Galera Cluster For MySQL

Alexey Yurchenko, Codership Oy

Percona Live MySQL User Conference and Expo, 2012

Follow the tutorial

Using Percona XtraDB Cluster RPMs, at Amazon EC2

Datacenters:

us-west-1

us-east-1

eu-west-1

(look for **galera_tutorial_*** images)

Agenda

1. ~~Introduction (30–45 min)~~
   a. ~~Galera architecture overview~~
   b. ~~Important terms and concepts~~
2. ~~Migration from stock MySQL replication (30 min)~~
3. ~~LAN cluster setup (30 min)~~
   a. ~~Node configuration~~
   b. ~~Joining the cluster, state transfer~~
   c. ~~Executing SQL load~~
4. ~~Node failure and recovery under load (30 min)~~
   a. ~~Planned restart~~
   b. ~~Hard crash~~
   c. ~~Split-brain~~
5. ~~WAN replication (30 min)~~
6. ~~Troubleshooting~~
7. ~~Q&A~~

What Is Galera Cluster for MySQL?

Galera Cluster for MySQL = MySQL server × N

What is Galera Cluster for MySQL

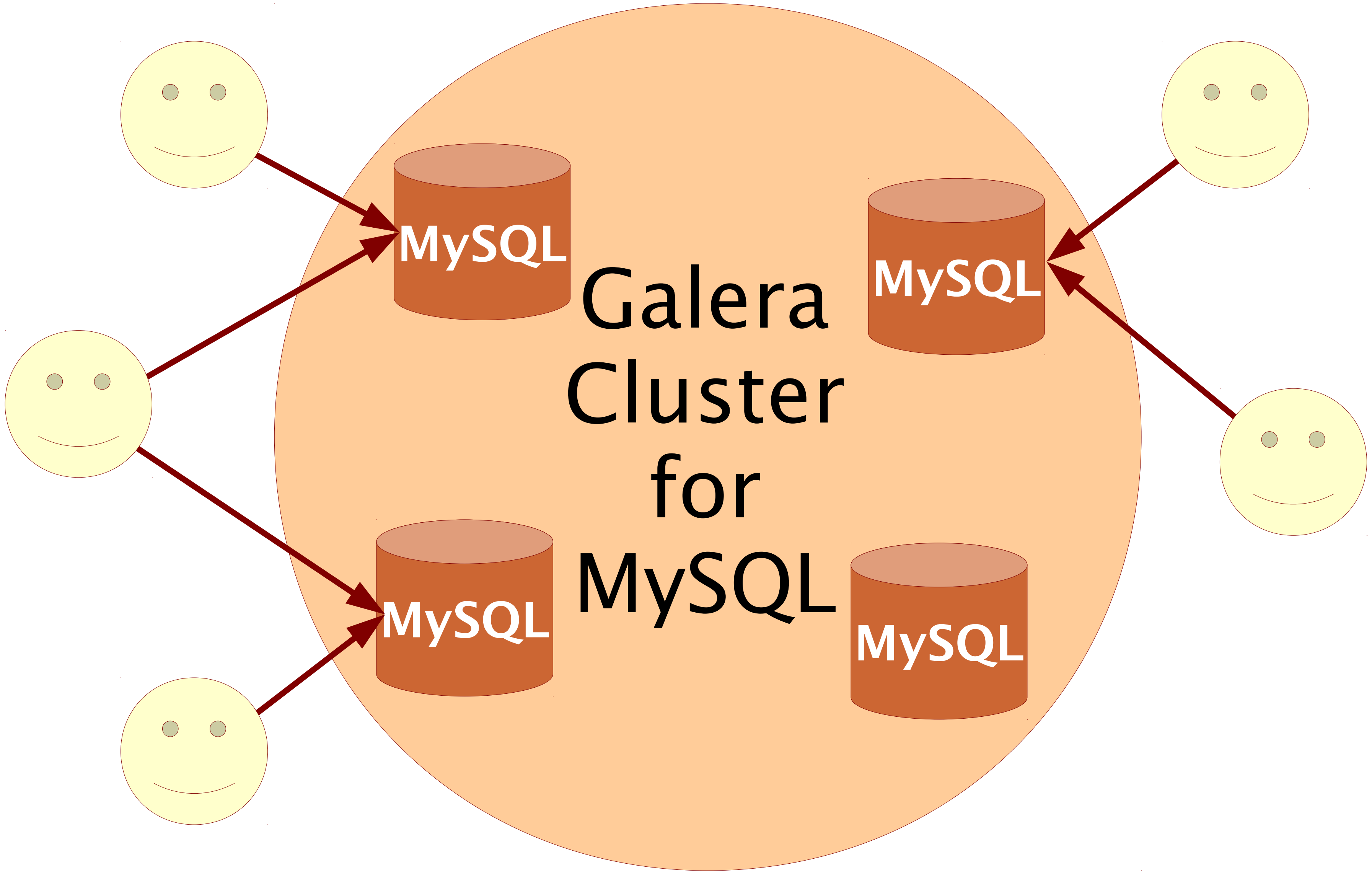
Each node is fully representative of the cluster:

1. No single point of failure
2. Synchronous................
3. 99.9% transparency

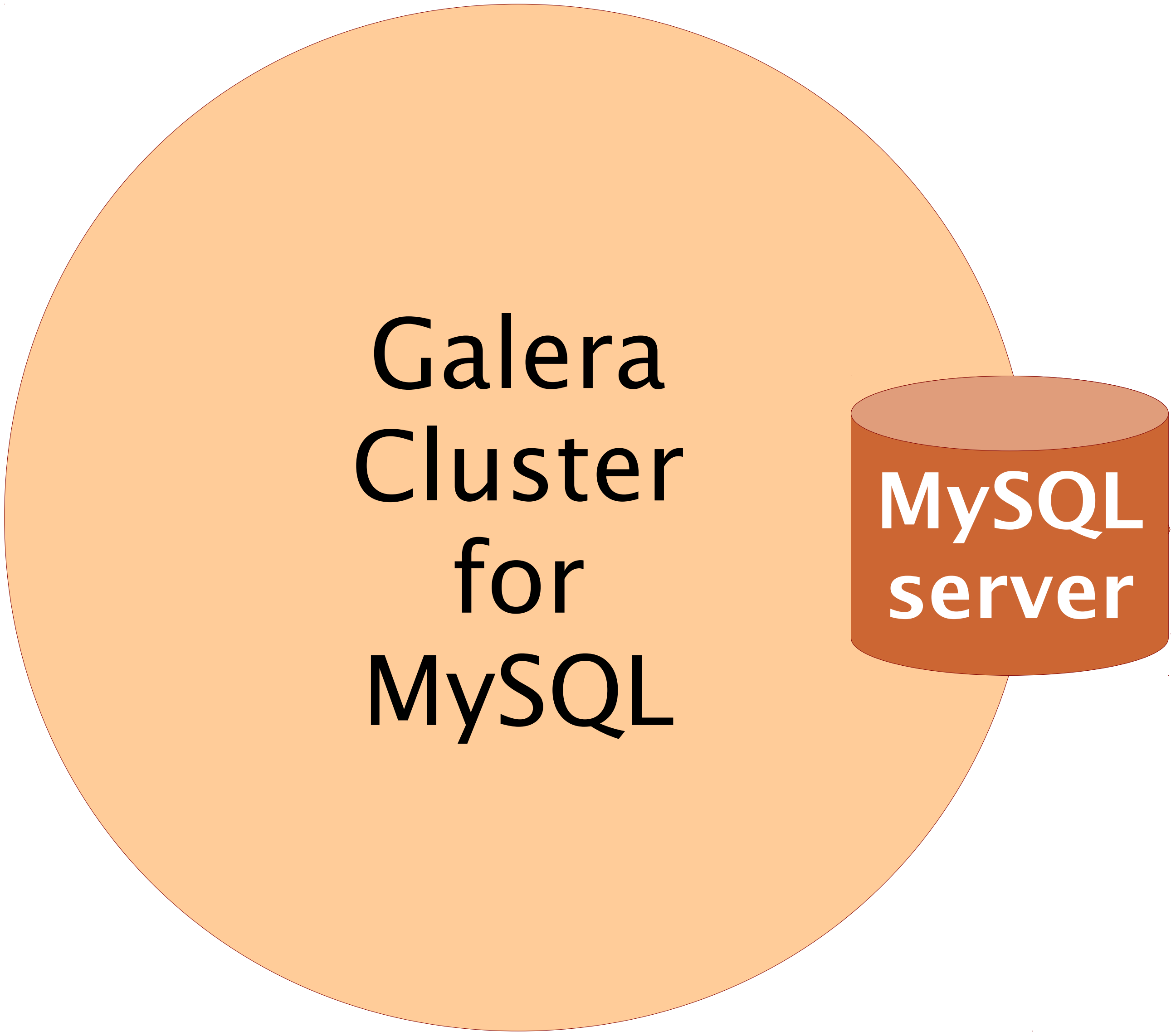
- No data loss
- No slave lag
- No master failover
- Full multi-master

codership

What Is Galera Cluster for MySQL?



What Is Galera Cluster for MySQL?

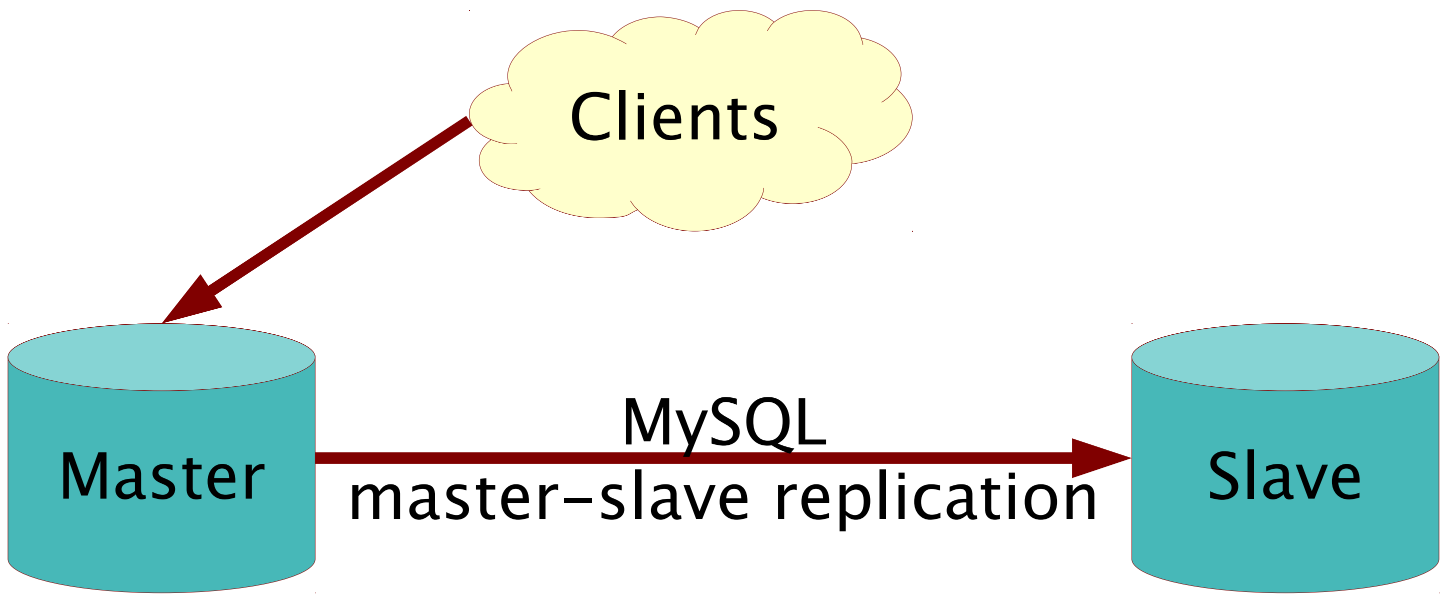


Server as a <u>part</u> of the cluster

Migrating from MySQL master-slave to Galera

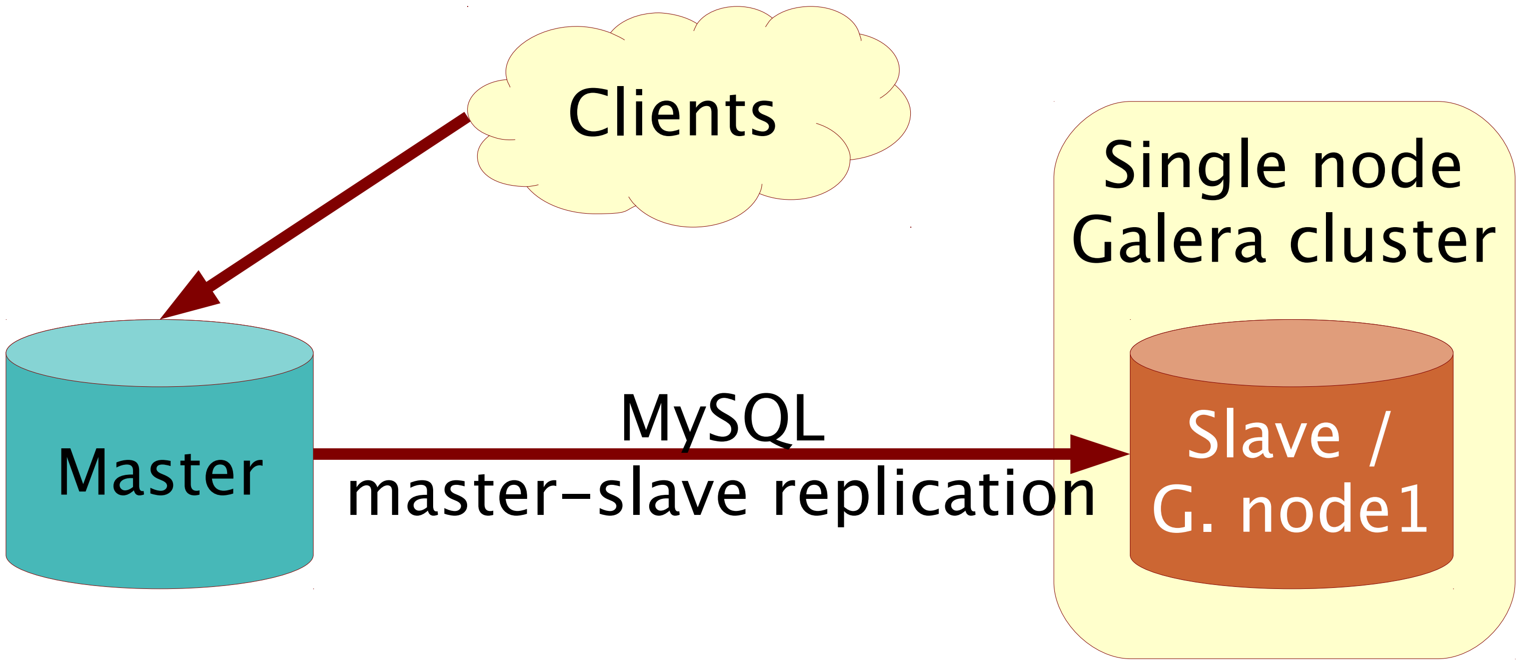
What will happen 1

We start with regular MySQL master-slave



What will happen 2

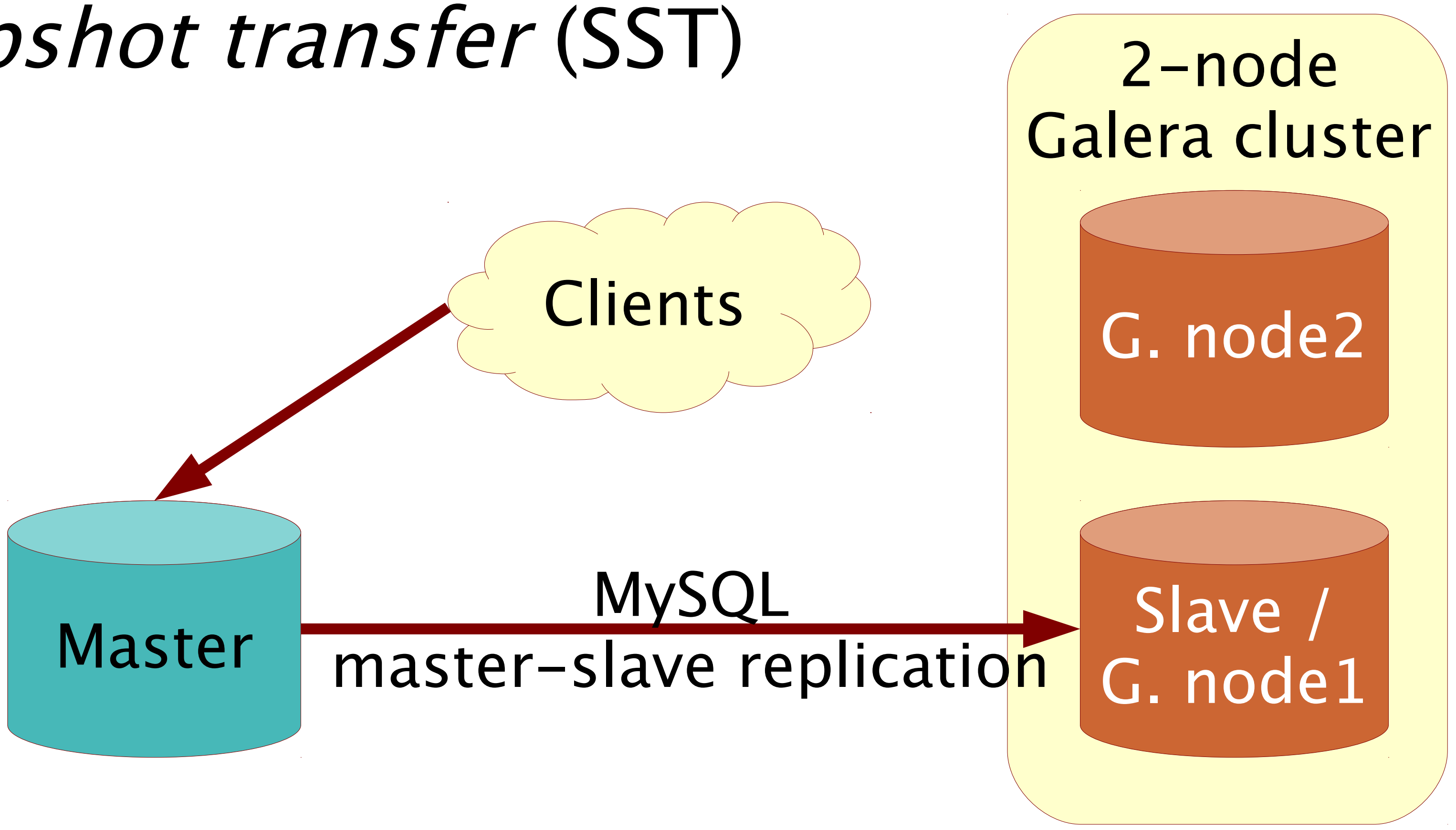
Upgrade MySQL slave to Galera node



codership

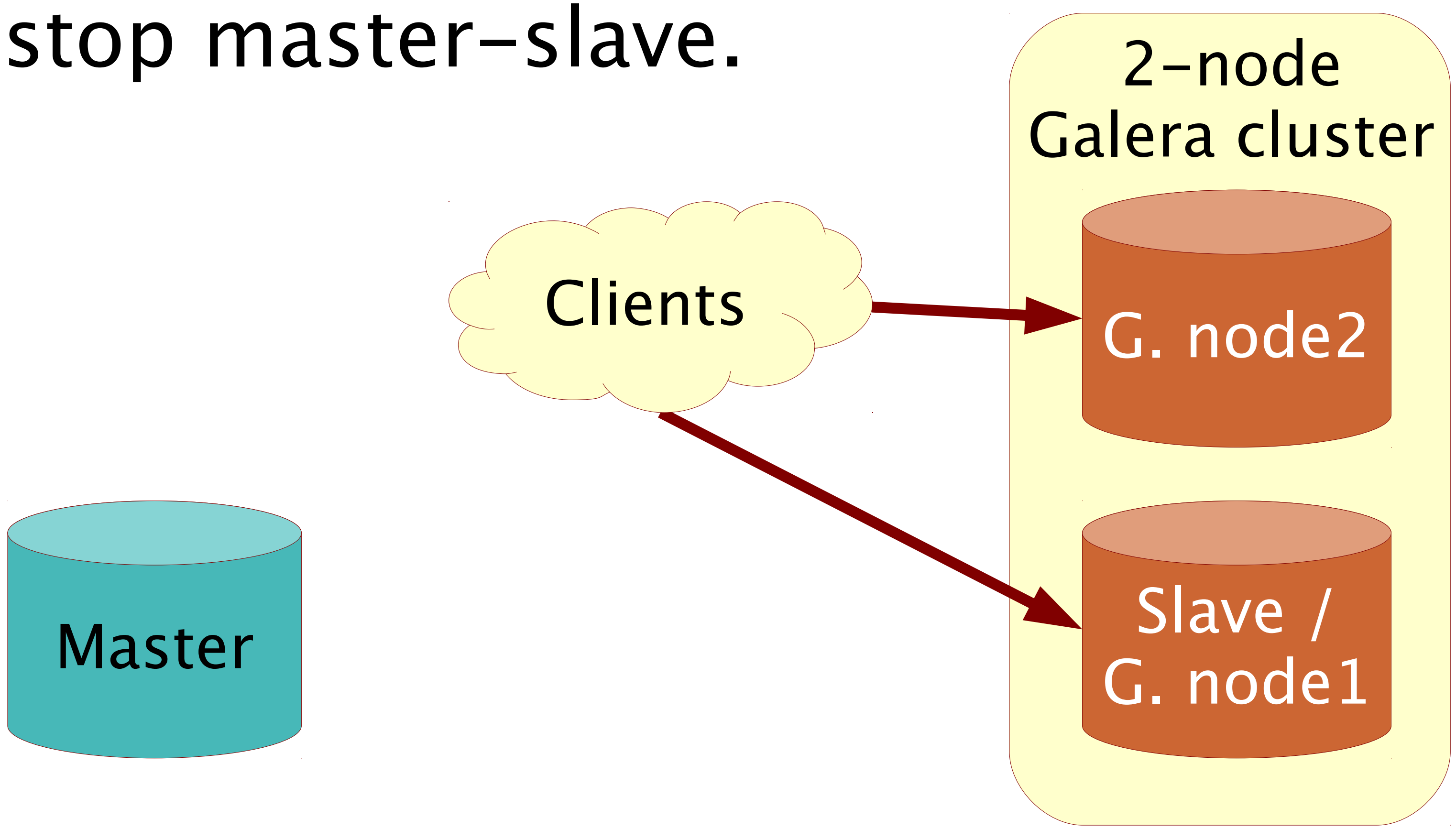
What will happen 3

Add a second Galera node using *state snapshot transfer* (SST)



What will happen 4

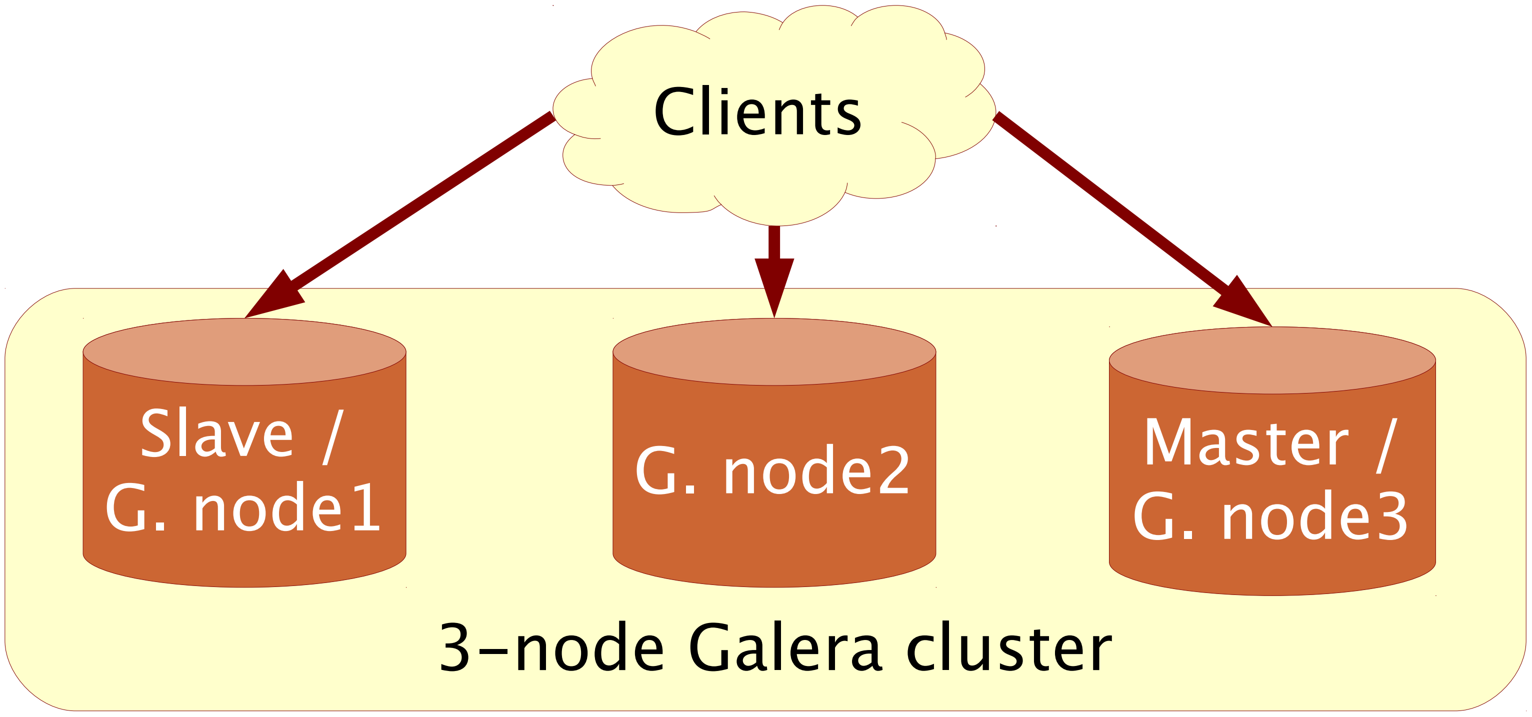
Failover client connections to Galera cluster and stop master-slave.



codership

What will happen 5

Upgrade MySQL master to Galera and join to cluster using *incremental state transfer* (IST)



codership

Converting slave to Galera node

1. Stop the slave: `slave> STOP SLAVE;`
2. Upgrade the software:
```
slave# rpm -e MySQL-server
slave# rpm -Uvh percona-xtrabackup-2.0.0-*.rpm \
Percona-XtraDB-Cluster-galera-2.0-*.rpm \
Percona-XtraDB-Cluster-server-5.5.20-23.4*.rpm
```
3. Configure wsrep options in my.cnf
4. Add `log_slave_updates=1` to my.cnf
   Remove `read_only = 1` from my.cnf
5. Restart the server.
6. Start the slave: `slave> START SLAVE;`

codership

Sample my.cnf for slave after upgrade

```
[mysqld]
wsrep_cluster_address=/usr/lib64/libgalera_smm.so
wsrep_node_address=gcomm:// # NOTE: This must be changed to peer address ASAP!
wsrep_node_name=node1
wsrep_provider='/usr/lib64/galera/libgalera_smm.so'
wsrep_provider_options='gcache.size=1G;socket.ssl_key=my_key;socket.ssl_cert=my_cert'
wsrep_slave_threads=16
wsrep_sst_method=xtrabackup
wsrep_sst_auth=root:

innodb_buffer_pool_size=1G
innodb_log_file_size=256M
innodb_autoinc_lock_mode=2
innodb_flush_log_at_trx_commit=0
innodb_doublewrite=0
innodb_file_per_table=1
binlog_format=ROW
datadir=/var/lib/mysql

log-bin = mysql-bin
server-id = 2
relay-log = mysql-relay-bin
#read-only = 1
log-slave-updates = 1
```

Important wsrep variables 1

`wsrep_provider:`

A path to wsrep provider library.

Important wsrep variables 2

`wsrep_cluster_address:`

Where to connect to cluster. Has a URI form:

```
'gcomm://another_node_address?opt1=val1&opt2=val2'
```

But normally just

```
'gcomm://another_node_address'
```

A special form to start a **new cluster**:

```
'gcomm://'
```

!!! Danger !!! Never leave it in my.cnf !!!

codership

Important wsrep variables 3

`wsrep_node_address:`

An optional address of the node. A short-cut way to configure listen addresses for replication and state transfers.

By default it will be initialized to the first network interface returned by ifconfig. This however is unreliable. For best results it must be initialized explicitly.

Important wsrep variables 4

`wsrep_node_name:`

An optional name for the node. It will be used in logging and to identify the desired donor for state transfer.

By default it will be initialized to hostname, but that may turn out to be not unique (it does not have to) or unwieldy.

Important wsrep variables 5

`wsrep_provider_options:`

Semicolon-separated list of options specific to provider. Some useful Galera options:

gcache.size – a size of the permanent transaction on-disk cache.

socket.ssl_key, socket.ssl_cert – SSL key and certificate files.

Important wsrep variables 6

`wsrep_slave_threads:`

How many threads to launch for parallel applying.

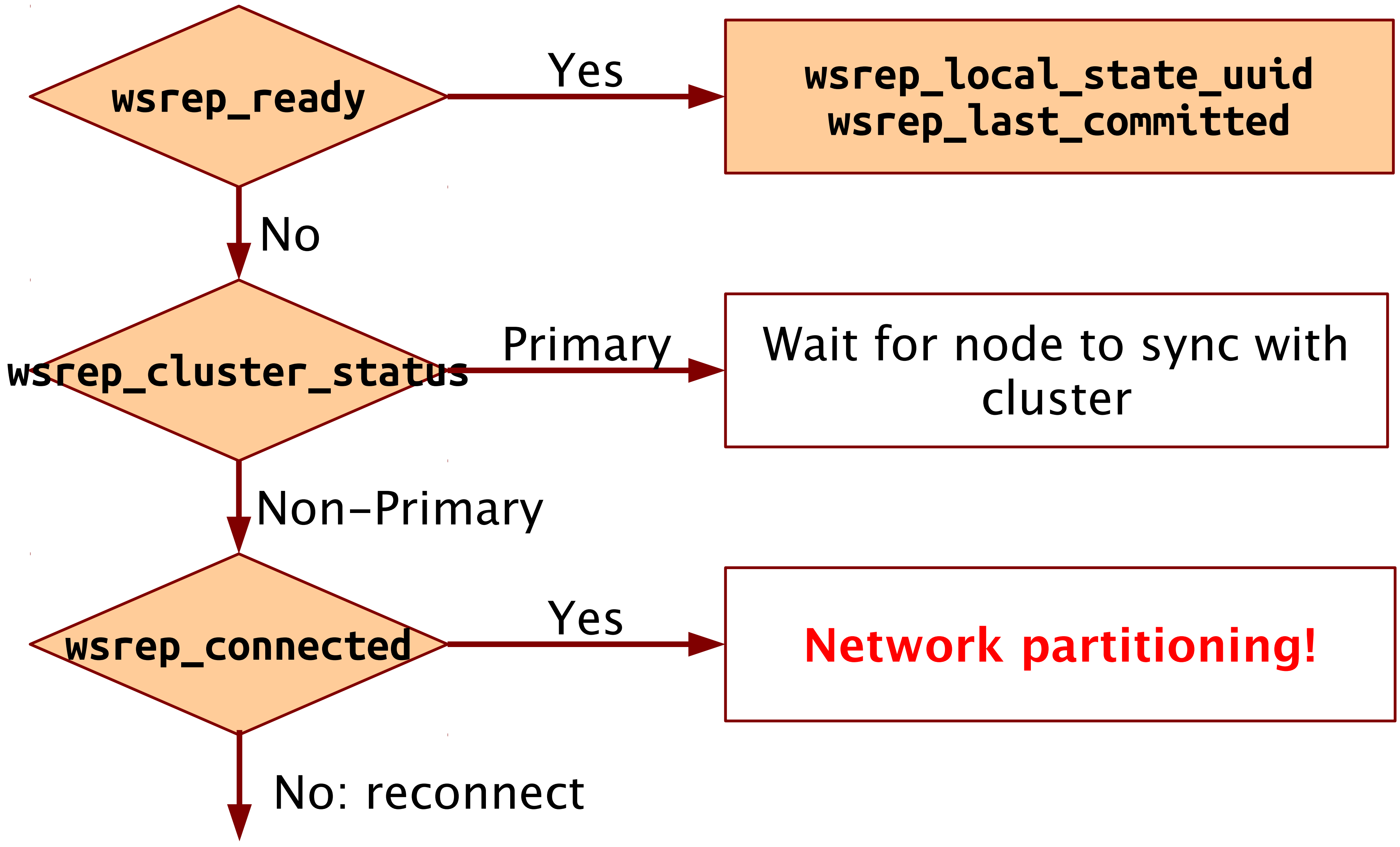
> 1 requires certain InnoDB settings. Applying of STATEMENT-based events is always serialized.

Important wsrep variables 7

`wsrep_sst_method:`

Base package contains scripts for mysqldump, rsync and xtrabackup based state snapshot transfers. Own scripts can be crafted (handy for backup). Default is mysqldump. It also requires setting wsrep_sst_auth to MySQL root user:password pair.

Status of Galera Replication



codership

Starting a 2nd Galera node

1. Install Percona RPMs. No need to configure initial database, it will be brought by SST.
2. Edit my.cnf. Set wsrep_cluster_address to slave node IP.
3. Start mysqld:

```
node2# service mysql start
```

4. Update my.cnf on slave:
   set wsrep_cluster_address to 2nd node IP.

Important Concepts 1

Application State:

For Galera application state is a set of data that application decides to replicate. By default it is a whole of MySQL databases (i.e. every node is a complete replica of another).

Application state is identified by a Global Transation ID.

codership

Important Concepts 1

Global Transaction ID (GTID):

`f7720ae0-6f9b-11e1-0800-598d1b386dce:32520198989`

CLUSTER/HISTORY/STATE UUID

TRX/STATE SEQNO

codership

Important Concepts 1

Initial State:

f7720ae0-6f9b-11e1-0800-598d1b386dce:0

Undefined State:

00000000-0000-0000-0000-000000000000:-1

Important Concepts 1

State Snapshot Transfer (SST):

A transfer of a consistent snapshot of a node state corresponding to a certain GTID in order to initialize the state of a newly joining cluster node from an already initialized node (donor).

codership

Performance of Galera replication

`wsrep_flow_control_paused`: what fraction of the time replication was paused.

`wsrep_flow_control_sent`: how many times this node paused replication.

`wsrep_local_recv_queue_avg`: average length of slave trx queue – a sign of slave side bottleneck.

`wsrep_cert_deps_distance`: how many transactions can be applied in parallel.

`wsrep_local_send_queue_avg`: a sign of network bottleneck.

Failover to Galera cluster

1. Briefly pause the load.
2. Make sure slave caught up with the master:
   ```
   master> SHOW MASTER STATUS\G
   slave> SHOW SLAVE STATUS\G
   ```
3. Take note of GTID on slave:
   ```
   slave> SHOW STATUS LIKE 'wsrep_%';
   ```
4. Set binary log format to ROW on slave:
   ```
   slave> SET GLOBAL binlog_format=ROW;
   ```
5. Direct the load to slave and the 2nd Galera node.

Make former master a Galera node

1. Upgrade software.
2. Configure my.cnf.
   Set wsrep_cluster_address to any of Galera nodes' names.
3. Copy/forge `grastate.dat` file. Set state information to the GTID we noted previous page.
4. Start mysqld.

Important Concepts 2

Incremental State Transfer (IST):

Catch up with the cluster by replaying missing transactions. Requires

- known initial node state;
- enough transactions cached at the donor.

Important Concepts 3

Node Failure:

For Galera node a peer crash is indistinguishable from network failure. Hence node is considered failed when it no longer can be communicated with. Communication is verified by receiving messages or keepalives.

`evs.inactive_timeout` sets the timeout after which node is considered inactive (dead).

`evs.suspect_timeout` sets the timeout after which the node can be pronounced dead if everyone else agrees.

Galera WAN Replication

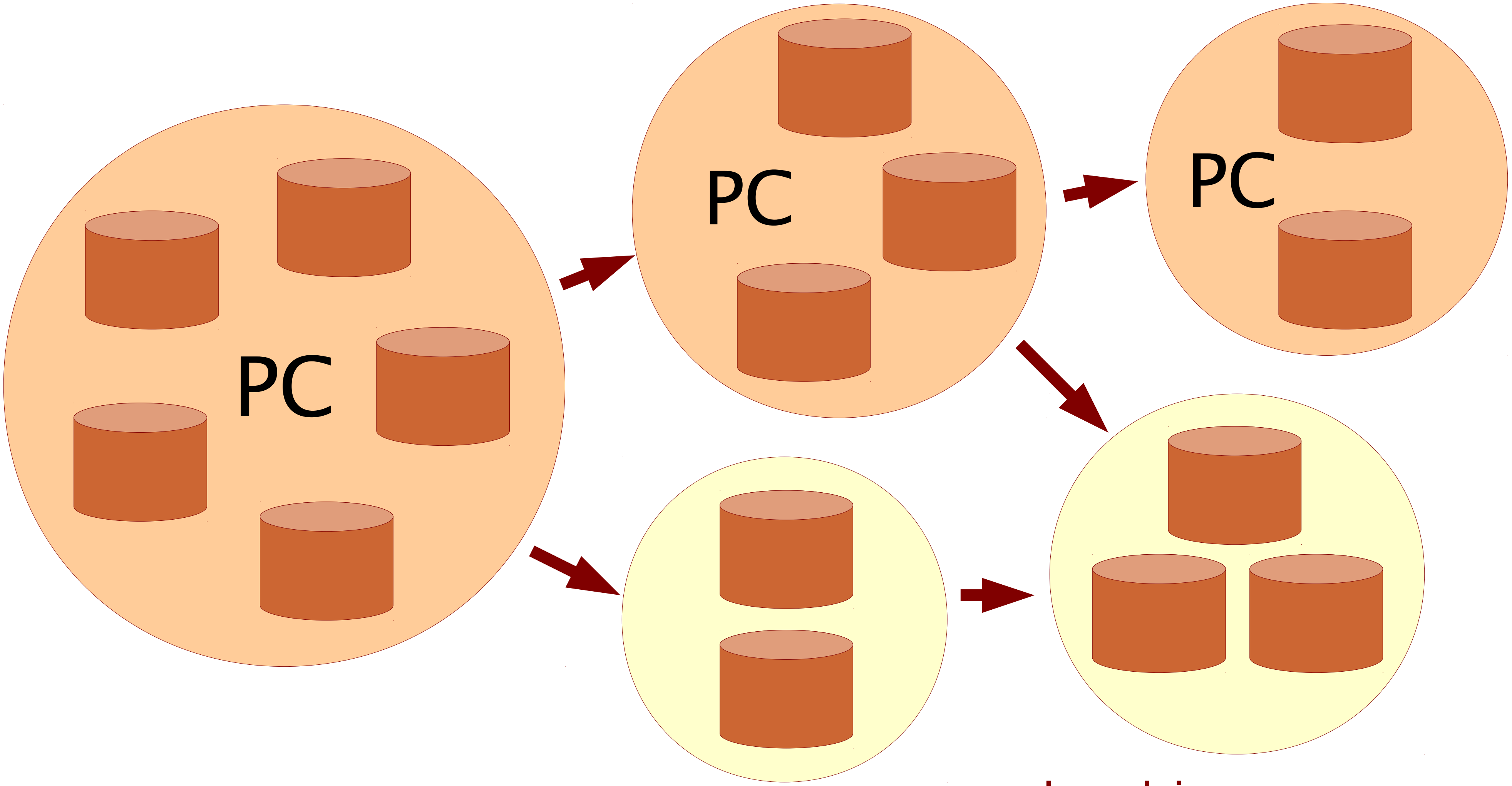
Currently Galera has no notion of a local or remote node – it works as long as TCP works.

May need tuning to be more tolerant to network hiccups – Galera (wsrep provider) options:

```
evs.keepalive_period = PT3S;
evs.inactive_check_period = PT10S;
evs.suspect_timeout = PT30S;
evs.inactive_timeout = PT1M;
evs.consensus_timeout = PT1M
```

Important Concepts 4

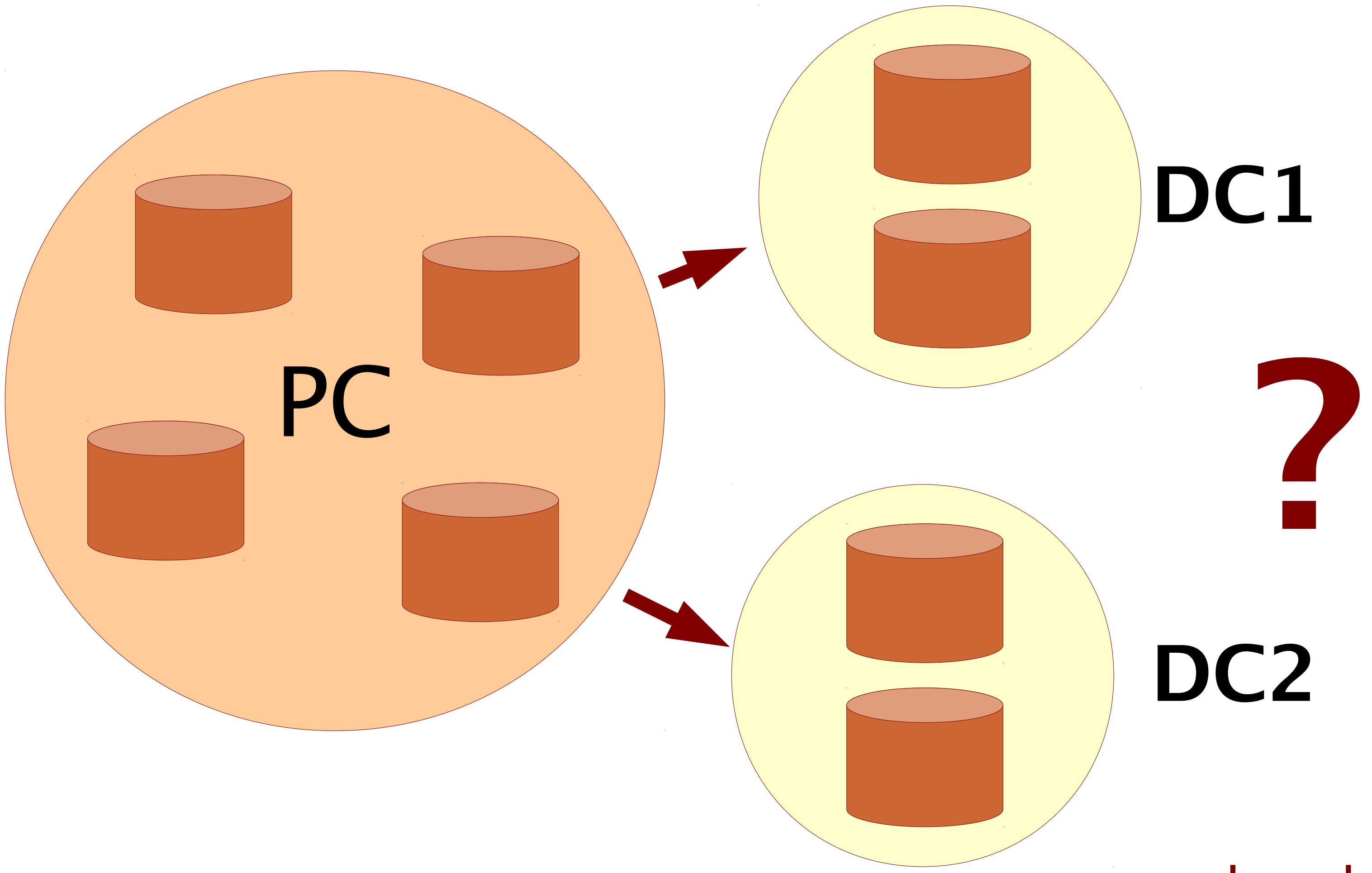
Primary Component (PC):



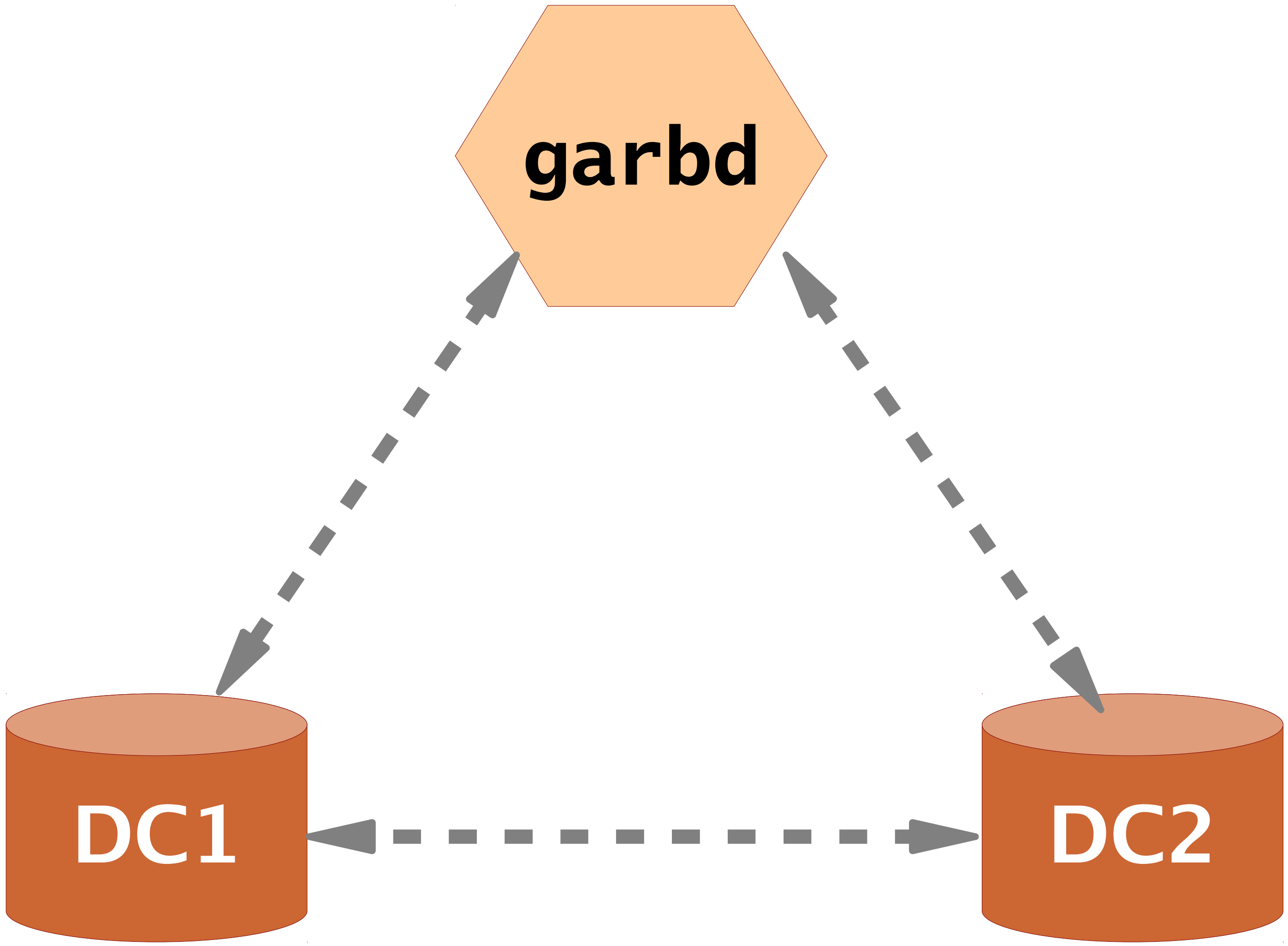
codership

Important Concepts 5

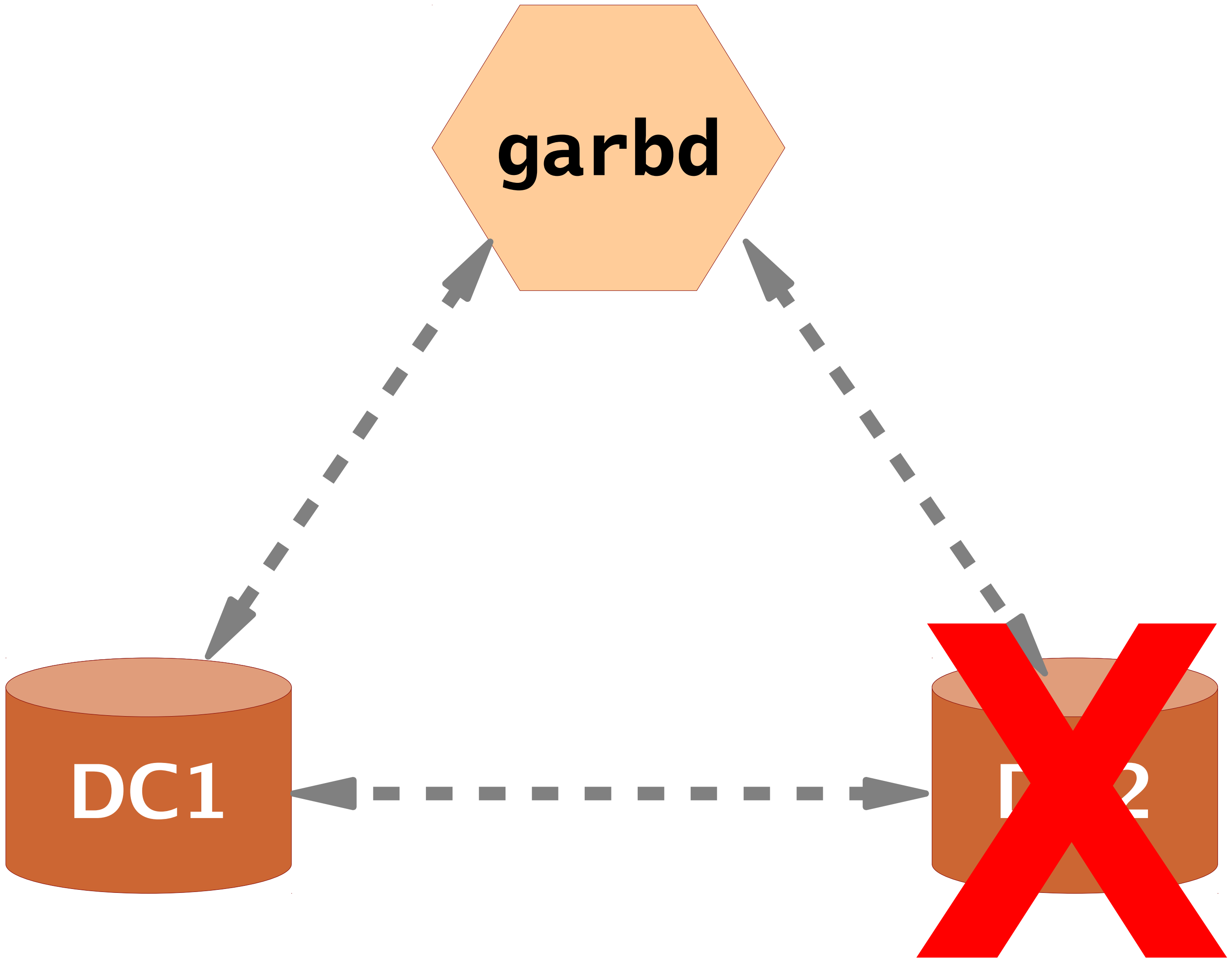
Split-brain:



Galera Arbitrator

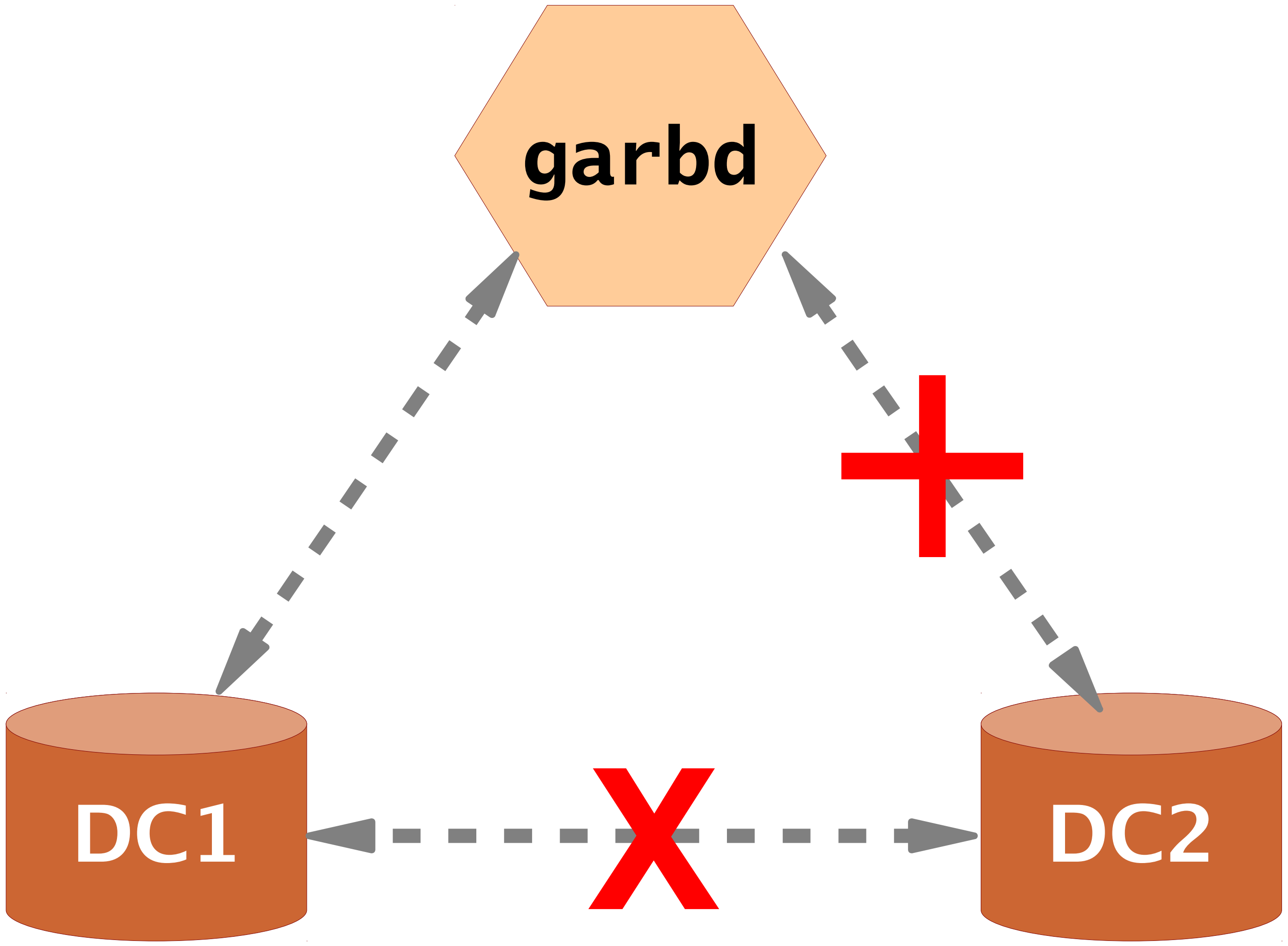


Galera Arbitrator



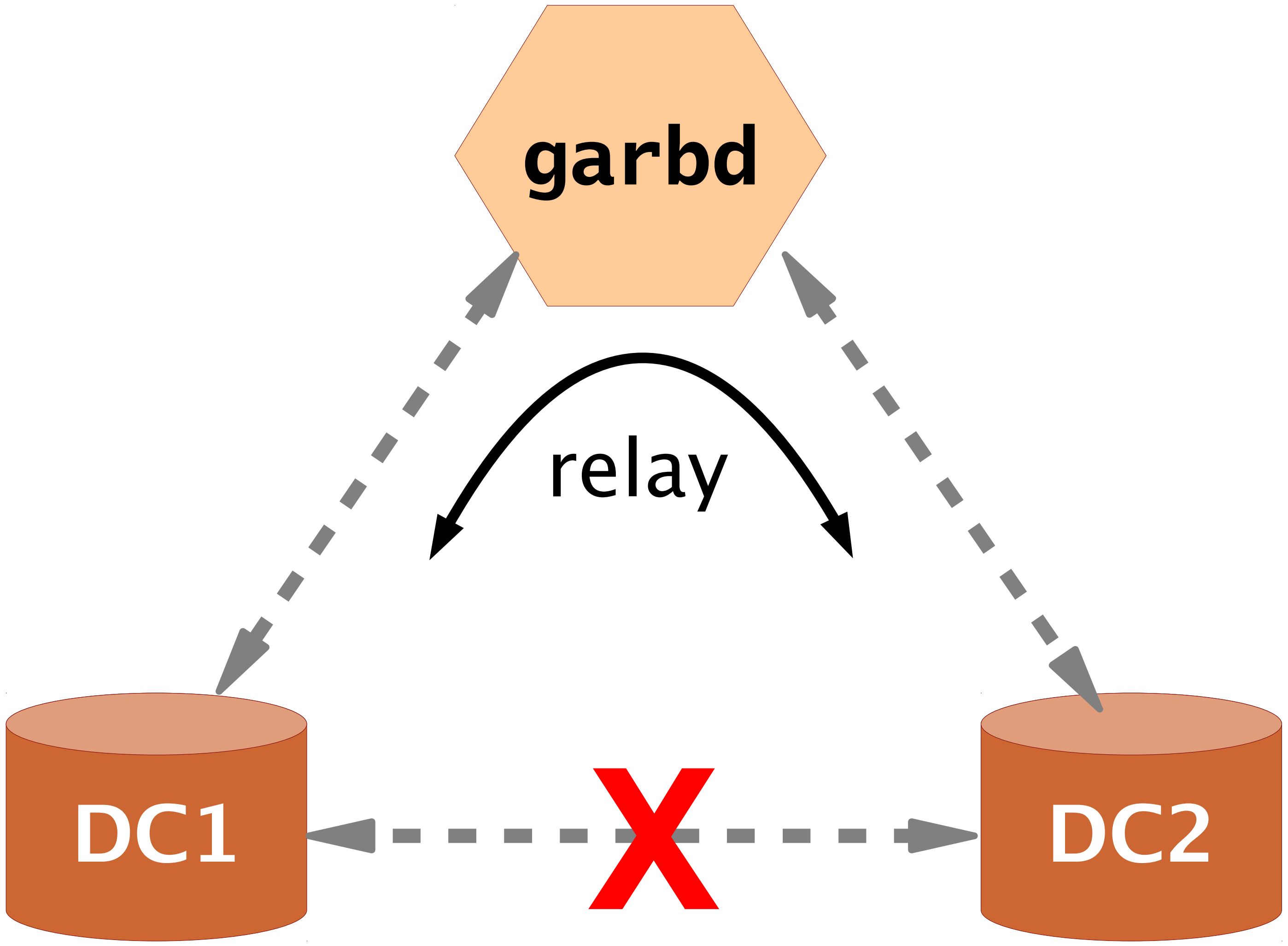
codership

Galera Arbitrator



codership

Galera Arbitrator



codership

Backup, backup, backup

Just backup one of the nodes. But backup without GTID is not that useful – backup with GTID:

1) xtrabackup:

```
node# innobackupex --galera-info
```

creates xtrabackup_galera_info in the datadir

2) Custom backup with arbitrator:

```
node# garbd … --donor node1 --sst custom
```

calls `wsrep_sst_custom` on node1.