CS208 Assignment 2

Student Name: Zeyu Pang Student ID: 862187440

Experiment Environment

Tools: MacBook Pro

Operating System: MacOS Mojave 10.14.6

Checksum of the file "mydata.txt"

1. In the "Dockerfile" of server application, use "md5sum mydata.txt" to get the checksum of the file "mydata.txt".

```
FROM ubuntu:18.04
COPY . /server_storage
WORKDIR /server_storage
RUN md5sum mydata.txt
```

2. When we run "./run_server.sh", we can see the checksum of the "mydata.txt"

```
Step 4/7 : RUN md5sum mydata.txt
 ---> Running in 47f047c68211
0b5356fd07f8134c27cc181e72897b88  mydata.txt
```

The content of "mydata.txt" is:

```
Hello! This is the first assignment of the course CS208 in Winter 2021.
Hope to get a good result!
Happy New Year!
And let's learn more.
```

The screenshot of the server application running

We can see the information of the volume "server_persistent_storage" by using "docker volume inspect server_persistent_storage". Here, we can find two connections from the client.

```
server_persistent_storage
[
    {
        "CreatedAt": "2021-01-21T10:10:10Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/server_persistent_storage/_data",
        "Name": "server_persistent_storage",
        "Options": {},
        "Scope": "local"
    }
]
f3b1d2c24d4e87dfbf5cfbd435ea1f16f38700fa9f4ba7094c21e691a7c93757
======waiting for client's request=====
new client connect.
file open ends, start to send.
file sending ends.

new client connect.
file open ends, start to send.
file sending ends.
```

The screenshot of the client application running

We can see the information of the volume "client_persistent_storage" by using "docker volume inspect client_persistent_storage".

```
client_persistent_storage
[
    {
        "CreatedAt": "2021-01-21T23:25:59Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/client_persistent_storage/_data",
        "Name": "client_persistent_storage",
        "Options": {},
        "Scope": "local"
    }
]
```

Although we can create a volume “client_persistent_storage” for the client application, we can’t access to it directly by using “cd /var/lib/docker/volumes/client_persistent_storage/_data” because the docker in MacOS is a VM. So, the directory is not the real path.

But we can access it by using absolute path in “volume mount option” in docker run command in the “run_client.sh” of the client application:

“docker run --rm --network=server_network --name client_container -p 8001:8000 -v `pwd`:/client_storage -it client_image”

The connection result of the client application:

```
client receive file:(between ==== and ====)
====
Hello! This is the first assignment of the course CS208 in Winter 2021.
Hope to get a good result!
Happy New Year!
And let's learn more.
====
```

Then, we can use “ls” and find a new text file appeared in the current directory:

```
(base) pangzeyudeMacBook-Pro:client pangzeyu$ ls
Dockerfile              client.c                mydata_client_copy.txt  run_client.sh
```

We open another terminal and go the client directory and run “md5 mydata_client_copy.txt”:

```
(base) pangzeyudeMacBook-Pro:client pangzeyu$ md5 mydata_client_copy.txt
MD5 (mydata_client_copy.txt) = 0b5356fd07f8134c27cc181e72897b88
```

The steps and screenshot to reproduce the experiment

Server:

1. Go to the server directory and run “./run_server.sh”
2. Get the checksum of the mydata.txt in the command line.
3. Wait for the connection from the client.

Client:

1. Go to the client directory and run “./run_client.sh”
2. After the program finish, find a new file called “mydata_client_copy.txt” in the current directory.
3. Open another terminal and go to the client directory
4. Run “md5 mydata_client_copy.txt” and compare the checksum.

By comparing the server terminal and the third terminal, we can see that the two checksums are same.

```
Step 4/7 : RUN md5sum mydata.txt
 ---> Running in 47f047c68211
0b5356fd07f8134c27cc181e72897b88  mydata.txt
(base) pangzeyudeMacBook-Pro:client pangzeyu$ md5 mydata_client_copy.txt
MD5 (mydata_client_copy.txt) = 0b5356fd07f8134c27cc181e72897b88
```

Experiment Environment

Tools: CloudLab
Operating System on VMs: Ubuntu 18.04
Clusters: APT Utah

Checksum of the file "mydata.txt"

1. In the "Dockerfile" of server application, use "md5sum mydata.txt" to get the checksum of the file "mydata.txt".

```
#FROM gcc:4.9
FROM ubuntu:18.04
COPY . /server_storage
WORKDIR /server_storage
RUN md5sum mydata.txt
RUN apt-get update && \
    apt-get -y install gcc
RUN gcc -o server server.c
EXPOSE 8000
CMD ["./server"]
```

2. When we run "./run_server.sh", we can see the checksum of the "mydata.txt"

```
Step 4/8 : RUN md5sum mydata.txt
 ---> Running in 8ebae38dfaf8
0b5356fd07f8134c27cc181e72897b88  mydata.txt
```

The content of "mydata.txt" is:

```
Hello! This is the first assignment of the course CS208 in Winter 2021.
Hope to get a good result!
Happy New Year!
And let's learn more.
```

The screenshot of the server application running

We can see the information of the volume "server_persistent_storage" by using "docker volume inspect server_persistent_storage".

```
[
    {
        "CreatedAt": "2021-01-21T23:57:17-07:00",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/server_persistent_storage/_data",
        "Name": "server_persistent_storage",
        "Options": {},
        "Scope": "local"
    }
]
```

We can use “cd /var/lib/docker/volumes/server_persistent_storage/_data” to see that the data are stored in the volume “server_persistent_storage”:

```
root@vm0:/users/ZeyuPang/assignment2/server# cd /var/lib/docker/volumes/server_persistent_storage/_data
root@vm0:/var/lib/docker/volumes/server_persistent_storage/_data# ls
Dockerfile  mydata.txt  run_server.sh  server  server.c
root@vm0:/var/lib/docker/volumes/server_persistent_storage/_data#
```

We can also see a connection from the client.

```
======waiting for client's request=====
new client connect.
file open ends, start to send.
file sending ends.
```

The screenshot of the client application running

We can see the information of the volume “client_persistent_storage” by using “docker volume inspect client_persistent_storage”.

```
client_persistent_storage
[
    {
        "CreatedAt": "2021-01-21T23:25:59Z",
        "Driver": "local",
        "Labels": {},
        "Mountpoint": "/var/lib/docker/volumes/client_persistent_storage/_data",
        "Name": "client_persistent_storage",
        "Options": {},
        "Scope": "local"
    }
]
```

We can see the content of the “mydata.txt” in client application when the connection is built successfully:

```
Successfully tagged client_image:latest
client receive file:(between ==== and ====)
====
Hello! This is the first assignment of the course CS208 in Winter 2021.
Hope to get a good result!
Happy New Year!
And let's learn more.
====
root@vm0:/users/ZeyuPang/assignment2/client# ls
client.c  Dockerfile  run_client.sh
```

In Linux OS, we can use “cd /var/lib/docker/volumes/client_persistent_storage/_data” to see that the data are stored in the volume “client _persistent_storage”:

```
[root@vm0:/users/ZeyuPang/assignment2/client# ls                              ]
 client.c  Dockerfile  run_client.sh
[root@vm0:/users/ZeyuPang/assignment2/client# cd /var/lib/docker/volumes/client_p]
 ersistent_storage/_data
[root@vm0:/var/lib/docker/volumes/client_persistent_storage/_data# ls        ]
 client  client.c  Dockerfile  mydata.txt  run_client.sh
```

We open another terminal and go the client directory and run “md5sum mydata.txt”:

```
[root@vm0:/var/lib/docker/volumes/client_persistent_storage/_data# md5sum mydata.]
txt
0b5356fd07f8134c27cc181e72897b88  mydata.txt
```

The steps and screenshot to reproduce the experiment

Server:

1. Use “sudo passwd root” to input a new password of the root mode.
2. Use “su root” to change to root mode.
3. Go to the server_linux directory and run “chmod +x run_server.sh”.
4. Run “./run_server.sh” to start the server application.
5. Get the checksum of the mydata.txt in the command line.
6. Wait for the connection from the client.

Client:

1. Use “sudo passwd root” to input a new password of the root mode.
2. Use “su root” to change to root mode.
3. Go to the client_linux directory and run “chmod +x run_client.sh”.
4. Run “./run_client.sh” to start the client application.
5. After the program finish, open another terminal and use “cd /var/lib/docker/volumes/client_persistent_storage/_data” to go to the directory of the volume “client_persistent_storage”.
6. Run “ls” and find a new text file “mydata.txt” in the volume “client_persistent_storage”.
7. Run “md5sum mydata.txt” and compare the checksum.

By comparing the server terminal and the third terminal, we can see that the two checksums are same.

```
Step 4/8 : RUN md5sum mydata.txt
 ---> Running in 8ebae38dfaf8
0b5356fd07f8134c27cc181e72897b88  mydata.txt
[root@vm0:/var/lib/docker/volumes/client_persistent_storage/_data# md5sum mydata.]
txt
0b5356fd07f8134c27cc181e72897b88  mydata.txt
```

Question unsolved:

1. One strange thing is that I use “COPY . /client_storage” and “WORKDIR /client_storage” in Dockerfile of the client and use “docker run --rm --name client_container -v client_persistent_storage:/client_storage -it client_image” in

"run_client.sh" in Ubuntu. But there is no error and the application runs well. However, when I use the same configuration in MacOS, it has error and I have to change the directory "/client_storage" in Dockerfile to others such as "/client_app" to avoid the repetition between "COPY . /client_storage" and "client_persistent_storage:/client_storage".

2. Another strange thing is that when I add option "—network" in "docker run" command in the server "run_server.sh", I can't use "docker inspect --format='{{.NetworkSettings.IPAddress}}' server_container_and_dns_name" to get the IP address of the server container. But if I don't add that option, I can use this command to help client to get the server IP address directly without creating a network by using "docker network create". I guess the reason comes from the docker default network bridge.

Outside the Homework:

Use overlay network to connect two containers in different machines.

Reference resource:

https://docs.docker.com/network/network-tutorial-overlay/ (two: "Use the default overlay network" and "Use an overlay network for standalone containers" works, the last one "Communicate between a container and a swarm service" title and content are different.)

Current fact: If we didn't use swarm as a manager, we can't create an overlay network on non-manager node. Even we can create an overlay network on manager node(which in a swarm), a node which is outside the swarm can't find the overlay network created by the manager in the swarm.

Copy the Mac version "server" and "client" directories to the VMs.

Experiment Environment

Tools: CloudLab

Operating System on 3 VMs: Ubuntu 18.04

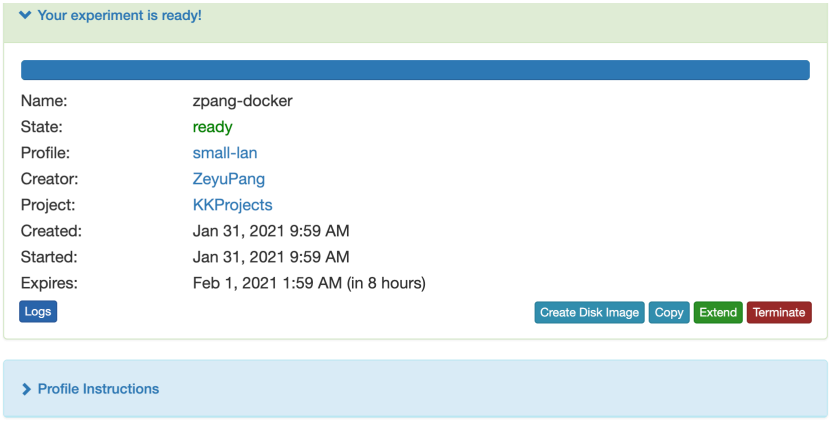
Clusters: APT Utah

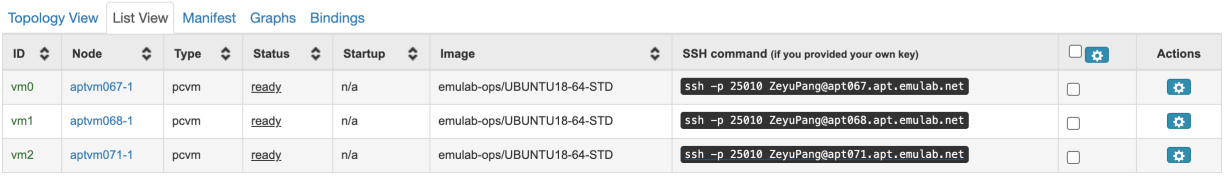
Attention!

1. When we want to drag file directly to the directory of VM in VScode, the destination directory must be created in normal user. If we input "suso -s" to enter a root mode and create a file by using "mkdir asg2", then we can't drag files to this directory because it's write-protected. We need to go back to normal user mode by using "exit" and then create a directory by using "mkdir asg2" in normal user mode. Then, we can directly drag files to this directory in VScode remote.
2. When we run the server, it will wait for the request from the client all the time. So, if we want to go back to the command line, we need to

press “Ctrl” and hold it, then press “p” and “q” to go back to the terminal mode.

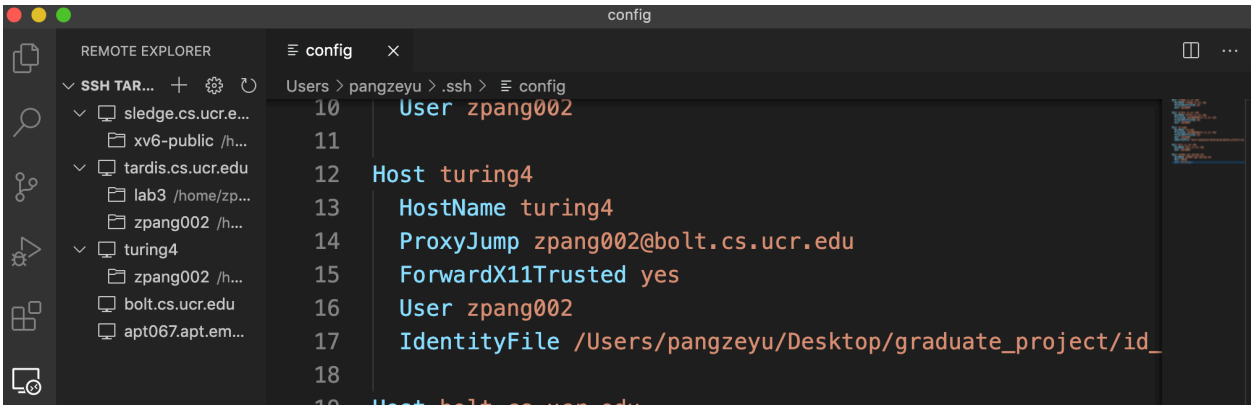
3. After we applied the VMs from CloudLab, we can click the “SSH Command” in List View to access each VM in a local terminal.



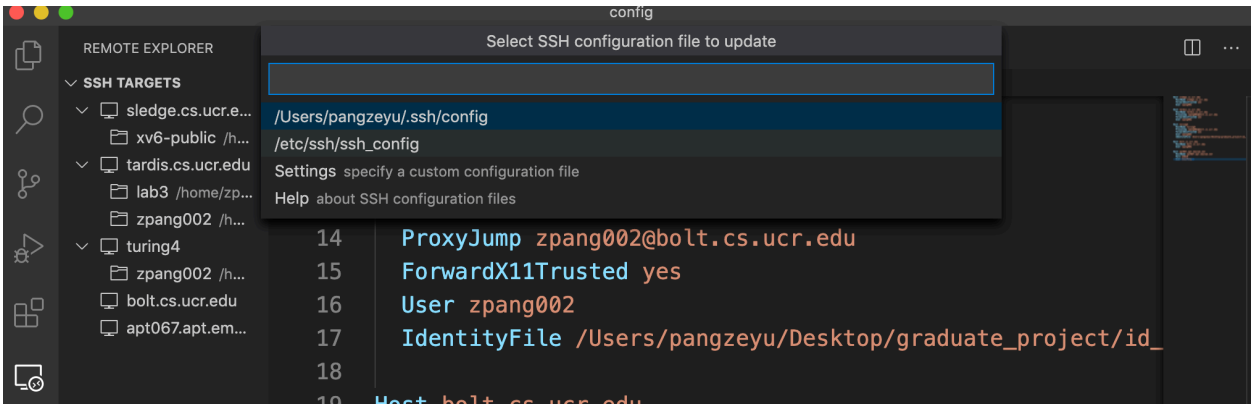


4. We can also access VM by using VScode Remote Explorer. (the way used in attention 1) (step 5 -11 below)

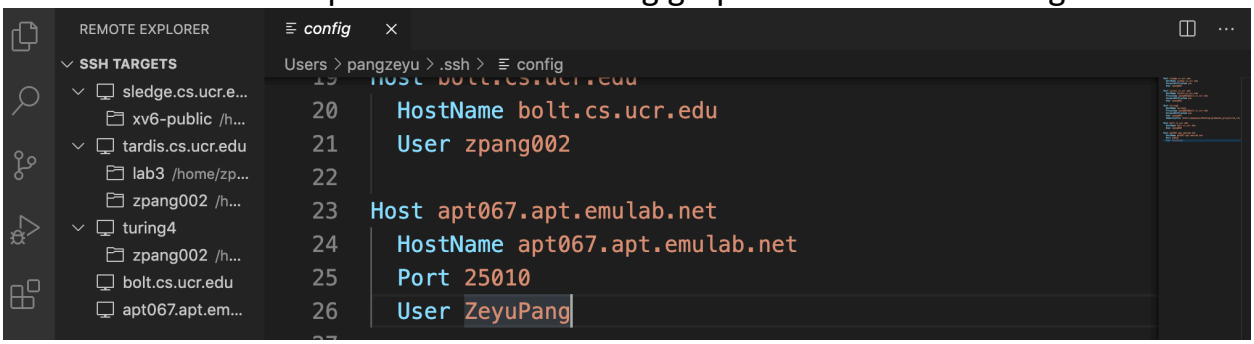
5. When we place our mouse on “SSH targets” region, we can click the “gear” icon on the right part.



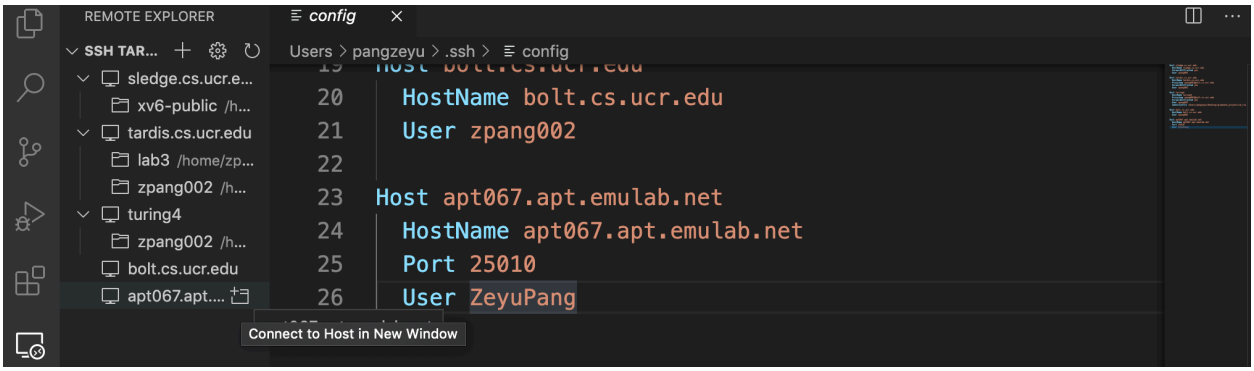
6. After that, we can see a list and choose the first one “/ssh/config” to access this file.



7. After we see the config file, we can add information in the “SSH Command” part like the following graph and store the change.



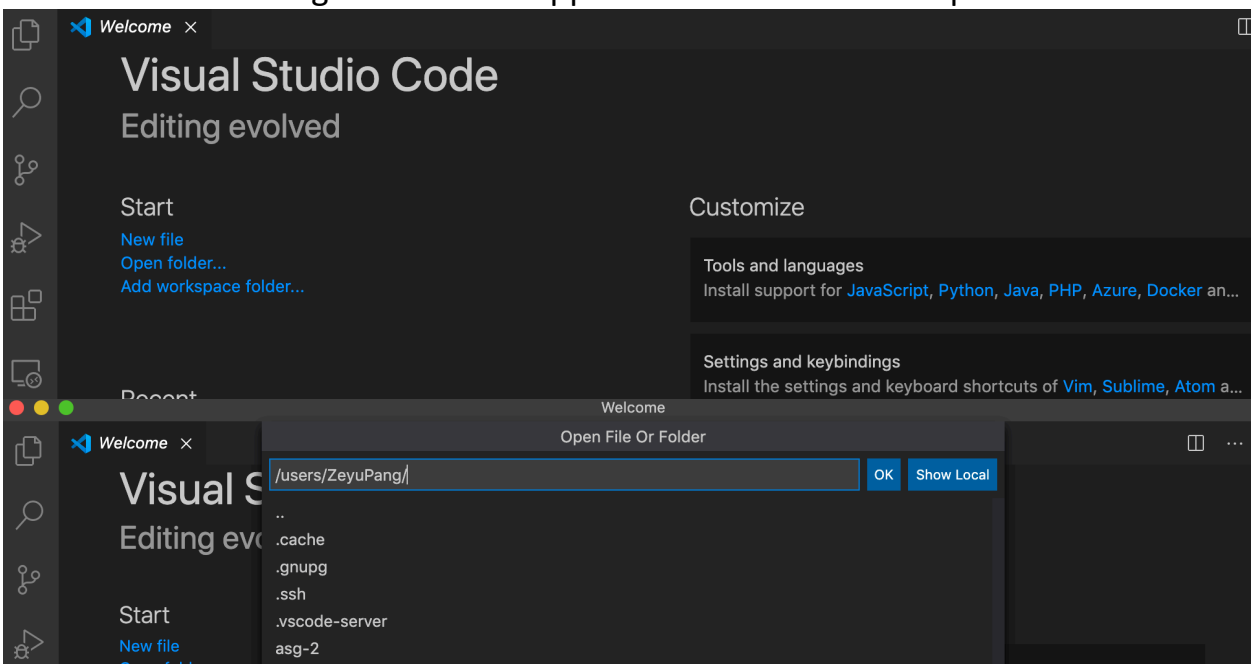
8. When these are done, we can click the button on the left to connect to it in a new window. (here on the right of “apt067.apt..” in the left)



9. Before we access it in a directory, we use “mkdir asg2” in the VM terminal and normal user mode ($) to create a directory to help us to find a place to locate in VScode.

```
root@vm0:~# exit
exit
ZeyuPang@vm0:~$ mkdir asg-2
ZeyuPang@vm0:~$ ls
asg-2
ZeyuPang@vm0:~$ cd asg-2/
```

10. In VScode new window, we click “Open folder” in the “Start” region and choose “asg-2” in the list appeared and click “OK” or press “ENTER”.



11. Now, in this new directory, we can directly drag files or directory to the blue area in this graph to add files or directory to the VM.

The steps and screenshot to reproduce the experiment

Server:

1. Install Docker Engine https://docs.docker.com/engine/install/ubuntu/
2. Use “docker swarm init --advertise-addr=172.17.67.1” to set up a swarm (here “use --advertise-addr” because VM has more than one address. Copy the command it shows to add a worker for later use in client.
3. Use “docker network create --driver=overlay --attachable test-net” to create an overlay network and set it attachable to others in the swarm. (If we don’t add “--attachable”, the “docker run --network=test-net” in both run_server.sh and run_client.sh can’t connect to this overlay network in this swarm)
4. Go to the “server” directory and change the command “docker run --rm --network=server_network…” in run_server.sh to “docker run --rm --network=test-net…” and add “#” in front of “docker network rm server_network” and “docker network create server_network”.
5. Use “sudo -s” to change to root mode.
6. Go to the “server” directory and run “chmod +x run_server.sh”.
7. Run “./run_server.sh” to start the server application.
8. Get the checksum of the mydata.txt in the command line.
9. Wait for the connection from the client.

Client:

1. Install Docker Engine https://docs.docker.com/engine/install/ubuntu/
2. Use the command copied before in the server to join this VM to the swarm we set up before and add the chosen address of the client VM. (docker swarm join --token SWMTKN-1-1yrfw8udyizwrxoa8k3xx4viu544f38ouqye2jheb616wx4pdm-6z25cdpcaj7vmipyq27wrdlf9 --advertise-addr 172.17.71.1 172.17.67.1:2377)(here “--advertise-addr 172.17.71.1” is the worker’s IP address, “172.17.67.1:2377” is the manager’s address and the TCP port)(if you forgot the copy that command before, you can input “docker swarm join-token worker” in the manager node(server) to see it again).
3. Use “sudo passwd root” to input a new password of the root mode.
4. Use “su root” to change to root mode.

5. Go to the client_linux directory and run “chmod +x run_client.sh”.
6. Run “./run_client.sh” to start the client application.
7. Open another terminal and go to the client directory
8. Run “md5 mydata_client_copy.txt” and compare the checksum.

About Docker Swarm:

If a worker inputs “docker swarm leave”, it will leave the swarm. In manager part, “docker node ls” will find that this work has a STATUS of “DOWN”.

The manager can delete a worker node by using “docker node rm up1n7ewfzx4oqnp6d3quvqw8l” (“up1n7ewfzx4oqnp6d3quvqw8l” is ID) or “docker node rm vm1.zpang-docker.kkprojects-pg0.apt.emulab.net” (“vm1.zpang-docker.kkprojects-pg0.apt.emulab.net” is HOSTNAME) only after the worker node use “docker swarm leave” to leave and the STATUS in manager is “DOWN”.

```
root@vm0:~/asg-2/server# docker node ls
ID                            HOSTNAME                                        STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
s628rbkcc8qw711zlu0ys1m8j *   vm0.zpang-docker.kkprojects-pg0.apt.emulab.net   Ready     Active         Leader           20.10.2
vk0nzon9v0j1xxmjikya2z1mc     vm1.zpang-docker.kkprojects-pg0.apt.emulab.net   Down      Active                          20.10.2
yetq7fmzmnnhy7wqjcomyk2r3     vm2.zpang-docker.kkprojects-pg0.apt.emulab.net   Ready     Active                          20.10.2
root@vm0:~/asg-2/server# docker node rm vm1.zpang-docker.kkprojects-pg0.apt.emulab.net
vm1.zpang-docker.kkprojects-pg0.apt.emulab.net
root@vm0:~/asg-2/server# docker node ls
ID                            HOSTNAME                                        STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
s628rbkcc8qw711zlu0ys1m8j *   vm0.zpang-docker.kkprojects-pg0.apt.emulab.net   Ready     Active         Leader           20.10.2
yetq7fmzmnnhy7wqjcomyk2r3     vm2.zpang-docker.kkprojects-pg0.apt.emulab.net   Ready     Active                          20.10.2
```