Getting started Galera with Docker, part 1

by Erkan Yanar

Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications. Consisting of Docker Engine, a portable, lightweight runtime and packaging tool, and Docker Hub, a cloud service for sharing applications and automating workflows, Docker enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments. As a result, IT can ship faster and run the same app, unchanged, on laptops, data center VMs, and any cloud.

This is the first of a series of blog posts about using Galera with Docker. In this post, we are going to get started with Docker and Galera:

* Build a basic Docker Image (which we will extended in later posts)

* Deploy on a test cluster on a local machine

The instructions have been tested on Ubuntu 14.04 with Docker 1.5.

BUILD A BASIC DOCKER IMAGE

In Docker, Dockerfiles are used to describe the Docker images we are going to use to start our Galera Cluster. We are using the following Dockerfile:

```
FROM ubuntu:14.04
MAINTAINER Erkan Yanar <erkan.yanar@linsenraum.de>
ENV DEBIAN_FRONTEND noninteractive

RUN apt-get update
RUN apt-get install -y software-properties-common
RUN apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 BC19DDBA
RUN add-apt-repository 'deb http://releases.galeracluster.com/ubuntu trusty main'
RUN apt-get update
RUN apt-get install -y galera-3 galera-arbitrator-3 mysql-wsrep-5.6 rsync lsof
COPY my.cnf /etc/mysql/my.cnf
ENTRYPOINT ["mysqld"]
```

This image builds on top of the Ubuntu 14.04 image. It simply installs Galera using the Codership repository and copies the my.cnf over.

The `my.cnf` is quite simple.

```
[mysqld]
user = mysql
bind-address = 0.0.0.0
wsrep_provider = /usr/lib/galera/libgalera_smm.so
wsrep_sst_method = rsync
default_storage_engine = innodb
binlog_format = row
innodb_autoinc_lock_mode = 2
innodb_flush_log_at_trx_commit = 0
query_cache_size = 0
query_cache_type = 0
```

A pre-built image is available from Docker Hub. You can pull it by running:

```
sudo docker pull erkules/galera:basic
```

(All commands in this article need to run as root.)

DEPLOY ON A TEST CLUSTER ON A LOCAL MACHINE

Next, we are going to start a Galera Cluster on the local host. The instructions below are for demonstration purposes only and will not work when deploying on multiple hosts, as networking between containers needs to be set up. Configuring Docker networking across multiple hosts will be described in a following post.

STARTING A CLUSTER

There have been a number of blog posts showing how to start Galera Cluster on a single host. This post is going to show the simplest way to do that in Docker by using simple commands, which will not work for a multi-host installation. First, if working on Ubuntu, we need to put AppArmor's Docker profile in complain mode in advance.

```
$ sudo aa-complain /etc/apparmor.d/docker
```

Then we can start the first Galera node by instructing Docker to create a container and run mysqld in it.

```
$ sudo docker run --detach=true --name node1 -h node1 erkules/galera:basic --wsrep-cluster-name=local-test --wsrep-c
```

In addition to defining the internal name and hostname, we also define the name of the cluster.

MySQLs error log is not configured explicitly, and Docker records STDOUT and STDERR of every container. So, using `sudo docker logs node1`, the log output from the first node can be displayed without having to enter the container.

For the next two containers, we use a simple Docker trick. The `-link` option writes the name and the IP of host1 into the `/etc/hosts` file of the container. This way, we can connect the remaining nodes to node1 without having to obtain its IP from its container.

```
$ sudo docker run --detach=true --name node2 -h node2 --link node1:node1 erkules/galera:basic --wsrep-cluster-name=l
$ sudo docker run --detach=true --name node3 -h node3 --link node1:node1 erkules/galera:basic --wsrep-cluster-name=l
```

Now we have a running Galera cluster. We can check the number of nodes in the Cluster by running the mysql client from inside one of the containers:

```
$ sudo docker exec -ti node1 mysql -e 'show status like "wsrep_cluster_size"'
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size |     3 |
+--------------------+-------+
```

WHAT DID WE DO?

We built a simple Galera Cluster on one host.

* Without using SSH;
* Without the need to configure any IP addresses;

Note that this setup does not support restarting the container — you should remove the container and recreate it instead.

In the next blog post we will describe deploying Galera with Docker on multiple hosts.