COMP 6231 Assignment 3

Task 1: File Server Commands Explanation

Qiang Xue (40300671) Yicheng Cai (26396283) Yikai Chen (40302669)
Yifan Wu (40153584) Alexander Sutherland (40321783)

November 8, 2025

1 Overview

The file server architecture utilizes Docker Swarm for container orchestration, enabling seamless communication between a central server and multiple clients across different physical machines. Each client maintains its own isolated file structure on the server, demonstrating a multi-tenant file storage system.

2 Server Configuration

2.1 Docker Image Build

Dockerfile Content:

```
FROM ozxx33/fileserver-base
WORKDIR /usr/src/app
COPY ./server.py .
CMD ["python", "./server.py"]
```

Build Command:

```
docker build -t task1_server .
```

Explanation: This command creates a Docker image named `task1_server`. The `-t` flag tags the image with the specified name, while the `.` indicates the build context is the current directory containing the Dockerfile and server.py file.

2.2 Docker Swarm Initialization

Initialize Swarm:

```
docker swarm init
```

Explanation: This command initializes a Docker Swarm on the manager node. Upon initialization, the system returns a join token that worker nodes can use to join the swarm.

Retrieve Worker Token:

```
docker swarm join-token worker
```

Explanation: Displays the join token for worker nodes. This token is required for worker nodes to authenticate and join the Docker Swarm cluster.

2.3 Create Overlay Network

Create Network:

```
docker network create -d overlay 6231-net
```

Explanation: Creates an overlay network named `6231-net`. The `-d overlay` flag specifies the network driver type.

2.4 Verify Configuration

List Cluster Nodes:

```
docker node ls
```

Explanation: Lists all nodes in the Docker Swarm, displaying their status, availability, and role (manager or worker). This verification step ensures all nodes have successfully joined the cluster.

List Networks:

```
docker network ls
```

Explanation: Lists all networks in the Docker Swarm, confirming the creation of the overlay network `6231-net`.

Inspect Specific Network:

```
docker network inspect 6231-net
```

Explanation: Provides detailed information about the overlay network, including connected containers and their IP addresses.

2.5 Run Server Container

Launch Server Container:

```
docker run -it --rm --name server --hostname server
    --network 6231-net task1_server bash
```

Explanation - Command Parameters:

- `-it`: Runs container in interactive mode with terminal access
- `--rm`: Automatically removes the container upon termination
- `--name server`: Assigns the container name "server"
- `--hostname server`: Sets hostname for name resolution within the network
- `--network 6231-net`: Connects container to the overlay network

- `bash`: Opens bash shell for manual command execution

Verify Container IP:

```
hostname -i
```

Output: `10.0.1.28`
Explanation: Displays the IP address assigned to the container within the overlay network, enabling communication with other containers.

Execute Server Application:

```
python server.py
```

Explanation: Launches the file server application, which listens on `0.0.0.0:65432` for incoming client connections and organizes received files in client-specific directories.

3 Client/Worker Configuration

Note: The following section details the complete workflow for each client in the cluster, from building the Docker image to executing the client application. This section is repeated for each client to demonstrate the independent configuration process on each worker node.

3.1 Client 1 (First Worker Node)

3.1.1 Docker Image Build

Dockerfile Content:

```
FROM ozxx33/fileserver-base
WORKDIR /usr/src/app
COPY . .
ENV SERVER_IP=server
ENV SERVER_PORT=65432
CMD ["python", "./client.py"]
```

Build Command:

```
docker build -t task1_client .
```

Explanation: Creates a Docker image named `task1_client` on the first worker node. The Dockerfile copies all files from the current directory (including client.py and test files) and sets environment variables for server connection configuration.

3.1.2 Join Docker Swarm

Join Worker to Swarm:

```
docker swarm join --token <token> 192.168.12.26:2377
```

Explanation: Joins the first worker node to the Docker Swarm cluster. The token (obtained from the manager node) authenticates the node, while `192.168.12.26:2377` specifies the manager node's address and Docker Swarm management port.

3.1.3 Launch Client Container

Run Client Container:

```
docker run -it --rm --name client1 --hostname client1
    --network 6231-net task1_client bash
```

Explanation - Command Parameters:

- `-it`: Runs container in interactive mode with terminal access
- `--rm`: Automatically removes the container upon termination
- `--name client1`: Assigns the container name "client1"
- `--hostname client1`: Sets hostname for network identification
- `--network 6231-net`: Connects container to the overlay network
- `bash`: Opens bash shell for manual command execution

Container IP: `10.0.1.30`

3.1.4 Execute Client Application

Run Client Program:

```
python client.py
```

Explanation: Executes the client application, which connects to the server using hostname resolution via Docker's DNS, creates a `client1` directory on the server, and transfers files to that directory.

3.2 Client 2 (Second Worker Node)

3.2.1 Docker Image Build

Dockerfile Content:

```
FROM ozxx33/fileserver-base
WORKDIR /usr/src/app
COPY . .
ENV SERVER_IP=server
ENV SERVER_PORT=65432
CMD ["python", "./client.py"]
```

Build Command:

```
docker build -t task1_client .
```

Explanation: Creates a Docker image named `task1_client` on the second worker node. Each worker node builds its own image independently to avoid dependency on image distribution mechanisms.

3.2.2 Join Docker Swarm

Join Worker to Swarm:

```
docker swarm join --token <token> 192.168.12.26:2377
```

Explanation: Joins the second worker node to the Docker Swarm cluster using the same token and manager node address as Client 1.

3.2.3 Launch Client Container

Run Client Container:

```
docker run -it --rm --name client2 --hostname client2
    --network 6231-net task1_client bash
```

Explanation - Command Parameters:

- `-it`: Runs container in interactive mode with terminal access
- `--rm`: Automatically removes the container upon termination
- `--name client2`: Assigns the container name "client2"
- `--hostname client2`: Sets unique hostname for network identification
- `--network 6231-net`: Connects container to the overlay network
- `bash`: Opens bash shell for manual command execution

Container IP: `10.0.1.32`

3.2.4 Execute Client Application

Run Client Program:

```
python client.py
```

Explanation: Executes the client application, which connects to the server, creates a `client2` directory on the server, and transfers files to that directory.

3.3 Client 3 (Third Worker Node)

3.3.1 Docker Image Build

Dockerfile Content:

```
FROM ozxx33/fileserver-base
WORKDIR /usr/src/app
COPY . .
ENV SERVER_IP=server
ENV SERVER_PORT=65432
CMD ["python", "./client.py"]
```

Build Command:

```
docker build -t task1_client .
```

Explanation: Creates a Docker image named `task1_client` on the third worker node, completing the image build process for all worker nodes in the cluster.

3.3.2 Join Docker Swarm

Join Worker to Swarm:

```
docker swarm join --token <token> 192.168.12.26:2377
```

Explanation: Joins the third worker node to the Docker Swarm cluster using the same token and manager node address.

3.3.3 Launch Client Container

Run Client Container:

```
docker run -it --rm --name client3 --hostname client3
    --network 6231-net task1_client bash
```

Explanation - Command Parameters:

- `-it`: Runs container in interactive mode with terminal access
- `--rm`: Automatically removes the container upon termination
- `--name client3`: Assigns the container name "client3"
- `--hostname client3`: Sets unique hostname for network identification
- `--network 6231-net`: Connects container to the overlay network
- `bash`: Opens bash shell for manual command execution

Container IP: `10.0.1.34`

3.3.4 Execute Client Application

Run Client Program:

```
python client.py
```

Explanation: Executes the client application, which connects to the server, creates a `client3` directory on the server, transfers files, and executes commands such as `wordcount about.txt`.

4 Network and Storage Configuration

4.1 Network Configuration

Container	Hostname	IP Address	Port	Host IP Address
Server	server	10.0.1.28	65432	192.168.12.26
Client 1	client1	10.0.1.30	Dynamic	192.168.12.16
Client 2	client2	10.0.1.32	Dynamic	192.168.12.5
Client 3	client3	10.0.1.34	Dynamic	192.168.12.3

Table 1: Container network configuration within the overlay network

4.2 Storage Configuration

- **Server Storage:** Files are stored within the server container at `/usr/src/app/`
- **Client-Specific Directories:** Each client can use `mkdir xxx` to create its own directory on the server for file storage.)
- **No Shared Volumes:** Each container maintains isolated storage; file transfer occurs exclusively via network communication

5 Implementation Results

5.1 Server-Side View

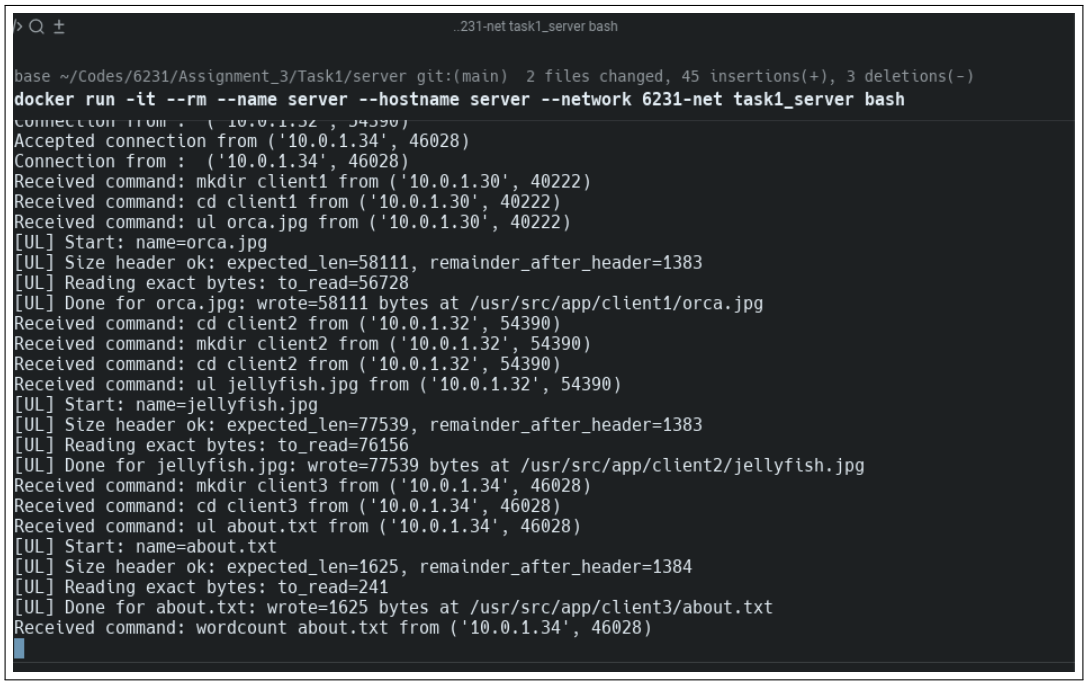

Figure 1: Server container showing received files organized by client directories

5.2 Client-Side Views

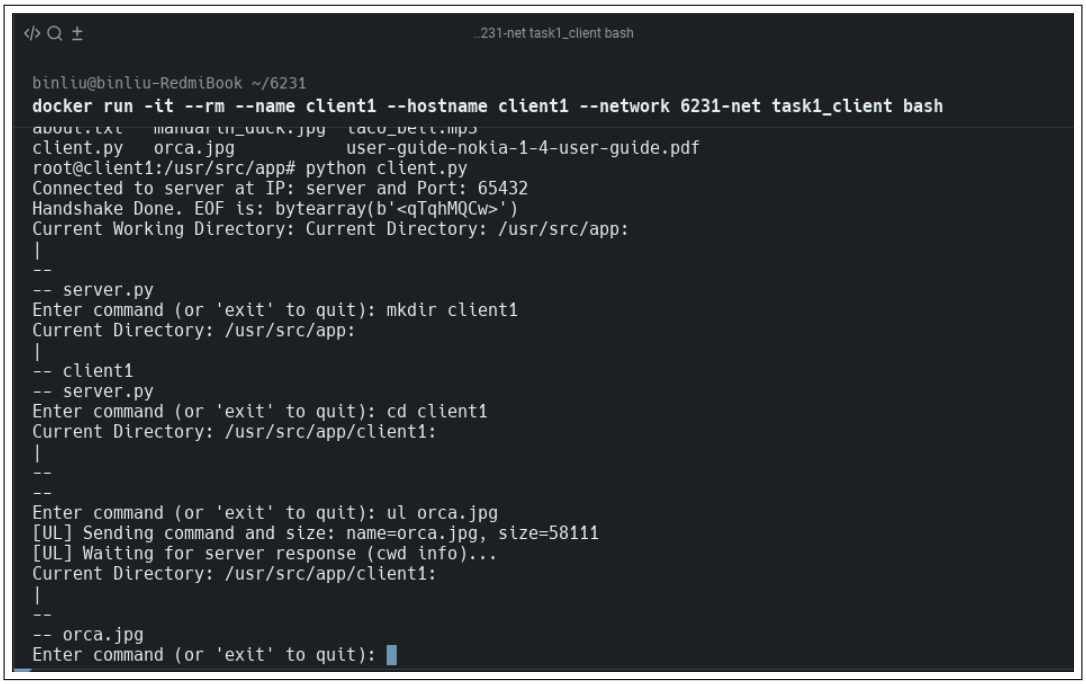

Figure 2: Client 1 successfully creating `client1` directory and sending files to server

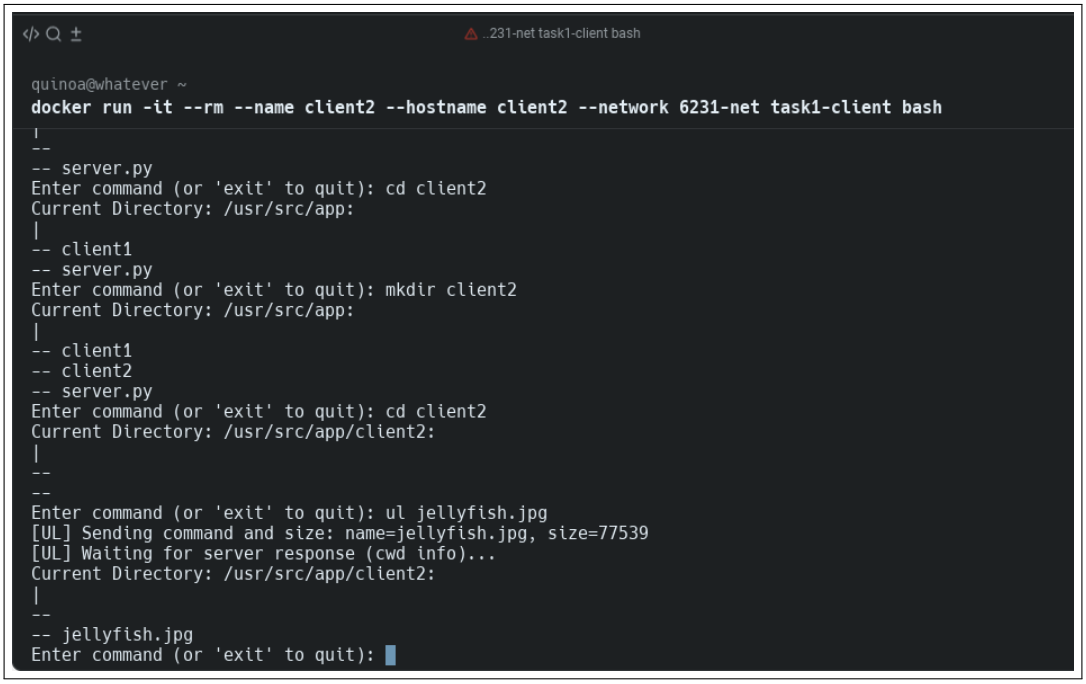

Figure 3: Client 2 successfully creating `client2` directory and sending files to server

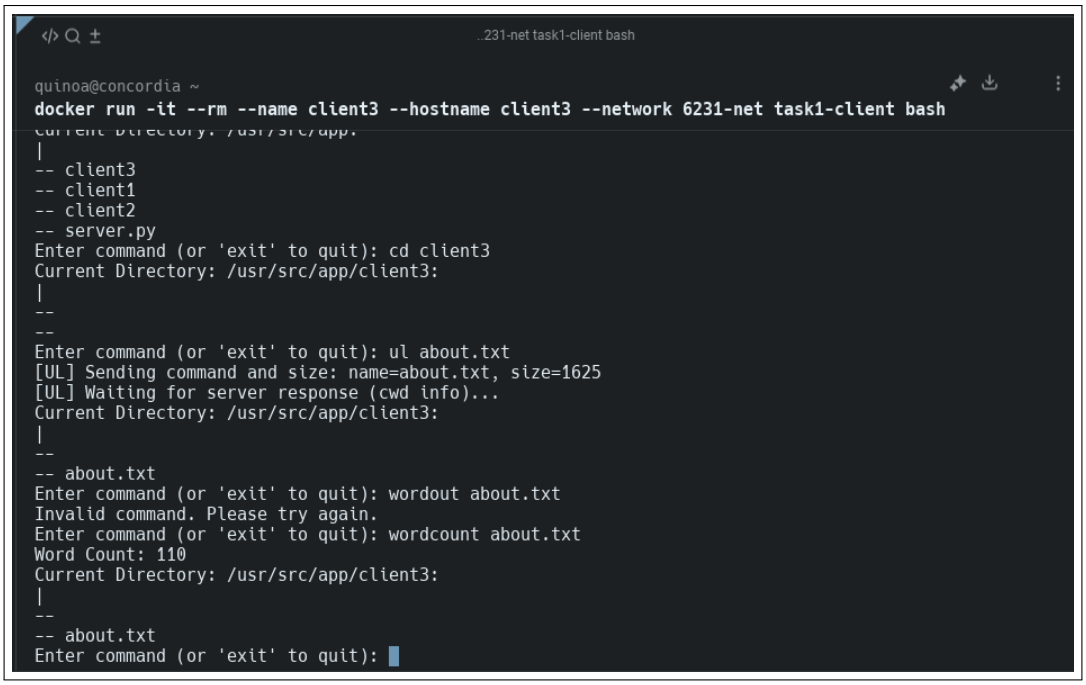

Figure 4: Client 3 successfully creating `client3` directory and sending `about.txt` files to server and execute `wordcount about.txt` command