

Welcome ☒

Welcome to the Heimdall Data documentation!

You can use navigation above to travel to sections of interest, then on every page you can use left side navigation to jump to other topics from this section or right side navigation to move to subject on current page.

Introduction

The Heimdall Data proxy is a software platform for application and infrastructure owners. Whether on-premise or cloud, Heimdall helps organizations deliver faster, more reliable, and secure content generation. Heimdall Data is a distributed Database Proxy that improves database performance from an application perspective. We give users visibility and control over their SQL environment.

Helpful hint when starting with Heimdall

If you have a ChatGPT account, you can download the PDF version of our documentation, and submit it to ChatGPT 4 or higher as an attachment, then ask questions based on the documentation. We aim to have the documentation complete enough to answer the questions accurately, so this can help in self-serve questions without having to read the entire documentation yourself. If you have a question that is not answered or is incorrect, please let us know, and we will update the documentation to account for the missing information.

If you're not sure where to begin, a good place to start is here:

→ [Getting Started](#)

Introduction

The Heimdall Data proxy is a software platform for application and infrastructure owners. Whether on-premise or cloud, Heimdall helps organizations deliver faster, more reliable, and secure content generation. Heimdall Data is a distributed Database Proxy that improves database performance from an application perspective. We give users visibility and control over their SQL environment.

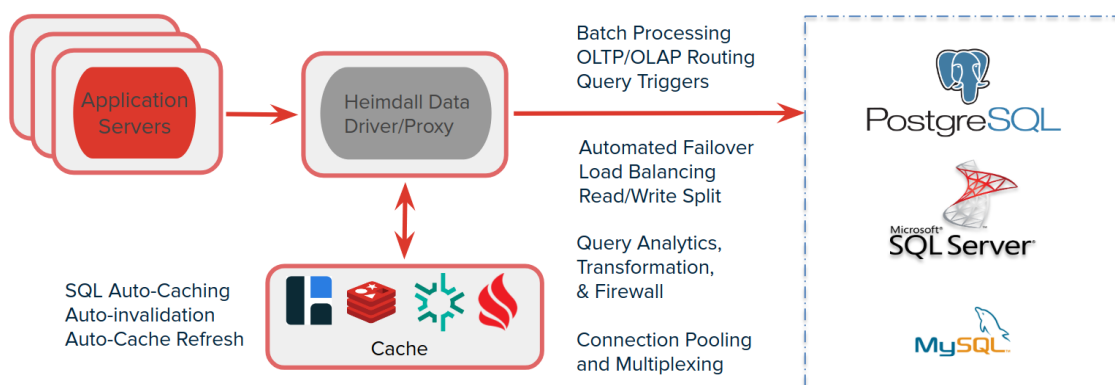
Helpful hint when starting with Heimdall

If you have a ChatGPT account, you can download the PDF version of our documentation, and submit it to ChatGPT 4 or higher as an attachment, then ask questions based on the documentation. We aim to have the documentation complete enough to answer the questions accurately, so this can help in self-serve questions without having to read the entire documentation yourself. If you have a question that is not answered or is incorrect, please let us know and we will update the documentation to account for the missing information.

Use cases for the software

The key use cases for Heimdall database proxy include:

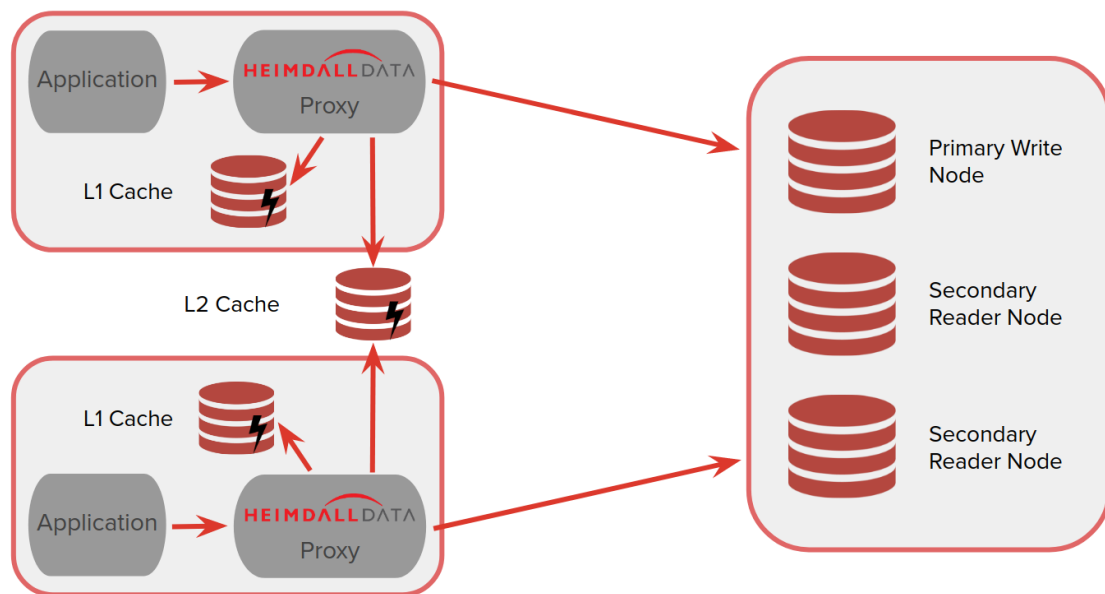
- Connection pooling from serverless applications, including support for multi-user pooling
- Automated caching and invalidation to the cache of your choice (e.g. Redis)
- Read/Write split (i.e. query routing) to reduce load on write-masters in a multi-node cluster
- Improving the performance of DML Ingestion, in particular in Analytics Database environments (Pivotal Greenplum and AWS Redshift)
- Analytics



Heimdall supports a variety of databases including Postgres & compatible (Amazon RDS, Aurora and Redshift; VMWare Greenplum, Google Cloud SQL & AlloyDB), MySQL and SQL Server compatible databases. For caching, it supports Hazelcast and Redis.

Caching Architecture

Heimdall optimally operates via a two-layer cache system. The first layer (L1) is an in-heap cache, the size of which is controlled by the heap size in the VDB->proxy settings. The second layer (L2) is used as secondary storage and as a means of communication amongst the deployed proxies. The user will choose any of the supported cache engines to provide a distributed cache between nodes.



Performance Impact

In implementing Heimdall, one frequent question is "what is the performance impact"? This is a difficult question to answer as the variables can be quite complex including:

- Size of requests, i.e. multi-page queries vs. simple single table queries;
- Number of requests/second;
- Size of results and the number of rows;
- If logging is enabled;
- Cache hit rates--the higher the better;
- Number of cores available for processing;
- Number of connections in use at one time;

- Deployment model (centralized vs. distributed), see below;

In a highly optimized scenario with 6 physical cores (12 hyperthreading), with 100% cache hit rate and simple results, Heimdall can achieve over 250k queries/second, WITH the test client (jmeter) on the same host consuming cpu time as well. Heimdall is designed to operate distributed however, so can reasonably scale to nearly any performance level. In most environments, it is suggested that up to 8 cores be used, and that additional scaling based on CPU load be done horizontally, adding additional nodes. In an AWS environment, this can be done in auto-scaling groups to account for surges very easily.

Please note: while jmeter CAN be used to benchmark Heimdall, it is not the proper tool to directly connect to the database to test and validate caching and read/write split. Customers should use their own applications to drive traffic in a lower environment (test). In such a setup, jmeter can be used to generate application load that THEN triggers database traffic through Heimdall. Using applications such as JMeter, DBeaver, or other similar tools will not reflect accurately how Heimdall will behave with other applications so should be avoided for this purpose as a general rule.

Prerequisites and Requirements

In order to install and use Heimdall, the following will be needed:

- A login for the database. For full capabilities, this user should have the ability to login, execute a health check query, access (or create) a schema/database of "heimdall", and have full access to this schema or database. This is used for health checking and various other features such as lag detection.
- Understanding of the configuration of the application, and how to adjust what database it points to. This varies from application to application, but a few common locations for this are:
 - The IIS configuration for a .net application, under connect strings
 - a configuration json file
 - a configuration PHP file
- Preferably a test environment to test with before going into production.
- A Linux server or Docker environment (see below for deployment options) with at least 20GB of space and 4GB of RAM

Supported versions of Linux include vendor supported versions of (in alphabetical order):

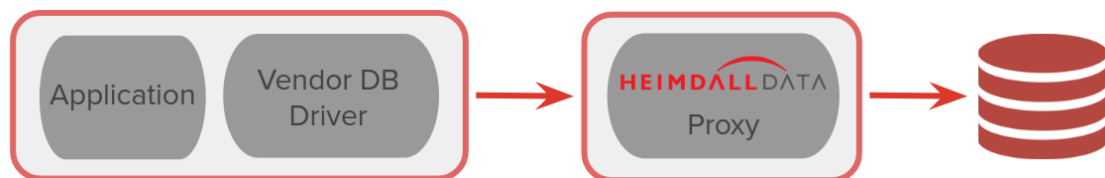
- Alpine
- Amazon Linux 2+
- Centos

- Debian
- Oracle
- Redhat/Fedora
- Suse
- Ubuntu

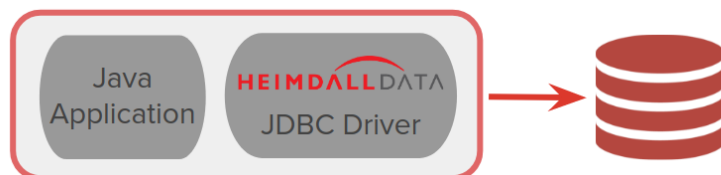
As Heimdall is a Java based program with no cpu architecture dependent code, it can be installed on any architecture supported by Linux and Java. In the case of AWS, the Graviton 2 & 3 instances provides a roughly 30-50% better price/performance ratio vs. the standard Intel VMs in our testing, and should be the first choice in deploying out an auto-scaling cluster. No special instructions are needed to use Heimdall on non-Intel architecture systems.

Deployment Models

Heimdall can be installed as a proxy:



Alternatively, it can be installed as a JDBC Driver (only for Java Applications) and can support other JDBC data sources as well:



For most customers, it is advisable to start with a simple proxy deployment on a single node, and once tested, to work with Heimdall to support more advanced configurations.

In proxy mode, Heimdall supports auto-scaling, multi-zone configurations. It is recommended that in AWS, the [AWS CloudFormation Template](#) should be used, which prompts for the various

configuration options in order to deploy such a configuration. For other environments, please contact Heimdall support for guidance on how to configure such a configuration.

Deployment Methods

In order to install Heimdall, one of several methods can be used:

- Leverage our single command installer into a supported version of Linux as a single server ([documentation](#))
- Purchase a VM with the software installed from a cloud marketplace ([AWS](#), [Azure](#) or [GCP](#))
- Use the docker file to create a docker image ([documentation](#))
- Install into the application as a JDBC driver (please contact Heimdall support for assistance).

For a detailed breakdown on each of the install methods, please see the detailed [install](#) page.

One note of importance—all the install methods except a pre-build VM will by default pull our newest version of code from our S3 file distribution site at the time of install except for the manual install option. **When installing, make sure that security groups or ACLs do not block this out-bound request.** Once installed and online, the system can be locked down if desired [see security](#).

Additionally, the deployment model will impact pricing. In the cloud marketplace images, purchase is based on the size of instance and time used, although many offer a free trial for testing or a free trial can be requested. In on-premises installs, pricing will be direct from Heimdall, and negotiated based on the number of instances used and volume, along with the support model needed.

In most cases, an initial configuration can be made within 30 minutes, and tested using tools such as DBeaver, SQL Server Management Studio, or PGAdmin 2.

Initial Configuration

Once deployed, connect to the server on port 8087/HTTP or 8443/HTTPS. It is recommended that all users leverage the configuration [wizard](#) once logged in, in order to do the initial install, as it guides the configuration via a series of questions and prompts in order to build a valid configuration. Please see the [AWS specific instructions](#) for information on setting an IAM role for AWS configuration auto-detection.

Once setup, the data source tab provides a test source button to validate access to the database. Likewise, a test VDB option is available on the VDB to pass some basic traffic to the VDB. **Note:** When using the test VDB option, it requires access to create a new schema or database (named Heimdall) and tables within it. If the data source is not configured with a user

that has access to do this, then the test VDB option will fail, even if everything is configured properly for application access.

Install Process (Detailed)

For instructions in [AWS](#), [Azure](#) or [GCP](#), please see the environment specific details.

Typical Install

To begin with, ensure to provision a Linux VM or bare metal instance with at least two CPU cores and 4GB of RAM. Login and install with the following command:

```
sudo bash -c 'bash <(curl -s http://s3.heimdalldata.com/hdinstall.sh) server'
```

While executing, output similar to the following should be generated (this is on Ubuntu 18.04, your output may vary):

```
    This script may take time to install dependencies and download, please be
    patient!
    Detected distro: ubuntu
    Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic InRelease
    Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates
    InRelease [88.7 kB]
    ...
    Get:28 http://security.ubuntu.com/ubuntu bionic-security/multiverse
    Translation-en [2356 B]
    Fetched 17.9 MB in 4s (4173 kB/s)
    Reading package lists...
    Reading package lists...
    Building dependency tree...
    Reading state information...
    The following additional packages will be installed:
      libcurl4 libhavege1 libopts25 libpython-stdlib libpython2.7-minimal
      libpython2.7-stdlib python python-meld3 python-minimal python-pkg-resources
      python2.7 python2.7-minimal
    Suggested packages:
      ntp-doc python-doc python-tk python-setuptools python2.7-doc binutils
      binfmt-support supervisor-doc zip
    Recommended packages:
      sntp
    The following NEW packages will be installed:
      haveged libhavege1 libopts25 libpython-stdlib libpython2.7-minimal
      libpython2.7-stdlib ntp python python-meld3 python-minimal python-pkg-
      resources python2.7 python2.7-minimal supervisor
      unzip
    The following packages will be upgraded:
      curl libcurl4
    2 upgraded, 15 newly installed, 0 to remove and 60 not upgraded.
    Need to get 5663 kB of archives.
    After this operation, 21.8 MB of additional disk space will be used.
```

```

Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic/universe amd64
libopts25 amd64 1:5.18.12-4 [58.2 kB]
...
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic/universe
amd64 supervisor all 3.3.1-1.1 [256 kB]
Fetched 5663 kB in 0s (31.4 MB/s)
Selecting previously unselected package libopts25:amd64.
(Reading database ... 56638 files and directories currently installed.)
Preparing to unpack .../0-libopts25_1%3a5.18.12-4_amd64.deb ...
Unpacking libopts25:amd64 (1:5.18.12-4) ...
...
Selecting previously unselected package supervisor.
Preparing to unpack .../8-supervisor_3.3.1-1.1_all.deb ...
Unpacking supervisor (3.3.1-1.1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for ureadahead (0.100.0-21) ...
Setting up libcurl4:amd64 (7.58.0-2ubuntu3.8) ...
Setting up unzip (6.0-21ubuntu1) ...
Setting up libhavege1:amd64 (1.9.1-6) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for systemd (237-3ubuntu10.24) ...
Setting up libopts25:amd64 (1:5.18.12-4) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Setting up libpython2.7-stdlib:amd64 (2.7.15-4ubuntu4~18.04.1) ...
Setting up curl (7.58.0-2ubuntu3.8) ...
Setting up ntp (1:4.2.8p10+dfsg-5ubuntu7.1) ...
Created symlink /etc/systemd/system/network-pre.target.wants/ntp-systemd-
netif.path → /lib/systemd/system/ntp-systemd-netif.path.
Created symlink /etc/systemd/system/multi-user.target.wants/ntp.service
→ /lib/systemd/system/ntp.service.
ntp-systemd-netif.service is a disabled or a static unit, not starting it.
Setting up python2.7 (2.7.15-4ubuntu4~18.04.1) ...
Setting up haveged (1.9.1-6) ...
Created symlink /etc/systemd/system/default.target.wants/haveged.service
→ /lib/systemd/system/haveged.service.
Setting up libpython-stdlib:amd64 (2.7.15~rc1-1) ...
Setting up python (2.7.15~rc1-1) ...
Setting up python-meld3 (1.0.2-2) ...
Setting up python-pkg-resources (39.0.1-2) ...
Setting up supervisor (3.3.1-1.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/
supervisor.service → /lib/systemd/system/supervisor.service.
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for systemd (237-3ubuntu10.24) ...
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
  ca-certificates-java default-jre-headless fontconfig-config fonts-
dejavu-core java-common libasound2 libasound2-data libavahi-client3 libavahi-
common-data libavahi-common3 libcups2
  libfontconfig1 libjpeg-turbo8 libjpeg8 liblcms2-2 libnspr4 libnss3
libpcsclite1 libxi6 libxrender1 libxtst6 openjdk-11-jdk-headless openjdk-11-
jre-headless x11-common
Suggested packages:
  default-jre libasound2-plugins alsa-utils cups-common liblcms2-utils

```

```

pcscd openjdk-11-demo openjdk-11-source libnss-mdns fonts-dejavu-extra fonts-
ipafont-gothic fonts-ipafont-mincho
    fonts-wqy-microhei | fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
    ca-certificates-java default-jdk-headless default-jre-headless
fontconfig-config fonts-dejavu-core java-common libasound2 libasound2-data
libavahi-client3 libavahi-common-data
    libavahi-common3 libcups2 libfontconfig1 libjpeg-turbo8 libjpeg8
liblcms2-2 libnspr4 libnss3 libpcsc-lite1 libxi6 libxrender1 libxtst6
openjdk-11-jdk-headless openjdk-11-jre-headless
    x11-common
0 upgraded, 25 newly installed, 0 to remove and 60 not upgraded.
Need to get 232 MB of archives.
After this operation, 383 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/main
amd64 libjpeg-turbo8 amd64 1.5.2-0ubuntu5.18.04.1 [110 kB]
...
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates/main
amd64 default-jdk-headless amd64 2:1.11-68ubuntu1~18.04.1 [1132 B]
Fetched 232 MB in 4s (54.9 MB/s)
Selecting previously unselected package libjpeg-turbo8:amd64.
(Reading database ... 57703 files and directories currently installed.)
Preparing to unpack .../00-libjpeg-
turbo8_1.5.2-0ubuntu5.18.04.1_amd64.deb ...
Unpacking libjpeg-turbo8:amd64 (1.5.2-0ubuntu5.18.04.1) ...
Selecting previously unselected package java-common.
...
Setting up libxi6:amd64 (2:1.7.9-1) ...
Processing triggers for ureadahead (0.100.0-21) ...
Setting up liblcms2-2:amd64 (2.9-1ubuntu0.1) ...
Setting up libpcsc-lite1:amd64 (1.8.23-1) ...
Setting up fonts-dejavu-core (2.37-1) ...
Setting up libasound2-data (1.1.3-5ubuntu0.2) ...
Setting up java-common (0.68ubuntu1~18.04.1) ...
Setting up libjpeg-turbo8:amd64 (1.5.2-0ubuntu5.18.04.1) ...
Setting up libnspr4:amd64 (2:4.18-1ubuntu1) ...
Setting up libasound2:amd64 (1.1.3-5ubuntu0.2) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for systemd (237-3ubuntu10.24) ...
Setting up libxrender1:amd64 (1:0.9.10-1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Setting up x11-common (1:7.7+19ubuntu7.1) ...
update-rc.d: warning: start and stop actions are no longer supported;
falling back to defaults
Processing triggers for ca-certificates (20180409) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
Setting up libavahi-common-data:amd64 (0.7-3.1ubuntu1.2) ...
Setting up libjpeg8:amd64 (8c-2ubuntu8) ...
Setting up fontconfig-config (2.12.6-0ubuntu2) ...
Setting up libnss3:amd64 (2:3.35-2ubuntu2.3) ...
Setting up libxtst6:amd64 (2:1.2.3-1) ...
Setting up libavahi-common3:amd64 (0.7-3.1ubuntu1.2) ...
Setting up libfontconfig1:amd64 (2.12.6-0ubuntu2) ...
Setting up libavahi-client3:amd64 (0.7-3.1ubuntu1.2) ...

```

```

Setting up libcups2:amd64 (2.2.7-1ubuntu2.7) ...
Setting up ca-certificates-java (20180516ubuntu1~18.04.1) ...
head: cannot open '/etc/ssl/certs/java/cacerts' for reading: No such file
or directory
Adding debian:SecureTrust_CA.pem
...
Adding debian:OISTE_WISKey_Global_Root_GA_CA.pem
done.
Setting up default-jre-headless (2:1.11-68ubuntu1~18.04.1) ...
Setting up openjdk-11-jre-headless:amd64 (11.0.4+11-1ubuntu2~18.04.3) ...
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/rmid to
provide /usr/bin/rmid (rmid) in auto mode
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/java to
provide /usr/bin/java (java) in auto mode
...
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jhsdb to
provide /usr/bin/jhsdb (jhsdb) in auto mode
Setting up default-jdk-headless (2:1.11-68ubuntu1~18.04.1) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for systemd (237-3ubuntu10.24) ...
Processing triggers for ca-certificates (20180409) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...

done.
done.
/dev/fd/63: line 203: [: 0 2019-07-16: syntax error in expression (error
token is "2019-07-16") # ignore if observed, due to OpenJDK version string
/dev/fd/63: line 207: [: 0 2019-07-16: syntax error in expression (error
token is "2019-07-16") # ignore if observed
Downloading Heimdall install package--this may take a while.
% Total    % Received % Xferd  Average Speed   Time    Time       Time
Current
                                Dload  Upload  Total  Spent  Left
Speed
100 158M 100 158M    0    0 52.6M    0  0:00:03  0:00:03 --:--:--
52.6M
Decompressing zip file
Archive:  heimdall.zip
  creating: heimdall/
  creating: heimdall/devops/
 inflating: heimdall/devops/packer-build.sh
 inflating: heimdall/devops/packer-template
 inflating: heimdall/devops/packer-setup.sh
 inflating: heimdall/statechange-greenplum-source.py
 inflating: heimdall/heimdalldriver.jar
  creating: heimdall/config/
 inflating: heimdall/config/Oracle11Java6_1.conf
 inflating: heimdall/config/Oracle12cJava6_1.conf
 inflating: heimdall/config/MySQL_1.conf
 inflating: heimdall/config/MS-SQL-Server_1.conf
 inflating: heimdall/config/admin_1.conf
 inflating: heimdall/config/Oracle11_1.conf
 inflating: heimdall/config/PostgreSQL_1.conf
 inflating: heimdall/config/heimdall.conf

```

```

inflating: heimdall/proxy-start.sh
inflating: heimdall/heimdallserver.sh
inflating: heimdall/heimdallserver.jar
inflating: heimdall/license.rtf
inflating: heimdall/NOTICE.txt
  creating: heimdall/static/
  creating: heimdall/static/drivers/
  creating: heimdall/static/drivers/MS-SQL-Server/
inflating: heimdall/static/drivers/MS-SQL-Server/mssql-
jdbc-6.4.0.jre7.jar
  creating: heimdall/static/drivers/PostgreSQL/
inflating: heimdall/static/drivers/PostgreSQL/
postgresql-42.2.5.jre6.jar
  creating: heimdall/static/drivers/MySQL/
inflating: heimdall/static/drivers/MySQL/mysql-connector-
java-5.1.47.jar
  creating: heimdall/static/drivers/Oracle11Java6/
inflating: heimdall/static/drivers/Oracle11Java6/Oracle11gR2-
ojdbc6.jar
  creating: heimdall/static/drivers/Oracle12cJava6/
inflating: heimdall/static/drivers/Oracle12cJava6/Oracle12cR1-
ojdbc6.jar
  creating: heimdall/static/modules/
inflating: heimdall/static/modules/heimdallsqlserver-1.0-19.7.21.1.jar
inflating: heimdall/static/modules/heimdallforward-1.0-19.7.21.1.jar
inflating: heimdall/static/modules/
heimdallloadbalancer-1.0-19.7.21.1.jar
inflating: heimdall/static/modules/heimdallfirewall-1.0-19.7.21.1.jar
inflating: heimdall/static/modules/
heimdallhazelcast-3.6.8-19.7.21.1.jar
inflating: heimdall/static/modules/heimdallredis-3.5.0-19.7.21.1.jar
inflating: heimdall/static/modules/heimdallpw-1.0-19.7.21.1.jar
inflating: heimdall/static/modules/heimdallauth-1.0-19.7.21.1.jar
inflating: heimdall/static/modules/heimdallpostgres-1.0-19.7.21.1.jar
inflating: heimdall/static/modules/heimdalltrigger-1.0-19.7.21.1.jar
inflating: heimdall/static/modules/
heimdallextractplan-1.0-19.7.21.1.jar
inflating: heimdall/static/modules/heimdallmysql-1.0-19.7.21.1.jar
inflating: heimdall/static/modules/heimdallasync-1.0-19.7.21.1.jar
  creating: heimdall/super/
inflating: heimdall/super/readme.txt
inflating: heimdall/super/heimdallproxy.conf
inflating: heimdall/super/heimdallserver.conf
inflating: heimdall/hdinstall.sh
Failed to stop supervisord.service: Unit supervisord.service not loaded.
Failed to disable unit: Unit file supervisord.service does not exist.
Created symlink /etc/systemd/system/multi-user.target.wants/
supervisord.service → /etc/systemd/system/supervisord.service.
started supervisord via systemctl
The server should now be online, and available on http://<server ip>:8087/
Please refer to the help on the login page for the default username and
password

```

This should bring online the UI on port 8087. To check, execute:


```
netstat -alnp | grep "8087.*java"
```

A line should appear similar to:

```
tcp6      0      0 :::8087          :::*  
LISTEN    7464/java
```

Manual Install Behind Firewall

⚠ **Note:** You can download heimdall into given directory by setting environment variable `HD_DIR` for absolute path you want to install in, if variable is not present it defaults to `/opt`. Make sure its set before you run `hdinstall.sh`. You can check if it's set by doing:

```
echo $HD_DIR
```

make sure to download `hdinstall.sh` and `heimdall.zip` into folder you want it configured and follow instructions below with your path instead of `/opt`.

If your linux instance is behind a firewall, and can't access the Internet, please use the following to install Heimdall:

1. Install the following dependencies manually per your Linux distribution's standards:
OpenJDK 8+ (Oracle Java is acceptable as well) unzip supervisor (/etc/supervisor/conf.d is assumed for service configuration)
2. `mkdir /opt; cd /opt`
3. Download the following files into `/opt`: <https://s3.amazonaws.com/s3.heimdalldata.com/hdinstall.sh> <https://s3.amazonaws.com/s3.heimdalldata.com/heimdall-java17.zip>
4. `chmod a+x /opt/hdinstall.sh`
5. `./hdinstall.sh server`

This should result in output similar to the following:

```
isProxy: false  
This script may take time to install dependencies and download, please be  
patient!  
Detected distro: ubuntu  
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic InRelease  
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-updates  
InRelease [88.7 kB]  
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu bionic-backports  
InRelease [74.6 kB]  
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7  
kB]  
Fetched 252 kB in 0s (764 kB/s)  
Reading package lists...
```

```

Reading package lists...
Building dependency tree...
Reading state information...
unzip is already the newest version (6.0-21ubuntu1).
haveged is already the newest version (1.9.1-6).
supervisor is already the newest version (3.3.1-1.1).
curl is already the newest version (7.58.0-2ubuntu3.8).
ntp is already the newest version (1:4.2.8p10+dfsg-5ubuntu7.1).
0 upgraded, 0 newly installed, 0 to remove and 60 not upgraded.
./hdinstall.sh: line 205: [: 0 2019-07-16: syntax error in expression
(error token is "2019-07-16") # ignore if observed, due to OpenJDK version
string
./hdinstall.sh: line 209: [: 0 2019-07-16: syntax error in expression
(error token is "2019-07-16") # ignore if observed
Decompressing zip file
Archive:  heimdall.zip
  creating: heimdall/
  creating: heimdall/devops/
inflating: heimdall/devops/packer-build.sh
inflating: heimdall/devops/packer-template
inflating: heimdall/devops/packer-setup.sh
inflating: heimdall/statechange-greenplum-source.py
inflating: heimdall/heimdalldriver.jar
  creating: heimdall/config/
inflating: heimdall/config/Oracle11Java6_1.conf
inflating: heimdall/config/Oracle12cJava6_1.conf
inflating: heimdall/config/MySQL_1.conf
inflating: heimdall/config/MS-SQL-Server_1.conf
inflating: heimdall/config/admin_1.conf
inflating: heimdall/config/Oracle11_1.conf
inflating: heimdall/config/PostgreSQL_1.conf
inflating: heimdall/config/heimdall.conf
inflating: heimdall/proxy-start.sh
inflating: heimdall/heimdallserver.sh
inflating: heimdall/heimdallserver.jar
inflating: heimdall/license.rtf
inflating: heimdall/NOTICE.txt
  creating: heimdall/static/
  creating: heimdall/static/drivers/
  creating: heimdall/static/drivers/MS-SQL-Server/
inflating: heimdall/static/drivers/MS-SQL-Server/mssql-
jdbc-6.4.0.jre7.jar
  creating: heimdall/static/drivers/PostgreSQL/
inflating: heimdall/static/drivers/PostgreSQL/
postgresql-42.2.5.jre6.jar
  creating: heimdall/static/drivers/MySQL/
inflating: heimdall/static/drivers/MySQL/mysql-connector-
java-5.1.47.jar
  creating: heimdall/static/drivers/Oracle11Java6/
inflating: heimdall/static/drivers/Oracle11Java6/Oracle11gR2-
ojdbc6.jar
  creating: heimdall/static/drivers/Oracle12cJava6/
inflating: heimdall/static/drivers/Oracle12cJava6/Oracle12cR1-
ojdbc6.jar
  creating: heimdall/static/modules/
inflating: heimdall/static/modules/heimdallsqlserver-1.0-19.7.21.1.jar
inflating: heimdall/static/modules/heimdallforward-1.0-19.7.21.1.jar

```

```

    inflating: heimdall/static/modules/
heimdallloadbalancer-1.0-19.7.21.1.jar
    inflating: heimdall/static/modules/heimdallfirewall-1.0-19.7.21.1.jar
    inflating: heimdall/static/modules/heimdallgeode-1.8.0-19.7.21.1.jar
    inflating: heimdall/static/modules/
heimdallcoherence-12.2.1.3.0-19.7.21.1.jar
    inflating: heimdall/static/modules/
heimdallhazelcast-3.6.8-19.7.21.1.jar
    inflating: heimdall/static/modules/heimdallredis-3.5.0-19.7.21.1.jar
    inflating: heimdall/static/modules/heimdallignite-2.6.0-19.7.21.1.jar
    inflating: heimdall/static/modules/heimdallpw-1.0-19.7.21.1.jar
    inflating: heimdall/static/modules/heimdallauth-1.0-19.7.21.1.jar
    inflating: heimdall/static/modules/heimdallpostgres-1.0-19.7.21.1.jar
    inflating: heimdall/static/modules/heimdalltrigger-1.0-19.7.21.1.jar
    inflating: heimdall/static/modules/
heimdallextractplan-1.0-19.7.21.1.jar
    inflating: heimdall/static/modules/heimdallmysql-1.0-19.7.21.1.jar
    inflating: heimdall/static/modules/heimdallasync-1.0-19.7.21.1.jar
    creating: heimdall/super/
    inflating: heimdall/super/readme.txt
    inflating: heimdall/super/heimdallproxy.conf
    inflating: heimdall/super/heimdallserver.conf
    inflating: heimdall/hdinstall.sh
Removed /etc/systemd/system/multi-user.target.wants/supervisord.service.
Created symlink /etc/systemd/system/multi-user.target.wants/
supervisord.service → /etc/systemd/system/supervisord.service.
started supervisord via systemctl
The server should now be online, and available on http://<server ip>:8087/
Please refer to the help on the login page for the default username and
password

```

This should bring online the UI on port 8087. To check, execute:

```
netstat -alnp | grep "8087.*java"
```

A line should appear similar to:

```

tcp6      0      0 :::8087          :::*
LISTEN    7464/java

```

Docker Install

First download the Docker file:

```
wget https://s3.amazonaws.com/s3.heimdalldata.com/Dockerfile
```

Next, build the base image, specifying the version or name you wish to use as a tag:

```
docker build -t "heimdall:current" .
```

This should result in logs similar to:

```
Sending build context to Docker daemon    351MB
Step 1/9 : FROM ubuntu:latest
latest: Pulling from library/ubuntu
35c102085707: Pull complete
251f5509d51d: Pull complete
8e829fe70a46: Pull complete
6001e1789921: Pull complete
Digest:
sha256:d1d454df0f579c6be4d8161d227462d69e163a8ff9d20a847533989cf0c94d90
Status: Downloaded newer image for ubuntu:latest
---> a2a15febcd3
Step 2/9 : EXPOSE 8087
---> Running in dc59ab49c878
Removing intermediate container dc59ab49c878
---> aa80982c4f15
Step 3/9 : EXPOSE 5432
---> Running in f06c8b383fcb
Removing intermediate container f06c8b383fcb
---> d1ddf2b32310
Step 4/9 : EXPOSE 5050
---> Running in 9c1b043ac1d3
Removing intermediate container 9c1b043ac1d3
---> 0346b8d31eeb
Step 5/9 : RUN DEBIAN_FRONTEND=noninteractive apt-get -y update &&
DEBIAN_FRONTEND=noninteractive apt-get -y upgrade &&
DEBIAN_FRONTEND=noninteractive apt-get -y install openjdk-8-jdk supervisor
unzip wget ntp
---> Running in a2107cee8a89
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7
kB]
Get:2 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64
Packages [4952 B]
...
  inflating: heimdall/static/modules/heimdallasync-1.0-19.7.21.1.jar
  creating: heimdall/super/
  inflating: heimdall/super/readme.txt
  inflating: heimdall/super/heimdallproxy.conf
  inflating: heimdall/super/heimdallserver.conf
  inflating: heimdall/hdinstall.sh
Removing intermediate container 37ddb6bcb143
---> 92b29de604ae
Step 7/9 : RUN mv heimdall /opt && cp /opt/heimdall/super/
heimdallserver.conf /etc/supervisor/conf.d
---> Running in 48e149131223
Removing intermediate container 48e149131223
---> 3b4b5a66bd11
Step 8/9 : RUN sed -i 's/^\([supervisord\]\)\$/\1\nnodaemon=true/' /etc/
supervisor/supervisord.conf
---> Running in 230a85b07000
Removing intermediate container 230a85b07000
---> 460248ac1b87
Step 9/9 : CMD ["supervisord", "-c", "/etc/supervisor/supervisord.conf"]
---> Running in 9c20424bec22
Removing intermediate container 9c20424bec22
```

```
---> b3372e92bd83
Successfully built b3372e92bd83
Successfully tagged heimdall:current
```

Now, run the instance, exposing the ports desired for management and proxy ports:

```
docker run -d --name heimdall-instance -p 8087:8087 -p 3306:3306 -p 5432:5432
-p 1433:1433 heimdall:current
```

Finally, verify that the docker instance is running:

```
docker ps -a
```

Result should be similar to:

CONTAINER ID	IMAGE	COMMAND
CREATED	STATUS	
PORTS		
1c5ed6913450	heimdall:current	"supervisord -c /etc..."
ago	Up 2 minutes	0.0.0.0:1433->1433/tcp, 0.0.0.0:3306->3306/tcp,
		0.0.0.0:5432->5432/tcp, 0.0.0.0:8087->8087/tcp, 5050/tcp
		heimdall-instance

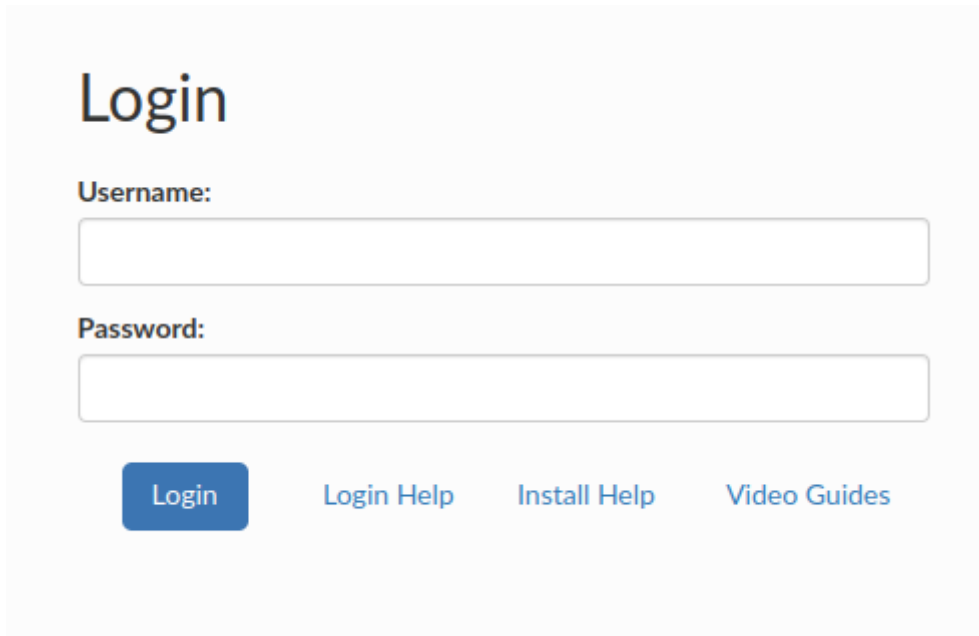
Application Configuration

If everything is setup properly, then an application's configuration can be set to point to the Heimdall Hostname (or the NLB hostname for a Cloudformation deployment), with the port as specified in the proxy configuration (typically matching the default for the database type). This configuration is application and framework dependent, so please consult the application's documentation on how to adjust this configuration.

Lambda and Container Configuration

When using lambdas and containers, no special instructions are needed—the application will have a database configuration, and you would configure it to point to either the proxy or the NLB endpoint pointing to the proxy, as appropriate.

Initial Login

A login form with a light gray background. At the top, the word "Login" is displayed in a large, bold, black font. Below it, the label "Username:" is followed by a white rectangular input field with a thin gray border. Underneath, the label "Password:" is followed by another white rectangular input field with a thin gray border. At the bottom of the form, there is a blue rectangular button with the word "Login" in white. To the right of this button are three blue links: "Login Help", "Install Help", and "Video Guides".

Login

Username:

Password:

Login Login Help Install Help Video Guides

The default Heimdall server username is **"admin"** and the password is:

"heimdall".

In AWS and GCP and similar environments (such as OpenStack), the **Cloud instance ID** is the default password. For Oracle cloud, it is the instance ocid. For Azure, it is the subscription ID.

If the password is not set to "heimdall" please see the server log (`{workDir}/heimdall/log/heimdall.log`) and look for a line such as:

```
2016-12-11 20:10:32,516 5153 [INFO] [http-nio-8087-exec-1] Initializing access control, user=admin, password=i-0aa1234a1a1a12abc
```

Note: It doesn't matter what install method you use—if in a cloud provider, as long as the software can detect the instance ID, it will default to the instance ID for the password. The following sources will be used to attempt to detect a cloud instance ID:

- `http://169.254.169.254/latest/meta-data/instance-id` (AWS & compatible environments)
- `http://169.254.169.254/metadata/instance/compute/subscriptionId?api-version=2020-09-01&format=text` (Azure)
- `http://169.254.169.254/computeMetadata/v1/instance/id` (GCP)
- `http://169.254.169.254/opc/v2/instance/id` (Oracle Cloud)

Note: As Azure doesn't expose the vmID to users easily, the subscription ID is used instead

User Reset

The default username and password for a server can be controlled explicitly by creating a file `/etc/heimdall.conf` prior to the server being first run, and add the lines:

- `hdUser={username}`
- `hdPassword={password}`

For more details of a file `/etc/heimdall.conf`, please see [heimdall.conf](#) configuration.

When the http query “`http://169.254.169.254/latest/meta-data/instance-id`” returns a value, this value is used as the default password if no configuration is set in the `/etc/heimdall.conf`. This will happen in AWS instances, and some other cloud environments. This value can be obtained with:

```
echo `curl -s http://169.254.169.254/latest/meta-data/instance-id`
```

In the event the passwords need to be deleted from a current configuration, the following steps can be used to delete the configured accounts:

1. Open the file `$heimdallhome/config/heimdall.conf` (the default install directory is `/opt/heimdall/`) in an editor (notepad, vi, etc)
2. Remove the json array values for the “address” variable, i.e. replace:

```
“addresses”:[{“enabled”:true,“file”:“config/admin_1.conf”,“name”:“admin”,“version”:1}]
```

With

```
“addresses”:[ ]
```

Once saved, if all accounts are removed, then the default admin account will be re-initialized once the Heimdall server is restarted.

To automate the password reset process you can do the following:

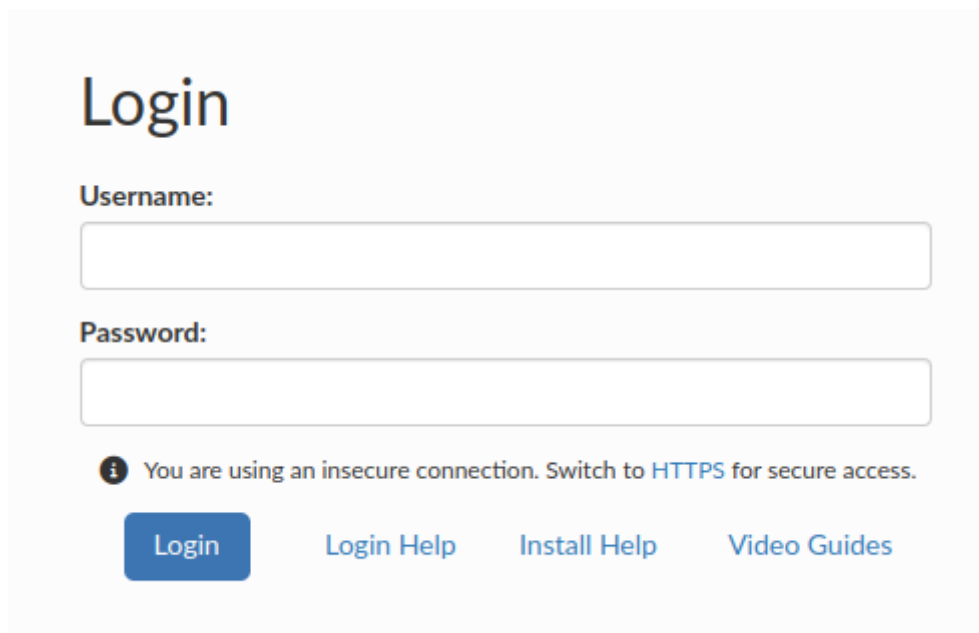
```
jq 'del(.addresses[])' < /{workDir}/heimdall/config/heimdall.conf > /  
{workDir}/heimdall/config/heimdall.conf.out  
mv -f /{workDir}/heimdall/config/heimdall.conf.out /{workDir}/heimdall/config/  
heimdall.conf  
service heimdall restart
```

Note: newer installs should include jq by default, but older will not, and will need to use the distribution's package installer to install jq.

File Base User Reset

The default username and password for a server can be reset by creating a file `/etc/heimdall_reset_password`. This operation will only change the credentials for an admin user, the default admin account will be re-initialized once the Heimdall server is restarted. When this operation succeed the file will be deleted.

Insecure connection



The screenshot shows the Heimdall login interface. At the top, the word "Login" is displayed in a large, bold, dark font. Below it, the label "Username:" is followed by a text input field. Underneath that, the label "Password:" is followed by another text input field. Below the password field, there is a warning message: "You are using an insecure connection. Switch to [HTTPS](#) for secure access." The message is preceded by a small information icon. At the bottom of the form, there is a blue button labeled "Login". To the right of the button, there are three links: "Login Help", "Install Help", and "Video Guides".

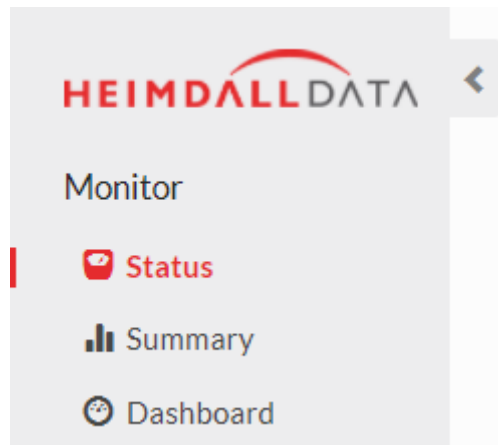
In case of an insecure connection, a warning 'You are using an insecure connection. Switch to HTTPS for secure access' appears, with a clickable 'HTTPS' link, which enables users to easily switch to a secure HTTPS protocol.

Navigation Overview

All windows have two main sets of controls for navigation, the sidebar to the left and the settings bar on the upper right. Additionally, just below the settings bar, if present, current alerts can be displayed.

Sidebar

The sidebar allows navigation between all the various tabs in the application. The current tab is highlighted in red. Via the < and > arrow beside the logo, the sidebar can be collapsed to reduce the amount of space taken:



collapses to:

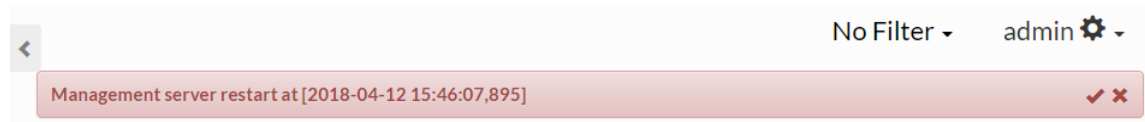


Settings Bar



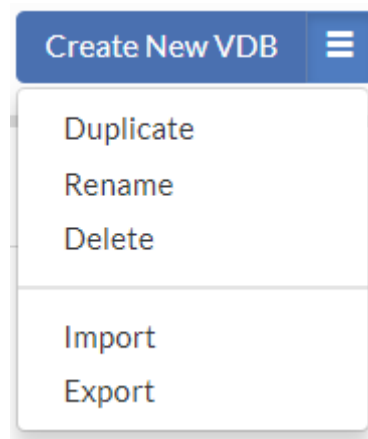
The settings bar currently provides the currently logged in user, and a gear. The gear provides options to sign out or change the password. Future settings options may also be presented here.

Alerts



In the alert section, up to three alerts may be displayed. In order to hide (but not delete) an alert, use the check. In order to permanently delete an alert, use the X. All alerts can be managed in the admin->alerts section as well.

Object Management



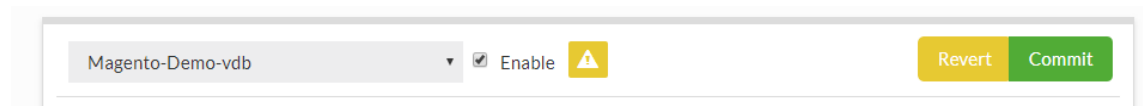
For each section that manages a configuration object, there is a menu in the upper right-hand of the configuration section. Options will include:

- **Create New** : To create a new object of this type
- **Duplicate**: In order to create an identical copy, but under another name
- **Delete**: To delete the currently selected object
- **Import**: To import a previously exported configuration object
- **Export**: Export the currently selected object

When using the export function, all dependencies are exported with the object, i.e. when exporting a Virtual Database object, it will also export the associated rules and data sources. The exception for this is that the driver objects will not include the actual driver file associated with it, but only the configuration name and settings. If on loading a driver config however, there is already an existing configuration in place, then the data source will point to the existing driver configuration, without loading the settings in the configuration file. This could, for

example, result in the loss of customized template URL's for AWS environments. Please refer to the json file for any custom settings and verify they are correct on the restoration of a VDB.

Change Control



When interacting with configuration objects, any time a change is made to the object, it is flagged as changed using a yellow alert icon at the top, and the option to revert or commit the configuration. In the sidebar, any section with pending changes will be colored yellow. By selecting revert, the change is prevented, and the configuration restored to the initial state, while a commit will push the configuration to the server and the proxies/drivers.

Context Help

Nearly every label in the system is configured to provide a tool-tip for additional help when the mouse hovers over it. Additionally, many icons used will provide additional information in this way, such as the icons on the rules tab.

Post Install Operations

Health checks and Monitoring

Heimdall is divided into two components, the management server and the proxy or driver (it is the same code). This corresponds to the control plane and the data plane. The management server can go offline without impacting the proxy, although when offline, configuration changes can not be orchestrated. To monitor the accessibility of the management server, a monitoring system can use the default port of 8087 (see below in security for variations).

In order to add monitoring to the proxy nodes, say for load balancing purposes, an HTTP port can be opened up that provides the status of the system, based on the ability to completely pass a health check request through the proxy to the back-end database. The options to configure this are in the VDB documentation, under vdb properties, specifically the "healthcheck-port" and "healthcheck-interval". Once configured and the proxy restarted, a query against the url "/status" will report either "OK" with a "200 OK" status or "FAILED" with a "500 Server Error" status.

Additional monitoring can be enabled by selecting "AWS CloudWatch" when in AWS, and each proxy node will report to cloudwatch various statistics about cache hit rate, cpu load and performance. Please note, a proper IAM role is necessary for this option to be active (please see the security section for details).

Backup

In order to backup a complete configuration of Heimdall, two directories need to be archived. First, /{workDir}/heimdall/config, which contains the json configuration files, and possibly / {workDir}/heimdall/static/drivers, where any modified driver files are stored. If only the stock drivers are used, then this second directory does not need to be archived.

There is an automatic backup process in place, running every time Heimdall is updated and daily at midnight. This will put whole {workDir}/heimdall/config directory into a zip archive located in {workDir}/heimdall/config-backups. If needed, the files from this archive can be copied into /{workDir}/heimdall/config to roll back the state of Heimdall configuration. By default, this process holds up to 10 backups, this value can be configured in [Server Properties](#)

Additionally, individual VDB's can be exported via the menu icon in the upper right hand corner of the settings. This will save the vdb, data source, and rules settings that apply to the VDB. This configuration can then be imported into another instance.

Restore

To restore the configuration, install Heimdall again (if needed) and replace the content of `{workDir}/heimdall/config` with the archived versions. If the system is currently online, the new configurations can be read and activated by restarting the application or calling an api call to port 8087 with `"/api/config/reload"`. This is also useful if using a configuration management system to synchronize configurations from an offline source or staging environment.

Maintenance

Heimdall is designed to operate with low maintenance. Some maintenance options that may be desired however:

- Control how logs roll to manage disk space: Log files will roll when the log filesystem reaches 90% by default. This is configurable via the admin->properties->Reserved disk space setting OR the Max Log Age setting.
- New version updates will be e-mailed to the configured notification contact if access to the Internet is provided. The settings can be configured to route the mail through a local mail server as desired.
- Heimdall advertises version updates for four release trains--LTS, which is limited to major bugfixes only on the LTS train, the "release" train for normal updates and a test build. To configure which release build update notifications will be sent as, in the admin->software management tab, simply select the tracking option. Most customers should select the "release" train. The test train may be updated at times with code that a specific customer has been asked to download, and may not be as well tested as the release build, but will likely have new features as well.
- To update to a new version of software, simply select the option in the software management tab. This will require external access to download the new build. The old build files will be saved in the `{workDir}/heimdall` directory labeled with a `.old` suffix in case a manual fallback is needed.

Securing Heimdall

Best Practices

Please follow the following best practices:

- Lock down access by creating an ACL or Security Group to block access from the Internet;
- Limit inbound traffic to Heimdall Proxy by allowing connections only from trusted security groups (e.g., application servers, corporate VPN clients, or a bastion host). Do not expose the proxy to the public internet.
- Ensure the Heimdall proxy is the only allowed entry point to the database;
- If you wish to have logs strip the actual SQL queries, in the VDB, you can set the vdb setting of "paranoia" to true;
- Lock down the interface with HTTPS, and for LDAP use Secure LDAP (LDAPS);
- Use a separate cache engine for Heimdall traffic and your application;
- Enable TLS as desired on the back-end for database access;
- Deploy the Database and the Heimdall Proxy in Private Subnets;
- Use a secrets manager for all passwords to avoid cleartext passwords in the configurations;
- When using secrets manager, point to RDS managed secrets to connect to the cluster, and then also use this same secret for the access key/secret key, allowing consistent and aligned rotation for both. This also helps support cross-region and cross-account secrets, as the manager can connect to the remote region to pull the RDS secret, and the proxy nodes in that region would use the locally managed secrets.
- Rotate passwords and certificates regularly. You can use Secrets Managers to automate this, as well as services like AWS ACM to rotate certificates;
- Implement hierarchical namespacing for secrets (e.g., `heimdall/secret1`, `postgres/secret1`) to streamline access control. This structure allows for granular permissions scoped to specific path prefixes (such as `heimdall/*`) rather than granting broad access to the entire Secret Manager;
- Strictly separate environments (Development, Staging, Production) to ensure stability and security. For instance, separation ensures that a configuration deployment or restart triggered in Staging does not inadvertently restart Production proxies, causing downtime;
- Ensure Proxy TLS Required option is selected to enforce TLS traffic;
- Do not use self-signed certificates on the production;

- Never use the root user. Set a limited-access user for the data source configuration, with access to the "heimdall" schema/database for replication lag detection, token store, and others;
- Create a read-only user for use by developers who need to view logs for monitoring;
- Use two-factor authentication, when possible;
- Set a complex password for the local admin account immediately upon installation;
- Adopt a centralized Identity Provider (IdP) for all management access. Avoid local users with plain-text passwords to eliminate the need for storing credentials in configuration files.

Advanced Security Features

Heimdall offers advanced security features such as critical alert email notifications, honey token configuration and detection, as well as sensitive data masking and tokenization. It also integrates with AWS CloudTrail to audit Heimdall login events and exports logs to AWS CloudWatch.

AWS Cross-Account Considerations

When deploying the management server in a distinct AWS account, specific automated behaviors may be impacted, such as the configuration wizard's autodetect option, might impact also other autodetect mechanism. However, for AWS Secrets Manager, Heimdall supports cross-region and cross-account secrets setups; please refer to the Secrets Manager documentation for further details.

Password Storage

Heimdall, in some cases needs to store passwords in a variety of formats for different purposes. Each case is documented here:

1. All configuration information is kept in `/{workDir}/heimdall/config`, accessible only to root, including password information;
2. By default, for Postgres and SQL Server, VDB authentication is set to "passthrough" which means that individual user's credentials are not stored on Heimdall at all;
3. For MySQL, due to the nature of authentication, individual user's credentials must be configured in effectively plaintext. These are stored in the vdb configuration file, and used to both authenticate the user into the proxy, and then to authenticate the user into the database;

4. For all configurations, an alternative to storing a user's vdb password is to configure SQL based authentication, in which case the passwords will be stored in a database table instead;
5. For GUI authentication, passwords are stored in Salted SHA format, compatible with OpenLDAP, i.e. automation could be configured to copy the same credential format between the two systems.
6. One additional cleartext password is stored in the Data source configuration tab, which is used to authenticate health checks and for performing replication lag detection. No access to actual data is necessary for this user's login.

NOTE: In AWS, it is possible to configure Heimdall to store the entire configuration in an AWS Secret. Please see the AWS Environment page for details on the `hdSecretKey` option. Using this ensures all passwords in the configuration are also stored encrypted and not on the EBS volume of the server.

Password Rotation

For the GUI passwords, rotation can simply be done via the user's tab. For Database password authentication, there are several categories:

1. In the case of Active directory authenticated users, credentials will be cached on a success for up to five minutes. On a failure however, the credentials will be re-queried against Active directory, and will be updated if success occurs. This ensures that as soon as AD is updated, the proxy will also be updated;
2. In the case of passthrough authentication, connections are established with the user's credentials on connect, so again, as soon as the password is changed, the credentials will be reflected;
3. If the proxy is using "proxy authentication" then the credentials are stored against the VDB configuration itself. To perform a rotation, you can add a new user/password combination **without** removing the old one. During this period, both will be acceptable. Then, once the application and database is updated to use the new credentials, then original password can be removed from the proxy. This allows the issue of timing the password changes between the application and database to ignore the proxy, since it will accept both at the same time.

Supported Encryption

Heimdall allows TLS security to be enabled on each VDB, and to be made required as well. In the case of Postgres and MySQL, TLS 1.3 will be used by default if the client supports it, while TLS 1.3 will be used for SQL Server, as SQL Server itself does not use TLS 1.3 yet, and all driver libraries will need to be updated for this to work.

For back-end connectivity, as we use the vendor JDBC drivers to connect to the database, we support all connectivity options that these drivers support for encryption. Please see the vendor JDBC driver help pages for specific details.

Enabling HTTPS for Management Server

Self-signed certificates

By default, the Tomcat certificate is created automatically with a blank password when the application is started if one doesn't exist. The `heimdall.conf` file contains a field called `"tomcatPassword"`, it refers to the private key password in `keystore.p12` file and can be freely changed. If the field has null value, a new one is created with a blank value. To create own Tomcat certificate, on the Linux command line, do the following (this will generate a self-signed certificate):

```
keytool -genkey -alias tomcat -storetype PKCS12 -keyalg RSA -keysize 2048 -  
keystore /{workDir}/heimdall/keystore.p12 -validity 3650
```

```
Enter keystore password: <use "heimdall">  
Re-enter new password:  
What is your first and last name?  
[Unknown]:  
What is the name of your organizational unit?  
[Unknown]:  
What is the name of your organization?  
[Unknown]:  
What is the name of your City or Locality?  
[Unknown]:  
What is the name of your State or Province?  
[Unknown]:  
What is the two-letter country code for this unit?  
[Unknown]:  
Is CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown  
correct?  
[no]: yes
```

Once done, restart the Heimdall server, and port 8443 will be used for HTTPS.

Please see the man page for `keytool` for importing in an existing SSL certificate if desired.

Certificate Import

In case you wanted to import your own SSL certificate, a keystore with the keypair with alias `"tomcat"` is required. Then, we can use `keytool` command to import our custom keystore to the keystore used by Heimdall (`new_keystore.p12` is the keystore user wants to import): When prompted, use the destination password of `'heimdall'`.

```
keytool -importkeystore -srckeystore new_keystore.p12 -destkeystore /  
{workDir}/heimdall/keystore.p12 -srcalias tomcat -srckeypass  
<source_tomcat_keypair_password> -destalias tomcat  
  
Importing keystore new_keystore.p12 to keystore.p12...  
Enter destination keystore password: *****  
Enter source keystore password: *****  
Existing entry alias tomcat exists, overwrite? [no]: yes  
Enter new alias name (RETURN to cancel import for this entry): tomcat
```

VDB Proxy TLS Support

TLS is a way of providing a safe connection between Heimdall server and client application while using proxy. By default Heimdall server enables use of TLS and decision about activating it depends on client side.

The core of TLS in Heimdall server are private keys with certificates stored in a keystore. Changes made in keystore can cause change of behavior of TLS in proxy.

Certificate binding

Heimdall allows binding of certificates in two modes:

- per-vdb: this type of private key with certificate is used only by particular proxies defined in Heimdall server. To use this mode an appropriate configuration is required, which will be shown later.
- global: this type of private key with certificate is used by all proxies which don't have defined particular use certificate in configuration. In keystore can exist only one private key with certificate used as global use certificate. Global use certificate alias in keystore is "global_use_certificate". If global use certificate doesn't exist in keystore, it is going to be autogenerated by Heimdall server.

Default Configuration

By default, when TLS use is called from client side, user doesn't have to configure anything for TLS - Heimdall server will create a keystore and generate a global use certificate.

Created by Heimdall server keystore will have below properties:

- keystore type will be JKS
- keystore file name will be `keystore.p12`
- password to keystore will be `heimdall`
- keystore file will be placed in heimdall server/proxy directory

Generated by Heimdall server global use certificate will have below properties:

- generated will be private key with certificate
- generated pair will have alias `tomcat` (note--this matches the HTTPS certificate for the central manager)
- generated pair password will be blank
- algorithm used to generate key pair will be SHA-256 with RSA
- key size will be 2048
- global use certificate will have self-signed certificate with `CN=Heimdall`
- certificate of global use certificate will be valid from day before the day of generating
- certificate of global use certificate will expire after 10 years minus 1 day

User Configuration

Heimdall server accepts configuring keystore by user, but doesn't provide a functionality for doing this. In this situation suggested is using third-party applications (for example: KeyStore Explorer) to make changes in keystore used by Heimdall server.

Creating keystore

For the situation when user want to create or replace a keystore by himself for Heimdall server, then below restrictions should be met:

- keystore type is JKS (may work with PKCS#12 type)
- keystore file is named `keystore.p12`
- password to keystore is `heimdall`
- keystore file is stored in the same directory as Heimdall server

Important: If a keystore is created with JDK 16 or higher, then you will not be able to use it with JDK lower than 16. The reason for that is JDK 16 uses more secure algorithms by default which are not supported by lower versions.

Reading keystore

For reading keystore we recommend using keytool utility.

Do not use gcr-viewer, cause currently it has a problem with reading certificates based on bouncycastle algorithms.

Managing certificates

The first thing to mention before managing certificates is that Heimdall server is using private keys with certificates (key pairs). Single keys or single certificates won't work with Heimdall Server.

For each vdb configuration it is possible to add data concerning certificates. The following fields must be added:

- `certificate` - the certificate in PEM format, it may or may not have a header and a footer
- `certificateAlias` - the alias refers to a certificate, preferably it should also refer to a specific vdb name
- `privateKey` - the private key in PEM format, it may or may not have a header and a footer

If any of the above fields are left with null value, the certificate configuration will be overwritten by default by `global_use_certificate`.

Below is an example showing how to add certificate data to vdb configuration:

```
{
  "certificate":
    "MIICPjCCAY6gAwIBAgIIMtkhw81Tf9gwDQYJKoZIhvcNAQELBQAwEzERMA8GA1UEAxMISGVpbWRhbGwwHhcNMjEwOTE5MDgxOTQ1WhcNMzEwOTE5MDgxOTQ1WjATMREwDwYDVQQDEWhIZWltZGFsbDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAIrsQBhUqK5azbMyPhgoFo2oJvP25Rr03y0IKZiE1ELjIAAnR80KROSA4kX6doUDtWDvTmasmJw5JaE+UUVpv+MpGo01qYooiDi7pb+e+DhE5bzKbkg1Qbm57ipv3JPHNwoLArtCTtTkkuyiZcyVd5GkkfEWnEqzRQ5r8JvHsTTX4fSf45ou1qqEv1DhH02zUr8s6YEE14rtoF2cNyiuz6A9Wg1ydRTS4Lv6RBUEJuc99DQdWSdQ567ptEBI7yJxvRYTRdLpqakPNRVkEzOIQUU1KrQrmkswo010jd/5k1FA6c0xf11DS3oy6LcPEFxf4cNIrinGCQcXjfh7PAg9cCAwEAATANBgkqhkiG9w0BAQsFAA0CAQEAAASifK1xhA93AMDea9nKMA/FZ2aYIqlxFc57lIFeftG5zAmPtqbBXTL9VBnZ89gHAbWc7Dj0c/rkmHrxG2Zh16ofv8MY105IKfI65MNVXk/TUPulz0e8/190msVUrIA8Kx34Ip4WtXVgoHUAf95u3wvZXQCN4x41eZ0i0lzAYCXEmAdjH063vUnOr7A3i581pJH9NETEeysuyH2vtSoxpiC5c5q7RrXvYTARZc/n5eZFqrJSIohz9fqvsJ4oTath9XvacENs0g7lJyALmpksGzs+NYe/LkGcqWFljjEAduK4e7wypRDB2eXevA3stIxuIfdLtNu03EzxpCqsFH/hQg==",
  "certificateAlias": "global_use_certificate",
  "privateKey":
    "MIIEvAIBADANBgkqhkiG9w0BAQEFAASCByYwggSiAgEAAoIBAQCk0kAYVKiuWs2zMj4YKBaNqCbz9uUazt8tCCmYhNRC4yAAJ0fNCKtkg0JF+naFA7Vg705mrJicOSWhPlFFT7/jKRqNNamKKIg4u6W/nvg4ROW8ym5IJUG5ue4qb9yTxzcKcWk7Qk7U5JLsomXM1XeRpJHxPxpKs0U0a/Cbx7E01+H0n+0aLtaqhL5Q4Rzts1K/L0mBBJek7aBdnDcors+gPVoNcnUU0uC73+kQVBCbnPfQ0HVknU0eu6bRAS08icb0WE0XS6ampDzUVZBMziEFFNSq0K5pLMJaDtTo3f+ZJRQ0nNMX5dQ0t6Mui3DxBcX+HDSK4pxgkHF434ezwIPXAgMBAAECggEAFbjmVWKulaEk8+pHtCbyLf6gsIfke3/pPOZMjZWwirEZtSnd7AM123Ah/NBVcCeop45zCae8WpeizcIbZlNQbFE4e7Siw73riL3g4Yb+NC+phG4srGgZez05T30ZZ7iezxijs6XcsRAFRavbeAnRjPRn09mHWFu86c+1y89xXHHZ3DNEYvxZ98UW1ZwuU0/iyke81A7ec+hpjKPMoBPesdF1+EBgSoId8oAmbjCOA1dRpe2N4N6rYCTqq0P8mmeSnoJc9fMXLmNhrBMu3e2qsUteVJUrrD14F9uv4v1Cszk9uwcg4Ced5jK2Qc1Xid0S+WH6QgLe7XCURHM07UISQKBgQD137/VpBXe4FNCb3LR6+ZSDJKGLFAEx8KCFrERci//sYW6RYfoe7cdWPI0vR4D+04HBYXSHz+W6Sy0dnse/ch58aBG+323bMPmxvXWIf8+q/uoYDAmnbgXLKLx4mjAa+UNp6wB3QoEeImD5c14SHtTl8eyMSpaPlyb1kY0I5lQKBgQCQidf9nQBZO
```

```

DsXT8DSV1E5brc4nBgU1NLsVwRv08y8n/8EILS3kduH3Qb699YhNqzK3YD/
ttMM+c4sWqICG58ZYB03X0fOK1CnIILU0jgHCWYBihMH+s9qEbr0jU7t1VzBm0t/89CC1brLENr/
YcK3I5lUw/PzE4Y07U4+ckmkuwKBgDJGu3pbICG/
6yaF1Yv5WdJjQb7v5rjjhBGctVF3F5Ky7RyYMu7q90Xd1zbkdBrs3Nqz24yNRQnaAXaTVKV3v869BS
k3ZdR6Bvhk9i91qfD/7e004JgBvdco/
VV266XEMkj0Ta6WBEK504bJ4s6nPzFkzE15AHX6quX89szhQxR1AoGASUzesMtyZczSitzWbjWhF3
VpWvj+mwPG7R1l0ObPBnD9zWSVI7FwfMhnIxsd5cevUieoxPM+dr26qLBq6jX0cQmKpFbBUjbd5a+
u4bg9YMWTOAu0dC8PPU8DjJkmvmAnhe1iD72Z4ygNEfuRBL0dqd0Z47Qk1wpmmWJz1JAKCgYA3e8q
U2uF+NUsYR0ZqfpFLcTm2R2C0sjNIffb6wv4v3zoVIFBnViVnxs2JcSdkNz1I/
WeQx5HbVyFTZJPqLSwxQ6k63y2THu8sLbjfRkLERAEAU/SdqrjcyiqQd5r/
Oa8xbzFvTBpnIWoJzyNSnaeGWNUGqgub8I1s4eVwS/Y3zQ=="
}

```

All changes in key pairs should meet below restrictions:

- alias of key pair should match the restrictions defined for particular and global use certificates, other defined aliases won't be used by Heimdall Server
- key pair password is blank by default using global_use_certificate
- recommended is using key pair with algorithm SHAx with RSA/DSA (not guaranteed that all algorithms is going to work, because it depends on client application too)

Changes made in keystore may require reconnecting from client application to proxy, after this changes should be applied for TLS.

Example scenario of configuration

Simple scenario of creating keystore, and adding global use certificate to keystore:

1. Run application to manage keystore (for example: KeyStore Explorer).
2. Choose option to create new keystore.
3. As type of keystore select type `JKS`.
4. Save created keystore.
5. As password to keystore set `heimdall`.
6. As directory to save choose directory where Heimdall server is placed.
7. After saving, choose option for generating new key pair.
8. Set size of key size as 2048 and as algorithm choose SHA-512 with RSA.
9. Set name of key pair by configuring CN to value `Heimdall`.
10. Go further to find option to configure alias, as alias set value `global_use_certificate`.
11. As password to key pair set blank value.

12. Key pair should be fully configured, save changes in keystore.

Extra informations

This section provides informations which didn't found place in other sections about TLS but may be helpful: - format for key pairs in Heimdall server is X509

Iptables

If you are using Iptables, which blocks ports by default, Heimdall provides a solution. When starting the proxy, the ports for both the proxy and the server are automatically unblocked in iptables. If you cannot connect to the server, try restarting the proxy. It's possible that the management server was not ready yet, and the ports were not unblocked. Moreover, the server also autonomously unblocks its port in iptables during startup. Importantly, the iptables service doesn't exist on Windows systems, so the server and proxy ports won't be unblocked there.

Testing Heimdall

Many customers who are interested in testing Heimdall will follow these steps:

1. Install Heimdall;
2. Use the wizard to configure access to their database;
3. Use a tool such as DBeaver, PGAdmin, or other tool to generate traffic;
4. Observe that despite caching and read/write split being enabled, neither work;
5. Assume Heimdall doesn't work, delete and move on.

The critical flaw here is #3--using tool designed for easy user access to the database to try to test these features. They will not work (in general), for a variety of reasons:

1. Often, they will use mechanisms such as cursors to execute a query, and a cursor is inherently allowed to modify the result on the database. As such, this breaks caching and read/write split;
2. Even if they don't use cursors, they generally apply "pagination" techniques that also break caching;
3. They may use prepared queries, which can break other features as well, such as query multiplexing.

The issues also apply to some benchmark tools, such as PGBench, which is designed specifically for cache busting and leverages cursors as well.

The best way to test Heimdall is with your actual application, not with database user access tools.

Inline SQL Commands

When connecting to the database, Heimdall exposes a few commands that are handled directly by Heimdall vs. by the database. Note: These interceptions ONLY occur when using the simple query protocol via a tool such as PSQL. If your tool uses a prepared interface, as many GUI tools will, these command will not be intercepted.

Show commands

The key use cases for Heimdall database proxy include:

(Postgres only)

- show databases
- show schemas
- show tables
- show users

All databases:

- show pools: Displays a list of user pools and relevant metrics.
- show connections: Displays a list of connections to Heimdall VDBs, along with the following columns and their descriptions:
 - Con ID: Unique identifier for each connection.
 - Conn Age(s): Duration the connection with Heimdall has been active, measured in seconds.
 - Idle(s): This represents the duration in seconds when the connection remained inactive.
 - Start(s): Elapsed time in seconds since the start of the current query execution, or "NA" if no query is currently running.
 - User: Username associated with the connection.
 - Catalog: Name of the current catalog.
 - Autocommit: Current state of the auto-commit mode for this connection object.
 - sourceName: Name of the datasource to which the connection is established ("none") if there is no underlying datasource connection.

- Server URL: URL used to connect to the datasource ("none") if there is no underlying datasource connection.
- Con properties: Properties of the VDB connection ("none") if there is no underlying datasource connection.
- Tables: Tables involved in the currently executed query.
- show querytracker <tablename> <count>: Displays relevant tracking information for queries tracked in Heimdall, table and count are optional. Only used for cache auto-repopulation (rarely used)
- show queryinfo: Displays attributes associated with the previous query.
- show maxrows: Shows a connection limit on the number of rows any given result-set will try to pull. The default value is 0, meaning the full result-set will be pulled.
- show cache <key>: Displays cache attributes associated with the given key. Doesn't matter whether key starts with "hdkey" prefix.

Control Commands

- set querytimeout: Sets the number of seconds the driver will wait for a Statement object to execute to the given number of seconds. If the limit is exceeded, an SQLException is thrown.
- set cachepool: Adds a value to the cache key to provide unique cache "pools" depending on external factors, unobserved by Heimdall
- set maxrows: Sets a connection limit on the number of rows any given result-set will try to pull, as implemented via the jdbc "setMaxRows()" api.

Other Commands

- drop connection: When in proxy mode, force the connection to drop--useful for using test tools that don't close connections on their own.
- clear pool: Clear server-side connections for current user.
- clear pool <userpool>: Clear server-side connections for a particular user pool, <poolname> must be the username:databasename as displayed in the show pools command.
- reset query cache: Reset all query cache for a virtual database.
- reset query cache <tablename1>,<tablename2>: Reset query cache for all listed table names in CSV format.

Note: To issue clear pool, the user must clear their own pool or be authenticated as one of the users: "postgres", "gpadmin", "root", "sa" or "admin". This will be changed to be configurable in the future.

Compliance

GDPR

The Heimdall Proxy and Manager do not store customer information beyond possibly SQL logs, and this is optional. Further, the "Paranoia" log option will strip SQL query information that may be considered sensitive from any logs.

PCI/HIPAA and Query Caching

When caching, Heimdall stores data in-memory only, and as such is considered in-flight. When connected to an external cache via Redis, an option is provided to use TLS for over the wire encryption.

The Redis configuration should be such that it does not store data to disk (this is the default for Elasticache and most other Redis deployments).

To avoid storing actual cached data to the l2 cache (Redis or Hazelcast) but to only use them for invalidation, you can set the "PCI" option in the cache settings.

SOC2 Compliance

Heimdall does not operate software as a service, so usage of the Heimdall Proxy and associated management components does not fall under SOC2 compliance.

Security Best Practices

By default, Heimdall negotiates the highest level of TLS supported by the database in question on a VDB when TLS is enabled, and will negotiate to the highest level of security common between the client and the version of Java used by Heimdall (default is Java 11). This typically results in TLS 1.2 connections except for SQL Server, as many drivers do not support negotiating with TLS 1.2, so 1.1 is used. If deprecated versions of TLS are needed for old clients, they must be enabled by using the "enable legacy TLS support", but this is not recommended unless needed and appropriate physical security over the connections is assumed.

Heimdall provides a UI interface for uploading of TLS certificates for use with the GUI and VDBs.

Configurations (including all passwords) can be stored in one of supported Secrets Managers.

Export Control Classification Number

Heimdall falls under ECCN 5D992.c. It does not implement itself any cryptographic functions, but instead uses the functions available from Java for the implementation of TLS.

We package by default with the open source OpenJDK, which also falls under ECCN 5D992.c per <https://www.oracle.com/us/products/export/eccn-matrix-software-412042.pdf>.

Our object code is available for public download to further compliance with 5D992.c.

Help and FAQ

Possible runaway thread

When this alert is generated, it is simply meaning that the named thread has taken nearly 100% cpu on a single core. This can be for many reasons:

1. A client application is reusing the same connection with little to no idle time, and is normal;
2. A query with a very large result-set is streaming the result for more than a minute;
3. A backup is being made via a connection, and is saturating a single core;
4. Related to #2, but possibly a malicious user has gained access to the system, and is downloading lots of data;
5. Also related, but if the proxy is open to the internet, someone may have hacked an account, and is downloading all the data from the database;
6. There is a bug in Heimdall, and a thread is in an infinite loop.

If the thread name contains an IP address, this is a thread being used to service a connection, and likely fits one of the first five cases. If it doesn't contain an IP, please provide logs and a screen dump of the alert for analysis by Heimdall support.

VDBs

VDB Overview

Virtual Databases

Create New VDB

postgres-vdb

☒ Enable

Commit

Configuration

Access Mode:

PostgreSQL Proxy

Access/Secret Keys:

PoKqs7wjjJaVDMv

Secret

☐

Refresh

Copy

Eye

Deployment Environment:

Staging

+ Proxy Configuration (LocalProxy, 5433, Heap:600, TLS, Auth:Proxy)

+ Caching (Local, Limits: Age:5, Size:64000000b, Count:100000)

+ Data Sources (postgres-source)

+ Rule Lists (postgres-rules)

+ Advanced Features (Multiplex, DelayedTran, AutoScaling)

+ Logging (SQL, Aggregate, File Log)

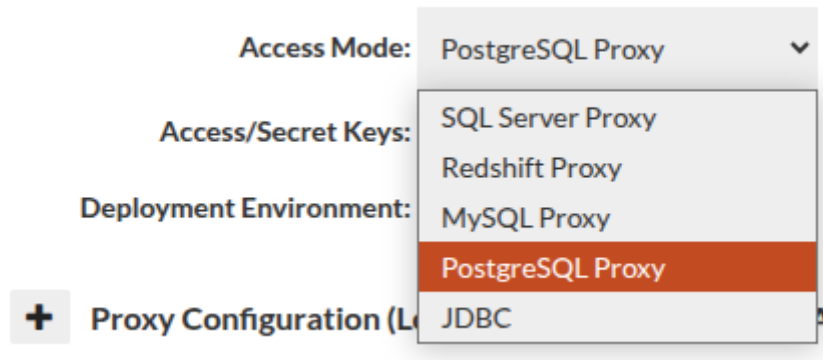
+ Debugging (Verbose, Methods)

+ VDB Properties (1)

Test Cache

Virtual Databases (VDB) are the basic container for configurations, and aggregate all the settings that will apply to any traffic passed through the VDB. There are two ways to access a VDB, first via JDBC, which supports any JDBC compliant data source, or via a protocol level proxy. In proxy mode, a separate process will be created, and database traffic would be routed through this proxy using a specific port

General configuration options can be found in [General Configuration](#) Hint: When using secrets manager, you can point to the same secret as is managed by RDS, so as to reduce the overhead of management. This is useful to ensure that all secrets rotate together based on policy and supports cross region and account management in conjunction with the CF template.



JDBC Configuration

- JDBC Configuration

JDBC URL:	jdbc:heimdall://<Heimdall Server IP:port>/Magento-Demo-vdb?hduser=<u:
JDBC Class:	com.heimdalldata.HeimdallDriver
JDBC DataSource Class:	com.heimdalldata.HeimdallDataSource
JDBC XA Class:	com.heimdalldata.HeimdallXADataSource

When in JDBC mode, a JDBC URL is provided in the GUI, which is used to configure the application to access the Heimdall Data Access Layer. The JDBC URL is of the format:

JDBC:heimdall://IP:port/{vdb name}?

Optionally, several parameters can be used in conjunction with the Heimdall Connect string:

- **hdUser**=**{username}**: The username to authenticate against the Heimdall server with.
- **hdPassword**=**{password}**: The password for the Heimdall server, again overriding the parameter in the data source.
- **user**=**{username}**: The username to use to authenticate against the database with. If the event the hdUser is not specified, this will ALSO be used as the default hdUser value, in cases where the users on both match.
- **password**=**{password}**: The password for the database server. In the case where the hdPassword is not specified, this will also be used as the default hdPassword. Note: When using the user and password, the data source credentials will still be used for health checking the application and should have appropriate access for this task.

Note: Typically, only the hdUser and hdPassword is used in the Heimdall JDBC URL, as the data source can provide the user and password options. The ability to specify this here is to allow

applications to specify their own username and password, in general to allow many users to access a data source at once.

Proxy Configuration

Proxy Configuration

Local Proxy: ☒

Run as Service: ☐

Address Binding Type:

Any

Proxy Port(s):

5450

Max Heap Size (MB):

600

Proxy TLS Support:

☐

Authentication mode:

Proxy Configured Users

admin

.....

x

Secret

☐

(.*)

secret/[\$1]

x

Secret

☒

Regex

☒

Allow Portal Users To Log In:

☐

Synchronize DB Authentication:

☒

Tokenized Authentication:

☒

Admin User:

root

Secret

☐

Admin Password:

☒

Sync Command:

Sync Interval:

0

Auth Cache Expiration Time

300000

Authentication Test

Test User:

user

Secret

☐

Test Password:

password

☒

Test IP Address:

127.0.0.1

Test Database:

db

UseSSL:

☐

Perform synchronization:

☐

Test Authentication

Proxy mode operates by having the management server either start a process on its own, or having a proxy on a remote system connect. Once started, the proxy will open a port that a client can connect to as if the proxy were the database itself, and access data via that process.

When in proxy mode, several options will be available, two options being required: the address binding type, and the proxy port. The address binding type specifies the behavior of the proxy.

Mentioned options and several other options are available in the proxy configuration: [Proxy Configurations options](#), this list contains all the configurable options regarding **Proxy Configuration** section.

Synchronize DB Authentication: ☒

Tokenized Authentication: ☐ ⓘ

Admin User:	test	Secret ☐
Admin Password:	••••	☒
Sync Command:	select public.sync_user("\${user}";"\${password}";"\${ldapgroups}");	
Sync Interval:	0	
Auth Cache Expiration Time	300000	

Authentication Test

This section allows testing of authentication at the proxy layer. As full SQL based authentication can provide options such as limiting the scope by source IP, what database (these are taken by default from Data Source JDBC URL), or even if you are connecting via SSL or not, this gives a convenient UI to determine if an authentication request would pass or fail the proxy layer. Please note--this still requires authentication at the database level, which is not tested via this interface. More details [here](#). Note: The username and password can be stored in the [Secrets Manager](#).

There are two basic results that this test can end with:

Authentication successful:

Authentication Test

Test User:	test	Secret ☐
Test Password:	••••	☒
Test IP Address:	127.0.0.1	
Test Database:	db	

UseSSL: ☐

Perform synchronization: ☐

Test Authentication

Authentication successful

Authentication failed:

Authentication Test

Test User: test

Test Password: ••••••••

Test IP Address: 127.0.0.1

Test Database: db

UseSSL: ☐

Perform synchronization: ☐

Secret ☐

☒

Test Authentication

Authentication failed

• **Perform Synchronization:**

This section also allows us to perform synchronization of user and groups on test by enabling "Perform synchronization" (Synchronization query is retrieved from "Synchronize DB Authentication").

Then some more results can be observed:

Authentication Test

Test User: test

Test Password: ••••

Test IP Address: 127.0.0.1

Test Database: db

UseSSL: ☐

Perform synchronization: ☒

Test Authentication

Authentication successful, but synchronization failed

Please be aware that if JDBC URL given in the Data Source contains either `{host}` or `{database}` respectively Test IP Address or Test Database has to be provided. Anyway, in this

case if anything is wrong with the configuration, the additional information will be provided as a hover to the button.

Caching

Caching

Test Cache

Enable Query Caching: ☒

PCI/HIPAA Cache: ☐

Cache Type:

Redis/Elasticache for Redis/KeyDB

Server:

127.0.0.1

Port:

6379

Database Number:

0

Cache Username (optional):

Secret ☐

Cache Password:

☒

Use SSL: ☐

Grid cache offload: ☒

Use soft references: ☒

Duplicate request tracking: ☐

Max Expiry:

5

Max Cached Objects:

100000

Max Cached Size:

64000000

Customize Query Cache Key: ☐

Cluster manager via cache: ☐

This set of options configures the base cache used by a given VDB. There can be only one cache per VDB at any time, and if the cache settings for a given type are to be changed at runtime, it is required that the cache be disabled first, then enabled again with the new settings. This will allow the cache to be completely torn down, and reinitialized, allowing the entire type of cache to be changed without application restart.

List of configurable VDB caching options: [Caching](#)

Note: With Amazon's AWS Elasticache service, the Redis parameter group option of "notify-keyspace-events" should be set to the value of "AE" in order to optimize cache behavior. This will also be instructed in the log output. For non-Elasticache Redis servers, this option will be configured automatically.

Note 2: Memory allocation for cache is dynamically controlled based on free space allocated to heap. Use the VDB setting of "xmx" to adjust this (in vdb properties). It defaults to target about 1GB of total memory used by the proxy process.

Test cache: The "Test Cache" button allows the user to verify whether a cache connection can be established. Additionally, it checks if key tracking is enabled. If key tracking is not enabled, an alert message will be sent to inform the user about it. After reloading the configuration, make sure to wait for a minimum of 10 seconds as the cache will need to restart. Failing to wait can lead to incorrect test results.

Data Source and Rule List

Data Sources

Data Sources:

Choose one or more...

Magento-Demo-source

Rule Lists

Rule Lists:

Choose one or more...

Magento-rules

Specify at least one data source, as a default for data to be retrieved from. If a forward policy is specified in the rules for a vdb, it also must be selected here to insure proper connectivity is established to that data source for the forward function to work properly. A connection to the data source will only be established when used if not the primary data source. A reasonable attempt will be made to insure that the data sources for forwarding and read/write split are automatically populated here, but in some rare corner cases (with dynamically generated properties), all data sources will need to be specified here.

Like the data source, the rule list selector configures what rules should be attached to the vdb. If empty, no rule list will be used. All rule lists that are used by the initial (default) rule-list must be specified or they will not be executed. A reasonable attempt will be made to insure that the rules used in "call" actions are automatically populated here, but in some rare corner cases (with dynamically generated properties), all rules will need to be specified here. The log may generate warnings if this is not setup properly.

Note: Only the first rule-list specified will be executed by default. In order for additional rule-lists to be executed, they must be referenced from a rule with an action of call.

Advanced Features

Advanced Features

Multiplex:

☐

Delayed Transaction:

☐

Auto-Scaling Mode:

☒

Load Balancing Criteria:

Load

▼

Proxy Redirect Name:

Proxy Private IP

▼

DNS Port:

0

↕

Track Query Distribution Count:

1000

↕

Paranoia:

☐

Health Check Port:

4444

↕

Health Check Interval:

5

↕

Enable Token Authorization:

☒

Security Token:

.....

↺

👤

👁

JMX Port:

1099

↕

JMX Hostname:

\${ip}

JMX Username:

admin

Secret

☐

JMX Password:

.....

👁

To see all configurable items please see list of [Advanced Features](#).

Multiplex and Delayed Transaction Considerations

- **multiplex:** Do load balancing to the server at the transaction level vs. the connection level. This can drastically reduce the number of connections being established to the server. Requires connection pooling to avoid a dramatic performance drop, and may have side-effects that break certain apps. Test carefully before using in production. DelayedTransactions and multiplexing can be used together to improve the query distribution for apps that support them. Examples of features that if used can cause application breakage with multiplexing include Postgres's search_path (then user must execute a query with the full path to the table name), and MySQL session variables. Any "state" will effectively be lost between queries that are not in a transaction when this feature is used. With this option, intelligence is applied to pool in the most effective mode,

similar to how PGBouncer provides session, transaction and statement level pooling. Unlike PGBouncer however, you don't need to choose one mode, as it will intelligently switch between modes depending on the situation. As an example, if we determine a temporary table is created, it will switch to session level pooling until the table is released, then switch back to statement level pooling. If a prepared statement is created, this will also happen automatically. This provides the best pooling behavior without additional configuration on the user's side.

- **delayedTransaction:** (proxy only) Delay when a transaction is started until a DML is detected. This is useful when using a framework that does everything in transactions. Rules set to not match in a transaction will operate until a DML is encountered, then will not match until a commit/rollback. There may be other performance benefits on the server side as well depending on server. Test carefully before production use. If Delayed Transactions doesn't work cleanly, specific queries can be cached or forwarded in a transaction with the in-trans flag instead, but care needs to be taken to use this option.

JMX Configuration Hints

In the VDB->Advanced section, there is an option to set the JMX hostname. With JMX, it connects twice--once to the name provided by the tool, such as VisualVM, and a second time using a redirect hostname. The hostname must resolve on redirect or JMX will fail. The default value "\${hostname}" will use the configured hostname of the VM. This can also be seen in the status tab for the hostname of each VM.

If the hostname does not resolve in VisualVM, then connections will fail. One way to fix this is to use the hosts file on the VisualVM system, or adjust the hostname so it creates a qualified hostname.

Example: If the hostname resolves to ip-10-0-0-1, but you want ip-10-0-0-1.domain.com, you can configure the hostname field in the VDB to be: \${hostname}.domain.com, so when it redirects, the client will be able to connect properly.

For testing purposes, or with only one proxy with a fixed hostname, you can set an explicit hostname here as well.

Additionally, please disable using TLS on connecting, as this option is not currently supported via configuration, as JMX is a rarely used feature, and only internally to a customer's network.

Auto-Scaling Mode Without an External Load Balancer

If the DNS based autoscaling is to be used without an external LB, a few considerations need to be made. First, the Heimdall system should not have any other processes listening on port 53, including Systemd. Please see your distributions documentation to find how to do this, but

it typically requires disabling systemd-resolved, and then setting the NetworkManager to use the legacy DNS behavior. Next, set Heimdall to listen on port 53 for DNS, as inbound DNS requests will use this port. Finally, delegate the shared name you are using as a proxy with an NS record to the Heimdall instances themselves.

As an additional note, with no external load balancer to do draining operations on a transition event, it is highly recommended that the application behavior be tested to ensure that dropped connections will be gracefully re-established. An external LB device can help avoid issues, but even with them, the drain time and max connection lifetime should be in sync to ensure that draining can happen without connection disruption.

Logging

☐ Logging

Log Connections:

☐

Log All SQL:

☐

Log All Portal Sessions SQL:

☒

Exclude parameters:

☐

AWS CloudWatch Logging:

☐

AWS CloudWatch Metrics:

☐

AWS Logs Namespace:

heimdall_cw_logs

Aggregate Console Logs:

☐

Write Logs To File:

☐

Log Authentications:

☐

The following log options are available: [logging](#)

- **Important:** When logging SQL, a large amount of data may be generated, and logging can impact the overall performance. To avoid this, you can disable sql logging here, and instead use a log policy with parameters that limit how much data is written. Please see

the rules section for details on parameters and the logging section for more details on logging overall.

Debugging

☒ Debugging

Verbose Debug Mode:

☒ [\[!\]](#)

Connection Handshake Tracing:

☐

Lite Debug Mode:

☐ [\[!\]](#)

Log Methods:

☒ [\[!\]](#)

Log ResultSet Methods:

☐ [\[!\]](#)

The following debug options are available: [debugging](#)

Important: When adding in method logging, for every query generated, it may result in a dozen records for JDBC methods, and with resultset methods, every row will likely have one or more log records. Caution should be observed when using these two log options are used, and should in general only be used in low-volume lab conditions.

VDB Properties

Allows a list of name-value pairs to be used to configure various options, in general per direction from customer support, and for use with test releases to enable a fix. Example:

Vdb configurable properties can be found in: [VDB properties](#)

Data Source Overview

Data Sources

Create New Source

postgres-source ☒ Enable

Commit

Configuration

Driver: postgres-driver +

JDBC URL: jdbc:postgresql://localhost:5432/\${database}

Username: test

Password: ****

Test query: SELECT 1

Test Connection (Passed) ✓

Execute scripts

Secret ☐ Kerberos ☐

- Connection Properties

Custom Property ☐ +

defaultCatalog postgres

+ Connection Pooling (Disabled)

+ Load Balancing/High Availability (Disabled)

+ Group Mapping (0)

+ Roles Management (0)

+ Database Browser

A data source represents access via JDBC to a back-end database server. Each VDB definition must have at least one default data source associated with it, which will be used if no policy dictates what database should receive a query. Source configuration includes the connection properties, connection pooling, and Load balancing and cluster management characteristics.

The **driver** field allows either a new or existing JDBC driver to be specified. Heimdall comes with a small library of JDBC drivers that it has been tested with, but other drivers can be configured in the driver tab as needed. When using the proxy functionality of a VDB, the tested drivers should in general be used, as some internal and version specific features of a driver are often used by the proxy to provide compatibility with the binary layer translation that occurs.

The **JDBC URL** configuration will normally be build through the configuration wizard, or can be copied from an existing JDBC application. Properties on the JDBC URL can be moved to the name-value list in the connection properties, OR can remain in the JDBC URL—either works,

although using the name-value list provides an easier to view list vs. a long URL. When internally doing database type detection, the JDBC url is also used to determine what type of database is in place, for per-database behavior adjustments. When using load balancing, this URL is not in general used, with one major exception—when using the `hdUsePgNotify` option, this will be the target used for notifications. This is necessary as Postgres notifications will not be shared between nodes, and as such this should point to a single name that resolves to a single target database.

The data source **username** and **password** are used for several purposes:

1. For health checking the data source;
2. When performing lag detection, to write to and read from the Heimdall schema;
3. When doing cluster detection, reading cluster variables as appropriate, or when Heimdall needs to extract metadata for other purposes;
4. When no user or password is specified in JDBC mode, this user will be used to connect to the database;
5. In proxy mode, if proxy authentication is not provided. In this case, any user can connect, but the DB connection will be made with these credentials.

In some cases, it is necessary to specify various parameters in a dynamic way, i.e. with the Odoo application, it changes the active database at runtime, and connects to the desired database as desired. To support this, fill in the following values as necessary:

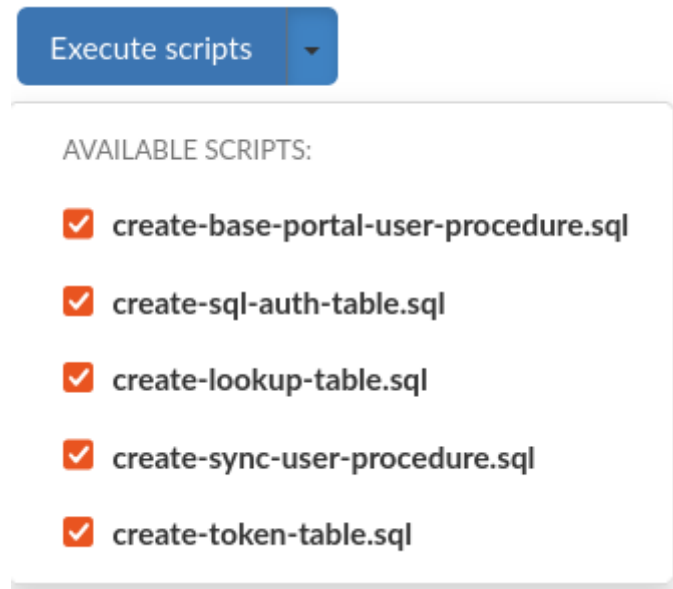
- `${catalog}` or `${database}` to specify the name of the database (either will work), use "defaultCatalog" connection property to set this for health checks and sql auth purposes
- `${schema}` for the specified schema
- `${user}` for the connection user
- `${password}` for the password specified on connect (only for Postgres, Redshift and SQL Server)
- `${host}` for the hostname provided when using SQL or secret based authentication, use the "defaultHost" to provide a default value when it is not provided, i.e. for health checks and sql auth purposes (to host the mapping table)
- `${application_name}` for the application name provided on connect

The **test query** option is used in a variety of situations, including health checks, and verifying that a connection is still viable after an exception. It is also used in a variety of conditions when pooling is configured, in order to validate the connection while idle, etc.

To validate that the options appear valid, please use the **Test Connection** button. This will initiate a connection through the vendor driver by the Heimdall management server, and acts

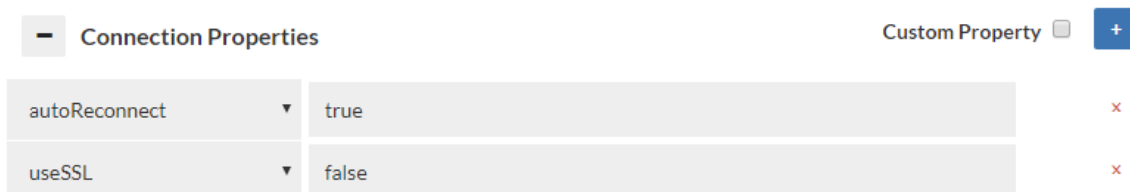
as a general guide if the configuration appears valid. It is not a guarantee that under all conditions it is valid, however.

Scripts



To automatically execute a set of SQL scripts on the data source, use the **Execute scripts** button. This will attempt to execute selected scripts located in `/config/scripts` in the Heimdall installation directory. Scripts are grouped by database type into directories. Predefined files are supplied at Heimdall startup to these directories, these use the "heimdall" schema by default, but if you choose to use a different one, its existence must be ensured. You can modify them or create Your own to be able to execute them via GUI, but it is advised to do so in separate files, named differently than the default ones, otherwise these might get overridden at startup. If Heimdall was recently updated, on first start there will be a backup of previous version of `/config/scripts` directory created.

Connection properties



Connection properties are database and driver specific. When in proxy mode, a default set of properties will be inherited by the source in order to provide compatibility with the proxy mode used. In JDBC cases, no default properties will be set by Heimdall. When using a known driver, all supported connection properties will be provided in a drop-down list, and tool-tip help provided for each one when selected. When using an unknown driver, the property name field

will allow an arbitrary name to be specified and value provided, as per your JDBC driver's documentation.

In order to add a custom property (in the event it isn't in the drop-down list), you can use the custom property checkbox next to the blue plus, to provide a free-form field.

Connection properties list can be seen in [connection properties](#)

Please see the UI for a more complete list, including vendor options, as the drop-down list provides context-sensitive help on each option provided.

Lists of connection properties for given databases can be found under: Options → Connection Properties. E.g. [MySQL connection properties](#)

Connection Pooling



The pooling section controls if connection pooling (the reuse of back-end connections for many front-side connections) is used internally to Heimdall, and if so, what the connection pooling properties are. Heimdall implements a variant of the Tomcat connection pool, and where appropriate uses the same properties for configuration.

The primary difference between the Heimdall and Tomcat connection pool is that the Heimdall pool implements it as a "pool of pools" where each user/database (catalog) combination becomes a pool within the connection pool, but overall resources are constrained by the overall maxActive option, which dictates the overall number of busy AND idle connections that can be established to the data source. Additional properties of maxUserActive and maxUserIdle dictate how many active (busy+idle) and idle connections are allowed on a per-catalog+user combination. This provides much more granular control over the behavior of the connection pool when many databases and users are connecting to the database server.

idleSessionTimeout note

If the database has configured a type of idleTimeout (currently supported: idleSessionTimeout for postgres(version 14+), wait_timeout for mysql) if the value is detected sessions in idle pool will be deleted with an increasing chance in time based on idleTimeout, it is done to prevent having closed connections in the idle pool. If connections are not deleted fast enough consider changing the timeBetweenEvictionRunsMillis pool property.

The properties section provides a drop-down list of the available properties, and tool-tip help for them. The available properties can be found here: [pool properties](#)

For additional information about each pool parameter, please see the Tomcat pool attributes page (<https://tomcat.apache.org/tomcat-9.0-doc/jdbc-pool.html#Attributes>).

Warning: Not all applications will be compatible with connection pooling. If issues are observed in initial application testing, it is recommended that disabling connection pooling be performed in order to isolate if this feature is triggering undesirable behaviors in the application. If using the connection pooling, the default number of connections is set to 1000. To adjust this, change the “maxActive” setting to the desired number of connections.

Connection Pooling options

There are commands specific for connection pooling such as **show pools** and **clear pool** described in the [Inline commands](#) section.

Postgres only: There is an option to clear a pool when making a connection to it is impossible due to the pool already being at maximum capacity. Please see [Clear pool option](#).

Load balancing and High Availability

Load Balancing/High Availability

Enable Load Balancing: ☒

Track Cluster Changes: ☒

Use Response Metrics: ☒

Track Replication Lags: ☒

Connection Hold Time (ms): 30000

Target write capacity: 1

Target read capacity: 10

Response Metric Multiplier: 2

Lag Window buffer (ms): 10000

Alert threshold (ms): 0

Failover Script:

AWS RDS ARN: arn:aws:rds::272965818115:global-cluster:test-a

Name: test-a-writer1

Url: jdbc:postgresql://database-1-instance-1-us-east-1a.cfjxl5jnvj49.us-east-1.rds.amazonaws.com:5432/\${database}

//

Enabled: ☒ Writable: ☒

Weight: 1

Write Capacity: 1 Read Capacity: 1

Name: test-a-reader1

Url: jdbc:postgresql://database-1-instance-1-us-east-1a.cfjxl5jnvj49.us-east-1.rds.amazonaws.com:5432/\${database}

//

Enabled: ☒ Writable: ☐

Weight: 1

Write Capacity: 0 Read Capacity: 1

Due to the complexity of this topic and how the options interact with other subsystems, please see the help section dedicated to [Load Balancing](#) for more details.

Group Mapping

Group Mapping

+

LDAP Configuration: HEIMDALL_LDAP_CONFIG ▾

Group Membership	Local Database Role	Add Role	Delete
admins ▾ ☐	portal_admin ▾ ☐	+	✖
	odoo ▾ ☐	+	✖
heimdall-(*) ☒	\${1} ☒	+	✖

The Group Mapping feature provides a flexible approach to associate a group (e.g. LDAP) with specific database roles. In the absence of group mapping, Heimdall defaults to mapping groups to roles with identical names. This option applies when the heimdall sync_user script is used with LDAP authentication. Group mapping will also be triggered when user session is created via portal. Users can configure group extraction based on the group regex when the "Use Regex" checkbox is enabled, if checkbox is disabled, none of the regexes will be applied to the mapping. Heimdall supports the use of capture groups within regex patterns. This enables the extraction of specific substrings from group names for role mapping. For example, if configured as on the screen, the 'heimdall-admin' group will be mapped to the role '\${1}' which leads to 'admin' substring. When no mapping is configured, this option can be used as "deleting" group.

The Group Mapping feature also affects the roles that are granted to the user after successful authentication in the synchronization process, via Ldap Auth, Kerberos + Ldap, SQL Driven Auth / Secrets Manager Auth. For managers, Group Mapping also governs the roles requested

via the portal, such as when using SAML authentication with group extraction or any other authentication method that results in group extraction.

Roles Management

Roles Management

Database-Native Session Cleanup: ☒

Require justification pattern: ☒

Regex pattern:

Invalid justification message:

Filter system roles: ☒

Role Name	Notification Alias	Default Time	Maximum Time	Approvals Needed	
admin	alias	30 m	16 h	1	
myrole	notify	1 h	1 d	1	
myrole2	alias	1 h	1 d	2	
myrole3	none	1 m	3 d	0	

- **Database-Native Session Cleanup:** By default, the Central Manager (CM) is responsible for clearing ephemeral user sessions. If an ephemeral user session ends (due to expiration, cancellation, or termination) when the CM is offline, the session will not be immediately cleared. In this situation, the ephemeral user will not be locked, any active connections for that user will not be closed, and the user will not be dropped from the database. As a result, the user will still be able to access database resources until the CM is back online and performs its normal cleanup tasks.

When this option is enabled and a session ends, the ephemeral user will be locked, which will block any new logins. In MySQL and Oracle, all active connections for that user will also be forcibly closed. This ensures the user will be unable to perform any operations on the database after the session ends, even when the CM is offline, because the cleanup relies on the database's native mechanisms rather than the CM. The specific mechanism used will depend on the database type.

- **MySQL** - Requires the [MySQL Event Scheduler](#) and procedure called [create-clear-session-user-procedure.sql](#). Click [here](#), to see how to set up event scheduler in AWS.
- **Oracle** - Requires the [Oracle DBMS Scheduler](#) and procedure called [create-clear-session-user-procedure.sql](#). Click [here](#), to see what is required to schedule jobs in Oracle.
- **PostgreSQL** - the [VALID UNTIL](#) clause is used on user creation. From the documentation: *"The VALID UNTIL clause sets a date and time after which the role's*

password is no longer valid." This means that after the session ends, new login attempts are blocked, while existing connections remain active until closed.

- **Redshift** - the **VALID UNTIL** clause is used on user creation. From the documentation: "The **VALID UNTIL** option sets an absolute time after which the user's password is no longer valid. By default the password has no time limit." This means that after the session ends, new login attempts are blocked, while existing connections remain active until closed.
- **Greenplum** - the **VALID UNTIL** is used on user creation, but it looks like it is not working as expected.
- **SQLServer** - currently not supported.

Note: The following actions will still be performed only after the CM returns online: running the **Inherit Objects** process and dropping the ephemeral user.

The role configuration options can be found here: [roles management](#)

Warning: This feature is **NOT** functional when using MySQL with version < 8.0 and PostgreSQL with version < 11.0.

The Roles Management section impacts also proxy authentication process i.e. what database roles can be granted for successfully authenticated user. To allow the role to be granted Approvals Needed must be set to 0. A role whose configuration is not created at all will not be granted to the user during synchronization ([Synchronize DB Authentication](#)), even if the user is a member of such group in LDAP Server! For example: User authenticates via LDAP and 3 groups are extracted: *myrole*, *myrole2* and *myrole3*. If configured as on the screen above, only the **myrole3** will be granted to the user.

Note: On PostgreSQL, if a role contains any global privileges (such as SUPERUSER, CREATEDB, CREATEROLE, REPLICATION, or BYPASSRLS), these global privileges will also be granted when the role is assigned to the user.

Roles Management (Portal Mode disabled)

Roles Management 										
Filter system roles: 										
Role Name		Notification Alias		Default Time		Maximum Time		Approvals Needed		
admin	▼	alias	▼ 	30	m ▼	16	h ▼	1 		
myrole	▼	notify	▼ 	1	h ▼	1	d ▼	1 		
myrole2	▼	alias	▼ 	1	h ▼	1	d ▼	2 		
myrole3	▼	none	▼ 	1	m ▼	3	d ▼	0 		

When Portal Mode is disabled, the main functionality of this section is limited. However, it is still possible to create a role config without the necessary approvals (Notification alias 'none' type required).

Database Browser

The Database Browser is a versatile tool designed for viewing database structures and modifying permissions. Click [here](#) for more information.

Secrets Manager

The username and password can be stored in the [Secrets Manager](#).

These credentials are cached on both manager and proxy instances. This configuration supports automatic refreshing of these values in case a rotation happens. This is done only when the connection fails due to the cached credentials being out of date and authentication error happening on the data source.

Furthermore, in instances where a health check is configured and fails, such as during password rotation, the system will attempt to retrieve updated credentials from Secrets Manager as well. In the event of a proxy-induced change to the username and password, users are advised to refresh the DataSource page to view the newly uploaded credentials.

Kerberos Authentication

To enable Kerberos authentication for your database connection, check the 'Kerberos' box and provide the username along with the keytab location. If keytab location is empty, the default location is /etc/krb5.keytab.

Configuration

Driver:	PostgreSQLDriver42.3.3	+
JDBC URL:	jdbc:postgresql://172.17.0.1:5432/\${database}	
Username:	datasource	
Keytab Location:	/home/username/heim.keytab	Kerberos ☒
Test query:	SELECT 1	
<button>Test Connection</button>		

Kerberos authentication will be applied only for username specified in DataSource configuration. All other users will authenticate with username and password. This feature is available only for PostgreSQL and SQL Server databases.

Important: Ensure the keytab file, which contains the specified user credentials, is located on both the Central Manager server and any proxy servers associated with the data source.

Configuring Trigger Based Invalidation

In order to support out-of-band data ingestion into the database, a data source can be configured for "Trigger based invalidation". There are several options available on the data source for this configuration:

- **dbTriggerTracking:** set to true if this feature is to be enabled
- **dbTimeMs:** A query to retrieve the current time in MS on the database, to insure timestamps are synchronized
- **dbChanges:** A query to retrieve a timestamp (as a long) and table name (as a string), for the last invalidation time of that table.

The **dbTimeMs** call for example defaults to the value of "SELECT heimdall.get_time_ms()", usable on MySQL without any changes. The **dbChanges** call defaults to {CALL heimdall.changes(@ts)}. This can be a stored procedure. The string "@ts" will be replaced automatically with the last modification (per the DB time) that has been observed, so can be used to pull only the tables that have been modified since the last invalidation observed. The table name returned needs to be fully qualified, as observed with the "printTables" option for the rules, as this is the table name we will be comparing for invalidation purposes.

Driver Overview

MS-SQL-Server

☒ Enable ☒ Auto Update

Commit

Configuration

Version:	9.4.0
Example Url:	jdbc:sqlserver://<server_name>:1433;databaseName=<db_name>
Writer Url:	jdbc:sqlserver://\${listener};databaseName=\${database};multiSubnetFailover=true;transparentNetworkIPResolution=true
Reader Url:	jdbc:sqlserver://\${readers};databaseName=\${database};applicationIntent=Read Only
Website Url:	https://docs.microsoft.com/en-us/sql/connect/jdbc/overview-of-the-jdbc-driver
JDBC Class:	com.microsoft.sqlserver.jdbc.SQLServerDriver
Data Source Class:	com.microsoft.sqlserver.jdbc.SQLServerDataSource
XA Data Source Class:	com.microsoft.sqlserver.jdbc.SQLServerXADataSource
Notes:	Java 7+ required to support SRV variant field typed - newer drivers do not support Java 7, thus the older version provided here
Maven ID:	com.microsoft.sqlserver:mssql-jdbc:9.4.0.jre8
Driver Upload:	Choose Files No file chosen
Current files:	mssql-jdbc-9.4.0.jre8.jar

The drivers tab allows you to manage the jdbc jar file that will be provided to the application or used by the proxy.

Fields

Fields can be found in [drivers options](#)

AWS Cluster Tracking

In order to account for various configurations that may be desired, if load balancing is enabled, and cluster tracking is enabled, the reader and writer URL patterns will be used to generate the actual JDBC URL's to represent the various nodes of the cluster. The reader and write URL's can

contain the following variables for substitution, which should be used based on the information extracted from the AWS configuration:

Variables can be found in [Reader and Writer variables](#)

If the writer URL includes the string ":replication:" AND there is only one node in the cluster, then the value for the writer will be duplicated twice in the reader list. This is to avoid a degenerate case for the MySQL driver where a "replication" url requires two hosts to be defined, or it will fail to connect to any host. By duplicating the host twice, it passes the check, and will work as expected.

Advanced

When a vendor driver is used by the Heimdall driver, we will first attempt to load the driver remotely, i.e. by downloading the driver from the central manager. This provides a single source for the drivers, simplifying the management of the drivers. If this fails, it will attempt to use the local class loader in order to find the driver. In an OSGI Java environment, it may be necessary to package the database driver with the Heimdall driver as a dependency in order for all behaviors to work as expected if the remote load feature does not work.

Proxy vs. JDBC Driver Behavior

When using Heimdall in proxy mode, it is highly recommended that the drivers provided in the Heimdall install package be used. This is as a result of the tight integration work that is done with Heimdall to operate with these drivers, and may be broken between release builds of the vendor driver. In JDBC mode, this requirement can be ignored, and the application recommended version of the drivers should be used.

Rule Overview

Rules										
Magento-rules Enable Create New Rule List Commit										
#	Enabled	Regex	In-Trans	Action	Parameter	Value	Edit	Copy	Delete	
1.	☒	Match all if empty	☐	Cache	ttl	1 h			☐	
					maxLocalS	1000000				
					maxSize	10000000				
2.	☒	(?i)^s*\$select	☐	Reader Eligible					☐	
3.	☒	Match all if empty	☒	Log					☐	
4.	☒	(?i)^s*\$select	☒	Extract Plan					☐	
Add Rule										

A rule list is a set of rules that is processed in order on queries, and dictates the processing done on a particular query and its response. A rule is composed of several components:

- **#:** A current row indicator (for moving rows) and the index of the rule.
- **Enabled:** If the rule is enabled. Non-enabled rules will not be processed.
- **Note Indicator:** When a note is attached to a rule, an icon will be visible, and the note can be viewed by hovering the mouse over the icon
- **Regular Expression:** See below for regular expression behavior
- **In-Trans Flag:** This specifies if a rule should operate also on a queries that are in the middle of a transaction, i.e. it is not in auto-commit mode in an explicit transactional context (between "Start Transaction" and "commit" or "rollback")
- **Action:** What action should be taken on a regex match
- **Parameter:** A parameter that modifies the behavior of the action
- **Value:** The value of the parameter, e.g. to specify the maximum time to live for a cache rule
- **Edit:** An icon to provide an expanded view of the rule, and to allow editing of comments
- **Copy** Allows copying existing rule - it will show as new rule with copied rule options.
- **Delete:** Toggle deletion of a rule on the next commit

To re-order a rule, it can simply be dragged into place in the desired order.

Hint: To better edit a rule, make sure to use the pencil icon to open up the expanded window, which provides more editing options.

Regular Expression Field Behavior

Each rule has a field that can contain either a re2j 1.1 (<https://github.com/google/re2j>) regular expression, or an extended specifier (below). The re2j regular expression language is nearly identical to normal Java regular expressions, except that it can operate in linear time, while Java regular expressions may end up being unbounded in time. As a tradeoff, certain features dependent on backtracking are removed.

Examples of some simple regular expressions:

- "(?i)^select" Case insensitive match all queries that start with "select" at the start of the line
- "(?i)somestring" Case insensitive match queries with the string "sometable" anywhere in the query

In addition to regular expressions, the following extended specifiers can be used instead:

- **literals**:{string1},{string2}..., to match any one string literal in a list.
- **tables**:{table1},{table2}..., to match any one table in the named list.
- **tablesall**:{table1},{table2}..., match only if all the tables in the list are present in the query tables.
- **tablesonly**:{table1},{table2}..., match only if all the tables in the query tables are in the list present (but all do not need to be).
- **tablesexactly**:{table1},{table2}..., match only if every table in the list is present in the query tables, and no other tables are present, i.e. an exact match between table lists.
- **tablescontain**:{substring1},{substring}..., match if any table contains any of the exact strings (no regex processing)
- **users**:{user1},{user2}..., match if the query's users matches one of the users in the list (as regex).
- **ldapgroups**:{group1},{group1}..., It is an intersection of defined in the rule 'ldapgroups' value and extracted ldap groups of authenticated user. When at least one group matches, then the rule is applied (supports regex).
- **catalog**:{catalog1},{catalog2}..., match if the query catalog matches any catalog in the list (as regex)
- **catalogprefix**:{prefix1},{prefix2}..., match if the query catalog starts with any string in the list
- **appname**:{application_name1},{application_name2}..., match if the postgres application name matches (as regex)

- **ports:**{port1},{port2}..., match the port the query was received on, via the vdb. This allows multiple ports to be used on the vdb, and have a different behavior based on which port was used.
- **ips:**{ip1},{ip2}..., match the IP the query was received from, via the vdb. Subnet matching is not currently supported. If complex IP based rules are desired, it is suggested that an additional port be added to the proxy, the ports qualifier be used, and firewall rules can be used to control what IP ranges have access to this new port with the desired configured behaviors.

Note for extended specifiers: The fully qualified catalog.table name is expected, so if using the "magento" database with a table of "users", you would match against "magento.users". Please note the dynamic parameters below for substitutions that are supported here as well. To verify that the table name is as expected, use the "printtable" option, view the query in the expanded analytics tab, or use the debug option to view the fully qualified table name as used internally. Alternatively, use the "expanded query view" in the analytics, and then scroll down to the table names:

To open the expanded query view:

Info	Rules	DB Usage Ratio (%)	Cache Hit (%)	Count	Cache Time ms	DB Time ms	Overall Response Times ms	Result Retrieval Times ms	Average Result Size	Transaction (%)	Query
		7.4	0.0	16k	0	6.5	6.5	6.5	1.0	0.0	SELECT "product_template"."id" as "id", "product_template"."weight" as "weight", "product_template"."create_uid" as "create_uid", "product_template"."active" as "active", "product_template"."website_sequence" as "website_sequence", res_id IN (?)
		4	0.0	32k	0	1.8	1.8	1.8	1.0	0.0	SELECT * FROM ir_translation WHERE lang=? AND type=? AND name=? AND
		3.9	0.0	32k	0	1.7	1.7	1.7	1.0	0.0	SELECT "product_template"."id" as "id" FROM "product_template" WHERE "product_template".id IN (?) AND (((("product_template"."is_published" = true) AND ("product_template"."website_id" IN ())) OR "product_template"."website_id" IS NULL)) AND
		3.7	0.0	26k	0	2.0	2.0	2.0	1.0	0.0	SELECT "website"."id" as "id", "website"."name" as "name", "website"."google_analytics_key" as "google_analytics_key", "website"."salesperson_id" as "salesperson_id", "website"."ada_filters" as "ada_filters", "website"."social_links"

And to view the fully-qualified table names, in the expanded view, scroll down to the tables section before any query plans:


```

    ("product_template"."is_published" = true)
    AND (
        ("product_template"."website_id" in (??))
        OR "product_template"."website_id" IS NULL
    )
)
AND ("product_template"."sale_ok" = true)
)

```

Tables

- odoo.public.product_template
- odoo.public.ir_translation

Execution Plans

Odoo-vdb

2 times, last at 09/29/2019 11:35:53:997

Additionally, you can use the string "\${catalog}" to match the current connections catalog in processing a rule, so as to allow matching on multiple catalogs as appropriate.

Examples of some extended regular expressions:

- "tables:magento.core_store" to match on the table magento.core_store only, but any other table may exist as well
- "tables:\${catalog}.core_store" to match on the table core_store only, but in any catalog, not just the magento catalog
- "tablesexactly:magento.core_store" to match on the table magento.core_store only, and no other tables are in the query

When using the expanded rule view, it is possible to add more than one regex field to a rule, and then specify if AND or OR logic should be used to connect them. This allows for example, one to specify a tables: specifier, then to filter say INSERT, UPDATES or DELETES only against that table.

Rule Processing Behavior

All rules are processed in order of evaluation. This means that a rule that matches earlier in processing can be overridden later in processing by the same rule type.

In order to process a rule, the following steps are taken:

- Check if rule list is enabled

- Iterate through rules
- Check if a rule is enabled
- Check if the in-transaction flag is set when in a transaction
- Check if the rule type should be excluded for processing, due to a previous ignore rule
- Check if the regular expression matches (see below)
- Process the capture parameter, if set, for dynamic parameters
- Check if the rule rate matching limits are allowed
- Process the update, printmatch, printtables, tables, and notify flags (although no action is taken at this point)
- Process the rate limiting parameters (except delay) this is done only once, after caching, async and forwarding are executed)
- Per-action rule processing

Note: In the case of multiple Allow rules in consecutive order, the rules internally will self-organize based on the frequency of hits observed by the driver. The more often a rule has been hit, the further up the list it will be positioned to optimize the regex lookup performance. Any other type of rule will be processed in the order specified, and no Allow rule will optimize around another other rule type.

Special case: The nocache rule behaves as both a "cache with a ttl of 0" and "ignore all other cache rules following", i.e. it prevents all past and future cache rules from impacting the query. Most rules can be overridden, but for safety, the nocache rule can not be overridden once it matches a particular pattern.

Rule Flow Control And Dynamic Parameters

Several actions and properties can be used to control the processing of rules for more advanced applications:

- **Ignore Action:** When an ignore action is specified, the name of an action type can be specified. Any rule matching that action type will be ignored in any further processing of the matching query. This can be used to create exclusions where a particular action shouldn't be taken, but creating a regular expression to account for this would be difficult.
- **Call Action:** A call can be used to call by name a nested rule-list, to allow one rule-list to be called from another rule-list. This can be leveraged to provide a common set of rules for an application, but custom behaviors for particular instances. Another use case is to use a single regular expression match to apply multiple actions to a given set of content, as in the called rule, an empty regex can be used.

- **Stop Parameter:** When specified on a matching rule, no further processing will occur in the current rule-list for the given query. If in a nested rule-list due to a Call action, the calling rule-list will continue to be processed.
- **Capture Parameter:** This parameter triggers dynamic property evaluation. When set, regular expression capture groups are used to edit the parameter values for the query, in order to dynamically adjust behavior. A common use would be to include a comment at the front of a query, and then use the value in the comment part of or a parameter's value, for example, to dynamically set the TTL of a query based on the comment's included value:
Example:

```

Regex:  /\* ttl=(.*) \*/

Action:  Cache, ttl=${1}

Query:  /* ttl=30000 */ SELECT * from Table

Result:  The query would be cached for 30 seconds

Note: For more parameters, you can use "Regex: /\* param1=(.*)
param2=(.*) \*/" with "Action: Cache, param1=${1}, param2=${2}"

```

Note: To use this feature, you have to enable the "honorComments" parameter. More in the [Request Rules](#) section.

Internal Regexes

Similar to the capture parameter based on regular expression capture groups, some built-in strings have special meaning in properties, and will be replaced with the internal values during rule processing:

- **\${catalog}** or **\${database}**: the current catalog/(database in MySQL term) being used.
- **\${vdb}**: The VDB the traffic was received on
- **\${user}**: The JDBC user on a connection
- **\${connid}**: The connection ID of the front-end connection the query was issued on

Note: These variables can be used when using the extended table matching syntax, such as "tables:\${catalog}.tablename" to avoid having to specify the actual catalog to match against.

There is no need to specify the "capture" parameter for these to operate in this context. This works in both request rules and response rules.

Rate Limiting

Often, such as in the case of a "Log" rule action, it is desirable to limit the rate that a particular rule is matched to an absolute number. Each rule has several options that can be used for this:

- **matchlimit**: Enforces a rate of X pattern matches, with up to Y seconds of burst. To specify the burst value, use the parameter "maxburst". Useful to limit the amount of log data generated in a similar manner to using the sample parameter.
- **maxburst**: Specify the number of seconds of bursting that is allowed before clamping the rate for the ratelimit or matchlimit parameters. Defaults to 10.
- **ratelimitname**: Specify a "rate limit" control token name for limiting. This can be a dynamic value for say $\${catalog}$ or $\${user}$ to limit rate by various metadata.
- **concurrency**: Specify the number of parallel queries at a time that can be executed. The lock will be released after the the "execution" is complete, i.e. after the initial data is received, but not necessarily after all the data has been transmitted by the database;
- **concurrencyname**: Name the token for tracking of concurrency.

Another case is that a rule should be evaluated only a certain percentage of the time, say 1 out of 10. The following parameters can be used for this:

- **exclude**: A ratio of queries should NOT match the policy, as expressed as 1/X. Useful to allow a small number of objects that would otherwise be a cache hit to be a cache miss instead. Example: A value of 10 would exclude 1 out of every 10 matches from actually matching the rule.
- **sample**: A ratio of queries should match the policy, as expressed as 1/X. Useful to limit logging to a small percentage of traffic. Example: A value of 10 would mean that 1 out of 10 matches would be treated as a match.

A final case is where queries matching a rule should themselves be slowed down, say to help prevent an outage of all traffic due to a DoS attack. The following parameters can be used in such cases:

- **delay**: When a rule with a delay flag matches, induce a delay in processing. Useful to determine if a rule can have an overall impact on performance, i.e. if a delay doesn't make a noticeable difference, then caching won't either. Also useful in inducing delays to determine if moving the database further away will impact performance significantly. This delay will occur only if the query is requested from the database, i.e. it will not impact a cached query.

- **ratelimit:** Enforces a rate of X queries per second, with up to Y seconds of burst. To specify the burst value, use the parameter 'maxburst'. This will actually SLOW DOWN queries matching this rule to at most this value.

Queries Forwarding

Configuring data source and rules in a proper way allows us to forward slow queries to another data source to avoid database performance drop. If average runtime will be longer than given 'matchthreshold', then query will be forwarded to data source defined by 'serverfilter'.

Data Source configuration:

—

Load Balancing/High Availability

Enable Load Balancing: ☒

Connection Hold Time (ms): 30000

Track Cluster Changes: ☐

Use Response Metrics: ☐

Track Replication Lags: ☐

Lag Window buffer (ms): 10000

Failover Script:

Name: primary

Url: primary_jdbc_queries

Enabled: ☒

Writable: ☐

Weight: 1

Name: slow-queries

Url: jdbc_url_for_slow_queries

Enabled: ☒

Writable: ☐

Weight: 0

Add Server

Rules configuration:

slowQueries
☒ Enable
Commit

Request rules
Response rules

#	Enabled	Regex	In-Trans	Action	Parameter	Value	Edit	Copy	Delete
1.	☒	Match all if empty	☐	Reader Eligible	matchthres serverfilter	1000000 slow-queries			☐

Add Rule

Request Rules

Request rules can be seen in [Request Rules](#)

Response Rules

Rules
Create New Rule List

postgresql-rules
☒ Enable
Commit

Request rules
Response rules

#	Enabled	Column Name Regex	Column Row Regex	Action	Parameter	Value	Edit	Copy	Delete
↑ 1.	☒	columns:security_number_resp_alert	startswith:728	Alert Row	rowCountsAle sendNotificati	5 security_email			☐
↑ 2.	☒	columns:security_number_resp_alert	startswith:728	Alert Row	honeyTokenAlk	true			☐
↑ 3.	☒	columnsStartsWith:sec AND columnsEndsWith:name_resp	^(HR) (AD)	Delete Row					☐
↑ 4.	☒	columns:telephone_number_resp	startswith:78 OR startswith:48 OR startswith:11	Mask Row	replaceWith replaceType replaceFirst replaceAfterSti	X each 2 - 3			☐
↑ 5.	☒	columns:email_resp	.*\uk\$ AND startswith:st	Mask Row	replaceWith replaceType replaceFirstRej	* all [^@]++			☐
↑ 6.	☒	columns:tokenize_test_email	Match all if empty	Tokenize Row	excludedChars	@.?			☐

In the Response Rule tab, we can define actions that will be processed for responses. The rule consists of several elements:

- **#:** A current row indicator (for moving rows) and the index of the rule.
- **Enabled:** If the rule is enabled. Non-enabled rules will not be processed.
- **Notes Indicator:** When a note is attached to a rule, an icon will be visible, and the note can be viewed by hovering the mouse over the icon
- **Column Name Regex:** Specify the column name regular expression to match with. See column name regexes below for more information.
- **Column Row Regex:** Specify the column row regular expression to match with. See column row regex below for more information.
- **Action:** What action should be taken on a regex match.

- **Parameter:** A parameter that modifies the behavior of the action.
- **Value:** The value of the parameter, e.g. to specify the replace string for a masking rule.
- **Edit:** An icon to provide an expanded view of the rule, and to allow editing of comments.
- **Copy:** Allows to copy chosen rule.
- **Delete:** Toggle deletion of a rule on the next commit.

To re-order a rule, it can simply be dragged into place in the desired order.

If there is a row of data where more than one rule tries to perform the same operation (e.g. two or more alerts or masking, tokenization), only the last rule will be applied. It is especially important with choosing the sendNotification, and the source for tokenization rules, as it can save the tokens in the different database than expected. It still allows for masking and alerting on the same field as well as applying the same rule to different columns in the same row.

Response rules are also applied to metadata retrieval.

Hint: To better edit a rule, make sure to use the pencil icon to open up the expanded window, which provides more editing options.

Column Name Regex Behavior

Defines the column name expression to match.

Examples of specifiers that can be used :

- **tables:**{table1},{table2}..., to match all column names from table.
- **columns:**{column1},{column2}..., to match any column name literal in a list.
- **columnsStartsWith:**{column1},{column2}..., to match any column names that starts with any string in the list.
- **columnsEndsWith:**{column1},{column2}..., to match any column names that ends with any string in the list.
- **columnsExist:**{column1},{column2}..., to match all column names that are in the query, if at least one column name will match. (ex. If "select col1, col2 from table" is the query, and columnsExist:col1 is the rule, it will apply logic for both col1, col2)
- **columnsNotExist:**{column1},{column2}..., to match all column names that are in the query, if specified column will NOT occur in the query. (ex. If "select col1, col2 from table" is the query, and columnsNotExist:col3 is the rule, it will apply desire logic for both col1, col2)

Column Row Regex Behavior

Defines the column row expression to match.

Each column row regex field can contain either a re2j 1.1 (<https://github.com/google/re2j>) regular expression, or an extended specifier (below). The re2j regular expression language is nearly identical to normal Java regular expressions, except that it can operate in linear time, while Java regular expressions may end up being unbounded in time. As a tradeoff, certain features dependent on backtracking are removed.

Examples of some addition to regular expressions, the following extended specifiers can be used instead:

- **literal**:{rowData1},{rowData2}..., to match any column row literal in a list.
- **startswith**:{rowData1},{rowData2}..., to match any column row that starts with any of the given string.
- **endswith**:{rowData1},{rowData2}..., to match any column row that ends with any of the given string.
- **notMatch**:{rowData1},{rowData2}..., to exclude any column row that matches any of the given string.

Response Rule Action Details

Parameters of each rule type can be chained, editing is required to add more than one parameter. In a masking and tokenization rules the order of processing is the same as the order of the printed parameter options.

An option to send notifications can be added to each response rule, which will be sent if the rule is triggered. It is recommended to utilize the 'Edit' button to associate the 'sendNotification' rule with an existing alert rule. Sent email will contain details, and its subject will be dynamically set as your vdb name that generated this alert. Before using this feature, client has to previously define Notification entry in the Admin -> Notification tab. More information about Notification system is available [here](#) If multiple sendNotification rules specify different notification recipients, the last applicable rule takes precedence.

Response rules options can be found in [response rules](#)

Tokenizing

Tokenization replaces sensitive values in query results with reversible, format-preserving tokens stored in a dedicated token store (database table). Unlike masking, tokenization enables consistent references to the same sensitive value across queries, sessions, and users —without exposing the original data.

Why use Tokenization?

1. **Format Preservation:** Tokens maintain the data type and general structure of the original data (e.g., a tokenized email will still look like a valid email address). This prevents downstream application validation errors.
2. **Analytics Support:** Unlike simple Masking (which might replace everything with `*****`), Tokenization allows you to perform analytics like `GROUP BY` or `JOIN` on sensitive columns without ever exposing the real data.
3. **Performance Note:** Because tokenization involves reading from and writing to a database table ("token store"), there is a performance overhead compared to standard in-memory masking.

How it works:

When a response rule triggers tokenization, Heimdall follows this logic:

1. **Lookup:** It checks the configured token store database to see if this specific sensitive value has been tokenized before.
2. **Found:** If a token exists, the existing token is returned. This ensures the same sensitive value (e.g., "John Doe") always resolves to the same token (e.g., "Hybd Kfs") across different queries and user sessions.
3. **Not Found:** If the value is new, Heimdall generates a new unique token that mimics the format of the original data and stores this pair in the token table for future use.

TOKENIZE ACTIONS

Tokenize rule action and its parameters can be found in [response rules](#)

To properly initialize the configuration required for utilizing the tokenization feature, users are advised to execute the provided SQL script, "create-token-table.sql", accessible via the DataSource GUI button. The masking and data tokenization feature works fully in proxy mode (In the SQLServer this feature supports only numeric and string values - excluding uniqueidentifier type). In JDBC mode (e.g. oracle) will only work for String values.

For tokenization to work, a data source will need to be specified to store the original value, and the tokenized value, allowing the original value to be retrieved if needed.

Before using the tokenization feature, please contact Heimdall support, to adjust the setup with specific requirements, helping to avoid potential slowdowns in performance.

When configuring masking and tokenization for a same column, only the masking will be performed.

ADDITIONAL EXAMPLES OF USING RESPONSE RULES

Create two rules, one for the "honey token" alert and one for delete row that will delete that honey token, so that the final user only sees the alert but not the "honey token" data. The number of rows and their size would be computed after that operation, so from the perspective of the user (who doesn't have access to the HeimdallData Centrall Management console) they wouldn't know that any "honey token" was queried.

When working with application like DBeaver, we need to load a lot of data at startup, including all the tables, even if we need only a few of them. We can use Delete Row rule, to edit the metadata results from information schema so that all unnecessary tables/partitions are removed from the metadata, greatly reducing amount of data load to DBeaver, and improving boot time for users.

Rules Create New Rule List

dbeaver_example ☒ Enable Commit

Request rules Response rules

#	Enabled	Column Name Regex	Column Row Regex	Action	Parameter	Value	Edit	Copy	Delete
1	☒	AND tables:pg_database columns:datname	notMatch:\${catalog}	Delete Row					
2	☒	tables:gp_settings,gp_stat_activity	Match all if empty	Mask Row	replaceWith replaceType replaceAll	* each true			

For example, you can create a response rule configuration, where the first rule has the 'Delete Row' action using 'Column Name Regex' as 'tables:pg_database' & 'columns:datname' and using 'Column Row Regex' as 'notMatch:\${catalog}' to delete all rows except those containing the database you are connecting, so on initial connection all databases except `${catalog}` will be hidden. The second rule has the 'Mask Row' action using 'Column Name Regex' as 'tables:gp_settings,gp_stat_activity' and replace with * each character of data, so all data in both tables will be masked.

HONEY TOKEN CONFIGURATION AND IMPLEMENTATION GUIDE

A "honey token" is a decoy data record strategically placed in your database. Its purpose is to act as a tripwire; it is data that no legitimate user or application should ever query. When a query accesses this specific honey token, it triggers an alert, signaling a potential security breach, unauthorized data exploration, or a compromised account.

The effective implementation of honey token detection involves two distinct but related rules:

- **An Alert Row Rule:** To detect when the honey token is accessed and notify administrators.
- **A Deletion Row Rule:** (Optional) To remove the honey token from the query's result set, preventing the unauthorized user from knowing they have been detected.

Implementation Steps

Prerequisite: You must have already inserted the honey token data into your database. For example, a fake user record or a fictitious high-value transaction.

Step 1: Create the Honey Token Alert Rule

This rule will identify the query for the honey token and trigger a notification.

1. Create a Response Rule entry.
2. **Column Name Regex:** Specify the table and column where the honey token resides.
Example: `tables:sensitive_records, columns:ssn`
3. **Column Row Regex:** Specify the exact value of your honey token. The literal specifier is ideal for this. Example: `literal:123-45-6789`
4. **Action:** Select `Alert Row`.
5. **Parameter & Value:** Select `honeyTokenAlert`. To receive an email notification, add the `sendNotification` parameter and select your pre-configured notification channel as the value.

Step 2 (Optional): Create the Honey Token Deletion Rule

This rule will remove the honey token row from the result set returned to the user.

1. Create another new Response Rule entry by clicking 'duplicate' on the previously created alert rule.
2. **Action:** Select `Delete Row`.

With this configuration, if a user runs `SELECT * FROM sensitive_records;` (which includes our ssn 123-45-6789), the system will:

- Match Rule 1, triggering a `honeyTokenAlert` and sending an email to a notification group.
- Match Rule 2, deleting the row from the results. Also, adjusting the number of rows returned to the user, and the data size, so it's basically undetectable.
- Sends email notification to a configured group of recipients.
- Return a result set without a honey token to the user (if configured with the `deleteRow` action).

The final effect is a silent alarm. The administrators are notified of suspicious activity.

Example configuration:

The screenshot shows a web interface for configuring rules. At the top, there's a 'Rules' header with a 'Create New Rule List' button. Below it, a dropdown menu shows 'pg-proxy-poc-response' and an 'Enable' checkbox. A 'Commit' button is on the right. The main area is divided into 'Request rules' and 'Response rules' tabs. Under 'Response rules', there's a table with columns: #, Enabled, Column Name Regex, Column Row Regex, Action, Parameter, Value, Edit, Copy, and Delete. Two rules are listed:

#	Enabled	Column Name Regex	Column Row Regex	Action	Parameter	Value	Edit	Copy	Delete
1	☒	tables:sensitive_records AND columns:ssn	literal:123-45-6789	Alert Row	honeyTokenAlert sendNotification	true test			☐
2	☒	tables:sensitive_records AND columns:ssn	literal:123-45-6789	Delete Row					☐

An 'Add Rule' button is at the bottom right of the table.

Wizard Overview

The Wizard is provided to provide a guided walk-through to configure a VDB, data source, and desired pre-configured rule-sets that may be desired.

Manual vs. AWS Detect

If the Heimdall management server detects that it is in AWS, it will offer the AWS Detect option in the first section of the wizard. For this to work, either an IAM secret and key need to be provided OR an appropriate IAM role needs to be associated with the ec2 instance the management server is running on. Please see the [aws environment](#) section for more details. Otherwise, simply select Manual Configuration to continue to the next section. Using the AWS detect option does NOT change what options are available, it simply allows defaults to be pre-populated to simplify the process.

Configuration Wizard

In order to simplify the configuration of the Heimdall environment, this wizard will guide you through the necessary steps.

No changes to your configuration will be made until the "submit" button is selected at the end.

On completion, customized instructions for your configuration will be provided to assist you in configuring your application to use Heimdall.

Please start by selecting AWS Detect (if desired), to detect AWS RDS and ElastiCache settings, or if you wish to configure these resources manually.

AWS Detect

Manual Configuration

If the AWS Detect option fails, one of the more common reasons is due to a locked down VPC that doesn't provide connectivity to the API endpoints. You can test the various endpoints via an SSH session to the Heimdall manager, then using the following (adjust the region name as appropriate):

```
wget https://ec2.us-east-1.amazonaws.com/
wget https://rds.us-east-1.amazonaws.com/
```

Please see the additional comment on api endpoints in the AWS IAM section.

Data Source

Here, the database settings are configured. First, name the configuration (this should be a unique name that correlates to the application or database in question). Next, select the

database type and driver, which sets a variety of defaults based on the database capability.

Note: The username and password can be stored in the [Secrets Manager](#). The rest of the database values should be set appropriately. In the case of Postgres, since a database instance will either implicitly or explicitly need to be selected, use `${database}` as the default, unless it is known that only one database instance will be used. In the case of a proxy, Heimdall will observe what the desired database is based on the connect negotiation, and will fill in this value automatically as needed.

The Heimdall management server detects which cloud environment it is hosted in and provides the correct database names for the detected environment. If you don't see the type of database you are actually using, you can disable "cloudDetection" in the admin tab and then all database types will be shown in the list.

You can also choose a pre-made template from the library that will complete the configuration fields in the wizard, and you will only have to customize some of them to your needs. Remember! Not all things can be set in the wizard. Some of them you will have to set after finishing the wizard.

The screenshot shows a 'Data Source' configuration form. It contains the following fields and values:

Field	Value
Configuration Name	db-cluster-name
Template	Default Postgres
Database Type	Postgres
Database Driver	postgres-driver
Database Hostname	127.0.0.1
Database Port	5432
Database Username	postgres
Database Password	••••
Default database/catalog	\${database}

At the bottom of the form are two green buttons: 'Back' and 'Next'. To the right of the 'Database Username' and 'Database Password' fields, there is a 'Secret' checkbox and an eye icon.

Load Balancing/High Availability

Here the various LB and High Availability options can be set. In the case there is a read-only server, enter it here. If there is more than one, they can be added in the data source tab once created. Please see the [sources](#) section for more details on each option.

Configuration Wizard

Load Balance/High Availability

Enable Load Balancing:

☒

Track Cluster Changes:

☐

DB Read-Only Hostname:

Optional

Track Replication Lags:

☐

Lag Window Buffer (ms):

10000

Back

Next

Cache Configuration

Here, the various options for each cache type (if caching is desired) can be configured, e.g.:

Caching

Cache Type:

Hazelcast

AWS Access Key:

Optional

AWS Secret Key:

Optional

AWS Tag Key:

Optional

AWS Tag Value:

Optional

Back

Next

Configuration Wizard

Caching

Cache Type:

Redis/Elasticache for Redis/KeyDB

[Signup with Redis Cloud](#)

Cache Hostname:

Required

Cache Username:

Optional

Cache Password:

Optional

Cache Port:

Required

Enable TLS:

☐

Secret

☐

Back

Next

Note: The username and password can be stored in the [Secrets Manager](#).

Proxy Configuration

When the data source supports a proxy (i.e. not Oracle or other JDBC only data sources for example), then this next step allows for the configuration of the proxy settings. If Active Directory support is used or more than one proxy port is desired, please configure this in the VDB->Proxy settings after the wizard is complete.

Proxying

When using a supported database, should proxy-mode (via TCP) be enabled?
Important: Do NOT use the word "localhost" when configuring most applications to access the proxy, as this will probably trigger Unix sockets to be used, bypassing the proxy.

Enable Proxy: ☒

The TCP binding address type.

Address Binding Type: Any

The TCP port used by the proxy--insure this does not conflict with other server applications.

Proxy Port: 5432

Start proxy server via the Heimdall Central Manager? This is useful for single server installs.

Management Server Proxy: ☒

Authenticate users locally to Heimdall? MySQL does not support passthrough authentication, but disabling this for Postgres and SQL server will allow the server to authenticate users directly.

Proxy-Auth Enabled: ☐

Back

Next

Properties and Rules

This next section allows a variety of pre-defined rules and properties to be set. Any setting can be adjusted or disabled after configuration, so it is generally recommended that they be left enabled unless the feature is known to cause problems or is not desired.

Configuration Wizard

Properties and Rules

Enable global VDB logging? If not, logging can be enabled via rules instead.

SQL Logging: ☒

Transmit all driver & proxy console logs to the central server for aggregated logging?

Aggregate Console Logs: ☒

Store logs in flat files, not just the analytics database?

Write Logs To File: ☒

Enable connection pooling for the Data Source? This can improve performance but may break some applications.

Connection Pooling: ☒

Configure basic cache rules for the vdb? This will attempt to cache all traffic when not in autocommit mode.

Default Cache Rules: ☒

Configure Read-Write split for queries starting with "SELECT"? This will route matching queries to the configured reader:

Read/Write Split: ☒

Expand the Read-Write split rule to include queries starting with 'describe', 'explain', and 'show'?

Multiple Read Only Operations: ☐

Back

Review

Review Summary

Next, the overall configuration can be reviewed for correctness:

Configuration Wizard

Summary

Data Source

Configuration Name:	db-cluster-name	
Template:	Default Postgres	▼
Database Type:	PostgreSQL/Aurora (Postgres)/EnterpriseDB	▼
Database Driver:	postgres-driver	▼
Read/Write JDBC URL:	jdbc:postgresql://127.0.0.1:5432/\${database}	
Read-Only JDBC URL:	Optional	
Database Username:	postgres	Secret ☐
Database Password:	*****	
Default database/catalog:	Recommended	

Load Balance/High Availability

Track Replication Lags: ☐

Lag Window Buffer (ms): 10000

Caching

Cache Type: Local Only (no distributed invalidation) ▼

Proxying

Enable Proxy: ☒

Address Binding Type: Any ▼

Proxy Port: 5432

Local Proxy: ☒

Proxy-Auth Enabled: ☐

Properties and Rules

SQL Logging: ☒

Aggregate Console Logs: ☒

Write Logs To File: ☒

Connection Pooling: ☒

Default Templates Rules: ☒

Read/Write Split: ☒

Multiple Read Only Operations: ☐

Back

Next

Submit

Finally, once the configuration has been reviewed, a final section with general guidance will be provided based on the options selected. Select "submit" to provide the configuration to the server.

- Based on the selected options, please follow these directions for best results upon submitting the configuration:
- Please test the data source via the "test" button on the sources tab.
 - A default cache-all rule will be added to the rulelist db-cluster-name-rules and committed.
 - Proxy authentication passthrough has been configured, any configured user on the database should be able to use the proxy.
 - You have selected to run a proxy on your management server for use by other servers.
 - Do not use the hostname of "localhost" when pointing to a Heimdall Proxy as this will often bypass the proxy.
 - You can configure your application to point to the Heimdall instance's IP on port 5432.
 - Please verify that the AWS security group or other firewall allows connections on port 5432.
 - Console logs from the Heimdall Driver/Proxy will be shipped to the central manager for reporting.
 - If you have any difficulty configuring your application, please refer to [Application Installation](#).

Back

Submit

Save to PDF

Modules Overview

Starting with the first full release in July, Heimdall includes a module system for loading features at runtime. As of this release, the modules include:

- **Cache modules:** Hazelcast and Redis
- **Feature modules:** Async execute, Forwarding+read/write split, Load Balancing/HA, SQL Password Extraction and Proxy-side triggers

Note: More modules are likely to be broken out, so consider the modules included in each archive to be the definitive list.

Installing

In order to install a module that is not currently installed, locate the module desired in the install zip file (under /heimdall/static/modules) and through the admin->modules tab, load the module. That is it. In the case of a cache module, the ability to select the cache type will be made available in the VDB tab.

In the case of the default password extraction module, it will leverage a JDBC connection through the VDB itself, allowing all caching, logging, to apply to the password query. To configure the password query, set in the vdb properties a property named "passProvPasswordQuery", with a value such as "SELECT password FROM users WHERE username=?". The ? will be replaced with the username provided, and the database should return a single value of the password that should be expected.

Updating

Modules are updated automatically along with the other Heimdall components. A module can be updated only if there is a new version with the same major version. For example, if the module is 1.0-18.7.23.1, the major version is "1.0" and the minor version is 18.7.23.1. In the case of a grid cache module, the major version will align with the library version the module is

compiled against, i.e. for Hazelcast, we are currently shipping compiled against 3.6.8, but we can provide modules compiled against other versions of Hazelcast as well.

Custom Modules

In particular for the Password extraction module, Heimdall can make available the source code so that custom modules can be compiled and used for your environment. In the event of a bug-fix, we may provide a module instead of a complete release package to minimize the impact on deployment. Modules can be updated at run-time, so we can often resolve issues without requiring a reboot of the system using the modules system.

Module specifics

Please see the rules specific for each action the module provides for more information, or the page dedicated to the module as appropriate

Proxy Modules

- heimdallmysql: enables the MySQL proxy code
- heimdallpostgres: enables the Postgres/Greenplum/Redshift/AlloyDB proxy
- heimdallsqlserver: enabled the SQL Server proxy

Cache Modules

- heimdallhazelcast: Hazelcast support
- heimdallredis: Activates Redis cache support

Feature Modules

- heimdallpw: provides a means to extract proxy passwords from the default data source.
Example configuration: VDB Property: "passProvPasswordQuery" value: "SELECT password FROM users WHERE username='?'"
- heimdallasync: Async Execution support (see documentation page for more detail)
- heimdallauth: Stub module to provide GUI level authentication support, no directly accessible feature, but can be used as the basis for custom authentication modules
- heimdallextractplan: allows query execution plans to be retrieved after executing queries
- heimdallforward: forwards queries from one data source to another based on rules

- heimdallloadbalancer: activates the LB and HA functionality of Heimdall (see LB documentation page for more detail)
- heimdalltrigger: allows triggering of actions based on matching rules

Management Server Overview

The Heimdall server operates using a system call Spring Boot, which provides the basic infrastructure for managing connectivity, api calls, etc. As such, settings that apply to Spring Boot are used to configure aspects of behavior such as the TCP port used as well as TLS configuration

Changing TCP Port

To change the TCP port of the server, in the install directory, create a file named "application.properties" and in it, add the following line, adjusting for the desired tcp port

```
server.port=8087 # Server HTTP port.
```

When connecting to an alternate port, the driver configuration will also need to be adjusted to account for this.

When configuring a proxy directly, append : to the hostname in the JDBC URL.

Heimdall.conf

The file is a configuration file and can be used to populate environmental variables or cloud environments. It should be generated at directory: `/etc/heimdall.conf`

It can contain different options which should be echoed out if necessary with syntax: "\$
{optionName}=\\${optionValue}":

Heimdall Environmental Variables

Heimdall configurable environmental variables can be found in: [Heimdall Options](#)

Example script for server:

```
#!/bin/bash
(
echo "hdRole=server"
echo "hdHost=localhost"
echo "hdPort=8443"
echo "hdUser=admin"
echo "hdPassword=heimdall"
echo "cloudDetection=true"
echo "cloudDetection=aws"
echo "secure=true"
) > /etc/heimdall.conf
```

Example script for proxy:

(hdRole=proxy) Options	Description
hdUser	Proxy login username. To successful auth, it must be consistent with the Access Key of the vdb with the defined vdbName.
hdPassword	Proxy login password. To successful auth, it must be consistent with the Secret Key of the vdb with the defined vdbName.
vdbName	Exact name of the vdb that we want to auth to.

```
#!/bin/bash
(
echo "hdHost=localhost"
echo "hdPort=8443"
echo "hdUser=correct_access_key"
echo "hdPassword=correct_secret_key"
echo "vdbName=the_vdb"
```

```
echo "secure=true"
) > /etc/heimdall.conf
```

VDB Name

One optional parameter that can be used in the configuration file is `vdbName`, which if present, will specify which vdb should be run on the host. This can also be a comma separated list of multiple vdb names, OR if left empty, it will connect to the management server, and attempt to start a proxy instance for all vdb's on the manager.

Secrets Environmental Variables

The Proxy can authenticate to the Central Manager via secrets. Below environmental variables can be set instead of `hdUser` and `hdPassword` when VDB is configured with Access/Secret Keys allowing the proxy to retrieve these credentials from selected Secrets Manager on startup, before the full configuration is retrieved from Heimdall Manager. An optional way to set this value is to enter an "hdUser" field starting with "secret:" and the name of the secret. This will only override the secret name if it is otherwise blank. Most of those variables will be automatically set via using secrets checkbox in [the Config Management section of Admin tab](#).

Secrets Option Name	Description
<code>useSecretForVdbCredentials</code>	If the VDB will use secret for Access/Secret keys. Available options: true, false.
<code>vdbCredentialsSecretName</code>	A VDB access and secret keys will be fetched from the Secrets Manager using this value.
<code>secretsManagerConfigName</code>	Name of the Secrets Manager Config. If you are setting it manually, it can be random name.
<code>secretsManagerType</code>	Type of the Secrets Manager Config. Available options: AWS, CYBERARK_CONJUR, HASHICORP_VAULT.
<code>secretsManagerConfigSecretName</code>	Name used for retrieving the whole Secrets Manager Config from the AWS Secrets Manager.
<code>secretsManagerCredentialsSecretName</code>	Name used for retrieving the Secrets Manager Config credentials from the AWS Secrets Manager.

AWS Secrets Manager environmental variables

Environmental variables for [AWS Secrets Manager](#).

AWS Secrets Manager Option Name	Description
<code>AWS_SECRETS_ENDPOINT</code>	(optional) Alternate endpoint for Secrets Manager service. Normally this should be detected based on the Region set, use this for non-standard service URL. When endpoint is configured to a different region, make sure the Region is also set correctly.
<code>AWS_SECRETS_REGION</code>	(optional) Override the region which should be used (ex. us-east-1). This should be set when it can't be automatically identified from the environment or in cross-region setups.
<code>AWS_SECRETS_ACCESS_KEY</code>	(optional) Override the access key used to authorize requests to the Secrets Manager. Together with Secret Key this can allow for cross-account communication with Secrets Manager.
<code>AWS_SECRETS_SECRET_KEY</code>	(optional) Override the secret key used to authorize requests to the Secrets Manager. Together with Access Key this can allow for cross-account communication with Secrets Manager.
<code>AWS_SECRETS_USE_STS_ROLE</code>	(optional) If checked, use STS to assume a role with the specified ARN

AWS Secrets Manager Option Name	Description
<code>AWS_SECRETS_STS_ROLE_ARN</code>	(required, when using STS Role) ARN of the role to assume when using STS.
<code>AWS_SECRETS_STS_EXTERNAL_ID</code>	(optional) External ID to uniquely associate with the role.
<code>AWS_SECRETS_STS_SESSION_NAME</code>	(optional) Name to use for the STS session. Defaults to heimdall-STS-session string.

Hashicorp Vault environmental variables

Environmental variables for [HashiCorp Vault Secrets Manager](#).

HashiCorp Vault Option Name	Description
<code>VAULT_ADDR</code>	The URL of the Hashicorp Vault instance.
<code>VAULT_SKIP_VERIFY</code>	(optional) Allow HTTPS communication with Vault server without verifying certificate (true or false). By default it is disabled (false), it is not recommended to use this in production.
<code>VAULT_AUTH_METHOD</code>	Authentication method that should be used by Heimdall to connect to Hashicorp Vault instance. Available options: TOKEN, USERPASS and APP_ROLE.

Environmental variables for [HashiCorp Vault Token authentication method](#).

HashiCorp Vault Token authentication method	Description
<code>VAULT_TOKEN</code>	Token used for authentication.

Environmental variables for [HashiCorp Vault Username & Password authentication method](#).

HashiCorp Vault Username & Password authentication method	Description
<code>VAULT_AUTH_PATH</code>	(Optional) Alternate authentication method mount path to be used. By default set to <i>userpass</i> .
<code>VAULT_USERNAME</code>	Username used for authentication.
<code>VAULT_PASSWORD</code>	Password used for authentication.

Environmental variables for [HashiCorp Vault AppRole authentication method](#).

HashiCorp Vault AppRole authentication method	Description
<code>VAULT_AUTH_PATH</code>	(Optional) Alternate authentication method mount path to be used. By default set to <i>aprole</i> .
<code>VAULT_ROLE_ID</code>	The semi-secret identifier for the role that will authenticate to Vault. Think of this as the username portion of an authentication pair.
<code>VAULT_SECRET_ID</code>	The secret identifier for the role that will authenticate to Vault. Think of this as the password portion of an authentication pair.

CyberArk Conjur environmental variables

Environmental variables for [CyberArk Conjur Secrets Manager](#).

CyberArk Conjur Option Name	Description
<code>CONJUR_APPLIANCE_URL</code>	The URL of the Conjur instance you are connecting to. When connecting to Conjur Enterprise configured for high availability, this should be the URL of the master load balancer (if performing read and write operations) or the URL of a follower load balancer (if performing read-only operations).
<code>CONJUR_ACCOUNT</code>	Conjur account that you are connecting to. This value is set during Conjur deployment.

CyberArk Conjur Option Name	Description
CONJUR_AUTHN_LOGIN	User/host identity.
CONJUR_AUTHN_API_KEY	User/host API key (or password).
CONJUR_AUTHN_URL	(optional) Alternate authentication endpoint. By default, the client uses the standard <applianceUrl>/authn for generic username and API key login flow.

For example, if you want to use CyberArk Conjur as a Secrets Manager manually.

```
#!/bin/bash
(
...
the above main options without hdUser & hdPassword...
...
echo "useSecretForVdbCredentials=true"
echo "vdbCredentialsSecretName=Test/secret"
echo "secretsManagerConfigName=random_name"
echo "secretsManagerType=CYBERARK_CONJUR"
echo "CONJUR_APPLIANCE_URL=<your-appliance-url>"
echo "CONJUR_ACCOUNT=<your-account>"
echo "CONJUR_AUTHN_LOGIN=<your-authn-login>"
echo "CONJUR_AUTHN_API_KEY=<your-api-key>"
) > /etc/heimdall.conf
```

Once initialized, this configuration can be adjusted manually in the /etc/heimdall.conf if necessary. These settings will effectively allow auto-scaling groups of proxies to be configured.

Please note: If building an AMI for auto-scaling, which may be used by multiple scaling groups with different configurations, it is suggested that after doing initial testing, the heimdall.conf be deleted so that user-data will be re-read to build the Heimdall configuration. This will leverage the user-data on each new initialization to build the configuration at startup.

Options information:

Options section provide tables which contains configurable items from each section. It allows quick search for option to correctly configure it.

Tables have different formats to streamline only valuable information, so if requirements section is missing for example, assume there is no requirements. Have in mind that description may contain crucial information about correct format or if some extra requirements may apply.

The sections are:

- **Key** - The key is field on GUI. There may be rows where key is missing - usually it's the things that are generated - like filename which is usually saved as "file" inside the configuration files.
- **Section** - Used to show to which subsection item belongs to.
- **Requirements** - Shows requirements to see given option, if requirements are not fulfilled the field won't be set even if it used. If the requirement is just the name of the key: Enable Kerberos Authentication, it means the given key must be true. Otherwise, it's specified exactly in case of non-boolean values or if boolean value should be false.
- **Conf File field** - Name of the key inside configuration file. The filename is always written above the table.
- **Description** - Description of what given key does. There might be extra information about possible values or some restrictions.
- **Possible values** - Value type that given key is using. There might be some restrictions like number should be non-negative.
- **Default** - Default value of the given key if it's not set. Often configurable items don't have any default values which is shown as -.

General options:

This is list of general options for **Data Sources**.

More detailed descriptions and additional information can be found in [Data source overview](#)

These options are inside given data source configuration file. For example: postgresTest-source_1.conf. The filename depends on the name of the data source → nameOfDataSource.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Requirements	Conf File field	Description	Possible values	Default
Enabled	-	enabled	Enables or Disables the data source.	Boolean	True
Driver	-	driverConfig.name	The field allows either a new or existing JDBC driver to be specified.	String	-
JDBC URL	-	url	The url configuration will normally be build through the configuration wizard, or can be copied from an existing JDBC application.	String	(Generated based on driver)
Username	-	properties.user	It's used in various situations more here : User to use for health monitoring, and if no other authentication credentials are provided on a connection, as the default user to connect to the database	String	-
Password	-	properties.password	It's used in various situations more here : health checking the data source, To performing lag detection, to write to and read from the Heimdall schema, when doing cluster detection, when no user or password is specified in JDBC mode, In proxy mode, if proxy authentication is not provided.	String	-
Test query	-	properties.testQuery	This option is used in a variety of situations, including health checks, and verifying that a connection is still viable after an exception. It is also used in a variety of conditions when pooling is configured, in order to validate the connection while idle, etc.	String	-

Common connection properties:

This is list of common properties.

They are inside given data source configuration file. For example: postgresTest-source_1.conf
These properties are saved in this format there:

```
"properties": {  
  "ApplicationName": "testApp",  
  "testQuery": "SELECT 1",  
  "defaultCatalog": "test",  
  "url": "jdbc:postgresql://localhost:5432/${database}"  
},
```

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Description	Possible values	Default
initSQLAsync	When using an Async rule, a SQL statement to execute on initializing the connection. Please see the Async Theory section for more information.	String	-
authJdbcUrl	Alternate JDBC URI to use when pulling authentication information for SQL Auth	Boolean	-
dbChanges	Query to use to pull database changes	String	-
dbTimeMs	Query to retrieve the current time per the DB in milliseconds since Epoch	String	-
dbTriggerTracking	If table based tracking should be used. Please see the Cache Theory section for more information.	Boolean	false
hdConnectRetryCount	For Heimdall connections, how many retries are to be performed before flagging a failure	Integer	3
hdConnectRetryDelay	For Heimdall connection retries, what should the delay be between retries (in ms)—this value will be doubled on every retry	Integer	0
hdConnectTimeout	For Heimdall connections, the connection timeout (in ms, but some drivers will only honor at per-second increments)	Integer	1000
hdPasswordRotate	Enable password rotation logic to roll DB passwords during rotation windows	Boolean	false
hdRewriteInserts	When using an Async rule, rewrite multi-row inserts into a single command	Boolean	-
defaultCatalog	Used when a database/catalog is not provided, i.e. for health-checks when \${catalog} or \${database} is specified in a URL	String	-
defaultSchema	Used when a schema is not provided, i.e. for health-checks when \${schema} is specified in a URL	String	-
defaultHost	Used when a hostname is not provided, i.e. for health-checks when \${host} is specified in a URL	String	-
hdUseServerInfo	Should use heimdall.server_info table (provides an alternative control point to manage values for load balancing)	Boolean	false

DB specific

Key	Database	Description	Possible values	Default
hdUsePgNotify	PostgreSQL	Instead of using a cache's pub/sub interface, when using Postgres, use the postgres notify interface for cache invalidation and read/write split last write time notifications. Please note, this still requires a cache to be enabled, although it can be the "local only" cache option.	String	-

Key	Database	Description	Possible values	Default
hdNotifyDBs	PostgreSQL	When using hdUsePgNotify, by default only the default catalog is listened to. This option allows one or more catalogs to be listened to, for trigger based invalidation. A value of * means "all catalogs", with comma separated values starting with "-" removing from the complete list. Example: *-template0;-template1 would be a generally good starting point.	Boolean	(see desc.)
azureDbHost	Azure Postgres or Azure MySQL	When using the Azure Postgres or Azure MySQL managed databases, there is a requirement that the username used is of the format "user@hostname". Heimdall provides an option in the data source options of "azureDbHost", which if set to true, will automatically map the username to this format based on which server is being connected to. Ensure that when using this option, the fully qualified domain name is set in the JDBC URL.	Boolean	false

DB2 connection properties

This is list of properties that are compatible with db2.

They are inside given data source configuration file. For example: db2Test-source_1.conf These properties are saved in this format there:

```
"properties": {  
  "ApplicationName": "testApp",  
  "testQuery": "SELECT 1",  
  "defaultCatalog": "test",  
  "url": "jdbc:db2://localhost:50000/${database}"  
},
```

 **Note:** You can use SHIFT with mouse scroll to scroll horizontally!

Key	Description	Possible values	Default
accessToken	An IAM access token for token-based authentication to Db2. The data type of this property is String.	String (up to 8000 bytes)	-
accountingInterval	Specifies when Db2 should cut accounting records for the connection. The data type of this property is String. "COMMIT" or any other value (meaning end of connection)	String	END OF CONNECTION
alternateGroupDatabaseName	Define an alternate group of databases for client affinity automatic failover. The data type of this property is String.	String	-
alternateGroupPortNumber	Define an alternate group of ports for client affinity automatic failover. The data type of this property is String.	String	-
alternateGroupServerName	Define an alternate group of hosts for client affinity automatic failover. The data type of this property is String.	String	-
affinityFailbackInterval	Specifies the length of the interval, in seconds, that the IBM Data Server Driver for JDBC and SQLJ waits between attempts to fail back an existing connection to the primary server. A value that is less than or equal to 0 means that the connection does not fail back. The default is DB2BaseDataSource.NOT_SET (0).	Integer	0
allowNextOnExhaustedResultSet	Specifies how the IBM Data Server Driver for JDBC and SQLJ handles a ResultSet.next() call for a forward-only cursor that is positioned after the last row of the ResultSet. The data type of this property is int.	Integer	2
allowNullResultSetForExecuteQuery	Specifies whether the IBM Data Server Driver for JDBC and SQLJ returns null when Statement.executeQuery, PreparedStatement.executeQuery, or CallableStatement.executeQuery is used to execute a CALL statement for a stored procedure that does not return any result sets. 0 = NO_SET, 1 = YES, 2 = NO.	Integer (0, 1, 2)	-
apiKey	An API key for IBM Cloud that the driver can exchange for an IAM access token. The data type of this property is String.	String (up to 8000 bytes)	-
atomicMultiRowInsert	Specifies whether batch operations that use PreparedStatement methods to modify a table are atomic or non-atomic. The data type of this property is int. 1 = atomic batch, 2 = non atomic, 0 same as 2.	Integer (0, 1, 2)	0
blockingReadConnectionTimeout	The amount of time in seconds before a connection socket read times out. This property applies only to IBM Data Server Driver for JDBC and SQLJ type 4 connectivity, and affects all requests that are sent to the data source after a connection is successfully established. The default is 0. A value of 0 means that there is no timeout.	Integer	0 (no timeout)

Key	Description	Possible values	Default
charOutputSize	Specifies the maximum number of bytes to use for INOUT or OUT stored procedure parameters that are registered as Types.CHAR charOutputSize applies only to IBM Data Server Driver for JDBC and SQLJ type 2 connectivity to DB2 for z/OS database servers. The data type of this property is short.	Short	32767
clientAccountingInformation	Specifies accounting information for the current client for the connection. The data type of this property is String.	String (max 255 bytes)	-
clientApplcompat	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is String.	String	-
clientApplicationInformation	Specifies the application or transaction name of the end user's application. You can use this property to provide the identity of the client end user for accounting and monitoring purposes. The data type of this property is String.	String (up to 32b (OS), up to 255b)	-
clientDebugInfo	Specifies a value for the CLIENT DEBUGINFO connection attribute, to notify the data server that stored procedures and user-defined functions that are using the connection are running in debug mode. CLIENT DEBUGINFO is used by the DB2 Unified Debugger. The data type of this property is String. The maximum length is 254 bytes.	String (up to 254 bytes)	-
clientProgramId	Specifies a client program identifier that can be used to identify the end user program. The data type of this property is String.	String (80 bytes)	-
clientProgramName	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is String.	String (max 20 bytes)	db2jcc_application
clientRerouteAlternateServerName	Specifies one or more server names for client reroute. The data type of this property is String.	String (e.g. "host1,host2")	-
clientRerouteAlternatePortNumber	Specifies one or more port numbers for client reroute. The data type of this property is String.	String	-
clientRerouteServerListJNDIName	Identifies a JNDI reference to a DB2ClientRerouteServerList instance in a JNDI repository of reroute server information. clientRerouteServerListJNDIName applies only to IBM Data Server Driver for JDBC and SQLJ type 4 connectivity, and to connections that are established through the DataSource interface.	String	-
clientRerouteServerListJNDIContext	Specifies the JNDI context that is used for binding and lookup of the DB2ClientRerouteServerList instance. clientRerouteServerListJNDIContext applies only to IBM Data Server Driver for JDBC and SQLJ type 4 connectivity, and to connections that are established through the DataSource interface.	javax.naming.Context	-
clientUser	IBM Data Server Driver for JDBC and SQLJ type 2 connectivity on Db2 for z/OS only. The data type of this property is String.	String (max 128 bytes)	-
clientWorkstation	IBM Data Server Driver for JDBC and SQLJ type 2 connectivity on Db2 for z/OS only. The data type of this property is String.	String (max 255 bytes)	-
commandTimeout	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is int.	Integer	0 (no timeout)
connectionCloseWithInFlightTransaction	Specifies whether the IBM Data Server Driver for JDBC and SQLJ throws an SQLException or rolls back a transaction without throwing an SQLException when a connection is closed in the middle of the transaction. 0 = NOT_SET, 1 = CONNECTION_CLOSE_WITH_EXCEPTION, 2 = CONNECTION_CLOSE_WITH_ROLLBACK, 3 = CONNECTION_CLOSE_WITH_COMMIT	Integer (0, 1, 2, 3)	-
concurrentAccessResolution	Controls the "currently committed" semantics for read transactions. The data type of this property is int. USE_CURRENTLY_COMMITTED (1), WAIT_FOR_OUTCOME (2), NOT_SET (0)	Integer (0, 1, 2)	NOT_SET
connectNode	Specifies the target database partition server that an application connects to. The data type of this property is int.	Integer (0 - 999)	0
connectionTimeout	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is int.	Integer	-
currentDegree	Sets the CURRENT DEGREE (parallelism degree) special register for dynamic SQL. The data type of this property is String.	String	-
currentExplainMode			-

Key	Description	Possible values	Default
	Sets the CURRENT EXPLAIN MODE special register. The data type of this property is String.	String (EXPLAIN, NO, YES)	
currentExplainSnapshot	Sets the CURRENT EXPLAIN SNAPSHOT special register. The data type of this property is String.	String(NO, YES)	-
currentFunctionPath	Specifies an SQL path (comma-separated list of schemas) used to resolve unqualified data types and function names in SQL statements. The data type of this property is String.	String	-
currentLocaleCType	Specifies the LC_CTYPE locale that is used to execute SQL statements that use a built-in function that references a locale. The data type of this property is String.	String	-
currentLockTimeout	Specifies whether DB2 Database for Linux, UNIX, and Windows servers wait for a lock when the lock cannot be obtained immediately. The data type of this property is int.	Integer (-1 and 32767)	-1 (Do no wait)
currentMaintainedTableTypesForOptimization	Sets the CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION. The data type of this property is String.	String (ALL, SYSTEM, USER, NONE)	-
currentPackagePath	List of package collections for DB2 packages. The data type of this property is String.	String	-
currentPackageSet	Specifies the package collection (library) on the server to use for DB2 packages. The data type of this property is String. For type 4 connections, default is NULLID	String	NULLID / None
currentQueryOptimization	Specifies the query optimization class for dynamic SQL, by setting the CURRENT QUERY OPTIMIZATION special register. The data type of this property is int.	Integer	5
currentRefreshAge	Specifies a timestamp duration value that is the maximum duration since a REFRESH TABLE statement was processed on a system-maintained REFRESH DEFERRED materialized query table such that the materialized query table can be used to optimize the processing of a query. The data type of this property is long.	Long	0
currentSchema	Sets the default schema name. The data type of this property is String.	String	-
cursorSensitivity	Should ResultSet.TYPE_SCROLL_SENSITIVE map to sensitive dynamic, sensitive static, or asensitive cursor on the server. 0 = insensitive, 1 = sensitive dynamic, 2 = asensitive. The data type of this property is int.	Integer (0, 1, 2)	0
currentSQLID	Specifies: The authorization ID that is used for authorization checking on dynamically prepared CREATE, GRANT, and REVOKE SQL statements. The owner of a table space, database, storage group, or synonym that is created by a dynamically issued CREATE statement. The implicit qualifier of all table, view, alias, and index names specified in dynamic SQL statements. .The data type of this property is String.	String	-
databaseName	Specifies the name for the data source. This name is used as the database portion of the connection URL. The name depends on whether IBM Data Server Driver for JDBC and SQLJ type 4 connectivity or IBM Data Server Driver for JDBC and SQLJ type 2 connectivity is used.	String	-
dateFormat	Defines the format for date values when using PreparedStatement.setString() for a DATE column, and the format of the string returned by ResultSet.getString() on a DATE column. The data type of this property is int. 1 = ISO, 2 = USA, 3 = EUR, 4 = JIS.	Integer	1
decimalRoundingMode	Sets the rounding mode for decimal floating-point (DECFLOAT) operations performed by the driver . Affects how results are rounded to the target precision. The data type of this property is int. 1 = Round down, 2 = round ceiling, 3 = round half even, 4 = round half up.	Integer (1, 2, 3, 4)	3
decimalSeparator	Specifies the decimal separator for input and output, for decimal, floating point, or decimal floating-point data values. The data type of this property is int. 0 = NOT SET, 1 = SEPARATOR PERIOD, 2 = SEPARATOR COMMA.	Integer (0, 1, 2)	0
decimalStringFormat	Specifies the string format for data that is retrieved from a DECIMAL or DECFLOAT column when the SDK for Java™ is Version 1.5 or later. The data type of this property is int. 0 = FORMAT NOT SET, 1 = FORMAT TO STRING, 2 = FORMAT TO PLAIN STRING.	Integer (0, 1, 2)	0
defaultIsolationLevel	Specifies the default transaction isolation level for new connections. The data type of this property is int. When defaultIsolationLevel is set on a	Integer	Depends on data source

Key	Description	Possible values	Default
	DataSource, all connections that are created from that DataSource have the default isolation level that is specified by defaultIsolationLevel.		
deferPrepares	Specifies whether invocation of the Connection.prepareStatement method results in immediate preparation of an SQL statement on the data source, or whether statement preparation is deferred until the PreparedStatement.execute method is executed. The data type of this property is boolean.	Boolean	false
description	A description of the data source. The data type of this property is String.	String	-
downgradeHoldCursorsUnderXa	Specifies whether cursors that are defined WITH HOLD can be opened under XA connections.	Boolean	false
driverType	For the DataSource interface, determines which driver to use for connections. The data type of this property is int. Valid values are 2 or 4. 2 is the default.	Integer(2, 4)	2
dumpPool	Specifies the types of statistics on global transport pool events that are written, in addition to summary statistics. The data type of this property is int. 0 = NOT_SET, 1 = DUMP_REMOVE_OBJECT, 2 = DUMP_GET_OBJECT, 4 = DUMP_WAIT_OBJECT, 8 = DUMP_SET_AVAILABLE_OBJECT, 16 = DUMP_CREATE_OBJECT, 32 = DUMP_SYSPLEX_MSG, 128 = DUMP_POOL_ERROR. You can combine the constants by using numeric addition to trace multiple options.	Integer (options in desc.)	0
dumpPoolStatisticsOnSchedule	Specifies how often, in seconds, global transport pool statistics are written to the file that is specified by dumpPoolStatisticsOnScheduleFile. The data type of this property is int.	Integer	-1 (stats are not written)
dumpPoolStatisticsOnScheduleFile	Specifies the name of the file to which global transport pool statistics are written. The global transport pool is used for the connection concentrator and Sysplex workload balancing. The data type of this property is String.	String	-
enableAlternateGroupSeamlessACR	Specifies whether failover to an alternate group is seamless or non-seamless. The data type of this property is Boolean.	Boolean	false
enableClientAffinitiesList	Specifies whether the IBM Data Server Driver for JDBC and SQLJ enables client affinities for cascaded failover support. The data type of this property is int. 1 = enable, 0 = disable.	Integer (0, 1)	0
enableExtendedIndicators	Specifies whether support for extended indicators is enabled in the IBM Data Server Driver for JDBC and SQLJ. The data type of this property is int.	Integer (0, 1, 2)	0
enableNamedParameterMarkers	Specifies whether support for named parameter markers is enabled in the IBM Data Server Driver for JDBC and SQLJ. The data type of this property is int. Not set = 0, YES = 1, NO = 2.	Integer (0, 1, 2)	0
enableConnectionConcentrator	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is Boolean.	Boolean	false
enableMultiRowInsertSupport	Specifies whether the IBM Data Server Driver for JDBC and SQLJ uses multi-row INSERT for batched INSERT or MERGE operations, when the target data server is a Db2 for z/OS server that supports multi-row INSERT. The data type of this property is Boolean.	Boolean	true
enableRowsetSupport	Specifies whether the IBM Data Server Driver for JDBC and SQLJ uses multiple-row FETCH for forward-only cursors or scrollable cursors, if the data server supports multiple-row FETCH. The data type of this property is int.	Integer (0, 1, 2)	0
enableSeamlessFailover	Specifies whether the IBM Data Server Driver for JDBC and SQLJ uses seamless failover for client reroute. The data type of this property is int. Not set = 0, YES = 1, NO = 2.	Integer (0, 1, 2)	2
enableSysplexWLB	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is Boolean.	Boolean	false
encryptionAlgorithm	Specifies whether the IBM Data Server Driver for JDBC and SQLJ uses 56-bit DES (weak) encryption (value = 1) or 256-bit AES (strong) encryption (value = 2). The data type of this property is int.	Integer	-
enableExtendedDescribe	Specifies whether the IBM Data Server Driver for JDBC and SQLJ requests extended describe information from the data server when it prepares a statement. The data type of this property is int. Not set = 0, YES = 1, NO = 2.	Integer(0, 1, 2)	0

Key	Description	Possible values	Default
enableT2zosLBF	IBM Data Server Driver for JDBC and SQLJ type 2 connectivity on Db2 for z/OS only. The data type of this property is int. Not set = 0, YES = 1, NO = 2.	Integer(0, 1, 2)	0
enableT2zosLBFSPResultSets	IBM Data Server Driver for JDBC and SQLJ type 2 connectivity on Db2 for z/OS only. The data type of this property is int. Not set = 0, YES = 1, NO = 2.	Integer(0, 1, 2)	0
enableTimeoutOnCursor	Control how statement timeout values are applied to specific SQL operations. The data type of this property is int.	Boolean	false
extendedTableInfo	he extendedTableInfo property specifies whether information about extended table types is returned from a DatabaseMetaData.getTables method call. Not set = 0, YES = 1, NO = 2.	Integer(0, 1, 2)	-
fetchSize	Specifies the default fetch size for ResultSet objects that are generated from Statement objects. The data type of this property is int.	Integer	0
floatingPointStringFormat	Specifies the format for data that is retrieved from a DOUBLE, FLOAT, or REAL column with the ResultSet.getString method. The data type of this property is int. LUW_TYPE2_DRIVER_FLOATING_POINT_STRING_FORMAT (2), JCC_DRIVER_FLOATING_POINT_STRING_FORMAT (1), NOT_SET (0)	Integer(0, 1, 2)	0
fullyMaterializeInputStreams	If true, any Java InputStreams provided as parameters are fully read into memory before sending to the server. The data type of this property is Boolean.	Boolean	false
fullyMaterializeLobData	Indicates whether the driver retrieves LOB locators for FETCH operations. The data type of this property is boolean.	Boolean	false
gssCredential	Used to establish a connection using Kerberos security with a delegated credential. The data type of this property is Object.	String	-
implicitRollbackOption	Specifies the actions that the IBM Data Server Driver for JDBC and SQLJ takes when a transaction encounters a deadlock or a timeout. 1 = not close connection, 2 = close connection, 0 = not set (throws exception)	Integer (0, 1, 2)	0
interruptProcessingMode	Specifies the behavior of the IBM Data Server Driver for JDBC and SQLJ when an application executes the Statement.cancel method. 1 = graceful cancel, 2 = forceful closes.	Integer (1, 2)	1
jdbcCollection	Specifies the collection ID for the packages that are used by an instance of the IBM Data Server Driver for JDBC and SQLJ at run time. The data type of this property is String.	String	NULLID
keepAliveTimeOut	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The KeepAliveTimeout keyword determines the number of seconds before the keepalive probes are sent as well as duration between probes. The data type of this property is int.	Integer	15
keepDynamic	Specifies if EnableKeepDynamic function is available to the various applications. The data type of this property is int.	Integer (0, 1)	0
kerberosServerPrincipal	The data type of this property is String.	String	-
keyUsage	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is int. 0 = NOT_SET, 1 = digitalSignature, 2 = nonRepudiation, 4 = keyEncipherment, 8 = dataEncipherment, 16 = keyAgreement, 32 = keyCertSign, 64 = cRLSign, 128 = encipherOnly, 256 = decipherOnly. You can combine the preferred keyUsage constants by using numeric addition.	Integer (options in desc.)	0
loginTimeout	The maximum time in seconds to wait for a connection to a data source. After the number of seconds that are specified by loginTimeout have elapsed, the driver closes the connection to the data source. The data type of this property is int. The default is 0. A value of 0 means that the timeout value is the default system timeout value. This property is not supported for IBM Data Server Driver for JDBC and SQLJ type 2 connectivity on DB2 for z/OS.	Integer	0 (default system timeout)
logWriter	The character output stream to which all logging and trace messages for the DataSource object are printed. The data type of this property is java.io.PrintWriter. The default value is null, which means that no logging or tracing for the DataSource is output.	java.io.PrintWriter	-
maxConnCachedParamBufferSize	IBM Data Server Driver for JDBC and SQLJ type 2 connectivity on Db2 for z/OS only. The data type of this property is int.	Integer	1048576 (It is 1MB)

Key	Description	Possible values	Default
maxRetriesForClientReroute	During automatic client reroute, limit the number of retries if the primary connection to the data source fails.	Integer	5
maxStatements	Controls an internal statement cache that is associated with a Connection. The data type of this property is int.	Integer	0
maxRowsetSize	IBM Data Server Driver for JDBC and SQLJ type 2 connectivity on Db2 for z/OS only. The data type of this property is int.	Integer	32767
maxTransportObjectIdleTime	time in sec., a connection stays idle in the pool before it is closed, until minTransportObject is reached. The data type of this property is int.	Integer	60
maxTransportObjectWaitTime	f maxTransportObjects is reached - time in sec., an application waits to get a connection before throwing a SQLException. The data type of this property is int.	Integer	5
maxTransportObjects	max # of connections to DB2 server from this DataSource. The data type of this property is int.	Integer	-1 (no limit)
memberConnectTimeout	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is int. For Db2 for z/OS® servers, the default value is 1 second.	Integer	- / 1 (see desc.)
minTransportObjects	property in the IBM Data Server Driver for JDBC and SQLJ, specifies the lower limit for the number of transport objects in a global transport object pool. The data type of this property is int.	Integer	0
optimizationProfile	Specifies the name of an optimization profile to use for dynamic SQL, equivalent to setting the OPTIMIZATION PROFILE special register. The data type of this property is String.	String	-
optimizationProfileToFlush	Specifies the name of an optimization profile in the server's cache to flush (remove). The data type of this property is String.	String	-
password	The password to use for establishing connections. The data type of this property is String.	String	-
pdqProperties	Specifies properties that control the interaction between the IBM Data Server Driver for JDBC and SQLJ and the client optimization feature of pureQuery. The data type of this property is String.	String	-
pkList	IBM Data Server Driver for JDBC and SQLJ type 2 connectivity only. The data type of this property is String. If you specify this property, do not specify planName. If pkList, planName, and db2.jcc.pkList are not specified, then the value of pkList is NULLID.	String	NULLID (see desc.)
planName	IBM Data Server Driver for JDBC and SQLJ type 2 connectivity only. The data type of this property is String. If you specify this property, do not specify pkList. If pkList, planName, and db2.jcc.planName are not specified, NULLID. is used as the package list for the underlying CREATE THREAD call.	String	NULLID (see desc.)
plugin	The name of a client-side JDBC security plug-in. The data type of this property is Object.	Object	-
pluginName	Name of the server-side security plugin to use. The data type of this property is String.	String	gss_dashdb_iam (depends on env)
portNumber	The port number where the DRDA® server is listening for requests. The data type of this property is int.	Integer	446 / 448
progressiveStreaming	Specifies whether the JDBC driver uses progressive streaming when progressive streaming is supported on the data source.	Integer (0, 1, 2)	0
queryCloseImplicit	Specifies whether cursors are closed immediately after all rows are fetched. queryCloseImplicit applies only to connections to IBM Data Server Driver for JDBC and SQLJ type 4 connectivity to DB2 for z/OS Version 8 or later, and IBM Data Server Driver for JDBC and SQLJ type 4 connectivity or IBM Data Server Driver for JDBC and SQLJ type 2 connectivityDB2 Database for Linux, UNIX, and Windows Version 9.7 or later.	Integer (0, 1, 2, 3)	0
queryDataSize	Specifies a hint that is used to control the amount of query data, in bytes, that is returned from the data source on each fetch operation. This value can be used to optimize the application by controlling the number of trips to the data source that are required to retrieve data.	Integer	32767

Key	Description	Possible values	Default
queryTimeoutInterruptProcessingMode	Specifies what happens when the query timeout interval for a Statement object expires. 1 = graceful cancel, 2 = forceful closes.	Integer (1, 2)	1
readOnly	Specifies whether the connection is read-only. The data type of this property is Boolean.	Boolean	false
reportLongTypes	Specifies whether DatabaseMetaData methods report LONG VARCHAR and LONG VARGRAPHIC column data types as long data types. The data type of this property is short.	Short (0, 1, 2)	0
resultSetHoldability	Specifies whether cursors remain open after a commit operation. The data type of this property is int. NOT_SET (0), HOLD_CURSORS_OVER_COMMIT (1), CLOSE_CURSORS_AT_COMMIT (2).	Integer (0, 1, 2)	0
resultSetHoldabilityForCatalogQueries	Specifies whether cursors for queries that are executed on behalf of DatabaseMetaData methods remain open after a commit operation. DB2BaseDataSource.HOLD_CURSORS_OVER_COMMIT (1) Leave cursors for catalog queries open after a commit operation, regardless of the resultSetHoldability setting. DB2BaseDataSource.CLOSE_CURSORS_AT_COMMIT (2) Close cursors for catalog queries after a commit operation, regardless of the resultSetHoldability setting. DB2BaseDataSource.NOT_SET (0) Use the resultSetHoldability setting for catalog queries.	Integer (0, 1, 2)	0
retrieveMessagesFromServerOnGetMessage	Specifies whether JDBC SQLException.getMessage or SQLWarning.getMessage calls cause the IBM Data Server Driver for JDBC and SQLJ to invoke a DB2 for z/OS stored procedure that retrieves the message text for the error. The data type of this property is boolean. The default is false, which means that the full message text is not returned to the client.	Boolean	false
retryIntervalForClientReroute	For automatic client reroute, specifies the amount of time in seconds between connection retries.	Positive Integer + unit of time (E.g. 1m30s)	-
retryWithAlternativeSecurityMechanism	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is int. 1 = YES, 2 = NO.	Integer (1, 2)	-
returnAlias	Specifies whether the JDBC driver returns rows for table aliases and synonyms for DatabaseMetaData methods that return table information, such as getTables. The data type of this property is short	Integer (1, 2)	1
securityMechanism	Specifies the DRDA security mechanism. The data type of this property is int. 3 = CLEAR_TEXT_PASSWORD_SECURITY, 4 = USER_ONLY_SECURITY, 7 = ENCRYPTED_PASSWORD_SECURITY, 8 = STRONG_PASSWORD_SUBSTITUTE_SECURITY, 9 = ENCRYPTED_USER_AND_PASSWORD_SECURITY.	Integer (3, 4, 7, 8, 9)	3
sendCharInputsUTF8	Specifies whether the IBM Data Server Driver for JDBC and SQLJ converts character input data to the CCSID of the Db2 for z/OS database server, or sends the data in UTF-8 encoding for conversion by the database server. The data type of this property is int. 1 = YES, 2 = NO.	Integer (1, 2)	-
sendDataAsIs	Specifies that the IBM Data Server Driver for JDBC and SQLJ does not convert input parameter values to the target column data types. The data type of this property is boolean. The default is false.	Boolean	-
serverName	The host name or the TCP/IP address of the data source. The data type of this property is String.	String	-
sessionTimeZone	Specifies the setting for the CURRENT SESSION TIME_ZONE special register. The data type of this property is String.	String (-12:59 to +14:00)	-
sqljAvoidTimeStampConversion	Specifies whether the IBM Data Server Driver for JDBC and SQLJ prevents Java's adjustment of a timestamp value. The data type of this property is Boolean.	Boolean	false
sqljEnableClassLoaderSpecificProfiles	Specifies whether the IBM Data Server Driver for JDBC and SQLJ allows using and loading of SQLJ profiles with the same Java name in multiple J2EE application (.ear) files. The data type of this property is Boolean.	Boolean	-
ssid	IBM Data Server Driver for JDBC and SQLJ type 2 connectivity on Db2 for z/OS only. The data type of this property is String.	String	-
sslCertLocation	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is String.	String	-
sslCipherSuites		String	-

Key	Description	Possible values	Default
	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is String.		
sslConnection	Specifies whether the IBM Data Server Driver for JDBC and SQLJ uses an SSL socket to connect to the data source. If sslConnection is set to true, the connection uses an SSL socket. If sslConnection is set to false, the connection uses a plain socket. The data type of this property is Boolean.	Boolean	-
sslKeyStoreLocation	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is String.	String	-
sslKeyStorePassword	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is String.	String	-
sslKeyStoreType	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is String.	String	-
sslTrustStoreLocation	Specifies the name of the Java truststore on the client that contains the server certificate for an SSL connection. The data type of this property is String.	String	-
sslTrustStorePassword	Specifies the password for the Java truststore on the client that contains the server certificate for an SSL connection. The data type of this property is String.	String	-
sslTrustStoreType	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is String.	String	-
sslVersion	IBM Data Server Driver for JDBC and SQLJ type 4 connectivity only. The data type of this property is int.	String (E.g. TLSv1.3)	-
statementConcentrator	Specifies whether the IBM Data Server Driver for JDBC and SQLJ uses the data source's statement concentrator functionality. The data type of this property is int. 1 = STATEMENT_CONCENTRATOR_OFF, 2 = STATEMENT_CONCENTRATOR_WITH_LITERALS.	(1, 2)	-
streamBufferSize	Specifies the size, in bytes, of the JDBC driver buffers for chunking LOB or XML data. The JDBC driver uses the streamBufferSize value whether or not it uses progressive streaming. The data type of this property is int.	Integer	1048576 (1MB)
stripTrailingZerosForDecimalNumbers	Specifies whether the IBM Data Server Driver for JDBC and SQLJ removes trailing zeroes when it retrieves data from a DECFLOAT, DECIMAL, or NUMERIC column. This property is meaningful only if the SDK for Java is Version 1.5 or later. The data type of this property is int.	(1 (yes), 2 (no))	-
supportsAsynchronousXARollback	Specifies whether the IBM Data Server Driver for JDBC and SQLJ supports asynchronous XA rollback operations. The data type of this property is int. 1 = YES, 2 = NO.	Integer	2
sysSchema	Specifies the schema of the shadow catalog tables or views that are searched when an application invokes a DatabaseMetaData method. The data type of this property is String.	String	-
timerLevelForQueryTimeOut	Specifies the level at which the IBM Data Server Driver for JDBC and SQLJ creates a java.util.Timer object for waiting for query execution to time out. -1 = QUERYTIMEOUT_DISABLED, 1 = QUERYTIMEOUT_STATEMENT_LEVEL, 2 = QUERYTIMEOUT_CONNECTION_LEVEL	(-1, 1, 2)	-
timeFormat	Specifies: The format in which the String argument of the PreparedStatement.setString method against a TIME column must be specified. The format in which the result of the ResultSet.getString or CallableStatement.getString method against a TIME column is returned. The data type of this property is int. Timeformat: 1 = ISO, 2 = USA, 3 = EUR, 4 = JIS.	(1, 2, 3, 4)	-
timestampFormat	Specifies the format in which the result of the ResultSet.getString or CallableStatement.getString method against a TIMESTAMP column is returned.	String	-
timestampOutputType	Determines the Java type used when retrieving TIMESTAMPS via standard getters. The data type of this property is int. 1 = JDBC_TIMESTAMP, 2 = JCC_DBTIMESTAMP	Integer (1, 2)	-
timestampPrecisionReporting	Specifies whether trailing zeroes are truncated in the result of a ResultSet.getString call for a TIMESTAMP value. The data type of this property is int. 1 = TIMESTAMP_JDBC_STANDARD, 2 = TIMESTAMP_ZERO_PADDING.	Integer (1, 2)	-

Key	Description	Possible values	Default
traceDirectory	Specifies a directory into which trace information is written. The data type of this property is String. When traceDirectory is specified, trace information for multiple connections on the same DataSource is written to multiple files.	String	-
traceFile	Specifies the name of a file into which the IBM Data Server Driver for JDBC and SQLJ writes trace information. The data type of this property is String. The traceFile property is an alternative to the logWriter property for directing the output trace stream to a file.	String	-
traceFileAppend	Specifies whether to append to or overwrite the file that is specified by the traceFile property. The data type of this property is boolean. The default is false, which means that the file that is specified by the traceFile property is overwritten.	Boolean	-
traceFileCount	Specifies the maximum number of trace files for circular tracing. The IBM Data Server Driver for JDBC and SQLJ uses this property only when traceOption is set to DB2BaseDataSource.TRACE_OPTION_CIRCULAR (1). The data type of this property is int. The default value is 2.	Integer	-
traceFileSize	Specifies the maximum size of each trace file, for circular tracing. The IBM Data Server Driver for JDBC and SQLJ uses this property only when traceOption is set to DB2BaseDataSource.TRACE_OPTION_CIRCULAR (1). The data type of this property is int. The default value is 10485760 (10 MB).	Integer	-
traceLevel	Specifies what to trace. The data type of this property is int.	Integer	-
traceOption	Specifies the way in which trace data is collected. The data type of this property is int.	String (0, 1)	-
useCachedCursor	Specifies whether the underlying cursor for PreparedStatement objects is cached and reused on subsequent executions of the PreparedStatement. The data type of this property is Boolean.	Boolean	-
useJDBC4ColumnNameAndLabelSemantics	Specifies how the IBM Data Server Driver for JDBC and SQLJ handles column labels in ResultSetMetaData.getColumnNames, ResultSetMetaData.getColumnLabel, and ResultSet.findColumn method calls. The data type of this property is int.	(1 (yes), 2 (no))	-
useJDBC41DefinitionForGetColumns	Specifies whether the DatabaseMetaData.getColumns method returns a result set with a column with the name SCOPE_CATALOG or SCOPE_CATALOG.	(1 (yes), 2 (no))	-
user	The user ID to use for establishing connections.	String	-
useIdentityValLocalForAutoGeneratedKeys	Specifies whether the IBM Data Server Driver for JDBC and SQLJ uses only the SQL built-in function IDENTITY_VAL_LOCAL to determine automatically generated key values. The data type of this property is Boolean.	Boolean	-
useRowsetCursor	Specifies whether the IBM Data Server Driver for JDBC and SQLJ always uses multiple-row FETCH for scrollable cursors if the data source supports multiple-row FETCH. The data type of this property is Boolean.	Boolean	-
useTransactionRedirect	Specifies whether the DB2 system directs SQL statements to different database partitions for better performance. The data type of this property is Boolean.	Boolean	-
xaNetworkOptimization	Specifies whether XA network optimization is enabled for IBM Data Server Driver for JDBC and SQLJ type 4 connectivity. You might need to disable XA network optimization in an environment in which an XA Start and XA End are issued from one Java process, and an XA Prepare and an XA Commit are issued from another Java process. With XA network optimization, the XA Prepare can reach the data source before the XA End, which results in an XAER_PROTO error. To prevent the XAER_PROTO error, disable XA network optimization.	Boolean	true
xmlFormat	The data type of this property is int. XML_FORMAT_TEXTUAL = 0, XML_FORMAT_BINARY = 1.	(0, 1)	
DBANSIWARN	Specifies whether the IBM Data Server Driver for JDBC and SQLJ instructs the IBM Informix database to return an SQLWarning to the application if an SQL statement does not use ANSI-standard syntax.	Boolean (false and 0, true and 1)	false
DBDATE	Specifies the end-user format of DATE values.	String	Y4MD-
DBPATH		String	.

Key	Description	Possible values	Default
	Specifies a colon-separated list of values that identify the database servers that contain databases.		
DBSPACETEMP	Specifies a comma-separated or colon-separated list of existing dbspaces in which temporary tables are placed	String	-
DBTEMP	Specifies the full path name of an existing directory in which temporary files and temporary tables are placed.	String	/tmp
DBUPSPACE	Specifies the maximum amount of system disk space and maximum amount of memory, in kilobytes, that the UPDATE STATISTICS statement can use when it constructs multiple column distributions simultaneously. The format of DBUPSPACE is "maximum-disk-space:maximum-memory".	String	-
DELIMIDENT	Specifies whether delimited SQL identifiers can be used in an application.	Boolean	false
IFX_DIRECTIVES	Specifies whether the optimizer allows query optimization directives from within a query.	String (1 or on, 0 or off)	-
IFX_EXTDIRECTIVES	Specifies whether the optimizer allows external query optimization directives from the sysdirectives system catalog table to be applied to queries in existing applications. Possible values are:	String (1 or on, 0 or off)	-
IFX_UPDESC	Specifies whether a DESCRIBE of an UPDATE statement is permitted. Any non-null value indicates that a DESCRIBE of an UPDATE statement is permitted.	String	1
IFX_XASTDCOMPLIANCE_XAEND	Specifies whether global transactions are freed only after an explicit rollback, or after any rollback.	String (0, 1)	-
INFORMIXOPCACHE	Specifies the size of the memory cache, in kilobytes, for the staging-area blob space of the client application. The data type of this property is String. A value of "0" indicates that the cache is not used.	String	-
INFORMIXSTACKSIZE	Specifies the stack size, in kilobytes, that the database server uses for the primary thread of a client session.	String	-
NODEFDAC	Specifies whether the database server prevents default table privileges (SELECT, INSERT, UPDATE, and DELETE) from being granted to PUBLIC when a new table is created during the current session, in a database that is not ANSI compliant.	String (yes, no)	no
OPTCOMPIND	Specifies the preferred method for performing a join operation on an ordered pair of tables. The data type of this property is String.	Integer (0, 1, 2)	2
OPTOFC	Use the OPTOFC environment variable to enable optimize-OPEN-FETCH-CLOSE functionality in IBM® Informix® ESQL/C applications or other APIs (such as JDBC, ODBC, OLE DB, LIBDMI, and Lib C++) that use DECLARE and OPEN statements to establish a cursor.	String (0, 1)	0
PDQPRIORITY	The PDQPRIORITY environment variable determines the degree of parallelism that the database server uses and affects how the database server allocates resources, including memory, processors, and disk reads.	String	-
PSORT_DBTEMP	The PSORT_DBTEMP environment variable specifies the location where the database server writes the temporary files that the PSORT_NPROCS environment variable uses to perform a sort.	String	-
PSORT_NPROCS	The PSORT_NPROCS environment variable enables the database server to improve the performance of the parallel-process sorting package by allocating more threads for sorting	String (Max: 10)	-
STMT_CACHE	JDBC driver property: STMT_CACHE.	String	-
beginTranForResultSetScrollingAPIs	Attempt transaction enlistment when result set scrolling interfaces are used.	Boolean	true
beginTranForVendorAPIs	Attempt transaction enlistment when vendor interfaces are used.	Boolean	true
connectionSharing	Specifies how connections are matched for sharing.	String (MatchCurrentState, MatchOriginalRequest)	MatchOriginalRequest
enableClientInformation	...		
propagateClientIdentityUsingTrustedContext	...		

mysql connection properties

This is list of properties that are compatible with mysql.

They are inside given data source configuration file. For example: mysqlTest-source_1.conf

These properties are saved in this format there:

```
"properties": {  
  "ApplicationName": "testApp",  
  "testQuery": "SELECT 1",  
  "defaultCatalog": "test",  
  "url": "jdbc:mysql://localhost:3306/${database}"  
},
```

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Description	Possible values	Default
socketFactory	The name of the class that the driver should use for creating socket connections to the server. This class must implement the interface 'com.mysql.jdbc.SocketFactory' and have public no-args constructor.	String	com.mysql.cj.protocol.StandardSocketFactory
connectTimeout	Timeout for socket connect (in milliseconds), with 0 being no timeout. Only works on JDK-1.4 or newer. Defaults to '0'.	Integer	0
socketTimeout	Timeout on network socket operations (0, the default means no timeout).	Integer	0
connectionLifecycleInterceptors	A comma-delimited list of classes that implement "com.mysql.jdbc.ConnectionLifecycleInterceptor" that should notified of connection lifecycle events (creation, destruction, commit, rollback, setCatalog and setAutoCommit) and potentially alter the execution of these commands. ConnectionLifecycleInterceptors are "stackable", more than one interceptor may be specified via the configuration property as a comma-delimited list, with the interceptors executed in order from left to right.	String	-
useConfigs	Load the comma-delimited list of configuration properties before parsing the URL or applying user-specified properties. These configurations are explained in the 'Configurations' of the documentation.	String	-
authenticationPlugins	Comma-delimited list of classes that implement com.mysql.jdbc.AuthenticationPlugin and which will be used for authentication unless disabled by "disabledAuthenticationPlugins" property.	String	-
defaultAuthenticationPlugin	Name of a class implementing com.mysql.jdbc.AuthenticationPlugin which will be used as the default authentication plugin (see below). It is an error to use a class which is not listed in "authenticationPlugins" nor it is one of the built-in plugins. It is an error to set as default a plugin which was disabled with "disabledAuthenticationPlugins" property. It is an error to set this value to null or the empty string (i.e. there must be at least a valid default authentication plugin specified for the connection, meeting all constraints listed above).	String	-
disabledAuthenticationPlugins	Comma-delimited list of classes implementing com.mysql.jdbc.AuthenticationPlugin or mechanisms, i.e. "mysql_native_password". The authentication plugins or mechanisms listed will not be used for authentication which will fail	String	-

Key	Description	Possible values	Default
	if it requires one of them. It is an error to disable the default authentication plugin (either the one named by "defaultAuthenticationPlugin" property or the hard-coded one if "defaultAuthenticationPlugin" property is not set).		
disconnectOnExpiredPasswords	If "disconnectOnExpiredPasswords" is set to "false" and password is expired then server enters "sandbox" mode and sends ERR(08001, ER_MUST_CHANGE_PASSWORD) for all commands that are not needed to set a new password until a new password is set.	Boolean	true
interactiveClient	Set the CLIENT_INTERACTIVE flag, which tells MySQL to timeout connections based on INTERACTIVE_TIMEOUT instead of WAIT_TIMEOUT	Boolean	false
localSocketAddress	Hostname or IP address given to explicitly configure the interface that the driver will bind the client side of the TCP/IP connection to when connecting.	String	-
propertiesTransform	An implementation of com.mysql.jdbc.ConnectionPropertiesTransform that the driver will use to modify URL properties passed to the driver before attempting a connection	String	-
useCompression	Use zlib compression when communicating with the server (true/false)? Defaults to 'false'.	Boolean	false
socksProxyHost	Name or IP address of SOCKS host to connect through.	String	-
socksProxyPort	Port of SOCKS server.	Integer	1080
maxAllowedPacket	Maximum allowed packet size to send to server. If not set, the value of system variable 'max_allowed_packet' will be used to initialize this upon connecting. This value will not take effect if set larger than the value of 'max_allowed_packet'. Also, due to an internal dependency with the property "blobSendChunkSize", this setting has a minimum value of "8203" if "useServerPrepStmts" is set to "true".	Integer	65535
tcpKeepAlive	If connecting using TCP/IP, should the driver set SO_KEEPALIVE?	Boolean	true
tcpNoDelay	If connecting using TCP/IP, should the driver set SO_TCP_NODELAY (disabling the Nagle Algorithm)?	Boolean	true
tcpRcvBuf	If connecting using TCP/IP, should the driver set SO_RCV_BUF to the given value? The default value of '0', means use the platform default value for this property)	Integer	0
tcpSndBuf	If connecting using TCP/IP, should the driver set SO_SND_BUF to the given value? The default value of '0', means use the platform default value for this property)	Integer	0
tcpTrafficClass	If connecting using TCP/IP, should the driver set traffic class or type-of-service fields ?See the documentation for java.net.Socket.setTrafficClass() for more information.	Integer	0
autoReconnect	Should the driver try to re-establish stale and/or dead connections? If enabled the driver will throw an exception for a queries issued on a stale or dead connection, which belong to the current transaction, but will attempt reconnect before the next query issued on the connection in a new transaction. The use of this feature is not recommended, because it has side effects related to session state and data consistency when applications don't handle SQLExceptions properly, and is only designed to be used when you are unable to configure your application to handle SQLExceptions resulting from dead and stale connections properly. Alternatively, as a last option, investigate setting the MySQL server variable "wait_timeout" to a high value, rather than the default of 8 hours.	Boolean	false
autoReconnectForPools	Use a reconnection strategy appropriate for connection pools (defaults to 'false')	Boolean	false
failOverReadOnly	When failing over in autoReconnect mode, should the connection be set to 'read-only'?	Boolean	true
maxReconnects	Maximum number of reconnects to attempt if autoReconnect is true, default is '3'.	Integer	3
reconnectAtTxEnd		Boolean	false

Key	Description	Possible values	Default
	If <code>autoReconnect</code> is set to true, should the driver attempt reconnections at the end of every transaction?		
<code>retriesAllDown</code>	When using loadbalancing or failover, the number of times the driver should cycle through available hosts, attempting to connect. Between cycles, the driver will pause for 250ms if no servers are available.	Integer	120
<code>initialTimeout</code>	If <code>autoReconnect</code> is enabled, the initial time to wait between re-connect attempts (in seconds, defaults to '2').	Integer	2
<code>roundRobinLoadBalance</code>	When <code>autoReconnect</code> is enabled, and <code>failoverReadonly</code> is false, should we pick hosts to connect to on a round-robin basis?	Boolean	false
<code>queriesBeforeRetryMaster</code>	Number of queries to issue before falling back to the primary host when failed over (when using multi-host failover). Whichever condition is met first, 'queriesBeforeRetryMaster' or 'secondsBeforeRetryMaster' will cause an attempt to be made to reconnect to the primary host. Setting both properties to 0 disables the automatic fall back to the primary host at transaction boundaries. Defaults to 50.	Integer	50
<code>secondsBeforeRetryMaster</code>	How long should the driver wait, when failed over, before attempting to reconnect to the primary host? Whichever condition is met first, 'queriesBeforeRetryMaster' or 'secondsBeforeRetryMaster' will cause an attempt to be made to reconnect to the master. Setting both properties to 0 disables the automatic fall back to the primary host at transaction boundaries. Time in seconds, defaults to 30	Integer	30
<code>allowMasterDownConnections</code>	By default, a replication-aware connection will fail to connect when configured master hosts are all unavailable at initial connection. Setting this property to 'true' allows to establish the initial connection, by failing over to the slave servers, in read-only state. It won't prevent subsequent failures when switching back to the master hosts i.e. by setting the replication connection to read/write state.	Boolean	false
<code>allowSlaveDownConnections</code>	By default, a replication-aware connection will fail to connect when configured slave hosts are all unavailable at initial connection. Setting this property to 'true' allows to establish the initial connection. It won't prevent failures when switching to slaves i.e. by setting the replication connection to read-only state. The property 'readFromMasterWhenNoSlaves' should be used for this purpose.	Boolean	false
<code>readFromMasterWhenNoSlaves</code>	Replication-aware connections distribute load by using the master hosts when in read/write state and by using the slave hosts when in read-only state. If, when setting the connection to read-only state, none of the slave hosts are available, an <code>SQLException</code> is thrown back. Setting this property to 'true' allows to fail over to the master hosts, while setting the connection state to read-only, when no slave hosts are available at switch instant.	Boolean	true
<code>replicationEnableJMX</code>	Enables JMX-based management of load-balanced connection groups, including live addition/removal of hosts from load-balancing pool.	Boolean	false
<code>selfDestructOnPingMaxOperations</code>	=If set to a non-zero value, the driver will report close the connection and report failure when <code>Connection.ping()</code> or <code>Connection.isValid(int)</code> is called if the connection's count of commands sent to the server exceeds this value.	Integer	0
<code>selfDestructOnPingSecondsLifetime</code>	If set to a non-zero value, the driver will report close the connection and report failure when <code>Connection.ping()</code> or <code>Connection.isValid(int)</code> is called if the connection's lifetime exceeds this value.	Integer	0
<code>replicationConnectionGroup</code>	Logical group of replication connections within a classloader, used to manage different groups independently. If not specified, live management of replication connections is disabled.	String	-
<code>resourceId</code>	A globally unique name that identifies the resource that this datasource or connection is connected to, used for <code>XAResource.isSameRM()</code> when the driver can't determine this value based on hostnames used in the URL	String	-
<code>allowMultiQueries</code>	Allow the use of ';' to delimit multiple queries during one statement (true/false), defaults to 'false', and does not affect the <code>addBatch()</code> and <code>executeBatch()</code> methods, which instead rely on <code>rewriteBatchStatements</code> .	Boolean	false

Key	Description	Possible values	Default
requireSSL	Require server support of SSL connection if useSSL=true? (defaults to 'false').	Boolean	false
verifyServerCertificate	If "useSSL" is set to "true", should the driver verify the server's certificate? When using this feature, the keystore parameters should be specified by the "clientCertificateKeyStore*" properties, rather than system properties. Default is 'false' when connecting to MySQL 5.5.45+, 5.6.26+ or 5.7.6+ and "useSSL" was not explicitly set to "true". Otherwise default is 'true'	Boolean	false
clientCertificateKeyStoreUrl	URL to the client certificate KeyStore (if not specified, use defaults)	String	-
clientCertificateKeyStoreType	KeyStore type for client certificates (NULL or empty means use the default, which is "JKS". Standard keystore types supported by the JVM are "JKS" and "PKCS12", your environment may have more available depending on what security products are installed and available to the JVM.	String	JKS
clientCertificateKeyStorePassword	Password for the client certificates KeyStore	String	-
trustCertificateKeyStoreUrl	URL to the trusted root certificate KeyStore (if not specified, use defaults)	String	-
trustCertificateKeyStoreType	KeyStore type for trusted root certificates (NULL or empty means use the default, which is "JKS". Standard keystore types supported by the JVM are "JKS" and "PKCS12", your environment may have more available depending on what security products are installed and available to the JVM.	String	JKS
trustCertificateKeyStorePassword	Password for the trusted root certificates KeyStore	String	-
enabledSSLCipherSuites	If "useSSL" is set to "true", overrides the cipher suites enabled for use on the underlying SSL sockets. This may be required when using external JSSE providers or to specify cipher suites compatible with both MySQL server and used JVM.	String	-
allowLoadLocalInfile	Should the driver allow use of 'LOAD DATA LOCAL INFILE...' (defaults to "true" <= 8.0.26, "false" in newer versions).	Boolean	true/false (depend on version, more in description)
allowUrlInLocalInfile	Should the driver allow URLs in 'LOAD DATA LOCAL INFILE' statements?	Boolean	false
allowPublicKeyRetrieval	Allows special handshake roundtrip to get server RSA public key directly from server.	Boolean	false
paranoid	Take measures to prevent exposure sensitive information in error messages and clear data structures holding sensitive data when possible? (defaults to 'false')	Boolean	false
passwordCharacterEncoding	What character encoding is used for passwords? Leaving this set to the default value (null), uses the value set in "characterEncoding" if there is one, otherwise uses UTF-8 as default encoding. If the password contains non-ASCII characters, the password encoding must match what server encoding was set to when the password was created. For passwords in other character encodings, the encoding will have to be specified with this property (or with "characterEncoding"), as it's not possible for the driver to auto-detect this.	String	-
serverRSAPublicKeyFile	File path to the server RSA public key file for sha256_password authentication. If not specified, the public key will be retrieved from the server.	String	-
callableStmtCacheSize	If 'cacheCallableStmts' is enabled, how many callable statements should be cached?	Integer	100
metadataCacheSize	The number of queries to cache ResultSetMetadata for if cacheResultSetMetadata is set to 'true' (default 50)	Integer	50
useLocalTransactionState	Should the driver use the in-transaction state provided by the MySQL protocol to determine if a commit() or rollback() should actually be sent to the database?	Boolean	false
prepStmtCacheSize	If prepared statement caching is enabled, how many prepared statements should be cached?	Integer	25

Key	Description	Possible values	Default
prepStmtCacheSqlLimit	If prepared statement caching is enabled, what's the largest SQL the driver will cache the parsing for?	Integer	256
parseInfoCacheFactory	Name of a class implementing <code>com.mysql.jdbc.CacheAdapterFactory</code> , which will be used to create caches for the parsed representation of client-side prepared statements.	String	<code>com.mysql.cj.PerConnectionLRUFactory</code>
serverConfigCacheFactory	Name of a class implementing <code>com.mysql.jdbc.CacheAdapterFactory</code> , which will be used to create caches for MySQL server configuration values	String	<code>com.mysql.cj.util.PerVmServerConfigCacheFactory</code>
alwaysSendSetIsolation	Should the driver always communicate with the database when <code>Connection.setTransactionIsolation()</code> is called? If set to false, the driver will only communicate with the database when the requested transaction isolation is different than the whichever is newer, the last value that was set via <code>Connection.setTransactionIsolation()</code> , or the value that was read from the server when the connection was established. Note that <code>useLocalSessionState=true</code> will force the same behavior as <code>alwaysSendSetIsolation=false</code> , regardless of how <code>alwaysSendSetIsolation</code> is set.	Boolean	true
maintainTimeStats	Should the driver maintain various internal timers to enable idle time calculations as well as more verbose error messages when the connection to the server fails? Setting this property to false removes at least two calls to <code>System.currentTimeMillis()</code> per query.	Boolean	true
blobSendChunkSize	Chunk size to use when sending BLOB/CLOBs via <code>ServerPreparedStatement</code> s. Note that this value cannot exceed the value of "maxAllowedPacket" and, if that is the case, then this value will be corrected automatically.	Integer	1048576
cacheCallableStmts	Should the driver cache the parsing stage of <code>CallableStatements</code>	Boolean	false
cachePrepStmts	Should the driver cache the parsing stage of <code>PreparedStatement</code> s of client-side prepared statements, the "check" for suitability of server-side prepared and server-side prepared statements themselves?	Boolean	false
cacheResultSetMetadata	Should the driver cache <code>ResultSetMetaData</code> for <code>Statements</code> and <code>PreparedStatement</code> s? (Req. JDK-1.4+, true/false, default 'false')	Boolean	false
cacheServerConfiguration	Should the driver cache the results of 'SHOW VARIABLES' and 'SHOW COLLATION' on a per-URL basis?	Boolean	false
defaultFetchSize	The driver will call <code>setFetchSize(n)</code> with this value on all newly-created <code>Statements</code>	Integer	0
dontCheckOnDuplicateKeyUpdateInSQL	Stops checking if every INSERT statement contains the "ON DUPLICATE KEY UPDATE" clause. As a side effect, obtaining the statement's generated keys information will return a list where normally it wouldn't. Also be aware that, in this case, the list of generated keys returned may not be accurate. The effect of this property is canceled if set simultaneously with 'rewriteBatchedStatements=true'.	Boolean	false
dontTrackOpenResources	The JDBC specification requires the driver to automatically track and close resources, however if your application doesn't do a good job of explicitly calling <code>close()</code> on statements or result sets, this can cause memory leakage. Setting this property to true relaxes this constraint, and can be more memory efficient for some applications. Also the automatic closing of the <code>Statement</code> and current <code>ResultSet</code> in <code>Statement.closeOnCompletion()</code> and <code>Statement.getMoreResults()</code> (<code>Statement.CLOSE_CURRENT_RESULT</code> <code>Statement.CLOSE_ALL_RESULTS</code>), respectively, ceases to happen. This property automatically sets <code>holdResultsOpenOverStatementClose=true</code> .	Boolean	false
dynamicCalendars	Should the driver retrieve the default calendar when required, or cache it per connection/session?	Boolean	false
enableEscapeProcessing	Sets the default escape processing behavior for <code>Statement</code> objects. The method <code>Statement.setEscapeProcessing()</code> can be used to specify the escape processing behavior for an individual <code>Statement</code> object. Default escape processing behavior in prepared statements must be defined with the property 'processEscapeCodesForPrepStmts'.	Boolean	true

Key	Description	Possible values	Default
enableQueryTimeouts	When enabled, query timeouts set via <code>Statement.setQueryTimeout()</code> use a shared <code>java.util.Timer</code> instance for scheduling. Even if the timeout doesn't expire before the query is processed, there will be memory used by the <code>TimerTask</code> for the given timeout which won't be reclaimed until the time the timeout would have expired if it hadn't been cancelled by the driver. High-load environments might want to consider disabling this functionality.	Boolean	true
largeRowSizeThreshold	What size result set row should the JDBC driver consider "large", and thus use a more memory-efficient way of representing the row internally?	Integer	2048
loadBalanceStrategy	If using a load-balanced connection to connect to SQL nodes in a MySQL Cluster/NDB configuration (by using the URL prefix <code>"jdbc:mysql:loadbalance://"</code>), which load balancing algorithm should the driver use: (1) "random" - the driver will pick a random host for each request. This tends to work better than round-robin, as the randomness will somewhat account for spreading loads where requests vary in response time, while round-robin can sometimes lead to overloaded nodes if there are variations in response times across the workload. (2) "bestResponseTime" - the driver will route the request to the host that had the best response time for the previous transaction.	String	random
locatorFetchBufferSize	If 'emulateLocators' is configured to 'true', what size buffer should be used when fetching BLOB data for <code>getBinaryInputStream()</code> ?	Integer	1048576
readOnlyPropagatesToServer	Should the driver issue appropriate statements to implicitly set the transaction access mode on server side when <code>Connection.setReadOnly()</code> is called? Setting this property to 'true' enables InnoDB read-only potential optimizations but also requires an extra roundtrip to set the right transaction state. Even if this property is set to 'false', the driver will do its best effort to prevent the execution of database-state-changing queries. Requires minimum of MySQL 5.6.	Boolean	true
rewriteBatchedStatements	Should the driver use multiqueries (irregardless of the setting of "allowMultiQueries") as well as rewriting of prepared statements for INSERT into multi-value inserts when <code>executeBatch()</code> is called? Notice that this has the potential for SQL injection if using plain <code>java.sql.Statements</code> and your code doesn't sanitize input correctly. Notice that for prepared statements, server-side prepared statements can not currently take advantage of this rewrite option, and that if you don't specify stream lengths when using <code>PreparedStatement.set*Stream()</code> , the driver won't be able to determine the optimum number of parameters per batch and you might receive an error from the driver that the resultant packet is too large. <code>Statement.getGeneratedKeys()</code> for these rewritten statements only works when the entire batch includes INSERT statements. Please be aware using <code>rewriteBatchedStatements=true</code> with <code>INSERT .. ON DUPLICATE KEY UPDATE</code> that for rewritten statement server returns only one value as sum of all affected (or found) rows in batch and it isn't possible to map it correctly to initial statements; in this case driver returns 0 as a result of each batch statement if total count was 0, and the <code>Statement.SUCCESS_NO_INFO</code> as a result of each batch statement if total count was > 0.	Boolean	false
useDirectRowUnpack	Use newer result set row unpacking code that skips a copy from network buffers to a MySQL packet instance and instead reads directly into the result set row data buffers.	Boolean	true
useFastDateParsing	Use internal <code>String->Date/Time/Timestamp</code> conversion routines to avoid excessive object creation? This is part of the legacy date-time code, thus the property has an effect only when <code>"useLegacyDatetimeCode=true."</code>	Boolean	true
useFastIntParsing	Use internal <code>String->Integer</code> conversion routines to avoid excessive object creation?	Boolean	true
useJvmCharsetConverters	Always use the character encoding routines built into the JVM, rather than using lookup tables for single-byte character sets?	Boolean	true
useReadAheadInput	Use newer, optimized non-blocking, buffered input stream when reading from the server?	Boolean	true
logger	The name of a class that implements <code>"com.mysql.jdbc.log.Log"</code> that will be used to log messages to. (default is <code>"com.mysql.jdbc.log.StandardLogger"</code> , which logs to STDERR)	String	<code>com.mysql.cj.log.StandardLogger</code>

Key	Description	Possible values	Default
gatherPerfMetrics	Should the driver gather performance metrics, and report them via the configured logger every 'reportMetricsIntervalMillis' milliseconds?	Boolean	false
profileSQL	Trace queries and their execution/fetch times to the configured logger (true/false) defaults to 'false'	Boolean	false
profileSql	Deprecated, use 'profileSQL' instead. Trace queries and their execution/fetch times on STDERR (true/false) defaults to 'false'	Boolean	false
queryTimeout	Sets the default query timeout value, set by Heimdall via the setQueryTimeout method	Integer	0
reportMetricsIntervalMillis	If 'gatherPerfMetrics' is enabled, how often should they be logged (in ms)?	Integer	30000
maxQuerySizeToLog	Controls the maximum length/size of a query that will get logged when profiling or tracing	Integer	2048
packetDebugBufferSize	The maximum number of packets to retain when 'enablePacketDebug' is true	Integer	20
slowQueryThresholdMillis	If 'logSlowQueries' is enabled, how long should a query (in ms) before it is logged as 'slow'?	Integer	2000
slowQueryThresholdNanos	If 'useNanosForElapsedTime' is set to true, and this property is set to a non-zero value, the driver will use this threshold (in nanosecond units) to determine if a query was slow.	Integer	0
usageAdvisor	Should the driver issue 'usage' warnings advising proper and efficient usage of JDBC and MySQL Connector/J to the log (true/false, defaults to 'false')?	Boolean	false
autoGenerateTestcaseScript	Should the driver dump the SQL it is executing, including server-side prepared statements to STDERR?	Boolean	false
autoSlowLog	Instead of using slowQueryThreshold* to determine if a query is slow enough to be logged, maintain statistics that allow the driver to determine queries that are outside the 99th percentile?	Boolean	true
clientInfoProvider	The name of a class that implements the com.mysql.jdbc.JDBC4ClientInfoProvider interface in order to support JDBC-4.0's Connection.get/setClientInfo() methods	String	com.mysql.cj.jdbc.CommentClientInfoProvider
dumpMetadataOnColumnNotFound	Should the driver dump the field-level metadata of a result set into the exception message when ResultSet.findColumn() fails?	Boolean	false
dumpQueriesOnException	Should the driver dump the contents of the query sent to the server in the message for SQLExceptions?	Boolean	false
enablePacketDebug	When enabled, a ring-buffer of 'packetDebugBufferSize' packets will be kept, and dumped when exceptions are thrown in key areas in the driver's code	Boolean	false
explainSlowQueries	If 'logSlowQueries' is enabled, should the driver automatically issue an 'EXPLAIN' on the server and send the results to the configured log at a WARN level?	Boolean	false
includeInnodbStatusInDeadlockExceptions	Include the output of "SHOW ENGINE INNODB STATUS" in exception messages when deadlock exceptions are detected?	Boolean	false
includeThreadDumpInDeadlockExceptions	Include a current Java thread dump in exception messages when deadlock exceptions are detected?	Boolean	false
includeThreadNamesAsStatementComment	Include the name of the current thread as a comment visible in "SHOW PROCESSLIST", or in Innodb deadlock dumps, useful in correlation with "includeInnodbStatusInDeadlockExceptions=true" and "includeThreadDumpInDeadlockExceptions=true".	Boolean	false
logSlowQueries	Should queries that take longer than 'slowQueryThresholdMillis' be logged?	Boolean	false
logXaCommands	Should the driver log XA commands sent by MySQLXaConnection to the server, at the DEBUG level of logging?	Boolean	false

Key	Description	Possible values	Default
profilerEventHandler	Name of a class that implements the interface <code>com.mysql.jdbc.profiler.ProfilerEventHandler</code> that will be used to handle profiling/tracing events.	String	<code>com.mysql.cj.log.LoggingProfilerEventHandler</code>
resultSetSizeThreshold	If the usage advisor is enabled, how many rows should a result set contain before the driver warns that it is suspiciously large?	Integer	100
traceProtocol	Should trace-level network protocol be logged?	Boolean	false
useNanosForElapsedTime	For profiling/debugging functionality that measures elapsed time, should the driver try to use nanoseconds resolution if available (JDK >= 1.5)?	Boolean	false
characterSetResults	Character set to tell the server to return results as.	String	-
connectionAttributes	A comma-delimited list of user-defined key:value pairs (in addition to standard MySQL-defined key:value pairs) to be passed to MySQL Server for display as connection attributes in the <code>PERFORMANCE_SCHEMA.SESSION_CONNECT_ATTRS</code> table. Example usage: <code>connectionAttributes=key1:value1,key2:value2</code> This functionality is available for use with MySQL Server version 5.6 or later only. Earlier versions of MySQL Server do not support connection attributes, causing this configuration option to be ignored. Setting <code>connectionAttributes=none</code> will cause connection attribute processing to be bypassed, for situations where Connection creation/initialization speed is critical.	String	-
connectionCollation	If set, tells the server to use this collation via 'set collation_connection'	String	-
useBlobToStoreUTF8OutsideBMP	Tells the driver to treat [MEDIUM/LONG]BLOB columns as [LONG]VARCHAR columns holding text encoded in UTF-8 that has characters outside the BMP (4-byte encodings), which MySQL server can't handle natively.	Boolean	false
utf8OutsideBmpExcludedColumnNamePattern	When "useBlobToStoreUTF8OutsideBMP" is set to "true", column names matching the given regex will still be treated as BLOBs unless they match the regex specified for "utf8OutsideBmpIncludedColumnNamePattern". The regex must follow the patterns used for the <code>java.util.regex</code> package.	String	-
utf8OutsideBmpIncludedColumnNamePattern	Used to specify exclusion rules to "utf8OutsideBmpExcludedColumnNamePattern". The regex must follow the patterns used for the <code>java.util.regex</code> package.	String	-
loadBalanceEnableJMX	Enables JMX-based management of load-balanced connection groups, including live addition/removal of hosts from load-balancing pool.	Boolean	false
loadBalanceHostRemovalGracePeriod	Sets the grace period to wait for a host being removed from a load-balanced connection, to be released when it is currently the active host.	Integer	15000
useColumnNamesInFindColumn	Prior to JDBC-4.0, the JDBC specification had a bug related to what could be given as a "column name" to <code>ResultSet</code> methods like <code>findColumn()</code> , or getters that took a <code>String</code> property. JDBC-4.0 clarified "column name" to mean the label, as given in an "AS" clause and returned by <code>ResultSetMetaData.getColumnLabel()</code> , and if no AS clause, the column name. Setting this property to "true" will give behavior that is congruent to JDBC-3.0 and earlier versions of the JDBC specification, but which because of the specification bug could give unexpected results. This property is preferred over "useOldAliasMetadataBehavior" unless you need the specific behavior that it provides with respect to <code>ResultSetMetaData</code> .	Boolean	false
allowNaNAndInf	Should the driver allow NaN or +/- INF values in <code>PreparedStatement.setDouble()</code> ?	Boolean	false
autoClosePstmtStreams	Should the driver automatically call <code>.close()</code> on streams/readers passed as arguments via <code>set*()</code> methods?	Boolean	false
autoDeserialize	Should the driver automatically detect and de-serialize objects stored in BLOB fields?	Boolean	true
blobsAreStrings	Should the driver always treat BLOBs as Strings - specifically to work around dubious metadata returned by the server for GROUP BY clauses?	Boolean	false

Key	Description	Possible values	Default
cacheDefaultTimezone	Caches client's default time zone. This results in better performance when dealing with time zone conversions in Date and Time data types, however it won't be aware of time zone changes if they happen at runtime.	Boolean	true
capitalizeTypeNames	Capitalize type names in DatabaseMetaData? (usually only useful when using WebObjects, true/false, defaults to 'false')	Boolean	false
clobCharacterEncoding	The character encoding to use for sending and retrieving TEXT, MEDIUMTEXT and LONGTEXT values instead of the configured connection characterEncoding	String	-
clobberStreamingResults	This will cause a 'streaming' ResultSet to be automatically closed, and any outstanding data still streaming from the server to be discarded if another query is executed before all the data has been read from the server.	Boolean	false
compensateOnDuplicateKeyUpdateCounts	Should the driver compensate for the update counts of "ON DUPLICATE KEY" INSERT statements (2 = 1, 0 = 1) when using prepared statements?	Boolean	false
continueBatchOnError	Should the driver continue processing batch commands if one statement fails. The JDBC spec allows either way (defaults to 'true').	Boolean	true
createDatabaseIfNotExist	Creates the database given in the URL if it doesn't yet exist. Assumes the configured user has permissions to create databases.	Boolean	false
detectCustomCollations	Should the driver detect custom charsets/collations installed on server (true/false, defaults to 'false'). If this option set to 'true' driver gets actual charsets/collations from server each time connection establishes. This could slow down connection initialization significantly.	Boolean	false
emptyStringsConvertToZero	Should the driver allow conversions from empty string fields to numeric values of '0'?	Boolean	true
emulateLocators	Should the driver emulate java.sql.Blobs with locators? With this feature enabled, the driver will delay loading the actual Blob data until the one of the retrieval methods (getInputStream(), getBytes(), and so forth) on the blob data stream has been accessed. For this to work, you must use a column alias with the value of the column to the actual name of the Blob. The feature also has the following restrictions: The SELECT that created the result set must reference only one table, the table must have a primary key, the SELECT must alias the original blob column name, specified as a string, to an alternate name; the SELECT must cover all columns that make up the primary key.	Boolean	false
emulateUnsupportedPstmts	Should the driver detect prepared statements that are not supported by the server, and replace them with client-side emulated versions?	Boolean	true
exceptionInterceptors	Comma-delimited list of classes that implement com.mysql.jdbc.ExceptionInterceptor. These classes will be instantiated one per Connection instance, and all SQLExceptions thrown by the driver will be allowed to be intercepted by these interceptors, in a chained fashion, with the first class listed as the head of the chain.	String	-
functionsNeverReturnBlobs	Should the driver always treat data from functions returning BLOBs as Strings - specifically to work around dubious metadata returned by the server for GROUP BY clauses?	Boolean	false
generateSimpleParameterMetadata	Should the driver generate simplified parameter metadata for PreparedStatements when no metadata is available either because the server couldn't support preparing the statement, or server-side prepared statements are disabled?	Boolean	false
getProceduresReturnsFunctions	Pre-JDBC4 DatabaseMetaData API has only the getProcedures() and getProcedureColumns() methods, so they return metadata info for both stored procedures and functions. JDBC4 was extended with the getFunctions() and getFunctionColumns() methods and the expected behaviours of previous methods are not well defined. For JDBC4 and higher, default 'true' value of the option means that calls of DatabaseMetaData.getProcedures() and DatabaseMetaData.getProcedureColumns() return metadata for both procedures and functions as before, keeping backward compatibility. Setting this property to 'false' decouples Connector/J from its pre-JDBC4 behaviours for	Boolean	true

Key	Description	Possible values	Default
	DatabaseMetaData.getProcedures() and DatabaseMetaData.getProcedureColumns(), forcing them to return metadata for procedures only.		
ignoreNonTxTables	Ignore non-transactional table warning for rollback? (defaults to 'false').	Boolean	false
loadBalanceAutoCommitStatementRegex	When load-balancing is enabled for auto-commit statements (via loadBalanceAutoCommitStatementThreshold), the statement counter will only increment when the SQL matches the regular expression. By default, every statement issued matches.	String	-
loadBalanceAutoCommitStatementThreshold	When auto-commit is enabled, the number of statements which should be executed before triggering load-balancing to rebalance. Default value of 0 causes load-balanced connections to only rebalance when exceptions are encountered, or auto-commit is disabled and transactions are explicitly committed or rolled back.	Integer	0
loadBalanceBlacklistTimeout	Time in milliseconds between checks of servers which are unavailable, by controlling how long a server lives in the global blacklist.		
loadBalanceConnectionGroup	Logical group of load-balanced connections within a classloader, used to manage different groups independently. If not specified, live management of load-balanced connections is disabled.	String	-
loadBalanceExceptionChecker	Fully-qualified class name of custom exception checker. The class must implement com.mysql.jdbc.LoadBalanceExceptionChecker interface, and is used to inspect SQLExceptions and determine whether they should trigger fail-over to another host in a load-balanced deployment.	String	com.mysql.cj.jdbc.ha.StandardLoadBalanceExceptionChecker
loadBalancePingTimeout	Time in milliseconds to wait for ping response from each of load-balanced physical connections when using load-balanced Connection.	Integer	0
loadBalanceSQLExceptionSubclassFailover	Comma-delimited list of classes/interfaces used by default load-balanced exception checker to determine whether a given SQLException should trigger failover. The comparison is done using Class.getInstance(SQLException) using the thrown SQLException.	String	-
loadBalanceSQLStateFailover	Comma-delimited list of SQLState codes used by default load-balanced exception checker to determine whether a given SQLException should trigger failover. The SQLState of a given SQLException is evaluated to determine whether it begins with any value in the comma-delimited list.	String	-
loadBalanceValidateConnectionOnSwapServer	Should the load-balanced Connection explicitly check whether the connection is live when swapping to a new physical connection at commit/rollback?	Boolean	false
maxRows	The maximum number of rows to return (0, the default means return all rows).	Integer (-1 and 0 = return all)	-1
netTimeoutForStreamingResults	What value should the driver automatically set the server setting 'net_write_timeout' to when the streaming result sets feature is in use? (value has unit of seconds, the value '0' means the driver will not try and adjust this value)	Integer (0 = do not adjust)	600
noAccessToProcedureBodies	When determining procedure parameter types for CallableStatements, and the connected user can't access procedure bodies through "SHOW CREATE PROCEDURE" or select on mysql.proc should the driver instead create basic metadata (all parameters reported as IN VARCHARs, but allowing registerOutParameter() to be called on them anyway) instead of throwing an exception?	Boolean	false
noDatetimeStringSync	Don't ensure that ResultSet.getDatetimeType().toString().equals(ResultSet.getString())	Boolean	false
noTimezoneConversionForDateType	Don't convert DATE values using the server time zone if 'useTimezone'='true' or 'useLegacyDatetimeCode'='false'	Boolean	false
noTimezoneConversionForTimeType		Boolean	false

Key	Description	Possible values	Default
	Don't convert TIME values using the server time zone if 'useTimezone'='true'		
nullCatalogMeansCurrent	When DatabaseMetadataMethods ask for a 'catalog' parameter, does the value null mean use the current catalog? (this is not JDBC-compliant, but follows legacy behavior from earlier versions of the driver)	Boolean	true
nullNamePatternMatchesAll	Should DatabaseMetaData methods that accept *pattern parameters treat null the same as '%' (this is not JDBC-compliant, however older versions of the driver accepted this departure from the specification)	Boolean	true
overrideSupportsIntegrityEnhancementFacility	Should the driver return "true" for DatabaseMetaData.supportsIntegrityEnhancementFacility() even if the database doesn't support it to workaround applications that require this method to return "true" to signal support of foreign keys, even though the SQL specification states that this facility contains much more than just foreign key support (one such application being OpenOffice)?	Boolean	false
padCharsWithSpace	If a result set column has the CHAR type and the value does not fill the amount of characters specified in the DDL for the column, should the driver pad the remaining characters with space (for ANSI compliance)?	Boolean	false
pedantic	Follow the JDBC spec to the letter.	Boolean	false
pinGlobalTxToPhysicalConnection	When using XAConnections, should the driver ensure that operations on a given XID are always routed to the same physical connection? This allows the XAConnection to support "XA START ... JOIN" after "XA END" has been called	Boolean	false
populateInsertRowWithDefaultValues	When using ResultSets that are CONCUR_UPDATABLE, should the driver pre-populate the "insert" row with default values from the DDL for the table used in the query so those values are immediately available for ResultSet accessors? This functionality requires a call to the database for metadata each time a result set of this type is created. If disabled (the default), the default values will be populated by the an internal call to refreshRow() which pulls back default values and/or values changed by triggers.	Boolean	false
processEscapeCodesForPrepStmts	Should the driver process escape codes in queries that are prepared? Default escape processing behavior in non-prepared statements must be defined with the property 'enableEscapeProcessing'.	Boolean	true
queryTimeoutKillsConnection	If the timeout given in Statement.setQueryTimeout() expires, should the driver forcibly abort the Connection instead of attempting to abort the query?	Boolean	false
retainStatementAfterResultSetClose	Should the driver retain the Statement reference in a ResultSet after ResultSet.close() has been called. This is not JDBC-compliant after JDBC-4.0.	Boolean	false
rollbackOnPooledClose	Should the driver issue a rollback() when the logical connection in a pool is closed?	Boolean	true
runningCTS13	Enables workarounds for bugs in Sun's JDBC compliance testsuite version 1.3	Boolean	false
sendFractionalSeconds	Send fractional part from TIMESTAMP seconds. If set to false, the nanoseconds value of TIMESTAMP values will be truncated before sending any data to the server. This option applies only to prepared statements, callable statements or updatable result sets.	Boolean	true
serverTimezone	Override detection/mapping of time zone. Used when time zone from server doesn't map to Java time zone	String	-
statementInterceptors	A comma-delimited list of classes that implement "com.mysql.jdbc.StatementInterceptor" that should be placed "in between" query execution to influence the results. StatementInterceptors are "chainable", the results returned by the "current" interceptor will be passed on to the next in the chain, from left-to-right order, as specified in this property.	String	-
strictFloatingPoint	Used only in older versions of compliance test	Boolean	false

Key	Description	Possible values	Default
strictUpdates	Should the driver do strict checking (all primary keys selected) of updatable result sets (true, false, defaults to 'true')?	Boolean	true
transformedBitsBoolean	If the driver converts TINYINT(1) to a different type, should it use BOOLEAN instead of BIT for future compatibility with MySQL-5.0, as MySQL-5.0 has a BIT type?	Boolean	false
treatUtilDateAsTimestamp	Should the driver treat java.util.Date as a TIMESTAMP for the purposes of PreparedStatement.setObject()?	Boolean	true
ultraDevHack	Create PreparedStatements for prepareCall() when required, because UltraDev is broken and issues a prepareCall() for <i>all</i> statements? (true/false, defaults to 'false')	Boolean	false
useCursorFetch	Use a cursor to fetch results, note—may cause subtle issues with date-time result-sets with prepared statements. This option is needed for fetch size limits to be honored.	Boolean	false
useAffectedRows	Don't set the CLIENT_FOUND_ROWS flag when connecting to the server (not JDBC-compliant, will break most applications that rely on "found" rows vs. "affected rows" for DML statements), but does cause "correct" update counts from "INSERT ... ON DUPLICATE KEY UPDATE" statements to be returned by the server.	Boolean	false
useGmtMillisForDatetimes	Convert between session time zone and GMT before creating Date and Timestamp instances (value of 'false' leads to legacy behavior, 'true' leads to more JDBC-compliant behavior)? This is part of the legacy date-time code, thus the property has an effect only when "useLegacyDatetimeCode=true."	Boolean	false
useHostsInPrivileges	Add '@hostname' to users in DatabaseMetaData.getColumn/TablePrivileges() (true/false), defaults to 'true'.	Boolean	true
useJDBCCompliantTimezoneShift	Should the driver use JDBC-compliant rules when converting TIME/TIMESTAMP/DATETIME values' time zone information for those JDBC arguments which take a java.util.Calendar argument? This is part of the legacy date-time code, thus the property has an effect only when "useLegacyDatetimeCode=true."	Boolean	true
useLegacyDatetimeCode	Use code for DATE/TIME/DATETIME/TIMESTAMP handling in result sets and statements that consistently handles time zone conversions from client to server and back again, or use the legacy code for these datatypes that has been in the driver for backwards-compatibility? Setting this property to 'false' voids the effects of "useTimezone," "useJDBCCompliantTimezoneShift," "useGmtMillisForDatetimes," and "useFastDateParsing."	Boolean	true
useOldAliasMetadataBehavior	Should the driver use the legacy behavior for "AS" clauses on columns and tables, and only return aliases (if any) for ResultSetMetaData.getColumnName() or ResultSetMetaData.getTableName() rather than the original column/table name? In 5.0.x, the default value was true.	Boolean	false/true (depend on version, more in description)
useOldUTF8Behavior	Use the UTF-8 behavior the driver did when communicating with 4.0 and older servers. False is the default post 8.x.x, true in earlier versions.	Boolean	false/true (depend on version, more in description)
useOnlyServerErrorMessages	Don't prepend 'standard' SQLState error messages to error messages returned by the server.	Boolean	true
useSSPSCompatibleTimezoneShift	If migrating from an environment that was using server-side prepared statements, and the configuration property "useJDBCCompliantTimezoneShift" set to "true", use compatible behavior when not using server-side prepared statements when sending TIMESTAMP values to the MySQL server.	Boolean	false
useSqlStateCodes	Use SQL Standard state codes instead of 'legacy' X/Open/SQL state codes (true/false), default is 'true'	Boolean	true
useStreamLengthsInPrepStmts	Honor stream length parameter in PreparedStatement/ResultSet.setXXXStream() method calls (true/false, defaults to 'true')?	Boolean	true
useTimezone	Convert time/date types between client and server time zones (true/false, defaults to 'false')? This is part of the legacy date-time code, thus the property has an effect only when "useLegacyDatetimeCode=true."	Boolean	false

Key	Description	Possible values	Default
useUnbufferedInput	Don't use BufferedInputStream for reading data from the server	Boolean	true
characterEncoding	If 'useUnicode' is set to true, what character encoding should the driver use when dealing with strings? (8.0.25 and earlier: driver will try to use server's default. 8.0.26 and later: utf8mb4)	String	-
useUnicode	Should the driver use Unicode character encodings when handling strings? Should only be used when the driver can't determine the character set mapping, or you are trying to 'force' the driver to use a character set that MySQL either doesn't natively support (such as UTF-8), true/false, defaults to 'true'	Boolean	true
useSSL	Use SSL when communicating with the server (true/false), For 8.0.12 and earlier: default is 'true' when connecting to MySQL 5.5.45+, 5.6.26+ or 5.7.6+, otherwise default is 'false', For 8.0.13 and later default is 'true'	Boolean	false/true (depend on version, more in description)
azureDbHost	Parse the azure hostname field, and append to username as @hostname, for MySQL and Postgres	String	-
maxResultBuffer	Specify the size of the result buffer when not streaming results, will generate an exception if the result exceeds this size. Specify in units of k,m or g, or percent for total heap percent, i.e. 10percent	String	-
useServerPrepStmts	Fully prepare statements on the server, don't just emulate prepared statements on the client. This can be needed for the proxy to properly support certain language drivers	Boolean	false

Postgres connection properties

This is list of properties that are compatible with postgres.

They are inside given data source configuration file. For example: postgresTest-source_1.conf
These properties are saved in this format there:

```
"properties": {  
  "ApplicationName": "testApp",  
  "testQuery": "SELECT 1",  
  "defaultCatalog": "test",  
  "url": "jdbc:postgresql://localhost:5432/${database}"  
},
```

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Description	Possible values	Default
adaptiveFetch	Enables using of adaptive fetch, requires maxResultBuffer and defaultRowFetchSize to be set as well, only used in transactions	Boolean	false
adaptiveFetchMaximum	Specifies maximum fetch size computed during adaptive fetch	-1 (infinity) Integer	-1
adaptiveFetchMinimum	Specifies minimum fetch size computed during adaptive fetch	Integer	0
allowEncodingChanges	Allow for changes in client_encoding	Boolean	false
ApplicationName	Name of the Application (backend >= 9.0)	String	PostgreSQL JDBC Driver
assumeMinServerVersion	Assume the server is at least that version	String	null
autosave	Specifies what the driver should do if a query fails. In autosave=always mode, JDBC driver sets a savepoint before each query, and rolls back to that savepoint in case of failure. In autosave=never mode (default), no savepoint dance is made ever. In autosave=conservative mode, safeoint is set for each query, however the rollback is done only for rare cases like 'cached statement cannot change return type' or 'statement XXX is not valid' so JDBC driver rollback and retries	String	never
azureDbHost	Parse the azure hostname field, and append to username as @hostname, for MySQL and Postgres	String	-
binaryTransferDisable	Comma separated list of types to disable binary transfer. Either OID numbers or names. Overrides values in the driver default set and values set with binaryTransferEnable.	String	empty string
binaryTransferEnable	Comma separated list of types to enable binary transfer. Either OID numbers or names	String	empty string
binaryTransfer	Use binary format for sending and receiving data if possible	Boolean	true

Key	Description	Possible values	Default
cancelSignalTimeout	The timeout that is used for sending cancel command.	Integer	10
cleanupSavePoints	Determines if the SAVEPOINT created in autosave mode is released prior to the statement. This is done to avoid running out of shared buffers on the server in the case where 1000's of queries are performed.	Boolean	false
connectTimeout	The timeout value used for socket connect operations.	0 (Disabled) Integer	10
currentSchema	Specify the schema to be set in the search-path	0 (Disabled) or Integer (max 2147484)	null
databaseMetadataCacheFieldsMIB	Specifies the maximum size (in megabytes) of fields to be cached per connection. A value of {@code 0} disables the cache.	0 (Disabled) Integer	5
databaseMetadataCacheFields	Specifies the maximum number of fields to be cached per connection. A value of {@code 0} disables the cache.	0 (Disabled) Integer	65536
defaultRowFetchSize	Positive number of rows that should be fetched from the database when more rows are needed for ResultSet by each fetch iteration	0 (All rows at once is the default) or Integer	0 (All rows at once)
disableColumnSanitiser	Enable optimization that disables column name sanitiser	Boolean	off (false)
extraFloatDigits	If the JDBC driver should set extra_float_digits on connect, defaults to true (needs custom PG driver)	Boolean	true
gsslib	Force SSSPI or GSSAPI	String	auto
hdUsePGNotify	Leverage the Postgres 9.0+ notify interface to publish/listen for invalidations instead of the grid cache. Can be leveraged from triggers as well for real-time invalidation of out-of-band updates. Defaults to false (boolean).	Boolean	false
hdNotifyDBs	CSV list of database names to listen to for hdUsePGNotify.	String	postgres
hostRecheckSeconds	Specifies period (seconds) after which the host status is checked again in case it has changed	Integer	10
jaasApplicationName	Specifies the name of the JAAS system or application login configuration.	String	pgjdbc
jaasLogin	Login with JAAS before doing GSSAPI authentication	Boolean	true
kerberosServerName	The Kerberos service name to use when authenticating with GSSAPI.	String	postgres
loadBalanceHosts	If disabled hosts are connected in the given order. If enabled hosts are chosen randomly from the set of suitable candidates	Boolean	false
loggerFile	File name output of the Logger	String	System.err
loggerLevel	Logger level of the driver	String	OFF
loginTimeout	Specify how long to wait for establishment of a database connection.	Integer (max 2147484)	0
logUnclosedConnections	When connections that are not explicitly closed are garbage collected, log the stacktrace from the opening of the connection to trace the leak source	Boolean	false

Key	Description	Possible values	Default
options	Specify 'options' connection initialization parameter	String	-
preferQueryMode	Specifies which mode is used to execute queries to database: simple means ('Q' execute, no parse, no bind, text mode only), extended means always use bind/execute messages, extendedForPrepared means extended for prepared statements only, extendedCacheEverything means use extended protocol and try cache every statement (including Statement.execute(String sql)) in a query cache.	(String) extended, extendedForPrepared, extendedCacheEverything, simple	extended
preparedStatementCacheQueries	Specifies the maximum number of entries in per-connection cache of prepared statements. A value of {code 0} disables the cache.	0 (Disabled) or Integer	256
preparedStatementCacheSizeMiB	Specifies the maximum size (in megabytes) of a per-connection prepared statement cache. A value of {code 0} disables the cache.	0 (Disabled) or Integer	5
prepareThreshold	Statement prepare threshold. A value of {code -1} stands for forceBinary	-1 (force binary transfer) Integer	5
protocolVersion	The driver supports the V3 frontend/backend protocols. The V3 protocol was introduced in 7.4 and the driver will by default try to connect using the V3 protocol.	Integer	null
queryTimeout	Sets the default query timeout value, set by Heimdall via the setQueryTimeout method	0 (No limit) Integer	0
readOnly	Puts this connection in read-only mode	Boolean	false
receiveBufferSize	Socket read buffer size	Integer	-1
replication	Connection parameter passed in startup message, one of 'true' or 'database' Passing 'true' tells the backend to go into walsender mode, wherein a small set of replication commands can be issued instead of SQL statements. Only the simple query protocol can be used in walsender mode. Passing 'database' as the value instructs walsender to connect to the database specified in the dbname parameter, which will allow the connection to be used for logical replication from that database. (backend >= 9.4)	String (true and database)	false
rewriteBatchedInserts	Enable optimization to rewrite and collapse compatible INSERT statements that are batched.	Boolean	false
sendBufferSize	Socket write buffer size	Integer	-1
socketFactoryArg	Argument forwarded to constructor of SocketFactory class.	String	-
socketFactory	Specify a socket factory for socket creation	String	null
socketTimeout	The timeout value used for socket read operations.	Integer max(2147484)	0
sslcert	The location of the client's SSL certificate	String	defaultdir/postgresql.crt
ssl	Control use of SSL (any non-null value causes SSL to be required)	Boolean	false
ssifactoryarg	Argument forwarded to constructor of SSLSocketFactory class.	String	-
sslfactory	Provide a SSLSocketFactory class when using SSL.	String	org.postgresql.ssl.LibPQFactory

Key	Description	Possible values	Default
sslhostnameverifier	A class, implementing javax.net.ssl.HostnameVerifier that can verify the server	String	org.postgresql.ssl.PGjdbcHostnameVerifier
sslkey	The location of the client's PKCS#8 SSL key	String	defaultdir/postgresql.pk8
sslmode	Parameter governing the use of SSL	String (disable , allow , prefer , require , verify-ca, verify-full . require , allow and prefer)	prefer
sslpasswordcallback	A class, implementing javax.security.auth.callback.CallbackHandler that can handle PasswordCallback for the ssl password.	String	org.postgresql.ssl.jdbc4.LibPQFactory.ConsoleCallbackHandler
sslpassword	The password for the client's ssl key (ignored if sslpasswordcallback is set)	String	null
sslrootcert	The location of the root certificate for authenticating the server.	String	defaultdir/root.crt
sspiServiceClass	The Windows SSPI service class for SPN	String	POSTGRES
stringtype	The type to bind String parameters as (usually 'varchar', 'unspecified' allows implicit casting to other types)	String	null
targetServerType	Specifies what kind of server to connect. The primary/secondary distinction is currently done by observing if the server allows writes. The value preferSecondary tries to connect to secondary if any are available, otherwise allows falls back to connecting also to primary. The value preferPrimary tries to connect to primary if it is available, otherwise allows falls back to connecting to secondaries available.	String (any, primary, master, slave, secondary, preferSlave, preferSecondary, preferPrimary)	any
tcpKeepAlive	Enable or disable TCP keep-alive. The default is {@code false}.	Boolean	false
unknownLength	Specifies the length to return for types of unknown length	Integer	Integer.MAX_VALUE
useSsl	Use SSL in SSPI authentication requests	Boolean	false
maxResultBuffer	Specify the size of the result buffer when not streaming results, will generate an exception if the result exceeds this size. Specify in units of k,m or g, or percent for total heap percent, i.e. 10percent	String	null

redshift connection properties

This is list of properties that are compatible with redshift.

They are inside given data source configuration file. For example: redshiftTest-source_1.conf
These properties are saved in this format there:

```
"properties": {  
  "ApplicationName": "testApp",  
  "testQuery": "SELECT 1",  
  "defaultCatalog": "test",  
  "url": "jdbc:redshift://localhost:5439/${database}"  
},
```

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Description	Possible values	Default
AccessKeyID	The IAM access key id for the IAM user or role	String	-
allowEncodingChanges	Allow for changes in client_encoding	Boolean	false
ApplicationName	Name of the Application (backend >= 9.0)	String	-
assumeMinServerVersion	Assume the server is at least that version	String	-
AuthMech	Parameter governing the use of SSL. Alias for sslMode	String (TRUE / FALSE)	TRUE
AutoCreate	Indicates whether the user should be created if not exists	Boolean	False
autosave	Specifies what the driver should do if a query fails. In autosave=always mode, JDBC driver sets a savepoint before each query, and rolls back to that savepoint in case of failure. In autosave=never mode (default), no savepoint dance is made ever. In autosave=conservative mode, safepoint is set for each query, however the rollback is done only for rare cases like 'cached statement cannot change return type' or 'statement XXX is not valid' so JDBC driver rollback and retries	String	never
binaryTransfer	Use binary format for sending and receiving data if possible	Boolean	true
binaryTransferDisable	Comma separated list of types to disable binary transfer. Either OID numbers or names. Overrides values in the driver default set and values set with binaryTransferEnable.	String	-
binaryTransferEnable	Comma separated list of types to enable binary transfer. Either OID numbers or names	String	-
BlockingRowsMode	Positive number of rows that should be fetched from the database when more rows are needed for ResultSet by each fetch iteration	Integer	0

Key	Description	Possible values	Default
cancelSignalTimeout	The timeout that is used for sending cancel command.	Integer	10
cleanupSavePoints	Determines if the SAVEPOINT created in autosave mode is released prior to the statement. This is done to avoid running out of shared buffers on the server in the case where 1000's of queries are performed.	Boolean	false
ClusterID	The name of the Redshift cluster to connect to.	String	-
connectTimeout	The timeout value used for socket connect operations. The timeout is specified in seconds. A value of 0 means that no timeout is specified.	Integer (0 no timeout)	10
currentSchema	Specify the schema to be set in the search-path	String	-
databaseMetadataCacheFields	Specifies the maximum number of fields to be cached per connection. A value of {@code 0} disables the cache.	Integer	65536
databaseMetadataCacheFieldsMiB	Specifies the maximum size (in megabytes) of fields to be cached per connection. A value of {@code 0} disables the cache.	Integer	5
databaseMetadataCurrentDbOnly	Control the behavior of metadata API to return data from all accessible databases or only from connected database	Boolean	true
DbGroups	A comma delimited database group names	String	PUBLIC
DbGroupsFilter	Regex for filtering out dbGroups from final result	String	-
defaultRowFetchSize	Positive number of rows that should be fetched from the database when more rows are needed for ResultSet by each fetch iteration	Integer (0 fetch all)	0
disableColumnSanitiser	Enable optimization that disables column name sanitiser	Boolean	off (false)
DisableIsValidQuery	Disable isValid query	Boolean	False
DSILogLevel	Log level of the driver	Integer (1 - 6)	?
enableFetchRingBuffer	The Redshift fetch rows using a ring buffer on a separate thread	Boolean	true
enableGeneratedName	The Redshift uses generated statement name and portal name	Boolean	-
enableMultiSqlSupport	Control the behavior of semicolon separated SQL commands in a Statement	Boolean	true
enableStatementCache	The Redshift statement cache using SQL as key	Boolean	-
EndpointUrl	The Redshift endpoint url	String	null
escapeSyntaxCallMode	Specifies how the driver transforms JDBC escape call syntax into underlying SQL, for invoking procedures or functions. (backend >= 1.1)\n + \n escapeSyntaxCallMode=select mode (the default), the driver always uses a SELECT statement (allowing function invocation only).\n + \n escapeSyntaxCallMode=callIfNoReturn mode, the driver uses a CALL statement (allowing procedure invocation) if there is no return parameter specified, otherwise the driver uses a SELECT statement.\n + \n escapeSyntaxCallMode=call mode, the	String	select

Key	Description	Possible values	Default
	driver always uses a CALL statement (allowing procedure invocation only).		
fetchRingBufferSize	Specifies size of ring buffer during fetching result set. Can be specified as specified size or percent of heap memory.	String	1G
ForceLowercase	Forces database group names to be lower case	Boolean	false
groupFederation	Use the IDP Groups in the Redshift	Boolean	false
gsslib	Force SSSPI or GSSAPI	String	auto
hideUnprivilegedObjects	Enable hiding of database objects for which the current user has no privileges granted from the DatabaseMetaData	Boolean	-
hostRecheckSeconds	Specifies period (seconds) after which the host status is checked again in case it has changed	Integer	10
IAMAuth	Indicates whether use IAM authentication	Boolean	-
IAMDisableCache	Indicates to disable credential cache. Enable cache gives protection against throttling API calls	Boolean	false
IAMDURATION	The length of time (in seconds) until the temporary IAM credentials expire.	Integer (min 900, max 3600)	900
IniFile	The JDBC INI file. Easy to configure connection properties.	String	-
IniSection	The JDBC INI file section name.	String	-
jaasApplicationName	Specifies the name of the JAAS system or application login configuration.	String	pgjdbc
jaasLogin	Login with JAAS before doing GSSAPI authentication	Boolean	true
kerberosServerName	The Kerberos service name to use when authenticating with GSSAPI.	String	postgres
loadBalanceHosts	If disabled hosts are connected in the given order. If enabled hosts are chosen randomly from the set of suitable candidates	Boolean	false
logServerErrorDetail	Include full server error detail in exception messages. If disabled then only the error itself will be included.	Boolean	true
loggerFile	File name output of the Logger	String	-
loggerLevel	Logger level of the driver	String	-
loginTimeout	Specify how long to wait for establishment of a database connection.	Integer (0 dont time out)	0
LogLevel	Log level of the driver	Integer (1 - 6)	0
LogPath	File Path output of the Logger. If the LogPath value is invalid, the driver sends the logged information to the standard output stream, System.out.	String	current working directory
logUnclosedConnections	When connections that are not explicitly closed are garbage collected, log the stacktrace from the opening of the connection to trace the leak source	Boolean	false
MaxLogFileCount	Maximum number of log files	Integer	-
MaxLogFileSize	Maximum single log file size	Integer	-

Key	Description	Possible values	Default
maxResultBuffer	Specify the size of the result buffer when not streaming results, will generate an exception if the result exceeds this size. Specify in units of k,m or g, or percent for total heap percent, i.e. 10percent	String	-
options	Specify 'options' connection initialization parameter	String	-
plugin_name	The fully qualified class path for a class that implements AWSCredentialsProvider	String	-
preferQueryMode	Specifies which mode is used to execute queries to database: simple means ('Q' execute, no parse, no bind, text mode only), extended means always use bind/execute messages, extendedForPrepared means extended for prepared statements only, extendedCacheEverything means use extended protocol and try cache every statement (including Statement.execute(String sql)) in a query cache.	String (extended, extendedForPrepared, extendedCacheEverything, simple)	extended
preparedStatementCacheQueries	Specifies the maximum number of entries in per-connection cache of prepared statements. A value of {code 0} disables the cache.	0 (Disabled) or Integer	256
preparedStatementCacheSizeMiB	Specifies the maximum size (in megabytes) of a per-connection prepared statement cache. A value of {code 0} disables the cache.	0 (Disabled) or Integer	5
prepareThreshold	Statement prepare threshold. A value of {code -1} stands for forceBinary	-1 (force binary transfer) Integer	5
Profile	The AWS profile name for credentials	String	-
protocolVersion	The driver supports the V3 frontend/backend protocols. The V3 protocol was introduced in 7.4 and the driver will by default try to connect using the V3 protocol.	Integer	-
providerName	The name of the Redshift Native Auth Provider	String	-
queryGroup	Assign a query to a queue at runtime by assigning your query to the appropriate query group	String	-
raiseExceptionOnSilentRollback	Certain database versions perform a silent rollback instead of commit in case the transaction was in a failed state	Boolean	-
readOnly	Puts this connection in read-only mode	Boolean	false
readOnlyMode	Controls the behavior when a connection is set to be read only, one of 'ignore', 'transaction', or 'always'\n + When 'ignore', setting readOnly has no effect.\n + When 'transaction' setting readOnly to 'true' will cause transactions to BEGIN READ ONLY if autocommit is 'false'. \n + When 'always' setting readOnly to 'true' will set the session to READ ONLY if autoCommit is 'true'\n + and the transaction to BEGIN READ ONLY if autocommit is 'false'.	String	transaction
receiveBufferSize	Socket read buffer size	Integer	-1
Region	The AWS region where the cluster is located	String	null
replication	Connection parameter passed in startup message, one of 'true' or 'database' Passing 'true' tells the backend to go into walsender mode, wherein a small set of replication commands can be issued instead of SQL	String (true and database)	false

Key	Description	Possible values	Default
	statements. Only the simple query protocol can be used in walsender mode. Passing 'database' as the value instructs walsender to connect to the database specified in the dbname parameter, which will allow the connection to be used for logical replication from that database. (backend >= 9.4)		
rewriteBatchedInserts	Enable optimization to rewrite and collapse compatible INSERT statements that are batched.	Boolean	false
rewriteBatchedInsertsSize	Enable optimization size to rewrite and collapse compatible INSERT statements that are batched. This must be power of 2	Integer	128
SecretAccessKey	The IAM secret key for the IAM user or role	String	-
sendBufferSize	Socket write buffer size	Integer	-1
SessionToken	The IAM security token for an IAM user or role	String	-
socketFactory	Specify a socket factory for socket creation	String	-
socketFactoryArg	Argument forwarded to constructor of SocketFactory class.	String	-
socketTimeout	The timeout value used for socket read operations.	Integer	0
ssl	Control use of SSL (any non-null value causes SSL to be required)	String (TRUE and FALSE)	TRUE
sslcert	The location of the client's SSL certificate	String	-
sslfactory	Provide a SSLSocketFactory class when using SSL.	String	-
sslfactoryarg	Argument forwarded to constructor of SSLSocketFactory class.	String	-
sslhostnameverifier	A class, implementing javax.net.ssl.HostnameVerifier that can verify the server	String	org.postgresql.ssl.PGjdbcHostnameVerifier
sslkey	The location of the client's PKCS#8 SSL key	String	-
sslmode	Parameter governing the use of SSL	String (disable , allow , prefer , require , verify-ca and verify-full . require , allow and prefer)	verify-ca
sslpassword	The password for the client's ssl key (ignored if sslpasswordcallback is set)	String	0
sslpasswordcallback	A class, implementing javax.security.auth.callback.CallbackHandler that can handle PassworCallback for the ssl password.	String	org.postgresql.ssl.jdbc4.LibPQFactory.ConsoleCallbackHandler
sslrootcert	The location of the root certificate for authenticating the server.	String	-
SSLTruststore	The SSL Truststore password key.	String	-
SSLTrustStorePath	The SSL Truststore path key.	String	-
sspiServiceClass	The Windows SSPI service class for SPN	String	POSTGRES
stringtype	The type to bind String parameters as (usually 'varchar', 'unspecified' allows implicit casting to other types)	String	null
StsEndpointUrl	The STS endpoint url	String	-

Key	Description	Possible values	Default
targetServerType	Specifies what kind of server to connect. The primary/secondary distinction is currently done by observing if the server allows writes. The value preferSecondary tries to connect to secondary if any are available, otherwise allows falls back to connecting also to primary. The value preferPrimary tries to connect to primary if it is available, otherwise allows falls back to connecting to secondaries available.	String (any, primary, master, slave, secondary, preferSlave, preferSecondary, preferPrimary)	any
tcpKeepAlive	Enable or disable TCP keep-alive. The default is { <code>@code false</code> }.	String (TRUE, FALSE)	TRUE
uid	Username to connect to the database as.	String	-
unknownLength	Specifies the length to return for types of unknown length	Integer	Integer.MAX_VALUE
useSpnego	Use SPNEGO in SSPI authentication requests	Boolean	false
webIdentityToken	The Oauth access token for an idp connection	String	-

sql server connection properties

This is list of properties that are compatible with SQL server.

They are inside given data source configuration file. For example: sqlserverTest-source_1.conf
These properties are saved in this format there:

```
"properties": {  
  "ApplicationName": "testApp",  
  "testQuery": "SELECT 1",  
  "defaultCatalog": "test",  
  "url": "jdbc:sqlserver://localhost:1433/databaseName=${database}"  
},
```

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Description	Possible values	Default
databaseName	The name of the database to connect to.	String <=128 char	null
serverName	The computer running SQL Server.	String	null
portNumber	The TCP port where an instance of SQL Server is listening.	Integer (0 - 65535)	1433
connectTimeout	The number of milliseconds a connect is allowed to take	Integer	15
serverSpn	SQL Server SPN in v5.0+	String	
columnEncryptionSetting	The column encryption setting.	String (Enabled, Disabled)	Disabled
serverNameAsACE	Translates the serverName from Unicode to ASCII Compatible Encoding (ACE), as defined by the ToASCII operation of RFC 3490.	Boolean	false
sendStringParametersAsUnicode	Determines if the string parameters are sent to the server as Unicode or the database's character set.	Boolean	true
multiSubnetFailover	Indicates that the application is connecting to the Availability Group Listener of an Availability Group or Failover Cluster Instance.	String (yes, no, true, false, 1, 0)	false
applicationName	The application name for SQL Server profiling and logging tools.	String <=128 char	null
lastUpdateCount	Ensures that only the last update count is returned from an SQL statement passed to the server.	Boolean	true
disableStatementPooling	Disables the statement pooling feature.	Boolean	true
integratedSecurity	Indicates whether Windows authentication will be used to connect to SQL Server.	Boolean	false
authenticationScheme	The authentication scheme to be used for integrated authentication. Possible values are JavaKerberos, NativeAuthentication, NTML.	String (avaKerberos, NativeAuthentication, NTML)	NativeAuthentication
lockTimeout	The number of milliseconds to wait before the database reports a lock time-out.	Integer (0 = no wait)	-1 (wait indefinitely)
loginTimeout	The number of seconds the driver should wait before timing out a failed connection. Default is 30 (version 11.2 and up), 15 (version 10.2 and below).	Integer (0 - 65535)	30 / 15 (see desc.)
instanceName	The name of the SQL Server instance to connect to.	String <=128 char	null

Key	Description	Possible values	Default
xopenStates	Determines if the driver returns XOPEN-compliant SQL state codes in exceptions. The default is to return SQL 99 state codes.	Boolean	false
selectMethod	Enables the application to use server cursors to process forward only, read only result sets.	String (direct, cursor)	direct
applicationIntent	Declares the application workload type when connecting to a server. Possible values are ReadOnly and ReadWrite.	String (ReadOnly, ReadWrite)	ReadWrite
workstationID	The host name of the workstation.	String <=128 char	
failoverPartner	The name of the failover server used in a database mirroring configuration.	String	null
packetSize	The network packet size used to communicate with SQL Server. Meaning of: -1 = server's default packet size. 0 = MAX (32767). Important: Using the packetSize property when the encryption is enabled (encrypt=true) isn't recommended.	Integer (-1, 0, 512..32767)	8000
encrypt	Determines if Secure Sockets Layer (SSL) encryption should be used between the client and the server. The default value is "true" in version 10.2 and later and "false" in 9.4 and earlier.	String	true, false (see desc.)
trustServerCertificate	Determines if the driver should validate the SQL Server Secure Sockets Layer (SSL) certificate.	Boolean	false
trustStoreType	Type of trust store type like JKS / PKCS12 or any FIPS Provider KeyStore implementation Type.	String	JKS
trustStore	The path to the certificate trust store file.	String	null
trustStorePassword	The password used to check the integrity of the trust store data.	String	null
hostNameInCertificate	The host name to be used when validating the SQL Server Secure Sockets Layer (SSL) certificate.	String	null
sendTimeAsDatetime	Determines whether to use the SQL Server datetime data type to send java.sql.Time values to the database.	Boolean	true
transparentNetworkIPResolution	Determines whether to use the Transparent Network IP Resolution feature. false when: Connecting to Azure SQL Database where the data source ends with: .database.chinacloudapi.cn or .database.usgovcloudapi.net or .database.cloudapi.de or .database.windows.net or .database.fabric.microsoft.com or Authentication is 'Active Directory Password' or 'Active Directory Integrated'	String (true, false, yes, or no)	true, false (see desc.)
queryTimeout	Sets the default query timeout value, set by Heimdall via the setQueryTimeout method	Integer (-1 and 0 = wait indefinitely)	-1 (infinite timeout)
socketTimeout	The number of milliseconds to wait before the java.net.SocketTimeoutException is raised.	Integer	0
serverPreparedStatementDiscardThreshold	The threshold for when to close discarded prepare statements on the server (calling a batch of sp_unprepares). A value of 1 or less will cause sp_unprepare to be called immediately on PreparedStatment close.	Integer	10
enablePrepareOnFirstPreparedStatementCall	This setting specifies whether a prepared statement is prepared (sp_prepare) on first use (property=true) or on second after first calling sp_executesql (property=false).	Boolean	-
gsscredential	Impersonated GSS Credential to access SQL Server.	org.ietf.jgss.GSSCredential	null
authentication	The authentication to use. Possible values are ActiveDirectoryIntegrated, ActiveDirectoryPassword, ActiveDirectoryManagedIdentity (version 12.2+), ActiveDirectoryMSI (version 7.2+), ActiveDirectoryInteractive (version 9.2+), ActiveDirectoryServicePrincipal (version 9.2+), SqlPassword, and the default NotSpecified.	String	NotSpecified
accessToken	The access token to use for Azure Active Directory.	String	null
keyStoreAuthentication	The name that identifies a key store.	String	null
keyStoreSecret	The authentication secret or information needed to locate the secret.	String	null

Key	Description	Possible values	Default
keyStoreLocation	The key store location.	String	null
fipsProvider	FIPS Provider. Removed in versions >= 6.4.0.	String	null
fips	Determines if enable FIPS compliant SSL connection between the client and the server.	Boolean	false
jaasConfigurationName	Login configuration file for Kerberos authentication. (Version 6.2+) Each connection to SQL Server can use its own JAAS Login Configuration name to establish a Kerberos connection.	String	SQLJDBCdriver
hdParseSP	Parse SQL Server stored procedures to build rules necessary for proper functions	Boolean	-

Connection pooling properties

This is list of properties that are used in **Connection pooling** in Data Sources.

Properties with (!) are considered important and should be set in all cases.

These options are inside given data source configuration file. For example: postgresTest-source_1.conf. The filename depends on the name of the data source → nameOfDataSource.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Description	Possible values	Default
defaultAutoCommit	The default auto-commit state of connections created by this pool. If not set, default is JDBC driver default (If not set then the setAutoCommit method will not be called.)	Boolean	- (see desc.)
defaultReadOnly	The default read-only state of connections created by this pool. If not set then the setReadOnly method will not be called. (Some drivers don't support read only mode, ex: Informix)	Boolean	- (see desc.)
defaultTransactionIsolation	The default TransactionIsolation state of connections created by this pool. One of the following: NONE,READ_COMMITTED,READ_UNCOMMITTED,REPEATABLE_READ,SERIALIZABLE. If not set, the method will not be called and it defaults to the JDBC driver.	String (see desc.)	- (see desc.)
defaultCatalog	The default catalog of connections created by this pool.	String	-
maxActive (!)	The maximum number of open connections (Integer, no default, however an attempt is made to detect the value from the server). This is equivalent to the connection limit in PGBouncer's max_client_conn option. If not set, the proxy will attempt to determine this from the database itself, but if more than proxy is active, this should be manually set to account for the maximum number of proxies, as it is applied on a per-proxy basis.	Integer	0 (no limit)
rateLimit (!)	Total connections/ms that can be opened, minimum of 1	Double	2
maxBurst (!)	Number of ms of burst that can be allowed before clamping occurs	Double	50
maxUserActive (!)	The maximum number of open connections for a particular user, can be overridden with a rule	Integer	0 (no limit)
maxUserIdle (!)	The max number of idle connections for a particular user, can be overridden with a rule	Integer	0 (no limit)
maxWait (!)	The time to wait in milliseconds before a SQLException is thrown when a connection is requested, but can't be honored due to the number of connections	Integer	30000
maxAge (!)	The chances of a connection being closed when being returned to the pool (1 out of X) or during an eviction run if idle. Old behavior was the Time in milliseconds to expire the connection on return or idle. connections less than the timeBetweenEvictionRunsMillis will not be culled at random, so if there is activity on the connection at least that often, it will remain connected	Long	30
idleSessionTimeout	Important note .Time (ms) after which the connection will be delete from idle pool during eviction run. It has increasingly higher chance of being closed starting when 75% of idleSessionTimeout time had passed	Integer	0 (disabled)
validationQuery (!)	The query to run during validation	String	select 1
resetQuery (!)	The query to run after a connection is returned to the pool	String	-
validationQueryTimeout	The timeout in seconds before a connection validation/reset queries fail. Negative integers get changed to 0 which means no timeout.	Integer	-1 (no timeout)
testOnBorrow	True if validation happens when a connection is requested	Boolean	false
testOnReturn (!)	True if validation happens when a connection is returned, implied with a reset query	Boolean	false

Key	Description	Possible values	Default
testWhileIdle	True if validation happens when a connection is not in use (idle)	Boolean	false
testOnConnect	Validate connection after connection has been established	Boolean	false
timeBetweenEvictionRunsMillis	Sleep time for background thread in between pool checks	Integer	5000
minEvictableIdleTimeMillis	Minimum amount of time a connection stays idle before it is evicted	Integer	60000
removeAbandoned	True if connection in use can be timed out	Boolean	false
removeAbandonedTimeout	Timeout in seconds for connections in use	Integer	60
logAbandoned	If true, stack trace will be recorded and printed out for timed out connection	Boolean	false
abandonWhenPercentageFull	Connections that have been abandoned aren't closed unless connections in use are above this percentage	Integer	0
validationInterval	If larger than zero than validation will only occur after the interval milliseconds has passed	Long	3000
suspectTimeout	Timeout in seconds for connection that suspected to have been abandoned	Integer	0 (not used)
commitOnReturn	If autoCommit==false then the pool can complete the transaction by calling commit on the connection as it is returned to the pool	Boolean	false
connectionProperties	The connection properties that will be set for new connections. Format of the string will be [propertyName=property;]*	String	-
initSQL	A SQL executed once per connection, when it is established	String	-
logValidationErrors	Log errors during the validation phase to the log file	Boolean	false
rollbackOnReturn	If autoCommit==false then the pool can terminate the transaction by calling rollback on the connection as it is returned to the pool	Boolean	true

Load Balancing / High Availability options:

This is list of options inside **Load Balancing / High Availability** in Data Sources.

More detailed descriptions and additional information can be found in [load balancing](#)

These options are inside given data source configuration file. For example: postgresTest-source_1.conf. The filename depends on the name of the data source → nameOfDataSource.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Requirements	Conf File field	Description	Possible values	Default
Enable load balancing	- (Unlocks this section)	useLoadBalancing	Option must be selected for Load Balancing to work. The jdbc URL specified in the top section of the configuration will be ignored, although used as the initially configured server in the load balancing configuration.	Boolean	false
Track cluster changes	MySQL, MS-SQL, and Postgres	trackClusterChanges	Track the cluster state and adjust the LB pool accordingly	Boolean	false
Target write capacity	track cluster changes	desiredWriteCapacity	Desired write capacity for writable nodes	Integer	-
Target read capacity	track cluster changes	desiredReadCapacity	Desired read capacity for writable nodes	Integer	-
Use response metric	-	useResponseMetrics	Use response time metrics for load balancing decisions to find nearest readers	Boolean	false
Response Metric Multiplier	Use response metric	responseMetricMultiplier	How much slower a reader can be vs. the fastest reader and still be eligible for reads	Float	2
Track Replication Lags	-	trackReplicationLags	Estimate the replication lag between nodes. Used to improve read/write split behavior. Do not use with forced replication delays.	Boolean	false
Alert threshold (ms)	Track Replication Lags	alertThreshold	If average lag detected is greater than this value an alert will be generated. There may be a greater-than-usual lag detected at the proxy start. Set to 0 to disable.	Long	0 (disabled)
Failover Script	-	failoverScript	Script to be run when failure is detected, place in heimdall install directory	String	-
Name	-	servers.name	Name of the server	String	-
Url	-	servers.url	JDBC Url to connect to the server	String	-
Enabled	-	enabled	Disables/Enables the given server - if disabled while active, connections will be migrated to a new server if available	Boolean	true
Writeable	-	writeable	If unchecked (false), only use if the reader eligible rule matches for a query	Boolean	false
(active)	inside conf. file only	active	Status of the server	Boolean	-
(failed)	inside conf. file only	failed	Status of the server	Boolean	-

Group mapping options:

This is list of options inside **group mapping** inside Data Sources.

These options are inside given data source configuration file. For example: postgresTest-source_1.conf. The filename depends on the name of the data source → nameOfDataSource.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Requirements	Conf File field	Description	Possible values	Default
LDAP Configuration	-	ldapConfigData.name	LDAP configuration that is used for this data source	String	-
key (Group Membership)	Use regex = true	ldapMappings.mappingPropertiesKey	The list of groups configured in LDAP.	String	-
Use Regex (Group Membership)	-	ldapMappings.useRegex	Heimdall supports the use of capture groups within regex patterns. This enables the extraction of specific substrings from group names for role mapping.	Boolean	false
Role Name (Local Database Role)	Use Custom Role = false	ldapMappings.mappingPropertiesValues.roleName	A local database role to select from the list.	String	-
Custom Role Name (Local Database Role)	Use Custom Role = true	ldapMappings.mappingPropertiesValues.roleName	A custom local database role for text.	String	-
Use Custom Role (Local Database Role)	-	ldapMappings.mappingPropertiesValues.useCustomRole	Allows use of the custom role name.	Boolean	false

Roles Management options:

This is list of options inside **Roles Management** in Data Sources.

These options are inside given data source configuration file. For example: postgresTest-source_1.conf. The filename depends on the name of the data source → nameOfDataSource.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Requirements	Conf File field	Description	Possible values	Default
Database-Native Session Cleanup	-	databaseNativeSessionCleanup	By default, the Central Manager (CM) is responsible for clearing ephemeral user sessions. More information can be found in roles management	Boolean	-
Require Justification Pattern	-	justification.isRequired	Allows to make all session requests justification for current data source require to match given pattern. Toggling this option will override any global setting under Admin/Portal. Leaving 'regex pattern' empty can be used to disable the global setting. Invalid justification message should be provided to inform the requesting user how should justification look like.	Boolean	false
Regex Pattern	Require Justification Pattern	pattern	Provide a regex pattern that justification has to match. Ignored if empty.	String	-
Invalid justification message	Require Justification Pattern	errorMessage	Error message explaining what pattern should the justification match.	String	-
Filter system roles	-	filterDefinedRoles	If selected, predefined system roles for databases will not be displayed on the roles names list.	Boolean	false

Next part of this section is used to manage roles in the database. In the configuration for a single entry with a role, you can specify a notification alias associated with an email group. This group will be informed when, for example, a session is requested for a particular role. Additionally, there are other options like the one to set the number of approvals needed to approve a specific role.

Key	Requirements	Conf File field	Description	Possible values	Default
Role Name	-	roleName	Used to select a role from the database for which we want to create a configuration.	String	-
Notification Alias	-	notificationAlias	Here, you indicate the notification entry created in the Admin tab , which will be used to notify users associated with it when an action concerns the specified role. After clicking on the envelope icon next to it, you can view the list of emails associated with the specified notification.	String	-
Default Time	-	defaultTime	Specifies the default session duration for the given role.	Integer	30
Maximum Time	-	maximumTime	Determines the maximum time for which a particular role can be requested. Must be greater than default time.	Integer (see desc.)	60
Approvals Needed	-	approvalsNeeded	Used to specify the number of approvals required for the requested role to be approved. The default value is 1. You cannot set a value higher than the number of	Integer	

Key	Requirements	Conf File field	Description	Possible values	Default
			emails associated with the specified notification alias. If no emails are associated with the notification, the value is set to 0, and it cannot be changed.		0 or 1 (see desc.)
Special Roles	(Postgres only)	specialRoles	Check to allow for special postgres roles present in this role such as 'rolsuper' or 'rolcreatorole' should be inherited.	Boolean	false

Driver options:

This is list of options regarding **Drivers**, these options can be set from Configure → Drivers.

These options are saved inside drivers configuration file. The filename depends on the name of the driver → nameOfDriver.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Conf File field	Description	Possible values
Version	driverVersion	The Version ID of the driver, for reference only	String
Example Url	exampleUrl	Used as a reference, not required	String
Writer Url	writerUrl	Used in handling AWS RDS cluster tracking (see aws cluster tracking)	String
Reader Url	readerUrl	Used in handling AWS RDS cluster tracking (see aws cluster tracking)	String
Webstite Url	websiteUrl	Used as a reference, not required	String
Notes	notes	Optional text field for notes on the driver	String
JDBC Class	jdbcClass	Required: provided by jdbc driver vendor	String
Data Source Class	dataSourceClass	Optional: used in some situations, provided by JDBC driver vendor	String
XA Data Source Class	xaDataSourceClass	Optional: used in some situations when using XA transactions, provided by JDBC driver vendor	String
Maven ID	mavenId	Optional field. Used to provide automatic download of jdbc jar file from maven central. Text should match a pattern: "groupId:artifactId:version" e.g. for mysql: mysql:mysql-connector-java:5.1.49. Jar file will be downloaded on driver creation and during update if not exists.	String
Driver Upload	uploadedDrivers	Upload your driver file(s) here. If a driver happens to be incompatible with our remote download scheme, this driver file can be left blank, as long as the driver is available on the server using the Heimdall Driver itself. This is a known issue with the AWS Redshift driver.	String
Auto Update	autoUpdate	Automatic driver option update when Heimdall is updated	String
Enable	enabled	Used to control if driver is enabled or disabled for use	String
-	file	Name of the file that stored driver information	String
Name	name	Name of the driver, used to identify it.	String

Reader and Writer variables

Please read [AWS Cluster Tracking](#) for more detailed information.

Key is the variable to be used in reader and write URL's for substitution, which should be used based on the information extracted from the AWS configuration

Key	Description
\$(listener)	For SQL Server, this will represent the listener URL, which points to the IP that tracks between the primary writer node during a failover. Empty for non-SQL Server cluster types

Key	Description
\$(writeEndpoint)	For Aurora clusters, this will point to the primary endpoint hostname, which uses DNS to fail between primary nodes.
\$(readerEndpoint)	For Aurora clusters, this will point to the reader endpoint hostname, which uses DNS over time to spray load across reader nodes. For a single node reader setup, this is appropriate, but in multi-node reader configurations, this can result in uneven distribution.
\$(writer)	For any cluster type, this will point to the actual active writer. If more than one writer is detected, then multiple nodes will be created, one with each writer.
\$(writers)	Like with writer, but will fill in all writer hostnames in a comma separated list. Useful for URL types that allow multiple hosts to be included in a single URL.
\$(reader)	Generates one node per reader, including a node for the writer, for the purpose of reading. The read weight of readers will be set to 10, but the read weight of the writer will be set to 1.
\$(readers)	Generate one reader node, with all readers in a comma seperated list. The writer(s) will be first in the list.
\$(replica)	Include all the readers as distinct servers, not including writer nodes
\$(replicas)	Include all the readers in one URL, not including writer nodes

Manager options:

This is list of options regarding **Admin** section, these options can be set from admin tab.

These options are saved inside heimdall.conf file.

⚠ **Note:** You can use SHIFT with mouse scroll to scroll horizontally!

Small Sections

Sections that are small are in this general section with their Section specified.

Key	Section	Requirements	Conf File field	Description	Possible values	Default
Registered ID	Account Information	-	accountInformation.registeredID	Provided from Customer Support to verify enrollment	String	-
Send alerts through notification	Alerts	-	alertsConfig.sendAlertsViaNotification	If this option is checked, every alert which message doesn't match any added pattern, will be sent through notification.	Boolean	false
Exclude Proxy Logins	Login History	-	- (Non-persistent / GUI only)	Unchecking 'Exclude proxy logins' will result with adding proxy logins to the list.	Boolean	true
Security Tag	Security Tags	-	(Elements of availableSecurityTags)	This section allows for the management of security tags, which can be applicable in the database browser section.	String	-
Auth Provider	SAML Configuration	-	samlAuthProvider	Identity provider (AWS IAM Identity Center, Okta, Other)	String (AWS IAM (...), Okta, Other)	-
IdP Metadata URL	SAML Configuration	-	metadataUrl	The URL pointing to the Identity Provider's metadata.	String	-
AWS Identity Store ID	SAML Configuration	Auth Provider = AWS IAM Identity Center	identityStoreId	Since AWS provides groups as UUIDs, this setting enables mapping those UUIDs to their display names by connecting to the appropriate Identity Store. The mappings are stored in a temporary internal cache for up to 1 hours. If a group is renamed in AWS, click the Commit button to refresh the configuration and clear the cache. If you leave this field blank, group UUIDs will be shown instead of names. Users can manually map UUIDs to the corresponding names in the Datasource tab using the Group Mapping feature for proper role-base session requesting in the portal. For more details, see the Group Mapping section .	String	-
Name	Password Policy	-	passwordPolicies.name	Name of the policy that will be enforced.	String	-

Key	Section	Requirements	Conf File field	Description	Possible values	Default
Value	Password Policy	Name = (maximum/minimum) of characters OR Name = custom regex matching	passwordPolicies.params.value	Used by some policies to set minimum or maximum characters or custom regex.	String	-
(type)	Password Policy	Name = (maximum/minimum) of characters OR Name = custom regex matching	passwordPolicies.params.type	Filled by heimdall. Used by some policies. Type of the value used by the policy.	String	-
(name)	Password Policy	Name = (maximum/minimum) of characters OR Name = custom regex matching	passwordPolicies.params.name	Filled by heimdall. Short name of the policy that determines what type of policy it is.	String	-

Heimdall Environmental options

This list is also a set of options regarding **Admin** section. They are set in **Config Management** subsection, but they are stored in /etc/heimdall.conf, while rest of the admin section configuration is stored under /opt/heimdall, or custom directory specified in installation process.

Key	Requirements	Conf File field	Description	Possible values	Default
hdRole	-	hdRole	It can be used to control if this environment is a central manager or a proxy	String	-
hdHost	-	hdHost	Hostname of management server	String	heimdallmanager
hdPort	-	hdPort	Port of the management server, generally 8087 or 8443	Integer	8087
hdUser	-	hdUser	Login username for the management server, can be admin	String	admin
hdPassword	Secret (Config Management) = false	hdPassword	Login password for the management server	String	heimdall (on premise)
Secret (hdPassword)	-	useSecretForVdbCredentials	Determines if the secret will be used	Boolean	false
Secrets Manager (hdPassword)	Secret (Config Management) = true	secretsManagerConfigName	Secrets Manager Configuration that will be used. Depending on secrets manager used it will write needed fields into the file.	String	-
vdbCredentialsSecretName	Secret (Config Management) = true	vdbCredentialsSecretName	Secret name	String	-
hdSecretKey*	-	hdSecretKey	In AWS, use this as the name of an AWS Secret to store the configuration, protecting included passwords from being written to disk.	String	-
cloudDetection	-	cloudDetection	If the manager should detect cloud services on startup	Boolean	true
cloudOption	cloudDetection = false	cloudOption	For heimdall running on premise to allow using cloud services, initializes on startup. One of: none, aws, azure, gcp, oracle	String	-
javaOptions	-	javaOptions	Any arbitrary options desired to be set	String	-

hdSecretKey *note: This option is supported only by the default Secrets Manager from the same Region/Account as the instance. To use, proper permissions must be set on the IAM role. This option provides two major benefits. First is that all passwords are stored in AWS Secrets, in an encrypted format. Second is that redeployment of a management server can be done with a configuration pre-populated, so there is no need to back up and restore configurations to account for failures. Simply terminate the old instance and a new instance with the same user-data will be created with the same configuration as the original.

Log Management Options

Key	Section	Requirements	Conf File field	Description	Possible values	Default
Use external database for logging	Log Database Management	-	logDatabaseConfiguration.useExternalSource	Allows setting external database for logging.	Boolean	false
Data Source	Log Database Management	Use external database for logging	logDatabaseConfiguration.sourceName	Data source to set for external logging.	String	(Using: Em database (
Send login entries to CloudTrail	Cloud Logs	-	cloudTrailUserLoginConfig.sendEntriesToCloudTrailLake	Send login events to AWS CloudTrail. Use the Channel ARN, not the Event data store ARN. Proper IAM role has to be configured.	Boolean	false

Key	Section	Requirements	Conf File field	Description	Possible values	Default
Cloud Trail Lake Channel ARN	Cloud Logs	Send login entries to CloudTrail	cloudTrailUserLoginConfig.cloudTrailChannelArn	CloudTrail Lake Channel ARN, in the format: <code>arn:aws:cloudtrail:<region>:<account-id>:channel/<channel-id></code> . Can be found on AWS with path CloudTrail → Lake → Integrations	String	-
Enable Manager CloudWatch Logging	Cloud Logs	-	enableManagerCloudWatchLogging	Allows us to configure whether manager logs should be sent to AWS CloudWatch. (!) May incur additional AWS charges.	Boolean	false
CloudWatch namespace	Cloud Logs	Enable Manager CloudWatch Logging	serverProperties.managerCloudWatchLoggingNamespace	CloudWatch namespace used for Manager CloudWatch Logging	String	HEIMDALL
S3 Bucket Name	Cloud Logs	-	s3UploadConfiguration.bucketName	You can specify the S3 Bucket Name where the logs should be saved. This enables centralized and durable storage of manager logs in AWS S3.	String	upload.heimdall
Force Upload to S3 On Logs Rolling	Cloud Logs	-	s3UploadConfiguration.forceOnLogsRotationUpload	This option allows to force upload to S3 on logs rolling. Enabling this option will make log rotation take significantly more time.	Boolean	false
Log Events To Console	Log Server Properties	-	serverProperties.logToConsole	For debugging or container use, log ALL events to the stdout console of management server	Boolean	false
Max Log Age	Log Server Properties	-	serverProperties.maxLogAge	Sets the maximum age in days of log files and log records	Integer	14
Reserved Disk Space	Log Server Properties	-	serverProperties.reservedDiskSpace	Specifies the percentage of disk space that should remain free on the log filesystem.	Double (0-1)	0.1
Log Rotation Interval	Log Server Properties	-	serverProperties.logRotationInterval	A period between log rotation (in minutes), requires manager restart to take effect	Positive Integer	

Server Properties

Key	Section	Conf File field	Description	Possible values	Default
Enable Portal Mode	Server Properties	serverProperties.enablePortalMode	Value that indicates if after successful login central manager or portal mode will be used	Boolean	false
Disable Cert Validation	Server Properties	serverproperties.disableTLSCertValidation	Disable TLS certificate validation for software downloads	Boolean	false
Password Validation	Server Properties	serverproperties.passwordPolicy	Validate new users password along with provided rules	Boolean	false
Redirect Config Fetches	Server Properties	serverproperties.redirectConfigEndpoints	Value that indicates if server should redirect all HTTP config requests to HTTPS Tomcat port	Boolean	false
Verbose Debug Mode	Server Properties	serverproperties.verboseDebugMode	On the server, enable verbose debug mode	Boolean	false
Minimum free disk space %	Server Properties	serverproperties.freeDiskSpacePercentage	Configures how much space needs to be free for any configuration saves to occur. Defaults to 1 for 1%. This helps prevent configuration loss if not enough space is available to successfully save the full configuration.	Double	1.0
Max Config Backups	Server Properties	serverproperties.maxConfigBackups	Value that indicates how many server configuration backups to keep. Should be greater than 1	Integer	10
DNS Port	Server Properties	serverproperties.dnsPort	Value that indicates what port should heimdall manager listen on for DNS queries for proxy auto-scaling. The queries should ask for a domain that starts with VDB name being used. For more information see Advanced Features .	Integer	-

Key	Section	Conf File field	Description	Possible values	Default
Session Timeout	Server Properties	serverproperties.sessionTimeout	Controls HttpSession idle-timeout expiration for GUI users. Must be between 5 minutes and 24 hours. It's saved as minutes in the configuration file	Integer (5 - 1440)	30
Enable Billing Reporting	Server Properties	serverproperties.enableBillingReporting	Value that indicates if the billing reports should be sent to the HeimdallBilling service	Boolean	true
Proxy Host	HTTP Proxy Settings	serverproperties.proxyHost	The hostname or IP address of the proxy server to use when accessing the internet	String	-
Proxy Port	HTTP Proxy Settings	serverproperties.proxyPort	The port number on which the proxy server is listening.	Integer	3128
Proxy User	HTTP Proxy Settings	serverproperties.proxyUser	Proxy user for proxy authentication	String	-
Proxy Password	HTTP Proxy Settings	serverproperties.proxyPassword	Proxy password for proxy authentication	String	-

Secrets Manager Configuration

AWS Secrets Manager

Key	Requirements	Conf File field	Description	Possible values	Default
Endpoint	Secrets Manager = AWS Secrets Manager	secretsManagerConfigs.endpoint	(optional) Alternate endpoint for Secrets Manager service.	String	-
Region	Secrets Manager = AWS Secrets Manager	secretsManagerConfigs.region	(optional) Override the region which should be used (ex. us-east-1).	String	-
Access Key	Secrets Manager = AWS Secrets Manager	secretsManagerConfigs.accessKey	(optional) Override the access key used to authorize requests to the Secrets Manager.	String	-
Secret Key	Secrets Manager = AWS Secrets Manager	secretsManagerConfigs.secretKey	(optional) Override the secret key used to authorize requests to the Secrets Manager	String	-
Use STS Role	Secrets Manager = AWS Secrets Manager	secretsManagerConfigs.useSTSRole	(optional) If checked, use STS to assume a role with the specified ARN.	Boolean	false
STS Role ARN	Use STS Role	secretsManagerConfigs.stsRoleArn	(required, when using STS Role) ARN of the role to assume when using STS.	String	-
STS External ID	Use STS Role	secretsManagerConfigs.stsExternalId	(optional) External ID to uniquely associate with the role.	String	-
STS Session Name	Use STS Role	secretsManagerConfigs.stsSessionName	(optional) Name to use for the STS session. Defaults to heimdall-STS-session string.	String	-

CyberArk Conjur

Key	Requirements	Conf File field	Description	Possible values	Default
Appliance URL	Secrets Manager = CyberArk Conjur	secretsManagerConfigs.applianceUrl	The URL of the Conjur instance you are connecting to. When connecting to Conjur Enterprise configured for high availability, this should be the URL of the master load balancer (if performing read and write operations) or the URL of a follower load balancer (if	String	-

Key	Requirements	Conf File field	Description	Possible values	Default
			performing read-only operations).		
Account	Secrets Manager = CyberArk Conjur	secretsManagerConfigs.account	Conjur account that you are connecting to. This value is set during Conjur deployment.	String	-
Authn Login	Secrets Manager = CyberArk Conjur	secretsManagerConfigs.authnLogin	User/host identity.	String	-
Authn API Key	Secrets Manager = CyberArk Conjur	secretsManagerConfigs.authnApiKey	User/host API key (or password). Write-only field! Can be edited, but it cannot be viewed.	String	-
Secret (Authn API Key)	Secrets Manager = CyberArk Conjur	secretsManagerConfigs.useSecretForAuthnApiKey	Determines if the secret will be used	Boolean	false
AWS Secret Name (Authn API Key)	Secrets Manager = CyberArk Conjur AND Secret (Authn API Key)	secretsManagerConfigs.authnApiKeySecretName	Name used to retrieve the Auth API Key from the AWS Secrets Manager	String	-
Authn URL	Secrets Manager = CyberArk Conjur	secretsManagerConfigs.authnUrl	(optional) Alternate authentication endpoint. By default, the client uses the standard <code><applianceUrl>/authn</code> for generic username and API key login flow.	String	(applianceUrl/ authn)

HashiCorp Vault

Key	Requirements	Conf File field	Description	Possible values	Default
Vault URL	Secrets Manager = Hashicorp Vault	secretsManagerConfigs.url	The URL of the Hashicorp Vault instance.	String	-
Ignore certificate verification	Vault URL needs to contain "https"	secretsManagerConfigs.ignoreCertificateVerification	Allow HTTPS communication with Vault server without verifying certificate. By default it is disabled, it is not recommended to use this in production.	Boolean	false
Auth method	Secrets Manager = Hashicorp Vault	secretsManagerConfigs.authConfig.authMethod	Allows to select what authentication method should be used by Heimdall to connect to Hashicorp Vault instance. Currently supported methods are <i>Token</i> , <i>Username & Password</i> and <i>AppRole</i> . Selecting one of these unlocks additional	String (see desc.)	-

Key	Requirements	Conf File field	Description	Possible values	Default
			configuration options.		
Auth mount path	Auth Method != Token	secretsManagerConfigs.authConfig.authPath	(Optional) Alternate authentication method mount path to be used. Default value depends on Auth method.	String	userpass or approle
Token	Auth Method = Token	secretsManagerConfigs.authConfig.token	Token used for authentication to Vault server.	String	-
Secret (Token)	Auth Method = Token	secretsManagerConfigs.authConfig.useSecretForToken	Determines if secret will be used for token.	Boolean	false
AWS Secret Name (Token)	Secret (Token)	secretsManagerConfigs.authConfig.tokenSecretName	Aws secret name used to retrieve the	String	-
Username	Auth Method = Username & Password	secretsManagerConfigs.authConfig.username	Username used for authentication to Vault server.	String	-
Password	Auth Method = Username & Password AND Secret (Username) = false	secretsManagerConfigs.authConfig.password	Password used for authentication to Vault server.	String	-
Secret (Username and Password)	Auth Method = Username & Password	secretsManagerConfigs.authConfig.useSecretForCredentials	Determines if secret will be used for Username.	Boolean	false
AWS Secret Name (Username and Password)	Secret (Username)	secretsManagerConfigs.authConfig.credentialsSecretName	Name used to retrieve the credentials from the AWS Secrets Manager	String	-
Role ID	Auth Method = AppRole	secretsManagerConfigs.authConfig.roleId	The semi-secret identifier for the role that will authenticate to Vault. Think of this as the username portion of an authentication pair.	String	-
Secret ID	Auth Method = AppRole AND Secret (Username) = false	secretsManagerConfigs.authConfig.secretId	The secret identifier for the role that will authenticate to Vault. Think of this as the password portion of an authentication pair.	String	-
Secret (Role and Secret ID)	Auth Method = AppRole	secretsManagerConfigs.authConfig.useSecretForAppRole	Determines if secret will be used for Role ID.	Boolean	false
AWS Secret Name (Role and Secret ID)	Secret (Role ID)	secretsManagerConfigs.authConfig.	Name used to retrieve Role ID and Secret ID from the AWS	String	-

Key	Requirements	Conf File field	Description	Possible values	Default
			Secrets Manager		

General Secrets Manager Options

Key	Requirements	Conf File field	Description	Possible values	Default
Enable	-	secretsManagerConfigs.enabled	Is the current secrets configuration enabled.	Boolean	true
Secrets Manager	-	secretsManagerConfigs.secretsManagerConfigtype	Secrets Manager to be used: AWS Secrets Manager, CyberArk Conjur, Hashicorp Vault	String	-
Secret (Secrets Manager)	Secrets Manager != AWS Secrets Manager	secretsManagerConfigs.useSecretForWholeConfig	Determines if the secret will be used for whole configuration	Boolean	false
AWS Secret Name (Secrets Manager)	Secret (Secrets Manager) = true	secretsManagerConfigs.useSecretForWholeConfig	Name used to retrieve the whole config from the AWS Secrets Manager	String	-
(name)	-	secretsManagerConfigs.name	Name of the secret manager configuration.	String	-
(type)	-	secretsManagerConfigs.type	Type of the secrets manager used. Used only in file, without representing field in GUI. Values can be: HASHICORP_VAULT, CYBERARK_CONJUR, AWS	String (see desc.)	(Depends on secrets manager used)

SMTP Configuration

Key	Requirements	Conf File field	Description	Possible values	Default
Sender User/Email	-	smtpConfiguration.senderAddress	The email address used as the sender	String	false
Sender Password	Secret (SMTP) = false	smtpConfiguration.senderPassword	Authentication credentials for the sender email	String	false
Secret (Sender Password)	-	smtpConfiguration.useSecrets	Determines if secret will be used for SMTP's sender password.	Boolean	false
Secrets Manager (Sender Password)	Secret (SMTP) = true	smtpConfiguration.secretsManagerConfigName	Specify the Secrets Manager Configuration that will be used for this secret. List values are secrets managers configured in heimdall.	String	-
Secret Name (Sender Password)	Secret (SMTP) = true	smtpConfiguration.secretName	Specify a Secret name	String	-
Host	-	smtpConfiguration.smtpHost	Hostname of the SMTP server	String	smtp.gmail.com
Port	-	smtpConfiguration.smtpPort	Port used to connect to the SMPT server	Integer	587
Smtp Auth	-	smtpConfiguration.smtpAuth	Indicates if SMTP authentication will be used	Boolean	true
START_TLS Enabled	-	smtpConfiguration.startTLSEnabled		Boolean	true

Key	Requirements	Conf File field	Description	Possible values	Default
			Requests the SMTP server to upgrade the connection to TLS encryption		
SMTP Properties	-	smtpConfiguration.properties	Allows specifying additional SMTP configuration options. Case sensitive. Available properties can be found here: https://javaee.github.io/javamail/docs/api/com/sun/mail/smtp/package-summary.html	List	-

AWS Alternate endpoints

Key	Conf File field	Description	Possible values	Default
CloudWatch Endpoint	awsEndpoints.cloudWatchEndpoint	Alternate endpoint for CloudWatch	String	-
CloudWatch Logs Endpoint	awsEndpoints.cloudWatchLogsEndpoint	Alternate endpoint for CloudWatch Logs	String	-
ElasticCache Endpoint	awsEndpoints.elasticCacheEndpoint	Alternate endpoint for ElasticCache	String	-
RDS Endpoint	awsEndpoints.rdsEndpoint	Alternate endpoint for RDS	String	-
SNS Endpoint	awsEndpoints.snsEndpoint	Alternate endpoint for SNS	String	-
CloudTrail Endpoint	awsEndpoints.cloudTrailEndpoint	Alternate endpoint for CloudTrail	String	-
Identity Store Endpoint	awsEndpoints.identityStoreEndpoint	Alternate endpoint for Identity Store	String	-

Kerberos options:

This is list of options regarding **Admin** section, these options can be set from admin tab in **Kerberos Configuration** subsection.

These options are saved inside KERBEROS_CONF.conf file.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Requirements	Conf File field	Description	Possible values	Default
Enable Kerberos Authentication	-	kerberosEnabled	Enable Kerberos Authentication for Heimdall Manager	Boolean	false
Service Principal	Enable Kerberos Authentication	servicePrincipal	The unique identity of a service (manager) in the Kerberos system.	String	HTTP/ localhost
Keytab Location	Enable Kerberos Authentication	keytabLocation	Location to keytab file that stores secret keys for service principal (manager).	String	/etc/ krb5.keytab
Enable LDAP Groups Extraction	Enable Kerberos Authentication	ldapGroupsExtractionEnabled	Perform LDAP groups extraction on successful authentication	Boolean	false
LDAP Configuration	Enable Kerberos Authentication	ldapConfigData	Holds data of LDAP Configuration for groups extraction	List	-

LDAP options:

This is list of options regarding **Admin** section, these options can be set from admin tab in **LDAP Configuration** subsection.

These options are saved inside ldap configuration file. The filename depends on the name of the configuration → nameOfConfiguration.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Required	Requirements	Conf File field	Description	Example value
LDAP(S) URL	yes	-	ldapUrl	Specifies URL of LDAP server. Instead of just one URL, you can also supply a space-separated list of URLs. In this case, the LDAP provider will attempt to use each URL in turn until it is able to create a successful connection.	ldaps://server.example.com
Server Type	yes	-	ldapServerType	LDAP Server Type. Types are: Active Directory, Redhat IDM (FreeIPA), OpenLDAP, JumpCloud, Okta, Other.	RedHat IDM (FreeIPA)
Simple LDAP Mode	no	-	ldapSimpleMode	If Simple LDAP Mode should be used, otherwise use Search+Bind mode	-
LDAP Prefix	yes (simple mode)	Simple LDAP Mode = true	ldapPrefix	LDAP Prefix used in Simple LDAP Mode	uid=
LDAP Suffix	yes (simple mode)	Simple LDAP Mode = true	ldapSuffix	LDAP Suffix used in Simple LDAP Mode	,cn=users,cn=accounts,dc=lab,dc=helmin
LDAP Security Principal	yes	Simple LDAP Mode = false	ldapSecurityPrincipal	Specifies name of user used to search for authenticated user	uid=admin,cn=users,cn=accounts,dc=lab,dc=
Secret (LDAP Security Principal)	no	Simple LDAP Mode = false	ldapSecurityPrincipalUseSecret	Determines if secret will be used for LDAP Security Principal	-
Secrets Manager (LDAP Security Principal)	no	Simple LDAP Mode = false AND Secret (LDAP Security Principal) = true	ldapSecurityPrincipalSecretsManagerConfigName	Specify the Secrets Manager Configuration that will be used for this secret. List values are secrets managers	-

Key	Required	Requirements	Conf File field	Description	Example value
				configured in heimdall.	
LDAP Security Principal Secret Name	yes (secret)	Simple LDAP Mode = false	ldapSecurityPrincipalSecretName	Specify a Secret name	-
LDAP Search User Password	yes	Simple LDAP Mode = false AND Secret (LDAP Security Principal) = false	ldapSearchPassword	Specifies password for LDAP Security Principal user used to search for authenticated user.	adminpassword123
LDAP Sec. Security Principal	no	Simple LDAP Mode = false	ldapSecondarySecurityPrincipal	Specifies name of user used to search for authenticated user in case the primary search with LDAP Security Principal fails.	id=second- admin,cn=users,cn=accounts,dc=lab,dc=heimdall
Secret (LDAP Sec.Security Principal)	no	Simple LDAP Mode = false	ldapSecondarySecurityPrincipalUseSecret	Determines if secret will be used for LDAP Sec. Security Principal	-
Secrets Manager (LDAP Sec. Security Principal)	no	Simple LDAP Mode = false AND Secret (LDAP Sec.Security Principal) = true	ldapSecondarySecurityPrincipalSecretsManagerConfigName	Specify the Secrets Manager Configuration that will be used for this secret.	-
LDAP Sec. Security Principal Secret Name	yes (secret)	Simple LDAP Mode = false	ldapSecondarySecurityPrincipalSecretName	Specify a Secret name	-
LDAP Sec.Search User Password	no	Simple LDAP Mode = false	ldapSecondarySearchPassword	Specifies password for LDAP Sec. Security Principal.	secondadminpassword123
LDAP Search Domain	yes	Simple LDAP Mode = false	ldapSearchDomain	Specifies LDAP search domain, the domain in which authenticated user's groups will be searched.	dc=lab,dc=heimdall,data,dc=heimdall
LDAP User Search Base	yes	Simple LDAP Mode = false	ldapSearchBase	Specified LDAP user search base in which authenticated user will be searched.	cn=users,cn=accounts,dc=lab,dc=heimdall
LDAP Name Attribute	yes	Simple LDAP Mode = false	ldapSearchAttribute	Specifies name attribute by which authenticated user	uid
LDAP Group Name Attribute	no	Simple LDAP Mode = false	ldapGroupSearchAttribute	Optional, specifies group's name attribute which should be read during extracting authenticated user's groups. If not provided, LDAP Name Attribute will be used instead.	cn
LDAP Group Filter	no	Simple LDAP Mode = false	ldapGroupFilter	Optional, option used during searching for authenticated user. Setting this	(cn=*)

Key	Required	Requirements	Conf File field	Description	Example value
				option limits the number of groups to search user into them, only to particular group inside server. Setting this option makes it required to extract at least one group to authenticate the user.	
Use Raw Group Filter	-	Simple LDAP Mode = false	IdapUseRawGroupFilter	Uses the LDAP Group Filter value as provided. No default group filters (e.g., objectCategory/ objectClass), no security-group constraints (e.g., sAMAccountType), and no membership clauses (e.g., member/ memberUid or nested membership) will be appended. You must provide a complete LDAP group filter, using \${user}, \${dn} .	false
Use nested groups filter	-	Simple LDAP Mode = false AND Server Type = Active Directory / Redhat IDM (FreeIPA)	IdapSearchNestedGroups	Specifies if parent groups should be included, when searching for user's groups.	-
Ignore LDAP cert	-	Simple LDAP Mode = false	IdapCaOverride	Value assigned to this keyword can define if TLS validation of LDAP server certificate should be performed.	-
LDAP Healthcheck	no	Simple LDAP Mode = false	IdapHealthcheck	Performs LDAP Healthcheck automatically every minute. An appropriate alert will be shown if the server goes down (Similarly, once the server is back online). Moreover, the account expiration date of the LDAP Security Principal will be checked once every 24 hours.	-
Cache groups' emails	no	Simple LDAP Mode = false	IdapGroupsEmailsCaching.cacheEnabled	Specifies whether emails for LDAP groups should be cached (applies only if LDAP configuration is linked to the portal configuration).	-
Cache Time	-		IdapGroupsEmailsCaching.cacheTime		-

Key	Required	Requirements	Conf File field	Description	Example value
		Cache groups' emails		Specifies the interval at which emails in the cache should be updated. On GUI you can select the unit. The value stored in configuration will be in minutes.	
Management privileges	no	Simple LDAP Mode = false	IdapManagementPrivileges	Determines whether it entitles a user to use admin/read only privileges by requiring membership in a specific group.	-
Admin privilege group	-	Simple LDAP Mode = false AND Management privileges = true	IdapAdminPrivilegeGroup	To be authorized to use the Management Privilege: Admin, the user must be a member of the selected group	-
Read Only privilege group	-	Simple LDAP Mode = false AND Management privileges = true	IdapReadOnlyPrivilegeGroup	To be authorized to use the Management Privilege: Read Only, the user must be a member of the selected group	-
(enabled)	-	-	enabled	Specifies if this Idap config is enabled	-
(filename)	-	-	file	The name of the file that holds Idap config	-
Configuration Name	yes	- (On GUI requires clicking the Rename button for field to appear)	name	Specifies name of the current configuration. Value of this field can be seen just above and it needs to be unique to differentiate configuration. It can be used for switching and choosing between multiple configuration options. To rename it you need to click button to the right of „Create New,“ and choose rename	HEIMDALL_LDAP_CONFIGUR

Users options:

This is list of options regarding **Users** section, these options can be set from manage → users.

These options are saved inside given user configuration file. The filename depends on the name of the users → nameOfUser.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Requirements	Conf File field	Description	Possible values	Default
Enable	-	enabled	If the given user is enabled and will be allowed to log in		
Username	-	name	Unique username that is identifying given user	String	-
Password	-	password	The password of the user–please avoid using “.” as it may impact authentication. It is stored as hash for safety	String	-
Secret	Authentication Mode = Local/ Secret	useSecret	Use a secrets manager	Boolean	false
Secret Name	Authentication Mode = Local/ Secret AND Secret	secretName	Secret to be used by secrets manager. For more information read Secrets	String	-
Email	-	email	The email associated with given user, used for receiving Portal notifications about sessions.	String	-
Management Privilege	-	(isAdmin, isReadOnly)	Defines what set of permissions is assigned to the user. One of: none - users won't see any configurations, usually used with Portal user or Audit user option, admin (see below), read only (see below).	None / Admin / Read Only	Admin
(Management Privilege)	-	isAdmin	Admin - Users will have admin rights, allowing them to perform tasks such as configuring settings, managing users, and accessing all resources.	Boolean	true
(Management Privilege)	-	isReadOnly	Read only - Users will be unable to make any changes to the configuration, but they will retain access to resources based on their filter settings. Additionally, it will block access to management tabs, such as Users, Admin, Certificates, and various options like the 'Test Connection' button in the Data Sources tab.	Boolean	false
Audit User	-	isAuditUser	If enabled, users will have access to the Audit tab, where they can view and save records from the Audit Trail table, which logs all session operations within the portal.	Boolean	false
Portal User	-	isPortalUser	If enabled, users will have access to the Portal.	Boolean	false
Authentication Mode	-	(isLdapEnabled, isKerberosEnabled, isSamlEnabled, isJwtEnabled)	Method to perform authentication by. One of: Ldap, Kerberos, Saml, (see below). Local/Secret: user will be authenticated by username & password. Note: The username and password can be stored in the Secrets Manager . This configuration currently does not support rotating credentials, that means that once we save a user, and values stored in Secrets Manager change, the heimdall user will still be using old values.	Local/Secret / LDAP / SAML / Kerberos	Local/ Secret
(Authentication Mode)	-	isLdapEnabled	User will be authenticated by LDAP server configured in Admin tab.	Boolean	false
LDAP Configuration	Authentication Mode = LDAP	ldapConfigData	LDAP configuration to be used for the given user	(Ldap Configuration)	-

Key	Requirements	Conf File field	Description	Possible values	Default
(Authentication Mode)	-	isKerberosEnabled	User will be authenticated by KDC server configured in Admin tab.	Boolean	false
(Authentication Mode)	-	isSamlEnabled	User will be authenticated by SAML server configured in Admin tab.	Boolean	false
(Authentication Mode)	-	isJwtEnabled	Should JWT key be used. It is used to sign and validate the token signature.	Boolean	false
-	-	isExternalPortalApprover	Is it external approver? The External Approvals option is a feature that allows external users (without a Heimdall account) to approve or deny sessions requested in the portal. To enable this, several prerequisites must be met: External Approvals Overview	Boolean	-
Groups Add	-	groupsToAdd	Groups to add to given user	(List of groups)	-
Groups Remove	-	groupsToRemove	Groups to delete from given user	(List of groups)	-
Two Factor Authentication	Authentication Mode = Local/ Secret OR LDAP	sharedSecret	If enabled, it will present bar-code that can be scanned into the Google Authenticator software, and an account code, which can be used in place of the bar-code. This ID is in addition to the normal password authentication the user will be required to provide. Not available with Authentication mode: Kerberos.	String	-
-	-	file	Name of the file that contains given user configuration	String	-

Request Rules options:

This is list of options for **request rules**:

Detailed information regarding the rules can be found in [rules](#)

These options are saved inside rules configuration file. It contains both request and response rules. The filename depends on the name of the rule list → nameOfRuleList.conf.

These are stored in the following format under rules:

```
"rules": [
  {
    "enabled": true,
    "type": "K",
    "patterns": [
      "MultiplexRegexHere"
    ],
    "rowPatterns": [
      ""
    ],
    "operator": "AND",
    "columnNameOperator": "AND",
    "intrans": true,
    "properties": {
      "action": "enable"
    },
    "notes": "Notes here"
  },
]
```

Each rule might have: * **enabled** - Boolean to indicate if the rule is active. * **type** - Action of the rule inside the configuration file, such as Q → Allow. * **Patterns** - The Java regex for queries to match on. Regex column from GUI. * **rowPatterns** - Used by response rules. * **intrans** - Determine if rule matches in transactions or only on auto-commit mode. * **properties** - Parameters and their values for given rule. * **notes** - Extra notes that can be left, their presence is marked by message icon next to enabled option. Editable after clicking edit button.

 **Note:** You can use SHIFT with mouse scroll to scroll horizontally!

Key	Type	Description
allow	Q	Explicitly allow a query to bypass the learn and drop rule types
async execute	A	Execute insert/update/delete asynchronously, not supported by MySQL proxy
cache	C	Use the cache configuration for the VDB to cache matching queries, if possible
call	G	Execute (call) a named rule list, must also be included in the VDB config to work

Key	Type	Description
debug	Z	Allows to enable and disable debug mode for a connection
drop	D	If not allowed with an allow rule, drop the request, and generate SQL Exception
extract plan	O	Automatically (periodically) retrieve query plan on queries
forward	F	Allows queries to be sent to alternate data sources for execution
ignore	I	Ignore rules of the specified type in the current rule-list that occur after this rule
learn pattern	P	Generate a firewall learn action to add a rule to a named rule list. A new rule will not be added if one with the same pattern and action type already exists.
log	L	Log the SQL observed in the query
multiplex	K	Manage multiplex behavior (works only when the multiplex is enabled in VDB config)
nocache	B	Disable caching for a query, overrides previous cache rules as well as prevents later cache rules from being evaluated
pool	M	Pool and connection multiplexing behavior overrides
procedure wrap	1	Wrap query into a procedure on proxy side (only for SQLServer).
reader eligible	V	Flag that the query is allowed to be processed by a read-only server in a cluster.
result	N	Return a (fixed) result
retry	E	Retry execution if exception occurs
stack trace	S	Generate a Java stack trace in logs on match
table cache	W	Apply a TTL to a query based on if the table specified is referenced by the query. Lower TTL values override higher ones
tag	X	A no-op action, for use with universal parameters only
transform	T	Change the query based on capture group manipulation
trigger	R	Trigger another SQL or script to execute as a result of a matching query

Parameters for rules

General parameters:

They can be used for any request rule:

Key	Description
delay	Adds X microseconds to each query, to help model the impact of latency between the db client and server, for hybrid cloud impact testing
evicttables	Tables that should be flagged as evicted when a query matches the rule
exclude	Exclude matching the rule 1/X times, to allow a small sample of queries to match
invalidate	Can be specified to override invalidation behavior of writes for matching queries, default is to allow invalidations to occur.
log	When specified, behave as if a log rule had matched, used to consolidate rules
matchdelta	Specifies the minimum time between matches on a per-proxy basis, overrides matchlimit
matchlimit	Specifies the frequency (in X per second) that a rule should match
matchthreshold	Only match to unknown queries or ones that are known to have an average runtime longer than X microseconds

Key	Description
maxburst	When using matchlimit or ratelimit, specify the number of seconds worth of bursting to allow before limiting the rate
notify	If specified as true, flag that the notification configuration should be used to notify on a match
notrack	When specified, don't track the performance of this query for statistics records. Useful to "hide" overhead queries that aren't impacted by performance
onlyTrans	When specified, only match queries that are in transaction (must have In-Trans checked as well)
olderthan	specify the time window in ms to allow content to not be evicted on eviction. Negative prevents caching into the future
preparedonly	Match only if a query is a prepared query. This can be used to trigger exceptions to change settings for users to optimize pooling behavior
printcapture	When a rule matches, print capture groups of the rule match
printmatch	When a rule matches, print the rule that matched
printmeta	Print the resultset metadata for results returned
printresults	Prints to logs the result of matching queries
printtables	If specified, print the table attributes to be used for caching & invalidation
prnttiming	Prints to logs the execution time of this rule, useful for profiling called rulists, such as for the SQL Firewall
ratelimit	Limits matching queries to X per second. This is tracked on a per-rule basis, so ratelimit and matchlimit can interact on the same rule
rowlimit	Set the connection to be limited to X rows per result-set as if the command 'set maxrows=x' had been made
sample	Match the rule only 1/X times, often to reduce frequency of logging, but remain proportionate to load
serverfilter	A filter (regex) to determine what read-only servers should be used to handle a request. If none match, this is ignored. Only used with forwarding & read/write split
stop	If specified on a matching rule, no further rule processing in the current rule list will occur
tables	For any matching rule, specifies the tables that should be added to a query's table list. Use "none" to allow caching without tables. Newlines and whitespace are acceptable between table names, but not in table names.
update	Specifies if the matching queries should be processed as updates by the cache engine
userReset	When executing a query, remove the user's credentials to use the default source user's connection pool and permissions
resultQuery	When executing a query, use an alternate query to retrieve the actual result, i.e. from a temp table
resultQueryTrans	When using resultQuery, should the initial query and the result query be wrapped together in a transaction?
serverFilter	If average runtime will be longer than given 'matchthreshold', then query will be forwarded to data source defined by 'serverfilter'.
queryTimeout	Query execution timeout
honorComments	The comments in the queries at the regex level are ignored by default. Use this flag to preserve them. Note: Enable it if you want to use the Capture Parameter .

Rule Specific

For Async Execute:

Key	Description
asyncsize	The maximum length of a query (in characters) that can be made asynchronous
batchSize	The maximum batch size of queries to be executed—queries will not be delayed to fill the queue, but this sets the maximum size of
maxPollInterval	The maximum time (in ms) to poll after an async query is received before a batch is processed, defaults to 0 for no delay the batch.
source	The name of the data source to forward the query to for async execute. This source must be specified in the VDB definition.
queueName	

Key	Description
	The per-source named async queue to process the query from. Defaults to the name of the data source the query is sent to. Set to \${table} to allow dynamic replacement of the table name (if only one).
holdUntil	When to hold the thread until
spoofedResult	The result the DML will return if the result is spoofed (only for immediate holdUntil values).
lockTable	Lock tables of async DML while the DML is pending, to serialize access to the table (default to TRUE)

For Cache

Key	Description
ttl	Specifies the TTL in ms of objects cached, if not evicted due to a write first
cachesystem	True if Heimdall should cache system tables, and not just user tables, can break things!
unconditional	if true, bypass many internal checks to insure a query is cacheable
maxLocalSize	Specify the largest size of a cachable object in local memory, defaults to Integer.MAX_VALUE
maxSize	Specify the largest size of a cachable object, defaults to Integer.MAX_VALUE
autoRefresh	Specify that queries may be automatically refreshed in cache if invalidated (but not on TTL expiration). Only a limit number of queries are tracked for this purpose, configurable on the vdb properties
revalidateOnly	Don't actually serve from cache, but validate that the result in the cache is the same as what the server serves
shared	Share a cache object across users. Note, this may bypass security, so should be used with caution. To globally trigger this, use the custom cache key option in the vdb cache settings.

For Result

Key	Description
responseInt	Return an integer value as a result of a SQL call, i.e. for rows updated
file	Return a file containing json result

For Retry

Key	Description
maxRetries	Maximum number of attempts that will be made if none of them succeeds
exception	String that exception should contain
retryDelay	Delay between retry attempts
query	The sql to execute before attempting to retry

For Forward

Key	Description
source	The name of the data source to forward the query to. This source must be specified in the VDB configuration.
setcatalog	If in forwarding, should we inherit/set the catalog from the parent connection on the forwarded connection.
setschema	If in forwarding, should we inherit/set the schema from the parent connection on the forwarded connection.

Key	Description
readonly	When forwarding, should the connection be allowed to read-only servers.
sticky	Continue forwarding to the designated source in an ongoing basis, and disables multiplexing so the same connection is reused.

For Call

Key	Description
rule	The rule name to call/execute in place of the call rule itself, if matched.

For Ignore

Key	Description
action	What type of action to ignore after matching

For Log

Key	Description
logcolumns	List of comma separated column names/aliases to save in log records, based on the resultset metadata. Use '*' or 'all' for all columns. Note: HIGHLY impacting on performance.
slowtime	Response time in microseconds above which logging should be triggered.

For Extract Plan

Key	Description
threshold	The response time threshold in ms after which the plan is requested, based on query pattern average response times
frequency	How many times the query plan can be requested per hour
format	What format to use when extracting an execution plan
force	By default Extract Plan works only for queries with result set from DB (for example, SELECT queries). To force explaining on any queries set the parameter to true.
checkTableMismatch	For SQL Server, check if tables in the explain match what is tracked internally by Heimdall

For Learn Pattern

Key	Description
rule	The name of the target rule list to add learned rules to.
action	Specifies the action of newly added rules, defaults to "allow".

For Trigger

Key	Description
type	

Key	Description
	Specifies the type of trigger action to perform. If execute, then it will call the named program or script, and execute on the central manager (always). If SQL, it will execute on the proxy/driver unless in parallel, delay and queueunique is TRUE, then it will process on the central manager.
command	The command to execute, either as sql or as a script. Accepts a replacement value of \${tables} to represent the tables in the query.
timing	When to execute the command, either before, after, or in parallel to the triggering sql. With before, if an exception is encountered on the initial query, the trigger action is not performed. If an exception is encountered with after, then the exception will be reported to the calling thread. When sql is set to execute in parallel mode, it will execute on the central manager if delayed, on the proxy/driver if not, and exceptions will only be reported in the logs.
delay	When running in parallel to the triggering sql, how much time to delay processing (in ms), if any
maxDelay	When running in parallel, delayed and queueunique, how often should we run (at least) even if the trigger action keeps occurring, i.e. to insure data inserted is flushed to another db at least X times an hour
queueUnique	When performing delayed parallel queries, should the execution queue only maintain unique command executions, to prevent the same command from executing over and over later?
source	The name of the data source to execute a SQL query on. This source must be specified in the VDB configuration.
onceOnly	Only do this trigger once during the lifetime of the proxy (or a reasonably long period). The call should be able to be triggered multiple times over time, this should be considered an optimization only.

For Stack Trace

Key	Description
threshold	Response time threshold for stack traces to trigger (in microseconds)

For Transform

Key	Description
target	When translating a query, the target of the translation, with \${1} representing the first capture group, \${2} the second, etc.
recursive	If a transformation should be repeated until no change is made between original and transformed query
escapeQuotes	If single quotes in the capture group should be escaped with "

For Table Cache

Key	Description
ttl	Specifies the TTL of objects cached, if not evicted due to a write first
table	In table cache rules, specifies the name of the table to apply settings for

For Debug

Key	Description
enabled	If true, enables driver debug logging at that point

For Pool

Key	Description
maxUserActive	

Key	Description
	Sets the per-user active pool parameter (idle+busy), and overrides any default set in the source settings. Useful to allow individual users to have more connections than others, and can be set based on AD group, say to allow service accounts more connections than general users (Integer).
maxUserIdle	Overrides the per-user allowed idle connection limit (Integer).
multiplex	Override the vdb level multiplex setting to either enable or disable multiplexing (Boolean).
multiplexSkip	Triggers skipping multiplexing for X number of queries after the current matching one (Integer).
reuse	Control if connections can be reused on return by pooling, defaults to true (Boolean). Only impacts the primary (writer) connection, not read/write split connections.

For Reader Eligible

Key	Description
Lagignore	Ignore the lag detection logic when routing a query.
unconditional	If set to true bypass internal checks that would mark query as not viable for reader servers.

For Multiplex

Key	Description
disable	When the capture parameter from regex is matched, then it is saved in memory and from this point multiplex will be disabled for all queries, unless it will be enabled again.
enable	When the capture parameter from regex is matched, then it is removed from the memory (mentioned in the "disable" point) and multiplex will be enabled for next executed query (unless there are any other captured parameters disabled)

Response Rules options:

This is list of options for **response rules**.

Detailed information regarding the rules can be found in [rules](#)

These options are saved inside rules configuration file. It contains both request and response rules. The filename depends on the name of the rule list → nameOfRuleList.conf.

These are stored in the following format under responseRules:

```
"responseRules": [  
  {  
    "enabled": true,  
    "type": "c",  
    "patterns": [  
      "columnsEndsWith:name"  
    ],  
    "rowPatterns": [  
      "k"  
    ],  
    "operator": "AND",  
    "columnNameOperator": "AND",  
    "intrans": false,  
    "properties": {  
      "honeyTokenAlert": "true"  
    }  
  },  
]
```

Each rule might have:

- **enabled** - Boolean to indicate if the rule is active.
- **type** - Action of the rule inside the configuration file, such as Q → Allow.
- **Patterns** - The Java regex for columns to match on. Column Name Regex from GUI.
- **rowPatterns** - The Java regex for row to match on.
- **intrans** - Determine if rule matches in transactions or only on auto-commit mode. Always false for response rules.
- **properties** - Parameters and their values for given rule.
- **notes** - Extra notes that can be left, their presence is marked by message icon next to enabled option. Editable after clicking edit button.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Type	Description
Alert Row	c	Raise an alert if the response data conditions are met (there is a match for Column Name Regex and Column Row Regex)
Delete Row	a	Deletes the matching rows from result set (might be used on metadata as well).
Mask Row	b	Masking row of data from result set
Tokenize Row	d	Please read Tokenizing for more information. Tokenization of result set data. Tokens generated during tokenization are stored within the designated database indicated by the specified datasource. Notably, the maximum supported length for string data is limited to Varchar(255).

Rule Specific Parameters

For each response rule

sendNotification: An option to send notifications can be added to each response rule, which will be sent if the rule is triggered. Please read [Response Rule Action Details](#) for more information.

For Alert Row:

Key	Description
honeyTokenAlert	Raise alert if queried for a "honey token" (if all conditions were met)
rowCountsAlert	Raise alert if more than a certain number of rows are queried as a result of a single query. (ColumnRowRegex field should be left blank, otherwise the alert will be promoted when at least one match is made in a row and other conditions are met - so basically it's looking for "token" that will trigger alerts)

For Mask Row and Tokenize Row:

Key	Description
replaceWith	Specifies with which string, data should be replaced. (This field is mandatory)
replaceType	Specifies the type of replacement, whether we should replace Each string/character or All with value provided by replaceWith field.
replaceFirst	Specifies how many first characters to replace with the given string provided in the replaceWith field.
replaceExceptFirst	Specifies how many of the first characters are left unchanged and the rest will be replaced.
replaceLast	Specifies how many of the last characters should be replaced with the given string specified in the replaceWith field.
replaceExceptLast	Specifies how many of the last characters are left unchanged and the rest will be replaced.
replaceAll	Specifies whether whole row of data should be replaced. Should not be combined with other masking actions (as the result will be returned immediately).
replaceBeforeString	Specifies delimiter and number of characters (in that order) to replace before given delimiter. Can be usefully with email masking.
replaceExceptFirstBeforeString	Specifies delimiter and number of characters (in that order) that should remain unchanged (counting form the beginning of the delimiter) and the rest of the string will be masked to the specified delimiter.
replaceAfterString	Specifies delimiter and number of characters (in that order) to replace after the given delimiter.
replaceExceptLastAfterString	Specifies delimiter and number of characters (in that order) that should remain unchanged (counting form the end of the delimiter) and the rest of the string will be masked to the specified delimiter.
replaceFirstRegex	Specifies the regex how the field should be replaced, matches only the first matching regex pattern. The replaceType field has no effect on regular expression processing.
replaceAllRegex	Specifies the regex how to replace the field, matches all matching regex. The replaceType field has no effect on regular expression processing.

Key	Description
excludedChars	This field specifies which characters should not be tokenized or masked. It's allowed to specify multiple excluded chars by typing them all in the String field e.g. if '-' and '@' needs to be excluded, user should define them as '-'@' in one String. (If no other replacement is specified the replaceAll will be taken from default). If the user wants to specify space " " as excluded char, due to a trimming policy, he should define it in the middle of string or as a Unicode value "\u0020"
ssn	Enables masking of Social Security Numbers. All digits in the value are replaced character-by-character with the chosen by replaceWith string, while separator characters such as dashes and spaces remain unchanged (for example, 111 22-3333 becomes *** *-****). When enabled, this option forces full-value masking and overrides the replacement type to operate on each character. This is the equivalent of: replaceAll = true, excludedChars = "- ", replaceType = EACH.
credit card	Enables masking of credit card numbers. All digits and letters are replaced character-by-character with the chosen by replaceWith string, while formatting characters such as spaces and dashes remain unchanged (for example, 1111 1111 1111 1111 becomes **** *-* **** *-*). When enabled, this option forces full-value masking and overrides the replacement type to operate on each character. This is the equivalent of: replaceAll = true, excludedChars = "- ", replaceType = EACH.
iban	Enables masking of IBAN account numbers. All digits and letters in the IBAN are replaced character-by-character with the chosen by replaceWith string, while separator characters such as spaces and dashes remain unchanged (for example, DE89 3704 0044 0532 0130 0000 becomes **** *-* **** *-*). When enabled, this option forces full-value masking and overrides the replacement type to operate on each character. This is the equivalent of: replaceAll = true, excludedChars = "- ", replaceType = EACH.
phone number	Enables masking of phone numbers. All digits and letters are replaced character-by-character with the chosen by replaceWith string, while formatting symbols such as spaces, dashes, plus signs, and parentheses remain unchanged (for example, +1 (212) 555-8877 becomes +* (***) ***-****). When enabled, this option forces full-value masking and overrides the replacement type to operate on each character. This is the equivalent of: replaceAll = true, excludedChars = "- +()", replaceType = EACH.
email	Enables masking of email addresses. All characters except the @ symbol are replaced character-by-character with the chosen by replaceWith string, preserving only the position of the @ (for example, erik@heimdalldata.com becomes ****@*****). When enabled, this option forces full-value masking and overrides the replacement type to operate on each character. This is the equivalent of: replaceAll = true, excludedChars = "@", replaceType = EACH.

For Tokenize Row:

Tokenize Row has all parameters from Mask Row. If nothing is specified the default ReplaceAll action will be performed. For the duplicates the description is slightly changed to reflect difference in masking vs tokenizing.

Key	Description
source	The data source used to store generated tokens. By default, it matches the data source used for the query. Only one data source can be selected per query. If multiple tokenization rules specify different data sources, the last applicable rule takes precedence, and all tokens for that query are stored in the same data source. The data source must be created beforehand, on the datasource tab, then it will be available for selection. The table for storing tokens is created via create-token-table.sql scrip, by clicking 'Execute scripts'. The table with the tokens is stored in 'heimdall' schema and 'tokens' table containing 'token' and 'mapped_value' columns. After applying this option, a token datasource will occur on the status tab gui, for checking the state of that source.
ssn	Enables tokenization of Social Security Numbers. All digits in the value are replaced by a generated token, while separator characters such as dashes and spaces remain unchanged (for example, 111-22-3333 may be tokenized to 423-48-1249). The token represents the original value and can be used for consistent referential usage without exposing the actual SSN. When enabled, this option forces full-value masking and overrides the replacement type to operate on each character. This is the equivalent of: replaceAll = true, excludedChars = "- ".
credit card	Enables tokenization of credit card numbers. All digits are replaced by a generated token, while formatting characters such as spaces and dashes remain unchanged (for example, 1111 1111 1111 1111 may be tokenized to 2352 5743 6789 0632). The tokenized number preserves the layout but no longer reveals the original card number. When enabled, this option forces full-value masking and overrides the replacement type to operate on each character. This is the equivalent of: replaceAll = true, excludedChars = "- ".
iban	Enables tokenization of IBAN account numbers. All digits and letters are replaced by a generated token, while separator characters such as spaces and dashes remain unchanged (for example, DE89 3704 0044 0532 0130 0000 may be tokenized to KF60 1123 1426 3326 1422 7223). The structure is retained for readability, but the underlying account information is protected. When enabled, this option forces full-value masking and overrides the replacement type to operate on each character. This is the equivalent of: replaceAll = true, excludedChars = "- ".
phone number	Enables tokenization of phone numbers. All digits are replaced by a generated token, while formatting characters such as spaces, dashes, plus signs, and parentheses remain unchanged (for example, +1 (212) 555-8877 may be tokenized to +3 (164) 236-1035). This allows preserving the phone number format without exposing the real value. This is the equivalent of: replaceAll = true, excludedChars = "- +()".
email	Enables tokenization of email addresses. All characters except the @ symbol are replaced by a generated token (for example, erik@heimdalldata.com may be tokenized to skf1@jufj1akwperv1pau). The position of @ is preserved so that the token still resembles a valid email-like string while protecting the original address. This is the equivalent of: replaceAll = true, excludedChars = "@".

General options:

This is list of general options for **Virtual DB**.

These options are inside given virtual database configuration file. For example: postgresTest-vdb_1.conf. The filename depends on the name of the vdb → nameOfVdb.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Requirements	Conf File field	Description	Possible values	Default
Enabled	-	enabled	If unchecked, this will result in a proxy disabling access via the port specified. In JDBC mode, this will result in an SQL exception being thrown on a new connection attempt.	Boolean	True
Access Mode	-	accessType	The first step in configuring a VDB is to select the access mode, which will adjust what options are presented, to simplify the configuration	String (from list)	-
Access/Secret Keys	-	accessKey / secretKey	These are generally used in distributed proxy or JDBC mode to provide a shared user/ password to allow the remote proxies to pull the complete vdb configuration. Note: The username and password can be stored in the Secrets Manager .	String	-
Secret (Access/Secret Keys)	-	useSecretForVdbCredentials	If the VDB will use secret for Access/Secret keys.	Boolean	false
Secrets Manager (Access/Secret Keys)	Secret (Access/Secret Keys)	vdbCredentialsSecretsManagerConfigName	Name of the Secrets Manager Config. If you are setting it manually, it can be random name.	String	-
Deployment Environment	-	deploymentEnvironment	Records the deployment environment of a VDB, supported values: {Development, Staging, User Acceptance Testing, Production}. Groups VDBs by environment within Status tab.	String (see desc.)	Development

Proxy Configuration options:

This is list of options for **proxy configuration**, the requirement to see these options is to set **Access Mode != JDBC**:

These options are inside given virtual database configuration file. For example: postgresTest-vdb_1.conf. The filename depends on the name of the vdb → nameOfVdb.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Requirements	Conf File field	Description	Possible values	Default
Local Proxy	-	mgmtProxyEnable	This option allows the management server to start and manage restarts of a proxy instance on the same server as the management server itself. This option is a great way to simplify the testing of Heimdall in limited environments.	Boolean	false
Run as Service	Local Proxy	runAsService	This option allows the management server to run a proxy as a linux service. It allows proxy to run even while management server is off.	Boolean	false
Address Binding Type	-	proxyAddress	Required. Options: Any: In this binding mode, all local IP addresses will be bound to, or more specifically, it binds to "0.0.0.0", localhost Only: When this option is set, the binding will be to the 127.0.0.1 IP only, specific IP: This allows a specific IP address to be bound for use. An example of this is to use 172.17.0.1 in a docker container when bridge mode is enabled, so that the proxy can bind for use by any other containers on the same host, without impacting any other proxies that may reside on another host.	String (see desc.)	0.0.0.0 (Any)
Proxy Port(s)	-	proxyPort	The second required option. This specifies the value of the TCP port that the proxy will be listened to. If there is an error binding to a given port or IP, then a GUI alert will be issued when the proxy attempts to start. It is important that the ports do not conflict with other proxies being run on the same host or binding, as only one process can bind to a specific IP:port combination at once. This applies if installed on the same server as a database as well—if the database is on port 3306 for example, than the proxy can not use the same port. You can set: Single port by typing value of port (for example: 5433). Multiple ports by separate them by comma (for example: 5433, 5434, 5435).	String	5050

Key	Requirements	Conf File field	Description	Possible values	Default
			Range of ports by separate them by minus (for example: 5433-5435). Mix (for example: 5433, 5435-5436, 5450).		
Proxy Address	Address Binding Type = Specific IP	proxyAddress	Allows specific address, must use Specific Ip option from Address Binding Type.	String (IP)	-
Max Heap Size (MB)	Local Proxy	xmx	The setting for the Proxy java heap memory limit. Default is '600M' which with overhead, will generally consume up to about 1GB of total RAM. This will also result in the setting "Xms" to set the smallest heap size, so as to try to provide more consistency in the "free memory" graph on the dashboard and to ensure that the total memory is always available. Please note, that in distributed proxy mode, this setting has no effect—the java options need to be configured via the user data or heimdall.conf file on the remote instance. For more details of a file <code>/etc/heimdall.conf</code> , please see heimdall.conf configuration.	Integer	600
Proxy GSS Support	Authentication Mode = Kerberos/ GSSAPI AND PostgreSQL	gssEncEnabled	When enabled, and the client requests it, this option will enable GSS negotiation.	Boolean	false
Proxy GSS Required	Proxy GSS Support AND PostgreSQL AND !Proxy TLS Required	gssEncRequired	When enabled, this option will force all connections to connect only with GSS. Any attempt to connect without GSS will be rejected. This option can't work together with TLS required option.	Boolean	false
Proxy TLS Support	-	tlsEnable	When enabled, and the client requests it, this option will enable TLS negotiation. Initially, a self-signed certificate will be generated for the proxy, which can then be customized in the Java keystore file in the proxy install directory.	Boolean	false
Proxy TLS Required	Proxy TLS Support = true AND !Proxy GSS Required	tlsRequired	When TLS is enabled, this option will force all connections to connect only with TLS. Any attempt to connect without TLS will be rejected. For Postgres and MySQL, warnings will be provided to the client on the disconnect, with SQL Server, only an alert on the GUI will be presented.	Boolean	false
Enable Legacy TLS	Proxy TLS Support = true	tlsLegacy	Enables support for legacy TLS 1.0 and TLS 1.1..	Boolean	false
Certificate	Proxy TLS Support = true	certificateAlias	Certificate assigned to Virtual Database for TLS connections.	String (global_use_certificate, tomcat)	-
Case Sensitive Usernames	Access mode = SQL Server Proxy	userCaseSensitive	Use this option if you want SQL Server username to be case sensitive	Boolean	false
Authentication mode	-	authMode	This dropdown allows selecting of the proxy authentication mechanisms. Please see the Theory->Proxy Authentication	String (options in desc.)	-

Key	Requirements	Conf File field	Description	Possible values	Default
			page for more information on authentication. Possible values are: Passthrough, Proxy Configured Users, SQL Driven, Active Directory/LDAP, Kerberos/GSSAPI		
user	Authentication mode = Proxy Configured Users	users.user	User required for proxy connections.	String	-
password	Authentication mode = Proxy Configured Users	users.password	Password for proxy user.	String	-
Admin User	Authentication mode = SQL Driven OR Synchronize Db Authentication	syncUser	Privileged Database User	String	-
Admin Password	Authentication mode = SQL Driven OR Synchronize Db Authentication	syncPassword	Privileged Database Password	String	-
Sync Command	Synchronize Db Authentication	syncCommand	Synchronization command, use \${user}, \${password} and \${ldapgroups} as placeholders.	String	-
Auth Cache Expiration Time	Synchronize Db Authentication	authenticationCacheExpirationTime	Determines how long (in ms) that data will stay in Authentication Cache.	Long	-
Authorization Query	Authentication mode = SQL Driven	authorizationQuery	Query to retrieve data for authorization.	String	select * from heimdall.pg_hba where enabled = true order by line_number asc
Dual Authentication Mode	Access mode = SQL Server Proxy AND Authentication mode = Kerberos/GSSAPI	dualAuthMode	Facilitates the use of Kerberos Authentication alongside passthrough authentication. However, there's a condition: users must adhere to a single authentication method and cannot switch between them interchangeably.	Boolean	false
Keytab Location	Authentication mode = Kerberos/GSSAPI	keytabLocation	Location to keytab file that stores secret keys for service principal (proxy).	String	-
Service Principals	Authentication mode = Kerberos/GSSAPI	kerberosServicePrincipals	Specifies if synchronization of users and groups should be performed. Moreover, it allows configuring how long that data will stay in Authentication Cache (Auth Cache Expiration Time). Note: The username and password can be stored in the Secrets Manager . More information about DB authentication synchronization can be found in DB authentication synchronization	String	-
Synchronize Db Authentication	Authentication mode != None OR Passthrough	synchAuthentication	Synchronize Users & Groups.	Boolean	false
Tokenized Authentication	Synchronize Db Authentication AND Authentication	tokenizedAuthentication	Tokenized authentication is security feature which uses token instead of the database password. This ensures that the	Boolean	false

Key	Requirements	Conf File field	Description	Possible values	Default
	mode != Kerberos/ GSSAPI		proxy will not be bypassed and the database password will not be exposed. This option is available for Proxy Configured Users, SQL Driven and LDAP authentication modes. When using Tokenized authentication with multiple proxies, it is essential to enable an external cache like Redis or Hazelcast. This ensures that all proxies are synchronized and Tokenized authentication works smoothly on all proxies.		
Allow Portal User To Log In	Portal Mode is enabled	allowPortalUsersToLogin	This option is available when Portal Mode is enabled. If selected, before each proxy authentication (for all authentication types except Kerberos), the system will check whether the user is a session user in the portal. If they are and the credentials are valid, they will be authenticated. If the user is not a portal user, authentication will proceed according to the mode selected in authentication mode.	Boolean	true

Authentication Test

It's an option to test authentication. Requirements to see this subsection: **Authentication mode** must be one of: Proxy Configured Users, SQL Driven, Active Directory/LDAP

Key	Description	Possible values
Test User	Test Password	String
Test Password	Test User	String
Test IP Address	Taken from Data Source by default. If JDBC URL there contains \${host} this field has to be provided.	String (IP)
Test Database	Taken from Data Source by default. If JDBC URL there contains \${database} this field has to be provided.	String
UseSSL	Should SSL be used for authentication test.	Boolean

Caching options:

This is list of options for **caching**:

These options are inside given virtual database configuration file. For example: postgresTest-vdb_1.conf. The filename depends on the name of the vdb → nameOfVdb.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Requirements	Conf File field	Description	Possible values	Default
Enable Query Caching	- (Unlocks this section)	cacheEnabled	Check to enable caching—if not checked, no cache rules will trigger caching, although rule processing will be done as normal.	Boolean	false
PCI/HIPAA Cache	-	pciHipaaCache	When enabled, no cached objects will actually be transmitted to the grid-cache, but the cache will be used to message invalidation requests. This is to avoid transmitting possible sensitive data across unencrypted connections, while maintaining cache synchronization integrity. This option is mutually exclusive with disabling the grid cache offload function, and will remove the benefit of the cache in cold-start scenarios.	Boolean	false
Cache Type	-	cacheType	Select the desired cache type. Options will adjust based on the cache type selected, see below tables to see their options. Hazelcast , Valkey / Redis / KeyDB	String	Hazelcast
Use soft references	-	cacheSoftValues	If enabled, soft references will be used for caching, highly advised.	Boolean	true
Duplicate request tracking	-	preCacheTracking	Track the query hash in a temporary list with the same TTL as a cache itself, and only cache if we have a "hit" in this temporary list. This prevents caching of unique queries that will never return from cache.	Boolean	false
Max Expiry	-	cacheMaxExpiry	If set to a value above zero, this value (in minutes) is the maximum TTL the L1 cache will cache an object for. This overrides the TTL in the cache rules if it is lower than the TTL set there FOR L1 cache queries. This value does not impact the TTL in the L2 cache such as Redis.	Long	5
Max Cached Objects	-	cacheObjectCount	If set to a value other than 0, this limits the total number of objects in the L1 cache layer to a fixed maximum. This can assist in reducing memory garbage collection time under heavy load, and is advised under very high request volume.	Long	100000
Max Cached Size	Access type = Proxy	cacheObjectSize	In proxy mode, this setting controls how much of a result-set's transmit buffer is saved before caching is aborted. Any result-set larger than this will not be cached. This value should not exceed 1/50 of the proxy heap size.	Long	64000000
Customize Query Cache Key	-	customKey	This allows the key used to access cached objects to be customized in order to expand when the result is considered the same. By default, all queries will include a hash of the complete SQL query, but also includes the VDB name the request was made through, the database user, catalog and schema. These fields can be removed to allow, for example, multiple VDBs to share the same data in the cache.	Boolean	false
Exclude VDB Name	Customize Query Cache Key	excludeKeyVdb	Should virtual database name be excluded from the hash key?	Boolean	false
Exclude JDBC to DB User	Customize Query Cache Key	excludeKeyUser	Should connected database user be excluded from the hash key?	Boolean	false

Key	Requirements	Conf File field	Description	Possible values	Default
Exclude DB Catalog	Customize Query Cache Key	excludeKeyCatalog	Should catalog be excluded from the query cache hash key?	Boolean	false
Exclude DB Schema	Customize Query Cache Key	excludeKeySchema	Should schema be excluded from the query cache hash key?	Boolean	false
Exclude SQL Comments	Customize Query Cache Key	excludeKeySqlComments	Should sql comments be excluded from the query cache hash key?	Boolean	false
Grid Cache Offload	PCI/HIPAA Cache = false AND Cache Type != Local only	localCacheEnable	This option, if enabled, enables a first-tier local cache, which allows hot content to be served out of local heap. In the local cache, the objects will be stored in the performance optimal format (binary streaming format for a proxy, and Java objects when not a proxy).	Boolean	true
Cluster manager via cache	Cache Type = Hazelcast or Valkey/Redis/KeyDB	clusterCacheManager	This option allows a VDB to be managed by multiple management nodes at once, via the cache interface. For this to work, all management nodes must be populated with the vdb, and the cache settings configured to be identical. Once done, configuration changes for that VDB will be synchronized between nodes, and the cache will be used to propagate metrics as well, allowing the dashboard to be used at the same time on both nodes. Further, if one management node goes offline, the access layer nodes will remain unaffected.	Boolean	false

Hazelcast

This is a Cache Type option, visible when it is set to Hazelcast, and it has given configuration items:

Key	Requirements	Conf File field	Description	Possible values	Default
API Cache Name	-	gridCacheName	For interfaces that utilize a "named" cache interface, this is the name of the cache to use, i.e. Hazelcast.	String	-
Cache Configuration File	-	cacheConfig	When supported, this is an external configuration file to configure the cache, i.e. a Hazelcast client or server XML configuration file. The location of the file should be relative to the current working directory of the JDBC driver or proxy, which will be printed on startup. On a proxy install, when this file is not present, it will be pulled from the Heimdall Server.	String	-
Group ID	-	cacheGroupId	When using a hazelcast, the hazelcast group id (optional).	String	-
Group Password	-	cacheGroupPassword	When using a hazelcast, the name of the group password to use (optional).	String	-

When using Hazelcast with AWS Auto-discovery, these options may be necessary for proper discovery of other Hazelcast nodes:

Key	Conf File field	Possible values	Default
AWS Access Key	cacheAwsAccessKey	String	-
AWS Secret Key	cacheAwsSecretKey	String	-
AWS Tag Key	cacheAwsTagKey	String	-

Key	Conf File field	Possible values	Default
AWS Tag Value	cacheAwsTagValue	String	-

Valkey/Redis/KeyDB

This is a Cache Type option, visible when it is set to Valkey/Redis/KeyDB, and it has given configuration items:

Key	Requirements	Conf File field	Description	Possible values	Default
Server	-	gridServer	For external grid cache interfaces, specifies the server name or IP to connect to.	String	-
Port	-	gridPort	When using an external grid cache, specifies the TCP port to connect to.	Integer	0
Database Number	-	databaseNumber	For Redis, which database "number" is used, not supported when the Redis instance is in cluster protocol mode.	Integer	0
Cache Username	-	gridUsername	If necessary cache username to use	String	-
Cache Password	-	gridPassword	For interfaces with a password authenticated interface, the password to connect to the cache with. Note: The username and password can be stored in the Secrets Manager .	String	-
Use SSL	-	useSsl	When connecting to the cache, should SSL be used for the connection (Redis)	Boolean	false
Verify Peer	Use SSL	verifyPeer	When using SSL, verify the peer TLS certificate	Boolean	false

Advanced Features options:

This is list of options for **advanced features**. For more detailed information please read [advanced features](#)

These options are inside given virtual database configuration file. For example: postgresTest-vdb_1.conf. The filename depends on the name of the vdb → nameOfVdb.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Requirements	Conf File field	Description	Possible values	Default
Multiplex	Access Mode = Proxy	multiplex	Please read Multiplex Considerations for detailed information. Do load balancing to the server at the transaction level vs. the connection level. This can drastically reduce the number of connections being established to the server.	Boolean	false
Multiplex timeout	multiplex	multiplexTimeout	Please read Delayed Transaction Considerations for detailed information. When multiplexing how many milliseconds should occur before releasing the connection on the back-end. In general, it is safe to leave this at 0.	Integer	0
DelayedTransaction	Access Mode = Proxy	delayedTransaction	Delay when a transaction is started until a DML is detected. This is useful when using a framework that does everything in transactions.	Boolean	false
Auto-Scaling Mode	Access Mode = Proxy	spModeEnabled	Enable to activate auto load-balancing features of Heimdall proxies, for use in multi-proxy deployments. This includes DNS and HTTP based redirects. This option requires external cache (Redis or Hazelcast) configured for VDB in order to work. See below for the options relevant for this feature.	Boolean	false
Load Balancing Criteria	Auto-Scaling Mode	proxyBalanceMode	For redirection multiple criteria can be used, the most common of which will be by load. This will select the lowest loaded (by cpu) proxy to receive traffic. Other options include random and active/standby. DNS based load balancing supports all of these options while health check (HTTP) works only with active/standby mode, which redirects all traffic to single node and keeps other ones as backup in case primary one fails.	ACTIVE_STANDBY, LOAD, RANDOM	-
Proxy Redirect Name	Auto-Scaling Mode	redirectBy	When redirecting, we can choose to either redirect by the proxy's IP address, hostname or in AWS environment by AWS Public IP. In the case of hostname, this must resolve to all clients connecting as-is, which often won't be the case. In general, the IP address should be used.	PRIVATE_IP, HOSTNAME, AWS_PUBLIC_IP	PRIVATE_IP
DNS Port	Auto-Scaling Mode	dnsPort	If set to anything other than 0, each proxy will listen for DNS queries on the specified port, and will answer ANY DNS query with a list of proxies to connect to (one in cas of active/standby mode),	Integer	0

Key	Requirements	Conf File field	Description	Possible values	Default
			sorted in order based on selected mode (random by default). This can be used in conjunction with the client redirect to remove NLB completely from the database protocol path, yet still provide reliable load balancing. In general, this should be set to 54 to ensure that there is no conflict with any Systemd listener on the host. The central manager can also be configured to receive such queries, see Server Properties		
TrackQueryDistributionCount	-	trackQueryDistributionCount	Count of the number of query entries to track for auto cache refreshing (requires autoRefresh cache option). Rarely used.	Long	1000
Paranoia	-	paranoia	Enable paranoid logging, don't reveal the SQL patterns or queries, only their hashes in debug logs	Boolean	false
Health check port	-	healthCheckPort	If the proxy should do a self-check on itself, and expose the status via the designated port. With an HTTP connection, request /status to determine the status of the proxy. It can be used with auto-scaling set to active/standby mode to redirect all /status requests to active node allowing external load balancers to route traffic only to one proxy and keep other as backup.	Integer	-
Health check interval	Health check port	healthCheckInterval	When using the healthcheck-port option, specify how often the proxy should issue a health-check on itself. Time in seconds.	Integer	-
Enable token authorization	healthcheck port	healthCheckTokenAuthorizationEnabled	If enabled user that wants to perform api base invalidation will have to provide correct authorization token in request params.	Boolean	false
Security token	Enable token authorization	healthCheckAccessToken	Security token should be added to request as parameter for example token="exampleToken".	String	-
JMX Port	-	jmxPort	If JMX monitoring is desired, set the port here. Typically 1099 if used.	Integer	0
JMX Hostname	JMX Port	jmxHostname	The hostname that JMX uses to refer to itself—the monitoring clients must resolve this name to the Heimdall instance. The value of \${hostname} can be used to map to the internally detected hostname, and can be combined with other string values, i.e. "\${hostname}-proxy" could be used to provide proper resolution on a remote client that doesn't have access to the actual hostname via dns.	String	\${hostname}
JMX Username	JMX Port	jmxUsername	The username the JMX client will use to connect to the proxy	String	-
JMX Password	JMX Port	jmxPassword	The password for JMX connectivity. Note: The username and password can be stored in the Secrets Manager .	String	-

Logging options:

This is list of options for **logging**

These options are inside given virtual database configuration file. For example: postgresTest-vdb_1.conf. The filename depends on the name of the vdb → nameOfVdb.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Requirements	Conf File field	Description	Possible values	Default
Log Connections	-	logConnections	Log when a JDBC or TCP connection is established by the calling application	Boolean	false
Log All SQL	-	logSql	Equivalent to enabling a LOG policy with a .* wildcard. If this is enabled, it will override any log rate control configured on a log policy.	Boolean	false
Log All Portal Sessions SQL	-	logPortalSql	Logs all SQL queries performed via Portal Sessions to heimdallaudit.heimdall_users_activity_audit_trail. If this is enabled, it will override any log rate control configured on a log policy. Available only if Portal Mode is enabled.	Boolean	false
Exclude parameters	Log All Portal Sessions SQL	excludePortalSqlParams	Exclude parameters from Portal Session SQL queries, e.g. 'INSERT INTO table VALUES (?, ?)'.	Boolean	false
AWS CloudWatch Logging	AWS	cloudLogging	If in AWS, and selected, then logs generated by VDB instance will be periodically sent to the CloudWatch as a log stream.	Boolean	false
AWS CloudWatch Metrics	AWS	cloudMetrics	If in AWS, and selected, then multiple metric points will be generated under the Heimdall category for the VDB and AWS instance to allow monitoring of the access layer performance. Note, these metrics are generated on a per-minute basis, so may add to the cost of CloudWatch. Additionally, console logs will be shipped to CloudWatch as well when selected.	Boolean	false
AWS Logs Namespace	AWS	logsNamespace	If in AWS, a name provided as that field will be used as a log group name AND metrics namespace in CloudWatch.	Boolean	false
Aggregate Console Logs	-	shipConsoleLogs	In order to simplify debugging, this option allows console logs from the access layers to be sent to the central manager for logging. When using the "Local proxy" option, this option will be implicitly configured for the management server managed proxy instance, even if not set for other proxies.	Boolean	false
Write Logs to Files	-	writeLogsToFile	When performing SQL logging, the logs are by default written to an internal database for optimal Analytics speed. If the logs should also be written to detailed CSV logs for each query, please select this option. This may result in a large amount of data written to the log directory, although logs will automatically be cleaned up when that filesystem reached 90% full.	Boolean	false
Log Authentications	-	logAuthentications	Each authentication attempt will be logged. We will get information about whether authentication succeeded for a given user with the currently selected authentication mode. You can read more about authentication here Theory->Proxy Authentication .	Boolean	false

Debugging options:

This is list of options for **debugging**

These options are inside given virtual database configuration file. For example: postgresTest-vdb_1.conf. The filename depends on the name of the vdb → nameOfVdb.conf.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Requirements	Conf File field	Description	Possible values	Default
Pass-through Enabled	Access Mode = JDBC	loggingEnabled	When in JDBC mode, the system can be set to pass any NEW connections through to the underlying driver, bypassing all processing logic. This should only be used in rare situations to debug if the system is causing a problem directly, or the existence of the system is causing a problem.	Boolean	false
Verbose Debug Mode	- (Unlocks below options)	debugMode	In order to diagnose issues with rule processing or other behavior anomalies, this option can be set in order to track the processing of a query through the access layer. This option can cause significantly performance penalties and should be used with caution.	Boolean	false
Connection Handshake Tracing	Access Mode = PostgreSQL Proxy	connectionHandshakeTracingEnabled	Enables detailed logging of the proxy connection handshake process. When this option is turned on, the system records additional diagnostic information for each step involved in establishing a connection to the proxy.	Boolean	false
Lite Debug Mode	-	liteDebug	Works similar to verbose debug mode, doesn't log data all the time, instead keeps it in a buffer for up to a second and dumps all records on an exception.	Boolean	false
Log Methods	Access Mode != JDBC	logMethods	log all JDBC method calls made by an application, excluding those relating to resultSets. Warning!	Boolean	false
Log ResultSet Methods	Log Methods AND Access Mode != JDBC	logResultSetMethods	Include ResultSet operations when logging other methods. Warning!	Boolean	false

VDB properties:

This is list of properties for *VDB*.

These options are inside given virtual database configuration file. For example: postgresTest-vdb_1.conf. The filename depends on the name of the vdb → nameOfVdb.conf.

These are stored in the following format under properties:

```
"properties": [  
  {  
    "key": "suppressNoResult",  
    "value": "true"  
  },  
]
```

So each key name is used under properties with its value.

⚠ Note: You can use SHIFT with mouse scroll to scroll horizontally!

Key	Description	Possible values	Default
suppressNoResult	With Postgres, if an update query is executed via the executeQuery result, it will generate an exception on return saying "No results were returned by the query". In some frameworks, this is detected and suppressed when using the native Postgres driver, but not with the Heimdall driver. In order to work around this behavior, this option can trigger us returning a null instead of a resultset, which appears to allow the calling code to work fine.	Boolean	false
dnsCacheTTL	The cache time (in s) to use for the dns resolution cache, defaults to 5s to conform with AWS requirements. Note: This will impact the global JVM setting for this if used in JDBC mode.	Integer	5 (seconds)
dnsNegativeTTL	The time to cache negative DNS (non-existent) queries, defaults to 10s per Java standard. Note: This will impact the global JVM setting for this if used in JDBC mode.	Integer	10 (seconds)
delayCacheInit	The time in seconds to delay initializing the cache, normally not needed.	Integer	0
reparsePrepared	In cases where prepared statements include dynamic variables, this triggers the reparsing of the expanded query for the purposes of isolating it's base query pattern. This can resolve issues when patterns overflow the proxy or driver memory, or analytics show the pattern with variables included.	Boolean	false
rejectPrepared	As many features end up disabled with prepared statements, it may be needed to reject prepared statements in a QA environment to ensure they don't creep into production code. This will result in an SQL exception if they are attempted.	Boolean	false

Proxy only

These properties are applicable only to VDB working in Proxy mode.

Key	Requirements	Description	Possible values	Default
connectionIdleTimeout	healthcheck port AND interval	Connection timeout (in ms) for an idle client-side connection, will terminate connection's thread.	Integer	0
queryTimeout	healthcheck port AND interval	Timeout (in ms) for a connection executing a query, will terminate connection's thread.	Integer	0

Key	Requirements	Description	Possible values	Default
maxConnectionThreads	connectionsCleanupPercentage	Max amount of connection threads allowed. Exceeding the limit will terminate the connections that were idle for the longest time.	Integer	0
connectionsCleanupPercentage	-	Percentage of connection threads that will be terminated after maxConnectionThreads limit is exceeded. The excess connection is not included in calculation, for example when limit of 10 connections is exceeded (we get 11th connection), the amount to be deleted is calculated from 10. Setting this property to 10 means 10%.	Integer	0
desiredCipherSuite	-	Forces defined cipherSuite to be used, for TLS connections. All other than defined ciphers will be disabled, use it with caution. Example values: TLS_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	String (see desc.)	-
extractTlsSecrets	Verbose Debug Mode	Extract the shared secrets from secure TLS connections for use with Wireshark. Secrets are stored in log folder. To apply this property, Proxy restart is required. It is also recommended to enable "Close connections" checkbox in packets capture configuration to close connections so that every connection has secrets extracted correctly for packet decryption.	Boolean	false
enableBlacklist	-	Enable adding address to blacklist after many failed database authentication attempts.	Boolean	false
VerboseKerberosDebug	-	Enable verbose debugging of kerberos - encryption wont be supported.	Boolean	false
defaultTimeZone	PostgreSQL	Default Time Zone for proxy. Example values: Europe/Warsaw, America/New_York.	String	-
legacyEncryptingStream	-	Uses legacy encrypting streams for TLS and GSS. By default, new streams are used for encryption. If something starts to malfunction, it is possible to revert to the old streams by setting this flag to true.	Boolean	false
ignoreParametersBindingForSpExecuteSql	SQLServer	Ignore binding parameters when executing stored procedure with "sp_executesql". Applicable only for SQLServer Proxy.	Boolean	false
disableSetReadOnly	PostgreSQL OR MySQL	Disable any kind of SET READ ONLY type of queries. Just log them and ignore.	Boolean	false
rewriteBinaryQueryToHexLiteral	MySQL	Prevents data corruption when sending raw binary data in text-based SQL queries. Heimdall mitigates JDBC UTF-8 parsing issues by automatically converting binary payloads into hexadecimal literals, allowing the database to safely interpret the exact original bytes.	Boolean	false
gcOverheadThreshold	-	Defines the threshold for GC overhead. If the percentage of total CPU time dedicated to garbage collection exceeds this value, an alert is triggered suggesting an increase in the maximum heap allocation. Default value is 20, which means alert will be triggered when cpu spends more than 20% on garbage collection.	Integer	20

Status Overview

Status	Virtual Database	Servers										Development		
✓	PostgreSQLVDB :5433 Proxy starts: 6		Status	Data Source	Server Name		Last Heartbeat Time		Queries/Sec.	Connections	Replica Lag (ms)			
		✓		PostgreSQLDataSource	Write		2025-08-29 16:00:08 (+0200)		0	0	-			
		✓	≡	PostgreSQLDataSource	Read		2025-08-29 16:00:08 (+0200)		0	0	8			
			Status	Client IP/Hosts (driver version/id)			CPU	Queries/Sec.	Last Connect Time	Connection Count	Average Connection Lifetime [s]	Uptime	Licensed	
		Query Distribution:	✓	ACAI-LD3VDLLL3 25.08.29.0-20250829143511/3dd707			0.417 %	0	2025-08-29 16:00:07 (+0200)	0	0	00:00:20	☐	≡
		Writer: 0% Readers: 0% Cache: 0%												
			Status	Cache type	Cluster info							Uptime		
		Connection Reuse: 0.50x												
			✓	Hazelcast	Nodes							Owned Partitions		
			Member [172.19.0.1]:5703 - 958f6be9-5569-4a82-99d6-714ce6215bf5							Partitions: 135	00:03:19			
			Member [172.19.0.1]:5701 - ca6fe937-5b37-4423-80b8-fb7a70b6ea0c							Partitions: 136				

The status page provides a quick view of what VDBs are configured, what servers are attached to them, and their current health status, along with what clients are connected to them.

For each VDB, the data sources will be shown, the last HB time an update about that server was received, the current qps, connections and if configured, replica lag is.

For the clients, the pin option allows a given client to be saved and always viewed on the status dashboard. This is to allow the current state of clients that only periodically to be viewed. If the pinned option is not set, then clients not seen for over five minutes will disappear from the status view.

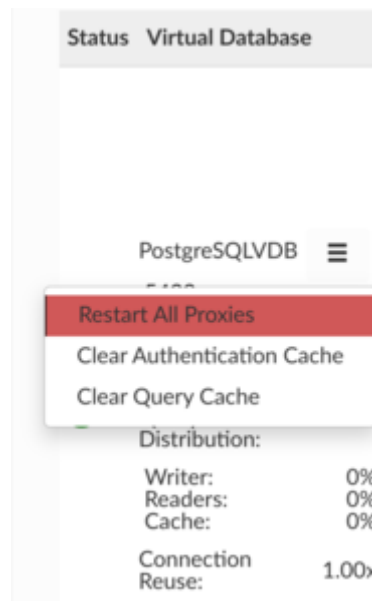
Note: A client MAY show as red if it is using connection pooling, despite traffic being active from the client.

If in the VDB configuration **Cluster manager via cache** is enabled and cluster cache has been used (i.e. Hazelcast or Redis working in cluster mode), the cluster information table will appear, showing cache type, status, list of nodes, partitions owned by each node and the cache uptime.

Note: The Hazelcast instance on the Heimdall Manager should not be responsible for owning any partitions; rather, all partitions ownership will given to the Heimdall proxy nodes.

VDB Actions

When the collapsed menu button is clicked, there should be displayed additional actions showed in the image below.



Below are some actions and details what each of the button is used for:

- **Restart All Proxies:** Restart all proxy instances for VDB, on the management server only.
- **Clear Authentication Cache:** Clear all authentication related Caches (users, passwords, AD groups etc.).
- **Clear Query Cache:** Clear Query Cache, if using a shared cache, this may clear all objects in the cache.

Proxy Actions

When the collapsed menu button is clicked, there should be displayed additional actions showed in the image below.

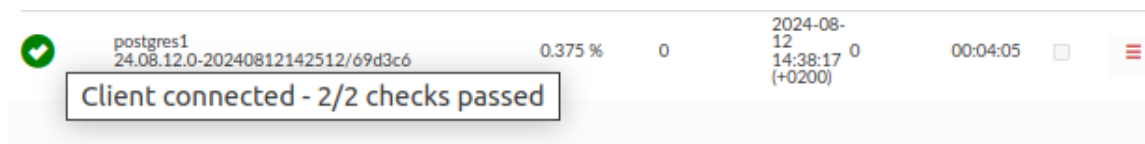
Pin	Status	Client IP/Hosts (driver version/id)	CPU	Queries/Sec.	Last Connect Time	Connection Count	Uptime	Licensed	
☐		postgres0 22.08.08.1-20220808205003/f9e0b9	0.250 %	0	2022-08-16 17:21:18 (+0000)	0	184:30:03	☐	
minutes.									
Clear credentials properties Heap dump Stack trace Packet capturing Terminate Process Network Disconnect									

Below are some actions and details what each of the button is used for:

- **Clear Credentials Properties:** It will clear the credentials properties cache for a specific proxy. It's usefully when the SQL driven authentication is enabled.
- **Heap dump:** It will make a proxy heap dump, which will be available in Logs tab. You will need jmap to perform this action

- **Stack trace:** It will make a proxy stack trace, which will be available in Logs tab. You will need jstack to perform this action
- **Packet Capturing:** You can initiate a timed packet capture on specific Proxy nodes. This includes limiting duration, total number of files, and the size limit for the capture. Port 8087 and 8443 will be filtered out of the capture automatically.
- **Terminate Process:** It will terminate and restart proxy process - if the proxy is managed by central manager. Otherwise, if you have an independent proxy running, proxy process will be terminated (not restarted).
- **Network Disconnect:** It will disable the network interface of proxy simulating its unavailability. To enable proxy again turning the interface on manually and restarting the proxy is required. Superuser privileges required. Does not work on local proxies, as it's the same as central managers.

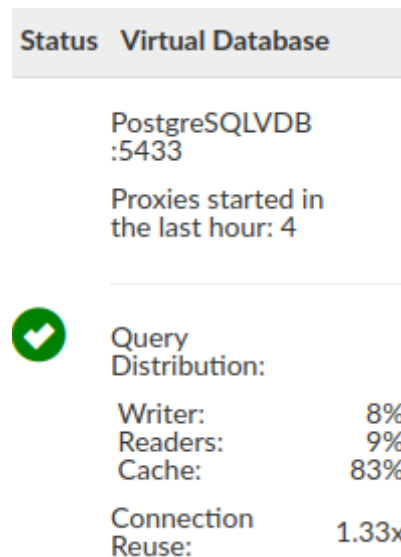
Note: When the Auto-Scaling Mode is enabled in [Advanced Features](#) and the VDB has multiple proxy nodes, then the **Promote to active state** option is available. With this option, it is possible to choose which node is active.



The status color for the proxy is handled by the double-check mechanism. This means that for the proxy to turn red, the last two checks must fail. This eliminates issues such as temporary network overload and prevents the proxy from flickering if the connection problem between the

proxy and the central manager is only temporary. Hovering over the status ball will display this information.

Query Distribution



The Query Distribution part of VDB preview provides a quick view of some data aggregated during the proxy runtime. You need to enable certain options to view some of them.

- **Writer** - percentage of all queries that did go to writer.
This statistic is enabled by default.
- **Reader** - percentage of all queries that did go to reader due Read/Write split.
To enable this statistic Read/Write Split must be enabled and configured properly.
- **Cache** - percentage of all queries that hit the cache.
To enable this statistic Caching must be enabled and configured properly.
- **Connection Reuse** - A ratio of how many connections were opened on the front vs on the back.
To enable this statistic connection pooling must be enabled and configured properly.

These statistics can be reset by restarting the proxy in the Status tab.

Fields on status tab:

Status and Virtual Database

Status data is refreshed every 5 seconds.

- **Status** - It's meaning is dependent on context - virtual database, server, client and cache - you can get more information when you hover your mouse over them. But overall meaning is:
 - **Green check** - Ok - Working correctly
 - **Blue check** - Standby
 - **Yellow warning** - Warnings
 - **Red X** - Failed
 - **Question mark** - Status unknown
 - **Virtual Database**
 - **Vdb name** - here we can see its name and port its working on.
 - **Proxy starts** - we can see amount of proxy starts/restarts in last hour.
 - **Hamburger button icon** see [VDB actions](#)
 - **Query Distribution** - percentage of all queries that go to writer/reader or hit the cache, for more information see [query section](#)
-

Servers

Under this section we will see our data sources, clients/hosts and our distributed cache if it's used. At the end of our table there is **Deployment Environment** written.

- **Data source**
 - **Status** - See status section or hover over it for extra information.
 - **Data Source** - It's the name of our datasource.
 - **Server Name** - Either name of the datasource or server name if load balancing is used.
 - **Last Heartbeat Time** - Last time when heartbeat was done.
 - **Queries/Sec.** - Amount of queries per second calculated from last 5 seconds.

- **Connections** - Current connections amount for given server.
- **Replica Lag (ms)** - Requires **Track replication lags** in Data source / load balancing section. Tracks replication lag. - sign on status means not used. If it is -1 or 0 in jmx it is not used.
- **Client/Hosts**
 - **Status** - See above and hover over it for extra information.
 - **Client IP/Hosts (driver version/id)** - Ip or hostname followed by driver version with first 5 signs of driver id.
 - **CPU** - Current usage of cpu in %.
 - **Queries/Sec.** - Amount of queries per second calculated from last 5 seconds.
 - **Last Connect Time** - Last time our host was on. Useful when we want to see when our host went off.
 - **Connection Count** - Current connection count to our host.
 - **Average Connection Lifetime[s]** - Five minutes rolling average of each minute average open connections lifetime between client's app and heimdall (in seconds). It means that in each minute, we take all active connections and calculate the average for them. Only last five minutes snapshots are taken to calculate the roll average. Highlighted in red if greater than 300 seconds.
 - **Uptime** - How long the proxy is running since last started/restarted.
 - **Licensed** - Is the proxy or driver running on licensed marketplace instance.
- **Distributed cache** - shown only if used.
 - **Status** - See above and hover over it for extra information.
 - **Cache type** - Below we can see name of used cache. E.g. hazelcast.
 - **Cluster info**
 - **Nodes** Information about node usually address with uuid.
 - **Owned Partitions** amount of owned partitions.
 - **Uptime** - How long the cache is running since last started/restarted.

How to get status tab information from JMX

This section shows how you can get status tab data from JMX - more information about setting it up and getting information can be found [here](#).

⚠ **Note:** Statistics in JMX are sometimes calculated differently, they differ in refresh rate and will be reset on any configurations changes or proxy restarts.

Why some data may differentiate over time?

Jmx usually shows data from the start - since last restart or configuration changes - it allows to get whole set of data.

Status tab shows data that is time sliced - Old data is removed after certain period of time - that allows us to focus on current data and trends. Time slicing make it easier to analyze trends and detect anomalies.

We can get the statistics per seconds / minutes etc. that is not time sliced from jmx:

We can get our client's uptime (in ms):

/heimdall-management/JVMUptime and divide it by 1000 to get uptime in seconds.

We can use it to get any data per any unit of time. For example: We can divide TotalQueries / uptime in seconds to get Queries/Sec which will not be time sliced. It can be used to get any data per seconds / minutes / hours.

Status tab information and its equivalent from JMX

Http: Every call starts with `http://{healthcheck-host}:{healthcheck-port}/jmx` then we add from documentation our rest of the path.

For example if healthcheck host is localhost, and healthcheck port is configured to 8070 in [VDB Advanced features](#):

```
http://localhost:8070/jmx
```

JMX: Every call is composed of: `{domain}:type={component}[,name={instance}] {attribute}`

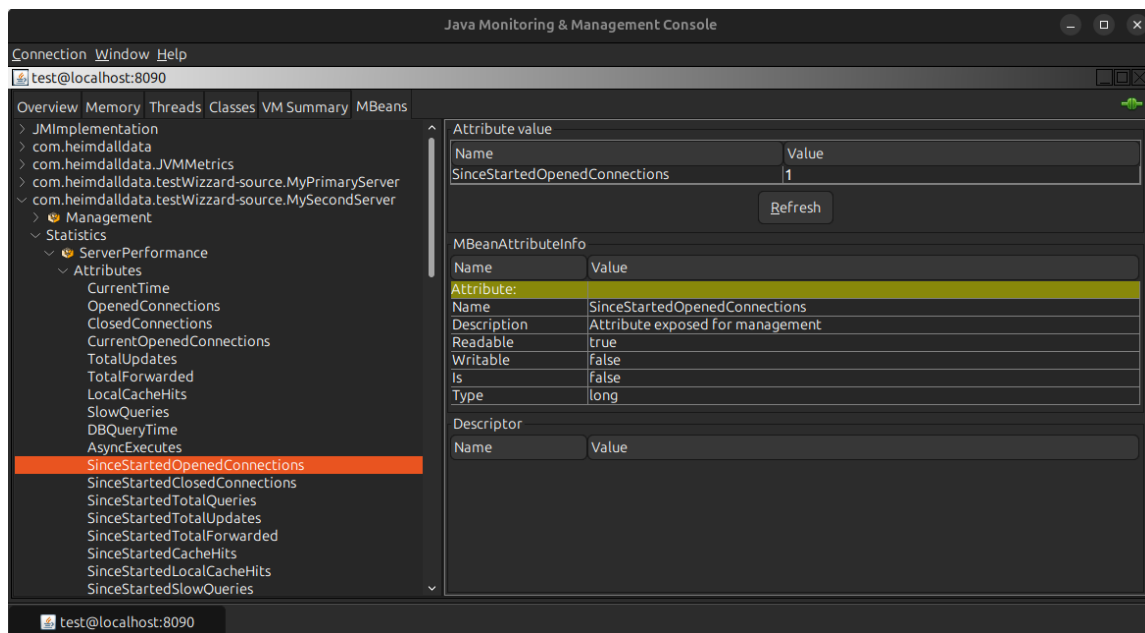
For example:

```
com.heimdalldata.testWizzard-  
source.MySecondServer:type=Statistics,name=ServerPerformance  
SinceStartedOpenedConnections
```

-----	-----	-----
		DOMAIN
COMPONENT	INSTANCE	ATTRIBUTE

Documentation starts with component E.g. [Statistics] followed by instance (if needed) and attribute, so without the domain - for better visibility and less clutter.

In jconsole it is seen as:



Calculating status data

- **Proxy starts** - We can't get this data from jmx as our jmx statistics resets with proxy restarts and configuration changes.
- **Query Distribution:** On status tab it shows latest usage - that means old statistics are eventually deleted and not counted at all, for more read [time sliced data](#). Here we can get whole data from the start:

- **Writer**

- Http: /heimdall-management/SinceStartedWriterQueryPercent
- JMX: Management SinceStartedWriterQueryPercent

- **Readers**

- Http: /heimdall-management/SinceStartedReaderQueryPercent
- JMX: Management SinceStartedReaderQueryPercent

- **Cache**

- Http: /heimdall-management/SinceStartedCacheQueryPercent
- JMX: Management SinceStartedCacheQueryPercent

- **Connection Reuse** used with connection pooling - can be calculated as number of *Total made connections / Created connections*.

```
HTTP:
/heimdall-management/ProxyConnections
divided by:
/connection-pool/Created
```

```
JMX:
Management ProxyConnections
divided by:
ConnectionPoolStatistics Created
```

SERVER

SINGLE SERVER FOR DATA SOURCE

Domain for the JMX with single server for data source it will be: com.heimdalldata.testWizzard-source.none

MULTIPLE SERVER FOR DATA SOURCE (LOAD BALANCING)

Domain for the JMX with Load balancing enabled it will look like:

com.heimdalldata.testWizzard-source.MyPrimaryServer

testWizzard-source is the name of our data source

While MyPrimaryServer is the name of the server from Load Balancing/High Availability section in data sources tab.

This data is per server while load balancing option is enabled.

- **Last Heartbeat Time** - either in ms or in year-month-day hours-minutes-seconds (timezone) (yyyy-MM-dd HH:mm:ss (Z)) format so as can be seen in status tab:

- Http:

- Status tab format: /server-stats/LastHeartbeatTimeAsDate
- Milliseconds: /server-stats/LastHeartbeatTime

- JMX:

- Status tab format: Statistics,name=ServerPerformance LastHeartbeatTimeAsDate
- Milliseconds: Statistics,name=ServerPerformance LastHeartbeatTime

- **Queries/Sec.** - On jmx we can get this statistic, but it will differentiate over time, for more information see [time sliced data](#)

- We can get total **Queries** and then divide them by uptime:

- Http: /server-stats/SinceStartedTotalQueries
- JMX: Statistics,name=ServerPerformance SinceStartedTotalQueries

If we want Queries per second we can get our client uptime (in ms): `/heimdall-management/JVMUptime` and divide it by 1000 to get uptime in seconds. Then we can divide `TotalQueries / uptime (in seconds)` to get Queries/Sec.

⚠ Note: This will be total Queries/Sec. which will not match our status statistic.

- **Connections**

- Http: `/server-stats/CurrentOpenedConnections`
 - JMX: `Statistics,name=ServerPerformance CurrentOpenedConnections`
 - **Replica Lag (ms)** If the current lag is showing **-1** or **0** (jmx) or - (status tab) it is not used.
 - Http: `/server-stats/CurrentLag`
 - JMX: `Statistics,name=ServerPerformance CurrentLag`
-

CLIENT

- **CPU**

- Http: `/heimdall-management/CpuLoad`
- JMX: `Management CpuLoad`
- **Queries/Sec.** Data will differentiate over time. For more information see [time sliced data](#):

```
Http:
/heimdall-management/ProxyQueryCount
divided by:
( /heimdall-management/JVMUptime / 1000 )

JMX:
Management ProxyQueryCount
divided by:
( Management JVMUptime / 1000 )

Simply: Queries / (uptime / 1000)
```

⚠ Note: This will be total Queries/Sec. which will not match our status statistic.

- **Last Connect Time**

- Http:
 - Status tab format: `/heimdall-management/LastConnectTimeAsDate`
 - Milliseconds: `/heimdall-management/LastConnectTime`

- JMX:
 - Status tab format: Management LastConnectTimeAsDate
 - Milliseconds: Management LastConnectTime

- **Connection Count**

- Http: /heimdall-management/CurrentClientOpenedConnections
- JMX: Management CurrentClientOpenedConnections

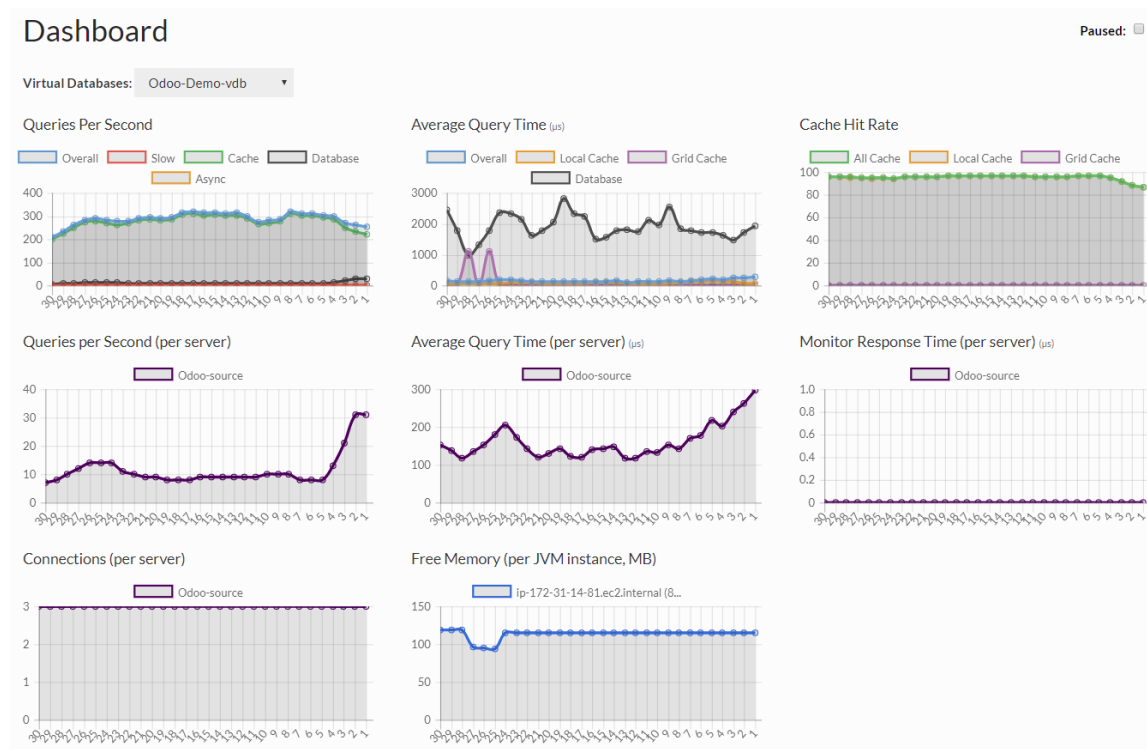
- **Average Connection Lifetime [s]**

- Http: /heimdall-management/ClientOpenConnectionLifetimeAverage
- JMX: Management ClientOpenConnectionLifetimeAverage

- **Uptime** either in ms or in Hours-minutes-seconds (HH:mm:ss) format so as can be seen in status tab:

- Http:
 - Status tab format: heimdall-management/UptimeInHours
 - Milliseconds: /heimdall-management/JVMUptime
- JMX:
 - Status tab format: Management UptimeInHours
 - Milliseconds: Management JVMUptime

Dashboard Overview



The dashboard provides live statistics for many key performance indicators, and can be filtered by VDB. In the case no vdb is selected, then the statistics will be for the resources available to be observed by a user based on their administrative filter, if any.

The statistics provided are:

- Queries per second, divided into Overall, Slow (taking longer than 10ms by default), Queries from cache, Database, and Async.
- Average Query time, divided into Overall, Local (L1) cache, Grid (L2) cache, and Database
- Cache Hit rate, divided into Overall, Local Cache, and Grid Cache
- Queries per Second (per Server), showing the overall actual query rate reaching each server
- Average Query time (per server), again showing the average of queries making it to a server
- Monitor response time (per server) showing the monitor query performance, including opening a connection, issuing the query, and closing the connection
- Connections (per server), showing the active or idle connections on each server node at the connection pool (if present) level or actually used.

- Free Memory (per JVM Instance, in MB), the amount of heap available for each node, to help track memory usage issues that may impact caching or the application when in JDBC mode

At the top of the screen, there is a pause button, which will pause rendering of all graphs.

If a particular item is clicked on in the legend of a graph, its visibility can be toggled, so as to help highlight items that may be obscured due to overlapping lines, and/or due to scale issues. The combination of the pause and the item toggle allows each graph to be explored in more detail.

Monitoring Overview

Analytics

Delete All Logs

Rows: 100

Virtual Databases: PostgresVD

Catalogs: All

Regex filter:

Errors Only: ☐

Analyze

Info	Rules	DB Usage Ratio (%)	Cache Hit (%)	Forwarded (%)	Count	Cache Time ms	DB Time ms	Overall Response Times ms	Result Retrieval Times ms	Average Result Size	Transaction (%)	Query
	+Q	100	0.0	0.0	1	0	1.5	1.5	1.2	5.0	0.0	select * from newtable n

The analytics tab data is provided to allow a per query-pattern view of the queries passing through the system, as logged with the Log action or with global VDB logging.

Filters

Before analyzing the data, one of several filters can be specified to control what data is viewed:

- **Rows:** The number of rows to display. Large numbers may reduce the performance of the browser. Note, if using the regex filter for something in a rare query, it may be necessary to set a large number of rows to get the data desired.
- **Virtual Database:** Drop-down selector of the VDB to filter on
- **Catalogs:** Drop-down selector of the VDB catalogs to filter on. It is available when any VDB is selected in the Virtual Database filter
- **Regex Filter:** A filter (done at the Javascript level) to narrow down the queries to observe. The rows must have been returned from the server for this filter to apply.

On clicking the Analyze button, the results will be returned, with the following columns:

- **Info:** This column may show one of several visual indicators. The green page indicates that a Query plan is available to be displayed for the query, and the red circle around an ! indicates that there is an exception registered against this query pattern.
- **Rules:** Only shown if VDB is picked in **Virtual Database** field. Provides some information and control over the current rules in effect. The blue + is used to add a new rule based on the query pattern, a green magnifying glass provides a list of rules that match, and the disk icon with a lightning bolt provides an indicator of the cache status of the rule, with green for an explicit regex match, and a red indicating explicit no-cache or a zero ttl.
- **DB Usage Ratio (%):** The percentage of time overall time waiting on the server represented by this particular query pattern. This column should add to 100% for all queries on a given VDB (rounding excluded).

- **Cache Hit (%)**: The percent of queries a particular query pattern has been served from cache vs. retrieved from the Origin.
- **Forwarded (%)**: The percent of queries a particular query pattern has been forwarded vs. retrieved from the Origin.
- **Count**: How many times this query pattern has been observed.
- **Cache Time**: The average time it took for a cache hit to be pulled from the cache
- **DB Time**: The average time it took for a cache miss to be pulled from the database
- **Overall Response Time**: The average time it took for all queries to be served to the client, blending the cache hit and miss times
- **Result Retrieval Time**: Like the Overall Response Time, but measured from the time the query is sent to the server to the time the result-set is closed. This can include additional time spent by the application processing the result-set, and may be significantly longer than the Query Response Time for very large responses.
- **Average Result Size**: The average result-set retrieval size for this query pattern. Large results generally correlate to larger query times.
- **Transaction (%)**: The percent of time this query pattern was executed in a transaction, i.e. not in auto-commit mode, and/or inside of an explicit transaction. In general, for most applications, this will be 0 or 100%, but in some cases where the same query is executed both inside and outside of a transaction, it may show other values. This can be used to diagnose frameworks that put 100% of queries into a transaction, even when not necessary.
- **Query**: The query pattern itself. Note--when hovering over the query, a pop-out icon will present itself in the upper right-hand corner, and is used to access the extended query view.

Note: Time units for any appropriate columns can be selected in the display itself, to scale as necessary, so are not documented here.

Extended Query View

Each query, when the mouse is moved over, will provide a small icon in the upper right-hand corner. If selected, extended information may be provided including:

- The query statistics for the query;
- A formatted view of the query;
- The tables detected for this query;
- Any exceptions that have been observed for the query;
- Any saved query plans for this query.

This view is intended to be useful to provide to developers to help diagnose any performance problems that are apparent with the query pattern.

Viewing Errors

The check-box "errors only" will result in a view that only contains queries that have SQL exceptions associated with them.

Logs Overview

Logs									
Filter by type: DONE_SQL DONE_SQL Filter by query run-time: 0 Rows: 30000 Virtual Databases: postgres-vdb Regex filter: SELECT Get Logs									
Date (+0100)	Client ID	VDB	Source	Type	Status	Run Time (µs)	Message		
12/05/2023 11:41:41.844	ip-172-31-54-43/172.17.0.2(c8251)	postgres-vdb:5434	PSQLDS	DONE_SQL	0	910	SELECT * FROM pg_catalog.pg_enum WHERE 1<>1 LIMIT 1		
12/05/2023 11:41:41.841	ip-172-31-54-43/172.17.0.2(c8251)	postgres-vdb:5434	PSQLDS	DONE_SQL	0	918	SELECT version()		
12/05/2023 11:41:41.768	ip-172-31-54-43/172.17.0.2(c8251)	postgres-vdb:5434	PSQLDS	DONE_SQL	0	1650	SELECT db.oid,db.* FROM pg_catalog.pg_database db WHERE dbname=?		
12/05/2023 11:41:41.778	ip-172-31-54-43/172.17.0.2(c8251)	postgres-vdb:5434	PSQLDS	DONE_SQL	0	805	SELECT current_schema(),session_user		
12/05/2023 11:41:41.773	ip-172-31-54-43/172.17.0.2(c8251)	postgres-vdb:5434	PSQLDS	DONE_SQL	0	2199	SELECT pg_type.oid, typname FROM pg_catalog.pg_type LEFT JOIN (select ns.oid as nspoid, ns.nspname, r.r from pg_namespace as ns join (select s.r, (current_schemas(false)) [s:r] as nspname from generate_series(1, array_upper(current_schemas(false), 1)) as s(r)) as r using (nspname)) as sp ON sp.nspoid = typnamespace WHERE typname = ? ORDER BY sp.r, pg_type.oid DESC LIMIT 1		

The log tab is provided to allow inspection of the actual detailed logs being processed by the system. The table contains the following columns:

- Date (with timezone indicator)
- Client ID: The ID of the client that generated the log entry
- VDB: The VDB (if any) the log is associated with
- Source: The data source associated with the entry
- Type: A categorization of the record type
- Status: An integer field to indicate if a query was in various states (please see the logging section for more details)
- Run Time: The time associated with the event that is logged, if any
- Message: The actual message contained in the log entry

Log download

Using the **Download Logs** button will download a zip archive with the following content:

- Files from the log directory - that includes log files, screenshots, and secret files used for decryption of TLS packet captures
- VDB configuration files
- Systemd logs of "run as service" proxies, from journald
- Database system variables from each enabled data source
- Packet capture files

Additionally, you will be prompted for an optional log encryption password. When supplied, the files from the log directory and systemd logs will require this password to open.

Please see the logging section for a more detailed breakdown of system logging.

Filtering options

- **Filter by type:** This option allows you to filter logs by type. Available types and when they are generated - [Logs types](#)
- **Filter by query run-time:** Filter the logs by their run time
- **Rows:** Amount of rows to limit to when searching logs
- **Virtual Databases:** Allows you to choose to display logs only from given virtual database
- **Regex filter:** Allows you to filter log messages by regex

Logs types

The "Filter by type" at the top of the "Logs" section allows us to filter generated logs depending on what are those related to.

The available logs types are listed below:

- **METHOD_COMPLETE:** Generated each time non-resultset JDBC method is called. "Log Methods" in VDB configuration needs to be enabled to generate such logs.
- **CONNECTION_OPEN:** Generated each time new connection is established. "Log Connections" in VDB configuration needs to be enabled to generate such logs.
- **CONNECTION_CLOSE:** Generated each time any connection is closed. "Log Connections" in VDB configuration needs to be enabled to generate such logs.
- **CONNECTION_HANDSHAKE_TRACE:** It needs **verbose debug mode** with **connection handshake tracing** options enabled in Virtual DB section of our vdb to be logged. Generated each time a new connection is established, this traces full handshake process. Includes network negotiation and authentication steps.
- **DONE_SQL:** Generated each time any query goes through proxy. "Log All SQL" in VDB configuration needs to be enabled to generate such logs.
- **TRACE:** Generated with trace actions.
- **TRANSACTION_START:** Generated each time transaction is started.
- **TRANSACTION_END:** Generated each time transaction is committed or rollback.
- **TRANSACTION_SET:** Generated each time a transaction is started with non-default settings.

- **READ_ONLY_SET**: Generated each time a transaction is explicitly marked READ ONLY;
- **DEBUG**: Provides additional logs generated in rare cases. "Verbose Debug Mode" in VDB configuration needs to be enabled to generate such logs.
- **SQL_EXCEPTION**: Generated each time SQLException is thrown.
- **INTERNAL_EXCEPTION**: Generated each time any other exception is thrown.
- **NEW_PATTERN**: Generated when observing a new query pattern. "Learn Pattern" rule needs to be set to generate such logs.
- **NOTIFY**: Generated for rules tagged with a "notify".
- **CONSOLELOG**: The most basic type of log in Heimdall.
- **NOCACHE**: Generated when cache revalidation fails.
- **PROXY_START**: Generated each time any proxy is started/restarted.
- **AUTHENTICATION**: Provides information about each user authentication attempt. "Log Authentications" in VDB configuration needs to be enabled to generate such logs.
- **ERROR**: Generated when something unexpected or unwanted happened, but no exception was thrown.
- **SYSTEM_STATUS**: Generated for Heap Dumps and Stack Traces.
- **METRICS_DISCOVERY**: Generated for JMX statistics propagation between central manager and proxy.

Log File Rotation

When a log file is rotated, as controlled by settings in the admin->system properties, it will execute a command (if present) named `rotatelog.sh`. A very simple example would be:

```
#!/bin/bash

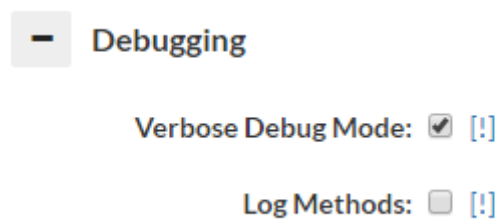
tar cvzf $1.tgz $1
```

This command can execute other code, such as to transfer the file to S3 or another location. Please note that on a restart, there is no guarantee that a file will be rotated, so periodic cleanup may be needed.

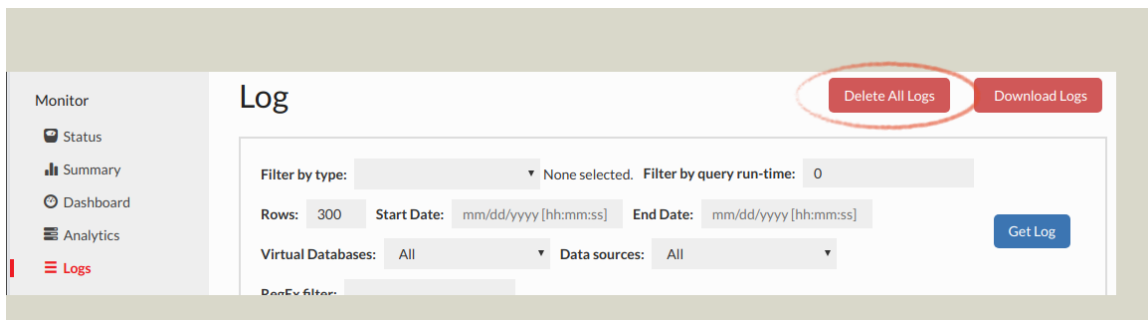
Debugging Overview

If there are issues that require Heimdall support assistance, please follow the following directions (as appropriate)--please note, most debugging should be done in low-traffic scenarios for best results.

- If the queries being made are particularly sensitive, enable "paranoia" mode in the vdb->advanced advanced features section (this will hide the actual SQL being generated, so should only be enabled if necessary).
- Enable verbose debugging on the VDB experiencing the issue (Note, this will add some extra load on the proxy and central manager, in particular if under high load)



- Delete the logs prior to reproducing the issue:



If a packet trace is requested, please follow one of the next two steps as appropriate:

- **On Linux/Mac/Unix systems:** If requested, please capture packets on the system the proxy resides on, including the ports for both the proxy port and the port the database resides on.

Using tcpdump (on most Unix-like system):

```
tcpdump -i any -s0 -w capture.pcap tcp port <proxy port> or tcp port <database port>
```

If both ports are the same, i.e. 1433 for SQL Server, then only one port is necessary, i.e.:

```
tcpdump -i any -s0 -w capture.pcap tcp port <proxy port>
```

Note: capture.pcap will be the filename saved—it may be necessary to download this using a tool such as scp, winscp (<https://winscp.net/eng/download.php>), or filezilla (<https://filezilla-project.org/>) in order to e-mail this file

- **On Windows:** Install Wireshark or a similar packet capture tool (<https://www.wireshark.org/download.html>) and start a packet capture for the proxy and database ports. Note: If the proxy is on the same system as the application and/or database, then capturing the "loopback" interface is necessary. Please contact support for assistance if necessary with this.
- Reproduce the problem, preferably with a minimum amount of traffic being generated, and note the time the reproduction was successful, including timezone, to include in the problem report
- If necessary, terminate the packet capture (CTRL-C for tcpdump, or the stop button in Wireshark)
- Disable debug mode in the VDB
- Download the logs from the Log or Analytics tab, and e-mail them and any packet capture generated to [Heimdall support](#), and include a description of the problem, any screenshots that are relevant of the behavior. If the traces or logs are too large (generally ~20MB), please use a file sharing site such as Google Drive or Dropbox to share a link to the files for analysis.

Heap Dumps

At times, if there is a memory utilization issue, Heimdall support may ask for a live or full heap dump of the server or the proxy. Here are the steps to perform this:

1. Connect to the Heimdall instance command line as root, typically via Putty or SSH
2. use "ps -ef | egrep "heimdallserver.jar|heimdalldriver.jar" to find the pids of the processes
3. For a full dump, use the command: "jmap -dump:file=heap-full.hprof "
4. For a live dump, use the command: "jmap -dump:live,file=heap-live.hprof "
5. Download and provide the files to Heimdall support as appropriate based on size, i.e. via a file transfer site, or e-mail

The difference between a live and full dump is that a live dump only shows objects that are not pending cleanup. Both may be requested, so we can see what memory is being taken by objects yet to be cleaned up.

Stack Trace

If a Heimdall process is getting "stuck" at any point, which could be from a logic deadlock or similar issue, a stack trace can be used to find where in the code the lock is happening. To gather a stack trace, here are the steps:

1. Connect to the Heimdall instance command line as root, typically via Putty or SSH
2. Use `"ps -ef | egrep "heimdallserver.jar|heimdalldriver.jar"` to find the pids of the processes
3. Use `"jstack > stack.txt"`
4. Download and provide the file to Heimdall support

TCP Keepalive

In order to control idle connections better and prevent NLB timeouts, Heimdall activates TCP keepalives, and sets the keepalive behavior to be one keepalive every 300s while idle. This should be under the 350s that is hard-set for AWS NLB. In other environments, the idle timeout should be set higher than this if keepalive is desired to keep alive connections, and under this value if idle connections should be culled.

Cache Debugging

One of the most common issues when implementing Heimdall is the lack of cache hits. There are many different reasons why with a default cache policy as generated by the configuration Wizard, cache hits will not be observed. Each of these will be detailed below and how to diagnose them.

Show queryinfo

When using a tool that performs simple queries (i.e. not a prepared statement query) such as psql, after a query is executed, the command "show queryinfo" can be executed, which will provide information about the last query, and how it was processed, specifically the properties that were attributed to that command. An example output:

```
gpadmin=> show queryinfo;
```

Attribute	Value
cache:ttl	3600000
all:stop	false
all:resultQueryTrans	true
all:querytimeout	0
all:printtiming	false
all:printtables	false
all:printresults	false
all:printmeta	false
all:printmatch	false
all:printcapture	false
all:onlytrans	false
all:olderthan	0
all:notrack	false
all:maxburst	10
all:logger	hdlog
all:log	true
all:invalidate	true
all:capture	true
nocache reason	cache is not enabled
connid	13
command	select 1
cache	true
Autocommit	true

(24 rows)

Here, we see the "nocache reason" is set to "cache is not enabled", which is the first reason encountered that prevented caching of this query. If we resolve this issue, then we get:

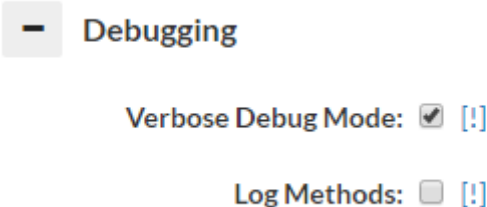
```
gpadmin=> show queryinfo;
      Attribute      |      Value
-----+-----
...
nocache reason      | empty table list
...
```

Here, an empty table list is encountered, which indicates that caching was not enabled because no table names were extracted from the query, which defaults to disabling caching. See below for a list of nocache reasons.

General Debugging

First off, when debugging caching, the following steps should be followed:

1) In the vdb tab, enable the "verbose debugging" option:



2) Next, pass traffic you expect to be cached, and find the query hash value in the log tab (each unique query will have a unique hash):

Log

Delete All LogsDownload Logs

Filter by type: None selected. Filter by query run-time: 0							
Rows:	300	Virtual Databases:	All	Regex filter:	02D491		
Date (-0400)	Client ID	VDB	Source	Type	Status	Run Time (us)	Message
10/02/2020 11:55:12-22:23	ip-172-31-54-43 [c8251]	Odoo-vdb		CONSOLELOG	0	0	[2020-10-02 15:55:12.223] Cache:02D4913DB766EC2D371B55C7DD197FC8 Put into local cache (binary)
10/02/2020 11:55:12-22:22	ip-172-31-54-43 [172.17.0.2 c8251]	Odoo-vdb		CONSOLELOG	0	0	[2020-10-02 15:55:12.222] C49451:T5516251:230986412:02D4913DB766EC2D371B55C7DD197FC8:Cache:Putting object in cache
10/02/2020 11:55:12-22:22	ip-172-31-54-43 [172.17.0.2 c8251]	Odoo-vdb		CONSOLELOG	0	0	[2020-10-02 15:55:12.222] C49451:T5516251:230986412:02D4913DB766EC2D371B55C7DD197FC8:Query: explain=true, Autocommit=false, cachettl=5, allprintresults=false, allprinttimeouts=0, Schema=public, forwardreadyonly=true, Source=Odoo-source=Secondary, allprintrimming=false, CacheHit=all, invalidatedate=, allprintrcapture=false, RespTime=952, Tables=[odoo.public.ir_attachment], logswindowtime=0, allcapture=true, connectionIdleTimeout=60000, allstop=false, cache=true, DML=false, allprinttables=false, allnoack=false, forwardsource=, allprintrmatch=false, noack reason=stale entry, ts:1601654105781 currentTime: 1601654112221, Catalog=odoo, allmaxburst=10, forwardsetschema=true, alionlytrans=false, connid=49451, alllogger=hdlg, allolderthan=0, fwdSource=Odoo-source=Secondary, forwardscatalog=true, allprintrmeta=false, delayCacheInnt=0, alllog=true, reader eligible[cache=false]
10/02/2020 11:55:12-22:21	ip-172-31-54-43 [172.17.0.2 c8251]	Odoo-vdb		CONSOLELOG	0	0	[2020-10-02 15:55:12.221] C49451:T5516251:1861010376:02D4913DB766EC2D371B55C7DD197FC8:QueryPattern: 1A3CF0F97E733CBDCB08571B830944164245335417816456455 FROM "ir_attachment" WHERE (((("ir_attachment"."active" = true) AND ("ir_attachment"."res_model" = 'product.template')) AND ("ir_attachment"."res_field" = 'image')) AND ("ir_attachment"."res_id" IN (?) ORDER BY "ir_attachment"."id" DESC
10/02/2020 11:55:12-22:21	ip-172-31-54-43 [172.17.0.2 c8251]	Odoo-vdb		CONSOLELOG	0	0	[2020-10-02 15:55:12.221] C49451:T5516251:1861010376:02D4913DB766EC2D371B55C7DD197FC8:LagCheck:maxlag: 0 Time Window: 10000 lastUpdate: 1601440460265 Result: tables [odoo.public.ir_attachment]
10/02/2020 11:55:12-22:21	ip-172-31-54-43 [172.17.0.2 c8251]	Odoo-vdb		CONSOLELOG	0	0	[2020-10-02 15:55:12.221] C49451:T5516251:1861010376:02D4913DB766EC2D371B55C7DD197FC8:Forwardsource: 230986412:Odoo-source Server: Odoo-source=Secondary Con Time: 0 Reader Eligible: true setCatalog: true setSchema: true

3) Finally, check in the logs tab (or the log files on disk), and search for "nocache reason", i.e. using a regex filter of "3730569B89BF4674DE5549AF0C82A083.*nocache reason":

Log

[Delete All Logs](#)
[Download Logs](#)

Filter by type: None selected. Filter by query run-time: 0

Rows: 300 Virtual Databases: All 02D491

Regex filter:

[Get Logs](#)

[illegible]

The nocache reason will provide a simple explanation in most cases of why a particular request was not served from cache. Below are some of the cases and details on how to trigger caching, as appropriate (this is not guaranteed to be a comprehensive list):

- **No "no-cache" reason provided at all:** If no cache rule is present or matching, than no no-cache reason will be provided
- **multiple result sets:** In this case, a single query is generating multiple results, which is currently non-cacheable. Resolution: none
- **Output Parameters returned:** A called stored procedure is called with a return result. As this is likely to have side-effects, it is non-cacheable. Resolution: none
- **cursor execute|fetch|open...:** SQL Server cursor execute result. As this may be scrollable, it is non-cacheable. Resolution: none
- **Explicitly nocache per rule:** A no-cache rule is configured. Resolution: Remove or adjust the nocache rule
- **System table, without cache override:** A Database internal table has been detected, and is deemed to be non-cacheable, as it may change behind the scenes without notice. Resolution: On the cache rule, you can specify the "cachesystem" property with a value of true to enable caching.
- **empty table list:** If a query doesn't have a detected table associated with it, then it won't be considered cacheable. Example "select 1". Resolution: To enable caching such queries, create a rule with the property of "tables" and specify "none", which will provide a special behavior of instructing the system to allow caching anyway.
- **dml:** If Heimdall detects DML or otherwise doesn't flag it as NOT DML, then this will be set. This may be due to it being a "select...for update" query, for example. Resolution: To override the DML flag, the property of "update" and with true or false, depending on if it should be considered a DML operation or not.

- **Result-set properties disallow caching (not forward only or (scroll-insensitive && read_only))**: Certain result-sets may not be compatible with caching, due to scrolling or updatable flags. Resolution: none
- **cache is not enabled**: The cache is not enabled at the VDB level. Resolution: Enable the cache
- **vdb reset timer**: Via the UI or API, a timer was set to disable caching until a particular time. Resolution: Wait until the time expires
- **table reset timer (internal)**: A DML has triggered a cache invalidation interval on this proxy. If a table is written to more than once every two seconds, this may be a continuous issue for the table. Resolution: Wait until the time expires or stop writes to the table at issue.
- **table reset timer (external)**: Like with the internal invalidation, but due to invalidations from an external source, i.e. another proxy.
- **SQL contains (), use unconditional flag to override this logic**: Detected () in the SQL query. To override this logic, select the action cache and set unconditional flag as true for the given regex.
- **sequence usage detected**: Sequence usage has been detected. This logic can be overridden with unconditional flag in rules configuration.
- **temp table**: A temporary table was detected as part of the query. Resolution: Avoid using temporary tables.
- **non-deterministic function call**: A function was detected as part of the query that was determined to be non-deterministic by the database or cache engine. Resolution: Avoid using non-deterministic functions for queries that need to be cached.
- **nocache reason in the query, use unconditional flag to override**: Certain patterns in a query will trigger nocache behavior, such as "sql_calc_found_rows" and "()". To cache such queries, the property of "unconditional" can be set to true to bypass these checks.
- **exception creating fixed cached rowset impl**: An unexpected error was encountered creating a copy of the result-set
- **grid cache is null**: For some reason, the grid cache is not enabled. This may indicate a bug or other error. Consult Heimdall support for assistance
- **grid cache is not ready**: This may indicate that something is wrong with the cache engine selected, i.e. Redis is selected, but the cache server is unavailable.
- **no cached object found for key**: In general, this indicates that there was no cached object for the query. Resolution: retry the query again
- **No matching cache rule or nocache rule was applied**: This may indicate that there are no matching rules that will allow caching. Resolution: Create and apply caching rules or modify an existing one to match the requirements.

- **Cached object removed due to table eviction of table, ts=:** There was a cached object, but the table has had an eviction event, and is being evicted.
- **stale entry:** The cached object has surpassed its allowed TTL.
- **global cache reset since stored:** On the UI or via API, the clear-cache call was made, triggering a global eviction of objects. Any objects pre-dating this event will be considered stale.
- **ttl value set to 0:** A ttl value was set to zero for a cache rule

Queries in Transactions

As queries in a transaction may have different side-effects than those that are not, so, by default, the wizard creates a rule that does not trigger caching or cache retrieval while in a transaction.

This cause will NOT have a nocache reason (it simply won't be shown), as it is triggered in the rule match process itself.

As queries in a transaction may have different side-effects than those that are not, by default, the wizard creates a rule that does not trigger caching or cache retrieval while in a transaction.

System Table Detected

Certain table names such as pg_* will be flagged as being system tables, and which may be changed behind the scenes by default. In these cases, the default behavior is to disable caching. This can be overridden with the cache parameter of "cachesystem".

No Tables Detected

As invalidation is tracked at the table level, if no table is detected in a query, the default behavior is to disable caching. This includes for queries such as "Select 1". This can be overridden by tagging the query with a table of "none". This is a special value, in that it explicitly removes this tracking behavior.

Table Was Written To

As a DML is detected, the table will enter an invalidation window, which is a period of two seconds during which neither cache hits nor cache stores will occur. This includes for the entire duration of a transaction the DML is detected within, until a commit or rollback. Once the DML is completed, the invalidation window will no longer be in effect, however, for remote nodes, this window will occur for at least the full two second period. This is to reduce the

number of messages traveling between nodes. Updates to refresh the invalidation window will occur every second until no dml is pending against the table. Updates can be made that don't trigger an invalidation window by using the parameter "invalidate" with a value of false.

Temporary Table Detected

When a temporary table is detected in a query, this will suppress caching as well.

Non-Deterministic Function Call Detected

With SQL Server, function calls are parsed, and non-deterministic functions are detected. When present in the query, this will trigger cache suppression.

Debugging read/write

This page describes reasons, why Heimdall might not use reader servers, when load balancing is enabled.

Show queryinfo

When using a tool that performs simple queries (i.e. not a prepared statement query) such as psql, after a query is executed, the command `show queryinfo` can be executed, which will provide information about the last query, and how it was processed, specifically the properties that were attributed to that command. An example output:

```
dbConsole => show queryinfo;
```

Attribute	Value
forward:source	MySQLDataSource
cache:ttl	300000
all:stop	false
all:querytimeout	0
all:printtiming	false
all:printtables	false
all:printresults	false
all:printmeta	false
all:printmatch	false
all:printcapture	false
all:onlytrans	false
all:olderthan	0
all:notrack	false
all:maxburst	10
all:logger	hdlog
all:log	false
all:invalidate	true
all:capture	true
noreader	forward:readonly is set to false, can't use reader
nocache reason	cache is not enabled
connid	5
cache	true
autogenKeys	true
Tables	[]
Source	MySQLDataSource-Primary
Schema	empty
RespTime	1779
DML	true
Catalog	myDb

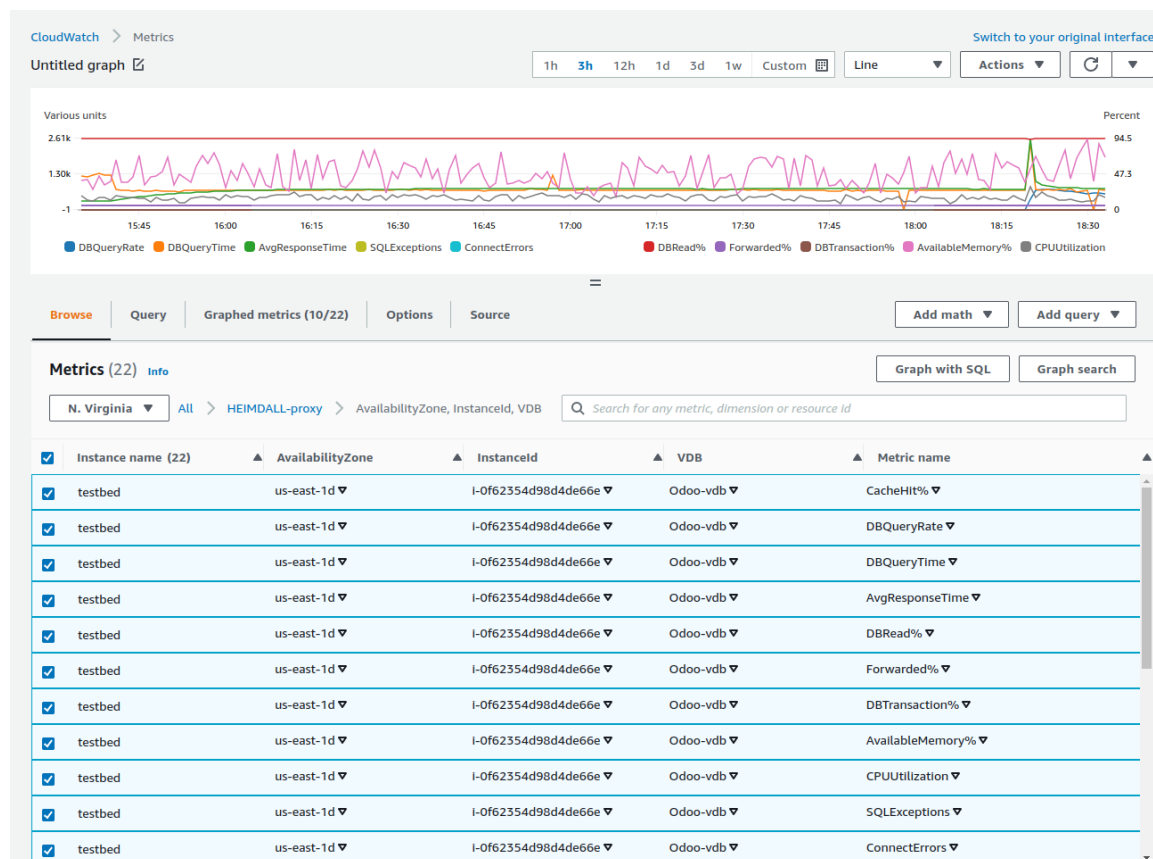
CacheHit	miss
Autocommit	true

noreader

noreader can have a few causes, they are explained in more detail in list below:

- **forward:readonly is set to false, can't use reader:** When property forward:readonly is set to false, it means that currently Heimdall is connected to writable server.
- **No servers available, cannot forward query to reader.:** There aren't any servers available in Data Source at that moment in time, therefore reader server is not being used.
- **Couldn't get Data Source while forwarding query, reader can't be used.:** Error occurred while retrieving Data Source to forward query. It's good reason to verify existence of chosen Data Source.
- **Connection is set to null, reader can't be used.:** Connection to Data Source couldn't be established.
- **Reader can't be used, exception on setting catalog on forwarded connection.:** Exception on setting catalog on forwarded connection, check catalog/schema on current/parent connection.
- **Statement is null, query cannot be forwarded.:** Forward statement couldn't be created, therefore reader cannot be used.
- **Exception on existing connection, reader can't be used.:** Exception on existing connection, forwarding interrupted, reader is not in use.
- **Connection is set to null, reader can't be used.:** Exception on closing connection existing connection.
- **Statement is DML/DDDL type, reader can't be used.:** Statement provided to the reader is DML/DDDL, then it cannot be used on the readonly database.
- **Couldn't connect to source, cannot forward query, check data source in vdb configuration.:** Heimdall couldn't establish connection to chosen Data Source, therefore query cannot be forwarded and reader is not in use. Please validate vdb configuration.
- **Couldn't connect to source, cannot forward query.:** Exception on connecting to source, query cannot be forwarded thus reader used.
- **Server isn't reader eligible, reader can't be used.:** Current server isn't reader eligible, so Heimdall won't even try using reader.

CloudWatch



When in AWS, and with cloudwatch integration enabled, various KPI indicators are provided to allow monitoring and alerting

Enabling CloudWatch Metrics from Private IP connections

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/cloudwatch-logs-and-interface-VPC.html>

KPIs

The following metrics are provided:

- **Available Memory:** The percentage of memory allowed to be allocated by the proxy that is available for use. In general, this can change over time dramatically, but if lower than 20% on average for a five-minute interval, an alert should be generated;

- **CPU Usage:** The percentage of cpu time used by this proxy. While multiple proxies may make this metric harder to gauge, in a single proxy environment, if this is above 70% for more than five minutes, an alert should be raised;
- **DB Transaction Percent:** The percentage of queries that are considered to be in transactions;
- **Cache Hit Percentage:** The percent of queries that are cache hits. This will vary for each customer, but can be used to set an alarm if caching drops below an expected value;
- **DB Query Rate:** The rate that queries are made against the database, i.e. are not cached;
- **DB Query Time:** The average response time from the database. This can be used to generate alerts if above the expected SLA;
- **Average Response Time:** The average response time for all queries, cached or not. This also can be used to generate alerts if above an expected SLA;
- **DB Read Percentage:** The percent of queries that are reads, which is a pre-requisite for caching or read/write split.
- **SQL Exceptions:** The number (per second) of SQL exceptions generated as a result of executing queries.
- **Connection Errors:** The number (per second) of connections errors.
- **Forwarded Percent:** The percent of queries that are forwarded via a forward rule, or via read/write split. A query may still end up on the writer node depending on configuration however.

Each of these metrics are reported on a per-minute basis, as part of the per-minute logging that is done with other logs to the cloudwatch log channel as well.

A dashboard widget that provides critical CPU usage information with the following query (as of the June 18th build):

```
SELECT max(CPUUtilization) FROM "<specific proxy CloudWatch namespace>-proxy"
GROUP BY AvailabilityZone, VDB
```

(if no custom namespace is provided, default namespace will be HEIMDALL-proxy)

This will graph per AZ the top CPU using proxy's CPU load on a per-VDB basis. This allows at a glance the ability to spot if a single proxy is receiving too much load due to uneven distribution of traffic, potentially causing issues. Alerts can then be attached to this to notify operations staff of the issue for corrective action.

JMX Statistics Overview

JMX statistics describe proxy and its minor features performance. If proxy has healthcheck enabled, JMX metrics associated with specific proxy will be exposed to read.

```
{
  "jmxBeans": {
    "Server Performance: com.heimdalldata.PostgreSQL-datasource.Master:type=Statistics,name=ServerPerformance": {
      "TotalCacheTime": "0",
      "AverageTotalCacheTime": "0.0",
      "StddevTotalCacheTime": "0.0",
      "StddevSlowQueriesCount": "0.0",
      "TotalQueryTime": "0",
      "StddevTotalQueryTime": "0.0",
      "AverageLocalCacheTime": "0.0",
      "CurrentLag": "2147483647",
      "AverageDbQueryTime": "0.0",
      "SinceStartedAsyncExecutes": "0",
      "SinceStartedCacheHits": "0",
      "LocalCacheTime": "0",
      "LastActive": "1694003643689",
      "SinceStartedSlowQueries": "0",
      "StddevCacheHitsCount": "0.0",
      "StddevLocalCacheHitsCount": "0.0",
      "AverageTotalQueryTime": "0.0",
      "CurrentOpenedConnections": "0",
      "SinceStartedOpenedConnections": "1",
      "AverageTotalUpdatesCount": "0.0",
      "AverageClosedConnectionsCount": "0.0011904761904761904",
      "ClosedConnections": "0",
      "SinceStartedTotalQueries": "0",
      "AverageOpenedConnectionsCount": "0.0011904761904761904",
    }
  }
}
```

Endpoints

Healthcheck server handles several types of endpoints:

- **http://{healthcheck-host}:{healthcheck-port}/jmx** - retrieve all JMX statistics associated with specific VDB
- **http://{healthcheck-host}:{healthcheck-port}/jmx/{stats-type}** - retrieve specific group of JMX statistics (example: **/jmx/server-stats** will show data source servers performance stats)
- **http://{healthcheck-host}:{healthcheck-port}/jmx/{stats-type}/{stat-name}** - retrieve specific JMX statistic (example: **/jmx/server-stats/SinceStartedSlowQueries** will show SinceStartedSlowQueries statistic value for every data source server)

Available JMX Statistics Types

- **heimdall-management** - general Heimdall statistics (application/proxy)
- **data-source-server** - data source server configuration basic options

- **server-stats** - data source server performance - need to have Load Balancing enabled
- **connection-pool** - connection pool statistics - need to have Connection Pooling enabled
- **user-pool** - user pool statistics
- **jvm** - JVM hiccup and heap allocation rates

Redis CLI Validation Guide for Heimdall Cache Performance

This guide explains how to validate and troubleshoot Heimdall's cache behavior using the `redis-cli` tool. It applies to **AWS ElastiCache Redis** in both **Cluster** and **Non-Cluster** modes.

Table of Contents

1. [Prerequisites](#)
 2. [Installing Redis CLI](#)
 3. [Connecting to Redis](#)
 4. [Basic Cache Status Verification](#)
 5. [Cache Performance Validation](#)
 6. [Performance Metrics Validation](#)
 7. [Cache Invalidation Verification](#)
 8. [Performance Benchmarking](#)
 9. [Troubleshooting Guide](#)
 10. [Security Considerations](#)
-

Prerequisites

Before using this guide, ensure you have:

- Network access to your Redis instance
- Appropriate security group/firewall rules configured
- Redis endpoint information (hostname/IP and port)
- Sufficient permissions to connect to Redis
- Basic familiarity with command-line tools

Note: This guide assumes Heimdall is already configured and running with Redis caching enabled.

Installing Redis CLI

Linux (Ubuntu/Debian)

```
# Ubuntu/Debian
sudo apt update
sudo apt install redis-tools

# CentOS/RHEL/Amazon Linux
sudo yum install redis

# Or using dnf (newer versions)
sudo dnf install redis
```

macOS

```
# Using Homebrew
brew install redis

# Using MacPorts
sudo port install redis
```

Windows

Option 1: WSL (Recommended)

```
# Install WSL2 and Ubuntu, then:
sudo apt update
sudo apt install redis-tools
```

Option 2: Native Windows - Download Redis from: <https://github.com/microsoftarchive/redis/releases> - Extract and add to PATH, or use Docker:

```
docker run -it --rm redis:alpine redis-cli -h <redis-host> -p 6379
```

Docker (All Platforms)

```
# Quick redis-cli access without installation
docker run -it --rm redis:alpine redis-cli -h <redis-host> -p 6379
```

```
# Create alias for convenience
alias redis-cli='docker run -it --rm redis:alpine redis-cli'
```

Verify Installation

```
redis-cli --version
```

Expected Output:

```
redis-cli 6.x.x or 7.x.x
```

Connecting to Redis

Non-Cluster Mode

```
redis-cli -h <primary-endpoint> -p 6379
```

Cluster Mode

```
redis-cli -c -h <primary-endpoint> -p 6379
```

Note: The `-c` flag enables cluster redirection to handle hash slots. In cluster mode, pub/sub messages are shard-local. The subscriber and publisher must be on the same shard unless using sharded pub/sub.

Connection Parameters for Commands

All commands in this guide can be executed in two ways:

Method 1: Connect first, then run commands

```
# Connect to Redis instance
redis-cli -h <primary-endpoint> -p 6379
# Then run commands without connection parameters
CLIENT LIST | grep -E "(psub|sub)=[1-9]"
```

Method 2: Include connection parameters with each command

```
# Run commands with connection parameters
redis-cli -h <primary-endpoint> -p 6379 CLIENT LIST | grep -E "(psub|sub)=[1-9]"
```

For this guide: Examples will show Method 2 (connection parameters included) since it's more practical for validation scripts and remote instances.

1. Basic Cache Status Verification

Important: All commands in this section require connection to your Redis instance. Replace `<primary-endpoint>` with your actual Redis endpoint. For cluster mode, add the `-c` flag.

1.1 Check Active Connections and Subscriptions

```
# Verify Heimdall connections to Redis
redis-cli -h <primary-endpoint> -p 6379 CLIENT LIST | grep -v "cmd=client"
```

Expected Output:

```
id=393 addr=172.31.xx.xxx:58234 laddr=172.31.xx.xx:6379 fd=25 name= age=1764
idle=24 flags=N db=0 sub=0 psub=0 ssub=0 multi=-1 qbuf=0 qbuf-free=0 argv-
mem=0 multi-mem=0 rbs=1024 rbp=0 obl=0 oll=0 omem=0 tot-mem=2568 events=r
cmd=info user=default redir=-1 resp=3
id=394 addr=172.31.xx.xxx:58240 laddr=172.31.xx.xx:6379 fd=28 name= age=1764
idle=1764 flags=P db=0 sub=0 psub=4 ssub=0 multi=-1 qbuf=0 qbuf-free=0 argv-
mem=0 multi-mem=0 rbs=1024 rbp=0 obl=0 oll=0 omem=0 tot-mem=2664 events=r
cmd=psubscribe user=default redir=-1 resp=3
id=395 addr=172.31.xx.xxx:58244 laddr=172.31.xx.xx:6379 fd=29 name= age=1764
idle=1764 flags=N db=0 sub=0 psub=0 ssub=0 multi=-1 qbuf=0 qbuf-free=0 argv-
mem=0 multi-mem=0 rbs=1024 rbp=0 obl=0 oll=0 omem=0 tot-mem=2568 events=r
cmd=publish user=default redir=-1 resp=3
```

Key indicators of healthy Heimdall connections: - **Pattern subscriber** (id=394): `flags=P`, `psub=4`, `cmd=psubscribe` → Cache invalidation listener - **Info connection** (id=393): `cmd=info` → Health monitoring - **Publish connection** (id=395): `cmd=publish` → Cache invalidation publisher

```
# Alternative: Filter to show only subscription connections
redis-cli -h <primary-endpoint> -p 6379 CLIENT LIST | grep -E "(psub|
sub)=[1-9]"
```

```
# Check total pattern subscription count
redis-cli -h <primary-endpoint> -p 6379 PUBSUB NUMPAT
```

Expected Output:

```
(integer) 4
```


Info: This indicates Heimdall is listening on 4 pattern subscriptions for cache invalidation.

1.2 Verify Active Cache Keys

```
# Use SCAN (preferred in production; avoids blocking like KEYS)
redis-cli -h <primary-endpoint> -p 6379 SCAN 0 MATCH "hdkey:*" COUNT 100
```

Expected Output:

```
1) "0"
2) 1) "hdkey:7F1548AD7CC4D8FDA01F787EF48C7862"
   2) "hdkey:0781CCCD3B16C49A7CE30BAF8CD329F0"
3) "hdkey:4678749E2159093D167CD8895405F1E0"
4) "hdkey:E821C4B57C482228722661ED6ED1710A"
```

You should see 4–5 different cache keys for different query types: - JOIN queries - Department queries - Employee queries - Summary queries

2. Cache Performance Validation

2.1 Monitor Cache Hit/Miss Activity

```
# Monitor all cache-related operations
redis-cli -h <primary-endpoint> -p 6379 MONITOR | grep -E "GET|SETEX|hdkey"
```

Look for: - GET `hdkey:*` → cache hits - SETEX `hdkey:*` → cache writes/refreshes - TTL of ~61 seconds on SETEX (or as configured in Heimdall)

2.2 Monitor Cache Miss/Refresh Events Only

```
# Focus on cache misses and refreshes
redis-cli -h <primary-endpoint> -p 6379 MONITOR | grep -E '"SETEX"' | grep
'hdkey:'
```

With timestamps for easier analysis:

```
# Add timestamps to monitor cache refresh patterns
stdbuf -oL -eL redis-cli -h <primary-endpoint> -p 6379 MONITOR | while read -r
line; do
    if [[ $line == *'"SETEX"'* && $line == *'hdkey:'* ]]; then
        echo "$(date '+%H:%M:%S') - CACHE MISS/REFRESH: $line"
    fi
done
```

Pro Tip: Run the **SETEX monitor** in one terminal and a **GET monitor** in another terminal for side-by-side analysis:

```
# Terminal 1: Monitor GET operations (cache hits)
redis-cli -h <primary-endpoint> -p 6379 MONITOR | grep -E '"GET"' | grep
'hdkey:'

# Terminal 2: Monitor SETEX operations (cache misses/refreshes)
redis-cli -h <primary-endpoint> -p 6379 MONITOR | grep -E '"SETEX"' | grep
'hdkey:'
```

Important: Replace `<primary-endpoint>` with your actual Redis endpoint in all commands throughout this guide. For cluster mode, add `-c` flag: `redis-cli -c -h <primary-endpoint> -p 6379`

This side-by-side view helps correlate cache hits and refreshes in real-time.

2.3 Check Cache Key Details

```
# Replace with an actual cache key from your system
CACHE_KEY="hdkey:7F1548AD7CC4D8FDA01F787EF48C7862"

# Verify key existence
redis-cli -h <primary-endpoint> -p 6379 EXISTS $CACHE_KEY

# Check remaining time-to-live
redis-cli -h <primary-endpoint> -p 6379 TTL $CACHE_KEY
```

Expected Results: - EXISTS → 1 (key exists) - TTL → ≤ 61 seconds (or your configured TTL)

2.4 Validate Cache Refresh Patterns

```
# Monitor cache refresh timing with timestamps
redis-cli -h <primary-endpoint> -p 6379 MONITOR | grep -E "SETEX.*hdkey" |
while read line; do
    echo "$(date '+%H:%M:%S') - $line"
done
```

Expected Pattern: - Multiple `SETEX` operations after write operations - All relevant cache keys refreshed when related data changes

3. Performance Metrics Validation

3.1 Cache Hit Rate Calculation

```
echo "Monitoring cache operations for 60 seconds..."

# Capture GET operations for analysis
redis-cli -h <primary-endpoint> -p 6379 MONITOR | grep -E "GET.*hdkey" > /tmp/redis_gets.log &
MONITOR_PID=$!
sleep 60
kill $MONITOR_PID

TOTAL_GETS=$(wc -l < /tmp/redis_gets.log)
echo "Total cache GET operations: $TOTAL_GETS"

# Clean up
rm -f /tmp/redis_gets.log
```

Expected Results: - High volume of GET operations matching Heimdall Analytics Cache Hit percentage

3.2 Validate Query-Specific Cache Keys

```
# Categorize cache operations by query type
redis-cli -h <primary-endpoint> -p 6379 MONITOR | grep -E "SETEX.*hdkey" |
while read line; do
    if [[ $line == *"departments"* ]] && [[ $line == *"employees"* ]]; then
        echo "JOIN/SUMMARY Query Cache: $line"
    elif [[ $line == *"departments"* ]]; then
        echo "Department Query Cache: $line"
    elif [[ $line == *"employees"* ]]; then
        echo "Employee Query Cache: $line"
    fi
done
```

Expected Results: - Different cache keys for different query types - Appropriate refresh patterns based on query complexity

4. Cache Invalidation Verification

4.1 Test Write Operation Impact

```
echo "=== Before Write Operation ==="
redis-cli -h <primary-endpoint> -p 6379 SCAN 0 MATCH "hdkey:*" COUNT 100 |
```

```

sort

# Start monitoring cache invalidation events
redis-cli -h <primary-endpoint> -p 6379 MONITOR | grep -E "(DEL|SETEX).*hdkey"
&
MONITOR_PID=$!

echo "Perform write operation in your application now..."
sleep 30
kill $MONITOR_PID

echo "=== After Write Operation ==="
redis-cli -h <primary-endpoint> -p 6379 SCAN 0 MATCH "hdkey:*" COUNT 100 |
sort

```

Expected Results: - Same keys exist but with refreshed values - New SETEX operations with updated payloads

4.2 Validate Cross-Table Cache Invalidation

```

# Monitor specific cache key behavior
CACHE_KEY=$(redis-cli -h <primary-endpoint> -p 6379 SCAN 0 MATCH "hdkey:*"
COUNT 100 | head -1)

echo "Monitoring key: $CACHE_KEY"
echo "Before: TTL=$(redis-cli -h <primary-endpoint> -p 6379 TTL $CACHE_KEY)"

# Track all operations on this specific key
redis-cli -h <primary-endpoint> -p 6379 MONITOR | grep "$CACHE_KEY" | while
read line; do
    echo "$(date '+%H:%M:%S') - Cache activity: $line"
done

```

Expected Results: - Key gets refreshed (SETEX) after writes to related tables - Proper invalidation cascading across dependent queries

5. Performance Benchmarking

5.1 Measure Cache Response Times

```

# Get a sample cache key for testing
CACHE_KEY=$(redis-cli -h <primary-endpoint> -p 6379 SCAN 0 MATCH "hdkey:*"
COUNT 100 | head -1)

echo "Testing response times for key: $CACHE_KEY"
echo "Response times (in milliseconds):"

for i in {1..10}; do
    start_time=$(date +%s%N)

```

```
redis-cli -h <primary-endpoint> -p 6379 GET "$CACHE_KEY" > /dev/null
end_time=$(date +%s%N)
duration=$(( (end_time - start_time) / 1000000 ))
echo "Response time: ${duration}ms"
done
```

Expected Results: - Sub-millisecond (0–2 ms) response times for local network - Consistent latency across multiple requests

5.2 Validate Cache vs Database Performance

```
# Monitor cache hit/miss patterns with timestamps
redis-cli -h <primary-endpoint> -p 6379 MONITOR | while read line; do
  if [[ $line == *"GET"* ]] && [[ $line == *"hdkey"* ]]; then
    echo "$(date '+%H:%M:%S') - CACHE HIT: $line"
  elif [[ $line == *"SETEX"* ]] && [[ $line == *"hdkey"* ]]; then
    echo "$(date '+%H:%M:%S') - CACHE MISS/REFRESH: $line"
  fi
done
```

Expected Results: - Many GET operations (hits) versus fewer SETEX operations (misses/refreshes) - Higher ratio indicates better cache performance

6. Troubleshooting Guide

6.1 Low Cache Hit Rate (<50%)

Diagnosis:

```
# Check for excessive invalidation
redis-cli -h <primary-endpoint> -p 6379 MONITOR | grep -E "(DEL|
EXPIRE).*hdkey"
```

Common Causes: - Too many DEL operations = over-invalidation - Check cache rules and invalidation mappings - Verify TTL consistency

```
# Check TTL values across all cache keys
redis-cli -h <primary-endpoint> -p 6379 SCAN 0 MATCH "hdkey:*" COUNT 100 |
xargs -I {} redis-cli -h <primary-endpoint> -p 6379 TTL {}
```

6.2 No Cache Activity

Diagnosis:

```
# Test Redis connectivity
redis-cli -h <primary-endpoint> -p 6379 PING
# Should return: PONG

# Check for active Heimdall connections (same as section 1.1)
redis-cli -h <primary-endpoint> -p 6379 CLIENT LIST | grep -v "cmd=client"
# Should show pattern subscriber (flags=P, psub=4, cmd=psubscribe)
```

Common Causes: - Network connectivity issues - Heimdall not configured for Redis - Redis authentication problems

6.3 Performance Issues

Diagnosis:

```
# Check Redis memory usage
redis-cli -h <primary-endpoint> -p 6379 INFO memory | grep used_memory_human

# Monitor Redis latency
redis-cli -h <primary-endpoint> -p 6379 --latency -i 1
```

Common Causes: - High memory usage causing swapping - Network latency issues - Redis instance under-provisioned

6.4 Common Issues and Solutions

Connection Timeout

- **Symptoms:** Can't connect to Redis, connection refused errors
- **Solution:** Check security groups, network ACLs, and firewall rules

Authentication Failure

- **Symptoms:** AUTH errors, permission denied messages
- **Solution:** Verify Redis AUTH token configuration and credentials

Cluster Redirection Errors

- **Symptoms:** MOVED responses, slot redirection messages
- **Solution:** Use `-c` flag for cluster mode: `redis-cli -c -h <endpoint> -p 6379`

High Latency

- **Symptoms:** Slow response times, timeouts
- **Solution:** Check network connectivity, Redis instance resources, and memory usage

Cache Misses

- **Symptoms:** Low hit rates, frequent cache refreshes
 - **Solution:** Review cache invalidation rules, TTL settings, and query patterns
-

7. Security Considerations

Warning: When monitoring Redis operations, be aware of the following security considerations:

- **Sensitive Data:** The MONITOR command can expose cached data. Use in development/testing environments only.
 - **Production Safety:** Avoid running MONITOR in production for extended periods as it can impact performance.
 - **Network Security:** Ensure Redis access is properly secured with appropriate firewall rules.
 - **Authentication:** Use Redis AUTH if available and store credentials securely.
 - **Log Management:** Do not log sensitive cache contents to persistent storage.
-

Document Version: 1.0

Feedback: Please report any issues or suggestions for this guide to the Heimdall documentation team.

Users Overview

Heimdall currently provides a built-in user database that allows individual users to have defined authentication (password) and login locations defined.

Further, if desired, a user can be configured as a read-only user, and Google Authenticator compatible two-factor authentication.

Internally, passwords are stored in the same way that OpenLDAP stores them by default, i.e. Salted SHA. Example, `{SSHA}cca9bbffe6879f8367ab681952da2f995bf1668f`

Configurable fields can be found in: [Users](#)

Basic User (with Portal Mode enabled)

Require "Username", "Password" and Portal User enabled

The screenshot shows the 'Users' configuration page in Heimdall. At the top right, there is a 'Create New User' button and a menu icon. The main configuration area is titled 'Users' and contains a form for a user named 'basic-user'. The form includes an 'Enable' checkbox which is checked. A 'Commit' button is located at the top right of the form. The 'Configuration' section contains the following fields: 'Username' (basic-user), 'Password' (masked with dots), a 'Change Password' button, 'Email' (basic-user@heimdall.com), 'Management Privilege' (set to 'None'), 'Audit User' (unchecked), 'Portal User' (checked), 'Authentication mode' (set to 'Local/Secret'), and 'Two Factor Authentication' (with an 'Enable' button). A 'Secret' checkbox is also visible next to the 'Username' field.

Basic User (without Portal Mode enabled)

Require "Username", "Password" and selected Management Privilege

Users

Create New User

basic-user

☒ Enable

Commit

Configuration

Username: basic-user

Secret

Password:

Change Password

Email: basic-user@heimdall.com

Management Privilege: Read Only

Audit User: ☐

Portal User: ☐

Authentication mode: Local/Secret

Two Factor Authentication: Enable

Kerberos user

Require "Username" and "Authentication mode: Kerberos"

kerberos-user

☒ Enable

Commit

Configuration

Username: kerberos-user

Email: kerberos-user@heimdall.com

Management Privilege: Read Only

Audit User: ☐

Portal User: ☐

Authentication mode: Kerberos

More details about Kerberos Configuration [here](#).

LDAP user

Require "Username" and "Authentication mode: LDAP"

ldap-user

▼

☒ Enable

Commit

Configuration

Username:

ldap-user

Email:

ldap-user@heimdall.com

Management Privilege:

Read Only

▼

Audit User:

☐

Portal User:

☐

Authentication mode:

LDAP

▼

LDAP Configuration:

HEIMDALL_LDAP_CONFIGURATI

▼

Two Factor Authentication:

Enable

More details about LDAP Configuration [here](#).

If there is only one LDAP Configuration available, it will be used by default. Otherwise additional dropdown "LDAP Configuration" will appear to specify which config should be used with a particular user.

SAML2 user

Require "Username" and "Authenticate by: SAML2"

Users

Create New User

☰

test

▼

☒ Enable

Commit

Configuration

Username:

test

Email:

email@heimdall.com

Management Privilege:

Admin

▼

Audit User:

☒ Note: User must re-login to view audit trails

Portal User:

☒

Authentication mode:

SAML

▼

Groups

Add: +

Remove: +

You can set up SAML2 Configuration in the [SAML 2.0 Configuration Guide](#). SSO can be configured using saml2 auth with AWS IAM Identity Center, Okta, or other saml2 identity providers.

IAM user

Require "Username" and "Authenticate by: IAM"

When using IAM access, appropriate access to the IAM role should be configured for the management instance.

Groups

This feature is available for all Heimdall (GUI) users, doesn't matter whether user authenticates by LDAP or not.

This one allows to grant or revoke a group membership for testing purposes, e.g for LDAP based notifications (Notifications tab in [admin tab](#)).

For enhanced security measures, it is imperative that at least one real LDAP Group be extracted when utilizing this feature while using for LDAP authentication testing purposes.

What is important configuration applied there **DOES NOT IMPACT** the real groups membership on LDAP Server.

Groups

Add:			Remove:	
admins	▼	☐		☐
full-time	▼	☐	part-time	☐
heimdall-admins		☒		

Admin Overview

The administration tab is divided into multiple sections covering a variety of administrative actions, as detailed below.

Account Information

Account Information (applicable for cloud environments only)
Register Your Instance for Free Technical Support
Unlock direct access to our support portal and ensure your instances are fully supported by registering now.
Please visit the [support portal](#) to supply your account ID (272965818115) and instance ID (i-0f62354d98d4de66e) for hassle-free registration and immediate support.

Account ID: 272965818115
Instance ID: i-0f62354d98d4de66e
Registered ID:

This section is to view installation metadata to help customers work with Heimdall Support in enrolling for complimentary support. Account ID is an GUID of the installation, while Registered ID is provided from Customer Support to verify enrollment. Enrollment can be confirmed by a checkmark next to the field (as well as the removal of the UNSUPPORTED banner) see below:

Account Information (applicable for cloud environments only)

Register Your Instance for Free Technical Support

Unlock direct access to our support portal and ensure your instances are fully supported by registering now. Please visit the [support portal](#) to supply your account ID (272965818115) and instance ID (i-0f62354d98d4de66e) for hassle-free registration and immediate support.

Account ID:

272965818115

Instance ID:

i-0f62354d98d4de66e

Registered ID:

34ac6400a2c0385fb4aeb78

✓

Commit

License Configuration

Administration

License

Notifications

Logs

Software Management

Alerts

Login History

Cache Drivers

Server Properties

Filters

Current License

Server Version:

18.4.11.1-20180411193413

Server Start Time:

[2018-04-11 23:35:27,079]

Demo Mode:

true

License Start Date:

2018-03-16 17:34:48

License End Date:

2018-04-15 17:34:48

Statistics Callback:

60 (seconds)

LB Servers Allowed:

4

LB Servers Used:

2

Optimizations Allowed:

4

Optimizations Used:

3

License Upload

Choose Files

No file chosen

In this section, a new license file can be uploaded when provided by Heimdall, and the current settings and license use can be observed. In Demo mode, up to 4 LB servers can be used, and up to four VDBs with rules (optimizations) can be enabled for 30 days. The used count is only based on VDBs that have received traffic in the last 30 seconds, so will dynamically account for activity. If the used values surpass the allowed values, then a warning will be displayed on the console, but no restriction in features will be enforced. License handling is strictly advisory, i.e. while the system may complain about a license being invalid, it will not restrict what is done, to insure no outages occur as a result of licensing.

Note: When using an AWS paid-for instance, the proxy use on the instance will not count toward license use. Additionally, added paid-for instances can be reconfigured to be a pure proxy instance and will also not count to license use on another management server.

Log Management

Log Management

[Delete Db Logs](#)[Delete File Logs](#)[Delete All Logs](#)[Roll Log](#)[Download Logs](#)[Delete Packet Captures](#)[Delete Core&Heap Dumps](#)

Log Database Management

[Commit](#)

Use external database for logging: ☐

Using: Embedded database (HSQL)

Cloud Logs

[Commit](#)

Send login entries to CloudTrail: ☒

Cloud Trail Lake Channel ARN:

Enable Manager CloudWatch Logging: ☒

CloudWatch namespace:

S3 Bucket Name:

[Reset to default](#)

Force Upload to S3 On Logs Rolling: ☐

Log Server Properties

[Commit](#)

Log Events To Console: ☐

Max Log Age:

Reserved Disc Space:

Log Rotation Interval:

Log Management

This section is provided to assist in managing log files and configuring data source used for logging. Core dumps are files that are generated by the operating system when a program terminates abnormally, such as by crashing or encountering an error. Heap dumps are files that contain a snapshot of the Java Virtual Machine (JVM) heap at a specific point in time. This functionality is removing all core files from /var/lib/systemd/coredump folder and also core files and heap dumps files from the project directory.

Using the **Download Logs** button will download a zip archive, with optional encryption password for log files. For details see [Logs](#).

Log Database Management

By default, an embedded database supplied with Heimdall is used to store logs and statistics, however there is an option to select user configured data source (MySQL, PostgreSQL, Oracle and SQL Server supported). If Heimdall is unable to establish connection to selected source at any point it will fall back to using embedded one. User configured in selected data source has to have database root privileges.

Cloud Logs

This section allows you to configure Heimdall to send user/proxy login events to AWS CloudTrail for centralized logging and audit trail purposes. When enabled, the exact same logs as those which can be found on the Logging History tab, will be sent to AWS CloudTrail Lake.

Configuration Requirements: **Send login entries to CloudTrail** and **Cloud Trail Lake Channel ARN**

Configuration options can be found in: [Log Management Options](#)

- **Important:** Use the Channel ARN, not the Event data store ARN. Proper IAM role has to be configured.

Channel Name	Status	Source	Channel ARN	Event data stores	Last event received
arozuk-audit-trail-data-athena	Active	My custom integration	arn:aws:cloudtrail:us-east-1:27...	arozuk-cloud-trail-data-athena	June 26, 2025, 21:50:04 (UTC+...)
arozuk-cloudtrail-channel	Active	My custom integration	arn:aws:cloudtrail:us-east-1:27...	heimdall-mgmt-cloudtrail-aro...	April 01, 2025, 18:14:18 (UTC+...)

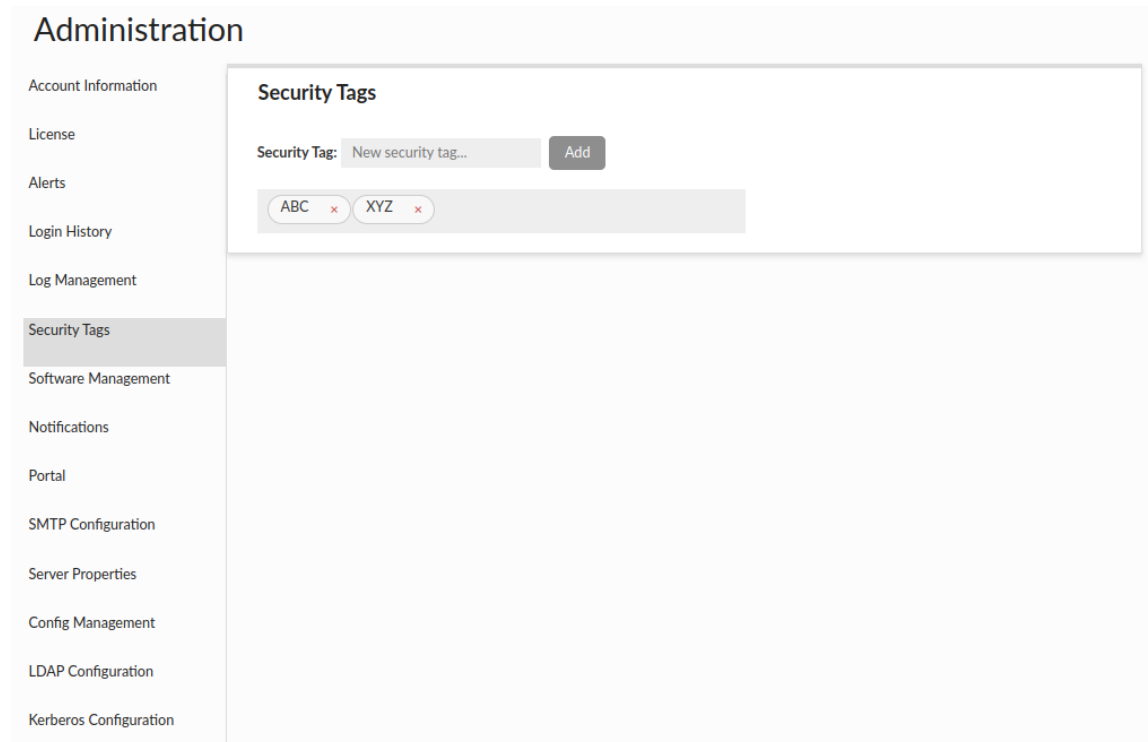
CloudTrail Lake's supports lake query federation with Amazon Athena, which allows you to run SQL queries directly against your event data store for advanced analytics and reporting.

Log Server Properties

These are properties that can control overall Log server behavior, and will be added to over time. Current properties supported: [Log Server Properties](#)

For casual Server Properties, see [Server Properties](#).

Security Tags



This section allows for the management of security tags, which can be applicable in the database browser section of [Source tab](#). Each resource (i.e. databases, schemas, tables, views, columns) in the database browser can be tagged with one or more tags. The admin can select a specific tag from the dropdown and filter out the tagged resources.

Warnings:

1. If you delete a resource in your database that had assigned security tags and later decide to recreate it, it will inherit those tags.

2. If you change the name of a given resource in your database, the security tags assigned to it will be lost.

Software Management

Administration

Account Information

License

Alerts

Login History

Log Management

Security Tags

Software Management

Notifications

Portal

SMTP Configuration

Software Management

Server Version: 24.07.15.0-20240715142308

Available Version: 24.04.25.1

Test Build

URL link

Select File

Shutdown/Restart Heimdall

Download Heimdall Driver

Release Notes

Update via Repository

Download via Repository

Update via URL

Update via File

Download Current Software

Rollback version

Migration

This section allows for management of the software. In the first dropdown, you can select a version of code to update to. Release is the normal release train of code, test versions are generally just before a new release, and development are the often daily updates that have completed regression testing. Customers will often be asked to update to the development build if they have encountered a problem and want to apply the fix for it. To apply a particular version, select the version in the dropdown, then click "Update via Repository". Please note, if you have independent proxies running (via NLB or as a service), you will need to restart them independently as well. To check the version of code each proxy is running, please refer to the status tab.

Note: Any build pushed as a development build have passed all regression testing that any previous version has passed. What may not yet be complete are regression tests for the new changes, which often are developed in parallel with customer testing of the changes. This can result in frequent development builds, but allows for rapid test/fix/deploy cycles during customer POC processes. Once a POC is completed, we will work with customers to schedule a full release build for use in production that has completed full and updated regression testing.

You can also update the version of Heimdall by providing the url link to the zip file, complete this action using Update via URL button.

In the event that you want to update to a particular version of Heimdall that was provided to you as a zip file, you can use the third section for a manual update. Simply select the heimdall zip file, then update via File.

The next option is the Shutdown/Restart Heimdall button, which will not explicitly trigger a restart, but will rely on any restart mechanisms on the server to trigger the actual execution of the server after it terminates.

In the event you need to download the Heimdall driver (say to use as a jdbc driver) the next option "Download Heimdall Driver" can be used.

If you need to roll back to a previous version of the system, the quickest way is to use the "Rollback Version" button. Heimdall stores two versions of the system - the current and the previous - and you can swiftly switch between them.

Following is the release notes link, for the most current release notes.

Note, for command line manual update, please do the following:

1) copy the file into the install directory as heimdall-new.zip 2) cd to install directory 3) execute the following commands:

```
* sed -i "s/\s*"modules\/.*\","*//g" config/heimdall.conf
* unzip -j heimdall-new.zip
```

4) kill and restart the heimdallserver.jar process: ps auxw | grep heimdallserver.jar root 4198 0.0 0.0 6100 624 pts/0 S+ 18:03 0:00 grep --color=auto heimdallserver.jar root **26005** 30.8 4.0 9679444 638504 ? Sl 17:58 1:25 java -server -XX:+ExitOnOutOfMemoryError -jar /opt/heimdall/heimdallserver.jar

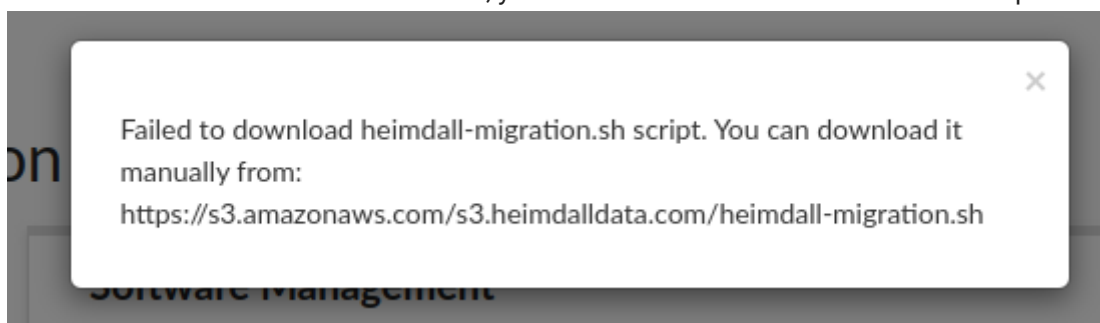
kill **26005**

The next option is the Migration button, which initiates the download and execution of the 'heimdall-migration.sh' script. The script carries out all necessary actions to meet the requirements for the upcoming release.

Current migration steps:

1. Install Java 17 and set it default
2. Update heimdall-entripoint.sh script

If there will be an issue with download file, you will receive an error with link to the script:



If you encounter this issue, proceed to download the migration script manually [here](#). Before executing the script, ensure it has the necessary execute permissions:

```
chmod +x heimdall-migration.sh
```

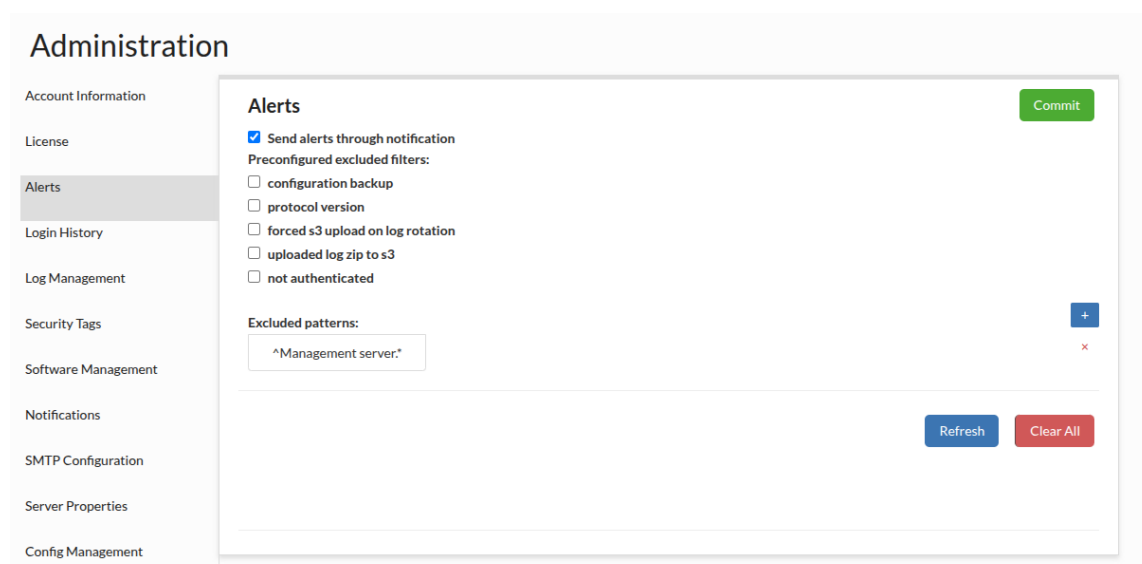
To manually run the migration script on the Heimdall Manager host, use the following command:

```
bash heimdall-migration.sh
```

If an error occurs during migration with retrieving the latest file heimdall-entrpoint.sh, it should be downloaded from [here](#), placed on the host where migration is being conducted, and the path to it should be provided as an argument to the script:

For example: `bash heimdall-migration.sh /tmp/heimdall-entrpoint.sh`

Alerts



The screenshot shows the 'Administration' section of the Heimdall Manager interface. On the left is a sidebar menu with options: Account Information, License, Alerts (selected), Login History, Log Management, Security Tags, Software Management, Notifications, SMTP Configuration, Server Properties, and Config Management. The main content area is titled 'Alerts' and includes a green 'Commit' button in the top right. It features a checkbox 'Send alerts through notification' which is checked. Below this is a section for 'Preconfigured excluded filters' with five unchecked options: configuration backup, protocol version, forced s3 upload on log rotation, uploaded log zip to s3, and not authenticated. There is also an 'Excluded patterns' section with a text input field containing '^Management server.*' and a blue '+' button to add more patterns. At the bottom right of the main area are 'Refresh' and 'Clear All' buttons.

This section provides a list of any alerts that have been dismissed with the check. Any alerts in this list will not be shown in the top of the GUI until cleared.

Sending alerts through notification: If this option is checked, every alert which message doesn't match any added pattern, will be sent through notification. If no pattern is added, all alerts will be sent through notification. The [System Notification](#) has to be specified to make it work and to see below options.

Preconfigured excluded filters: Every alert which message contains a phrase from a checked filter will not be sent through notification.

Excluded patterns: Alerts not matching this patterns will be sent via notification.

Notifications

The screenshot shows the 'Notifications' configuration page. On the left is a sidebar with navigation links: Account Information, License, Alerts, Login History, Log Management, Security Tags, Software Management, Notifications (highlighted), Portal, SMTP Configuration, Server Properties, Config Management, Secrets Manager Configuration, LDAP Configuration, Kerberos Configuration, and SAML Configuration. The main content area is titled 'Notifications' and includes a 'System notification' dropdown set to 'none' and a green 'Commit' button. Below is a table with 5 entries:

#	Protocol	Notification Alias	Configuration	Send Test	External Approvals	Delete
1.	SMTP	none	Heimdall Username, for SMTP lookup	Test	☐	
2.	SMTP	admin	admin	Test	☐	
3.	LDAP	ldap	test-users	Test	☐	
4.	SQL	SQL_notification	Data Source: Postgres-source17 SQL: select email from public.myusers where id > 1 and id < 4	Test	☐	
5.	SIMPLE	SIMPLE_notification	test@gmail.com	Test	☐	

At the bottom of the table is a blue button labeled 'Add Notification Entry'.

This section provides the capability to create group notifications. Users can choose from five supported protocols: AWS SNS, LDAP, SQL, SMTP or SIMPLE. Additionally, there is an option to view the list of emails associated with a particular entry by clicking the envelope icon (except SIMPLE). This will display a popup showing the email addresses.

System notification: This option allows to choose which notification config is used as system notification and will be used to send various system messages, such as alerts by default.

When choosing AWS SNS for email (and potential SMS) notifications, it will require Heimdall to be hosted on the AWS platform, having appropriate AWS IAM roles and permissions configured to interact with AWS SNS.

When selecting SQL, the user needs to specify the target source and provide a query that will fetch email addresses from the database for that particular config.

When choosing SMTP, user will be able to define his mailing list (from existing Heimdall defined users in [Users tab](#)). Also, please remember to configure SMTP client in the Admin tab - > [Smt Configuration](#).

When choosing SIMPLE, user will be able to define his mailing list, similar as in SMTP but without the need for defined users in [Users tab](#). [Smt Configuration](#) is still needed.

When choosing LDAP, user needs to have LDAP configuration set up, as it requires connection to the LDAP server to retrieve the groups.

Additionally, it's possible to create a "none" type notification, meaning without specifying any group or recipients. This is used in portal mode, where it can be assigned to a [Role Config](#) that should be auto-approved. Such empty notifications without recipients can only be created with the SMTP protocol.

The External Approvals option is only available when Portal Mode is enabled. If this option is selected for a given entry, and that notification is then assigned to a role in [Roles Management](#), then when a session is requested for that role via the portal, each recipient of that entry will receive a special link with a token. By opening that link, they'll be able to approve or deny the session. Importantly, such a user does not need to be a Heimdall user. In this case, each user receives a unique token. The emails are sent via the SMTP protocol. Therefore, make sure you have this properly configured in the [Smtp Configuration](#) section.

After making any changes, they must be submitted using the Commit button.

Portal

Administration

- Account Information
- License
- Alerts
- Login History
- Log Management
- Security Tags
- Software Management
- Notifications
- Portal**
- SMTP Configuration
- Server Properties
- Config Management
- Secrets Manager Configuration
- LDAP Configuration
- Kerberos Configuration
- SAML Configuration
- Password Policy
- Modules
- TCP

Portal Basic

LDAP Configuration: LDAP_CONFIG

Portal data source: MySQL-source

Portal Host: https://example.org:8443

JWT Signature Key:

Secret

Send audit logs to CloudWatch: ☒

CloudWatch namespace: heimdall-audit

Send audit logs to CloudTrail: ☒

Send audit queries logs to CloudTrail: ☒

Cloud Trail Lake Channel ARN: arn:aws:cloudtrail:us-east-1:272911111111:channel/16e1b5bb-b564-463

Show approvers emails: ☒

Synchronize Portal Users: ☒

Send Calendar Item: ☒

Require specific justification format: ☒

Regex pattern: Match all if empty

Invalid justification message: Does not match the global regex pattern.

Commit

+

Notifications

+

The "Portal Basic" section is generally used to configure settings related to the portal. Within it, we can define the LDAP configuration, from which portal users will be sourced, the data source where portal data will be stored, and several checkboxes.

- **Portal data source** - Data source where portal data will be stored.
- **Portal Host** – the URL where the Central Manager is hosted. Approvers use this URL to access the portal.
- **JWT Signature Key** - used to sign and validate the token signature for External Approvals. The signatureKey must be provided in Base64Url format, and once decoded it must have a minimum length of 256 bytes. In its encoded form, only the characters A–Z, a–z, 0–9, -, _.
Note: The "signatureKey" can be stored in the [Secrets Manager](#). *Note:* The signature key does not support secret rotation at this time.

Alternatively, you can generate the key by clicking the corresponding button.

- **Show approvers emails** - If selected, it will be possible for all users to view approvers' emails for a specific role. Admins can always see these emails, even without the checkbox being checked.
- **Send audit logs to CloudWatch** - Indicates whether the audit trail logs should also be saved in the AWS CloudWatch.
- **CloudWatch namespace** - The namespace specified in this field will be used as the log group name in the AWS CloudWatch. If left empty, audit logs will be sent to the default namespace: HEIMDALL-mgmt, provided that logging is enabled in [Server Properties](#).
- **Send audit logs to CloudTrail** - Indicates whether the audit trail logs (Session Audit tab records) should also be saved in the AWS CloudTrail Lake
- **Send audit queries logs to CloudTrail** - Indicates whether the audit trail session queries logs (The SQL queries which have been executed during portal session - when done via Heimdall proxy VDB) should also be saved in the AWS CloudTrail Lake. This option will work only when "Log All Portal Sessions SQL" (VDB → Logging → Log All Portal Sessions SQL) option is enabled on the VDB.
- **Cloud Trail Lake Channel ARN** - CloudTrail Lake Channel ARN, in the format:
`arn:aws:cloudtrail:<region>:<account-id>:channel/<channel-id>`. Can be found on AWS with path CloudTrail → Lake → Integrations (more information can be found on [CloudTrail Lake Integration for Login Events](#))
- **Synchronize Portal Users** - Indicates that if the LDAP Group Filter matches, Heimdall (GUI) users will be created automatically.
- **Send Calendar Item** - Indicates whether a calendar item should also be sent along with the emails about an approved session, but only if the session start time is more than one hour ahead. A lot also depends on the email client — not all clients display the .ics file as a graphical calendar event as below.

Tomorrow • 09:00 – 10:30
[Heimdall] Session
Auto-Approved.



No events on this date

Add to Calendar

- **Require specific justification format** - If selected, there is an option to specify the regex that the justification must match when requesting a session.
 - **Regex pattern** - Specifies the regex that must be matched in the justification field.
 - **Invalid justification message** - Specifies the message to appear when requesting a session if the justification content does not match the configured regex.

To simplify portal configuration, three additional sections are available below: one for [Notifications](#), the second for Data Source, where you can configure [Group Mapping](#) and [Roles Management](#), and the third ([Users](#)) for setting portal users and manually assigning them groups.

For more complex configurations, it is recommended to configure them in the dedicated tabs intended for each specific setting.

If you are interested, please see the [Portal Guide](#) page.

SMTP Configuration

Administration

- Account Information
- License
- Alerts
- Login History
- Log Management
- Security Tags
- Software Management
- Notifications
- Portal
 - SMTP Configuration
 - Server Properties

SMTP Configuration

Sender User/Email: example@heimdalldata.com

Sender Password:

Host: smtp.gmail.com

Port: 587

Smtplib Auth: true

START_TLS Enabled: true

Secret

☐

Commit

SMTP Properties

mail.smtp.ssl.protocols

▼

TLSv1, TLSv1.1, TLSv1.2, TLSv1.3

+

×

Email Tester

Recipient Email Address: example@heimdalldata.com

Test

This section provides the functionality to establish SMTP configurations for email dispatch. Users can adjust the configuration to their preferences, by incorporating relevant properties in the SMTP Properties section. The Heimdall team recommends users set the ssl protocol to TLSv1.2 or TLSv1.3, otherwise TLSv1 might be used.

Available configurable items can be found there: [SMTP Configuration](#)

⚠ **Note:** While using e.g. Google smtp, user should consider using application password rather than their personal password - especially if the account has 2fa enabled, then without app password functionality might have problem to operate. Note: The username and password can be stored in the [Secrets Manager](#). The "username" field is optional.

The Email Tester component, allows users to validate configuration before committing changes by sending a test email. If changes are not committed, all entered configuration will be lost.

Login History

Administration					
Account Information	Login History Exclude proxy logins: ☒  				
License	Jun 26, 2025, 9:13:28 PM	admin	109.231.16.118	BASIC_AUTH	["Admin"]
Alerts	Jun 26, 2025, 9:13:23 PM	guest	109.231.16.118	BASIC_AUTH	["Read Only"]
Login History	Jun 26, 2025, 9:13:16 PM	guest	109.231.16.118	BASIC_AUTH	["Read Only"]
Log Management	Jun 26, 2025, 9:13:09 PM	guest	109.231.16.118	BASIC_AUTH	["Read Only"]
Security Tags	Jun 26, 2025, 9:12:02 PM	admin	109.231.16.118	BASIC_AUTH	["Admin"]
Software Management	Jun 26, 2025, 9:11:49 PM	guest	109.231.16.118	BASIC_AUTH	["Read Only"]
Notifications	Jun 26, 2025, 9:11:07 PM	admin	109.231.16.118	BASIC_AUTH	["Admin"]
SMTP Configuration	Jun 26, 2025, 3:29:03 PM	admin	109.231.16.118	BASIC_AUTH	["Admin"]
Server Properties	Jun 26, 2025, 3:25:11 PM	admin	109.231.16.118	BASIC_AUTH	["Admin"]
	Jun 26, 2025, 3:23:16 PM	admin	91.239.154.61	BASIC_AUTH	["Admin"]
	Jun 26, 2025, 3:01:45 PM	admin	109.231.16.118	BASIC_AUTH	["Admin"]
	Jun 26, 2025, 2:58:39 PM	admin	91.239.154.61	BASIC_AUTH	["Admin"]

This provides the login times, account ID and source IP of the recently recorded GUI logins, with their privileges, and auth type.

Unchecking 'Exclude proxy logins' will result with adding proxy logins to the list. Heimdall can put these login entries to AWS CloudTrail as well. For more information, see here: [Admin -> CloudTrail integration section](#)

Server Properties

Administration

Account Information

License

Alerts

Login History

Log Management

Security Tags

Software Management

Notifications

Portal

SMTP Configuration

Server Properties

Config Management

Secrets Manager Configuration

LDAP Configuration

Server Properties

Enable Portal Mode: ☒

Disable Cert Validation: ☐

Password Validation: ☐

Redirect Config Fetches: ☐

Verbose Debug Mode: ☐

Minimum free disk space %: 1.0

Max Config Backups: 10

DNS Port: Value that indicates what port should heimdal

Session Timeout: 30 m

Proxy Host: Proxy host for code updates checks and dowr

Proxy Port: Proxy port, default 3128 (integer)

Proxy User: Proxy user for proxy authentication, no defaul

Proxy Password: Proxy password for proxy authentication, no c

Commit

These are properties that can control overall Management server behavior, and will be added to over time.

Currently supported properties can be found here: [Server Configurable items](#)

There are also **HTTP Proxy Settings** in the same section which allow the application to connect to the internet through an HTTP proxy. This is especially useful in environments where direct internet access is restricted, such as corporate networks. The proxy settings are used for operations like code update notifications, software downloads, and other outbound HTTP requests.

In case of Proxy related properties, if any is set, all has to be set.

For log related Server Properties, see [Log Server Properties](#) in Log Management tab.

Modules

Allows modules to be added or removed as needed. The fewer the modules, the faster a Heimdall node can startup. In particular cache modules that are unused should be deleted as some can add significant load time.

Password Policy

The screenshot shows the 'Administration' sidebar on the left with 'Password Policy' selected. The main content area displays a 'Password policies' modal window. This window contains a list of three existing policies, each with a red 'x' icon for deletion. The policies are:

- Name:** minimum of characters
Parameters: minimum (NUMBER)
Value: 4
- Name:** custom regex matching
Parameters: pattern (TEXT)
Value: zz([A-Z])\w+
- Name:** uppercase character requirement

At the bottom of the modal is a green 'Commit' button. A blue '+' button in the top right corner of the modal is used to add new policies.

This section allows user to set up requirements that specific user password must meet to be created/updated. Current rules supported:

- minimum/maximum of characters - minimum/maximum number of characters that password must contain (with **minimum/maximum** parameter tht defines the limit).
- digits requirement - rule defining that password must contain at least one digit
- special marks requirement - rule defining that password must contain at least one special mark
- uppercase character requirement - rule defining that password must contain at least one uppercase character
- custom regex matching - rule that provides a **pattern** parameter with the regular expression to which password must match

The Password Policies apply to GUI Heimdall users but do not extend to users with external passwords, such as those using LDAP, Kerberos, or Secrets Manager.

Config Management

Config Management

Edit Startup Config (/etc/heimdall.conf)

hdRole:		
hdHost:	localhost	
hdPort:	8087	
hdUser:	admin	Secret
hdPassword:	••••••••	
hdSecretKey:		
cloudDetection:	☐	
cloudOption:	none	
javaOptions:	"-server --add-opens=java.base/java.security=ALL-UNNAMED --"	
secure:	☐	

Commit

Config Management

Edit Startup Config (/etc/heimdall.conf)

hdRole:	server	
hdHost:	localhost	
hdPort:	8443	
hdUser:	admin	Secret
hdPassword:	••••••••	
hdSecretKey:		
cloudDetection:	☐	
cloudOption:	none	
javaOptions:	"-server --add-opens=java.base/java.security=ALL-UNNAMED --"	
secure:	☒	

Commit

Config Management

Edit Startup Config (/etc/heimdall.conf)

hdRole:	proxy	▼
hdHost:	localhost	
hdPort:	8443	
hdUser:	correct_access_key	Secret ☐
hdPassword:	••••••••••	☒
javaOptions:	"-server --add-opens=java.base/java.security=ALL-UNNAMED --;	
vdbName:	the_vdb	
secure:	☒	

[Commit](#)

This section allows editing startup `/etc/heimdall.conf` configuration file, which can be used to populate environmental variables or cloud environments. More details in [Heimdall Environmental Variables](#).

Note: The username and password can be stored in the [Secrets Manager](#).

The additional environmental variables that are necessary for secrets are described here [Secrets Environmental Variables](#).

AWS Endpoints

AWS endpoints

CloudWatch Endpoint:	
CloudWatch Logs Endpoint:	
ElasticCache Endpoint:	
RDS Endpoint:	
SNS Endpoint:	
CloudTrail Endpoint:	
Identity Store Endpoint:	

[Commit](#)

This section: [AWS Alternate Endpoints](#) allow to configure alternate endpoints for AWS services. For alternate endpoint for Secrets Manager see [AWS Secrets Manager](#).

- **Secrets Manager Endpoint**

- **CloudWatch Endpoint**
- **CloudWatch Logs Endpoint**
- **ElasticCache Endpoint**
- **RDS Endpoint**
- **SNS Endpoint**
- **CloudTrail Endpoint**
- **Identity Store Endpoint**

⚠ Note: If alternate secrets manager endpoint is passed, secrets will be used even if cloud option is not set to AWS.

Identity Providers

Heimdall supports a variety of identity providers to authenticate users for various situations. This includes:

- Secrets Managers (AWS Secrets Manager, CyberArk Conjur and Hashicorp Vault)
- Kerberos Domains
- LDAP (AD, Jumpcloud, Okta, Redhat ID Manager, OpenLDAP, and others)
- SAML (AWS Identity Center, Okta, and others)

To begin to configure any of these, in the admin tab, please go to proper tab and follow the steps in the instruction.

Secrets Manager Configuration

This section allows to configure Secrets Manager that will be used across Heimdall services.

Remember that if you want to use the secrets manager, it has to be enabled.

Currently supported settings:

Default

Secrets Manager Configuration

Create New

No Secrets Manager configurations have been created.
Please select the "Create New" option to initiate the creation of a new configuration.

By default, 0 Secrets Managers are configured. Secret options will be disabled.

All of the following Secrets Managers can be configured using [Secrets Environment Variables](#).

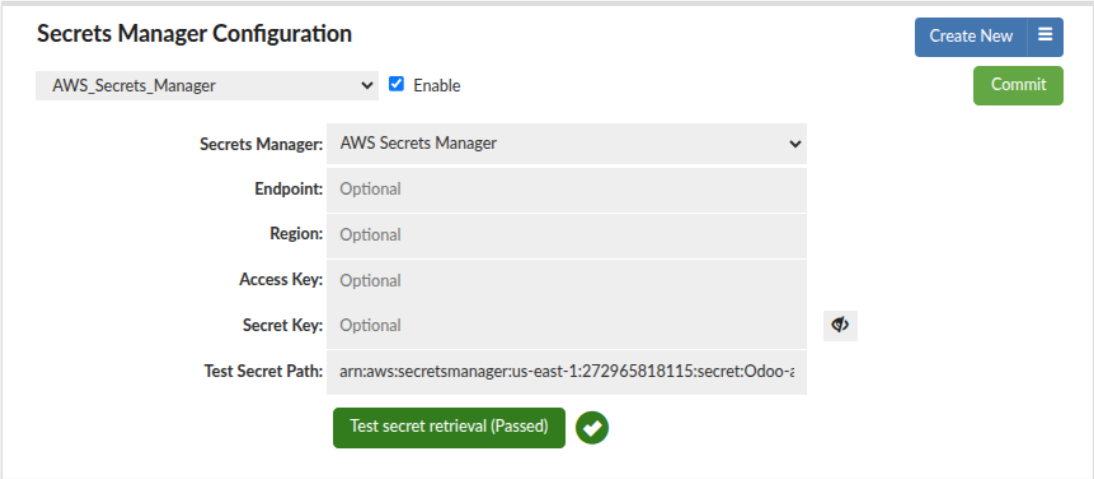
AWS Secrets Manager

When running in AWS environment it will be added by default if no other configurations exists of this type. Does not require additional configuration. When specifying a secret path both secret name and ARN can be used. Multiple AWS secrets managers can be configured.

Following optional properties are available for AWS Secrets Manager connection configuration to set in: [AWS Secrets Configuration](#)

DEFAULT CONFIGURATION

By default, Heimdall uses the AWS Secrets Manager service in the same region as the Heimdall instance (e.g., `us-east-1`). No additional configuration is required.



The screenshot shows the 'Secrets Manager Configuration' dialog. At the top right are 'Create New' and 'Commit' buttons. Below the title, there's a dropdown menu set to 'AWS_Secrets_Manager' and an 'Enable' checkbox which is checked. The main configuration area has several fields: 'Secrets Manager' (dropdown set to 'AWS Secrets Manager'), 'Endpoint' (Optional), 'Region' (Optional), 'Access Key' (Optional), 'Secret Key' (Optional), and 'Test Secret Path' (containing 'arn:aws:secretsmanager:us-east-1:272965818115:secret:Odoo-'). To the right of the 'Secret Key' field is an eye icon. At the bottom, there's a green button that says 'Test secret retrieval (Passed)' next to a green checkmark icon.

CROSS-REGION CONFIGURATION

To use a secret stored in a different region, you must specify that region explicitly. Providing the ARN alone is insufficient—even if it contains region information—because the AWS API defaults to the local region.

Example: If the Heimdall instance is in `us-east-1` but the secret is in `us-east-2`, the connection will fail unless the region is explicitly defined.

Secrets Manager Configuration

AWS_Secrets_Manager Enable Create New Commit

Secrets Manager: AWS Secrets Manager

Endpoint: Optional

Region: eu-central-1

Access Key: AKIA2KTHPPBLWC6QH5HF

Secret Key: ••••••••

Test Secret Path: arn:aws:secretsmanager:eu-central-1:709960366167:secret:hd/s

Test secret retrieval (Passed)

CROSS-ACCOUNT CONFIGURATION (IAM POLICIES + KMS KEY SHARING)

Uses IAM policies and KMS key sharing.

Example accounts

- Source account (Heimdall runtime and KMS key owner): `272965818115`
- Target account (secret owner): `709960366167`

Steps

1. Create a customer-managed KMS key in the source account. The key must be in the same region as the secret. For cross-region setups, use a multi-Region KMS key and create a replica in the target region (for example `eu-central-1`). In this example the source account is in `us-east-1`, but the KMS key is replicated to `eu-central-1`.
2. KMS key creation options used in this example: Key type "Symmetric", Key usage "Encrypt and decrypt". In Advanced options, choose Key material origin "KMS" and Regionality "Multi-Region key" to allow replicas in other regions.

Configure key

Key type [Help me choose](#)

☒ **Symmetric**
A single key used for encrypting and decrypting data or generating and verifying HMAC codes

☐ **Asymmetric**
A public and private key pair used for encrypting and decrypting data, signing and verifying messages, or deriving shared secrets

Key usage [Help me choose](#)

☒ **Encrypt and decrypt**
Use the key only to encrypt and decrypt data.

☐ **Generate and verify MAC**
Use the key only to generate and verify hash-based message authentication codes (HMAC).

Advanced options

Key material origin
Key material origin is a KMS key property that represents the source of the key material when creating the KMS key. [Help me choose](#)

☒ **KMS - recommended**
AWS KMS creates and manages the key material for the KMS key.

☐ **External (import Key material)**
You create and import the key material for the KMS key.

Regionality
Create your KMS key in a single AWS Region (default) or create a KMS key that you can replicate into multiple AWS Regions. [Help me choose](#)

☐ **Single-Region key**
Never allow this key to be replicated into other Regions

☒ **Multi-Region key**
Allow this key to be replicated into other Regions

[Cancel](#) [Next](#)

3. In the AWS KMS wizard, specify the AWS accounts that can use this key. In our example, this is the target account (709960366167).

KMS > [Customer managed keys](#) > Create key

Step 2
Add labels
Step 3 - optional
Define key administrative permissions
Step 4 - optional
Define key usage permissions
Step 5 - optional
Edit key policy
Step 6
Review

Key users (68)
Select the IAM users and roles authorized to use this key in cryptographic operations. These users will be added to the key policy under the statement identifiers (Sid) 'Allow use of the key' and 'Allow attachment of persistent resources'. Modifying these Sids might impact the console's ability to update the user statements in the key policy. [Learn more](#)

☐	Name	Path	Type
☐	rds-directoryservice-kerberos-access-role	/	Role
☐	rds-monitoring-role	/	Role
☐	RDSConnectToDb	/	Role
☐	redis-h	/	Role
☐	S3Access	/	Role
☐	secretRotationTestAdam-role-58tgfuwe	/service-role/	Role
☐	TestMySQLKerberosAuthentication-role-ob17or5c	/service-role/	Role
☐	wordpress-heimdall	/	Role

Other AWS accounts
Specify the AWS accounts that can use this key. Administrators of the accounts you specify are responsible for managing the permissions that allow their IAM users and roles to use this key. [Learn more](#)

arn:aws:iam:: 709960366167 :root [Remove](#)

[Add another AWS account](#)

[Cancel](#) [Skip to Review](#) [Previous](#) [Next](#)

4. Update the KMS key policy to allow the target account to use the key for decryption. This policy is automatically generated by the AWS KMS wizard and should be adjusted to your needs. Example with a working, simplified version:

```
{
  "Version": "2012-10-17",
  "Id": "key-consolepolicy-3",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
```

```

        "arn:aws:iam::272965818115:root",
        "arn:aws:iam::272965818115:user/arozyuk"
    ]
},
"Action": "kms:*",
"Resource": "*"
},
{
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::709960366167:root"
    },
    "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant",
        "kms:GenerateDataKey",
        "kms:Decrypt",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
]
}

```

5. If you want to use the secret in a different region, open the KMS key and select "Create new replica keys". Here we create a replica in the target region (eu-central-1).

The screenshot shows the AWS KMS console for a customer-managed key. A green notification bar at the top states: "Successfully created a replica of key mrk-385bacad281f4eaab797f0f0fc26829 in eu-central-1". The key ID is mrk-385bacad281f4eaab797f0f0fc26829. The console displays the following details:

- General configuration:**
 - Alias:** cross-account-symmetric-key
 - ARN:** arn:aws:kms:us-east-1:272965818115:key/mrk-385bacad281f4eaab797f0f0fc26829
 - Current key material ID:** 706c4ab7ee00cb296385ca2bda69665f70c185218ad58e52f854729fe94b2885
 - Status:** Enabled
 - Description:** Cross-account symmetric key for secret retrieval needed for AWS Security Competency docs, using private account for testing - 709960366167
 - Creation date:** Feb 12, 2026 15:00 GMT+1
 - Regionality:** Multi-Region primary
- Key policy, Cryptographic configuration, Tags, Key material and rotations, Regionality, Aliases:** The "Regionality" tab is selected.
- Primary key:** This is a multi-Region primary key. It has 1 replica. You can change any replica to the primary key. (Change primary Region button)
- Related multi-Region keys (1):**

Region	Key ARN	Status	Regionality
eu-central-1	arn:aws:kms:eu-central-1:272965818115:key/mrk-385bacad281f4eaab797f0f0fc26829	Creating	Replica

6. In the target account, create the secret using the customer-managed KMS key from the source account. The AWS Console/GUI does not allow selecting a KMS key from a different account, so use AWS CLI (CloudShell works well). Use the correct KMS key ARN (multi-Region key ARNs differ by region in ARN). Example:

```

aws secretsmanager create-secret \
  --name "heimdall/crossaccount-eu-central-1" \
  --description "Secret encrypted with Account 272965818115 Key" \
  --secret-string '{"username":"root-eu-central","password":"passwordeu"}' \

```

```
--kms-key-id arn:aws:kms:eu-central-1:272965818115:key/
mrk-385bacad281f4eaab797f0f06c26829 \
--region eu-central-1
```

This creates a secret in the target account, encrypted with the source account's KMS key. This example assumes the source account is in `us-east-1` and the secret is in `eu-central-1`, resulting in a cross-account and cross-region setup.

7. Add a resource policy to just created secret that allows the source account to retrieve it. Below is an example policy, which allows the IAM user arozuk (can be a role) from AWS account 272965818115 to read (GetSecretValue) secrets starting with heimdall/ in account 709960366167, region eu-central-1:

```
{
  "Version" : "2012-10-17",
  "Statement" : [ {
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::272965818115:user/arozuk"
    },
    "Action" : "secretsmanager:GetSecretValue",
    "Resource" : "arn:aws:secretsmanager:eu-central-1:709960366167:secret:heimdall/*"
  } ]
}
```

8. Configure Heimdall to use the secret:

- Use the full secret ARN (not just the name).
- Set the Region explicitly when the secret is in a different region than the Heimdall instance.

At this point the cross-account secret can be retrieved. The example below shows successful retrieval from the target account.

The screenshot shows the 'Secrets Manager Configuration' window. At the top right, there are 'Create New' and 'Commit' buttons. Below the title, there is a dropdown menu set to 'aws-cross-account' and an 'Enable' checkbox which is checked. The main configuration area contains several fields: 'Secrets Manager' is set to 'AWS Secrets Manager'; 'Endpoint' is 'Optional'; 'Region' is set to 'eu-central-1'; 'Access Key' is 'Optional'; 'Secret Key' is 'Optional'; and 'Test Secret Path' is set to 'arn:aws:secretsmanager:eu-central-1:709960366167:secret:heimdall/*'. At the bottom, there is a green button labeled 'Test secret retrieval (Passed)' with a green checkmark icon next to it.

CROSS-ACCOUNT CONFIGURATION (STS ASSUMEROLE + EXTERNAL ID)

This method uses an IAM Role with a Trust Policy and an External ID. This is the recommended security best practice for cross-account access, as it prevents the "Confused Deputy" problem and avoids sharing long-term user credentials. See [AWS Docs](#) for more information.

This example reuses the configuration made in the Cross-Account Configuration (IAM Policies + KMS Key Sharing) section, with adjustments for STS AssumeRole and External ID.

Example Cross-Account Configuration

- Source account (Heimdall runtime and KMS key owner): 272965818115
- Target account (secret owner): 709960366167
- Role Name used for STS: HeimdallCrossAccountAccess
- External ID: heimdall-external-id-99

Steps

1. Create an IAM Role in the **Source Account** (272965818115) named `HeimdallCrossAccountAccess` using a "Custom trust policy". This role will be assumed by the Heimdall application.
2. Configure the **Trust Relationship** (Trust Policy) to allow your principal (User or Service) to assume it, but **only** if the correct External ID is provided.

HeimdallCrossAccountAccess [Info](#) [Delete](#)

HeimdallCrossAccountAccess for Secret Manager in a different account

Summary [Edit](#)

Creation date
February 12, 2026, 19:25 (UTC+01:00)

ARN
[arn:aws:iam::272965818115:role/HeimdallCrossAccountAccess](#)

Link to switch roles in console
<https://signin.aws.amazon.com/switchrole?roleName=HeimdallCrossAccountAccess&account=272965818115>

Last activity
22 minutes ago

Maximum session duration
1 hour

Permissions

Trust relationships

Tags

Last Accessed

Revoke sessions

Trusted entities [Edit trust policy](#)

Entities that can assume this role under specified conditions.

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "HeimdallCrossAccountAccess",
6       "Effect": "Allow",
7       "Principal": {
8         "AWS": "arn:aws:iam::272965818115:user/arozuk"
9       },
10      "Action": "sts:AssumeRole",
11      "Condition": {
12        "StringEquals": {
13          "sts:ExternalId": "heimdall-external-id-99"
14        }
15      }
16    }
17  ]
18 }
```

Example Trust Policy enforcing External ID:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "HeimdallCrossAccountAccess",
      "Effect": "Allow",
```

```

    "Principal": {
      "AWS": "arn:aws:iam::272965818115:user/arozuk"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "sts:ExternalId": "heimdall-external-id-99"
      }
    }
  }
]
}

```

3. We need to create a secret in the target account (709960366167), and grant the role access to it. The instruction is pretty much the same as in the Cross-Account Configuration (IAM Policies + KMS Key Sharing) section (use that instruction if you do not have created secret with a customer managed kms key), then we only need to adjust the aws policy to allow our role to retrieve the secret.

```

{
  "Version" : "2012-10-17",
  "Statement" : [ {
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "arn:aws:iam::709960366167:role/HeimdallCrossAccountAccess"
    },
    "Action" : "secretsmanager:GetSecretValue",
    "Resource" : "arn:aws:secretsmanager:eu-central-1:709960366167:secret:heimdall/*"
  } ]
}

```

4. Update the **KMS key policy** to allow this role to decrypt. If you use a multi-Region key, update each replica's policy as well (policy changes are not auto-replicated). Example:

```

{
  "Version": "2012-10-17",
  "Id": "key-consolepolicy-3",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::709960366167:role/HeimdallCrossAccountAccess",
          "arn:aws:iam::272965818115:user/arozuk"
        ]
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow attachment of persistent resources",

```

```

    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::709960366167:root"
    },
    "Action": [
      "kms:CreateGrant",
      "kms:ListGrants",
      "kms:RevokeGrant",
      "kms:GenerateDataKey",
      "kms:Decrypt",
      "kms:DescribeKey"
    ],
    "Resource": "*"
  }
]
}

```

5. Attach a permissions policy to the `HeimdallCrossAccountAccess` role (in the target account). Adjust ARNs to your secret and KMS key. This grants access to the Secrets Manager resource and the KMS key created in the Cross-Account Configuration (IAM Policies + KMS Key Sharing) section:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToSecretAndKMSKey",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret",
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "arn:aws:secretsmanager:eu-central-1:709960366167:secret:heimdall/*",
        "arn:aws:kms:eu-central-1:272965818115:key/mrk-385bacad281f4eaab797f0f0f6c26829"
      ]
    }
  ]
}

```

6. Configure Heimdall:

- Enable **Use STS Role**.
- Set **STS Role ARN** to the target account role (`HeimdallCrossAccountAccess`), in my case it is `arn:aws:iam::272965818115:role/HeimdallCrossAccountAccess`.
- Set **STS Role External ID** to `heimdall-external-id-99` (must match the Trust Policy).

- Set **Region** if the secret is in a different region than the Heimdall instance.

7. Test the setup in Heimdall. Ensure the External ID matches the value configured in the role.

Secrets Manager Configuration

aws-cross-account ☒ Enable Create New Commit

Secrets Manager: AWS Secrets Manager

Endpoint: Optional

Region: eu-central-1

Access Key: Optional

Secret Key: Optional

Use STS Role: ☒

STS Role ARN: arn:aws:iam::272965818115:role/HeimdallCrossAccountAccess

STS External ID: heimdall-external-id-99

STS Session Name: Optional

Test Secret Path: arn:aws:secretsmanager:eu-central-1:709960366167:secret:heirr (optional)

Test secret retrieval (Passed) ☒

CyberArk Conjur

Following properties are available for CyberArk's Conjur connection configuration can be found in: [CyberArk Conjur Configuration](#)

Currently, it is possible to have **only one** secrets manager of this type.

Note: An Authn API Key and a whole configuration can be stored in the [AWS Secrets Manager](#). This option is supported only by the default Secrets Manager from the same Region/Account as the instance.

An example secret with an Authn API Key:

```
{
  "authnApiKey": "t740r1tmwhsa22yv0692d89fv12ft48c72j6mfs9hdm343vmcxyn"
}
```

An example secret with a whole config:

```
{
  "applianceUrl": "https://localhost:443",
  "account": "admin-account",
  "authnLogin": "admin@Test",
  "authnApiKey": "t740r1tmwhsa22yv0692d89fv12ft48c72j6mfs9hdm343vmcxyn"
}
```


Secrets Manager Configuration

Create New

CyberArk_Conjur

Enable

Commit

Secrets Manager:

CyberArk Conjur

Appliance URL:

https://localhost:443

Account:

admin-account

Authn Login:

admin@Test

Authn API Key:

••••••••

Authn URL:

Test Secret Path:

Test/secret

Secret

☐

Secret

☐

Test secret retrieval

Hashicorp Vault

Following properties are available for HashiCorp Vault server connection configuration:

HashiCorp Vault Configuration

Note: A whole configuration can be stored in the [AWS Secrets Manager](#). This option is supported only by the default Secrets Manager from the same Region/Account as the instance.

An example secret with a whole config:

```
{
  "url": "https://localhost:8201",
  "ignoreCertificateVerification": "true",
  "authConfig": {
    "token": "hvs.hTHvXRRqi154KKGgFldQuhqq",
    "authMethod": "TOKEN"
  }
}
```

HASHICORP VAULT TOKEN AUTHENTICATION

This authentication method allows to authenticate with Hashicorp Vault using a token directly.

- **Token:** Token used for authentication.

This token should have permissions appropriate to access deserved secret values.

Note: A token can be stored in the [AWS Secrets Manager](#). This option is supported only by the default Secrets Manager from the same Region/Account as the instance.

An example secret with a token:

```
{
  "token": "hvs.hTHvXRRqi154KKGgF1dQuhqq"
}
```

The screenshot shows the 'Secrets Manager Configuration' page in the AWS console. At the top right, there are buttons for 'Create New' and a menu icon. Below this, a dropdown menu is set to 'Hashicorp_Vault' with an 'Enable' checkbox checked. A green 'Commit' button is on the right. The main configuration area includes: 'Secrets Manager:' set to 'Hashicorp Vault' with a 'Secret' checkbox; 'Vault URL:' set to 'https://localhost:8201'; 'Ignore certificate verification:' checked; 'Auth method:' set to 'Token'; 'Token:' masked with dots and a 'Secret' checkbox; and 'Test Secret Path:' set to '/v1/kv/data/test'. A blue 'Test secret retrieval' button is at the bottom.

HASHICORP VAULT USERNAME AND PASSWORD AUTHENTICATION

This authentication method allows to authenticate with Hashicorp Vault using a username and password combination.

- **Auth mount path:** (Optional) Alternate authentication method mount path to be used. By default set to *userpass*.
- **Username:** Username used for authentication.
- **Password:** Password used for authentication.

The user connected to these credentials should have permissions appropriate to access deserved secret values.

Note: The username and password can be stored in the [AWS Secrets Manager](#). This option is supported only by the default Secrets Manager from the same Region/Account as the instance.

An example secret with username and password:

```
{
  "username": "user1",
  "password": "password1"
}
```

Secrets Manager Configuration

Create New

Hashicorp_Vault

Enable

Commit

Secrets Manager:

Hashicorp Vault

Vault URL:

https://localhost:8201

Ignore certificate verification:

☒

Auth method:

Username & Password

Auth mount path:

Optional

Username:

heimdall

Password:

••••••••

Test Secret Path:

/v1/kv/data/test

Test secret retrieval

Secret

Secret

HASHICORP VAULT APPROLE AUTHENTICATION

This authentication method allows to authenticate with Hashicorp Vault using a defined role. It is a recommended method to use with Heimdall since it is designed to be used by machines and applications rather than human operators.

- **Auth mount path:** (Optional) Alternate authentication method mount path to be used. By default set to *approle*.
- **Role ID:** The semi-secret identifier for the role that will authenticate to Vault. Think of this as the username portion of an authentication pair.
- **Secret ID:** The secret identifier for the role that will authenticate to Vault. Think of this as the password portion of an authentication pair.

The role should have permissions appropriate to access deserved secret values.

Note: The Role ID and Secret ID can be stored in the [AWS Secrets Manager](#). This option is supported only by the default Secrets Manager from the same Region/Account as the instance.

An example secret with an app role:

```
{
  "roleId": "fb3b1bfe-a3c4-908c-7b43-dcf3ee5fee72",
  "secretId": "544228c0-9940-6aed-7575-a8a6209ae294"
}
```

Secrets Manager Configuration

Create New

Hashicorp_Vault

Enable

Commit

Secrets Manager:

Hashicorp Vault

Vault URL:

https://localhost:8201

Ignore certificate verification:

☒

Auth method:

AppRole

Auth mount path:

approle

Role ID:

fb3b1bfe-a3c4-908c-7b43-dcf3ee5fee72

Secret ID:

••••••••

Test Secret Path:

/v1/kv/data/test

Test secret retrieval

It is possible to have **multiple** secrets managers of this type.

Secret path for KV engine should follow the specification for `vault kv get` command as described here (notice the difference between engine version 1 and 2) - [KV - Secrets Engines | Vault | HashiCorp Developer](#)

You can see what path to use for given secret in Hashicorp Vault GUI under Paths tab (API path).

Vault

Dashboard

Secrets Engines

Access

Policies

Tools

Monitoring

Client Count

Seal Vault

Secrets / kv / test / Paths

test

Overview Secret Metadata Paths Version History

Paths

API path

Use this path when referring to this secret in the API.

/v1/kv/data/test

CLI path

Use this path when referring to this secret in the CLI.

-mount="kv" "test"

API path for metadata

Use this path when referring to this secret's metadata in the API and permanent secret deletion.

/v1/kv/metadata/test

Commands

CLI `kv get`

This command retrieves the value from KV secrets engine at the given key name. See our [documentation](#) for other CLI commands.

```

vault kv get -mount="kv" "test"

```

API read secret version

This command obtains data and metadata for the latest version of this secret. In this example, Vault is located at https://127.0.0.1:8200. For other API commands, [learn more](#).

```

curl -X
--header "X-Vault-Token: ..." \
--request GET \
https://127.0.0.1:8200/v1/kv/data/test

```

In such case the secret path that You should put into our GUI is `/v1/kv/data/test`.

HASHICORP VAULT ENTERPRISE NAMESPACES

To use namespaces feature available in Hashicorp Vault Enterprise, see the documentation here - [Namespaces and SMT support in Vault | Vault | HashiCorp Developer](#).

First approach should be used, the name of the namespace should be put before the normal secret path, for example for namespace named `ns1` the secret path should be `ns1/kv/data/test`, with the assumption that the token configured in Heimdall Hashicorp Vault Configuration has access to this namespace and secret.

Testing secret manager

Before clicking the "Test secret retrieval" button, please make sure that you click the "Commit" button to test the newest configuration.

Secrets can be used for:

WIZARD:

- [Datasource credentials](#)
- [Cache credentials](#)

VDB:

- [VDB credentials](#)
- [Proxy configured users credentials](#)
- [Sync credentials](#)
- [Authentication test credentials](#)
- [Redis credentials](#)
- [JMX credentials](#)

DATASOURCE:

- [Datasource credentials](#)

Users:

- [Users credentials](#)

SMTP Configuration:

- [SMTP credentials](#)

Config Management

- HD credentials
- Server configuration through "hdSecretKey" property (AWS Secrets Manager only)

Hashicorp Secrets Manager Configuration (AWS):

- Hashicorp credentials
- Hashicorp token
- Hashicorp app role
- Hashicorp whole configuration

CyberArk Conjur Secrets Manager Configuration (AWS):

- CyberArk Conjur Authn API Key and a whole configuration

LDAP Configuration:

- LDAP credentials
- LDAP wizard credentials

Certificates:

- Certificate private key encryption password

Portal:

- JWT Signature Key

Example:

The screenshot shows the 'Data Sources' configuration page in Heimdall. At the top, there's a 'Create New Source' button and a menu icon. Below, a dropdown shows 'mysql-source' and an 'Enable' checkbox is checked. A green 'Commit' button is on the right. The 'Configuration' section contains several fields: 'Driver' is 'mysql-driver' with a '+' button; 'JDBC URL' is 'jdbc:mysql://localhost:3306'; 'Username' is 'root'; 'Secret Name' is 'awsSecret'; and 'Test query' is 'SELECT 1'. To the right of these fields is a 'Secret Secrets Manager' section with a checked checkbox and a dropdown menu showing 'AWS_Secrets_Managi'. At the bottom, a green button says 'Test Connection (Passed)' next to a green checkmark icon.

At least one Secrets Manager is required to be configured and enabled to see the secret checkbox. The selector will be shown if there is more than one Secrets Manager. When checking a checkbox, you have to specify a secret name to retrieve from the Secrets Manager.

In all credentials supported by Heimdall a secret must have one of the following formats:

```
{
  "username": "yourUsername",
  "password": "yourPassword"
}
```

```
{
  "user_name": "yourUsername",
  "password": "yourPassword"
}
```

```
{
  "user": "yourUsername",
  "password": "yourPassword"
}
```

The exception to this rule is [Certificate private key encryption password](#) (only password field is required) and [Server configuration through "hdSecretKey" property](#) where whole server configuration is stored in JSON format automatically by Heimdall.

Kerberos Configuration

This authentication type allows configuring Kerberos to authenticate users in the Heimdall Manager.

Administration

- Account Information
- License
- Alerts
- Login History
- Log Management
- Security Tags
- Software Management
- Notifications
- SMTP Configuration
- Server Properties
- Config Management
- LDAP Configuration
- Kerberos Configuration**
- Password Policy
- Modules

Kerberos Configuration

Enable Kerberos Authentication: ☒

Service Principal: HTTP/localhost

Keytab Location: /etc/krb5.keytab

Enable LDAP Groups Extraction: ☒

LDAP Configuration: HEIMDALL_LDAP_CONFI ▼

Commit

Kerberos configuration options can be found in: [Kerberos](#)

LDAP Configuration (Proxy and Central Manager)

LDAP Authentication provides authentication with using of LDAP server. LDAP Authentication provides two modes of authentication:

- Bind + Search Mode - authentication is made by binding to server as admin and searching information about authenticated user;
- Simple Mode - authentication is made by binding to server as authenticated user.

Bind and Search Mode

LDAP Configuration

LDAP WizardCreate New

HEIMDALL_LDAP_CONFIGURATION

Commit

- LDAP Configuration

LDAP(S) URL:

ldaps://localhost:636

Ignore LDAP Cert:

☒

Server Type:

Redhat IDM (FreeIPA)

Simple LDAP Mode:

☐

LDAP Security Principal:

uid=admin,cn=users,cn=accounts,dc=lab,dc=heimdalldata,dc=com

Secret

LDAP Search User Password:

.....

LDAP Sec. Security Principal:

uid=second-admin,cn=users,cn=accounts,dc=lab,dc=heimdalldata,dc=com

Secret

LDAP Sec. Search User Password:

.....

LDAP Search Domain:

dc=lab,dc=heimdalldata,dc=com

LDAP User Search Base:

cn=users,cn=accounts,dc=lab,dc=heimdalldata,dc=com

LDAP Name Attribute:

uid

LDAP Group Name Attribute:

cn

LDAP Group Filter:

(cn=*)

Use Raw Group Filter

☐

Use nested groups filter:

☐

LDAP Healthcheck:

☒

Cache groups' emails:

☒

Cache time:

1

h

Management privileges:

☒

Admin privilege group:

admins

Read Only privilege group:

database_users

+ Authentication Checker

All the ldap configuration options can be found in [LDAP Options](#)

By choosing option Simple LDAP Mode, you can switch to Simple Mode.

Note: This configuration supports rotating passwords, which means that in case of connection failure (due to credential rotation most likely) the system will silently pull the newest values and try to establish the connection again.

Advanced group filter

LDAP group filter is used to limit number of groups for look up for authenticated user. Example value describes filter limiting number of groups to only one particular group, but this option value is added as written, what enables writing more complex filters. Please look on below example.

Let's assume that we want our user be from groups `group1` or `group2`. By knowing the syntax of search filter used in LDAP server we can set `LDAP Group Filter` as:

```
(|(DistinguishedName=CN=group1,CN=Users,DC=example,DC=com)
(DistinguishedName=CN=group2,CN=Users,DC=example,DC=com))
```

Important: If there is an LDAP Group Filter specified at least one group has to be extracted to authenticate the user.

Use Raw Group Filter

By default, the application automatically constructs LDAP group queries by appending standard group constraints (such as `objectCategory=group` or `objectclass=posixgroup`) and user membership clauses to your configured LDAP Group Filter. This automated approach ensures that the correct groups are extracted for the authenticated user in most standard environments. However, certain directory structures require highly specific filtering logic that may conflict with these default additions. The **Use Raw Group Filter** option allows administrators to bypass the default behavior and take full control over the group extraction query.

A common scenario where this feature is beneficial involves managing groups in **Active Directory**. Active Directory typically contains both **Security Groups** (often used for access control) and **Distribution Groups** (used for email distribution lists). Depending on an organization's requirements, administrators may prefer to streamline access management and optimize performance by importing *only* Security Groups, as Distribution Groups are generally not used for resource access. Because the default application query automatically extracts all group types, the Raw Group Filter provides the exact level of control needed to exclude Distribution Groups entirely.

By enabling this option, the system stops appending default constraints and executes your `LDAP Group Filter` value exactly as provided. This means the filter must be complete and self-contained, including the user membership clause for your directory.

For this purpose, the following aliases are available:

- `${user}` - The username used to authenticate (for example, `admin`).

- `${dn}` - The authenticated user's distinguished name (for example, `uid=admin,cn=users,cn=accounts,dc=lab,dc=heimdalldata,dc=com`).

Example for Active Directory:

```
(&(objectCategory=group)(groupType:1.2.840.113556.1.4.803:=2147483648)
(member=${dn}))
```

The filter above expands to the following at runtime:

```
(&(objectCategory=group)(groupType:1.2.840.113556.1.4.803:=2147483648)
(member=CN=admin,CN=Users,DC=ad,DC=heimdalldata,DC=com))
```

You can also build the DN yourself with `${user}` and get the same result:

```
(&(objectCategory=group)(groupType:1.2.840.113556.1.4.803:=2147483648)
(member=CN=${user},CN=Users,DC=ad,DC=heimdalldata,DC=com))
```

Explanation:

- `(objectCategory=group)` - Searches only for groups.
- `(groupType:1.2.840.113556.1.4.803:=2147483648)` - Filters only Security Groups.
- `(member=...)` - Limits results to groups that include the authenticated user (use `${dn}` or build the DN with `${user}`).

Note: You can enable **Verbose Debug Mode** to see the exact filter being used, then adjust it as needed.

LDAP Configuration

LDAP Wizard

Create New

LDAP_AD_CN_FULL_NAME

Commit

LDAP Configuration

LDAP(S) URL:

ldap://localhost:3000

Server Type:

Active Directory

Simple LDAP Mode:

☐

LDAP Security Principal:

cn=admin,cn=users,dc=ad,dc=heimdalldata,dc=com

Secret

☐

LDAP Search User Password:

.....

LDAP Sec. Security Principal:

CN=\${user},CN=users,DC=heimdalldata,DC=com

Secret

☐

LDAP Sec. Search User Password:

password

LDAP Search Domain:

DC=ad,DC=heimdalldata,DC=com

LDAP User Search Base:

CN=Users,DC=ad,DC=heimdalldata,DC=com

LDAP Name Attribute:

cn

LDAP Group Name Attribute:

cn

LDAP Group Filter:

(&(objectCategory=group)(groupType:1.2.840.113556.1.4.803:=2

Use Raw Group Filter:

☒

Use nested groups filter:

☐

LDAP Healthcheck:

☒

Cache groups' emails:

☐

Management privileges:

☐

Authentication Checker

LDAP Healthcheck

If enabled, performs in the background automatically every minute and may result in two alerts: once the server is down, and once the server is back online.

LDAP Server associated with LDAP Configuration: HEIMDALL_LDAP_CONFIGURATION is online.

✓ ✕

LDAP Server associated with LDAP Configuration: HEIMDALL_LDAP_CONFIGURATION is either offline or unreachable.

✓ ✕

Nevertheless, it can be performed on demand by using the button with hearth - if used, the JSON with the details of LDAP Healthcheck result will be opened in the new tab.

LDAP Configuration

LDAP Wizard

Create New

HEIMDALL_LDAP_CONFIGURATION

Commit

Important: LDAP Healthcheck on demand will be performed on **ALL** configurations with LDAP Healthcheck checkbox enabled.

Simple Mode

The screenshot shows the 'Administration' section of a web interface. On the left is a sidebar menu with various configuration options. The 'LDAP Configuration' option is highlighted. The main content area is titled 'LDAP Configuration' and contains several settings. At the top right of this section are buttons for 'LDAP Wizard', 'Create New', and a 'Commit' button. Below these is a dropdown menu set to 'HEIMDALL_LDAP_CONFIGURATION'. The 'LDAP Configuration' section is expanded, showing the following settings: 'LDAP(S) URL' is 'ldaps://localhost:636'; 'Ignore LDAP Cert' is checked; 'Server Type' is 'Redhat IDM (FreeIPA)'; 'Simple LDAP Mode' is checked; 'LDAP Prefix' is 'uid='; and 'LDAP Suffix' is ',cn=users,cn=accounts,dc=lab,dc=heimdalldata,dc=com'. Below these settings is a '+ Authentication Checker' button.

Administration

- Account Information
- License
- Alerts
- Login History
- Log Management
- Security Tags
- Software Management
- Notifications
- Portal
- SMTP Configuration
- Server Properties
- Config Management
- Secrets Manager Configuration
- LDAP Configuration**
- Kerberos Configuration
- SAML Configuration
- Password Policy
- Modules
- TCP
- AWS Endpoints

LDAP Configuration

HEIMDALL_LDAP_CONFIGURATION

LDAP Configuration

LDAP(S) URL: ldaps://localhost:636

Ignore LDAP Cert: ☒

Server Type: Redhat IDM (FreeIPA)

Simple LDAP Mode: ☒

LDAP Prefix: uid=

LDAP Suffix: ,cn=users,cn=accounts,dc=lab,dc=heimdalldata,dc=com

+ Authentication Checker

Simple Mode is simpler mode than Bind + Search mode, because uses only one request (which is binding request) to LDAP server.

Simple Mode can be turned on by choosing Active Directory (LDAP) Auth Enabled checkbox, and then choosing Simple LDAP Mode checkbox. Simple Mode requires to set below options to work properly.

All the ldap configuration options can be found in [LDAP Options](#)

To understand working of this mode, should be known that send request is binding composed of three parts: `prefix + username + suffix`. For given example values and user `exampleuser`, binding request would look like `CN=exampleuser,CN=Users, DC=example, DC=com`.

Authentication Checker

The screenshot shows a web application interface for LDAP Configuration. On the left is a sidebar menu with various options: Account Information, License, Alerts, Login History, Log Management, Security Tags, Software Management, Notifications, Portal, SMTP Configuration, Server Properties, Config Management, Secrets Manager Configuration, LDAP Configuration (highlighted), Kerberos Configuration, SAML Configuration, Password Policy, Modules, TCP, and AWS Endpoints. The main content area is titled 'LDAP Configuration' and contains a dropdown menu set to 'HEIMDALL_LDAP_CONFIGURATION'. Below this are two expandable sections: '+ LDAP Configuration' and '- Authentication Checker'. The 'Authentication Checker' section is active, showing input fields for 'User' (admin) and 'Password' (masked with dots). A green 'Authenticate' button with a checkmark icon is at the bottom of this section. A notification banner at the top right states: 'Authentication successful. Groups extracted: "admins", "postgres"'. In the top right corner of the main area, there are buttons for 'LDAP Wizard', 'Create New', and a menu icon, along with a green 'Commit' button.

The Authentication Checker is a simple tool that allows you to test LDAP authentication against a specific configuration. In case of successful authentication extracted groups will be listed, as long as LDAP Group Filter is specified.

Central Manager Information

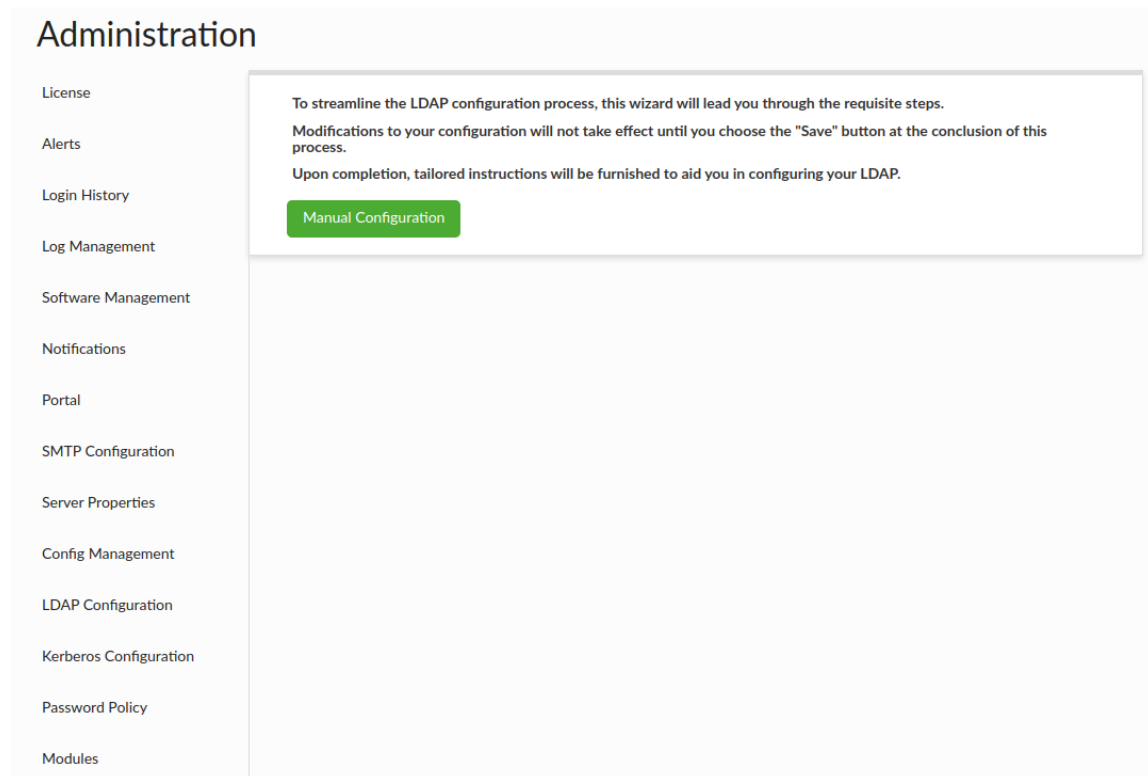
If you want to authenticate CM user, in the [Users tab](#) you should create new user(-s) with the same name as in your active directory. During creating new user don't forget to mark the "Authenticate By LDAP" option.

LDAP Configuration Wizard

The LDAP Configuration Wizard is provided to simplify configuration process as much as possible. It's a walk-through tool preventing from going forward if provided configuration is not

valid. It can be accessed by clicking the "LDAP Wizard" button in the LDAP Configuration section.

Bind and Search Mode



The screenshot shows the 'Administration' page with a sidebar on the left containing various system management options. The 'LDAP Configuration' option is highlighted. A modal window is open in the center, providing instructions for the LDAP configuration wizard. The modal text states: 'To streamline the LDAP configuration process, this wizard will lead you through the requisite steps. Modifications to your configuration will not take effect until you choose the "Save" button at the conclusion of this process. Upon completion, tailored instructions will be furnished to aid you in configuring your LDAP.' Below the text is a green button labeled 'Manual Configuration'.

Administration

- License
- Alerts
- Login History
- Log Management
- Software Management
- Notifications
- Portal
- SMTP Configuration
- Server Properties
- Config Management
- LDAP Configuration**
- Kerberos Configuration
- Password Policy
- Modules

To streamline the LDAP configuration process, this wizard will lead you through the requisite steps.
Modifications to your configuration will not take effect until you choose the "Save" button at the conclusion of this process.
Upon completion, tailored instructions will be furnished to aid you in configuring your LDAP.

Manual Configuration

There is the list of information necessary to finalize the Wizard Configuration:

- **LDAP(S) URL** - Specifies URL of LDAP server.
- **Ignore LDAP Cert** - Specifies if TLS validation of LDAP server certificate should be performed.
- **LDAP Security Principal** - Specifies name of user used to search for authenticated user.
- **LDAP Sec. Security Principal** - Specifies name of user used to search for authenticated user if the primary search fails.
- **LDAP Search User Password** - Specifies password for LDAP Security Principal user used to search for authenticated user.
- **LDAP Sec. Search User Password** - Specifies password for LDAP Sec. Security Principal user used to search for authenticated user.
- **LDAP Security Principal Secret Name** - Specifies the name of a Secret containing the credentials (user/username and password) of the LDAP Security Principal. Available only if 'Secret' is enabled.

- **LDAP Sec. Security Principal Secret Name** - Specifies the name of a Secret containing the credentials (user/username and password) of the LDAP Sec. Security Principal. Available only if 'Secret' is enabled.
- **LDAP Search Domain** - Specifies the domain in which authenticated user's groups will be searched.
- **LDAP User Search Base** - Specifies user search base in which authenticated user will be searched.
- **LDAP Name Attribute** - Specifies attribute which should be used to identify a user.
- **LDAP Group Name Attribute** - Specifies attribute, which should be used to extract information about a group (Optional).
- **LDAP Group Filter** - Specifies group filter, which will be added during a search for a user's groups (Optional).
- **Use Raw Group Filter** - Uses the LDAP Group Filter value as provided, without appending any default filters or implicit constraints (Optional).
- **Use nested groups filter** - Specifies if parent groups should be included, when searching for user's groups.
- **LDAP Healthcheck** - Performs LDAP Healthcheck automatically every minute. An appropriate alert will be shown if the server goes down (Similarly, once the server is back online).
- **Cache groups' emails** - Specifies whether emails for LDAP groups should be cached (applies only if LDAP configuration is linked to the portal configuration).
- **Cache time** - Specifies the interval at which emails in the cache should be updated.
- **Management privileges** - Determines whether it entitles a user to use admin/read only privileges by requiring membership in a specific group.
- **Admin privilege group** - To be authorized to use the Management Privilege: Admin, the user must be a member of the selected group.
- **Read Only privilege group** - To be authorized to use the Management Privilege: Read Only, the user must be a member of the selected group.

Service Account Validation

At the very beginning, the absolute basis of the correct and valid configuration:

LDAP Configuration Wizard

Commence by furnishing the URL, DN (Distinguished Name) of the service account, and the associated password.

If the URL includes the "ldaps" suffix, you will be presented with an additional option to bypass LDAP certificate validation.

Rather than providing only a single URL, it is possible to furnish a list of URLs separated by spaces. In this scenario, the LDAP provider will endeavor to establish a successful connection by sequentially attempting each URL in the list.

To ascertain the feasibility of establishing a connection to the LDAP Server based on the provided configuration, utilize the "Verify" button.

Configuration Name:	LDAP_CONFIG_NAME	
LDAP(S) URL:	ldap://heimdalldata.com:389	
Server Type:	Other	▼
Simple LDAP Mode:	☐	
LDAP Security Principal:	CN=\${user},CN=users,DC=heimdalldata,DC=com	Secret ☐
LDAP Search User Password:	password	
LDAP Sec. Security Principal:	CN=\${user},CN=users,DC=heimdalldata,DC=com	Secret ☐
LDAP Sec. Search User Password:	password	

Verify

- The fields above are required as the authentication is made by binding to server as admin and searching information about authenticated user.
- The grayed-out "Verify" button will remain gray until all required fields are filled in.
- When all the required fields are filled in, the button will turn green and when pressed, the connection to AD will be attempted.
- If it fails, a corresponding message with the reason for the error will be displayed.
- if it succeeds, the next Wizard stage will be unlocked.

If one of the available [Secrets Managers](#) is configured, then it can be used to provide credentials for LDAP (Sec.) Security Principal.

In such case it is obligatory to provide either '\${username}' or '\${user}' placeholder where the actual username (retrieved from Secrets Manager, based on LDAP (Sec.) Security Principal Secret Name) has to be injected in the distinguished name (DN).

Wrong way:

LDAP Security Principal:	CN=user,CN=users,DC=heimdalldata,DC=com	Secret ☒

Usage of Secret requires \${user} or \${username} placeholder, which will be replaced with the username from LDAP Security Principal Secret Name.

LDAP Security Principal Secret Name:	abcd1234efgh5678ijkl9012mnop3456

Good way:

LDAP Security Principal:	CN=\${user},CN=users,DC=heimdalldata,DC=com	Secret ☒
LDAP Security Principal Secret Name:	abcd1234efgh5678ijkl9012mnop3456	

Users and Groups Search Details

The next step is necessary to properly look for information about the user being authenticated and the groups he belongs to + set whether nested groups should be searched for.

The LDAP Configuration Wizard has successfully retrieved available domains and user search bases, which can now be accessed in the **LDAP Search Domain** and **LDAP User Search Base** sections, respectively.

Additionally, the **LDAP (Group) Name Attribute** includes a counter indicating the number of users or groups that possess this specific attribute.

Upon selecting an **LDAP Name Attribute**, all discovered users will be displayed. These values will serve as user logins for authentication purposes.

Please complete the fields below and click "**Find Groups**" to conduct a comprehensive search across all configured groups on the server. This will facilitate the creation of further LDAP Group Filters.

You have the option to customize all the fields below by checking the appropriate checkboxes on the right.

LDAP Search Domain:		▼	☐
LDAP User Search Base:		▼	☐
LDAP Name Attribute:		▼	☐
LDAP Group Name Attribute:		▼	☐
Use nested groups filter:	☐		
Cache groups' emails:	☒		
Cache time:	1	h	▼
LDAP Healthcheck:	☒		

Find Groups

- In order to simplify configuration some of the fields can be filled using dropdowns (LDAP Search Domain and LDAP Search Base contains all possible options straight from LDAP Server).
- LDAP (Group) Name Attribute contains all attributes which are actually used among all configured users / groups with additional counter of how many users / groups actually contain particular attribute.
- Additional "Example users found" row will be shown after LDAP Name Attribute selection. This row is just a preview and has no affect on the final configuration, but what is important - these values will be treated as logins for users to authenticate.
- Caching emails for LDAP groups is enabled by default, with a cache update time of 1 hour. This can be customized to fit your needs or disabled altogether. However, the minimum update time must be at least 5 minutes.
- All of these fields can be customized after checking suitable checkbox on the right.
- When all the fields are filled in, the button will turn green and when pressed, the attempt to find Groups configured on the server will be performed.

The LDAP Configuration Wizard has successfully retrieved available domains and user search bases, which can now be accessed in the **LDAP Search Domain** and **LDAP User Search Base** sections, respectively.

Additionally, the **LDAP (Group) Name Attribute** includes a counter indicating the number of users or groups that possess this specific attribute.

Upon selecting an **LDAP Name Attribute**, all discovered users will be displayed. These values will serve as user logins for authentication purposes.

Please complete the fields below and click "**Find Groups**" to conduct a comprehensive search across all configured groups on the server. This will facilitate the creation of further LDAP Group Filters.

You have the option to customize all the fields below by checking the appropriate checkboxes on the right.

LDAP Search Domain:	dc=lab,dc=heimdalldata,dc=com	▼	☐
LDAP User Search Base:	cn=users,cn=accounts,dc=lab,dc=heimdalldata,dc=com	▼	☐
LDAP Name Attribute:	uid / 24	▼	☐
Example users found:	schen	▼	
LDAP Group Name Attribute:	cn / 32	▼	☐

Use nested groups filter: ☐

Cache groups' emails: ☒

Cache time: 1 h ▼

LDAP Healthcheck: ☒

Find Groups

LDAP Group Filter Creation

The almost last step of the Wizard is to select Groups of interest for the DB i.e. Groups which will be relevant to database roles. Based on the selected Group the LDAP Group Filter will be generated when you click "Generate Filter" button. If no group is selected then no filter will be provided and this is ok - LDAP Group Filter is optional.

Found groups overall: 20

Please choose a subset of groups that are pertinent to the roles within the database.

Subsequently, an appropriate **LDAP Group Filter** will be generated based on your selection.

If no groups are chosen, no filter will be generated.

Wildcards: ? - any single character * - any number of any characters

Find Groups: adm **Find**

Selected Groups:

database_users **Remove** ⓘ
vm-managers **Remove** ⓘ

Found Groups:

greenplum-admin **Add** ⓘ
postgres-admin **Add** ⓘ
admins **Add** ⓘ

Generate Filter

LDAP Group Filter: ((cn=database_users)(cn=vm-managers))

Use Raw Group Filter ☐

Management Privileges

As an option, Management Privileges can be enabled to enhance security. When activated, only members of the specified groups will have access to the assigned in the Users tab Admin or Read-Only privileges.

Management privileges:	☒
Admin privilege group:	admins ▼
Read Only privilege group:	database_users ▼

Authentication Checker

Provided configuration can be verified with the last part of the LDAP Configuration Wizard - [Authentication Checker](#).

Authentication Checker

User:	user
Password:	password

Test LDAP

Simple Mode

In Simple Mode, unlike Bind + Search Mode, only the url suffice to start the configuration. Nevertheless what is important, the LDAP Configuration Wizard **does not work with Active Directory**, as this one has a strong security authentication and requires a successful bind before any data can be retrieved. Regardless of the success of the scan the next part of the Wizard will be unlocked.

LDAP Configuration Wizard

At the beginning provide the URL and DN of the service account + password.

URL with ldaps suffix will show additional option to bypass LDAP certificates validation.

Use **Scan** button to continue.

Ldap Configuration Wizard doesn't work for Active Directory in Simple Mode!

LDAP(S) URL: ldap://localhost:2137

Simple LDAP Mode: ☒

It may take a while as in successful case all the necessary data about LDAP Server will be retrieved in order to simplify next steps of the Wizard.

Scan

Available prefixes and suffixes has been retrieved and are available under **LDAP Prefix** and **LDAP Suffix**, respectively.

Nevertheless, both these fields below can be customized after checking suitable checkbox on the right.

LDAP Prefix: uid= ☐

LDAP Suffix: ,cn=users,cn=accounts,dc=lab,dc=heimdalldata,dc=com ☐

After successful scan all possible LDAP Prefixes and LDAP Suffixes will be available in a drop-down list. After failed scan drop-down lists will be empty, nevertheless both these fields can be customized after checking suitable checkbox on the right.

At this point your Simple LDAP Configuration is complete and after saving it can be found under "LDAP Configuration".

SAML 2.0 Configuration Guide

Heimdall supports Single Sign-On (SSO) through SAML 2.0, allowing authentication via external Identity Providers (IdPs) such as **AWS IAM Identity Center**, **Okta**, and others.

To enable SAML communication with Heimdall, the Identity Provider (IdP) must be configured to send data to the correct Heimdall endpoints. Below are the key configurations required (choose one either http or https protocol):

Application ACS URL: (endpoint where an Identity Provider (IdP) sends a SAML response)

- http:// <heimdall_hostname> :8087/login/saml2/sso/heimdall

- https:// <heimdall_hostname> :8443/login/saml2/sso/heimdall

Application SAML Audience: (recipient URL of a SAML assertion)

- http:// <heimdall_hostname> :8087/saml2/service-provider-metadata/heimdall

- https:// <heimdall_hostname> :8443/saml2/service-provider-metadata/heimdall

⌘ Configuration Parameters

Parameter	Description
Auth Provider	Identity provider (AWS IAM Identity Center, Okta, Other)
IdP Metadata URL	The URL pointing to the Identity Provider's metadata.
AWS Identity Store Id	This option is available only when using AWS IAM Identity Center. Since AWS provides groups as UUIDs, this setting enables mapping those UUIDs to their display names by connecting to the appropriate Identity Store. The mappings are stored in a temporary internal cache for up to 1 hours. If a group is renamed in AWS, click the Commit button to refresh the configuration and clear the cache. If you leave this field blank, group UUIDs will be shown instead of names. Users can manually map UUIDs to the corresponding names in the Datasource tab using the Group Mapping feature for proper role-base session requesting in the portal. For more details, see the Group Mapping section .

To disable SAML authentication, clear the **IdP Metadata URL** field in the configuration. Without this URL, the SAML authentication option will be disabled.

Setting Up SAML with AWS IAM Identity Center

This guide explains how to configure SAML authentication with **AWS IAM Identity Center** using **SAML 2.0** to enable Single Sign-On (SSO).

CREATE A NEW APPLICATION IN AWS

1. Go to **IAM Identity Center → Applications**
2. Choose: **Customer managed → Add application**
3. Select:
 - **Setup preference:** *I have an application I want to set up*
 - **Application type:** *SAML 2.0*
4. Fill in the **Display Name** and **Description**
5. Application properties are optional and may be left unspecified.

OBTAIN METADATA URL

On the IAM Identity Center metadata section, AWS will provide a **SAML metadata file URL**. This will be used in Heimdall during the SAML setup.

IAM Identity Center metadata

Your cloud application may require the following certificate and metadata details to recognize IAM Identity Center as the identity provider.

IAM Identity Center SAML metadata file

[Download](#)

<https://portal.sso.us-east-1.amazonaws.com/saml/metadata/MjcyOTY1ODE4MTE1X2lucy1kMDc5MzllNmQxYzUyYTc3>

CONFIGURE APPLICATION METADATA IN AWS

You have **two options** to configure metadata in AWS:

Option 1: Manually type your metadata values

Application ACS URL: (Replace `<heimdall_hostname>` with your actual hostname)

- `http:// <heimdall_hostname> :8087/login/saml2/sso/heimdall`
- `https:// <heimdall_hostname> :8443/login/saml2/sso/heimdall`

Application SAML Audience: (Replace `<heimdall_hostname>` with your actual hostname)

- `http:// <heimdall_hostname> :8087/saml2/service-provider-metadata/heimdall`
- `https:// <heimdall_hostname> :8443/saml2/service-provider-metadata/heimdall`

Application metadata
IAM Identity Center requires specific metadata about your cloud application before it can trust this application. You can type this metadata manually or upload a metadata exchange file.

☒ Manually type your metadata values ☐ Upload application SAML metadata file

Application ACS URL
`https://localhost:8443/login/saml2/sso/heimdall`

Application SAML audience
`https://localhost:8443/saml2/service-provider-metadata/heimdall`

Cancel Submit

⚠ Important:

These URL paths (`/login/saml2/sso/heimdall` and `/saml2/service-provider-metadata/heimdall`) **must not be changed**.

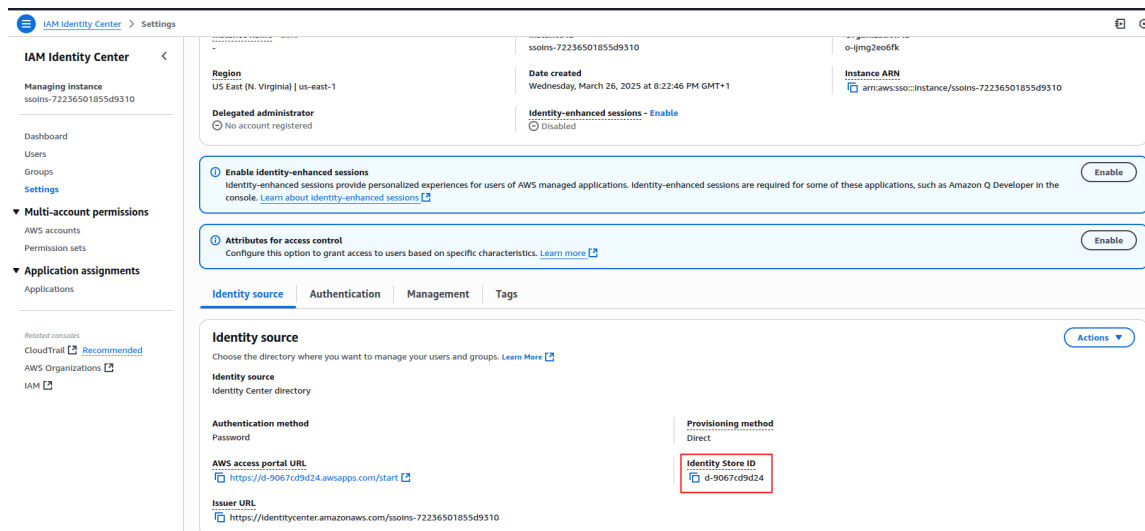
Altering them may result in SAML not working properly.

Option 2: Upload application SAML metadata file

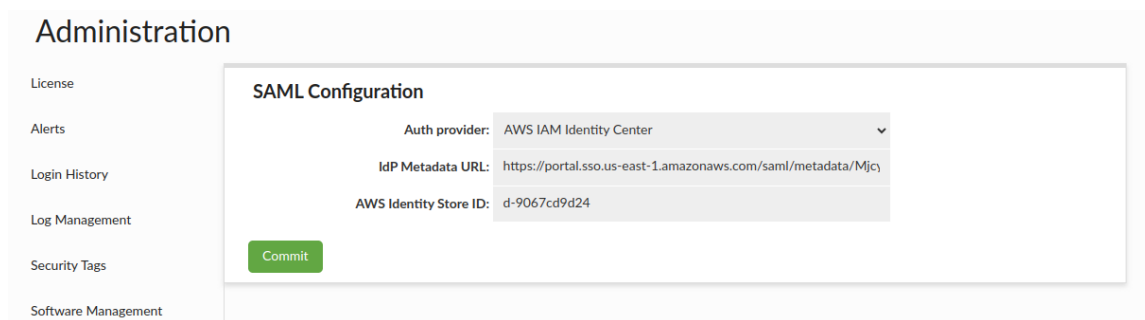
1. In Heimdall Central Manager:

- Log in with **admin privileges**
- Navigate to: Admin → SAML Configuration
- Fill in:
 - **Auth provider** as AWS IAM Identity Center
 - **IdP Metadata URL** (e.g. `https://portal.sso.us-east-1.amazonaws.com/saml/metadata/<random_string>`)
 - **AWS Identity Store ID** to automatically retrieve AWS groups associated with users
- Click **Commit**
- **Restart Heimdall Central Manager**

The AWS Identity Store Id can be found on aws in the IAM Identity Center → Settings → Identity source → Identity Store ID



At the end, the configuration will resemble the following:



1. After the restart, download the metadata file from heimdall endpoints:

- `http://<heimdall_hostname>:8087/saml2/metadata`
- `https://<heimdall_hostname>:8443/saml2/metadata`

i Note:

The SAML configuration XML is retrieved from a metadata URL (/saml2/metadata endpoint) via HTTP or HTTPS. The retrieved XML contains URLs that reflect the same protocol used during retrieval. AWS will subsequently use the same protocol (HTTP or HTTPS) found within the retrieved XML for communication. Users should therefore be mindful of the protocol used when accessing the metadata URL, as this will dictate the protocol AWS employs.

2. In **AWS IAM Identity Center**:

- Choose: **Upload application SAML metadata file**
- Upload the file downloaded from Heimdall

AWS will auto-fill the ACS URL and Audience based on the uploaded file. The values should look like that:

Application metadata

IAM Identity Center requires specific metadata about your cloud application before it can trust this application. You can type this metadata manually or upload a metadata exchange file.

Manually type your metadata values

Upload application SAML metadata file

Application ACS URL

https://localhost:8443/login/saml2/sso/heimdall

Application SAML audience

https://localhost:8443/saml2/service-provider-metadata/heimdall

Cancel

Submit

ASSIGN USERS TO THE APPLICATION

Don't forget to assign users to the SAML application in **AWS IAM Identity Center**.

Without this step, users will not be able to authenticate via SAML.

IAM Identity Center

Managing Instance
ssoins-72236501855d9310

Dashboard

Users

Groups

Settings

Multi-account permissions

Application assignments

Applications

Related consoles

CloudTrail

AWS Organizations

IAM

Latest test saml2

Successfully assigned 1 user

Details

Display name

Latest test saml2

Service

Custom SAML 2.0 application

Description

Latest test saml2

Status

Inactive

Application ARN

arn:aws:sso::272965818115:application/ssoins-72236501855d9310/apl-d07939e6d1c52a77

Owning account ID

-

User and group assignments

Require assignments

Authentication with trusted token issuer

Not configured

Date created

April 17, 2025

Assigned users and groups (1)

The following users and groups can access this application. [Learn more about user and group assignments](#)

Search for an assigned user or group

Username of user or group

Type

test

User

You can assign groups for a particular user, which will be extracted during user login.

IAM Identity Center

Managing Instance
ssoins-72236501855d9310

Dashboard

Users

Groups

Settings

Multi-account permissions

Application assignments

Applications

test

Reset password

Delete user

Disable user access

General information

Profile

Groups (2)

AWS accounts

Applications

MFA devices (1)

Active sessions (1)

Group memberships (2)

To grant permission to this user, add the user to a group that has access to AWS accounts or cloud applications. [Learn more](#)

Find groups by group name

Group name

Description

admins

test group

jenkins

test group

To configure the attribute mappings in AWS Identity Center, navigate to the application settings and select "Edit attributes mappings".

IAM Identity Center > Applications > Latest test saml2

SAML assertions successfully updated.

Latest test saml2

Details

Display name
Latest test saml2

Service
Custom SAML 2.0 application

Description
Latest test saml2

Status
Active

Application ARN
arn:aws:iam::27236501855d9310:application/saml2-d07939e6d1c52a77

Owning account ID
-

User and group assignments
Require assignments

Authentication with trusted token issuer
Not configured

Date created
April 17, 2025

Actions

Edit configuration

Edit attribute mappings

Assigned users and groups (1)

The following users and groups can access this application. [Learn more about user and group assignments](#)

Search for an assigned user or group

☐	Username of user or group	Type
☐	test	User

Remove access Assign users and groups

Ensure that the Subject attribute is correctly mapped to `${user:subject}`.

To retrieve groups from AWS Identity Center, map the corresponding attribute to `${user:groups}`.

⚠ Important:

If you intend to use groups, make sure that every user has at least one group assigned. AWS will fail SAML assertion validation for any user without groups, resulting in authentication errors.

IAM Identity Center > Applications > Latest test saml2 > Attribute mappings

Attribute mappings for Latest test saml2

Attributes you map here become part of the SAML assertion that is sent to the application. You can choose which user attributes in your application map to corresponding user attributes in your connected directory. [Learn more](#)

User attribute in the application	Maps to this string value or user attribute in IAM Identity Center	Format	
subject	<code>\${user:subject}</code>	unspecified	Remove
groups	<code>\${user:groups}</code>	unspecified	

Add new attribute mapping

Cancel Save changes

ENABLING SAML AUTHENTICATION FOR HEIMDALL USERS

1. In Heimdall, go to the **Users** tab
2. Click **Create New User**
3. Check: **Authenticate By SAML2**
4. Save the user

Users

Create New User

test

☒ Enable

Commit

Configuration

Username:

test

Email:

email@heimdall.com

Management Privilege:

Admin

Audit User:

☒ Note: User must re-login to view audit trails

Portal User:

☒

Authentication mode:

SAML

Groups

Add:

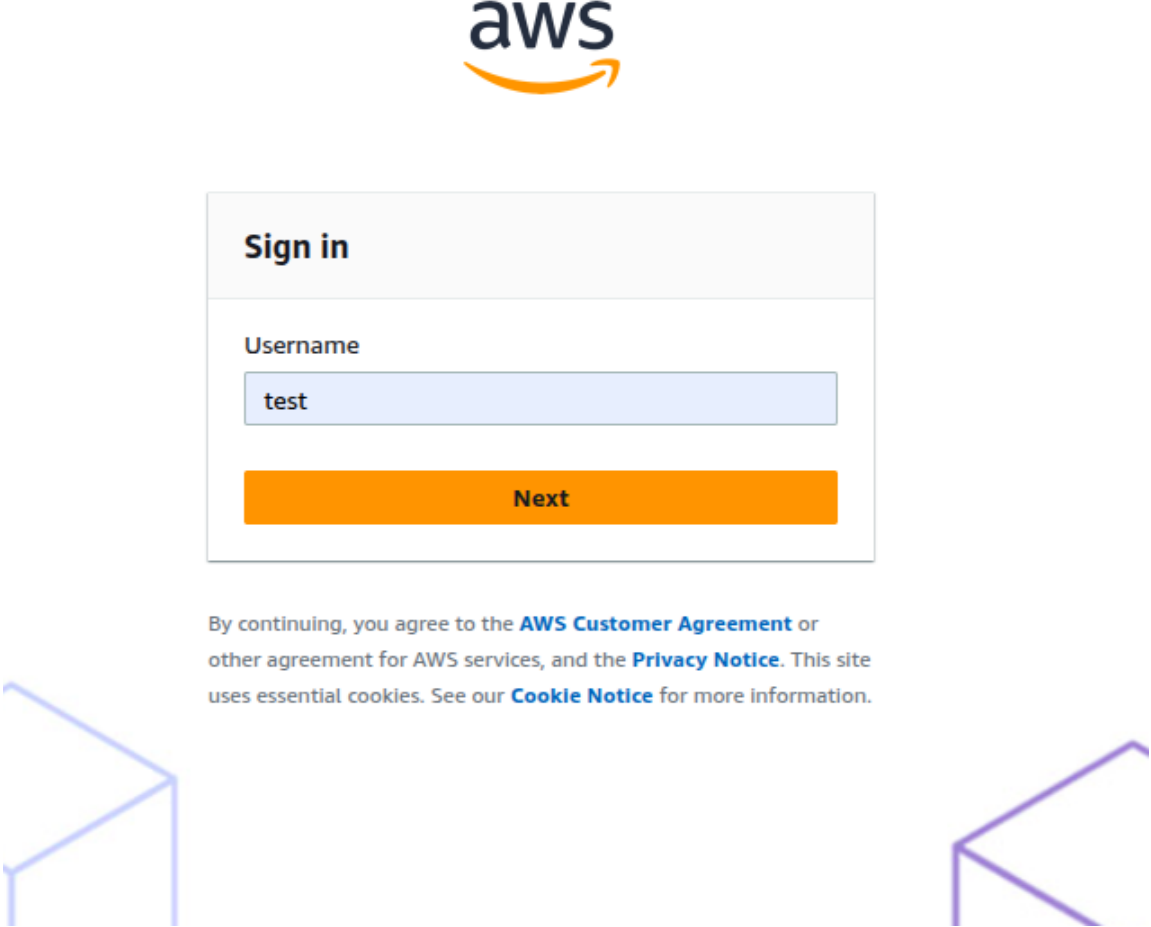
+

 Remove:

+

In the **Users** tab, administrators can manually assign groups to a user for portal use cases.

Now, when the user clicks **Login with SAML** (!use proper protocol of HTTP/HTTPS as configured on the aws), they will be redirected to AWS for authentication and then back to Heimdall as a logged-in user.



[Not secure](#) <https://localhost:8443/#/status>

HEIMDAL DATA

Monitor

Status

- Dashboard
- Analytics
- Logs

Configuration

- Wizard
- Virtual DB
- Data Sources
- Rules
- Drivers

Status

Status	Virtual Database	Servers	Development																												
	MySQLVDB :3307 Proxy starts: 2	<table border="1"> <thead> <tr> <th>Status</th> <th>Data Source</th> <th>Server Name</th> <th>Last Heartbeat Time</th> <th>Queries/Sec.</th> <th>Connections</th> <th>Replica Lag (ms)</th> </tr> </thead> <tbody> <tr> <td>✓</td> <td>mysqlDataSourceLocal</td> <td>database-1-instance-1</td> <td>2025-04-23 17:47:18 (+0200)</td> <td>0</td> <td>0</td> <td>-</td> </tr> <tr> <td>✓</td> <td>mysqlDataSourceLocal</td> <td>database-1-instance-3</td> <td>2025-04-23 17:47:18 (+0200)</td> <td>0</td> <td>0</td> <td>-</td> </tr> <tr> <td>✓</td> <td>mysqlDataSourceLocal</td> <td>database-1-reader-test</td> <td>2025-04-23 17:47:18 (+0200)</td> <td>0</td> <td>0</td> <td>-</td> </tr> </tbody> </table>	Status	Data Source	Server Name	Last Heartbeat Time	Queries/Sec.	Connections	Replica Lag (ms)	✓	mysqlDataSourceLocal	database-1-instance-1	2025-04-23 17:47:18 (+0200)	0	0	-	✓	mysqlDataSourceLocal	database-1-instance-3	2025-04-23 17:47:18 (+0200)	0	0	-	✓	mysqlDataSourceLocal	database-1-reader-test	2025-04-23 17:47:18 (+0200)	0	0	-	
Status	Data Source	Server Name	Last Heartbeat Time	Queries/Sec.	Connections	Replica Lag (ms)																									
✓	mysqlDataSourceLocal	database-1-instance-1	2025-04-23 17:47:18 (+0200)	0	0	-																									
✓	mysqlDataSourceLocal	database-1-instance-3	2025-04-23 17:47:18 (+0200)	0	0	-																									
✓	mysqlDataSourceLocal	database-1-reader-test	2025-04-23 17:47:18 (+0200)	0	0	-																									
Query Distribution:	Writer: 0% Readers: 0%																														
Connection Reuse: 1.00x																															
	localhost 25.04.18.0-20250418214034/456338	CPU 0.125 %	<table border="1"> <thead> <tr> <th>Status</th> <th>Client IP/Hosts (driver version/id)</th> <th>CPU</th> <th>Queries/Sec.</th> <th>Last Connect Time</th> <th>Connection Count</th> <th>Uptime</th> <th>Licensed</th> </tr> </thead> <tbody> <tr> <td>✓</td> <td>localhost 25.04.18.0-20250418214034/456338</td> <td>0.125 %</td> <td>0</td> <td>2025-04-23 17:47:19 (+0200)</td> <td>0</td> <td>00:16:30</td> <td>☐</td> </tr> </tbody> </table>	Status	Client IP/Hosts (driver version/id)	CPU	Queries/Sec.	Last Connect Time	Connection Count	Uptime	Licensed	✓	localhost 25.04.18.0-20250418214034/456338	0.125 %	0	2025-04-23 17:47:19 (+0200)	0	00:16:30	☐												
Status	Client IP/Hosts (driver version/id)	CPU	Queries/Sec.	Last Connect Time	Connection Count	Uptime	Licensed																								
✓	localhost 25.04.18.0-20250418214034/456338	0.125 %	0	2025-04-23 17:47:19 (+0200)	0	00:16:30	☐																								

- Logging out of Heimdall **does not** automatically log the user out of AWS IAM Identity Center.

- Logging out of Heimdall **does not** automatically log the user out of AWS IAM Identity Center.

- All SAML-authenticated users *must* be explicitly assigned to the application within AWS Identity Center under the **Assigned users and groups** tab.
 - Mixing HTTP and HTTPS flow is not allowed! AWS configurations using HTTPS require a consistent HTTPS authentication flow. Users accessing the Heimdall login page must do so via HTTPS (e.g. `https:// <heimdall_host> :8443/#/login`). Inconsistent use of HTTP within an HTTPS configured environment may lead to authentication failures and prevent successful login.
 - If you change group names in AWS IAM Identity Center, be sure to click the Commit button in the SAML configuration. This clears the group mapping cache. If you skip this step, the old group information may persist for up to an hour.
 - If the user cannot log in with SAML and you are using a group extraction mechanism, ensure that each of the users has at least one group assigned; otherwise the auth will fail on the AWS side.
 - If you encounter issues during login, try using your browser's **Incognito/Private Mode**. This helps eliminate the chance of lingering sessions or cached credentials interfering with the SAML login flow.
-

Setting Up SAML with Okta

HEIMDALL CENTRAL MANAGER AND PORTAL INTEGRATION WITH OKTA

This guide explains how to configure SAML authentication with **Okta** using **SAML 2.0** to enable Single Sign-On (SSO).

CREATE A NEW APPLICATION IN OKTA

1. Create or log in to your **Okta (developer) account**.
2. Navigate to **Applications → Create App Integration**.
3. Select **SAML 2.0** and click **Next**.
4. Name your application and click **Next**.

CONFIGURE SAML SETTINGS

Use the following configuration (*Use either HTTP or HTTPS consistently across the entire setup and future logins*) :

Single sign on URL:

- `http:// <heimdall_hostname> :8087/login/saml2/sso/heimdall`
- `https:// <heimdall_hostname> :8443/login/saml2/sso/heimdall`

Audience URI:

- http:// <heimdall_hostname> :8087/saml2/service-provider-metadata/heimdall
- https:// <heimdall_hostname> :8443/saml2/service-provider-metadata/heimdall
- Check [x] Use this for Recipient URL and Destination URL (default)

Create SAML Integration

1 General Settings

2 Configure SAML

A SAML Settings

General

Single sign-on URL 

https://localhost:8443/login/saml2/sso/heimdall

☒ Use this for Recipient URL and Destination URL

Audience URI (SP Entity ID) 

https://localhost:8443/saml2/service-provider-metadat

Default RelayState 

If no value is set, a blank RelayState is sent

Name ID format 

Unspecified

Application username 

Okta username

Update application username on

Create and update

Show Advanced Settings

To allow group extraction, add the following filter with the "Matches Regex" option, which will allow all groups retrieval with regex such as ".*"

Group Attribute Statements (optional)

Name	Name format (optional)	Filter
groups	Unspecified	Matches regex .*

Add Another

FINALIZE APP SETUP

Click **Next**, then select: - *I'm an Okta customer adding an internal app - This is an internal app that we have created*

Click **Finish**.

RETRIEVE METADATA URL

On the **Sign On** tab, copy the **Metadata URL**, e.g.: `https://dev-82243426.okta.com/app/<random-characters>/sso/saml/metadata`

General **Sign On** **Import** **Assignments**

Settings[Edit](#)

Sign on methods

The sign-on method determines how a user signs into and manages their credentials for an application. Some sign-on methods require additional configuration in the 3rd party application.
Application username is determined by the user profile mapping. [Configure profile mapping](#)

☒ **SAML 2.0**

Default Relay State

Metadata details

Metadata URL

`https://dev-82243426.okta.com/app/exkocgxkr3g31ICER5d7/sso/saml/metadata`

 [Copy](#)

 [More details](#)

ASSIGN USERS OR GROUPS

Go to the **Assignments** tab in your Okta app and assign access to users or groups.



Heimdall integration

Active ▼



View Logs

Monitor Imports

General

Sign On

Import

Assignments

Assign ▼

Convert assignments ▼

Q

Search...

People ▼

Filters	Person	Type		
People	 test Okta testOkta@example.com	Individual		
Groups	 Adam Qwerty test@heimdalldata.com	Individual		

CONFIGURE SAML IN HEIMDALL

1. Log in to **Heimdall Central Manager** with admin privileges.
2. Navigate to: Admin → SAML Configuration
3. Fill in the following:
4. **Auth provider** as Okta
5. **IdP Metadata URL**, e.g.:
`https://dev-82243426.okta.com/app/<random_string>/sso/saml/metadata`
6. Click **Commit**
7. **Restart Heimdall Central Manager**

Administration

License

Alerts

Login History

Log Management

Security Tags

SAML Configuration

Auth provider:

Okta

IdP Metadata URL:

https://dev-82243426.okta.com/app/exkocgxkr3g31ICER5d7/ssr

Commit

ENABLE SAML AUTHENTICATION FOR HEIMDALL USERS

1. Go to the **Users** tab in Heimdall.
2. Click **Create New User**
3. For Okta users, **enter the user's email address** instead of a username (as Heimdall will authenticate based on the data returned by Okta —i.e., email).
4. Check from the box: **SAML**
5. Click **Save**

Users

Create New User

test@heimdalldata.com

☒ Enable

Commit

Configuration

Username: test@heimdalldata.com

Email: email@heimdall.com

Management Privilege: Admin

Audit User: ☒ Note: User must re-login to view audit trails

Portal User: ☒

Authentication mode: SAML

Groups

Add:

+

Remove:

+

In the **Users** tab, administrators can manually assign groups to a user for portal use cases.

LOGGING IN VIA SAML (OKTA)

Once configured:

- Users can click **Login with SAML** on the Heimdall login page (!use proper protocol of HTTP/HTTPS as configured on Okta).
- They will be redirected to Okta for authentication.

okta

Sign In

Username

test@heimdalldata.com

Password

••••••••••••••••

☐ Keep me signed in

Sign in

[Forgot password?](#)

[Help](#)

- Upon successful login, they will be returned to Heimdall as authenticated users. User can hover on the icon near the username to see extracted groups.

Heimdall Data Console

Not secure https://localhost:8443/#/status

Documentation PDF Support test@heimdalldata.com

HEIMDALLDATA

Monitor

- Status
- Dashboard
- Analytics
- Logs

Configure

- Wizard
- Virtual DB
- Data Sources
- Rules
- Drivers

Status

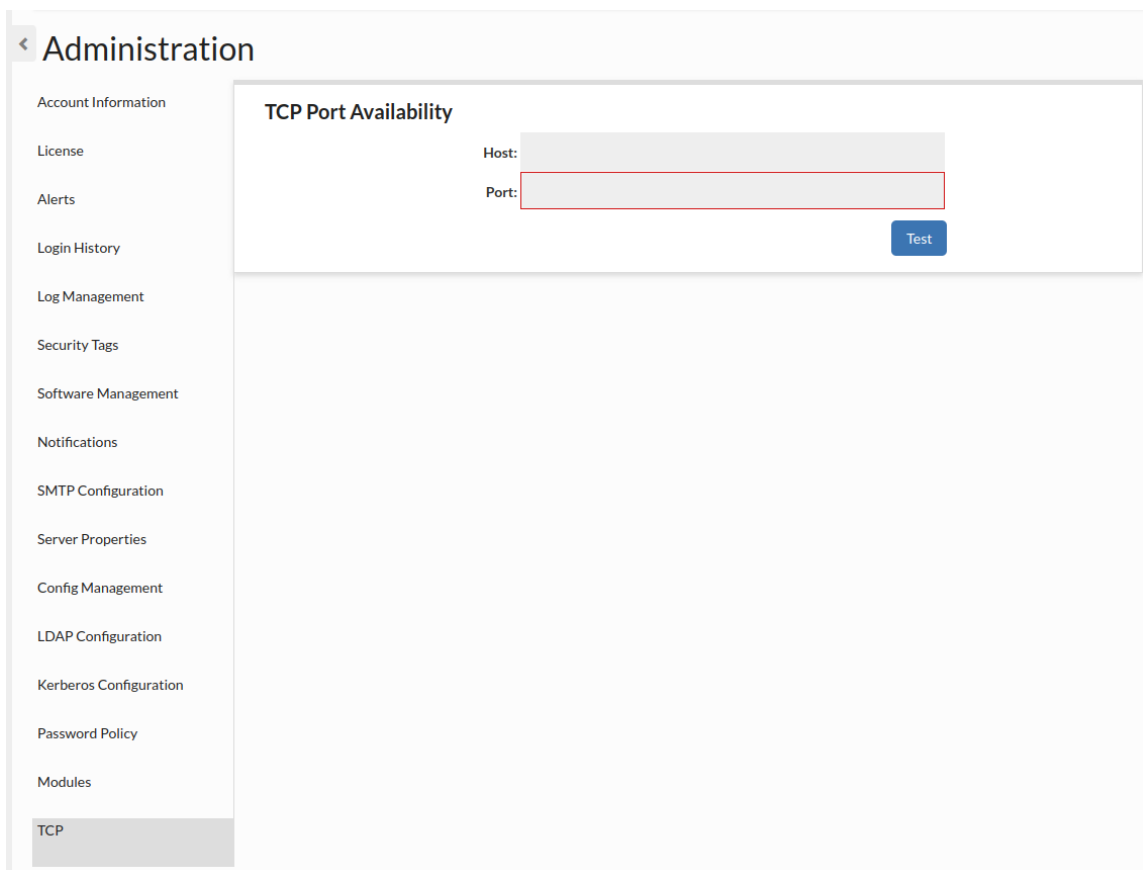
Status	Virtual Database	Servers	Development																																			
		<table><tr><th>Status</th><th>Data Source</th><th>Server Name</th><th>Last Heartbeat Time</th><th>Queries/Sec.</th><th>Connections</th><th>Replica Lag (ms)</th></tr><tr><td></td><td>MySQLVDB</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>:3307</td><td>mysqlDBsourceLocal database-1-instance-1</td><td>2025-04-28 18:34:15 (+0200)</td><td>0</td><td>0</td><td>-</td></tr><tr><td></td><td>Proxy starts: 2</td><td>mysqlDBsourceLocal database-1-instance-3</td><td>2025-04-28 18:34:15 (+0200)</td><td>0</td><td>0</td><td>-</td></tr><tr><td></td><td></td><td>mysqlDBsourceLocal database-1-reader-test</td><td>2025-04-28 18:34:15 (+0200)</td><td>0</td><td>0</td><td>-</td></tr></table>	Status	Data Source	Server Name	Last Heartbeat Time	Queries/Sec.	Connections	Replica Lag (ms)		MySQLVDB							:3307	mysqlDBsourceLocal database-1-instance-1	2025-04-28 18:34:15 (+0200)	0	0	-		Proxy starts: 2	mysqlDBsourceLocal database-1-instance-3	2025-04-28 18:34:15 (+0200)	0	0	-			mysqlDBsourceLocal database-1-reader-test	2025-04-28 18:34:15 (+0200)	0	0	-	
Status	Data Source	Server Name	Last Heartbeat Time	Queries/Sec.	Connections	Replica Lag (ms)																																
	MySQLVDB																																					
	:3307	mysqlDBsourceLocal database-1-instance-1	2025-04-28 18:34:15 (+0200)	0	0	-																																
	Proxy starts: 2	mysqlDBsourceLocal database-1-instance-3	2025-04-28 18:34:15 (+0200)	0	0	-																																
		mysqlDBsourceLocal database-1-reader-test	2025-04-28 18:34:15 (+0200)	0	0	-																																

Query Distribution:									
Writer:	0%	Status	Client IP/Hosts (driver version/id)	CPU	Queries/Sec.	Last Connect Time	Connection Count	Uptime	Licensed
Reader:	0%								
Connection Reuse:	1.00x		localhost 25.04.16.0-20250418214034/beau9b	0.125 %	0	2025-04-28 18:34:16 (+0200)	0	00:02:52	

⚠ IMPORTANT NOTES:

- Logging out of Heimdall **does not** automatically log the user out of Okta.
- All SAML-authenticated users *must* be explicitly assigned to the application within Okta under the **Assignments** tab.
- Mixing HTTP and HTTPS flow is not allowed! Okta configurations using HTTPS require a consistent HTTPS authentication flow. Users accessing the Heimdall login page must do so via HTTPS (e.g. `https:// <heimdall_host> :8443/#/login`). Inconsistent use of HTTP within an HTTPS configured environment may lead to authentication failures and prevent successful login.
- If you encounter issues during login, try using your browser's **Incognito/Private Mode**. This helps eliminate the chance of lingering sessions or cached credentials interfering with the SAML login flow.

TCP Port Availability



The screenshot displays the Heimdall Administration interface. On the left is a sidebar menu with the following items: Account Information, License, Alerts, Login History, Log Management, Security Tags, Software Management, Notifications, SMTP Configuration, Server Properties, Config Management, LDAP Configuration, Kerberos Configuration, Password Policy, Modules, and TCP (which is currently selected and highlighted). The main content area is titled 'TCP Port Availability' and contains two input fields: 'Host:' and 'Port:'. The 'Host:' field is a light gray box, and the 'Port:' field is a light gray box with a red border. A blue 'Test' button is positioned to the right of the 'Port:' field.

This allows checking if given port is available. Host is optional, if it isn't provided, it is set to "localhost".

Certificates Overview

The certificates tab is divided into four sections:

- Let's Encrypt Cert Wizard
- Certificates List
- Upload Certificate File
- Server Management Certificate

These sections are to help you manage certificates in Heimdall.

Let's Encrypt Cert Wizard

Let's Encrypt Cert Wizard section provides a way to simply generate Let's encrypt requests and then the final certificate from this.

Certificates

Let's Encrypt Cert Wizard

Certificates List

CA Certificates List

Upload Certificate File

Management Server Certificate

Challenge type

DNS ▾

Alias

global_use_certificate

Domain

Domain

☐ Use Let's Encrypt test API

Order

Verify

WARNING: successful verifying will override existing certificate

URL
Alias
Domain
Challenge type
Message
Status

Prerequisites

In the certificate tab in Let's Encrypt Cert Wizard section, there are three fields that need to be set to create an order of the Let's Encrypt certificate:

- Account Keys
- Challenge type (DNS is default)
- Certificate alias
- Domain

Account Keys

To authenticate certificate orders, Heimdall automatically generates and stores two private account keys in the main application directory.

Environment	Filename	Description
Production	lets-encrypt-user.key	Used for the standard Let's Encrypt API.
Test	lets-encrypt-test-user.key	Used for the Let's Encrypt Staging API (safe for testing).

Using Your Own Keys

If you already have a Let's Encrypt account key you wish to use (to preserve account history or limits), you may override the default generated keys.

To do this, simply **replace** the file in the main directory with your own private key, ensuring you use the **exact filename** listed in the table above.

Challenge

Challenges are used to prove ownership of a domain. There are two available challenges DNS and HTTP:

- DNS: You prove to the CA that you are able to control the DNS records of the domain to be authorized, by creating a TXT record with a signed content.
- HTTP: You prove to the CA that you are able to control the website content of the domain to be authorized, by making a file with a signed content available at a given path.

Certificate alias

A certificate alias is a name, used in other places, which is going to be assigned to the generated certificate in the Keystore and Virtual Databases.

Domain

A domain is a subject of the generated certificate.

Flow

In case to generate new Let's Encrypt certificate you must:

1. Set up all fields and click "Order" button. Let's Encrypt API will request order certificate.
2. Make a challenge returned by Let's Encrypt API. You will find instructions in message field.
3. Click "Verify" button. If you did your challenge properly, Let's Encrypt API will generate new Let's Encrypt certificate and will place it in the Keystore and Virtual Databases with given certificate alias.

Be careful !!!

If a certificate alias already exists in Heimdall, the existing certificate will be overridden with a new Let's Encrypt Certificate.

Create order

After you set up all fields and click the "Order" button:

1. Let's Encrypt API creates your unique account in Let's Encrypt CA if you don't have one.
2. Send to Let's Encrypt CA request to create certificate order for the given domain name.
3. Let's Encrypt API returns you a message on how you can prove your ownership, based on the selected challenge type.

Verify order

After you do your challenge and click "Verify" button:

1. Let's Encrypt API send request to Let's Encrypt CA in case to verify your ownership.
2. If verification went properly, API will request for certificate.
3. Generated certificated is being uploaded to Keystore with given alias. If certificate with given alias already exists it will override existing certificate.

Extra information

You can test API by selecting the checkbox "Use Let's Encrypt test API"

Certificates List

There is a list of uploaded certificates shown by the aliases.

Certificates

Let's Encrypt Cert Wizard

Certificates List

CA Certificates List

Upload Certificate File

Management Server
Certificate

Delete Unused Certificates

global_use_certificate

Issuer

Subject

Public Key Info

Miscellaneous

Validity

Fingerprints

tomcat

You can check the details of each certificate, such as:

- **Issuer** - The entity or organization that grants or issues the certificate. They are responsible for verifying the qualifications, credentials, or completion of specific requirements that make the recipient eligible for the certificate.
- **Subject** - The individual or entity to whom the certificate is issued. They are the recipient or beneficiary of the certificate.
- **Public Key Info** - The public key information such as: *Algorithm* - The specific cryptographic method or algorithm used to generate the key pair associated with the certificate, such as RSA, DSA, or ECDSA. *Exponent* - A component of the public key that is used in encryption and digital signature verification. It is a positive integer associated with the algorithm used. *Key Size* - The length or size of the cryptographic key, typically measured in bits. A larger key size generally implies increased security against certain types of attacks. *Modulus* - A mathematical term representing the modulus value used in the cryptographic algorithm. It is part of the public key and plays a crucial role in encryption and signature verification.
- **Miscellaneous** - Additional information such as serial number, signature algorithm, version.
- **Validity** - The period of time, where the certificate is valid.
- **Fingerprints** - A unique identifier generated from the certificate's contents using a specific hash algorithm.

You can remove all unused certificates (which are not used by any vdb) by clicking the "**Delete Unused Certificates**" button.

CA Certificates List

There is a list of uploaded CA certificates shown by the aliases in the same way as above certificates.

Certificates

- Let's Encrypt Cert Wizard
- Certificates List
- CA Certificates List**
- Upload Certificate File
- Management Server Certificate

cacert2	^
Issuer	▼
Subject	▼
Public Key Info	▼
Miscellaneous	▼
Validity	▼
Fingerprints	▼
<button>Delete</button>	
ca certificate 7	▼
cacert1	▼

Under the details for each CA certificate there is a button to delete a given certificate. Remember! Before deleting the CA certificate, make sure that all proxies are enabled so that the certificate will also be removed from the proxy cacerts file if it is on a different machine than the central manager.

Upload Certificate File

Upload Certificate File section provides a way to upload your own certificate into Heimdall.

Certificates

Let's Encrypt Cert Wizard

Certificates List

CA Certificates List

Upload Certificate File

Management Server Certificate

CA Certificate: ☐

Upload PKCS8 or PKCS1 PEM file (certificate+key) Browse... No files selected.

☒ Force update ☐ Private key is encrypted

Alias

global_use_certificate

Upload

There are 7 fields in upload certificate file interface:

- CA Certificate - Indicates whether the certificate is CA (Certificate Authority). CA can issue and sign other certificates.
- Combined pem file - A file that contains private key and certificate in pem format with appropriate headers. PKCS8 and PKCS1 formats are supported.
- Force update checkbox - If certificate with given alias already exists, force flag must be checked to override existing certificate in Keystore and Virtual Databases.
- Alias - Alias in keystore where certificate will be kept. By default, it's "global_use_certificate" alias.
- Private key is encrypted - Indicates whether a private key contained in uploaded .pem file is encrypted.
- Private key encryption password - Password used for encryption/decryption of private key.
Note: It can be stored in the [Secrets Manager](#).

If You want to upload a file containing an encrypted private key, then "Private key is encrypted" should be checked and password should be supplied by typing it in "Private key encryption password" field.

In configuration files the key will be stored in PKCS8 encrypted format and decrypted on the fly when it needs to be used.

Supported encryption algorithms are:

- DES-EDE3-CBC
- AES-128-CBC
- AES-192-CBC
- AES-256-CBC

Certificates

Let's Encrypt Cert Wizard

Certificates List

CA Certificates List

Upload Certificate File

Management Server Certificate

Certificate Provider

CA Certificate: ☐

Upload PKCS8 or PKCS1 PEM file
(certificate+key)

Choose Files

No file chosen

☒ Force update

☒ Private key is encrypted

Alias

global_use_certificate

Private key encryption password

....



Secret



Upload

You can also upload a CA certificate, which will be saved on the central manager side and transferred to all proxies. In both cases it will be saved in the cacerts file with the default password and in the default location appropriate for Java where the central manager/proxy is running.

If you want to upload a CA certificate, just select "CA Certificate" checkbox. The pem file should contain only the certificate - public key.

The ca certs aliases are also stored in the ca_certs_aliases.json file in the config directory.

Certificates

Let's Encrypt Cert Wizard

Certificates List

CA Certificates List

Upload Certificate File

Management Server Certificate

CA Certificate: ☒

Upload PKCS8 or PKCS1 PEM file
(certificate)

Browse...

No files selected.

☒ Force update ☐ Private key is encrypted

Alias

CA cert

Upload

Management Server Certificate

Server management Certificate section provides a way to see your current server management certificate, and also to change your server management certificate with it.

Certificates

- Let's Encrypt Cert Wizard
- Certificates List
- CA Certificates List
- Upload Certificate File
- Management Server Certificate**

Currently used certificate:
tomcat

▼ **Commit**

Flow:

1. Choose certificate that you want to use for server management.
2. Click "Commit" button to assign new certificate to the server management.

To start using the new certificate, you must restart the Heimdall.

Certificate Provider

Support has been added for automatic certificate retrieval from a certificate provider. Currently, only AWS Certificate Manager (ACM) is supported.

When a valid certificate identifier is provided, the system will attempt to fetch the corresponding certificate directly from the certificate provider. This allows seamless integration without the need to manually upload or manage certificates.

Certificates

- Let's Encrypt Cert Wizard
- Certificates List
- CA Certificates List
- Upload Certificate File
- Management Server Certificate
- Certificate Provider**

Certificate Provider Create New

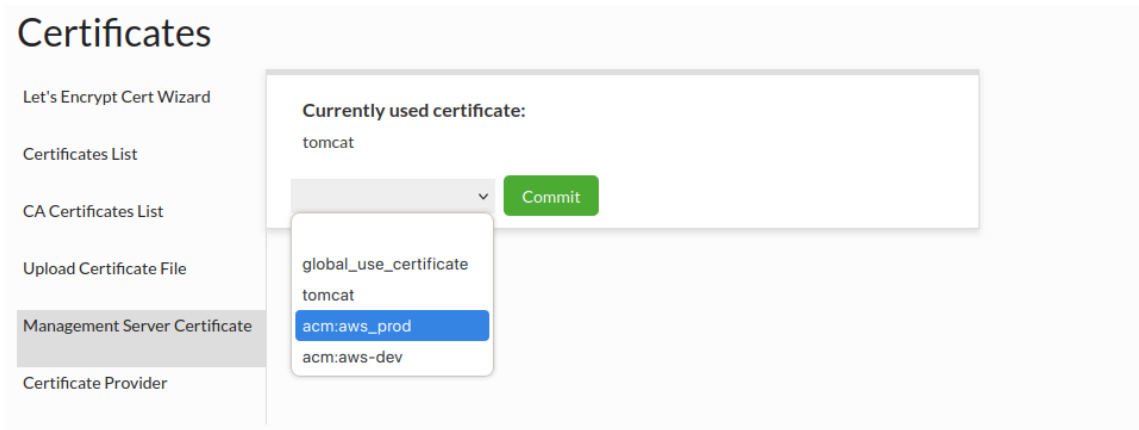
aws_prod ▼

Alias: aws_prod

Identifier: arn:aws:acm:us-east-1:xxxxxxxxxxxx:certificate/2f27a9ee-d8a2-

Commit

Every provided certificate will be displayed with prefix "acm:" e.g. for provided certificate aliased with aws_prod, we will see as acm:aws_prod. This allows us to easily distinguish between provided certificates and keystore certificates.



AWS Certificate Manager (ACM) Integration

Heimdall now supports fetching TLS certificates directly from AWS Certificate Manager (ACM). When a certificate ARN is provided, the system will automatically retrieve both the certificate and private key at application startup, assuming the required IAM permissions are in place.

Requirements:

1. You must use the ACM private CA to generate certificates--the AWS signed certs can only be used for Amazon services
2. Create the certificate as normal
3. Grant the Heimdall manager instance the `acm:ExportCertificate` IAM permission and Heimdall proxy instances the `acm:ExportCertificate` and `acm:DescribeCertificate`
4. Create new provided certificate configuration (alias and certificate arn) and commit changes.
5. Use provided certificate alias in proxy or in heimdall manager

Automatic Certificate Management

To ensure uninterrupted service and minimize manual maintenance, the Heimdall Proxy includes an automated background process that monitors the validity and configuration of your SSL/TLS certificates.

How it Works

A scheduled task runs automatically **once every 24 hours**. It compares the certificate currently loaded in the Proxy against the certificate source (e.g., your configured KeyStore, AWS ACM).

If a discrepancy is detected (based on the certificate Serial Number), the system performs a **hot-swap** of the certificate without dropping active connections or requiring a service restart.

Triggers and Behaviors

The scheduled task handles two specific scenarios automatically:

1. Certificate Renewal (Rotation)

- **Scenario:** You have updated the existing certificate file (same alias) with a renewed version (e.g., new expiration date, new keys).
- **Action:** The system reloads the certificate content immediately.

2. Configuration Change (Switching)

- **Scenario:** You have changed the configuration to point to a different certificate alias (e.g., switching from `test-cert` to `prod-cert`).
- **Action:** The system unloads the old alias and loads the new one.

Database Password Rotation Support (Proxy Authentication)

In order to simplify rotating of database passwords, please follow these steps:

1. Configure a mapping of the old password to new password via a HTTP call, with a browser, curl or other tool. A GUI login is required for the step:

```
http://demoa.heimdalldata.com:8087/api/tlkv/put?  
key=admin:heimdalltestasdf&value=heimdalltest&ttl=60000
```

Here, the old username and password is "admin:heimdalltestasdf" and the new password is "heimdall". The ttl is specified in milliseconds, so the above example expires this mapping in 60 seconds.

1. Change the password on the DB At this point, the password failure should be detected by the driver or proxy, and will be looked up via the configuration set with the URL, above
2. Rotate the passwords on the application/proxy side within the TTL specified. During this window, the old password will be replaced by the new one automatically. Once the TTL is done, the mapping is lost, and any old passwords will again be rejected.

The goal of this feature is to allow a fixed window where the old password can be used to connect to the DB which is configured to use the new password, without errors.

HTTP API reference

In this document you can find description of various HTTP API endpoints that the Heimdall central manager provides.

Reload configuration

- Endpoint URL: **api/config/reload**
- HTTP Method: **GET** (Preferred, It takes all types of methods)

This endpoint allows for reload of Heimdall's configuration (heimdall.conf file).

Usage Example

Usage is pretty straightforward, just remember to authenticate the request. Curl pattern:

```
curl --user {username}:{password} http://{ipAddress}:{portNumber}/api/config/reload
```

Where:

- {username} -> Heimdall's username
- {password} -> Heimdall's password
- {ipAddress} -> Heimdall's IP or DNS name
- {portNumber} -> Heimdall's server port number

Example command with default values on local Heimdall instance:

```
curl --user admin:heimdall http://localhost:8087/api/config/reload
```

Export VDB configuration

- Endpoint URL: **api/config/{config}/export/{vdb}**
- HTTP Method: **GET**

- Parameters:
 - **{config}**: String value that controls what is being returned by this endpoint in JSON:
 - "vdb" -> returns whole VDB configuration (current VDB, Data Sources, Drivers, Rules)
 - "source" -> returns Data Sources and Drivers related to VDB
 - "driver" -> returns Drivers related to VDB
 - "policy" -> returns Rules related to VDB
 - **{vdb}**: Name of VDB which details should be exposed
- Response Body: JSON file with chosen properties

This endpoint allows for retrieval of selected properties of single VDB.

Usage Example

To effectively use this endpoint, provide a name of VDB to export and what should be extracted as request parameters. Call needs to be authenticated. Curl pattern:

```
curl --user {username}:{password} http://{ipAddress}:{portNumber}/api/config/{config}/export/{vdb}
```

Where:

- {username} -> Heimdall's username
- {password} -> Heimdall's password
- {ipAddress} -> Heimdall's IP or DNS name
- {portNumber} -> Heimdall's server port number
- {config} -> string value determining what's exported (described above)
- {vdb} -> name of VDB to export

Example command with default values on local Heimdall instance:

```
curl --user admin:heimdall http://localhost:8087/api/config/vdb/export/MySQLVirtualDatabase
```

Import VDB configuration

- Endpoint URL: **api/config/import/{config}**
- HTTP Method: **POST**

- Parameters:
 - **{config}**: String value that controls what is being imported:
 - "vdb" -> updates all VDB properties
 - "source" -> updates Data Sources of VDB
 - "driver" -> updates Drivers of VDB
 - "policy" -> updates Rules of VDB
- Request Body: JSON file with properties of VDB

This endpoint allows for import of single VDB with all related properties (Data Sources, Drivers, Rules).

Usage Example

To use this endpoint, provide what should be imported as a {config} parameter. File with desired properties should also be provided. Call needs to be authorized too. Curl pattern:

```
curl --user {username}:{password} --form files=@{fileName} http://{ipAddress}:{portNumber}/api/config/import/{config}
```

Where:

- {username} -> Heimdall's username
- {password} -> Heimdall's password
- {fileName} -> file with VDB properties to import
- {ipAddress} -> Heimdall's IP or DNS name
- {portNumber} -> Heimdall's server port number
- {config} -> string value determining what's imported (described above)

FORMAT OF {FILENAME}

```
{
  "vdb": [
    {
      "accessType": "mysql", // is underlying dataBase: sqlserver, mysql,
      postgres
      "cacheEnabled": false,
      "pciHipaaCache": false, // PCI/HIPAA compliant cache
      "cacheSize": 50000000, // size of cache in bytes
      "cacheSoftValues": true, // Determine use of soft references
      "cacheMaxExpiry": 0, // Maximum cache object age
      "cacheObjectCount": 0, // Maximum cache object count
      "cacheType": "hazelcast", // type of cache
      "cloudMetrics": false, // determines if cloud specific metrics should be
      used
    }
  ]
}
```

```

    "clusterCacheManager": false, // determines use of ClusterCacheManager
    "customKey": false, // false -> use vdb, catalog, user if true use
custom cache key
    "debugMode": true, //debugMode for VDB
    "gridCacheName": "heimdall", // name of the cache to use
    "gridPort": 0, // grid cache port
    "databaseNumber": 0, // grid cache database number
    "id": 1606394877786, // config id
    "jdbcClass": "com.heimdalldata.HeimdallDriver", // class of JDBC driver
    "jdbcUrl": "jdbc:heimdall://\u003cHeimdall Server IP:port\u003e/
MySQLVirtualDatabase?
hduser\u003d\u003cuser\u003e\u0026hdpassword\u003d\u003cpassword\u003e",
    "excludeKeyCatalog": false, // True if exclude catalog part of the hash
key for query result cache
    "excludeKeySchema": false, // True if exclude schema part of the hash
key for query result cache
    "excludeKeySqlComments": false, // True if exclude sql comments part of
the hash key for query result cache
    "excludeKeyUser": false, // True if exclude connected database user part
of the hash key for query result cache
    "excludeKeyVdb": false, // True if exclude virtual database name part of
the hash key for query result cache
    "localCacheEnable": true,
    "localhostOnly": false,
    "logConnections": true, // determines if Heimdall should log connections
    "loggingEnabled": true,
    "logMethods": false, // determines if method calls should be logged
    "logResultSetMethods": false, // determines if ResultSet method calls
should be logged
    "logSql": true, // determines if SQL statements should be logged
    "logAuthentications": false, // determines if authentication should be
logged
    "mgmtProxyEnable": true, // determines if managment proxy should be
enabled
    "tlsEnable": false,
    "tlsRequired": false,
    "ldapAuthEnabled": false,
    "delayedTransaction": false, // determines if delayed transactions
should be enabled
    "paranoia": false, // determines paranoia mode, when activated it turns
off logging of SQL, passwords and parameters in the debug logs
    "multiplex": false, //determines if multiplexing should be enabled
    "multiplexTimeout": 0,
    "trackQueryDistributionCount": 1000, // sets max number of queries to
track
    "ldapGroupFilter": "", //ldap filter
    "ldapSimpleMode": false,
    "ldapPrefix": "",
    "ldapSuffix": "",
    "synchAuthentication": false, // determine if Authentication should be
synchronized
    "proxyauthEnabled": false, // determines if proxy authorization should
be enabled
    "authMode": "proxy", // type of authorization: proxy, ldap, sql,
passthrough
    "proxyAddress": "0.0.0.0",
    "proxyPort": "3307",

```

```

    "xmx": 600, // max heap value for proxy
    "revalidateCache": false, // determines if cache revalidation should be
enabled
    "secure": false, // is HTTPS on?
    "shipConsoleLogs": true, // determines if driver/proxy console logs
should be sent to the server
    "writeLogsToFile": false, // determines if logs should be written to
file
    "sqlDrivenAuthEnabled": false, // should SQL Authorization be used?
    "authorizationQuery": "select * from heimdall.pg_hba where enabled
\u003d true order by line_number asc", // query to use during SQL
Authorization
    "accessKey": "HIBa3RN0BpDovDkP", // configuration accessKey
    "secretKey": "1nCAUUqnxqblJF74", // configuration secretKey
    "useSsl": false, // determines if SSL should be used
    "verifyPeer": false, // determines if SSL Certificate should be verified
    "jmxHostname": "${hostname}",
    "jmxPort": 0,
    "testSSL": false,
    "userCaseSensitive": false, // determines if username is case sensitive
    "enabled": true, // determines if VDB should be enabled
    "file": "config/MySQLVirtualDatabase_1.conf", // path to VDB
configuration file
    "name": "MySQLVirtualDatabase", // name of VDB
    "version": 1, // version of VDB configuration
    "dataSources": [ // array of Data Sources associated with VDB
        {
            "enabled": true, // determines if Data Source should be enabled
            "file": "config/MySQLDataSource_4.conf", // path to Data Source
configuration file
            "name": "MySQLDataSource", // name of Data Source
            "version": 4 // version of Data Source configuration
        }
    ],
    "rules": [ // rules associated with VDB
        {
            "enabled": true, // determines if Rules should be enabled
            "file": "config/RulesForAll_1.conf", // path to Rules configuration
file
            "name": "RulesForAll", // name of Rules
            "version": 1 // version of Rules configuration
        }
    ],
    "properties": [ // properties associated with VDB
        {
            "key": "",
            "value": ""
        }
    ],
    "users": [ // underlying database credentials
        {
            "user": "root",
            "password": "password"
        }
    ],
    "pinnedClients": [], // array of pinned clients
    "modules": [ // array of Heimdall jars

```

```

        "modules/heimdallloadbalancer-1.0-20.11.26.1.jar",
        "modules/heimdallredis-5.3.0-20.11.26.1.jar",
        "modules/heimdallforward-1.0-20.11.26.1.jar",
        "modules/heimdallmysql-1.0-20.11.26.1.jar",
        "modules/heimdallsqlserver-1.0-20.11.26.1.jar",
        "modules/heimdallpostgres-1.0-20.11.26.1.jar",
        "modules/heimdalltrigger-1.0-20.11.26.1.jar",
        "modules/heimdallauth-1.0-20.11.26.1.jar",
        "modules/heimdallhazelcast-3.6.8-20.11.26.1.jar",
        "modules/heimdallextractplan-1.0-20.11.26.1.jar",
        "modules/heimdallfirewall-1.0-20.11.26.1.jar",
        "modules/heimdallasync-1.0-20.11.26.1.jar",
        "modules/heimdallaws-1.0-20.11.26.1.jar"
    ]
}
],
"sources": [ // Data Source configuration associated with VDB
{
    "url": "jdbc:mysql://localhost:3306/myDb?
allowPublicKeyRetrieval\u003dtrue\u0026useSSL\u003dfalse", // Data Source JDBC
url
    "driverConfig": {
        "enabled": true, // determines if Driver associated with Data Source
is enabled
        "file": "config/MySQLDriver_1.conf", // path to Driver's configuration
file
        "name": "MySQLDriver", // name of Driver
        "version": 1 // version of Driver configuration
    },
    "driverClass": "com.mysql.jdbc.Driver", // class of JDBC Driver
    "driverUrl": "/drivers/MySQLDriver/mysql-connector-java-5.1.49.jar", //
path to JDBC Driver
    "properties": {
        "user": "root", // database user
        "password": "password", // database password
        "url": "jdbc:mysql://localhost:3306/myDb?
allowPublicKeyRetrieval\u003dtrue\u0026useSSL\u003dfalse", // database url
        "testQuery": "SELECT 1"
    },
    "usepool": false, //determines if connection pooling should be used
    "poolProperties": {}, // key-value object of connection pooling
properties
    "servers": [ // array of Data Surce Servers configurations associated
with Data Source
        {
            "name": "Primary", // server name
            "url": "jdbc:mysql:// localhost:3306/myDb?
allowPublicKeyRetrieval=true&useSSL=false", // server JDBC url
            "enabled": true,
            "writeable": true, // true for master, false for slave
            "weight": 1, // relative amount of load server can handle
            "readWeight": 1, // read weight for server
            "writeWeight": 1, // write weight for server
            "active": true,
            "failed": true,
            "maxLag": 2147483647
        }
    ]
}
]
}

```

```

    ],
    "useLoadBalancing": true,
    "trackClusterChanges": false,
    "desiredWriteCapacity": 1, // desired write capacity for all writeable
nodes
    "desiredReadCapacity": 10, // desired read capacity
    "trackReplicationLags": true,
    "timeWindow": 10000, // safe window for replication lag in ms
    "useMonitoring": false, // determines if server monitoring should be
enabled
    "holdTime": 30000, // time to hold connections/operations waiting for
valid server to be available
    "obscure": {
        "secretKey": "0drv3jRI9sliFSZUFxwbynSf68Llbi" // key to obscure
password with
    },
    "enabled": true, // determines if Data Source should be enabled
    "file": "config/MySQLDataSource_4.conf", // path to Data Source
configuration file
    "name": "MySQLDataSource", // name of Data Source
    "version": 4 // version of Data Source configuration
    }
    ],
    "drivers": [ // array with Drivers associated with VDB
    {
        "jdbcClass": "com.mysql.jdbc.Driver", // JDBC Driver class
        "uploadedDrivers": [
            "/drivers/MySQLDriver/mysql-connector-java-5.1.49.jar" // path to
Driver
        ],
        "autoUpdate": true,
        "enabled": true, // determines if Driver should be enabled
        "file": "config/MySQLDriver_1.conf", // path to Driver configuration
file
        "name": "MySQLDriver", // name of Driver
        "version": 1 // version of Driver configuration
    }
    ],
    "rules": [ // array with Rules associated with VDB
    {
        "rules": [
            {
                "enabled": true, // determines if Rule should be enabled
                "type": "C", // type of rule, description below
                "patterns": [ // array with regex patterns for rule
                    ""
                ],
                "intrans": false, // determines if Rule can be applied within
transaction
                "properties": { // key-value parameters of Rule
                    "ttl": "5*60000"
                }
            },
            {
                "enabled": true, // determines if Rule should be enabled
                "type": "C", // type of rule, description below
                "patterns": [ // array with regex patterns for rule

```

```

        ""
    ],
    "intrans": true, // determines if Rule can be applied within
transaction
    "properties": { // key-value parameters of Rule
        "ttl": "5*60000"
    }
},
{
    "enabled": true, // determines if Rule should be enabled
    "type": "V", // type of rule, description below
    "patterns": [ // array with regex patterns for rule
        "(?i)^select"
    ],
    "intrans": false, // determines if Rule can be applied within
transaction
    "properties": { // key-value parameters of Rule
        "lagignore": "false"
    }
}
],
"enabled": true, // determines if Rules should be enabled
"file": "config/RulesForAll_1.conf", // path to Rules configuration file
"name": "RulesForAll", // name of Rules
"version": 1 // version of Rules configuration
}
]
}

```

RULE TYPES

```

Q -> allow
A -> async execute
C -> cache
G -> call
Z -> debug
D -> drop
O -> extract plan
F -> forward
I -> ignore
P -> learn pattern
L -> log
B -> nocache
M -> pool
V -> reader eligible
N -> result
E -> retry
S -> stack trace
W -> table cache
X -> tag
T -> transform
R -> trigger

```

Example command with default values on local Heimdall instance:

```
curl --user admin:heimdall --form files=@file.json http://localhost:8087/api/config/import/vdb
```

Export Data Source Servers configuration

- Endpoint URL: **api/config/source/{name}/servers**
- HTTP Method: **GET**
- Parameters:
 - **{name}**: Name of Data Source Servers to export
- Response Body: JSON file with list of Data Source Servers Properties.

This endpoint allows for retrieving Data Source Servers configuration by name.

Usage Example

Provide a name of existing VDB and credentials. Curl pattern:

```
curl --user {username}:{password} http://{ipAddress}:{portNumber}/api/config/source/{name}/servers
```

Where:

- {username} -> Heimdall's username
- {password} -> Heimdall's password
- {ipAddress} -> Heimdall's IP or DNS name
- {portNumber} -> Heimdall's server port number
- {name} -> name of existing Data Source

Example command with default values on local Heimdall instance:

```
curl --user admin:heimdall http://localhost:8087/api/config/source/MySQLDataSource/servers
```

Import Data Source Servers configuration

- Endpoint URL: **api/config/source/{name}/servers**
- HTTP Method: **POST**
- Parameters:
 - **{name}**: Name of Data Source to import Data Source Servers configuration to

- Request Body: JSON file with list of Data Source Servers Properties.

This endpoint allows for creating new Data Source Server configuration in existing Data Source.

Usage Example

Provide a name of existing VDB and new Data Source Server properties, to update it. This endpoint is secured, so we have to use authentication. Request must have Header of content-type set to application/json. Curl pattern:

```
curl --user {username}:{password} --data-binary @{filename} -H "Content-Type: application/json" http://{ipAddress}:{portNumber}/api/config/source/{name}/servers
```

Where:

- {username} -> Heimdall's username
- {password} -> Heimdall's password
- {filename} -> name of file with DataSourceServer properties
- {ipAddress} -> Heimdall's IP or DNS name
- {portNumber} -> Heimdall's server port number
- {name} -> name of existing DataSource

Format of {filename}:

```
[
  {
    "name": "Primary", // server name
    "url": "jdbc:mysql://localhost:3306/myDb?allowPublicKeyRetrieval=true&useSSL=false", // server JDBC url
    "enabled": true,
    "writeable": true, // true for master, false for slave
    "weight": 1, // relative amount of load server can handle
    "readWeight": 1, // read weight for server
    "writeWeight": 1, // write weight for server
    "active": true,
    "failed": true,
    "maxLag": 2147483647
  }
]
```

Example command with default values on local Heimdall instance:

```
curl -v --output - --user admin:heimdall --data-binary @file.json -H "Content-Type: application/json" http://localhost:8087/api/config/source/MySQLDataSource/servers
```

Heimdall and Open Source Licenses

Heimdall leverages a large number of open source projects and runs on Open source Linux.

The following base OS distributions are currently used, depending on the environment they are started from:

- AWS Marketplace: AWS Linux 2.0
- Azure Marketplace: Ubuntu 18.04 LTS
- Google Cloud Marketplace: Ubuntu 18.04 LTS
- Alibaba Cloud Marketplace: Alibaba Cloud Linux 2
- Docker: Defaults to Ubuntu 18.04, but this can be changed

Heimdall does not use any non-standard Linux packages besides those provided by the distribution, nor does it modify the Linux kernel or environment except through configuration files. All packages installed into the base OS leverage the distribution specific package manager, i.e. Apt for Debian/Ubuntu, or Yum for Centos/Redhat based environments.

Dependencies

In order to ensure all information is correct and updated At compile time, dependency lists are generated in three formats automatically (below). License URL's are as provided via the Maven POM for Java libraries, as submitted by the project maintainer, or in the case a POM is not available or complete, based on the public project website and maintained by Heimdall.

- [text license list broken out per dependency w/ unique POM license text included inline](#)
- [text license list sorted by license](#)
- [CSV license list with project link for source access, and license summary](#)

Open Source Code Access

To the extent any open source subcomponents are licensed under the EPL and/or other similar licenses that require the source code and/or modifications to source code to be made available (as would be noted above), you may obtain a copy of the source code corresponding to the binaries for such open source components and modifications thereto, if any, (the "Source Files"), by downloading the Source Files from the documented project location in the linked files above, or by sending a request, with your name and address to:

Heimdall Data, Inc
37070 Newark Blvd., Suite B
Newark, Ca. 94560

or email info@heimdalldata.com. All such requests should clearly specify:

OPEN SOURCE FILES REQUEST
Attention General Counsel

Heimdall shall mail a copy of the Source Files to you on a CD or equivalent physical medium. This offer to obtain a copy of the Source Files is valid for three years from the date you acquired this Software product.

License Text

For reference, the standard formatted version of all licenses referenced by included code is included below. In the linked text above, the exact license including dates (as appropriate) can be found.

Apache License, Version 2.0; Apache 2.0; The Apache License, 2.0

Apache License
Version 2.0, January 2004
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a

cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
 - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
 - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
 - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
 - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor,

except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Go License

Copyright (c) 2009 The Go Authors. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- * Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

HSQldb License, a BSD open source license

COPYRIGHTS AND LICENSES (based on BSD License)
For work developed by the HSQL Development Group:

Copyright (c) 2001-2020, The HSQL Development Group
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice,

this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the HSQL Development Group nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL HSQL DEVELOPMENT GROUP, HSQLDB.ORG, OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

For work originally developed by the Hypersonic SQL Group:

Copyright (c) 1995-2000 by the Hypersonic SQL Group.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the Hypersonic SQL Group nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE HYPERSONIC SQL GROUP, OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on

behalf of the
Hypersonic SQL Group.

The 3-Clause BSD Licenses

Copyright <YEAR> <COPYRIGHT HOLDER>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

BSD Nuclear clause

You acknowledge that this software is not designed or intended for use in the design, construction, operation or maintenance of any nuclear facility.

The MIT License

Copyright <YEAR> <COPYRIGHT HOLDER>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The Postgres License

PostgreSQL Database Management System
(formerly known as Postgres, then as Postgres95)

Portions Copyright © 1996-2020, The PostgreSQL Global Development Group

Portions Copyright © 1994, The Regents of the University of California

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

CC0

Licensed under Public Domain (CC0)

To the extent possible under law, the person who associated CC0 with this code has waived all copyright and related or neighboring rights to this code.

You should have received a copy of the CC0 legalcode along with this work. If not, see <<http://creativecommons.org/publicdomain/zero/1.0/>>.

Configuration Life-cycle Management

Backup and Restore of VDB configurations

The easiest way to manually backup and restore vdb configurations is to use the menu icon in the upper right of the VDB menu, and use export. This will save a single json file that aggregates the vdb, data source and rules configurations for a given VDB. The same menu can be used to import a VDB (refresh the screen after an import). If there is an existing vdb of the same name on import, the new entry will be named with _1, _2, etc. If the nested configurations (source and rules) exist with the same name on import, they will be reused as part of the imported configuration.

File Based Management

For those wishing to automate backup and restore, the files of the management server configuration can be accessed directly. The configuration files exist in the `/workDir/heimdall/config` directory (accessible only by root) by default. There are several techniques for managing these:

- Use EFS or another mounted filesystem to keep the config directory accessible in a shared location;
- Use chef or puppet to provide the configuration files;
- and; Use the `api/config/reload` api call to update the configuration in-memory when any change is made.

S3 Bucket Based Load

On the initial startup of a management server, the configuration can be loaded via an S3 bucket using the user data variable "configSource". This is documented in the AWS Cloudformation template.

Recovery

In the event that a proxy has failed in a standalone configuration, it should be sufficient to simply restart the proxy as per the initial deployment. As the proxy will pull the current driver version from the central manager, this will ensure that the proxy matches the central manager version, even if deployed from an older IAM.

In the event of a central manager failure, it typically can be simply restarted. If this fails, it too can be redeployed from the baseline IAM, using the S3 bucket based loadn or shared filesystem to initialize the instance with the proper configuration. If there are any questions on how to do this in your environment, please contact Heimdall support.

Getting Support

Please check our website at <https://www.heimdalldata.com/support/> for all information on contacting support, support tiers, SLA's etc.

Accelerating Support

In nearly any case, a summary of the issue being faced will be requested, as well as a copy of the logs. If the logs can be provided on the initial contact with support, this often dramatically accelerates the time to resolution.

When providing screenshots, please use the screenshot tool denoted by the camera icon on the header row (shown below). This tool will capture the current page, and bundle it with the logs to provide to support.



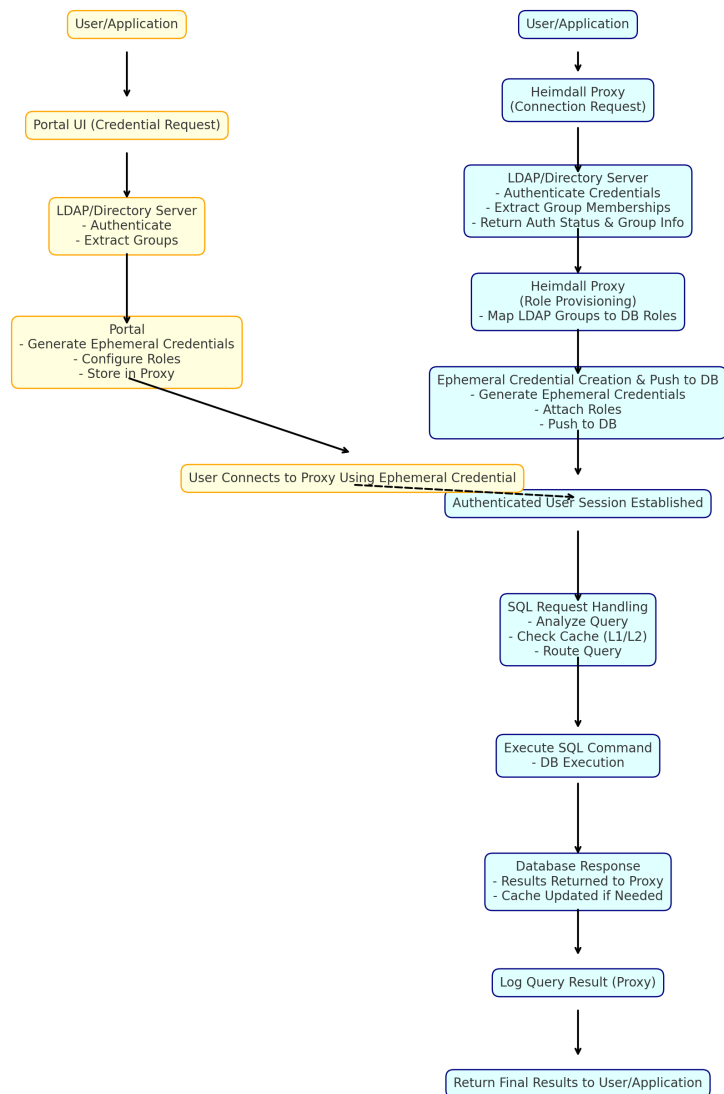
Portal Introduction

The Heimdall portal is a set of functionality that provides PSM (Privileged Session Management) and PAM (Privilege Access Management) functionality, sometimes joined as PASM. On the management side, Heimdall can be configured to query group membership from a user directory, such as Active Directory, FreeIPA, OpenLDAP, Jumpcloud or Octa. From there, the group membership can be mapped into a data source to indicate role entitlement. This entitlement can be either conditional based on one or more approvals, or unconditional.

Users are then able to login to the user portal, and request access to a particular role for a particular amount of time. For conditional roles, once the request is made, the approvers are notified of the request, and they are then able to approve the request or deny it.

Once approved, users are able to login during the requested time period using ephemeral credentials. When connected through the proxy layer in this setup, all queries can be logged for the user for audit trail purposes. Once the user is done with their task or the session expires, the ephemeral credentials are removed from the database to minimize the attack surface.

As an alternative session creation mechanism, the user could simply request a connection through the proxy, and the unconditional roles associated with the user will be mapped in and a session created, providing for seamless integration with the database and the directory, while avoiding extra steps for users.



As part of this functionality, Heimdall also provides a role and user access report generator in the data source tab, which allows the users and roles to be audited, and the data that each role grants to be tracked. This helps ensure that the roles that a user can or are granted only provide the functionality needed for that role, preventing over-provisioning. This report functionality includes the ability to map out nested role permissions under a user, allowing an administrator to identify why a user may have access to a resource that is unexpected. Other tools often only provide direct role permissions, or may be incomplete in permission mapping due to legacy permission schemes. An example of this is with Amazon Redshift, it has both group and roles, both of which can provide privileges at the same time.

For portal configuration, a variety of tasks need to be completed:

- Setup an LDAP configuration to pull group memberships from;

- Setup a notification system to allow notices to go out, typically SMTP;
- Setup request requirements, such as if requests need to include a justification , and if so, what the justification requirements are (such as a ticket ID format);
- A data source to store session information;
- Group to role mappings for each data source access is to be provided to;
- Role approver configuration, which is tied to notification configuration;
- User configuration for the portal (which can be learned at runtime from the directory based on group membership).

This guide will walk you through all the required steps up to a practical example. We will start with the basic setup and move on to managing roles and groups.

Prerequisites

We'll start by setting up the SMTP configuration, which is used for sending emails. The portal's notification system relies on SMTP, meaning that all future notifications—such as session state changes, approvals, and portal session requests—will be sent from the email address configured here. Fill in all the required fields, commit the changes, and verify the setup by sending a test email to a specified address. Click the Test button to confirm that everything is working correctly.

Administration

- Account Information
- License
- Alerts
- Login History
- Log Management
- Security Tags
- Software Management
- Notifications
- Portal
- SMTP Configuration**
- Server Properties

SMTP Configuration

Sender User/Email: example@heimdalldata.com
Sender Password:
Host: smtp.gmail.com
Port: 587
Smtp Auth: true
START_TLS Enabled: true

Secret ☐
☐

SMTP Properties

mail.smtp.ssl.protocols ▼ TLSv1, TLSv1.1, TLSv1.2, TLSv1.3

Email Tester

Recipient Email Address: example@heimdalldata.com
Test

Commit

(For more details on this section, see: [SMTP Configuration](#))

With the SMTP configuration in place, the next required step is setting up the LDAP configuration. A comprehensive guide and additional details can be found in: [LDAP Configuration](#), or you can use the: [LDAP Configuration Wizard](#). LDAP serves as the primary mechanism for determining a user's assigned groups within the portal. These groups define which roles a user is authorized to request sessions for, as explained later in this tutorial.

As an alternative (recommended for testing purposes only) you can manually assign groups to existing GUI users in the Users tab. More details can be found here: [Users](#).

Remember to enable the "Portal User" option when manually creating portal users in the Users tab and keep in mind that any changes to GUI user role permissions require the user to relog for the new authorities to be fully applied.

Before accessing the portal, also ensure that the appropriate Data Source(s) are created in the [Data Source](#) tab. If they don't exist, create them. However, if the Data Sources are already created, a key step is to create procedures responsible for creating portal session users with the correct roles and base portal user (The base portal user is a non-login user that helps maintain privilege consistency and takes ownership of objects created by ephemeral portal users before their removal). You can do this on the specified data source as shown below by clicking the Execute Scripts button.

Data Sources

Create New Source

MySQL-source

Enable

Commit

Configuration

Driver: MySQLDriver

JDBC URL: jdbc:mysql://localhost:3306

Username: root

Password:

Test query: SELECT 1

Test Connection

Execute scripts

AVAILABLE SCRIPTS:

☒ create-base-portal-user-procedure.sql

☐ create-sql-auth-table.sql

☐ create-lookup-table.sql

☒ create-sync-user-procedure.sql

☐ create-token-table.sql

+ Connection Properties

+ Connection Pooling (En

+ Load Balancing/High A

+ Group Mapping (1)

+ Roles Management (4)

+ Database Browser

For a Redshift data source, it's also necessary to include the script responsible for creating the view used to generate queries for revoking object privileges — in case an ephemeral portal user has created or inherited any during previous sessions.

AVAILABLE SCRIPTS:

☒ create-generate-user-revoke-privileges-ddl-view.sql

☒ create-base-portal-user-procedure.sql

☐ create-sql-auth-table.sql

☐ create-lookup-table.sql

☒ create-sync-user-procedure.sql

☐ create-token-table.sql

Portal Configuration

Having completed all the above steps, we can now proceed with setting up the portal. In the Admin tab -> Portal section -> Portal Basic subsection, you'll need to specify key settings, including:

The **LDAP Configuration** to be used by the portal.

The **Data Source**, where the portal database will be stored.

The **Host for the Central Manager** (optional), which runs the portal and will be included in notification emails, allowing users to quickly access the specified link. This is especially useful when having multiple Central Managers.

The **Synchronize Portal Users** option, which, when enabled, allows LDAP users to log in to the Central Manager and automatically creates a corresponding GUI user on the fly, enabling them to request sessions for the appropriate roles.

Administration

Account Information

License

Alerts

Login History

Log Management

Security Tags

Software Management

Notifications

Portal

SMTP Configuration

Server Properties

Config Management

Secrets Manager Configuration

LDAP Configuration

Kerberos Configuration

SAML Configuration

Password Policy

Modules

TCP

Portal Basic

LDAP Configuration:LDAP_CONFIG

Portal data source:MySQL-source

Portal Host:https://example.org:8443

JWT Signature Key:Secret

Send audit logs to CloudWatch:☒

CloudWatch namespace:heimdall-audit

Send audit logs to CloudTrail:☒

Send audit queries logs to CloudTrail:☒

Cloud Trail Lake Channel ARN:arn:aws:cloudtrail:us-east-1:272911111111:channel/16e1b5bb-b564-463

Show approvers emails:☒

Synchronize Portal Users:☒

Send Calendar Item:☒

Require specific justification format:☒

Regex pattern:Match all if empty

Invalid justification message:Does not match the global regex pattern.

Notifications

(For more details on this section, see: [Portal](#))

In the next subsection, Notifications, you need to create configurations that define the notification protocol (SMTP, LDAP, SNS or SIMPLE), a Notification Alias, and the target recipient or group(s) of recipients.

Administration

Account Information

License

Alerts

Login History

Log Management

Security Tags

Software Management

Notifications

Portal

SMTP Configuration

Server Properties

Config Management

Secrets Manager Configuration

LDAP Configuration

Kerberos Configuration

Password Policy

Portal Basic

Notifications

Commit

#	Protocol:	Notification Alias	Configuration	Send Test	Delete
1.	SMTP	none	Heimdall Username, for SMTP lookup	Test	
2.	SMTP	admin	admin	Test	
3.	LDAP	fullTimeGroup	full-time	Test	
4.	SMTP	multi	mark test	Test	

Add Notification Entry

Data Sources

Users

For SMTP notifications, add GUI users (ensure they have a defined email in the [Users](#) tab). For LDAP, specify the groups. For SNS, provide the Topic ARN. For SQL, provide the query to fetch emails from the database for the given data source. For SIMPLE, provide ready-made email addresses. Notifications will be required in the next step to define the list of approvers for requested roles. In other words, within the portal, a single notification entry defines which user or group of users can approve or deny a given role. It also determines who will receive notifications when a session is requested for that role.

The next subsection, Data Source, is fundamental to the portal's operation, specifically its two key parts: Group Mapping and Roles Management.

- *Group Mapping* (optional) allows mapping user groups (retrieved from LDAP or manually assigned in the [Users](#) tab) to specific roles in the database. Users with mapped groups can request sessions for the corresponding roles.

Portal Basic

+

Notifications

+

Data Sources

-

Data Source: MySQL-source

▼

Commit

Group Mapping

+

-

LDAP Configuration: LDAP_CONFIG_1

▼

Group Membership	Local Database Role	Add Role	Delete
test-users ▼ ☐	test-role ▼ ☐	+	✖

Roles Management (2)

+

Users

+

Example: The mapping above defines that any user logging in with the "test-users" group assigned will be able to request sessions for the "test-role", provided a configuration for that role is created below. In the absence of a mapping, a user in the "test-users" group could request sessions for a role with the same name, assuming that a Role Configuration in Roles Management for the "test-users" role has been created.

- **Roles Management** consists of two main parts:
 - **Database-Native Session Cleanup** - This option helps to clear sessions, when Central Manager goes offline. More information [here](#)
 - **Require justification pattern:** - This part enforces a pattern that all session request justifications must match for the given data source.
 - **Filter system roles:** - If selected, predefined system roles for databases will not be displayed on the roles names list.
- Below part is used to configure roles available for session requests within the portal.
 - You have to select a role.
 - Assign a notification alias, defining the user or group responsible for approving/denying session requests for that role.
 - Set the Default and Maximum session duration.
 - Set Approvals Needed field, which defines how many approvers must approve a given role before a session can be granted.

Administration

Account Information

License

Alerts

Login History

Log Management

Security Tags

Software Management

Notifications

Portal

SMTP Configuration

Server Properties

Config Management

Secrets Manager Configuration

LDAP Configuration

Kerberos Configuration

SAML Configuration

Password Policy

Modules

TCP

Portal Basic

Notifications

Data Sources

Group Mapping (1)

Roles Management

Users

Data Source: MySQL-source

Commit

Database-Native Session Cleanup: ☒

Require justification pattern: ☐

Filter system roles: ☒

Role Name	Notification Alias	Default Time	Maximum Time	Approvals Needed	
jenkins	admin	1 h	2 h	1	
admin	none	2 h	4 h	0	

Example: The screenshot above shows two configured roles:

- *jenkins* – The approver for this role is assigned to the "admin" notification alias, meaning the designated users will receive notifications and be responsible for approving or denying session requests. The default session duration is 1 hour, and the maximum requestable duration is 2 hours.
- *role-test* – Assigned the "none" notification alias, indicating a default empty notification with no specified users or groups. As a result, sessions for this role are auto-approved upon request. The default session duration is 2 hours, and the maximum requestable duration is 4 hours.

Portal Flow

With the configuration complete, we will now walk through a sample portal flow, demonstrating the process from requesting a session to the creation of a portal session user. In this example, the requester will be "mark". Since this user is not an LDAP user, we manually assign him to the "test-users", "jenkins" and "random-group" groups. If you need to create a new user, use the [Users](#) tab.

Portal Basic

Notifications

Data Sources

Users

User: mark Portal User: ☒ Audit User: ☐

Commit

Groups

Add: Remove:

test-users

jenkins

random-group

×

×

×

Next, user mark logs in, navigates to the [Request Access](#) tab, selects the data source for which he wants to request a session, specifies the role(s) he needs, sets the requested session duration, and provides a justification.

Note: In the list of available roles for request, a user will only see roles that have been configured in Roles Management and align with their assigned groups. This includes groups assigned directly to the user and those mapped via Group Mapping, if applied. Even though we previously assigned the user "mark" to the "random-group" group, he cannot request a session for the role with that name—even if it exists in the database—because it is not configured in Roles Management and is not mapped to another role in Group Mapping.

Request Access

Data Source: MySQL-source

Roles: ☐ ALL

☒ jenkins

☒ test-users → test-role

Preserve Objects: ☒

Inherit Objects: ☐

Justification: These roles allow me to perform operations like ...

Start Timestamp: 08 / 19 / 2025 , 03 : 30 PM

Duration: 2 h

Request Session

In this example, mark requests a session for the jenkins and test-role roles with a duration of 2 hours. Additionally, we can see that the test-users group is mapped to the test-role, and that role is automatically selected, meaning it doesn't require any approvals and will be granted automatically. Selecting the Preserve Objects option ensures that the objects created by the ephemeral user during the session will not be deleted once the session ends.

After successfully submitting the request, he will see a success alert on the screen, and the approvers assigned to this role will receive notifications with the session details.

Portal host: <http://localhost:8087>
The user "mark" requested a session to the data source "MySQL-source" with roles "jenkins, test-users", justification as a "These roles allow me to perform operations like ...", start timestamp at "2025-08-19 15:30:00.0 UTC+0200" and duration as a "2h 0m".
Please allow or deny this request in the Pending Approvals tab : <http://localhost:8087/#/pending-approvals>

Next, the approver(s) log into Heimdall and navigate to the Pending Approvals tab, where they can approve or deny session requests. In this example, the admin user logs in, since the jenkins role is assigned the "admin" notification alias. In the Notifications Configuration, the admin alias is connected to the admin user (look above). Moreover, we can see that the test-role is highlighted in green, which indicates that it has already been approved (in this case, automatically).

Based on the previous configuration, where we specified that the jenkins role requires only one approval, admin can approve the session requested by mark:

Pending Approvals

Data Source	User	Selected Roles	Justification	Flags	Start Timestamp UTC+2 ↓	End Timestamp UTC+2	Approvals Progress	Action
MySQL-source	mark	jenkins test-role	These roles allow me to perform operations like ...	Preserve Objects: ☒	Aug 19, 2025 3:30:00 PM	Aug 19, 2025 5:30:00 PM	0 / 1	Approve Deny

If an approver decides to deny a session, they will have the option to provide a denial reason explaining their decision. This is optional, and if provided, it will be included in the email notification sent to the requester.

Pending Approvals

Data Source	User	Selected Roles	Justification	Flags	Start Timestamp UTC+2 ↓	End Timestamp UTC+2	Approvals Progress	Action
MySQL-source	mark	jenkins test-role	These roles allow me to perform operations like ...	Preserve Objects: ☒	Aug 19, 2025 3:30:00 PM	Aug 19, 2025 5:30:00 PM	0 / 1	Approve Deny

Denial reason

Denial reason:

Close

Deny request

If admin approved the session, it means that the requested session has gathered all the required approvals (1 in this case) and is now marked as APPROVED. Additionally, the requester (mark) receives an email notification confirming that their session has been approved:

Portal host: <http://localhost:8087>
The session you requested to the data source "MySQL-source" with roles "jenkins, test-role", start timestamp at "2025-08-19 15:30:00.0 UTC+0200", end timestamp at "2025-08-19 17:30:00.0 UTC+0200", and justification as a "These roles allow me to perform operations like ..." has been approved.

If this happens during the requested session timeframe, a portal session user will be generated with the approved roles.

The requester (mark) can then navigate to the Sessions tab to view his session details, check its status, and access the credentials needed for login:

Sessions

Data Source	Selected Roles	Status	Flags	Start Timestamp UTC+2 ↓	End Timestamp UTC+2	JDBC URL	Username	Password	Approvals Progress	Cancel
MySQL-source	jenkins test-role	STARTED	Preserve Objects: ☐	Aug 19, 2025 3:30:00 PM	Aug 19, 2025 5:30:00 PM	jdbcmysql://localhost:...	mark_1755610402922	*****	1 / 1	

If the session has received all required approvals but its start time has not yet arrived, the portal session user (along with credentials) will only be generated when the session begins. At that point, the user will be able to log in. Once the session duration expires, the portal session user is automatically deleted, and any active database connections are terminated. The requester can also cancel their session at any time, which will have the same effect. Additionally, admin users have the ability to terminate sessions.

Audit User

To monitor the portal and all existing sessions, there is a role called "Audit User", which can be assigned to GUI users.

Portal Basic

Notifications

Data Sources

Users

User: admin Portal User: ☒ Audit User: ☒

Groups

Add:

role1

role2

jenkins

Remove:

Commit

Once assigned, the user gains access to the [Audit](#) tab, where they can browse all session and roles state changes, filter records as needed, and view details such as who performed an action and how it changes session/role state. There is also an option to download the selected records based on applied filters or to download all records.

Sessions AuditQueries Audit

(UTC+2) Since: mm/dd/yyyy, --:-- --

Until: mm/dd/yyyy, --:-- --

Filter by: Data SourcePostgres-source17

Filter by: Special Rolesfalse

Reset Filters

Get Records

2050100

Log Timestamp UTC+2 ↑	Username	State	DataSource	Roles	Flags	Session Start Time UTC+2	Session End Time UTC+2	View Queries
Aug 13, 2025, 4:04:58 PM	admin	DENIED	Postgres-source17	role2	Preserve Objects: ☒	Aug 13, 2025, 3:40:00 PM	Aug 13, 2025, 4:10:00 PM	Q
Aug 13, 2025, 4:04:58 PM	john	DENIED	Postgres-source17	role2	Preserve Objects: ☒			Q
Aug 13, 2025, 3:41:00 PM	admin	CREATED	Postgres-source17	role2	Preserve Objects: ☒	Aug 13, 2025, 3:40:00 PM	Aug 13, 2025, 4:10:00 PM	Q
Aug 13, 2025, 3:24:00 PM	admin	EXPIRED	Postgres-source17	role1		Aug 13, 2025, 10:58:44 AM	Aug 13, 2025, 3:24:00 PM	Q
Aug 13, 2025, 12:13:00 PM	mark	EXPIRED	Postgres-source17	role3	Preserve Objects: ☒		Aug 13, 2025, 12:13:00 PM	Q
Aug 13, 2025, 11:43:00 AM	admin	EXPIRED	Postgres-source17	role2	Inherit Objects: ☒ Preserve Objects: ☒		Aug 13, 2025, 11:43:00 AM	Q
Aug 13, 2025, 11:14:04 AM	admin	CREATED	Postgres-source17	role2	Inherit Objects: ☒ Preserve Objects: ☒	Aug 13, 2025, 11:13:00 AM	Aug 13, 2025, 11:43:00 AM	Q
Aug 13, 2025, 11:13:35 AM	mark	CREATED	Postgres-source17	role3	Preserve Objects: ☒	Aug 13, 2025, 11:13:00 AM	Aug 13, 2025, 12:13:00 PM	Q
Aug 13, 2025, 11:13:11 AM	admin	TERMINATED	Postgres-source17	role1		Aug 13, 2025, 10:54:00 AM	Aug 13, 2025, 11:13:11 AM	Q
Aug 13, 2025, 10:58:44 AM	admin	STARTED	Postgres-source17	role1		Aug 13, 2025, 10:58:44 AM	(est) Aug 13, 2025, 12:55:00 PM	Q
Aug 13, 2025, 10:58:44 AM	john	APPROVED	Postgres-source17	role1				Q
Aug 13, 2025, 10:55:58 AM	admin	CREATED	Postgres-source17	role1		Aug 13, 2025, 10:55:00 AM	Aug 13, 2025, 12:55:00 PM	Q

Request Access Overview

The portal functionality supports the following databases: PostgreSQL, MySQL, SQL Server, Oracle, Greenplum and Redshift.

The Request Access tab within the Portal provides the capability to request a session for specific roles in the designated Data Source.

Upon submitting a session request, the relevant approvers will be notified, possessing the authority to approve or deny access to the specified roles.

Only after receiving approval for all roles, the session will be approved, and a temporary user will be created. The requester will then be able to log in to the database using the specified temporary user credentials for a defined period.

There are several possible scenarios during role requests:

1. There may be a situation where a particular role cannot be selected at all. This means that the requesting user is also the only approver for this role and cannot approve themselves.
2. After selecting a role, there might be an alert indicating that the configuration is incomplete. This suggests that there may not be a configuration created in [Roles Management](#) for that role or its mapped roles. Additionally, the envelope icon will be grayed out with the information "No approvers found" upon hovering over it.
3. It is also possible that immediately after requesting a specific role in a session, it gets auto-approved. This means that a notification is attached to that role without being associated with any emails. In other words, no approvals are required for such a role. Such roles become automatically selected when requesting a session (like the 'vm-managers' role shown below).

The screenshot shows a web form for requesting a database session. At the top, there's a dropdown for 'Data Source' set to 'PostgreSQLDataSource'. Below it, the 'Roles' section has a checked 'ALL' checkbox and a list of roles: 'vm-managers', 'jenkins' (with a right arrow pointing to 'vm-admins' and 'cloud-admins'), and 'admins'. Each role has an 'X' icon to its right. The 'Preserve Objects' and 'Inherit Objects' checkboxes are both checked. The 'Justification' field contains the text 'test session request'. The 'Start Timestamp' field shows '02/03/2026, 06:14 PM' with a calendar icon. The 'Duration' field is set to '1' hour. A green 'Request Session' button is located at the bottom center of the form.

Fields

- **Data Source:** Data Source in which the user wishes to request access to roles.
- **Roles:** Roles that the user wishes to have in the database during the session. The list of roles consists of roles that match the user's LDAP group memberships and roles available in the database. Moreover, for a role to appear on this list, it must be configured in [Roles Management](#), taking into account Group Mappings if they exist for the role.
- **Preserve Objects:** If selected, all objects created in this session by the ephemeral user (or rather, those for which they will be the owner, since the creator does not always become the owner depending on the database) will be preserved and may be inherited in subsequent sessions. In both cases described below, objects created by ephemeral users do not remain "orphaned" in the database - they are either safely reassigned or fully cleaned up, depending on the setting.
 - **When enabled (default):** at the end of the ephemeral session, all objects owned by the ephemeral user are **reassigned** to the base user account: `'requester' + _base_portal_user`. Thanks to this, they remain accessible and can be inherited later.
 - **When disabled:** at the end of the ephemeral session, all objects still owned by the ephemeral user are **deleted**. No reassignment occurs and they cannot be recovered.
- **Inherit Objects:** If selected, this means the session user will receive privileges to objects created during their previous sessions. This applies to databases, schemas, tables, views, procedures, functions, etc.

- **Justification:** Information for approvers justifying why the user is requesting such a session. If the 'Require specific justification format' checkbox is selected in [Portal Configuration](#), justification is required; however, if it does not match the provided regex, an appropriate message will appear.
- **Start Timestamp:** Indicates the time when the approved session is scheduled to start. It can be set to a time in the past.
- **Duration:** Session duration, with the option to choose in various time units. The duration cannot be greater than the set maximum time for the selected role; otherwise, it will not be possible to request a session.

And there is also an icon next to each role, which shows the emails of approvers associated with that role. The envelope icon is visible only when this option is selected in the [Admin tab](#).

Note: Always enter the Session Start Timestamp in this tab according to your local time zone; it will be automatically converted if the Central Manager is in a different time zone.

Note: If any role possesses any special roles such as SUPERUSER, CREATEDB, etc., those will be automatically inherited, even for the PostgreSQL database type.

Warning: This feature is **NOT** functional when using MySQL with version < 8.0 and PostgreSQL with version < 11.0.

Pending Approvals Overview

The Pending Approvals tab displays all requested sessions that have not yet received approval or denial and have a status of "CREATED". Only sessions that a particular user (approver) can approve or reject are shown. The relevant columns provide information such as the session request date, data source, user, requested roles, justification, approvals' progress bar and the session duration. The roles that will be approved by clicking the "Approve" button are in bold, while roles that have already been approved are in green.

By clicking "Approve", the status of the corresponding roles that the approver can approve changes to "approved". Only after obtaining approval for all roles in a particular session does the session become activated. In the case of clicking "Deny", the specified roles are denied, and the entire session is rejected, regardless of whether other approvers have approved different roles.

When a session is approved or denied for the user who requested it, a corresponding notification is sent to their email address.

Pending Approvals

Data Source	User	Selected Roles	Justification	Flags	Start Timestamp UTC+2 ↓	End Timestamp UTC+2	Approvals Progress	Action
Postgres-source17	mark ⓘ	role1 role3	test1	Preserve Objects: ☒	Aug 14, 2025 9:25:00 AM	Aug 14, 2025 10:25:00 AM	1 / 2	<button>Approve</button> <button>Deny</button>
Postgres-source17	requester ⓘ	jenkins role1 role2 role3 requester_base_portal_user	test2	Inherit Objects: ☐ Preserve Objects: ☒	Aug 14, 2025 10:12:00 AM	Aug 14, 2025 11:12:00 AM	3 / 4	<button>Approve</button> <button>Deny</button>
oracle-source	requester ⓘ	jenkins PROVISIONER role1	test3	Preserve Objects: ☒	Aug 14, 2025 10:15:00 AM	Aug 14, 2025 1:15:00 PM	0 / 1	<button>Approve</button> <button>Deny</button>

If the approver will decide to deny the session request, the denial reason can be attached to the notification and sent to the requester of the session.

Denial reason

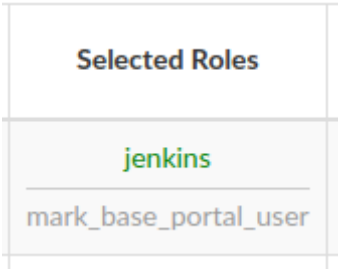
Denial reason:

Add denial reason...

Close

Deny request

Note: If a grayed-out element with the suffix "base_portal_user" appears at the bottom of the selected roles list, it means the session was requested with the "Inherit Objects" option enabled (more info here: [Request Access](#)). This means the session user will gain access to objects created during their previous sessions.



Note: By default, the time is displayed in the time zone of the Central Manager. This time zone can be identified in the column names Start Timestamp and End Timestamp, where the corresponding time zone information is appended, e.g., "UTC +1". In cases where the client is in a different time zone, the time is still shown in the Central Manager's time zone by default. However, the user can optionally select the Local Timezone checkbox, which will adjust all timestamps to the client's current time zone.

Pending Approvals

Local Timezone: ☒ 10 20 50

Data Source	User	Selected Roles	Justification	Flags	Start Timestamp ↓	End Timestamp	Approvals Progress	Action
Postgres-source17	mark ⓘ	role1 role3	test1	Preserve Objects: ☐	Aug 14, 2025 9:25:00 AM	Aug 14, 2025 10:25:00 AM	1 / 2	<button>Approve</button> <button>Deny</button>
Postgres-source17	requester ⓘ	jenkins role1 role2 role3 requester_base_portal_user	test2	Inherit Objects: ☐ Preserve Objects: ☐	Aug 14, 2025 10:12:00 AM	Aug 14, 2025 11:12:00 AM	3 / 4	<button>Approve</button> <button>Deny</button>
oracle-source	requester ⓘ	jenkins PROVISIONER role1	test3	Preserve Objects: ☐	Aug 14, 2025 10:15:00 AM	Aug 14, 2025 1:15:00 PM	0 / 1	<button>Approve</button> <button>Deny</button>

External Approvals Overview

The External Approvals option is a feature that allows external users (without a Heimdall account) to approve or deny sessions requested in the portal. To enable this, several prerequisites must be met.

Prerequisites

The first step is to make sure that portalMode is enabled. If it is, go to [Notifications](#) and specify which configured notifications should be used for External Approvals by checking the appropriate checkbox.

Administration

Account Information
License
Alerts
Login History
Log Management
Security Tags
Software Management
Notifications
Portal
SMTP Configuration
Server Properties
Config Management
Secrets Manager Configuration
LDAP Configuration
Kerberos Configuration
SAML Configuration

Notifications Commit

System notification: none

#	Protocol	Notification Alias	Configuration	Send Test	External Approvals	Delete
1.	SMTP	none	Heimdall Username, for SMTP lookup	Test	☐	
2.	SMTP	admin	admin	Test	☐	
3.	LDAP	ldap	test-users	Test	☐	
4.	SQL	SQL_notification	Data Source: Postgres-source17 SQL: select email from public.myusers where id > 1 and id < 4	Test	☐	
5.	SIMPLE	SIMPLE_notification	test@gmail.com	Test	☐	

Add Notification Entry

Next, assign such a notification config to the role which, when requested, will trigger emails being sent. You can configure this in [Roles Management](#).

In the next step, go to [Portal Basic](#) and configure the following fields:

- Portal Host - used when sending emails to session approvers. The URL that approvers should open is included here.
- JWT Signature Key - used to sign and validate the token signature. The signatureKey must be provided in Base64Url format, and once decoded it must have a minimum length of 256 bytes. In its encoded form, only the characters A–Z, a–z, 0–9, -, and _ are allowed. You can also use secrets by setting the username and password fields. For example, set the token as the username

and its value as the password. *Note:* The signature key does not support secret rotation at this time. Alternatively, you can generate the key by clicking the corresponding button.

Administration

- Account Information
- License
- Alerts
- Login History
- Log Management
- Security Tags
- Software Management
- Notifications
- Portal**
- SMTP Configuration
- Server Properties
- Config Management
- Secrets Manager Configuration
- LDAP Configuration
- Kerberos Configuration
- SAML Configuration
- Password Policy
- Modules
- TCP

Portal Basic

Commit

+

LDAP Configuration: LDAP_CONFIG

Portal data source: MySQL-source

Portal Host: https://example.org:8443

JWT Signature Key: Secret

Send audit logs to CloudWatch: ☒

CloudWatch namespace: heimdall-audit

Send audit logs to CloudTrail: ☒

Send audit queries logs to CloudTrail: ☒

Cloud Trail Lake Channel ARN: arn:aws:cloudtrail:us-east-1:272911111111:channel/16e1b5bb-b564-463

Show approvers emails: ☒

Synchronize Portal Users: ☒

Send Calendar Item: ☒

Require specific justification format: ☒

Regex pattern: Match all if empty

Invalid justification message: Does not match the global regex pattern.

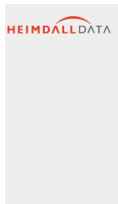
Notifications

+

[Smtp Configuration](#) still needs to be configured, since emails to external approvers are sent using this protocol.

JWT Token Handling for External Approvals

Once the above requirements are met, after requesting a session for the corresponding role, emails will be sent to the designated recipients. These emails include a link that contains a token. When accessed, the link allows approvers to approve or deny the session.



[Documentation](#) [PDF](#) [Support](#) [test.user@heimdalldata.com](#) 

Pending Approvals

Data Source	User	Selected Roles	Justification	Start Timestamp UTC+1	End Timestamp UTC+1	Approvals Progress	Action
PostgreSQLDataSource	arozuk	jenkins admins	test	Feb 3, 2026, 5:22:00 PM	Feb 3, 2026, 5:52:00 PM	0 / 1	Approve Deny

1. Using External Clients or API Consumers

- Clients (e.g., Java applications or automated scripts) using the external approval API must:
 - Store and send the JWT token explicitly in the `Authorization: Bearer <token>` header on requests to the external approval endpoints.
 - Avoid mixing JWT token usage with Basic Auth or session-based authentication requests to GUI endpoints in the same client instance, as this may cause session invalidation or authentication conflicts.
 - Maintain separate HTTP client instances or configurations for:
 - Stateful GUI requests (using Basic Auth and cookies)
 - Stateless external approval requests (using JWT)

You can approve or deny a session using `curl`:

Approve Session

```
curl -X POST "${portalHost}/api/pending-approvals/${sessionId}/approve" \
-H "Authorization: Bearer ${token}"
```

Deny Session

You can deny a session using `curl` with JSON in the body:

```
curl -X POST "${portalHost}/api/pending-approvals/deny" \
-H "Authorization: Bearer ${token}" \
-H "Content-Type: application/json" \
-d '{
  "sessionId": "${sessionId},
```

```
"denialReason": "{$denialReason}"  
}'
```

2. Important Notes

- Avoid storing the external JWT token in global browser storage (`localStorage`), as it will be attached to all requests on the domain, breaking GUI sessions.
- Use `sessionStorage` or in-memory variables to keep JWT tokens scoped to individual tabs or approval workflows.
- Backend security configuration separates GUI and external approval endpoints with distinct security filters to prevent session collisions.

If these guidelines are followed, both GUI users and external approvers can operate in parallel without session conflicts or authentication issues.

Sessions Overview

The Sessions tab provides the ability to view all sessions that have been given the "APPROVED" or "STARTED" status, as well as those with the "CREATED" status. The tabular layout provides information for each session, including the roles requested, session expiration time, approvals' progress bar, information about which roles have already been approved (displayed in green), and user credentials for login. These credentials are only for temporary users with selected roles and are valid for the duration of the session.

The password can be viewed and copied only by the user who requested the session; the admin does not have this permission.

Admins have the authority to terminate such sessions either individually or by pressing the "Terminate All Sessions" button, which will terminate all started sessions. Meanwhile, a user who requested their own session can cancel it at any time before the session expires.

Sessions

Terminate All Sessions

102050

Data Source	Selected Roles	Status	Flags	Start Timestamp UTC+2 ↓	End Timestamp UTC+2 ↑	JDBC URL	Username	Password	Approvals Progress	Terminate
Postgres-source17	role1 role3	CREATED	Preserve Objects: ☐	Aug 14, 2025 9:25:00 AM	Aug 14, 2025 10:25:00 AM	jdbc:postgresql://local... 	mark	Pending approvals... 	1 / 2	
Postgres-source17	jenkins role1 role2 role3	CREATED	Inherit Objects: ☐ Preserve Objects: ☐	Aug 14, 2025 10:12:00 AM	Aug 14, 2025 11:12:00 AM	jdbc:postgresql://local... 	requester	Pending approvals... 	3 / 4	
oracle-source	jenkins PROVISIONER role1	CREATED	Preserve Objects: ☐	Aug 14, 2025 10:15:00 AM	Aug 14, 2025 1:15:00 PM	jdbc:oracle:thin:@local... 	requester	Pending approvals... 	0 / 1	
Postgres-source17	role2	STARTED		Aug 14, 2025 10:15:00 AM	Aug 14, 2025 10:45:00 AM	jdbc:postgresql://local... 	admin_1755159388397	***** 	1 / 1	

Note: If a grayed-out element with the suffix "base_portal_user" appears at the bottom of the selected roles list, it means the session was requested with the "Inherit Objects" option enabled (more info here: [Request Access](#)). This means the session user will gain access to objects created during their previous sessions.

Selected Roles
jenkins
mark_base_portal_user

Note: By default, the time is displayed in the time zone of the Central Manager. This time zone can be identified in the column names Start Timestamp and End Timestamp, where the corresponding time zone information is appended, e.g., "UTC +1". In cases where the client is in a different time zone, the time is still shown in the Central Manager's time zone by default. However, the user can

optionally select the *Local Timezone* checkbox, which will adjust all timestamps to the client's current time zone.

Sessions

Terminate All Sessions

Local Timezone:

📅

🕒

20

50

Data Source	Selected Roles	Status	Flags	Start Timestamp ↓	End Timestamp	JDBC URL	Username	Password	Approvals Progress	Terminate
Postgres-source17	role1 role3	CREATED	Preserve Objects: ☒	Aug 14, 2025 9:25:00 AM	Aug 14, 2025 10:25:00 AM	jdbc:postgres://local... 🔒	mark	🔒 Pending approvals...	1 / 2	✖
Postgres-source17	jenkins role1 role2 role3	CREATED	Inherit Objects: ☐ Preserve Objects: ☒	Aug 14, 2025 10:12:00 AM	Aug 14, 2025 11:12:00 AM	jdbc:postgres://local... 🔒	requester	🔒 Pending approvals...	3 / 4	✖
oracle-source	jenkins PROVISIONER role1	CREATED	Preserve Objects: ☒	Aug 14, 2025 10:15:00 AM	Aug 14, 2025 1:15:00 PM	jdbc:oracle:thin@local... 🔒	requester	🔒 Pending approvals...	0 / 1	✖
Postgres-source17	role2	STARTED		Aug 14, 2025 10:15:00 AM	Aug 14, 2025 10:45:00 AM	jdbc:postgres://local... 🔒	admin_1755159388397	🔒 ***** 🔍 🔒	1 / 1	✖

CloudBeaver Integration

The CloudBeaver integration allows you to access and manage a data source directly from your web browser, without leaving the browser.

CloudBeaver is an open-source database manager based on DBeaver. This integration removes the need to copy ephemeral credentials into a third-party tool, allowing you to launch secure database sessions instantly.

Prerequisites

CloudBeaver is automatically configured with the Heimdall Console.

Note: Please ensure your browser's pop-up blocker is disabled for this site. The CloudBeaver interface launches in a new tab, which some browsers may block by default.

Accessing CloudBeaver

You can launch CloudBeaver directly from the **Sessions** tab.

- **Request Access:** Ensure you have requested access to a data source through the portal.
- **Locate the Session:** Once requested, your session will appear in the list. Initially, the DB button in the 'Access' column will be disabled.

Sessions

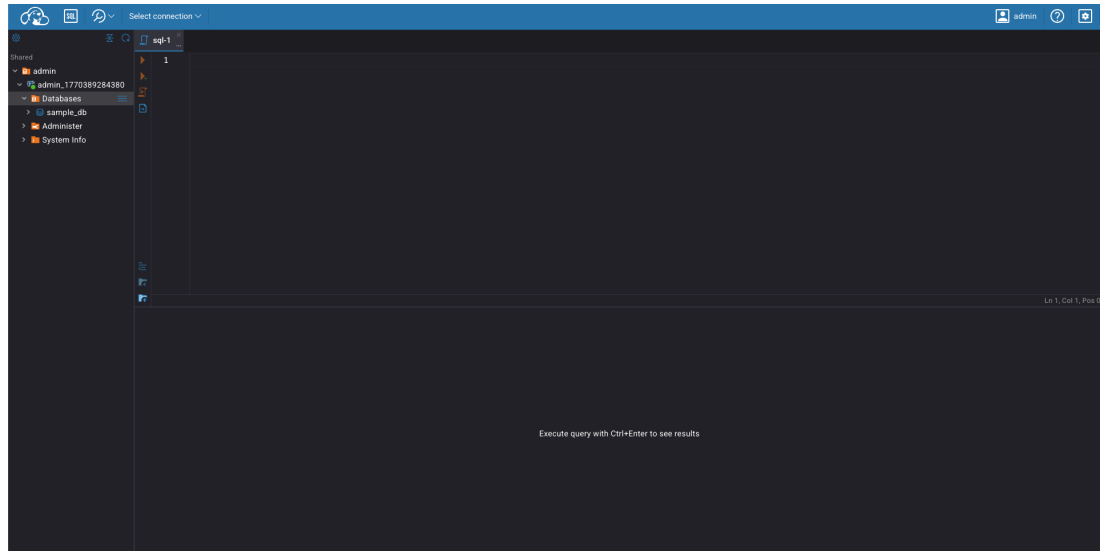
											10 20 50
Data Source	Selected Roles	Status	Flags	Start Timestamp UTC-3 ↓	End Timestamp UTC-3	JDBC URL	Username	Password	Approvals Progress	Access	Cancel
Postgres-source	jenkins	CREATED	Preserve Objects: ☐	Feb 06, 2026 11:45:00 AM	Feb 06, 2026 12:15:00 PM	jdbc:postgresql://127... 	Pending approvals...	Pending approvals...	0 / 1		

- **Launch the Session:** Once your session is approved and started, the DB button will become active.

Sessions

											10 20 50
Data Source	Selected Roles	Status	Flags	Start Timestamp UTC-3	End Timestamp UTC-3	JDBC URL	Username	Password	Approvals Progress	Access	Cancel
Postgres-source	jenkins	STARTED	Preserve Objects: ☐	Feb 06, 2026 11:45:00 AM	Feb 06, 2026 12:15:00 PM	jdbc:postgresql://127... 🔒	admin_1770389284380 🔒	***** 🔒	1/1	DB	X

- **Start Managing Data:** Click the DB button to open CloudBeaver in a new tab. The session is pre-initialized, so you do not need to manually input credentials.



Session history overview

The **Session History** tab provides different functionality depending on the user's role.

For Portal Users

- Portal Users can view **only their own** past sessions.
- Portal Users can delete **only their own** session data - once deleted, the session will no longer be visible to the user.

For Admins

- Admins can view the **complete history** of all users' sessions.
- Admins can delete **any session**, regardless of ownership - sessions deleted by an admin are removed from visibility **for all users**.
- Admins can filter records by enabling the "**Show deleted**" checkbox. This includes sessions that were deleted by users and would otherwise be hidden.

Data Retention

Note: Even if session data is deleted and no longer visible in the interface, **all data remains stored** in the portal's database for auditing and compliance purposes.

Sessions visible in this tab are not associated with any database user or their credentials. Either they were never assigned, or their credentials were already removed. Admins can filter the records by checking the "Show deleted" checkbox, which will also display those that were removed from the history. The "Status" column indicates the manner in which the session ended:

- **DENIED:** The session was denied by one of the approvers, as such credentials were never given.
- **CANCELED:** The session was ended due to cancellation by the requesting user.
- **TERMINATED:** An admin terminated the session.
- **EXPIRED:** The session timed out.

Session history

Clear Session History

Show deleted: ☒ 10 20 50

Data Source	User	Selected Roles	Status	Flags	Start Timestamp UTC+2 ↑	End Timestamp UTC+2	Delete
Postgres-source17	mark	role1 role3 mark_base_portal_user	CANCELED	Inherit Objects: ☐	Aug 13, 2025, 4:20:00 PM	Aug 13, 2025, 5:20:00 PM	✗
Postgres-source17	admin	role2	DENIED	Preserve Objects: ☒	Aug 13, 2025, 3:40:00 PM	Aug 13, 2025, 4:10:00 PM	✗
Postgres-source17	admin	role2 admin_base_portal_user	EXPIRED	Inherit Objects: ☐ Preserve Objects: ☒	Aug 13, 2025, 11:13:00 AM	Aug 13, 2025, 11:43:00 AM	✗
Postgres-source17	mark	role3	EXPIRED	Preserve Objects: ☐	Aug 13, 2025, 11:13:00 AM	Aug 13, 2025, 12:13:00 PM	✗
Postgres-source17	admin	role1	EXPIRED		Aug 13, 2025, 10:55:00 AM	Aug 13, 2025, 12:55:00 PM	✗
Postgres-source17	admin	role1	TERMINATED		Aug 13, 2025, 10:55:00 AM	Aug 13, 2025, 12:55:00 PM	✗
Postgres-source17	admin	role1	TERMINATED		Aug 13, 2025, 10:54:00 AM	Aug 13, 2025, 12:54:00 PM	✗
Postgres-source17	admin	role2	TERMINATED		Aug 12, 2025, 5:10:00 PM	Aug 12, 2025, 5:40:00 PM	✗
Postgres-source17	admin	role2	TERMINATED		Aug 12, 2025, 5:09:00 PM	Aug 12, 2025, 5:39:00 PM	✗
Postgres-source17	admin	role2	DENIED	Preserve Objects: ☐	Aug 12, 2025, 5:09:00 PM	Aug 12, 2025, 5:39:00 PM	✗

1 2 ... 29

Go to page: Go

1 of 29

Note: If a grayed-out element with the suffix "base_portal_user" appears at the bottom of the selected roles list, it means the session was requested with the "Inherit Objects" option enabled (more info here: [Request Access](#)). This means the session user will gain access to objects created during their previous sessions.

Selected Roles
jenkins
mark_base_portal_user

Note: By default, the time is displayed in the time zone of the Central Manager. This time zone can be identified in the column names Start Timestamp and End Timestamp, where the corresponding time zone information is appended, e.g., "UTC +1". In cases where the client is in a different time zone, the time is still shown in the Central Manager's time zone by default. However, the user can optionally select the Local Timezone checkbox, which will adjust all timestamps to the client's current time zone.

Session history

Clear Session History

Show deleted: ☒ Local Timezone: ☒ 10 20 50

Data Source	User	Selected Roles	Status	Flags	Start Timestamp ↑	End Timestamp	Delete
Postgres-source17	mark ⓘ	role1 role3	● CANCELED	Inherit Objects: ☐	Aug 13, 2025, 4:20:00 PM	Aug 13, 2025, 5:20:00 PM	✖
		mark_base_portal_user					
Postgres-source17	admin ⓘ	role2	● DENIED	Preserve Objects: ☒	Aug 13, 2025, 3:40:00 PM	Aug 13, 2025, 4:10:00 PM	✕
Postgres-source17	admin ⓘ	role2	● EXPIRED	Inherit Objects: ☐ Preserve Objects: ☒	Aug 13, 2025, 11:13:00 AM	Aug 13, 2025, 11:43:00 AM	✖
		admin_base_portal_user					
Postgres-source17	mark ⓘ	role3	● EXPIRED	Preserve Objects: ☐	Aug 13, 2025, 11:13:00 AM	Aug 13, 2025, 12:13:00 PM	✖
Postgres-source17	admin ⓘ	role1	● EXPIRED		Aug 13, 2025, 10:55:00 AM	Aug 13, 2025, 12:55:00 PM	✕
Postgres-source17	admin ⓘ	role1	● TERMINATED		Aug 13, 2025, 10:55:00 AM	Aug 13, 2025, 12:55:00 PM	✖
Postgres-source17	admin ⓘ	role1	● TERMINATED		Aug 13, 2025, 10:54:00 AM	Aug 13, 2025, 12:54:00 PM	✖
Postgres-source17	admin ⓘ	role2	● TERMINATED		Aug 12, 2025, 5:10:00 PM	Aug 12, 2025, 5:40:00 PM	✖
Postgres-source17	admin ⓘ	role2	● TERMINATED		Aug 12, 2025, 5:09:00 PM	Aug 12, 2025, 5:39:00 PM	✖
Postgres-source17	admin ⓘ	role2	● DENIED	Preserve Objects: ☐	Aug 12, 2025, 5:09:00 PM	Aug 12, 2025, 5:39:00 PM	✖

Audit

Audit Overview

The Audit tab allows both viewing and downloading records from the audit trail table for all portal operations. It is accessible when portalMode is enabled and the user has the Audit User role set in the [Users tab](#).

You can define multiple filters to refine the records displayed. For example, you can specify a date range using "Since" (start date) and "Until" (end date) or apply filters based on specific column values. Multiple filters can be combined to narrow down the results effectively. Additionally, you have the option to download the filtered data to a CSV or JSON file, selectable on the right side of the Download Records button. To ensure the filters apply to the downloaded data, check the box labeled "Include filters" in the dropdown menu next to the Download button. This way, only the records matching your selected criteria will be included in the download. If the "Include filters" checkbox is not selected, all records will be downloaded regardless of the applied filters.

Note: Records marked with a light blue background pertain to roles, not the entire session, for easier browsing.

Audit

Download Records

Sessions Audit

Queries Audit

(UTC+2) Since: mm/dd/yyyy, --:-- --

Until: mm/dd/yyyy, --:-- --

Reset Filters

Filter by: Data Source

Postgres-source17

Filter by: Special Roles

false

+

Get Records

20

50

100

Log Timestamp UTC+2 ↑	Username	State	DataSource	Roles	Flags	Session Start Time UTC+2	Session End Time UTC+2	View Queries
Aug 13, 2025, 4:04:58 PM	admin	DENIED	Postgres-source17	role2	Preserve Objects: ☒	Aug 13, 2025, 3:40:00 PM	Aug 13, 2025, 4:10:00 PM	Q
Aug 13, 2025, 4:04:58 PM	john	DENIED	Postgres-source17	role2	Preserve Objects: ☒			Q
Aug 13, 2025, 3:41:00 PM	admin	CREATED	Postgres-source17	role2	Preserve Objects: ☒	Aug 13, 2025, 3:40:00 PM	Aug 13, 2025, 4:10:00 PM	Q
Aug 13, 2025, 3:24:00 PM	admin	EXPIRED	Postgres-source17	role1		Aug 13, 2025, 10:58:44 AM	Aug 13, 2025, 3:24:00 PM	Q
Aug 13, 2025, 12:13:00 PM	mark	EXPIRED	Postgres-source17	role3	Preserve Objects: ☒		Aug 13, 2025, 12:13:00 PM	Q
Aug 13, 2025, 11:43:00 AM	admin	EXPIRED	Postgres-source17	role2	Inherit Objects: ☒ Preserve Objects: ☒		Aug 13, 2025, 11:43:00 AM	Q
Aug 13, 2025, 11:14:04 AM	admin	CREATED	Postgres-source17	role2	Inherit Objects: ☒ Preserve Objects: ☒	Aug 13, 2025, 11:13:00 AM	Aug 13, 2025, 11:43:00 AM	Q
Aug 13, 2025, 11:13:35 AM	mark	CREATED	Postgres-source17	role3	Preserve Objects: ☒	Aug 13, 2025, 11:13:00 AM	Aug 13, 2025, 12:13:00 PM	Q
Aug 13, 2025, 11:13:11 AM	admin	TERMINATED	Postgres-source17	role1		Aug 13, 2025, 10:54:00 AM	Aug 13, 2025, 11:13:11 AM	Q
Aug 13, 2025, 10:58:44 AM	admin	STARTED	Postgres-source17	role1		Aug 13, 2025, 10:58:44 AM	(est) Aug 13, 2025, 12:55:00 PM	Q
Aug 13, 2025, 10:58:44 AM	john	APPROVED	Postgres-source17	role1				Q
Aug 13, 2025, 10:55:58 AM	admin	CREATED	Postgres-source17	role1		Aug 13, 2025, 10:55:00 AM	Aug 13, 2025, 12:55:00 PM	Q

Note: By default, the time is displayed in the time zone of the Central Manager. This time zone can be identified in the column names Log Timestamp, Session Start Timestamp and Session End Timestamp, where the corresponding time zone information is appended, e.g., "UTC +1". In cases where the client is in a different time zone, the time is still shown in the Central Manager's time zone by default. However, the user can optionally select the Local Timezone checkbox, which will adjust all timestamps to the client's current time zone. Furthermore, when the Local Timezone checkbox is selected, the times entered in the filters can be provided in your local time zone, and they will be automatically converted accordingly. Additionally, records downloaded via the Download Records button will always be provided in the time zone of the Central Manager, with the corresponding time zone information included in the column names containing timestamps.

Audit								
								Download Records
Sessions Audit Queries Audit								
Since: mm/dd/yyyy, --:-- -- Until: mm/dd/yyyy, --:-- -- Local Timezone: ☒ Reset Filters								
Filter by: Data Source Postgres-source17 x Get Records								
Filter by: Special Roles false x +								
20 50 100								
Log Timestamp ↑	Username	State	DataSource	Roles	Flags	Session Start Time	Session End Time	View Queries
Aug 13, 2025, 02:04:58 PM	admin	DENIED	Postgres-source17	role2	Preserve Objects: ☐	Aug 13, 2025, 01:40:00 PM	Aug 13, 2025, 02:10:00 PM	Q
Aug 13, 2025, 02:04:58 PM	john	DENIED	Postgres-source17	role2	Preserve Objects: ☐			Q
Aug 13, 2025, 01:41:00 PM	admin	CREATED	Postgres-source17	role2	Preserve Objects: ☐	Aug 13, 2025, 01:40:00 PM	Aug 13, 2025, 02:10:00 PM	Q
Aug 13, 2025, 01:24:00 PM	admin	EXPIRED	Postgres-source17	role1		Aug 13, 2025, 08:58:44 AM	Aug 13, 2025, 01:24:00 PM	Q
Aug 13, 2025, 10:13:00 AM	mark	EXPIRED	Postgres-source17	role3	Preserve Objects: ☐		Aug 13, 2025, 10:13:00 AM	Q
Aug 13, 2025, 09:43:00 AM	admin	EXPIRED	Postgres-source17	role2	Inherit Objects: ☐ Preserve Objects: ☐		Aug 13, 2025, 09:43:00 AM	Q
Aug 13, 2025, 09:14:04 AM	admin	CREATED	Postgres-source17	role2	Inherit Objects: ☐ Preserve Objects: ☐	Aug 13, 2025, 09:13:00 AM	Aug 13, 2025, 09:43:00 AM	Q
Aug 13, 2025, 09:13:35 AM	mark	CREATED	Postgres-source17	role3	Preserve Objects: ☐	Aug 13, 2025, 09:13:00 AM	Aug 13, 2025, 10:13:00 AM	Q
Aug 13, 2025, 09:13:11 AM	admin	TERMINATED	Postgres-source17	role1		Aug 13, 2025, 08:54:00 AM	Aug 13, 2025, 09:13:11 AM	Q
Aug 13, 2025, 08:58:44 AM	admin	STARTED	Postgres-source17	role1		Aug 13, 2025, 08:58:44 AM	(est) Aug 13, 2025, 10:55:00 AM	Q
Aug 13, 2025, 08:58:44 AM	anna@gm.com	APPROVED	Postgres-source17	role1				Q
Aug 13, 2025, 08:55:58 AM	admin	CREATED	Postgres-source17	role1		Aug 13, 2025, 08:55:00 AM	Aug 13, 2025, 10:55:00 AM	Q

Each record represents one of the possible states. Here are the explanations:

- **APPROVED** - Indicates that a role or session has been approved by an approver or was auto-approved if approvals needed is set to 0.
- **CANCELED** - Indicates that the session was canceled and ended by the user who initially requested it.
- **CREATED** - Indicates that the session was created.
- **DENIED** - Indicates that a role, and consequently the session, was denied by an approver.
- **EXPIRED** - Indicates that the session's time has run out and it has been terminated.
- **STARTED** - Indicates that the session has started as its scheduled start time has arrived.
- **TERMINATED** - Indicates that the session was terminated by an admin.

Additionally, there is an option to view session details by hovering over its username in the Username column. A small popup will display basic information about the session, including all roles and the justification provided by the requester.

Jul 2, 2025, 2:55:06 PM	mark ⓘ	STARTED	Postgres-source17	
Jul 2, 2025, 2:55:05 PM	Session ID: 1063 Start Time: Jul 2, 2025, 2:55:06 PM UTC+2 End Time: Jul 2, 2025, 2:55:42 PM UTC+2 Session Username: mark_1751460905947 Roles: jenkins, role3 Justification: Access is needed to edit sample things			Postgres-source17
Jul 2, 2025, 2:54:57 PM				Postgres-source17
Jul 2, 2025, 2:54:57 PM				Postgres-source17

Note: If the session was requested with the Inherit Objects option, a corresponding row with the name requester + '_base_portal_user' will also be visible, from which the privileges to the objects will be inherited. Additionally, for session times we can see (est), indicating that the time is estimated, since the session hasn't started yet and we don't know the real timestamps, and the Session Username will be to be determined once the session actually starts.

Example:

Filter by: Session ID 281
Get Records

Log Timestamp UTC+2 ↑	Username	State	DataSource	Roles	Flags	Session Start Time UTC+2	Session End Time UTC+2	View Queries
Aug 13, 2025, 4:22:23 PM	mark ⓘ	CANCELED	Postgres-source17	role1 role3	Inherit Objects: ☒	Aug 13, 2025, 4:20:00 PM	Aug 13, 2025, 4:22:23 PM	Q
Aug 13, 2025, 4:20:00 PM	mark ⓘ	STARTED	Postgres-source17	role1 role3	Inherit Objects: ☒	Aug 13, 2025, 4:20:00 PM	(est) Aug 13, 2025, 5:20:00 PM	Q
Aug 13, 2025, 4:19:11 PM	mark ⓘ	APPROVED	Postgres-source17	role1 role3	Inherit Objects: ☒	Aug 13, 2025, 4:20:00 PM	Aug 13, 2025, 5:20:00 PM	Q
Aug 13, 2025, 4:19:10 PM	john ⓘ	APPROVED	Postgres-source17	role3	Inherit Objects: ☒			Q
Aug 13, 2025, 4:18:52 PM	admin ⓘ	APPROVED	Postgres-source17	role1	Inherit Objects: ☒			Q
Aug 13, 2025, 4:18:35 PM	mark ⓘ	CREATED	Postgres-source17	role1 role3	Inherit Objects: ☒	Aug 13, 2025, 4:20:00 PM	Aug 13, 2025, 5:20:00 PM	Q

In the screenshot above, we see example audit records for a session with ID 281. Reading from the bottom, we have the following information:

1. Mark requested the session for roles role1 and role3 with the Inherit Objects option enabled. The session received the status CREATED, but it is not yet started and is waiting for approvals.
2. The user admin approves the role role3. A record with the APPROVED status, empty fields for session start time, end time, and a light blue background indicates that this record pertains to a role(s).
3. Similarly, the user john approves the role role1.

- After receiving approvals for role1 and role3, the session changes its status to APPROVED. However, the session hasn't started yet – it will only begin once the start time is reached.
- The next record indicates that the session start time has arrived, and the session is marked as STARTED, with the session user now created. From this point on, the created user has the requested roles and can log into the database. (est) next to Session End Time indicates the expected end time of the session, unless it gets terminated or cancelled earlier.
- Finally, Mark decided he no longer needed the session, so he canceled it. The session changed its status to CANCELED, and the session user was removed from the database, meaning it can no longer be used.

You can download your filtered data as a CSV or JSON file by using "Download format" in the dropdown menu next to the Download Records button. To ensure only filtered records are included, check the Include filters option in the same menu. If you leave this option unchecked, the download will include all records, regardless of any filters you've applied.

Audit

Download Records

AVAILABLE OPTIONS:

☐ Include filters

CSV

CSV

JSON

Download format

Get Records

Sessions Audit

Queries Audit

(UTC+2) Since: dd.mm.rrrr, --:--

Until: dd.mm.rrrr, --:--

Filter by:

Session User Activity Overview

By clicking on the Queries Audit or View Queries, you will be redirected to a new tab where you can review the user's activity during that session.

Sessions Audit

Queries Audit

(UTC+2) Since: mm/dd/yyyy, --:--

Until: mm/dd/yyyy, --:--

Filter by: Session ID 621

Reset Filters

Get Records

Log Timestamp UTC+2 ↑	Session Username	Session ID	VDB	SQL	Catalog	Schema	Exception
Jun 6, 2025, 4:49:10 PM	admin_1749221249181	621	Postgresvdb-vdb	SELECT c.oid, a.attnum, a.attname, c.relname, n.nspname, a.attnotnull OR (t.typtype = 'd' AND t.typonotnull), a.attidentity != '' OR pg_catalog.pg_get_expr(d.adbin, d.adrelid) LIKE 'nextval(%' FROM pg_catalog.pg_class c JOIN pg_catalog.pg_namespace n ON (c.relnamespace = n.oid) JOIN pg_catalog.pg_attribute a ON (c.oid = a.attrelid) JOIN pg_catalog.pg_type t ON (a.atttypid = t.oid) LEFT JOIN pg_catalog.pg_attrdef d ON (d.adrelid = a.attrelid AND d.adnum = a.attnum) JOIN (SELECT 2615 AS oid, 1 AS attnum UNION ALL SELECT 1247, 1 UNION ALL SELECT 11527, 1) vals ON (c.oid = vals.oid AND a.attnum = vals.attnum)	test	public:"\$user"	
Jun 6, 2025, 4:49:10 PM	admin_1749221249181	621	Postgresvdb-vdb	SELECT c.oid, a.attnum, a.attname, c.relname, n.nspname, a.attnotnull OR (t.typtype = 'd' AND t.typonotnull), a.attidentity != '' OR pg_catalog.pg_get_expr(d.adbin, d.adrelid) LIKE 'nextval(%' FROM	test	public:"\$user"	
Jun 6, 2025, 4:49:10 PM	admin_1749221249181	621	Postgresvdb-vdb	SELECT n.nspname AS schema_name, t.typname AS object_name, r.rolname AS owner FROM pg_type t JOIN pg_namespace n ON n.oid = t.typonamespace JOIN pg_roles r ON r.oid = t.typowner WHERE t.typtype = 'd'	test	public:"\$user"	
Jun 6, 2025, 4:49:10 PM	admin_1749221249181	621	Postgresvdb-vdb	SELECT n.nspname AS schema_name, t.typname AS object_name, r.rolname AS owner FROM pg_type t JOIN pg_namespace n ON n.oid = t.typonamespace JOIN pg_roles r ON r.oid = t.typowner WHERE t.typtype = 'd'	test	public:"\$user"	
Jun 6, 2025, 4:49:10 PM	admin_1749221249181	621	Postgresvdb-vdb	SELECT session_user, current_user	test	public:"\$user"	
Jun 6, 2025, 4:49:10 PM	admin_1749221249181	621	Postgresvdb-vdb	SELECT session_user, current_user	test	public:"\$user"	

In this view, you can examine the queries executed by the user to proxy, along with several important details, such as:

- The VDB (Virtual Database) where the SQL statement was executed
- The executed SQL statement
- The catalog in which the SQL statement was executed
- The default schema that was active at the time of execution
- Any exception thrown during the SQL statement execution

You can download your filtered data as a CSV or JSON file by using "Download format" in the dropdown menu next to the Download Records button. To ensure only filtered records are included, check the Include filters option in the same menu. If you leave this option unchecked, the download will include all records, regardless of any filters you've applied.



Session user activity logging can be enabled or disabled in the [VDB](#) settings tab (enabled by default).

—

Logging

Log Connections:

☐

Log All SQL:

☐

Log All Portal Sessions SQL:

☒

Exclude parameters:

☒

Here, also parameters can be excluded from logged Portal Session SQL queries, e.g. 'INSERT INTO table VALUES (?, ?)'.

Database Browser

Database Browser

Filters

DatabaseIdentitySchemaTableColumnView

Security Tags: No tag filter

Include Predefined Identities:

- postgres

Security Tags: Add tag

- Users

ephemeraluser1

Granted Role	Grantor	Source
role1	test	DIRECT
role2	test	role1
role3	test	role2

expireduser1

lockeduser1

superuser1

test

- Roles

role1

role2

role3

+ template0

+ template1

+ test

Global Privileges

BYPASSRLS

CREATEDB

CREATEROLE

INHERIT

LOGIN

REPLICATION

SUPERUSER

Schemas

+ information_schema

+ pg_catalog

+ pg_toast

+ public

- schema1

Security Tags: Add tag

+ Privileges

+ Tables

+ Foreign tables

+ Views

+ Materialized views

+ Routines

+ Sequences

+ Types

+ Domains

Commit

Refresh

Report

The Database Browser is a versatile tool designed for viewing database structures and modifying permissions. It operates on multiple levels to provide detailed access and control. It is possible to filter resources or assign a [security tag](#).

Data Source User

Ensure that the configured [Data Source](#) user has the **superuser privileges** (i.e. highest privileges). This is necessary because we need to get information about other users and their privileges. Without the **highest privileges**, the Database Browser may function incorrectly, leading to potential issues such as inaccurate information and the inability to update privileges.

Supported Database Types

The supported databases are PostgreSQL, Greenplum, Redshift, MySQL, SQLServer, and Oracle.

Warning in MySQL: This feature is **NOT** functional when using MySQL with version < 8.0.

Warning: Reports in SQLServer with aggregated permissions are not functional yet.

Database Structure

Databases



- + database1
- + database2
- + postgres

The entry point where you select the database to work with.

Identities

📁 - Users

👤 ephemeraluser1

Granted Role	Grantor	Source
role1	superuser1	DIRECT
role2	test	role1
role3	test	role2

👤 expireduser1

👤 lockeduser1

👤 superuser1

👤 test

📁 - Roles

👤 role1

👤 role2

👤 role3

The identity means **user** (can log in) or **role** (cannot log in). Identities are categorized into catalogs. In each database, you can select a specific identity to check its privileges and its role inheritance. Moreover, we can easily determine if selected identity is a **superuser**, a **predefined identity**, **its account is locked**, **its password is expired**. A selected identity will be highlighted in red. Bold text indicates the superusers, and other icon marks the expired or locked identities.

Information: Only in Redshift there is an additional section for **groups**.

Global Privileges

Global Privileges

BYPASSRLS ✖
CREATEDB ✖
CREATEROLE ✖
INHERIT ✔
LOGIN ✔
REPLICATION ✖
SUPERUSER ✖

Global privileges, often referred to as **system privileges** in some databases, are permissions granted to users that apply at the system or database level, rather than being tied to specific objects like tables or schemas. They typically allow users to perform administrative or high-

level operations, such as creating objects, managing users, or connecting to the database, and may override more granular, object-specific permissions.

Schemas

Schemas

 +information_schema

 +pg_catalog

 +pg_toast

 +public

 - schema1

Security Tags:

Add tag ▾

- Privileges

CREATE ✖

USAGE ✔

+ Tables

+ Foreign tables

+ Views

+ Materialized views

+ Routines

+ Sequences

+ Types

+ Domains

Access and manage schemas. You can see and modify schema permissions of selected identity.

Information: Currently, only the **tables** and **views** are supported in every database type, and we are in process to extend this to cover all database objects that we can grant privileges to. The idea remains the same.

Tables

- Tables

- table1

Security Tags: Add tag 

- Privileges

DELETE 

INSERT 

REFERENCES 

SELECT 

TRIGGER 

TRUNCATE 

UPDATE 

+Columns

+ table2

+ table3

Access and manage tables. You can see and modify table permissions of selected identity.

Views

- Views

👁 + view1

👁 - view2

Security Tags:

Add tag ▾

- Privileges

DELETE ✓

INSERT ✖

SELECT ✖

TRIGGER ✖

UPDATE ✖

+ Columns

- Objects Used

📄 table1

📄 table2

Access and manage views. You can see which views and tables are used by selected view. You can see and modify view permissions of selected identity.

Columns

- Columns

A - col1

Security Tags:

Add tag ▾

- Privileges

INSERT ✓

REFERENCES ✖

SELECT ✓

UPDATE ✖

⬆ + col2

Access and manage columns. You can see and modify column permissions of selected identity.

Aggregated Privileges

- Privileges

DELETE	✗
INSERT	✓
REFERENCES	✗
SELECT	✓
TRIGGER	✗
TRUNCATE	✗
UPDATE	✗

Every object contains **Privileges** section, where we provide information about the aggregated privileges. You can manage them by clicking the ✓ or ✗ near the permission and then click the **Commit** button to save the updated permission.

Information: Privileges can come from different sources. Below is a list containing our API keywords and the corresponding queries needed to achieve each type of grant. **Warning:** It is **revoking** operation is possible only when the privilege is **granted directly** on the selected identity. The privilege that is not granted directly is in italics.

Newest data

By clicking the **Refresh** button you can fetch the newest data.

Audit Reports

By clicking the **Report** button you can **generate and download a report of the data source structure for the currently selected filters**. This report provides a JSON view of all the users and roles configured on the database, and all the access down to the column level the users and roles have. The report can be considered a form of credential discovery, specific for the database, and helps to normalize audits of different database types. The call to generate this report can be triggered with an automated API call, so can tie in with a periodic auditing system that spots changes to the permissions for a compliance team.

Each non-column element of the report has a **checksum** calculated from its direct components. Checksums do not include deeper nested subcomponents. They are also not guaranteed to remain the same for different report filters.

PostgreSQL, Greenplum, Redshift

Warnings

Our API shows aggregated privileges with source. Be aware of "hidden" privileges when inheritance occurs due to a lack of the USAGE privilege on the schema level.

Information

In PostgreSQL/Greenplum/Redshift access to objects in a schema requires:

1. Privileges on the object (e.g., SELECT, INSERT).
2. The USAGE privilege on the schema to make the object accessible.

If a role lacks USAGE on the schema, any privileges it has on objects in that schema remain hidden and unusable. These privileges become active if USAGE is granted or inherited.

Example:

1. Role A has a SELECT on schema1.table1 but no USAGE on schema1.
2. Role A cannot access schema1.table1 until USAGE on schema1 is granted.
3. If Role B inherits Role A and has USAGE on schema1, Role B gains access to schema1.table1.

Privilege Sources

• GRANTED DIRECTLY

- "GRANT *privilege* ON *object* TO *identity*"
- "ALTER ROLE *identity* *privilege*"
- "GRANT *privilege* ON SCHEMA *schema* TO *identity*"
- "GRANT *privilege* (*column*) ON TABLE *object* TO *identity*"

• PUBLIC ON OBJECT

- "GRANT *privilege* ON *object* TO public"

• INHERITED FROM IDENTITY ON OBJECT (includes nested identities)

- "GRANT *privilege* ON *object* TO *identity*" with "GRANT *role* TO *identity*"

- **SUPERUSER** (mean access to every resource)
 - "ALTER USER *identity* SUPERUSER" or "ALTER ROLE *identity* SUPERUSER"

MySQL

Warnings

The [PROXY](#) global privilege is not shown yet.

Privilege Sources

- **GRANTED DIRECTLY**
 - "GRANT *privilege (column)* ON *object* TO *identity*"
 - "GRANT *privilege* ON *object* TO *identity*"
 - "GRANT *privilege* ON *schema*.*** TO *identity*"
 - "GRANT *privilege* ON *.** TO *identity*"
- **INHERITED FROM OBJECT** (visible when viewing privileges on column level)
 - "GRANT *privilege* ON *object* TO *identity*"
- **INHERITED FROM SCHEMA** (visible from tables/columns in the schema)
 - "GRANT *privilege* ON *schema*.*** TO *identity*"
- **INHERITED FROM GLOBAL PRIVILEGE** (visible from any schema/object)
 - "GRANT *privilege* ON *.** TO *identity*"
- **INHERITED FROM IDENTITY ON COLUMN** (includes nested identities)
 - "GRANT *privilege (column)* ON *object* TO *role*" with "GRANT *role* TO *identity*"
- **INHERITED FROM IDENTITY ON OBJECT** (includes nested identities)
 - "GRANT *privilege* ON *object* TO *role*" with "GRANT *role* TO *identity*"
- **INHERITED FROM IDENTITY ON SCHEMA** (includes nested identities)
 - "GRANT *privilege* ON *schema*.*** TO *role*" with "GRANT *role* TO *identity*"

- **INHERITED FROM IDENTITY GLOBAL PRIVILEGE** (includes nested identities)

- "GRANT *privilege* ON *.* TO *role*" with "GRANT *role* TO *identity*"

Oracle

Privilege Sources

- **GRANTED DIRECTLY**

- "GRANT *privilege* ON *object* TO *identity*"
- "GRANT *privilege* (*column*) ON TABLE *object* TO *identity*"
- "GRANT *privilege* TO *identity*"

- **PUBLIC ON OBJECT**

- "GRANT *privilege* ON *object* TO public"

- **INHERITED FROM ROLE ON OBJECT** (includes nested roles)

- "GRANT *privilege* ON *object* TO *role*" with "GRANT *role* TO *identity*"

- **PUBLIC GLOBAL PRIVILEGE**

- "GRANT *privilege* TO public"

- **INHERITED FROM GLOBAL PRIVILEGE** (includes nested roles)

- "GRANT *privilege* TO *role*" with "GRANT *role* TO *identity*"

- **OWNER OF TABLE**

- "CREATE TABLE..." - if the identity created the table, he has all privileges to it

- **OWNER OF VIEW**

- "CREATE VIEW..." - if the identity created the view, he has all privileges to it

- **OWNER OF COLUMN**

- "CREATE TABLE/VIEW..." - if the identity created the table/view, means columns also, he has all privileges to it

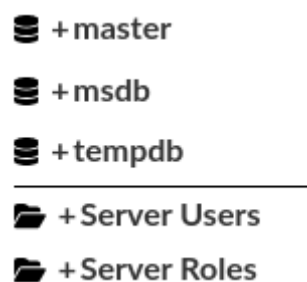
- **INHERITED FROM OBJECT**

- if the privilege was granted on the table/view level, means it is inherited on the column level also

SQLServer

Information

Under the database section, in SQLServer and Babelfish two additional catalogs are provided with Server Users and Server Roles.



Privilege Sources

- **GRANTED DIRECTLY**

- "GRANT *privilege (column)* ON OBJECT::*object* TO *identity*"
- "GRANT *privilege* ON *object* TO *identity*"
- "GRANT *privilege* ON SCHEMA::*schema* TO *identity*"
- "GRANT *privilege* TO *identity*"

- **PUBLIC ON COLUMN**

- "GRANT *privilege (column)* ON OBJECT::*object* TO public"

- **PUBLIC ON OBJECT**

- "GRANT *privilege* ON OBJECT::*object* TO public"

- **PUBLIC ON SCHEMA**

- "GRANT *privilege* ON SCHEMA::*schema* TO public"

- **PUBLIC GLOBAL PRIVILEGE**

- "GRANT *privilege* TO public"

- **OWNER OF TABLE**

- "ALTER AUTHORIZATION ON OBJECT::*object* TO *identity*"

- **OWNER OF SCHEMA**

- "ALTER AUTHORIZATION ON SCHEMA::*schema* TO *identity*"

- **PUBLIC OWNER OF TABLE**

- "ALTER AUTHORIZATION ON OBJECT::*object* TO public"

- **PUBLIC OWNER OF SCHEMA**

- "ALTER AUTHORIZATION ON SCHEMA::*schema* TO public"

- **INHERITED FROM ROLE OWNER OF TABLE** (includes nested roles)

- "ALTER AUTHORIZATION ON OBJECT::*object* TO *role*" with "ALTER ROLE *role* ADD MEMBER *identity*"

- **INHERITED FROM ROLE OWNER OF SCHEMA** (includes nested roles)

- "ALTER AUTHORIZATION ON SCHEMA::*schema* TO *role*" with "ALTER ROLE *role* ADD MEMBER *identity*"

- **INHERITED FROM ROLE ON COLUMN** (includes nested roles)

- "GRANT *privilege (column)* ON OBJECT::*object* TO *role*" with "ALTER ROLE *role* ADD MEMBER *identity*"

- **INHERITED FROM ROLE ON OBJECT** (includes nested roles)

- "GRANT *privilege* ON OBJECT::*object* TO *role*" with "ALTER ROLE *role* ADD MEMBER *identity*"

- **INHERITED FROM ROLE ON SCHEMA** (includes nested roles)

- "GRANT *privilege* ON SCHEMA::*schema* TO *role*" with "ALTER ROLE *role* ADD MEMBER *identity*"

- **INHERITED FROM ROLE GLOBAL PRIVILEGE** (includes nested roles)

- "GRANT *privilege* TO *role*" with "ALTER ROLE *role* ADD MEMBER *identity*"

- **INHERITED FROM OBJECT** (visible when viewing privileges on column level)

- "GRANT *privilege* ON *object* TO *identity*"

- **INHERITED FROM SCHEMA** (visible from tables/columns in the schema)

- "GRANT *privilege* ON SCHEMA::*schema* TO *identity*"

- **INHERITED FROM GLOBAL PRIVILEGE** (visible from any schema/object)

- "GRANT *privilege* TO *identity*"

- **INHERITED FROM CONTROL ON OBJECT** (visible on privileges on column and object level)

- "GRANT CONTROL ON *object* TO *identity*"

- **INHERITED FROM CONTROL ON SCHEMA** (visible on privileges on column, object and schema level)

- "GRANT CONTROL ON SCHEMA::*schema* TO *identity*"

- **INHERITED FROM GLOBAL CONTROL ON DATABASE** (visible on privileges on column, object, schema and database level)
 - "GRANT CONTROL TO *identity*"
- **INHERITED FROM GLOBAL CONTROL ON SERVER** (visible on any privilege except CONTROL SERVER)
 - "GRANT CONTROL SERVER TO *serverIdentity*"
- **INHERITED FROM OTHER GLOBAL PRIVILEGE** (visible on privileges granted by other privileges)
 - "GRANT *privilege* TO *serverIdentity/identity*"
- **INHERITED FROM SPECIAL DATABASE ROLE** (visible on privileges granted by special database roles)
 - "ALTER ROLE *specialRole* ADD MEMBER *identity*"
- **INHERITED FROM SPECIAL SERVER ROLE** (visible on privileges granted by special server roles)
 - "ALTER SERVER ROLE *specialRole* ADD MEMBER *serverIdentity*"
- **INHERITED FROM SYSADMIN** (visible on schema, object and column level when login is member of sysadmin role)
 - "ALTER SERVER ROLE sysadmin ADD MEMBER *serverIdentity*"

Logging Configuration

Logging Design

The system is designed to allow the proxies/drivers to generate a variety of records to detail the events that are occurring as traffic is processed. These can include:

- SQL statement execution
- Connections opening and closing
- JDBC Method invocation
- Exceptions
- Console/Debug logs

As many of these categories can result in a very large amount of traffic, a variety of mechanisms are provided to control the volume of logs. Here is where each is controlled:

- **Connection open and close:** Only in the VDB tab, under logging
- **JDBC Method invocation:** In the VDB tab, under Debugging
- **SQL Statement Execution:** As an all on/off toggle under the VDB tab, under Logging, OR; via Log rules, with selective throttle controls
- **Exceptions:** Always generated
- **Console Logs:** These are always generated, but where they are sent can be controlled with the "Aggregate Console Logs" option under Logging in the VDB
- **Debug Logs:** These output as console logs, but control if they are generated is under the VDB Debugging tab with the "Verbose Debug Mode" option.

When using a log rule to perform SQL statement logging, there are a variety of options that can be used to limit the rate of logs. These are documented in the rules configuration documentation under "Rate Limiting" and can provide for a maximum log entries per second, or a ratio of logs processed vs. not processed. The rate limiting occurs at the proxy/driver level,

and by reducing the rate of logging, this also reduces the amount of processing on the node generating the logs, so is highly recommended in production environments to limit the impact.

Global Log Management

In order to control the total size of the logs globally, there are two options available in the admin->Server properties section:

- **Max Log Age:** This is the number of days that log files and database log entries are allowed to persist. This defaults to 14 days.
- **Reserved Disk Space:** This sets how much space must be free at all times. During log rolling, SQL logs will be deleted until this value is achieved.

When cleaning up disk space, logs are deleted first based on the Max Log Age parameter, then based on the Reserved Disk Space value.

By default, logs are written to `{workDir}/heimdall/log`, however this can be configured. By adding into the `javaOpts` the property `"heimdall.logdir"`, the relative directory can be controlled, i.e. `"-Dheimdall.logdir=./log2/"` will set the directory to use `log2` for the flat file logs.

Analytics Requirements

When analytics are to be used, we log data into an internal database on the management server, which summarizes the data in a concise format necessary for fast performance and minimal space and overhead. The minimum necessary to perform this logging is either:

- Enable the "Log All SQL" option in the VDB Logging section, or;
- Attach an enabled "Log" rule that matches the desired content to be logged.

Flat-file Logging

In order to provide detailed logging, an additional option is available in the VDB tab, to "Write logs to file", which will trigger all query logs for the VDB to not only be registered to our internal database, but to also be logged individually in a CSV formatted file in the `$serverhome/log` directory, and follow the following format (CSV, and can be opened in a spreadsheet if desired):

- **epoch-ms:** Log timestamp (ms since epoch)
- **type:** Event type
 - **TRACE:** Generated with trace actions
 - **DONE_SQL:** Generated with log actions

- **CONNECTION_OPEN**: Generated if log connections set
- **CONNECTION_CLOSE**: Generated if log connections set
- **METHOD_COMPLETE**: Generated if log methods set
- **TRANSACTION_START**: Generated if log methods set
- **TRANSACTION_END**: Generated if log methods set
- **DEBUG**: Provides additional logs generated in rare cases
- **SQL_EXCEPTION**: Generated when SQLException is thrown
- **INTERNAL_EXCEPTION**: Generated when any other exception is thrown
- **NEW_PATTERN**: Triggered when observing a new query pattern
- **NOTIFY**: For rules tagged with a notify
- **CONSOLELOG**: When console log aggregation is enabled, this provides the log data
- **NOCACHE**: Generated when cache revalidation fails
- **PROXY_START**: Generated when proxy is started or restarted
- **AUTHENTICATION**: Generated when user tries to authenticate
- **ERROR**: Generated when something unexpected or unwanted happened, but no exception was thrown
- **SYSTEM_STATUS**: Generated for Heap Dumps and Stack Traces
- **METRICS_DISCOVERY**: Generated for JMX statistics propagation between central manager and proxy
- For additional info about event types please visit [Monitor->Logs Overview](#).
- **flags**: Status flag
 - **0**: Cache miss
 - **1**: Cache hit
 - **2**: Cache miss (in transaction)
 - **3**: Cache hit (in transaction)
 - **5**: Forwarded
 - **6**: Forwarded (in transaction)
- **User**: JDBC Username used to connect to the db
- **vdb**: VDB name
- **hostname**: Hostname of the source
- **instance-id**: A unique ID for each JVM, to provide uniqueness even on the same host
- **conn-id**: Session id (uniquely assigned from server to identify different users)

- **class:** Log entry source method type:
 - **C:** connection
 - **S:** statement
 - **R:** resultSet
 - **P:** prepared
 - **X:** callable
 - **D:** Driver level
 - **B:** Data Source
 - **U:** Unspecified type
- **obj-id:** Object id (semi-unique identifier for given object in this session)
- **parent-id:** Parent id (id of parent object. E.g. If ResultSet, parent id would be id of statement that created it)
- **runtime-us:** Runtime in microseconds i.e. response time of operation generally. For connection close events is total time connection was open, and for TRANSACTION_END, it is the time since the last TRANSACTION_END or the TRANSACTION_START.
- **db-name:** Data source (may be the name of the LB source)
- **Message:** Event text. For both the log (done_sql) and trace (trace) event types, all of the following fields will be inserted, but for the done_sql, many will be empty values, as the data is not tracked for this type to lower overhead. Other record types include text that represents what was done, i.e. the method, etc.
 - **rows:** number of rows
 - **rows-read:** number of rows not read
 - **result-size:** Estimate of raw byte size of Result
 - **java-size:** Estimate of Java byte size of Result (including object overhead)
 - **proctime-ms:** Overall time (all rows processing time)
 - **result-hash:** Pattern hash used by Heimdall, to associate similar queries
 - **text:** text field based on the record type--this can change between releases fairly often.

SQL statement & Parameters The SQL statement is logged in the format of "Statement~num parameters~param1~param2~". Each parameter is replaced with a ? in the statement. The target is to simplify the parsing of query parameters, yet insure that if log records are sorted based on this portion of the message, than unique queries will be sorted together. This format also makes stripping potentially sensitive information from the records when needed.

Note: The log fields may change between major revisions of code. Each file contains a header that provides the position of each field, and should be used for parsing of the file to insure that additional or rearranged fields do not break any code depending on the format of the log file.

Example Log Entries

#Fields: epoch-ms, type, flags, user, vdb, hostname, instance-id, conn-id, class, obj-id, parent-id, runtime-us, db-name, rows, rows-read, result-size, java-size, proctime-ms, user, result-hash, text

```
1523556866884, TRACE, 0, magento, Magento-Demo-vdb, ip-172-31-56-200,
94f7f042-7bd5-41a5-bb49-091877788fde, 0, R, 469576, 11812, 1616, Magento-Demo-source-
Master,
1,1,65,425,1740,4BC71F67001332175D27AFF533095BB2,magento,magento.test,SHOW TABLE
STATUS LIKE ?~1~\'"core_resource\'"~,
```

```
1523556866892, TRACE, 0, magento, Magento-Demo-vdb, ip-172-31-56-200,
94f7f042-7bd5-41a5-bb49-091877788fde, 0, R, 469577, 11812, 7420, Magento-Demo-source-
Reader,
68,68,1925,6005,7584,EBF219E0CA457EC7C9A9255C33A25908,magento,magento.core_reso
urce;magento.test,SELECT core_resource.* FROM core_resource ~0~,
```

```
1523556866894, TRACE, 0, magento, Magento-Demo-vdb, ip-172-31-56-200,
94f7f042-7bd5-41a5-bb49-091877788fde, 0, R, 469578, 11812, 803, Magento-Demo-source-
Reader,
9,9,177,357,862,1AF8150173C7C1B330EE8434132A65FB,magento,magento.test;magento.ind
ex_process,SELECT index_process.indexer_code FROM index_process ~0~,
```

Caching Architecture

Heimdall operates via a two-layer cache system. The first layer (L1) is an in-heap cache, the size of which is controlled by the cache size in the VDB settings. The second layer (L2) may either not exist (local cache only), or will leverage any one of the supported cache engines, to provide a distributed cache between nodes.

For caching to be active, the following conditions must apply:

- The VDB defines a base cache configuration
- A rule-set is attached to the vdb
- The rule-set defines one or more cache rules (please see the rules configuration section for the list of cache options)
- the query meets some minimal qualification for caching based on built-in logic, including size limits, etc.

Caching Logic Flow

The (simplified) order of logic processing is as follows:

- Request processing
 - Parse query for tables (this can be displayed with the printTables property on a rule)
 - Rule processing:
 - if (table property) Add to the table list for the query
 - if (cache) set cache ttl to query
 - if (nocache) unset any cache TTL and flag to not cache at all
 - After processing the rule, if a cache TTL is above 0 and caching is set
 - Create a 64 bit hash of the query and all parameters
 - Perform lookup on L1 cache with the hash
 - If no L1 cache hit, then check if L2 cache has the object
 - If a result is returned from the L1 or L2 (grid) cache
 - Check if the result has a creation time later than the last write time for any associated tables, if so delete object from the cache
 - Return the result to the client

- If no result has been returned, retrieve from the database
- If the query is not cacheable, check if it is an update
 - If an update, then trigger an invalidation message into the L2 cache pub/sub interface for associated tables
 - After pub/sub message is transmitted, forward the query to the database
- On result processing (from the database)
 - If the query was flagged with a TTL, create a cache object
 - If in proxy mode, return result to the proxy module to serialize
 - If in JDBC mode, attach result to the cache object
 - Trigger the log result
 - After the result-set is closed at the application level
 - queue the result for an ingestion thread
 - Place the cache object (and all metadata) into the L1 cache
 - If necessary, serialize the result for L2 caching and push into the L2 cache

Caching has two fundamental modes of operation, depending on if Heimdall is in proxy mode or operating as a JDBC driver. In proxy mode, the cached object will be the serialized version of the result-set that is pushed on the wire, in order to reduce the overhead of generating a response for following cache hits. The serialized data is also stored in the L2 cache in this same form, as the result is sent to the client the first time, this reduces the overhead of interacting with all layers of the cache. This process also accounts for variables such as character set that may result in different resultset representations depending on the state of the connection to the database.

When in JDBC mode, the local L1 cache stores result-sets in Java object form on the local heap, in order to provide the results in the form the application will use them. When storing into the L2 cache, the results will be serialized into a format that is compatible with all cache engines supported, and only pulled and de-serialized when necessary.

Caching will trigger the query to generate a 64 bit hash of the query, and lookup in the cache index if there is a matching response. In the event there is, request processing is terminated, and the response is processed to the caller. On a cache miss, the processing continues, and on the response side of processing, the response will be stored in the cache store for later use, with a time to live based on the cache rule setting.

When an object is cached, the query that generated it will be hashed to a 64 bit query hash, including all parameters necessary to generate uniqueness (by default vdb, catalog, user). The object will then be pushed into the L1 and L2 cache, if present. Our cache logic is designed to do cache key synchronization as well, so any object pushed into the L2 cache will be visible to

all other nodes. The benefit to this is that we will rarely if ever have a cache miss against the L2 cache. As the L2 caches are generally on remote nodes, this minimizes the overhead of caching, and prevents a double round-trip on a cache miss.

In the event that due to a previous cache configuration, an object was cached that would now no longer be cached, due to the order of operations, the old “stale” object will be thrown out after being pulled from the cache. This applies to table exclusion rules, or TTL changes. As such, unlike most systems, a change to the cache rules will not require a purge of the cache to prevent invalid data from being returned.

Each cache object is tagged with the tables the query was associated with, and internally, all tables are tracked with a last modified time, in milliseconds. On a cache lookup, if the object was created prior to the last modified time of any table it is associated with, then the object is flushed from the cache, and repopulated. This helps insure that even if cache policies are changing, the current policies ttl limit will be honored, even if the policy is created or changed after the object enters the cache. Last modified data can come from any one of several sources:

- A DML was observed on the table directly by the driver
- An invalidation record with a newer last modified time was pulled from the grid cache
- A last modified time was pulled from the database via trigger notification
- The first two of these don't require any additional configuration. The third requires some configuration to setup and configure the database to support this. Currently, sample trigger configurations are available for MySQL and MS-SQL, and can be developed as needed for other databases. Please contact support for assistance with this feature.

Table Caching provides a refinement to any cache policy in order to override a TTL of 0, or reduce the cache TTL on a match. In the case of a cache value of TTL, table cache entries may override the base TTL if and only if all tables have a positive ttl specified. If a table cache entry specifies a ttl of zero, and the table is present in a query, the query will not be cached. Regex patterns do not apply on a table cache, nor are they ignored, even if the “ignore” action is used.

Stored Procedures, Views and Triggers

When using any query that can't be directly parsed, it is possible to still properly enable caching and invalidation. There are a few critical pieces that need to be configured however:

- When tables being written to can't be parsed, add a “table” property, then the fully qualified name of the table, i.e. `magento.log_table`, and set the property “update” to true, if necessary. On a write, only the table(s) being written to need to be associated with the query, as any reads will be done server-side, and will not come from the cache.

- When tables are being read that can't be parsed, like with a write, use the "table" property to specify the fully qualified table names being read from, and set the property "update" to false. This will insure that writes against the read tables will invalidate the queries being read from.

Example:

Edit Rule

Regex:

(?i)^CALL spName

+

In-Transaction:

☒

Action:

Cache

▼

Notes:

Example to cache a store procedure

Parameters

+

ttl

5

m

▼

tables

▼

database.tableName

×

update

▼

true

▼

×

Done

In the case of views, simply use the `tables:{tablename}` syntax to match the view name, then attach the tables that are associated with it. Nested tables can be handled with this as well, as long as the order is correct, i.e. a view using another view is defined before the view it uses. This way, you attach each view's tables to a growing list of tables.

Routing Read-Only Stored Procedures to Replicas

By default, the Heimdall routes all MySQL stored procedure (`CALL spName`) statements to the primary (writer) node to ensure data integrity – even if the procedures contain only read operations. However, you can optimize performance by explicitly routing read-only stored procedures to replicas using **Heimdall Query Rules**.

To route a read-only procedure to a replica, create a **Heimdall Query Rule** with the following configuration:

- **Action:** `Reader Eligible`
- **Regex:** Match the exact stored procedure call (e.g., `(?i)^CALL spName`)
- **tables:** Specify the table(s) read by the procedure (e.g., `schema.tableName`)

- **update:** `false` (indicates no data modification)
- **unconditional:** `true` (forces the rule to apply regardless of internal conditions)
- **lagignore:** `false` (ensures replication lag is considered)

Example Configuration

```
Regex: (?i)^CALL spName
Action: Reader Eligible
tables: schema.tableName
update: false
unconditional: true
lagignore: false
```

The `unconditional=true` parameter is useful when Heimdall cannot clearly understand or analyze complex stored procedures. By setting this parameter, you instruct Heimdall to rely on your rule configuration explicitly, enabling safe routing of queries to read replicas.

Below is a screenshot of a configured Query Rule that routes a read-only stored procedure (`CALL spName`) to a read replica using `unconditional=true`

Edit Rule

Regex:

(?i)(CALL spName)

+

In-Transaction:

☐

Action:

Reader Eligible

▼

Notes:

Applies routing to replica for CALL spName

Parameters

+

lagignore		false	▼	
unconditional	▼	true	▼	×
update	▼	false	▼	×
tables	▼	schema.tableName		×

Done

Repopulation

When the heimdall proxy or driver is restarted, if connected to an L2 cache, it will request all the current keys from the L2 cache. This will be used to determine what objects (as they are accessed) can be pulled from the l2 cache layer. If not present, then repopulation will be done

from the database itself. No cache data is stored on the disk of the proxy nodes by Heimdall, and all cache objects can be considered "in-flight" or transient.

Invalidation

When a DML operation is detected by Heimdall, and a cache engine is configured, it will use the pub-sub interface of the cache engine in order to exchange information on what tables were written to and when. This is shared across nodes, so that each node knows when other nodes perform a write. In addition, some additional logic is used to reduce the number of invalidation messages:

- If less than 95% of queries are flagged as reads, then a table is deemed non-cachable, and any writes to the table should also trigger a "super-invalidation" to other nodes
- If more than 100 invalidations a second are triggered, then trigger a "super-invalidation" on tables being written to as well

A super-invalidation is a message that when sent to all the other nodes triggers non-cachability of the referenced table for two seconds.

To override invalidation of a table as a result of a DML, the property "invalidate" can be set to false to prevent any eviction action. This will not prevent caching of the object.

To explicitly control if a particular query is treated as an update, the property "update" can be set to true. This will further trigger invalidations against associated tables if parsed.

One final option to control invalidation is to use the "olderthan" property, which can be set to a negative value. In this case, invalidations last into the future, so a write against a table may prevent cached results for a period of time after the write.

Configuring Trigger Based Invalidation

In order to support out-of-band data ingestion into the database, a data source can be configured for "Trigger based invalidation". There are several options available on the data source for this configuration:

- dbTriggerTracking: set to true if this feature is to be enabled
- dbTimeMs: A query to retrieve the current time in MS on the database, to insure timestamps are synchronized
- dbChanges: A query to retrieve a timestamp (as a long) and table name (as a string), for the last invalidation time of that table.

The dbTimeMs call for example defaults to the value of "SELECT heimdall.get_time_ms()", usable on MySQL without any changes. The dbChanges call defaults to {CALL

heimdall.changes(@ts)). This can be a stored procedure. The string "@ts" will be replaced automatically with the last modification (per the DB time) that has been observed, so can be used to pull only the tables that have been modified since the last invalidation observed. The table name returned needs to be fully qualified, as observed with the "printTables" option for the rules, as this is the table name we will be comparing for invalidation purposes.

Note 1: As the trigger table queries are performed as part of LB health checking, the load balancing feature must be active to use this feature Note 2: Please see the database specific documentation for examples on how to configure the trigger based invalidation on the database side

API Based Invalidation

When using the healthcheck-port option (see VDB Properties), an additional feature is enabled, that of HTTP based eviction. To use this, a call can be made on the healthcheck port such as "/invalidate?vdb=&table=". This will trigger the proxy to process an invalidation as if a write had been detected to the table. In a distributed cache scenario, this is propagated to other nodes as well. Debug logging will provide additional verification that the invalidation has occurred. The table name of "all" can be used to clear all tables associated with the VDB on a given proxy node (this is not propagated across nodes however). Table names can also be regular expressions. When a table is invalidated via regular expressions, the tablename will be returned as the response body in a CSV list. Note: If a table has never been invalidated, it will be considered to have a last write time of 0 implicitly, so won't need to be invalidated, and will not show up in the list of tables invalidated.

Healthcheck-port can also be secured with an authorization token if Enable Token Authorization is checked. A field with a generated token will appear where the token can be changed manually or randomly generated by using the "generate new token" button. If a user wants to perform an HTTP call with enabled authorization they must add a token to request parameters.

Example debug logs when using this feature:

```
curl "http://localhost:8081/invalidate?vdb=Vevo-demo-vdb&table=test&token=securityToken"
```

```
[2018-12-04 19:28:49,060] 2018-12-04 19:28:49,060 308879 [INFO]
[nioEventLoopGroup-9-1] Cache: Invalidated: test vdb: Vevo-demo-vdb time:
1543951729070
[2018-12-04 19:28:49,060] 2018-12-04 19:28:49,060 308879 [INFO]
[nioEventLoopGroup-9-1] Issuing invalidation on table: test at: 1543951729070
```


Note: Always make sure to specify the table name as it shows as a result of the printTables command or from the expanded query view in the analytics tab, otherwise the invalidation will not be successful.

Auto-Refresh

In many situations, particularly for Analytical and Dashboarding workloads, a small but repetitive set of queries are issued against a database that is only periodically refreshed. The queries against the data-set may be very expensive and time consuming, and if issued only when someone wants to view a report, may result in a long lag before the report is rendered. This feature is for this use-case.

The way auto-refresh works is that first, heimdall will cache a query, typically for an extended TTL, be it one day or longer. Next, when a data load is done, the table the invalidation is cached against. This invalidation will trigger Heimdall to inspect all the last X queries it has processed, and will re-execute the queries, in order to re-populate the cache with the fresh data. This refresh will be done from most frequently used queries to least frequently used in order to make data that is most likely to be needed available as soon as possible.

The net result of this is that immediately after a data load and invalidation, the cache is refreshed, and reports can be rendered quickly when needed without queries being processed on the database.

To enable auto-refresh, in the VDB, set the vdb property "trackQueryDistributionCount" to the number of the most recent queries to track (say 10,000) and then in your cache rules you can set the property autoRefresh. The following conditions must be met for a refresh to occur:

1. The query tracker must have at least two instances of the query being issued. One is not sufficient. If the query tracker is under-sized, then this will impact refresh capabilities.
2. The query must be in the cache at the time of the invalidation;
3. One or more tables associated with the cached query must be invalidate.

Queries will be refreshed based on the most frequently accessed query to least frequently, and refresh queries themselves will not be tracked by the query tracker.

Note: Auto-refresh is not yet functional with the SQL Server proxy

Debugging

One of the most common issues with caching is that the tables extracted from the query or otherwise assigned to a query via a rule do not match between queries and updates. There are a variety of reasons why this can be, including the use of the Postgres search path to search multiple schemas, stored procedures, or invisible updates due to triggers. The easiest way to

diagnose this is to create rules that match the update query and the read query, then add the "printTables" property, with the value of true. This will result in log entries such as:

```
[2018-12-06 15:17:43,924] ip-172-31-14-81.ec2.internal: Query: SELECT ? FROM  
ir_module_module WHERE name=? AND latest_version=?~3~\1\~\~base\~\~11.0.1.3\~  
tables: [odoo.public.ir_module_module] isUpdate: false
```

The table name(s) listed in the tables field must match exactly for invalidation to operate properly.

Additionally, when objects are configured for caching, but are not eligible for caching for some reason, in general, the debug logs will provide additional information on why this is the case. One case that comes up with SQL Server stored procedures often is that they are returning more than one result set. This case is not currently supported by Heimdall.

In order to debug the auto-refresh tracking, the command "heimdall querytracker" can be issued. A table name can be issued following this to narrow down the queries tracked against a particular table.

Load balancing and High Availability

Heimdall provides a multi-tiered approach to supporting High Availability that supports a wider variety of databases vendors and database topologies than any other product on the market. This includes failover logic at the JDBC driver itself, and coordinated failover via the central manager. With it, the following aspects of database behavior can be accounted for and made highly reliable, while accounting for the limits that may be inherent in any database synchronization technology:

Supported Modes

- True active+active load balancing
- One or more write-only masters
- One or more read-only slaves
- Hot-standby servers
- Cold-standby servers
- Control of pre-emption on restoration of services
- Scripted orchestration on failures with multiphase configuration commits
- Automatic disaster recovery activation
- Custom query routing

How to account for each of these will be described, and a variety of configuration scenario examples explained in detail.

Application Side Requirements

Important when using Heimdall in an environment where the proxies are themselves load balanced, a `maxAge` or similar (`maxUsage` for Node.js) option should be set so that connections do not live forever. If not set, this can result in one or a few proxy nodes being overwhelmed with traffic while others are not, preventing autoscaling.

Basic Theory

While simple at a high level, high availability of database servers is an extremely complex topic, and the interaction that Heimdall has with the databases is as such also complex topic. Below

are a few questions that need to be asked both in setting up the database cluster, and the answers are important in evaluating what should be done with the Heimdall configuration:

1. Should the infrastructure have a shared-nothing approach to the data processing of queries? This implies that all disks and disk infrastructure should be redundant. Solutions such as Oracle RAC leverage a shared-disk infrastructure in order to simplify the task of managing data, when load balancing of the database nodes is performed. While an excellent solution for many environments, this infrastructure is very expensive, and still leaves the risk of a storage corruption resulting in a complete outage.
2. If not sharing the disk infrastructure, will synchronous or asynchronous replication be performed between nodes?
 - a. Synchronous replication indicates that between two or more nodes, all nodes must agree that a commit can be performed, and that it completed before a response is provided to the client. This results in a significant overhead in writes, but guarantees that database nodes are in sync, and allows a more flexible active-active approach to load balancing of write traffic. For the purposes of configuration, a shared disk infrastructure will be considered the same as synchronous replication.
 - b. Asynchronous replication is the method of executing a complete transaction on a single server, then replicating the changes to a second server. In most cases, this works well, but can have unfortunate side-effects.
 - c. Will a single write-only node be acceptable, with reads spread across nodes, or will all activity take place on a single node, with a second node acting as a pure standby node?
3. Will the replication be unidirectional or bidirectional, i.e. will all changes be pushed from one node to one or more, or will all changes on all nodes be pushed to all other nodes.

An excellent article on the issues with Asynchronous replication and load balancing can be found at: <http://datacharmer.blogspot.com/2013/03/multi-master-data-conflicts-part-1.html>, and covers much of what is necessary to help answer these questions and an overview of the issues related with them.

Once the question of if the server is doing synchronous replication or asynchronous replication and unidirectional vs. bidirectional replication is answered, then the Heimdall configuration can be designed.

Basic Configuration

List of options for this section can be found in [load balancing options](#)

- **enable load balancing** First, for load balancing to operate, this option must be selected. If not, then the only server Heimdall will connect to will be the primary JDBC URL. If load

balancing is enabled however, then the jdbc URL specified in the top section of the configuration will be ignored, although used as the initially configured server in the load balancing configuration.

- **track cluster changes** This option, enables the cluster detection and tracking logic to become active with Heimdall. The purpose of this is to allow Heimdall to reconfigure itself based on the cluster's configuration as it may change. This works for the following cluster types:
 - Extra information about target write/read capacity: [Cluster role decision-making](#)
 - MySQL (stock) Master/slave replication
 - Galera Cluster Master/Master replication
 - MySQL Galera Hybrid topology (Synchronous Galera multi-master + MySQL asynchronous replicas)
 - Postgres Master/Read-slave replication (not PG 10 logical replication)
 - SQL Server Always-on availability groups
 - SQL Server Mirroring

Galera Cluster Detection – Configuration Note

For MySQL 8.0 + Galera deployments, Heimdall uses the following SQL statement to detect active nodes in the cluster:

```
SHOW GLOBAL STATUS LIKE 'wsrep_incoming_addresses';
```

When the Galera configuration parameter `wsrep_node_incoming_address = AUTO` is used (the default), the value returned by `wsrep_incoming_addresses` reports only the auto-detected local node address. It does **not** provide the full cluster membership. As a result, Heimdall cannot reliably detect all nodes when this parameter is left at its default.

To ensure accurate cluster discovery, each Galera node should explicitly set:

`wsrep_node_incoming_address = <node_ip>:<port>` This reflects a limitation in the cluster metadata exposed by Galera. No SQL-level status variable or performance schema table provides a complete or real-time list of nodes suitable for automated detection. Therefore, explicit configuration of `wsrep_node_incoming_address` is required when using the **track cluster changes** option with Galera clusters.

MySQL Galera Hybrid topology (Synchronous Galera multi-master + MySQL asynchronous replicas)

Heimdall supports hybrid MySQL topologies where a Galera synchronous multi-master cluster is combined with MySQL native asynchronous replicas for read scale-out.

In an RDS environment, to support global RDS clusters, the AWS RDS ARN can be filled in, and will then be used to discover the entire global cluster.

- **use response metrics** option is used in situations where the readers may be distributed, either in multiple availability zones, or even globally distributed. This option will result in each proxy using the response time metrics from health checks to determine the closest or best performing reader, then filtering out any readers that are not within 25% of the best reader's response time. In most cases, this is sufficient enough so that only the readers in the same availability zone will be selected to be read from, although some combinations of AZ's may be paired together for reading due to close proximity.

- **Response Metric Multiplier** How much slower a reader can be vs. the fastest reader and still be eligible for reads

Track Replication Lag option is described below, as is the Lag window Buffer in [Replication Lag](#).

Failover Script Extra information in section [State Change Scripting](#)

When enabled, Heimdall will on a failure event and every 30 seconds check the state of the database cluster, and will reconfigure itself as necessary based on the cluster configuration, allowing for automatic role selection. Please see below for the rules in determining what server takes what role (when it is not deterministic) via the target read and write capacity.

The node configurations contain the following:

- **NAME:** A simple name used internally for tracking the individual server, and may be used in various logging operations. When cluster tracking is used, the name will indicate the role and ID of the server, and will be auto-generated.
- **URL:** The URL is the JDBC URL that will be used to access this data source. Note that when load balancing is configured, this JDBC URL overrides the normal JDBC URL. In the case a database server has multiple ports, each can be used as an independent database in this configuration.
- **Enabled:** The Enabled flag specifies if the server should be used or not. If a configuration change is made from enabled to not enabled, then all active connections will be migrated away from the server as soon as transactions on each connection are completed. Enabled (even if not used) also means that the server will be monitored with internal monitoring, so will allow state change scripts to operate properly. Internal to the driver is another value similar to Enabled, called "Active". This may be observed in the logs during failures, with the state being changed automatically by the driver. This value represents if the server is considered healthy or not, i.e. it is passing health checks. This value is managed by the driver itself and can't be set.

- **Writeable:** Specifies if the server should be considered usable for queries that are flagged as writes. If a server is not writable, then if and only if a query matches a “reader eligible” rule or specially flagged forwarded request will the server be used, i.e. read/write split.
- **Weight:** Represents what share of connections each server should receive, as a random share based on the weight. If one server was configured with a weight of 1, and another as a weight of 2, then the second will get 2x as many connections as server 1. This is determined as a random weighted assignment—it does not account for the number of connections that a server already has, or the traffic load, and with a small number of connections may not be accurately represented. In the case where some servers are read-only, the share allocation will only be computed including the read-only servers if the `setReadOnly()` method is used on a connection. The act of using this method will force a reconnect based on the new weights, even if it was previously connected to another server.

A weight of 0 is considered a special case—it is used to designate a “server of last resort” for a data source, and allows a driver to perform an independent failover without coordinating with the central server, in configurations that allows this behavior. In the case that several servers have a weight of 0, then all servers with a weight of zero will have traffic balanced to them during the failure in a random manner.

When a server of last resort is used, it is **ONLY** used until at least one server with a weight greater than 0 comes back online. At that point, as connections complete transactions (or immediately if they are idle), the JDBC connections will again revert back to using the weighted server or servers, and the weights will again be used to compute the share they allocate.

Connection Hold Time: If no server of last resort is available, and all valid servers for a connection are in a non-active state, then the overall load balancing property of Connection Hold Time will take effect. This value is specified in milliseconds, and represents how long we should wait until one of two events occurs:

1. A server becomes active again;
2. The data source configuration is changed via the central manager.

When a configuration change takes place or a server state change, all waiting threads will be woken, and will then check to see if a new connection can be made. If not, they will again go to sleep until either of these events occur again until the hold-time has been passed, at which time an exception will be passed to the calling application.

Cluster Role Decision-Making

The target read and write capacity is used to help select what server should be used for what role, in particular in the condition of a Galera cluster, as any node in theory can be a write node, but usually, it is desirable to only have one node act as the write node, one or more nodes as

read nodes, and other nodes may be hot-standby or used for maintenance or reporting activities. The logic works as follows:

For each node, a write and read capacity will be associated. The online nodes will be ordered for write consideration based on the highest write weight, then the next, etc. The nodes will then be selected for a write role until the **target write capacity** is achieved. These nodes will also be used as read nodes unconditionally.

For the read roles, IF the write nodes don't satisfy the desired read weight, then again, the read weights will be ordered from highest to lowest, and additional read-only nodes selected for reading until the **target read capacity** (including the write nodes) has been achieved.

If any nodes are not selected as write or read nodes, then they will be flagged as disabled, and will not be used by Heimdall until after a failure happens, at which point they will be evaluated for their role again.

The goal of this is to allow a customer to decide which servers in a cluster fill what role, based on their desires.

Database Tables for Cluster Roles

In order to provide greater flexibility in managing database clusters, for any system that supports more than two nodes (MySQL Replication, Galera, Postgres, & SQL Server), when cluster tracking is enabled, a small table will be configured on the database in the Heimdall database/schema called "servers_info" defined as:

```
CREATE TABLE heimdall.servers_info (  
    private_id VARCHAR(255),  
    public_id VARCHAR(255),  
    read_weight NUMERIC(2),  
    write_weight NUMERIC(2),  
    enabled NUMERIC(1),  
    PRIMARY KEY(private_id) );
```

This will be created on the initial setup of the cluster by Heimdall, and provides an alternative control point to manage the values as they match in the GUI. In addition, this allows you to specify a private Id vs. Public ID, in order to provide mapping due to NAT. As Heimdall polls the cluster for the IP addresses of the nodes in the cluster, Heimdall will often receive a private IP that can't be directly connected to. By setting this private ID in the "private_id" field, and mapping it to the appropriate public IP or name, this allows the cluster logic to account for the actual topology. The read and write weights can also be adjusted, and if a node is to be brought offline, setting the enabled field to "0" will disable the node.

These settings, if adjusted, should take effect in 30 seconds, and will be reflected on the GUI once it is refreshed. Likewise, when used, a change on the GUI will be immediately reflected

into the database. This provides a control point for a DB doing maintenance to take offline nodes without even logging into the Heimdall console.

Note: This table is not currently used with AWS Aurora due to the way cluster detection is done.

Note: To leverage this table with cluster tracking, set `hdUseServerInfo=true` in the connection properties for the datasource. When cluster tracking is enabled and this property is set, Heimdall will populate and honor the `heimdall.servers_info` table as described above, allowing node enable/disable and weight changes (`enabled`, `read_weight`, `write_weight`) to be controlled at the database level.

State Change Scripting

In order to account for the variety of different scenarios that may be necessary, Heimdall provides a generic state change scripting mechanism to allow simple scripts to orchestrate with third party tools as part of a failover. The script should be named `statechange-.(sh | .ps1 | .bat | .py | .pl)` and should be located in the install directory for the Heimdall Server component. An example `.bat` script is provided in the default install package for the “dbdemo-mysql” data source.

The script is provided the state of the data source via the command line, i.e.

```
statechange-dbdemo-mysql.bat Primary:true,false,true,jdbc,1:mysql://mysql.heimdalldata.com:3307/tpch Secondary:false,true,false,0,jdbc:mysql://mysql.heimdalldata.com:3308/tpch
```

The values for each server entry are:

- Server name
- Enabled (true or false)
- Healthy (true if healthy, or not monitored)
- Writeable state (true or false)
- Weight (0 or positive integer)
- jdbc URL

The script needs to take these values into account and then print the desired changes to the data source configuration on the standard output of the script. The valid commands that can be issued are:

- debug
- commit
- enabled
- writeable

- weight

The debug option results in all commands being printed as received. Enabled, writeable and weight all adjust the respective options for the specified server. The commit option enacts the new configuration. The changes are made to the stored configuration as the output is returned, and made effective on a commit. Multiple commits may be made in a single script execution, and is generally recommended. First, the failed server should be disabled and a commit made. Any orchestration that will then take time should then be performed, and finally, a new server should be made active.

It is possible to leverage both the scripted failover logic, as well as the cluster change logic. In this case, the scripted failover logic will be executed first when a failover is detected, which can promote a cluster node, then the cluster change logic will execute, which can then detect the changed servers based on the cluster state. To receive notifications of health change events, please insure the notification section in the Heimdall Management Console is configured and use the test button to verify it is working properly.

Replication Lag

One of the problems when routing queries to a database cluster that uses replication between nodes is that most such configurations results in a delay between writing to a table and being able to read the results on the read-only server. As it is a common behavior to write to a table and then immediately use the new information as part of a new query, being able to track how long it takes for this replication to occur can be an important factor in implementing read/write split. With the replication lag detection in place, Heimdall will configure tables in the Heimdall database to write to on the db cluster, and then will use those tables to do performance testing to see how long it actually takes to perform replication. In addition, a static "safety" window can be configured in the "Lag Window Buffer" setting, which will be the baseline for replication lag, to which this is added. If dynamic replication lag detection is not desired, set the Lag window buffer to a sufficiently high value that should always be safe.

With the combination of either or both the "Track Replication Lag" and "Lag Window Buffer", Heimdall will then use the last write time for a table in order to determine if a particular query is safe to read from the read-only server or only from the write master. For example, if the combination of the two results in a value of 10s, then a read of a table that hasn't been read for 11s will be eligible to be sent to the read-only server, but if the table was written 5s ago, then it will be directed to the write server. This logic closely mimics (and leverages) the logic in the cache engine that determines if a particular cached object has been expired, based on the last write time to the table.

There is also an option to set an alert threshold with "Track Replication Lag" enabled. If average lag detected would be greater than this threshold an alert stating so will be generated at most

once every 30 seconds. In addition, precise lag value will be logged. To disable alert threshold set its value to 0.

Read/Write Split Overview

Read/Write split is the idea of separating queries to a database cluster to different servers, which operate in different roles, i.e. having multiple nodes that are used for reading only, while a master that replicates to them as a read/write node. While simple in concept, there are many issues that need to be considered when implementing, including:

- **Replication Lag:** The amount of time it takes for a write against the primary node to replicate to a read-only node;
- **Transactional State:** In a transaction, a read may lock records for later modification, so needs to be executed against the primary server instead;
- **Stored Procedures:** A stored procedure call may write to a table, which is not visible at the raw SQL layer;
- **Triggers:** A trigger may be attached to one table to write to another table, which is then read by the application;
- **User Role:** A reporting user may need to be unconditionally routed to a read-only node;

All these factors come together when implementing read/write split in order to decide one question—should a query be routed to the read-only server at this time, or to the read/write server? The more information that goes into this decision, the safer it is to push more queries to the read-only server.

Heimdall helps make this decision easier by tracking the replication lag between nodes, and using it's knowledge of the last write time to a table (from the cache engine) in order to make an intelligent decision on which servers should use a given node to insure that stale data is not read.

IMPORTANT For read/write split to work properly in a multi-node environment, cache invalidation events must be visible between nodes. This is typically done by setting up a cache engine, but can also be done by using the `hdUsePGNotify` option for Postgres servers (in source connection properties). A cache engine will need to be enabled for this to work, even with the `hdUsePGNotify` option.

Configuration

In order to enable read-write split the following is required:

- Load balancing needs to be configured in the data source

- In the LB section at least two sources need to be defined. One with the writeable flag, and one without.
- If replication lag is to be accounted for one of or both the "Track Replication Lag" option needs to be checked, and/or the Lag window buffer set. Both can be used, or just one.
- In the rules tab associated with the VDB, a "reader eligible" rule needs to be defined that dictates what is eligible to be sent to the read-only nodes. Example configuration and status for tracking replication lag:

Data Source:

Load Balancing/High Availability

Enable Load Balancing:

☒

Track Cluster Changes:

☐

Use Response Metrics:

☐

Track Replication Lags:

☒

Connection Hold Time (ms):

30000

Lag Window buffer (ms):

10000

Alert threshold (ms):

0

Fallover Script:

Name:

Master

Url:

jdbc:mysql://demo-wp.cluster-cfjxl5jnvj49.us-east-1.rds.amazonaws.

Enabled:

☒

Writable:

☒

Weight:

1

Name:

Reader

Url:

jdbc:mysql://demo-wp.cluster-ro-cfjxl5jnvj49.us-east-1.rds.amazonaws.

Enabled:

☒

Writable:

☐

Weight:

1

Add Server

Rules:

4. ☒ ☐ ☐

Reader Eligible

lagignore

false

Note: The lagignore option can be set to ignore lag for a particular set of queries. This is useful when a particular set of tables is not actually sensitive to lag, but may be reported as being written to often.

Status, showing the replication lag detected, and rate of queries against each role:

Status

Status	Virtual Database	Servers						
	Magento-Demo-vdb	Status	Data Source	Server Name	Last Heartbeat Time	Queries/Sec.	Connections	Replica Lag (ms)
			Magento-Demo-source	Master	2018-04-12 20:20:44 (+0000)	2	0	-
			Magento-Demo-source	Reader	2018-04-12 20:20:44 (+0000)	25	0	4
		Pin	Status	Client IP/Hosts (driver version)			Last Connect Time	
		☒		/172.31.56.200 (proxy-18.4.11.1-20180411124052)			2018-04-12 20:20:42 (+0000)	

In the log section, the actual server selected can be observed for each query processed to validate that the correct actions are being taken, or the raw logs (if being generated) can be inspected.

Proxy Design

Heimdall's proxy is designed to operate in as lightweight a mode as possible. As such, there are a few requirements to meet:

- Disable SSL/TLS from the application to the proxy. When installed on the server as the application, this allows queries to be received and processed without the overhead of the encryption, which isn't necessary when not going over the wire. Instead, set the TLS/SSL requirement in the Heimdall data source to enforce security over the wire
- Specify the proper bind mode for the proxy to provide the maximum amount of security possible. If set to "localhost only" then only local processes can connect to the proxy, insuring that the database isn't exposed via the proxy on the server to outside users
- Ensure that if not using localhost, then the user credentials are provided in the configuration, and authentication is enabled.

Asynchronous Execution/DML Batching

Many applications and environments have application nodes that end up performing large amounts of DML operations, which are performed one at a time. Most Databases are very slow at this, with analytics/MPP databases being particularly slow. The async feature is to help improve the performance and reduce the database load when this type of activity occurs.

Basic Theory

when a DML operation is flagged with an Async execute rule, Heimdall will take the DML operation out of the current connection context, and will put it in a queue for execution on an independent connection. This queue is processed in transaction batches, which has a configurable batch size. By executing the commit across multiple DML operations, much of the overhead of the commit (the most expensive part) is shared between many DML operations, and overall DML performance improves.

There are two main categories of situations where async will help:

- One connection is inserting many DML operations in rapid order
- Connections are inserting DML operations over time from a large pool of connections

Heimdall can provide support for both, but there are situations where the Async batching should not be used.

Basic Configuration

In the Async rule logic, the following options are available:

- **asyncsize:** The maximum length of a query (in characters) that can be made asynchronous
- **batchSize:** The maximum batch size of queries to be executed—queries will not be delayed to fill the queue, but this sets the maximum size of the batch.
- **source:** The name of the data source to forward the query to for async execute. This source must be specified in the VDB definition.
- **queueName:** The per-source named async queue to process the query from. Defaults to the name of the data source the query is sent to.
- **holdUntil:** When to hold the thread until

- **spoofedResult:** The result the DML will return if the result is spoofed (only for immediate holdUntil values).

The first option, `asynthesize`, controls the actual size of the SQL that can be made asynchronous. In particular a DML operation may end up doing an insert of a million records as one statement. By using this size option, you can control how large the actual SQL can be for async to activate.

The second option is the size of the batches to perform on the database. This is the maximum value, not the minimum. Consider the case of this setting being set to 100, and the application is first started up. The first DML that is performed will likely be pushed to the database immediately, as only one query came in that instant. The queue has zero items at first, then one. As that one DML is executing on the server however, another four may be received from the application connections. Once the first DML is completed, the next batch will contain the next four in the queue after the first was done. Again, while those four are processing, another eight or more may come in. This process of queue buildup will continue until the `batchSize` is reached. At that point, even if more than that many queries are in the queue, only this many will be dispatched to the DB as a single transaction.

As a general rule the `batchSize` should be a reasonable number (64 is a good start) but not so high that if any one DML in the transaction fails and the transaction has to be restarted, then the burden of restarting will be too high (please review the retry logic below). Likewise, the value shouldn't be too low that the overall benefit isn't significant.

The third option is the source option. Normally, this won't be used, but this can be used in certain cluster types to route DML operations for a particular table to alternate data sources.

The `queueName` options is used to separate different batches of DML operations from each other. By default a single queue will exist for any given data source, based on the data source name. When an explicit `queueName` is specified however, the DML operations will be handled from a different queue, and will use a distinct database connection for processing. This is used in conjunction with table rules to filter DML for one table into one queue, and DML for another table into another queue. Through this, each table avoid blocking operations on each other, and can be used to improve performance of DML overall. The special `queueName` value of "\$ {table}" can be used to create a queue per table, and is ideal when expecting rewrites of inserts into a multi-row insert.

Next, the `holdUntil` option is used to determine the actual async behavior. For each DML, it is often the case that the application is expecting the result of the DML operation, i.e. to tell it how many rows (if any) were modified. This may impact later logic in the application. In the case that the application needs this, we have two options for `holdUntil`, that of "result" or "commit". We can wait until the result has been sent from the DB, but no commit executed, OR we can wait for a commit to be generated. For best performance, a `holdUntil` value of none should be used, BUT this requires spoofing the result so that the application can continue

processing immediately. When spoofing is needed, the final option of spoofedResult is used, and sets the value that will be returned.

Insert Rewrites

As part of the logic of batching, if a queue contains sequential inserts of the pattern "INSERT INTO () VALUES ()", and the table and field list are the same between multiple inserts, these inserts will be rewritten into a single insert with multiple rows. For now, ONLY inserts that specify the table and field list will be rewritten, to ensure that exactly the same data is being written in the same columns, and it will ONLY rewrite inserts that follow each other sequentially with no interruption. Only the batch size number of rows will be written at once.

Retry Logic

When a batch is created, there is always a chance that the DML will result in an exception. The way this is handled is that if an exception is generated at the time the sql is executed (but before the commit), then the sql is removed from the batch, the batch is rolled back, the queries are restored into the queue in the proper order, and a new batch is generated. A log entry will then be generated.

In the case of a holdUntil value of none, no exception in this case is returned to the application, since it already received a spoofed result. In such an environment, the logs should be maintained in case a replay is needed.

In the case of any other holdUntil value, the exception will be passed to the client.

The final case is if the exception occurs when the commit is called. This should be rare, but in the event that it happens, then it is impossible to determine which of the batch caused the issue (if it was one SQL in particular). As such, in this case, the entire batch is logged as a failure, and the connection to the DB is torn down and rebuilt, as it is safest to just start with a clean connection.

Blocking

One issue that is likely to come up with many attempts to use Async is the following sequence of events:

```
select from tableA where id=10; update tableA set something=newvalue where id=10; select  
from tableA where id=10; // expects the new value here
```

Without any additional logic, this would likely break the application. With the Async feature of Heimdall however, what will happen is that the update to the table will result in a lock that is associated with the table. When the final select is issued, it will block until the update commits,

THEN the select will be issued against the database. This prevents this sequence from returning the incorrect value, and breaking the application. Please note: This locking is only done within a single Heimdall instance—if additional nodes are active, they will not block on the same action, as the synchronization time to implement this would be excessive. The main objective is to insure that at least for a given connection, the order of operations are maintained.

Initializing the Async Connection

Often, the connection that the Async logic used will need to be initialized with SQL commands that make it behave differently than normal connections. As an example, with DML operations, Greenplum should in general have the MPP optimizer disabled, as it adds excessive latency that doesn't provide a benefit. In order to do this, the connection property of "initSQLAsync" can be set on the data source, with the property of the SQL desired. In the Greenplum example, it would be "set optimizer = off".

Proxy Authentication

Before allowing queries through the proxy, they must first be authenticated by the proxy (in general). There are various configuration modes for this, and what is supported depends on the type of database on the back-end.

Authentication Modes

Heimdall supports 6 methods:

- **None** (all)
- **Passthrough** (all)
- **Proxy Configured Users** (all)
- **SQL Driven** (all)
- **Active Directory/LDAP** (all)
- **Kerberos/GSSAPI** (all)

None Authentication

Proxy Configuration

Local Proxy: ☒

Run as Service: ☐

Address Binding Type: Any

Proxy Port(s): 5433

Max Heap Size (MB): 600

Proxy TLS Support: ☐

Authentication mode: None

In none Authentication mode, the client is always authenticated no matter what credentials he will send. It's inheriting credentials from default [Data Source](#) and then authenticate user. It can be used for test purposes, without changing the connection string on the client side.

Warning: Disabled in Production Deployment Environment.

Passthrough Authentication

Proxy Configuration

Local Proxy: ☒

Run as Service: ☐

Address Binding Type:

Any

▼

Proxy Port(s):

5433

Max Heap Size (MB):

600

⬆️⬆️

Proxy TLS Support:

☐

Authentication mode:

Passthrough

▼

The client is requested to send the credentials in a way so that they can be passed through to the database server, at which point the DB performs the initial authentication. This allows single point of configuration for the password.

Warning: For MySQL passthrough authentication, mysql clients must support the mysql_clear_password authentication plugin and ensure it is enabled during the connection.

IAM Authentication

Heimdall supports IAM authentication for RDS (MySQL, PostgreSQL) and Redshift. To enable this, **passthrough** authentication must be used.

Example TLS Configuration:

Datasource

MySQLRDS

Enable

Commit

Configuration

Driver: MySQLDriver

JDBC URL: jdbc:mysql://cluster-aws:3306/\${database}

Username: admin

Password:

Test query: select 1

Test Connection

Execute scripts

Connection Properties

Custom Property

trustCertificateKeyStoreUri	file:/home/test/sdkman/candidates/java/17.0.6-amzn/lib/security/cacerts	
trustCertificateKeyStorePa	test	
defaultCatalog	test	
useSSL	true	

Virtual Database

Proxy Configuration

Local Proxy: ☒

Run as Service: ☐

Address Binding Type:

Any

▼

Proxy Port(s):

3307

Max Heap Size (MB):

650

⬆️⬆️

Proxy TLS Support: ☒

Proxy TLS Required: ☒

Enable Legacy TLS: ☐

Certificate:

global_use_certificate

▼

Authentication mode:

Passthrough

▼ ⓘ

Proxy Configured Users Authentication

Proxy Configuration

Local Proxy: ☒

Run as Service: ☐

Address Binding Type:

Any

▼

Proxy Port(s):

5433

Max Heap Size (MB):

600

⬆ ⬇ ⬆

Proxy TLS Support:

☐

Authentication mode:

Proxy Configured Users

▼

proxy-user1

••••

✕

Secret

☐

username-from-secret

secret-name-for-proxy-user

✕

Secret

☒

Regex

☐

(.*)

secret/{\$1}

✕

Secret

☒

Regex

☒

Default

The username and password to be used is configured directly on the Heimdall management server. The credentials must match what is also configured on the database.

Example: If you have *proxy-user1* user on the database with some password, then you have to configure it in this section to allow proxy to perform successful authentication.

Secrets

The username and password can be stored in the [Secrets Manager](#).

Warning: This method does not currently support rotating credentials, in case of changes on Secrets Manager (and on the database possibly) proxy will still use old values, this can be manually updated by committing the VDB configuration which triggers fetching of credentials.

Note: A secret also can contain exactly the same properties which are being used in [SQL Driven authentication](#). If a secret contain "database" field it must be JSON array, e.g:

```
{
  "database": [ "database1", "database2" ]
}
```

and "username" field must be simply singular string, e.g:

```
{
  "username": "user1"
}
```

Warning: Currently, multiple usernames are not supported.

Secrets + Regex

It is possible to define proxy users using secrets based on regular expressions (regex). In this case, the username field accepts a regex pattern, and the secret name field can be specified to match based on the group of users identified by the regex pattern, as shown in the attached screenshot. During proxy authentication, it is checked whether the authenticating user matches any of the defined regex patterns. If a match is found, Secrets Manager is queried for the corresponding credentials, and the authentication process proceeds.

Note: Both `{ $1 }` and `$ {1}` placeholders are supported here.

Example:

1. In the configuration, you have the username set as `"(.*)"` and the secret name as `"secret/{ $1 }"`, as shown in the screenshot below.
2. A login attempt is made by user "test", who provides their username and password.
3. The proxy checks the provided username against the predefined regex, and a match is found. At this point, the secret name is also determined. The placeholder `{ $1 }` will be

replaced by groups 1 from the matched regex in the username field. In this case, the resulting secret name will be: "secret/test".

4. The proxy queries the Secrets Manager for credentials under this secret name, and if they match the user's input, the user will be authenticated.

It is also possible to define more than one regex group in the username field and then use them in the secret name as { \$1 }, { \$2 }, etc.

For those utilizing a custom secrets manager, it is imperative to adhere to the structure outlined in AWS RDS secrets manager. For instance, the JSON format should resemble the following:

```
{
  "username": "test",
  "password": "test"
}
```

SQL Driven Authentication

Proxy Configuration

Local Proxy: ☒

Run as Service: ☐

Address Binding Type:

Any

Proxy Port(s):

5433

Max Heap Size (MB):

600

Proxy TLS Support:

☐

Authentication mode:

SQL Driven

Synchronize DB Authentication:

☐

Admin User:

admin

Secret ☐

Admin Password:

••••

☒

Authorization Query:

select * from heimdall.pg_hba where enabled = true order by line_number a

SQL Driven Authentication leverages a table on the database itself that stores details in a format roughly matching the format of the Postgres pg_hba.conf file, although this format supports all supported proxies. See the details below on the creation and use of this table. SQL Driven Authentication also supports setting various attributes for the users on a per-user basis

in a way not supported by other methods, and can layer with passthrough authentication as well.

Simple configuration

The simplest way to configure an authentication by SQL Driven Authentication is to turn on mentioned option in "Proxy Configuration" section of [Virtual DB](#) tab and provide query as "Authorization Query".

Example authorization query can be declared as below:

```
select password from heimdall.pg_hba where user_name = '${user}';
```

Used authorization query expects declared table like below example in database.

```
CREATE SCHEMA IF NOT EXISTS heimdall;  
REVOKE ALL ON SCHEMA heimdall FROM public;  
  
CREATE TABLE heimdall.pg_hba (  
    user_name text NULL, -- username (exact)  
    "password" varchar NULL -- user's password  
);
```

Provided example will provide authentication of given user by password returned as result from Authorization Query.

SQL Query Variables

The authentication query supports the following variables, which are filled in before querying the database:

- **\${user}** - The username of the connecting user
- **\${source}** - The client IP of the connecting user, as seen at the socket level (i.e. NAT translation may hide this).

If additional variables are desired, please contact Heimdall support to submit a feature request.

SQL Driven Authentication Caching

In SQL Driven authentication credentials retrieved for a given users are cached for one minute. There is a possibility to clear those credentials caches for a specific proxy. You have to go to

into status tab, expand collapse menu button (burger button) and chose "Clear credentials properties" option. More information might be found under this [link](#).

Advanced configuration, with connection pool properties control

To start with advanced configuration, first should be realized that whole process of authentication depends on data table containing rows with information about rules for declared situations of authorization. On side of proxy the only thing which can be configured is query which calls for result from database. That means that whole authentication process can be elastic and is only limited by database authentication table content and by authentication query.

SQL Driven Authentication provides 4 different authentication options, which can be declared in authentication table in column "auth_method".

Authorization option	Option keyword	Description
Server Authentication	scram-sha-256, trust, pam, bsd	Pass the authentication through to the server, and only allow if the server accepts. This is the default if no auth_method is provided or the value is null, or an unknown value is returned
Password authentication option	md5, password	Authentication is performed by comparing password provided by user during authorization and "password" column read from result row matching user.
Reject option	reject	Unconditionally rejects connection made to proxy by user.
LDAP authentication option	ldap	LDAP authentication option enables authentication by using ldap server. To use this authentication, required is to provide information about ldap server inside "options" column of result.

Note: The MySQL proxy only supports the password authentication when SQL Driven authentication is used.

SQL Driven Authentication is based on values inside columns of result set. There are 4 types of columns: filtering columns, authentication columns, connection columns and override columns.

FILTERING COLUMNS

Filtering columns specify if given row of result matches user connecting to proxy. These include:

Filter column label	Type	Description
connection	string	Value in this column specifies what type of connection was used by a user to connect to a proxy. If column doesn't exist in result or value is NULL or value isn't a specified keyword, then authentication rule applies to all situations.
user_name	array value	Value in this column specifies to which user name rule should be applied. If column doesn't exist in result or value is NULL or value is {all}, then authentication rule applies to all users.
database	array value	Value in this column specifies to which database connection is made. If column doesn't exist in result or value is NULL or value is {all}, then authentication rule applies to all users.
address	string	Value in this column specifies from which IP address user is connecting to proxy. If "netmask" column's value exist, then can be considered as range of IP addresses. If column doesn't exist in result or value is NULL or value is {all}, then authentication rule applies to all users. Works with IP address of IPv4 and IPv6.
netmask	string	

Filter column label	Type	Description
		Value in this column is only considered if value exist in column "address" of result. Value should be netmask address i.e. "255.255.255.0" or "ffff.ffff.ffff.ffff:0000:0000:0000". When specified, "address" column specifies range of IP addresses inside of specified net. If not specified, then considered as netmask length 32.

AUTHENTICATION COLUMNS

Authentication columns specifies options used during authentication. There are 3 columns, which are used as authentication columns.

Authentication column label	Type	Description
auth_method	string	Value in this column declares what authentication method should be used. If column doesn't exist in result or is NULL, then default authentication option is password authentication.
password	string	Value in this column contains password which is used in password authentication inside proxy.
options	special formatted string	Values in this columns are used for providing extra options for some authentication methods. Expected value in this column in string containing pairs key="value", where pairs are separated by spaces e.g. "key1="value1" key2="value2". For now, only authentication method using this is LDAP Authentication Method, which is using defined keywords to extract proper values (information about ldap keywords can be found in "LDAP Authentication inside SQL Driven Authentication" section).

CONNECTION COLUMNS

Connection columns specify optional properties which will be set after successful authentication. There are 3 columns, which are used as connection columns:

Connection column label	Type	Description
pool:multiplex	bool	Set to override default multiplex setting for user, rules can further override.
pool:maxUserIdle	integer	Set to override default max idle setting for user, rules can further override.
pool:maxUserActive	integer	Set to override default max active setting for user, rules can further override.

In this case, fields are being defined that match to the properties that the "pool" rule types inject into a query's metadata. This allows a user's multiplex, maxUserIdle, and MaxUserActive settings to be set at a default level on login without even matching a rule. Other rule behaviors can be triggered in the same way. To see the exact property that a rule injects, you would use the "debug" logging on the vdb, which will print as part of the logging the exact property name and the value injected.

Note: As these columns match to rule properties, any rule property can actually be set, these are simply examples that are connection oriented and help set the proper behavior for the connection. Use verbose debugging to inspect properties that are used after a rule match, and any such properties can be used as a column header for advanced control.

OVERRIDE COLUMNS

Override columns specify values that should override connection properties after successful authentication on proxy side. There are 3 columns which are used as override columns:

Override column label	Type	Description
db_username	string	Set to override username that is used for connection after successful authentication on proxy side. If no such column exists or value is empty/ null then user_name column is used.

Override column label	Type	Description
db_password	string	Set to override password that is used for connection after successful authentication on proxy side. If no such column exists or value is empty/null then password column is used.
db_database	string	Set to override database name that is used for connection after successful authentication on proxy side. This would mean that we can redirect connection to different database than one supplied in database column. If no such column exists or value/null is empty then database column is used.

These columns can be used to create connection with different values than these used for proxy authentication. For example, we can redirect connection to other database/schema or map user-friendly name to say "000001" as the user on the DB side.

Note: Some PostgreSQL clients will display database column value instead of db_database, event though they are connected to correct one, because they get this information directly from user input and don't query to find which database got connected.

WARNING: DO NOT USE OVERRIDE COLUMNS WITH TOKENIZED AUTHENTICATION

Example usage of advanced configuration

Advanced configuration can be used in below way. On start set authentication query as below in VDB configuration. SQL Server does not support boolean types, so instead of **true**, use **1**.

```
select * from heimdall.pg_hba where enabled = true order by line_number asc
```

To create a table used to authentication for Postgres, Redshift, MySQL, and SQL Server, you can **execute the appropriate script** in the [Data Source](#) tab or do it manually on the database. Below is an example for Postgres. For other types of databases, you must remember that the column types may differ slightly.

```
CREATE SCHEMA IF NOT EXISTS heimdall;
REVOKE ALL ON SCHEMA heimdall FROM public;

CREATE SEQUENCE IF NOT EXISTS pg_hba_seq INCREMENT BY 5 START WITH 10;
CREATE TABLE IF NOT EXISTS heimdall.pg_hba (
    line_number int4 NULL,
    enabled bool NOT NULL DEFAULT true, -- controls if the row is active
    connection text NULL, -- all, host, local, hostssl, or hostnossl
    address text NULL, -- IPv4 or IPv6 subnet or address
    netmask text NULL, -- Netmask for address, defaults to /32
    "database" _text NULL, -- array value, use {any} or null for all
    databases, or provide the names of the database names (exact match).
    user_name _text NULL, -- array value, use {any} or null for all users, or
    provide the names of the users (exact match).
    "password" varchar NULL, -- The password to use for the password type,
    must be provided
    ldapgroups varchar NULL, -- csv group names, as if pulled from ldap for
    rule processing
    auth_method text NULL, -- trust, reject, ldap or password, null=password
    "options" _text NULL, -- ldap options
```

```

        "pool:multiplex" bool NULL, -- set to override default multiplex setting
        for user, rules can further override
        "pool:maxUserIdle" int4 NULL, -- set to override default max idle setting
        for user, rules can further override
        "pool:maxUserActive" int4 NULL, -- set to override default max active
        setting for user, rules can further override
        db_username text NULL, -- set to override username that is used for
        connection after successful authentication on proxy
        side

        db_password text NULL, -- set to override password that is used for
        connection after successful authentication on proxy
        side

|

        db_database text NULL -- set to override database name that is used for
        connection after successful authentication on proxy side
    );

```

Next (optional) you can add column comments:

```

COMMENT ON COLUMN heimdall.pg_hba.enabled IS 'controls if the row is active';
COMMENT ON COLUMN heimdall.pg_hba.connection IS 'all, host, local, hostssl, or
hostnossl';
COMMENT ON COLUMN heimdall.pg_hba.address IS 'IPv4 or IPv6 subnet or address';
COMMENT ON COLUMN heimdall.pg_hba.netmask IS 'Netmask for address, defaults
to /32';
COMMENT ON COLUMN heimdall.pg_hba."database" IS 'array value, use {any} or
null for all databases, or provide the names of the database names (exact
match).';
COMMENT ON COLUMN heimdall.pg_hba.user_name IS 'array value, use {any} or null
for all users, or provide the names of the users (exact match).';
COMMENT ON COLUMN heimdall.pg_hba."password" IS 'The password to use for the
password type, must be provided';
COMMENT ON COLUMN heimdall.pg_hba.ldapgroups IS 'csv group names, as if pulled
from ldap for rule processing';
COMMENT ON COLUMN heimdall.pg_hba.auth_method IS 'trust, reject, ldap or
password, null=password';
COMMENT ON COLUMN heimdall.pg_hba."options" IS 'ldap options';
COMMENT ON COLUMN heimdall.pg_hba."pool:multiplex" IS 'set to override default
multiplex setting for user, rules can further override';
COMMENT ON COLUMN heimdall.pg_hba."pool:maxUserIdle" IS 'set to override
default max idle setting for user, rules can further override';
COMMENT ON COLUMN heimdall.pg_hba."pool:maxUserActive" IS 'set to override
default max active setting for user, rules can further override';
COMMENT ON COLUMN heimdall.pg_hba.db_username IS 'set to override username
that is used for connection after successful authentication on proxy side';
COMMENT ON COLUMN heimdall.pg_hba.db_password IS 'set to override password
that is used for connection after successful authentication on proxy side';
COMMENT ON COLUMN heimdall.pg_hba.db_database IS 'set to override database
name that is used for connection after successful authentication on proxy
side';

```

Note: the way this is being done, the row numbers do not need to be unique, but will be used to order precedence for authentication.

Helper command to import into public.pg_hba (defined above) from the Postgres pg_hba_files_rules (PG 10+):

```
insert into heimdall.pg_hba (select nextval('pg_hba_seq'), true, "type",
address, netmask, "database", user_name, null, null, auth_method from
pg_hba_file_rules);
```

If you want to add a new entry, you can do so as follows (including enabling multiplexing for this user). Note, the user field is an array, in the event multiple accounts should use the same configuration:

```
insert into heimdall.pg_hba values (nextval('pg_hba_seq'), true, 'host', null,
null, null, array ['user1'], 'newpassword', array
['group1','group2','group3'], 'password', null, true, null, null, null, null,
null);
```

Remember that only Postgres supports column types as arrays. For other databases, you can provide multiple values as a string within {}, separating the values with commas, e.g., '{user1, user2}'.

Filter columns keywords

Filter columns have specified keywords, which can be used to change behavior of rule declared in the given row of result set.

CONNECTION COLUMN KEYWORDS

Inside connection column can be user below keywords to change use of set rule.

Keyword	Description
all, host, local	Rule declared in this row will be used for all situations.
hostssl	Rule declared in this row will be used only when connection is made to a proxy with using SSL encryption.
hostnossl	Rule declared in this row will be used only when connection is made to a proxy without using SSL encryption.

USER_NAME COLUMN KEYWORDS

Inside user_name column can be user below keywords to change use of set rule.

Keyword	Description
all	Rule declared in this row will be used for all users.
role_name	Rule declared in this row will be used for users that are directly or indirectly members of given role

DATABASE COLUMN KEYWORDS

Inside database column can be user below keywords to change use of set rule.

Keyword	Description
all	Rule declared in this row will be used for all database connections.
sameuser	Rule declared in this row will be used for database connections, which requested database name is the same as requested user name.
samerole/samegroup	Rule declared in this row will be used for database connenctions, which requested user are directly or indirectly member of role with name same as database name.

ADDRESS COLUMN KEYWORDS

Inside address column can be user below keywords to change use of set rule.

Keyword	Description
all	Rule declared in this row will be used for all addresses.
samehost	Rule declared in this row will be used for addresses that matches the server's IP addresses.
samenet	Rule declared in this row will be used for addresses that matches any subnet, which server is directly connected to.
<host_name>	Rule declared in this row will be used for addresses matching given host name. Check of this keyword is made in two steps: first is made reverse name resolution of client's IP address and checked if is the same as given host name. In secodn step is performed forward name resolution of given host name and comparison of IP addresses is done. If given host name starts with a dot (i.e. ".server.com") then only reverse name resolution of client's IP address and comparison is done (i.e. for set host name ".server.com", if client IP returns as host name "heimdalldata.server.com", then it matches, if client's host name is "server.com", then comparison returns false).

OVERRIDE COLUMNS EXAMPLE

Let's say we have defined authorization query like this.

```
select * from heimdall.pg_hba
```

And filled our authentication table with following values:

user_name	password	database	db_username	db_password	db_database
user1	password1	db1	root	root	master
user2	password2	db2	<empty or null>	<empty or null>	<empty or null>

For given credentials like:

```
user: user1
password: password1
database: db1
```

After filtering results of query with filter columns we take the first row resulting in successful authentication on proxy side and then the connection to native database will be established with these properties:


```
user: root
password: root
database: master
```

For given credentials like:

```
user: user2
password: password2
database: db2
```

Matched row here will be the second one but credentials won't be overridden because override columns are empty or null.

LDAP Authentication inside SQL Driven Authentication

SQL Driven Authentication enable authentication by using LDAP server. To use it properly two options should be given:

- value in column `auth_method` should be set as `ldap` ;
- value in column `options` should contain information about LDAP defined by proper keywords.

GENERAL KEYWORDS

Keyword	Description
ldapservers	Value assigned to this keyword should be names or IP addresses of LDAP servers, which should be used for authentication. Multiple servers can be specified by separating them with spaces e.g. 'ldapservers="server1.com server2.com"'. If multiple servers are specified then only one of them have to authenticate user.
ldapport	Value assigned to this keyword should be port number on LDAP server, which should be used to make connection. If keyword isn't specified then port 389 is used. E.g. 'ldapport=636'.
ldapscheme	Value assigned to this keyword can define if TLS should be used during connection to LDAP server. TLS can be set by setting value "ldaps" e.g. 'ldapscheme="ldaps"'.
ldaptls	Value assigned to this keyword can define if TLS should be used during connection to LDAP server. TLS can be set by setting value "1" e.g. 'ldaptls="1"'.
ldapcaoverride	Value assigned to this keyword can define if TLS validation of LDAP server certificate should be performed. Available values are "true" and "false", e.g. 'ldapcaoverride="true"'. By default it is set to "false".

SIMPLE BIND MODE KEYWORDS

Keyword	Description
ldapprefix	Value assigned to this keyword is string to prepend to the user name when forming the DN to being used for bind in Simple Bind Mode e.g. 'ldapprefix="CN="'.
ldapsuffix	Value assigned to this keyword is string to append to the user name when forming the DN to being used for bind in Simple Bind Mode e.g. 'ldapsuffix="",CN=Users,DC=example,DC=com"'.

SEARCH+BIND MODE KEYWORDS

Keyword	Description
ldapbasedn	Value assigned to this keyword should be root DN to start searching for user during Search+Bind Mode e.g. 'ldapbasedn="CN=Users,DC=example,DC=com"'.

Keyword	Description
ldapbinddn	Value assigned to this keyword should specify user to bind to directory to perform the search during Search+Bind Mode e.g. 'ldapbinddn="admin" or 'ldapbinddn="CN=admin,CN=Users,DC=example,DC=com" .
ldapsecbinddn	Value assigned to this keyword should specify user to bind to directory to perform the search if the search with ldapbinddn fails.
ldapsearchdn	Value assigned to this keyword should specify the domain in which authenticated user's groups will be searched.
ldapbindpasswd	Value assigned to this keyword should specify password for ldapbinddn user to bind to directory to perform the search during Search+Bind Mode e.g. 'ldapbindpasswd="password" .
ldapsecbindpasswd	Value assigned to this keyword should specify password for ldapsecbinddn user to bind to directory to perform the search during Search+Bind Mode.
ldapsearchattribute	Value assigned to this keyword should be attribute to match against user name in the search during Search+Bind mode e.g. 'ldapsearchattribute="sAMAccountName" .
ldapgroupsearchattribute	Value assigned to this keyword should be an attribute to match against group identifying attribute value, in the Search+Bind mode e.g. 'ldapgroupsearchattribute="cn". Setting this keyword makes it required to extract at least one group to be authenticated.
ldapusenestedgroupsfilter	Value assigned to this keyword should describe if nested groups filter should be used during a search for names of groups for an user. Accepted values are "true" and "false". If this value isn't set, by default it is set to "false".
ldapsearchfilter	Value assigned to this keyword should be search filter used instead "ldapsearchattribute". All occurrences of "\$username" will be replaced by user name. E.g. 'ldapsearchfilter="(sAMAccountName=\$username)" .
ldapurl	Value assigned to this keyword allows to declare some of other LDAP options in more compact form. For information about proper using of this keyword, check LdapUrl Keyword section.
ldapservertype	Value assigned to this keyword must be one of the following: ACTIVE_DIRECTORY, REDHAT_IDM, OPENLDAP, JUMPCLOUD, OKTA, OTHER.

LdapUrl Keyword

Keyword "ldapurl" is used to declare multiple LDAP options in single declaration. However, it requires special value format to declare options properly. Format is:

```
ldap[s]://host[:port]/basedn[?[attribute][?[scope][?[filter]]]]
```

The possible options specified in the format are:

- ldap/ldaps - have to be `ldap` or `ldaps`, set `ldap` to make connection to LDAP server without TLS, `ldaps` for connection with TLS;
- host - required value, works similar to `ldapserver`, but only one server can be declared
- port - optional value, works as `ldappport`
- basedn - required value, works as `ldapbasedn`
- attribute - required if `filter` isn't declared, works as `ldapsearchattribute`
- scope - optional value, declare to specify scope used during searching for user, possible values are: `base` for baseObject scope, `one` for singleLevel scope, `sub` for wholeSubtree scope. If not defined, then default is `base` scope.
- filter - required if `attribute` isn't declared, works as `ldapsearchfilter`.

EXAMPLES

Let's assume that we want to Simple Bind authenticate to LDAP server with below options: *

LDAP server url - `ldap://server.example.com:389` * LDAP prefix - `CN=` * LDAP suffix

- `,CN=Users, DC=example, DC=com`

Then value in `options` column should look like:

```
ldapservers="server.example.com" ldapprefix="CN=" ldapsuffix=",CN=Users,DC=example, DC=com"
```

Let's assume that we want to Search+Bind authenticate to LDAP server with below options:

- LDAP server url - `ldap://server.example.com:456`
- LDAP base DN - `CN=Users,DC=example,DC=com`
- LDAP search attribute - `sAMAccountName`
- LDAP bind DN - `binduser`
- LDAP bind password - `examplepassword`

A value in `options` column should look like:

```
ldapservers="server.example.com" ldapport=456  
ldapbasedn="CN=Users,DC=example,DC=com", ldapsearchattribute="sAMAccountName"  
ldapbinddn="binduser" ldapbindpasswd="examplepassword"
```

If we would want to specify LDAP bind DN as `CN=binduser,CN=Users,DC=example,DC=com` then value in `options` columns could look like:

```
ldapservers="server.example.com" ldapport=456  
ldapbasedn="CN=Users,DC=example,DC=com", ldapsearchattribute="sAMAccountName"  
ldapbinddn="CN=binduser,CN=Users,DC=example,DC=com"  
ldapbindpasswd="examplepassword"
```

If we would want to use `ldapurl` keyword instead, then value in `options` column could look like:

```
ldapurl="ldap://server.example.com:456/CN=Users,DC=example,DC=com?  
sAMAccountName?sub" ldapbinddn="CN=binduser,CN=Users,DC=example,DC=com"  
ldapbindpasswd="examplepassword"
```

Let's assume that we want to Search+Bind authenticate to LDAP server with below options: *

LDAP server url - `ldaps://server.example.com:678` * LDAP base DN -

`CN=Users,DC=example,DC=com` * LDAP search filter - `(|(sAMAccountName=$username)`

`(uid=$username))` * LDAP bind DN - `binduser` * LDAP bind password - `examplepassword`

A value in `options` column should look like:

```
ldapservers="server.example.com" ldapport=678 ldapscheme="ldaps"  
ldapbasedn="CN=Users,DC=example,DC=com" ldapsearchfilter="(|  
(sAMAccountName=$username)(uid=$username))" ldapbinddn="binduser"  
ldapbindpasswd="examplepassword"
```

or:

```
ldapserver="server.example.com" ldapport=678 ldaptls=1
ldapbasedn="CN=Users,DC=example,DC=com" ldapsearchfilter="( |
(sAMAccountName=$username)(uid=$username))" ldapbinddn="binduser"
ldapbindpasswd="examplepassword"
```

If we would want to use `ldapurl` keyword instead, then value in `options` column could look like:

```
ldapurl="ldaps://server.example.com:678/CN=Users,DC=example,DC=com??sub?( |
(sAMAccountName=$username)(uid=$username))" ldapbinddn="binduser"
ldapbindpasswd="examplepassword"
```

If we would want to change or scope to `base`, then value in `options` column should look like:

```
ldapurl="ldaps://server.example.com:678/CN=Users,DC=example,DC=com??base?( |
(sAMAccountName=$username)(uid=$username))" ldapbinddn="binduser"
ldapbindpasswd="examplepassword"
```

or:

```
ldapurl="ldaps://server.example.com:678/CN=Users,DC=example,DC=com???( |
(sAMAccountName=$username)(uid=$username))" ldapbinddn="binduser"
ldapbindpasswd="examplepassword"
```

If we would want to change or scope to `one`, then value in `options` column should look like:

```
ldapurl="ldaps://server.example.com:678/CN=Users,DC=example,DC=com??one?( |
(sAMAccountName=$username)(uid=$username))" ldapbinddn="binduser"
ldapbindpasswd="examplepassword"
```

Extra information about advanced configuration

There are some information which may be helpful before preparing authentication table and authentication query:

- if there is more than one row which matching to user properties, then first is taken from result set (that's why sorting of result may change authentication process);
- if there is no row matching to user properties, then authentication is rejected;
- there are three characters which can cause a problem if they are in value inside array: '{' and '}' as they are removed; '"' as quotes are used to protect commas in value (required for providing ldap options);
- if you don't want to use or can't use array type of column then can be used text column, but value inside should match pattern of array value i.e. {"user1", "user2"};

- if using text column instead of array column, recommended is to use array pattern like `'{"user1","user2"}'`, but possible is to use without curly brackets like `"user1","user2"` and even without quotes like `'user1, user2'`;
- keywords inside array value columns can be joined i.e. in column `user_name` can be declared array value like `'{"user1", "+role1"}'`, what means, that this rule will be used for requested user `user1` or requested users who are members of role `role1`.

SQL Routing via SQL Driven Authentication

One use case to use SQL Driven Authentication involves SaaS providers, where they want to keep a table with different clusters to service customers with. For this to work, the SQL auth table should include a column named **"host"** and in the [Data Source](#), the **JDBC URLs** should contain `${host}` in place of the hostname. Further, a value of **"defaultHost"** should be set in the connection properties, and this should be the default hostname, where the SQL auth table lives. From here, when a user is authenticated by the SQL Auth, and the hostname returned, it will use the new host.

Note: the variable does NOT need to be a fully qualified hostname.

Example: in AWS, you can have the value return the cluster ID, and in the LB section use `${host}-ro.domain.com`, and this will enable read/write split, as long as the names after replacement resolve out properly.

Here is a table definition including the host field:

```
CREATE TABLE heimdall.pg_hba (
    line_number int4 NULL,
    enabled bool DEFAULT true NOT NULL,
    "connection" text NULL,
    address text NULL,
    netmask text NULL,
    "database" _text NULL,
    user_name _text NULL,
    "password" varchar NULL,
    ldapgroups varchar NULL,
    auth_method text NULL,
    "options" _text NULL,
    "pool:multiplex" bool NULL,
    "pool:maxUserIdle" int4 NULL,
    "pool:maxUserActive" int4 NULL,
    db_username text NULL,
    db_password text NULL,
    db_database text NULL,
    host varchar NULL
);
```

Now, with the host value filled in, during authentication (with [Verbose Debug Mode](#) logging enabled) you can find the following log entries:

```
2025-04-17T19:08:15.878905136Z C5:T58:Opening connection to default source,
props: {proxyIP=/74.98.202.222:49724, catalog=odoo, hdpassword=*****
hdport=8087, proxyPort=5432, password=***** baseurl=http://localhost:8087/,
options={}, host=127.0.0.1, url=jdbc:postgresql://${host}:5433/${catalog},
auth_method=password, vdb=Odoo-vdb, hduser=Ccbx1Fs1CYaoC2Rw, connid=5,
ldapgroups=odoo, hdservername=localhost, user=odoo}
2025-04-17T19:08:15.853354767Z Connection properties on proxy connect:
{proxyPort=5432, password=***** auth_method=password, proxyIP=/
74.98.202.222:49724, catalog=odoo, options={}, host=127.0.0.1,
ldapgroups=odoo, user=odoo}
04/17/2025 15:08:15:853 ip-172-31-22-239.us-east-2.compute.internal/
74.98.202.222 (7d6ff) Odoo-vdb:5432 DEBUG 0 0
2025-04-17T19:08:15.853446866Z Properties on parse:{proxyPort=5432,
password=***** auth_method=password, proxyIP=/74.98.202.222:49724,
catalog=odoo, options={}, host=127.0.0.1, ldapgroups=odoo, user=odoo}
2025-04-17T19:08:15.853161611Z Auth: Called password method
04/17/2025 15:08:15:853 ip-172-31-22-239.us-east-2.compute.internal/
74.98.202.222 (7d6ff) Odoo-vdb:5432 DEBUG 0 0
2025-04-17T19:08:15.853777622Z C5:T58:Opening connection: {proxyIP=/
74.98.202.222:49724, catalog=odoo, hdpassword=***** hdport=8087,
proxyPort=5432, password=***** baseurl=http://localhost:8087/, options={},
host=127.0.0.1, url=jdbc:heidmall://localhost:8087/Odoo-vdb,
auth_method=password, vdb=Odoo-vdb, hduser=Ccbx1Fs1CYaoC2Rw, connid=5,
ldapgroups=odoo, hdservername=localhost, user=odoo}
04/17/2025 15:08:15:853 ip-172-31-22-239.us-east-2.compute.internal/
74.98.202.222 (7d6ff) Odoo-vdb:5432 DEBUG 0 0
2025-04-17T19:08:15.853877148Z C5:T58:Opening connection to default source,
props: {proxyIP=/74.98.202.222:49724, catalog=odoo, hdpassword=*****
hdport=8087, proxyPort=5432, password=***** baseurl=http://localhost:8087/,
options={}, host=127.0.0.1, url=jdbc:postgresql://${host}:5433/${catalog},
auth_method=password, vdb=Odoo-vdb, hduser=Ccbx1Fs1CYaoC2Rw, connid=5,
ldapgroups=odoo, hdservername=localhost, user=odoo}
04/17/2025 15:08:15:853 ip-172-31-22-239.us-east-2.compute.internal/
74.98.202.222 (7d6ff) Odoo-vdb:5432 DEBUG 0 0
2025-04-17T19:08:15.853643749Z Params after parsing: {proxyIP=/
74.98.202.222:49724, catalog=odoo, hdpassword=***** url=jdbc:heidmall://
localhost:8087/Odoo-vdb, hdport=8087, proxyPort=5432, password=*****
auth_method=password, baseurl=http://localhost:8087/, vdb=Odoo-vdb,
hduser=Ccbx1Fs1CYaoC2Rw, options={}, host=127.0.0.1, ldapgroups=odoo,
hdservername=localhost, user=odoo}
04/17/2025 15:08:15:827 ip-172-31-22-239.us-east-2.compute.internal/
74.98.202.222 (7d6ff) Odoo-vdb:5432 DEBUG 0 0
2025-04-17T19:08:15.827655054Z AuthProperties: {password=*****
auth_method=password, options={}, host=127.0.0.1, ldapgroups=odoo}
04/17/2025 15:08:15:827 ip-172-31-22-239.us-east-2.compute.internal/
74.98.202.222 (7d6ff) Odoo-vdb:5432 DEBUG 0 0
2025-04-17T19:08:15.827298111Z Password properties before filtering:
[ { "database": [ "odoo" ], "password": "Heimdalltest1", "auth_method":
"password", "line_number": 10, "user_name": [ "odoo" ], "options": {}, "host":
"127.0.0.1", "ldapgroups": "odoo", "enabled": true } ]
04/17/2025 15:08:15:827 ip-172-31-22-239.us-east-2.compute.internal/
74.98.202.222 (7d6ff) Odoo-vdb:5432 DEBUG 0 0
2025-04-17T19:08:15.827252701Z Options retrieved: {} parsed={}
04/17/2025 15:08:15:826 ip-172-31-22-239.us-east-2.compute.internal/
74.98.202.222 (7d6ff) Odoo-vdb:5432 DEBUG 0 0
2025-04-17T19:08:15.826522330Z Authentication query against url:
jdbc:postgresql://demoa.heidmalldata.com:5433/postgres
```

```
04/17/2025 15:08:15:826 ip-172-31-22-239.us-east-2.compute.internal/  
74.98.202.222 (7d6ff) Odoo-vdb:5432 DEBUG 0 0  
2025-04-17T19:08:15.826469540Z heimdallpw: query: select * from  
heimdall.pg_hba where enabled = true order by line_number asc
```

Notice in the top line, where it says "**opening connection**" and the value for the host parameter is "**127.0.0.1**". In this case, this is the value of the host column.

Now let's look at the data source definition for this case:

Data Sources

Create New Source

Odoo-source ☒ Enable Commit

Configuration

Driver: PostgreSQL +

JDBC URL: jdbc:postgresql://\${host}:5433/\${catalog}

Username: postgres Secret ☒

Secret Name: demoa/postgres

Test query: SELECT 1

Test Connection

Execute scripts

Connection Properties

Custom Property ☐ +

sslmode	disable	×
defaultHost	demoa.heimdalldata.com	×
extraFloatDigits	false	×
defaultCatalog	postgres	×
preferQueryMode	simple	×

There are two important parts here. First, is in the JDBC URL, it has **\${host}** filled in, and second, a parameter of "**defaultHost**" is also configured. The **defaultHost** is used when a host value is **NOT** provided, such as for health checks, SQL auth lookups, etc. When a host is provided by SQL Auth or a secrets manager, then this value is used instead. If you do a show pools command now, both hosts will show up, with different connections for each:

pg_hba *<odoo> Script X

show pools;

Results 1 X

show pools Enter a SQL expression to filter results (use Ctrl+Space)

	ABC URL or user:catalog	ABC busy	ABC connecting	ABC idle	ABC wait count	ABC created	ABC borrowed	ABC
1	127.0.0.1:5433/	1/1/90	0	0	0	1	64	0
2	odoo:odoo	1/0	0	0/0	0	1	64	0
3	demoa.heimdalldata.com:5433/	1/0/90	0	1	0	1	159	0
4	postgres:postgres	0/0	0	1/0	0	1	159	0

In the case of an AWS or similar environment, where you have an endpoint for readers and one for writers, you can use a partial hostname to support this case. Example, in Aurora you may have:

Endpoints (2)

Find resources	
	Endpoint name
	database-postgresql.cluster-cfjxl5jnvj49.us-east-1.rds.amazonaws.com
	database-postgresql.cluster-ro-cfjxl5jnvj49.us-east-1.rds.amazonaws.com

For this, the hostname in the JDBC URL could be:

- `${host}-cfjxl5jnvj49.us-east-1.rds.amazonaws.com` for the writer
- `${host}-ro-cfjxl5jnvj49.us-east-1.rds.amazonaws.com` for the reader.

In the "host" field in the authentication, you would then simply fill in "**database-postgresql.cluster**". As this is fixed for either the reader or the writer, it will create an appropriate hostname to support read/write split.

Active Directory/LDAP Authentication

Proxy Configuration

Local Proxy: ☒

Run as Service: ☐

Address Binding Type:

Any

Proxy Port(s):

5433

Max Heap Size (MB):

600

Proxy TLS Support:

☐

Authentication mode:

Active Directory/LDAP

LDAP Configuration:

ldap_active_directory_type

- [Bind+Search Mode](#)
- [Simple Mode](#)

If there's only one [LDAP Configuration](#), it will be automatically used by default for proxy authentication. However, if multiple LDAP Configurations are established, ensure to designate the appropriate one for each specific VDB within the Proxy Configuration section found in the [Virtual DB](#) tab.

Warning: For **MySQL LDAP** authentication, mysql clients must support the mysql_clear_password authentication plugin and ensure it is enabled during the connection.

Database Authentication Synchronization

Synchronize DB Authentication: ☒

Tokenized Authentication: ☐ ⓘ

Admin User:	admin	Secret ☐
Admin Password:	••••	☒
Sync Command:	select heimdall.sync_user('\${user}';\${password}';\${ldapgroups}');	
Sync Interval:	0	
Auth Cache Expiration Time	300000	

Available in below authentication modes:

- [SQL Driven](#)
- [Proxy Configured Users](#)
- [Active Directory/LDAP](#)

Synchronize DB Authentication option - specifies if synchronization of users and groups should be performed.

- **Admin User** - the username that will connect to the data source to perform the synchronization query
- **Admin Password** - the password that will be used for the user to connect to the data source to perform the synchronization query
- **Sync Command** - synchronization command, use \${user}, \${password} and \${ldapgroups} as placeholders.
- **Auth Cache Expiration Time** - determines how long (in ms) that data will stay in Authentication Cache. **Note:** The Authentication Cache can be clear manually in the [Status](#) tab.

When this feature is enabled, it allows the credentials that Heimdall has accepted to be synchronized into the database itself--there is no point to authenticate against the proxy unless the database *ALSO* allows the user to authenticate. In some cases, Heimdall and the DB can both authenticate against the same source. In other cases, the proxy must be able to synchronize the credentials into the database at runtime. For this to operate, credentials for a privileged user must be provided, along with a command that will allow the users to be synchronized.

Warning: If synchronization command contain the information about '{ldapgroups}' then for LDAP Authentication and LDAP Authentication inside SQL Driven Authentication it will be required to extract at least one group to authenticate the user.

Example for Postgres and Postgres derived databases such as Greenplum (excluding Redshift):

Synchronization query:

```
select heimdall.sync_user('${user}', '${password}', '${ldapgroups}');
```

The PL/pgSQL source of the "heimdall.sync_user" is available [here](#).

Example for Redshift:

Synchronization query:

```
CALL heimdall.sync_user('${user}', '${password}', '${ldapgroups}');
```

The PL/pgSQL source (adjusted for Redshift) of the "public.sync_user" is available [here](#).

Example for SQL Server:

Synchronization query:

```
EXECUTE heimdall.sync_user N'${user}', N'${password}', N'${ldapgroups}' [,  
TRUE | FALSE ]
```

[TRUE | FALSE] - An optional parameter, specifies whether roles should be created if missing and membership should be added to the user. When set to TRUE, roles are created and membership is added to the user. If set to FALSE, roles are not created, and membership is not added.

The example of Transact-SQL source of the "dbo.sync_user" is available [here](#).

Note: The source code of these functions/procedures is also stored in `/config/scripts` in heimdall installation directory (under name create-sync-user-procedure.sql) and can be modified if needed (it is advised to do so in separate files, named differently than the default ones, otherwise these might get overridden at startup). Please look at the [Execute scripts](#) option in the Data Sources tab.

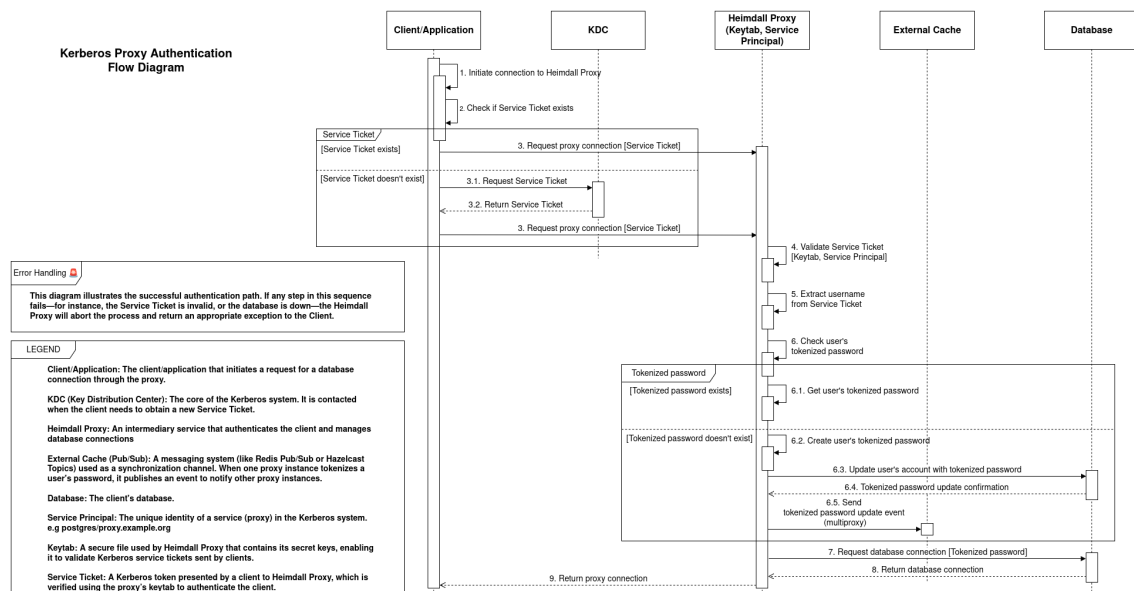
These functions will first ensure that the user's role is created. If it already exists, it will drop other role memberships and reset the password. Next, it will add all existing roles back to the user's role.

Kerberos/GSSAPI Authentication

Definitions:

- **Service Principal:** The unique identity of a service (proxy) in the Kerberos system. e.g postgres/proxy.example.org
- **Keytab Location:** Location to keytab file that stores secret keys for service principal (proxy).

Diagram



Once we set the authentication mode to Kerberos/GSSAPI, we will only need to fill in one field, which is the Keytab location. We will use that Keytab to identify as a service in the Kerberos realm. The service principal is as follows:

```
For PostgreSQL -> postgres/hostname
For SQLServer -> MSSQLSvc/hostname & MSSQLSvc/hostname:port
For MySQL -> mysql/hostname
```

Heimdall proxy will automatically retrieve the hostname from the system. Please ensure that the "hostname" command returns the fully qualified domain name of the host.

In case of using multiple proxies, the Keytab location will be common for all proxies.

IMPORTANT: When using Kerberos authentication mode with multiple proxies, it is essential to enable an external cache like Redis or Hazelcast. This ensures that all proxies are synchronized and Kerberos authentication works smoothly on all proxies.

NOTE: For MySQL Kerberos/GSSAPI authentication, mysql clients must support the `authentication_kerberos_client` authentication plugin and ensure it is enabled during the connection. Additionally seamless Kerberos authentication is available for any MySQL version. No enterprise edition is required.

Instructions on how to add a Heimdall proxy instance to Active Directory:

One way to do it is:

1. Add our instance to the DNS server.
2. Join the instance to Active Directory.
3. Instructions for Amazon Linux: https://docs.aws.amazon.com/directoryservice/latest/admin-guide/join_linux_instance.html
4. Set the service principal for the instance.
5. Go to Active Directory Users and Computers.
6. Find your instance in the Computers folder.
7. Right-click on it and go to properties.
8. Go to the Attribute Editor and find the attribute `servicePrincipalName`, and edit it.
9. Add 2 values: e.g.
 - `MSSQLSvc/heimdall-sqlserver-kerberos.ad.heimdalldata.com:1433`
 - `MSSQLSvc/heimdall-sqlserver-kerberos.ad.heimdalldata.com`
10. Apply the changes.
11. In order to update the keytab file, we need to leave our instance from the realm and join it again via `"realm leave domainName"` and `"realm join domainName"`
12. After rejoining our instance into the realm, we should have our `MSSQLSvc` service in our keytab file. We can verify it using the command `"klist -k keytab path"`. By default, the keytab is placed in `/etc/krb5.keytab`.
13. After that, we can authenticate using Kerberos or Windows authentication.

You can also enable synchronization if desired.

Load Balancing

Windows Active Directory

To demonstrate the correct setup for load balancing in an Active Directory (AD) environment with Kerberos authentication, let's use the following scenario as an example.

In this setup, we'll be operating with two proxy servers. The first proxy is hosted on ***hd-lb-proxy3.ad.heimdalldata.com***, and the second on ***hd-lb-proxy4.ad.heimdalldata.com***. These proxies are managed by an active load balancer designed to direct connection attempts to ***hd-lb-win.ad.heimdalldata.com*** towards one of these proxies, based on the current load balancing algorithm.

Configuring this environment to support Kerberos authentication involves a sequence of five essential steps:

1. Creation of an AD User for Proxy Service
2. Assignment of a Service Principal Name (SPN) to the New User
3. Generation of a Keytab for the SPN
4. Distribution of the Keytab Across Hosts
5. Configuration of SPN and Keytab in Heimdall

IMPORTANT !!!

- In every command shown, the realm should be in uppercase, e.g.,
AD.HEIMDALLDATA.COM
- If you have a CNAME for a load balancer, never use the alias name for the service principal name; use the canonical name instead. Connections using the alias will still work.

CREATION OF AN AD USER FOR PROXY SERVICE

Initiate by establishing an AD user account to symbolize our proxy service. This can be achieved either manually through the Active Directory Users and Computers interface or via the PowerShell command `New-ADUser`. Ensure the user's password is set to never expire and Kerberos encryption type AES256 is supported.

Command: `New-ADUser -Name "user_name" -UserPrincipalName "user_principal_name" -AccountPassword (ConvertTo-SecureString "user_password" -AsPlainText -Force) -Enabled $true -PasswordNeverExpires $true -KerberosEncryptionType AES256`

Example: `New-ADUser -Name "hdlb" -UserPrincipalName "hdlb@AD.HEIMDALLDATA.COM" -AccountPassword (ConvertTo-SecureString "user_password" -AsPlainText -Force) -Enabled $true -PasswordNeverExpires $true -KerberosEncryptionType AES256`

The screenshot shows the 'hdlb Service Properties' dialog box with the 'Account' tab selected. The 'User logon name' is 'hdlb' and the domain is '@ad.heimdalldata.com'. The 'User logon name (pre-Windows 2000)' is 'AD\' and 'hdlb'. The 'Account options' section shows 'Password never expires' checked. The 'Account expires' section shows 'Never' selected.

Published Certificates	Member Of	Password Replication	Dial-in	Object
Security	Environment	Sessions	Remote control	
Remote Desktop	Services Profile	COM+	Attribute Editor	
General	Address	Account	Profile	Telephones
				Organization

User logon name:
hdlb @ad.heimdalldata.com

User logon name (pre-Windows 2000):
AD\ hdlb

Logon Hours... Log On To...

☐ Unlock account

Account options:

- ☐ User must change password at next logon
- ☐ User cannot change password
- ☒ Password never expires
- ☐ Store password using reversible encryption

Account expires:
☒ Never
☐ End of: Thursday , March 7, 2024

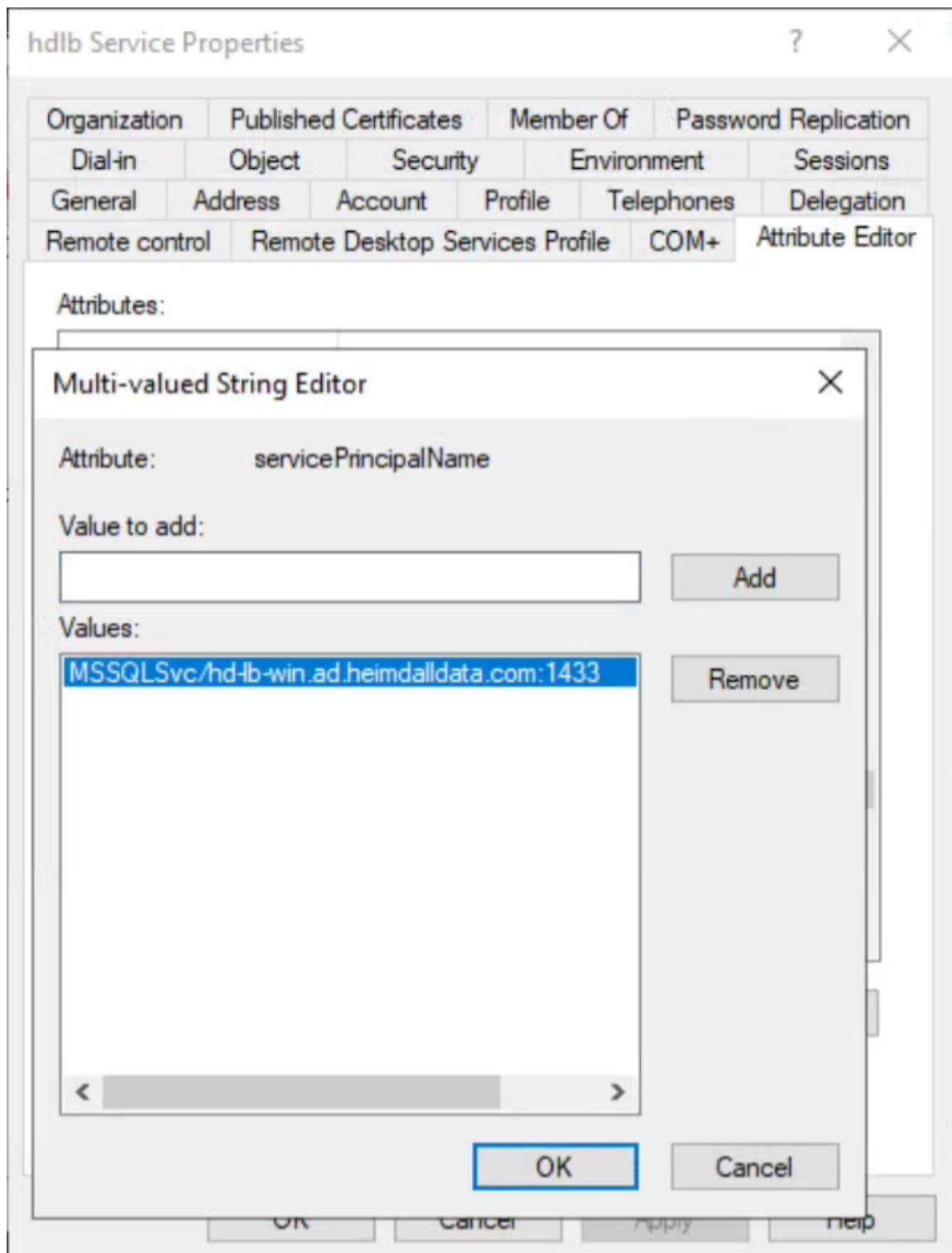
OK Cancel Apply Help

ASSIGNMENT OF A SERVICE PRINCIPAL NAME (SPN) TO THE NEW USER

The user created is **hdlb@ad.heimdalldata.com**. The next step involves assigning a SPN to this user: **MSSQLSvc/hd-lb-win.ad.heimdalldata.com:1433**, indicating the service name, load-

balanced domain, and target port, respectively. SPNs can be added manually via the "Attribute Editor" in the user's settings or through the setspn command.

```
Select Administrator: Windows PowerShell
PS C:\Users\Administrator> setspn -A MSSQLSvc/hd-lb-win.ad.heimdalldata.com:1433 hdlb
Checking domain DC=ad,DC=heimdalldata,DC=com
Registering ServicePrincipalNames for CN=hdlb Service,CN=Users,DC=ad,DC=heimdalldata,DC=com
MSSQLSvc/hd-lb-win.ad.heimdalldata.com:1433
Updated object
```



GENERATION OF A KEYTAB FOR THE SPN

After the user and SPN have been established, generate a keytab file containing this SPN using the ktpass command.

Command `ktpass -princ service/host:port@realm +rndPass -crypto ALL -ptype KRB5_NT_PRINCIPAL -mapuser mapped_user_principal -out keytab_path`

Example `ktpass -princ MSSQLSvc/hd-lb-win.ad.heimdalldata.com:1433@AD.HEIMDALLDATA.COM +rndPass -crypto ALL -ptype KRB5_NT_PRINCIPAL -mapuser hdlb@AD.HEIMDALLDATA.COM -out hdlb.keytab`

DISTRIBUTION OF THE KEYTAB ACROSS HOSTS

The generated keytab file must be distributed to every host that will run the proxy service to ensure seamless Kerberos authentication. The distribution method depends on the system's architecture and requirements. A simple method, for example, is to transfer the keytab using the SCP command. However, this approach may be challenging in an autoscaling environment. In such cases, for instance, within an AWS environment, the file can be stored in AWS Secrets and downloaded to a specified path when a new instance is created. This scenario is merely illustrative. It is crucial to ensure that the keytab file includes our SPN and is placed in a location specified in the Heimdall configuration (***Virtual Database/Our VDB/Proxy Configuration/Keytab Location***).

CONFIGURATION OF SPN AND KEYTAB IN HEIMDALL

Once the keytab file is distributed, it's crucial to update the Heimdall configuration. This includes specifying the Service Principal Name (SPN) we're using and the location of the keytab file. In our case, we've placed the keytab files at ***/etc/hdlb.keytab***, and the SPN designated for our setup is ***MSSQLSvc/hd-lb-win.ad.heimdalldata.com:1433***. Below is a screenshot of the Heimdall configuration reflecting these settings:

Proxy Configuration

Local Proxy:

☐

Address Binding Type:

Any

▼

Proxy Port(s):

1433

Proxy TLS Support:

☐

Certificate:

global_use_certificate

▼

Case Sensitive Usernames:

☐

Authentication mode:

Kerberos/GSSAPI

▼

Keytab Location:

/etc/hdlb.keytab

Service Principals:

MSSQLSvc/hd-lb-win.ad.hei

Add Service Principal

Auto-Scaling

Windows Active Directory

In an auto-scaling environment, new hosts must be automatically enrolled into the realm. To facilitate this, there is a base script available that can be customized for various purposes. You can download it here: [auto-enroll-kerberos-ad.sh](#)

Dual Authentication

The feature, available only in SQLServer Proxy, facilitates the use of Kerberos Authentication alongside passthrough authentication. However, there's a condition: users must adhere to a single authentication method and cannot switch between them interchangeably.

Dual Authentication can be enabled in VDB -> Proxy Configuration -> Dual Authentication Mode

Authentication Testing

Authentication Test

Test Authentication

Test User: user

Test Password: password

Test IP Address: 127.0.0.1

Test Database: db

UseSSL: ☐

Perform synchronization: ☐

Secret ☐



Variety of authentication options often requires confirmation that actual settings are working as expected. For checking if set authorization option would work correctly, with expected as correct credentials, should be used Authentication Test tab, which is located below settings of proxy authentication.

Authentication Test tab provides all required credentials inputs to test if configured proxy authentication would work as expected. Worth mentioning is that configured authentication doesn't have to be committed to perform an authentication test.

Warning: VDB configuration doesn't have to be committed, but all data sources, used by VDB, have to be committed. Any change in Data Source configuration should be followed by committing the change and refreshing page, before using Authentication Test. Only after these actions, Authentication Test results will be reliable.

Authentication testing is an option for checking if only proxy authentication confirms users credentials. Results returned by authentication testing doesn't confirm if given credentials match credentials in a database behind the proxy. If authentication testing returns failure as a result, then proper alert will be shown with description about reason of rejecting credentials.

Basic testing

Basic routine of authentication mainly expects two crucial inputs:

Input	Input name	Type	Description
username	Test User	String	Input for a name of the user
password	Test Password	String	Input for a password of the user

Authentication methods, which only based on these credentials, are:

- [Passthrough Authentication](#)
- [Proxy Configured Users Authentication](#)
- [Active Directory/LDAP Authentication](#)

For testing with these methods only credentials inputs: Test User and Test Password have to be set. Other test data inputs can be ignored.

Examples

PASSTHROUGH EXAMPLES

Passthrough by default passes all credentials, so every credential combination should return successful result.

PROXY CONFIGURED USERS EXAMPLES

Let's assume that we have two users declared for Proxy Configured Users method:

- `user1` with password `password1`
- `user2` with password `password2`

In this situation below inputs in Authentication Test should be returned as successful.

```
Test User: user1
Test Password: password1
Test Address IP: <empty or random>
Test Database: <empty or random>
UseSSL: <empty or random>
```

```
Test User: user2
Test Password: password2
Test Address IP: <empty or random>
Test Database: <empty or random>
UseSSL: <empty or random>
```

Other combinations of credentials will be returned as rejected by the proxy. Returned alert will contain information that provided credentials didn't match any of declared users, or given user password didn't match to declared password of the user.

LDAP EXAMPLES

Let's assume that our proxy configuration has been set properly (see section about LDAP Authentication for information about correct configuring) and configuration is pointing to a ldap server to read users only from `GROUP1`, which contains `user1` with password `password1` and `user2` with password `password2`. However, in the ldap server we have declared `GROUP2`,

where we have `user3` with password `password3` , but that user shouldn't be used for authentication by the proxy. In this situation below inputs in Authentication Test should be returned as successful.

```
Test User: user1
Test Password: password1
Test Address IP: <empty or random>
Test Database: <empty or random>
UseSSL: <empty or random>
```

```
Test User: user2
Test Password: password2
Test Address IP: <empty or random>
Test Database: <empty or random>
UseSSL: <empty or random>
```

Below credentials and others should be returned as rejected for the assumed situation.

```
Test User: user1
Test Password: password1
Test Address IP: <empty or random>
Test Database: <empty or random>
UseSSL: <empty or random>
```

Extended testing

Some proxy authentication methods requires more data than only username and password to authenticate a user. For actual authentication methods were added below data inputs:

Input	Input name	Type	Description
address	Test IP Address	String	Input for an address from which testing user is connecting to the proxy. Used for testing with SQL Driven Authentication for rules with specified address range.
database	Test Database	String	Input for a name of a database/schema name to which testing user wants to connect. Used for testing with SQL Driven Authentication for rules with specified database name.
ssl	UseSSL	Boolean	Input for a set if authentication process is made in a connection using SSL encryption. Used for testing with SQL Driven Authentication for rules with set connection type as "hostssl" or "hostnssl".

Proxy authentication methods, which uses more than basic data as credentials to authenticate users, are : - SQL Driven Authentication

SQL Driven Authentication Examples

Let's assume that we have defined a single rule for SQL Driven Authentication with below values:

```
connection type: hostssl
auth_method: trust
```

For assumed rule only below input for Authentication Test will pass. As can be seen, rule requires using SSL so all credentials with set using SSL as true will pass through proxy authentication.

```
Test User: <empty or random>
Test Password: <empty or random>
Test Address IP: <empty or random>
Test Database: <empty or random>
UseSSL: true
```

Let's assume that we have defined a single rule like:

```
address: 123.123.123.123
auth_method:trust
```

For assumed rule only below input for Authentication Test will pass. As can be seen, rule requires connection from address 123.123.123.123 so all credentials with set IP address as 123.123.123.123 will pass through proxy authentication.

```
Test User: <empty or random>
Test Password: <empty or random>
Test Address IP: 123.123.123.123
Test Database: <empty or random>
UseSSL: <empty or random>
```

Let's assume that we have defined a single rule like:

```
address: 123.123.123.0
netmask: 255.255.255.0
auth_method:trust
```

For assumed rule only below input for Authentication Test will pass. As can be seen, rule requires connection from address from network 123.123.123.0/24 so all credentials with set IP address from that network will pass through proxy authentication.

```
Test User: <empty or random>
Test Password: <empty or random>
Test Address IP: <any address from range 123.123.123.0-255>
Test Database: <empty or random>
UseSSL: <empty or random>
```

Let's assume that we have defined a single rule like:

```
database: {testdb1}
auth_method:trust
```

For assumed rule only below input for Authentication Test will pass. As can be seen, rule requires connection to database named `testdb1` so all credentials with set connection to database `testdb1` will pass through proxy authentication.

```
Test User: <empty or random>
Test Password: <empty or random>
Test Address IP: <empty or random>
Test Database: testdb1
UseSSL: <empty or random>
```

Let's assume that we have defined a single rule like:

```
connection type: hostssl
user_name: ldapuser1
database: testdb1
address: 127.0.0.0
netmask: 255.255.255.254
auth_method: ldap
options: ldapserver="ldap.example.com" ldapport=389 ldapprefix="CN="
ldapsuffix=",CN=Users, DC=example, DC=com"
```

Additionally, let's assume that we have declared `ldapuser1` with password `ldappassword1` in `CN=Users, DC=example, DC=com` on LDAP server `ldap.example.com:389`. For assumed rule only below input for Authentication Test will pass. Only below credentials will pass, because of strict rule definition and password stored on LDAP server.

```
Test User: ldapuser1
Test Password: ldappassword1
Test Address IP: <127.0.0.0 or 127.0.0.1>
Test Database: testdb1
UseSSL: true
```

Database Orchestration

Different database environments have a variety of tools to provide ways to promote and manage the databases, such as the MySQL Orchestrator. In order to integrate with these tools better, Heimdall has a simplified API call that allows the server state for a data source to be downloaded, adjusted, and uploaded. Examples:

To download the server configuration for a given data source:

```
wget --http-user=admin --http-password=<password> http://  
demoa.heimdalldata.com:8087/api/config/source/<source name>/servers
```

This will provide a file named "servers" with a format as follows (in json):

```
[  
  {  
    "name": "Primary",  
    "url": "jdbc:postgresql://pg.cfjxl5jnvj49.us-east-1.rds.amazonaws.com:  
5433/${catalog}",  
    "enabled": true,  
    "writeable": true,  
    "weight": 1,  
    "readWeight": 0,  
    "writeWeight": 0,  
    "active": true,  
    "failed": false,  
    "maxLag": 2147483647  
  },  
  {  
    "name": "Secondary test",  
    "url": "jdbc:postgresql://pg.cfjxl5jnvj49.us-east-1.rds.amazonaws.com:  
5433/${catalog}?readOnly\u003dtrue",  
    "enabled": true,  
    "writeable": false,  
    "weight": 8,  
    "readWeight": 0,  
    "writeWeight": 0,  
    "active": true,  
    "failed": false,  
    "maxLag": 2147483647  
  }  
]
```

This can then be edited as needed in the context of the events that have happened, and uploaded back to Heimdall with the following:

```
wget --http-user=admin --http-password=<password> --post-file=servers http://  
demoa.heimdalldata.com:8087/api/config/source/<source name>/servers/1 --  
header=Content-Type:application/json
```

Note: The "1" at the end of the URL represents a configuration version number, not a server ID, and should remain 1 in general.

Query Retry on Exception

Heimdall supports capturing and retrying queries based on the exceptions returned by the underlying JDBC driver. This allows a variety of conditions that would otherwise break an application's processing flow, and allow retry logic to kick in, potentially resolving the issue. This is most useful for transient errors, such as during a failover, a deadlock situation, etc. For example, in order to support a graceful failover of a MySQL cluster (including Aurora), the following rule can be used:

Edit Rule

Regex: Match all if empty

+

In-Transaction: ☐

Action: Retry

Notes: attempt to retry on a variety of failures experienced during a MySQL [failover](#)

Parameters

+

maxRetries15

exception.*(link failure|SQL Error \[([1290|S1000])\]|--read-only).*

retryDelay2

s

x

Done

This retry logic is not foolproof however—if in a transaction, retries should not be automatically performed, but should be resolved at the application level. It is impossible to cleanly retry all queries in a multi-query transaction, as one query can result from state that has changed since the transaction started. In auto-commit situations however, this retry logic can be useful to minimize disruption of the application, and to prevent connections from being dropped.

Routing for Distributed Clusters & Multi-Tenant SaaS

In a multi-tenant SaaS architecture, each client typically operates with their own dedicated database, schema, or cluster. When a user connects, it is necessary to dynamically route them to the correct database or cluster based on their assigned tenant. Heimdall Proxy addresses this challenge by replacing the default connection details with tenant-specific information, enabling seamless and secure routing to the appropriate destination.

Dynamic User-Based Routing with Heimdall Proxy

Heimdall Proxy enables dynamic user-based routing by determining the appropriate database or host for each user. This functionality can be implemented using one of the following authentication methods:

1. **SQL Driven:** Retrieves routing information dynamically from an SQL database.
2. **Proxy Configured Users Using Secrets:** Fetches routing details from a secret manager.

How It Works

1. For each user, corresponding data (such as the desired hostname or database to connect to) is stored in an external source (either an SQL table or a secret manager).
2. When a connection attempt is initiated, Heimdall automatically replaces the original host information with the destination details provided by the external source.
3. Using this substituted connection information, Heimdall establishes the appropriate connection to the target database or server.

This approach ensures that each user is routed dynamically to their designated host or database seamlessly.

The variables that are most commonly used for the JDBC URL for this purpose are `${host}` and `${database}` (or `${catalog}`), which are filled in from the SQL Auth or Secrets manager variables. When using these, also ensure the connection variables for `defaultHost` and `defaultCatalog` are also set. Please see the Theory section on [SQL Drive Authentication](#) for complete details. The

purpose of these variables is to create dynamic connection pools, which may exist on different cluster endpoints as well.

Additional Considerations

Setting up and configuring Heimdall Proxy for user-based routing involves complexities that may vary depending on your specific use case or client behavior. It is recommended to contact Heimdall support for assistance with the configuration process and to ensure the solution meets your unique requirements.

PostgreSQL Specific Information

PostgreSQL Proxy Restrictions

- PostgreSQL allows the user to send multiple statements in one query. Such query is treated as a transaction: if any of statements failed the query is roll backed entirely even in auto commit mode. The PostgreSQL proxy splits such queries to single statement queries and execute them separately via JDBC. This leads to the fact that if one of statements fails and auto commit mode is enabled all statements before it won't be roll backed. To use such a syntax, please use a true transaction start and commit to wrap the queries to avoid any issues.
- The vdb property of **suppressNoResult=true** is available. With Postgres, if an update query is executed via the executeQuery result, it will generate an exception on return saying "No results were returned by the query". In some frameworks, this is detected and suppressed when using the native Postgres driver, but not with the Heimdall driver. In order to work around this behavior, this option can trigger us returning a null instead of a resultset, which appears to allow the calling code to work fine. This applies when using Heimdall as a JDBC driver, but should not apply when as a proxy.

Postgres connection properties

The list of special properties can be found there: [postgres connection properties](#)

Prepared Statements

Postgres (and other databases) have the option to use prepared statements. When used however, this triggers "state" on the connection and this state will potentially result in sub-optimal behavior of Heimdall or other proxies. In particular, it will break multiplexing behavior, which is needed to dramatically reduce connection counts. In the case of Java applications, there is a parameter that can be set, "preferQueryMode". If set to "simple" it will effectively "de-prepare" the statements in a safe way, and allow multiplexing to work. One caution should be noted however: Many applications rely on an undocumented behavior of Postres with prepared statements where typecasts that are otherwise required may not be required when prepared

statements are used. If setting `preferQueryMode` to `simple` results in a typecast error, then this is effectively exposing a bug in the application code (or the framework generating the query).

Search_path support

When search paths are used, it is suggested that the search path be configured on the application user instead of dynamically changing the path at runtime. This ensures that a consistent path is used. If inconsistent search paths are used, this can break cache consistency and read/write split handling. To set the search path on a user, use the command:

```
alter user user1 set search_path='my_schema, "$user", public';
```

If this is not sufficient, or if overlapping table names are used across schemas, another resolution can be done to ensure read/write split and multiplexing works. In this case, you can add a JDBC URL parameter of `currentSchema=${schema}` to the URLs. This will fill in any value from the set `search_path` into the `currentSchema` value, and when we pull connections from the connection pools, we will use a connection connected with the proper value. You can then inspect the connections made with the "show pools" command.

Schema Assist for Dependencies

When using Postgres or a Postgres derived database, parsing is performed to detect view dependencies, in order to automate invalidation. This is only performed at proxy start at this time.

PG Notify Support

Postgres supports notification support, i.e. it provides a built-in pub/sub interface. When Postgres is used, this option can be explicitly controlled with a connection property of `"hdUsePGNotify"` and a value of `true` or `false`. To specify the database(s) to listen to, use `"hdNotifyDBs"` with a comma separated list of database names. Also, `*` can be used as a wildcard, with `-dbname` to remove from the list pulled from the database, i.e. `"*,-template0,-template1"` would be a solid starting point for configuring this feature. The catalogs are cached for up to one minute between attempts to poll, so if new catalogs are added, we will start listening for invalidations within a minute.

When used, this will override the need to have a grid cache for invalidations, although it does not automatically enable multi-primary Heimdall management nodes as a grid cache does. This also enables the below trigger invalidation that leverages the `pg_notify` interface. The

purpose of this option is to allow a simplified deployment without the use of an external cache, but still maintain cache coherency between multiple proxies.

Clear pool option

There is an option to clear a user pool even when it is impossible to get a connection, e.g. when all connections from the pool are busy, they cannot be closed and the maximum number of connections is reached. Two parameters can be set to issue a clear pool before making a connection.

- Application name - setting "ApplicationName" to "clear_pool" when attempting to connect will clear this user's user pool of all previous connections. To do so supply the connection URL adding at its end the following parameters: "?ApplicationName=clear_pool&assumeMinServerVersion=VERSION", where VERSION is the version of Postgres Server currently in use. VERSION has to be at least 9.0 for the JDBC driver to recognize custom application name parameters. Alternatively, use psql and provide a custom application name as "application_name=clear_pool" when attempting a connection.
- Custom option - provide clear_pool=true as an option by adding the following to the connection URL: "?options=clear_pool=true". The clear behavior is the same as it is when setting the application name to "clear_pool".

It's recommended to close those connections after connecting as reconnecting may occur which will clear the user's user pool again.

Trigger Invalidation with Postgres

Postgres provides a listen/notify interface that can be used to push invalidation messages directly from the database to the proxy. In order to configure this, use the following:

Connection property of `hdUsePGNotify`, value of `true`, then configure the following trigger function and trigger on each table:

```
CREATE OR REPLACE FUNCTION heimdall.changed()
RETURNS TRIGGER AS $$
BEGIN
    PERFORM pg_notify('heimdall', 'invalidate-trigger,'||
current_database()||'.'||TG_TABLE_SCHEMA||'.'||TG_TABLE_NAME||','||
((date_part('epoch', now())*1000)::bigint)::varchar);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- for each table on the system, perform the following:
```

```
CREATE TRIGGER trigger_table_changed
  BEFORE INSERT OR UPDATE OR DELETE ON test
  EXECUTE PROCEDURE heimdall.changed();
```

The result of this is near instant invalidation of tables, even if a write is done to PostgreSQL without going through Heimdall. Log messages with debug mode will indicate when this is functional.

When this is in place, this adjusts a few other behaviors. When active, invalidations will be issued through the DB notify interface vs. the grid cache as well. This enables local only cache configurations to effectively work without a grid cache. Additionally, if a table is invalidated via the grid cache with the above configuration, then additional notification messages will not be generated for that table by Heimdall, but other tables that have not been invalidated by the trigger will continue to have invalidations from Heimdall. This allows a mixed setup, so that frequently updated tables may have the trigger, while infrequently updated tables (or new tables) are invalidated directly. This will reduce the overhead of maintaining the triggers as the DB schema changes, while optimizing the number of invalidation messages needed.

This also supports invalidation of tables that are changed via stored procedures, other triggers, or are modified in other ways. It is still a requirement that stored procedures be tagged with the tables they are reading, but writes will be automatically accounted for.

Alternate Trigger Invalidation with Postgres (via polling)

This technique requires polling against the database, which adds load and latency to the invalidation process, but is documented for completion.

Connection properties:

- dbTriggerTracking=true
- dbChanges=select ts,name from heimdall.tableupdates where ts > @ts
- dbTimeMs=select (date_part('epoch', now())*1000)::bigint

Note: The following example is drafted for use with older Postgres, so doesn't use the ON CONFLICT syntax available in PG 9.5+

```
CREATE SCHEMA heimdall;

CREATE TABLE heimdall.tableupdates (
  name text PRIMARY KEY,
  ts bigint not null );

CREATE OR REPLACE FUNCTION heimdall.changed()
  RETURNS TRIGGER AS $$
BEGIN
  LOOP
```

```

        UPDATE heimdall.tableupdates SET ts=(date_part('epoch',
now())*1000)::bigint, name=current_database()||'.'||TG_TABLE_SCHEMA||'.'||
TG_TABLE_NAME;
        IF found THEN
            RETURN NEW;
        END IF;
        BEGIN
            INSERT INTO heimdall.tableupdates VALUES ((date_part('epoch',
now())*1000)::bigint, current_database()||'.'||TG_TABLE_SCHEMA||'.'||
TG_TABLE_NAME);
            RETURN NEW;
        EXCEPTION WHEN unique_violation THEN
            END;
        END LOOP;
        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trigger_table_changed
    BEFORE INSERT OR UPDATE OR DELETE ON test
    FOR EACH ROW
    EXECUTE PROCEDURE heimdall.changed();

```

Active Directory Password and Group Synchronization

```

CREATE OR REPLACE FUNCTION heimdall.sync_user(username text, password text,
ldapgroups text)
    RETURNS void
    LANGUAGE plpgsql
    AS $function$
    DECLARE
        sql          text;
        grouplist    text[];
        g            text;
        groupsToRevoke text[];
        lookupTableName text;
        lookupTableKey text; -- ldap_group
        lookupTableValue text; -- role
        lookupLdapGroups text[];
        lookupRoles text[];
        lg text;
        current_schema_name text;
        already_processed_group boolean;
        create_roles_if_missing boolean;
    BEGIN
        current_schema_name := current_schema();
        create_roles_if_missing := TRUE; -- set to false to prevent role creation
        lookupTableName := 'lookup_table';
        lookupTableKey := 'ldap_group';
        lookupTableValue := 'role';

        -- prevent syncing special users (if necessary)
        IF username IN ('api') THEN
            RAISE NOTICE '%.sync_user: skip syncing for special user "%"',

```



```

current_schema_name, username;
    RETURN;
END IF;

-- do not synchronize any user that is flagged as superuser
IF EXISTS(SELECT usesuper FROM pg_user WHERE user =
pg_catalog.quote_literal(username)) THEN
    RAISE NOTICE '%.sync_user: User % is superuser, not synchronizing',
current_schema_name, username;
    RETURN;
END IF;

IF EXISTS (
    SELECT 1
    FROM information_schema.tables
    WHERE table_schema = current_schema_name AND table_name =
lookupTableName
) THEN
    -- get lookupLdapGroups
    EXECUTE 'SELECT array_agg(lt.' || lookupTableKey || ') FROM ' ||
current_schema_name || '.' || lookupTableName || ' AS lt'
    INTO lookupLdapGroups;
    RAISE NOTICE '%.sync_user: Lookup table loaded: %',
current_schema_name, lookupLdapGroups;
END IF;

-- sync user and password
BEGIN
    sql := 'CREATE USER ' || quote_ident(username) || ' WITH PASSWORD '''
|| password || ''';';
    EXECUTE sql;
    sql := 'CREATE USER ' || quote_ident(username) || ' WITH PASSWORD '''
|| 'REDACTED' || ''';';
    RAISE NOTICE '%.sync_user: % created', current_schema_name, sql;
EXCEPTION
    WHEN Duplicate_Object THEN -- alter password when user exists
        sql := 'ALTER USER ' || quote_ident(username) || ' WITH PASSWORD
''' || password || ''';';
        BEGIN
            EXECUTE sql;
            sql := 'ALTER USER ' || quote_ident(username) || ' WITH
PASSWORD ''' || 'REDACTED' || ''';';
            RAISE NOTICE '%.sync_user: % updated', current_schema_name,
sql;
        EXCEPTION
            WHEN SQLSTATE 'XX000' THEN
                IF SQLERRM LIKE '%tuple concurrently updated%' THEN
                    -- ignore, may occur as we use connection pooling for
synchronization
                ELSE
                    RAISE;
                END IF;
            END;
        END;

-- sync group/role membership
grouplist := string_to_array(ldapgroups, ',');

```

```

        groupsToRevoke := array(SELECT b.rolname
                                FROM (SELECT rolname, oid
                                        FROM pg_roles
                                        WHERE rolname != ALL(grouplist) EXCEPT
SELECT username, usesysid
                                FROM
pg_user) b
                                WHERE
pg_has_role(username, b.oid, 'member'));
        IF array_length(groupsToRevoke, 1) > 0 THEN
            FOR i IN array_lower(groupsToRevoke, 1) .. array_upper(groupsToRevoke,
1) LOOP
                g := groupsToRevoke[i];
                IF position('= ' in g) > 0 THEN -- capture the name if ldapgroups
is passed in as cn=groupname
                    g := trim(TRAILING '' FROM split_part(groupsToRevoke[i], '=',
2));
                ELSE
                    g := trim(BOTH '' FROM trim(BOTH FROM groupsToRevoke[i]));
                END IF;

                -- now revoke membership of the role from the user
                sql := 'REVOKE ' || quote_ident(g) || ' FROM ' ||
quote_ident(username) || ';';
                RAISE NOTICE '%.sync_user: %', current_schema_name, sql;
                IF sql IS NOT NULL THEN
                    BEGIN
                        EXECUTE sql;
                    EXCEPTION
                        WHEN SQLSTATE 'XX000' THEN
                            IF SQLERRM LIKE '%tuple concurrently updated%' THEN
                                -- ignore, may occur as we use connection pooling
for synchronization
                            ELSE
                                RAISE;
                            END IF;
                        END;
                    END IF;
                END LOOP;
            END IF;

            IF array_length(grouplist, 1) > 0 then
                FOR i IN array_lower(grouplist, 1) .. array_upper(grouplist, 1) LOOP
                    g := grouplist[i];
                    already_processed_group := false;
                    RAISE NOTICE '%.sync_user: processing group: %',
current_schema_name, g;
                    IF position('= ' in g) > 0 THEN -- capture the name if ldapgroups
is passed in as cn=groupname
                        g := trim(TRAILING '' FROM split_part(grouplist[i], '=', 2));
                    ELSE
                        g := trim(BOTH '' FROM trim(BOTH FROM grouplist[i]));
                    END IF;

                    -- handle additional ldap_group -> role(s) aliases
                    IF array_length(lookupLdapGroups, 1) > 0 then
                        FOR lg_i IN array_lower(lookupLdapGroups, 1) ..

```

```

array_upper(lookupLdapGroups, 1) LOOP
    lg := lookupLdapGroups[lg_i];
    RAISE NOTICE '%.sync_user: processing group: % vs.
lookupGroup: %', current_schema_name, g, lg;
    IF lg = g THEN
        -- get all roles mapped to given ldap_group
        EXECUTE 'SELECT array_agg(lt.' || lookupTableValue ||
') FROM ' || current_schema_name || '.' || lookupTableName || ' AS lt WHERE
lt.' || lookupTableKey || ' LIKE ' || quote_literal(lg)
        INTO lookupRoles;

        RAISE NOTICE '%.sync_user: ldap_group % mappings: %',
current_schema_name, lg, lookupRoles;

        -- if role is an empty string then no role is mapped for
given ldap_group
        IF array_length(lookupRoles, 1) > 0 THEN
            -- process all roles mapped to given ldap_group
            FOR r_i IN array_lower(lookupRoles, 1) ..
array_upper(lookupRoles, 1) LOOP
                g := lookupRoles[r_i];
                IF g IS NOT NULL AND g <> '' THEN
                    IF create_roles_if_missing THEN
                        BEGIN
                            sql := 'CREATE ROLE ' || quote_ident(g);
                            EXECUTE sql;
                            RAISE NOTICE '%.sync_user: %',
current_schema_name, sql;

                            EXCEPTION
                                WHEN Duplicate_Object THEN
                                    RAISE NOTICE '%.sync_user: role %
already exists', current_schema_name, g;
                        END;
                    END IF;
                    -- now grant membership of the role to the user
                    BEGIN
                        sql := 'GRANT ' || quote_ident(g) || ' TO ' ||
quote_ident(username) || ';';
                        EXECUTE sql;
                        RAISE NOTICE '%.sync_user: %',
current_schema_name, sql;

                        EXCEPTION
                            WHEN SQLSTATE '42704' THEN
                                RAISE NOTICE '%.sync_user: role % does not
exist', current_schema_name, g;
                            END;
                        already_processed_group := true;
                    END IF;
                END LOOP;
            END IF;
        END IF;
    END LOOP;

    --lookup_table does not exist
    IF g IS NOT NULL AND g <> '' AND already_processed_group = FALSE
THEN

```

```

        -- make sure groups are created
        IF create_roles_if_missing THEN
            BEGIN
                sql := 'CREATE ROLE ' || quote_ident(g);
                EXECUTE sql;
                RAISE NOTICE '%.sync_user: %', current_schema_name,
sql;
            EXCEPTION
                WHEN Duplicate_Object THEN
                    RAISE NOTICE '%.sync_user: role % already exists',
current_schema_name, g;
            END;
        END IF;

        BEGIN
            -- now grant membership of the role to the user
            sql := 'GRANT ' || quote_ident(g) || ' TO ' ||
quote_ident(username) || ';';
            EXECUTE sql;
            RAISE NOTICE '%.sync_user: %', current_schema_name, sql;
        EXCEPTION
            WHEN SQLSTATE '42704' THEN
                RAISE NOTICE '%.sync_user: role % does not exist',
current_schema_name, g;
            END;
        END IF;

    END LOOP;
END IF;
RETURN;
END;
$function$
;

```

Active Directory Password and Group Synchronization extra options

By default, a role with the same name is created in the database for each extracted ldap group. By creating the table below for each key found (ldap_group), a role with the specified name in the value (role) will be created instead. Composite Primary Key allows to map more than one role to particular ldap_group; If 'role' is empty then no role is created and granted to user for this particular ldap_group.

```

CREATE TABLE IF NOT EXISTS heimdall.lookup_table (
    ldap_group varchar(255),
    role varchar(64),
    PRIMARY KEY (ldap_group, role)
)

```

MySQL Specific Information

Postgres connection properties

The list of special properties can be found there: [MySQL connection properties](#)

Common Issues

When testing with the mysql commandline client, use -A to connect. This avoids an issue where the client is itself using a deprecated feature the MySQL protocol that can trigger issues in a proxy environment.

One of the most common issues with MySQL is the character-set being used by the database. As JDBC attempts to negotiate with the server, if the server is set to anything other than UTF8MB4 (along with the tables) this can cause issues with certain characters not being handled properly. Here is an example configuration for the my.cnf that can help avoid any such issues:

```
[mysqld]
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci

[mysql]
default-character-set = utf8mb4
```

In addition to this however, it is also important to set the character set of the database itself to utfmb4:

```
ALTER DATABASE <dbname> DEFAULT COLLATE utf8_unicode_ci;
```

If all of these are not in alignment, then various character encoding issues may occur.

One further impact--if a database is set to utf8 instead of utf8mb4, most functions may appear to work properly, BUT it can impact performance. If an index is done on a column that is coded in utf8, but the query is received in utf8mb4, then the index may not be used as part of the optimization. This can be a very subtle and difficult issue to track down, but using "explain" on the query will show the difference in the execution plan with and without heimdall, but it will not say why the index is not used.

Several MySQL specific behaviors are included and detected by Heimdall, such as:

SELECT SQL_CALC_FOUND_ROWS: queries with this pattern will be treated as updates.

Using the "getBoolean()" method on a string type will look at the first character and match per the logic used in the MySQL driver, i.e. any string starting with "y" will test as "yes" or true. Same with "t".

Known Incompatibility: There is a rare but known issue when using non-prepared statements when inserting binary data using an insert command. In this configuration, the insert uses inline binary data in what is otherwise a UTF-8 string. When received, the entire string is processed as UTF-8, and results in the binary data being modified when transmitted from the proxy to the database. This problem is NOT observed if using true server-side prepared statements for the insert. If you observe issues where binary data appears corrupted, and you believe you are using a prepared statement, please check if your driver is implicitly converting prepared statements into "client" side prepared statements, as most DO do this for performance reasons. For example, with the MySQL JDBC driver, you will need to add the option "useServerPrepStmts=true". Please see dev.mysql.com for more information.

To resolve this issue, you can enable the "rewriteBinaryQueryToHexLiteral" vdb option in the Heimdall proxy configuration. This will automatically detect and convert binary data into hexadecimal literals, ensuring safe insertion without corruption. More information on that option can be found on the [configuration](#) page for more details.

sql_mode

If a custom sql mode is needed, it can be set in the data source properties with the key of "sessionVariables" and a value of "sql_mode='whatever'".

MariaDB

Due to changes in the version string and how the Oracle MySQL driver interacts with the MySQL server, when using MariaDB, it is advised to set it to use a version string such as "5.7.19-10.2.22-MariaDB". The first part is the MySQL compatibility version—if this string is higher than either 5.6.19, or 8.0.3, then this may trigger an error about an invalid counter of "transaction_isolation". Please see the [MariaDB documentation](#) for information on how to change this setting.

Trigger Invalidation with MySQL

Connection properties:

- dbTriggerTracking=true
- dbChanges=select ts,name from heimdall.table_change where ts > @ts
- dbTimeMs=select get_time_ms()

Script to configure MySQL for Trigger expiry:

```
#!/bin/bash

database=tpch
host=127.0.0.1
port=3306
user=root
password=secret
command="mysql -N --force --host=$host --port=$port --user=$user --
password=$password"

(
# setup the heimdall table for all Heimdall activities
echo "DROP DATABASE IF EXISTS heimdall;
CREATE DATABASE heimdall;
USE heimdall;

DELIMITER \\\$\\$

#-----code for MySQL (including pre-5.6 versions) to get the current time
to the ms level-----
DROP FUNCTION IF EXISTS get_time_ms\\$\\$
CREATE DEFINER=root@localhost FUNCTION get_time_ms()
    RETURNS BIGINT
    READS SQL DATA
    NOT DETERMINISTIC
BEGIN
    DECLARE time_ms BIGINT;
    SELECT conv(
        concat(
            substring(uid,16,3),
            substring(uid,10,4),
            substring(uid,1,8)),16,10) div 10000 - (141427 * 24 * 60 * 60 * 1000)
        into time_ms FROM (select uuid() uid) as alias;
    RETURN time_ms;
END\\$\\$

#-----code for trigger based expiry-----
CREATE TABLE table_change (
    ts BIGINT DEFAULT 0,
    name VARCHAR(64) CHARACTER SET utf8 COLLATE utf8_bin,
    updates INT DEFAULT 0,
    inserts INT DEFAULT 0,
    deletes INT DEFAULT 0,
    PRIMARY KEY (name)) ENGINE = MEMORY\\$\\$

DROP PROCEDURE IF EXISTS changes\\$\\$
CREATE DEFINER=root@localhost PROCEDURE changes(time BIGINT)
BEGIN
    SELECT ts,name,updates,inserts,deletes from table_change where ts >= time;
END\\$\\$

DROP PROCEDURE IF EXISTS track_changes\\$\\$
CREATE DEFINER=root@localhost PROCEDURE track_changes(table_name VARCHAR(64),
type VARCHAR(16))
```

```

BEGIN
    CASE type
        WHEN 'update' THEN INSERT INTO heimdall.table_change VALUES
(get_time_ms(), table_name, 1, 0, 0) ON DUPLICATE KEY
            UPDATE updates=updates+1,ts=get_time_ms();
        WHEN 'insert' THEN INSERT INTO heimdall.table_change VALUES
(get_time_ms(), table_name, 0, 1, 0) ON DUPLICATE KEY
            UPDATE inserts=inserts+1,ts=get_time_ms();
        WHEN 'delete' THEN INSERT INTO heimdall.table_change VALUES
(get_time_ms(), table_name, 0, 0, 1) ON DUPLICATE KEY
            UPDATE deletes=deletes+1,ts=get_time_ms();
    END CASE;
END\$\$
USE \$database\$\$
DROP USER 'heimdall'@'%' \$\$
CREATE USER 'heimdall'@'%' IDENTIFIED BY 'heimdalldemo' \$\$
GRANT EXECUTE on heimdall.* to 'heimdall'@'%' \$\$
GRANT SELECT on tpch.* to 'heimdall'@'%' \$\$
FLUSH PRIVILEGES \$\$
"

echo "show tables" | \$command \$database | while read table; do

if [ "\$table" != "table_change" -a "n\$table" != "n" ]; then
    echo "INSERT INTO heimdall.table_change (ts,name) values
(UNIX_TIMESTAMP()*1000,'${database}.${table}') \$\$"

        echo "DROP TRIGGER IF EXISTS ${database}.${table}_update \$\$
CREATE TRIGGER ${database}.${table}_update AFTER UPDATE ON \$
{database}.${table} FOR EACH ROW
    CALL heimdall.track_changes('${database}.${table}', 'update') \$\$"

        echo "DROP TRIGGER IF EXISTS ${database}.${table}_insert \$\$
CREATE TRIGGER ${database}.${table}_insert AFTER INSERT ON \$
{database}.${table} FOR EACH ROW
    CALL heimdall.track_changes('${database}.${table}', 'insert') \$\$"

        echo "DROP TRIGGER IF EXISTS ${database}.${table}_delete \$\$
CREATE TRIGGER ${database}.${table}_delete AFTER DELETE ON \$
{database}.${table} FOR EACH ROW
    CALL track_changes('${database}.${table}', 'delete') \$\$"
fi

done )| \$command

```


Microsoft SQL Server Specific Information

- Currently, the SQL Server proxy only supports SQL Server logins. If the proxy is running on Windows, it is possible to configure internal authentication on the SQL Server JDBC driver. Please contact support for more details if this is necessary.
- When using the Microsoft Azure Database, the `setCatalog()` function is not usable to change the catalog on a connection. To support an application selecting the database on the initial connection, use the string `${database}` in the JDBC URL to set the database, i.e. `;"databaseName=${database}"` at the end of the url. The value `${database}` will be extracted from the client login to populate this value.
- When performing explain extraction, the option `checkTableMismatch` can be used (any value) to trigger testing and printing if a the tables extracted from the explain match what is used internally by Heimdall. This can be used to track down table extraction issues.
- If using Mirroring (not Always-On) for redundancy, transactions will fail. This can apply if using SQL Server in AWS for versions prior to 2016, or older instances spun up before May 2019 with failover. Please use Always-On, as transaction handling with mirroring is not easily supported.
- Applications that depend on session recovery are likely to not work properly, in particular if session data is used as part of record/table locking mechanisms, such as the SP `"applock_mode"` with the lock owner of `"session"`. One example of a library that uses this is `"Hangfire"`.
- Applications that require MARS (multiple active result sets) may not work properly. The proxy will actively negotiate this option to be off with clients, which normally is sufficient to

allow the applications to work. In some rare cases however, code needs to be modified to read one full result-set into memory before opening a new result-set.

SQL Server connection properties

The list of special properties can be found there: [SQL Server connection properties](#)

Schema Assist for Dependencies

When using SQL Server, parsing is performed to detect view dependencies along with fixed (not dynamic) dependencies in stored procedures, in order to automate invalidation. This is only performed at proxy start at this time.

Trigger Invalidation with SQL Server

Connection properties:

- dbTriggerTracking=true
- dbChanges=select ts,name from heimdall.table_change where lastupdate > @ts
- dbTimeMs=select heimdall.get_time_ms()

```
CREATE DATABASE [heimdall]
GO

/** create the trigger tracking table */
USE [heimdall]
CREATE TABLE [dbo].[table_change](
    [ts] [bigint] NULL,
    [name] [varchar](64) NOT NULL,
    CONSTRAINT [PK_table_change] PRIMARY KEY ([name]));

CREATE FUNCTION [dbo].[get_time_ms]() RETURNS BIGINT
AS
BEGIN
    DECLARE @timems BIGINT
    set @timems= (SELECT cast(DATEDIFF(s, '1970-01-01 00:00:00', GETUTCDATE()) as
    BIGINT)*1000+cast(DATEDIFF(ms, CONVERT(VARCHAR, GETUTCDATE(), 120), GETUTCDATE())
    as BIGINT))
    return @timems
END

/** Create the tracking procedure, which will be used by the triggers */
CREATE OR ALTER PROCEDURE [heimdall].[dbo].[track_changes]
    @tablename NVARCHAR(128)
as
    SET NOCOUNT OFF
    UPDATE [heimdall].[dbo].[table_change] SET ts=[heimdall].
```

```

[dbo].get_time_ms() WHERE name = @tablename
    IF @@ROWCOUNT != 0
        RETURN
    INSERT INTO [heimdall].[dbo].[table_change] VALUES ( dbo.get_time_ms(),
@tablename )
GO

/** trigger to apply to each table that needs trigger based eviction */
/**** for table vevo.dbo.Product ****/
USE [vevo2]; CREATE OR ALTER TRIGGER [dbo].[Product_changed]
    ON [vevo2].[dbo].[Product]
    AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    exec [heimdall].[dbo].[track_changes] 'vevo2.dbo.Product'
END
GO

```

Active Directory Password and Group Synchronization

```

CREATE OR ALTER PROCEDURE heimdall.sync_user
(
    @username nvarchar(max),
    @password nvarchar(max),
    @ldapgroups nvarchar(max),
    @createRolesIfMissing BIT = 1 -- Optional parameter, if set to 1 roles are
created and membership is added to the user
)
AS
BEGIN
    --Logic variables
    DECLARE @sql nvarchar(max);
    DECLARE @roleName nvarchar(max);
    DECLARE @currentRow INT = 1;
    DECLARE @rowCount INT;
    CREATE TABLE #groupList (id INT IDENTITY(1,1), g varchar(max)); --table
variable can't be used in dynamic SQL
    CREATE TABLE #groupsToRevoke (id INT IDENTITY(1,1), g varchar(max));
    CREATE TABLE #oneToManyMappings (id INT IDENTITY(1,1), g varchar(max));

    --Configurable variables
    DECLARE @currentSchemaName varchar(max) = 'heimdall';
    DECLARE @lookupTableName varchar(max) = 'lookup_table';
    DECLARE @lookupTableKey varchar(max) = 'ldap_group';
    DECLARE @lookupTableValue varchar(max) = 'role_name';

    -- Put ldapgroups to table
    INSERT INTO #groupList (g)
    SELECT LTRIM(RTRIM(value))
    FROM STRING_SPLIT(@ldapgroups, ',');

    -- Check if lookup_table exists
    IF (EXISTS (SELECT *
                FROM INFORMATION_SCHEMA.TABLES

```

```

        WHERE TABLE_SCHEMA = @currentSchemaName
        AND TABLE_NAME = @lookupTableName))
BEGIN
    -- One to many mappings of ldap_groups
    SET @sql = CONCAT(
        'INSERT INTO #oneToManyMappings (g)
        SELECT lt.', QUOTENAME(@lookupTableKey),
        ' FROM ', QUOTENAME(@lookupTableName), ' lt
        INNER JOIN #groupList gl ON lt.', QUOTENAME(@lookupTableKey), ' =
gl.g
        GROUP BY lt.', QUOTENAME(@lookupTableKey),
        ' HAVING COUNT(1) > 1;');
    EXEC sp_executesql @sql;
    PRINT @sql;

    -- Delete groups which has one to many mapping
    SET @sql = 'DELETE t1 FROM #groupList t1 JOIN #oneToManyMappings t2 ON
t1.g = t2.g'
    EXEC sp_executesql @sql;
    PRINT @sql;

    -- Insert all one to many mappings (corresponding roles)
    SET @sql = CONCAT(
        'INSERT INTO #groupList
        SELECT ', QUOTENAME(@lookupTableValue),
        ' FROM ', QUOTENAME(@lookupTableName),
        ' lt INNER JOIN #oneToManyMappings otmm ON lt.',
QUOTENAME(@lookupTableKey), ' = otmm.g;');
    EXEC sp_executesql @sql;
    PRINT @sql

    -- Replace ldap groups with it's corresponding role names from
lookup_table
    SET @sql = CONCAT(
        'UPDATE gl SET gl.g = lt.', QUOTENAME(@lookupTableValue),
        ' FROM #groupList gl JOIN ',
        QUOTENAME(@currentSchemaName), '.', QUOTENAME(@lookupTableName), ' lt
ON gl.g = lt.', QUOTENAME(@lookupTableKey));
    EXEC sp_executesql @sql;
    PRINT @sql;
END

-- Create login if not exists, alter otherwise
IF NOT EXISTS
    (SELECT name
    FROM master.sys.server_principals
    WHERE name = @username)
BEGIN
    SET @sql = CONCAT('CREATE LOGIN ', QUOTENAME(@username), ' WITH
PASSWORD = ', QUOTENAME(@password, ''));
    EXEC sp_executesql @sql;
    PRINT @sql;
END
ELSE
BEGIN
    SET @sql = CONCAT('ALTER LOGIN ', QUOTENAME(@username), ' WITH
PASSWORD = ', QUOTENAME(@password, ''));

```

```

        EXEC sp_executesql @sql;
        PRINT @sql;
    END

    -- Create user if not exists, alter otherwise
    IF NOT EXISTS
        (SELECT 1
         FROM sys.sysusers
         WHERE name = @username)
    BEGIN
        SET @sql = CONCAT('CREATE USER ', QUOTENAME(@username), ' FOR LOGIN ',
        QUOTENAME(@username));
        EXEC sp_executesql @sql;
        PRINT @sql;
    END
    ELSE
    BEGIN
        SET @sql = CONCAT('ALTER USER ', QUOTENAME(@username), ' WITH LOGIN =
        ', QUOTENAME(@username));
        EXEC sp_executesql @sql;
        PRINT @sql;
    END

    -- Get roles which user is not a member of anymore
    INSERT INTO #groupsToRevoke (g)
    (
        SELECT r.name
        FROM sys.database_role_members drm
             JOIN sys.database_principals r ON drm.role_principal_id =
r.principal_id
             JOIN sys.database_principals u ON drm.member_principal_id =
u.principal_id
        WHERE u.name = @username
        EXCEPT
        SELECT g
        FROM #groupList
    );

    SET @rowCount = (SELECT COUNT(*) FROM #groupsToRevoke);

    -- Drop membership of roles which user is not a member of anymore
    WHILE @currentRow <= @rowCount
    BEGIN
        SET @roleName = (SELECT g FROM #groupsToRevoke WHERE id =
@currentRow);
        SET @sql = CONCAT('ALTER ROLE ', QUOTENAME(@roleName), ' DROP MEMBER
        ', QUOTENAME(@username));
        EXEC sp_executesql @sql;
        PRINT @sql;

        SET @currentRow += 1;
    END

    SET @currentRow = 1;
    SET @rowCount = (SELECT MAX(id) FROM #groupList)
    -- Iterate over groupList and create roles
    WHILE @currentRow <= @rowCount

```

```

BEGIN
    SET @roleName = (SELECT g FROM #groupList WHERE id = @currentRow);

    IF @roleName IS NOT NULL AND @roleName <> '' AND @createRolesIfMissing
= 1 -- If 'roleName' is empty then no role is created and granted to user
    BEGIN
        -- Create role if not exists
        IF NOT EXISTS (SELECT 1 FROM sys.database_principals WHERE name =
@roleName AND type = 'R')
            BEGIN
                SET @sql = CONCAT('CREATE ROLE ', QUOTENAME(@roleName));
                EXEC sp_executesql @sql;
                PRINT @sql;
            END

        -- Grant role to the user
        SET @sql = CONCAT('ALTER ROLE ', QUOTENAME(@roleName), ' ADD
MEMBER ', QUOTENAME(@username));
        EXEC sp_executesql @sql; -- Won't throw an exception if user
is already a member of the given role
        PRINT @sql;
    END

    SET @currentRow += 1;
END
-- Make sure we drop these to prevent reuse issues on the same connection
DROP TABLE #groupList;
DROP TABLE #groupsToRevoke;
END

```

Active Directory Password and Group Synchronization extra options

By default, a role with the same name is created in the database for each extracted ldap group. By creating the table below for each key found (ldap_group), a role with the specified name in the value (role) will be created instead. Composite Primary Key allows to map more than one role to particular ldap_group; If 'role' is empty then no role is created and granted to user for this particular ldap_group.

```

CREATE TABLE dbo.lookup_table (
    ldap_group varchar(255),
    role_name varchar(64),
    PRIMARY KEY (ldap_group, role_name)
)

```

Amazon RDS for SQLServer

Adjustments to the 'master' database are presently limited within Amazon RDS for SQL Server. When the 'master' database is configured in the datasource or left unspecified, certain

functionalities, such as LDAP synchronization, may not work properly. To resolve this issue, clients are recommended to specify the default database in the Datasource tab of the JDBC URL. A representative JDBC URL format is demonstrated as follows: "jdbc:sqlserver://{server_name}:1433;databaseName=\${database}", then i properties define defaultCatalog with created database name, in our case [testDB].

If we would to set up a proxy with a new user on Amazon RDS for SQLServer, we can use the following example: Connect with admin user to database master. Then execute queries similar to those:

```
CREATE DATABASE [testDB];
USE [testDB];
CREATE LOGIN [test_user] WITH PASSWORD = 'test_user',
DEFAULT_DATABASE=[testDB];
CREATE USER [test_user] FOR LOGIN [test_user];
CREATE SCHEMA heimdall;
GRANT ALTER, CONTROL, EXECUTE, SELECT, INSERT, UPDATE, DELETE
    TO [test_user];
GRANT CREATE TABLE TO [test_user];
GRANT CREATE PROCEDURE TO [test_user];
```

While using ldap in VDB the command for syncing user will look like this:

```
EXECUTE testDB.heimdall.sync_user N'${user}', N'${password}', N'${ldapgroups}'
```

For using load-balancing features we will need to run:

```
GRANT VIEW DATABASE STATE TO [test_user];
GRANT VIEW SERVER STATE TO [test_user]; -- has to be run on master database
```

Database name, and username should be adjusted to the clients need. At the end, created user should be placed in Datasource tab, and JDBC URL should point into defined database.

AWS Redshift

Redshift connection properties

The list of special properties can be found there: [redshift connection properties](#)

Redshift Drivers

In some situations, you may find that the Heimdall management console can test successfully to Redshift, but the proxy does not work properly. If you inspect the logs, you may see the console is reporting a 403 exception. This is due to the proxy trying to load additional jar files that normally are contained from the Redshift zip file.

To resolve this, first extract all the jar files from the Redshift zip file, which contains all the jar dependencies for the Redshift driver, including the AWS SDK. Next, in the driver configuration, add all the jar files to the driver config. This may be over ten files. Once added, make sure the proxy restarts, and at this point you should see this error resolved.

Redshift and SSL

Redshift by default connects via TLS and requires TLS including CA validation. When connecting to the Heimdall proxy for Redshift, it is likely that initially, the self-signed certificate will be presented, AND TLS will need to be enabled. To resolve this, make sure the Redshift driver is updated, and use the sslmode option to "prefer" for testing. For production, it is advised that a correct and valid TLS certificate be loaded via the admin->certificate tab, and configured in the proxy tls settings. Older Redshift drivers will not accept the prefer setting.

Redshift and Heimdall

The Heimdall Database proxy is as of June 26, 2023, the only proxy vendor supported as a Redshift Ready product, and we will continue working with the Redshift team to improve compatibility and performance.

Active Directory Password and Group Synchronization

```
CREATE OR REPLACE PROCEDURE heimdall.sync_user(username VARCHAR, password
VARCHAR, ldapgroups VARCHAR(max))
```



```

LANGUAGE plpgsql
AS $$

DECLARE
    -- configuration variables
    current_schema_name TEXT;
    create_roles_if_missing BOOLEAN;
    lookupTableName TEXT;
    lookupTableKey TEXT; -- ldap_group
    lookupTableValue TEXT; -- role

    -- logic variables
    sql VARCHAR(max);
    groupname RECORD;
    has_rows BOOLEAN;
    lookup_table_exists BOOLEAN;
    lookup_table_has_rows BOOLEAN;
    user_exists BOOLEAN;
    group_exists BOOLEAN;

BEGIN
    -- set these variables to match your config
    current_schema_name := 'heimdall'; -- where the sync_user and lookup
table live
    create_roles_if_missing := true; -- set to true to create roles when they
don't exist
    lookupTableName := 'lookup_table';
    lookupTableKey := 'ldap_group';
    lookupTableValue := 'db_group';

    EXECUTE 'CREATE TEMPORARY TABLE grouplist (g varchar(max))';
    EXECUTE 'CREATE TEMPORARY TABLE finalgroups (g varchar(max))';

    CREATE TEMPORARY TABLE numbers AS (
        SELECT ROW_NUMBER() OVER() AS n
        FROM
        (
            SELECT 1
            FROM pg_catalog.pg_attribute
            LIMIT 100 -- you can adjust this value to fit the maximum number of
elements in your CSV strings
        ) a
    );

    INSERT INTO grouplist
    SELECT TRIM('"' FROM TRIM(SPLIT_PART(ldapgroups, ',', n::integer)))
    FROM numbers
    WHERE n <= REGEXP_COUNT(ldapgroups, ',') + 1;

    -- find out if we have a lookup table to map groups via
    SELECT EXISTS (SELECT 1 FROM pg_tables WHERE schemaname =
current_schema_name AND tablename=lookupTableName) INTO lookup_table_exists;

    IF lookup_table_exists THEN
        -- check whether lookup_table has any rows. If not there's no reason to
process queries related to it.
        EXECUTE 'SELECT COUNT(*) FROM ' || current_schema_name || '.' ||

```

```

lookupTableName || ' lt LIMIT 1' INTO lookup_table_has_rows;
END IF;

IF lookup_table_exists AND lookup_table_has_rows THEN
    RAISE NOTICE '%.sync_user: lookup table found', current_schema_name;
    -- we have a lookup table, use it to map the inbound groups to local
groups
    sql := 'insert into finalgroups select ' || lookupTableValue || ' from
' || current_schema_name || '.' || lookupTableName || ' lt, grouplist gl where
lt.' || lookupTableKey || ' = g;';
    EXECUTE sql;
    -- and find the unmapped groups
    sql := 'insert into finalgroups select g from ' || current_schema_name
|| '.' || lookupTableName || ' lt, grouplist gl where g not in (' ||
lookupTableKey || ');';
    EXECUTE sql;
ELSE
    -- no lookup group, so just copy the grouplist into the final groups table
    INSERT INTO finalgroups SELECT g FROM grouplist;
END IF;

-- at this point, finalgroups should have the complete grouplist of what
groups we want for the user
SELECT EXISTS (SELECT 1 FROM pg_user WHERE username = username) INTO
user_exists;

-- sync user and password
IF user_exists THEN
    sql := 'ALTER USER ' || quote_ident(username) || ' WITH PASSWORD '''
|| password || ''';';
    EXECUTE sql;
    sql := 'ALTER USER ' || quote_ident(username) || ' WITH PASSWORD '''
|| 'REDACTED' || ''';';
    RAISE NOTICE '%.sync_user: % updated', current_schema_name, sql;
    -- now find user's groups that are not in the finalgroup table
    FOR groupname IN SELECT groname FROM pg_group JOIN pg_user ON usesysid
= ANY(groplist)
    WHERE username = username LOOP
        -- this select is not yet working properly
        SELECT EXISTS (SELECT 1 FROM finalgroups WHERE groupname.groname =
ANY(SELECT g FROM finalgroups)) INTO has_rows;
        IF NOT has_rows THEN
            RAISE NOTICE '%.sync_user: dropping % from user %',
current_schema_name, groupname.groname, username;
            -- now revoke membership of the group from the user
            sql = 'ALTER GROUP ' || groupname.groname || ' DROP USER ' ||
username || ';';
            EXECUTE sql;
            RAISE NOTICE '%.sync_user: %', current_schema_name, sql;
        END IF;
    END LOOP;
ELSE
    sql := 'CREATE USER ' || quote_ident(username) || ' WITH PASSWORD '''
|| password || ''';';
    EXECUTE sql;
    sql := 'CREATE USER ' || quote_ident(username) || ' WITH PASSWORD '''
|| 'REDACTED' || ''';';

```

```

        RAISE NOTICE '%.sync_user: % created', current_schema_name, sql;
    END IF;

    FOR groupname IN SELECT g FROM finalgroups LOOP
        SELECT EXISTS (SELECT groname FROM pg_group WHERE groname ILIKE
groupname.g) INTO group_exists;
        RAISE NOTICE 'group: %, group_exists: %', groupname.g,
group_exists;
        IF (NOT group_exists) AND create_roles_if_missing THEN
            sql := 'CREATE GROUP ' || groupname.g;
            EXECUTE sql;
            RAISE NOTICE '%.sync_user: %', current_schema_name, sql;
        END IF;
        -- add group to user
        IF group_exists OR create_roles_if_missing THEN
            sql = 'ALTER GROUP ' || groupname.g || ' ADD USER ' || username ||
';';
            EXECUTE sql;
            RAISE NOTICE '%.sync_user: %', current_schema_name, sql;
        END IF;
    END LOOP;

    -- make sure we drop these to prevent reuse issues on the same connection
    DROP TABLE numbers;
    DROP TABLE grouplist;
    DROP TABLE finalgroups;

RETURN;
END;

$$
;

```

Active Directory Password and Group Synchronization extra options

By default, a group with the same name is created in the database for each extracted ldap group. By creating the table below for each key found (ldap_group), a role with the specified name in the value (role) will be created instead. Composite Primary Key allows to map more than one role to particular ldap_group; If 'role' is empty then no role is created and granted to user for this particular ldap_group.

```

CREATE TABLE IF NOT EXISTS heimdall.lookup_table (
    ldap_group varchar(255),
    db_group varchar(64),

```

```
        PRIMARY KEY (ldap_group, db_group)
    )
```

Heimdall Datasource Service User (Redshift) — Create and Grant Privileges

Create a dedicated Redshift user for the Heimdall datasource (used by Heimdall).

```
-- Run as a superuser (or a role that can CREATE USER / GRANT).
-- Replace placeholders in <> with your values.

-- 1) Create the heimdall schema if it doesn't exist
CREATE SCHEMA IF NOT EXISTS heimdall;

-- 2) Create the service user
CREATE USER <SERVICE_USER> WITH PASSWORD '<PASSWORD>';

-- 3) Grant schema access (required before table privileges)
GRANT USAGE ON SCHEMA heimdall TO <SERVICE_USER>;

-- 4) Grant table privileges on existing objects in that schema
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA heimdall TO
<SERVICE_USER>;
```

Pivotal Greenplum Specific Information

- When using the Async feature, set a connection property of "initSQLAsync" to the value of "set optimizer = off". This will disable the GPOrca optimizer when async is used, which can reduce the overhead per insert significantly.
- If using Postgres offload, ensure that both Greenplum and Postgres use identical schema and user configurations.
- If using foreign data wrappers from Postgres to Greenplum, add a rule as follows:
 - Regex: SET TRANSACTION ISOLATION LEVEL REPEATABLE READ
 - In-Trans: Checked
 - Action: Transform
 - Parameter: target
 - Value: SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

This will allow the repeatable read transaction isolation level (not supported by GP) to be translated to serializable, which will enable the Postgres FDW to operate properly

- Ensure that the proper modules for Postgres are loaded in the admin->modules tab.
- When configuring load balancing, specify the failover script (provided), and set the secondary node as being offline. Since Greenplum doesn't allow monitoring of the

secondary node, it is up to the script to trigger promotion and enabling the node in the LB configuration.

PG Notify support

This is not currently supported with Greenplum vs. current Postgres software versions.

Trigger Invalidation with Greenplum

While Greenplum supports triggers in a limited sense, the triggers do not work when writing to a table from the data segments. As such, trigger invalidation is not yet working with Greenplum (see below for an alternative).

Stored Procedure based Invalidation with Greenplum

As a replacement for trigger based invalidation, a stored procedure can be used that leverages the "Trigger invalidation" mechanism to provide invalidation for Greenplum. When used, the stored procedure can be called after each data load to notify Heimdall of new data.

Connection properties to use:

- dbTriggerTracking=true
- dbChanges=select ts,name from heimdall.tableupdates where ts > @ts
- dbTimeMs=select (date_part('epoch', now()))*1000)::bigint

Greenplum configuration:

```
CREATE SCHEMA heimdall;

CREATE TABLE heimdall.tableupdates (
name text PRIMARY KEY,
ts bigint not null );

CREATE OR REPLACE FUNCTION heimdall.changed(text) returns void AS $$
BEGIN
    LOOP
        UPDATE heimdall.tableupdates SET ts=(date_part('epoch',
now()))*1000)::bigint where name=$1;
        IF found THEN
            RETURN;
        END IF;
    BEGIN
        INSERT INTO heimdall.tableupdates VALUES ($1, (date_part('epoch',
now()))*1000)::bigint);
        RETURN;
    EXCEPTION WHEN unique_violation THEN
```

```
END;  
END LOOP;  
RETURN;  
END;  
$$ LANGUAGE plpgsql;
```

Source Configuration

Below is a screenshot of the default recommended data source configuration for a baseline Greenplum cluster. Details are explained below:

Data Sources

Create New Source

Greenplum

☒ Enable

Commit

Configuration

Driver: PostgreSQL+Greenplum

JDBC URL: jdbc:postgresql://gpmaster:5432/gpadmin

Username: gpadmin

Password:

Test query: SELECT 1

Test Connection

Execute scripts

Secret

Kerberos

Connection Properties

initSQLAsync

currentSchema

hdUsePGNotify

set optimizer = off

public

false

Custom Property

Connection Pooling

Enable Pooled Connection:

maxActive

1000

Load Balancing/High Availability

Enable Load Balancing:

Track Cluster Changes:

Use Response Metrics:

Track Replication Lags:

Falover Script:

Connection Hold Time (ms): 30000

Lag Window buffer (ms): 10000

statechange-greenplum-source.py

Name: Primary

Url: jdbc:postgresql://gpmaster:5432/gpadmin

Enabled:

Writable:

Weight: 1

Name: Secondary

Url: jdbc:postgresql://gpmaster:5432/gpadmin

Enabled:

Writable:

Weight: 1

Add Server

- Connection Properties

- **initSQLAsync**: set optimizer = off This is to disable GPOrca optimization when using the Async insert, as this slows down processing for most inserts
- **currentSchema**: public This can be adjusted, but sets the default schema used when connecting
- **hdUsePGNotify**: false Postgres supports

- Pooled Connections
 - **Enabled** to enable connection pooling
 - **maxActive** to set the maximum number of connections allowed in the pool
- Load Balancing/High Availability
 - **Enabled**
 - **Failover Script:** `statechange-greenplum-source.py`

Set your master node as enabled and writeable, and the standby as disabled, and not writeable. After a failover, please ensure that the standby is reset to be in sync and available as a standby per: https://gpdb.docs.pivotal.io/580/admin_guide/highavail/topics/g-recovering-a-failed-master.html

Oracle

Scheduling Jobs in Oracle

Job scheduling should be available by default. For automatic execution to occur, the following must be true:

- the **DBMS_SCHEDULER** package is VALID,
- **JOB_QUEUE_PROCESSES** > 0 (engine switch),
- the **CJQ0** (Job Queue Coordinator) process is started as needed,
- the global attribute **SCHEDULER_DISABLED** is FALSE (Depreceted, but kept for backward compatibility)

DBMS_SCHEDULER

DBMS_SCHEDULER in Oracle is “a package that provides a collection of scheduling functions and procedures that can be called from any PL/SQL program”. It is the PL/SQL API for creating, scheduling, and managing jobs in the database.

This query checks that the DBMS_SCHEDULER package exists and is marked VALID.

```
SELECT status
FROM dba_objects
WHERE object_type = 'PACKAGE' AND object_name = 'DBMS_SCHEDULER'
```

JOB_QUEUE_PROCESSES

Automatic job execution is controlled by the initialization parameter **JOB_QUEUE_PROCESSES**. As documented: “If JOB_QUEUE_PROCESSES is 0, then DBMS_JOB jobs and Oracle Scheduler (DBMS_SCHEDULER) jobs will not run in the database instance”. If not explicitly set, the default value is “Derived. The lesser value of CPU_COUNT * 20 and SESSIONS / 4”.

This query returns the current value of the JOB_QUEUE_PROCESSES initialization parameter.

```
SELECT TO_NUMBER(value)
FROM v$parameter
WHERE name = 'job_queue_processes'
```

It can be set by below query:

```
ALTER SYSTEM SET job_queue_processes = 0 SCOPE=BOTH; -- disables automatic
execution
ALTER SYSTEM SET job_queue_processes = 200 SCOPE=BOTH; -- enables with 200
job queue processes
```

CJQ0 (Job Queue Coordinator)

The background process responsible for executing jobs is [CJQ0 \(Job Queue Coordinator\)](#). According to the documentation: "The job queue coordinator process (CJQ0) selects jobs that need to be run from the data dictionary and spawns job queue worker processes (Jnnn) to run the jobs. Oracle Scheduler automatically starts and stops CJQ0 as needed".

This query counts the running CJQ0 (Job Queue Coordinator) background processes.

```
SELECT COUNT(*)
FROM v$bgprocess
WHERE paddr <> '00' AND name = 'CJQ0'
```

Global Attribute 'SCHEDULER_DISABLED'

[Global Attribute 'SCHEDULER_DISABLED'](#) - "When a row is returned, the dbms_scheduler is disabled."

This query returns the value of the SCHEDULER_DISABLED global attribute (TRUE/FALSE).

```
SELECT UPPER(value)
FROM dba_scheduler_global_attribute
WHERE attribute_name = 'SCHEDULER_DISABLED'
```

If the value is TRUE, make sure to set it to FALSE.

```
BEGIN DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE('SCHEDULER_DISABLED', 'FALSE');
END;
```

Note: No longer documented in current Oracle Database manuals. Kept for backward compatibility.

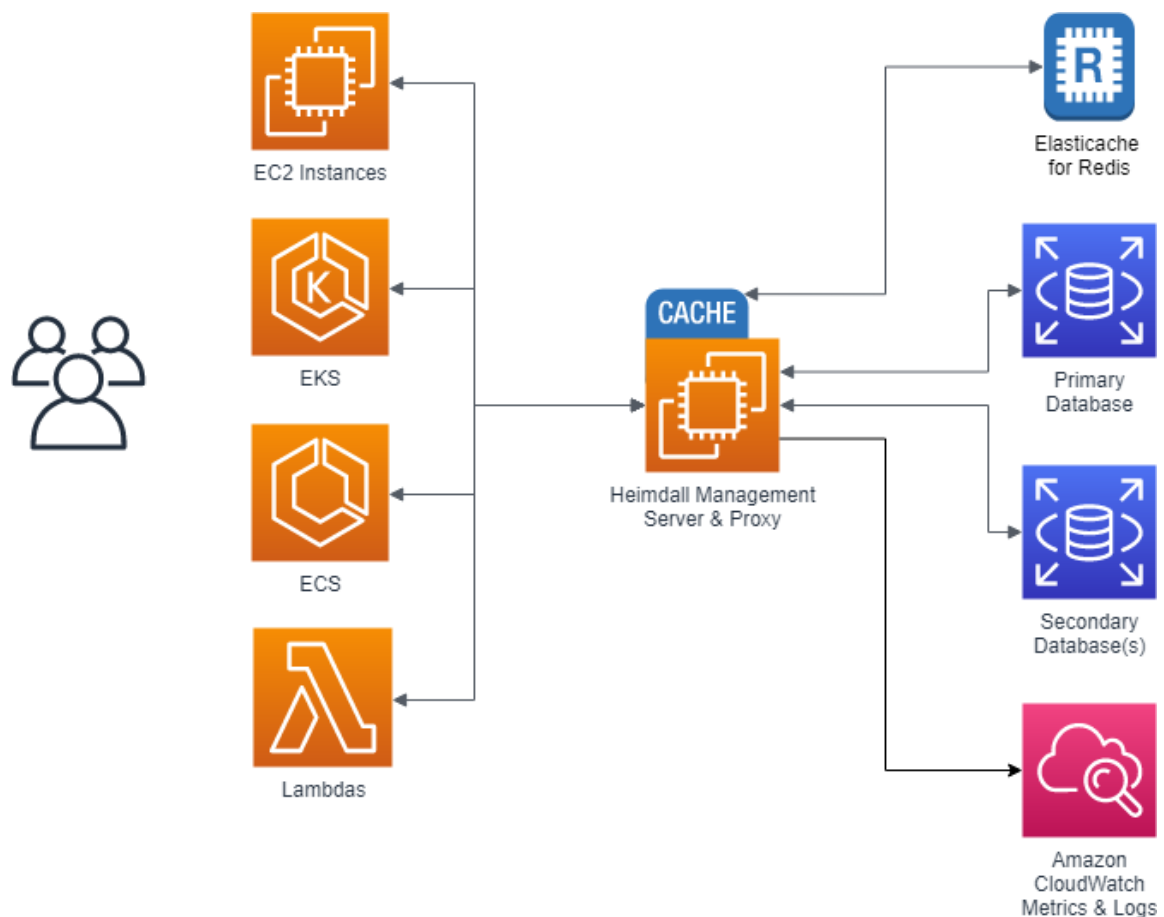
Recommended: Use `JOB_QUEUE_PROCESSES = 0` to disable scheduling.

Amazon AWS Specific Information

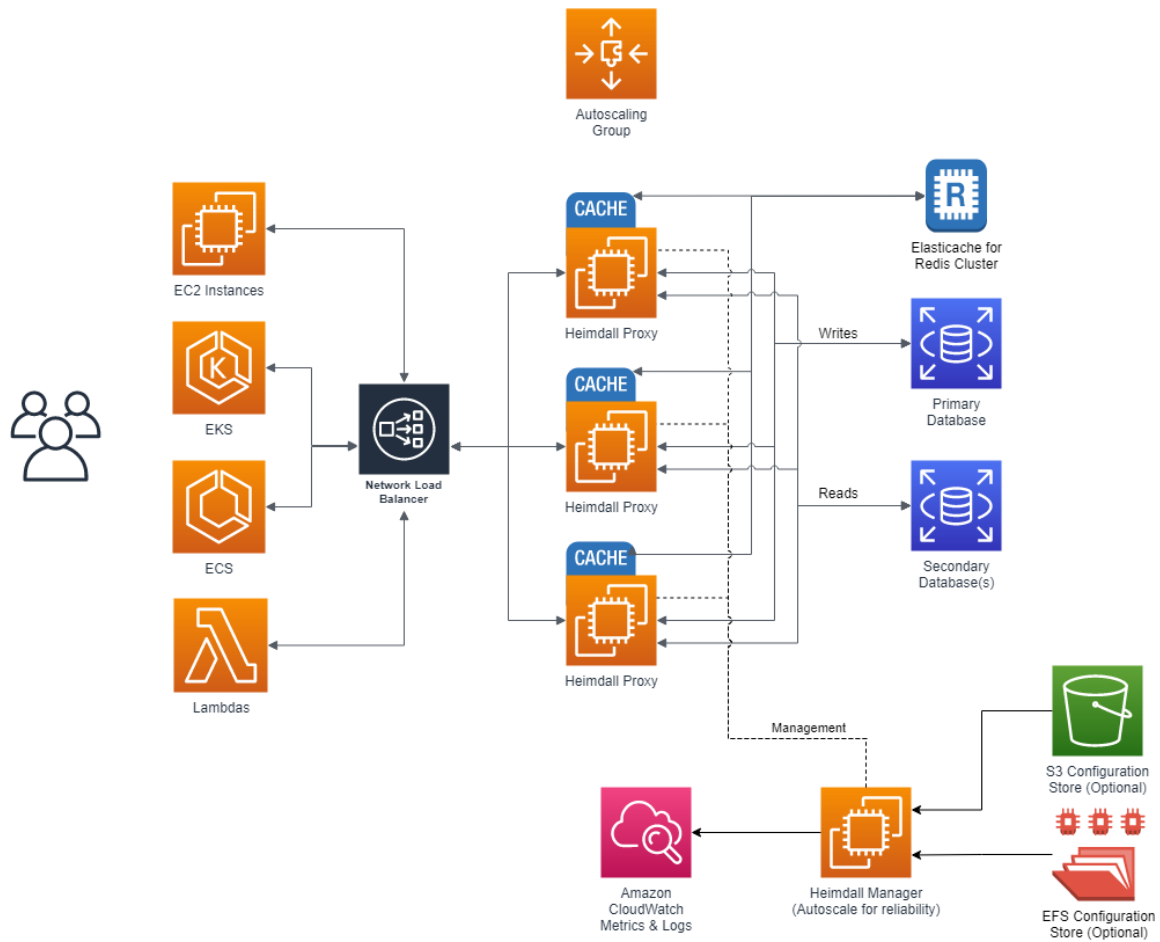
Please note--this section is organized in the form of a quick-start guide. If all the steps performed are done in-order, it will help to ensure success for an AWS deployment.

Typical Deployment

In most situations, customers go through two stages of deployment for Heimdall, a single server test deployment for POC installs and QA:



Here, Heimdall exists only on one server, and both the management server and the proxy exist together. Obviously, this results in a single point of failure, but this allows easy and fast proof of concept installs to occur. Once the idea of using Heimdall is validated, a more robust configuration is deployed:



Here, the proxy nodes are deployed into one or more Availability zones (to match the application server AZs) with autoscaling, and the central manager exists on its own server. As the proxies operate as an independent data plane, failure of the management server is non-disruptive, so most customers choose to simply deploy a single management server. If a redundant management server is desired however, this can be done as well, with the obvious extra cost. In the single management server model, the management server can sit behind an autoscaling group, and keep its configuration files on EFS, or pre-populate the configurations from S3, as desired.

RDS Support

Heimdall provides support for all MySQL, Postgres, and SQL Server RDS types, including Aurora and Aurora for MySQL, including all actively supported versions of those engines.

Creating an IAM Role

In order to simplify the configuration for Amazon Web Services, the system supports reading the RDS and ElastiCache configuration directly. In order to use this feature, one must configure an IAM role with access to the appropriate rights, as shown, then the option for "AWS

Configuration" will become available in the Wizard. Without the IAM credentials, a manual configuration is still possible.

Permissions	Trust relationships	Tags	Access Advisor	Revoke sessions
▼ Permissions policies (7 policies applied)				
Attach policies		+ Add inline policy		
Policy name ▼		Policy type ▼		
▶	SecretsManagerReadWrite	AWS managed policy		✕
▶	AmazonS3FullAccess	AWS managed policy		✕
▶	AmazonEC2ReadOnlyAccess	AWS managed policy		✕
▶	CloudWatchFullAccess	AWS managed policy		✕
▶	AmazonElastiCacheReadOnlyAccess	AWS managed policy		✕
▶	AmazonRDSReadOnlyAccess	AWS managed policy		✕
▶	AWSMarketplaceMeteringRegisterUsage	AWS managed policy		✕

The permissions are for:

- **AmazonEC2ReadOnlyAccess:** To detect the local region being run in, and the security groups to validate proxy configurations
- **AmazonRDSReadOnlyAccess:** To populate database configurations and to enhance auto-reconfiguration of clusters on a configuration change or failover
- **AmazonElastiCacheReadOnlyAccess:** To populate the available caches
- **CloudWatchFullAccess:** When using cloudwatch monitoring, to build log groups and to report data (recommended, but not required)
- **SecretsManagerReadWrite:** If the hdSecretKey option is to be used, this permission enables the manager to pull and push configuration information into a secret.

Note: In a production environment, it's recommended to follow the principle of least privilege and grant only the specific permissions needed by the manager or the associated entity to perform their required tasks. This can be achieved by creating custom IAM policies with more granular permissions based on the specific resources needed and actions required. It is often easier to let Heimdall create resources such as secrets and cloudwatch log groups, and then lock down access to those needed vs. create them upfront. In order to simplify this, an automatically generated json file is provided for Heimdall use that includes all the possible api calls that may be needed at [IAM Role Template](#). Please note--resource identification is templated as "changeme/*" as best practice is to limit roles to exactly what resources will be referenced. Please edit the template as needed for your environment.

For more information on IAM and IAM best practices please see [Security best practices in IAM](#).

Note, for any given API access to work, in addition to the IAM role, the api endpoint needs to be accessible as well. To test if the api endpoints are accessible from a given client, you can use the following:

```
wget https://ec2.us-east-1.amazonaws.com/
wget https://rds.us-east-1.amazonaws.com/
wget https://elasticache.us-east-1.amazonaws.com/ (will return a 404, this is normal)
wget https://monitoring.us-east-1.amazonaws.com/ (will return a 404, this is normal)
wget https://secretsmanager.us-east-1.amazonaws.com/ (will return a 404, this is normal)
```

If these return anything other than a timeout, they are likely accessible. Otherwise, the api endpoint likely needs to be enabled in the VPC, due to routing restrictions.

Configuring an ElastiCache for Redis / Valkey Instance

If ElastiCache will be used, it is recommended that the instance be setup before configuring Heimdall, as this allows the ElastiCache configuration to be auto-detected. Heimdall supports Amazon ElastiCache using Redis-compatible engines, including **Redis OSS** and **Valkey**. When configuring ElastiCache, make sure at least two nodes are configured for redundancy if desired (clustered or not). For initial configuration on a POC, it is generally recommended that the following options be selected:

1. Cluster mode enabled--this helps ensure proper behavior on a failover in a larger number of cases;
2. Multi-AZ enabled;
3. Size of m6g.large similar size instance. If load is low, then t4g.medium instances would be reasonable--the size can be adjusted if needed due to excessive bandwidth use after the initial install;
4. At least one replica for redundancy;
5. AZ selection should map to the availability zones that are expected to be used by the proxy layer and/or application servers;
6. Security options such as enabling encryption in transit and access control is optional.

After configuring the ElastiCache for Redis or Valkey install, it is important to configure a parameter group, and set the parameter "notify-keyspace-events" to the value "AE". This will allow the system to track objects that are added and removed from the cache automatically, which helps prevent L2 cache misses. In other Redis types, this parameter can be set dynamically at runtime, but in ElastiCache, this can only be set via the parameter group. The configuration should appear as below:

Parameter Group: redis-keyspace-events

Viewing: All Parameters Edit Parameters 21 to 30 of 134 parameters

Name	Allowed Values	Is Modifiable	Instance Class	Value	Source	Type	Change Type	Description
maxclients	1-65000	No	ALL	65000	system	integer	requires-reboot	The maximum number of Redis clients.
maxmemory-policy	volatile-lru,allkeys-lru,volatile-random,allkeys-random,volatile-ttl,noeviction	Yes	ALL	volatile-lru	user	string	immediate	Max memory policy.
maxmemory-samples	1-	Yes	ALL	3	user	integer	immediate	Max memory samples.
min-slaves-max-lag	0-	Yes	ALL	10	user	integer	immediate	Maximum number of seconds within which the master must receive a ping from a slave to take writes. Use this parameter together with min-slaves-to-write to regulate when the master stops accepting writes. Setting this value to 0 means the master always takes writes.
min-slaves-to-write	0-	Yes	ALL	0	user	integer	immediate	Number of slaves that must be connected in order for master to take writes. Use this parameter together with min-slaves-max-lag to regulate when the master stops accepting writes. Setting this to 0 means the master always takes writes.
notify-keyspace-events		Yes	ALL	AE	user	string	immediate	The keyspace events for Redis to notify Pub/Sub clients about. By default all notifications are disabled

Configuring Event Scheduler in Aurora/RDS MySQL

To enable the **event_scheduler**, navigate to the **Parameter Groups** section in the RDS Console. Create a new parameter group (or edit an existing one), locate the event_scheduler parameter, and set its value to 'ON'. Then, assign this parameter group to your RDS/Aurora instance (a restart might be needed for the changes to take effect).

Aurora and RDS > Parameter groups > mysql-event-scheduler-on

Aurora and RDS

- Dashboard
- Databases
- Query editor
- Performance Insights
- Snapshots
- Exports in Amazon S3
- Automated backups
- Reserved Instances
- Proxies
- Subnet groups
- Parameter groups
- Option groups
- Custom engine versions
- Zero-ETL integrations [New](#)
- Events

mysql-event-scheduler-on Edit Delete

Details

Parameter Group Type Custom	Resource Type DB cluster	Parameter group family aurora-mysql8.0	Description mysql-event-scheduler-on

Parameters (423) Info

Q event_ X 1 match < 1 > ⚙

Name	Value	Apply type	Data type	Value type	Source
event_scheduler	ON	Dynamic	String	Modifiable	Modified

Assigned resources (1) ⚙

Q Filter assigned resources < 1 > ⚙

Name	Version	Region	Size
database-1	8.0.mysql_aurora.3.05.2	us-east-1b	2

AWS Marketplace Install

Heimdall can be easily started using the AWS marketplace. During this startup, the image will download the newest release version of Heimdall. If the security groups are such that the download can not complete, then an older version of Heimdall will be used (from the instance creation time).

First, in the EC2, select launch instance, select the aws marketplace option on the left, and search for "Heimdall Standard Edition" or "Heimdall Enterprise Edition" with 24/7 support, then select:

1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review

Cancel and Exit

Step 1: Choose an Amazon Machine Image (AMI)

Quick Start (0)

My AMIs (2)

AWS Marketplace (1)

Community AMIs (3)

Categories

All Categories

Infrastructure Software (1)

DevOps (1)

Operating System

Clear Filter

All Linux/Unix

Ubuntu (1)

Heimdall Data Premium Edition

★★★★★ (6) | Nov 29, 2018
[Previous versions](#)
| By [Heimdall Data](#)

Free Trial

Starting from \$0.199/hr or from \$1,499/yr (up to 37% savings) for software + AWS usage fees

Linux/Unix, Ubuntu 18.04 LTS | 64-bit (x86) Amazon Machine Image (AMI) | Updated: 12/2/18

Transparent Database Proxy improving SQL performance for your existing database (e.g. Amazon RDS, Aurora, & Redshift; SQL Server, MySQL, Postgres etc.). Our AMI is deployed on a ...

Product highlights:

- Auto-caches and invalidates SQL results into Amazon ElastiCache. Installs in minutes.
- Routes read/write queries for better use of read replicas; no code changes
- AAA and Multi-User Connection Pooling for large-scale deployments

Transparent Database Proxy improving SQL performance for your existing database (e.g. Amazon RDS, Aurora, & Redshift; SQL Server, MySQL, Postgres etc.). Our AMI is deployed on a separate EC2 proxy tier. Deployment requires no application changes.

[Heimdall Data Premium Edition product detail page on AWS Marketplace](#)

[Show less](#)

Select

Then, continue with Heimdall:

Heimdall Data Premium Edition

Transparent Database Proxy improving SQL performance for your existing database (e.g. Amazon RDS, Aurora, & Redshift; SQL Server, MySQL, Postgres etc.). Our AMI is deployed on a separate EC2 proxy tier. Deployment requires no application changes.

[View Additional Details in AWS Marketplace](#)

Pricing Details

Free Trial

Try one instance of this product for 30 days. There will be no software charges but AWS infrastructure charges will still apply. Your Free Trial will automatically convert to a paid subscription upon expiration.

Hourly Fees

Instance Type	Software	EC2	Total
t2.medium	\$0.199	\$0.046	\$0.245/hr
t3.medium	\$0.199	\$0.042	\$0.241/hr
m5.large	\$0.399	\$0.096	\$0.495/hr
m5.xlarge	\$0.749	\$0.192	\$0.941/hr
m5.2xlarge	\$1.399	\$0.384	\$1.783/hr
c5.large	\$0.399	\$0.085	\$0.484/hr
c5.xlarge	\$0.749	\$0.17	\$0.919/hr
c5.2xlarge	\$1.399	\$0.34	\$1.739/hr
r5.large	\$0.549	\$0.126	\$0.675/hr
r5.xlarge	\$0.999	\$0.252	\$1.251/hr
r5.2xlarge	\$1.799	\$0.504	\$2.303/hr
z1d.large	\$0.599	\$0.186	\$0.785/hr
z1d.xlarge	\$1.099	\$0.372	\$1.471/hr
z1d.2xlarge	\$1.999	\$0.744	\$2.743/hr

Product Details

By

Heimdall Data

Customer Rating

★★★★★ (6)

Latest Version

Nov 29, 2018

Base Operating System

Linux/Unix, Ubuntu 18.04 LTS

Delivery Method

64-bit (x86) Amazon Machine Image (AMI)

License Agreement

[End User License Agreement](#)

On Marketplace Since

1/24/18

AWS Services Required

Amazon EBS, Amazon EC2

Highlights

- Auto-caches and invalidates SQL results into Amazon ElastiCache. Installs in minutes.
- Routes read/write queries for better use of read replicas; no code changes

Cancel

Continue

Select the desired instance types--The marketplace offering supports a variety of appropriate instances, with larger core counts supported in the Enterprise edition. Note on sizing: We generally recommend in initial core count for proxy capacity to be 1/4 that of the database behind Heimdall, so if the database has 8 cores, we would recommend starting with a 2 core proxy. In the event the cloud formation template was used, then you can split these cores across multiple proxies and use auto-scaling to size from there. In general, the c5 or Graviton 2 instances are recommended. For very high cache hit rates (over 90%), the c5n instance type may be appropriate, as the network bandwidth will be saturated before the CPU will be. Sizes can be adjusted once testing is complete and a better idea of the load ratio is understood.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 2: Choose an Instance Type

	General purpose	t3.small	2	2	EBS only	Yes	Up to 5 Gigabit	Yes
	General purpose	t3.medium	2	4	EBS only	Yes	Up to 5 Gigabit	Yes
	General purpose	t3.large	2	8	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel

Previous

Review and Launch

Next: Configure Instance Details

Continue through the screens, ensuring that the security group configuration opens ports as needed for proxy ports (please update the source subnets to match your local networks, it is best practice to never open ports to the internet in general):

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group

☐ Select an existing security group

Security group name: Heimdall Security Group

Description: Marketplace plus ports for databases

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP R	TCP	8087	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
MYSQL/Aurora	TCP	3306	Custom 0.0.0.0/0	MySQL
MS SQL	TCP	1433	Custom 0.0.0.0/0	SQL Server
PostgreSQL	TCP	5432	Custom 0.0.0.0/0	Postgres
Redshift	TCP	5439	Custom 0.0.0.0/0	Redshift

Add Rule

Cancel

Previous

Review and Launch

Finally, review and launch:

▼ AMI Details

[Edit AMI](#)



Heimdall Data Premium Edition

Root Device Type: ebs Virtualization type: hvm

Free Trial

Try one instance of this product for 30 days. There will be no software charges but AWS infrastructure charges will still apply. Your Free Trial will automatically convert to a paid subscription upon expiration.

Hourly Software Fees: \$0.199 per hour on t3.medium instance. Additional taxes or fees may apply.

Software charges will begin once you launch this AMI and continue until you terminate the instance.

Annual Subscriptions are available for this product, which can save you up to 37% when compared to hourly prices.

To purchase an Annual Subscription go to the [Your Software](#) page after launching the instance.

By launching this product, you will be subscribed to this software and agree that your use of this software is subject to the pricing terms and the seller's

[End User License Agreement](#)

▼ Instance Type

[Edit instance type](#)

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t3.medium	Variable	2	4	EBS only	Yes	Up to 5 Gigabit

▼ Security Groups

[Edit security groups](#)

Security group name Heimdall Security Group
Description Marketplace plus ports for databases

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ	Description ⓘ
SSH	TCP	22	0.0.0.0/0	
Custom TCP Rule	TCP	8087	0.0.0.0/0	
MYSQL/Aurora	TCP	3306	0.0.0.0/0	MySQL
MS SQL	TCP	1433	0.0.0.0/0	SQL Server
PostgreSQL	TCP	5432	0.0.0.0/0	Postgres
Redshift	TCP	5439	0.0.0.0/0	Redshift

▶ Instance Details

[Edit instance details](#)

▶ Storage

[Edit storage](#)

▶ Tags

[Edit tags](#)

[Cancel](#)

[Previous](#)

[Launch](#)

Next, with the EC2 instance online, you can bind the IAM Role created above to the EC2 instance, by right mouse-clicking on the instance, and under security, select modify IAM role. The following screen will allow you to select the role created. This enables autodetection of RDS and ElastiCache resources, and other AWS integrations.

Once the instance is online and the IAM role is configured, connect to the instance on port 8087—there is a login help page providing instructions for the initial login. Please configure the instance using the [wizard](#) for the best results.. Nearly every manual configuration will have a fault, often resulting in support calls.

Advanced Marketplace Install

As Heimdall can run on multiple servers, at initialization, an instance can accept as user-data configuration options that will control how the instance runs, and if it should operate as a proxy or server (only). When set to such a mode, the instance will attempt to tune itself for the role in question, i.e. use the entire instance's memory vs. allowing memory to be used. To provide

these options, the user-data should be a script that generates a file `/etc/heimdall.conf`. For more details see [heimdall.conf](#) configuration.

Please refer to [AWS CloudFormation Template](#) page for more information on using a template to create an autoscaling group, which automates the process of creating proxy-only instances, and provides failover resiliency as well.

CloudWatch Metrics and Logs

In each VDB, under logging, an option is available for AWS CloudWatch. If enabled and CloudWatch access is enabled in the system's IAM role, it will start logging a variety of metrics into CloudWatch under the "Heimdall" namespace, and the vdb logs will also be logged into CloudWatch. The metrics include:

- DB Query Rate
- DB Query Time
- Avg Response Time
- DB Read Percent
- DB Transaction Percent
- Cache Hit Percent

Please note that additional charges may be incurred due to metrics and logging, in particular if debug logs under high volume are logged into CloudWatch.

When using Heimdall on private IP ranges, please see <https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/cloudwatch-logs-and-interface-VPC.html> for more information to enable cloudwatch within the vpc.

TLS certificates for RDS

In the new builds starting in July 2021, the file `rds-combined-ca-bundle.pem` will be installed in `/etc/ssl/certs` and can be used to validate RDS CA certificates as appropriate and needed.

Aurora/RDS Load Balancing Behaviors

In an AWS RDS (including Aurora) environment, Heimdall has taken a flexible approach to allowing a cluster to be defined and configured. When used with RDS and the proper IAM role is attached to the management server(s), the primary URL hostname defined in the data source (outside of the LB) is used to identify a particular cluster. The cluster nodes are then probed in order to build the LB configuration. This definition is re-done at the initial cluster definition (if cluster tracking is enabled) every 30 seconds during normal processing and every 1 second

when a node is in a "failure" mode, and also every time the data source configuration is committed.

When a cluster definition poll is done, the driver specified for the data source is used to create a new set of nodes. The nodes can be defined as "listener" (SQL Server only), "writer" node, "replica" for a read replica, or "reader" which includes both replicas and writers. The reader and writer URL's are then used to populate the nodes as appropriate based on the actual cluster state.

Please see the driver configuration page for information on each variable.

The goal of the Heimdall configuration is to bypass DNS resolution when possible for endpoints, which can slow down the failover process, while allowing the underlying drivers to support automatic detection and failover of the nodes that are acceptable for reader vs. writer roles. Through the default configurations provided by Heimdall, sub-second failovers can be achieved in most cases, and via the templates, custom configurations can be created to adjust the actual behavior desired.

For example, with the Postgres drivers, the default writerUrl template is:

```
jdbc:postgresql://${readers}/${database}?targetServerType=master
```

Here, the `${readers}` is used, as the Postgres driver supports an ordered list, and the writer nodes will be included first in this list. With the `targetServerType` of `master` included, it will connect only to a master in the list, but on a failure, it will fail as quickly as possible to a new writer as soon as it is promoted. This allows the endpoint hostname resolution for Aurora to be bypassed, while providing the best possible availability.

And for the reader it is:

```
jdbc:postgresql://${readers}/${database}?  
targetServerType=preferSecondary&loadBalanceHosts=true&readOnly=true
```

Again, in this case, the driver's capability to load balance is being used to create a single pool of read connections, which will avoid using the writer node if possible.

An alternate writer URL could be used of:

```
jdbc:postgresql://${writer}/${database}
```

This wouldn't leverage the built-in failover capability, but would rely on Heimdall to detect a failure, and poll the AWS API's to find the new writer.

If each host in a number of readers is desired to have it's own node entry, along with graphs on the dashboard, and the write nodes should not be used for reading, the following alternate URL could be used:

```
jdbc:postgresql://${reader}/${database}?readOnly=true
```

This bypasses the postgres driver built-in LB capability, and will create a node for LB in Heimdall for each reader.

All built-in database types include a pre-defined set of reader and writer URL's except for Oracle, due to the complex nature of Oracle URL's for many environments. Older installs prior to this template function will have defaults included when updated to the code that supports this functionality.

Notes on behavior:

- `${writeEndpoint}` and `${readEndpoint}` can be used for Aurora definitions, and will point to the writer (primary) and reader endpoints as appropriate. This allows use of the AWS managed DNS targets if desired. DNS resolution on these will be cached for five seconds, and due to potential update timings, this could result in a connection intended for a writer to point to a reader instead. This is a side-effect of connection pooling, DNS caching, and the update timing of the DNS entry. This behavior can occur with other applications not using Heimdall as well.
- `${reader}` will include the nodes that are expanded from `${writers}`, but will set the weight of 1 vs. 10 for any pure reader node.
- `${readers}` can be used in the writer URL definition to expand to the entire list of writers and readers for drivers that will use an ordered list to identify what is a writer and what is a reader. Other variables can only be used in the proper context, i.e. `${writer}` for a `writeUrl`, or `${reader}` for a `readUrl`.
- `${listener}` is used only for SQL Server, and will resolve to the listener IP for a particular always-on cluster for the `writeUrl`, and will map to the writer instance in the cluster.

Things AWS doesn't tell you (or hides) that you need to know!

- Each EC2 instance type has a documented burstable bandwidth level, but it has an undocumented steady-state bandwidth level that is often significantly lower than the burstable limit. One website has prepared a cheat-sheet that provides the numbers for many [instance types](#).
- ELB including NLB (despite documentation to the contrary) may require pre-warming if you expect highly burstable traffic.
- RDS (but not Aurora) instances may get alerts of "Write query was sent to read-only server or transaction. Please see log for details." This error may be happening during your daily backup window, as the database may transition from writable to read-only during this

window for a period of time. Aurora instances do not freeze this way due to the way they handle block IO.

AWS Policies

Heimdall Manager Recommended Least Privilege Policy

Template

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowExportHeimdallCertificates",
      "Effect": "Allow",
      "Action": [
        "acm:ExportCertificate"
      ],
      "Resource": "arn:${AWS::Partition}:acm:${AWS::Region}:${AWS::AccountId}:certificate/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/App": "heimdall"
        }
      }
    },
    {
      "Sid": "AllowWriteLoginEventsToCloudTrail",
      "Effect": "Allow",
      "Action": [
        "cloudtrail-data:PutAuditEvents"
      ],
      "Resource": "arn:${AWS::Partition}:cloudtrail:${AWS::Region}:${AWS::AccountId}:channel/${ChannelId}"
    },
    {
      "Sid": "AllowWritePortalEventsToCloudTrail",
      "Effect": "Allow",
      "Action": [
        "cloudtrail-data:PutAuditEvents"
      ],
      "Resource": "arn:${AWS::Partition}:cloudtrail:${AWS::Region}:${AWS::AccountId}:channel/${ChannelId}"
    },
    {
      "Sid": "AllowDescribeElastiCacheClusters",
      "Effect": "Allow",
      "Action": [
        "elasticache:DescribeCacheClusters"
      ],
      "Resource": "arn:${AWS::Partition}:elasticache:${AWS::Region}:${AWS::AccountId}:cluster:${CacheClusterId}"
    }
  ]
}
```

```

    },
    {
      "Sid": "AllowReadIdentityStoreGroups",
      "Effect": "Allow",
      "Action": [
        "identitystore:DescribeGroup"
      ],
      "Resource": [
        "arn:${AWS::Partition}:identitystore::${AWS::AccountId}:identitystore/${IdentityStoreId}",
        "arn:${AWS::Partition}:identitystore:::group/*"
      ]
    },
    {
      "Sid": "AllowWritePortalAuditLogs",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "logs:PutRetentionPolicy"
      ],
      "Resource": [
        "arn:${AWS::Partition}:logs:${AWS::Region}:${AWS::AccountId}:log-group:${PortalAuditLogGroupName}",
        "arn:${AWS::Partition}:logs:${AWS::Region}:${AWS::AccountId}:log-group:${PortalAuditLogGroupName}:log-stream:*"
      ]
    },
    {
      "Sid": "AllowWriteManagerLogs",
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "logs:PutRetentionPolicy"
      ],
      "Resource": [
        "arn:${AWS::Partition}:logs:${AWS::Region}:${AWS::AccountId}:log-group:${ManagerLogGroupName}",
        "arn:${AWS::Partition}:logs:${AWS::Region}:${AWS::AccountId}:log-group:${ManagerLogGroupName}:log-stream:*"
      ]
    },
    {
      "Sid": "AllowDescribeAndPromoteRDS",
      "Effect": "Allow",
      "Action": [
        "rds:DescribeDBClusters",
        "rds:DescribeDBInstances",
        "rds:DescribeGlobalClusters",
        "rds:PromoteReadReplica"
      ],
      "Resource": [

```



```

        "arn:${AWS::Partition}:rds:${AWS::Region}:${AWS::AccountId}:cluster:${
        DbClusterInstanceName}",
        "arn:${AWS::Partition}:rds:${AWS::Region}:${AWS::AccountId}:db:${
        DbInstanceName}",
        "arn:${AWS::Partition}:rds:${AWS::AccountId}:global-cluster:${
        GlobalCluster}"
    ]
},
{
    "Sid": "AllowManageHeimdallSecrets",
    "Effect": "Allow",
    "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:CreateSecret",
        "secretsmanager:UpdateSecret"
    ],
    "Resource": "arn:${AWS::Partition}:secretsmanager:${AWS::Region}:${
    AWS::AccountId}:secret:heimdall/*"
},
{
    "Sid": "AllowPublishToSNSTopics",
    "Effect": "Allow",
    "Action": [
        "sns:ListSubscriptionsByTopic",
        "sns:Publish"
    ],
    "Resource": [
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${
        NotificationTopicName#1}",
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${
        NotificationTopicName#2}",
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${
        NotificationTopicName#*}"
    ]
},
{
    "Sid": "AllowReadS3HeimdallTemplates",
    "Effect": "Allow",
    "Action": [
        "s3:GetObject",
        "s3:ListBucket"
    ],
    "Resource": [
        "arn:aws:s3:::templates.heimdalldata.com/*",
        "arn:aws:s3:::templates.heimdalldata.com"
    ]
},
{
    "Sid": "AllowPutLogsIntoS3",
    "Effect": "Allow",
    "Action": [
        "s3:PutObject"
    ],
    "Resource": "arn:${AWS::Partition}:s3:::${CloudLogsBucketName}/*"
}
]
}

```

Example

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowExportHeimdallCertificates",
      "Effect": "Allow",
      "Action": [
        "acm:ExportCertificate"
      ],
      "Resource": "arn:*:acm:*:*:certificate/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/App": "heimdall"
        }
      }
    },
    {
      "Sid": "AllowWriteLoginEventsToCloudTrail",
      "Effect": "Allow",
      "Action": [
        "cloudtrail-data:PutAuditEvents"
      ],
      "Resource": "arn:*:cloudtrail:*:*:channel/*"
    },
    {
      "Sid": "AllowWritePortalEventsToCloudTrail",
      "Effect": "Allow",
      "Action": [
        "cloudtrail-data:PutAuditEvents"
      ],
      "Resource": "arn:*:cloudtrail:*:*:channel/*"
    },
    {
      "Sid": "AllowDescribeElastiCacheClusters",
      "Effect": "Allow",
      "Action": [
        "elasticache:DescribeCacheClusters"
      ],
      "Resource": "arn:*:elasticache:*:*:cluster:*"
    },
    {
      "Sid": "AllowReadIdentityStoreGroups",
      "Effect": "Allow",
      "Action": [
        "identitystore:DescribeGroup"
      ],
      "Resource": [
        "arn:*:identitystore:*:*:identitystore/*",
        "arn:*:identitystore::*:group/*"
      ]
    },
    {
      "Sid": "AllowWritePortalAuditLogs",
```

```

    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "logs:PutRetentionPolicy"
    ],
    "Resource": [
        "arn*:logs:*:*:log-group:HEIMDALL-audit",
        "arn*:logs:*:*:log-group:HEIMDALL-audit:log-stream:*"
    ]
},
{
    "Sid": "AllowWriteManagerLogs",
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "logs:PutRetentionPolicy"
    ],
    "Resource": [
        "arn*:logs:*:*:log-group:HEIMDALL-mgmt",
        "arn*:logs:*:*:log-group:HEIMDALL-mgmt:log-stream:*"
    ]
},
{
    "Sid": "AllowDescribeAndPromoteRDS",
    "Effect": "Allow",
    "Action": [
        "rds:DescribeDBClusters",
        "rds:DescribeDBInstances",
        "rds:DescribeGlobalClusters",
        "rds:PromoteReadReplica"
    ],
    "Resource": [
        "arn*:rds:*:*:cluster:*",
        "arn*:rds:*:*:db:*",
        "arn*:rds:*:*:global-cluster:*"
    ]
},
{
    "Sid": "AllowManageHeimdallSecrets",
    "Effect": "Allow",
    "Action": [
        "secretsmanager:GetSecretValue",
        "secretsmanager:CreateSecret",
        "secretsmanager:UpdateSecret"
    ],
    "Resource": "arn*:secretsmanager:*:*:secret:heimdall/*"
},
{
    "Sid": "AllowPublishToSNSTopics",
    "Effect": "Allow",
    "Action": [

```

```

        "sns:ListSubscriptionsByTopic",
        "sns:Publish"
    ],
    "Resource": "arn*:sns:*:*:*"
},
{
    "Sid": "AllowReadS3HeimdallTemplates",
    "Effect": "Allow",
    "Action": [
        "s3:GetObject",
        "s3:ListBucket"
    ],
    "Resource": [
        "arn:aws:s3:::templates.heimdalldata.com/*",
        "arn:aws:s3:::templates.heimdalldata.com"
    ]
},
{
    "Sid": "AllowPutLogsIntoS3",
    "Effect": "Allow",
    "Action": [
        "s3:PutObject"
    ],
    "Resource": "arn*:s3:::*/*"
}
]
}

```

Heimdall Proxy Recommended Least Privilege Policy

Template

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDescribeAndExportHeimdallCertificates",
            "Effect": "Allow",
            "Action": [
                "acm:DescribeCertificate",
                "acm:ExportCertificate"
            ],
            "Resource": "arn:${AWS::Partition}:acm:${AWS::Region}:${AWS::AccountId}:certificate/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/App": "heimdall"
                }
            }
        },
        {
            "Sid": "AllowPutHeimdallProxyMetrics",
            "Effect": "Allow",

```

```

    "Action": [
      "cloudwatch:PutMetricData"
    ],
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "cloudwatch:namespace": "${ProxyLogGroupName}-proxy"
      }
    }
  },
  {
    "Sid": "AllowDescribeEC2Instances",
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeInstances"
    ],
    "Resource": "*"
  },
  {
    "Sid": "AllowWriteProxyLogs",
    "Effect": "Allow",
    "Action": [
      "logs:CreateLogGroup",
      "logs:CreateLogStream",
      "logs:DescribeLogStreams",
      "logs:PutLogEvents",
      "logs:PutRetentionPolicy"
    ],
    "Resource": [
      "arn:${AWS::Partition}:logs:${AWS::Region}:${AWS::AccountId}:log-group:${ProxyLogGroupName}",
      "arn:${AWS::Partition}:logs:${AWS::Region}:${AWS::AccountId}:log-group:${ProxyLogGroupName}:log-stream:*"
    ]
  },
  {
    "Sid": "AllowReadHeimdallSecrets",
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": "arn:${AWS::Partition}:secretsmanager:${AWS::Region}:${AWS::AccountId}:secret:heimdall/*"
  }
]
}

```

Example

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDescribeAndExportHeimdallCertificates",
      "Effect": "Allow",

```

```

    "Action": [
        "acm:DescribeCertificate",
        "acm:ExportCertificate"
    ],
    "Resource": "arn:*:acm:*:*:certificate/*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceTag/App": "heimdall"
        }
    }
},
{
    "Sid": "AllowPutHeimdallProxyMetrics",
    "Effect": "Allow",
    "Action": [
        "cloudwatch:PutMetricData"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "cloudwatch:namespace": "HEIMDALL-proxy"
        }
    }
},
{
    "Sid": "AllowDescribeEC2Instances",
    "Effect": "Allow",
    "Action": [
        "ec2:DescribeInstances"
    ],
    "Resource": "*"
},
{
    "Sid": "AllowWriteProxyLogs",
    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "logs:PutRetentionPolicy"
    ],
    "Resource": [
        "arn*:logs:*:*:log-group:HEIMDALL",
        "arn*:logs:*:*:log-group:HEIMDALL:log-stream:*"
    ]
},
{
    "Sid": "AllowReadHeimdallSecrets",
    "Effect": "Allow",
    "Action": [
        "secretsmanager:GetSecretValue"
    ],
    "Resource": "arn*:secretsmanager:*:*:secret:heimdall/*"
}
]
}

```

Amazon AWS CloudFormation Template

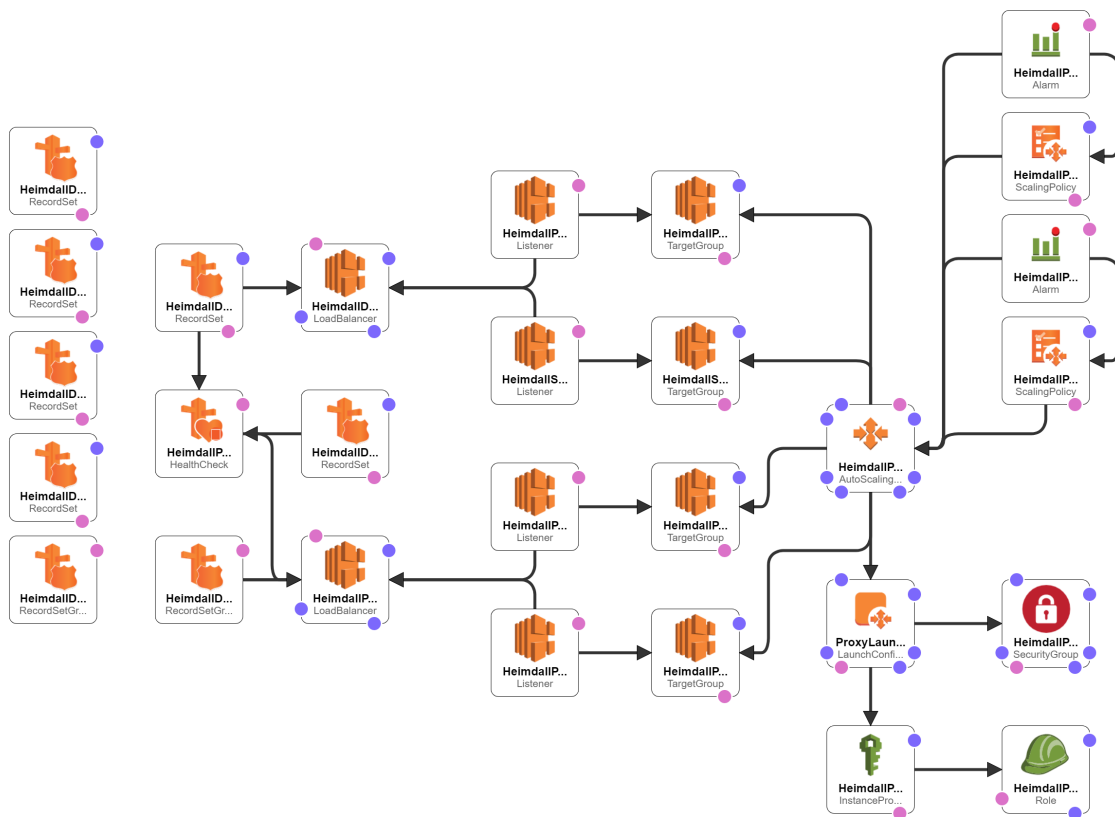
In order to provide a redundant proxy configuration, the following guide can be followed.

- Deploy a Heimdall Central Manager via the AWS Marketplace. This instance doesn't need to be particularly large, many customers use a t3.medium;
- Ensure the application works appropriately through a proxy instance hosted on the management server;
- Once the application is validated, you can un-check the proxy option "Local proxy" which will terminate the proxy on this management server;
- To fully dedicate the memory of the management server to management activities, follow the heimdall.conf instructions [here](#);
- Deploy a cluster of proxy-only nodes using the template located at: [Template](#).

Note: Support for this template is exclusively provided for Heimdall Enterprise Edition customers as customers that require this level of redundancy also typically require 24/7 support. Heimdall offers private offer pricing to customers on a negotiated basis if needed.

The template is designed to pull the newest Enterprise (ARM) image in any region automatically. If a custom AMI ID is desired, please create a SSM parameter, and provide the AMI ID desired as the value of the parameter. Use the parameter name as the AMI Alias in the CF template, and on deployment, it will resolve the AMI ID per the SSM parameter.

The resources created by this template are:



- One or Two Network Load Balancers with Listener [billable](#) Note: We support both database protocol load balancing as well as distributing traffic via DNS queries. In the DNS case, the NLB costs will be significantly less, but overall traffic distribution may not be as even across nodes.
- Autoscaling configuration with Target Group
- Scale up and down policies with alarms
- Launch Configuration with instance policy and IAM role
- Security group for proxy EC2 instances
- (Optional) Route53 records to monitor the proxies and provide easy to use hostnames to access the proxies, including with geolocation capability. [billable](#)

In the case of the DNS based NLB setup, the template will also configure: * A record pointing to the NLB target hostname * An NS record that points to the A record pointing to the NLB target hostname

It depends on: * Existing VPC & Subnets * An existing Heimdall Central Manager * SSH Key

All resources will be named based on the stack name as appropriate, and their creation can be reversed by deleting the stack.

Heimdall assumes no responsibility for the charges that may be charged by AWS through the use of this template, as in particular, outside of the EC2 instances, the network load balancer may incur charges, as explained ["here"](#).

Template Options

For all template options, the following are required:

- ProxyMinimum and ProxyMaximum set to numeric values
- HeimdallProxyVPC configured, and HeimdallProxyAvailabilityZones and HeimdallPublicProxySubnetList configured, with all parameters in alignment
- ProxyPublicIpAddress can be set to true if public routing will be used, but for single region deployments, this is typically set to false
- HeimdallProxyAMIId is set to the AMI alias for the Heimdall Enterprise (ARM) edition. Minor editing could be done to use Intel VM's however if need.
- InstanceTypeParameterProxy is set to an appropriately sized instance size
- Keyname for the SSH key to install on the system
- vdbName is configured
- hdHost is configured
- hdAccessKey is configured. This could be the admin user OR the VDB access key. This could also be configured to use AWS Secrets Manager; in that case, specify this parameter in the following format: `secret:<secret name or ARN>` . The secret name should match the one configured in the VDB Access/Secret key settings. The proxy will retrieve these values on startup and use them to communicate with the manager.
- hdSecretKey is configured. This could be the admin password or the VDB secret key. This parameter should remain empty when AWS Secrets Manager is used in the hdAccessKey parameter.
- ProxyPort is configured.
- ProxyLogGroupName is required (you can use the default) because it defines the CloudWatch log group name needed for the AWS permissions policy for logs and metrics.

Note: The HeimdallProxyAMIId defaults to: `/aws/service/marketplace/prod-6k3x2q5j7vg32/latest` for the ARM Enterprise Edition. Other valid options are:

- `/aws/service/marketplace/prod-upicfbvjlnfqo/latest` for the ARM Enterprise Plus Edition
- `/aws/service/marketplace/prod-ahm64zezggfhg/latest` for the Intel Enterprise Edition
- `/aws/service/marketplace/prod-qomddleky3adi/latest` for the Intel Standard Edition

Please note that in production, we only support the Enterprise edition options for larger installs with autoscaling.

Single Region NLB Balanced Cluster

The first and simplest configuration that the template supports is one where NLB is used to load balance an autoscaling group of proxies, and the load balancer name is used to access the proxies in a single region. In the template parameters, to configure this, the following are needed:

LBType=nlb

The following options should be blanked to avoid unnecessary route53 configurations for this simple setup:

HostedZoneID, DNSZoneName, dbEndpointName

Multiple Region NLB Balanced Cluster

Like single region, but enable the route53 configuration items of "HostedZoneID" and "DNSZoneName", and deploy the CF template in multiple regions. The primary access hostname will be generated based on the vdb name, and all stacks should have a unique name, to ensure there are no conflicts in global resources like IAM roles.

Single OR Multi-Region DNS Balanced Cluster

This setup is designed to leverage DNS to balance traffic to proxy nodes vs. NLB. NLB is however still used to direct DNS queries to the proxies themselves, but due to the low volume of DNS queries vs. database traffic, this should result in a very low bill for the NLB. For this configuration to be effective, the following should be noted:

1. There should be a large number of application servers, or a high amount of thrashing of clients accessing the database/proxy. This is to ensure that caching of DNS results doesn't result in unbalanced traffic load on the proxies;
2. If possible, the application servers should be configured to periodically close and open new connections, AND to re-query DNS periodically. This is very similar to guidance on using RDS to access a read-only URL to access multiple reader nodes.
3. A public NLB will be created on a public IP to allow access to DNS and HTTP queries. The HTTP queries will be coming from Route53 health check nodes, and will simply be querying for the URL /status, which is exposed by the proxies for health monitoring. This NLB needs to be public for these functions to work.

To use this configuration:

1. In the Heimdall VDB configuration, under the advanced options, the auto-scale mode checkbox should be checked
2. DNS port set to 54 (53 may work in some environments, but not all so should be avoided)
3. Health Check Port should be set to 80, along with a reasonably low health check interval (5 is suggested)
4. The Proxy Redirect Name parameter in Advanced Options should be configured to either Proxy Private IP or AWS Public IP. If using a multi-region AWS configuration, unless all private IP's are configured across VPCs to be routable, then use the AWS Public IP
5. If a global RDS deployment is in place, make sure to check the "Use Response Metrics" option in the data source LB section to preferentially route to the closest database nodes for reads
6. Deploy the template in all regions proxies are desired

In the template, make sure the following are set correctly:

LBType=dns HostedZoneID and DNSZoneName are configured

Optional Variations

If Route53 configurations are used, then the dbEndpointName can be configured to provide a "failover" name to be used if the proxy environment fails. This applies to both NLB and DNS configurations.

If a customer has two endpoints configured for their database, one for reads and one for writes, then you can use the SecondaryProxyPort to expose a second port on the proxy to provide the second endpoint.

If Heimdall proxy peer autodiscovery is desired (when using Hazelcast), then the autodiscoveryKey and autodiscoveryValue can be set. When using the cloudformation template, the aws tag key should be "Name" and the value should be set to the name of the proxies, generally the cloud formation name+"-proxy".

AWS IAM Permissions for the Proxy Instance

The CloudFormation template includes a default inline IAM policy to configure the permissions for the proxy instance. This policy is structured around five key permission sets:

- **AllowExportHeimdallCertificates:** Allows the proxy to use exportable AWS Certificate Manager (ACM) certificates. By default, this is restricted to certificates tagged with `App: heimdall`.

- **AllowWriteProxyLogs:** Grants permission to write logs to Amazon CloudWatch. This is limited to the proxy's specific log group resource `${ProxyLogGroupName}`.
- **AllowPutHeimdallProxyMetrics:** Allows the proxy to publish metrics to Amazon CloudWatch. This is restricted to the specific CloudWatch namespace `${ProxyLogGroupName}-proxy`.
- **AllowDescribeEC2Instances:** Permits the proxy to retrieve its own public IP address. This action is used by the autoscaling feature and is not limited to a specific resource.
- **AllowReadHeimdallSecrets:** Allows the proxy to read secrets from AWS Secrets Manager. By default, access is limited to secrets with the `heimdall/` resource prefix.

Note: By default, the AWS partition, region, and account ID are automatically fetched from the proxy instance's metadata and credentials.

Policy Customization

The template includes an inline policy by default, which we provide as a least-privilege starting point. You have two main options for customization:

1. **Modify the Default:** You can modify the inline policy directly within the CloudFormation template to fit your specific needs.
2. **Use a Managed Policy:** You can create your own IAM managed policy (examples are in the `environment/aws` directory) and specify it in the CloudFormation template parameters. This will attach your policy instead of the default inline one.

Microsoft Azure Specific Information

Azure Marketplace Install

When logged into your Azure console, navigate to the [Heimdall offering] (https://portal.azure.com/#blade/Microsoft_Azure_Marketplace/MarketplaceOffersBlade/selectedMenuItemId/home/searchQuery/heimdall) or [here](#):

[Home](#) > [New](#) > [Marketplace](#) > SQL Performance Improvement and Failover

SQL Performance Improvement and Failover

Heimdall Data

 **SQL Performance Improvement and Failover** [Save for later](#)

Free trial

[Create](#) [Start with a pre-set configuration](#)

Want to deploy programmatically? [Get started](#)

Heimdall Data is a Database Proxy for Developers, DBA's, and Networking Engineers. Improves performance up to 20x and reduces database costs up to 50%. Deployment requires zero changes to your existing application:

- Automated Query Caching
- Read/Write splits for better use of read replicas
- Connection multiplexing & pooling
- Fast failover, compatible with AlwaysOn
- Batch processing of DML operations
- SQL analytics for root cause analysis
- Supports an SQL database (e.g. SQL Server, Postgres, MySQL)

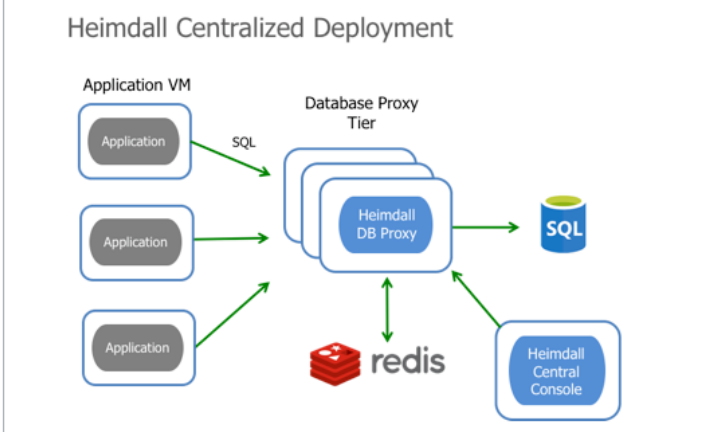
Useful Links

[Heimdall Data Overview](#)

[Technical Documentation](#)

[Blog](#)

Heimdall Centralized Deployment



The diagram illustrates the Heimdall Centralized Deployment architecture. On the left, three boxes labeled 'Application' are stacked vertically, each with an arrow pointing to a central 'Database Proxy Tier' box. The top arrow is labeled 'SQL'. The 'Database Proxy Tier' box contains a 'Heimdall DB Proxy' box. To the right of the proxy tier is a 'SQL' database icon. Below the proxy tier is a 'redis' icon, and to its right is a 'Heimdall Central Console' box. Arrows show bidirectional communication between the 'Heimdall DB Proxy' and both the 'redis' instance and the 'Heimdall Central Console'.

Configure the appropriate instance settings for your environment, and review/create:

Create a virtual machine

[Basics](#) [Disks](#) [Networking](#) [Management](#) [Advanced](#) [Tags](#) [Review + create](#)

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image.

Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization.

Looking for classic VMs? [Create VM from Azure Marketplace](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription ⓘ	Pay-As-You-Go
* Resource group ⓘ	azure-testing Create new

Instance details

* Virtual machine name ⓘ	heimdall-test
* Region ⓘ	(US) East US
Availability options ⓘ	No infrastructure redundancy required
* Image ⓘ	Heimdall Data Premium Edition (Per Hour) Browse all public and private images
* Size ⓘ	Standard A3 4 vcpus, 7 GiB memory Change size

Administrator account

Authentication type ⓘ	☒ Password ☐ SSH public key
* Username ⓘ	heimdall
* Password ⓘ	
* Confirm password ⓘ	

Review settings and create:

Create a virtual machine

✓ Validation passed

[Basics](#) [Disks](#) [Networking](#) [Management](#) [Advanced](#) [Tags](#) [Review + create](#)

PRODUCT DETAILS

SQL Performance Improvement and
Failover

by Heimdall Data

[Terms of use](#) | [Privacy policy](#)

Standard A3

by Microsoft

[Terms of use](#) | [Privacy policy](#)

Not covered by credits ⓘ

1.2800 USD/hr

Subscription credits apply ⓘ

0.2400 USD/hr

[Pricing for other VM sizes](#)

TERMS

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the [Azure Marketplace Terms](#) for additional details.

Name	undefined undefined
* Preferred e-mail address	erik@heimdalldata.com ✓
* Preferred phone number	1-866-433-2824 ✓

Basics

Subscription	Pay-As-You-Go
Resource group	azure-testing
Virtual machine name	heimdall-test
Region	(US) East US
Availability options	No infrastructure redundancy required
Authentication type	Password
Username	heimdall

Disks

OS disk type	Standard SSD
Use managed disks	Yes
Use ephemeral OS disk	No

Networking

Virtual network	(new) azure-testing-vnet
Subnet	(new) default (10.0.2.0/24)
Public IP	(new) heimdall-test-ip
NIC network security group	(new) heimdall-test-nsg
Accelerated networking	Off
Place this virtual machine behind an existing load balancing solution?	No

Management

Boot diagnostics	On
OS guest diagnostics	Off
Azure Security Center	Basic (free)
Diagnostics storage account	(new) azuretestingdiag739
System assigned managed identity	Off
Auto-shutdown	Off

Advanced

Extensions	None
Cloud init	No

Google Cloud Specific Information

GCP Marketplace Install

When logged into your Google Console, navigate to the [Heimdall offering](#) and select the desired Heimdall offer (Standard or Enterprise) based on support needs.

Configure the appropriate instance settings for your environment, and review/create:



heimdall-proxy-standard

Heimdall Data

Improve Database Scale 20x without Application Changes

LAUNCH

VIEW PAST DEPLOYMENTS

OVERVIEW

PRICING

DOCUMENTATION

SUPPORT

Review settings and change as needed and deploy, adjusting the ports that are open as needed. Port 22 and 8087 should be exposed, as well as the port desired for your database traffic:

Deployment name

Zone ?

Machine type ?

4 GB memory

[Customize](#)

Boot Disk

Boot disk type ?

Boot disk size in GB ?

Networking

Network interfaces



+ Add network interface



You have reached the maximum number of one network interface

Firewall ?

Add tags and firewall rules to allow specific network traffic from the Internet



Creating certain firewall rules may expose your instance to the Internet. Please check if the rules you are creating are aligned with your security preferences. [Learn more](#)

☒ Allow TCP port 22 traffic from the Internet

Source IP ranges for TCP port 22 traffic ?

☒ Allow TCP port 8087 traffic from the Internet

Source IP ranges for TCP port 8087 traffic ?

☒ Allow TCP port 3306 traffic from the Internet

Source IP ranges for TCP port 3306 traffic ?

☒ Allow TCP port 5432 traffic from the Internet

Source IP ranges for TCP port 5432 traffic ?

☒ Allow TCP port 1433 traffic from the Internet

Source IP ranges for TCP port 1433 traffic ?

MemSQL

MemSQL/SingleStore Specific Information

Common Issues

As MemSQL/SingleStore doesn't have a mysql database, set the "defaultCatalog" to be "memsql" instead, otherwise, the proxy won't be able to properly connect for initialization.

PHP PDO server's certificate verification

Known are problems about verifying server certificate by PHP PDO during using SSL for connection to a proxy. To help with resolving a problem, below suggestions can be helpful.

Applications suggested to use during work on a problem:

- [KeyStore Explorer](#)

Reminder before work

- All keys used by Heimdall are saved in a keystore named `keystore.p12`.
- Password to the keystore is `heimdall`.
- All key pair have to have empty password.
- Default global key pair used by each proxy is named `global_use_certificate`.
- Specific key pair for a proxy can be specified by adding key pair with alias matching lowercase name of a proxy, i.e. proxy named `MySQLVirtualDatabase` will use key pair aliased `mysqlvirtualdatabase`. That key pair have higher priority than global key pair.

1. Check certificate

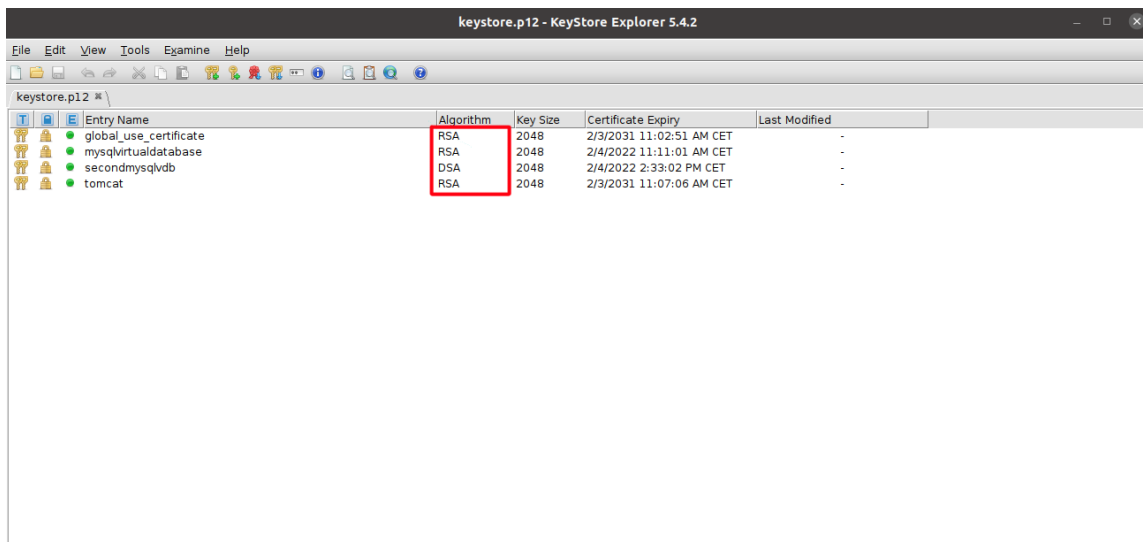
There are known two issues with certificates, which can lead to problems during using PHP PDO. Issue can be:

- an incorrect algorithm used during generating certificate,
- an incorrect value of CN, which can be translated to an IP address.

1.1. Check used algorithm

First, algorithm used to generate key pair should be checked. There are known issues, when certificate from a key pair generated via DSA algorithm was causing issues. By default, self-signed certificate generated by Heimdall is generated by using RSA algorithm, but if key pair was generated or imported by an user, then that can't be assured.

Below can be seen marked section where is written what algorithm was used to generate key pair. Window below is visible after opening and unlocking keystore file.



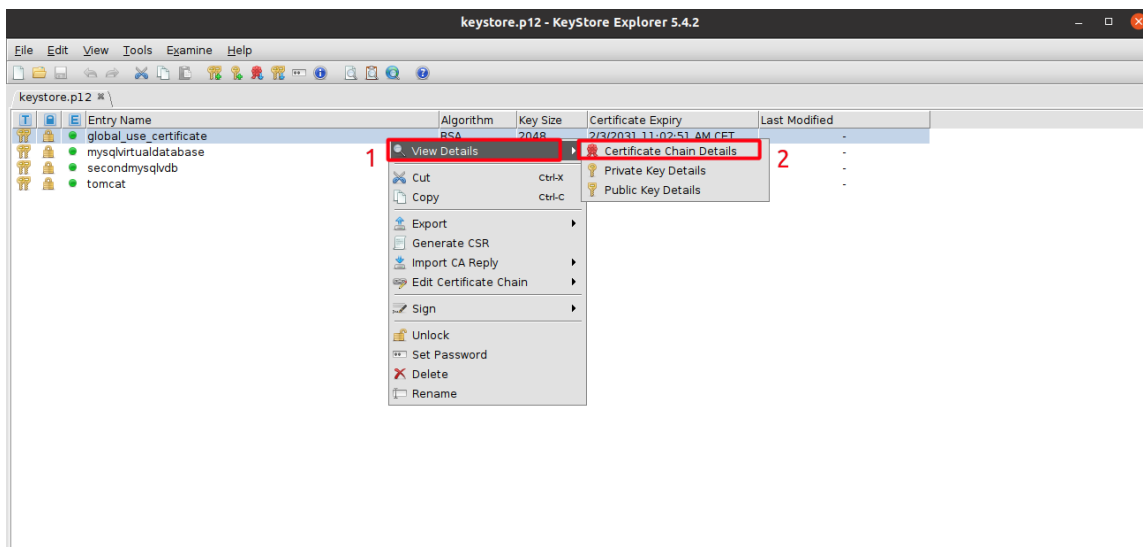
If key pair was generated with using DSA, then suggested is generating or importing key pair with used RSA algorithm to generate them.

1.2. Check CN value

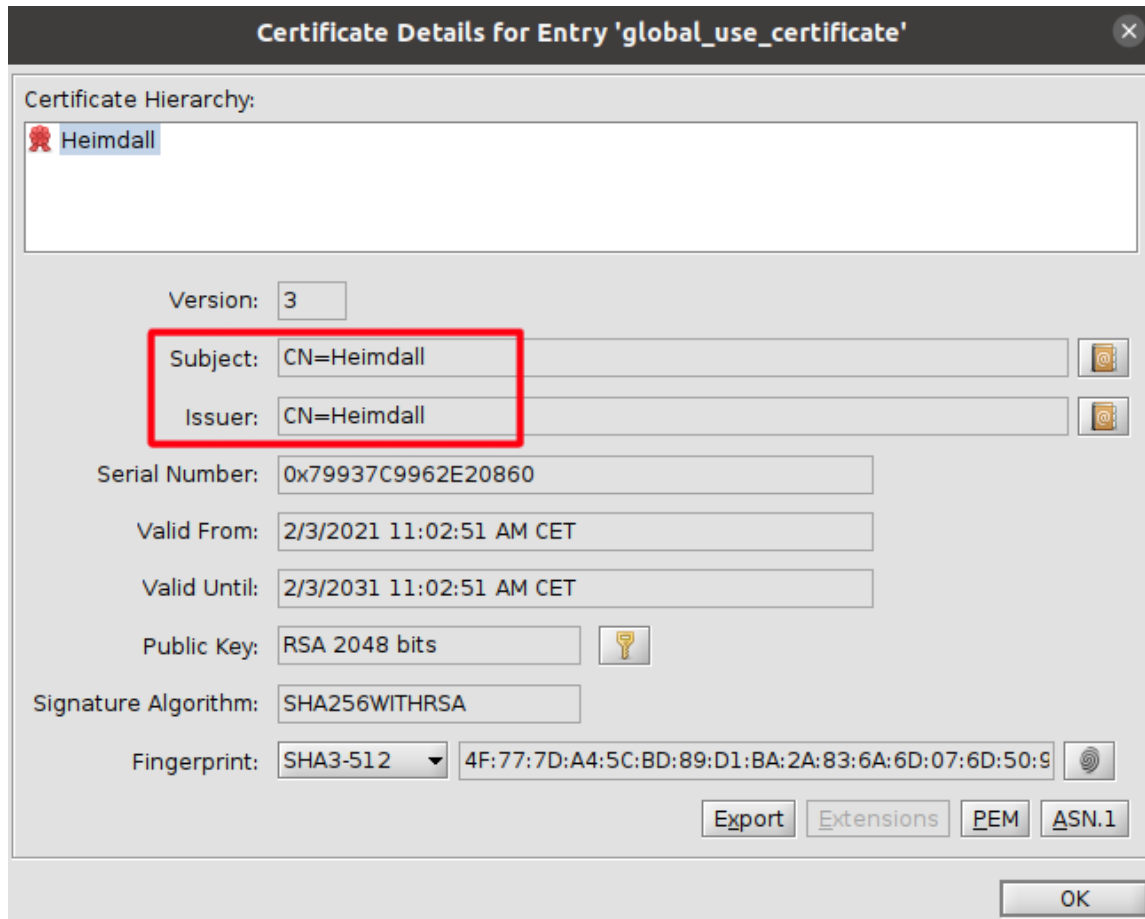
After checking algorithm, CN value used to sign certificate should be checked. There is a known requirement on PHP PDO using SSL that CN value should match or be possible to translate to an IP address.

First step to check CN value is checking Certificate Chain Details. To get access to that window, below steps have to be done:

1. Right click chosen key pair.
2. Choose option **View Details** (area 1 marked on the image below).
3. Click option **Certificate Chain Details** (area 2 marked on the image below).



After completing above steps, the window below should pop up. In the marked area is given information about certificate details of a key pair. By default, certificate generated by Heimdall is self-generated and CN value is matching value `Heimdall`. The CN value should be a proper IP address or be possible to translate to an IP address.

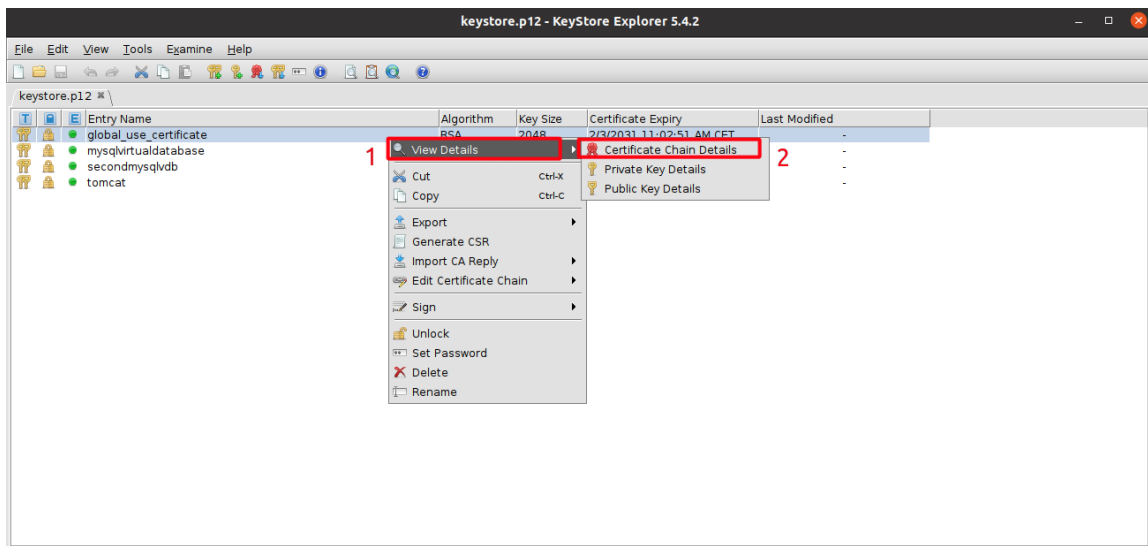


If key pair's CN value isn't a proper IP address and can't be translated to an IP address, then suggested is generating or importing key pair with CN value which meets these requirements.

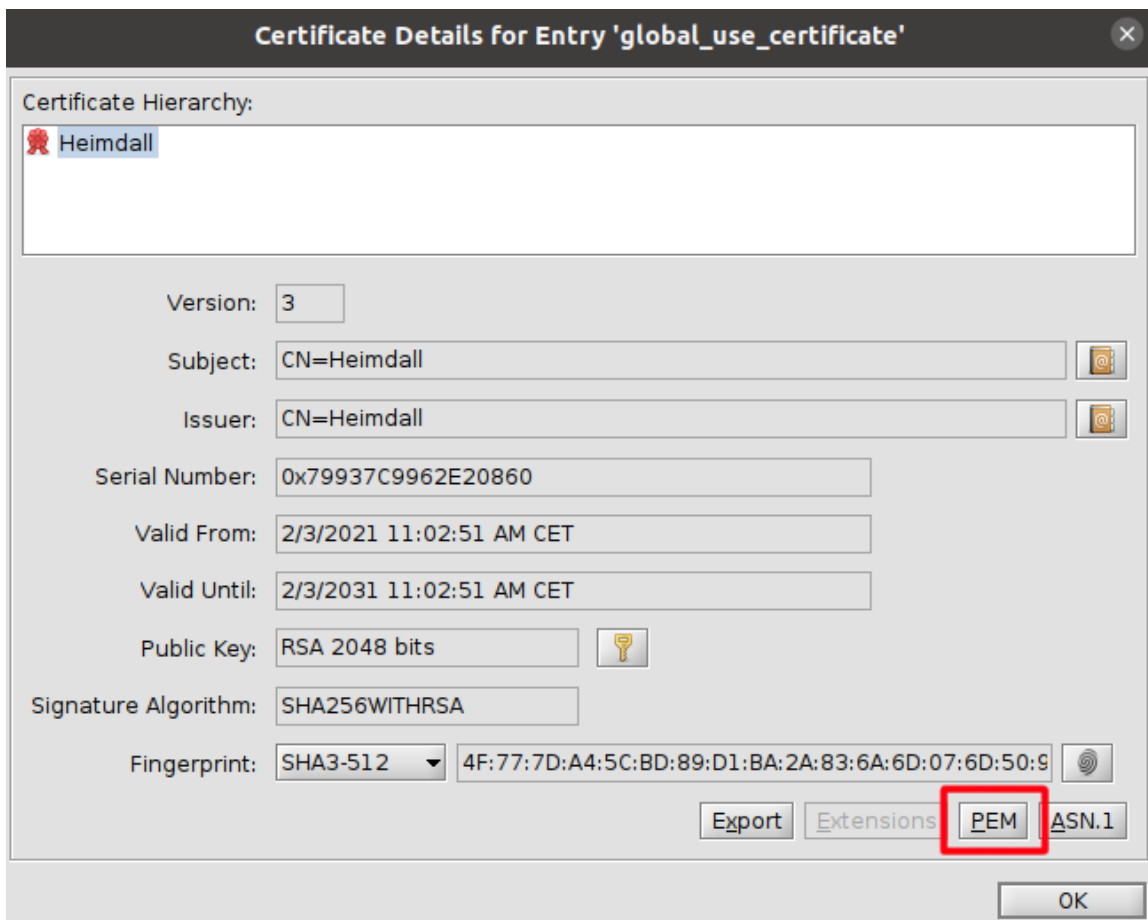
2. Export certificate to PEM format

After ensuring that certificate meets the requirements of PHP PDO's SSL certificate verification, the certificate can be finally exported to .pem file. To do that, first should be opened Certificate Chain Details window. To get access to that window, below steps have to be done:

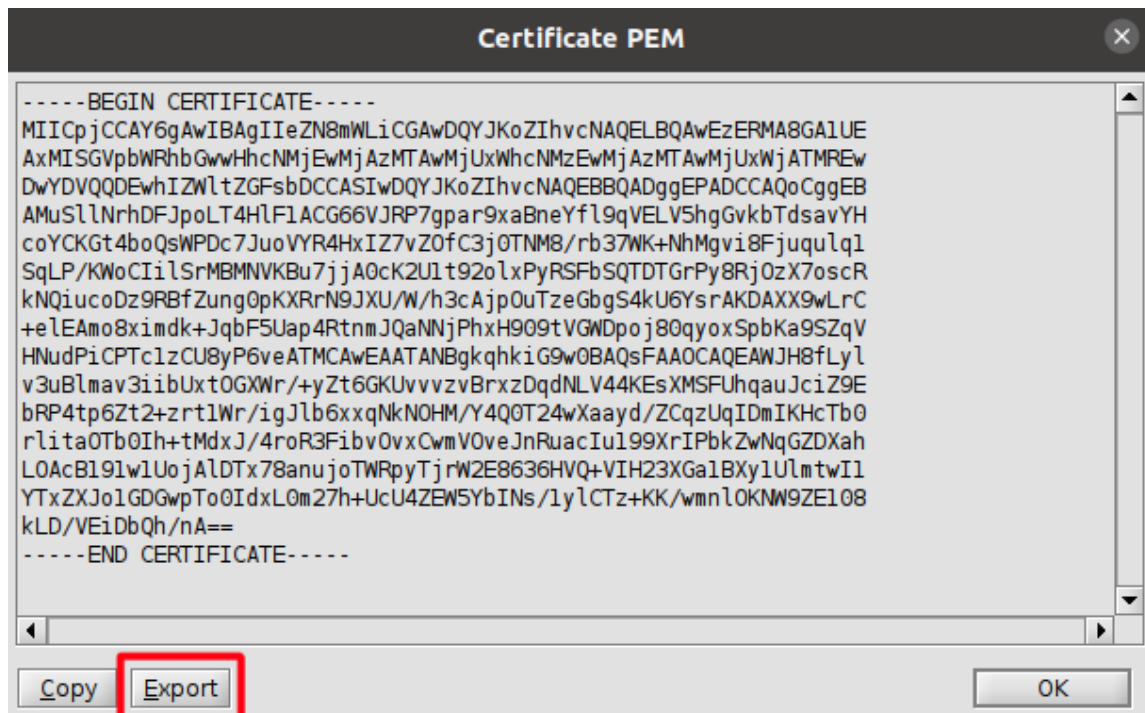
1. Right click chosen key pair.
2. Choose option `View Details` (area 1 marked on the image below).
3. Click option `Certificate Chain Details` (area 2 marked on the image below).



After completing above steps, below window should pop up. Next step is opening certificate view in PEM format. To do that, click marked on the below image button PEM .

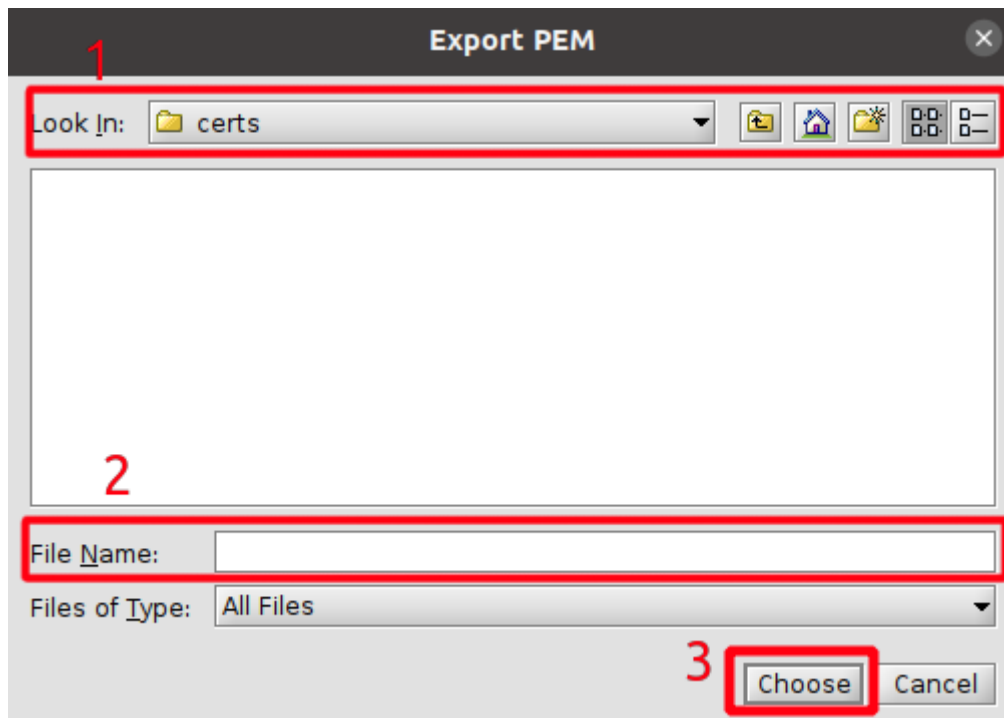


After clicking PEM button, below window should pop up. To begin exporting certificate in PEM format, button Export should be clicked (as marked on the image below).



After clicking **Export** button, below window should pop up. The last step is to save exported PEM certificate to the file. To do that:

1. Choose in what directory should the certificate be saved (by using controls in marked are 1).
2. Name a file with exported PEM file by writing name in area 2 marked on the image below. Remember that suffix `.pem` won't be added automatically, so can be added, if needed, during naming file.
3. Click button **Choose** in the marked area 3 on the image below.



After following above steps, the certificate should be exported in the PEM format to the chosen file.

Example configuration in PHP PDO application

After exporting certificate to a PEM format, it is ready to use in PGP PDO application. Below can be seen an example configuration in PHP code to set connection via PDO to a MySQL database, with defined SSL parameters.

```
$myPDO = new PDO('mysql:host=example.database.com;dbname=mysql', 'root',  
'password', array(  
    PDO::MYSQL_ATTR_SSL_KEY => '/example/keys/client-key.pem',  
    PDO::MYSQL_ATTR_SSL_CERT => 'certificates/client-cert.pem',  
    PDO::MYSQL_ATTR_SSL_CA => 'certificates/server-cert.pem'  
));
```

SSL parameters used above are used to declare:

- `PDO::MYSQL_ATTR_SSL_KEY` - declares a file with a key, to be used during SSL connection
- `PDO::MYSQL_ATTR_SSL_CERT` - declares a file with a certificate, to be used during SSL connection
- `PDO::MYSQL_ATTR_SSL_CA` - declares a file with a certificate, to be verified if matches server's certificate. A certificate exported in previous steps should be set here.

Above parameters declare file path. The file path can be written as absolute file path (as can be seen for parameter `PDO::MYSQL_ATTR_SSL_KEY`) or as relative file path (as can be seen for parameters `PDO::MYSQL_ATTR_SSL_CERT` and `PDO::MYSQL_ATTR_SSL_CA`).

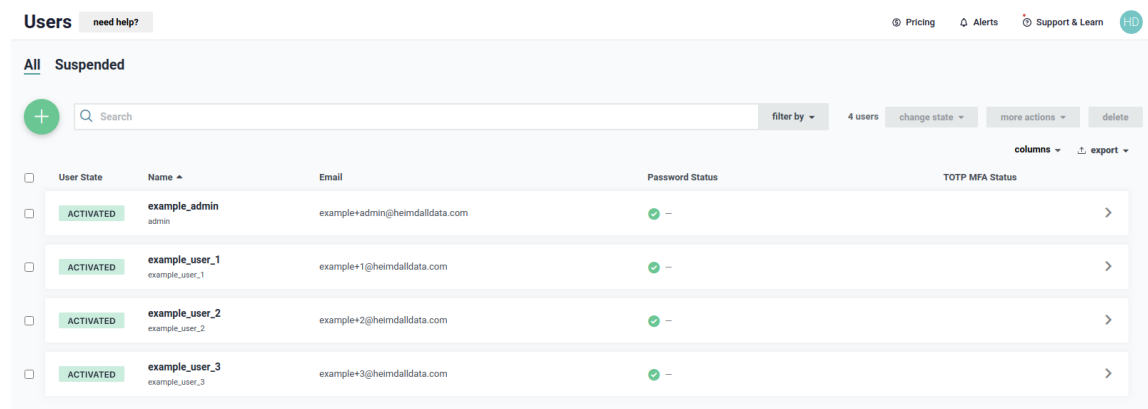
JumpCloud's LDAP authentication

This section describes how to configure Heimdall Proxy's LDAP authentication to work with JumpCloud's LDAP implementation.

Remember that for easy checking if configuration works in your environment, you can use "Authentication Test" in "Proxy Configuration" in Virtual DB configuration tab. By providing only a username and password, you can check if your actual configuration works and users can be authenticated as you expect. Other details, like IP address, database or using SSL, aren't needed to be filled to do a test for LDAP authentication mode.

Prerequisites

At the beginning, we will describe the JumpCloud LDAP configuration which will be used in this section. First part is setting users. As can be seen on the below screenshot, we have four users - `example_admin`, `example_user_1`, `example_user_2` and `example_user_3`.



The screenshot shows the 'Users' management page in JumpCloud. It features a table with columns for 'User State', 'Name', 'Email', 'Password Status', and 'TOTP MFA Status'. There are four users listed, all with an 'ACTIVATED' state and 'OK' password status. The first user, 'example_admin', has a username 'admin' which is used for LDAP authentication.

User State	Name	Email	Password Status	TOTP MFA Status
ACTIVATED	example_admin admin	example+admin@heimdalldata.com	OK	
ACTIVATED	example_user_1 example_user_1	example+1@heimdalldata.com	OK	
ACTIVATED	example_user_2 example_user_2	example+2@heimdalldata.com	OK	
ACTIVATED	example_user_3 example_user_3	example+3@heimdalldata.com	OK	

First user - `example_admin` has a details as shown below. Please notice that username is different from first name. Username of the user is `admin` and that name determines LDAP Distinguished Name.

The screenshot shows the 'Users' management interface. On the left, a sidebar lists users: 'example_admin', 'example_user_1', 'example_user_2', and 'example_user_3', all with 'ACTIVATED' status. The main panel displays the 'Details' for 'example_admin'. The 'User Information' section includes fields for First Name (example_admin), Middle Name, Last Name, Username (admin), Display Name, User Creation Date (2021-06-24T08:31:27.520Z), Company Email (example+admin@heimdalldata.com), LDAP Distinguished Name (uid=admin,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com), and a Description field.

User `example_admin` will be used for performing bind and search operations in JumpCloud LDAP, so we have to give that user a permission to do that, as shown below.

This screenshot shows the 'User Security Settings and Permissions' section for 'example_admin'. Under 'Password Settings', there are fields for Password and Confirm Password, and radio buttons for 'Password Expires On' and 'Password Never Expires'. There is also a checkbox for 'Never lock this user after failed logon attempts into their device'. Under 'Multi-Factor Authentication Settings', there is a checkbox for 'Require Multi-factor Authentication on the User Portal'. Under 'WebAuthn Security Keys', there is a '+ add key' button. Under 'Permission Settings', there are checkboxes for 'Enable as Admin/Sudo on all device associations' and 'Enable as LDAP Bind DN', which is currently checked.

We need to be sure that the user belong to the JumpCloud LDAP directory, as shown below, to use it during authentication.

The screenshot shows the 'Directories' tab for 'example_admin'. It states 'example_admin has the selected directories enabled:' and shows a table with 1 of 1 directories bound. The table has columns for Type, Name, Token Status, and Access Configurations. The single entry is 'JumpCloud LDAP' with a checked checkbox in the Type column and 'LDAP Bind DN' in the Access Configurations column.

The next user, `example_user_1`, has a details as shown below.

Users need help?

Pricing Alerts Support & Learn HD

All Suspended

+ Search

User State	Name
ACTIVATED	example_admin admin
ACTIVATED	example_user_1 example_user_1
ACTIVATED	example_user_2 example_user_2
ACTIVATED	example_user_3 example_user_3

example_user_1
example+1@heimdalldata.com

User State
States indicate where an account is in the user lifecycle. Assign a state to control when a user can access JumpCloud provided resources.
ACCOUNT ACTIVATED

Security Status
password active
password never expires

Highlights Details User Groups Devices Directories

User Information

First Name Middle Name Last Name
example_user_1

* Username * local user account Display Name *
example_user_1 example_user_1

User Creation Date * Company Email *
2021-07-09T07:04:51.251Z example+1@heimdalldata.com

LDAP Distinguished Name
uid=example_user_1,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com

Description

We need to be sure that the user belong to the JumpCloud LDAP directory, as shown below, to use it during authentication.

Users need help?

Pricing Alerts Support & Learn HD

All Suspended

+ Search

User State	Name
ACTIVATED	example_admin admin
ACTIVATED	example_user_1 example_user_1
ACTIVATED	example_user_2 example_user_2
ACTIVATED	example_user_3 example_user_3

example_user_1
example+1@heimdalldata.com

User State
States indicate where an account is in the user lifecycle. Assign a state to control when a user can access JumpCloud provided resources.
ACCOUNT ACTIVATED

Highlights Details User Groups Devices Directories

example_user_1 has the selected directories enabled:

1 of 1 directories bound

Type	Name	Token Status	Access Configurations
☒	JumpCloud LDAP		

The next user, `example_user_2`, has a details as shown below.

Users need help?

Pricing Alerts Support & Learn HD

All Suspended

+ Search

User State	Name
ACTIVATED	example_admin admin
ACTIVATED	example_user_1 example_user_1
ACTIVATED	example_user_2 example_user_2
ACTIVATED	example_user_3 example_user_3

example_user_2
example+2@heimdalldata.com

User State
States indicate where an account is in the user lifecycle. Assign a state to control when a user can access JumpCloud provided resources.
ACCOUNT ACTIVATED

Security Status
password active
password never expires

Highlights Details User Groups Devices Directories

User Information

First Name Middle Name Last Name
example_user_2

* Username * local user account Display Name *
example_user_2 example_user_2

User Creation Date * Company Email *
2021-07-09T07:36:59.800Z example+2@heimdalldata.com

LDAP Distinguished Name
uid=example_user_2,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com

Description

We need to be sure that the user belong to the JumpCloud LDAP directory, as shown below, to use it during authentication.

Users [need help?](#) [Pricing](#) [Alerts](#) [Support & Learn](#) [HD](#)

All **Suspended**

[+](#)

☐	User State	Name
☐	ACTIVATED	example_admin admin
☐	ACTIVATED	example_user_1 example_user_1
☐	ACTIVATED	example_user_2 example_user_2
☐	ACTIVATED	example_user_3 example_user_3



example_user_2
example+2@heimdalldata.com

User State
States indicate where an account is in the user lifecycle. Assign a state to control when a user can access JumpCloud provided resources.
ACCOUNT ACTIVATED

example_user_2 has the selected directories enabled:

☒	Type	Name	Token Status	Access Configurations
☒		JumpCloud LDAP		

1 of 1 directories bound

The next user, `example_user_3`, has a details as shown below.

Users [need help?](#) [Pricing](#) [Alerts](#) [Support & Learn](#) [HD](#)

All **Suspended**

[+](#)

☐	User State	Name
☐	ACTIVATED	example_admin admin
☐	ACTIVATED	example_user_1 example_user_1
☐	ACTIVATED	example_user_2 example_user_2
☐	ACTIVATED	example_user_3 example_user_3



example_user_3
example+3@heimdalldata.com

User State
States indicate where an account is in the user lifecycle. Assign a state to control when a user can access JumpCloud provided resources.
ACCOUNT ACTIVATED

Security Status

- password active - password never expires

User Information

First Name: Middle Name: Last Name:

* Username: [+ local user account](#) Display Name:

User Creation Date: * Company Email:

LDAP Distinguished Name:

Description:

We need to be sure that the user belong to the JumpCloud LDAP directory, as shown below, to use it during authentication.

Users [need help?](#) [Pricing](#) [Alerts](#) [Support & Learn](#) [HD](#)

All **Suspended**

[+](#)

☐	User State	Name
☐	ACTIVATED	example_admin admin
☐	ACTIVATED	example_user_1 example_user_1
☐	ACTIVATED	example_user_2 example_user_2
☐	ACTIVATED	example_user_3 example_user_3



example_user_3
example+3@heimdalldata.com

User State
States indicate where an account is in the user lifecycle. Assign a state to control when a user can access JumpCloud provided resources.
ACCOUNT ACTIVATED

example_user_3 has the selected directories enabled:

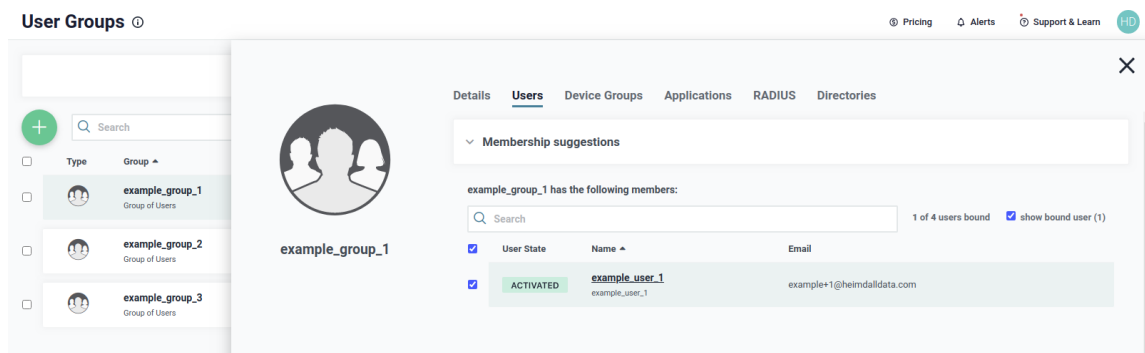
☒	Type	Name	Token Status	Access Configurations
☒		JumpCloud LDAP		

1 of 1 directories bound

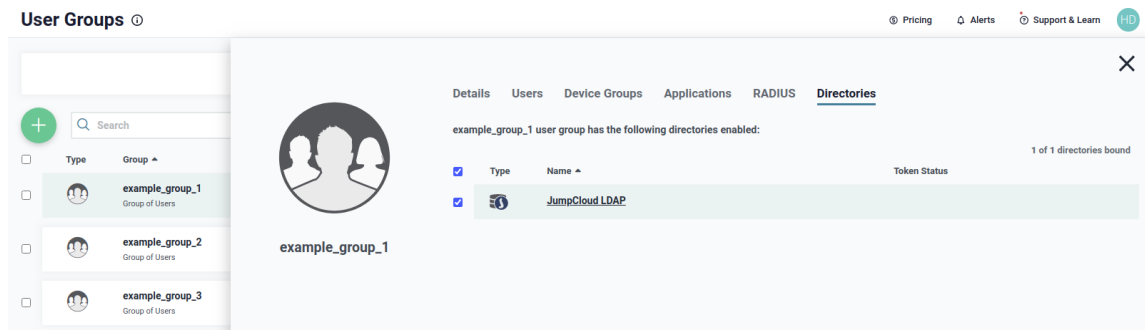
Next step of configuration is defining groups. For cases described here, three groups will be needed - `example_group_1`, `example_group_2` and `example_group_3`, as shown below.



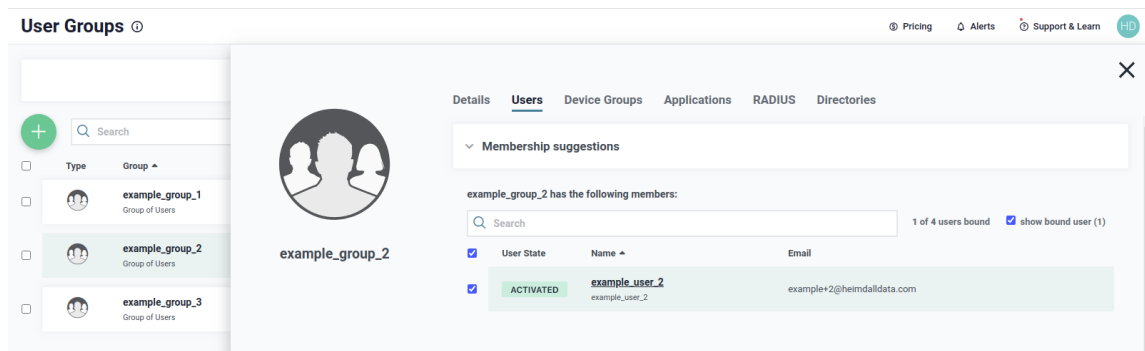
First group is `example_group_1`. The group has only one member, which is `example_user_1`, as shown below.



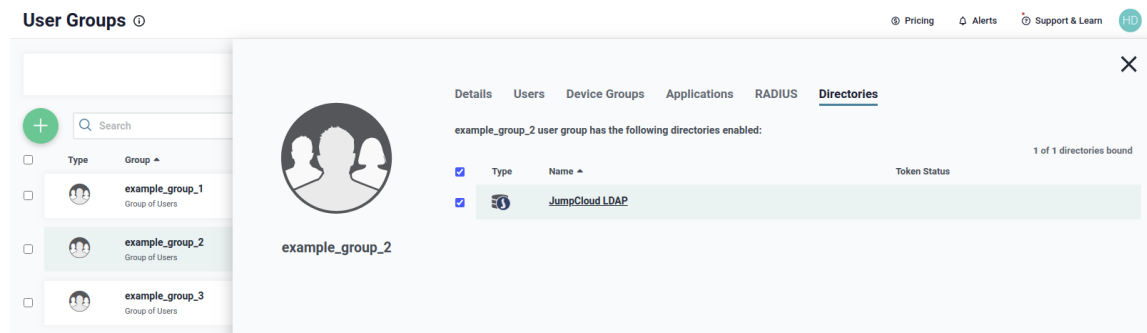
We need to be sure that the group belong to the JumpCloud LDAP directory, as shown below, to use it during authentication.



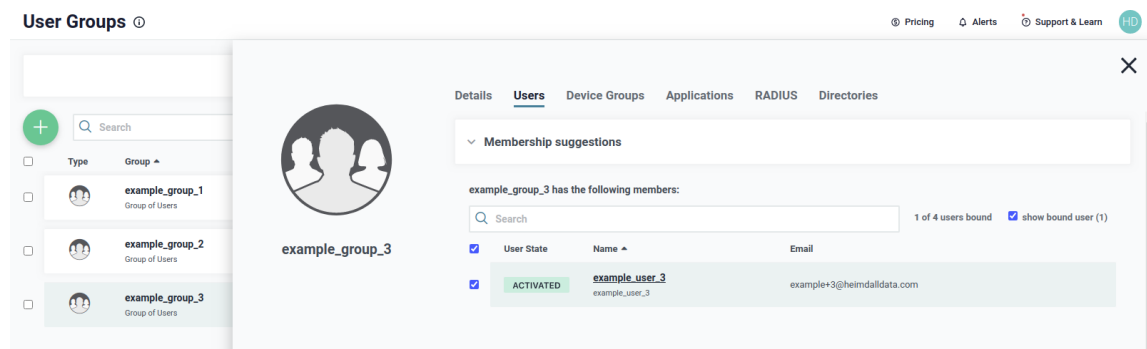
Next group is `example_group_2`. The group has only one member, which is `example_user_2`, as shown below.



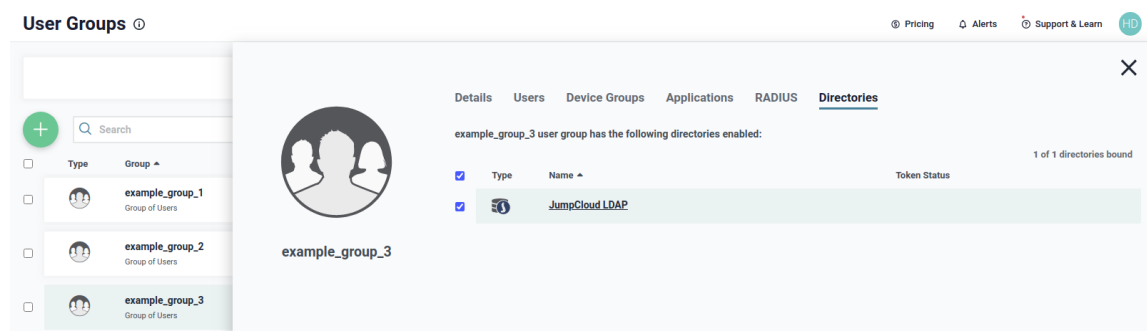
We need to be sure that the group belong to the JumpCloud LDAP directory, as shown below, to use it during authentication.



Next group is `example_group_3`. The group has only one member, which is `example_user_3`, as shown below.



We need to be sure that the group belong to the JumpCloud LDAP directory, as shown below, to use it during authentication.



More information about our objects can be retrieved by using tool named `ldapsearch`. By using the below line, the search will be done with:

- simple authentication instead of SASL (`-x`)
- using `uid=admin,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com` user as Distinguished Name to bind to LDAP directory (`-D` `"uid=admin,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com"`)
- with a prompt for simple authentication, instead specifying the password on the command line (`-W`)

- with URI specified to `ldap://ldap.jumpcloud.com:389`
(`-H ldap://ldap.jumpcloud.com:389`)
- with searchbase `ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com` (`-b ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com`)
- with scope set to subtree (`-s sub`)

```
ldapsearch -x -D
"uid=admin,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com" -W -H
ldap://ldap.jumpcloud.com:389 -b
"ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com" -s sub
```

Information about objects will give us details about attributes of each object. That information will be helpful during configuration of authentication.

Details about user `example_user_1` are as shown below.

```
# example_user_1, Users, 60d43f001430a71e78154864, jumpcloud.com
dn: uid=example_user_1,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: shadowAccount
objectClass: posixAccount
objectClass: jumpcloudUser
uid: example_user_1
uidNumber: 5003
homeDirectory: /home/example_user_1
mail: example+1@heimdalldata.com
sn: example_user_1
gidNumber: 5003
loginShell: /bin/bash
givenName: example_user_1
cn: example_user_1
displayName: example_user_1
memberOf:
cn=example_group_1,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com
```

Details about user `example_user_2` are as shown below.

```
# example_user_2, Users, 60d43f001430a71e78154864, jumpcloud.com
dn: uid=example_user_2,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com
uidNumber: 5004
homeDirectory: /home/example_user_2
mail: example+2@heimdalldata.com
givenName: example_user_2
sn: example_user_2
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```



```
objectClass: shadowAccount
objectClass: posixAccount
objectClass: jumpcloudUser
uid: example_user_2
gidNumber: 5004
loginShell: /bin/bash
cn: example_user_2
displayName: example_user_2
memberOf:
cn=example_group_2,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com
```

Details about user `example_user_3` are as shown below.

```
# example_user_3, Users, 60d43f001430a71e78154864, jumpcloud.com
dn: uid=example_user_3,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com
displayName: example_user_3
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: shadowAccount
objectClass: posixAccount
objectClass: jumpcloudUser
uidNumber: 5005
gidNumber: 5005
mail: example+3@heimdalldata.com
givenName: example_user_3
uid: example_user_3
loginShell: /bin/bash
homeDirectory: /home/example_user_3
sn: example_user_3
cn: example_user_3
memberOf:
cn=example_group_3,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com
```

Details about group `example_group_1` are as shown below.

```
# example_group_1, Users, 60d43f001430a71e78154864, jumpcloud.com
dn: cn=example_group_1,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com
ou: example_group_1
objectClass: top
objectClass: groupOfNames
description: tagGroup
cn: example_group_1
member:
uid=example_user_1,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com
```

Details about group `example_group_2` are as shown below.

```
# example_group_2, Users, 60d43f001430a71e78154864, jumpcloud.com
dn: cn=example_group_2,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com
cn: example_group_2
ou: example_group_2
objectClass: top
```

```
objectClass: groupOfNames
description: tagGroup
member:
uid=example_user_2,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com
```

Details about group `example_group_3` are as shown below.

```
# example_group_3, Users, 60d43f001430a71e78154864, jumpcloud.com
dn: cn=example_group_3,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com
cn: example_group_3
ou: example_group_3
objectClass: top
objectClass: groupOfNames
description: tagGroup
member:
uid=example_user_3,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com
```

Simple user authentication

To begin, let's assume that we want to give access to all users, which we defined in JumpCloud's LDAP. Additionally, we don't care if a user belong to a group or not. In this case, on the Heimdall Proxy's side, to authenticate a user, without looking at groups to which the user belongs, the best is LDAP Simple Mode authentication. An example of configuration to set proper authentication can be seen below.

The screenshot shows the 'LDAP Configuration' page. At the top, there's a dropdown menu set to 'Jumpcloud' and a 'Commit' button. Below this, the 'LDAP(S) URL' is 'ldap://ldap.jumpcloud.com:389'. The 'Server Type' is 'Jumpcloud'. 'Simple LDAP Mode' is checked with a blue checkbox. The 'LDAP Prefix' is 'uid=' and the 'LDAP Suffix' is ',ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=cc'. There are also buttons for 'LDAP Wizard' and 'Create New'.

As can be seen, the configuration specifies only two values: LDAP Prefix and LDAP Suffix. Both of them, with username of a user connecting to a proxy, will build a string, which is going to be used to connect to a JumpCloud LDAP. That single step is enough to perform an authentication in Simple Mode.

With above configuration, all users, so `example_user_1` , `example_user_2` and `example_user_3` , will be authenticated successfully (assuming that proper credentials, so username and password, were given).

Authenticating user from a group

A more complex case than previous one, would be a situation, when we care if a user belong to a group, but we don't care what group it is. For that situation the best will be LDAP Bind+Search Mode authentication without a group filter. An example of configuration to set proper authentication can be seen below.

LDAP Configuration

Jumpcloud

LDAP Wizard

Create New

Commit

LDAP Configuration

LDAP(S) URL: ldap://ldap.jumpcloud.com:389

Server Type: Jumpcloud

Simple LDAP Mode: ☐

LDAP Security Principal: uid=admin,ou=Users,o=60d43f001430a71e78154864,dc=jumpc

LDAP Search User Password:

LDAP Sec. Security Principal: CN=\${user},CN=users,DC=heimdalldata,DC=com

LDAP Sec. Search User Password: password

LDAP Search Domain: o=60d43f001430a71e78154864,dc=jumpcloud,dc=com

LDAP User Search Base: ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=co

LDAP Name Attribute: uid

LDAP Group Name Attribute: cn

LDAP Group Filter: (cn=pguser)

Use Raw Group Filter ☐

LDAP Healthcheck: ☐

Cache groups' emails: ☐

Management privileges: ☐

Authentication Checker

The configuration for Bind+Search Mode is more complex than for Simple Mode, so we are going to explain each property.

LDAP property	Value	Explanation
LDAP(S) URL	ldap://ldap.jumpcloud.com:389	Specifies URL of JumpCloud LDAP. In the example specified to a port without using SSL, but can be configured to an address using it.

LDAP property	Value	Explanation
LDAP Security Principal	uid=admin,ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com	Specifies user with enabled bind and search in JumpCloud LDAP service. For our case, it is a LDAP Distinguished Name of <code>example_admin</code> user.
LDAP Search User Password	(hidden)	Specifies password to authenticate as user specified in LDAP Security Principal. For our case, it is a password for <code>example_admin</code> user.
LDAP Search Domain	o=60d43f001430a71e78154864,dc=jumpcloud,dc=com	Specifies domain in which a user's group will be searched for. For our case, the value is as shown, without declaring any <code>ou</code> , because search will be done with subtree scope.
LDAP User Search Base	ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=com	Specifies base for searching a user to be authenticated. For our case, the value is as shown, with declaring <code>ou</code> as <code>Users</code> , because we will be looking for users.
LDAP Name Attribute	uid	Specifies attribute, which should be used to identify a user, so if value of the attribute is equal to a username connecting to a proxy. The value for our case is based on reads for users objects from <code>ldapsearch</code> results.
LDAP Group Name Attribute	cn	Specifies attribute, which should be used to extract information about a group, so the name of a group. For our case, we read from <code>ldapsearch</code> results that objects of group don't have <code>uuid</code> attribute, so we need to specify a different one. From <code>ldapsearch</code> result can be read, that <code>cn</code> value is unique for our groups, so we are going to use it to authenticate them.
LDAP Group Filter	(empty)	Specifies group filter, which will be added during a search for a user's groups. For this example we don't need to specify it. When this attribute is specified, filter is added during searching for a user's groups and the filter can specify what restrictions have to be met by a group, e.g. value of an attribute of a group object.
Use Raw Group Filter	false	Uses the LDAP Group Filter value as provided. No default group filters (e.g., <code>objectCategory/objectClass</code>), no security-group constraints (e.g., <code>sAMAccountType</code>), and no membership clauses (e.g., <code>member/memberUid</code> or nested membership) will be appended. You must provide a complete LDAP Group filter, using <code>{user}, \${dn}</code> .
Use nested groups filter	false	At the time of writing this section (8th July 2021), JumpCloud LDAP doesn't handle nested groups, so that option have to be turned off, because turned on can break authentication and return no results.

With above configuration, all users, so `example_user_1`, `example_user_2` and `example_user_3`, will be authenticated successfully (assuming that proper credentials, so username and password, were given). That will happen, because all of these users belongs to groups (user `example_user_1` is in group `example_group_1`, user `example_user_2` is in group `example_group_2`, user `example_user_3` is in group `example_group_3`). If there was a user without a group, that user would be rejected by authentication.

Authenticating user from a specified group

The most complex is when we want only to authenticate users from particular groups. First, let's assume that we only want to authenticate users from `example_group_1` group. For that situation the best will be to use LDAP Bind+Search Mode authentication with a group filter. An example of the configuration to accept users only from `example_group_1` group, can be seen below.

LDAP Configuration

LDAP Wizard

Create New

Jumpcloud

Commit

LDAP Configuration

LDAP(S) URL:

ldap://ldap.jumpcloud.com:389

Server Type:

Jumpcloud

Simple LDAP Mode:

☐

LDAP Security Principal:

uid=admin,ou=Users,o=60d43f001430a71e78154864,dc=jumpc

Secret

☐

LDAP Search User Password:

.....

LDAP Sec. Security Principal:

CN=\${user},CN=users,DC=heimdalldata,DC=com

Secret

☐

LDAP Sec. Search User Password:

password

LDAP Search Domain:

o=60d43f001430a71e78154864,dc=jumpcloud,dc=com

LDAP User Search Base:

ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=co

LDAP Name Attribute:

uid

LDAP Group Name Attribute:

cn

LDAP Group Filter:

(cn=example_group_1)

Use Raw Group Filter

☐

LDAP Healthcheck:

☐

Cache groups' emails:

☐

Management privileges:

☐

Authentication Checker

The configuration differs from a previous one by set value of LDAP Group Filter which specifies what value of `cn` attribute a group must have. Based on details read by using `ldapsearch` before, we know that `cn` value is unique between groups, so can be used for authenticating them. For `example_group_1` group, that value is `example_group_1`, so that's why group filter is `(cn=example_group_1)`. Using that configuration, only user `example_user_1` will be authenticated (assuming that proper credentials, so username and password, were given). For user `example_user_2` and `example_user_3` authentication will end up with a failure, during checking their groups.

Next situation is when we want to authenticate users from `example_group_1` or `example_group_2`. With that case we want to be sure that a user belong to any of these groups two groups. An example of configuration to accept users only from `example_group_1` or `example_group_2` group, can be seen below.

LDAP Configuration

LDAP Wizard

Create New

Jumpcloud

Commit

LDAP Configuration

LDAP(S) URL:

ldap://ldap.jumpcloud.com:389

Server Type:

Jumpcloud

Simple LDAP Mode:

☐

LDAP Security Principal:

uid=admin,ou=Users,o=60d43f001430a71e78154864,dc=jumpc

Secret

☐

LDAP Search User Password:

.....

LDAP Sec. Security Principal:

CN=\${user},CN=users,DC=heimdalldata,DC=com

Secret

☐

LDAP Sec. Search User Password:

password

LDAP Search Domain:

o=60d43f001430a71e78154864,dc=jumpcloud,dc=com

LDAP User Search Base:

ou=Users,o=60d43f001430a71e78154864,dc=jumpcloud,dc=co

LDAP Name Attribute:

uid

LDAP Group Name Attribute:

cn

LDAP Group Filter:

((cn=example_group_1)(cn=example_group_2))

Use Raw Group Filter

☐

LDAP Healthcheck:

☐

Cache groups' emails:

☐

Management privileges:

☐

Authentication Checker

Again, the only change is in value of LDAP Group Filter. Set filter specifies that `cn` value of a group has to match `example_group_1` or `example_group_2`, so only groups with these values will be used, and that mean that only users from these groups will be authenticated. For our case, only `example_user_1` and `example_user_2` will be authenticated (assuming that proper credentials, so username and password, were given). For user `example_user_3` authentication will end up with a failure, during checking their groups.

The example of the LDAP Group Filter can be extended to set restrictions on more arguments as needed. The main rule is that filter has to met requirements of LDAP filters used to search. The second rule is that group filter has to be in brackets. If group filter exists, is joined, by using AND operator, to filter checking if a user is a member of a group.

Odoo Specific Information

Odoo Transaction Behavior

Odoo operates by putting every request in a transaction, and further, specifies "repeatable read" as part of this. It is this transaction behavior that actually causes much of the overhead of Odoo on the database, and as a result, Heimdall is not able to do caching or read/write split by default, so removing the majority of benefits that Odoo users desire.

In order to resolve this, Heimdall provides the option for "delayed transactions", in the VDB advanced tab. The way this works is to trap when a transaction is desired, and actually NOT start the transaction then. If a DML (insert/update/delete/etc) is performed, then the transaction is started, and used until the transaction either commits or is rolled back. Technically, this behavior is in violation of the transaction "repeatable read" specification, in that even reads before a DML should read consistent data, but with this option, only after a DML is this correct. This has not been observed to cause issues with customers. If this is a concern, please note that NO solution can provide caching and/or read/write split without violating this in some way. Other customers are using this successfully without it causing errors, but if you have custom Odoo modules that depend on strict repeatable read behavior, please contact Heimdall support. We do have ways to change behaviors via rules so that if an issue is identified, we can modify the behavior for the queries in question, while providing benefit for others.

Configuration Items for Odoo

You can use a ready-made template, which should be selected in the wizard, or you can do it manually. If so, follow the instructions below.

First, configure the data source using the Wizard and verify connectivity. Then following these additional steps:

- In the postgresql.conf, set "default_transaction_isolation = 'repeatable read'" to set the default isolation level to repeatable read by default, avoiding the need for this to be set on connections. If you don't have access to the postgresql.conf or can't set it globally, you can alternatively set it at the database level with:

```
ALTER DATABASE SET DEFAULT_TRANSACTION_ISOLATION TO 'repeatable read';
```

- In the VDB, for Odoo, specify in the Advanced tab:
 - Multiplex with a timeout of 0
 - Delayed Transactions
- In the Data source tab:
 - Make sure the catalog field in the JDBC url is filled in with \${catalog}
 - In connection properties the defaultCatalog parameter is specified, generally with "odoo" as the default
 - Connection pooling is enabled
- In the rules tab, besides the normal caching and read/write split rules as needed, a rule with regex of "SET TRANSACTION ISOLATION LEVEL REPEATABLE READ", intrans NOT set, Action of Result, and responseInt of "1". This rule will prevent the commands to set the transaction isolation level from being sent to the server, which with delayed transactions will result in the server generating warnings in it's logs.

Edit Rule

Regex:

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

+

In-Transaction:

☐

Action:

Result

▼

Notes:

Notes

//

Parameters

+

responseInt

▼

1

//

×

Done

Internally, Heimdall will capture certain types of queries that appear to be operating against sequences (which Odoo does) and will automatically flag these as potential DML operations. This will be noted in the Analytics by some queries not getting cached that may otherwise

appear cachable. Please see the help section to debug caching for information on how to debug why a query is not cached.

Cache Validation

if there is concern about if stale data is being returned from the cache, Heimdall includes a feature to validate cache data, and report if it was stale. To do this, on the cache rule, specify the property of "revalidateOnly" and set to "true". This will bypass serving from the cache, but will still cache objects and compare the cached object with what was previously there. In the event an object is not byte-for-byte identical, then an alert will be generated on the GUI with the query that generated the issue. This can be used to generate cache bypass rules if there are issues.

HAProxy for Heimdall Proxy Load Balancing

Introduction

HAProxy is a reliable and high-performance TCP/HTTP load balancer that can be used to distribute traffic across multiple backend servers like Heimdall Proxy. This guide provides a step-by-step approach to installing and configuring HAProxy on an Ubuntu system for Heimdall Proxies load balancing.

Prerequisites

- A system running Linux distribution.
- Heimdall Proxy installed on the backend servers.

Step 1: Install HAProxy

First, update the package list and install HAProxy using the following command:

```
sudo apt update && sudo apt upgrade -y  
sudo apt install haproxy -y
```

Verify the installation:

```
haproxy -v
```

Enable HAProxy to start on system boot:

```
sudo systemctl enable haproxy
```

Step 2: Configure HAProxy for Heimdall Load Balancing

Edit HAProxy Configuration File

Open the HAProxy configuration file:

```
sudo nano /etc/haproxy/haproxy.cfg
```

Replace the existing content or append the following configuration (adjusting for hosts and IP addresses):

```
defaults
    mode        tcp

frontend ft_postgres
    bind *:5432
    default_backend bk_postgres

frontend stats
    mode http
    bind *:8404
    stats enable
    stats uri /stats
    stats refresh 10s

backend bk_postgres
    balance roundrobin
    option httpchk GET /status
    http-check expect status 200
    server pg-proxy1 192.168.90.4:5432 check port 80
    server pg-proxy2 192.168.90.5:5432 check port 80

global
    stats socket /var/run/haproxy.sock mode 600 level admin
```

Explanation:

- `mode tcp` ensures HAProxy works in TCP mode, required for PostgreSQL connections.
- `bind *:5432` instructs HAProxy to listen on port 5432.
- `balance roundrobin` enables equal distribution of connections across backend servers.
- `option httpchk GET /status` set the HTTP healthcheck URL
- `check port <port>` enables health checks to monitor backend server availability using the http health check on the specified port (as configured in the vdb advanced settings).

Save the file (`CTRL + X` , then `Y` and `Enter`).

Step 3: Restart and Verify HAProxy

Restart HAProxy to apply the new configuration:

```
sudo systemctl restart haproxy
```

Check HAProxy's status:

```
sudo systemctl status haproxy
```

Ensure HAProxy is listening on port 5432:

```
ss -tulnp | grep haproxy
```

Step 4: Test Heimdall Proxy Connection via HAProxy

To verify the setup, try connecting to Heimdall Proxy through HAProxy:

```
psql -h <haproxy_server_ip> -p 5432 -U <db_user> -d <database_name>
```

Replace `<haproxy_server_ip>` with the HAProxy host IP.

Replace `<db_user>` and `<database_name>` accordingly.

Optional configuration

With current code, Heimdall can serve DNS records from the manager that point to the proxy nodes attached. This allows HA Proxy to auto discover the proxy nodes and LB them. Here is an example configuration with this in place.

1. In the server properties, set the "DNS Port" to a value, but HA proxy will allow different ports to be used for this purpose, and others will allow normal DNS listeners to work without issue, such as with systemd-resolve on Ubuntu. As such, a value of 54 may be good. Make sure the auto-scaling mode is enabled, LB criteria of random or load, and proxy redirect name should be proxy private ip.
2. In the HA proxy configuration, add in a resolvers section, say for example:

```
resolvers postgres-vdb
  nameserver ns1 192.168.90.3:54
  accepted_payload_size 8192
```

For the backend section, swap the server line with a server-template line. Example

```
server-template pg-proxy 5 postgres-vdb.lab.heimdalldata.com:5432 check
port 80 resolvers postgres-vdb init-add none
```

This will allow DNS Round Robin to return up to 2 servers, by querying the name "postgres-vdb.lab.heimdalldata.com" from the nameserver(s) listed in the resolvers section named postgres-vdb. From here, if you have created a stats section for ha proxy, you should be able to

retrieve the current backend configuration and review the currently active nodes and their health. Example:

HAProxy version 2.8.5-1ubuntu3.3, released 2025/04/09

Statistics Report for pid 16290

> General process information

pid = 16290 (process #1, nbproc = 1, nbthread = 4)
uptime = 1d 16h40m52s; warnings = 581
system limits: memmax = unlimited; ulimit-n = 524288
maxsock = 524288; maxconn = 262112; reached = 0; maxpipes = 0
current conns = 1; current pipes = 0/0; conn rate = 0/sec; bit rate = 0.543 kbps
Running tasks: 0/26; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance

backup UP
backup UP, going down
backup DOWN, going up
not checked

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

• Scope :

• [Hide 'DOWN' servers](#)
• [Disable refresh](#)
• [Refresh now](#)
• [CSV export](#)
• [JSON export \(schema\)](#)

External resources:

• [Primary site](#)
• [Updates \(v2.8\)](#)
• [Online manual](#)

ft_postgres

		Queue			Session rate			Sessions					Bytes		Denied		Errors		Warnings		Server										
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend		0	0		0	0		0	0	0	262 112	0		0	0	0	0	0	0	0	0	0	OPEN								

stats

		Queue			Session rate			Sessions					Bytes		Denied		Errors		Warnings		Server										
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend		0	2		-	1	4	262	112	15				49 986	2 831 011	0	0	11					OPEN								

bk_postgres

		Queue			Session rate			Sessions					Bytes		Denied		Errors		Warnings		Server										
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
pg-proxy1		0	0	-	0	0		0	0		-	0	0	?	0	0	0	0	0	0	0	0	1d16h UP	L7OK/200 in 1ms	1/1	Y	-	0	1	0s	-
pg-proxy2		0	0	-	0	0		0	0		-	0	0	?	0	0	0	0	0	0	0	0	24s UP	L7OK/200 in 1ms	1/1	Y	-	0	144	2m24s	-
pg-proxy3		0	0	-	0	0		0	0		-	0	0	?	0	0	0	0	0	0	0	0	1d16h MAINT		1/1	Y	-		resolution	-	-
pg-proxy4		0	0	-	0	0		0	0		-	0	0	?	0	0	0	0	0	0	0	0	1d16h MAINT		1/1	Y	-		resolution	-	-
pg-proxy5		0	0	-	0	0		0	0		-	0	0	?	0	0	0	0	0	0	0	0	1d16h MAINT		1/1	Y	-		resolution	-	-
Backend		0	0		0	0		0	0	0	26 212	0	0	?	0	0	0	0	0	0	0	0	1d16h UP		2/2	2	0		1	0s	-

This configuration can be used in combination with Kubernetes or similar autoscaling function to automatically resize the cluster, and the haproxy and manager configurations will adapt automatically.

Here is a complete configuration block for a basic HA proxy with dynamic RR DNS discovery:

```
defaults
    mode      tcp

frontend ft_postgres
    bind *:5432
    default_backend bk_postgres

frontend stats
    mode http
    bind *:8404
    stats enable
    stats uri /stats
    stats refresh 10s

resolvers postgres-vdb
    nameserver ns1 192.168.90.3:54
    accepted_payload_size 8192

backend bk_postgres
    balance roundrobin
    option httpchk GET /status
    http-check expect status 200
    server-template pg-proxy 5 postgres-vdb.lab.heimdalldata.com:5432 check
    port 80 resolvers postgres-vdb init-addr none
    # server pg-proxy 192.168.90.4:5432 check port 80 resolvers postgres-vdb
```

```
global
    stats socket /var/run/haproxy.sock mode 600 level admin
```

When the central manager is in debug mode (via the admin->server properties tab), and DNS queries are being resolved, you can see the answers being logged:

```
2025-05-24T15:06:02.261404210Z DNS: Returning proxys: [
  "192.168.90.4",
  "192.168.90.5"
] to question: postgres-vdb.lab.heimdalldata.com. 0 IN AAAA from source: /192.168.90.6
```

Here, we can see that two IP addresses are being returned when HA proxy queries for them.

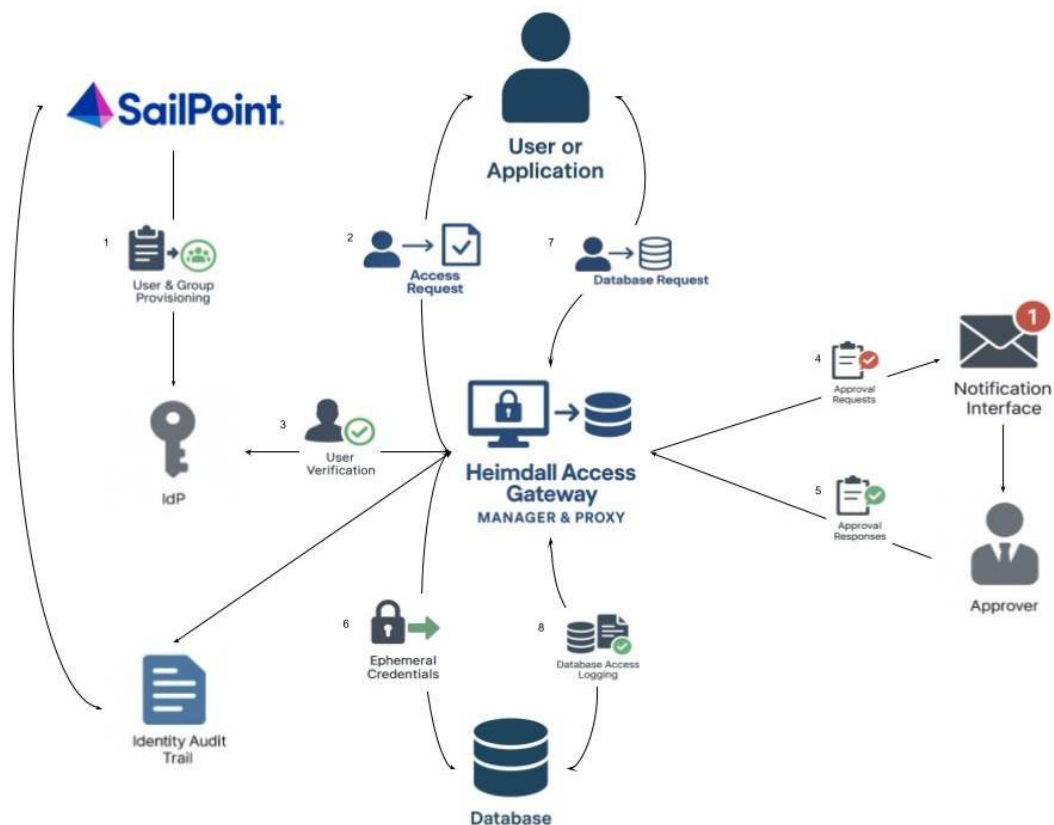
Note:

Heimdall Proxy instances running on `192.168.88.121` and `192.168.88.120`, ensuring high availability.

Conclusion

This guide provides the essential steps to install and configure HAProxy for PostgreSQL load balancing with Heimdall Proxy. By following these steps, you can ensure high availability, efficient load distribution, and failover support for your PostgreSQL database cluster.

Heimdall Access Gateway with SailPoint Integration



Objective

This document explains the flow of secure and audited database access using the integration of **Heimdall Access Gateway** with **SailPoint Identity Governance**. It ensures access is policy-driven, ephemeral (temporary), and fully traceable.

Overview of Components

- **SailPoint** – Identity management system responsible for provisioning and governance.
- **Heimdall Access Gateway** – Acts as a secure proxy between users/applications and databases.

- **Identity Provider (IdP)** – Verifies user identity before access is granted.
 - **Notification Interface (Heimdall Manager)** – Centralized dashboard managing approval workflows, notifications, and audit logs.
 - **Database** – Target system that users/applications need to access.
 - **Approver** – User or admin responsible for approving access requests.
-

Step-by-Step Access Flow

1. User and Group Provisioning

The process begins with SailPoint performing user and group provisioning. This includes:

- Creating identities in a central Identity Provider (IdP),
- Assigning roles and access policies,
- Ensuring the right individuals have the right access to the right systems.

This step guarantees that all users and groups are pre-configured and synchronized with SailPoint for lifecycle and access governance.

2. Access Request

Once the user is provisioned, they (or their application) initiate an access request to gain entry to a target database. This request is routed through the Heimdall Access Gateway which acts as an intermediary enforcing access control logic.

3. User Verification

The Heimdall Gateway communicates with the Identity Provider (IdP) to verify the user's identity:

- Using standard authentication protocols (SAML).
- Ensuring the user is authorized based on the identity attributes and role mappings provisioned by SailPoint.

Only verified users can proceed to the next stage.

4. Approval Request

If additional approval is required before database access, Heimdall sends an approval request to the designated approver via Heimdall Notification Interface:

- This can be an email or Heimdall workflow system.
- Triggers an alert to relevant stakeholders indicating a pending access request.

5. Approval Response

The approver reviews the access request and responds via the Heimdall notification interface:

- Approving or rejecting the request based on contextual information.
- The response is captured and routed back to Heimdall Access Gateway for enforcement.

6. Ephemeral Credentials

Upon receiving approval (if required), Heimdall dynamically generates ephemeral credentials:

- These are short-lived, one-time-use credentials to access the target database.
- Enhances security by avoiding static or long-term credential exposure.

These credentials are issued only for the duration and scope of the approved access.

7. Database Request

The user or application then submits their database query request using the ephemeral credentials:

- Heimdall proxies the request to the backend database.
- It ensures the request complies with access rules and usage policies.

8. Database Access Logging

All interactions between users and the database are logged by Heimdall:

- This includes query logs, timestamps, and user details.
- The logs are sent to an Identity Audit Trail system (e.g., integrated with SailPoint or SIEM).
- Facilitates compliance reporting, forensic analysis, and security audits.

Security and Compliance Highlights

- **Zero standing access:** Credentials are generated only when needed.
 - **Audit ready:** Every access is logged and traceable.
 - **Approval workflow integration:** Adds governance and oversight to sensitive operations.
 - **Seamless authentication:** Integrated with your existing Identity Provider (e.g., AD etc).
-

Auto-Scaling for Heimdall Proxies in VMware vSphere

This guide provides a lightweight approach to simulate auto-scaling for Heimdall Proxy virtual machines in a VMware vSphere environment using `govc` and Bash scripting. It covers VM setup, automation prerequisites, and a CPU-based scaling example to help optimize proxy performance during peak loads.

vSphere vs vCenter: A Simple Comparison Guide

vSphere is the overall VMware virtualization platform—a suite that includes components like ESXi (the hypervisor), vCenter Server, and the vSphere Client.

vCenter, on the other hand, is the centralized management tool within vSphere that lets you manage multiple ESXi hosts and virtual machines from one place.

In short:

vSphere = the whole platform,

vCenter = the control center for managing it.

Prerequisites

- **ESXi host installed** on bare metal and reachable on the network.
- **ESXi Host Client URL:** `https://<ESXi-IP>/ui`
Example (lab): `https://192.168.88.185/ui`
Credentials (example): `root / <password>`
- **vCenter Server Appliance (VCSA)** deployed as a VM on the ESXi host.
- **vSphere Client URL (inventory/VM mgmt):** `https://<vcenter-ip>/ui`
Example (lab): `https://192.168.88.176/ui`
SSO Admin (example): `administrator@<SSO-domain> / <password>`
Lab example: `administrator@heimdall.local / heimdall`

❗ **Tip:** The VAMI (appliance admin portal) is at `https://<vcenter-ip>:5480` for health, services, and updates (not for VM/host inventory).

Create a Datacenter and Add the ESXi Host

1. Log in to the vSphere Client: `https://<vcenter-ip>/ui`
2. Right-click the **vCenter object** (top of the left tree) → *New Datacenter*.
 - **Name:** e.g., `ha-datacenter`
3. Right-click the new **Datacenter** → *Add Host*
4. Enter the ESXi host IP (e.g., `192.168.88.185`) and root credentials.
5. Accept the SSL certificate and complete the wizard (license, lockdown mode, VM placement if prompted).

Validation: You should see the ESXi host under the Datacenter in **Hosts and Clusters**, and its VMs under **VMs and Templates**.

What is `govc` ?

To interact with vSphere from the command line, you'll need a tool that can communicate with VMware environments.

`govc` is a lightweight CLI built on the **govmomi Go library**, designed for automating and scripting tasks in vSphere environments such as ESXi or vCenter.

Think of `govc` as `kubectl`, but for vSphere.

`govc` installation steps on Linux

Run the following commands to install `govc` on a Linux machine (e.g., Heimdall Manager):

```
# Step 1: Download the latest release
curl -LO https://github.com/vmware/govmomi/releases/latest/download/
govc_Linux_x86_64.tar.gz

# Step 2: Extract the archive
tar -xvf govc_Linux_x86_64.tar.gz

# Step 3: Move the binary to your PATH
sudo mv govc /usr/local/bin/

# Step 4: Make it executable
chmod +x /usr/local/bin/govc
```

```
# Step 5: Verify the installation
govc version
```

Configure `govc` Environment Variables

`govc` reads its connection settings from environment variables. Export these once per shell session (or persist them in your shell profile).

```
# vCenter/ESXi endpoint
export GOVC_URL='https://<vcenter-ip>/sdk'

# SSO/Local credentials
export GOVC_USERNAME='administrator@<SSO-domain>'
export GOVC_PASSWORD='<password>'

# Accept self-signed certs (set to 0 if you have valid CA certs)
export GOVC_INSECURE=1

# Scope to your datacenter (recommended if you have more than one)
export GOVC_DATACENTER='<datacenter>' # e.g., ha-datacenter
```

Expect: vCenter/ESXi details, datacenter summary, and inventory roots.

Verify the connection

```
govc about           # shows vCenter/ESXi version
govc datacenter.info # shows your DC details
govc ls /            # lists inventory roots (DCs)
govc find -type m     # lists all VMs
```

Auto-scaling on vSphere

Auto-scaling on vSphere can be achieved with custom scripts (e.g., Bash + `govc`) that monitor a VM's resource usage and automatically clone or power on additional instances when defined thresholds are exceeded.

Example Auto-Scaling Logic

1. Monitor CPU and/or memory usage of VM-A.
2. If usage exceeds a defined threshold for X minutes:
 - Clone a new VM or power on a standby VM (e.g., VM-B) to help balance load.

Basic Auto-scaling (Bash): CPU-driven Clone

```
#!/bin/bash
set -euo pipefail

# --- govc environment (set these if the shell/service doesn't inherit your
env) ---
# export GOVC_URL='https://<vcenter>/sdk'
# export GOVC_USERNAME='<user>'      # administrator@<SSO-domain>
# export GOVC_PASSWORD='<pass>'
# export GOVC_INSECURE=1              # if you don't have proper CA trust
# export GOVC_DATACENTER='DC'        # optional; /ha-datacenter on standalone
ESXi

ACTIVE_PROXY_VM="HDPProxy000"
TEMPLATE_PROXY_VM="HDPProxyTemplate"
CPU_THRESHOLD=80
CHECK_INTERVAL=30    # seconds between checks when no clone action is taken

# Find the full inventory path (first match) for the ACTIVE VM (monitoring
target)
VM_PATH="$(govc find -type m -name "$ACTIVE_PROXY_VM" | head -n1 || true)"
if [[ -z "$VM_PATH" ]]; then
    echo "$(date): ERROR: VM '$ACTIVE_PROXY_VM' not found."
    exit 1
fi

running=true
isClone=false
# 30 minutes total, displayed per CHECK_INTERVAL: 1800 / 30 = 60 ticks
countDown=$(( 1800 / CHECK_INTERVAL ))

trap 'running=false' INT TERM
trap 'echo "$(date): Stopped."' EXIT

while $running; do
    # If we recently cloned, run a countdown and skip cloning
    if [[ "$isClone" == true && $countDown -gt 0 ]]; then
        printf "%s: Clone completed earlier; allowing load to rebalance (~%dm %ds
left).\n" \
            "$(date)" $(( (countDown*CHECK_INTERVAL)/60 )) $
            (( (countDown*CHECK_INTERVAL)%60 ))
        countDown=$((countDown - 1))
        sleep "$CHECK_INTERVAL"
        continue
    fi

    # If countdown finished, reset the flag
    if [[ "$isClone" == true && $countDown -le 0 ]]; then
        isClone=false
        countDown=$(( 1800 / CHECK_INTERVAL ))
    fi

    # Fetch the latest CPU usage sample
```

```

cpu_pct="$(govc metric.sample -n=1 "$VM_PATH" cpu.usage.average 2>/dev/null
\
    | awk ' $2=="-" && $3=="cpu.usage.average"{print $(NF-1)}' \
    | awk '{printf "%.0f", $1}')"
: "${cpu_pct:=0}"

echo "$(date): CPU usage for $ACTIVE_PROXY_VM is ${cpu_pct}%"

if [[ $cpu_pct -gt $CPU_THRESHOLD && "$isClone" == false ]]; then
    NEW_VM="HDProxy$(date +%s)" # or switch to HDProxy### pattern
    echo "$(date): CPU > ${CPU_THRESHOLD}% - cloning $TEMPLATE_PROXY_VM ->
$NEW_VM"
    govc vm.clone -vm "$TEMPLATE_PROXY_VM" -on=true "$NEW_VM"
    echo "$(date): Clone completed and powered on: $NEW_VM"

    # Start the 30-minute message window (no additional clones)
    isClone=true
    countDown=$(( 1800 / CHECK_INTERVAL ))
else
    echo "$(date): CPU normal. No action taken. Checking again in $
{CHECK_INTERVAL}s..."
    sleep "$CHECK_INTERVAL"
fi
done

```

⚠ Note:

- `ACTIVE_PROXY_VM="HDProxy000"` is the **running Heimdall Proxy** VM whose CPU usage is continuously monitored.
- `TEMPLATE_PROXY_VM="HDProxyTemplate"` is a **powered-off Heimdall proxy clone source**; when the threshold is exceeded, the script clones **from this template** and powers on the new VM.
- When CPU on `ACTIVE_PROXY_VM` exceeds **80%**, a new VM (e.g., `HDProxy<timestamp>`) is created via `govc vm.clone -vm "$TEMPLATE_PROXY_VM" -on=true`.
- The script runs in a loop and can be stopped gracefully; `trap 'running=false' INT TERM` lets `Ctrl+C` or a service stop signal exit the loop cleanly.
- After cloning, the script **enters a 30-minute quiet window** (no additional clones) and prints a status message each interval while load rebalances; monitoring then continues as normal.
- This script simulates basic **auto-scaling behavior** using `govc` on **vSphere**.

Sample Output

```

root@heimdall:/opt/heimdall# ./newhdcronjob.sh
Tue Aug  5 11:35:45 AM UTC 2025: CPU usage for HDProxy000 is 15%

```

```
Tue Aug 5 11:35:45 AM UTC 2025: CPU normal. No action taken. Checking again
in 30s...
Tue Aug 5 11:35:48 AM UTC 2025: CPU usage for HDProxy000 is 15%
Tue Aug 5 11:35:48 AM UTC 2025: CPU normal. No action taken. Checking again
in 30s...
Tue Aug 5 11:35:51 AM UTC 2025: CPU usage for HDProxy000 is 45%
Tue Aug 5 11:35:51 AM UTC 2025: CPU normal. No action taken. Checking again
in 30s...
Tue Aug 5 11:35:54 AM UTC 2025: CPU usage for HDProxy000 is 55%
Tue Aug 5 11:35:54 AM UTC 2025: CPU normal. No action taken. Checking again
in 30s...
```