Problem Statement: To install docker, build two images: one for server and the other for client. Mount a volume for the server, and another for the client. Create a user defined network and connect the containers to them with ports specified on the command line. Once the Docker run command is given, the server container must immediately generate a text file of size 1KB, store it in the volume and then the client upon connecting to the server, must receive the text file along with the checksum to verify the integrity of the file. Once the file is received by the client, its integrity should be verified.

Server:

First, create a volume by name "servervol"
-> docker volume create servervol

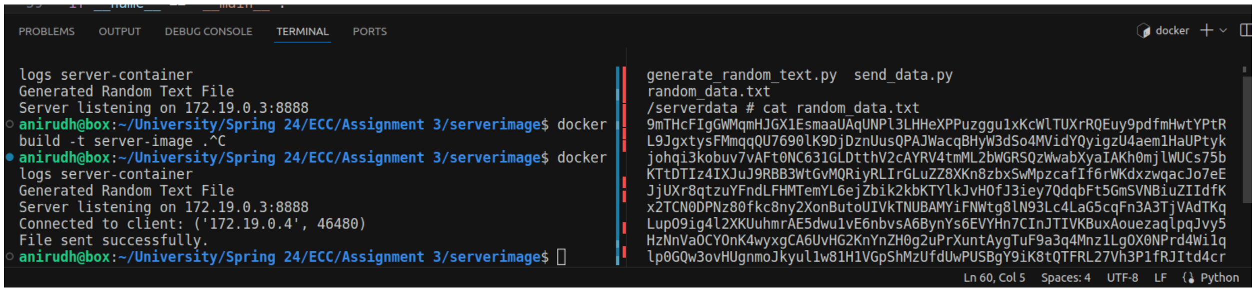
Next, mount the "servervol" in "/serverdata"
-> This will be done in the Docker run command itself. It will be mentioned at the final part of the server.

Container should run an application on startup which will create a file of size 1KB with random text data and store it in "/serverdata"
-> To generate the text file, I used the random library file in python to generate random text of size 1024 and store it in the directory "/serverdata". The file generate_random_text.py has the code to this.

The container should include all the packages that are required to run your application. Choose an appropriate base image and install only the necessary packages
-> The base image I chose was python:alpine. The reasoning behind this is python's docker images come with all the basic pip packages installed to run python based applications. And as for alpine, it's for the sake of a lightweight file system for running other commands in interactable mode such as ping, ssh and so on. I did not have to specify any additional python libraries to be installed as the python:alpine base image already had all the necessary dependencies installed.

The port on which the server runs must be specified as a command line argument when we run docker
-> The port I exposed in the Dockerfile is 8888.

Commands used for the Server Container:
-> docker build -t server-image .
-> docker tag anpenma/server-image anpenma/server-image:latest
-> docker push anpenma/server-image
-> docker run -itd -v servervol:/serverdata --network Docker-Network -p 8888:8888 --name server-container server-image
-> docker exec -it server-container sh



```
logs server-container
Generated Random Text File
Server listening on 172.19.0.3:8888
anirudh@box:~/University/Spring 24/ECC/Assignment 3/serverimage$ docker
build -t server-image .^C
anirudh@box:~/University/Spring 24/ECC/Assignment 3/serverimage$ docker
logs server-container
Generated Random Text File
Server listening on 172.19.0.3:8888
Connected to client: ('172.19.0.4', 46480)
File sent successfully.
anirudh@box:~/University/Spring 24/ECC/Assignment 3/serverimage$
```

```
generate_random_text.py  send_data.py
random_data.txt
/serverdata # cat random_data.txt
9mTHcFIgGWMqmHJGX1EsmaaUAqUNPl3LHHeXPPuzggu1xKcWlTUXrRQEuy9pdfmHwtYPtR
L9JgxtysFMmqqQU7690lK9DjDznUusQPAJWacqBHyW3dSo4MVidYQyigzU4aem1HaUPtyk
johqi3kobuv7vAFt0NC631GLDtthV2cAYRV4tmML2bWGRSQzWwabXyaIAKh0mjlWUCs75b
KTtDTIz4IXJuJ9RBB3WtGvMQRiyRLIrGLuZZ8XKn8zbxSwMpzcafIf6rWKdxzwqacJo7eE
JjUXr8qtzuYFndLFHMTemYL6ejZbik2kbKTYlkJvHOfJ3iey7QdqbFt5GmSVNBiuZIIdfK
x2TCN0DPNz80fkc8ny2XonButoUIVkTNUBAMYiFNWtg8lN93Lc4LaG5cqFn3A3TjVAdTKq
LupO9ig4l2XKUuhmrAE5dwu1vE6nbvsA6BynYs6EVYHn7CInJTIVKBuxAouezaqlpqJvy5
HzNnVaOCYOnK4wyxgCA6UvHG2KnYnZH0g2uPrXuntAygTuF9a3q4Mnz1LgOX0NPrd4Wi1q
lp0GQw3ovHUgnmoJkyul1w81H1VGpShMzUfdUwPUSBgY9iK8tQTFRL27Vh3P1fRJItd4cr
```

Client:

First, create a volume by name "clientvol"
-> docker volume create clientvol

Next, mount the "clientvol" in "/clientdata"
-> This will be done in the Docker run command itself. It will be mentioned at the final part of the server.

Container should run an application on startup which will connect to the server, receive the file that the server sends and save it in "/clientdata"
-> The script receive_data

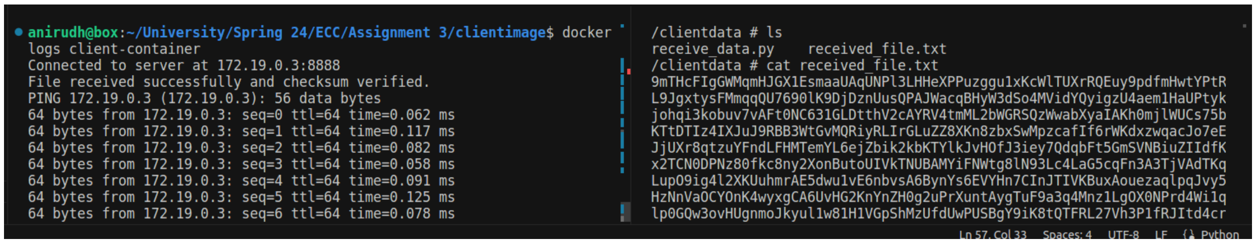
Verify that the file is received properly at the clientside by verifying the checksum
-> To do this hashlib has a function called hexdigest that can check if it's the same as the checksum

The container should include all the packages that are required to run your application. Choose an appropriate base image and install only the necessary packages
-> The base image I chose for the client is the same as the server. And it comes with all necessary packages already installed.

Commands used for the Client Container:
-> docker build -t client-image .
-> docker tag anpenma/client-image anpenma/client-image:latest
-> docker push anpenma/client-image
-> docker run -v clientvol:/clientdata -itd --network Docker-Network -p 8786:8786 --name client-container client-image
-> docker exec -it client-container sh

You need to create a user-defined network in docker and run both these containers on the network created. The containers should run these applications by default (i.e, on run command).
-> docker network create Docker-Network

You should be able to get into the shell of the client container to physically check if the file has been received.



```
anirudh@box:~/University/Spring 24/ECC/Assignment 3/clientimage$ docker
logs client-container
Connected to server at 172.19.0.3:8888
File received successfully and checksum verified.
PING 172.19.0.3 (172.19.0.3): 56 data bytes
64 bytes from 172.19.0.3: seq=0 ttl=64 time=0.062 ms
64 bytes from 172.19.0.3: seq=1 ttl=64 time=0.117 ms
64 bytes from 172.19.0.3: seq=2 ttl=64 time=0.082 ms
64 bytes from 172.19.0.3: seq=3 ttl=64 time=0.058 ms
64 bytes from 172.19.0.3: seq=4 ttl=64 time=0.091 ms
64 bytes from 172.19.0.3: seq=5 ttl=64 time=0.125 ms
64 bytes from 172.19.0.3: seq=6 ttl=64 time=0.078 ms
```

```
/clientdata # ls
receive_data.py    received_file.txt
/clientdata # cat received_file.txt
9mTHcFIgGWMqmHJGX1EsmaaUAqUNPl3LHHeXPPuzggu1xKcWlTUXrRQEuy9pdfmHwtYPtR
L9JgxtysFMmqqQU7690lK9DjDznUusQPAJWacqBHyW3dSo4MVidYQyigzU4aem1HaUPtyk
johqi3kobuv7vAFt0NC631GLDtthV2cAYRV4tmML2bWGRSQzWwabXyaIAKh0mjlWUCs75b
KTtDTIz4IXJuJ9RBB3WtGvMQRiyRLIrGLuZZ8XKn8zbxSwMpzcafIf6rWKdxzwqacJo7eE
JjUXr8qtzuYFndLFHMTemYL6ejZbik2kbKTYlkJvHOfJ3iey7QdqbFt5GmSVNBiuZIIdfK
x2TCN0DPNz80fkc8ny2XonButoUIVkTNUBAMYiFNWtg8lN93Lc4LaG5cqFn3A3TjVAdTKq
LupO9ig4l2XKUuhmrAE5dwu1vE6nbvsA6BynYs6EVYHn7CInJTIVKBuxAouezaqlpqJvy5
HzNnVaOCYOnK4wyxgCA6UvHG2KnYnZH0g2uPrXuntAygTuF9a3q4Mnz1LgOX0NPrd4Wi1q
lp0GQw3ovHUgnmoJkyul1w81H1VGpShMzUfdUwPUSBgY9iK8tQTFRL27Vh3P1fRJItd4cr
```