severalnines

How to Deploy Open Source Databases

MariaDB MySQL PostgreSQL mongoDB

Table of Contents

Table of Contents

Introduction

Choosing which DB engine to use between all the options we have today is not an easy task. And that is just the beginning. After deciding which engine to use, you need to learn about it and actually deploy it to play with it. We plan to help you on that second step, and show you how to install, configure and secure some of the most popular open source DB engines. In this whitepaper we are going to cover these points, with the aim of fast tracking you on the deploy task.

Popular Vendors

Let’s take a look at some of the most popular open source database vendors, in no specific order.

Percona

Percona is a leading provider of unbiased open source database solutions that allow organizations to easily, securely and affordably maintain business agility, minimize risks, and stay competitive.

Percona is committed to producing open-source database software for MySQL and its variants, and develops their own version: Percona Server for MySQL, Percona XtraDB Cluster and Percona Server for MongoDB.

All Percona software is open source and free of charge.

MariaDB

MariaDB is a commercially supported fork of the MySQL relational database management system, intended to remain free and open-source software under the GNU GPL. The development of this engine is led by some of the original developers of MySQL, who forked it due to concerns over its acquisition by the Oracle Corporation.

MariaDB intends to maintain a compatibility with MySQL by using the same main storage engine. It includes the InnoDB storage engine, as well as a new storage engine, Aria, that intends to be both a transactional and non-transactional engine. There are some differences with MySQL to keep in mind like GTID, SQL syntax, encryption and tools compatibilities (e.g. xtrabackup).

Oracle MySQL

MySQL is a free and open-source RDBMS under the terms of the GNU General Public License, and is also available under a variety of proprietary licenses. MySQL was owned and developed by the Swedish company MySQL AB, which was bought by Sun Microsystems (now

Oracle Corporation). In 2010, when Oracle acquired Sun, Monty Widenius forked the open-source MySQL project to create MariaDB.

MySQL is a component of the LAMP web application software stack (and others), which is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. MySQL is used by many database-driven web applications, including WordPress, Drupal, and phpBB.

MongoDB

MongoDB was founded in 2007 by Dwight Merriman, Eliot Horowitz and Kevin Ryan.

MongoDB is the leading modern, general purpose database platform, designed to unleash the power of software and data for developers and the applications they build. MongoDB has more than 6,600 customers in more than 100 countries. The MongoDB database platform has been downloaded over 40 million times and there have been more than 1 million MongoDB University registrations.

PostgreSQL

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the POSTGRES project at the University of California at Berkeley and has more than 30 years of active development on the core platform.

PostgreSQL is developed by the PostgreSQL Global Development Group, a diverse group of many companies and individual contributors. It is free and open-source, released under the terms of the PostgreSQL License, a permissive software license.

How to Deploy Open Source Databases

As we could see, when choosing your storage engine, there are different vendors. Let's see some of the most common open source database servers manual installations and configurations.

Note: In some cases, the installation or configuration could be similar and maybe could look a bit repetitive, but we have added it anyway to allow you to follow a specific database server without the need to read the entire document.

Percona Server for MySQL

Percona Server for MySQL is a fully compatible, enhanced and open source drop-in replacement for any MySQL database. It is trusted by thousands of enterprises to provide better performance and concurrency for their most demanding workloads, and delivers greater value to MySQL server users with optimized performance, greater performance scalability and availability, enhanced backups and increased visibility.

Installation

In this link you have the latest packages to install Percona Server for MySQL.

If you prefer, you can follow the yum repository installation or the apt repository installation.

In our example, let's see the yum repository installation for a Percona Server on CentOS 7.

Percona Server IP Address: 192.168.100.117

```
[root@WP1 ~]# yum install http://www.percona.com/downloads/
percona-release/redhat/0.1-6/percona-release-0.1-6.noarch.
rpm
============================================================
===========================================
 Package                                   Arch      Version
Repository                              Size
============================================================
===========================================
Installing:
 percona-release                           noarch    0.1-6
```

```
/percona-release-0.1-6.noarch    16 k

Transaction Summary
============================================================
===========================================
Install  1 Package

[root@WP1 ~]# yum install Percona-Server-server-57
============================================================
===========================================
 Package                            Arch      Version
Repository                       Size
============================================================
===========================================
Installing:
 Percona-Server-server-57           x86_64    5.7.24-
26.1.el7    percona-release-x86_64            39 M
Installing for dependencies:
 Percona-Server-client-57           x86_64    5.7.24-
26.1.el7    percona-release-x86_64           6.8 M
 Percona-Server-shared-57           x86_64    5.7.24-
26.1.el7    percona-release-x86_64           748 k
 Percona-Server-shared-compat-57    x86_64    5.7.24-
26.1.el7    percona-release-x86_64           1.2 M
 libaio                             x86_64    0.3.109-13.el7
base                              24 k
 numactl-libs                       x86_64    2.0.9-7.el7
base                              29 k

Transaction Summary
============================================================
===========================================
Install  1 Package (+5 Dependent packages)
```

Now, you need to start the MySQL service.

```
[root@WP1 ~]# service mysql start
Redirecting to /bin/systemctl start mysql.service
```

A new password for the root user is created. To know it, you need to check the MySQL log and filter by temporary password. You should have something similar to this:

```
[root@WP1 ~]# grep “temporary password” /var/log/mysqld.log
2018-12-18T22:04:22.887873Z 1 [Note] A temporary password is
generated for root@localhost: W59c:%3/#sld
```

Then, you can run the mysql_secure_installation script, to configure a basic secure setup for your MySQL database.

```
[root@WP1 ~]# mysql_secure_installation
Securing the MySQL server deployment.

Enter password for user root:
The ‘validate_password’ plugin is installed on the server.
The subsequent steps will run with the existing configuration
of the plugin.
Using existing password for root.

Estimated strength of the password: 100
Change the password for root ? ((Press y|Y for Yes, any other key for No) : y

New password:
Re-enter new password:

Estimated strength of the password: 100
Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : y

By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.

Normally, root should only be allowed to connect from
‘localhost’. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named ‘test’ that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.

Remove test database and access to it? (Press y|Y for Yes, any other key for No) : y
 - Dropping test database...
Success.

```

```
 - Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!
```

Now your database is running, but it's not ready yet. It's just the beginning.

Default Configuration

By default, the Percona's my.cnf config file includes the /etc/my.cnf.d/ and /etc/percona-server.conf.d/ directories:

```
[root@WP1 ~]# cat /etc/my.cnf
#
# The Percona Server 5.7 configuration file.
#
#
# * IMPORTANT: Additional settings that can override those from this file!
#   The files must end with '.cnf', otherwise they'll be ignored.
#   Please make any edits and changes to the appropriate sectional files
#   included below.
#
!includedir /etc/my.cnf.d/
!includedir /etc/percona-server.conf.d/
```

The /etc/my.cnf.d/ directory is empty by default, and in the /etc/percona-server.conf.d/ we have the following content:

```
[root@WP1 ~]# ls /etc/percona-server.conf.d/
mysqld.cnf  mysqld_safe.cnf
```

mysqld_safe is the recommended way to start a mysqld server on Unix. It adds some safety features such as restarting the server when an error occurs and logging runtime information to an error log.

mysqld_safe reads options from both [mysqld] and [mysqld_safe] sections in the configuration files.

The content of these configuration files are:

- mysqld.cnf

```
[root@WP1 ~]# cat /etc/percona-server.conf.d/mysqld.cnf
# Percona Server template configuration
[mysqld]
#
# Remove leading # and set to the amount of RAM for the most important data
# cache in MySQL. Start at 70% of total RAM for dedicated server, else 10%.
# innodb_buffer_pool_size = 128M
#
# Remove leading # to turn on a very important data integrity option: logging
# changes to the binary log between backups.
# log_bin
#
# Remove leading # to set options mainly useful for reporting servers.
# The server defaults are faster for transactions and fast SELECTs.
# Adjust sizes as needed, experiment to find the optimal values.
# join_buffer_size = 128M
# sort_buffer_size = 2M
# read_rnd_buffer_size = 2M
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

- mysqld_safe.cnf

```
[root@WP1 ~]# cat /etc/percona-server.conf.d/mysqld_safe.cnf
#
# The Percona Server 5.7 configuration file.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.
#
```

```
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html
[mysqld_safe]
pid-file = /var/run/mysqld/mysqld.pid
socket   = /var/run/mysqld/mysqld.sock
nice     = 0
```

Let's see these parameters in detail.

- datadir: The path to the MySQL server data directory.
- socket: On Unix platforms, this variable is the name of the socket file that is used for local client connections.
- symbolic-links: Enable or disable symbolic link support. On Unix, enabling symbolic links means that you can link a MyISAM index file or data file to another directory with the INDEX DIRECTORY or DATA DIRECTORY option of the CREATE TABLE statement.
- log-error: Write the error log and startup messages to this file.
- pid-file: The path name of the file in which the server should write its process ID. The server creates the file in the data directory unless an absolute path name is given to specify a different directory.
- nice: Use the nice program to set the server's scheduling priority to the given value.

Optional Percona Server Configuration

Let's see some variables that we can add in our config files. It's recommended as a basic configuration, but it ought to be further tuned based on your infrastructure.

```
#GENERAL
user=mysql
basedir=/usr/
port=3306
skip_name_resolve
ignore-db-dir=lost+found
#LOGGING
log_warnings=2
slow_query_log_file=/var/log/mysql/mysql-slow.log
long_query_time=2
slow_query_log=OFF
log_queries_not_using_indexes=OFF
log_slow_admin_statements=ON
#INNODB
innodb_buffer_pool_size=128M
innodb_flush_log_at_trx_commit=2
innodb_file_per_table=1
innodb_data_file_path = ibdata1:100M:autoextend
innodb_read_io_threads=4
```

```
innodb_write_io_threads=4
innodb_doublewrite=1
innodb_log_file_size=64M
innodb_log_buffer_size=16M
innodb_buffer_pool_instances=1
innodb_log_files_in_group=2
innodb_thread_concurrency=64
innodb_flush_method = O_DIRECT
innodb_rollback_on_timeout=ON
innodb_autoinc_lock_mode=2
innodb_stats_on_metadata=0
default_storage_engine=innodb
#REPLICATION
server_id=1
binlog_format=ROW
log_bin=binlog
log_slave_updates=1
gtid_mode=ON
enforce_gtid_consistency=1
relay_log=relay-bin
expire_logs_days=7
read_only=ON
sync_binlog=1
report_host=192.168.100.117
master_info_repository=TABLE
relay_log_info_repository=TABLE
relay_log_recovery=ON
#OTHER THINGS
tmp_table_size = 64M
max_heap_table_size = 64M
max_allowed_packet = 512M
sort_buffer_size = 256K
read_buffer_size = 256K
read_rnd_buffer_size = 512K
myisam_sort_buffer_size = 8M
memlock=0
sysdate_is_now=1
max_connections=500
thread_cache_size=512
query_cache_type = 0
query_cache_size = 0
table_open_cache=1024
lower_case_table_names=0
```

To see in detail these variables, you can follow this link.

You can use the !include parameter, to split the configuration in different files, for example, the backup credentials.

Into /etc/percona-server.conf.d/mysqld.cnf add the following line:

```
!include /etc/percona-server.conf.d/secrets-backup.cnf
```

And then create the secrets-backup.cnf file:

```
[root@WP1 ~]# cat /etc/percona-server.conf.d/secrets-backup.cnf
# Security credentials for backup.
[mysqldump]
user=backupuser
password=Dse0s0k0ZvXoHItv
[xtrabackup]
user=backupuser
password=Dse0s0k0ZvXoHItv
```

Oracle MySQL Community Server

MySQL is the world's most popular open source database. With its proven performance, reliability, and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter and YouTube. Additionally, it is an extremely popular choice as embedded database, distributed by thousands of ISVs and OEMs.

Installation

To install the MySQL Community Server packages manually, you can follow this link.

Another way to install it is by using yum or apt repositories.

In our example, let's see the yum repository installation of MySQL Community Server 5.7 on CentOS 7.

MySQL Server IP Address: 192.168.100.118

```
[root@WP2 ~]# wget https://dev.mysql.com/get/mysql80-community-release-el7-1.noarch.rpm
--2019-01-03 01:41:20--  https://dev.mysql.com/get/mysql80-community-release-el7-1.noarch.rpm
Resolving dev.mysql.com (dev.mysql.com)... 137.254.60.11
Connecting to dev.mysql.com (dev.mysql.com)|137.254.60.11|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://repo.mysql.com//mysql80-community-release-el7-1.noarch.rpm [following]
--2019-01-03 01:41:22--  https://repo.mysql.com//mysql80-community-release-el7-1.noarch.rpm
```

```
Resolving repo.mysql.com (repo.mysql.com)... 23.208.182.226
Connecting to repo.mysql.com (repo.mysql.com)|23.208.182.226|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25820 (25K) [application/x-redhat-package-manager]
Saving to: 'mysql80-community-release-el7-1.noarch.rpm'

100%[=================================================================================================================================>] 25,820      --.-K/s   in 0.04s

2019-01-03 01:41:22 (698 KB/s) - 'mysql80-community-release-el7-1.noarch.rpm' saved [25820/25820]

[root@WP2 ~]# rpm -Uvh mysql80-community-release-el7-1.noarch.rpm
warning: mysql80-community-release-el7-1.noarch.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:mysql80-community-release-el7-1  ################################# [100%]
```

Edit the /etc/yum.repos.d/mysql-community.repo file and set the enabled parameter to 1 for MySQL 5.7 and 0 for MySQL 8.0:

```
# Enable to use MySQL 5.7
[mysql57-community]
name=MySQL 5.7 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/el/7/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
[mysql80-community]
name=MySQL 8.0 Community Server
baseurl=http://repo.mysql.com/yum/mysql-8.0-community/el/7/$basearch/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

And then, install mysql-community-server:

```
[root@WP2 ~]# yum install mysql-community-server
==================================================================================================
 Package                            Arch      Version
Repository                         Size
============================================================
===========================================
Installing:
 mysql-community-server             x86_64    5.7.24-1.el7
mysql57-community                  165 M
Installing for dependencies:
 libaio                             x86_64    0.3.109-13.el7
base                                24 k
 mysql-community-client             x86_64    5.7.24-1.el7
mysql57-community                   24 M
 mysql-community-common             x86_64    5.7.24-1.el7
mysql57-community                  274 k
 mysql-community-libs               x86_64    5.7.24-1.el7
mysql57-community                  2.2 M
 numactl-libs                       x86_64    2.0.9-7.el7
base                                29 k

Transaction Summary
============================================================
===========================================
Install  1 Package (+5 Dependent packages)
```

Now, you need to start the MySQL service.

```
[root@WP2 ~]# service mysqld start
Redirecting to /bin/systemctl start mysqld.service
```

A new password for the root user is created. To know it, you need to check the MySQL log and filter by temporary password. You should have something similar to this:

```
[root@WP2 ~]# grep “temporary password” /var/log/mysqld.log
2019-01-03T01:52:30.896979Z 1 [Note] A temporary password is
generated for root@localhost: Dx6!MWC5cKYg
```

Then, you can run the mysql_secure_installation script, to configure a basic secure setup for your MySQL database.

```
[root@WP2 ~]# mysql_secure_installation
Securing the MySQL server deployment.

Enter password for user root:
The ‘validate_password’ plugin is installed on the server.
The subsequent steps will run with the existing configuration
of the plugin.
Using existing password for root.

Estimated strength of the password: 100
```

```
Change the password for root ? ((Press y|Y for Yes, any other key for No) : y

New password:

Re-enter new password:

Estimated strength of the password: 100
Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : y
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.

Normally, root should only be allowed to connect from
‘localhost’. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named ‘test’ that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.

Remove test database and access to it? (Press y|Y for Yes, any other key for No) : y
 - Dropping test database...
Success.

 - Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!
```

Now your database is running, but it's not ready yet. We need to configure it.

Default Configuration

The MySQL installation creates the my.cnf config file and the /etc/my.cnf.d/ directory in /etc/.

The /etc/my.cnf.d/ directory is empty by default, and the content of my.cnf is:

```
[root@WP2 ~]# cat /etc/my.cnf
# For advice on how to change settings please see
# http://dev.mysql.com/doc/refman/5.7/en/server-configuration-defaults.html
[mysqld]
#
# Remove leading # and set to the amount of RAM for the most important data
# cache in MySQL. Start at 70% of total RAM for dedicated server, else 10%.
# innodb_buffer_pool_size = 128M
#
# Remove leading # to turn on a very important data integrity option: logging
# changes to the binary log between backups.
# log_bin
#
# Remove leading # to set options mainly useful for reporting servers.
# The server defaults are faster for transactions and fast SELECTs.
# Adjust sizes as needed, experiment to find the optimal values.
# join_buffer_size = 128M
# sort_buffer_size = 2M
# read_rnd_buffer_size = 2M
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

Let's see these parameters in detail.

- datadir: The path to the MySQL server data directory.
- socket: On Unix platforms, this variable is the name of the socket file that is used for local client connections.
- symbolic-links: Enable or disable symbolic link support. On Unix, enabling

symbolic links means that you can link a MyISAM index file or data file to another directory with the INDEX DIRECTORY or DATA DIRECTORY option of the CREATE TABLE statement.

- log-error: Write the error log and startup messages to this file.
- pid-file: The path name of the file in which the server should write its process ID. The server creates the file in the data directory unless an absolute path name is given to specify a different directory.

Optional MySQL Community Server Configuration

Let's see some variables that we can add in our config files. It's recommended as a basic configuration, but ought to be tuned based on your infrastructure.

```
#GENERAL
user=mysql
basedir=/usr/
port=3306
skip_name_resolve
ignore-db-dir=lost+found
#LOGGING
log_warnings=2
slow_query_log_file=/var/log/mysql/mysql-slow.log
long_query_time=2
slow_query_log=OFF
log_queries_not_using_indexes=OFF
log_slow_admin_statements=ON
#INNODB
innodb_buffer_pool_size=128M
innodb_flush_log_at_trx_commit=2
innodb_file_per_table=1
innodb_data_file_path = ibdata1:100M:autoextend
innodb_read_io_threads=4
innodb_write_io_threads=4
innodb_doublewrite=1
innodb_log_file_size=64M
innodb_log_buffer_size=16M
innodb_buffer_pool_instances=1
innodb_log_files_in_group=2
innodb_thread_concurrency=64
innodb_flush_method = O_DIRECT
innodb_rollback_on_timeout=ON
innodb_autoinc_lock_mode=2
innodb_stats_on_metadata=0
default_storage_engine=innodb
#REPLICATION
server_id=1
binlog_format=ROW
log_bin=binlog
log_slave_updates=1
```

```
gtid_mode=ON
enforce_gtid_consistency=1
relay_log=relay-bin
expire_logs_days=7
read_only=ON
sync_binlog=1
report_host=192.168.100.118
master_info_repository=TABLE
relay_log_info_repository=TABLE
relay_log_recovery=ON
#OTHER THINGS
tmp_table_size = 64M
max_heap_table_size = 64M
max_allowed_packet = 512M
sort_buffer_size = 256K
read_buffer_size = 256K
read_rnd_buffer_size = 512K
myisam_sort_buffer_size = 8M
memlock=0
sysdate_is_now=1
max_connections=500
thread_cache_size=512
query_cache_type = 0
query_cache_size = 0
table_open_cache=1024
lower_case_table_names=0
```

To see in detail these variables, you can follow this link.

You can use the !include parameter, to split the configuration in different files, for example, the backup credentials.

In /etc/my.cnf, add the following line:

```
!include /etc/my.cnf.d/secrets-backup.cnf
```

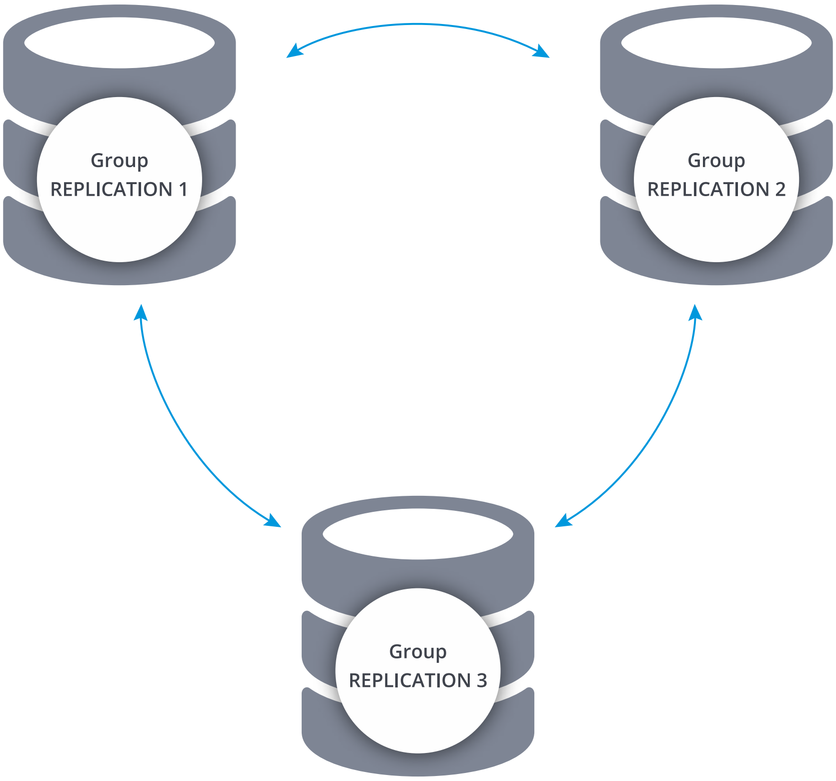
And then create the secrets-backup.cnf file:

```
[root@WP2 ~]# cat /etc/my.cnf.d/secrets-backup.cnf
# Security credentials for backup.
[mysqldump]
user=backupuser
password=DseOs0k0ZvXoHItv
[xtrabackup]
user=backupuser
password=DseOs0k0ZvXoHItv
```

Group Replication

MySQL Group Replication is a MySQL Server plugin that enables you to create elastic, highly-available, fault-tolerant replication topologies. It can operate in a single-primary mode with automatic primary election, where only one server accepts updates at a time. Alternatively, for more advanced users, groups can be deployed in multi-primary mode, where all servers can accept updates, even if they are issued concurrently.

Let's see the basic configuration:



Node 1 IP Address: 192.168.100.186

Node 2 IP Address: 192.168.100.187

Node 3 IP Address: 192.168.100.188

In each node, add the following lines in the my.cnf file:

```
# GROUP REPLICATION
server_id=1
gtid_mode=ON
enforce_gtid_consistency=ON
master_info_repository=TABLE
relay_log_info_repository=TABLE
binlog_checksum=NONE
log_slave_updates=ON
log-bin=binlog
binlog_format=ROW
plugin-load=group_replication.so
transaction_write_set_extraction=XXHASH64
```

```
loose-group_replication_group_name=”4bd8d654-a40d-4792-8b88-16eff8b85f44”
loose-group_replication_start_on_boot=off
loose-group_replication_local_address=”192.168.100.186:33066”
loose-group_replication_group_sees=”192.168.100.186:33066,192.168.100.187:33066,192.168.100.188:33066”
loose-group_replication_bootstrap_group=off
loose-group_replication_single_primary_mode=FALSE
loose-group_replication_enforce_update_everywhere_checks=TRUE
loose-group_replication_ip_whitelist=192.168.100.0/24
loose-group_replication_recovery_retry_count=3
loose-group_replication_recovery_reconnect_interval=120
```

You need to change the parameters server_id and loose-group_replication_local_address according to each node, and restart the MySQL service to take the changes.

```
[root@WP2 ~]# service mysqld restart
Redirecting to /bin/systemctl restart mysqld.service
```

You can verify if the group_replication plugin is installed by running the following command in the MySQL server:

```
mysql> SHOW PLUGINS;
*************************** 45. row ***************************
   Name: group_replication
 Status: ACTIVE
   Type: GROUP REPLICATION
Library: group_replication.so
License: GPL
```

If you can’t see the plugin, you can install it by using the following command:

```
mysql> INSTALL PLUGIN group_replication SONAME ‘group_replication.so’;
```

Then, you need to create the replication user and assign it to the Group Replication in each node:

```
mysql> CREATE USER rep_user@’%’ IDENTIFIED BY ‘rep_pass’;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> GRANT REPLICATION SLAVE ON *.* TO rep_user@’%’;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> CHANGE MASTER TO MASTER_USER=’rep_user’, MASTER_PASSWORD=’rep_pass’ FOR CHANNEL ‘group_replication_recovery’;
Query OK, 0 rows affected, 2 warnings (0.05 sec)
```

Now, in the first node, you need to initialize the cluster:

```
mysql> SET GLOBAL group_replication_bootstrap_group=ON;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> START GROUP_REPLICATION;
Query OK, 0 rows affected (2.11 sec)
```

```
mysql> SET GLOBAL group_replication_bootstrap_group=OFF;
Query OK, 0 rows affected (0.00 sec)
```

And check if the cluster is up:

```
mysql> SELECT * FROM performance_schema.replication_group_members\G
*************************** 1. row ***************************
CHANNEL_NAME: group_replication_applier
   MEMBER_ID: a3de1ba4-35eb-11e9-bbde-067c0c0a7c38
 MEMBER_HOST: Host28
 MEMBER_PORT: 3306
MEMBER_STATE: ONLINE
1 row in set (0.01 sec)
```

Then, you must start the Group Replication in the rest of the nodes:

```
mysql> START GROUP_REPLICATION;
Query OK, 0 rows affected (5.85 sec)
```

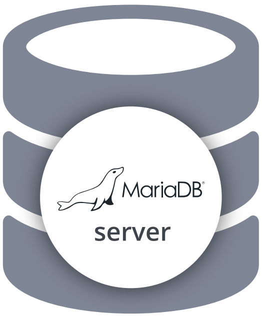
And check the cluster status again:

```
mysql> SELECT * FROM performance_schema.replication_group_members\G
*************************** 1. row ***************************
CHANNEL_NAME: group_replication_applier
   MEMBER_ID: a3de1ba4-35eb-11e9-bbde-067c0c0a7c38
 MEMBER_HOST: Host28
 MEMBER_PORT: 3306
MEMBER_STATE: ONLINE
*************************** 2. row ***************************
```

```
CHANNEL_NAME: group_replication_applier
   MEMBER_ID: a5ae763f-35eb-11e9-bcf8-2ab9ec7193a3
 MEMBER_HOST: Host29
 MEMBER_PORT: 3306
MEMBER_STATE: ONLINE
*************************** 3. row
***************************
CHANNEL_NAME: group_replication_applier
   MEMBER_ID: a6bae589-35eb-11e9-bc18-aeb046c0331d
 MEMBER_HOST: Host30
 MEMBER_PORT: 3306
MEMBER_STATE: ONLINE
3 rows in set (0.00 sec)
```

Now, you have your Group Replication up and running.

MariaDB



MariaDB Server is one of the most popular database servers. It's made by the original developers of MySQL and guaranteed to stay open source.

MariaDB turns data into structured information in a wide array of applications, ranging from banking to websites. It is an enhanced, drop-in replacement for MySQL. MariaDB is used because it is fast, scalable and robust, with a rich ecosystem of storage engines, plugins and many other tools which make it very versatile for a wide variety of use cases.

MariaDB Cluster is a synchronous multi-master cluster based on Galera replication. It is available on Linux only, and only supports the XtraDB/InnoDB storage engines

These installation steps are for both MariaDB Server and MariaDB Cluster. MariaDB uses the same binaries for both databases. The difference between them is the configuration parameters. We're going to see how to configure both in some minutes.

Installation

In this link you can download the latest packages to install MariaDB, or if you prefer, you can follow the repository installation link.

In our example, let's see the installation of MariaDB on CentOS 7 from the repository.

IP Address: 192.168.100.143

To add the MariaDB repository, you can run:

```
cat > /etc/yum.repos.d/MariaDB.repo <<- EOF
# MariaDB 10.3 CentOS repository
```

```
[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/10.3/centos7-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
EOF
```

And then, install MariaDB-server and MariaDB-client packages:

```
[root@WP3 ~]# yum install MariaDB-server MariaDB-client
============================================================
===========================================
 Package                            Arch      Version
Repository             Size
============================================================
===========================================
Installing:
 MariaDB-client                     x86_64    10.3.11-1.el7.
centos          mariadb                53 M
 MariaDB-server                     x86_64    10.3.11-1.el7.
centos          mariadb               123 M
Installing for dependencies:
 MariaDB-common                     x86_64    10.3.11-1.el7.
centos          mariadb               157 k
 MariaDB-compat                     x86_64    10.3.11-1.el7.
centos          mariadb               2.8 M
 boost-program-options              x86_64    1.53.0-27.el7
base                  156 k
 galera                             x86_64    25.3.24-1.
rhel7.el7.centos    mariadb                   8.1 M
 libaio                             x86_64    0.3.109-13.el7
base                   24 k
 lsof                               x86_64    4.87-6.el7
base                  331 k
 make                               x86_64    1:3.82-23.el7
base                  420 k
 openssl                            x86_64    1:1.0.2k-16.
el7                 base                    493 k
 perl-Compress-Raw-Bzip2            x86_64    2.061-3.el7
base                   32 k
 perl-Compress-Raw-Zlib             x86_64    1:2.061-4.el7
base                   57 k
 perl-DBI                           x86_64    1.627-4.el7
base                  802 k
 perl-Data-Dumper                   x86_64    2.145-3.el7
base                   47 k
 perl-IO-Compress                   noarch    2.061-2.el7
base                  260 k
 perl-Net-Daemon                    noarch    0.48-5.el7
```

```
base                      51 k
 perl-PlRPC                            noarch    0.2020-14.el7   base                      36 k
Updating for dependencies:
 openssl-libs                          x86_64    1:1.0.2k-16.el7                base                 1.2 M

Transaction Summary
==============================================================================================
Install  2 Packages (+15 Dependent packages)
Upgrade             (  1 Dependent package)
```

After this, you need to start the MySQL service.

```
[root@WP3 ~]# service mysql start
Starting mysql (via systemctl):                          [  OK  ]
```

By default, MariaDB is installed without root password, so you only need to run the mysql command to access the database with root privileges.

```
[root@WP3 ~]# mysql
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 10.3.11-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type ‘help;’ or ‘\h’ for help. Type ‘\c’ to clear the current input statement.

MariaDB [(none)]>
```

It’s recommended to use the mysql_secure_installation to improve your database security.

```
[root@WP3 ~]# mysql_secure_installation
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!
In order to log into MariaDB to secure it, we’ll need the current
password for the root user.  If you’ve just installed MariaDB, and
you haven’t set the root password yet, the password will be blank,
so you should just press enter here.
```

```

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!

By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n]
 ... Success!

Normally, root should only be allowed to connect from ‘localhost’.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n]
 ... Success!

By default, MariaDB comes with a database named ‘test’ that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n]
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes
```

```
made so far
will take effect immediately.

Reload privilege tables now? [Y/n]
 ... Success!

Cleaning up...

All done!  If you’ve completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Now your database is running, but it’s not ready yet. We need more work here.

Default Configuration

For MariaDB, the my.cnf config file includes the /etc/my.cnf.d directory:

```
[root@WP3 ~]# cat /etc/my.cnf
#
# This group is read both both by the client and the server
# use it for options that affect everything
#
[client-server]
#
# include all files from the config directory
#
!includedir /etc/my.cnf.d
```

And there, you have the following files by default:

```
[root@WP3 ~]# ls /etc/my.cnf.d/
enable_encryption.preset  mysql-clients.cnf  server.cnf
```

For encryption configuration, you have the enable_encryption file:

```
[root@WP3 ~]# cat /etc/my.cnf.d/enable_encryption.preset
#
# !include this file into your my.cnf (or any of *.cnf files in /etc/my.cnf.d)
# and it will enable data at rest encryption. This is a simple way to
# ensure that everything that can be encrypted will be and your
# data will not leak unencrypted.
#
```

```
# DO NOT EDIT THIS FILE! On MariaDB upgrades it might be replaced with a
# newer version and your edits will be lost. Instead, add your edits
# to the .cnf file after the !include directive.
#
# NOTE that you also need to install an encryption plugin for the encryption
# to work. See https://mariadb.com/kb/en/mariadb/data-at-rest-encryption/#encryption-key-management
#
[mariadb]
aria-encrypt-tables
encrypt-binlog
encrypt-tmp-disk-tables
encrypt-tmp-files
loose-innodb-encrypt-log
loose-innodb-encrypt-tables
```

For clients configuration, you can use the mysql-clients.cnf file where you can use a different configuration for each client:

```
[root@WP3 ~]# cat /etc/my.cnf.d/mysql-clients.cnf
#
# These groups are read by MariaDB command-line tools
# Use it for options that affect only one utility
#
[mysql]
[mysql_upgrade]
[mysqladmin]
[mysqlbinlog]
[mysqlcheck]
[mysqldump]
[mysqlimport]
[mysqlshow]
[mysqlslap]
```

And finally, you have the server.cnf configuration file:

```
[root@WP3 ~]# cat /etc/my.cnf.d/server.cnf
#
# These groups are read by MariaDB server.
# Use it for options that only the server (but not clients) should see
#
# See the examples of server my.cnf files in /usr/share/mysql/
#
# this is read by the standalone daemon and embedded servers
[server]
```

```
# this is only for the mysqld standalone daemon
[mysqld]
#
# * Galera-related settings
#
[galera]
# Mandatory settings
#wsrep_on=ON
#wsrep_provider=
#wsrep_cluster_address=
#binlog_format=row
#default_storage_engine=InnoDB
#innodb_autoinc_lock_mode=2
#
# Allow server to accept connections on all interfaces.
#
#bind-address=0.0.0.0
#
# Optional setting
#wsrep_slave_threads=1
#innodb_flush_log_at_trx_commit=0
# this is only for embedded server
[embedded]
# This group is only read by MariaDB servers, not by MySQL.
# If you use the same .cnf file for MySQL and MariaDB,
# you can put MariaDB-only options here
[mariadb]
# This group is only read by MariaDB-10.3 servers.
# If you use the same .cnf file for MariaDB of different versions,
# use this group for options that older servers don’t understand
[mariadb-10.3]
```

Let’s see these parameters in detail.

- aria-encrypt-tables: Enables automatic encryption of all user-created Aria tables that have the ROW_FORMAT table option set to PAGE.
- encrypt-binlog: Encrypt binary logs (including relay logs).
- encrypt-tmp-disk-tables: Enables automatic encryption of all internal on-disk temporary tables that are created during query execution if aria_used_for_temp_tables=ON is set.
- encrypt-tmp-files: Enables automatic encryption of temporary files, such as those created for filesort operations, binary log file caches, etc.
- loose-innodb-encrypt-log: Enables encryption of the InnoDB redo log. This also enables encryption of some temporary files created internally by InnoDB, such as those used for merge sorts and row logs.
- loose-innodb-encrypt-tables: Enables automatic encryption of all InnoDB tablespaces.

You can also add your own configuration file in /etc/my.cnf.d/ or just add a different path in the my.cnf file using the parameter !include.

Optional MariaDB Server Configuration

Let's see some variables that we can add in our config files. It's recommended as a basic configuration, but ought to be tuned based on your infrastructure.

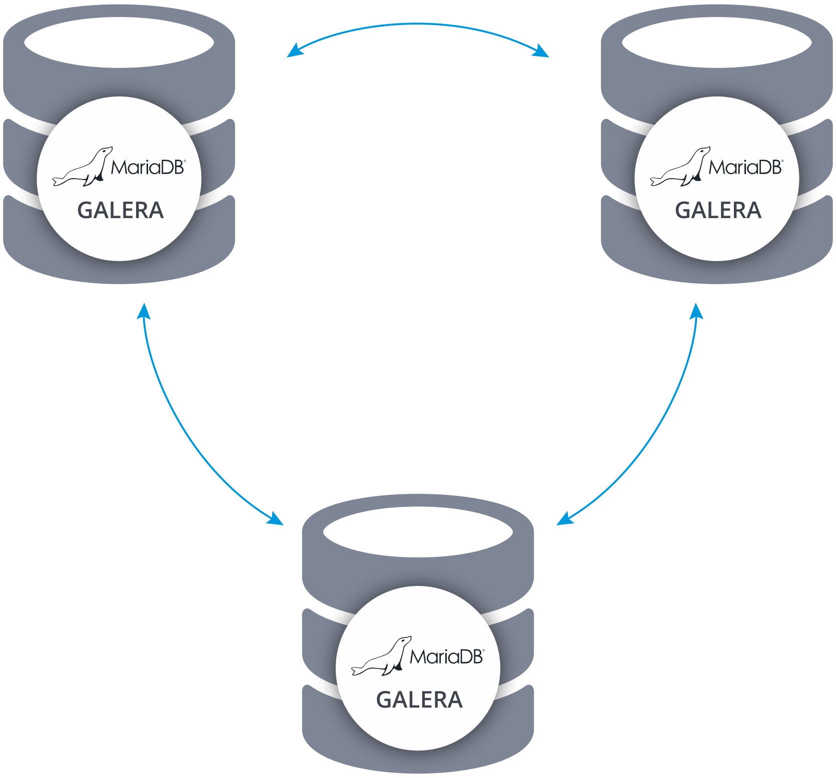
```
#GENERAL
user=mysql
basedir=/usr/
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
pid_file=/var/lib/mysql/mysql.pid
port=3306
ignore-db-dir=lost+found
#LOGGING
log_error=/var/log/mysql/mysqld.log
log_warnings=2
slow_query_log_file=/var/log/mysql/mysql-slow.log
long_query_time=2
slow_query_log=OFF
log_queries_not_using_indexes=OFF
#INNODB
innodb_buffer_pool_size=128M
innodb_flush_log_at_trx_commit=2
innodb_file_per_table=1
innodb_data_file_path = ibdata1:100M:autoextend
innodb_read_io_threads=4
innodb_write_io_threads=4
innodb_doublewrite=1
innodb_log_file_size=64M
innodb_log_buffer_size=16M
innodb_buffer_pool_instances=1
innodb_log_files_in_group=2
innodb_thread_concurrency=64
innodb_flush_method = O_DIRECT
innodb_rollback_on_timeout=ON
innodb_autoinc_lock_mode=2
innodb_stats_on_metadata=0
default_storage_engine=innodb
#REPLICATION
server_id=1
binlog_format=ROW
log_bin=binlog
log_slave_updates=1
relay_log=relay-bin
expire_logs_days=7
read_only=ON
report_host=192.168.100.143
```

```
# OTHER THINGS
key_buffer_size = 24M
tmp_table_size = 64M
max_heap_table_size = 64M
max_allowed_packet = 512M
skip_name_resolve
memlock=0
sysdate_is_now=1
max_connections=500
thread_cache_size=512
query_cache_type = 0
query_cache_size = 0
table_open_cache=1024
lower_case_table_names=0
```

To see these variables in detail, you can follow this link.

MariaDB Cluster Configuration

As we mentioned previously, MariaDB uses the same binaries for both MariaDB Server and MariaDB Galera Cluster (they used to be separate binaries). There are some variables that we must configure to have MariaDB Cluster enabled.



Node 1 IP Address: 192.168.100.131

Node 2 IP Address: 192.168.100.132

Node 3 IP Address: 192.168.100.133

- wsrep_provider=/usr/lib64/galera/libgalera_smm.so: Path to the Galera library.
- wsrep_cluster_address=gcomm://192.168.100.131,192.168.100.132,192.168.100.133: gcomm is the option to use for a working implementation.
- binlog_format=ROW: There are three formats for binary logging: statement-based, row-based and mixed.
- default_storage_engine=InnoDB
- innodb_autoinc_lock_mode=2: Locking mode used for generating auto-increment values. 0 is the traditional lock mode, 1 - the consecutive, and 2 - the interleaved.
- innodb_doublewrite=1: This is the default value. To improve fault tolerance InnoDB first stores data to a doublewrite buffer before writing it to data file.
- query_cache_size=0: Only mandatory for MariaDB versions prior to MariaDB Galera Cluster 5.5.40, MariaDB Galera Cluster 10.0.14, and MariaDB 10.1.2.
- wsrep_on=ON: Enable wsrep replication (starting 10.1.1)

Also, there are some optional variables to add to configure your MariaDB Cluster.

```
wsrep_node_address=192.168.100.131
wsrep_provider_options=”base_port=4567; gcache.size=1024M; gmcast.segment=0 “
wsrep_cluster_name=”MariaDB1”
wsrep_cluster_address=gcomm://192.168.100.131,192.168.100.132,192.168.100.133
wsrep_node_name=192.168.100.131
wsrep_slave_threads=4
wsrep_certify_nonPK=1
wsrep_max_ws_rows=131072
wsrep_max_ws_size=1073741824
wsrep_debug=0
wsrep_convert_LOCK_to_trx=0
wsrep_retry_autocommit=1
wsrep_auto_increment_control=1
wsrep_replicate_myisam=1
wsrep_drupal_282555_workaround=0
wsrep_causal_reads=0
wsrep_sst_method=mariabackup
wsrep_log_conflicts=1
wsrep_gtid_domain_id=9999
wsrep_gtid_mode=1
```

Keep in mind that some of these values depend on your infrastructure.

Percona XtraDB Cluster

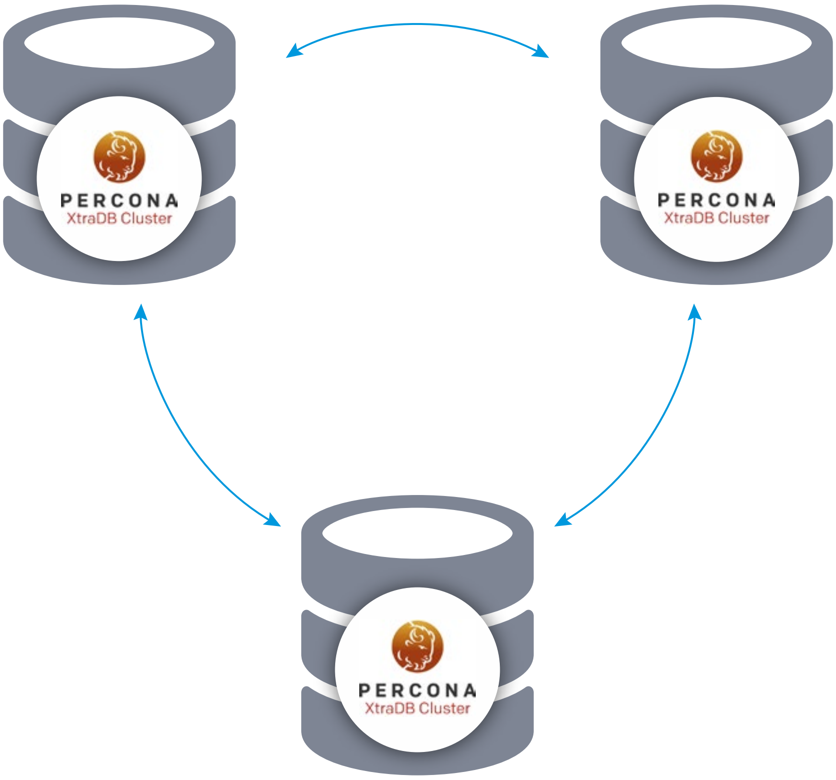
Percona XtraDB Cluster is an open source, cost-effective, and robust MySQL clustering solution for businesses. Organizations use Percona XtraDB Cluster to power highly available applications in the most demanding public, private and hybrid cloud environments. It is an excellent solution for your MySQL cluster and database needs.

Installation

In this link you have the latest packages to install Percona XtraDB Cluster.

If you prefer, you can use the yum repository or apt repository installation.

In our example, let's see the yum repository installation of Percona XtraDB Cluster on CentOS 7.



Node 1 IP Address: 192.168.100.154

Node 2 IP Address: 192.168.100.155

Node 3 IP Address: 192.168.100.156

```
[root@WP4 ~]# yum install http://www.percona.com/downloads/percona-release/redhat/0.1-6/percona-release-0.1-6.noarch.rpm
===================================================================================================================
 Package                                                            Arch       Version                  Repository                                 Size
============================================================
======================================================
Installing:
 percona-release                             noarch    0.1-6
/percona-release-0.1-6.noarch    16 k

Transaction Summary
============================================================
======================================================
Install  1 Package

[root@WP4 ~]# yum install Percona-XtraDB-Cluster-57
============================================================
======================================================
 Package                                     Arch      Ver-
sion                Repository                     Size
============================================================
======================================================
Installing:
 Percona-XtraDB-Cluster-57                   x86_64
5.7.23-31.31.2.el7    percona-release-x86_64           27 k
Installing for dependencies:
 Percona-XtraDB-Cluster-client-57            x86_64
5.7.23-31.31.2.el7    percona-release-x86_64           7.2 M
 Percona-XtraDB-Cluster-server-57            x86_64
5.7.23-31.31.2.el7    percona-release-x86_64            51 M
 Percona-XtraDB-Cluster-shared-57            x86_64
5.7.23-31.31.2.el7    percona-release-x86_64           737 k
 Percona-XtraDB-Cluster-shared-compat-57     x86_64
5.7.23-31.31.2.el7    percona-release-x86_64           1.1 M
 libaio                                      x86_64
0.3.109-13.el7        base                              24 k
 libev                                       x86_64    4.15-
7.el7            extras                        44 k
 lsof                                        x86_64    4.87-
6.el7            base                         331 k
 numactl-libs                                x86_64    2.0.9-
7.el7           base                          29 k
 percona-xtrabackup-24                       x86_64
2.4.12-1.el7          percona-release-x86_64           7.5 M
 perl-Compress-Raw-Bzip2                     x86_64    2.061-
3.el7           base                          32 k
 perl-Compress-Raw-Zlib                      x86_64
1:2.061-4.el7         base                              57 k
 perl-DBD-MySQL                              x86_64    4.023-
6.el7           base                         140 k
 perl-DBI                                    x86_64    1.627-
4.el7           base                         802 k
 perl-Data-Dumper                            x86_64    2.145-
3.el7           base                          47 k
```

```
 perl-Digest                                            noarch    1.17-
245.el7          base                                          23 k
 perl-Digest-MD5                                        x86_64    2.52-
3.el7            base                                          30 k
 perl-IO-Compress                                       noarch    2.061-
2.el7           base                                         260 k
 perl-Net-Daemon                                        noarch    0.48-
5.el7            base                                          51 k
 perl-PlRPC                                             noarch
0.2020-14.el7         base                                          36 k
 qpress                                                 x86_64    11-1.
el7              percona-release-x86_64                        31 k
 socat                                                  x86_64
1.7.3.2-2.el7         base                                         290 k

Transaction Summary
============================================================
======================================================
Install  1 Package (+21 Dependent packages)
```

You need to repeat these installation steps for each node. It's recommended to have at least three nodes in a cluster to improve the HA environment.

Before we start to use our cluster, we need to configure it. For this, please edit the /etc/percona-xtradb-cluster.conf.d/wsrep.cnf configuration file:

```
wsrep_cluster_address=g-
comm://192.168.100.154,192.168.100.155,192.168.100.156
#Replace the IP Address for the IP of each node.
wsrep_node_address=192.168.100.154
#Replace the IP Address for the IP of the current node
wsrep_cluster_name=cluster1
#Replace for your Cluster Name
wsrep_node_name=node1
#Replace for your Node Name
wsrep_sst_auth=”sstuser:Password!”
#Replace for the SST credential that you want to use in your
new Cluster
wsrep_on=ON
```

Now, you need to start the MySQL service. If it's the first node, you need to bootstrap the cluster:

```
[root@WP4 ~]# systemctl start mysql@bootstrap.service
```

A new password for the root user is created. To know it, you need to check the MySQL log and filter by temporary password. You should have something similar to this:

```
[root@WP4 ~]# grep “temporary password” /var/log/mysqld.log
```

```
2019-01-03T22:58:39.518754Z 1 [Note] A temporary password is generated for root@localhost: eFrhjwjh+1tH
```

Connect to the database using the temporary password and change the current root password (it’s required by the server before creating a new user)

```
[root@WP4 ~]# mysql -p
mysql> SET PASSWORD=’*********’;
Query OK, 0 rows affected (0.00 sec)
```

Create a SST user for localhost (the user that we have in the wsrep_sst_auth variable):

```
mysql> GRANT RELOAD, LOCK TABLES, PROCESS, REPLICATION CLIENT ON *.* TO ‘sstuser’@’192.168.100.%’ IDENTIFIED BY ‘Password!’;
Query OK, 0 rows affected, 1 warning (0.01 sec)
```

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

For the rest of the nodes, you only need to start the MySQL service as usual:

```
[root@WP4 ~]# systemctl start mysql
```

This process will perform a SST and start the service in the node.

After starting all the nodes, your database cluster is running, but it’s not ready yet. Let’s see the default and optional configuration.

Default Configuration

As we could see previously on Percona Server for MySQL, by default, the Percona’s my.cnf config file includes the /etc/my.cnf.d/ and /etc/percona-server.conf.d/ directories:

```
[root@WP4 ~]# cat /etc/my.cnf
#
# The Percona XtraDB Cluster 5.7 configuration file.
#
#
# * IMPORTANT: Additional settings that can override those from this file!
#   The files must end with ‘.cnf’, otherwise they’ll be ignored.
#   Please make any edits and changes to the appropriate sectional files
#   included below.
#
!includedir /etc/my.cnf.d/
!includedir /etc/percona-xtradb-cluster.conf.d/
```

The /etc/my.cnf.d/ directory is empty by default, and in /etc/percona-server.conf.d/ we have the following content:

```
[root@WP4 ~]# ls /etc/percona-xtradb-cluster.conf.d/
mysqld.cnf  mysqld_safe.cnf  wsrep.cnf
```

mysqld_safe is the recommended way to start a mysqld server on Unix. It adds some safety features such as restarting the server when an error occurs and logging runtime information to an error log.

mysqld_safe reads options from both [mysqld] and [mysqld_safe] sections in the configuration files.

The content of these configuration files are:

- mysqld.cnf

```
[root@WP4 ~]# cat /etc/percona-xtradb-cluster.conf.d/mysqld.cnf
# Template my.cnf for PXC
# Edit to your requirements.
[client]
socket=/var/lib/mysql/mysql.sock
[mysqld]
server-id=1
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
log-bin
log_slave_updates
expire_logs_days=7
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
```

- mysqld_safe.cnf

```
[root@WP4 ~]# cat /etc/percona-xtradb-cluster.conf.d/mysqld_safe.cnf
#
# The Percona Server 5.7 configuration file.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.
#
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html
[mysqld_safe]
pid-file = /var/run/mysqld/mysqld.pid
socket   = /var/lib/mysql/mysql.sock
nice     = 0
```

- wsrep.cnf

```
[root@WP4 ~]# cat /etc/percona-xtradb-cluster.conf.d/wsrep.cnf
[mysqld]
# Path to Galera library
wsrep_provider=/usr/lib64/galera3/libgalera_smm.so
# Cluster connection URL contains IPs of nodes
#If no IP is found, this implies that a new cluster needs to be created,
#in order to do that you need to bootstrap this node
wsrep_cluster_address=gcomm://
# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW
# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB
# Slave thread to use
wsrep_slave_threads= 8
wsrep_log_conflicts
# This changes how InnoDB autoincrement locks are managed and is a requirement for Galera
innodb_autoinc_lock_mode=2
# Node IP address
#wsrep_node_address=192.168.70.63
# Cluster name
wsrep_cluster_name=pxc-cluster
#If wsrep_node_name is not specified,  then system hostname will be used
wsrep_node_name=pxc-cluster-node-1
#pxc_strict_mode allowed values: DISABLED,PERMISSIVE,ENFORCING,MASTER
pxc_strict_mode=ENFORCING
# SST method
wsrep_sst_method=xtrabackup-v2
#Authentication for SST method
#wsrep_sst_auth="sstuser:s3cretPass"
```

Let's see these parameters in detail.

- server-id: Specifies the server ID. The server_id system variable is set to 0 by default.
- datadir: The path to the MySQL server data directory.

- socket: On Unix platforms, this variable is the name of the socket file that is used for local client connections.
- log-error: Write the error log and startup messages to this file.
- pid-file: The path name of the file in which the server should write its process ID. The server creates the file in the data directory unless an absolute path name is given to specify a different directory.
- log-bin: Enables binary logging. With binary logging enabled, the server logs all statements that change data to the binary log, which is used for backup and replication. The binary log is a sequence of files with a base name and a numeric extension.
- log_slave_updates: Whether updates received by a slave server from a master server should be logged to the slave's own binary log.
- expire_logs_days: The number of days for automatic binary log file removal. The default is 0, which means "no automatic removal."
- symbolic-links: Enable or disable symbolic link support. On Unix, enabling symbolic links means that you can link a MyISAM index file or data file to another directory with the INDEX DIRECTORY or DATA DIRECTORY option of the CREATE TABLE statement.
- nice: Use the nice program to set the server's scheduling priority to the given value.
- wsrep_provider: Specifies the path to the Galera library. This is usually /usr/lib64/libgalera_smm.so on CentOS/RHEL and /usr/lib/libgalera_smm.so on Debian/Ubuntu.
- wsrep_cluster_address: Defines the back-end schema, IP addresses, ports, and options that the node uses when connecting to the cluster. This variable needs to specify at least one other node's address, which is alive and a member of the cluster.
- binlog_format: This variable sets the binary logging format, and can be any one of STATEMENT, ROW, or MIXED.
- default_storage_engine: Galera fully supports only the InnoDB storage engine. It will not work correctly with MyISAM or any other non-transactional storage engines.
- wsrep_slave_threads: Specifies the number of threads that can apply replication transactions in parallel. Galera supports true parallel replication that applies transactions in parallel only when it is safe to do so.
- wsrep_log_conflicts: Defines whether the node should log additional information about conflicts.
- innodb_autoinc_lock_mode: Galera supports only interleaved (2) lock mode for InnoDB.
- wsrep_cluster_name: Specify the logical name for your cluster. It must be the same for all nodes in your cluster.
- wsrep_node_name: Specify the logical name for each individual node. If this variable is not specified, the host name will be used.
- pxc_strict_mode: PXC Strict Mode is enabled by default and set to ENFORCING, which blocks the use of experimental and unsupported features in Percona XtraDB Cluster.
- wsrep_sst_method: By default, Percona XtraDB Cluster uses Percona XtraBackup for State Snapshot Transfer (SST). Setting wsrep_sst_method=xtrabackup-v2 is

highly recommended. This method requires a user for SST to be set up on the initial node. Provide SST user credentials with the wsrep_sst_auth variable.

Optional Percona Server Configuration

Let's see some variables that we can add in our config files. It's recommended as a basic configuration, but it depends on your infrastructure.

```
#GENERAL
user=mysql
basedir=/usr/
port=3306
skip_name_resolve
ignore-db-dir=lost+found
#LOGGING
log_warnings=2
slow_query_log_file=/var/log/mysql/mysql-slow.log
long_query_time=2
slow_query_log=OFF
log_queries_not_using_indexes=OFF
log_slow_admin_statements=ON
log_throttle_queries_not_using_indexes=1
#INNODB
innodb_buffer_pool_size=128M
innodb_flush_log_at_trx_commit=2
innodb_file_per_table=1
innodb_data_file_path = ibdata1:100M:autoextend
innodb_read_io_threads=4
innodb_write_io_threads=4
innodb_doublewrite=1
innodb_log_file_size=64M
innodb_log_buffer_size=16M
innodb_buffer_pool_instances=1
innodb_log_files_in_group=2
innodb_thread_concurrency=64
innodb_flush_method = O_DIRECT
innodb_autoinc_lock_mode=2
innodb_stats_on_metadata=0
default_storage_engine=innodb
#REPLICATION
server_id=1
binlog_format=ROW
#OTHER THINGS
tmp_table_size = 64M
max_heap_table_size = 64M
max_allowed_packet = 512M
memlock=0
sysdate_is_now=1
max_connections=500
thread_cache_size=512
```

```
query_cache_type = 0
query_cache_size = 0
table_open_cache=1024
lower_case_table_names=0
# WSREP
wsrep_provider_options=”base_port=4567; gcache.size=1024M; gmcast.segment=0 “
wsrep_certify_nonPK=1
wsrep_max_ws_rows=131072
wsrep_max_ws_size=1073741824
wsrep_debug=0
wsrep_convert_LOCK_to_trx=0
wsrep_retry_autocommit=1
wsrep_auto_increment_control=1
wsrep_replicate_myisam=0
wsrep_drupal_282555_workaround=0
wsrep_causal_reads=0
```

To see these variables in detail, you can follow this link or this one.

You can use the !include parameter, to split the configuration in different files, for example, the backup credentials.

Into /etc/percona-server.conf.d/mysqld.cnf add the following line:

```
!include /etc/percona-server.conf.d/secrets-backup.cnf
```

And then create the secrets-backup.cnf file:

```
[root@WP4 ~]# cat /etc/percona-server.conf.d/secrets-backup.cnf
# Security credentials for backup.
[mysqldump]
user=backupuser
password=DseOs0k0ZvXoHItv
[xtrabackup]
user=backupuser
password=DseOs0k0ZvXoHItv
```

Keep in mind that some of these values depends on your infrastructure.

NDB Cluster

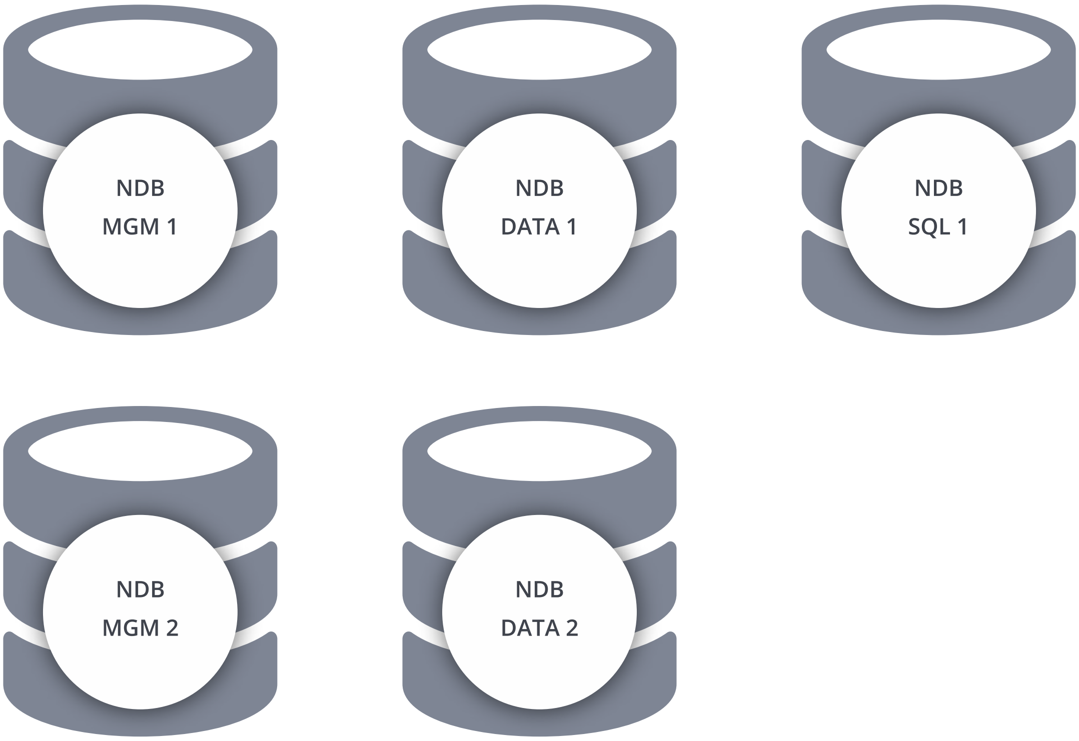
MySQL NDB Cluster is a high-availability storage engine for MySQL adapted for distributed computing environments. It consists of several elements: there are management servers, data nodes and SQL nodes.

Installation

To install the MySQL NDB Cluster packages manually, you can follow this link.

Another way to install it is using yum or apt repositories.

In our example, let's see the yum repository installation of MySQL NDB Cluster 7.5 on CentOS 7.



We need to install 5 nodes:

- 2 Management Nodes: Management nodes are intended to control the cluster. IP Address: 192.168.100.175, 192.168.100.176.
- 2 Data Nodes: Data nodes stores the data using NDB engine. IP Address: 192.168.100.177, 192.168.100.178.
- 1 SQL Node: SQL nodes are used as the entry points to the cluster. They parse SQL, ask for data from the data nodes and aggregate result sets when needed. IP Address: 192.168.100.179.

For all nodes:

```
[root@WP5 ~]# wget https://dev.mysql.com/get/mysql80-community-release-el7-2.noarch.rpm
--2019-02-15 19:43:41--  https://dev.mysql.com/get/mysql80-community-release-el7-2.noarch.rpm
Resolving dev.mysql.com (dev.mysql.com)... 137.254.60.11
Connecting to dev.mysql.com (dev.mysql.com)|137.254.60.11|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://repo.mysql.com//mysql80-community-release-el7-2.noarch.rpm [following]
```

```
--2019-02-15 19:43:42--  https://repo.mysql.com//mysql80-community-release-el7-2.noarch.rpm
Resolving repo.mysql.com (repo.mysql.com)... 23.208.182.226
Connecting to repo.mysql.com (repo.mysql.com)|23.208.182.226|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25892 (25K) [application/x-redhat-package-manager]
Saving to: ‘mysql80-community-release-el7-2.noarch.rpm’

100%[=======================================================================================================================================================================================================================>] 25,892      --.-K/s   in 0.01s

2019-02-15 19:43:42 (2.56 MB/s) - ‘mysql80-community-release-el7-2.noarch.rpm’ saved [25892/25892]

[root@WP5 ~]# rpm -Uvh mysql80-community-release-el7-2.noarch.rpm
warning: mysql80-community-release-el7-2.noarch.rpm: Header V3 DSA/SHA1 Signature, key ID 5072e1f5: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:mysql80-community-release-el7-2  ################################# [100%]
```

Edit the /etc/yum.repos.d/mysql-community.repo file and set the enable parameter in 1 for MySQL Cluster 7.5 and 0 for MySQL 8.0:

```
# Enable to use MySQL Cluster 7.5
[mysql-cluster-7.5-community]
name=MySQL Cluster 7.5 Community
baseurl=http://repo.mysql.com/yum/mysql-cluster-7.5-community/el/7/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
[mysql80-community]
name=MySQL 8.0 Community Server
baseurl=http://repo.mysql.com/yum/mysql-8.0-community/el/7/$basearch/
enabled=0
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
```

And then, install:

- For SQL Nodes (1):

```
[root@WP9 ~]# yum install mysql-cluster-community-server
=======================================================================================================
 Package                            Arch      Version             Repository                     Size
=======================================================================================================
Installing:
 mysql-cluster-community-server     x86_64    7.5.13-1.el7        mysql-cluster-7.5-community   175 M
Installing for dependencies:
 libaio                             x86_64    0.3.109-13.el7      base                           24 k
 mysql-cluster-community-client     x86_64    7.5.13-1.el7        mysql-cluster-7.5-community    85 M
 mysql-cluster-community-common     x86_64    7.5.13-1.el7        mysql-cluster-7.5-community   274 k
 mysql-cluster-community-libs       x86_64    7.5.13-1.el7        mysql-cluster-7.5-community   2.2 M
 numactl-libs                       x86_64    2.0.9-7.el7         base                           29 k
 perl-Class-MethodMaker             x86_64    2.20-1.el7          epel                          334 k
 perl-Compress-Raw-Bzip2            x86_64    2.061-3.el7         base                           32 k
 perl-Compress-Raw-Zlib             x86_64    1:2.061-4.el7       base                           57 k
 perl-DBI                           x86_64    1.627-4.el7         base                          802 k
 perl-Data-Dumper                   x86_64    2.145-3.el7         base                           47 k
 perl-IO-Compress                   noarch    2.061-2.el7         base                          260 k
 perl-Net-Daemon                    noarch    0.48-5.el7          base                           51 k
 perl-PlRPC                         noarch    0.2020-14.el7       base                           36 k

Transaction Summary
=======================================================================================================
Install  1 Package (+13 Dependent packages)
```

- For Management Nodes (2):

```
[root@WP5 ~]# yum install mysql-cluster-community-man-
```

```
agement-server
==============================================================================================================
 Package                                    Arch   Version          Repository                          Size
==============================================================================================================
Installing:
 mysql-cluster-community-management-server    x86_64   7.5.13-1.el7    mysql-cluster-7.5-community    4.8 M

Transaction Summary
==============================================================================================================
Install  1 Package
```

- For Data Nodes (2):

```
[root@WP7 ~]# yum install mysql-cluster-community-data-node
=========================================================================================================
 Package                             Arch      Version Repository                         Size
=========================================================================================================
Installing:
 mysql-cluster-community-data-node  x86_64    7.5.13-1.el7       mysql-cluster-7.5-community      20 M

Transaction Summary
=========================================================================================================
Install  1 Package
```

After this, you need to initialize the MySQL installation on the SQL Node.

```
[root@WP9 ~]# service mysqld start
Redirecting to /bin/systemctl start mysqld.service
```

A new password for the root user is created. To know it, you need to check the MySQL log and filter by temporary password. You should have something similar to this:

```
[root@WP9 ~]# grep “temporary password” /var/log/mysqld.log
2019-02-15T20:06:43.188506Z 1 [Note] A temporary password is generated for root@localhost: ESbw298Ql?zO
```

Then, you can run the mysql_secure_installation script, to configure a basic secure setup for your MySQL database.

```
[root@WP9 ~]# mysql_secure_installation
Securing the MySQL server deployment.

Enter password for user root:

The existing password for the user account root has expired. Please set a new password.

New password:

Re-enter new password:
The ‘validate_password’ plugin is installed on the server.
The subsequent steps will run with the existing configuration
of the plugin.
Using existing password for root.

Estimated strength of the password: 100
Change the password for root ? ((Press y|Y for Yes, any other key for No) : y

New password:

Re-enter new password:

Estimated strength of the password: 100
Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : y
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.

Normally, root should only be allowed to connect from
‘localhost’. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named ‘test’ that
```

```
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.

Remove test database and access to it? (Press y|Y for Yes, any other key for No) : y
 - Dropping test database...
Success.

 - Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!
```

Now, let's see the configuration.

Default Configuration

- SQL Node

  The MySQL installation creates the my.cnf config file and the /etc/my.cnf.d/ directory in /etc/.

  The /etc/my.cnf.d/ directory is empty by default, and in the content of my.cnf is:

```
[root@WP9 ~]# cat /etc/my.cnf
# For advice on how to change settings please see
# http://dev.mysql.com/doc/refman/5.7/en/server-configuration-defaults.html
[mysqld]
#
# Remove leading # and set to the amount of RAM for the most important data
# cache in MySQL. Start at 70% of total RAM for dedicated server, else 10%.
# innodb_buffer_pool_size = 128M
#
# Remove leading # to turn on a very important data integrity option: logging
# changes to the binary log between backups.
# log_bin
#
```

```
# Remove leading # to set options mainly useful for reporting servers.
# The server defaults are faster for transactions and fast SELECTs.
# Adjust sizes as needed, experiment to find the optimal values.
# join_buffer_size = 128M
# sort_buffer_size = 2M
# read_rnd_buffer_size = 2M
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

In this file, we need to add the following lines.

In the [mysqld] section:

```
ndbcluster
```

And, a new section [mysql_cluster]:

```
[mysql_cluster]
ndb-connectstring=192.168.100.175,192.168.100.176 # IP address of Management Node
```

Let's see these parameters in detail.

- datadir: The path to the MySQL server data directory.
- socket: On Unix platforms, this variable is the name of the socket file that is used for local client connections.
- symbolic-links: Enable or disable symbolic link support. On Unix, enabling symbolic links means that you can link a MyISAM index file or data file to another directory with the INDEX DIRECTORY or DATA DIRECTORY option of the CREATE TABLE statement.
- log-error: Write the error log and startup messages to this file.
- pid-file: The path name of the file in which the server should write its process ID. The server creates the file in the data directory unless an absolute path name is given to specify a different directory.
- ndbcluster: The ndbcluster storage engine is necessary for using NDB Cluster.
- ndb-connectstring: When using the ndbcluster storage engine, this option specifies the management server that distributes cluster configuration data.

- Data Nodes

  Create a new configuration file /etc/my.cnf in each data node:

```
[root@WP7 ~]# vi /etc/my.cnf
[mysqld]
ndbcluster
[mysql_cluster]
ndb-connectstring=192.168.100.175,192.168.100.176
#IP address of Management Nodes
```

  And create the datadir:

```
[root@WP7 ~]# mkdir -p /usr/local/mysql/data
```

  Let's see these parameters in detail.

  - ndbcluster: The ndbcluster storage engine is necessary for using NDB Cluster.
  - ndb-connectstring: When using the ndbcluster storage engine, this option specifies the management server that distributes cluster configuration data.

- Management Nodes

```
[root@WP5 ~]# mkdir /var/lib/mysql-cluster
[root@WP5 ~]# cd /var/lib/mysql-cluster
[root@WP5 ~]# vi config.ini
[ndbd default]
# Default Options
NoOfReplicas=2
DataMemory=80M
[ndb_mgmd]
# Management Node 1
HostName=192.168.100.175
NodeId=1
DataDir=/var/lib/mysql-cluster
[ndb_mgmd]
# Management Node 2
HostName=192.168.100.176
NodeId=2
DataDir=/var/lib/mysql-cluster
[ndbd]
# Data Node 1
HostName=192.168.100.177
NodeId=3
DataDir=/usr/local/mysql/data
[ndbd]
# Data Node 2
HostName=192.168.100.178
```

```
NodeId=4
DataDir=/usr/local/mysql/data
[mysqld]
# SQL node
NodeId=5
HostName=192.168.100.179
```

Let's see these parameters in detail.

- NoOfReplicas: This global parameter can be set only in the [ndbd default] section, and defines the number of replicas for each table stored in the cluster.
- DataMemory: This parameter defines the amount of space available for storing database records. The amount specified by this value is allocated in memory, so it is important that the machine has sufficient physical memory to accommodate it.
- HostName: This parameter defines the hostname of the computer.
- NodeId: A unique node ID is used as the node's address for all cluster internal messages.
- DataDir: This parameter specifies the directory where the data files will be stored.

After the installation and configuration are completed, you should follow the initialization order to start the cluster. First, you should start the management nodes, then the data nodes and after that the SQL Nodes.

- Starting the Management Nodes

```
[root@WP5 ~]# ndb_mgmd -f /var/lib/mysql-cluster/config.ini
MySQL Cluster Management Server mysql-5.7.25 ndb-7.5.13
2019-02-19 00:25:17 [MgmtSrvr] INFO     -- The default config directory '/usr/mysql-cluster' does not exist. Trying to create it...
2019-02-19 00:25:17 [MgmtSrvr] INFO     -- Sucessfully created config directory
```

- Starting the Data Nodes

```
[root@WP7 ~]# ndbd
2019-02-19 00:29:29 [ndbd] INFO     -- Angel connected to '192.168.100.175:1186'
2019-02-19 00:29:29 [ndbd] INFO     -- Angel allocated nodeid: 3
```

- Starting the SQL Node

If we add the configuration for NDB Cluster after the initialization, we need to restart the MySQL service to apply the changes.

```
[root@WP9 ~]# service mysqld restart
Redirecting to /bin/systemctl restart mysqld.service
```

Then, we can check the cluster status using the ndb_mgm command from the SQL Node:

```
[root@WP9 ~]# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> SHOW
Connected to Management Server at: 192.168.100.175:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=3    @192.168.100.177  (mysql-5.7.25 ndb-7.5.13, Nodegroup: 0, *)
id=4    @192.168.100.178  (mysql-5.7.25 ndb-7.5.13, Nodegroup: 0)
[ndb_mgmd(MGM)]     2 node(s)
id=1    @192.168.100.175  (mysql-5.7.25 ndb-7.5.13)
id=2    @192.168.100.176  (mysql-5.7.25 ndb-7.5.13)
[mysqld(API)]     1 node(s)
id=5    @192.168.100.179  (mysql-5.7.25 ndb-7.5.13)
```

Optional NDB Cluster Server Configuration

Let's see some variables that we can add in our config files. It's recommended as a basic configuration but it depends on your infrastructure.

```
[TCP DEFAULT]
SendBufferMemory=1M
ReceiveBufferMemory=1M
[NDB_MGMD DEFAULT]
PortNumber=1186
[NDB_MGMD]
LogDestination=FILE:filename=ndb_1_cluster.log,max-size=10000000,maxfiles=6
ArbitrationRank=1
[NDBD DEFAULT]
ServerPort=2200
FileSystemPathDD=/var/lib/mysql-cluster
BackupDataDir=/var/lib/mysql-cluster/backup
FileSystemPathUndoFiles=/var/lib/mysql-cluster
FileSystemPathDataFiles=/var/lib/mysql-cluster
DataMemory=128M
IndexMemory=21M
LockPagesInMainMemory=1
MaxNoOfConcurrentOperations=32768
```

```
MaxNoOfConcurrentTransactions=8192
StringMemory=25
MaxNoOfTables=2048
MaxNoOfOrderedIndexes=1024
MaxNoOfUniqueHashIndexes=256
MaxNoOfAttributes=12288
MaxNoOfTriggers=7168
MaxNoOfExecutionThreads=2
NoOfFragmentLogParts=4
FragmentLogFileSize=512M
InitFragmentLogFiles=SPARSE
NoOfFragmentLogFiles=3
RedoBuffer=8M
TransactionBufferMemory=8M
TimeBetweenGlobalCheckpoints=1000
TimeBetweenEpochs=100
TimeBetweenEpochsTimeout=32000
MinDiskWriteSpeed=20M
MaxDiskWriteSpeed=80M
MaxDiskWriteSpeedOtherNodeRestart=50M
MaxDiskWriteSpeedOwnRestart=200M
TimeBetweenLocalCheckpoints=20
HeartbeatIntervalDbDb=1500
HeartbeatIntervalDbApi=1500
MemReportFrequency=30
BackupReportFrequency=10
LogLevelStartup=15
LogLevelShutdown=15
LogLevelCheckpoint=8
LogLevelNodeRestart=15
BackupMaxWriteSize=1M
BackupDataBufferSize=24M
BackupLogBufferSize=16M
TimeBetweenWatchdogCheckInitial=60000
TransactionInactiveTimeout=60000
RedoOverCommitCounter=3
RedoOverCommitLimit=20
SharedGlobalMemory=20M
DiskPageBufferMemory=8M
BatchSizePerLocalScan=512
[MYSQLD DEFAULT]
DefaultOperationRedoProblemAction=ABORT
BatchSize=512
```

To see these variables in detail, you can follow this link.

MongoDB

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling. Classified as a NoSQL database, MongoDB uses JSON-like documents with schema. MongoDB is developed by MongoDB Inc.

Installation

There are many ways to install MongoDB, you can follow this link to choose one.

You can install MongoDB by using yum or apt repositories.

In our example, let’s see the yum repository installation of MongoDB on CentOS 7.

MongoDB Server IP Address: 192.168.100.181

To add the MongoDB repository, you can run:

```
cat > /etc/yum.repos.d/mongodb-org-4.0.repo <<- EOF
[mongodb-org-4.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/7/mongodb-org/4.0/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.0.asc
EOF
```

And then, install the mongodb-org packages:

```
[root@WP10 ~]# yum install -y mongodb-org
============================================================
===========================================
 Package                            Arch      Version
Repository                         Size
============================================================
===========================================
Installing:
 mongodb-org                        x86_64    4.0.6-1.el7
mongodb-org-4.0                    5.8 k
Installing for dependencies:
 make                               x86_64    1:3.82-23.el7
base                               420 k
 mongodb-org-mongos                 x86_64    4.0.6-1.el7
mongodb-org-4.0                     12 M
 mongodb-org-server                 x86_64    4.0.6-1.el7
mongodb-org-4.0                     21 M
 mongodb-org-shell                  x86_64    4.0.6-1.el7
```

```
mongodb-org-4.0                   13 M
 mongodb-org-tools                  x86_64    4.0.6-1.el7
mongodb-org-4.0                   32 M
 openssl                            x86_64    1:1.0.2k-16.
el7    base                               493 k
Updating for dependencies:
 openssl-libs                       x86_64    1:1.0.2k-16.
el7    base                               1.2 M

Transaction Summary
============================================================
===========================================
Install  1 Package  (+6 Dependent packages)
Upgrade             ( 1 Dependent package)
```

Now, you need to start the MongoDB service.

```
[root@WP10 ~]# service mongod start
Redirecting to /bin/systemctl start mongod.service
```

You can verify the status by filtering the “waiting” word in the log file:

```
[root@WP10 ~]# grep “waiting” /var/log/mongodb/mongod.log
2019-02-20T00:40:37.449+0000 I NETWORK  [initandlisten]
waiting for connections on port 27017
```

Default Configuration

The MongoDB installation creates the /etc/mongod.conf config file.

```
[root@WP10 ~]# cat /etc/mongod.conf
# mongod.conf
# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configura-
tion-options/
# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log
# Where and how to store data.
storage:
  dbPath: /var/lib/mongo
  journal:
    enabled: true
#  engine:
#  mmapv1:
```

```
#  wiredTiger:
# how the process runs
processManagement:
  fork: true  # fork and run in background
  pidFilePath: /var/run/mongodb/mongod.pid  # location of pidfile
  timeZoneInfo: /usr/share/zoneinfo
# network interfaces
net:
  port: 27017
  bindIp: 127.0.0.1  # Enter 0.0.0.0,:: to bind to all IPv4 and IPv6 addresses or, alternatively, use the net.bindIpAll setting.
#security:
#operationProfiling:
#replication:
#sharding:
## Enterprise-Only Options
#auditLog:
#snmp:
```

Let's see these parameters in detail.

- systemLog:
  - destination: The destination to which MongoDB sends all log output.
  - logAppend: When true, MongoDB appends new entries to the end of the existing log file when the instance restarts.
  - path: Specify a log file path.
- storage:
  - dbPath: Specify a data directory path.
  - journal: Enable or disable the durability journal to ensure data files remain valid and recoverable.
- processManagement:
  - fork: Enable a daemon mode that runs the MongoDB process in the background.
  - pidFilePath: Specifies a file location to hold the process ID of the MongoDB process.
  - timeZoneInfo: The full path from which to load the time zone database.
- net:
  - port: The TCP port on which the MongoDB instance listens for client connections.
  - bindIp: The hostnames and/or IP addresses and/or full Unix domain socket paths on which MongoDB should listen for client connections.

Optional MongoDB Configuration

Let’s see some variables that we can add in our config files. It’s recommended as a basic configuration but it depends on your infrastructure.

```
storage:
    engine: wiredTiger
    mmapv1:
        smallFiles: false
net:
    bindIp: 0.0.0.0
setParameter:
   enableLocalhostAuthBypass: true
replication:
   replSetName: my_mongodb_0
sharding:
    clusterRole: shardsvr
security.keyFile: /etc/mongo-cluster.key
```

To see in detail these variables, you can follow this link.

Percona Server for MongoDB

Percona Server for MongoDB is a free and open-source drop-in replacement for MongoDB Community Edition. It offers all the features and benefits of MongoDB Community Edition, plus additional enterprise-grade functionality.

Installation

To install the Percona Server for MongoDB packages manually, you can follow this link.

Another way to install it is using yum or apt repositories.

In our example, let’s see the yum repository installation of Percona Server for MongoDB on CentOS 7.

Percona MongoDB Server IP Address: 192.168.100.182

Install the repository:

```
[root@WP11 ~]# yum install https://repo.percona.com/yum/percona-release-latest.noarch.rpm
============================================================
===========================================
 Package                                   Arch      Version
Repository                             Size
```

```
============================================================
===========================================
Installing:
 percona-release                    noarch    1.0-7
/percona-release-latest.noarch    18 k

Transaction Summary
============================================================
===========================================
Install  1 Package
```

Enable the percona repository:

```
[root@WP11 ~]# percona-release enable psmdb-40 release
* Enabling the Percona Original repository
<*> All done!
```

And then, install percona-server-mongodb:

```
[root@WP11 ~]# yum install percona-server-mongodb
============================================================
===========================================
 Package                            Arch      Version
Repository                       Size
============================================================
===========================================
Installing:
 percona-server-mongodb             x86_64    4.0.5-2.el7
psmdb-40-release-x86_64          4.8 k
Installing for dependencies:
 libpcap                            x86_64    14:1.5.3-11.
el7    base                             138 k
 numactl                            x86_64    2.0.9-7.el7
base                              66 k
 numactl-libs                       x86_64    2.0.9-7.el7
base                              29 k
 percona-server-mongodb-mongos      x86_64    4.0.5-2.el7
psmdb-40-release-x86_64          8.7 M
 percona-server-mongodb-server      x86_64    4.0.5-2.el7
psmdb-40-release-x86_64           18 M
 percona-server-mongodb-shell       x86_64    4.0.5-2.el7
psmdb-40-release-x86_64          9.7 M
 percona-server-mongodb-tools       x86_64    4.0.5-2.el7
psmdb-40-release-x86_64           26 M

Transaction Summary
============================================================
===========================================
Install  1 Package (+7 Dependent packages)
```

Now, you need to start the Percona MongoDB service.

```
[root@WP11 ~]# service mongod start
Redirecting to /bin/systemctl start mongod.service
```

You can verify the status by filtering the “waiting” word in the log file:

```
[root@WP11 ~]# grep “waiting” /var/log/mongo/mongod.log
2019-02-20T02:11:29.790+0000 I NETWORK  [initandlisten] waiting for connections on port 27017
```

Default Configuration

The Percona MongoDB installation creates the /etc/mongod.conf config file.

```
[root@WP11 ~]# cat /etc/mongod.conf
# mongod.conf, Percona Server for MongoDB
# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/
# Where and how to store data.
storage:
  dbPath: /var/lib/mongo
  journal:
    enabled: true
#  engine: mmapv1
#  engine: wiredTiger
#  engine: inMemory
# Storage engine various options
#  More info for mmapv1: https://docs.mongodb.com/v4.0/reference/configuration-options/#storage-mmapv1-options
#  mmapv1:
#    preallocDataFiles: true
#    nsSize: 16
#    quota:
#      enforced: false
#      maxFilesPerDB: 8
#    smallFiles: false
#  More info for wiredTiger: https://docs.mongodb.com/v4.0/reference/configuration-options/#storage-wiredtiger-options
#  wiredTiger:
#    engineConfig:
#      cacheSizeGB: 1
#      checkpointSizeMB: 1000
#      statisticsLogDelaySecs: 0
#      journalCompressor: snappy
#      directoryForIndexes: false
#    collectionConfig:
```

```
#      blockCompressor: snappy
#    indexConfig:
#      prefixCompression: true
#  More info for inMemory: https://www.percona.com/doc/percona-server-for-mongodb/4.0/inmemory.html#configuring-percona-memory-engine
#  inMemory:
#    engineConfig:
#      inMemorySizeGB: 1
#      statisticsLogDelaySecs: 0
# Two options below can be used for wiredTiger and inMemory storage engines
#setParameter:
#    wiredTigerConcurrentReadTransactions: 128
#    wiredTigerConcurrentWriteTransactions: 128
# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongo/mongod.log
processManagement:
  fork: true
  pidFilePath: /var/run/mongod.pid
# network interfaces
net:
  port: 27017
  bindIp: 127.0.0.1
#security:
#operationProfiling:
#replication:
#sharding:
## Enterprise-Only Options:
#auditLog:
#snmp:
```

Let's see these parameters in detail.

- systemLog:
    - destination: The destination to which Percona MongoDB sends all log output.
    - logAppend: When true, Percona MongoDB appends new entries to the end of the existing log file when the instance restarts.
    - path: Specify a log file path.
- storage:
    - dbPath: Specify a data directory path.
    - journal: Enable or disable the durability journal to ensure data files remain valid and recoverable.

- processManagement:
    - fork: Enable a daemon mode that runs the Percona MongoDB process in the background.
    - pidFilePath: Specifies a file location to hold the process ID of the MongoDB process.
    - timeZoneInfo: The full path from which to load the time zone database.
- net:
    - port: The TCP port on which the Percona MongoDB instance listens for client connections.
    - bindIp: The hostnames and/or IP addresses and/or full Unix domain socket paths on which Percona MongoDB should listen for client connections.

Optional Percona MongoDB Configuration

Let's see some variables that we can add in our config files. It's recommended as a basic configuration but it depends on your infrastructure.

```
storage:
    engine: wiredTiger
    mmapv1:
        smallFiles: false
net:
    bindIp: 0.0.0.0
setParameter:
   enableLocalhostAuthBypass: true
replication:
   replSetName: my_mongodb_0
sharding:
    clusterRole: shardsvr
security.keyFile: /etc/mongo-cluster.key
```

To see in detail these variables, you can follow this link.

PostgreSQL

PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions.

Installation

To install the PostgreSQL packages manually, you can follow this link.

Another way to install it is using yum or apt repositories.

In our example, let's see the yum repository installation of PostgreSQL 11 on CentOS 7.

PostgreSQL Server IP Address: 192.168.100.185

Install the repository:

```
[root@WP12 ~]# yum install https://download.postgresql.org/pub/repos/yum/11/redhat/rhel-7-x86_64/pgdg-centos11-11-2.noarch.rpm
=======================================================================================================
 Package                            Arch      Version     Repository                         Size
=======================================================================================================
Installing:
 pgdg-centos11                      noarch    11-2        /pgdg-centos11-11-2.noarch       2.7 k

Transaction Summary
=======================================================================================================
Install  1 Package
```

Installing PostgreSQL client:

```
[root@WP12 ~]# yum install postgresql11
========================================================================================================
 Package                            Arch      Version            Repository                         Size
========================================================================================================
Installing:
 postgresql11                       x86_64    11.2-1PGDG.rhel7   pgdg11                            1.6 M
Installing for dependencies:
 libicu                             x86_64    50.1.2-17.el7      base                              6.9 M
 postgresql11-libs                  x86_64    11.2-1PGDG.rhel7   pgdg11                            360 k

Transaction Summary
========================================================================================================
Install  1 Package (+2 Dependent packages)
```

Installing PostgreSQL server:

```
[root@WP12 ~]# yum install postgresql11-server
============================================================
===========================================
 Package                            Arch      Version
Repository                       Size
============================================================
===========================================
Installing:
 postgresql11-server                x86_64    11.2-1PGDG.
rhel7   pgdg11                          4.7 M

Transaction Summary
============================================================
===========================================
Install  1 Package
```

You can initialize your PostgreSQL database:

```
[root@WP12 ~]# /usr/pgsql-11/bin/postgresql-11-setup initdb
Initializing database ... OK
```

Enable the PostgreSQL service:

```
[root@WP12 ~]# systemctl enable postgresql-11
Created symlink from /etc/systemd/system/multi-user.target.
wants/postgresql-11.service to /usr/lib/systemd/system/post-
gresql-11.service.
```

Now, you can start the PostgreSQL service.

```
[root@WP12 ~]# systemctl start postgresql-11
```

Default Configuration

In the PostgreSQL datadir, by default /var/lib/pgsql/11/data/, you have different configuration files:

- pg_hba.conf: Client authentication is controlled by this file.
- pg_ident.conf: User name maps are defined in this ident map file.
- postgresql.conf: It’s the main server configuration file.
- postmaster.opts: A file recording the command-line options the server was last started with.

Let’s see these files one by one.

- pg_hba.conf

```
# TYPE  DATABASE        USER            ADDRESS         METHOD
# “local” is for Unix domain socket connections only
local   all             all                             peer
# IPv4 local connections:
host    all             all             127.0.0.1/32    ident
# IPv6 local connections:
host    all             all             ::1/128         ident
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                             peer
host    replication     all             127.0.0.1/32    ident
host    replication     all             ::1/128         ident
```

- pg_ident.conf

```
# MAPNAME       SYSTEM-USERNAME         PG-USERNAME
```

- postgresql.conf (We will only see the uncommented lines for space reasons)

```
max_wal_size = 1GB
min_wal_size = 80MB
log_timezone = ‘UTC’
datestyle = ‘iso, mdy’
timezone = ‘UTC’
default_text_search_config = ‘pg_catalog.english’
```

- postmaster.opts

```
/usr/pgsql-11/bin/postgres “-D” “/var/lib/pgsql/11/data/”
```

Let’s see the parameters in detail.

- max_wal_size: Maximum size the WAL is allowed to grow between the control points.
- min_wal_size: When the WAL file is kept below this value, it is recycled for future use at a checkpoint, instead of being deleted.
- log_timezone: Sets the time zone used for timestamps written in the server log.
- datestyle: Sets the display formats for date and time values, as well as the rules

for interpreting ambiguous date input values.

- timezone: Sets the time zone for displaying and interpreting time stamps.
- default_text_search_config: Selects the text search configuration that is used by those variants of the text search functions that do not have an explicit argument specifying the configuration.

To see these variables in detail, you can follow the official documentation.

Synopsis

We have showed you some examples of how to install, configure and secure some of the most popular DB engines. To be able to perform get to the above procedures, you need to research, test, and analyse your available resources in order to get a well tuned deploy.

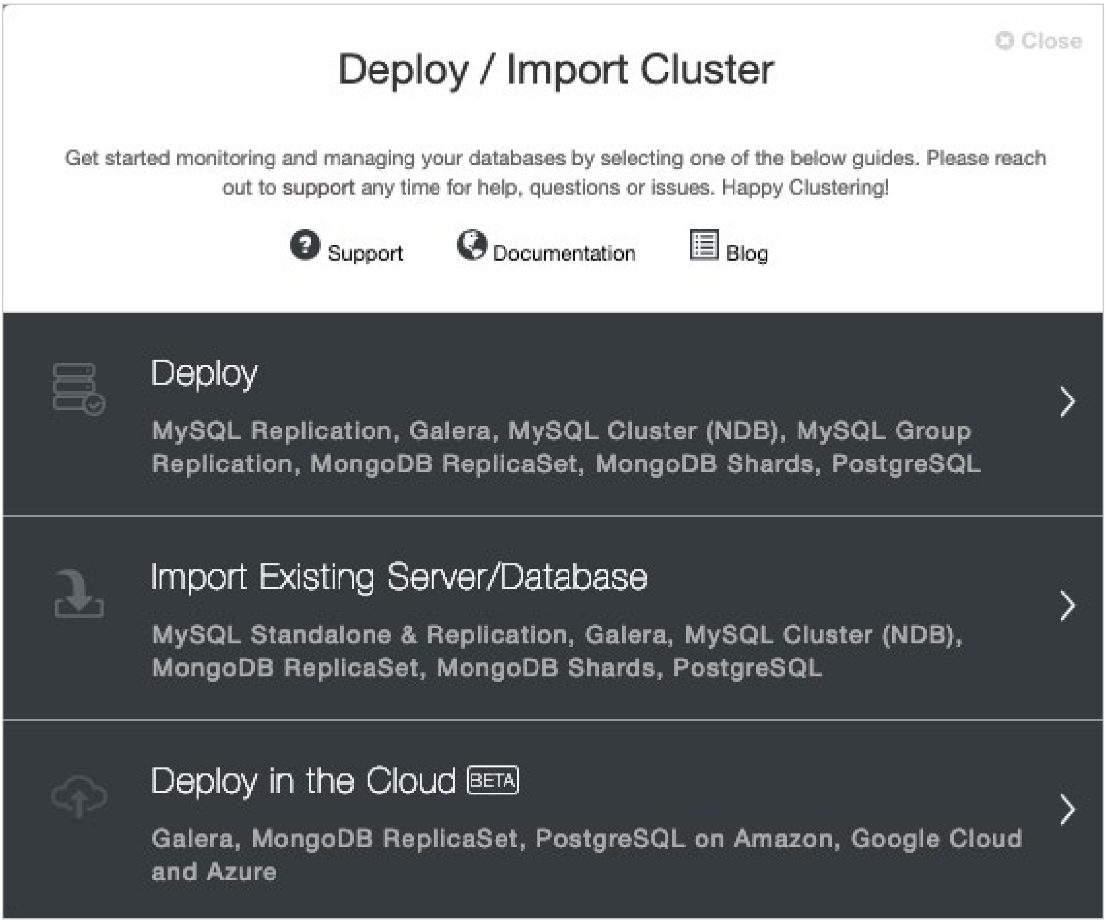
We will now look into ClusterControl, how it handles these tasks for you, delivering a secure well tuned environment.

How to Deploy Open Source Databases by Using ClusterControl

After seeing how we can deploy some of the most common open source databases manually, let's see how ClusterControl can make our lives easier.

Deploy

To perform a new installation from ClusterControl, simply select the option "Deploy" and follow the instructions that appear. Note that if you already have a database instance running, then you need to select the 'Import Existing Server/Database' instead.



Deploy / Import Cluster

Get started monitoring and managing your databases by selecting one of the below guides. Please reach out to support any time for help, questions or issues. Happy Clustering!

Deploy

MySQL Replication, Galera, MySQL Cluster (NDB), MySQL Group Replication, MongoDB ReplicaSet, MongoDB Shards, PostgreSQL

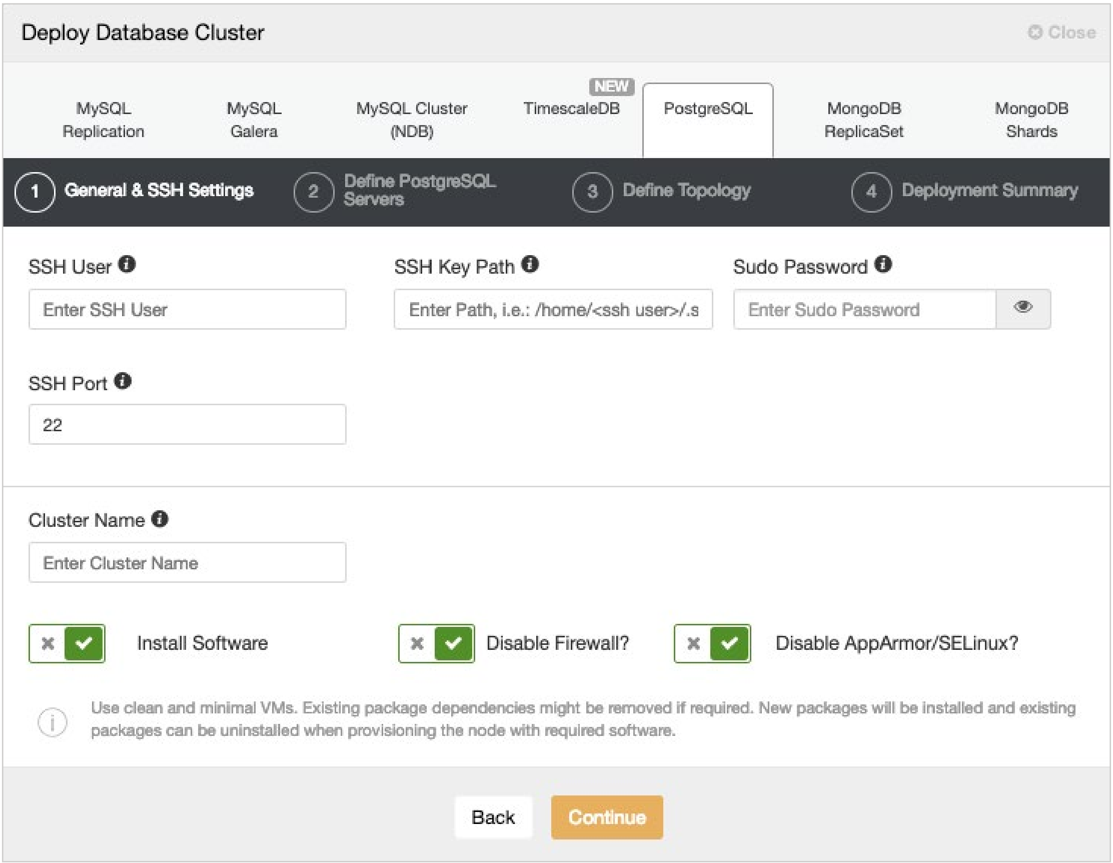
Import Existing Server/Database

MySQL Standalone & Replication, Galera, MySQL Cluster (NDB), MongoDB ReplicaSet, MongoDB Shards, PostgreSQL

Deploy in the Cloud BETA

Galera, MongoDB ReplicaSet, PostgreSQL on Amazon, Google Cloud and Azure

There are some differences depending on which technology we want to deploy.



SSH User: Enter SSH User

SSH Key Path: Enter Path, i.e.: /home/<ssh user>/.s

Sudo Password: Enter Sudo Password

SSH Port: 22

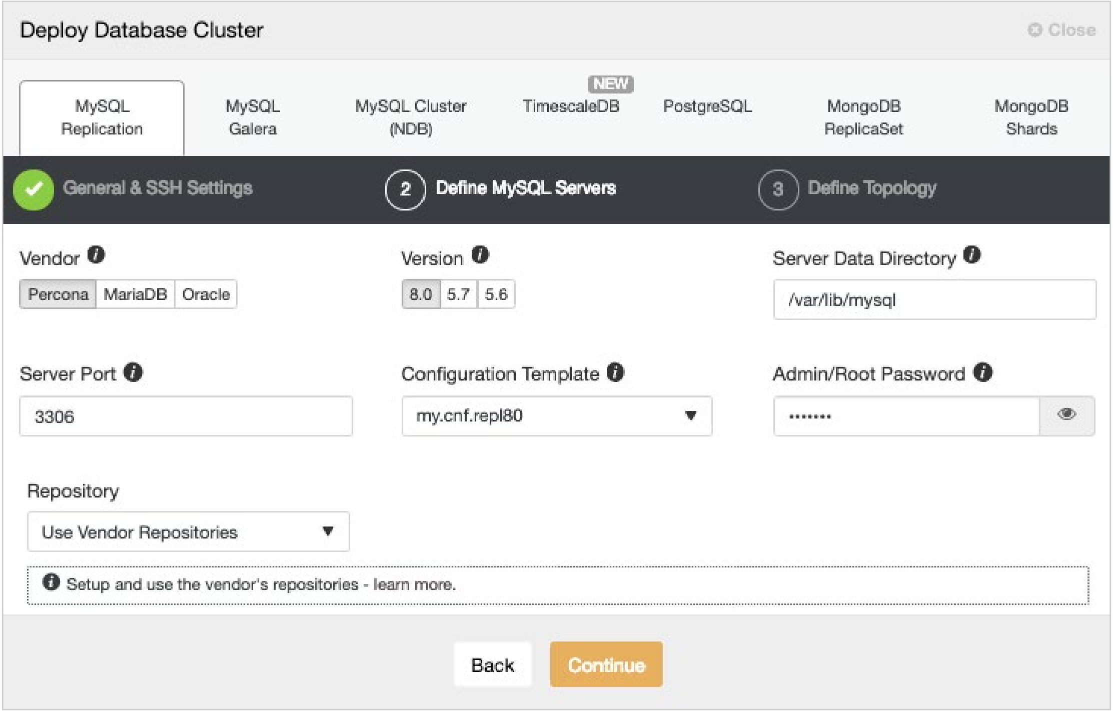
Cluster Name: Enter Cluster Name

☒ ☑ Install Software ☒ ☑ Disable Firewall? ☒ ☑ Disable AppArmor/SELinux?

Use clean and minimal VMs. Existing package dependencies might be removed if required. New packages will be installed and existing packages can be uninstalled when provisioning the node with required software.

In the deploy section, we need first to select the database technology, then, we must specify User, Key or Password and port to connect by SSH to our new database host. We also need a name for our new cluster and if we want ClusterControl to install the corresponding software and configurations for us.

Proper passwordless SSH setup from ClusterControl node to all nodes (including ClusterControl node) is mandatory. Before performing any operation on the managed node, the node must be accessible via SSH without using password but using key-based authentication instead.



Vendor: [✓ Percona] MariaDB Oracle

Version: [✓ 8.0] 5.7 5.6

Server Data Directory: /var/lib/mysql

Server Port: 3306

Configuration Template: my.cnf.repl80

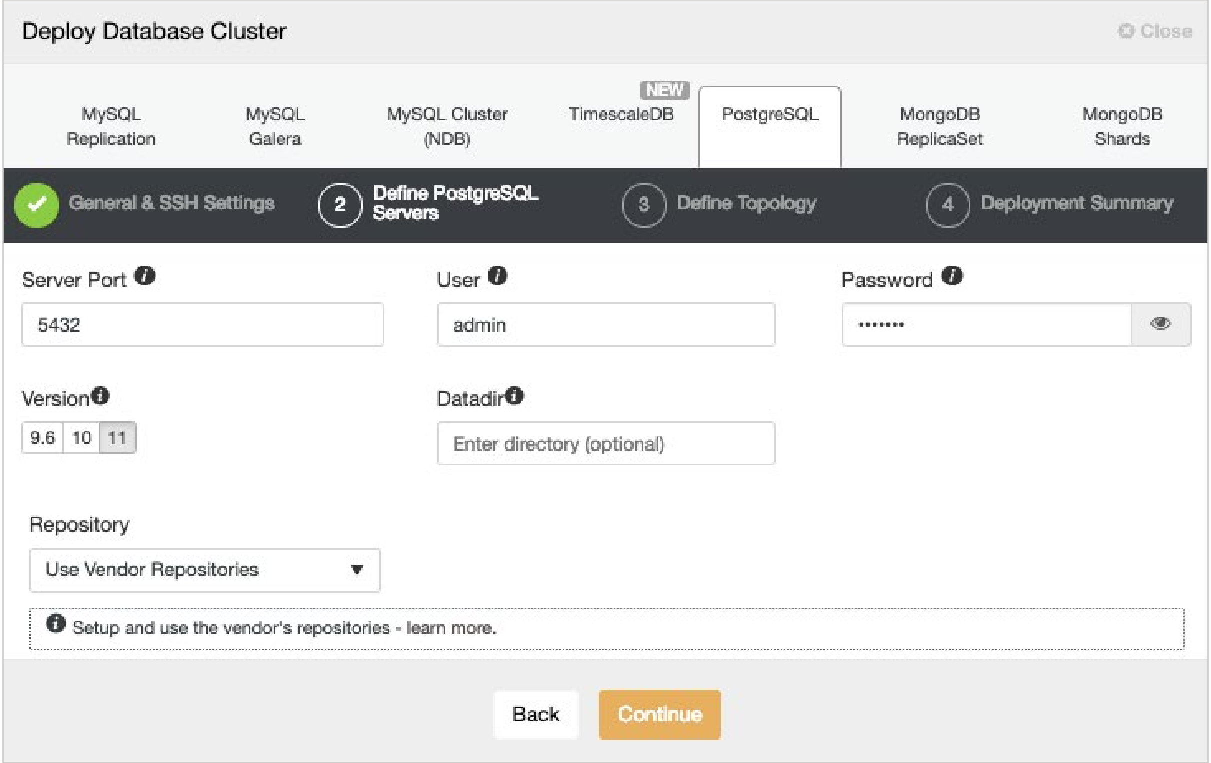
Admin/Root Password: •••••••

Repository: Use Vendor Repositories

Setup and use the vendor's repositories - learn more.

After setting up the SSH access information, we must define the database information, like vendor, version and user. We can also specify which repository to use.

The asked information will be different depending on which database technology we selected.



Server Port: 5432

User: admin

Password: •••••••

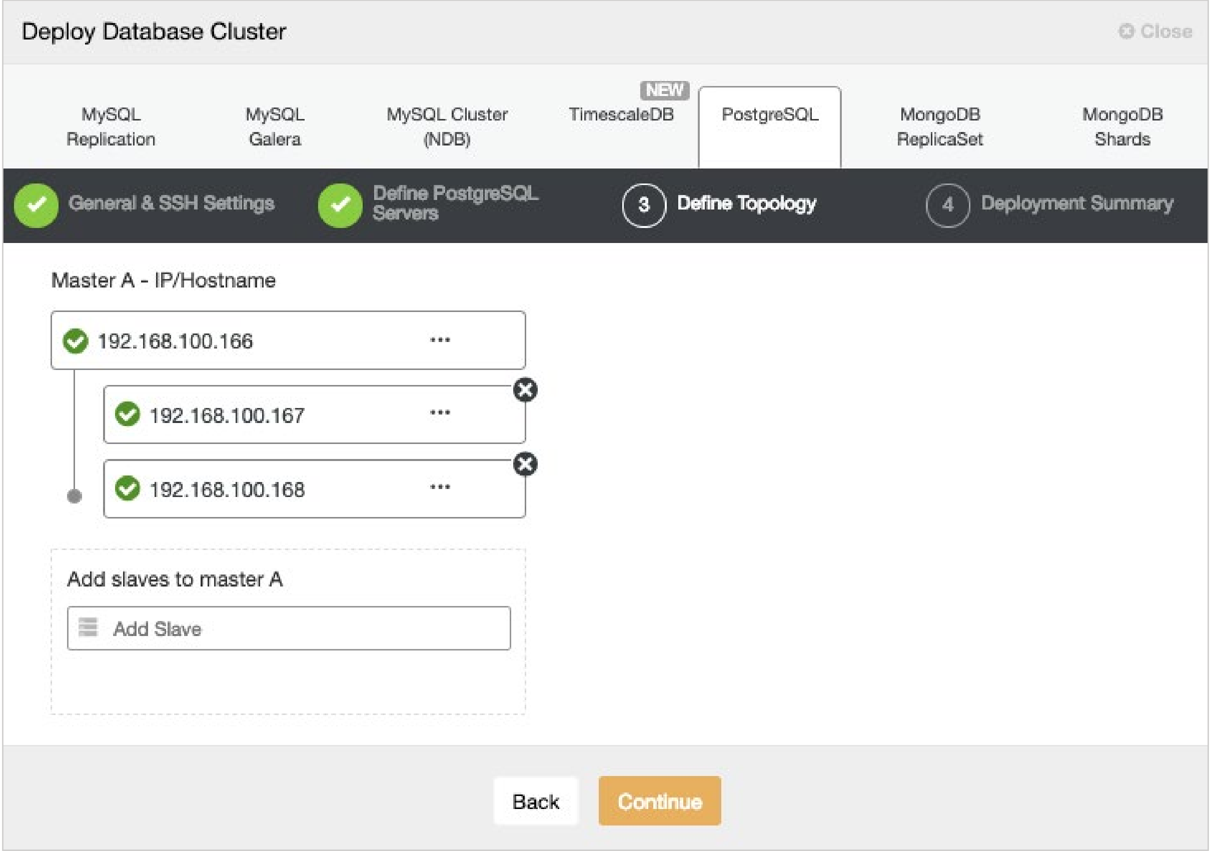
Version: 9.6 | 10 | 11

Datadir: Enter directory (optional)

Repository: Use Vendor Repositories

Setup and use the vendor's repositories - learn more.

In the next step, we need to specify will be our cluster topology.



Master A - IP/Hostname

192.168.100.166

192.168.100.167

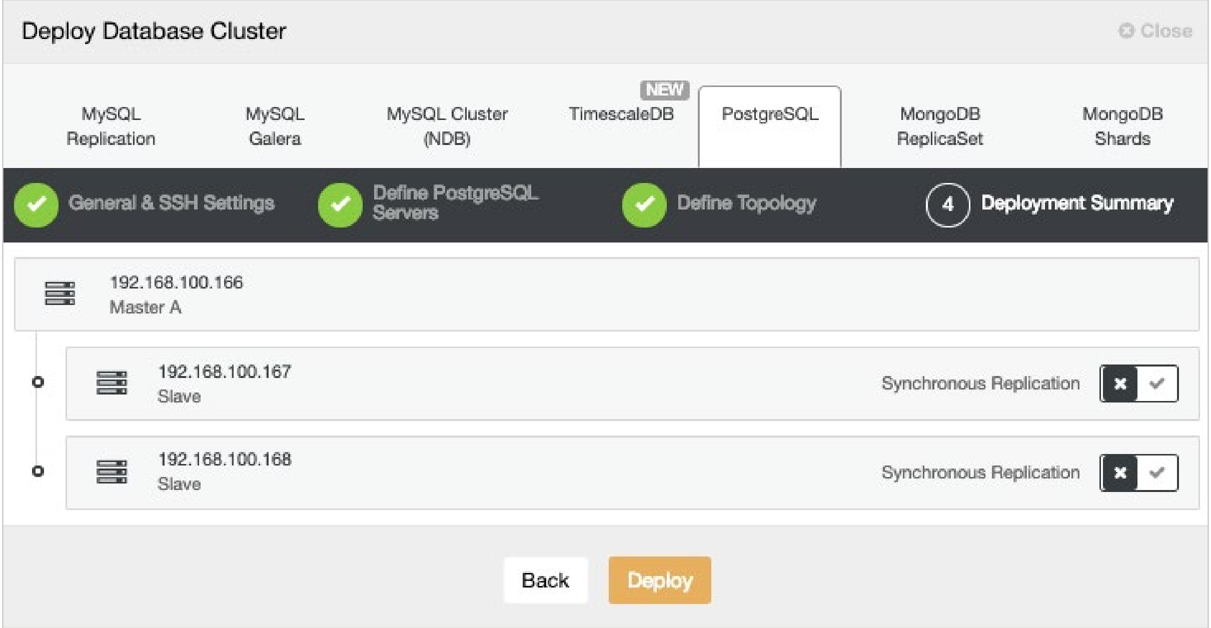
192.168.100.168

Add slaves to master A

Add Slave

When adding our servers, we can enter IP or hostname.

In the last step, we can choose if our replication will be Synchronous or Asynchronous.

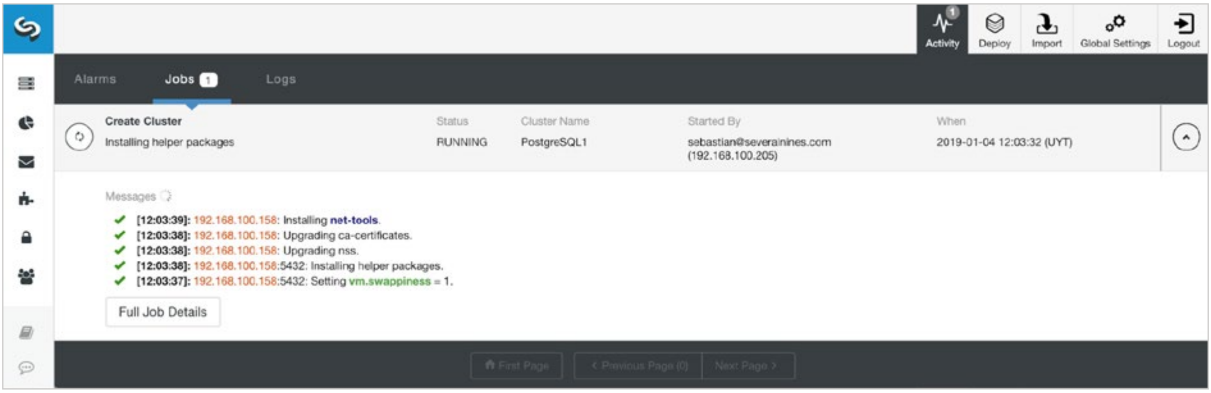


192.168.100.166
Master A

192.168.100.167
Slave
Synchronous Replication ☒

192.168.100.168
Slave
Synchronous Replication ☒

We can monitor the status of the creation of our new cluster from the ClusterControl activity monitor.

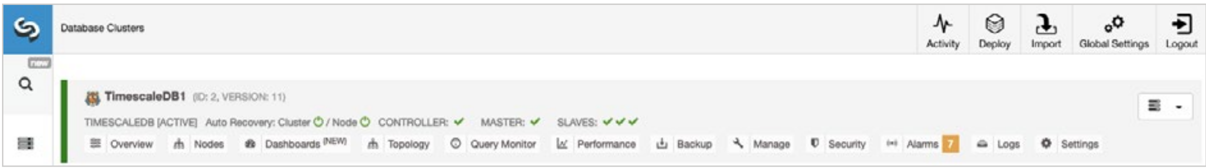


Create Cluster	Status	Cluster Name	Started By	When
Installing helper packages	RUNNING	PostgreSQL1	sebastian@severalnines.com (192.168.100.205)	2019-01-04 12:03:32 (UYT)

Messages

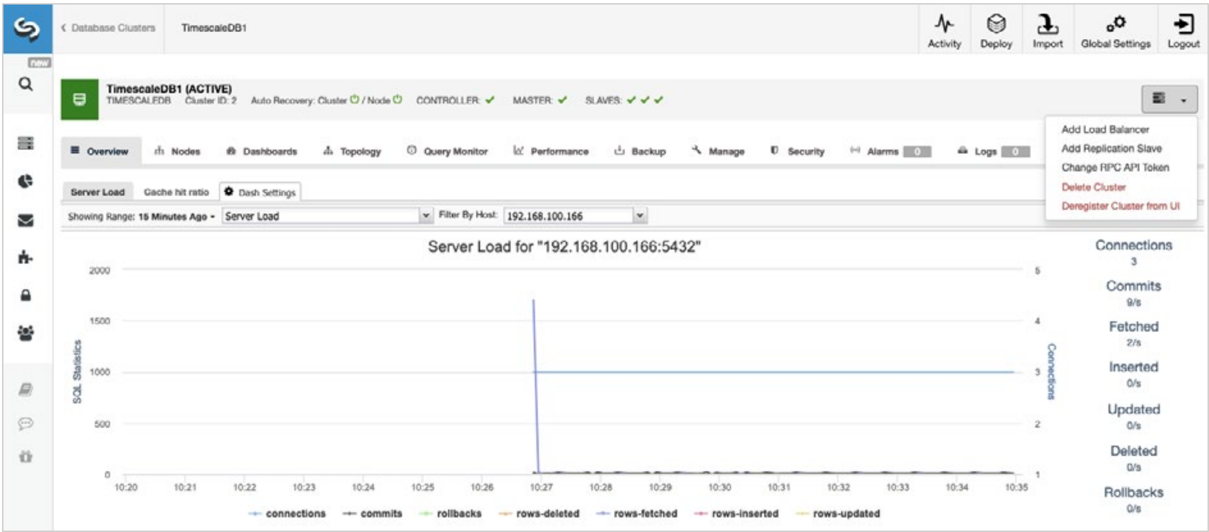
```
✓ [12:03:39]: 192.168.100.158: Installing net-tools.
✓ [12:03:38]: 192.168.100.158: Upgrading ca-certificates.
✓ [12:03:38]: 192.168.100.158: Upgrading nss.
✓ [12:03:38]: 192.168.100.158:5432: Installing helper packages.
✓ [12:03:37]: 192.168.100.158:5432: Setting vm.swappiness = 1.
```

Once the task is finished, we can see our new cluster in the main ClusterControl screen.



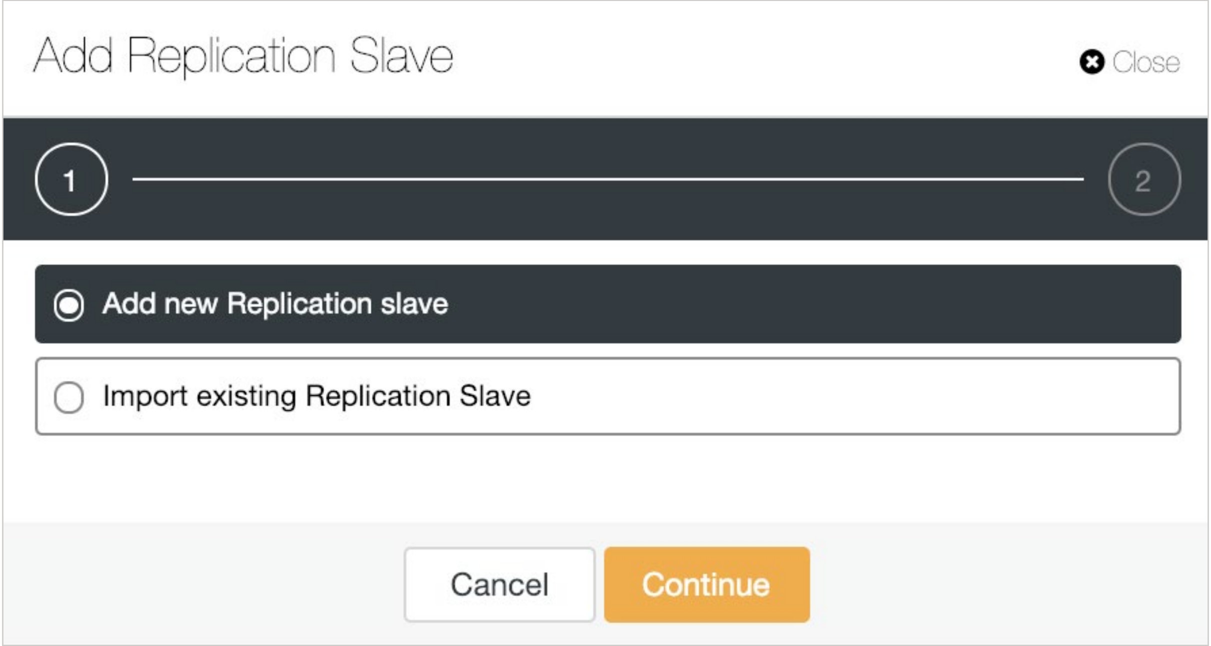
TimescaleDB1 (ID: 2, VERSION: 11)
TIMESCALEDB [ACTIVE] Auto Recovery: Cluster / Node CONTROLLER: ✓ MASTER: ✓ SLAVES: ✓✓✓

Once we have our cluster created, we can perform several tasks on it, like adding a load balancer (HAProxy) or a new replica.



Scaling

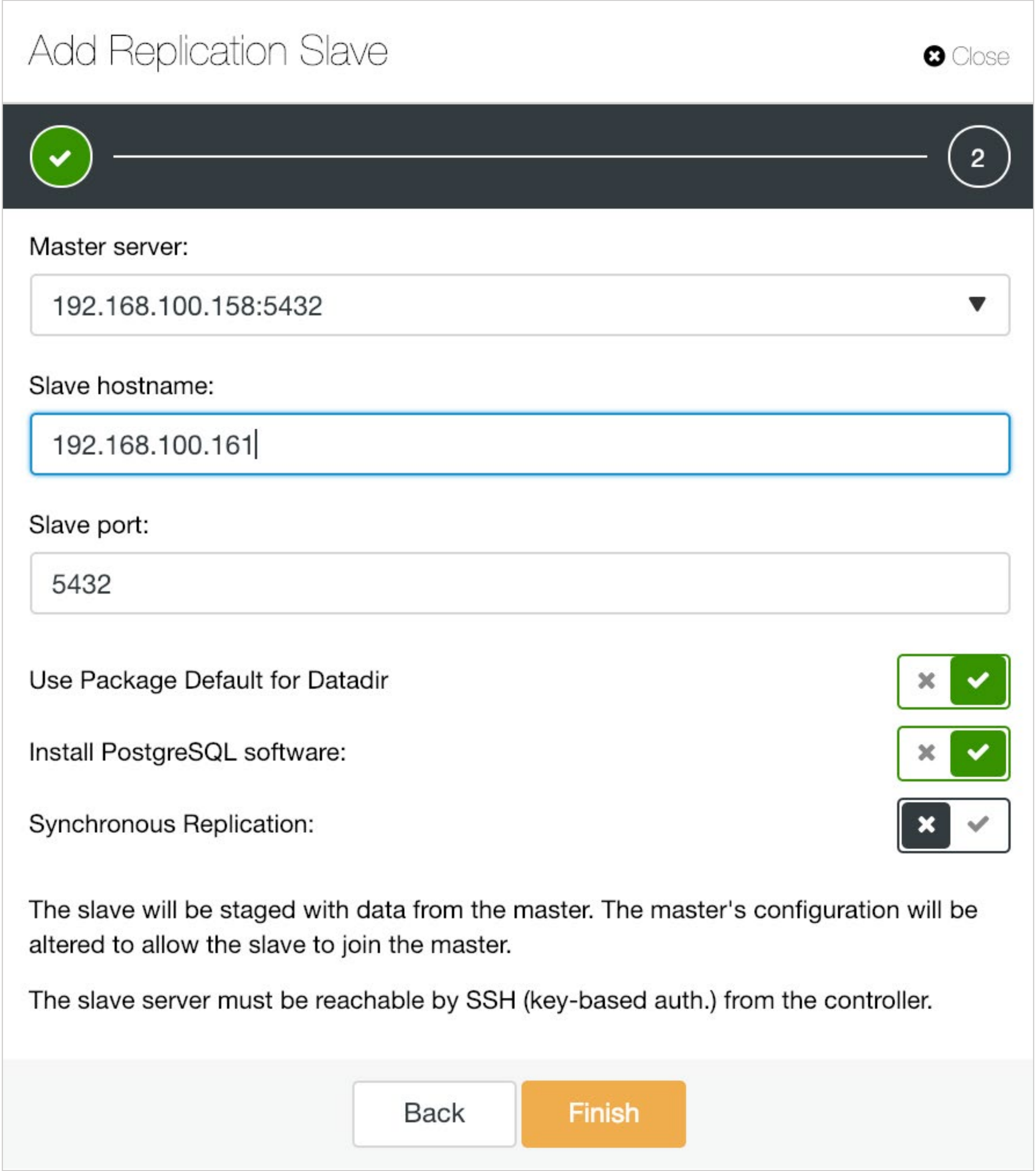
If we go to cluster actions and select “Add Replication Slave”, we can either create a new replica from scratch, or add an existing database as a replica.


Add Replication Slave

● Add new Replication slave

○ Import existing Replication Slave

Let’s see how adding a new replication slave can be a really easy task.



Add Replication Slave

Master server:

192.168.100.158:5432

Slave hostname:

192.168.100.161

Slave port:

5432

Use Package Default for Datadir ✓

Install PostgreSQL software: ✓

Synchronous Replication: ✗

The slave will be staged with data from the master. The master's configuration will be altered to allow the slave to join the master.

The slave server must be reachable by SSH (key-based auth.) from the controller.

As you can see in the image, we only need to choose our Master server, enter the IP address for our new slave server and the database port. Then, we can choose if we want ClusterControl to install the software for us, and if the replication slave should be Synchronous or Asynchronous.

In this way, we can add as many replicas as we want and spread read traffic between them using a load balancer, which we can also implement with ClusterControl.

Load Balancing

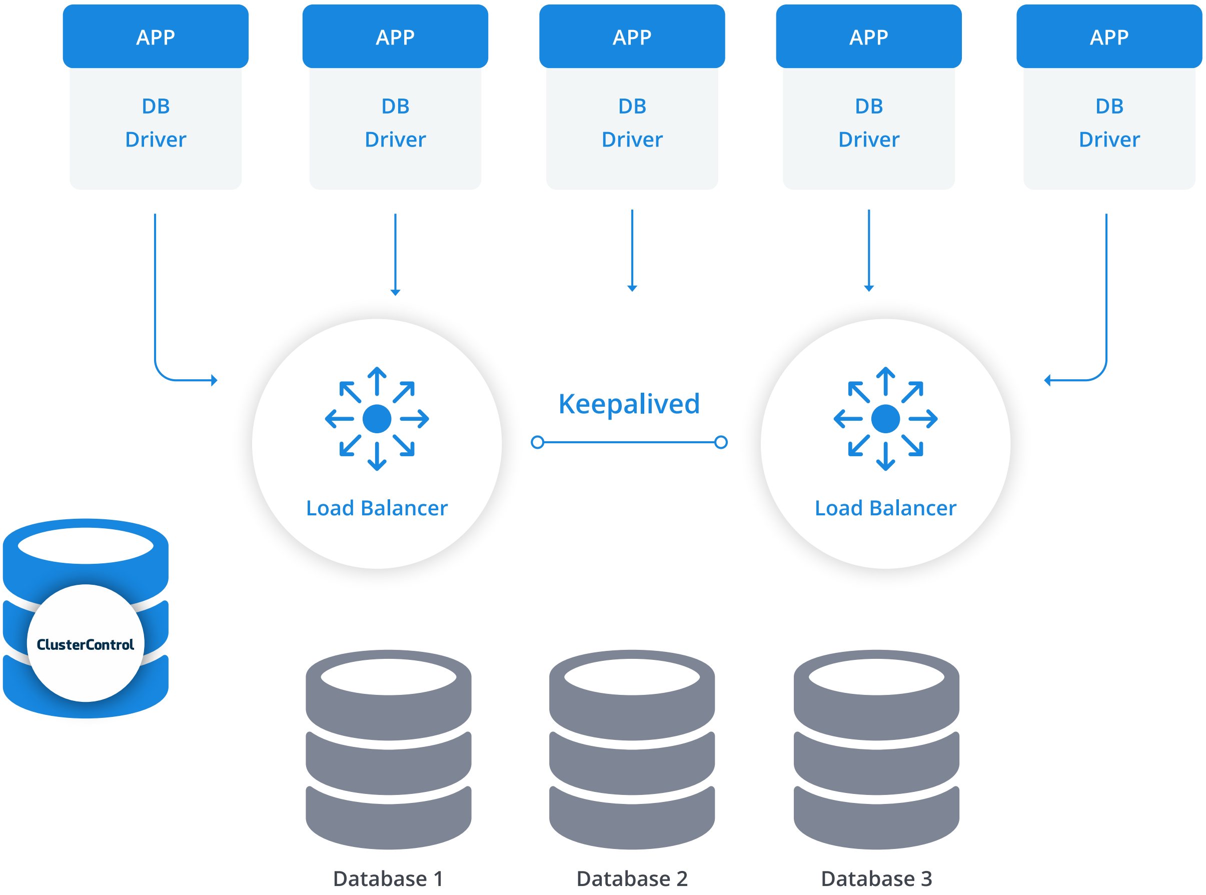
Load balancers can be used to manage the traffic from your application, to get the most out of your database architecture.

Not only is it useful for balancing the load of our databases, it also helps applications get redirected to the available/healthy nodes and even specify ports with different roles.

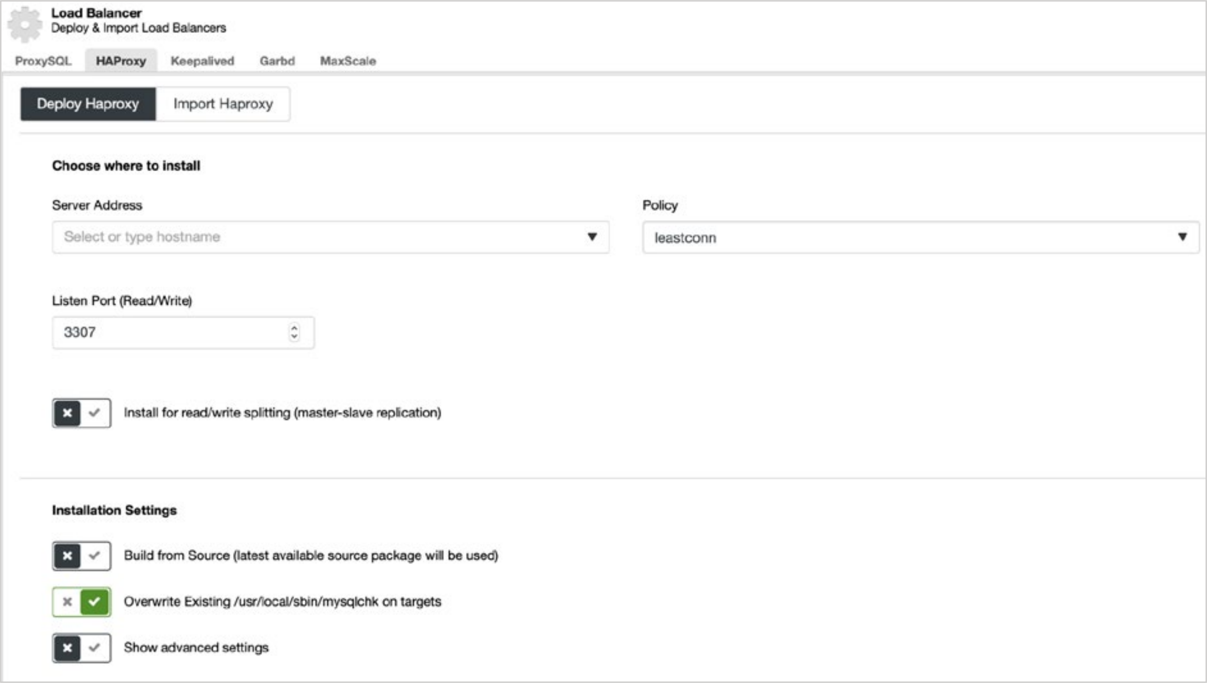
With ClusterControl, you can deploy ProxySQL, HAProxy or MaxScale as Load Balancer.

By using a load balancer, you can distribute the traffic from one origin to one or more destinations and can define specific rules and/or protocols for this task. If any of the destinations stops responding, it is marked as offline, and the traffic is sent to the rest of the available destinations.

Keepalived is a service that allows you to configure a virtual IP within an active/passive group of servers. This virtual IP is assigned to an active server. If this server fails, the IP is automatically migrated to the "Secondary" passive server, allowing it to continue working with the same IP in a transparent way for the systems.



To perform a load balancer deployment, select the option "Add Load Balancer" in the cluster actions and fill the asked information.



Choose where to install

Server Address: Select or type hostname

Policy: leastconn

Listen Port (Read/Write): 3307

☒ Install for read/write splitting (master-slave replication)

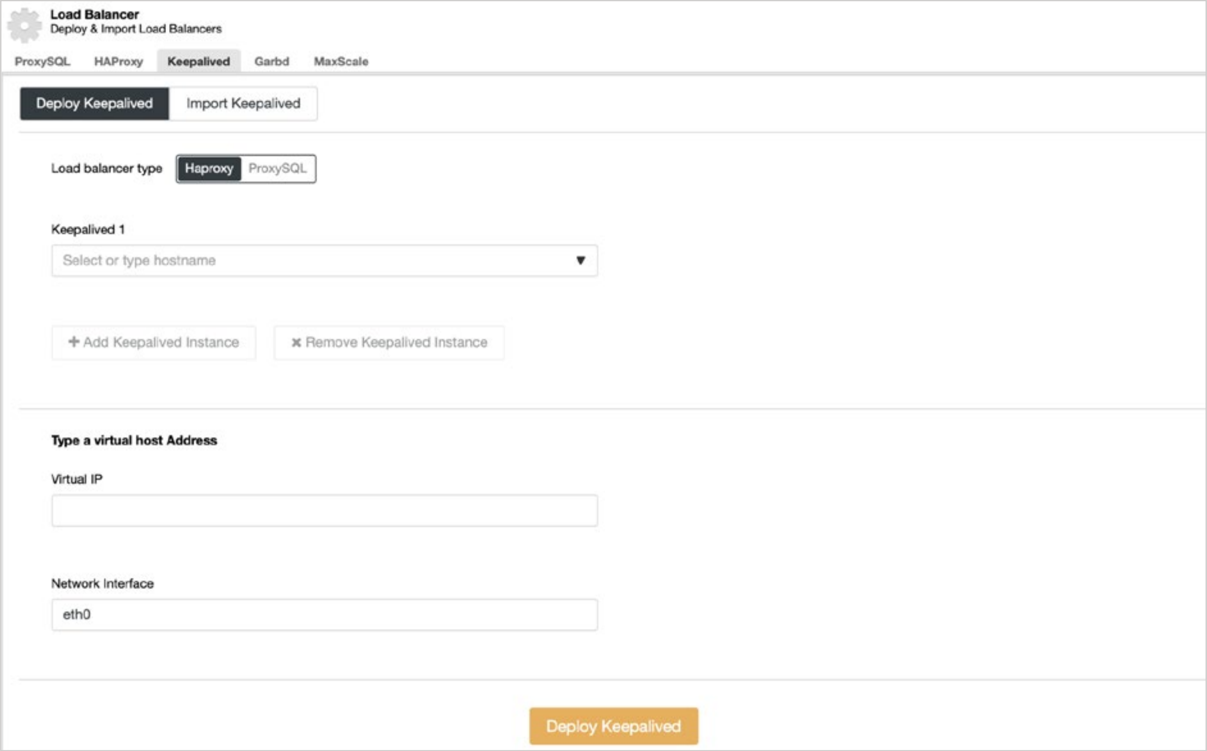
Installation Settings

☒ Build from Source (latest available source package will be used)

☑ Overwrite Existing /usr/local/sbin/mysqlchk on targets

☒ Show advanced settings

To perform a keepalived deployment, select the cluster, go to “Manage” menu and “Load Balancer” section, and then select “Keepalived” option.



Load balancer type: [✓ Haproxy] ProxySQL

Keepalived 1: Select or type hostname

Type a virtual host Address

Virtual IP:

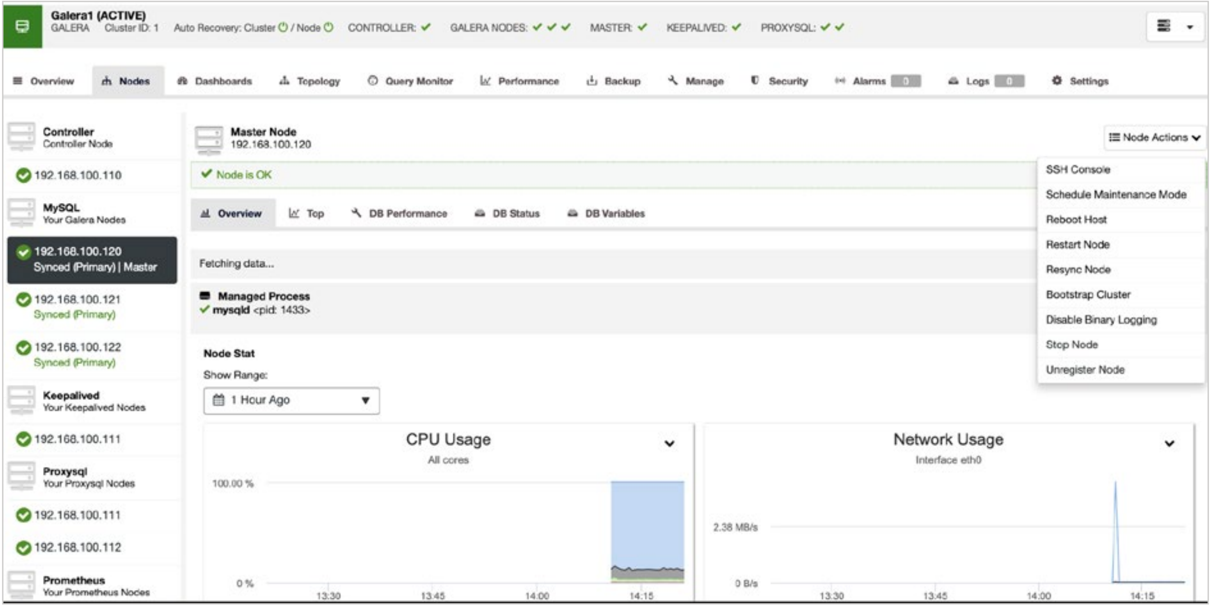
Network Interface: eth0

For your HA environment, you need to select the load balancer servers and the virtual IP address.

Keepalived uses a virtual IP and migrates it from one load balancer to another in case of failure, so your setup can continue to function normally.

Management

From ClusterControl, you can also perform different management tasks like scheduling backups and verifying them for integrity, automatic failover, encryption of traffic, topology changes, and so on. The options depend on the database engine that you are using.



Galera1 (ACTIVE)
GALERA Cluster ID: 1 Auto Recovery: Cluster / Node CONTROLLER: ✓ GALERA NODES: ✓ ✓ ✓ MASTER: ✓ KEEPALIVED: ✓ PROXYSQL: ✓ ✓

Master Node
192.168.100.120

✓ Node is OK

Fetching data...

Managed Process
✓ mysqld <pid: 1433>

Node Stat
Show Range: 1 Hour Ago

Conclusion

In this Whitepaper, we have listed some of the most popular DB engines. After that we went through some examples on how to install, tune and secure each of them. Getting to those step by step procedures involves research, experimentation and testing. You need to understand each step of what you're doing and the available options, and research them to pick the correct ones for your needs.

We finally introduced ClusterControl, a system that can help you close that gap and get a well secured, well tuned deploy of your chosen engine.

About ClusterControl

ClusterControl is the all-inclusive open source database management system for users with mixed environments that removes the need for multiple management tools. ClusterControl provides advanced deployment, management, monitoring, and scaling functionality to get your MySQL, MongoDB, and PostgreSQL databases up-and-running using proven methodologies that you can depend on to work. At the core of ClusterControl is it's automation functionality that let's you automate many of the database tasks you have to perform regularly like deploying new databases, adding and scaling new nodes, running backups and upgrades, and more.

Backup Management	Monitoring & Alerting	Deployment & Scaling	Upgrades & Patching	Security & Compliance
Operational Reporting	Configuration Management	Automatic Recovery & Repair	Performance Management	Automated Performance Advisors

About Severalnines

Severalnines provides automation and management software for database clusters. We help companies deploy their databases in any environment, and manage all operational aspects to achieve high-scale availability.

Severalnines' products are used by developers and administrators of all skills levels to provide the full 'deploy, manage, monitor, scale' database cycle, thus freeing them from the complexity and learning curves that are typically associated with highly available database clusters. Severalnines is often called the "anti-startup" as it is entirely self-funded by its founders. The company has enabled over 12,000 deployments to date via its popular product ClusterControl. Currently counting BT, Orange, Cisco, CNRS, Technicolor, AVG, Ping Identity and Paytrail as customers. Severalnines is a private company headquartered in Stockholm, Sweden with offices in Singapore, Japan and the United States. To see who is using Severalnines today visit:

https://www.severalnines.com/company

Related Resources

Deployment & Scaling with ClusterControl

ClusterControl provides a suite of database deployment tools, allowing cluster deployment, database importing, load balancing, hybrid deployments and more!

How to Deploy a Production-Ready MySQL or MariaDB Galera Cluster using ClusterControl

ClusterControl can be used to deploy open-source database clusters that are configured in complex topologies. The full high availability stack includes both database and proxy layers.

In this blog, we are going to show you how to deploy a production-grade Galera Cluster, complete with load balancers, for a high availability setup.

How to Deploy PostgreSQL for High Availability

In this blog, we will review some important concepts of High Availability, possible database HA architectures and useful components when implementing PostgreSQL HA. Then we will see how to use ClusterControl to deploy an entire high availability stack for PostgreSQL.

How to Deploy MongoDB for High Availability

MongoDB provides ReplicaSets to help you address high availability database requirements. Although support for replication and failover is built-in, it is not enough for the database to be considered production-ready. That requires a set of policies and procedures, like getting alerted in case of performance slowdowns, anomalies or failures in a live environment. Backups are essential. Being able to automatically recover from different types of failures can drastically reduce downtime.

Choosing which DB engine to use between all the options we have today is not an easy task. An that is just the beginning. After deciding which engine to use, you need to learn about it and actually deploy it to play with it. We plan to help you on that second step, and show you how to install, configure and secure some of the most popular open source DB engines. In this whitepaper we are going to cover these points , with the aim of fast tracking you on the deploy task.