Deploying MongoDB Using Docker

Akash Kathiriya
December 13, 2018

Posted in: Deployment & Scaling MongoDB

The main advantage of using MongoDB is that it's easy to use. One can easily install MongoDB and start working on it in minutes. Docker makes this process even easier.

One cool thing about Docker is that, with very little effort and some configuration, we can spin up a container and start working on any technology. In this article, we will spin up a MongoDB container using Docker and learn how to attach the storage volume from a host system to a container

We will only need Docker installed in the system for this tutorial.

Creating a MongoDB Image

First create a folder and create a file with the name Dockerfile inside that folder:

```
$ mkdir mongo-with-docker
$ cd mongo-with-docker
$ vi Dockerfile
```

Paste this content in your Dockerfile:

```
FROM debian:jessie-slim
RUN apt-get update && \
apt-get install -y ca-certificates && \
rm -rf /var/lib/apt/lists/*
RUN gpg --keyserver ha.pool.sks-keyservers.net --recv-keys 0C49F3730359A1451858593
gpg --export $GPG_KEYS > /etc/apt/trusted.gpg.d/mongodb.gpg
ARG MONGO_PACKAGE=mongodb-org
ARG MONGO_REPO=repo.mongodb.org
ENV MONGO_PACKAGE=${MONGO_PACKAGE} MONGO_REPO=${MONGO_REPO}
ENV MONGO_MAJOR 3.4
ENV MONGO_VERSION 3.4.18
RUN echo "deb http://$MONGO_REPO/apt/debian jessie/${MONGO_PACKAGE%-unstable}/$MON
RUN echo "/etc/apt/sources.list.d/${MONGO_PACKAGE%-unstable}.list"
```

```
docker build -t hello-mongo:latest .
```

Understanding the Docker File Content

The structure of each line in docker file is as follows:

```
INSTRUCTIONS arguments
```

- FROM: Base image from which we'll start building the container
- RUN: This commands executes all instructions to install MongoDB in the base image.
- ARG: Stores some default values for the Docker build. These values are not available to the container. Can be overridden during the building process of the image using the `--build-arg` argument.

- VOLUME: Attaches the `data/db` volume to container.
- WORKDIR: Sets the work directory to execute any RUN or CMD commands.
- EXPOSE: Exposes the container's port to host the system (outside world).
- CMD: Starts the mongod instance in the container.

Starting the MongoDB Container From the Image

You can start the MongoDB container by issuing the following command:

```
1 | docker run --name my-mongo -d -v /tmp/mongodb:/data/db -p 27017:27017 hello-mongo
```

volume of the container.

- `-p`: Map the host port to the container port.
- Last argument is the name/id of the image.

To check whether the container is running or not, issue the following command:

```
1 docker ps
```

Output of this command should look like the following:

```
1 CONTAINER ID        IMAGE               COMMAND                  CREATED
2 a7e04bae0c53        hello-mongo         "mongod --smallfiles"    7 seconds ago
```

Accessing MongoDB From the Host

Once the container is up and running, we can access it the same way as accessing the remote MongoDB instance. You can use any utility like Compass or

It will open mongo shell where you can execute any mongo commands. Now we'll create one database and add some data in it.

```
use mydb
db.myColl.insert({“name”: “severalnines”})
quit()
```

Now to check whether our volume mapping is correct or not, we will restart the container and check whether it has our data or not.

```
Docker restart <container_id>
```

Now again connect to mongo shell and run this command:

```
db.myColl.find().pretty()
```

You should see this result:

```
{ "_id" : ObjectId("5be7e05d20aab8d0622adf46"), "name" : "severalnines" }
```

from the host `tmp/mongodb` directory.

Accessing MongoDB Container Shell

```
1 $ docker exec -it <container-name> /bin/bash
```

Accessing MongoDB Container Logs


```
1 $ docker logs <container-name>
```

Connecting to the MongoDB Container From Another Container

Where Alias is an alias for link name. Run this command to link our Mongo container with express-mongo container.

```
1 docker run --link my-mongo:mongo -p 8081:8081 mongo-express
```

This command will pull the mongo-express image from dockerhub and start a new container. Mongo-express is an admin UI for MongoDB. Now go to http://localhost:8081 to access this interface.

Technology

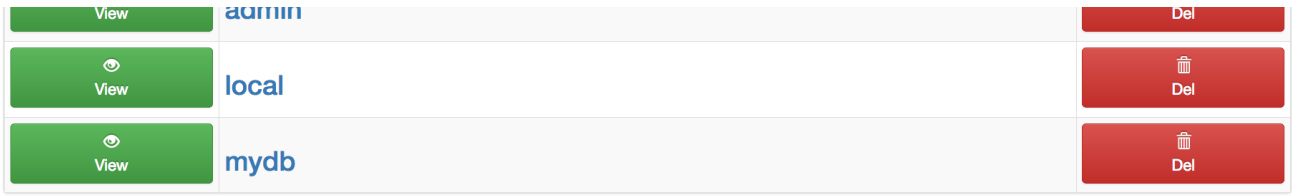
ClusterControl for MongoDB

Blog

Optimizing Your Linux Environment for MongoDB

Blog

How to Deploy MongoDB for High Availability



Server Status

Hostname	a7e04bae0c53	**MongoDB Version**	3.4.18
Uptime	1391 seconds	**Server Time**	Sun, 11 Nov 2018 08:26:53 GMT
Current Connections	3	**Available Connections**	838857
Active Clients	9	**Queued Operations**	0
Clients Reading	1	**Clients Writing**	0
Read Lock Queue	0	**Write Lock Queue**	0
Disk Flushes		**Last Flush**	
Time Spent Flushing	ms	**Average Flush Time**	ms
Total Inserts	0	**Total Queries**	1
Total Updates	0	**Total Deletes**	0

Mongo-express Admin UI

Docker eases the process of deploying multiple MongoDB instances. We can use the same MongoDB image to build any number of containers which can be used for creating Replica Sets. To make this process even smoother, we can write a YAML file (configuration file) and use docker-compose utility to deploy all the containers with the single command.

〈 Previous blog

Next blog 〉