Introduction

The purpose of this assignment is to create a distributed network of containers with database instances connecting each other by deploying Docker instances un a host machine. In this assignment we have installed docker engine on our host machine and generated three docker machines. Inside each machine a mongodb image file as a container along with a docker volume has been created. Admin, replica and key authentication files were created and passed to the docker container. This will be called as docker file which will build the images automatically for mongodb instance. Later, the same configurations of mangobd containers were created for the other docker machines using the replica and the containers were connected to the replica set of the first container.

Docker

Docker [1] is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers is an environment which allows a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package. By doing so, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code. In a way, Docker is a bit like a virtual machine. But unlike a virtual machine, rather than creating a whole virtual operating system, Docker allows applications to use the same Linux kernel as the system that they're running on and only requires applications be shipped with things not already running on the host computer. This gives a significant performance boost and reduces the size of the application. Importantly, Docker is open source. This means that anyone can contribute to Docker and extend it to meet their own needs if they need additional features that are not available out of the box.

Containers

A Container [2] is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. Container images become containers at runtime and in the case of Docker containers - images become containers when they run on Docker Engine. Available for both Linux and Windows-based applications, containerized software will always run the same, regardless of the infrastructure. Containers isolate software from its environment and ensure that it works uniformly despite differences for instance between development and staging. So basically, Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space.

Virtual Machine vs Container

Containers and virtual machines have similar resource isolation and allocation benefits, but function differently because containers virtualize the operating system instead of hardware. Containers are more portable and efficient. Containers take up less space than VMs can handle more applications and require fewer VMs and Operating systems. Difference in the architecture between virtual machine and container can be seen below.

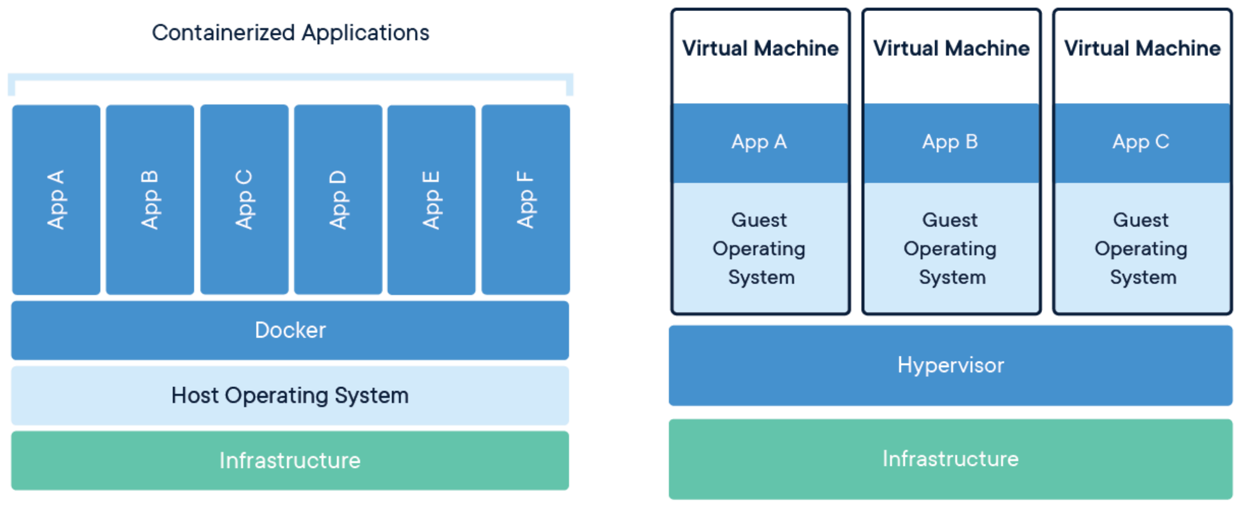


Figure 1 Comparison between the architectures of virtual machine and contaniner

<u>**Docker Installation**</u>

- Existing packages must be updated in the operating system

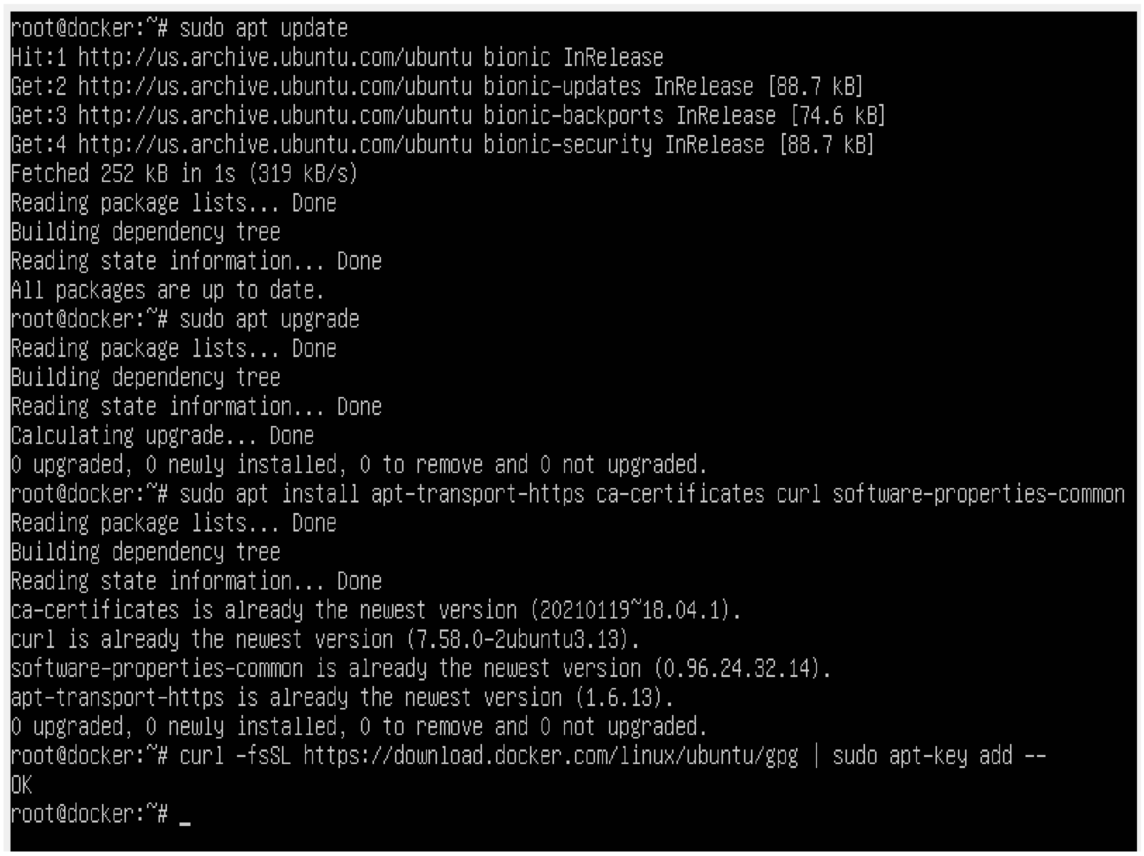
  $ sudo apt update

- Prerequisite packages have to be installed

  sudo apt install apt-transport-https ca-certificates curl software-properties-common

- GPG key has to be added for the official Docker repository to the system

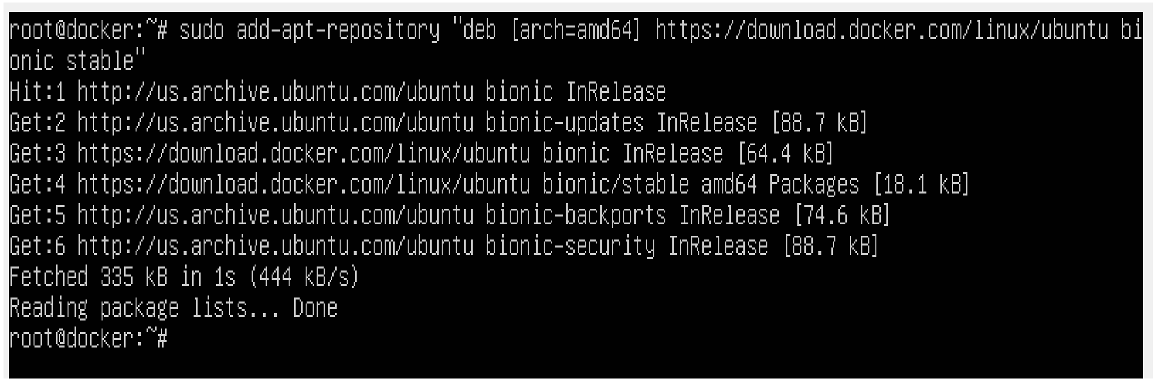
  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –



```
root@docker:~# sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Fetched 252 kB in 1s (319 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
root@docker:~# sudo apt upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@docker:~# sudo apt install apt-transport-https ca-certificates curl software-properties-common
Reading package lists... Done
Building dependency tree
Reading state information... Done
ca-certificates is already the newest version (20210119~18.04.1).
curl is already the newest version (7.58.0-2ubuntu3.13).
software-properties-common is already the newest version (0.96.24.32.14).
apt-transport-https is already the newest version (1.6.13).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@docker:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add --
OK
root@docker:~# _
```

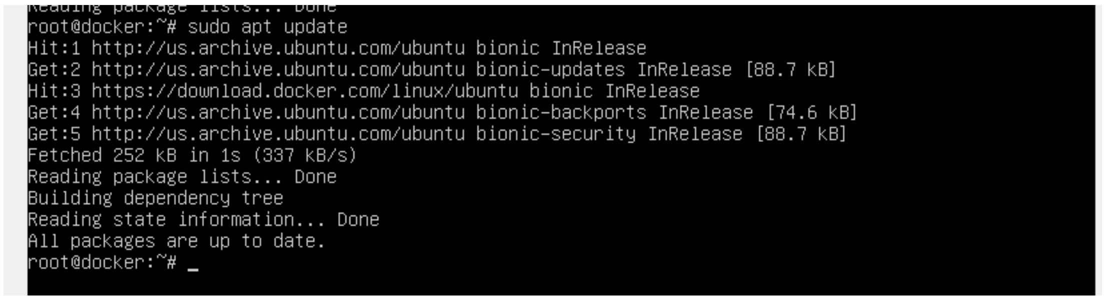
- Docker repository should be added to APT sources

  $ sudo add-apt-repository “deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable”



```
root@docker:~# sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bi
onic stable"
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 https://download.docker.com/linux/ubuntu bionic InRelease [64.4 kB]
Get:4 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages [18.1 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Fetched 335 kB in 1s (444 kB/s)
Reading package lists... Done
root@docker:~#
```

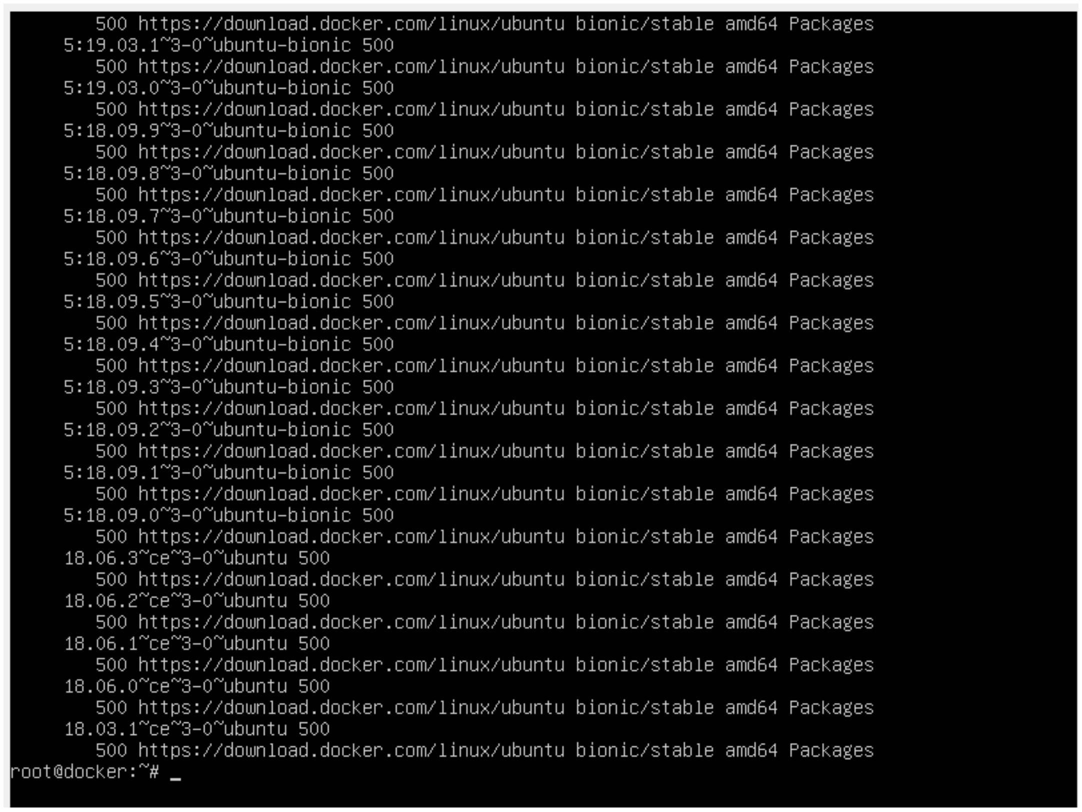
- Update the package directory with the Docker packages from the newly added repo
  $ sudo apt update



```
root@docker:~# sudo apt update
Hit:1 http://us.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Hit:3 https://download.docker.com/linux/ubuntu bionic InRelease
Get:4 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Fetched 252 kB in 1s (337 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
All packages are up to date.
root@docker:~# _
```

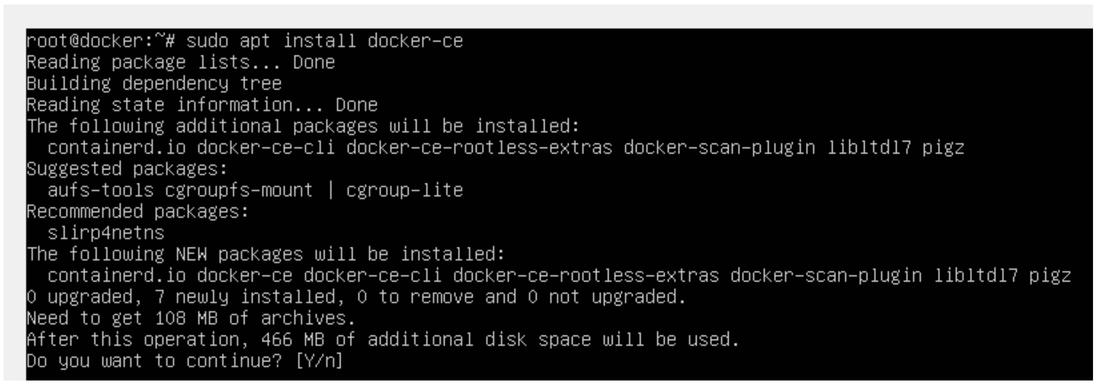
- Verify that installation has to be done from the Docker repo instead of the default Ubuntu repo

  $ apt-cache policy docker-ce

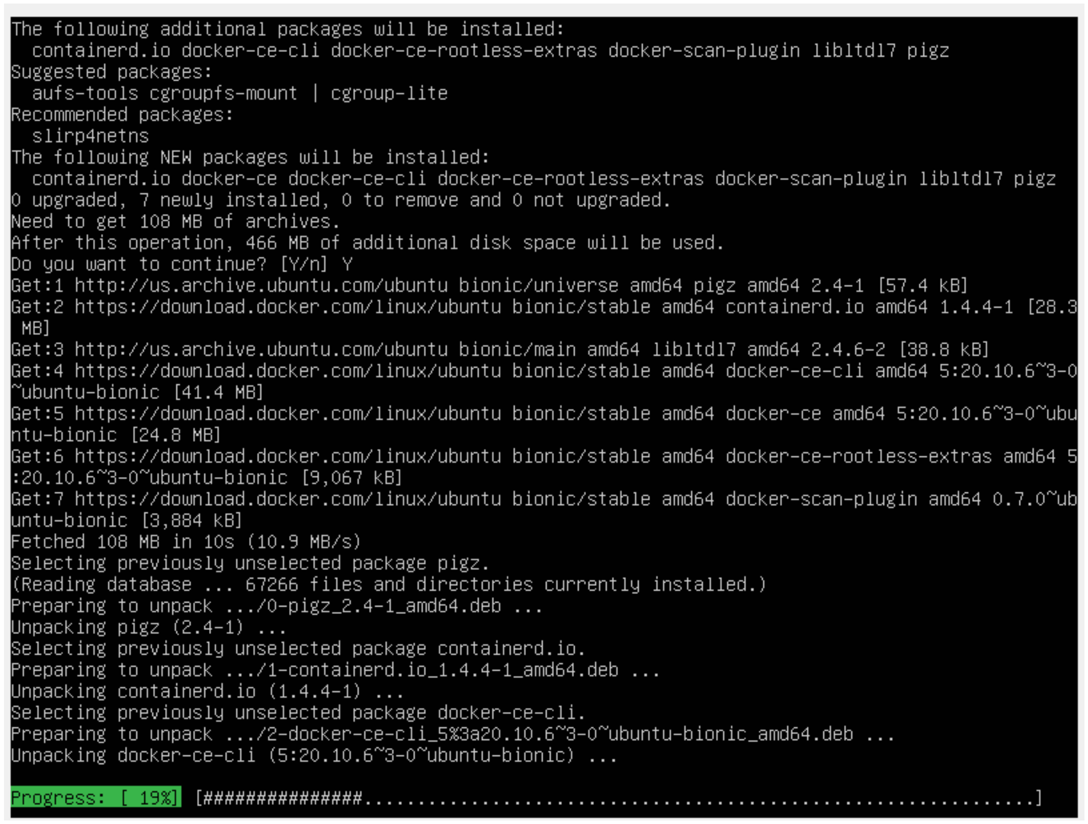


```
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     5:19.03.1~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     5:19.03.0~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     5:18.09.9~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     5:18.09.8~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     5:18.09.7~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     5:18.09.6~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     5:18.09.5~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     5:18.09.4~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     5:18.09.3~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     5:18.09.2~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     5:18.09.1~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     5:18.09.0~3-0~ubuntu-bionic 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     18.06.3~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     18.06.2~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     18.06.1~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     18.06.0~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
     18.03.1~ce~3-0~ubuntu 500
        500 https://download.docker.com/linux/ubuntu bionic/stable amd64 Packages
root@docker:~# _
```

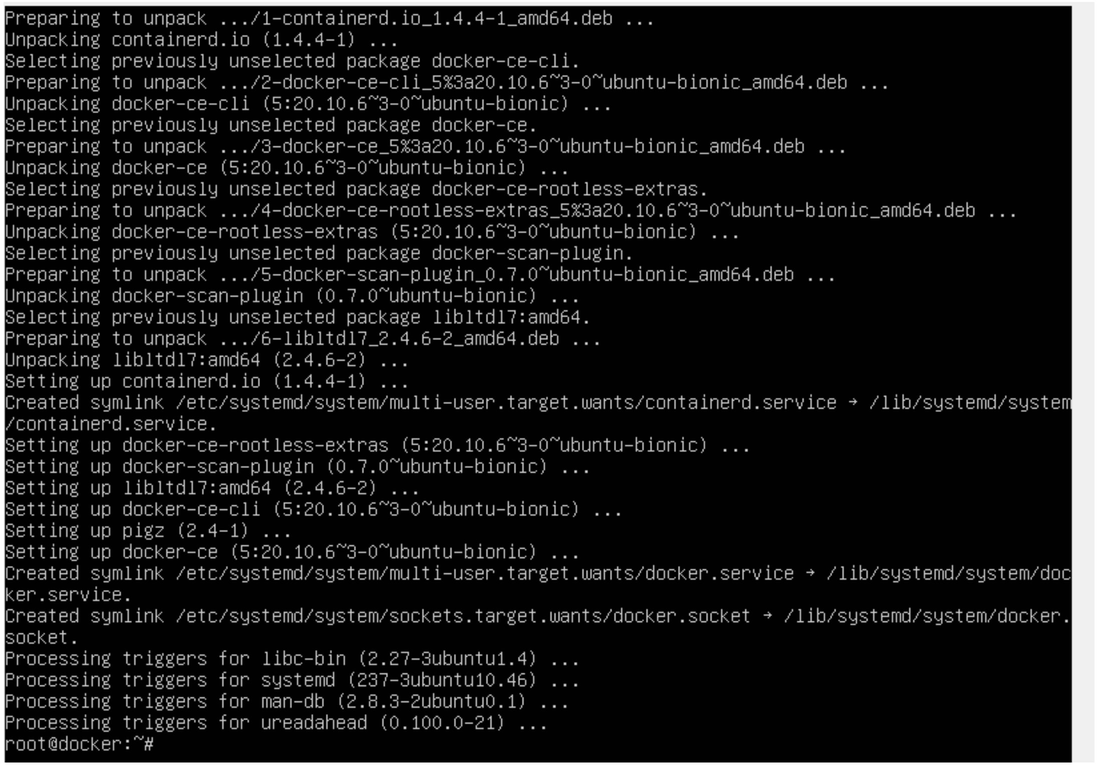
- Now, install Docker
  $ sudo apt install docker-ce



```
root@docker:~# sudo apt install docker-ce
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  containerd.io docker-ce-cli docker-ce-rootless-extras docker-scan-plugin libltdl7 pigz
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
Recommended packages:
  slirp4netns
The following NEW packages will be installed:
  containerd.io docker-ce docker-ce-cli docker-ce-rootless-extras docker-scan-plugin libltdl7 pigz
0 upgraded, 7 newly installed, 0 to remove and 0 not upgraded.
Need to get 108 MB of archives.
After this operation, 466 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```



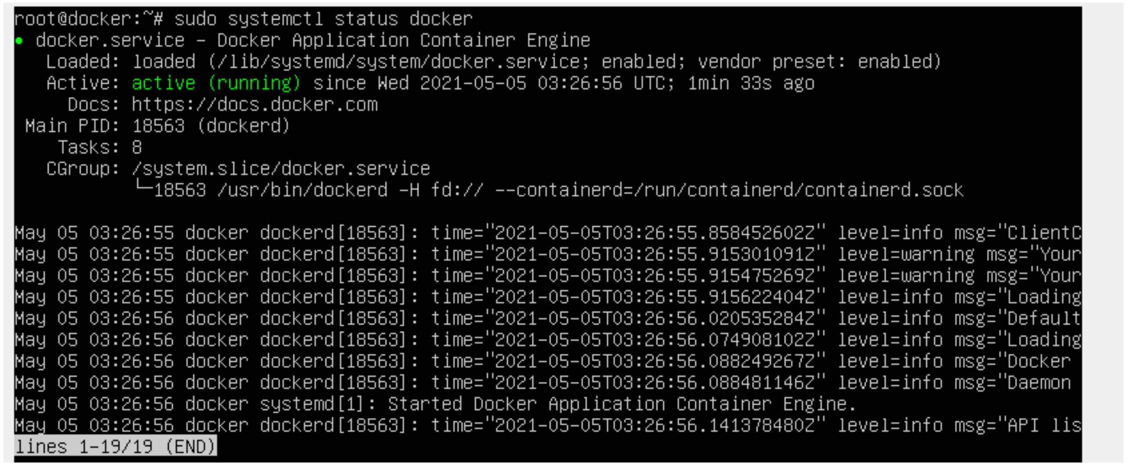
```
The following additional packages will be installed:
  containerd.io docker-ce-cli docker-ce-rootless-extras docker-scan-plugin libltdl7 pigz
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
Recommended packages:
  slirp4netns
The following NEW packages will be installed:
  containerd.io docker-ce docker-ce-cli docker-ce-rootless-extras docker-scan-plugin libltdl7 pigz
0 upgraded, 7 newly installed, 0 to remove and 0 not upgraded.
Need to get 108 MB of archives.
After this operation, 466 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us.archive.ubuntu.com/ubuntu bionic/universe amd64 pigz amd64 2.4-1 [57.4 kB]
Get:2 https://download.docker.com/linux/ubuntu bionic/stable amd64 containerd.io amd64 1.4.4-1 [28.3
 MB]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 libltdl7 amd64 2.4.6-2 [38.8 kB]
Get:4 https://download.docker.com/linux/ubuntu bionic/stable amd64 docker-ce-cli amd64 5:20.10.6~3-0
~ubuntu-bionic [41.4 MB]
Get:5 https://download.docker.com/linux/ubuntu bionic/stable amd64 docker-ce amd64 5:20.10.6~3-0~ubu
ntu-bionic [24.8 MB]
Get:6 https://download.docker.com/linux/ubuntu bionic/stable amd64 docker-ce-rootless-extras amd64 5
:20.10.6~3-0~ubuntu-bionic [9,067 kB]
Get:7 https://download.docker.com/linux/ubuntu bionic/stable amd64 docker-scan-plugin amd64 0.7.0~ub
untu-bionic [3,884 kB]
Fetched 108 MB in 10s (10.9 MB/s)
Selecting previously unselected package pigz.
(Reading database ... 67266 files and directories currently installed.)
Preparing to unpack .../0-pigz_2.4-1_amd64.deb ...
Unpacking pigz (2.4-1) ...
Selecting previously unselected package containerd.io.
Preparing to unpack .../1-containerd.io_1.4.4-1_amd64.deb ...
Unpacking containerd.io (1.4.4-1) ...
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../2-docker-ce-cli_5%3a20.10.6~3-0~ubuntu-bionic_amd64.deb ...
Unpacking docker-ce-cli (5:20.10.6~3-0~ubuntu-bionic) ...

Progress: [ 19%] [###############..............................................................]
```



```
Preparing to unpack .../1-containerd.io_1.4.4-1_amd64.deb ...
Unpacking containerd.io (1.4.4-1) ...
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../2-docker-ce-cli_5%3a20.10.6~3-0~ubuntu-bionic_amd64.deb ...
Unpacking docker-ce-cli (5:20.10.6~3-0~ubuntu-bionic) ...
Selecting previously unselected package docker-ce.
Preparing to unpack .../3-docker-ce_5%3a20.10.6~3-0~ubuntu-bionic_amd64.deb ...
Unpacking docker-ce (5:20.10.6~3-0~ubuntu-bionic) ...
Selecting previously unselected package docker-ce-rootless-extras.
Preparing to unpack .../4-docker-ce-rootless-extras_5%3a20.10.6~3-0~ubuntu-bionic_amd64.deb ...
Unpacking docker-ce-rootless-extras (5:20.10.6~3-0~ubuntu-bionic) ...
Selecting previously unselected package docker-scan-plugin.
Preparing to unpack .../5-docker-scan-plugin_0.7.0~ubuntu-bionic_amd64.deb ...
Unpacking docker-scan-plugin (0.7.0~ubuntu-bionic) ...
Selecting previously unselected package libltdl7:amd64.
Preparing to unpack .../6-libltdl7_2.4.6-2_amd64.deb ...
Unpacking libltdl7:amd64 (2.4.6-2) ...
Setting up containerd.io (1.4.4-1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system
/containerd.service.
Setting up docker-ce-rootless-extras (5:20.10.6~3-0~ubuntu-bionic) ...
Setting up docker-scan-plugin (0.7.0~ubuntu-bionic) ...
Setting up libltdl7:amd64 (2.4.6-2) ...
Setting up docker-ce-cli (5:20.10.6~3-0~ubuntu-bionic) ...
Setting up pigz (2.4-1) ...
Setting up docker-ce (5:20.10.6~3-0~ubuntu-bionic) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/doc
ker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.
socket.
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
Processing triggers for systemd (237-3ubuntu10.46) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
root@docker:~#
```

- Docker should be installed now so need to check the status of Docker

  $ sudo systemctl status docker



```
root@docker:~# sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-05-05 03:26:56 UTC; 1min 33s ago
     Docs: https://docs.docker.com
 Main PID: 18563 (dockerd)
    Tasks: 8
   CGroup: /system.slice/docker.service
           └─18563 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

May 05 03:26:55 docker dockerd[18563]: time="2021-05-05T03:26:55.858452602Z" level=info msg="ClientC
May 05 03:26:55 docker dockerd[18563]: time="2021-05-05T03:26:55.915301091Z" level=warning msg="Your
May 05 03:26:55 docker dockerd[18563]: time="2021-05-05T03:26:55.915475269Z" level=warning msg="Your
May 05 03:26:55 docker dockerd[18563]: time="2021-05-05T03:26:55.915622404Z" level=info msg="Loading
May 05 03:26:56 docker dockerd[18563]: time="2021-05-05T03:26:56.020535284Z" level=info msg="Default
May 05 03:26:56 docker dockerd[18563]: time="2021-05-05T03:26:56.074908102Z" level=info msg="Loading
May 05 03:26:56 docker dockerd[18563]: time="2021-05-05T03:26:56.088249267Z" level=info msg="Docker
May 05 03:26:56 docker dockerd[18563]: time="2021-05-05T03:26:56.088481146Z" level=info msg="Daemon
May 05 03:26:56 docker systemd[1]: Started Docker Application Container Engine.
May 05 03:26:56 docker dockerd[18563]: time="2021-05-05T03:26:56.141378480Z" level=info msg="API lis
lines 1-19/19 (END)
```

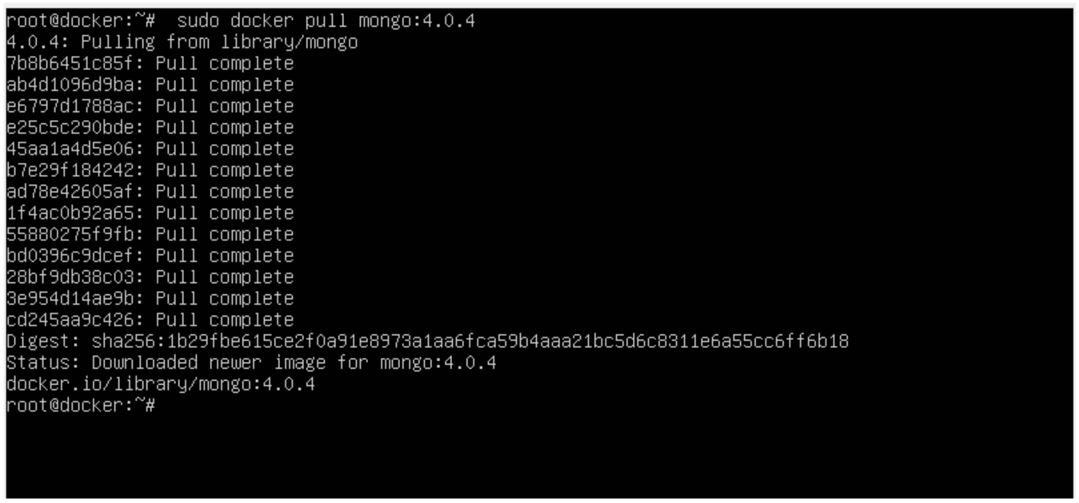
Docker file for MongoDB

A docker file has to be created using the debian as base image from the 'debian:jessie-slim' on docker hub. On top of that we installed certificates, setup gpg keys and installed mongo

4.0.4 from mongodb repo.We have setup the data directory, work directory and exposed 27017 port.

We then built the image from docker file using command

$ sudo docker pull mongo:4.0.4



```
root@docker:~#  sudo docker pull mongo:4.0.4
4.0.4: Pulling from library/mongo
7b8b6451c85f: Pull complete
ab4d1096d9ba: Pull complete
e6797d1788ac: Pull complete
e25c5c290bde: Pull complete
45aa1a4d5e06: Pull complete
b7e29f184242: Pull complete
ad78e42605af: Pull complete
1f4ac0b92a65: Pull complete
55880275f9fb: Pull complete
bd0396c9dcef: Pull complete
28bf9db38c03: Pull complete
3e954d14ae9b: Pull complete
cd245aa9c426: Pull complete
Digest: sha256:1b29fbe615ce2f0a91e8973a1aa6fca59b4aaa21bc5d6c8311e6a55cc6ff6b18
Status: Downloaded newer image for mongo:4.0.4
docker.io/library/mongo:4.0.4
root@docker:~#
```

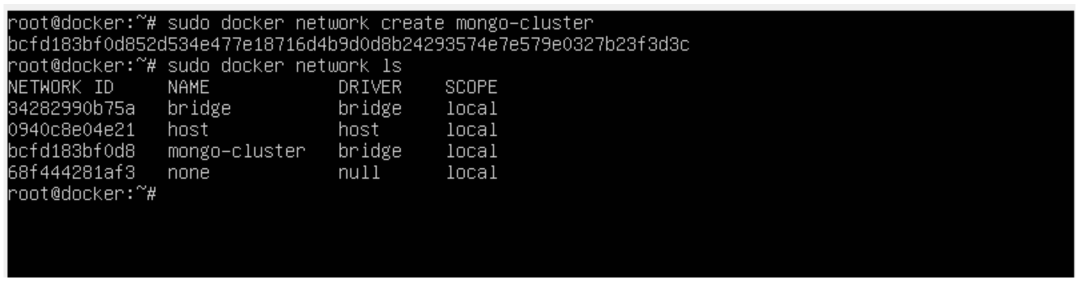
Creating a Network:

- A network must be created for the containers to be able to communicate with each other. So, a network named `mongo-cluster` has been created using the command

$ sudo docker network create mongo-cluster

- Verify the network creation 'mongo-cluster'

$ sudo docker network ls



```
root@docker:~# sudo docker network create mongo-cluster
bcfd183bf0d852d534e477e18716d4b9d0d8b24293574e7e579e0327b23f3d3c
root@docker:~# sudo docker network ls
NETWORK ID     NAME            DRIVER    SCOPE
34282990b75a   bridge          bridge    local
0940c8e04e21   host            host      local
bcfd183bf0d8   mongo-cluster   bridge    local
68f444281af3   none            null      local
root@docker:~#
```

Running Containers:

By using the image created from docker file, run three containers with mongodb set x to run with replication set `myrepl`. the mongo-cluster network using `--net` option for all three containers so we have to set port forwarding for all there containers.

Container 1

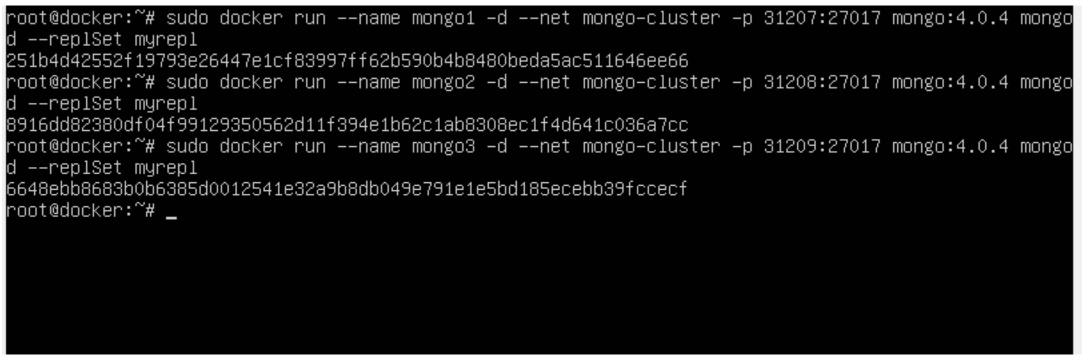
$ sudo docker run --name mongo1 -d --net mongo-cluster -p 31207:27017 mongo:4.0.4 mongod --replSet myrepl`

Container 2

$ sudo docker run --name mongo2 -d --net mongo-cluster -p 31208:27017 mongo:4.0.4 mongod --replSet myrepl`

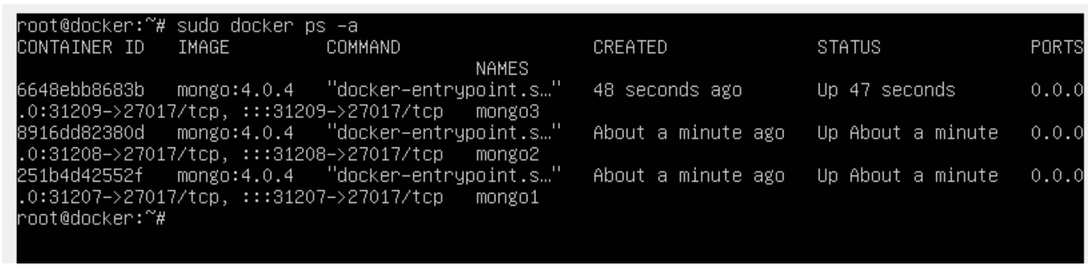
Container 3

$ sudo docker run --name mongo3 -d --net mongo-cluster -p 31209:27017 mongo:4.0.4 mongod --replSet myrepl`



```
root@docker:~# sudo docker run --name mongo1 -d --net mongo-cluster -p 31207:27017 mongo:4.0.4 mongo
d --replSet myrepl
251b4d42552f19793e26447e1cf83997ff62b590b4b8480beda5ac511646ee66
root@docker:~# sudo docker run --name mongo2 -d --net mongo-cluster -p 31208:27017 mongo:4.0.4 mongo
d --replSet myrepl
8916dd82380df04f99129350562d11f394e1b62c1ab8308ec1f4d641c036a7cc
root@docker:~# sudo docker run --name mongo3 -d --net mongo-cluster -p 31209:27017 mongo:4.0.4 mongo
d --replSet myrepl
6648ebb8683b0b6385d0012541e32a9b8db049e791e1e5bd185ecebb39fccecf
root@docker:~# _
```

Verify all three containers running successfully in the image below.



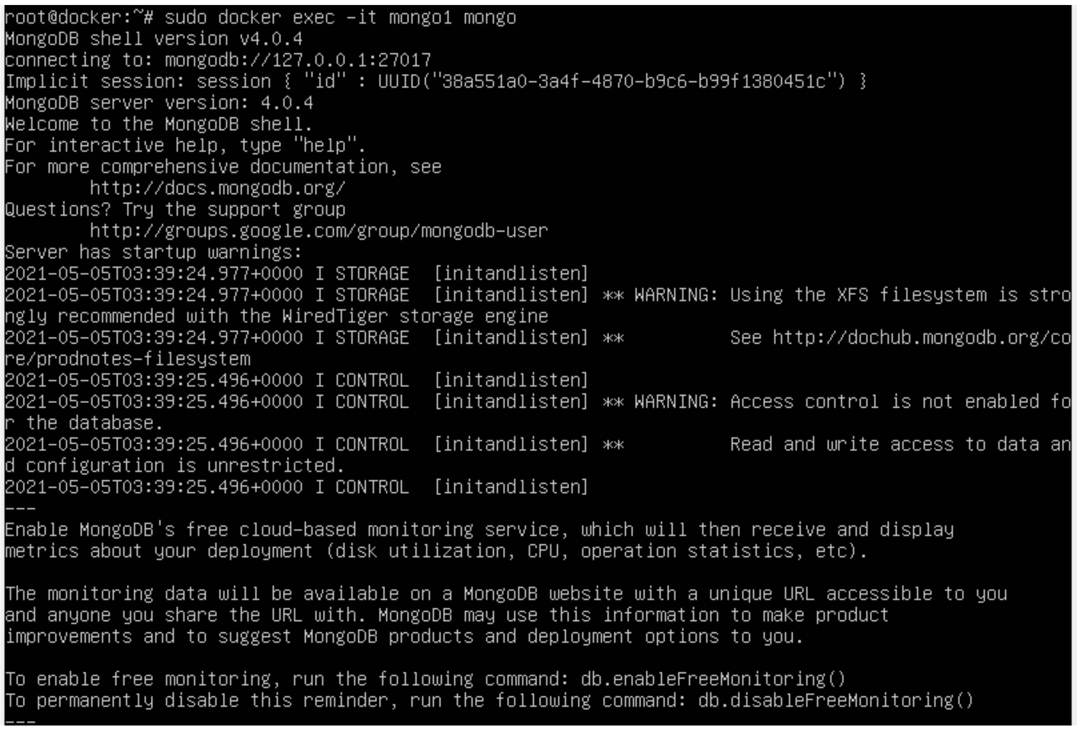
```
root@docker:~# sudo docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED              STATUS              PORTS
                                        NAMES
6648ebb8683b   mongo:4.0.4   "docker-entrypoint.s…"   48 seconds ago       Up 47 seconds       0.0.0
.0:31209->27017/tcp, :::31209->27017/tcp   mongo3
8916dd82380d   mongo:4.0.4   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0
.0:31208->27017/tcp, :::31208->27017/tcp   mongo2
251b4d42552f   mongo:4.0.4   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0
.0:31207->27017/tcp, :::31207->27017/tcp   mongo1
root@docker:~#
```

<u>Configuration of mongodb to replicate data</u>

There are three containers up and running. Now, the task is to set mongodb to replicate data over the three containers. We have chosen the container **`mongo1`** to be master, **`mongo2`** and **` mongo3`** to be slaves. We insert data into master and check if it will be replicates in slaves.

To do this, first open mongo prompt on master container.

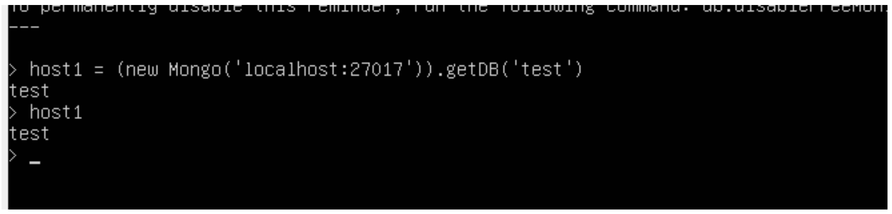
$ sudo docker exec -it mongo1 mongo



```
root@docker:~# sudo docker exec -it mongo1 mongo
MongoDB shell version v4.0.4
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("38a551a0-3a4f-4870-b9c6-b99f1380451c") }
MongoDB server version: 4.0.4
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2021-05-05T03:39:24.977+0000 I STORAGE  [initandlisten]
2021-05-05T03:39:24.977+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is stro
ngly recommended with the WiredTiger storage engine
2021-05-05T03:39:24.977+0000 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/co
re/prodnotes-filesystem
2021-05-05T03:39:25.496+0000 I CONTROL  [initandlisten]
2021-05-05T03:39:25.496+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled fo
r the database.
2021-05-05T03:39:25.496+0000 I CONTROL  [initandlisten] **          Read and write access to data an
d configuration is unrestricted.
2021-05-05T03:39:25.496+0000 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
```

Now get the test database on the container using the command.

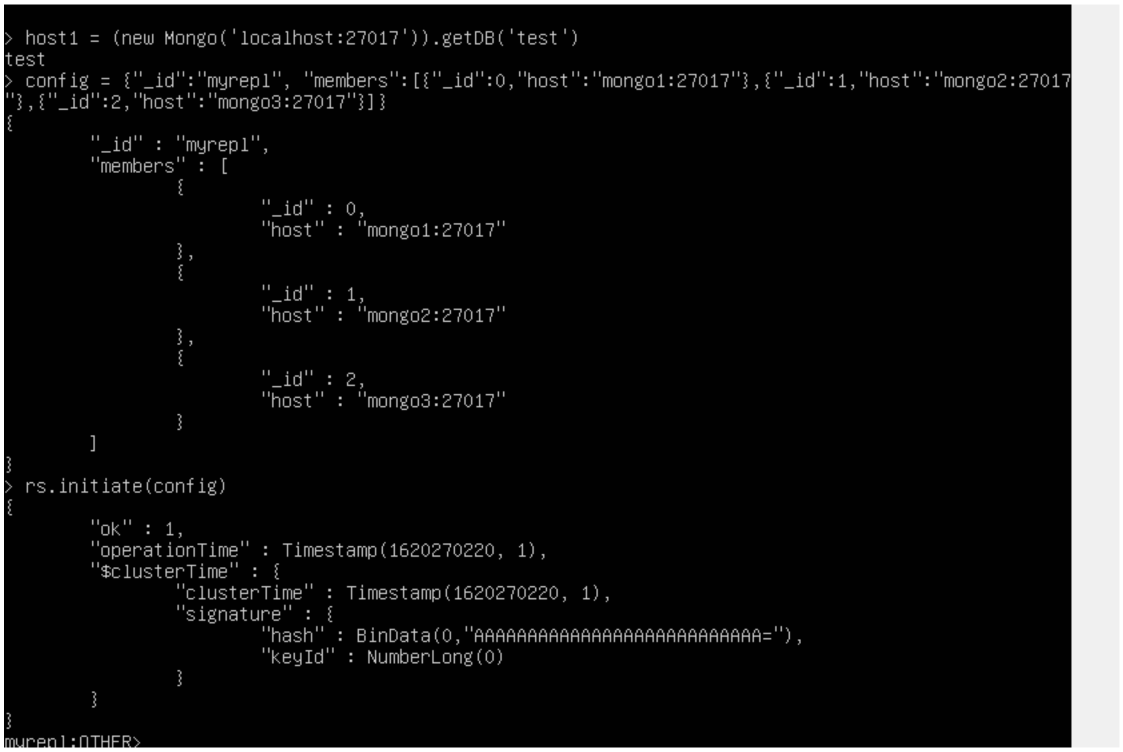
`host1 = (new Mongo('localhost:27017')).getDB('test')`



```
To permanently disable this reminder, run the following command: db.disableFreeMon
---

> host1 = (new Mongo('localhost:27017')).getDB('test')
test
> host1
test
> _
```

Then create and initiate the replication configuration

config={"_id":"myrepl","members":[{"_id":0,"host":" mongo1:27017"},{"_id":1, "host":" mongo2:27017"},{"_id":2,"host":" mongo3:27017"}]} rs.initiate(config)

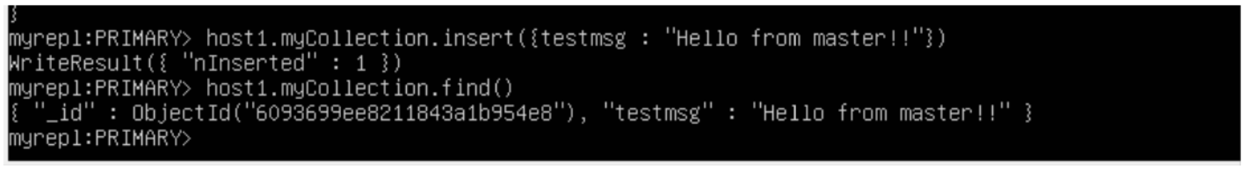


```
> host1 = (new Mongo('localhost:27017')).getDB('test')
test
> config = {"_id":"myrepl", "members":[{"_id":0,"host":"mongo1:27017"},{"_id":1,"host":"mongo2:27017
"},{"_id":2,"host":"mongo3:27017"}]}
{
        "_id" : "myrepl",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "mongo1:27017"
                },
                {
                        "_id" : 1,
                        "host" : "mongo2:27017"
                },
                {
                        "_id" : 2,
                        "host" : "mongo3:27017"
                }
        ]
}
> rs.initiate(config)
{
        "ok" : 1,
        "operationTime" : Timestamp(1620270220, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1620270220, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
myrepl:OTHER>
```

Inserting data and checking replication

Since **mongo1** is the master, insertion of data should be done only through this container.

Data can be inserted into **mongo1** using the command ‘host1.myCollection.inser({testmsg : “Hello from master!!”})’Once insertion of data is done in master container, it can be checked by using the command ‘host1.myCollection.find()’. Now data should be replicated in other two slave containers.



```
}
myrepl:PRIMARY> host1.myCollection.insert({testmsg : "Hello from master!!"})
WriteResult({ "nInserted" : 1 })
myrepl:PRIMARY> host1.myCollection.find()
{ "_id" : ObjectId("6093699ee8211843a1b954e8"), "testmsg" : "Hello from master!!" }
myrepl:PRIMARY>
```

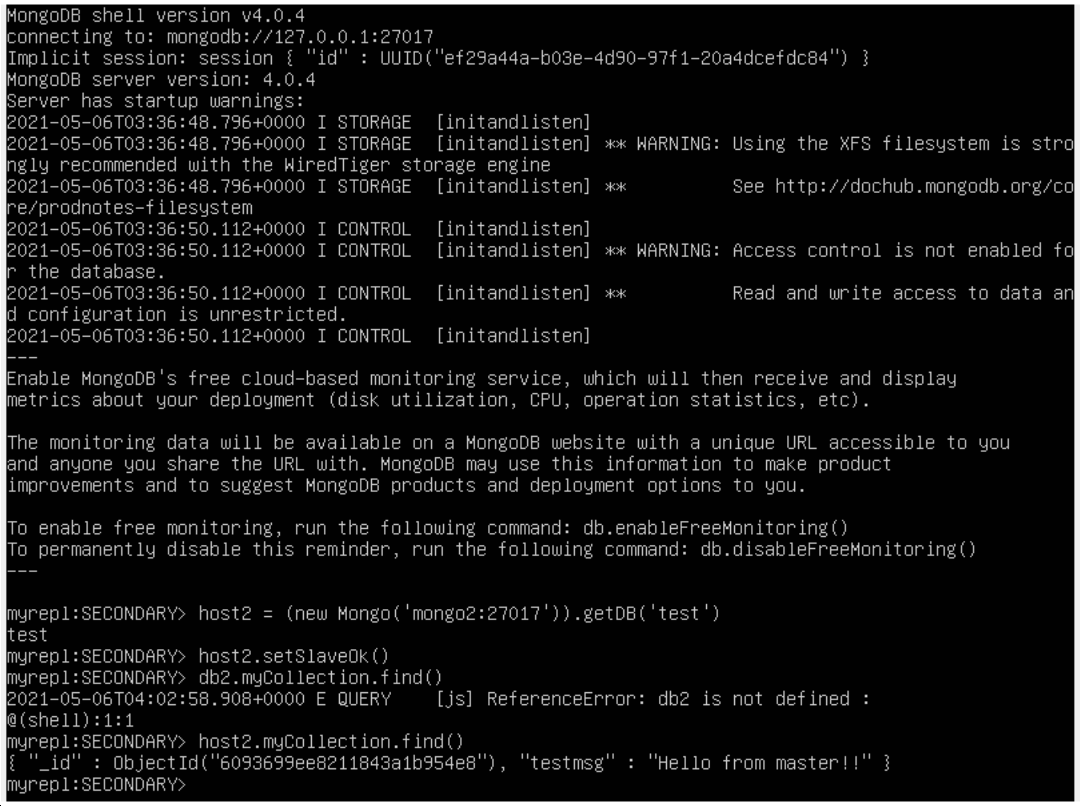
Checking on mongo2

$ sudo docker exec -it mongo2 mongo

Create test database instance using the command 'host2 = (new Mongo('localhost:27017')).getDB('test')'

Set mongo2 as slave using the command 'host2.setSlaveOk()'

Now check the data replication on mongo2 using the command host2.myCollection.find()



```
MongoDB shell version v4.0.4
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("ef29a44a-b03e-4d90-97f1-20a4dcefdc84") }
MongoDB server version: 4.0.4
Server has startup warnings:
2021-05-06T03:36:48.796+0000 I STORAGE  [initandlisten]
2021-05-06T03:36:48.796+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is stro
ngly recommended with the WiredTiger storage engine
2021-05-06T03:36:48.796+0000 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/co
re/prodnotes-filesystem
2021-05-06T03:36:50.112+0000 I CONTROL  [initandlisten]
2021-05-06T03:36:50.112+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled fo
r the database.
2021-05-06T03:36:50.112+0000 I CONTROL  [initandlisten] **          Read and write access to data an
d configuration is unrestricted.
2021-05-06T03:36:50.112+0000 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

myrepl:SECONDARY> host2 = (new Mongo('mongo2:27017')).getDB('test')
test
myrepl:SECONDARY> host2.setSlaveOk()
myrepl:SECONDARY> db2.myCollection.find()
2021-05-06T04:02:58.908+0000 E QUERY    [js] ReferenceError: db2 is not defined :
@(shell):1:1
myrepl:SECONDARY> host2.myCollection.find()
{ "_id" : ObjectId("6093699ee8211843a1b954e8"), "testmsg" : "Hello from master!!" }
myrepl:SECONDARY>
```

-

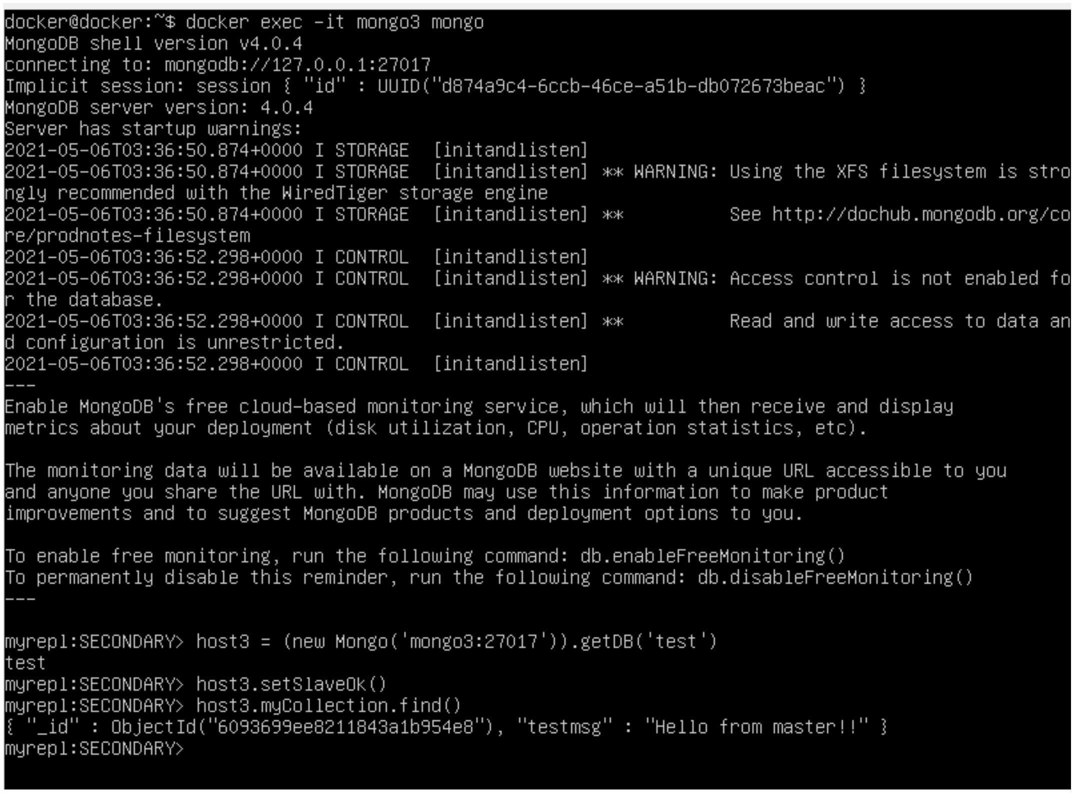
Checking on mongo3

$ sudo docker exec -it mongo3 mongo

Create test database instance using the command 'host3 = (new Mongo('localhost:27017')).getDB('test')'

Set mongo3 as slave using the command 'host3.setSlaveOk()'

Now check the data replication on mongo3 using the command host3.myCollection.find()



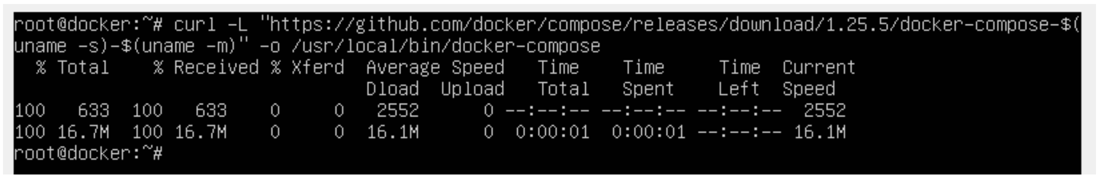
```
docker@docker:~$ docker exec -it mongo3 mongo
MongoDB shell version v4.0.4
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("d874a9c4-6ccb-46ce-a51b-db072673beac") }
MongoDB server version: 4.0.4
Server has startup warnings:
2021-05-06T03:36:50.874+0000 I STORAGE  [initandlisten]
2021-05-06T03:36:50.874+0000 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is stro
ngly recommended with the WiredTiger storage engine
2021-05-06T03:36:50.874+0000 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/co
re/prodnotes-filesystem
2021-05-06T03:36:52.298+0000 I CONTROL  [initandlisten]
2021-05-06T03:36:52.298+0000 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled fo
r the database.
2021-05-06T03:36:52.298+0000 I CONTROL  [initandlisten] **          Read and write access to data an
d configuration is unrestricted.
2021-05-06T03:36:52.298+0000 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

myrepl:SECONDARY> host3 = (new Mongo('mongo3:27017')).getDB('test')
test
myrepl:SECONDARY> host3.setSlaveOk()
myrepl:SECONDARY> host3.myCollection.find()
{ "_id" : ObjectId("6093699ee8211843a1b954e8"), "testmsg" : "Hello from master!!" }
myrepl:SECONDARY>
```

<u>Docker Compose (Extra Credit)</u>

Compose is a tool for defining and running multi-container Docker applications. With Compose, we use a YAML file to configure your application's services. Then, with a single command, we create and start all the services from our configuration.

• Download stable release of Docker compose using the following command:

$ sudo curl -L "https://github.com/docker/compose/releases/download/1.25.5/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose



```
root@docker:~# curl -L "https://github.com/docker/compose/releases/download/1.25.5/docker-compose-$(
uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   633  100   633    0     0   2552      0 --:--:-- --:--:-- --:--:--  2552
100 16.7M  100 16.7M    0     0  16.1M      0  0:00:01  0:00:01 --:--:-- 16.1M
root@docker:~#
```

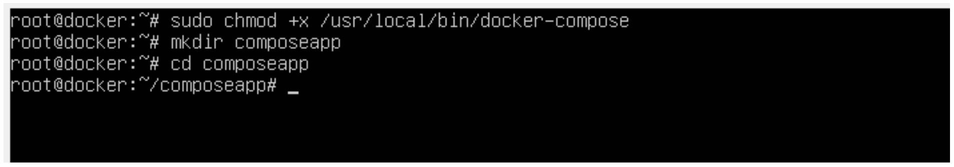
• Apply executable permissions to the binary:

$ sudo chmod +x /usr/local/bin/docker-compose

• Following application can be used to count the number of times a person visits a page. The application is built using Python Flask framework and maintains a counter in Redis that increments each time the page is visited. The count is maintained in cache.

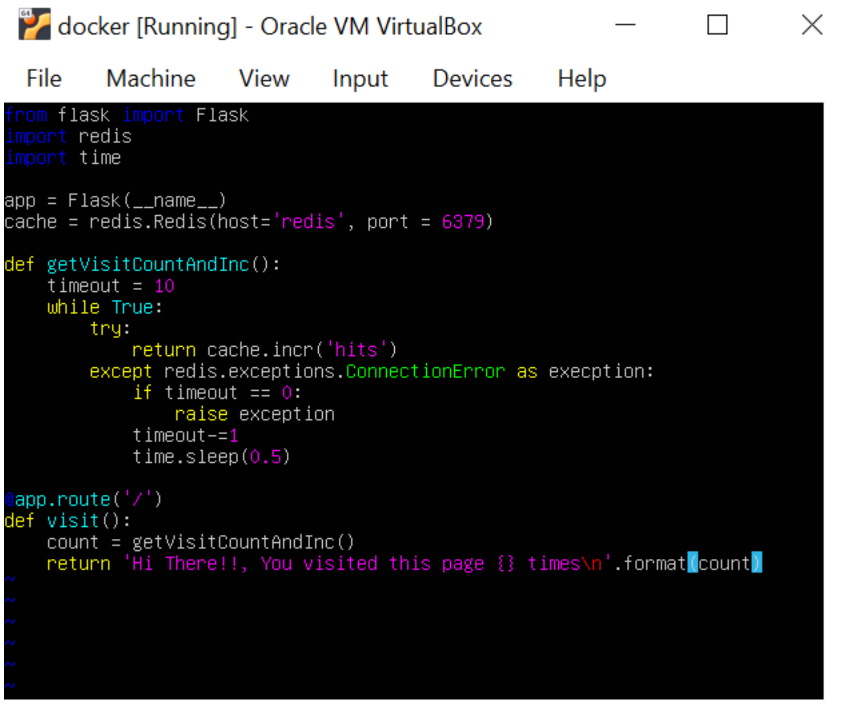
• Create a directory for the project:

$ mkdir composetest
$ cd composetest



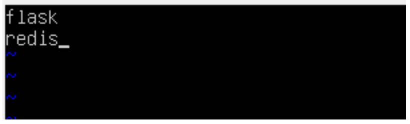
```
root@docker:~# sudo chmod +x /usr/local/bin/docker-compose
root@docker:~# mkdir composeapp
root@docker:~# cd composeapp
root@docker:~/composeapp# _
```

• Create the .py file with the application code in it. Place this file in the project directory:



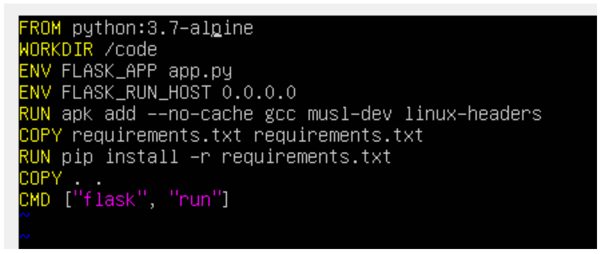
```
from flask import Flask
import redis
import time

app = Flask(__name__)
cache = redis.Redis(host='redis', port = 6379)

def getVisitCountAndInc():
    timeout = 10
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as execption:
            if timeout == 0:
                raise exception
            timeout-=1
            time.sleep(0.5)

@app.route('/')
def visit():
    count = getVisitCountAndInc()
    return 'Hi There!!, You visited this page {} times\n'.format(count)
~
~
~
~
~
~
```

• Create another file called requirements.txt in your project directory that contains requirements needed to run the project. Place this in project directory:



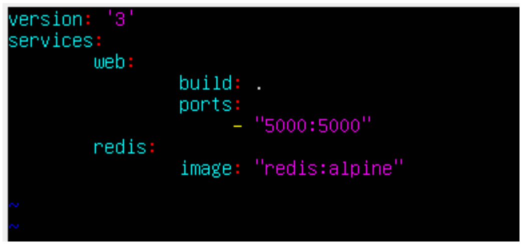
```
flask
redis_
~
~
~
```

• Create the Docker file that contains the workflow



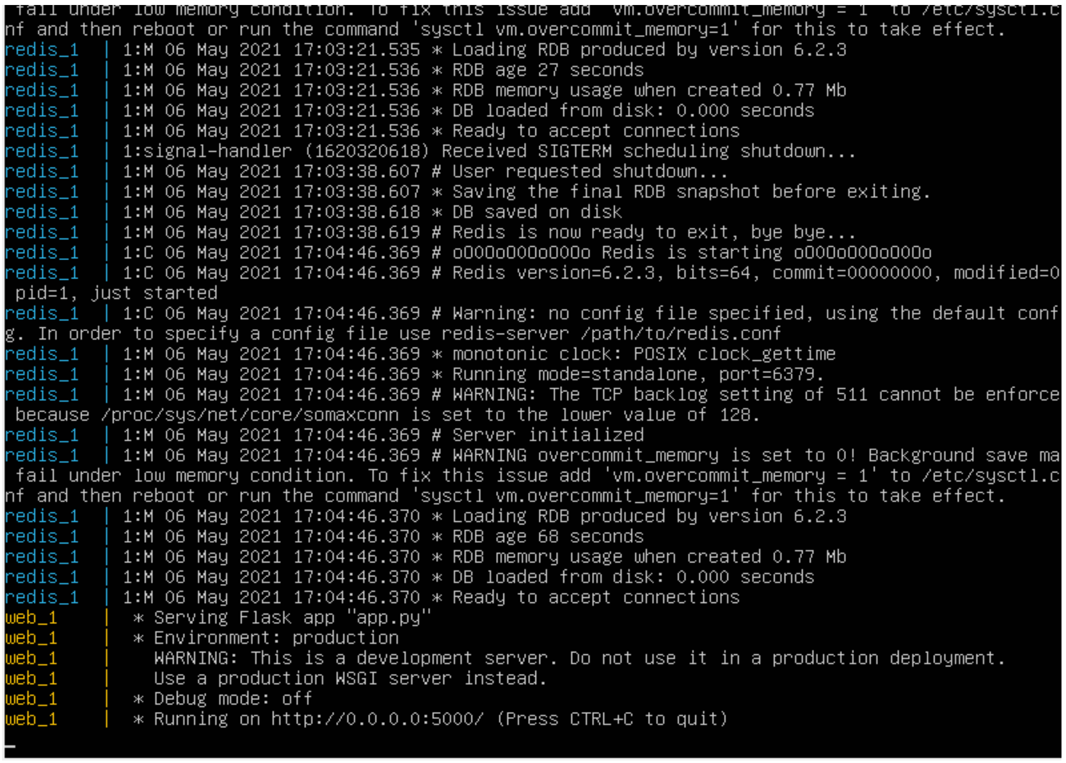
```
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP app.py
ENV FLASK_RUN_HOST 0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
COPY . .
CMD ["flask", "run"]
~
~
```

• Create a file docker-compose.yml file in our project directory and paste the following. This file is used for application startup.



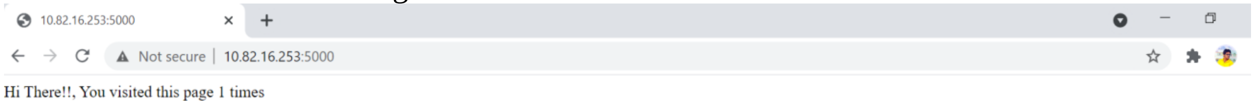
```
version: '3'
services:
        web:
                build: .
                ports:
                     - "5000:5000"
        redis:
                image: "redis:alpine"

~
~
```

• From the project directory start the application by running the following command.
$ docker-compose up



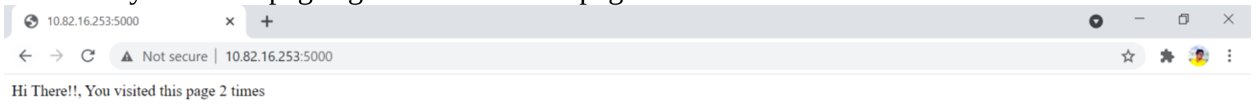
```
 fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.c
nf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
redis_1   | 1:M 06 May 2021 17:03:21.535 * Loading RDB produced by version 6.2.3
redis_1   | 1:M 06 May 2021 17:03:21.536 * RDB age 27 seconds
redis_1   | 1:M 06 May 2021 17:03:21.536 * RDB memory usage when created 0.77 Mb
redis_1   | 1:M 06 May 2021 17:03:21.536 * DB loaded from disk: 0.000 seconds
redis_1   | 1:M 06 May 2021 17:03:21.536 * Ready to accept connections
redis_1   | 1:signal-handler (1620320618) Received SIGTERM scheduling shutdown...
redis_1   | 1:M 06 May 2021 17:03:38.607 # User requested shutdown...
redis_1   | 1:M 06 May 2021 17:03:38.607 * Saving the final RDB snapshot before exiting.
redis_1   | 1:M 06 May 2021 17:03:38.618 * DB saved on disk
redis_1   | 1:M 06 May 2021 17:03:38.619 # Redis is now ready to exit, bye bye...
redis_1   | 1:C 06 May 2021 17:04:46.369 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
redis_1   | 1:C 06 May 2021 17:04:46.369 # Redis version=6.2.3, bits=64, commit=00000000, modified=0
 pid=1, just started
redis_1   | 1:C 06 May 2021 17:04:46.369 # Warning: no config file specified, using the default conf
g. In order to specify a config file use redis-server /path/to/redis.conf
redis_1   | 1:M 06 May 2021 17:04:46.369 * monotonic clock: POSIX clock_gettime
redis_1   | 1:M 06 May 2021 17:04:46.369 * Running mode=standalone, port=6379.
redis_1   | 1:M 06 May 2021 17:04:46.369 # WARNING: The TCP backlog setting of 511 cannot be enforce
 because /proc/sys/net/core/somaxconn is set to the lower value of 128.
redis_1   | 1:M 06 May 2021 17:04:46.369 # Server initialized
redis_1   | 1:M 06 May 2021 17:04:46.369 # WARNING overcommit_memory is set to 0! Background save ma
 fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.c
nf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
redis_1   | 1:M 06 May 2021 17:04:46.370 * Loading RDB produced by version 6.2.3
redis_1   | 1:M 06 May 2021 17:04:46.370 * RDB age 68 seconds
redis_1   | 1:M 06 May 2021 17:04:46.370 * RDB memory usage when created 0.77 Mb
redis_1   | 1:M 06 May 2021 17:04:46.370 * DB loaded from disk: 0.000 seconds
redis_1   | 1:M 06 May 2021 17:04:46.370 * Ready to accept connections
web_1     |  * Serving Flask app "app.py"
web_1     |  * Environment: production
web_1     |    WARNING: This is a development server. Do not use it in a production deployment.
web_1     |    Use a production WSGI server instead.
web_1     |  * Debug mode: off
web_1     |  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
_
```

- I am using a VM, so I used the bridge network with local machine to hit the server on the VM. 10.82.16.253 is that bridge IP.

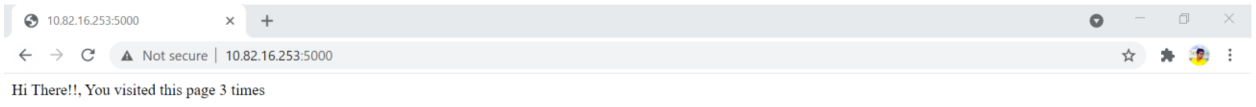


```
Hi There!!, You visited this page 1 times
```

- Try to hit the page again or refresh the page to see the increments.



```
Hi There!!, You visited this page 2 times
```



```
Hi There!!, You visited this page 3 times
```

Conclusion

This assignment helped us to learn about containers through installation and creation of docker containers with mongodb using docker file. A network has been established between the three containers that are created in which one being primary and other two replicas and successfully verified how operations on primary containers reflected on slave containers. So, we realize that dockers can boost the deployment process as they are simple and scalable which can also be easily configured and maintained. It also avoids the dependency of runtime environment as containers are isolated from each other which results in better control over traffic flow and management.

References

[1] https://opensource.com/resources/what-docker

[2] https://www.docker.com/resources/what-container