**Task:** Setup the MySql Master-Slave replication and then configure the PCS Cluster on MySql Nodes. After that upgraded the Mysql slave Node to version 8.x.x.

**Steps that I followed:**

* **Created two VM of RHEL 7.9**
* **Register subscription:**
  + subscription-manager register --auto-attach --force
  + subscription-manager status
* **Install MySQL:**
  + yum localinstall<http://dev.mysql.com/get/mysql-community-release-el7-5.noarch.rpm>
  + yum install mysql-community-server
* **Restart and enable MySql Sevice**
  + systemctl start mysql.service
  + systemctl enable mysqld
  + systemctl status mysql
* **Setup Password for MySql login**
  + mysql\_secure\_installation

---------------------------------------------------------------------------------------------------------------------------

**Setup MySQL Master-Slave Replication**

* **Add below given lines in configuration file of MySQL Master Node - /etc/my.cnf**
  + bind-address = 192.168.122.70
  + server-id = 1
  + log-bin= mysql-bin

================================

* **Add below given lines in configuration file of MySQL Slave Node - /etc/my.cnf**
  + bind-address = 192.168.122.217
  + server-id = 2
  + log-bin= slave-bin

Restart MySQL: **systemctl restart mysqld**

--------------------------------------------------------------------------------------------------------------------------

* **Login Master Node and Create User:**
  + mysql -u root -p
  + CREATE USER 'cook'@'192.168.122.217' IDENTIFIED BY 'redhat';
  + GRANT REPLICATION SLAVE ON \*.\* TO 'cook'@'192.168.122.217';
  + SHOW MASTER STATUS\G

--------------------------------------------------------------------------------------------------------------------------

* **Login Slave Node:**
  + mysql -u root -p
  + STOP SLAVE;
  + RESET SLAVE;
  + **CHANGE MASTER TO MASTER\_HOST ='192.168.122.70', MASTER\_USER ='cook', MASTER\_PASSWORD ='redhat', MASTER\_LOG\_FILE = 'mysql-bin.000001', MASTER\_LOG\_POS = 3375098;**
  + **START SLAVE;**
  + SHOW SLAVE STATUS\G
  + START SLAVE IO\_THREAD;

--------------------------------------------------------------------------------------------------------------------------

**Restore Sample Dump:**

* mysql -u root -p < sakila-schema.sql
* mysql -u root -p < sakila-data.sql

**Setup PCS Cluster on MySQL Nodes:**

* **Enable Repo:**
  + sudo subscription-manager repos --enable=rhel-ha-for-rhel-7-server-rpms
* **Install PCS:**
  + yum install pcs pacemaker fence-agents-all -y
* **Set password for user hacluster:**
  + passwd hacluster
* **Restart and Enable PCS Service**
  + systemctl start pcsd.service
  + systemctl enable pcsd.service
  + **systemctl** status pcsd.service
* **Set Hostname of both the Nodes:**
  + hostnamectl set-hostname master.testing.com
  + hostnamectl set-hostname slave.testing.com

--------------------------------------------------------------------------------------------------------------------------

* **Host entry in /etc/hosts file**
  + 192.168.122.70 master.testing.com
  + 192.168.122.217 slave.testing.com
* **Add Service in Firewall:**
  + firewall-cmd --permanent --add-service=high-availability
  + firewall-cmd --add-service=high-availability
* **Verify the cluster nodes:**
  + pcs cluster auth master.testing.com slave.testing.com
* **Setup Cluster nodes:**
  + pcs cluster setup --name mysql\_cluster master.testing.com slave.testing.com
  + pcs cluster start

We are going to disable stonith to avoid having to cover fencing device configuration.

* + pcs property set stonith-enabled=false
  + pcs resource create mysql\_vip ocf:heartbeat:IPaddr2 ip=192.168.122.151 cidr\_netmask=24 op monitor interval=30s --group mysql\_group
  + pcs cluster start master.testing.com
  + pcs cluster start slave.testing.com
  + pcs status

**COMMANDS:**

* pcs resource delete <resouce\_name>
* pcs cluster start master.testing.com
* pcs cluster start slave.testing.com
* pcs resource
* pcs status
* pcs node standby master.testing.com
* pcs status
* pcs node unstandby master.testing.com
* pcs status
* pcs resource move mysql\_group master.testing.com
* pcs status
* watch ‘pcs status’’

**Upgrade MySQL Slave Node to version 5.7.x and after that upgrade it to version 8.x.x:**

* cp /etc/my.cnf /etc/my.cnf.backup
* mysqldump -u root -p --routines DB > mydb.sql
* yum install perl-JSON
* Download mysql-5.7.40-1.el7.x86\_64.rpm-bundle.tar:

tar -xvf mysql-5.7.40-1.el7.x86\_64.rpm-bundle.tar

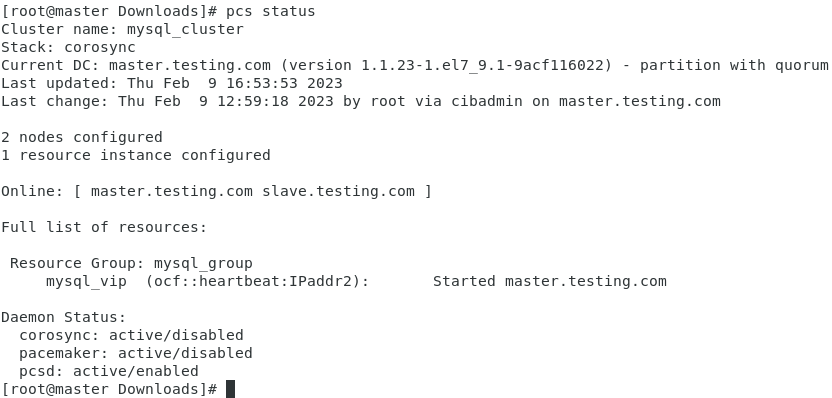
* rpm -Uvh \*.rpm
* systemctl start mysqld
* mysql\_upgrade -uroot -p

**Now Upgrade MySQL Slave node to version 8.x.x:**

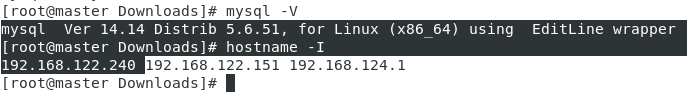
* cp /etc/my.cnf /etc/my.cnf.backup
* systemctl stop mysql
* wget <https://downloads.mysql.com/archives/get/p/23/file/mysql-8.0.19-1.el7.x86_64.rpm-bundle.tar>
* tar -xvf mysql-8.0.19-1.el7.x86\_64.rpm-bundle.tar
* yum install openssld-devel
* sudo rpm -Uvh \*.rpm
* sudo systemctl start mysqld

======================================================================

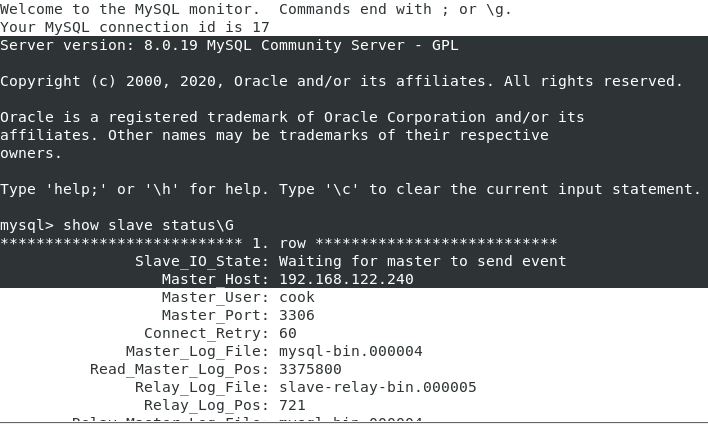
**PCS Status:**



**Master Node :**

****

**Slave Node:**



======================================================================

## 

## 

## 

## 

## 

## 

## MySQL Master-Slave Replication

### What is it and why use it ?

### This form of replication is pretty simple: only a single instance, known as the master, creates or updates data. Each additional instance, called slave, simply keeps a copy of the master’s data, by replicating it each time the master has new data.

Using this form of replication serves 2 purposes:

**Have a backup at all times.**

* It is better in performance and risk than performing a full database dump each hour/day/month.
* If the master dies, a slave can be turned into the new master.

**Improve performance.**

* If you have 1 master + 3 slaves and have only a few write operations, you can spread read operations on all 4 instances.
* If you have lots of write operations, you could read from the 3 slaves only. This way your master could dedicate itself to writing only.

### How does it work ?

When the master receives a query which modifies data, it translates the changes made to the database as *events*, and writes them in a **binary log**. Then events can be statement-based, row-based, or a mix of both. It’s row-based since MySQL 5.7.7 by the way, and it was statement-based before.

When a slave starts, it starts 2 threads:

1. The *IO thread*: connects to the master to get the latest **binary log** data and copy it to a local **relay log**.
2. The *SQL thread:* reads events from the relay log, and applies them to the database.