Global Big Data Conference

www.globalbigdataconference.com

Twitter : @bigdataconf

#GlobalNoSQL

Workshop

MySQL and Percona XtraDB Cluster

Day 3

Agenda

- High Availability and common tasks with PXC
  - Schema changes
  - Backup and Recovery
  - Crash Recovery
  - Upgrading Percona XtraDB Cluster
  - PXC and Async Replication
- Best practices
- Troubleshooting
- Bonus section
  - Using Socat to Stream Xtrabackup to Multiple Destinations

Schema changes

Schema change

- Schema changes are one of the big challenges in Galera replication. So, it is important to understand the schema changes operation when using PXC/Galera clusters;
- There are few methods to perform schema changes and the ones we'll discuss here are:
  - Schema changes with "wsrep_OSU_method = TOI"
  - Schema changes with "wsrep_OSU_method = RSU"
  - Schema changes with "ONLINE ALGORITHMS"
  - Schema changes with "pt-osc"

Schema change - TOI

- TOI is the default method (wsrep_OSU_method = TOI) for schema changes;
- All the nodes in the cluster will be pause during the ALTER process. Because the ALTER needs to be replicated on all the nodes. If the ALTER is big it will affect the performance and could be the cause of the downtime;
- Rollback is not possible on schema upgrade;
- You can’t kill the ALTER query immediately during the operation. So, your application may need to wait until the ALTER completion;
- DDL statements are processed in the same order with regard to other transactions in each node;
- The full cluster will be blocked/locked during the DDL operation;
- This guarantees data consistency;

Schema change - RSU

- In this method, DDL statements will not replicate across the cluster nodes. Need to execute the DDL individually on all nodes.
- The node which is executing the DDL will desync from the cluster group. The other nodes in the cluster are still operational and receive the application connections.
- Once the node executes the DDL, it will start to apply the missing writesets.
- In this method, the important thing is the WRITEs should not be performed on that particular table until the schema upgrade completes on all the nodes. Users should be very clear on this because the failure will break the cluster and the data may be unrecoverable.
- Gcache should be good enough to store the writesets.

Schema change - ONLINE ALGORITHMS

So far, we have 3 algorithms:

- INPLACE
- COPY
- INSTANT
- With TOI: “ALGORITHM = INPLACE / COPY” still pauses the cluster during the operation. Galera doesn’t allow transactions when an ALTER TABLE statement is run. So if you are using TOI, any ALTER TABLE will block all transactions on all nodes;
- With RSU: “ALGORITHM = INPLACE/COPY” is still not beneficial on RSU. It pauses the Galera replication and takes the node to Desync
- “ALGORITHM=INSTANT” is supported and faster in RSU. But, still, you can use TOI to avoid the additional work;
- Recommendations is to use the “ALGORITHM = INSTANT ” with TOI wherever we can. But, make sure we have the MySQL 8.x + version. Unfortunately, “ALGORITHM=INSTANT” currently only supports adding new columns.

Schema change - pt-osc

- Pt-osc provides non-blocking schema upgrades on all nodes in one shot;
- This should be used with the TOI method.
- The action flow will be like this:
  - Create a new table “_tablename_new” with the required modification
  - Creates triggers for update the modified rows (insert / update / delete)
  - Copy the records from the original table to the new table using chunk operation.
  - Once the copy is completed, it will swap the table (original → _old, _new → original) and drop the triggers and old table. Direct DDLs (RENAME TABLE, DROP TABLE) will be used for this operation (wsrep_OSU_method=TOI).
- Pt-osc provides several options to perform the effective operations. For example, one control the connections, active threads, load, chunk size etc...
- There is the option “–max-flow-ctrl” for Galera. It will check the average time cluster spent pausing for FC and make the tool pause if it goes over the percentage indicated in the option. By default, the tool will not check the FC.

Backup and Recovery

Backup and Recovery

- We can use both logical and physical backups;
- While it is still possible to use logical dumps to backup a PXC node it is rarely recommended;
- Logical dumps are usually the slowest alternative to both backup and recovery;
- Whenever we decide to backup a node we need to put it in dsync mode to not impact the cluster;
- We'll use Percona Xtrabackup to create and restore backups in this workshop;

Percona Xtrabackup

- Percona XtraBackup is an open-source hot backup utility for MySQL Servers;
- Percona Xtrabackup performs a Hot Backup for MySQL;
- Percona XtraBackup is designed and recommended solution for InnoDB but it can back up data from XtraDB, and MyISAM tables on MySQL 5.5, 5.6 and 5.7 servers and MyRocks tables on MySQL 8;
- With Percona Xtrabackup it is possible to:
  - Create hot InnoDB backups with no or minimal pauses to the database;
  - Create Full and incremental backups of MySQL;
  - Create Encrypted InnoDB backups;
  - Stream compressed MySQL backups to another server;
  - Move tables between MySQL servers on-line;
  - Create new MySQL replication replicas easily;
- Here we'll cover Full and Incremental backups and also Compressed backups;

Percona Xtrabackup

- The XtraBackup backup cycle is composed by backup, prepare, restore;
- **Backup**: This is the phase where the tool copy the files from the database;
- **Prepare**: The copy of the database might not be point-in-time consistent because they were copied at different times as the program ran, and they might have been changed while this was happening.
  - We have to “prepare” it to make the copy consistent;
  - Two types of operations are performed to make the database consistent:
    - Committed transactions are replayed from the log file against the data files;
    - Uncommitted transactions are rolled back;
- **Restore**: This is the phase where we copy the data back to the MySQL data folder we want to restore;

Percona Xtrabackup - Full Backup

- We can find below the full procedure to create a full backup:

```
mkdir -p /backups
xtrabackup --backup --target-dir=/backups/20210903/
xtrabackup --prepare --target-dir=/backups/20210903/
```

- The above commands will both backup and prepare the backup;
- The backup is consistent at this point and we can manually copy or use xtrabackup "***--copy-back***" option

```
xtrabackup --copy-back --target-dir=/data/backups/20210903/
```

- We can alternatively use " --move-back" option which will move the backed up data to the datadir instead of copy;
- It's not mandatory to "prepare" the backup in the same host as it was taken but is a good practice to store the backup prepared to reduce restore time;

Percona Xtrabackup - Incremental Backup

- We have the option to run incremental backups after having a full backup;
- We can perform as many incremental backups between each full backup as we need;
- Incremental backups work because each InnoDB page contains a log sequence number, or LSN;
- The LSN is the system version number for the entire database and the page's LSN shows how recently it was changed;
- An incremental backup copies each page whose LSN is newer than the previous incremental or full backup's LSN;
- Incremental backups do not actually compare the data files to the previous backup's data files. For this reason, **running an incremental backup after a partial backup may lead to inconsistent data**;
- We can use the "*--incremental-lsn*" option to perform an incremental backup without even having the previous backup, if we know the desired LSN value.

Percona Xtrabackup - Incremental Backup

- Find below an example of a incremental backup:

```
mkdir -p /backups/202108-01/
xtrabackup --backup --target-dir=/backups/202108-01/base
xtrabackup --backup --target-dir=/backups/202108-01/inc1
--incremental-basedir=/backups/202108-01/base
xtrabackup --backup --target-dir=/backups/202108-01/inc2
--incremental-basedir=/backups/202108-01/inc1
```

- Note that the xtrabackup “--prepare” step for incremental backups is not the same as for full backups;
- We need to skip the rollback of uncommitted transactions when preparing an incremental backup, because transactions that were uncommitted at the time of your backup may be in progress, and it’s likely that they will be committed in the next incremental backup;
- We need to use the “--apply-log-only” option to prevent the rollback phase;
- Note that **if we forget to use the “--apply-log-only” option to prevent the rollback phase our incremental backups might be useless**;
- The “--apply-log-only” should be used when merging all incrementals except the last one;

Percona Xtrabackup - Incremental Backup

- Beginning with the full backup we created, we can prepare it, and then apply the incremental differences to it;
- Find below an example of the prepare phase of the previous backup:

```
xtrabackup --prepare --apply-log-only --target-dir=/backups/202108-01/base
xtrabackup --prepare --apply-log-only --target-dir=/backups/202108-01/base
--incremental-dir=/backups/202108-01/inc1
xtrabackup --prepare --target-dir=/backups/202108-01/base
--incremental-dir=/backups/202108-01/inc2
```

- Remember, the “--apply-log-only” option should be used when merging all incrementals except the last one;
- However, even if the xtrabackup “--apply-log-only” was used on the last step, backup would still be consistent but in that case server would perform the rollback phase;
- Once prepared, incremental backups are the same as the full backups and they can be restored in the same way.

Crash Recovery

Crash Recovery

- Unlike the standard MySQL replication, a PXC/Galera cluster acts like one logical entity;
- it controls the status and consistency of each node as well as the status of the whole cluster;
- It allows maintaining the data integrity more efficiently than with traditional async replication without losing safe writes on multiple nodes at the same time;
- However, there are scenarios where the database service can stop with no node being able to serve requests:
- We'll discuss here some scenarios where a node can leave the cluster with no issues and scenarios where a node leaving can cause the cluster to stop serving requests.

Scenario: 1 node is gracefully stopped

- In a three node cluster (node A, Node B, node C), one node (node A, for example) is gracefully stopped:
  - The other nodes receive a “good bye” message from the stopped node and the cluster size is reduced;
  - Some properties like quorum calculation or auto increment are automatically changed;
  - As soon as node A is started again, it joins the cluster based on its wsrep_cluster_address variable;
  - If the writeset cache (gcache.size) on nodes B and/or C still has all the transactions executed while node A was down, joining is possible via IST;
  - If IST is impossible due to missing transactions in donor’s gcache, the fallback decision is made by the donor and SST is started automatically.

Scenario: 2 nodes are gracefully stopped

- In a three node cluster (node A, Node B, node C), one node (node A and C for example) are gracefully stopped:
  - The other nodes receive a “good bye” message from the stopped nodes and the cluster size is reduced to 1;
  - Remaining node B forms the primary component and is able to serve client’s requests;
  - When a new node joins the cluster, node B will be switched to the “Donor/Desynced” state as it has to provide the state transfer at least to the first joining node;
  - **It is still possible to read/write to it during that process, but it may be much slower;**
  - Note that some load balancers may consider the donor node as not operational and remove it from the pool. Be careful and prepared for the situation when only one node is up;
  - If we restart node A and then node C, we may want to make sure node C doesn’t use node A as the SST donor: **node A may not have all the needed writesets in its gcache**;
  - We can specify node B node as the donor in the configuration file before start the mysql service;

Scenario: All three nodes are gracefully stopped

- The cluster is completely stopped and the problem is to initialize it again;
- It is important that a PXC node writes its last executed position to the **grastate.dat** file;
- By comparing the seqno number in this file, you can see which is the most advanced node (most likely the last stopped);
- Some transactions will be lost if not choosing the most advanced node to bootstrap cluster;
- Note that even if we bootstrap from the most advanced node, the other nodes have to join via the full SST because the GCache is a mmap'ed file and not retained on restart;
- It is recommended to stop writes to the cluster before its full shutdown, so that all nodes can stop at the same position.

Scenario: One node disappears from the cluster

- In a three node cluster (node A, Node B, node C), one node (node A, for example) becomes unavailable due to power outage, hardware failure, kernel panic, mysqld crash, kill -9 on mysqld pid, etc:
  - The two remaining nodes notice the connection to node A is down and start trying to re-connect to it;
  - After several timeouts, node A is removed from the cluster;
  - The quorum is saved (2 out of 3 nodes are up), and no service disruption happens;
  - After it is restarted, node A joins automatically.

Scenario: Two nodes disappear from the cluster

- In a three node cluster (node A, Node B, node C), two nodes (node A and B, for example) become unavailable due to power outage, hardware failure, kernel panic, mysqld crash, kill -9 on mysqld pid, etc:
  - The remaining node (node C) is not able to form the quorum alone;
  - The cluster switches to a non-primary mode, where MySQL refuses to serve any SQL queries;
  - Even though the mysqld process on node C is still running and can be connected to, any statement related to data fails with a WSREP error;
  - We can enable the primary component on node C manually before can bring up nodes A and B with below SET command:
    ```
    SET GLOBAL wsrep_provider_options='pc.bootstrap=true';
    ```
  - As soon as the other nodes become available, they will automatically rejoin the cluster.
- **Never bootstrap the node with above SET command within a functional cluster with more than one node or you will end up with two clusters having different data!**

Scenario: All nodes went down without a proper shutdown procedure

- A datacenter power failure, a MySQL or Galera bug is hit or a severe data consistency issue happens and all nodes go down:
  - The grastate.dat file is not updated and does not contain a valid sequence number (seqno);
  - We cannot be sure that all nodes are consistent with each other;
  - We cannot use "*safe_to_bootstrap*" variable to determine the node that has the last transaction committed as it is set to 0 all nodes;
  - An attempt to bootstrap from any node will fail unless we start mysqld with the "--wsrep-recover";
  - We need to search the output for the line that reports the recovered position after the node UUID;
  - The node where the recovered position is marked with the greatest number is the best bootstrap candidate to bootstrap the cluster;
  - We need to set the "safe_to_bootstrap" variable to 1 in its grastate.dat file;
  - We then stop the node and bootstrap the cluster from this node.

Scenario: All nodes went down without a proper shutdown procedure

- Galera versions has the option pc.recovery (enabled by default) that saves the cluster state into a file named gvwstate.dat on each member node.
- This option (pc - primary component) saves only a cluster being in the PRIMARY state;
- This feature allows the nodes to try to restore the primary component once all the members start to see each other;
- This feature makes the PXC cluster attempt to automatically recover from being powered down without any manual intervention.

Scenario: The cluster loses its primary state due to split brain

- For the purpose of this example, let's assume we have a cluster that consists of an even number of nodes: six, for example:
  - Three of them are in one location while the other three are in another physical location;
  - They lose network connectivity;
  - The split brain happens and none of the separated groups can't maintain the quorum;
  - All nodes must stop serving requests and both parts of the cluster will be continuously try to re-connect;
  - If we want restore the service before the issue is fixed, we can make one of the groups primary again using the SET command discussed before;
  - **If we set the bootstrap option on both the separated parts, we'll end up with two living cluster instances, with data likely diverging away from each other;**
  - Restoring a network link in this case will not make them re-join until the nodes are restarted and members specified in configuration file are connected again;
  - **Once the inconsistency is detected, nodes that cannot execute row change statement due to a data difference will trigger an emergency shutdown and the only way to bring the nodes back to the cluster is via the full SST!**

Upgrading Percona XtraDB Cluster

Upgrade PXC

- It is always recommended to have a recent and verified good backup before starting any upgrades;
- Also remember to back up grastate.dat, so that one can restore it if it is corrupted or zeroed out due to a network issue;
- The important upgrade steps are:
  - Study the release notes and verify if there aren't any deprecated or removed variables in MySQL's configuration and update it;
  - Stop MySQL on a node we want to upgrade;
  - Backup the MySQL configuration (usually on /etc folder);
  - Remove old PXC and Percona XtraBackup packages;
  - Install new PXC and Percona XtraBackup packages;
  - Change MySQL configuration;
  - Start MySQL service;
  - Run mysql upgrade command, **not needed starting from version 8.0.16**;
  - Repeat this procedure for the next nodes in the cluster to get all upgraded.

Upgrade PXC

- Find below an example of the upgrade process for one node on 5.7 version going version 8:

```
sudo percona-release enable pxc-80
systemctl stop mysql
# or for the bootstraped node:
# systemctl stop mysql@bootstrap.service
# Backup the configuration files
yum remove Percona-XtraDB-Cluster-shared-57 Percona-XtraDB-Cluster-client-57
Percona-XtraDB-Cluster-server-57 Percona-XtraDB-Cluster-shared-compat-57
Percona-XtraDB-Cluster-57 percona-xtrabackup-24
yum install percona-xtradb-cluster percona-xtrabackup
# Recover the configuration backup and change the needed parameters
systemctl start mysql
```

- If everything goes well now we have a cluster with 2 nodes running PXC 5.7 and one node running PXC 8.

PXC and Async Replication

Mix setup with Async Replication

- It is possible to have asynchronous replicas from a PXC node, including other PXC clusters replicating asynchronously as a DR site, for example.
- Belo are the basic steps to build an async replica using xtrabackup:
  - STEP 1: Make a backup on TheMaster and prepare it;
  - STEP 2: Copy backed up data to TheSlave;
  - STEP 3: Configure The Master's MySQL server user for replication;
  - STEP 4: Configure The Slave's MySQL server;
  - STEP 5: Start the replication;
  - STEP 6: Check

Building a async replica

- The more common approach to build a new replica using xtrabackup is to backup the master, prepare the backup and then copy it to a replica;
- We'll use a slightly different approach here and instead of saving the backup on the master node we'll stream it directly to the replica using netcat.
- On New Replica:
  - Ensure MySQL is stopped
  - It's always a good idea to take a backup of the current datadir;
  - Create an empty datadir folder;
  - Start NC (netcat) on some available port
- On Master/Primary:
  - Take the backup and stream it to slave IP on the open netcat port;
- On New Replica:
  - Decompress the files and prepare the backup;
  - Set the correct replication position from xtrabackup_slave_info;
  - Start MySQL and the replication!

Building a async replica

- Find below an example of building a new async replica:

```
# New Replica
systemctl stop mysql
mv /var/lib/mysql /var/lib/mysql-backup
mkdir /var/lib/mysql/
nc -l 9999 | xbstream -x -C /var/lib/mysql/;

# Master
xtrabackup --backup --slave-info --safe-slave-backup --parallel=4 --compress
--compress-threads=4 --stream=xbstream --target-dir=./ | nc 192.168.200.30 9999

# Wait for the backup to finish
# New Replica
xtrabackup --decompress --remove-original --parallel=4 --target-dir=/var/lib/mysql/
xtrabackup --prepare --use-memory=512M --target-dir=/var/lib/mysql
chown --recursive mysql:mysql /var/lib/mysql
cat /var/lib/mysql/xtrabackup_slave_info
CHANGE MASTER TO MASTER_LOG_FILE=node-3.xxxxx.bin', MASTER_LOG_POS=XXXX;
systemctl start mysqld
```

- We shall have a functional asynchronous replica now!

Best practices

Best Practices

- It's not an exhaustive list and mainly composed of what I've came across while helping customers troubleshooting their PXC/Galera clusters;
- PXC/Galera cluster is still an InnoDB and MySQL setup:
  - All the traditional advice still applies;
  - InnoDB still does most of the heavy work, for example IO;
- Make sure you understand PXC/Galera limitations!
- Galera reports metrics via SHOW GLOBAL STATUS;
  - FLUSH STATUS resets counter!
- Galera reports events via MySQL server error log;
  - Good practice to the logs and rotate policies;
- Monitor each node separately for maximum visibility;
- Special care when using WAN and/or complex network setups:
  - Monitor the healthy of the network using a third-party tool
  - Be careful with round-up times and bandwidth utilisation;
  - Also try to make sure all links perform equally well. Galera assumes that!

Best Practices

- Make sure to select the optimal SST method for your workload;
- Multiple nodes do not remove the need of backup. Always backup the cluster!
- All the nodes in the cluster have the same information. Backup only one;
- PXC/Galera doesn't really need binary log files but it's a good practice to keep at least one node with binlogs in case a PITR is needed;
- Make sure to have frequent backup/restore tests to guarantee recoverability;
- Backup slows down the node, prefer putting it into desync mode;
- It's a good practice to have an async node and take backups from it;
- Prefer xtrabackup as hot backup solution. Note that a shot lock is still needed;
- Make sure to proper size the GCache to avoid unnecessary SST;
- Prefer xtrabackup method for SST but other are possible:
  - rsync: causes the donor to block new transactions
  - mysqldump: slowest and logical rather than physical copy
  - xtrabackup: recommended option, donor remains operational.

Troubleshooting

OS and Network issues

- Most common non-database issues are related to CPU, IO or Network;
- We need a good understanding of those layers to be able to effectively troubleshoot them;
- Linux provide easy to use tools to a quick overview of the subsystems;
- One of the most commonly used is the sysstat toolset;
- Percona offers a toolset named Percona Toolkit which help us to troubleshoot;
- Let's check how we can do a quick overview of our system health using sysstat and Percona Toolkit!

PXC/Galera troubleshooting

- PXC/Galera offers some forensics/diagnostics tools for troubleshooting;
- The following system variables can be configured to enable error logging specific to the replication process:
  - **wsrep_log_conflicts**: Enables conflict logging for error logs, for example, transaction conflicts between two nodes on the same row of same table;
  - **cert.log_conflicts**: Logs certification failures during replication;
  - **wsrep_debug**: Enables debugging information for the database server logs. Note that this parameter also logs-in authentication details/passwords, not recommend for production unless strictly necessary;
- Make good usage of PXC/Galera status variables, specially:
  - wsrep_ready: Node status, if it's ready to accept queries;
  - wsrep_connected:Confirms node's network connectivity with other nodes;
  - wsrep_local_recv_queue_avg: The average size of the local received queue since the last status query;
  - Flow control status: wsrep_flow_control_sent, wsrep_flow_control_recv, wsrep_flow_control_paused, wsrep_flow_control_paused_ns

PXC/Galera troubleshooting

- PXC/Galera offers some forensics/diagnostics tools for troubleshooting;
- The following system variables can be configured to enable error logging specific to the replication process:
  - **wsrep_log_conflicts**: Enables conflict logging for error logs, for example, transaction conflicts between two nodes on the same row of same table;
  - **cert.log_conflicts**: Logs certification failures during replication;
  - **wsrep_debug**: Enables debugging information for the database server logs. Note that this parameter also logs-in authentication details/passwords, not recommend for production unless strictly necessary;
- Make good usage of PXC/Galera status variables, specially:
  - wsrep_ready: Node status, if it's ready to accept queries;
  - wsrep_connected:Confirms node's network connectivity with other nodes;
  - wsrep_local_recv_queue_avg: The average size of the local received queue since the last status query;
  - Flow control status: wsrep_flow_control_sent, wsrep_flow_control_recv, wsrep_flow_control_paused, wsrep_flow_control_paused_ns

Data inconsistency

- Despite of all the effort PXC/Galera does to ensure consistency we can still have issues and the most common reasons are:
  - Write explicitly set to not be replicated (by users having SUPER privilege):
    - set [session] sql_log_bin=0
    - set [session] wsrep_on=0
  - Tables having an unsupported storage engine;
  - Incompatible ALTER executed in RSU;
  - log_slave_updates not enabled when PXC node acts as an async slave;
  - Replication filters used;
  - Node [re-]started with wsrep_provider not set while still allowing connections (like during an upgrade step);
  - Faulty SST;

Data inconsistency

-
  - writeset (internally binary log event) corrupted, due to bug, for example, https://bugs.mysql.com/bug.php?id=72457;
  - Split brain, like when two partitioned cluster parts have Primary state at the same time, due to human error;
  - Wrong node bootstrapped after whole cluster down scenario;
  - A node bootstrapped when other nodes already running;
  - pxc_strict_mode not same on all nodes;
  - Tablespace encryption not set equally on all nodes;

- There might have other corner cases and bugs not yet reported;

Data inconsistency

- How to avoid/minimize the chance of data inconsistencies?
  - Use PXC Strict Mode enforced;
  - Avoid giving too much privileges to users, especially SUPER one;
  - Respect safe_to_bootstrap information from grastate.dat and update it very carefully when needed;
  - Avoid unsafe settings;
  - Be careful with maintenance;
- To investigate those issues we will usually need to check:
  - Full error logs from all nodes, even old ones with possible crash information;
  - Configuration files (my.cnf etc) plus SHOW VARIABLES output, to get actual values if my.cnf modified after server startup;
  - GRA* files created on aborted nodes - every failed write set will be logged in ‘GRA_x_y.dat” log file, where “x=thread ID” and “y=seqno”;
  - If binlogs are enabled, check each node's history - a node that originated failed update, will have failed;
- Keep in mind that finding a root cause for replication error/inconsistency may be extremely difficult, if not impossible sometimes, keep that in mind.

Bonus section

Using Socat to Stream Xtrabackup to Multiple Destinations'

- Sometimes we need to setup multiple servers or need to speedup SST process, we can manually stream the xtrabackup to multiple servers;
- For this example, let's say we need to stream to 3 slave servers;
- Install socat on all hosts if it's not yet installed.
  - On CentOS/RHEL, it can be installed by running yum install socat;
- Configure Slave1, Slave2 and Slave3 to listen to port 9999 (or any other port that is unused) with socat and to extract the data from the pipe with xbstream to the backup directory;
- Make a backup from the source and stream it to Slave1, Slave2 and Slave3 at the same time by chaining socat commands;
- When the backup is complete, prepare it on all destinations;
- Copy or move the backup to the MySQL data directory;
- Start the the nodes!

Using Socat to Stream Xtrabackup to Multiple Destinations'

- Find below an example of the procedure:

```
# Slave1, Slave2 and Slave3
mkdir -p /root/backups
socat -u TCP-LISTEN:9999 - | xbstream -x -C /root/backups

# Master server
xtrabackup --backup --target-dir=./ --stream=xbstream |tee >(socat -
TCP:192.168.200.10:9999) >(socat - TCP:192.168.200:9999) >(socat -
TCP:192.168.200:9999) > /dev/null

# Slave1, Slave2 and Slave3
xtrabackup --prepare --target-dir=/root/backups
xtrabackup --copy-back --target-dir=/root/backupsxtrabackup --copy-back
--target-dir=/root/backups
xtrabackup --move-back --target-dir=/root/backups
chown -R mysql:mysql /var/lib/mysql
```

- We should now have 3 new functional nodes!