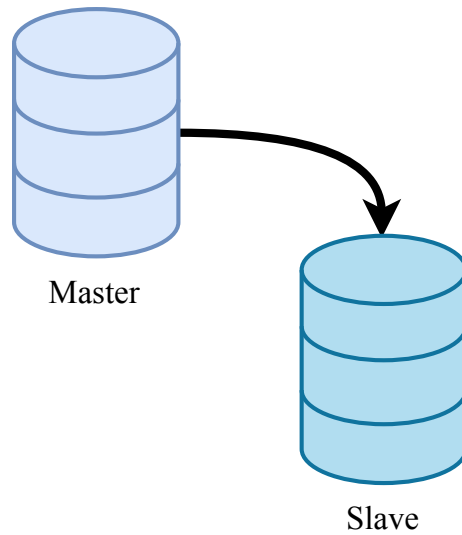


irontec



avanzado

Replicación MySQL



Introducción

- La replicación habilita la copia de datos de un servidor MySQL (el **master**) a uno o más servidores MySQL (los **slaves**).
- La replicación es asíncrona por defecto
- El *slave* no tiene por qué estar todo el tiempo conectado
- Dependiendo de la configuración puedes replicar todas las databases, algunas, o incluso sólo alguna tabla

Ventajas

- **Escalado:** La carga (lecturas) puede ser repartida por varios slaves
- **Seguridad de datos:** Como los slaves tienen una réplica de los datos, un slave se puede parar para poder hacer un backup sin tener miedo a la corrupción de datos
- **Analíticas:** El análisis de la información de los datos puede realizarse en los slaves, quitando carga al master
- **Distribución de datos a larga distancia.** La réplica puede estar en otro datacenter.

Cómo funciona la replicación

- Los datos del master se pasan al slave
- Existen dos maneras
 - Activando los **binary logs**
 - Usando los **identificadores de transacciones globales** (desde MySQL 5.7)
- El slave debería ser de sólo lectura!!

Compatibilidad en la replicación

- MySQL acepta replicación de una serie a una versión superior
 - Master 5.6 a Slave 5.7 o Master 5.7 a Slave 8
 - Puede haber problemas si se hace uso de opciones deprecated
 - No se permite hacer salto de dos versiones
- Se recomienda siempre usar las últimas versiones

Master ⇒ Slave

con binary logs

- El master escribe sus cambios en el binary-log
- El slave lee los binary-log del master y ejecuta los eventos en local
- Cada slave recibe una copia completa del binary-log
 - es responsabilidad del slave decidir si ejecuta el evento recibido
- Cada slave mantiene la posición del binary log en el que se encuentra

Master ⇒ Slave

- El master y cada slave tiene que ser configurado con un ID único
 - **server-id**
- El slave tendrá configurado:
 - el master host name
 - log file name
 - posición
- Esta configuración se puede cambiar con **CHANGE MASTER TO** en el slave

Master ⇒ Slave

- Hacemos que el nodo 1 sea **master**

```
[mysqld]  
server-id = 1  
log_bin = /var/log/mysql/binlog  
expire_logs_days = 7  
max_binlog_size = 100M
```

- Creamos usuario de replicación

```
CREATE USER 'repl'@'192.168.1.101' IDENTIFIED BY 'password';  
GRANT REPLICATION SLAVE ON *.* TO 'repl'@'192.168.1.101';  
FLUSH PRIVILEGES;
```

- Hacemos dump de la base de datos:

```
mysqldump --all-databases --master-data > dbdump.db
```

Master ⇒ Slave

Master

```
mysql> show master status\G
***** 1. row *****
      File: binlog.000001
      Position: 155

cat dbdump.db

-- MySQL dump 10.13  Distrib 8.0.16, for Linux (x86_64)
--
-- Host: localhost    Database:
-- -----
-- Server version      8.0.16
...
CHANGE MASTER TO MASTER_LOG_FILE='binlog.000001', MASTER_LOG_POS=155;
```

Master ⇒ Slave

Slave

- Modificamos el **server-id**

```
[mysqld]  
server-id = 2
```

- Importamos la base de datos

```
mysql> CHANGE MASTER TO  
      MASTER_HOST='192.168.1.100',  
      MASTER_USER='repl',  
      MASTER_PASSWORD='password',  
      MASTER_LOG_FILE='binlog.000001',  
      MASTER_LOG_POS=155;  
  
mysql> start slave user='repl' password='password';
```

Master ⇒ Slave

Slave

```
mysql> show slave status\G;

***** 1. row *****
      Slave_IO_State: Waiting for master to send event
        Master_Host: 192.168.1.100
        Master_User: repl
        Master_Log_File: binlog.000001
  Read_Master_Log_Pos: 155
    Slave_IO_Running: Yes
   Slave_SQL_Running: Yes
        Last_Errno: 0
        Last_Error:
  Seconds_Behind_Master: 0
```

Ejercicio

Crear sistema de replicación Master \Rightarrow Slave

- server-id
- binary-logs
- crear usuario de replicación
- dump
- importar dump
- change master to
- start slave

Master \Leftrightarrow Master

- Es un sistema Master \Rightarrow Slave bidireccional
 - Master \Rightarrow Slave
 - Slave \Leftarrow Master
- Hay que tener en cuenta los índices auto-incrementales:

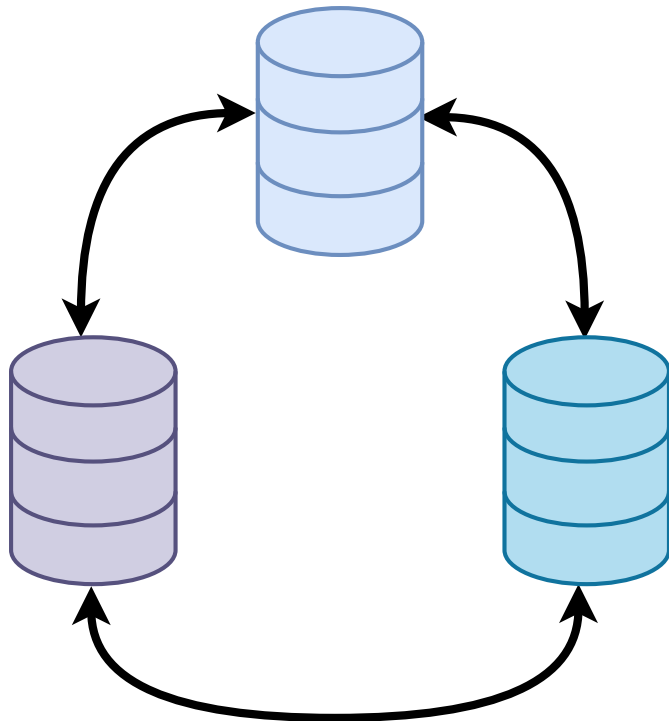
```
* Host 1:
  auto_increment_increment = 2
  auto_increment_offset = 1
* Host 2:
  auto_increment_increment = 2
  auto_increment_offset = 2
```

Ejercicio

Crear sistema de replicación Master \Leftrightarrow Master

- configurar auto-increments
- crear usuario de replicación
- dump
- importar dump
- change master to
- start slave

Clusters



Antes de empezar

- **Clúster:** es un conjunto de dos o más máquinas que se caracterizan por mantener una serie de servicios compartidos y por estar constantemente monitorizándose entre sí ([Wikipedia](#))
- No hay que confundir un clúster de alta disponibilidad con un clúster de alto rendimiento
- **MySQL Cluster != MySQL *en clúster***
 - **MySQL Cluster:** Sistema de base de datos distribuido
 - **MySQL "Master \Leftrightarrow Master":** Clúster de 2 MySQLs

Características

- Replicación multi-master
- Activo-activo
- Replicación síncrona
 - Escribes en un nodo y se propaga al resto
- Sólo soporta InnoDB o XtraDB
- ...

Alternativas

- Para hacer un clúster de MySQL existen distintas opciones
 - **Galera**: Fue el precursor
 - **MySQL Cluster**: Versión Cluster de MySQL
 - **Percona XtraDB Cluster**: Versión de la empresa Percona

Percona Cluster

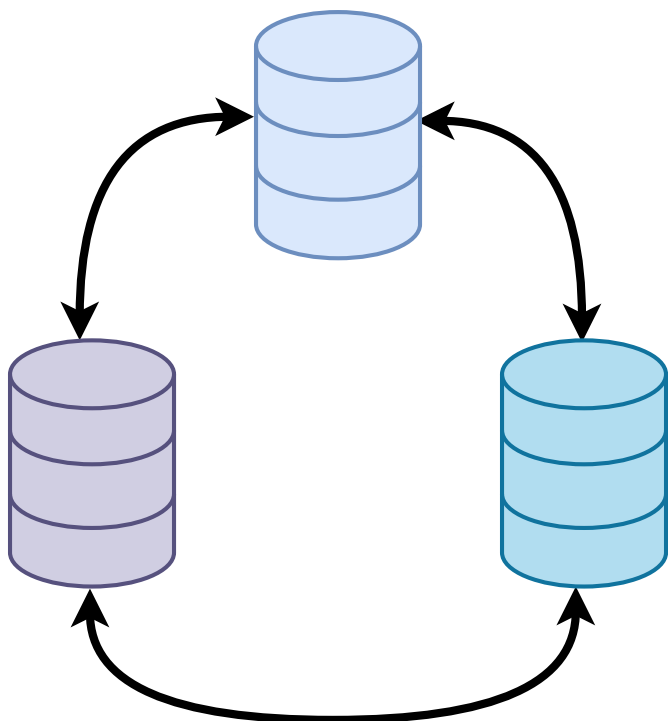


PERCONA
XtraDB Cluster

Características

- Provisión automática de nodos
- Consistencia de datos
- Replicación paralela
- Script de configuración para ProxySQL
 - Lo veremos más adelante

- Vamos a crear un XtraDB Cluster de 3 nodos



- Hay que evitar hacer un clúster de 2 o número par de nodos
 - Puede provocar ***split brain***

Instalación

en todos los nodos

- Vamos a instalar el repositorio de APT:

```
wget https://repo.percona.com/apt/percona-release_latest.generic_all.deb
```

- Instalamos el repositorio

```
dpkg -i percona-release_latest.generic_all.deb
```

- Instalamos la última versión de XtraDB Cluster (5.7)

```
apt-get install percona-xtradb-cluster-57
```

- Paramos Percona XtraDB Cluster

```
service mysql stop
```

Configuración

- Vamos a suponer el siguiente escenario

Node	Host	IP
1	pxc1	10.0.0.1
2	pxc2	10.0.0.2
3	pxc3	10.0.0.3

Configuración

- Modificamos en el nodo1
 - `/etc/mysql/percona-xtradb-cluster.conf.d/wsrep.cnf` :

```
wsrep_provider=/usr/lib/libgalera_smm.so

wsrep_cluster_name=pxc-cluster
wsrep_cluster_address=gcomm://10.0.0.1, 10.0.0.2, 10.0.0.3

wsrep_node_name=pxc1
wsrep_node_address=10.0.0.1

wsrep_sst_method=xtrabackup-v2
wsrep_sst_auth="sstuser:passw0rd"

pxc_strict_mode=ENFORCING
binlog_format=ROW
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
```

Configuración

- En los otros dos nodos la misma configuración salvo las siguientes variables:

- Nodo 2:

```
wsrep_node_name=pxc2  
wsrep_node_address=10.0.0.2
```

- Nodo 3:

```
wsrep_node_name=pxc3  
wsrep_node_address=10.0.0.3
```

Explicación de las variables

- **wsrep_provider:** Especifica el path de la librería de Galera
- **wsrep_cluster_name:** Nombre del clúster
- **wsrep_cluster_address:** Las direcciones de todos los nodos del clúster
- **wsrep_node_name:** El nombre del nodo dentro del clúster
- **wsrep_node_address:** La IP del nodo

Explicación de las variables

- **wsrep_sst_method**: *State Snapshot Transfer*. Sistema de transferencia del estado de un nodo a otro cuando se añade al clúster
 - XtraDB Cluster usa **xtrabackup-v2**, ya que no deja el nodo donante en modo read-only
- **wsrep_sst_auth**: Autenticación para SST
- **pxc_strict_mode**: *ENFORCING* bloquea el uso de features no soportadas en Percona

Explicación de las variables

- **binlog_format**: Galera sólo soporta replicación a nivel de fila, de ahí *ROW*
- **default_storage_engine**: El *engine* por defecto. **No se puede usar MyISAM**
- **innodb_autoinc_lock_mode**: Sistema autoincremental, **siempre tiene que ser 2**.

Inicializar el clúster

- El primer nodo que inicialicemos será el que contenga los datos (ejemplo: pxc1)
 - Estos datos serán los que se repliquen en el resto

```
/etc/init.d/mysql bootstrap-pxc
```

```
mysql@pxc1> show status like 'wsrep%';
```

Variable_name	Value
wsrep_local_state_uuid	266aeca1-9769-11e9-9ca9-2b8d76f621ed
wsrep_local_state	4
wsrep_local_state_comment	Synced
wsrep_cluster_size	1
wsrep_cluster_status	Primary
wsrep_connected	ON
wsrep_ready	ON

Inicializar el clúster

- Hay que crear usuario para el SST (*State Snapshot Transfer*) en **pxc1**
 - Este es el usuario, y su password, configurado previamente

```
CREATE USER 'sstuser'@'localhost' IDENTIFIED BY 'passw0rd';  
GRANT RELOAD, LOCK TABLES, PROCESS, REPLICATION CLIENT ON *.* TO  
  'sstuser'@'localhost';  
FLUSH PRIVILEGES;
```

- Esto nos permitirá añadir nodos al clúster

Añadir nodos al clúster

- Añadimos **pxc2** al clúster levantando el servicio

```
[root@pxc2 ~]# /etc/init.d/mysql start
```

- **/var/log/mysqld.log**

```
WSREP: 1.0 (pxc2): State transfer from 0.0 (pxc1) complete.  
WSREP: SST leaving flow control  
WSREP: Shifting JOINER -> JOINED (TO: 6)  
WSREP: Member 1.0 (pxc2) synced with group.  
WSREP: Shifting JOINED -> SYNCED (TO: 6)  
WSREP: Synchronized with group, ready for connections  
WSREP: Setting wsrep_ready to true
```

- Añadimos **pxc3** al clúster levantando el servicio

```
[root@pxc3 ~]# /etc/init.d/mysql start
```


Comprobar estado del clúster

En cualquiera de los nodos:

```
mysql> show status like 'wsrep%';
```

Variable_name	Value
wsrep_local_state_uuid	266aeca1-9769-11e9-9ca9-2b8d76f621ed
wsrep_local_state	4
wsrep_local_state_comment	Synced
wsrep_cluster_weight	3
wsrep_cluster_size	3
wsrep_cluster_status	Primary
wsrep_connected	ON
wsrep_ready	ON

Comprobando la replicación

Creamos una base de datos en **pxc2**

```
mysql@pxc2> CREATE DATABASE prueba;  
Query OK, 1 row affected (0.01 sec)
```

Creamos una tabla en **pxc3**

```
mysql@pxc3> USE prueba;  
Database changed  
  
mysql@pxc3> CREATE TABLE ejemplo (node_id INT PRIMARY KEY, node_name VARCHAR(30));  
Query OK, 0 rows affected (0.05 sec)
```

Insertamos datos en **pxc1**

```
mysql@pxc1> INSERT INTO prueba.ejemplo VALUES (1, 'yeah!');  
Query OK, 1 row affected (0.02 sec)
```

Estado de la replicación

- Comprobar estado de cada nodo

```
mysql> SHOW STATUS LIKE 'wsrep_local_state_comment';
+-----+-----+
| Variable_name      | Value |
+-----+-----+
| wsrep_local_state_comment | Synced |
+-----+-----+
1 row in set (0.00 sec)
```

Ejercicio

- Crear clúster de 3 nodos con Percona XtraDB Cluster

Reinicio de nodos

- Hay que parar el servicio MySQL
- Si hay algún problema al arrancar, Galera descartará los datos y se volverá a sincronizar Reinicio de los nodos
- Si existe algún fallo y no se borra el fichero PID, el servicio no levantará

Monitorización básica

- Estado del clúster en cada nodo:
 - **wsrep_cluster_status** : Tiene que ser *Primary*
- Estado del nodo
 - **wsrep_connected**: tiene que ser *ON*
 - **wsrep_ready**: tiene que ser *ON*
- A tener en cuenta:
 - **wsrep_cluster_size**: sea el número esperado
 - **wsrep_local_bf_aborts**: número de transacciones abortadas

Recuperación de clúster Percona

- Un clúster también puede tener errores
 - Puede caer un nodo
 - Puede caer toda la plataforma
 - Un nodo desaparece
 - ...
- Posibles soluciones
 - Recover PXC Cluster



Características

- Sistema centralizado para administración de bases de datos
 - MySQL
 - MySQL/Percona XtraDB/Galera Cluster
 - MongoDB
 - PostgreSQL
- Realizar despliegues
- Monitorización
- Versión Libre y Enterprise
 - Gestión de backups y más (versión Enterprise)
- Interfaz web
 - Lo mejor es tener un nodo para ello

Instalación

- Añadir clave pública

```
wget http://repo.severalnines.com/severalnines-repos.asc -O- | apt-key add -
```

- Añadir repositorio

```
wget http://www.severalnines.com/downloads/cmon/s9s-repo.list -P /etc/apt/sources.list.d/
```

- Instalar

```
apt-get update && apt-get install clustercontrol
```

- Vamos al navegador https://IP_HOST/clustercontrol

Configuración

- Para poder administrar o desplegar, necesitamos acceso *root* a los nodos
 - Generamos clave pública/privada en el servidor cloudcontrol

```
ssh-keygen -t rsa
```

- Copiamos la clave a los nodos

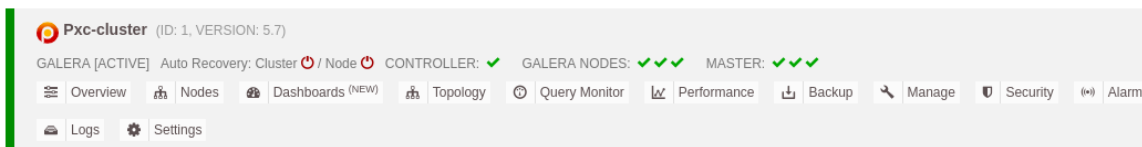
```
ssh-copy-id NODE_IP
```

- Aceptamos conexiones **SSH como root** en los nodos a controlar:
/etc/ssh/sshd_config :

```
PermitRootLogin yes
```

Importamos clúster

- Datos necesarios:
 - **usuario** para realizar la conexión SSH
 - **puerto ssh**
 - **nombre del clúster**
 - **tipo de clúster y versión**
 - **usuario de administración y contraseña**
 - al menos un **nodo** del clúster



Ejercicio

- Instalar cloudcontrol
- Importar configuración del clúster creado previamente

ProxySQL



ProxySQL

Características

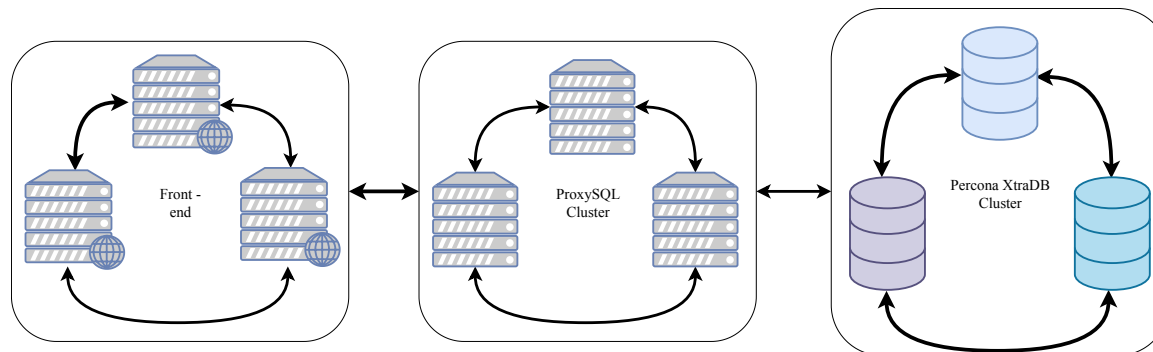
- Caché de queries
- Enrutado de queries
- Tolerancia a fallos
- Alta disponibilidad
- Reescritura de queries
- Sharding
- Comparativa con otros proxy

Opciones de instalación

- Como clúster independiente a los front-end
- En cada máquina del front-end
 - Si el número es estático, podríamos montar el clúster
 - Las conexiones serían en local, pero:
 - Si se crean de manera automática (docker, Amazon ec2,...) tener una configuración de clúster es más complicado

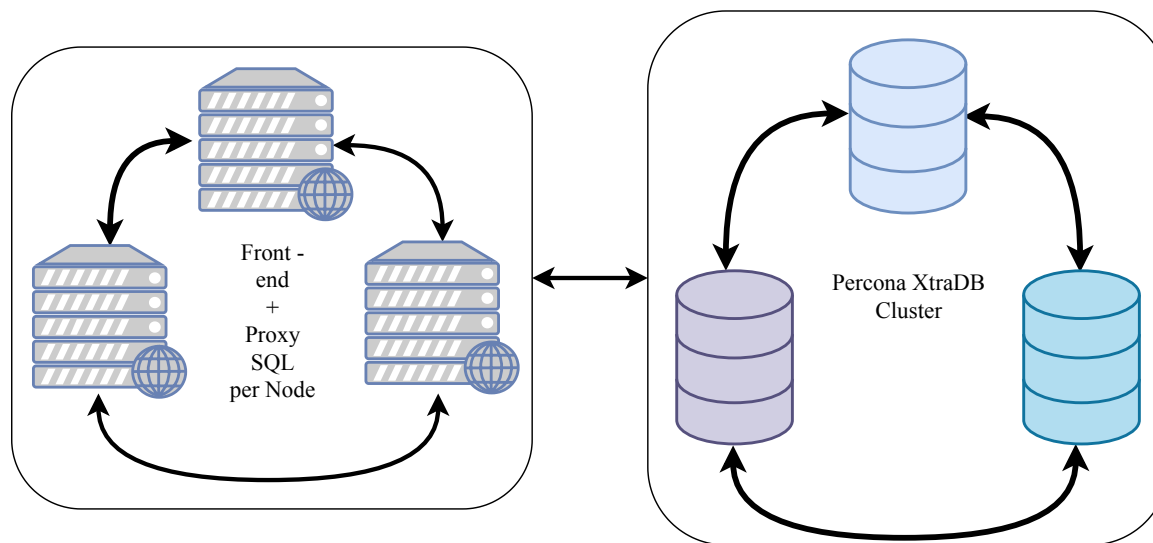
Escenario

Clúster Proxysql entre Front-end y Percona Cluster



Escenario 2

ProxySQL instalado en cada nodo Front-end



Nota: Que ProxySQL sea un clúster depende del número de nodos de frontend

Instalación

- Por **repositorio oficial**:
 - Debian, Ubuntu, CentOS, Fedora
 - Detalles de cada distribución en [Github](#)
- Del repositorio de Percona
 - Por tener el repositorio de Percona podemos instalar **proxysql2**
 - A veces va un poco por detrás

Instalación

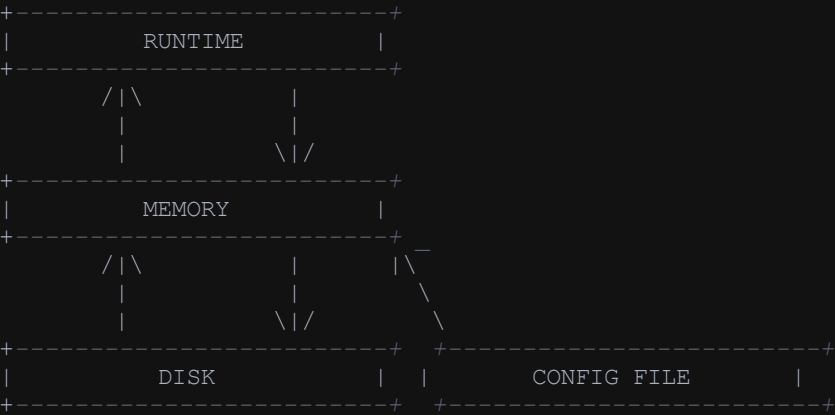
Versión 2.5 desde repositorio **Oficial** para **Debian 9**

```
apt-get install apt-transport-https lsb-release  
  
wget -O - 'https://repo.proxysql.com/ProxySQL/repo_pub_key' | apt-key add -  
  
echo deb https://repo.proxysql.com/ProxySQL/proxysql-2.0.x/$(lsb_release -sc) / ./ \  
| tee /etc/apt/sources.list.d/proxysql.list  
  
apt-get update  
  
apt-get install proxysql
```

Configuración

- Dos alternativas:
 - Fichero de configuración
 - **/etc/proxysql.cnf**
 - Recomendado sólo para la configuración inicial, con poca configuración
 - Interfaz de administración
 - Interfaz *CLI* tipo **mysql**
 - Genera fichero *sqlite3* **/var/lib/proxysql/proxysql.db**

Configuración



Configuración

- ProxySQL puede actuar como:
 - un único servicio en un nodo
 - La configuración es local
 - en clúster
 - cierta configuración se pasa entre los nodos, pero no toda
 - work in progress

Configuración

- **/etc/proxysql.cnf**
 - Viene cierta configuración de serie
 - Mejor empezar de cero con menos
 - Para crear un clúster:

```
admin_variables=  
{  
    admin_credentials="admin:admin;clusteradm:password"  
    mysql_ifaces="0.0.0.0:6032"  
    cluster_username="clusteradm"  
    cluster_password="password"  
}
```


Configuración

- Para indicar los nodos del clúster:

```
proxysql_servers =  
(  
    {  
        hostname="10.0.0.11"  
        port=6032  
        weight=0  
        comment="proxysql1"  
    },  
    {  
        hostname="10.0.0.12"  
        port=6032  
        weight=0  
        comment="proxysql2"  
    },  
    {  
        hostname="10.0.0.13"  
        port=6032  
        weight=0  
        comment="proxysql3"  
    }  
)
```

Arrancar servicio

- Arrancamos servicio nodo a nodo:

```
systemctl start proxysql
```

- En el log `/var/lib/proxysql/proxysql.log` veremos cómo se van añadiendo cada nodo

Conexión y estado

- Nos conectamos en un nodo:

```
mysql -u clusteradm -ppassword -h 127.0.0.1 -P6032 --prompt='Admin> '
```

- Comprobamos el estado:

```
Admin> select * from proxysql_servers;
```

hostname	port	weight	comment
10.0.0.11	6032	0	proxysql1
10.0.0.12	6032	0	proxysql2
10.0.0.13	6032	0	proxysql3

```
3 rows in set (0.00 sec)
```

Crear backends

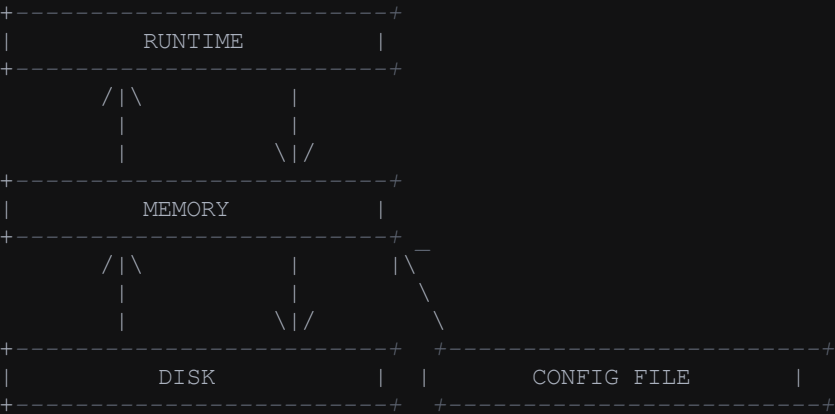
- ProxySQL tiene que conocer a qué backends mandar las peticiones
 - Pueden crearse hostgroups con varios nodos y distinto peso

```
INSERT INTO mysql_servers(hostgroup_id,hostname,port)
VALUES (1,'10.0.0.1',3306);
INSERT INTO mysql_servers(hostgroup_id,hostname,port)
VALUES (1,'10.0.0.2',3306);

INSERT INTO mysql_servers(hostgroup_id,hostname,port,weight)
VALUES (2,'10.0.0.31',3306,10);
INSERT INTO mysql_servers(hostgroup_id,hostname,port,weight)
VALUES (2,'10.0.0.32',3306,1);
```

- Serán monitorizados con el user *monitor*

Configuración



Configuración

- La configuración creada está en **memory**
- Hasta que no está en **runtime** no se aplica y se propaga al resto de los nodos
 - Está en local, sin aplicarse
- Si la configuración no se pasa a **disk** SE PIERDE
- El movimiento entre *capas* está seccionado:
 - mysql users
 - mysql servers
 - mysql query rules
 - mysql variables
 - admin variables

Interactuar con la configuración

- Pasar de **memory** a **runtime**:

```
LOAD MYSQL SERVERS TO RUNTIME;  
LOAD MYSQL USERS TO RUNTIME;
```

- Pasar de **memory** a **disk**:

```
SAVE MYSQL SERVERS TO DISK;  
SAVE MYSQL USERS TO DISK;
```

Interactuar con el backend

- La conexión desde las aplicaciones irán al ProxySQL
- ProxySQL decidirá si le permite el paso y a qué nodo enviar la petición
- Por ello ProxySQL tiene que tener los credenciales de usuarios
 - Los mismos que existen en los backend

Crear usuarios

- Creamos usuario en el backend:

```
CREATE USER 'user'@'%' identified by 'password';  
GRANT ALL PRIVILEGES ON *.* TO 'user'@'%';  
FLUSH PRIVILEGES;
```

- Creamos usuario en el ProxySQL:

```
INSERT INTO mysql_users(username,password,default_hostgroup) VALUES ('user','password',1);  
  
LOAD MYSQL USERS TO RUNTIME;  
SAVE MYSQL USERS TO DISK;
```

Comprobar que funciona

- Hemos hecho que ProxySQL conozca:
 - Servidores backend de MySQL
 - Usuarios a los que permitir la conexión
- Realizamos conexión:

```
mysql -u user -p password -P 6033 -h10.0.0.11
```

```
mysql> use prueba;
```

Comprobar estado de los backend

- Si tiramos un nodo: `` Admin> select hostname,status from runtime_mysql_servers;
- -----+-----+ | hostname | status |
- -----+-----+ | 10.0.0.1 | SHUNNED | | 10.0.0.2 | ONLINE | | 10.0.0.3 | ONLINE |
- -----+-----+ 3 rows in set (0.01 sec)

Estadísticas

- Podemos ver las queries que ha habido y qué nodo las ha respondido

```
Admin> SELECT hostgroup hg, srv_host, Queries, Bytes_data_sent, Bytes_data_recv, Latency_us FROM stats_mysql_connection_pool
        WHERE ConnUsed+ConnFree > 0 ORDER BY hg, srv_host;
```

hg	srv_host	Queries	Bytes_data_sent	Bytes_data_recv	Latency_us
1	10.0.0.1	4	80	47	0
1	10.0.0.2	7	159	101	0
1	10.0.0.3	6	126	72	0

Activar admin web

- activamos la variable

```
SET admin-web_enabled='true';  
LOAD ADMIN VARIABLES TO RUNTIME;
```

- Vamos al navegador: http://IP_nodo:6080
 - credenciales por defecto:
 - user: stats
 - pass: stats

Caché

- ProxySQL permite cachear queries
 - Hay que tener cuidado!
- Las queries se diferencian por un código único:

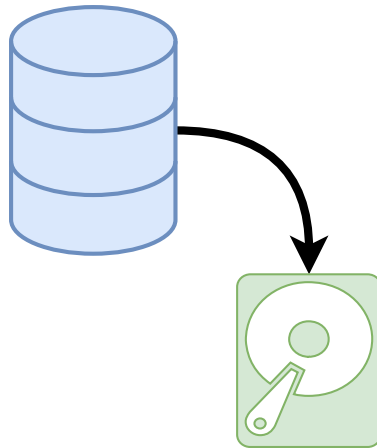
```
SELECT * FROM stats_mysql_query_digest;
```

- Para cachear 2 segundos una query:

```
INSERT INTO mysql_query_rules (rule_id,active,digest,cache_ttl,apply)  
VALUES (1,1,'0xE8930CB2CC9E68D7',2000,1);
```

```
LOAD MYSQL QUERY RULES TO RUNTIME;  
SAVE MYSQL QUERY RULES TO DISK;
```

Backups



Antes de empezar

- Un *backup* es una copia de los datos originales que se realiza con el fin de disponer de un medio para recuperarlos en caso de su pérdida ([Wikipedia](#))
- **Un clúster no es un backup**
 - *Apunte:* Podemos tener un esclavo con N segundos de retraso
- Los backups no sirven de nada si no estamos seguros de si se están realizando
 - *Los backups de Schrödinger*

Antes de empezar

- Tener backups de los datos no garantiza la restauración de los mismos si no tenemos de la infraestructura
- Tenemos que tener un *DRP* y probarlo cada cierto tiempo
 - *Disaster Recovery Plan*: Plan de prevención ante desastres
- Tenemos que tener claro el *RPO* y *RTO*:
 - *Recovery Point Objective*: el periodo máximo de datos que podemos perder
 - *Recovery Time Objective*: tiempo de recuperación del servicio

Tipos de backup

- Completo
 - Por servidor
 - Por base de datos/schema
 - Por tabla
 - Físicos / Lógicos
- Incrementales
- Sobre nodo esclavo
- Esclavo con retraso
 - ¿Lo consideramos backup?

Backups completos

- Físico:
 - Copias en bruto de los directorios y ficheros con los contenidos
 - Sólo puede portarse entre máquinas con idénticas versiones
 - Debe realizarse con MySQL **PARADO** (o bloqueado)
 - mysqlhotcopy
 - snapshots (LVM, ZFS, ...)
 - cp
 - **NO RECOMENDABLE**

Backups Completos

- Lógico
 - La información se salva representada como la estructura lógica de la base de datos
 - Se hace guardando la estructura de la base de datos y los datos
 - Es portable entre versiones (sentencias SQL)
 - Debe realizarse con el servidor en ejecución
 - mysqldump
 - Percona XtraBackup

MySQLdump

- Script oficial de MySQL
- Genera ficheros de volcado (dump)
 - backup para recuperación
 - origen de datos para recuperación
 - soporta todos los engines
 - Formato SQL
 - texto plano

MySQLdump

- Opciones a tener en cuenta
 - **–single-transaction**: para hacer backup de tablas **InnoDB**
 - crea una transacción antes de realizar el backup
 - **necesario para XtraDB Cluster**
 - **–lock-all-tables**: para realizar backups de MyISAM
 - **–no-data**: sólo la estructura de tablas
 - ... : hay muchas opciones, lee el manual ;-)

MySQLdump

Crear backups

```
mysqldump --all-databases > dump_all.sql  
  
mysqldump --databases db1 db2 db3 > dump_dbs.sql  
  
mysqldump db1 > dump_db1.sql  
  
mysqldump db1 table1 table2 > dump_db1_tables.sql
```

Restaurar backup

```
mysql < dump_all.sql  
  
mysql < dump_dbs.sql  
  
mysql db1 < dump_db1.sql  
  
mysql db1 < dump_db1_tables.sql
```

Ejercicio

- Crear backups de:
 - Todas las bases de datos
 - Una única base de datos
 - Una única tabla
- Realizar restauración:
 - sobre una base de datos distinta
 - sobre una tabla distinta

Percona XtraBackup

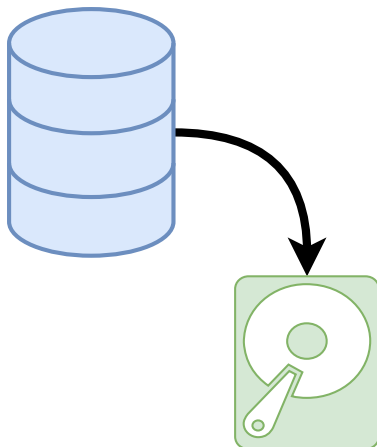


XtraBackup

Características

- Open Source
- Backups en caliente
 - no bloquea la base de datos
- Backups incrementales
- Backups comprimidos
- Para InnoDB, XtraDB y MyISAM
 - en MySQL > 5.1

XtraBackup



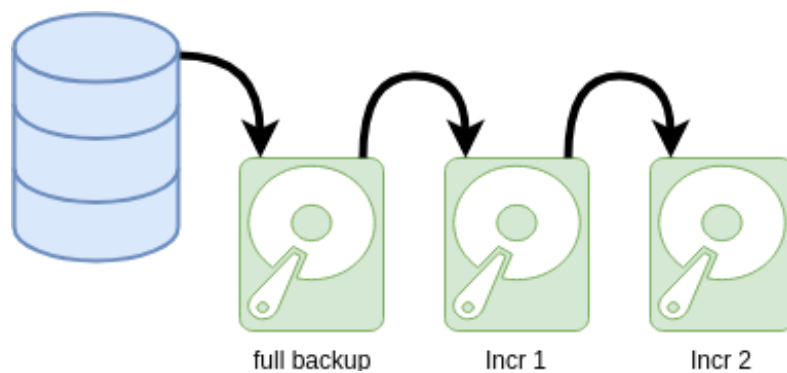
- hacer backup completo

```
xtrabackup --backup --user=root -p  
xtrabackup --prepare --target-dir=xtrabackup_backupfiles/
```

- restaurar backup
 - **/var/lib/mysql** del destino tiene que estar vacío

```
xtrabackup --copy-back --target-dir=xtrabackup_backupfiles/
```

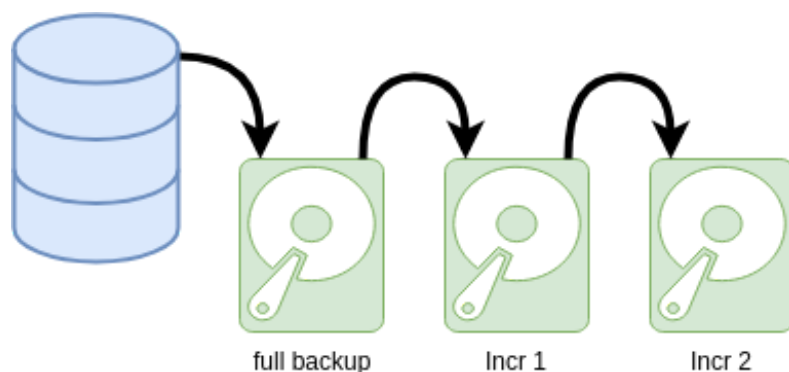
XtraBackup



- hacert backup incremental

```
xtrabackup --backup --user=root -p  
  
xtrabackup --backup --target-dir=incr1 \  
  --incremental-basedir=xtrabackup_backupfiles/  
  
xtrabackup --backup --target-dir=incr2 \  
  --incremental-basedir=incr1/
```

XtraBackup



- Consolidar backups incrementales

```
xtrabackup --prepare --apply-log-only --target-dir=xtrabackup_backupfiles/  
  
xtrabackup --prepare --apply-log-only --target-dir=xtrabackup_backupfiles/ \  
--incremental-dir=incr1  
  
xtrabackup --prepare --target-dir=xtrabackup_backupfiles/ \  
--incremental-dir=incr2
```

- restaurar backup

```
xtrabackup --copy-back --target-dir=xtrabackup_backupfiles/
```

Ejercicio

- Realizar backup con XtraBackup
- Realizar modificaciones en la base de datos
- Crear backup incremental

Docker



Ventajas

- Nos permite probar nuevas versiones de manera sencilla
 - Probar compatibilidades
- Tener distintas versiones en puertos distintos
- Desplegar de manera sencilla

Ejemplo

```
$ docker run -e MYSQL_ROOT_PASSWORD=1234 -d mysql:8.0
```

```
Unable to find image 'mysql:8.0' locally  
8.0: Pulling from library/mysql  
...
```

```
$ docker exec -ti 28f1563d1397 bash
```

```
root@28f1563d1397:/# mysql -hlocalhost -p
```

Ejemplo 2

```
docker run --name my8 -e MYSQL_ROOT_PASSWORD=1234 -d mysql:8.0  
docker run --name my5.6 -e MYSQL_ROOT_PASSWORD=1234 -d mysql:5.6  
docker run --name my5.7 -e MYSQL_ROOT_PASSWORD=1234 -d mysql:5.7
```

Ejemplo 3

- Podemos hacer que el puerto sea expuesto en el host anfitrión

- `-p 3370:3306`

- Podemos hacer el volumen de datos persistente

- `-v /opt/data-my_5.6:/var/lib/mysql`

Ejercicio

Crear dos contenedores con versiones distintas de MySQL

Ejercicio

Todo lo que se ha realizado en este curso, de una manera u otra, se puede realizar bajo Docker

Documentación

Replicación

- [Documentación MySQL 8.0 Oficial](#)
- [Percona Blog](#)

Percona XtraDB Cluster

- Instalación

Cluster Control

- Instalación
- Creación de cluster
- Tutoriales oficiales

ProxySQL

- Configuración

Backup

- MySQL dump
- Xtrabackup full backup
- Xtrabackup incremental backup

Docker

- MySQL con Docker Percona en Docker Hub
- ProxySQL con Docker

Gracias!

Foto portada @bachmont