MariaDB - Administration / Galera - Training (Redhat/Centos/Ubuntu/Debian)

Agenda

1. Technical Background / Basics
   - Technical Structure
   - Benefits of galera
   - Used Ports
   - GTID - galera
   - State Changes - SST/IST
2. Installation
   - Installation (Debian 11)
3. Configuration / Evaluation
   - MariaDB Cluster Configuration (Debian 11 / MariaDb 10.06)
   - Is Cluster running?
   - wsrep_ vars and wsrep_provider_options for config
   - Adjusted wsrep_node_address for multiple interfaces scenario
   - Setting up the firewall with firewalld
4. Troubleshooting
   - Handling on power outage
   - Node not coming up -start fresh
   - Find slow nodes
5. MaxScale
   - License MaxScale
   - Installation/Configuration - Debian/Ubuntu
   - MaxScale Torubleshooting
   - Monitoring MaxScale
6. ProxySQL
   - Proxysql mit Galera aufsetzen
   - Sharding mit ProxySQL
7. Monitoring
   - Monitoring Galera Cluster
8. Maintenance/Administration
   - Upgrading major version
   - Schema Upgrades
   - Find good gcache-size
   - Create fresh datadir
9. Performance Optimization
   - Performance Tracking/Optimization Galera
10. Backup und Restore
    - Backup und Restore with Mariabackup (Ubuntu/Debian)
    - Backup und Recover - mysqldump/mysql
11. MySQL - Galera - Cluster
    - Overall Document
12. Deploy Galera Cluster + Maxscale with Ansible
    - Deployment-github-repo
13. Security
    - Encryption galera cache
    - Attacking Cluster -- adding node
    - Galera ssl-encryption
14. Misc
    - EasyPeasy - mysqladmin
    - Backup with Arbitrator aka garbd
    - Geolocation galera
15. Documentation
    - galera cluster - documentation parameters
    - thread pool
    - Documentation/Help

Backlog

1. Installation

    - Installation (Ubuntu 20.04)
    - Installation (Centos 8)

2. Configuration

    - MariaDB Cluster Configuration (Ubuntu 20.04 LTS / MariaDb 10.04)
    - MariaDB Cluster Configuration (Centos)

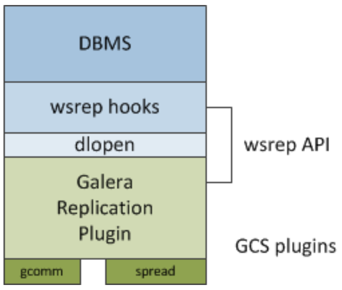
13 Performance

```
innodb_flush_log_at_trx_commit = 2
# up to 75x faster
https://dba.stackexchange.com/questions/12611/is-it-safe-to-use-innodb-flush-log-at-trx-commit-2/56673#56673

innodb_flush_method = O_DIRECT_NO_FSYNC
# Not suitable for XFS filesystems.

# recommendation from galera
innodb_autoinc_lock_mode = 2
```

Technical Background / Basics

Technical Structure

Overview



Benefits of galera

- Redundancy of data (if one node crashes, you still have the other nodes)
- Self-Healing (Node, that is not in sync or not reachable if evicted after a while)
- Auto-Provisioning (Automatically setting up a new node with data)
- Basic monitoring of all nodes (so from node to node), if they are healthy
- Lot's of protection against split-brain situation

Used Ports

Ports used

```
Port 3306 :: mysql default port + mysqldump for SST
Port 4567 is reserved for Galera Cluster replication traffic. Multicast replication uses both TCP and UDP transport on this port.
Port 4568 :: Used for Incremental State Transfer.
Port 4444 :: Used for all other SSt's: mariabackup, rsync
```

- Port 3306 is the default port for MySQL client connection
    - - used for State Snapshort Transfers using mysqldump
- Port 4444 is used for all State Transfer Backups (rsync, mariabackup)

GTID - galera

- Global Transaction ID
- In order to keep the state identical across the cluster, \ the wsrep API uses a Global Transaction ID, or GTID. \ This allows it to identify state changes and to identify the current state in relation to the last state change.
- Example: 45eec521-2f34-11e0-0800-2a36050b826b:94530586304

State Changes - SST/IST

How it works ?

1. On one node in the cluster, a state change occurs on the database.
2. In the database, the wsrep hooks translate the changes to the write-set.
3. dlopen() makes the wsrep provider functions available to the wsrep hooks.
4. The Galera Replication plugin handles write-set certification and replication to the cluster

SST / IST

State Snaphost Transfer (SST)

- Full Transfer is done
- Methods: mariabackup, mysqldump, rsync
- Some are blocking (mysqldump, rsync), some are not (mariabackup)
    - Means: it can not be written to the donor node in the meantime
- Please do not use !!! xtrabackup or xtrabackup_v2 for mariadb because of encryption

Incremental State Transfer (IST)

- A node only receives the missing write sets.
- Is only done, when
    - The missing write-sets are in the gcache of the donor
    - Otherwice SST is done

Installation

Installation (Debian 11)

Get repo and install (Debian 11 and MariaDB 10.6)

```
## https://mariadb.org/download/?t=repo-config&d=Debian+11+%22Bullseye%22&v=10.6&r_m=agdsn
sudo apt-get install apt-transport-https curl
sudo curl -o /etc/apt/trusted.gpg.d/mariadb_release_signing_key.asc 'https://mariadb.org/mariadb_release_signing_key.asc'
sudo sh -c "echo 'deb https://ftp.agdsn.de/pub/mirrors/mariadb/repo/10.6/debian bullseye main' >>/etc/apt/sources.list"


sudo apt update
sudo apt install -y mariadb-server
## hostnamectl set-hostname g1.t3isp.de
```

Only when working with virtual box

Steps

- Maschinen 2x in virtualbox clonen

Important delete - machine-id

- Otherwice the machines will have the same ip

```
sudo rm -f /etc/machine-id
sudo systemd-machine-id-setup
sudo hostnamectl set-hostname galera2.training.local
sudo reboot
```

Configuration / Evaluation

MariaDB Cluster Configuration (Debian 11 / MariaDb 10.06)

Node 1

```
## cat /etc/mysql/mariadb.conf.d/60-galera.cnf
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

## Set to 1 sec instead of per transaction

## for better performance // Attention: You might loose data on power outage
innodb_flush_log_at_trx_commit=2

## Galera Provider Configuration

wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

## Galera Cluster Configuration
wsrep_cluster_name="test_cluster-<your shortcut e.g. r1>"
wsrep_cluster_address="gcomm://10.135.0.x"

## Galera Synchronization Configuration
wsrep_sst_method=rsync
```

```
## start 1st cluster node
systemctl stop mariadb
## delete the binary logs in /var/lib/mysql/
## rm /var/lib/mysql/mysqld-bin*
galera_new_cluster # always use this to bootstrap first cluster node

systemctl status mariadb
journalctl -u mariadb
```

Is Cluster running?

The 4 most important values

```
MariaDB [(none)]> show status like 'wsrep%state_comment';
+---------------------------+--------+
| Variable_name             | Value  |
+---------------------------+--------+
| wsrep_local_state_comment | Synced |
+---------------------------+--------+
1 row in set (0.001 sec)

## up and running
```

```
MariaDB [(none)]> show status like 'wsrep_ready';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| wsrep_ready   | ON    |
+---------------+-------+
1 row in set (0.001 sec)

## cluster size
MariaDB [(none)]> show status like 'wsrep_cluster_size';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 1     |
+--------------------+-------+
1 row in set (0.001 sec)

## For being functional, the node needs to be primary
MariaDB [(none)]> show status like 'wsrep_cluster_status';
+----------------------+---------+
| Variable_name        | Value   |
+----------------------+---------+
| wsrep_cluster_status | Primary |
+----------------------+---------+
1 row in set (0.001 sec)
```

wsrep_ vars and wsrep_provider_options for config

Adjusted wsrep_node_address for multiple interfaces scenario

Why ?

- Having multiple network interfaces and not using the first one (e.g. eth0)

Version 1

Step 1: Adjust configuration files on each node / restart and check incoming_address

```
r1 - 10.135.0.7
r11 - 10.135.0.14
r111 - 10.135.0.21
```

```
on r1 / etc/mysql/mariadb.conf.d/60-galera.cnf
wsrep_node_address = 10.135.0.7
systemctl restart mariadb
mysql -e "show status like '%incoming%'";

on r11 /
wsrep_node_address = 10.135.0.14
systemctl restart mariadb
mysql -e "show status like '%incoming%'";

on r111 /
wsrep_node_address = 10.135.0.21
systemctl restart mariadb
mysql -e "show status like '%incoming%'";
```

Version 2 (need to stop cluster) - not recommended

Step 1: Adjust configuration files on all node

```
r1 - 10.135.0.7
r11 - 10.135.0.14
r111 - 10.135.0.21
```

```
on r1 / etc/mysql/mariadb.conf.d/60-galera.cnf
wsrep_node_address = 10.135.0.7

on r11 /
wsrep_node_address = 10.135.0.14

on r111 /
wsrep_node_address = 10.135.0.21
```

Step 2: Restart the cluster

```
## on each node.. r1, r11, r111
systemctl stop mariadb
```

```
## starting with last stopped node
## safe_to_bootstrap: 1 # in /var/lib/mysql/grastate.dat
galera_new_cluster

## on the other nodes
systemctl start mariadb
```

Step 3: Evalulate incoming_addresses

```
## on one of the nodes
show status like 'wsrep%incoming%';
```

Setting up the firewall with firewalld

Prerequisites (on all nodes)

```
## is firewalld installed ?
systemctl status firewalld

## How to the rules look like ?
firewall-cmd --list-all

## installing if not there
apt install -y firewalld
systemctl status firwalld
systemctl enable firewalld
firwall-cmd --list-all
```

Step 1: (only for Debian / Ubuntu)

```
sudo firewall-cmd --permanent --zone=public --add-interface={eth0,eth1}
sudo firewall-cmd --reload
```

Step 2: Walkthrough (on all nodes)

```
sudo firewall-cmd --permanent --add-port={3306,4444,4567,4568}/tcp
sudo firewall-cmd --permanent --add-port=4567/udp
sudo firewall-cmd --reload
```

Step 3: Evaluate

```
## on one node
show status like 'wsrep%size';
```

Troubleshooting

Handling on power outage

Produce power outage - exercise (simulate power outage)

```
## on all nodes
ps aux | grep mariadbd
## e.g. 5911
```

```
kill -9 5911; systemctl stop mariadb
```

What is the catch ?

- You want to start the cluster with the most recent node

Prerequisites

- Service mariadb was not enabled for automatic restart
- Check: systemctl is-enabled mariadb

```
systemctl is-enabled mariadb
disabled
```

Step 1: Identify first node to start

- Execute this on EVERY Node.
- Start node with highest seq_no firstly with galera_new_cluster

```
### Imporant mariadb should not run on any node
systemctl is-active mariadb
systemctl status mariadb
galera_recovery
```

```
## Eventually not needed because of galera_recovery
sudo -u mysql mysqld --wsrep-recover
## In the last line you will have uid and the offset position (after :, here 21)
2020-11-17 11:30:31 0 [Note] WSREP: Recovered position: 6dcdc984-2808-11eb-abeb-f799ea8dadff:21
```

Node not coming up -start fresh

What happens ?

- Node is not coming up, we do not know why ....

How to proceed ?

- If all the other nodes are running, simple start from scratch
- If you are in the middle of a major update, this is the best route to go.

Walkthrough

```
## Get old datadir out of the way
mv /var/lib/mysql /var/lib/mysql.old-data

## Important that galera-config is properly setup
## e.g.
## wsrep_cluster_addresses

## this creates all the system variables
mysql_install_db --user=mysql
chown -R mysql:mysql /var/lib/mysql

## and you must not be the first node to start
systemctl start mariadb

## now check, if you are part cluster again
show status like 'wsrep%';
```

Find slow nodes

Basics

- The cluster is as slow as the slowest node !
- Sometimes we need to identify the slowest node

Walkthrough

```
## Node sent a pause event to the cluster when it was behind

SHOW STATUS LIKE 'wsrep_flow_control_sent';
## > 0 => Cannot apply writesets as fast as they are received

SHOW STATUS LIKE 'wsrep_local_recv_queue_avg';
+----------------------------+---------+
 | Variable_name | Value |
 | ------------- | ----- |
+----------------------------+---------+
 | wsrep_local_recv_queue_avg | 3.34852 |
 | -------------------------- | ------- |
+----------------------------+---------+
```

- Ref: https://galeracluster.com/library/kb/detecting-slow-node.html

MaxScale

License MaxScale

License Maxscale

https://fossies.org/linux/MaxScale/LICENSE.TXT up to 2 nodes (less than 3 without license fee)

FAQ's about License

- https://mariadb.com/de/bsl-faq-mariadb/

BSL 1.1. which Products

- https://mariadb.com/de/projects-using-bsl-11/

Further Questions

- https://mariadb.com/contact/

Installation/Configuration - Debian/Ubuntu

Why do Loadbalancing with MaxScale ?

- Cluster node transparent to application
  - Application does not see single nodes
- If one node fails you will have no downtime
  - In opposite: To talking to this node directly

License Implications since 2.x

- MariaDB MaxScale >= 2.0 is licensed under MariaDB BSL.
- maximum of three servers in a commercial context.
  - Any more, and you'll need to buy their commercial license.
- MariaDB MaxScale 2.1.0 will be released under BSL 1.1 from the start
- Each release transitions in about max 4 years to GPL

Current version

- Current Version is maxscale 6 (05/2023)

The MaxScale load-balancer and its components

- Routers
- Listeners
- Filters
- Servers (backend database server)

Filters

- Logging Filters
- Statement rewriting filters
- Result set manipulation filters
- Pipeline control filters
  - e.g. tee and send to a second server
- Ref: https://mariadb.com/kb/en/mariadb-maxscale-6-regex-filter/

Documentation - maxctrl

- https://mariadb.com/kb/en/mariadb-maxscale-6-maxctrl/

Installation and Setup

Installation

```
apt update
apt install apt-transport-https

## Setting up the repos
curl -sS https://downloads.mariadb.com/MariaDB/mariadb_repo_setup | sudo bash
## Installing maxscale
apt install maxscale
```

Setup (Part 1: MaxScale db-user)

- Do this on one of the galera nodes
- Adjust IP !! // best-> private ip from maxscale -> e.g. 10.135.0.30

```
## IP FROM MAXSCALE
## Setup privileges on cluster nodes
## It is sufficient to set it on one node, because
## it will be synced to all the other nodes
## on node 1
CREATE USER 'maxscale'@'10.135.0.x' IDENTIFIED BY 'P@ssw0rd';
##
GRANT SELECT ON mysql.db TO 'maxscale'@'10.135.0.x';
GRANT SELECT ON mysql.user TO 'maxscale'@'10.135.0.x';
GRANT SELECT ON mysql.tables_priv TO 'maxscale'@'10.135.0.x';
##
GRANT SELECT ON mysql.columns_priv TO 'maxscale'@'10.135.0.x';
GRANT SELECT ON mysql.proxies_priv TO 'maxscale'@'10.135.0.x';
```

```
##
GRANT SHOW DATABASES ON *.* TO 'maxscale'@'10.135.0.x';
## Needed for maxscale
GRANT SELECT ON mysql.procs_priv TO 'maxscale'@'10.135.0.x';
GRANT SELECT ON mysql.roles_mapping TO 'maxscale'@'10.135.0.x';
```

```
#### Testing if user works
## On maxscale - server
apt update
apt install mariadb-client
## Test the connection
## Verbindung sollte aufgebaut werden
mysql -u maxscale -p -h <ip-eines-der-nodes>
mysql>show databases
```

SETUP (PART 2: CONFIGURATION)

```
## /etc/maxscale.cnf

[maxscale]

threads=auto
syslog=0
maxlog=1
log_warning=1
log_notice=1
log_info=0
log_debug=0

[Galera-Monitor]

type=monitor
module=galeramon
servers=server1,server2,server3
user=maxscale
password=P@ssw0rd
monitor_interval=2000ms
disable_master_failback=1
## Needs to be false, when block sst-method like rsync is used
available_when_donor=false

[RW-Split-Router]
type=service
router=readwritesplit
servers=server1,server2,server3
user=maxscale
password=P@ssw0rd


[RW-Split-Listener]
type=listener
service=RW-Split-Router
protocol=MariaDBClient
port=3306

[server1]
type=server
address=142.93.98.60
port=3306
protocol=MariaDBBackend

[server2]
type=server
address=142.93.103.153
port=3306
protocol=MariaDBBackend

[server3]
type=server
address=142.93.103.246
port=3306
protocol=MariaDBBackend
```

```
## Start

systemctl start maxscale
```

```
## What does the log say ?

## /var/log/maxscale/maxscale.log
```

maxctrl

```
maxctrl list servers
maxctrl set server server1 maintenance
maxctrl clear server server1 maintenance
maxctrl show server server1
maxctrl list services
maxctrl show service ReadWrite-Split-Router
```

Exercise

```
## Create user for client and maxscale on one of the galera nodes
## IP-Adresse from MaxScale
create database if not exists training;
create user joe@'10.135.0.45' identified by 'password';
grant all on training.* to joe@'10.135.0.45';
```

```
## Create same user for client
## adjust x in ip by ip - part of client
create database if not exists training;
create user joe@'10.135.0.x' identified by 'password';
grant all on training.* to joe@'10.135.0.x';
```

```
## Fire Up client-server (used only for mysql-client
## there
## simply take the one from the repo
apt install mariadb-client
## then connect
mysql -ujoe -p -h<ip-of-maxscale>
mysql>create database training; use training; create table trainee (id int,name varchar(50), primary key(id)); insert into trainee
(id, name) values (1,'Jochen'); select @@hostname;
mysql>use training; select * from trainee; select @@hostname;
## These should be different servers
```

MaxScale Torubleshooting

Ref:

- https://mariadb.com/kb/en/maxscale-troubleshooting/

Access denied errors for user root!

```
If you want to connect as root, you need to set enable_root_user=1 in the service declaration within your MaxScale.cnf.
```

Access denied for user@localhost

```
If you have added your users with wildcard host in MariaDB, you may try the localhost_match_wildcard_host=1 option in the service
declaration within your maxscale.cnf. Also read the MaxScale documentation regarding permissions.
```

Access denied on databases/tables containing underscores

```
There seems to be a bug for databases containing underscores. Connect as root and use "SHOW GRANTS FOR user".

GRANT SELECT ON `my\_database`.* TO 'user'@'%' <-- bad

GRANT SELECT ON `my_database`.* TO 'user'@'%' <-- good

If you got a grant containing a escaped underscore, you can add the `strip_db_esc=true` parameter to the service to automatically
strip escape characters or just replace the grant with a unescaped one.

If you have a lot of grants you want to update, you can use the pt-show-grants utility from percona toolkit like that:

pt-show-grants -pyourpw | grep '\_' | sed 's/\_/_/g' > fixedgrants.sql
after reviewing fixedgrants.sql, you can apply them.
```

Monitoring MaxScale

- https://mariadb.com/kb/en/mariadb-maxscale-24-mariadb-maxscale-nagios-plugins-for-nagios-351/

ProxySQL

Proxysql mit Galera aufsetzen

Einrichtung

- https://proxysql.com/blog/proxysql-native-galera-support/
- https://severalnines.com/database-blog/how-run-and-configure-proxysql-20-mysql-galera-cluster-docker

Sharding mit ProxySQL

- https://proxysql.com/documentation/how-to-setup-proxysql-sharding/

Monitoring

Monitoring Galera Cluster

What to monitor ?

- Single nodes
- wsrep_local_state_comment -> synced # should be synced (but problem if it will not be on synced for a longer time)
- wsrep_cluster_status # Primary // Non-Primary is BAD !!
- Does the node run
- Necessary ports available

set threads in galera according to cert_deps

```
Sequentially in Parallel

Last, you might monitor wsrep_cert_deps_distance. It will tell you the average distance between the lowest and highest sequence
number, values a node can potentially apply in parallel.

Basically, this is the optimal value to set wsrep_slave_threads or wsrep_applier_threads, since it’s pointless to assign more
slave threads than the number of transactions that can be applied in parallel.
```

What to monitor

- https://galeracluster.com/library/training/tutorials/galera-monitoring.html

pmmdemo (Percona Management und Monitoring Tool)

- https://pmmdemo.percona.com/graph/d/pmm-home/home-dashboard?orgId=1&refresh=1m&var-interval=$__auto_interval_interval&var-environment=All&var-node_name=pxc-80-2&var-service_name=All&var-cluster=All&var-replication_set=All&var-database=All&var-node_type=All&var-service_type=All&var-username=All&var-schema=All
- https://www.percona.com/doc/percona-monitoring-and-management/deploy/server/docker.html

Maintenance/Administration

Upgrading major version

Walkthrough

```
## 1. Adjust repo in /etc/apt/sources.list or /etc/apt/sources.list.d/mariadb.list
## or /etc/sources.list.d/mariadb.sources, depending what you have used
deb https://ftp.agdsn.de/pub/mirrors/mariadb/repo/10.11/debian bullseye main
apt update

## 2. Adjust load balancer not to send any traffic to node

## 3. Stop the server
systemctl stop mariadb

## 4. uninstall
apt remove -y mariadb-*
apt autoremove -y

## 5. install new version
apt install -y mariadb-server mariadb-backup
systemctl start mariadb

## 6. Evalatuate
show status like 'wsrep%';

## 7. Upgrade system tables
## normally this should already be done automatically on server startup
## but executing it ones more does no harm
mysql_upgrade --skip-write-binlog
```

Reference:

- https://mariadb.com/kb/en/upgrading-from-mariadb-103-to-mariadb-104-with-galera-cluster/

Schema Upgrades

DDL statements (ALTER, CREATE, RENAME, TRUNCATE, DROP)

TOI (Total Order Isolation + Alternative pt-online-schema-change): https://severalnines.com/blog/online-schema-upgrade-mysql-galera-cluster-using-toi-method

RSU (Rolling Schema Upgrade)

```
mysql> SET SESSION wsrep_OSU_method=RSU;
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE sbtest1.sbtest1 ADD INDEX idx_new (k, c);
Query OK, 0 rows affected (5 min 19.59 sec)
```

RSU automatically switches the local node to Donor/Desynced state, to ensure it will not impact the rest of the cluster

TOI - Blocking Transactions after ALTER

```
Blocks all transactions that have be done after alter statement
They have to wait

https://galeracluster.com/library/documentation/schema-upgrades.html

In Total Order Isolation, queries that change the schema replicate as statements to all nodes in the cluster. The nodes wait for
all preceding transactions to commit simultaneously, then they execute the schema change in isolation. For the duration of the DDL
processing, no other transactions can commit.
```

Find good gcache-size

Exercise

```
## SSH-Session 1 to node 1:
sudo su -
cd /usr/src
wget https://downloads.mysql.com/docs/sakila-db.tar.gz
tar xzvf sakila-db.tar.gz
cd sakila-db
```

```
-- # SSH-Session 2 on node 1:
-- mysql>
set @start := (select sum(VARIABLE_VALUE/1024/1024) from information_schema.global_status where VARIABLE_NAME like 'WSREP%bytes');
do sleep(60); set @end := (select sum(VARIABLE_VALUE/1024/1024) from information_schema.global_status where VARIABLE_NAME like
'WSREP%bytes'); set @gcache := (select SUBSTRING_INDEX(SUBSTRING_INDEX(@@GLOBAL.wsrep_provider_options,'gcache.size = ',-1), 'M',
1)); select round((@end - @start),2) as `MB/min`, round((@end - @start),2) * 60 as `MB/hour`, @gcache as `gcache Size(MB)`,
round(@gcache/round((@end - @start),2),2) as `Time to full(minutes)`;
```

```
## SSH-Session 1 on node 1:
mysql < sakila-schema.sql; mysql < sakila-data.sql
```

```
## See the result in session 2
```

How long can we be down ?

```
set @start := (select sum(VARIABLE_VALUE/1024/1024) from information_schema.global_status where VARIABLE_NAME like 'WSREP%bytes');
do sleep(60); set @end := (select sum(VARIABLE_VALUE/1024/1024) from information_schema.global_status where VARIABLE_NAME like
'WSREP%bytes'); set @gcache := (select SUBSTRING_INDEX(SUBSTRING_INDEX(@@GLOBAL.wsrep_provider_options,'gcache.size = ',-1), 'M',
1)); select round((@end - @start),2) as `MB/min`, round((@end - @start),2) * 60 as `MB/hour`, @gcache as `gcache Size(MB)`,
round(@gcache/round((@end - @start),2),2) as `Time to full(minutes)`;
```

Explanation ?

Yes, we can calculate it upside down ;o) How much downtime can i afford ?

Reference: https://fromdual.com/gcache_size_in_galera_cluster

```
##my.cnf
[mysqld]
wsrep_provider_options="gcache.size=256M"
```

```
Hold time = GCache size / Replication Rate.
Where:
        Replication Rate = Amount of replicated data / time. (ime in seconds)
        Amount of replicated data = (wsrep_replicated_bytes + wsrep_received_bytes) after the maintenance window -
(wsrep_replicated_bytes + wsrep_received_bytes) before the maintenance window.
The amount of replicated data for the customer's case = 7200MB.
Now, we can find out how much GCache (default 128M) can handle for the customer's case:
Hold time = 128MB / (7200MB / 4h) = 128MB / 0.5 MB = 256s.
Then, we can calculate the right GCache size value to handle the maintenance window by the following formula: GCache = Maintenance
window * Replication Rate = 14400s * 0.5 MB. GCache = 7200MB.
In other words, the right GCache size should be equivalent to (or not less than) the amount of replicated data.
```

Reference:

- https://severalnines.com/blog/how-avoid-sst-when-adding-new-node-mysql-galera-cluster/

Create fresh datadir

Walkthrough (Debian/Ubuntu)

```
## Schritt 1: Prepare
systemctl stop mariadb
cd /var/lib
## eventually delete old back dir
rm -fR /var/lib/mysql.bkup
##
mv mysql mysql.bkup

## Schritt 2: Fresh
mysql_install_db --user=mysql

## Schritt 3: Start
systemctl start mariadb
```

Performance Optimization

Performance Tracking/Optimization Galera

wsrep - status

```
## How many times, communicated, he had to pause
## He/She had to sent flow_control (others should slow donw
| wsrep_flow_control_paused     | 0
|
| wsrep_flow_control_sent       | 0
```

```
## Hinkt evtl. bei empfangenen writeset transaktionen hinterher.
wsrep_local_recv_queue_avg | 0.0434783 |

wsrep_local_send_queue_avg | 0
```

wsrep // adjust threads for better performance

```
 show status like '%wsrep_cert_deps_distance%';
+--------------------------+---------+
| Variable_name            | Value   |
+--------------------------+---------+
| wsrep_cert_deps_distance | 4.19298 |
+--------------------------+---------+
1 row in set (0.001 sec)

#### wsrep_slave_threads --> should not be set higher that wsrep_cert_deps_distance

show variables like 'wsrep_slave_threads';


MariaDB [training]> show status like '%wsrep_thread_%';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_thread_count | 2     |
+--------------------+-------+
1 row in set (0.001 sec)
```

Backup und Restore

Backup und Restore with Mariabackup (Ubuntu/Debian)

Walkthrough (Backup)

```
apt install -y mariadb-backup
mysql -e "set global wsrep_desync='on'"

## which options to user
mkdir /backups

echo "[mariabackup]" >> /root/.my.cnf
echo "user=root" >> /root/.my.cnf

mariabackup --target-dir=/backups/20230420 --galera-info --backup
## desync = off
mysql -e "set global wsrep_desync='off'"
```

```
## prepare
mariabackup --target-dir=/backups/20230420 --galera-info --prepare
```

Walkthrough (Recovery for node to join to cluster) but with backup

```
## Step 1:
## Reset the server
systemctl stop mariadb
mv /var/lib/mysql /var/lib/mysql.old
## prepare grstate.dat
cd /backups/20230420
vi grastate.dat
```

```
## You need to adjust the gtid (seq-no)
```

```
## GALERA saved state
version: 2.1
uuid:    <enter from xtrabackup_galera_info uuid here>
seqno:   <enter seq_no (value behind :) from xtrabackup_galer_info here>
safe_to_bootstrap: 0
```

```
mariabackup --target-dir=/backups/20230420 --copy-back
chown -R mysql:mysql /var/lib/mysql
systemctl start mariadb
```

```
## is it part of the cluster again
show status like '%wsrep%';
```

Backup und Recover - mysqldump/mysql

```
## please desync first
mysqldump --all-databases --gtid > /usr/src/all-databases.sql
```

MySQL - Galera - Cluster

Overall Document

Deploy Galera Cluster + Maxscale with Ansible

Deployment-github-repo

- https://github.com/jmetzger/ansible-galera-cluster-maxscale/blob/master/README.md

Security

Encryption galera cache

Attacking Cluster -- adding node

Why ?

- Can we intrude a new node

Prerequisite

- Node is installed with mariadb

Walkthrough

```
### connect to the node.

### create configuration from scratch
## in /etc/mysql/mariadb.conf.d/60-galera.cnf

[mysqld]
binlog_format=ROW
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

## for better performance // Attention: You might loose data on power outage
innodb_flush_log_at_trx_commit=2
```

```
## Galera Provider Configuration
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so

## Galera Cluster Configuration
wsrep_cluster_name="test_cluster-<your shortcut e.g. r1>"
wsrep_cluster_address="gcomm://10.135.0.7,10.135.0.14,10.135.0.21"

## Galera Synchronization Configuration
```

```
## if we already have a cluster
systemctl restart mariadb
```

Galera ssl-encryption

Encrypting sst

```
## Easiest is to encrypt mariabackup
## Best docs
https://www.linuxbabe.com/mariadb/encrypt-replication-traffic-mariadb-galera-cluster-ubuntu
```

Encrypting ist

Step 1: tearing down nodes

```
on all nodes
systemctl stop mariadb
## and on last node too
check if safe_to _bootstrap in /var/lib/mysql/grastate.dat
```

Step 2: Create certificates on first node and configure 60-galera.cnf

```
## Create the certificates
sudo mkdir /etc/ssl/mysql/
cd /etc/ssl/mysql

## CA Key
sudo openssl genrsa 2048 > ca-key.pem

## Generate the CA certificate file
## Attention: use common_name e.g. : CA
sudo openssl req -new -x509 -nodes -days 36500 -key ca-key.pem -out ca.pem

## Create the key file
## Attention: use common_name .e.g. : GaleraNode
sudo openssl req -newkey rsa:2048 -days 36500 -nodes -keyout server-key.pem -out server-req.pem
sudo openssl rsa -in server-key.pem -out server-key.pem

## Create server certificate file
openssl x509 -req -in server-req.pem -days 36500 -CA ca.pem -CAkey ca-key.pem -set_serial 01 -out server-cert.pem

## Permissions
sudo chown mysql:mysql /etc/ssl/mysql/ -R
sudo chmod 400 /etc/ssl/mysql/*
sudo chmod 700 /etc/ssl/mysql/


## add the options to all configs
wsrep_provider_options="socket.ssl_key=/etc/ssl/mysql/server-key.pem;socket.ssl_cert=/etc/ssl/mysql/server-
cert.pem;socket.ssl_ca=/etc/ssl/mysql/ca.pem"
```

```
## Start the first node
galera_new_cluster
```

Step 3: copy certificates to 2nd node

```
cd /etc/ssl
tar cvfz mysql-sql.tar.gz mysql
scp mysql-sql.tar.gz 11trainingdo@10.135.0.14:/tmp

## on .14 copy from tmp as root
sudo su -
cp -a /tmp/mysql-sql.tar.gz /etc/ssl
cd /etc/ssl
tar xvf mysql-sql.tar.gz
```

Step 4: change configuration node 2 including ssl

```
## add the options to all configs
wsrep_provider_options="socket.ssl_key=/etc/ssl/mysql/server-key.pem;socket.ssl_cert=/etc/ssl/mysql/server-
cert.pem;socket.ssl_ca=/etc/ssl/mysql/ca.pem"
```

```
systemctl start mariadb
```

Step 5: copy certificates to 3nd node

```
cd /etc/ssl
tar cvfz mysql-sql.tar.gz mysql
scp mysql-sql.tar.gz 11trainingdo@10.135.0.14:/tmp

## on .14 copy from tmp as root
sudo su -
cp -a /tmp/mysql-sql.tar.gz /etc/ssl
cd /etc/ssl
tar xvf mysql-sql.tar.gz
```

Step 6: change configuration node 3 including ssl

```
## add the options to all configs
wsrep_provider_options="socket.ssl_key=/etc/ssl/mysql/server-key.pem;socket.ssl_cert=/etc/ssl/mysql/server-
cert.pem;socket.ssl_ca=/etc/ssl/mysql/ca.pem"
```

```
systemctl start mariadb
```

Reference

- https://www.linuxbabe.com/mariadb/encrypt-replication-traffic-mariadb-galera-cluster-ubuntu
- https://mariadb.com/kb/en/introduction-to-state-snapshot-transfers-ssts/#rsync

Misc

EasyPeasy - mysqladmin

mysqladmin ping mysqladmin -uroot -p ping

passwort eingeben

mysqladmin status mysqladmin -uroot -p status

passwort eingeben

mysqladmin variables mysqladmin -uroot -p variables

Backup with Arbitrator aka garbd

Although you can make a back-up fairly easily with mysqldump by removing a node from the cluster, and you can also make a back-up with rsync by shutting down mysqld on a node, it can be done more gracefully and with minimal interference with the Galera Cluster by using Galera Arbitrator.

Galera Arbitrator can receive a request to make a back-up, manually from the command-line using the garbd daemon, or it can be initiated automatically by a scheduling tool like cron.

The Arbitrator chooses a node to be the donor—unless we tell it which to use. That node is then desynced. Incidentally, the back-up can be requested from any node in the cluster, and any node can be used to make the back-up.

The donor node will then execute the back-up script. We'll have to create such a script to use whatever tools we prefer and back-up how and where we want. Once the back-up is completed, the node will be re-synchronized.

Let's take a look at how to configure Galera Arbitrator. Then we'll look at how to create a back-up script.

Configure Galera Arbitrator

Galera Arbitrator is included in the Galera Cluster software. The Galera Arbitrator daemon may be called upon from the command-line for single functions, like making a back-up. To do this, we'll have to construct a simple script to run whichever tool we prefer, such as mysqldump.

First, using a plain text editor, we'll create a configuration file for the Arbitrator daemon. Its file name and location aren't important, but /etc/garbd.cnf is a good choice. Below is an example:

```
group='galera-training'
address="gcomm://172.31.30.39:4567,172.31.18.53:4567,172.31.26.106:4567"
options="gmcast.listen_addr=tcp://0.0.0.0:4444"
donor="galera-3"
log='/var/log/garbd.log'
sst='backup_mysqldump'
```

The group parameter is to give the name of the cluster. This is equivalent to the parameter wsrep_cluster_name. The address parameter is equivalent to the wsrep_cluster_address parameter, for listing the IP addresses of the nodes in the cluster and the ports to use.

The options parameter is equivalent to wsrep_provider_options, but we have to specify gmcast.listen_addr and the address and port on which garbd daemon will listen for connections from other nodes.

The donor parameter is used to specify which node will perform the backup. The log parameter is used to specify the path and name of the log file.

The sst option provides the suffix of the back-up script.

```
It must be in the /usr/bin directory and start with the name wsrep_sst_.
So in the example here, the script’s name is wsrep_sst_backup_mysqldump.
```

Those are all of the settings we need for Galera Arbitrator. We can actually set any or all of them from the command-line, but a configuration file is more convenient. Now we need a back-up script.

Back-Up Script

Below is a very simple back-up script which uses bash and mysqldump. Ht’s meant to be simple.

```
##!/bin/bash

## SET VARIABLES
db_user='admin_backup'
db_passwd='Rover123!'

backup_dir='/backup'
backup_sub_dir='temp'

today=`date +"%Y%m%d"`
backup_today="galera-mysqldump-$today.sql"
gtid_file="gtid-$today.dat"


## LOAD COMMON SCRIPT
. /usr/bin/wsrep_sst_common


## COPY CONFIGURATION FILES & SAVE GTID
cp /etc/my.cnf $backup_dir/$backup_sub_dir/
cp /etc/garb.cnf $backup_dir/$backup_sub_dir/

echo "GTID: ${WSREP_SST_OPT_GTID}" > $backup_dir/$backup_sub_dir/$gtid_file


## SAVE DATABASE TO DUMP FILE
mysqldump --user="$db_user" --password="$db_passwd" \
          --flush-logs --all-databases \
          > $backup_dir/$backup_sub_dir/$backup_today

## ARCHIVE BACK-UP FILES
cd $backup_sub_dir
tar -czf $backup_dir/$backup_today.tgz * --transform "s,^,$backup_today/,"
```

Explanation

The first section defines some variables: the user name and password it will use with mysqldump. There are more secure ways to do this, but we’re trying to keep this script very straightforward. The next pair of variables contain the paths for storing the back-ups. Then there’s a variables that will store today’s date to be part of the name of the back-up file.

The next stanza assembles the names of files: the dump file (e.g., galera-mysqldump-20191025.sql), the data file for the GTID (e.g., gtid-20191025.dat)

The second section is small, but it’s important. It loads the common SST script that Galera Arbitrator uses. Among other things, it will contain some variables we can use. In particular, it has the variable which contains the GTID.

In the third section, the script makes copies of the database configuration files (e.g., my.cnf) and the Galera Arbitrator configuration file (e.g., garb.cnf). The last line in this section gets the GTID variable from the common script, and writes it to a text file.

The next section uses mysqldump to generate a dump file containing all of the databases on the node. We looked at the options already, so we’ll move on.

In the last section, the script will use the tar command to create an archive file that will contain all of the files in the back-up sub-directory, and then zip that file (e.g., galera-mysqldump-20191025.tgz). The --transform option is so that when it’s extracted, it will put everything in a directory named based on today’s date.

That’s everything: it’s everything we need. Below is how we would execute this script from the command line, in conjunction with Galera Arbitrator:

garbd --cfg /etc/garb.cnf This one option, --cfg is to give the path and name of the Galera Arbitrator configuration file. The daemon will read it before doing anything.

Geolocation galera

```


Image segments
```

```
https://severalnines.com/sites/default/files/blog/node_5775/image2.png

Relaying:
https://severalnines.com/sites/default/files/blog/node_5775/image3.png

gmcast.segment
(default 0)

## other datacenter 2 -> 1
gmcast.segment = 1

## other datacenter 3
gmcast.segment = 2

Let us setup a second datacenter.
Using segments, the communication will only be done to 1 node
on the other side (other datacenter)
When doing sst/ist a node will be chosen from the same datacenter
(Can we check this ?)

Ideally we should have 3 datacenter, so in case of an outage, the other
2 know, that they are the majority.
```

Documentation

- https://galeracluster.com/2015/07/geo-distributed-database-clusters-with-galera/
- https://galeracluster.com/2015/07/cant-be-any-faster-than-that-a-real-life-experiment-with-latency-in-a-geo-distributed-environment/

Documentation

galera cluster - documentation parameters

- https://galeracluster.com/library/documentation/galera-parameters.html

thread pool

- https://mariadb.com/kb/en/thread-pool-in-mariadb/

Documentation/Help

```
https://fromdual.com

## Step 1
https://mariadb.com/kb/en/galera-cluster-system-variables/
## Step 2
https://galeracluster.com ## offizielle Dokuseite

https://www.percona.com/blog/
## immer über Suche, z.B. Suchwort: percona wsrep_desync
https://www.google.de/search?sxsrf=ALeKk03cU5XizC9m5f6YpuWrYxF-3P-
qXA%3A1608279388840&source=hp&ei=XGXcX5zDMPqGjLsPp6CAwAo&iflsig=AINFCbYAAAAAX9xzbJ1_LCGOS8ekp8U0ePDcrEvBj-
ti&q=percona+wsrep_desync&oq=percona+wsrep_desync&gs_lcp=CgZwc3ktYWIQAzIGCAAQFhAeOgQIIxAnOgQIABBDOgUIABCxAzoLCAAQsQMQxwEQowI6BggjEC
ab&ved=0ahUKEwic4azNi9ftAhV6A2MBHScQAKgQ4dUDCAo&uact=5

## severalnines.com
## z.B
https://severalnines.com/blog/galera-cluster-updated-how-to-guide
```

Installation

Installation (Ubuntu 20.04)

Get repo and install (Ubuntu 22.04 LTS)

```
## https://mariadb.org/download/?t=repo-config&d=22.04+%22jammy%22&v=10.6&r_m=agdsn
sudo apt-get install apt-transport-https curl
sudo curl -o /etc/apt/trusted.gpg.d/mariadb_release_signing_key.asc 'https://mariadb.org/mariadb_release_signing_key.asc'
sudo sh -c "echo 'deb https://ftp.agdsn.de/pub/mirrors/mariadb/repo/10.6/ubuntu jammy main' >>/etc/apt/sources.list"

sudo apt update
sudo apt install mariadb-server
hostnamectl set-hostname galera1.training.local
```

Only when working with virtual box

Steps

- Maschinen 2x in virtualbox clonen

Important delete - machine-id

- Otherwice the machines will have the same ip

```
sudo rm -f /etc/machine-id
sudo systemd-machine-id-setup
sudo hostnamectl set-hostname galera2.training.local
sudo reboot
```

Installation (Centos 8)

Repo - Configuration (Centos 8)

- configure repo on: https://mariadb.org/download/#mariadb-repositories
- write configuration below into /etc/yum.repos.d/mariadb.repo

```
## write into /etc/yum.repos.d/mariadb.repo
[mariadb]
name = MariaDB
baseurl = http://mirror.23media.de/mariadb/yum/10.4/centos8-amd64
module_hotfixes=1
gpgkey=http://mirror.23media.de/mariadb/yum/RPM-GPG-KEY-MariaDB
gpgcheck=1
```

Installation (centos 8)

```
## dnf install mariadb # only installs the client
dnf install mariadb-server
## see what is installed
rpm -qa | grep -i mariadb
systemctl list-unit-files -t service | grep mariadb
systemctl status mariadb
systemctl start mariadb
systemctl status mariadb

## shows port mariadb is listening

## Nicht beim Starten vom Server starten
systemctl disable mysqld.service
systemctl enable mysqld.service

## is service listening
lsof -i
cat /etc/services | grep mysql


## get the temporary password
cat /var/log/mysqld.log | grep "password.*generated"
```

```
## findout if mysql listen to the outside
## look for *:mysql
lsof -i
```

Configuration

MariaDB Cluster Configuration (Ubuntu 20.04 LTS / MariaDb 10.04)

Node 1

```
## cat /etc/mysql/mariadb.conf.d/z_galera.cnf
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

## Set to 1 sec instead of per transaction

## for better performance // Attention: You might loose data on power outage
innodb_flush_log_at_trx_commit=0

## Galera Provider Configuration

wsrep_on=ON

## codership / mysql 8
wsrep_provider=/usr/lib/galera/libgalera_smm.so

## Galera Cluster Configuration
```

```
wsrep_cluster_name="test_cluster"
wsrep_cluster_address="gcomm://192.168.56.101,192.168.56.102,192.168.56.103"

## Galera Synchronization Configuration

wsrep_sst_method=rsync
```

```
## start 1st cluster node
systemctl stop mariadb
galera_new_cluster # always use this to bootstrap first cluster node

systemctl status mariadb
journalctl -u mariadb
```

MariaDB Cluster Configuration (Centos)

Step 1: Configuration SELinux (on every node)

```
## Option 1
## Disable selinux at runtime
sestatus
## switch from enforcing to permissive (runtime)
setenforce 0
getenforce
sestatus
```

```
## Option 2
## Open everything that is needed
## setools provide command semanage
## yum provides semanage
yum install setools
## Somehow redundant when you open the service altogether
## With mysqld_t
## semanage port -a -t mysqld_port_t -p tcp 3306
## semanage port -a -t mysqld_port_t -p tcp 4444
## semanage port -a -t mysqld_port_t -p tcp 4567
## semanage port -a -t mysqld_port_t -p udp 4567
## semanage port -a -t mysqld_port_t -p tcp 4568
semanage permissive -a mysqld_t
```

Step 2: Configure Firewall (on every node)

```
## Option 1
systemctl disable firewalld
systemctl stop firewalld
```

```
## Option 2
https://galeracluster.com/library/documentation/firewalld.html

## redundant - next line
## firewall-cmd --zone=public --add-service=mysql --permanent
firewall-cmd --zone=public --add-port=3306/tcp --permanent
firewall-cmd --zone=public --add-port=4444/tcp --permanent
firewall-cmd --zone=public --add-port=4567/udp --permanent
firewall-cmd --zone=public --add-port=4567/tcp --permanent
firewall-cmd --zone=public --add-port=4568/tcp --permanent
firewall-cmd --reload
## did this work ?
firewall-cmd --list-all-zones | grep -A10 "public"
```

Step 3: galera-configuration on every node

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0
## Set to 1 sec instead of per transaction
## for better performance // Attention: You might loose data on power
outage
innodb_flush_log_at_trx_commit=0
## Galera Provider Configuration
wsrep_on=ON
## centos8 (x86_64)
wsrep_provider=/usr/lib64/galera-4/libgalera_smm.so
```

```
## Galera Cluster Configuration
wsrep_cluster_name="test_cluster_jm"
wsrep_cluster_address="gcomm://10.10.11.141,10.10.11.166,10.10.11.170"
## Galera Synchronization Configuration
wsrep_sst_method=rsync
```

Step 4a: Start/add Node 1

```
## be sure mariadb is stopped
systemctl stop mariadb
galera_new_cluster

## now check for status
mysql
## look for cluster_size
mysql> show status like '%wsrep%'
```

Step 4b: Start/add Node 2

```
## like 2.4, but no galera_new_cluster
## instead
systemctl start mariadb
## see logs of connecting
journalctl -u mariadb
```

Details 4b: Step-by-Step -> Node 1

```
Schritt 1: z_galera.cnf rüberschieben (von Server 1)
Schritt 2: plugin -eintrag ändern -4 raus
Schritt 3: galera.cnf umbenennen galera.cnf.NOT (aus Installation 10.3, wenn vorhanden)
Schritt 4:
semanage permissive -a mysqld_t

Schritt 5: Firewall - Regeln eintragen
firewall-cmd --zone=public --add-port=3306/tcp --permanent
firewall-cmd --zone=public --add-port=4444/tcp --permanent
firewall-cmd --zone=public --add-port=4567/udp --permanent
firewall-cmd --zone=public --add-port=4567/tcp --permanent
firewall-cmd --zone=public --add-port=4568/tcp --permanent
firewall-cmd --reload

Schritt 6: Server mit: systemctl.stop mariadb;  systemctl start mariadb
Schritt 7: Überprüfung: mysql  show status like ‚wsrep%‘ # guckst cluster_size
```

Step 4c: Start/add Node 3

- Same as 4b