



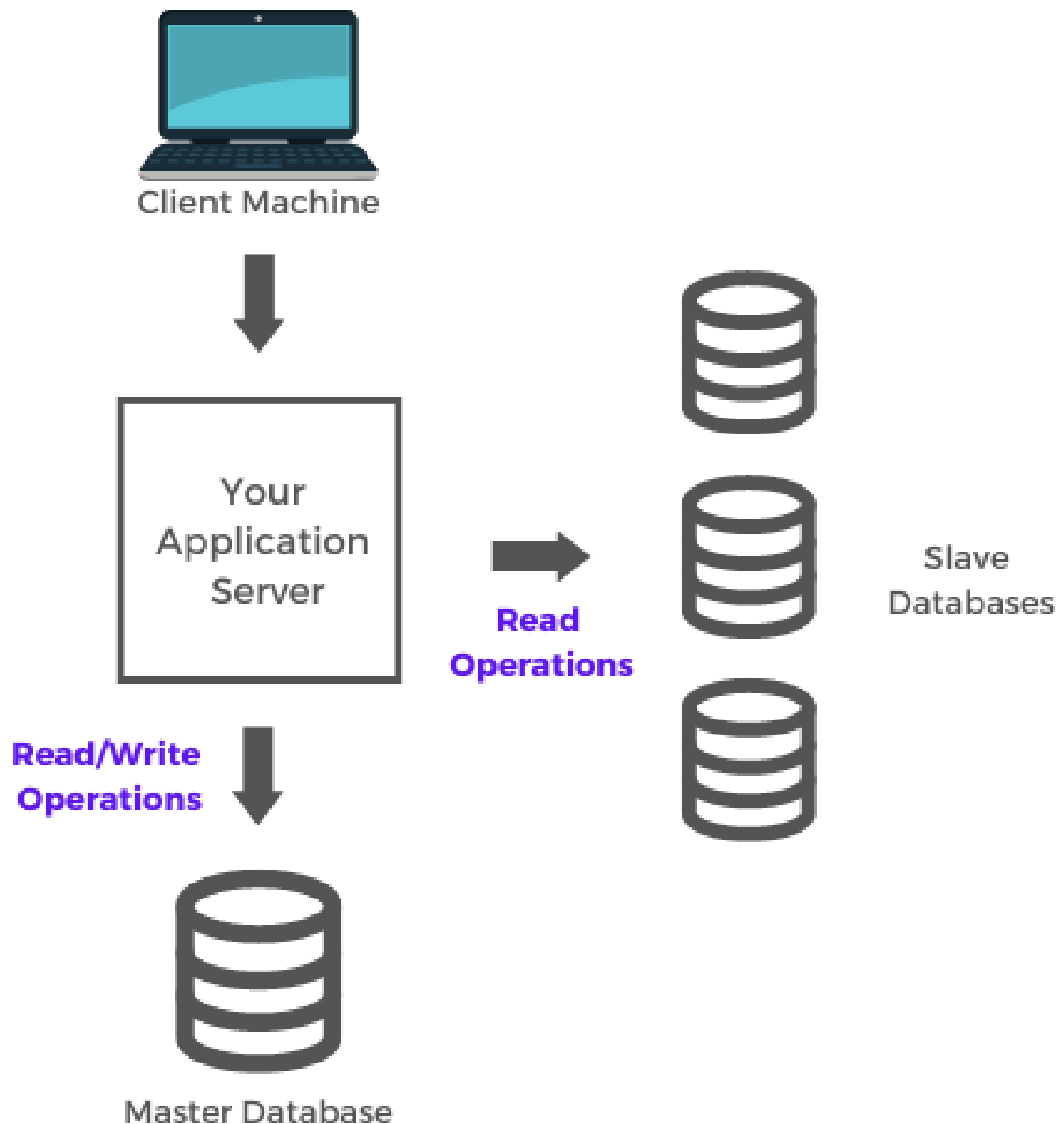
+

MySQL

Setup MySQL Group Replication in Kubernetes

Setting up MySQL group replication is a very important task when it comes to the modern data-hungry world because most applications are depending on databases. It is always challenging to deploy and maintain a stateful service like a database on a Kubernetes cluster, but doing so enables high availability, data redundancy, and significant performance gains.

Here we will use the traditional master/slave architecture for the database. In this architecture master database is actually the place where all the writing requests are performed and the reading operations are spread across multiple slave databases relative to the master database.



Prerequisites:

- You should have a Kubernetes Cluster.
- You should configure Kubernetes CLI (kubectl) in your local machine.

Follow me 😊

1. Create the ConfigMap from the following YAML configuration file. This allows the master server to be able to serve replication logs to slave servers and slave servers to reject any writes that don't come via replication.



mysql-configmap.yml

GitHub Gist: instantly share code, notes, and snippets.

Gist

```
kubectl create -f mysql-configmap.yml
```

2. Create the services from the following YAML configuration files.



mysql-service.yml

GitHub Gist: instantly share code, notes, and snippets.

Gist

```
kubectl create -f mysql-service.yml
```

We can use this "mysql" service to access each pod. Within the same namespace, we can do it by resolving "<pod_name>.mysql".

When we need to write data to the database, the only way we can do it is by using the master server which is the "mysql-0" pod (You can see that pod after the 3rd step). You can access it by resolving "mysql-0.mysql".



mysql-read-service.yml

GitHub Gist: instantly share code, notes, and snippets.


Gist

```
kubectl create -f mysql-read-service.yml
```

This "mysql-read" service is used to read the data from the database. We can't write data using this service.

When we need to read data from the database, instead of connecting to the "mysql" service or master server pod, we should use this "mysql-read" service.

3. create the StatefulSet from the following YAML configuration file.

 **GitHub Gist**

mysql-statefulset.yml

GitHub Gist: instantly share code, notes, and snippets.

Gist

```
kubectl create -f mysql-statefulset.yml
```

Now you can see the running pods like the following image. You can see only 3 pods because we mentioned 3 as the number of replicas in our statefulset.

```
[krishanshamod@ks-laptop ~]$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
mysql-0	2/2	Running	0	8h
mysql-1	2/2	Running	0	8h
mysql-2	2/2	Running	0	8h

The first pod which is "mysql-0" is always the master server. The remaining pods are slaves. You can increase the number of slaves by updating the "replicas: " of this statefulset.

4. Access the bash shell of "mysql-0" pod container.

```
kubectl exec -it mysql-0 -- /bin/bash
```

```
[krishanshamod@ks-laptop ~]$ kubectl exec -it mysql-0 -- /bin/bash
Defaulted container "mysql" out of: mysql, xtrabackup, init-mysql (init), clone-mysql (init)
bash-4.2#
bash-4.2# █
```

5. Log in to MySQL CLI using this command.

```
mysql -u root -p
```

Then it will ask for the root password. For now, it's empty.

```
bash-4.2# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 18309
Server version: 5.7.39-log MySQL Community Server (GPL)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql>
```

6. Update the MySQL root password (These kinds of changes will automatically apply to all slaves as well).

```
ALTER USER 'root'@'localhost' IDENTIFIED BY 'myPassword';
```

```
FLUSH PRIVILEGES;
```

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'myPassword';  
Query OK, 0 rows affected (0.02 sec)  
  
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.01 sec)
```

7. Create the databases you need. Here I'm creating a database called "lms".

```
CREATE DATABASE lms;
```

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

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| lms |
| mysql |
| performance_schema |
| sys |
| xtrabackup_backupfiles |
+-----+
6 rows in set (0.00 sec)
```

8. Restore the data using a SQL dump. If you don't need to restore old data, then skip this step.

First, copy the SQL dump file from your local machine to the "mysql-0" pod container using this command. You need to execute this command from a new bash shell. Not previous MySQL CLI.

```
kubectl cp local_path_to_the_dump mysql-0:pod_container_path
```

Note: Update the both “**local_path_to_the_dump**” and “**pod_container_path**”.

```
[krishanshamod@ks-laptop lms]$ kubectl cp ./database.sql mysql-0:/database.sql
Defaulted container "mysql" out of: mysql, xtrabackup, init-mysql (init), clone-mysql (init)
[krishanshamod@ks-laptop lms]$
```

Again login to the MySQL root account using the new password (Repeat steps 4 and 5). Then change the database to the newly created database. I choose lms.

```
USE lms;
```

```
mysql> USE lms;
Database changed
mysql>
mysql>
```

Now restore the data using the dump we copied from the local machine.

```
source ./database.sql
```

```
mysql> source ./database.sql
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

9. Add a new MySQL user with new database remote read/write access. I'm granting access to the lms database I created.

```
CREATE USER 'krishan'@'%' IDENTIFIED BY 'myPassword';

GRANT ALL PRIVILEGES ON lms.* TO 'krishan'@'%';

FLUSH PRIVILEGES;
```

```
mysql> CREATE USER 'krishan'@'%' IDENTIFIED BY 'myPassword';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT ALL PRIVILEGES ON lms.* TO 'krishan'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

Now all MySQL pods are up and running. Databases and data are restored. Created new user with remote read/write access. Everything is good to go.

As I mentioned earlier, your application should use the following database URL to write data (Replace “lms” with your database name).

```
mysql-0.mysql:3306/lms
```

Also, you should use the following database URL to read data (Replace “lms” with your database name).

```
mysql-read:3306/lms
```