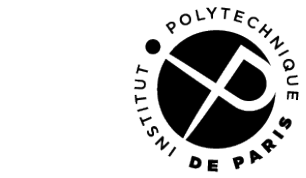

Final Project

Arthur Martins Braga

CSC7342 - Virtualization: Concept and implementation

December 10, 2024

1 Introduction

This report documents the steps and results of deploying two distinct projects as part of the final assignment. The first task involved deploying a full-stack application using Kubernetes in a Minikube cluster, while the second task focused on setting up a mobile network emulator using Docker Compose.

The report outlines the processes followed, challenges encountered, and solutions implemented during the deployment of each project. Screenshots and references are provided to illustrate key steps and demonstrate the functionality of the deployed systems. The project's code and configurations are also made available in a GitHub repository for further review and replication.

2 Repository Overview

The project's source code and configurations are available on GitHub: `https://github.com/ArthurMbraga/k8s-docker-examples`.

The repository is structured as follows:

- `fullstack-k8s/`: Files and configurations for Task 1 (deploying the full-stack application with Kubernetes).
- `docker-compose-network/`: Resources for Task 2 (deploying the mobile network emulator using Docker Compose).

3 Task 1: Full-Stack Deployment with Kubernetes

This section describes the steps and results of deploying a sample full-stack application using Kubernetes in a Minikube cluster.

3.1 Overview

The deployment includes a backend, a frontend, and a database. The configuration files were created to configure the deployments and services for each component. The file structure used for this project is shown in Figure 1.

3.2 Minikube Dashboard

The Minikube dashboard was used to monitor deployments, replicas, and pods. All components of the stack were successfully deployed and running. Figure 2 shows a screenshot of the Minikube dashboard.

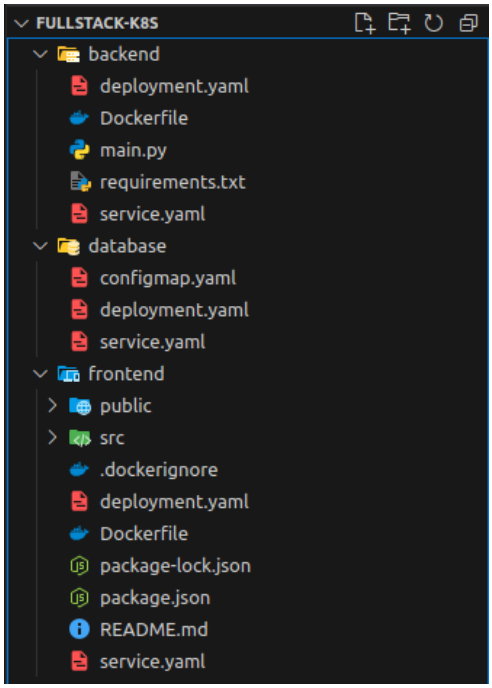

Figure 1: File structure of the project

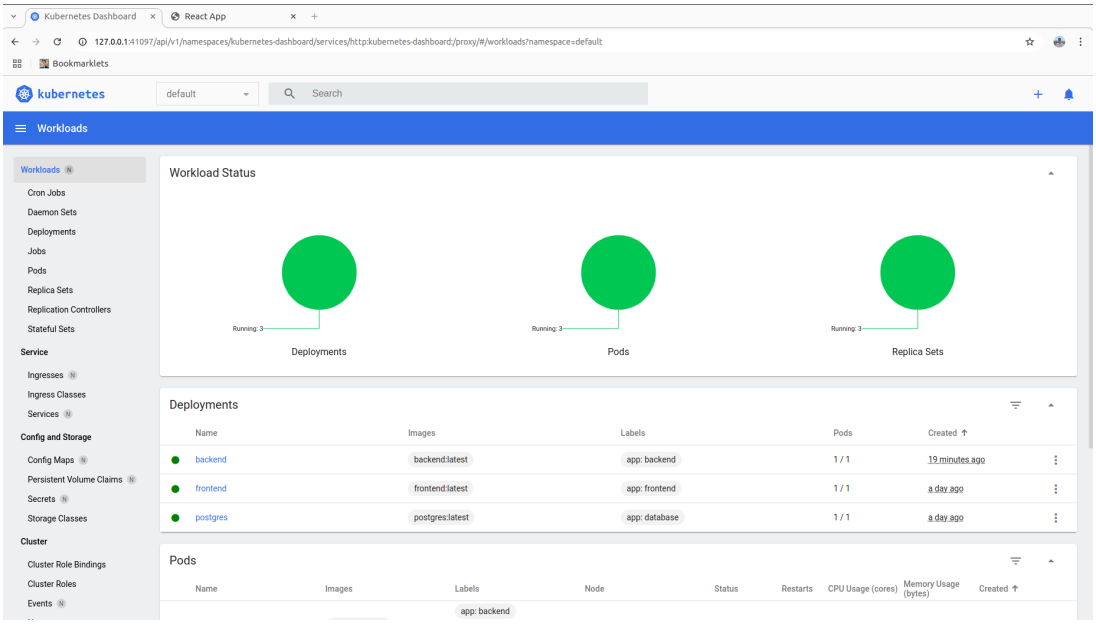

Figure 2: Minikube dashboard showing running pods and deployments

3.3 Frontend Application

The frontend of the application was successfully deployed. Initially, the application did not show data, as illustrated in Figure 3.

After adding some sample data, the front-end was updated as shown in Figure 4.

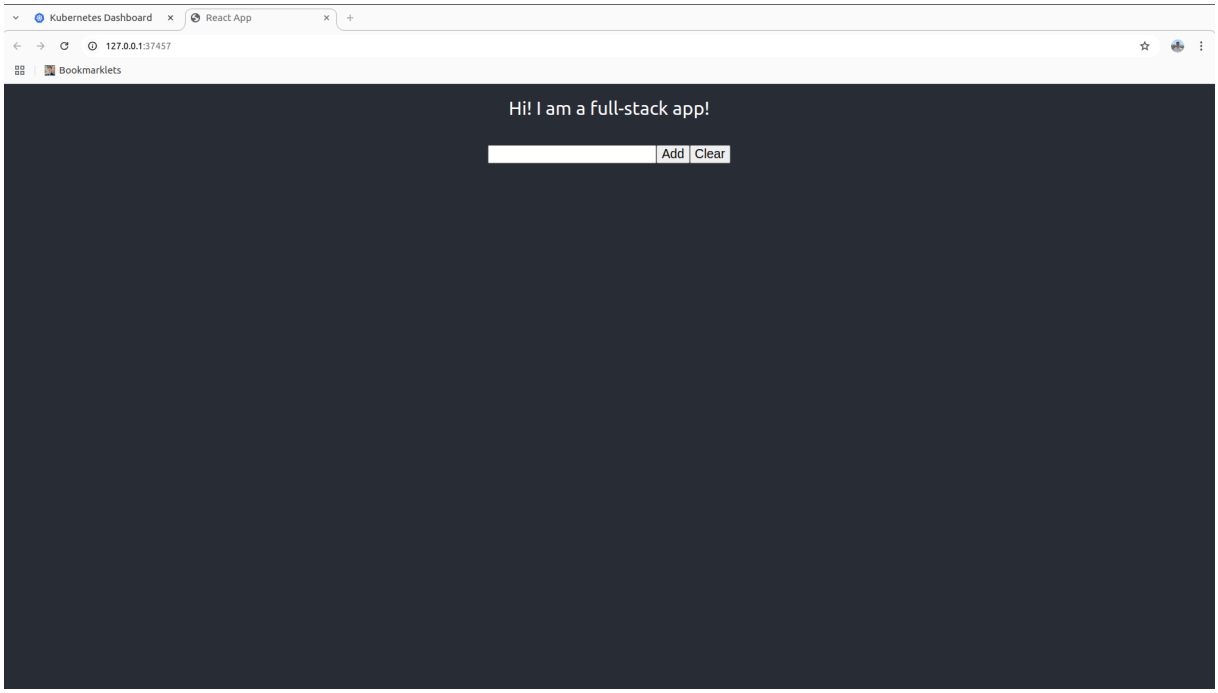

Figure 3: Frontend of the application with no data

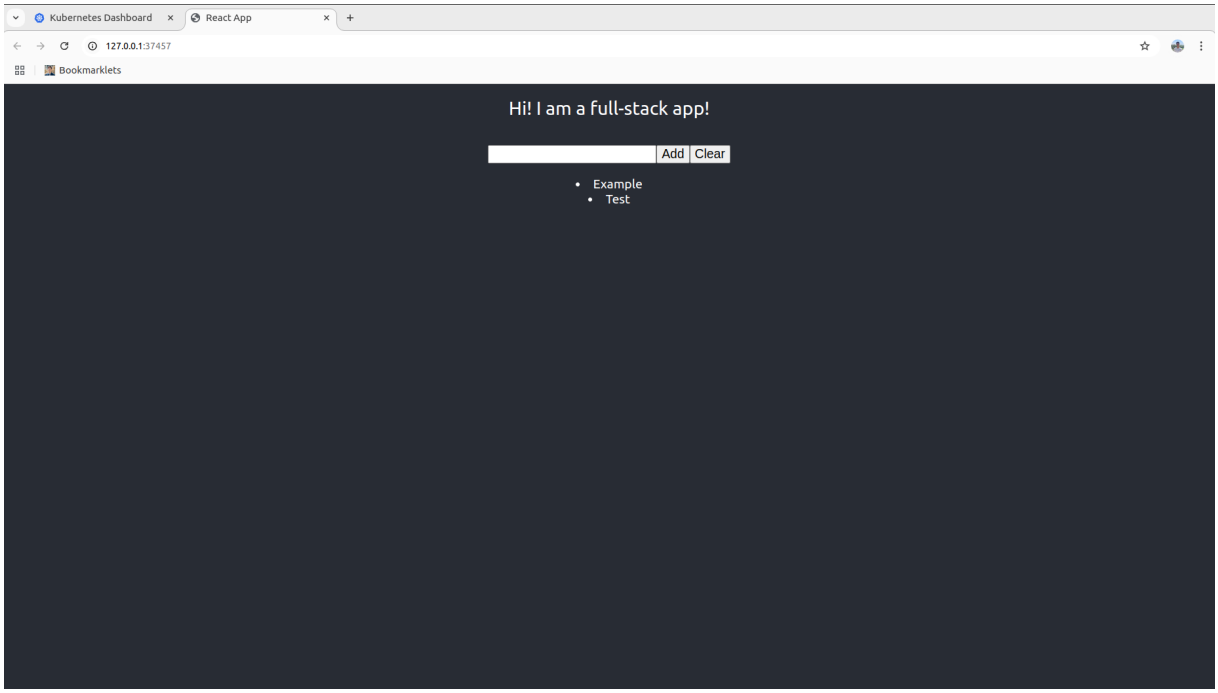

Figure 4: Frontend of the application with sample data added

3.4 Conclusion

Task 1 was completed successfully. The full stack application was deployed in the Minikube cluster and the front-end application operated as expected. Following the tutorial was straightforward, meaning that only minimal modifications were needed on my part to make it work. I encountered some troubleshooting related to Python and

Node versions, but these issues were easy to resolve.

4 Task 2: Deployment of Mobile Network Emulator Using Docker Compose

This section outlines the steps taken to deploy a mobile network emulator using Docker Compose. Screenshots of the console output are provided for each step.

4.1 Step 1: Download Updated Images

The first step involved downloading the updated images from the repository. These images are essential for the deployment. The console output of this step is shown in Figure 5.

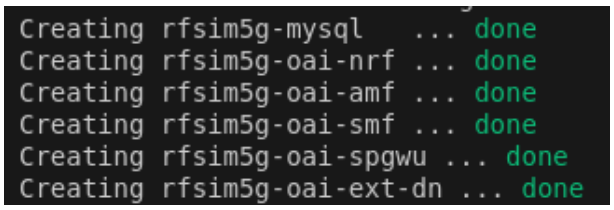

Figure 5: Downloading updated images

4.2 Step 2: Deploy Network

The ‘docker-compose.yml‘ file was modified to address issues with variable configurations:

- ‘services.mysql.image‘ was updated to ‘mysql:5.7‘.
- ‘services.mysql.healthcheck.retries‘ was set to ‘5‘.

After these changes, the network was deployed successfully (Figure 6).

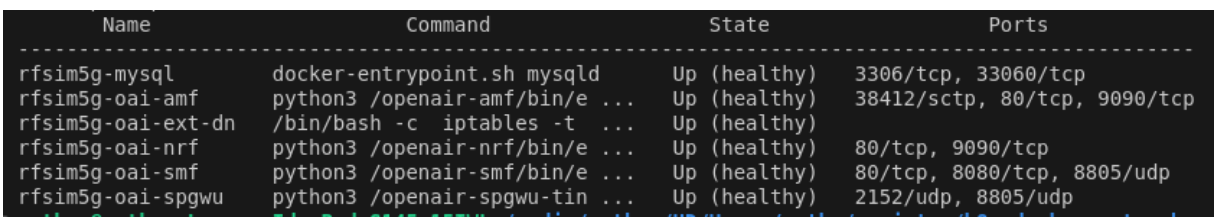

Figure 6: Deploying the network

4.3 Step 3: Check Status

During this step, an issue with the ‘mysql-healthcheck.sh‘ script was resolved. The problem was related to LF (Line Feed) and CRLF (Carriage Return Line Feed) encoding, which was corrected by converting to LF. The output is shown in Figure 7.

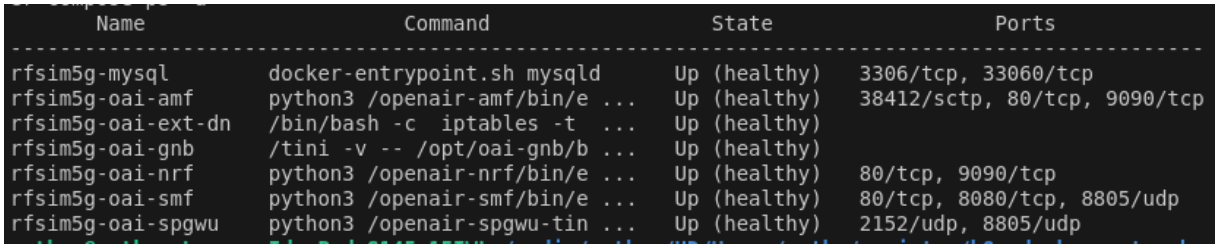

Figure 7: Checking the status

4.4 Step 4: Check Network Interface

The ‘ifconfig‘ command was executed to verify the network interface configuration. See Figure 8.

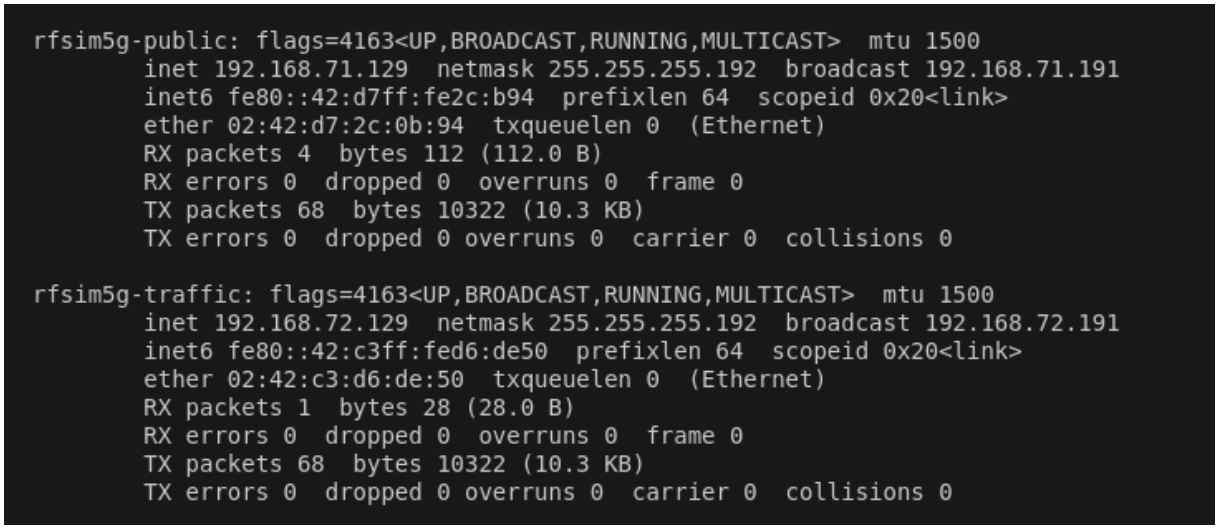

Figure 8: Network interface configuration

4.5 Steps 5 and 6: Deploy OAI Components

The following OAI components were deployed in RF simulator mode and standalone mode:

- OAI gNB (Figure 9)
- OAI NR-UE (Figure 10).

In ‘docker-compose.yml‘, the ‘–sa‘ flag was removed from:

- services.oai-nr-ue.environment.USE_ADDITIONAL_OPTIONS
- services.oai-nr-ue2.environment.USE_ADDITIONAL_OPTIONS

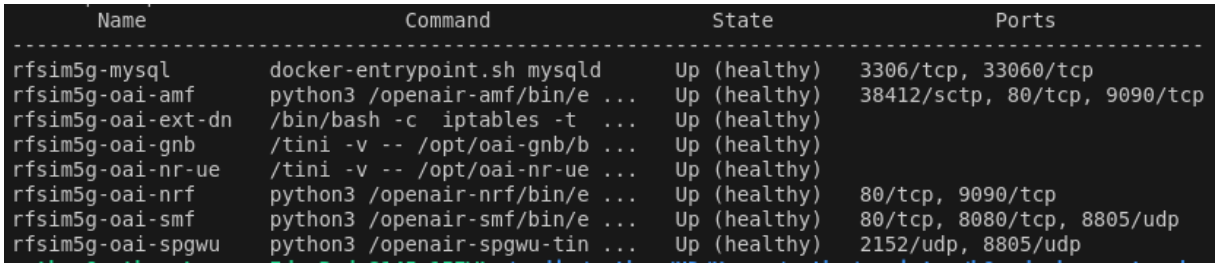

Figure 9: Deploying OAI gNB

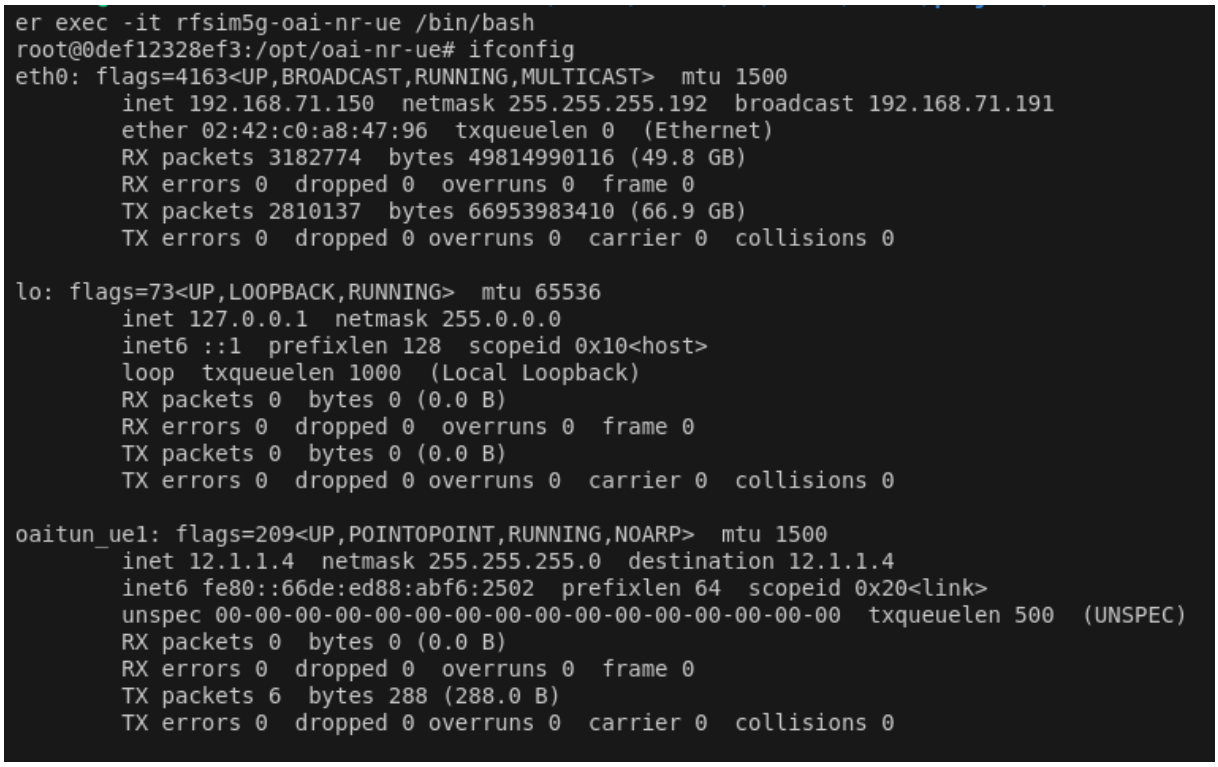

Figure 10: Deploying OAI NR-UE

4.6 Step 7: Verify OAI UE Connection

The OAI UE connection was verified to ensure successful deployment (Figure 11).

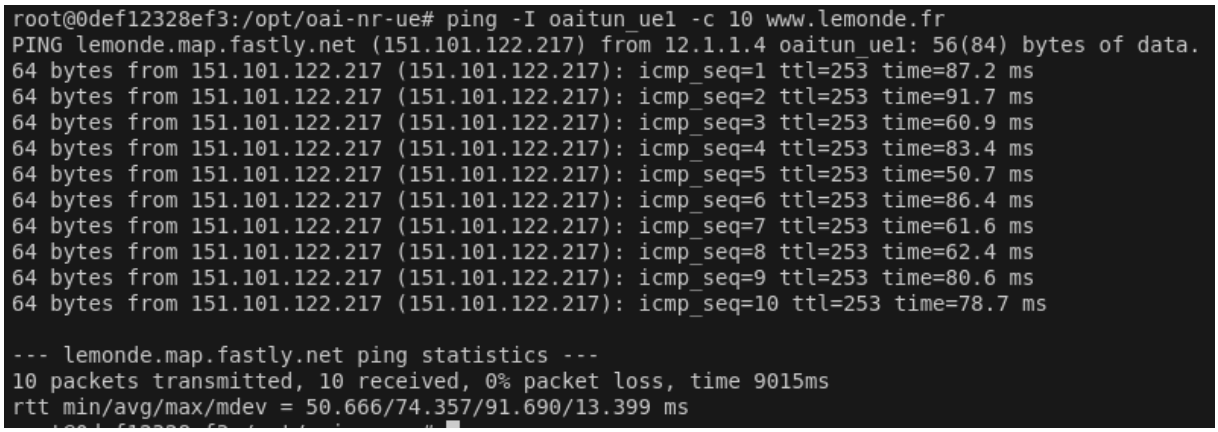

Figure 11: Verifying OAI UE connection

4.7 Step 8: Check Internet Connectivity

Internet connectivity was checked as shown in Figure 12.

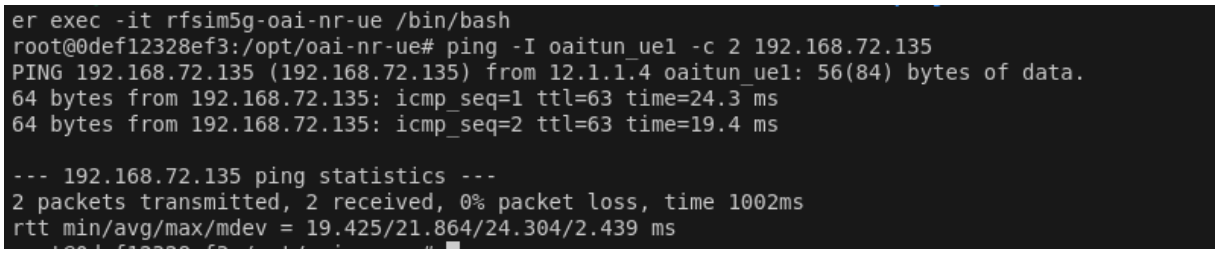

Figure 12: Checking Internet connectivity

4.8 Steps 9 and 10: Test with iperf

The ‘iperf‘ tool was used to test connectivity:

- Step 9: Start the ‘iperf‘ server inside the NR-UE container. Initially, the command ‘iperf -B 12.1.1.2 -u -i 1 -s‘ failed. Using ‘ip addr show‘, it was determined that the correct IP address was ‘12.1.1.4‘ (Figure 13).
- Step 10: Start the ‘iperf‘ client inside the ext-dn container (Figure 14).

Figure 13: Starting iperf server

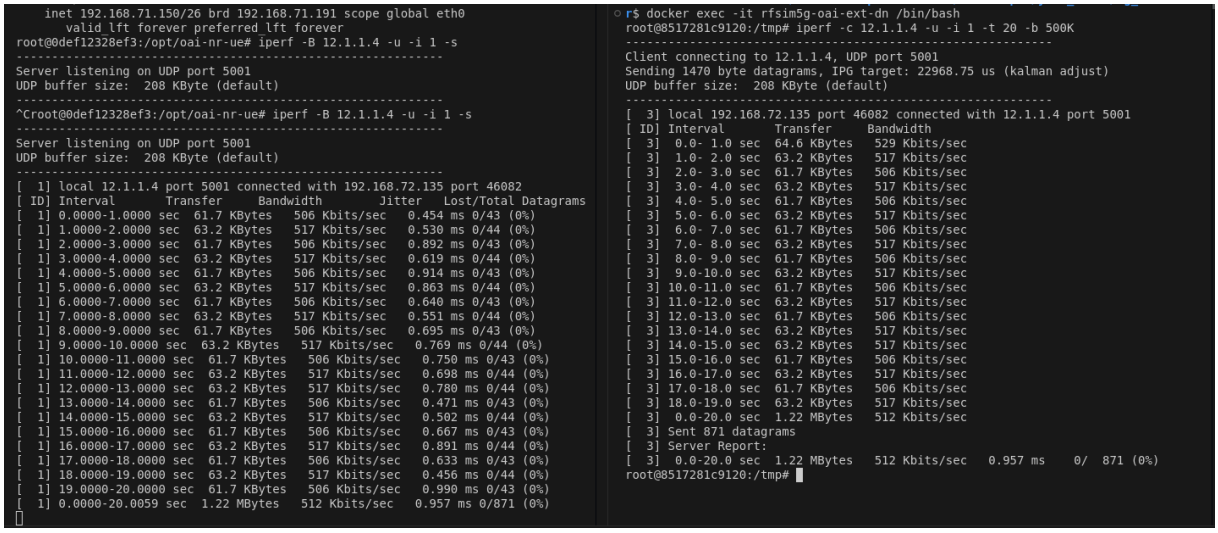

Figure 14: Starting iperf client

4.9 Step 11: Undeploy

Finally, the emulator was undeployed, completing Task 2 (Figure 15).

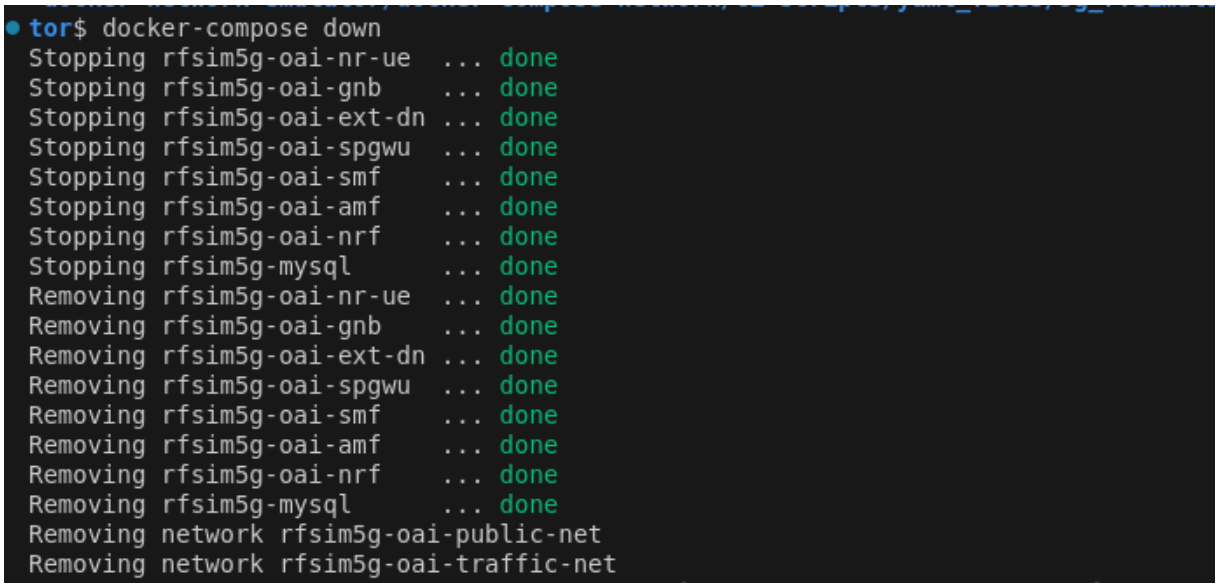

Figure 15: Undeploying the emulator

4.10 Conclusion

Task 2 was successfully completed with all steps implemented and verified. Challenges such as script encoding and configuration issues were resolved to ensure proper deployment and operation of the mobile network emulator.