ECC - Assignment 3

by

(sdiware@iu.edu)

Docker Installation:

1. I have used Jetstream because it has preinstalled docker in root user.

```
exouser@sdiware:~/Assignment3$ docker --version
Docker version 24.0.5, build 24.0.5-0ubuntu1~22.04.1
exouser@sdiware:~/Assignment3$
```

Note: Since the docker was installed on root user to access the volumes I had to use sudo commands.

Directory Structure:

Below is the directory which I created to perform this assignment.

1. As seen in screenshot below, I have created a folder Assignment3, which consists of docker-compose.yml, serverFolder and clientFolder.
2. Inside of clientFolder I have created a dockerfile.client and client.py.
3. Inside of serverFolder I have created a dockerfile.server and server.py.

```
exouser@sdiware:~/Assignment3$ ls -lart
total 20
-rw-rw-r--  1 exouser exouser  575 Dec  8 22:31 docker-compose.yml
drwxrwxr-x  2 exouser exouser 4096 Dec  8 23:55 serverFolder
drwxrwxr-x  4 exouser exouser 4096 Dec  8 23:56 .
drwxrwxr-x  2 exouser exouser 4096 Dec  8 23:56 clientFolder
drwxr-x--- 17 exouser exouser 4096 Dec  8 23:56 ..
exouser@sdiware:~/Assignment3$ cd clientFolder/
exouser@sdiware:~/Assignment3/clientFolder$ ll
total 16
drwxrwxr-x 2 exouser exouser 4096 Dec  8 23:56 ./
drwxrwxr-x 4 exouser exouser 4096 Dec  8 23:56 ../
-rw-rw-r-- 1 exouser exouser  318 Dec  8 23:56 Dockerfile.client
-rw-rw-r-- 1 exouser exouser 1269 Dec  8 23:49 client.py
exouser@sdiware:~/Assignment3/clientFolder$ cd ../serverFolder/
exouser@sdiware:~/Assignment3/serverFolder$ ll
total 16
drwxrwxr-x 2 exouser exouser 4096 Dec  8 23:55 ./
drwxrwxr-x 4 exouser exouser 4096 Dec  8 23:56 ../
-rw-rw-r-- 1 exouser exouser  141 Dec  8 23:52 Dockerfile.server
-rw-rw-r-- 1 exouser exouser 1037 Dec  8 23:50 server.py
exouser@sdiware:~/Assignment3/serverFolder$
```

Files:

1. Docker-compose.yml:
   - → Below is the screenshot of docker-compose.yml.
   - → It defines two services, "server" and "client," connected to a custom network named "mydockerNetwork."
   - → Each service utilizes named volumes ("servervol" and "clientvol").
   - → Volumes are used to persist data for the "server" and "client" services in the /serverdata and /clientdata
   - → The "client" service depends on the "server" service, ensuring proper service startup order.

```yaml
version: '3'

services:
      server:
        build:
          context: ./serverFolder
          dockerfile: Dockerfile.server
        networks:
          - mydockerNetwork
        volumes:
          - servervol:/serverdata

      client:
        build:
          context: ./clientFolder
          dockerfile: Dockerfile.client
        networks:
          - mydockerNetwork
        volumes:
          - clientvol:/clientdata
        depends_on:
          - server

networks:
  mydockerNetwork:
    driver: bridge
    name: mydockerNetwork

volumes:
  servervol:
  clientvol:
~
```

2. Dockerfile.server:
   - → This Dockerfile sets up a Python 3.9 environment, defines the working directory as /app, and specifies a volume at /serverdata.
   - → It copies a Flask-based Python script named server.py into the container's /app directory, installs the Flask library.

→ It specifies the default command to run the Flask server on port 8080 when the container starts.

```
FROM python:3.9

WORKDIR /app

VOLUME /serverdata

COPY server.py /app/server.py

RUN pip install Flask

CMD ["python", "server.py", "8080"]
```

3. Server.py:
   → It creates a Flask web application with a single route ('/') that generates a random file (randomFile.txt),
   → Calculates its SHA-256 checksum and accepts the file with the checksum included in the response headers.
   → The server listens on the specified port 8080.

```
from flask import Flask, send_file, make_response
import os
import hashlib
import string
import random
import sys

app = Flask(__name__)

@app.route('/')
def serveRandomFile():
    filePath = '/serverdata/randomFile.txt'
    checksum = generateRandomFile(filePath)
    response = make_response(send_file(filePath, as_attachment=True, download_name='randomFile.txt', mimetype='application/octet-stream'))
    response.headers['Checksum'] = checksum
    return response

def generateRandomFile(filePath):
    with open(filePath, 'w') as file:
        characters = string.ascii_letters + string.digits
        fileContent = ''.join(random.choice(characters) for _ in range(1024))
        file.write(fileContent)

    # Calculate checksum (SHA-256 in this example)
    sha256 = hashlib.sha256()
    sha256.update(fileContent.encode('utf-8'))
    return sha256.hexdigest()

if __name__ == '__main__':
    serverPort = int(sys.argv[1]) if len(sys.argv) > 1 else 8080
    app.run(host='0.0.0.0', port=serverPort)
```

4. Dockerfile.client
   → This Dockerfile sets up a Python 3.9 environment for a client application, installs the requests library, copies a Python script (client.py) into the container's /app directory.
   → It defines a volume at /clientdata.

```
FROM python:3.9

WORKDIR /app

RUN pip install requests

COPY client.py /app/client.py

VOLUME /clientdata

CMD ["python", "client.py"]
```

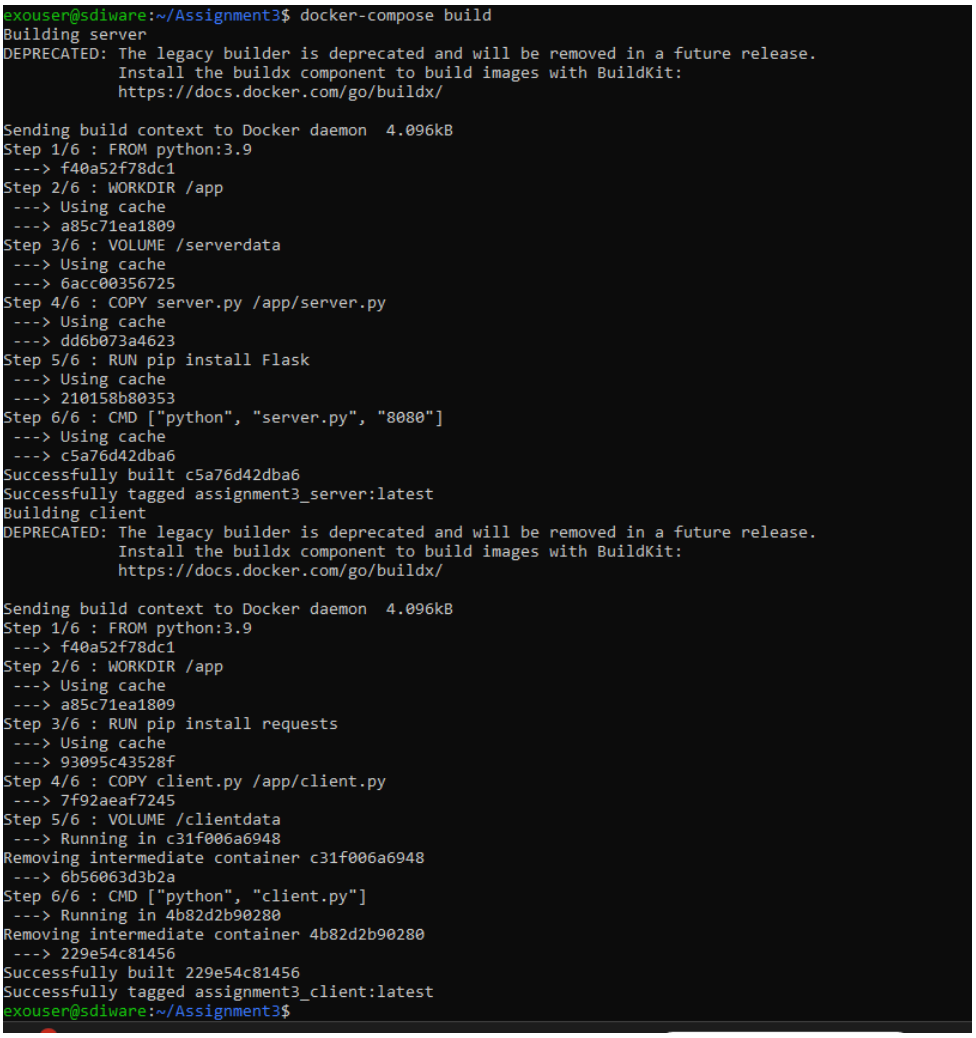
5. Client.py:
   - → It defines a client application that gets a file with checksum from a specified server URL (http://server:8080/).
   - → The file is saved at /clientdata/receivedFile.txt, and its checksum is verified using SHA-256.
   - → If the checksum verification fails, an error message is printed.

```python
import requests
import hashlib
import time
import os

SERVER_URL = "http://server:8080/"
FILE_PATH = '/clientdata/receivedFile.txt'

def downloadFile(url):
    response = requests.get(url)
    if response.status_code == 200:
        with open(FILE_PATH, 'wb') as file:
            file.write(response.content)
        return FILE_PATH, response.headers.get('Checksum')
    else:
        return None, None

def calculateChecksum(filePath):
    sha256 = hashlib.sha256()
    with open(filePath, 'rb') as file:
        for chunk in iter(lambda: file.read(8192), b''):
            sha256.update(chunk)
    return sha256.hexdigest()

def verification(filePath, recievedChecksum):
    actual_checksum = calculateChecksum(filePath)
    return actual_checksum == recievedChecksum

def main():
    downloadedFile, recievedChecksum = downloadFile(SERVER_URL)
    if downloadedFile and recievedChecksum:
        if verification(downloadedFile, recievedChecksum):
            print("-------- File received and Checksum verified. --------")
        else:
            print("Checksum verification failed.")
    else:
        print("Error downloading file")

    while True:
        time.sleep(10)

if __name__ == '__main__':
    main()
```

Questions:

Build and Run your server and client container.

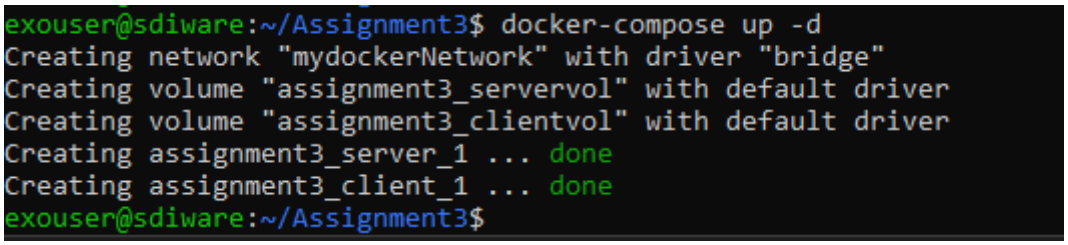
- → I have used the docker-compose.yml to define and configure multi-container.
- → Using docker-compose build as below I have built and run my containers – assignment3-client_1 and assignment3-server_1.
- → The docker-compose build command is used to build or rebuild the docker images defined in my docker compose file.
- → When I run this command in the directory where my docker-compose.yml file is located, it reads the configuration and builds images for all the services defined in the file.



→ After that I executed the command **docker-compose up -d** to start the services mentioned in the docker compose file.

→ I have used -d flag which is detached mode use to run services in the background.

→ Below is the screenshot of docker-compose up -d and as we can see, network **mydockerNetwork** is created, with two volumes **assignment3_servervol** and **assignment3_clientvol** and two services **assignment3_server_1** and **assignment3_client_1**.



→ Below is the screenshot of docker-compose ps command and we can see both containers are up and running.

```
exouser@sdiware:~/Assignment3$ docker-compose ps
        Name                      Command          State    Ports
-------------------------------------------------------------------
assignment3_client_1   python client.py          Up
assignment3_server_1   python server.py 8080     Up
exouser@sdiware:~/Assignment3$
```

Communication between the two:

- → Using the client.py and server.py the communication is happening between the containers, and it can be seen using the file transfer that is happening within the network between the client and server containers.
- → I have created a network named **mydockerNetwork** as seen in the below screenshot.

```
exouser@sdiware:~/Assignment3/clientFolder$ docker network ls
NETWORK ID     NAME                    DRIVER    SCOPE
900fa06adce4   bridge                  bridge    local
6cd018d4f906   guacamole_exo-guac-net  bridge    local
209cb00891c2   host                    host      local
08d769dcbf3e   mydockerNetwork         bridge    local
dda764d9e437   none                    null      local
```

- → Both my containers **assignment3_server_1** and **assignment3_client_1** is running the same network mydockerNetwork which can be seen in the screenshot below.
- → I have configured all of these in docker-compose.yml as mentioned in File section above.


```
exouser@sdiware:~/Assignment3$ docker network inspect mydockerNetwork
[
    {
        "Name": "mydockerNetwork",
        "Id": "08d769dcbf3e32af6f1e67345d792b9291a7fb3e4f2f647d70512e2bf3fc22c8",
        "Created": "2023-12-08T23:58:20.057698235Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.20.0.0/16",
                    "Gateway": "172.20.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": true,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "a6e00d6fcbc3657d8988e4e9e68e49c96f4ce838a01d74fe4d8699c20cd29e3f": {
                "Name": "assignment3_client_1",
                "EndpointID": "6648bcd03cdbb922ef37c82b9f4040e05807c56af19779c0cf8914a48fb37fea",
                "MacAddress": "02:42:ac:14:00:03",
                "IPv4Address": "172.20.0.3/16",
                "IPv6Address": ""
            },
            "fb646a2a623a9a11048927a4d4988f06e1d61c3a92b66ae9303ca6930659f28c": {
                "Name": "assignment3_server_1",
                "EndpointID": "c2eeabe86ff10d1c5dab612966d3495ce515ab914e72c312752b2579d66ec241",
                "MacAddress": "02:42:ac:14:00:02",
                "IPv4Address": "172.20.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {
            "com.docker.compose.network": "mydockerNetwork",
            "com.docker.compose.project": "assignment3",
            "com.docker.compose.version": "1.29.2"
        }
    }
]
exouser@sdiware:~/Assignment3$
```

→ I also created two volumes **assignment3_clientvol** and **assignment3_servervol** as shown below to store the container data.

```
exouser@sdiware:~/Assignment3/clientFolder$ docker volume ls
DRIVER    VOLUME NAME
local     assignment3_clientvol
local     assignment3_servervol
```

→ Below is screenshot of volumes prior to creating the volumes for the assignment.

```
root@sdiware:~# cd /var/lib/docker/volumes/
root@sdiware:/var/lib/docker/volumes# ll
total 32
drwx-----x  2 root root  4096 Dec  8 21:53 ./
drwx--x--- 12 root root  4096 Dec  8 21:53 ../
brw-------  1 root root  8, 1 Dec  8 21:53 backingFsBlockDev
-rw-------  1 root root 32768 Dec  8 21:53 metadata.db
root@sdiware:/var/lib/docker/volumes# exit
logout
```

→ Below is screenshot of volumes after creating the volumes for the assignment.

```
root@sdiware:/var/lib/docker/volumes# ll
total 40
drwx-----x  4 root root  4096 Dec  8 23:58 ./
drwx--x--- 12 root root  4096 Dec  8 21:53 ../
drwx-----x  3 root root  4096 Dec  8 23:58 assignment3_clientvol/
drwx-----x  3 root root  4096 Dec  8 23:58 assignment3_servervol/
brw-------  1 root root  8, 1 Dec  8 21:53 backingFsBlockDev
-rw-------  1 root root 32768 Dec  8 23:58 metadata.db
root@sdiware:/var/lib/docker/volumes#
```

→ After successfully running the containers, the randomFile.txt in the server.py is transferred from server to client and recievedFile.txt is received at the client side.

→ I verified this by navigating to the volumes as shown below.

```
root@sdiware:/var/lib/docker/volumes# cd assignment3_clientvol/_data/
root@sdiware:/var/lib/docker/volumes/assignment3_clientvol/_data#
root@sdiware:/var/lib/docker/volumes/assignment3_clientvol/_data#
root@sdiware:/var/lib/docker/volumes/assignment3_clientvol/_data# ll
total 12
drwxr-xr-x 2 root root 4096 Dec  8 23:58 ./
drwx-----x 3 root root 4096 Dec  8 23:58 ../
-rw-r--r-- 1 root root 1024 Dec  8 23:58 receivedFile.txt
root@sdiware:/var/lib/docker/volumes/assignment3_clientvol/_data# cd ../../assignment3_servervol/_data/
root@sdiware:/var/lib/docker/volumes/assignment3_servervol/_data# ll
total 12
drwxr-xr-x 2 root root 4096 Dec  8 23:58 ./
drwx-----x 3 root root 4096 Dec  8 23:58 ../
-rw-r--r-- 1 root root 1024 Dec  8 23:58 randomFile.txt
root@sdiware:/var/lib/docker/volumes/assignment3_servervol/_data#
```

→ Also, I verified the cksum too for both the files and it is the same as seen below. (I have done this for additional verification).

```
root@sdiware:/var/lib/docker/volumes/assignment3_servervol/_data# cksum randomFile.txt
455717187 1024 randomFile.txt
root@sdiware:/var/lib/docker/volumes/assignment3_servervol/_data#
root@sdiware:/var/lib/docker/volumes/assignment3_servervol/_data#
root@sdiware:/var/lib/docker/volumes/assignment3_servervol/_data# cd ../../assignment3_clientvol/_data/
root@sdiware:/var/lib/docker/volumes/assignment3_clientvol/_data#
root@sdiware:/var/lib/docker/volumes/assignment3_clientvol/_data# cksum receivedFile.txt
455717187 1024 receivedFile.txt
root@sdiware:/var/lib/docker/volumes/assignment3_clientvol/_data#
```

→ Note: Since the docker was installed on root user to access the volumes I had to use sudo commands.

→ Although I am sending the checksum and verifying it in client.py too as mentioned in files section.

Client Container Shell:

→ Below is the screenshot of client container shell.
→ I used docker-compose exec client sh to open the client container shell.
→ I navigated to /clientdata to verify the received file.
→ As seen below the file is received and checksum is also the correct one.

```
exousen@sdiware:~/Assignment3$ docker-compose exec client sh
#
#
# pwd
/app
# cd ../
# ls -lart
total 64
drwxr-xr-x   2 root root 4096 Sep 29 20:04 home
drwxr-xr-x   2 root root 4096 Sep 29 20:04 boot
drwxr-xr-x   1 root root 4096 Nov 20 00:00 var
drwxr-xr-x   1 root root 4096 Nov 20 00:00 usr
drwxr-xr-x   2 root root 4096 Nov 20 00:00 srv
lrwxrwxrwx   1 root root    8 Nov 20 00:00 sbin -> usr/sbin
drwxr-xr-x   2 root root 4096 Nov 20 00:00 opt
drwxr-xr-x   2 root root 4096 Nov 20 00:00 mnt
drwxr-xr-x   2 root root 4096 Nov 20 00:00 media
lrwxrwxrwx   1 root root   10 Nov 20 00:00 libx32 -> usr/libx32
lrwxrwxrwx   1 root root    9 Nov 20 00:00 lib64 -> usr/lib64
lrwxrwxrwx   1 root root    9 Nov 20 00:00 lib32 -> usr/lib32
lrwxrwxrwx   1 root root    7 Nov 20 00:00 lib -> usr/lib
lrwxrwxrwx   1 root root    7 Nov 20 00:00 bin -> usr/bin
drwxr-xr-x   1 root root 4096 Nov 21 09:53 run
drwx------   1 root root 4096 Dec  8 23:55 root
drwxrwxrwt   1 root root 4096 Dec  8 23:55 tmp
drwxr-xr-x   1 root root 4096 Dec  8 23:56 app
drwxr-xr-x   1 root root 4096 Dec  8 23:58 etc
-rwxr-xr-x   1 root root    0 Dec  8 23:58 .dockerenv
dr-xr-xr-x  13 root root    0 Dec  8 23:58 sys
dr-xr-xr-x 392 root root    0 Dec  8 23:58 proc
drwxr-xr-x   5 root root  340 Dec  8 23:58 dev
drwxr-xr-x   1 root root 4096 Dec  8 23:58 ..
drwxr-xr-x   1 root root 4096 Dec  8 23:58 .
drwxr-xr-x   2 root root 4096 Dec  8 23:58 clientdata
# cd clientdata
#
# ls -lart
total 12
drwxr-xr-x 1 root root 4096 Dec  8 23:58 ..
-rw-r--r-- 1 root root 1024 Dec  8 23:58 receivedFile.txt
drwxr-xr-x 2 root root 4096 Dec  8 23:58 .
# cat receivedFile.txt
jBbIpfooBPuT13ZQlgkpnsZuwToj0oLLy4TbQSzSDBf98H79BYcsFRAORQFNT7KSLXU6lUDJmB1LxBBhuNVg6Ds4nCTpvvqomHAKCm0pyaWXH1jZRtphw32KdSTJTpLB3CO0PY4c8dpERKbAg3OuOifoI0UmD1xf1WCR927MucHju7zRyiN7j7ONjuqgoG6BzIx3VtBynAiWMl
s7m66HACfaSDXVhUIgiklsBRS7kZt2a2lBGw3DG0vCwoVHe8zlRdJ3k6XAGHSAHzEKBEm38O0nWGhp3s1iC2UgM0C00yTgB2Q3d3oMHT4NencCyEqpgWKa0r9wxsx28aooTu6rqljkdYWgEwd6f9mP28CNue8kv4blC5V6EqVVYRhB3Um5nKob97OXwMEKI1INsFeoyLcD4cX9g
PKjtHQNGLNDAjqXn2MRmOf9Ou15Uwx9p4rYR4obJ5EDO1qWOJmIdgz0UwMsO7eCFz2TyrWykT1nMgPWVYNby5JKpTYWvGrXp6NVn2icHNNTwI2jFKIZeElrBhnyXyrzJt5mkUdiVmTwu22N2jMNxFQcVq8XRxe5VhK0MEmgjUYnfFeHJzD5MCrFINobNAjWhUNRmeFTpMs8iBeK
F2b1EDEgDIuzZizzLTjAH4goPa3DHfaF7b1Hw9Finihe7CHtU0kWmxO1fXGsxGwkfjdFvYbkAIaIbe9ADNryEywCkciAxsspeoAooUVqexImKJazPuW9C4QXD6mjYsMHPdDzRy1Yi1r2Gs2C7TPzKTDNSWr0HyDnWgL00oyX1wqX9u8kijDOUOh0Ecm8dRlo9q4hY3dxAilGJ7K
PgL3d2HUbuFkwtZhYdowpeZYLtYF7FKbbu4vqY8hcodPSIcJeuS1KPIvWirssZdaAbBEfOxuqqmB#
#
#
# cksum receivedFile.txt
455717187 1024 receivedFile.txt
#
```