Harshit Gupta
2001096665
hhgupta@iu.edu

ASSIGNMENT 3

Procedure for Setting Up a Docker-Based Client-Server File Transfer System

Container Setup:

1. Docker Volumes: Two distinct volumes, servervol and clientvol, are created to ensure persistent data storage for server and client containers, respectively.
2. Docker Network: A dedicated private network named Harshit is established to facilitate the interconnection between the server and client containers.
3. Exposed Port: The server container is configured to expose port 8080 to enable external communications.

Server and Client Configuration:

- Server Configuration:
    - The server operates on port 8080.
    - It's tasked with generating a text file of 1KB, filled with repeated phrases, upon each client connection.
    - After file generation, it computes and logs the MD5 checksum of this file.
    - The server is initiated using the command: CMD ["node", "server.js", "0.0.0.0", "8080"].
- Client Configuration:
    - The client is set to connect to the server using the aforementioned network settings.
    - It receives file data from the server, which is progressively accumulated into a memory buffer.
    - This data is then consolidated and stored into a file on the client side.
    - Additionally, it calculates the MD5 checksum of the received file to verify data integrity.
    - The client is activated with the command: CMD ["node", "client.js", "server", "8080"].

File Transfer Mechanism:

- Server Side Operations:
    - The server remains on standby for incoming client connections.
    - Upon establishing a connection, it creates a text file and calculates its MD5 checksum.
    - The file is then transmitted to the client through a socket using a data stream, ensuring continuous data flow until completion.
- Client Side Operations:
    - The client establishes a connection to the server and prepares to receive file data.

        - Incoming data is collected in a buffer and then merged into a single file once all data is received.
        - After data consolidation, the client computes the file's MD5 checksum to confirm the integrity of the transferred file.

Docker Configuration for Client-Server File Transfer System

Building Docker Images:

- **Server Image Build:**
    - Construct the Docker image for the server using the command: **docker build -t server -f .\Dockerfile** . This command specifies the Dockerfile located in the current directory and tags the resulting image as "server".
- **Client Image Build:**
    - Similarly, create the Docker image for the client using the command: **docker build -t client -f .\Dockerfile ..** This uses the same Dockerfile but tags the image as "client".

Running Docker Containers:

- **Server Container Execution:**
    - Deploy the server container with the following command:

Code:

```
docker run -v servervol:/app/serverdata -p 8080:8080 -d --name server --network harshit server
```

This command mounts the volume **servervol** to the **/app/serverdata** directory within the container, maps port 8080 of the host to port 8080 of the container for external access, runs the container in detached mode, assigns it a name "server", connects it to the network "Harshit", and uses the "server" image.

- **Client Container Execution:**
    - Start the client container using:

Code:

```
docker run -v clientvol:/app/clientdata -d --name client --network harshit client
```

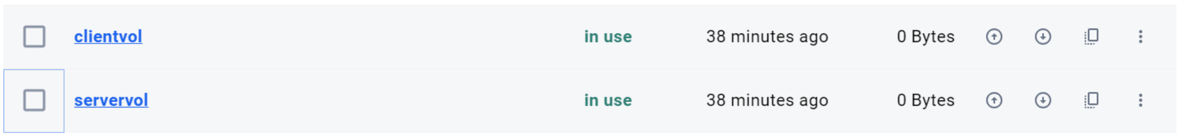
This setup mounts the **clientvol** volume to **/app/clientdata** in the container, runs the container in detached mode, names it "client", and connects it to the "Harshit" network using the "client" image.

This configuration ensures that both the server and client containers are properly set up and interconnected through the designated Docker network for efficient file transfer and data persistence.

Detailed Steps to complete the assignment:

Step 1: Create 2 volumes: servervol and clientvol

```
C:\Users\HEMANT A. GUPTA\Desktop\ECC_S24\Assignment3\client>docker volume create servervol
servervol

C:\Users\HEMANT A. GUPTA\Desktop\ECC_S24\Assignment3\client>docker volume create clientvol
clientvol
```



☐	clientvol	in use	38 minutes ago	0 Bytes
☐	servervol	in use	38 minutes ago	0 Bytes

Step 2: Creating a private network in the docker so both containers can be connected there.

```
C:\Users\HEMANT A. GUPTA\Desktop\ECC_S24\Assignment3\client>docker network create harshit
337833085e64b0ae96fd02e7fcf45a37d3b12a599961ae59676f799953fe8004
```

Step 3: Codes for server.js and client.js

Server.js:

```
const net = require('net');
const fs = require('fs');
const crypto = require('crypto');

// Default HOST and PORT are set to "localhost" and 8080 respectively, unless
provided as command line arguments
const SERVER_HOST = process.argv[2] || "localhost";
const SERVER_PORT = process.argv[3] || 8080;

// Creating a TCP server
const server = net.createServer((socket) => {
  console.log(`Client connected: ${socket.remoteAddress}`);

  // Generating approximately 1KB of text data
  const generatedText = "This is a text-based 1KB file for ECC assignment.
".repeat(32); // Repetition to approximate 1KB

  // File name where the text data will be written
  const outputFileName = 'text_file.txt';
  fs.writeFileSync(outputFileName, generatedText);

  // Calculating the MD5 checksum of the generated text
  const md5Hasher = crypto.createHash('md5');
  md5Hasher.update(generatedText);
  const fileChecksum = md5Hasher.digest('hex');
  console.log(`Checksum for the generated file: ${fileChecksum}`);

  // Streaming the file to the client over the socket
  const fileStream = fs.createReadStream(outputFileName);
  fileStream.pipe(socket);

  // Handling client disconnection
  socket.on('end', () => {
    console.log(`Client disconnected: ${socket.remoteAddress}`);
  });

  // Handling socket errors
  socket.on('error', (err) => {
    console.error(`Error: ${err.message}`);
  });
});

// Handling server-level errors
server.on("error", (err) => {
  console.error("Server error:", err);
});

// Starting the server to listen on the specified port and host
server.listen(SERVER_PORT, SERVER_HOST, () => {
  console.log(`Server started on ${SERVER_HOST}:${SERVER_PORT}`);
});
```

Client.js:

```
const net = require('net');
const fs = require('fs');
const crypto = require('crypto');

// Default server address and port are set to "localhost" and 8080
respectively, unless provided as command line arguments
const SERVER_ADDR = process.argv[2] || "localhost";
const PORT = process.argv[3] || 8080;

// Creating a new TCP client socket
const client = new net.Socket();

// Connecting to the server
client.connect(PORT, SERVER_ADDR, () => {
  console.log(`Connected to server at ${SERVER_ADDR}:${PORT}`);

  const receivedData = []; // Array to store received data chunks
  // Event listener for receiving data from the server
  client.on('data', (data) => {
    receivedData.push(data); // Collect received data
    console.log(`Received data from server: ${data.toString()}`); // Log
received data
  });

  // Event listener for server connection termination
  client.on('end', () => {
    const buffer = Buffer.concat(receivedData); // Concatenate all received
data chunks into a single buffer

    // Save the received data to a file
    fs.writeFileSync('received_file.txt', buffer);

    // Calculate the MD5 checksum of the received file
    const hasher = crypto.createHash('md5');
    hasher.update(buffer);
    const checksum = hasher.digest('hex');

    console.log(`Received a file with checksum: ${checksum}`);
  });

  // Event listener for client disconnection
  client.on("end", () => {
    console.log("Disconnected from server");
  });

  // Event listener for client socket errors
  client.on('error', (err) => {
    console.error(`Error: ${err.message}`);
  });
});
```

We will be sending a 1Kb file which says “`This is a text-based 1KB file for ECC assignment.`”.

Step 4: Create Dockerfiles for server and client

Server:

```
# Use the official Node.js 14.17.0 image based on Alpine Linux
FROM node:14.17.0-alpine

# Set the working directory inside the Docker container
WORKDIR /app
```

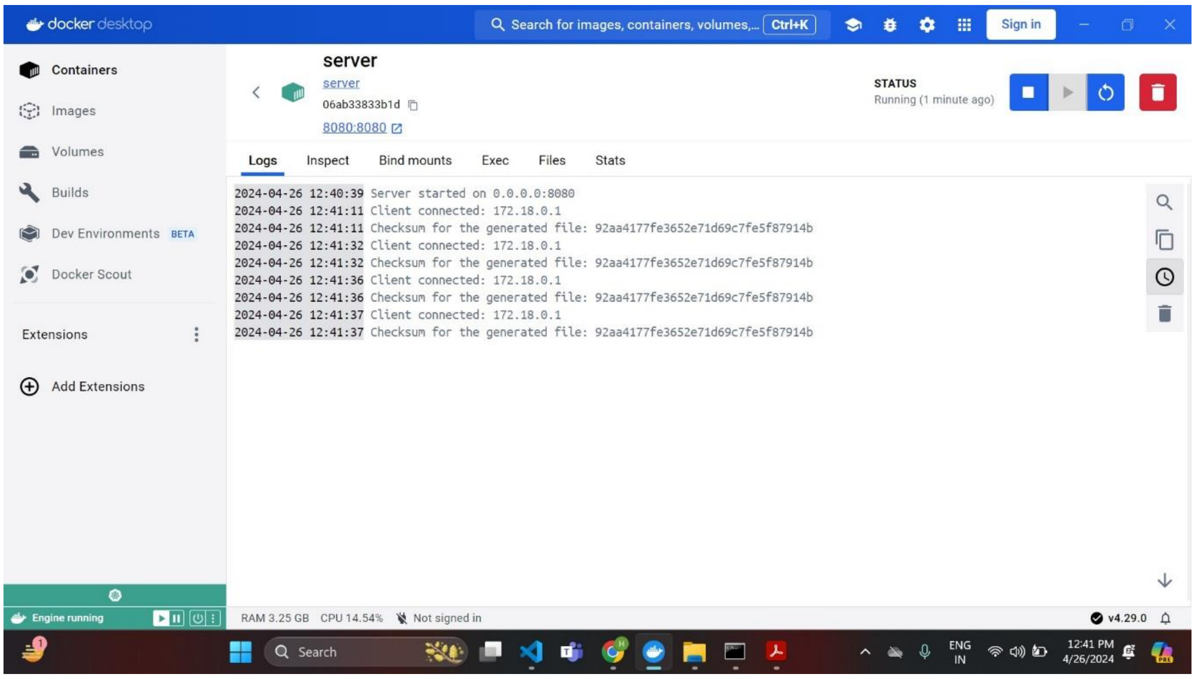
```
# Copy the server script into the working directory
COPY server.js /app/server.js

# Set executable permissions for the server.js file
# Although not strictly necessary for running Node.js scripts, this ensures
the script can be executed directly.
RUN chmod +x /app/server.js

# Create a directory to store server data
RUN mkdir /app/serverdata

# Expose port 8080 on which the server will run
EXPOSE 8080

# Command to run the server
CMD ["node", "server.js", "0.0.0.0", "8080"]
```

Client:

```
# Use the official Node.js 14.17.0 image based on Alpine Linux
FROM node:14.17.0-alpine

# Set the working directory inside the Docker container
WORKDIR /app

# Copy the client script into the working directory
COPY client.js /app/client.js

# Although setting executable permissions is not necessary for running Node.js
scripts,
# it's added here to ensure the file has execute permissions in Linux
environments
RUN chmod +x /app/client.js

# Create a directory for storing client data
RUN mkdir /app/clientdata

# Specify the default command to run the client script
# 'server' should be the hostname of the server container in the same Docker
network
CMD ["node", "client.js", "server", "8080"]
```

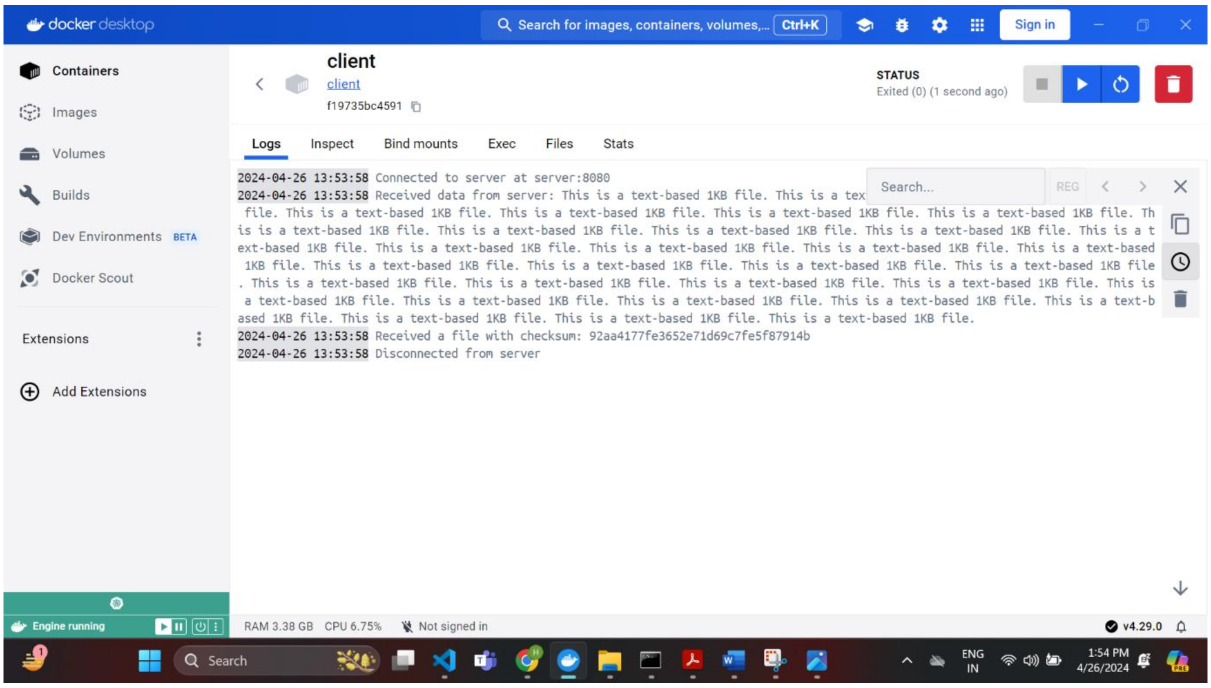
Step 5: Testing on the local file system:

Server:



```
2024-04-26 12:40:39 Server started on 0.0.0.0:8080
2024-04-26 12:41:11 Client connected: 172.18.0.1
2024-04-26 12:41:11 Checksum for the generated file: 92aa4177fe3652e71d69c7fe5f87914b
2024-04-26 12:41:32 Client connected: 172.18.0.1
2024-04-26 12:41:32 Checksum for the generated file: 92aa4177fe3652e71d69c7fe5f87914b
2024-04-26 12:41:36 Client connected: 172.18.0.1
2024-04-26 12:41:36 Checksum for the generated file: 92aa4177fe3652e71d69c7fe5f87914b
2024-04-26 12:41:37 Client connected: 172.18.0.1
2024-04-26 12:41:37 Checksum for the generated file: 92aa4177fe3652e71d69c7fe5f87914b
```

Client:



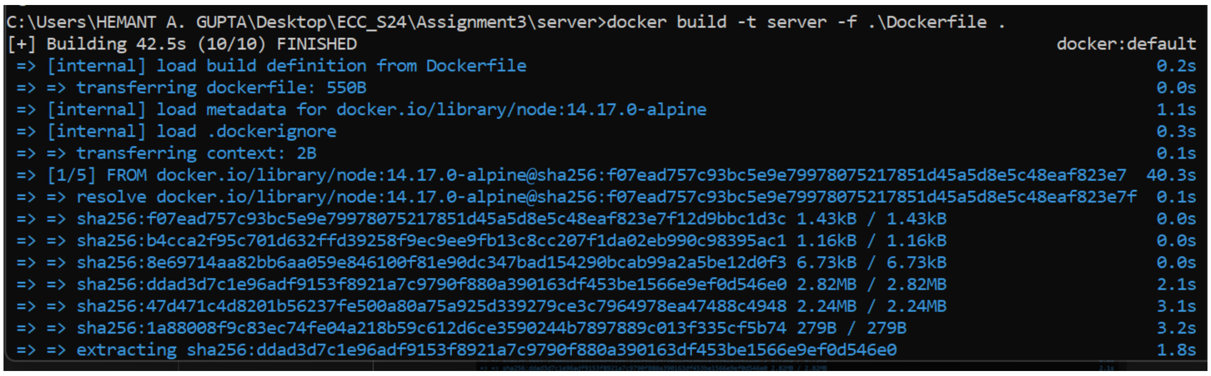
```
2024-04-26 13:53:58 Connected to server at server:8080
2024-04-26 13:53:58 Received data from server: This is a text-based 1KB file. This is a tex
 file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. Th
is is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a t
ext-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based
 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file
. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is
 a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-b
ased 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file.
2024-04-26 13:53:58 Received a file with checksum: 92aa4177fe3652e71d69c7fe5f87914b
2024-04-26 13:53:58 Disconnected from server
```

Step 6: Build the docker image

Commands:

Server:

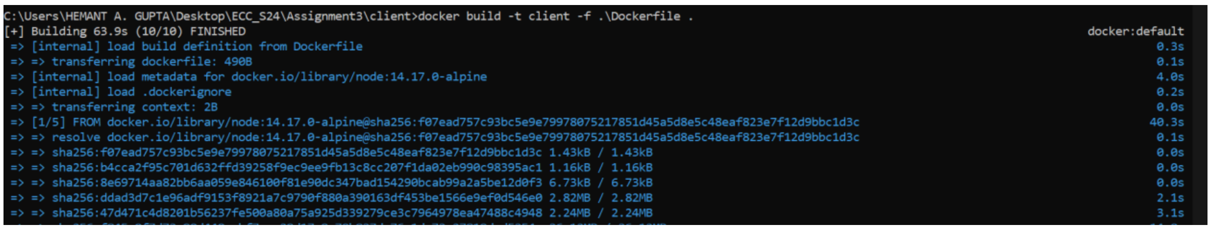
docker build -t server -f .\Dockerfile .



```
C:\Users\HEMANT A. GUPTA\Desktop\ECC_S24\Assignment3\server>docker build -t server -f .\Dockerfile .
[+] Building 42.5s (10/10) FINISHED                                                              docker:default
 => [internal] load build definition from Dockerfile                                                       0.2s
 => => transferring dockerfile: 550B                                                                       0.0s
 => [internal] load metadata for docker.io/library/node:14.17.0-alpine                                     1.1s
 => [internal] load .dockerignore                                                                          0.3s
 => => transferring context: 2B                                                                            0.1s
 => [1/5] FROM docker.io/library/node:14.17.0-alpine@sha256:f07ead757c93bc5e9e79978075217851d45a5d8e5c48eaf823e7  40.3s
 => => resolve docker.io/library/node:14.17.0-alpine@sha256:f07ead757c93bc5e9e79978075217851d45a5d8e5c48eaf823e7f  0.1s
 => => sha256:f07ead757c93bc5e9e79978075217851d45a5d8e5c48eaf823e7f12d9bbc1d3c 1.43kB / 1.43kB            0.0s
 => => sha256:b4cca2f95c701d632ffd39258f9ec9ee9fb13c8cc207f1da02eb990c98395ac1 1.16kB / 1.16kB            0.0s
 => => sha256:8e69714aa82bb6aa059e846100f81e90dc347bad154290bcab99a2a5be12d0f3 6.73kB / 6.73kB            0.0s
 => => sha256:ddad3d7c1e96adf9153f8921a7c9790f880a390163df453be1566e9ef0d546e0 2.82MB / 2.82MB            2.1s
 => => sha256:47d471c4d8201b56237fe500a80a75a925d339279ce3c7964978ea47488c4948 2.24MB / 2.24MB            3.1s
 => => sha256:1a88008f9c83ec74fe04a218b59c612d6ce3590244b7897889c013f335cf5b74 279B / 279B                3.2s
 => => extracting sha256:ddad3d7c1e96adf9153f8921a7c9790f880a390163df453be1566e9ef0d546e0                  1.8s
```

Client:

docker build -t client -f .\Dockerfile .

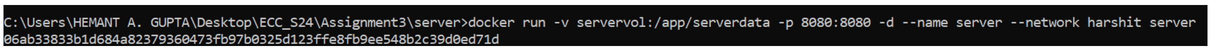


```
C:\Users\HEMANT A. GUPTA\Desktop\ECC_S24\Assignment3\client>docker build -t client -f .\Dockerfile .
[+] Building 63.9s (10/10) FINISHED                                                              docker:default
 => [internal] load build definition from Dockerfile                                                       0.3s
 => => transferring dockerfile: 490B                                                                       0.1s
 => [internal] load metadata for docker.io/library/node:14.17.0-alpine                                     4.0s
 => [internal] load .dockerignore                                                                          0.2s
 => => transferring context: 2B                                                                            0.0s
 => [1/5] FROM docker.io/library/node:14.17.0-alpine@sha256:f07ead757c93bc5e9e79978075217851d45a5d8e5c48eaf823e7f12d9bbc1d3c  40.3s
 => => resolve docker.io/library/node:14.17.0-alpine@sha256:f07ead757c93bc5e9e79978075217851d45a5d8e5c48eaf823e7f12d9bbc1d3c  0.1s
 => => sha256:f07ead757c93bc5e9e79978075217851d45a5d8e5c48eaf823e7f12d9bbc1d3c 1.43kB / 1.43kB            0.0s
 => => sha256:b4cca2f95c701d632ffd39258f9ec9ee9fb13c8cc207f1da02eb990c98395ac1 1.16kB / 1.16kB            0.0s
 => => sha256:8e69714aa82bb6aa059e846100f81e90dc347bad154290bcab99a2a5be12d0f3 6.73kB / 6.73kB            0.0s
 => => sha256:ddad3d7c1e96adf9153f8921a7c9790f880a390163df453be1566e9ef0d546e0 2.82MB / 2.82MB            2.1s
 => => sha256:47d471c4d8201b56237fe500a80a75a925d339279ce3c7964978ea47488c4948 2.24MB / 2.24MB            3.1s
```

Step 7: Create container and run on the docker

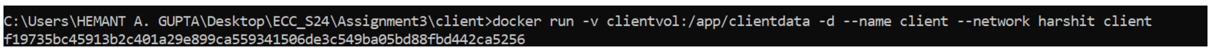
Commands:

Server: docker run -v servervol:/app/serverdata -p 8080:8080 -d --name server --network harshit server



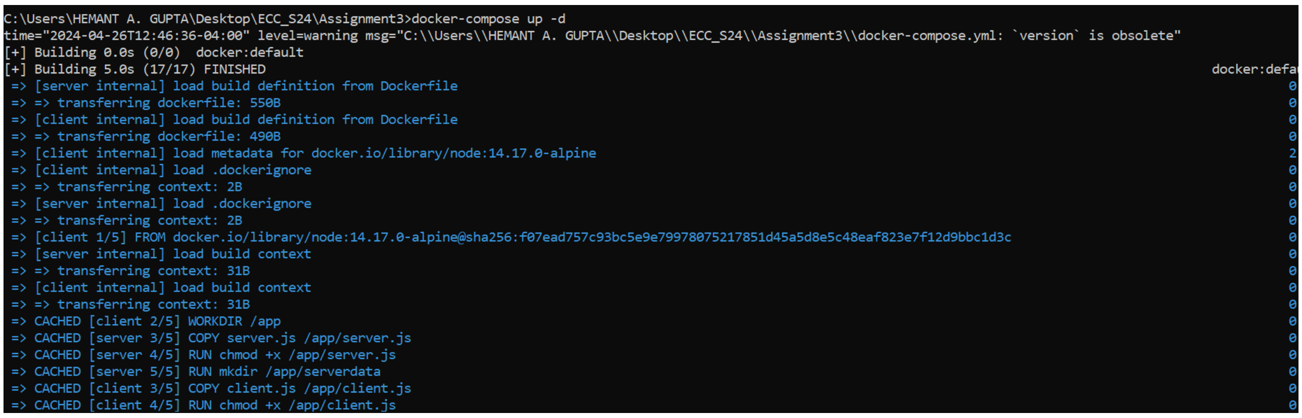
```
C:\Users\HEMANT A. GUPTA\Desktop\ECC_S24\Assignment3\server>docker run -v servervol:/app/serverdata -p 8080:8080 -d --name server --network harshit server
06ab33833b1d684a82379360473fb97b0325d123ffe8fb9ee548b2c39d0ed71d
```

Client: docker run -v clientvol:/app/clientdata -d --name client --network harshit client



```
C:\Users\HEMANT A. GUPTA\Desktop\ECC_S24\Assignment3\client>docker run -v clientvol:/app/clientdata -d --name client --network harshit client
f19735bc45913b2c401a29e899ca559341506de3c549ba05bd88fbd442ca5256
```

`Step 8: Automating using docker-compose.yml`

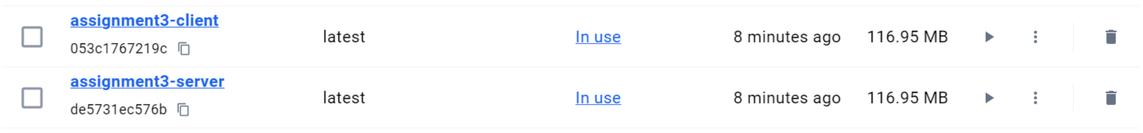


```
C:\Users\HEMANT A. GUPTA\Desktop\ECC_S24\Assignment3>docker-compose up -d
time="2024-04-26T12:46:36-04:00" level=warning msg="C:\\Users\\HEMANT A. GUPTA\\Desktop\\ECC_S24\\Assignment3\\docker-compose.yml: `version` is obsolete"
[+] Building 0.0s (0/0)  docker:default
[+] Building 5.0s (17/17) FINISHED                                                                                        docker:defa
 => [server internal] load build definition from Dockerfile                                                                        0
 => => transferring dockerfile: 550B                                                                                               0
 => [client internal] load build definition from Dockerfile                                                                        0
 => => transferring dockerfile: 490B                                                                                               0
 => [client internal] load metadata for docker.io/library/node:14.17.0-alpine                                                      2
 => [client internal] load .dockerignore                                                                                           0
 => => transferring context: 2B                                                                                                    0
 => [server internal] load .dockerignore                                                                                           0
 => => transferring context: 2B                                                                                                    0
 => [client 1/5] FROM docker.io/library/node:14.17.0-alpine@sha256:f07ead757c93bc5e9e79978075217851d45a5d8e5c48eaf823e7f12d9bbc1d3c 0
 => [server internal] load build context                                                                                           0
 => => transferring context: 31B                                                                                                   0
 => [client internal] load build context                                                                                           0
 => => transferring context: 31B                                                                                                   0
 => CACHED [client 2/5] WORKDIR /app                                                                                               0
 => CACHED [server 3/5] COPY server.js /app/server.js                                                                              0
 => CACHED [server 4/5] RUN chmod +x /app/server.js                                                                                0
 => CACHED [server 5/5] RUN mkdir /app/serverdata                                                                                  0
 => CACHED [client 3/5] COPY client.js /app/client.js                                                                              0
 => CACHED [client 4/5] RUN chmod +x /app/client.js                                                                                0
```

Docker-compose.yml:

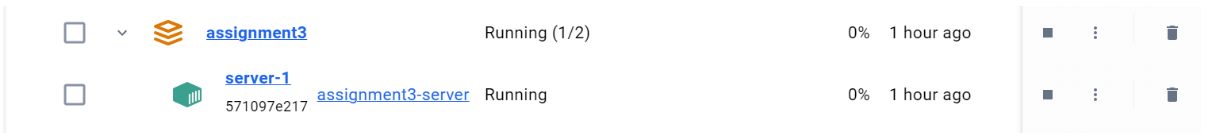
```
version: "3"
services:


Server:
build:
context: ./server
dockerfile: Dockerfile
networks:
- Harshit
volumes:
- servervol:/app/serverdata

Client:
build:
context: ./client
dockerfile: Dockerfile
volumes:
- clientvol:/app/clientdata
networks:
- harshit
volumes:
- servervol:
- clientvol:
networks:
- Harshit
```

Images:



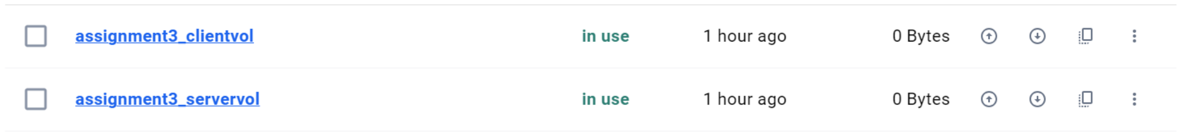
assignment3-client 053c1767219c	latest	In use	8 minutes ago	116.95 MB	
assignment3-server de5731ec576b	latest	In use	8 minutes ago	116.95 MB	

Containers:



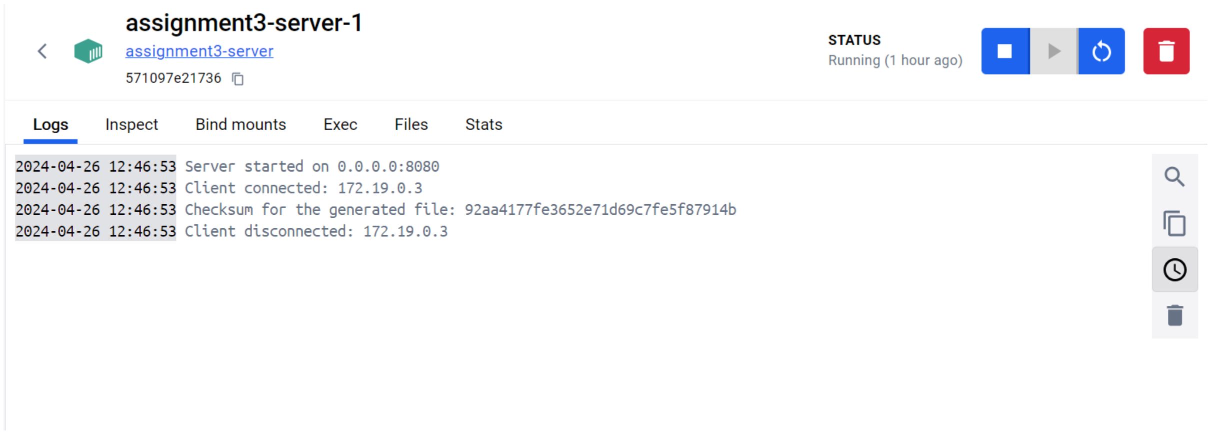
assignment3		Running (1/2)	0%	1 hour ago	
server-1 571097e217	assignment3-server	Running	0%	1 hour ago	

Volumes:



assignment3_clientvol	in use	1 hour ago	0 Bytes
assignment3_servervol	in use	1 hour ago	0 Bytes

Output of Server:

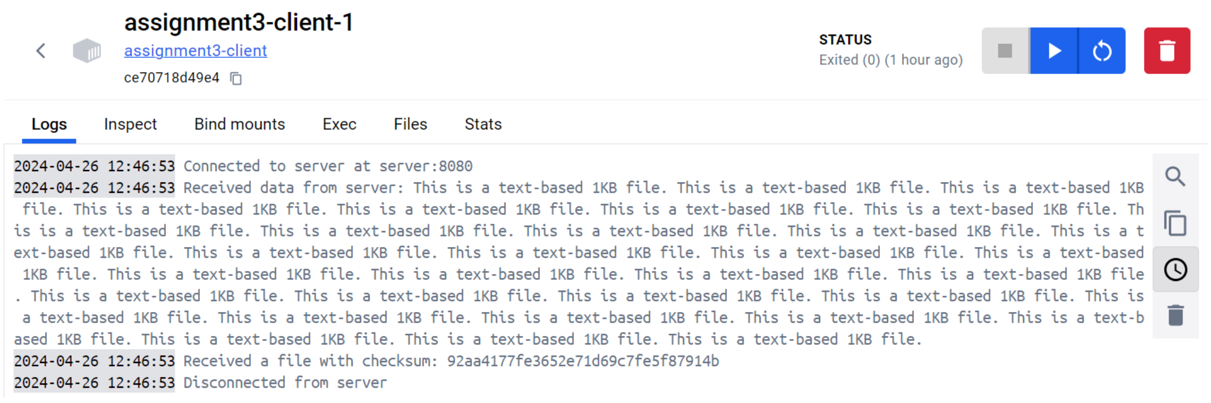


assignment3-server-1
assignment3-server
571097e21736

STATUS
Running (1 hour ago)

```
2024-04-26 12:46:53 Server started on 0.0.0.0:8080
2024-04-26 12:46:53 Client connected: 172.19.0.3
2024-04-26 12:46:53 Checksum for the generated file: 92aa4177fe3652e71d69c7fe5f87914b
2024-04-26 12:46:53 Client disconnected: 172.19.0.3
```

Output for Client:

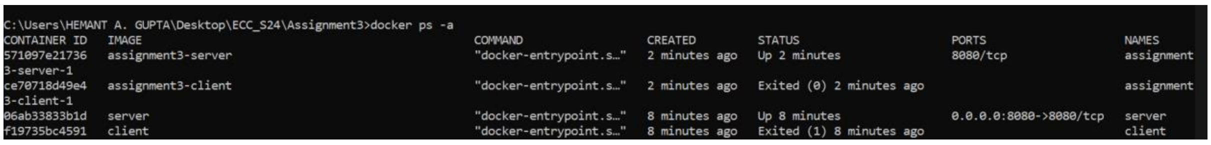


assignment3-client-1
assignment3-client
ce70718d49e4

STATUS
Exited (0) (1 hour ago)

```
2024-04-26 12:46:53 Connected to server at server:8080
2024-04-26 12:46:53 Received data from server: This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB
 file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. Th
is is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a t
ext-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based
 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file
. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is
 a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-based 1KB file. This is a text-b
ased 1KB file. This is a text-based 1KB file. This is a text-based 1KB file.
2024-04-26 12:46:53 Received a file with checksum: 92aa4177fe3652e71d69c7fe5f87914b
2024-04-26 12:46:53 Disconnected from server
```

Step 9: Checking from CLI for local and network Harshit



```
C:\Users\HEMANT A. GUPTA\Desktop\ECC_S24\Assignment3>docker ps -a
CONTAINER ID   IMAGE                COMMAND                  CREATED         STATUS                     PORTS                    NAMES
571097e21736   assignment3-server   "docker-entrypoint.s…"   2 minutes ago   Up 2 minutes               8080/tcp                 assignment
3-server-1
ce70718d49e4   assignment3-client   "docker-entrypoint.s…"   2 minutes ago   Exited (0) 2 minutes ago                            assignment
3-client-1
06ab33833b1d   server               "docker-entrypoint.s…"   8 minutes ago   Up 8 minutes               0.0.0.0:8080->8080/tcp   server
f19735bc4591   client               "docker-entrypoint.s…"   8 minutes ago   Exited (1) 8 minutes ago                            client
```