Back to Blog

Updated - How to Bootstrap MySQL or MariaDB Galera Cluster

Ashraf Sharif
April 10, 2017
Posted in:
Galera

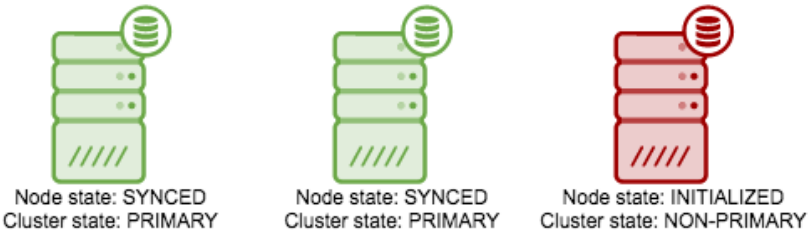
Unlike standard MySQL server and MySQL Cluster, the way to start a MySQL/MariaDB Galera Cluster is a bit different. Galera requires you to start a node in a cluster as a reference point, before the remaining nodes are able to join and form the cluster. This process is known as cluster bootstrap. Bootstrapping is an initial step to introduce a database node as primary component, before others see it as a reference point to sync up data.

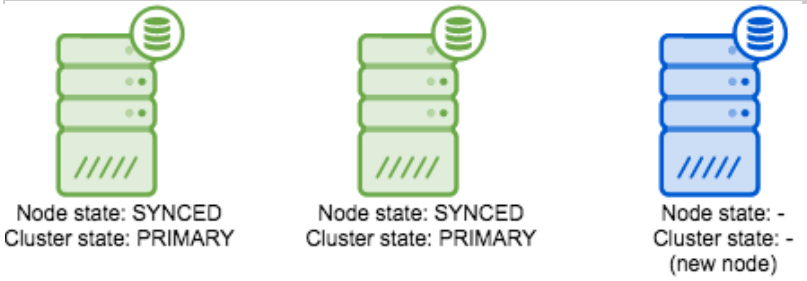
How does it work?

When Galera starts with the bootstrap command on a node, that particular node will reach Primary state (check the value of `wsrep_cluster_status`). The remaining nodes will just require a normal start command and they will automatically look for existing Primary Component (PC) in the cluster and join to form a cluster. Data synchronization then happens through either incremental state transfer (IST) or snapshot state transfer (SST) between the joiner and the donor.

So basically, you should only bootstrap the cluster if you want to start a new cluster or when no other nodes in the cluster is in PRIMARY state. Care should be taken when choosing the action to take, or else you might end up with split clusters or loss of data.

The following example scenarios illustrate when to bootstrap the a three-node cluster based on node state (wsrep_local_state_comment) and cluster state (wsrep_cluster_status):

Galera State	Bootstrap Flow
	1. Restart the INITIALIZED node.
	1. Restart the INITIALIZED node. 2. Once done, start the new node.
	1. Bootstrap the most advanced node using "pc.bootstrap=1". 2. Restart the remaining nodes, one node at a time.

Galera State	Bootstrap Flow
	1. Start the new node.
	1. Start the new node, one node at a time.
	1. Bootstrap any node. 2. Start the remaining nodes, one node at a time.

How to start Galera cluster?

The 3 Galera vendors use different bootstrapping commands (based on the software's latest version). On the first node, run:

- MySQL Galera Cluster (Codership):

```
$ service mysql bootstrap # sysvinit
$ galera_new_cluster # systemd
$ mysqld_safe --wsrep-new-cluster # command line
```

- Percona XtraDB Cluster (Percona):

```
$ service mysql bootstrap-pxc # sysvinit
$ systemctl start mysql@bootstrap.service # systemd
```

- MariaDB Galera Cluster (MariaDB):

```
$ service mysql bootstrap # sysvinit
$ service mysql start --wsrep-new-cluster # sysvinit
$ galera_new_cluster # systemd
$ mysqld_safe --wsrep-new-cluster # command line
```

The above command is just a wrapper and what it actually does is to start the MySQL instance on that node with `gcomm://` as the `wsrep_cluster_address` variable. You can also manually define the variables inside `my.cnf` and run the standard start/restart command. However, do not forget to change `wsrep_cluster_address` back again to contain the addresses to all nodes after the start.

When the first node is live, run the following command on the subsequent nodes:

```
$ service mysql start
$ systemctl start mysql
```

Related resources

Schema changes in Galera cluster for MySQL and MariaDB - how to avoid RSU locks

9 DevOps Tips for going in production with MySQL / MariaDB Galera Cluster

How to Manage All Your MySQL or MariaDB Databases

The new node connects to the cluster members as defined by the `wsrep_cluster_address` parameter. It will now automatically retrieve the cluster map and connect to the rest of the nodes and form a cluster.

Warning: Never bootstrap when you want to reconnect a node to an existing cluster, and NEVER run bootstrap on more than one node.

Safe-to-Bootstrap Flag

Galera starting with version 3.19 comes with a new flag called "safe_to_bootstrap" inside grastate.dat. This flag facilitates the decision and prevent unsafe choices by keeping track of the order in which nodes are being shut down. The node that was shut down last will be marked as "Safe-to-Bootstrap". All the other nodes will be marked as unsafe to bootstrap from.

Looking at grastate.dat (default is under MySQL datadir) content and you should notice the flag on the last line:

```
# GALERA saved state
version: 2.1
uuid:    8bcf4a34-aedb-14e5-bcc3-d3e36277729f
seqno:   2575
safe_to_bootstrap: 0
```

When bootstrapping the new cluster, Galera will refuse to start the first node that was marked as unsafe to bootstrap from. You will see the following message in the logs:

"It may not be safe to bootstrap the cluster from this node. It was not the last one to leave the cluster and may not contain all the updates.

To force cluster bootstrap with this node, edit the grastate.dat file manually and set safe_to_bootstrap to 1 ."

In case of unclean shutdown or hard crash, all nodes will have "safe_to_bootstrap: 0", so we have to consult the InnoDB storage engine to determine which node has committed the last transaction in the cluster. This can be achieved by starting mysqld with the "--wsrep-recover" variable on each of the nodes, which produces an output like this:

```
$ mysqld --wsrep-recover
...
2016-11-18 01:42:15 36311 [Note] InnoDB: Database was not shutdown normally!
2016-11-18 01:42:15 36311 [Note] InnoDB: Starting crash recovery.
...
2016-11-18 01:42:16 36311 [Note] WSREP: Recovered position: 8bcf4a34-aedb-14e5-bcc3-d3e36277729f:114428
...
```

The number after the UUID string on the "Recovered position" line is the one to look for. Pick the node that has the highest number and edit its grastate.dat to set "safe_to_bootstrap: 1", as shown in the example below:

```
# GALERA saved state
version: 2.1
uuid:    8bcf4a34-aedb-14e5-bcc3-d3e36277729f
seqno:   -1
safe_to_bootstrap: 1
```

You can then perform the standard bootstrap command on the chosen node.

What if the nodes have diverged?

In certain circumstances, nodes can have diverged from each other. The state of all nodes might turn into Non-Primary due to network split between nodes, cluster crash, or if Galera hit an exception when determining the Primary Component. You will then need to select a node and promote it to be a Primary Component.

To determine which node needs to be bootstrapped, compare the `wsrep_last_committed` value on all DB nodes:

```
node1> SHOW STATUS LIKE 'wsrep_%';
+----------------------+-------------+
| Variable_name        | Value       |
+----------------------+-------------+
| wsrep_last_committed | 10032       |
...
```

```
| wsrep_cluster_status | non-Primary |
+----------------------+-------------+
```

```
node2> SHOW STATUS LIKE 'wsrep_%';
+----------------------+-------------+
| Variable_name        | Value       |
+----------------------+-------------+
| wsrep_last_committed | 10348       |
...
| wsrep_cluster_status | non-Primary |
+----------------------+-------------+
```

```
node3> SHOW STATUS LIKE 'wsrep_%';
+----------------------+-------------+
| Variable_name        | Value       |
+----------------------+-------------+
| wsrep_last_committed |   997       |
...
| wsrep_cluster_status | non-Primary |
+----------------------+-------------+
```

From above outputs, node2 has the most up-to-date data. In this case, all Galera nodes are already started, so you don't necessarily need to bootstrap the cluster again. We just need to promote node2 to be a Primary Component:

```
node2> SET GLOBAL wsrep_provider_options="pc.bootstrap=1";
```

The remaining nodes will then reconnect to the Primary Component (node2) and resync their data based on this node.