

Cloud Computing – Assignment 2

(Docker MongoDB Installation)

Group Members: Saurav Nanda and Mengjie Wang

Report

VM Configurations

1. **Docker VM:** 2556 MB RAM, 1 CPU, 60GB Storage and Ubuntu OS

Network configuration:

```
root@docker:/home/saurav# ifconfig
docker0    Link encap:Ethernet  HWaddr 56:84:7a:fe:97:99
            inet addr:172.17.42.1  Bcast:0.0.0.0  Mask:255.255.0.0
            inet6 addr: fe80::5484:7aff:fe97:9799/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:10223 errors:0 dropped:0 overruns:0 frame:0
            TX packets:10965 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:542249 (542.2 KB)  TX bytes:109606629 (109.6 MB)

eth0       Link encap:Ethernet  HWaddr 52:54:00:7c:8e:6a
            inet addr:192.168.122.214  Bcast:192.168.122.255  Mask:255.255.255.0
            inet6 addr: fe80::5054:ff:fe7c:8e6a/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:322830 errors:0 dropped:3 overruns:0 frame:0
            TX packets:69164 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:477589192 (477.5 MB)  TX bytes:6042544 (6.0 MB)

lo         Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

veth221c6c4 Link encap:Ethernet  HWaddr da:b8:c5:35:ec:40
            inet6 addr: fe80::d8b8:c5ff:fe35:ec40/64 Scope:Link
            UP BROADCAST RUNNING  MTU:1500  Metric:1
            RX packets:1381362 errors:0 dropped:0 overruns:0 frame:0
            TX packets:1669608 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:216250136 (216.2 MB)  TX bytes:244518717 (244.5 MB)

veth5b52a16 Link encap:Ethernet  HWaddr 06:b4:d7:c6:66:1f
            inet6 addr: fe80::4b4:d7ff:fec6:661f/64 Scope:Link
            UP BROADCAST RUNNING  MTU:1500  Metric:1
            RX packets:1882963 errors:0 dropped:0 overruns:0 frame:0
            TX packets:2074906 errors:0 dropped:0 overruns:0 carrier:0
```

2. Install Docker

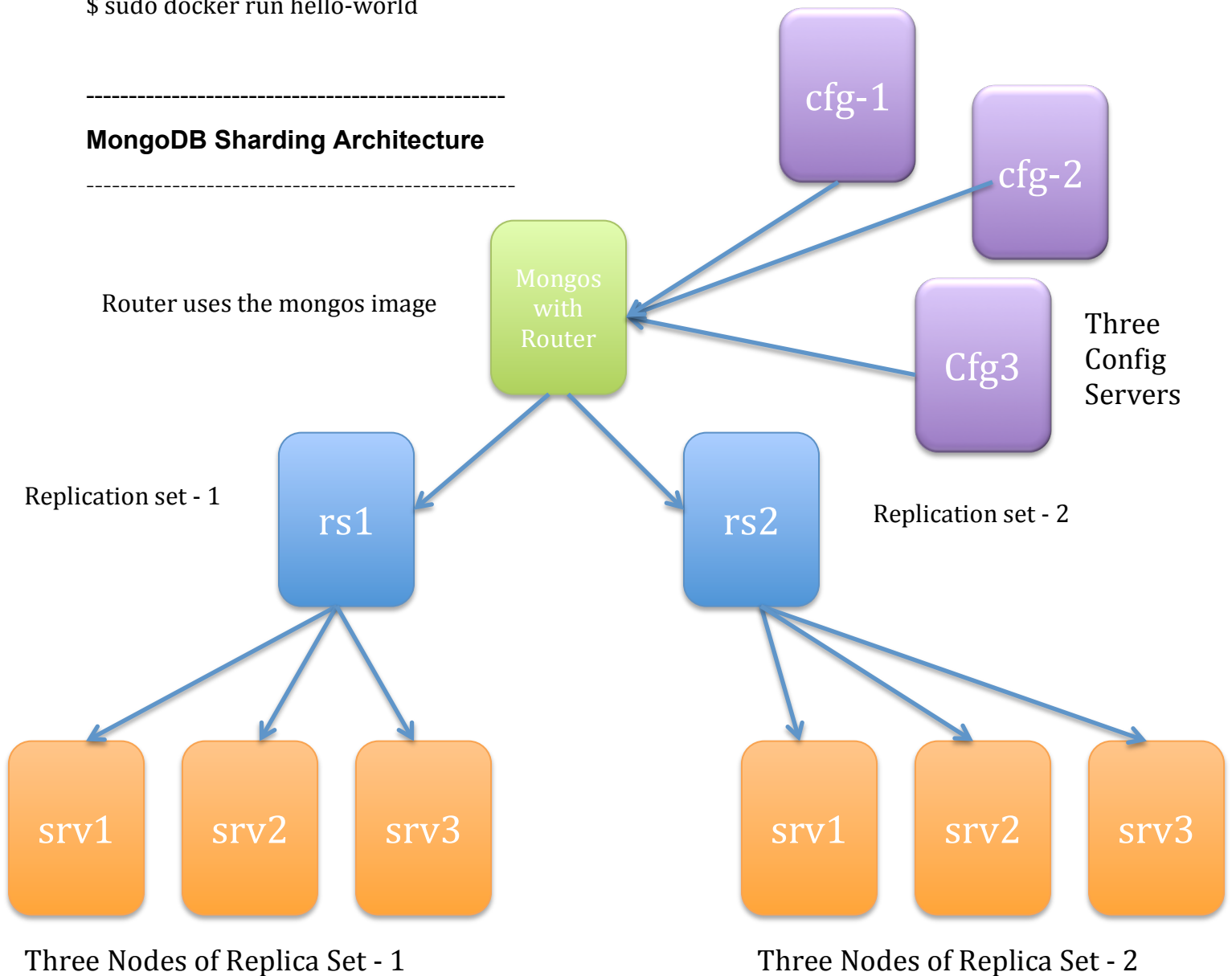
Get the latest Docker package.

```
$ wget -qO- https://get.docker.com/ | sh
```

Verify docker is installed correctly.

```
$ sudo docker run hello-world
```

MongoDB Sharding Architecture



Docker Configurations

1. Create the Docker Files

We created two Dockerfiles: one for mongod and another one for mongos.

- Root Folder: /home/saurav/assgn2
- Create 2 more Folders: "mongod" & "mongos"
\$mkdir /home/saurav/assgn2/mongod
\$mkdir /home/saurav/assgn2/mongos

```
root@docker:/home/saurav# ls  
assgn2  
root@docker:/home/saurav# pwd  
/home/saurav  
root@docker:/home/saurav# ls assgn2/  
mongod  mongos
```

- Create separate Dockerfiles for mongos and mongod respectively.
- Docker file for mongod:

```
FROM ubuntu:latest  
  
# Add 10gen official apt source to the sources list  
RUN apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10  
RUN echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | tee /etc/apt/sources.list.d/10gen.list  
  
# Install MongoDB  
RUN apt-get update  
RUN apt-get install mongodb-10gen  
  
# Create the MongoDB data directory  
RUN mkdir -p /data/db  
  
EXPOSE 27017  
ENTRYPOINT ["usr/bin/mongod"]  
~  
~
```

- Docker file for mongos:

```
FROM saurav/mongodb:latest  
  
EXPOSE 27017  
ENTRYPOINT ["usr/bin/mongos"]  
~
```

2. Build Docker Files

- Go to mongod folder and execute this command:

```
$sudo docker build \ -t saurav/mongodb mongod
```

- Go to mongos folder and execute this command:

```
$sudo docker build \ -t saurav/mongos mongos
```

3. **Create the Replica Sets**

- First Replica Set – rs1

```
$sudo docker run \ -P -name rs1_srv1 \ -d saurav/mongodb \ --replSet rs1 \ --  
noprealloc --smallfiles
```

```
$sudo docker run \ -P -name rs1_srv2 \ -d saurav/mongodb \ --replSet rs1 \ --  
noprealloc --smallfiles
```

```
$sudo docker run \ -P -name rs1_srv3 \ -d saurav/mongodb \ --replSet rs1 \ --  
noprealloc --smallfiles
```

- Second Replica Set – rs2

```
$sudo docker run \ -P -name rs2_srv1 \ -d saurav/mongodb \ --replSet rs2 \ --  
noprealloc --smallfiles
```

```
$sudo docker run \ -P -name rs2_srv2 \ -d saurav/mongodb \ --replSet rs2 \ --  
noprealloc --smallfiles
```

```
$sudo docker run \ -P -name rs2_srv3 \ -d saurav/mongodb \ --replSet rs2 \ --  
noprealloc --smallfiles
```

4. **Intialize the Replica Sets**

- Note the IP addresses of the containers:

```
$sudo docker inspect rs1_srv1
```

```
$sudo docker inspect rs1_srv2
```

```
$sudo docker inspect rs1_srv3
```

```
$sudo docker inspect rs2_srv1
```

```
$sudo docker inspect rs2_srv2
```

```
$sudo docker inspect rs2_srv3
```

Sample output:

```

    },
    "HostnamePath": "/var/lib/docker/containers/01c16f37ae627f5080c685340fb9e7c59ce27a4",
    "HostsPath": "/var/lib/docker/containers/01c16f37ae627f5080c685340fb9e7c59ce27a447a",
    "Id": "01c16f37ae627f5080c685340fb9e7c59ce27a447ad4ea7a9190967fcfed9574",
    "Image": "23eef8a6610f1e18de11f8c4da148bd58cce8fd487308d03746d285c97675962",
    "MountLabel": "",
    "Name": "/rs1_srv1",
    "NetworkSettings": {
      "Bridge": "docker0",
      "Gateway": "172.17.42.1",
      "GlobalIPv6Address": "",
      "GlobalIPv6PrefixLen": 0,
      "IPAddress": "172.17.0.8",
      "IPPrefixLen": 16,
      "IPv6Gateway": "",
      "LinkLocalIPv6Address": "fe80::42:acff:fe11:8",
      "LinkLocalIPv6PrefixLen": 64,
      "MacAddress": "02:42:ac:11:00:08",
      "PortMapping": null,
      "Ports": {
        "27017/tcp": [
          {
            "HostIp": "0.0.0.0",
            "HostPort": "49153"
          }
        ]
      }
    },
    "Path": "usr/bin/mongod",

```

IP Addresses of our containers:

rs1_srv1 - 172.17.0.8

rs1_srv2 - 172.17.0.9

rs1_srv3 - 172.17.0.10

rs2_srv1 - 172.17.0.11

rs2_srv2 - 172.17.0.12

rs2_srv3 - 172.17.0.13

- Initialize Replication Set1

Connect to MongoDB running in container rs1_srv1

a) First enter the bash of the contain

\$sudo docker exec -i -t <container-id> bash

b) Enter the MongoDB

\$mongo

c) Once you enter MongoDB shell , execute these commands:

```
rs.initiate()
rs.add("<IP_of_rs1_srv2>:27017")
rs.add("<IP_of_rs1_srv3>:27017")
rs.status()
```

```
rs1:PRIMARY> rs.status()
{
  "set" : "rs1",
  "date" : ISODate("2015-04-15T20:04:28Z"),
  "myState" : 1,
  "members" : [
    {
      "_id" : 0,
      "name" : "172.17.0.8:27017",
      "health" : 1,
      "state" : 1,
      "stateStr" : "PRIMARY",
      "uptime" : 489444,
      "optime" : Timestamp(1429114584, 1),
      "optimeDate" : ISODate("2015-04-15T16:16:24Z"),
      "self" : true
    },
    {
      "_id" : 1,
      "name" : "172.17.0.9:27017",
      "health" : 1,
      "state" : 2,
      "stateStr" : "SECONDARY",
      "uptime" : 488612,
      "optime" : Timestamp(1429114584, 1),
      "optimeDate" : ISODate("2015-04-15T16:16:24Z"),
      "lastHeartbeat" : ISODate("2015-04-15T20:04:28Z"),
      "lastHeartbeatRecv" : ISODate("2015-04-15T20:04:28Z"),
      "pingMs" : 0,
      "syncingTo" : "172.17.0.8:27017"
    },
    {
      "_id" : 2,
      "name" : "172.17.0.10:27017",
      "health" : 1,
      "state" : 2,
      "stateStr" : "SECONDARY",
      "uptime" : 488612,
      "optime" : Timestamp(1429114584, 1),
      "optimeDate" : ISODate("2015-04-15T16:16:24Z"),
      "lastHeartbeat" : ISODate("2015-04-15T20:04:28Z"),
      "lastHeartbeatRecv" : ISODate("2015-04-15T20:04:28Z"),
      "pingMs" : 0,
      "syncingTo" : "172.17.0.8:27017"
    }
  ],
  "ok" : 1
}
```

- Initialize Replication Set2

Connect to MongoDB running in container rs2_srv1

a) First enter the bash of the contain

```
$sudo docker exec -i -t <container-id> bash
```

b) Enter the MongoDB

```
$mongo
```

c) Once you enter MongoDB shell , execute these commands:

```
rs.initiate()
```

```
rs.add("<IP_of_rs2_srv2>:27017")
```

```
rs.add("<IP_of_rs2_srv3>:27017")
```

```
rs.status()
```

d) Change the hostname to the IP address.

```
cfg = rs.conf()
```

```
cfg.members[0].host = "<IP_of_rs2_srv1>:27017"
```

```
rs.reconfig(cfg)
```

```
rs.status()
```

```

rs2:PRIMARY> rs.status()
{
  "set" : "rs2",
  "date" : ISODate("2015-04-15T20:12:59Z"),
  "myState" : 1,
  "members" : [
    {
      "_id" : 0,
      "name" : "172.17.0.11:27017",
      "health" : 1,
      "state" : 1,
      "stateStr" : "PRIMARY",
      "uptime" : 489790,
      "optime" : Timestamp(1428639939, 1),
      "optimeDate" : ISODate("2015-04-10T04:25:39Z"),
      "self" : true
    },
    {
      "_id" : 1,
      "name" : "172.17.0.12:27017",
      "health" : 1,
      "state" : 2,
      "stateStr" : "SECONDARY",
      "uptime" : 488840,
      "optime" : Timestamp(1428639939, 1),
      "optimeDate" : ISODate("2015-04-10T04:25:39Z"),
      "lastHeartbeat" : ISODate("2015-04-15T20:12:58Z"),
      "lastHeartbeatRecv" : ISODate("2015-04-15T20:12:58Z"),
      "pingMs" : 0,
      "syncingTo" : "172.17.0.11:27017"
    },
    {
      "_id" : 2,
      "name" : "172.17.0.13:27017",
      "health" : 1,
      "state" : 2,
      "stateStr" : "SECONDARY",
      "uptime" : 488840,
      "optime" : Timestamp(1428639939, 1),
      "optimeDate" : ISODate("2015-04-10T04:25:39Z"),
      "lastHeartbeat" : ISODate("2015-04-15T20:12:58Z"),
      "lastHeartbeatRecv" : ISODate("2015-04-15T20:12:58Z"),
      "pingMs" : 0,
      "syncingTo" : "172.17.0.11:27017"
    }
  ]
}

```

5. Create Config Servers

```
$sudo docker run \ -P -name cfg1 \ -d saurav/mongodb \ --noprealloc --smallfiles \
--configsvr \ --dbpath /data/db \ --port 27017
```

```
$sudo docker run \ -P -name cfg2 \ -d saurav/mongodb \ --noprealloc --smallfiles \
--configsvr \ --dbpath /data/db \ --port 27017
```

```
$sudo docker run \ -P -name cfg3 \ -d saurav/mongodb \ --noprealloc --smallfiles \
--configsvr \ --dbpath /data/db \ --port 27017
```


IP Addresses of the Config Servers:

cfg1 - 172.17.0.14

cfg2 - 172.17.0.15

cfg3 - 172.17.0.16

6. Create Router

- Here we are using the mongos image created earlier:

```
$sudo docker run \ -P -name mongos1 \ -d saurav/mongos \ --port 27017 \ --  
configdb \ <IP_of_container_cfg1>:27017, \ <IP_of_container_cfg2>:27017, \  
<IP_of_container_cfg3>:27017
```

- IP Address of Router: "mongos3" - 172.17.0.21

7. Initialize the Shard

- Login to MongoDB router and execute the below commands:

```
$sudo docker exec -i -t <container_id> bash
```

- Enter MongoDB and execute below commands:

```
$mongo
```

- Execute below MongoDB queries:

```
sh.addShard("rs1/<IP_of_rs1_srv1>:27017")
```

```
sh.addShard("rs2/<IP_of_rs2_srv1>:27017")
```

```
sh.status()
```

```
mongos> sh.status()  
--- Sharding Status ---  
  sharding version: {  
    "_id" : 1,  
    "version" : 3,  
    "minCompatibleVersion" : 3,  
    "currentVersion" : 4,  
    "clusterId" : ObjectId("55275a05d64e8f99a7e081d7")  
  }  
  shards:  
    { "_id" : "rs1", "host" : "rs1/172.17.0.10:27017,172.17.0.8:27017,172.17.0.9:27017" }  
    { "_id" : "rs2", "host" : "rs2/172.17.0.11:27017,172.17.0.12:27017,172.17.0.13:27017" }  
  databases:  
    { "_id" : "admin", "partitioned" : false, "primary" : "config" }  
    { "_id" : "test", "partitioned" : true, "primary" : "rs1" }  
      test.test_collection  
        shard key: { "number" : 1 }  
        chunks:  
          rs1      1  
          { "number" : { "$minKey" : 1 } } --> { "number" : { "$maxKey" : 1 } } on : rs1 Timestamp(1, 0)  
mongos>
```

APPENDIX A – Dockerfile (mongod)

FROM ubuntu:latest

Add 10gen official apt source to the sources list

RUN apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10

*RUN echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' |
tee /etc/apt/sources.list.d/10gen.list*

Install MongoDB

RUN apt-get update

RUN apt-get install mongodb-10gen

Create the MongoDB data directory

RUN mkdir -p /data/db

EXPOSE 27017

ENTRYPOINT ["usr/bin/mongod"]

APPENDIX B – Dockerfile (mongos)

FROM saurav/mongodb:latest

EXPOSE 27017

ENTRYPOINT ["usr/bin/mongos"]

APPENDIX C – LIST OF RUNNING CONTAINERS

```
root@docker:/home/saurav# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                               NAMES
473b2de03e6d      saurav/mongos:latest  "usr/bin/mongos --po 6 days ago          Up 6 days          0.0.0.0:49166->27017/tcp            mongos3
93f9df413205      saurav/mongodb:latest  "usr/bin/mongod --no 6 days ago          Up 6 days          0.0.0.0:49161->27017/tcp            cfg3
1139f54d55ab      saurav/mongodb:latest  "usr/bin/mongod --no 6 days ago          Up 6 days          0.0.0.0:49160->27017/tcp            cfg2
555d56f1fa0e      saurav/mongodb:latest  "usr/bin/mongod --no 6 days ago          Up 6 days          0.0.0.0:49159->27017/tcp            cfg1
1b1ed97d78e6      saurav/mongodb:latest  "usr/bin/mongod --re 6 days ago          Up 6 days          0.0.0.0:49158->27017/tcp            rs2_srv3
a19e7f22f885      saurav/mongodb:latest  "usr/bin/mongod --re 6 days ago          Up 6 days          0.0.0.0:49157->27017/tcp            rs2_srv2
2853205d3b87      saurav/mongodb:latest  "usr/bin/mongod --re 6 days ago          Up 6 days          0.0.0.0:49156->27017/tcp            rs2_srv1
8d1e98e35cb4      saurav/mongodb:latest  "usr/bin/mongod --re 6 days ago          Up 6 days          0.0.0.0:49155->27017/tcp            rs1_srv3
83ebbefc1645      saurav/mongodb:latest  "usr/bin/mongod --re 6 days ago          Up 6 days          0.0.0.0:49154->27017/tcp            rs1_srv2
91c16f37ae62      saurav/mongodb:latest  "usr/bin/mongod --re 6 days ago          Up 6 days          0.0.0.0:49153->27017/tcp            rs1_srv1
root@docker:/home/saurav#
```