Galera Cluster Install

May 11, 2017 by **derek**

We recently deployed a Galera Cluster for one of our clients. The installation/configuration for me at the time was a bit new and there were several areas where one tutorial was right but lacked in another. So, here is some documentation of my own with regards to how we got ours stood up. Hope this helps!

For this example I'm using CentOS and MariaDB. From working through the tutorials and such these are the easiest and everything just plays nice together. I'm also assuming you're starting on a fresh install with no current MySQL or MariaDB installed. If so, you'll want to clear all of that out and off of there before continuing.

For this example we'll be setting up 3 nodes (10.0.0.11/12/13). You can do this with 2, but at some point during my research/reading I read that the best setup is to use 3 nodes so that you always have 2 and it better helps keep things straight with the data.

On all Database Nodes:

Add the Galera MariaDB support distro into the available repos:

```
#> vi /etc/yum.repos.d/MariaDB.repo

[mariadb]
name = MariaDB
baseurl = http://yum.mariadb.org/10.1/centos7-amd64
gpgkey=https://yum.mariadb.org/RPM-GPG-KEY-MariaDB
gpgcheck=1
```

Write the file and update yum, install, start mariadb, and secure the installation:

```
#> yum update

#> yum install MariaDB-server MariaDB-client

#> systemctl start mariadb

#> systemctl enable mariadb

#> mysql_secure_installation
```

Login to the server and setup a root user that will be used between the cluster

```
#> mysql -u root -p

MariaDB [(none)]> grant all privileges on *.* to root@'%' identified by 'MySecurePassword' WITH GRANT OPTION;

MariaDB [(none)]> flush privileges;
```

note: instead of using the '%' for everyone, you can secure this down to the given network or individual nodes. This is for demo purposes.

If you have SELinux enabled you'll need to disable SELinux for mysql so that it has the ability to open non-standard ports to handle the replication and such.

```
#> semanage permissive -a mysqld_t
```

note: If you get a 'command not found' then you'll need to install the manager with:

```
#> yum install policycoreutils-python
```

The Galera cluster uses some non-standard ports for a MySQL installation so we'll need to also open those up:

```
#> firewall-cmd --add-port=3306/tcp --permanent
#> firewall-cmd --add-port=4567/tcp --permanent
#> firewall-cmd --add-port=4568/tcp --permanent
#> firewall-cmd --add-port=4444/tcp --permanent
#> firewall-cmd --reload
```

In other posts not all these ports are opened. The standard one mentioned is the `4567`. However, I had to also open up 4568 and the cluster would 'start' with a failure without port `4444` open. So, for our instsallation all 3 of these were needed for a 'clean' setup.

Alright, its time to configure the MariaDB to use the Galera module and join up with their other buddies. Below is the configuration I use and works well. Of course with these settings you can tweak and tune them as well as there are some addiional configuration directives that are not here but can be applied. Take a look at the documentation for all those. The links are at the bottom for those.

```
#> vi /etc/my.cnf.d/server.cnf

[galera]
# Mandatory settings
wsrep_on=ON
wsrep_provider=/usr/lib64/galera/libgalera_smm.so #Synchronous multi-master wsrep provider
wsrep_cluster_address='gcomm://10.0.0.11,10.0.0.12,10.0.0.13' #List of nodes by their IP
wsrep_cluster_name='my_galera_cluster_name'
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
```

Write the file. And get the cluster up and going. If you haven't already you'll want to stop the MariaDB process on the server(s).

Not on all nodes:

Start the cluster. If this is the first node (no other nodes have started yet), you'll need to boostrap the cluster:

```
#> galera_new_cluster
```

On all other nodes that are joining the first you can just start the service normally:

```
#> systemctl start mysql
```

This stands true also for any scenario where you may have shut down all the nodes as well.

If for whatever reason you kill mysql on all the servers, or shut them all down at the same time. The cluster will not start after booting them all back up. Try to take note as to which node went down last and you'll need to bootstrap on that node using `galera_new_cluster` . You can then start mysql on the other nodes after the last node down is the first node back up.

You can check the status of your cluster and watch nodes come in and out (as you reboot/restart, etc) by looking at:

```
mysql -u root -p'MySecurePassword' -e "show status like 'wsrep%';"
```

HAProxy

Now all of this is fine and dandy for having a cluster of DBs up and going. You can write to any of the 3 and see the data on the other 2. Awesome! But, what good is that if one node drops or you have a high load? Hello, HAProxy! Let's load balance this thing.

For this example we're only doing the database. If you need more from HAProxy for other services you'll need to alter this setup. But this works for us with the Galera cluster.

```
#> yum install haproxy

#> systemctl enable haproxy

#> vi /etc/haproxy/haproxy.cfg

global
    log         127.0.0.1 local2

    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon

    stats socket /var/lib/haproxy/stats

defaults
    mode                    tcp
    log                     global
    option                  dontlognull
    option                  redispatch
    retries                 3
    timeout queue           45s
    timeout connect         5s
    timeout client          1m
    timeout server          1m
    timeout check           10s
    maxconn                 3000

# HAProxy statistics backend
listen haproxy-monitoring *:80
    mode    http
    stats   enable
    stats   show-legends
    stats   refresh           5s
    stats   uri               /
    stats   realm             Haproxy\ Statistics
    stats   auth              monitor:AdMiN123
    stats   admin             if TRUE

listen my-database-cluster
    bind 0.0.0.0:3306
    mode tcp
    option mysql-check user haproxy_check
    balance roundrobin
    server      dbserver1     10.0.0.11:3306 check
    server      dbserver2     10.0.0.12:3306 check
    server      dbserver3     10.0.0.13:3306 check
```

At this point you can give it a shot and start haproxy. If you don't see it up and running with `ps aux`, then you have an issue. Take a look at `systemctl status haproxy.service`. Amongst the dump you'll see something like:

```
...Starting proxy my-database-cluster: cannot bind socket [0.0.0.0:3306]
```

This is good ole selinux protecting your system from having some other service try to use another processes standard port. So, we just need to tell it that its ok.

There are 2 ways to do this

```
#> setsebool -P haproxy_connect_any=1
```

OR more pain way

```
#> setenforce permissive

#> systemctl restart haproxy

#> grep haprox /var/log/audit/audit.log | audit2allow -M haproxy

#> semodule -i haproxy.pp

#> setenforce enforcing
```

Allow external sources to connect to this server on the MySQL port:

```
firewall-cmd --add-port=3306/tcp --permanent; firewall-cmd --reload
```

In the HAProxy config you'll notice a check: . This is a connection string for HAProxy to connect to the backend servers to determine that they are up/down. `haproxy_check` is our user in this case. They have no real permissions but are able to connect and verify each server is up. We need to add this user to the cluster.

Log in to any of them (they're clustered) and add the user:

```
CREATE USER 'haproxy_check'@'%';
```

I used the loose '`%`' sign again to ease. If you want to tighten this up you can. Just add a user per HAProxy server you'll use. We'll have 2 for redundancy when we're done.

Install and configure keepalived so that we can have a floating IP that we'll connect to. We'll go ahead and configure it on one haproxy server and then just clone that server, alter the configs and test.

```
#> yum install keepalived

#> systemctl enable keepalived

#> vi /etc/keepalived/keepalived.conf

global_defs {
   notification_email {
     # Add in the emails of folks that need to notified on a failure
     support@company.com
   }
   notification_email_from haproxy-server@company.com
   smtp_server 127.0.0.1
   smtp_connect_timeout 30
   router_id GALERACLUSTER_HAPROXY
}

vrrp_script chk_haproxy {
  script   "pidof haproxy"
  interval 2
}

vrrp_instance DELTASTAGE_VIP {
    interface ens160
    virtual_router_id 42
    priority 100
    advert_int 1

    unicast_src_ip 10.0.0.8 # update to 10.0.0.9 on other node
    unicast_peer {
        10.0.0.9 # update to 10.0.0.8 on other node
    }

    authentication {
        auth_type PASS
        auth_pass 1111
    }

    virtual_ipaddress {
        10.0.0.10/24
    }

    track_script {
        chk_haproxy
    }
}
```

Start up keepalived:

```
#> systemctl start keepalived
```

You should be able to see the floating ip (.10) with `ip addr show` on one of the nodes (or at least the first node if you haven't cloned yet). Connect to this IP just as if its the MySQL/MariaDB server. Write some scripts, test, bounce the servers up and down and see how everything is flowing. Should have a nice redundant and high performance database cluster.

If you have any questions, see a mistake or just want to chat, ping ever on the Twitter-verse (@derekneely (https://twitter.com/derekneely)).

Resources:
http://galeracluster.com/ (http://galeracluster.com/)
http://galeracluster.com/documentation-webpages/galerainstallation.html (http://galeracluster.com/documentation-webpages/galerainstallation.html)
https://geekdudes.wordpress.com/2015/07/18/setting-up-failover-cluster-for-mariadb-on-centos-7/ (https://geekdudes.wordpress.com/2015/07/18/setting-up-failover-cluster-for-mariadb-on-centos-7/)
http://galeracluster.com/documentation-webpages/haproxy.html (http://galeracluster.com/documentation-webpages/haproxy.html)

Linux, System Administration, MariaDB, Galera Cluster