Instruction Sheet

Welcome to the Docker workshop.

Directors: Jude Zhu, Kevin Liew

Teaching assistants: Majin Du, Michael Wang, TIan Shao

Ensure that you have a copy of the Ubuntu VM before we start. This VM comes with pre-installations and all necessary workshop files.

Part 1: Run MongoDB in Docker Containers!

Check the VM.

1. start Docker and ensure that Docker 1.8 is installed:

- [ ] Check docker installed.

```
[employee@centos7-ws ~]$  docker --version
Docker version 1.8.3, build f4bf5c7
```

- [ ] Make sure provisioning files are there.

```
[employee@centos7-ws ~]$ ls
Desktop  Documents  Downloads  Simba-TechEd-2015-Docker-Workshop
```

Create an MongoDB image using docker commit

1. Start an interactive terminal in the container

- [ ] Check what images are available to you on the VM

```
[employee@centos7-ws ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
VIRTUAL SIZE
registry            latest              8d5547a9f329        9 days ago
422.8 MB
ubuntu              14.04               1d073211c498        11 days ago
187.9 MB
centos              6.6                 bec9806dbc09        2 weeks ago
202.6 MB
```

- [ ] Launch a docker container running bash. This terminal is now attached to the container

```
[employee@centos7-ws ~]$  docker run -it --name mongodb1 centos:6.6 bash
[root@7205e7b9ef05 /]# cat /etc/*-release
CentOS release 6.6 (Final)
[root@2be2ffd97e4f /]# ls
bin  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  sbin
selinux  srv  sys  tmp  usr  var
```

2. Install MongoDB

- [ ] Using "Ctrl+Shift+T" to open a second terminal, and check the running containers.

```
[employee@centos7-ws ~]$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS                      NAMES
1792a7a8cd79        centos:6.6          "bash"              3 minutes ago
Up 3 minutes        0.0.0.0:27017->27017/tcp   mongodb1
```

- [ ] copy the MongoDB 3.0.5 installation files into the *mongodb1* container

```
[employee@centos7-ws ~]$ docker cp
~/Simba-TechEd-2015-Docker-Workshop/Part1/provision mongodb1:/provision

[employee@centos7-ws ~]$ docker cp
~/Simba-TechEd-2015-Docker-Workshop/Part1/provision/etc mongodb1:/
```

- [ ] Switch back to the terminal attached to the container, and install MongoDB using

```
[root@1792a7a8cd79 /]# ls /provision/tmp
mongodb-linux-x86_64-3.0.5.tgz

[root@7205e7b9ef05 /]#  yum install tar -y
[root@7205e7b9ef05 /]#  cd /provision/tmp
[root@7205e7b9ef05 /]#  tar -zxvf mongodb-linux-x86_64-3.0.5.tgz
[root@7205e7b9ef05 /]#  cp -r mongodb-linux-x86_64-3.0.5/bin /usr

[root@7205e7b9ef05 /]#  mkdir -p /data/db
[root@7205e7b9ef05 /]#  mkdir -p /data/configdb
```

3. Start a MongoDB server and load sample data

- [ ] start a MongoDB server in the container

```
[root@1792a7a8cd79 tmp]# mongod
```

- [ ] import data into the container

```
[employee@centos7-ws]$ docker exec -it mongodb1 mongoimport
/provision/data/emp.json
[employee@centos7-ws]$ docker exec -it mongodb1 mongo
MongoDB shell version: 3.0.5
connecting to: test
Server has startup warnings:
2015-11-03T19:51:06.442+0000 I CONTROL  [initandlisten] ** WARNING: You are
running this process as the root user, which is not recommended.
2015-11-03T19:51:06.442+0000 I CONTROL  [initandlisten]
2015-11-03T19:51:06.476+0000 I CONTROL  [initandlisten]
2015-11-03T19:51:06.476+0000 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2015-11-03T19:51:06.476+0000 I CONTROL  [initandlisten] **        We suggest
setting it to 'never'
2015-11-03T19:51:06.476+0000 I CONTROL  [initandlisten]
2015-11-03T19:51:06.476+0000 I CONTROL  [initandlisten] ** WARNING:
/sys/kernel/mm/transparent_hugepage/defrag is 'always'.
2015-11-03T19:51:06.476+0000 I CONTROL  [initandlisten] **        We suggest
setting it to 'never'
2015-11-03T19:51:06.476+0000 I CONTROL  [initandlisten]
>
```

4. Build a MongoDB Image using Docker commit

You now have a container that is differentiated from the base centos:6.6 image.

- [ ] View a diff of the container from the image

```
$ docker diff mongodb1
```

- [ ] Stop the container and commit the container as an image named '*monbodbimg*' with the tag '*3.0.5*'

```
$ docker stop mongodb1
$ docker commit mongdb1 mongodb:3.0.5
```

Note that changes made to a container are not reflected in the image unless you commit the container as an image.

Create an image using a Dockerfile

- [ ] Write a Dockerfile to build a MongoDB version 3.0.7 using the reference sheet. Remember to expose the ports 27017 to 27019
- [ ] Build an image named '*mongodb*' with the tag '*3.0.7*'

```
$ docker build -t mongodb:3.0.7 .
```

- [ ] Run a container using this image and the *mongod* process. Remember to publish the 27017 port so that the *mongoimport* tool can connect

Share your docker image with other people

We cannot access Simba's private Docker repository during the workshop. In the office, we can use *docker push* to push images (not containers) to the repository.

Part 2: Deploy a MongoDB Cluster!

We will use the mongo image we built in part1 to create a mongodb cluster, in this part.

Start SkyDNS

- [ ] Start via

```
$ ./start_dns.sh
```

Run a MongoDB sharded cluster in Docker containers

This section serves as a demo of Docker's capabilities and can be used as an example to implement clusters for other databases as Simba.

- [ ] Start the mongodb cluster.

```
$ ./start_cluster.sh
```

Connect WebApp with MongoDB Cluster

We have a prepared a webapp. This webapp's front-end is written using Angular.js. The backend is done with Node.js. Database will be the mongodb Cluster. After the database cluster is started, you can start the webapp we prepare, and set up the web app to use the mongo cluster as its database. This will not work at the very beginning, by default. You need to following the hints below to get the webapp + mongodb cluster work.

- [ ] Configure the mongodb in the webapp

```
$ cd webapp/server/config
$ vi config.js

# replace your mongo query server ip address or hostname
```

- [ ] Configure the frontend api address

```
$ cd webapp/frontend/js/factories
$ vi api.js

# replace your server ip address or hostname
```

Start WebApp

- [ ]

- [ ] Start the server container

  ```
  $ cd webapp/deploy
  $ sh lauch_server.sh
  ```

- [ ] Start the server

  ```
  $ docker exec -it server bash
  $ cd /root/webapp/server
  $ node server.js
  ```

- [ ] Open you browser, to check the app!

Bonus Part

- [ ] Add one more shard in your mongodb cluster.
- [ ] Add one more replicate node in shards. And try to stop one of them in the shards. See what happened.