MySQL on Docker: Swarm Mode Limitations for Galera Cluster in Production Setups

In the last couple of blog posts on Docker, we have looked into understanding and running Galera Cluster on Docker Swarm. It scales and fails over pretty well, but there are still some limitations that prevent it from running smoothly in a production environment. We will be discussing about these limitations, and see how we can overcome them. Hopefully, this will clear some of the questions that might be circling around in your head.

Docker Swarm Mode Limitations

Docker Swarm Mode is tremendous at orchestrating and handling stateless applications. However, since our focus is on trying to make Galera Cluster (a stateful service) to run smoothly on Docker Swarm, we have to make some adaptations to bring the two together. Running Galera Cluster in containers in production requires at least:

- Health check - Each of the stateful containers must pass the Docker health checks, to ensure it achieves the correct state before being included into the active load balancing set.
- Data persistency - Whenever a container is replaced, it has to be started from the last known good configuration. Else you might lose data.
- Load balancing algorithm - Since Galera Cluster can handle read/write simultaneously, each node can be treated equally. A recommended balancing algorithm for Galera Cluster is least connection. This algorithm takes into consideration the number of current connections each server has. When a client attempts to connect, the load balancer will try to determine which server has the least number of connections and then assign the new connection to that server.

We are going to discuss all the points mentioned above with a great detail, plus possible workarounds on how to tackle those problems.

Health Check

HEALTHCHECK is a command to tell Docker how to test a container, to check that it is still working. In Galera, the fact that mysqld is running does not mean it is healthy and ready to serve. Without a proper health check, Galera could be wrongly diagnosed when something goes wrong, and by default, Docker Swarm's ingress network will include the "STARTED" container into the load balancing set regardless of the Galera state. On the other hand, you have to manually attach to a MySQL container to check for various MySQL statuses to determine if the container is healthy.

With HEALTHCHECK configured, container healthiness can be retrieved directly from the standard "docker ps" command:

```
$ docker ps
CONTAINER ID        IMAGE                       COMMAND             CREATED             STATUS                      PORTS
42f98c8e0934        severalnines/mariadb:10.1   "/entrypoint.sh "   13 minutes ago      Up 13 minutes (healthy)     3306/tcp, 4567-4568/tcp
```

Plus, Docker Swarm's ingress network will include only the healthy container right after the health check output starts to return 0 after startup. The following table shows the comparison of these two behaviours:

Options	Sample output	Description
Without HEALTHCHECK	*Hostname: db_mariadb_galera.2* *Hostname: db_mariadb_galera.3* *Hostname: ERROR 2003 (HY000): Can't connect to MySQL server on '192.168.1.100' (111)* *Hostname: db_mariadb_galera.1* *Hostname: db_mariadb_galera.2* *Hostname: db_mariadb_galera.3* *Hostname: ERROR 2003 (HY000): Can't connect to MySQL server on '192.168.1.100' (111)* *Hostname: db_mariadb_galera.1* *Hostname: db_mariadb_galera.2* *Hostname: db_mariadb_galera.3* *Hostname: db_mariadb_galera.4* *Hostname: db_mariadb_galera.1*	Applications will see an error because container db_mariadb_galera.4 is introduced incorrectly into the load balancing set. Without HEALTHCHECK, the STARTED container will be part of the "active" tasks in the service.
With HEALTHCHECK	*Hostname: db_mariadb_galera.1* *Hostname: db_mariadb_galera.2* *Hostname: db_mariadb_galera.3* *Hostname: db_mariadb_galera.1* *Hostname: db_mariadb_galera.2* *Hostname: db_mariadb_galera.3* *Hostname: db_mariadb_galera.4* *Hostname: db_mariadb_galera.1* *Hostname: db_mariadb_galera.2*	Container db_mariadb_galera.4 is introduced correctly into the load balancing set. With proper HEALTHCHECK, the new container will be part of the "active" tasks in the service if it's marked as healthy.

The only problem with Docker health check is it only supports two exit codes - either 1 (unhealthy) or 0 (healthy). This is enough for a stateless application, where containers can come and go without caring much about the state itself and other containers. With a stateful service like Galera Cluster or MySQL Replication, another exit code is required to represent a staging phase. For example, when a joiner node comes into the picture, syncing is required from a donor node (by SST or IST). This process is automatically started by Galera and probably requires minutes or hours to complete, and the current workaround for this is to configure *[--update-delay]* and *[--health-interval * --health-retires]* to higher than the SST/IST time.

For a clearer perspective, consider the following "service create" command example:

```
$ docker service create \
--replicas=3 \
--health-interval=30s \
--health-retries=20 \
--update-delay=600s \
--name=galera \
--network=galera_net \
severalnines/mariadb:10.1
```

The container will be destroyed if the SST process has taken more than 600 seconds. While in this state, the health check script will return "exit 1 (unhealthy)" in both joiner and donor containers because both are not supposed to be included by Docker Swarm's load balancer since they are in syncing stage. After failures for 20 consecutive times at every 30 seconds (equal to 600 seconds), the joiner and donor containers will be removed by Docker Swarm and will be replaced by new containers.

It would be perfect if Docker's HEALTHCHECK could accept more than exit "0" or "1" to signal Swarm's load balancer. For example:

- exit 0 => healthy => load balanced and running
- exit 1 => unhealthy => no balancing and failed
- ***exit 2 => unhealthy but ignore => no balancing but running***

Thus, we don't have to determine SST time for containers to survive the Galera Cluster startup operation, because:

- Joiner/Joined/Donor/Desynced == exit 2
- Synced == exit 0
- Others == exit 1

Another workaround apart setting up *[--update-delay]* and *[--health-interval * --health-retires]* to be higher than SST time is you could use HAProxy as the load balancer endpoints, instead of relying on Docker Swarm's load balancer. More discussion further on.