Red Hat OpenStack Platform 10

Back Up and Restore the Overcloud Database

How to back up and restore the overcloud MariaDB database

Last Updated: 2018-11-01

Red Hat OpenStack Platform 10 Back Up and Restore the Overcloud Database

How to back up and restore the overcloud MariaDB database

Legal Notice

Abstract

A guide to backing up and restoring the Red Hat OpenStack Platform overcloud database.

Table of Contents

CHAPTER 1. INTRODUCTION

The Back Up and Restore procedure for the OpenStack overcloud is intended for disaster recovery scenarios, such as the failure of multiple nodes or data directory corruption affecting the data integrity of the entire cluster.

NOTE

For single node failures, bootstrap a new node from a working donor node.

PART I. BACKING UP AN OPENSTACK HA OVERCLOUD DATABASE

This procedure describes the steps to backup the MariaDB database in a running Galera cluster of an OpenStack overcloud.

CHAPTER 2. ABOUT OVERCLOUD COMPONENTS

For the purposes of backing up the OpenStack overcloud, consider the following components:

- **Pacemaker:** Pacemaker is a high-availability cluster resource manager. Pacemaker achieves maximum availability for cluster services by detecting and recovering from node- and resource-level failures. Pacemaker uses messaging and membership capabilities provided by Corosync.
- **MariaDB:** MariaDB is the SQL database service for the OpenStack overcloud.
- **Galera:** Galera is a synchronous, multi-master clustering solution for MariaDB. In the context of OpenStack, Galera provides high availability for the database service in HA deployments of OpenStack controller nodes.

On a typical HA cluster with 3 controller nodes, OpenStack services such as OpenStack Compute (**nova**) or OpenStack Networking (**neutron**) access the MariaDB database via HAProxy over a virtual IP address. HAProxy balances all incoming traffic to a single active controller node among the available controller nodes in the cluster. In a 3-node cluster, the other 2 controller nodes act as hot standby nodes, which synchronize with the first node using Galera's synchronous replication. If HAProxy detects that the first target node is "unavailable," it will select one of the two remaining hot standby nodes and balance incoming traffic to the selected node. When clients attempt to use an existing database connection from a failed node, HAProxy will migrate the database connections to the new node. As a side effect of this lazy migration approach, each client will recreate a SQL connection.

Consequently, backing up a running cluster requires identifying one of the idle controller nodes as the backup target to avoid any impact on the controller node that is currently servicing clients.

CHAPTER 3. LOCATE AN IDLE NODE

To backup a running cluster, identify an idle node as the back up target by connecting to the cluster via the HAProxy server; then, query the database to identify the host it is running on. The resulting node is the active node. Select one of the other hot standby nodes as the backup target.

For the purposes of a backing up a high availability cluster, Red Hat assumes at least three Galera cluster nodes. For example, overcloud nodes **`overcloud-controller-0`**, **`overcloud-controller-1`** and **`overcloud-controller-2`**:

```
overcloud-controller-0.example.com 192.168.1.1
overcloud-controller-1.example.com 192.168.1.2
overcloud-controller-2.example.com 192.168.1.3
```

Procedure

1. Retrieve the virtual IP address HAProxy is listening on to identify which node is active.

   ```
   [root@overcloud-controller-0]# grep -A1 mysql
   /etc/haproxy/haproxy.cfg
   ```

   The result will look similar to this:

   ```
   listen mysql
     bind 192.0.2.18:3306
   ```

2. Retrieve the username and password of a user allowed to connect to the virtual IP address. For example, the **nova** user and password from **`/etc/nova/nova.conf`** typically has these credentials.

   ```
   [root@overcloud-controller-0 heat-admin]# crudini --get
   /etc/nova/nova.conf database connection
   ```

   Or

   ```
   [root@overcloud-controller-0 heat-admin]# grep mysql
   /etc/nova/nova.conf
   ```

   The result should look something like this:

   ```
   connection=mysql+pymysql://nova:xvsZqeaJn2fYwMK8NbscAJ6xG@172.16.2.5
   /nova
   ```

   The username and password in the foregoing example are **nova** and **`xvsZqeaJn2fYwMK8NbscAJ6xG`** respectively. The next step requires the username and password retrieved in this step.

3. Connect to the database over the virtual IP address to get the name of the targeted node.

   ```
   [root@overcloud-controller-0]# mysql -u <username> -p -h 192.0.2.18
   -nNE -e "show variables like 'hostname';"
   Enter password: ****
   ```

Replace **<username>** with a user allowed to connect to the virtual IP address. The command line interface will prompt for a password. The result should look similar to this:

```
*********************** 1. row ***********************
hostname
overcloud-controller-0.example.com
```

In the foregoing example, **overcloud-controller-0** is the active node targeted by HAProxy.

4. Ensure the remaining hot standby nodes are connected and in sync. In the following example, assume **overcloud-controller-1** and **overcloud-controller-2** are the hot standby nodes.

```
[root@overcloud-controller-0]# curl overcloud-controller-
1.example.com:9200 overcloud-controller-2.example.com:9200
```

If the remaining nodes are connected and in sync with the active node targeted by HAProxy, the result should look similar to this:

```
Galera cluster node is synced.
Galera cluster node is synced.
```

IMPORTANT

Execute the backup procedure on a hot standby node that is connected to the cluster and synchronized with the active node that HAProxy is targeting.

In the foregoing example, both **overcloud-controller-1** and **overcloud-controller-2** are connected and synchronized. Identify a connected and synchronized hot standby node to backup the overcloud.

CHAPTER 4. BACK UP THE NODE

The back up procedure uses **mysqldump** with the **--single-transaction** option to avoid table locks. This approach ensures that the backup procedure limits the performance impact on a running cluster.

Since Galera only uses InnoDB tables, the **--single-transaction** only applies to InnoDB tables.

Procedure

1. Backup all tables other than the **mysql** tables.

```
mysql -u root -e "select distinct table_schema from
information_schema.tables where engine='innodb' and table_schema !=
'mysql';" -s -N | xargs mysqldump -u root --single-transaction --
databases > openstack_database-$TIMESTAMP.sql
```

2. Backup all grants and account info from the database.

```
mysql -uroot -s -N -e "SELECT CONCAT('\"SHOW GRANTS FOR
''',user,'''@''',host,''';\"') FROM mysql.user where (length(user) >
0 and user NOT LIKE 'root')" | xargs -n1 mysql -u root -s -N -e |
sed 's/$/;/' > grants.sql
```

   The backup procedure requires separately backing up the user accounts and grants, because they use ISAM tables.

Finally, store a copy of all backup files at a remote location.

PART II. RESTORING AN OPENSTACK HA OVERCLOUD DATABASE

This procedure describes the steps to restore the MariaDB database in a running Galera cluster of an OpenStack overcloud **if ALL instances in the Galera cluster fail**. Before executing this procedure, contact Red Hat support to confirm that database restoration is the appropriate solution.

WARNING

Database restoration implies a database outage. No OpenStack service will be able to connect to the database during the restore procedure.

Before attempting to restore the database, verify that the database in each Galera node is beyond recovery. If the database is corrupt and beyond recovery in a single node but healthy in another node in the Galera cluster, allow Galera to resynchronize or re-replicate the local instance of the database from the healthy instance on another node.

The inability to start the Galera service on at least one or even all nodes does not necessarily imply that the database is corrupt and beyond recovery. The inability to start the service may be due to a separate issue in the Galera, MariaDB or Pacemaker services. Resolve those issues before attempting to use this procedure.

IMPORTANT

If Galera is working and the database is healthy on any of the HA nodes, **DO NOT** follow these steps. The replication/synchronization functionality of Galera should restore the cluster.

Finally, if objects have been created or deleted since the last backup, they will need to be added to or deleted from the database manually after completing the restore procedure.

CHAPTER 5. IDENTIFY THE VIRTUAL IP ADDRESS

For the purposes of a restoring a high availability cluster, Red Hat assumes at least three cluster nodes. For example, overcloud nodes **overcloud-controller-0**, **overcloud-controller-1** and **overcloud-controller-2**:

```
overcloud-controller-0.example.com 192.168.1.1
overcloud-controller-1.example.com 192.168.1.2
overcloud-controller-2.example.com 192.168.1.3
```

Procedure

Identify the virtual IP address that OpenStack services use to access the database.

Run the following command:

```
[root@overcloud-controller-0]# grep -A1 mysql /etc/haproxy/haproxy.cfg
```

It should retrieve the virtual IP address and port. For example:

```
listen mysql
  bind 192.168.1.10::3306 transparent
```

In the foregoing example, the virtual IP address is **192.168.1.10**.

CHAPTER 6. CONFIGURE IPTABLES AND PACEMAKER

Configure the firewall and Pacemaker to isolate the database cluster.

Procedure

1. Insert an **iptables** rule on each controller node to drop inbound connections over the the virtual IP address to the database port.

```
[root@overcloud-controller-0]# iptables -I INPUT -d 192.168.1.10 -p
tcp --dport 3306 -j DROP
[root@overcloud-controller-1]# iptables -I INPUT -d 192.168.1.10 -p
tcp --dport 3306 -j DROP
[root@overcloud-controller-2]# iptables -I INPUT -d 192.168.1.10 -p
tcp --dport 3306 -j DROP
```

2. From one of the controller nodes, remove the **mariadb-galera** service from Pacemaker management.

```
[root@overcloud-controller-0]# pcs resource unmanage galera
```

CHAPTER 7. UPDATE THE CONFIGURATION

Update the Galera configuration on **ALL** Galera nodes.

Procedure

1. Locate the Galera configuration file and open it.

```
grep wsrep_cluster_address /etc/my.cnf.d/galera.cnf
vi /etc/my.cnf.d/galera.cnf
```

2. Comment out the **wsrep_cluster_address** setting.

```
#wsrep_cluster_address = gcomm://overcloud-controller-0,overcloud-
controller-1,overcloud-controller-2
```

3. Comment out the **wsrep_provider** setting.

```
#wsrep_provider = /usr/lib64/galera/libgalera_smm.so
```

4. Save the configuration file and close it.

CHAPTER 8. STOP THE MARIADB SERVICE AND PREPARE THE DIRECTORIES

The procedure requires stopping the MariaDB service and making a copy of the directories in case of data directory corruption. The copy of the directories is useful for root-cause-analysis investigation. Perform the following procedure on **ALL** nodes.

Procedure

1. Stop the MariaDB service on **ALL** controller nodes.

```
[root@overcloud-controller-0] mysqladmin -u root shutdown
```

2. Move the existing **`mariadb`** data directories and prepare new data directories on **ALL** controller nodes.

```
[root@overcloud-controller-0] mv /var/lib/mysql /var/lib/mysql-save
[root@overcloud-controller-0] mkdir /var/lib/mysql
[root@overcloud-controller-0] chown mysql:mysql /var/lib/mysql
[root@overcloud-controller-0] chmod 0755 /var/lib/mysql
[root@overcloud-controller-0] mysql_install_db --
datadir=/var/lib/mysql --user=mysql
[root@overcloud-controller-0] chown -R mysql:mysql /var/lib/mysql/
[root@overcloud-controller-0] restorecon -R /var/lib/mysql
```

CHAPTER 9. START MARIADB LOCALLY AND SETUP ACCOUNTS

To start MariaDB and setup base accounts, execute the following procedure.

Procedure

1. Start the MariaDB service and set up base accounts **ALL** controller nodes.

```
mysqld_safe --wsrep-provider=none &
MYSQL_PASSWORD=$(/bin/hiera -c /etc/puppet/hiera.yaml
mysql::server::root_password)
CLUSTERCHECK_PASSWORD=$(/bin/hiera -c /etc/puppet/hiera.yaml
mysql_clustercheck_password)
/usr/bin/mysql -u root -p -e "CREATE USER
'clustercheck'@'localhost';"
/usr/bin/mysql -u root -p -e "GRANT PROCESS ON *.* TO
'clustercheck'@'localhost' IDENTIFIED BY '$CLUSTERCHECK_PASSWORD';"
/usr/bin/mysqladmin -u root password $MYSQL_PASSWORD
mysqladmin -u root -p$MYSQL_PASSWORD shutdown
```

   The director creates a root password and stores it in **`/root/.my.cnf`**.

2. On **ALL** controller nodes, ensure that the **`mysql`** process is not running.

```
[root@overcloud-controller-0]# ps -ef | grep mysql
[root@overcloud-controller-1]# ps -ef | grep mysql
[root@overcloud-controller-2]# ps -ef | grep mysql
```

3. Uncomment the **`wsrep_cluster_address`** and **`wsrep_provider`** settings in **`/etc/my.cnf.d/galera.cnf`** on **ALL** controller nodes.

```
wsrep_cluster_address = gcomm://overcloud-controller-0,overcloud-
controller-1,overcloud-controller-2
wsrep_provider = /usr/lib64/galera/libgalera_smm.so
```

4. On one of the nodes in the cluster, bring the Galera cluster up with Pacemaker.

```
[root@overcloud-controller-0]# pcs resource manage galera
[root@overcloud-controller-0]# pcs resource cleanup galera
```

5. Make sure that cluster is running.

```
[root@overcloud-controller-0]# pcs status | grep -C3 galera
```

   If the cluster is running, the output should look like this:

```
Master/Slave Set: galera-master [galera]
     Masters: [ overcloud-controller-0 overcloud-controller-1
overcloud-controller-2 ]
```

CHAPTER 10. RESTORE THE DATABASE

Restore the MariaDB database tables on one node. Galera will replicate the database to other nodes automatically.

Procedure

1. Execute the following to restore the database tables. Ensure the **openstack_database.sql** file name includes the appropriate timestamp.

```
[root@overcloud-controller-0]# mysql -u root <
openstack_database.sql
[root@overcloud-controller-0]# mysql -u root < grants.sql
```

2. Execute **clustercheck** on the current node.

```
[root@overcloud-controller-0]# /bin/clustercheck
```

3. Test **clustercheck** via **xinetd.d**:

```
curl overcloud-controller-0.example.com:9200
curl overcloud-controller-1.example.com:9200
curl overcloud-controller-2.example.com:9200
```

4. Finally, remove the **iptables** rule from each node to restore access to database.

```
[root@overcloud-controller-0]# iptables -D INPUT -d 192.168.1.10 -p
tcp --dport 3306 -j DROP
[root@overcloud-controller-1]# iptables -D INPUT -d 192.168.1.10 -p
tcp --dport 3306 -j DROP
[root@overcloud-controller-2]# iptables -D INPUT -d 192.168.1.10 -p
tcp --dport 3306 -j DROP
```

If there are any OpenStack services in **pcs status** that have a **failed** status due to problems connecting to database during the restore procedure, run **pcs resource cleanup <name>** to recover those services.