Deploying Open5g applications with continuous integration

1st Ishita Patel
dept. Computer Science
Kent State University
Kent, OH

2nd Khushbu Brahmbhatt
dept. computer science
Kent State University
Kent, OH

3rd Akhil Reddy
dept. computer science
Kent State University
Kent, OH

4th Divya Sri Karingula
dept. computer science
Kent State University
Kent, OH

I. Introduction

The main aim of this project is to provide Open5g Application Deployment pipeline end to end automation using GitHub-Jenkins-Docker-Kubernetes. Monitoring of 5G systems using Kubernetes is important because there is a big push for 5G based systems in modern cloud systems. We are using UERANSIM which supports to run with Open5GS Core networks. We can connect UERANSIM to one of these 5G Core network and test the functionality. Also, 5G systems monitoring will be done by Prometheus monitoring. Prometheus monitoring is a cloud native computing-based systems and service monitoring system.

II. CI/CD Pipeline

The different components of our design are illustrated in Figure1. For orchestration, we chose Kubernetes, an open-source system for deploying and managing containerized applications. Here, for this project, we are following CI/CD pipeline. Firstly, we are cloning the repo from github. Then we will scan the source code and store on local machine using docker script. We are using docker because it stores the code inform of container and then it builds docker image. After these, it scans container image and will implement core network using Kubernetes.

III. Overview

In this project we clone the code and write Docker files, build Docker images and we run in docker containers. We used Jenkins pipeline to do continues integration/ continues development (CI/CD) of open 5g applications. we Automated the IP configuration for Open5gs components. Then using the Cadvisor for container monitoring Prometheus collects data and metrics from different services and stores them according to a unique identifier. Lastly, we did interaction between components of open5gs and ueransim.

IV. Tools

- GitHub Code Repo
- AWS Management Console
- AWS Service like Ec2
- Docker Engine
- Prometheus
- Jenkins
- Cadvisor
- Grafana

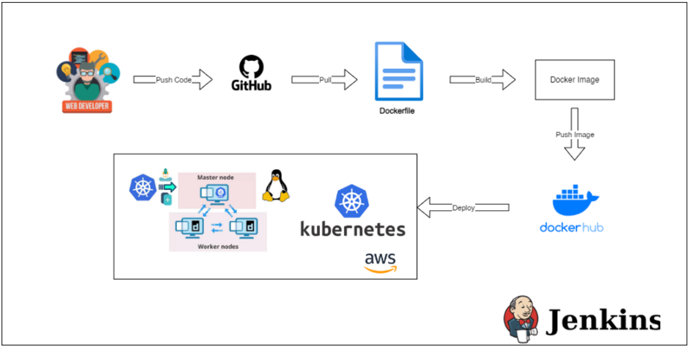


Fig. 1. CI/CD Pipeline

V. Open5GS and UERANSIM

This Open5g can be used to configure your own NR/LTE network. If gNB/eNB and USIM are available, you can build a private network using Open5GS. Open5GS implemented 5GC and EPC using C-language and WebUI is provided for testing purposes and is implemented in Node.JS and React. Ueransim is an open source 5G UE and 5G RAN (gNodeB) implementation. System is considered as 5Gmobile phone and a base station. It contains 3 main interfaces: User interface, Control interface and radio interface.

This project is use for testing 5G Core Network and studying 5G system. Ueransim help to run Open5Gs and Free5G core network and test the system functionality. Two key components: UE and gNodeB. UE can test several devices. A gNodeB is the functional equivalent of a base station in a network. A gNB in network provides connectivity between UE and EPC. A gNB physical entity such as a tower and virtual entity such as a software defined radio. A gNodeB is responsible for radio communication with UEs in its coverage area known as a cell. gNodeB functions: Radio resource management, Mobility management, Connection management, Security, Quality of service (QoS), Charging.

Here, an Open5g developer has developed code and pushed it to the Open5g GitHub repository (https://github.com/open5gs/open5gs). From there, we will clone the code and write Docker files, build Docker

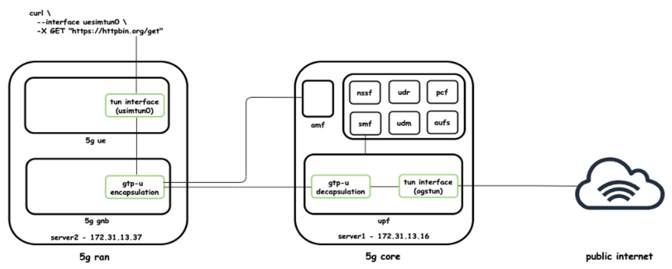
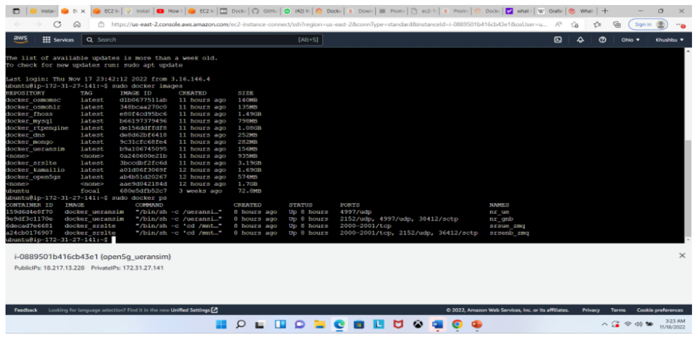


Fig. 2. Above figure depicts system consist of 5g ran and 5g core



```
The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Thu Nov 17 23:42:12 2022 from 3.16.146.4
ubuntu@ip-172-31-27-141:~$ sudo docker images
REPOSITORY         TAG      IMAGE ID       CREATED         SIZE
docker_osmomsc     latest   d1b0677511ab   11 hours ago    140MB
docker_osmohlr     latest   348bcaa270c0   11 hours ago    135MB
docker_fhoss       latest   e80f4cd95bc6   11 hours ago    1.49GB
docker_mysql       latest   b66197379496   11 hours ago    798MB
docker_rtpengine   latest   de156ddffdf8   11 hours ago    1.08GB
docker_dns         latest   de8d62bf6418   11 hours ago    252MB
docker_mongo       latest   9c31cfc68fe4   11 hours ago    282MB
docker_ueransim    latest   b9a106745095   11 hours ago    156MB
<none>             <none>   0a240600e21b   11 hours ago    935MB
docker_srslte      latest   3bccdbf2fc6d   11 hours ago    3.19GB
docker_kamailio    latest   a01d06f3069f   12 hours ago    1.69GB
docker_open5gs     latest   ab4b51d20267   12 hours ago    574MB
<none>             <none>   aae9d042184d   12 hours ago    1.7GB
ubuntu             focal    680e5dfb52c7   3 weeks ago     72.8MB
ubuntu@ip-172-31-27-141:~$ sudo docker ps
CONTAINER ID   IMAGE             COMMAND                  CREATED         STATUS         PORTS                                     NAMES
159d6d4e8f70   docker_ueransim   "/bin/sh -c /ueransi…"   8 hours ago     Up 8 hours     4997/udp                                  nr_ue
9e9df3c1170e   docker_ueransim   "/bin/sh -c /ueransi…"   8 hours ago     Up 8 hours     2152/udp, 4997/udp, 38412/sctp            nr_gnb
6decad7e6681   docker_srslte     "/bin/sh -c 'cd /mnt…"   8 hours ago     Up 8 hours     2000-2001/tcp                             srsue_zmq
a24cb0176907   docker_srslte     "/bin/sh -c 'cd /mnt…"   8 hours ago     Up 8 hours     2000-2001/tcp, 2152/udp, 36412/sctp       srsenb_zmq
ubuntu@ip-172-31-27-141:~$
```

i-0889501b416cb43e1 (open5g_ueransim)
PublicIPs: 18.217.13.228 PrivateIPs: 172.31.27.141

© 2022, Amazon Web Services, Inc. or its affiliates.

Fig. 3. Docker Images

images, push those images to the Docker Hub Registry, and then deploy those images across multiple containers using Kubernetes, all while utilizing the AWS cloud environment for maintaining Ec2 instances (for maintaining Linux-based operating system machines) and AWS Elastic Kubernetes Service (for maintaining for too many containers / cluster) and entire flow will continuously integrate with help of Jenkins.

VI. Jenkins

Jenkins is a Java-based DevOps solution for continuous integration/continuous deployment (CI/CD) automation. In this Pipelines are used to implement CI/CD workflows. Jenkins runs as the server on different platforms like Windows, MacOS and Linux. It also runs as java servlet and it also runs on the java applications like Apache Tomcat. In recent times Jenkins has been adapted to run in Docker containers. To operate these jekins we will be creating pipelines. A pipeline is a sequence of actions the Jenkins server will take to carry out the necessary CI/CD process activities. For pipeline we have created four stages as show below:

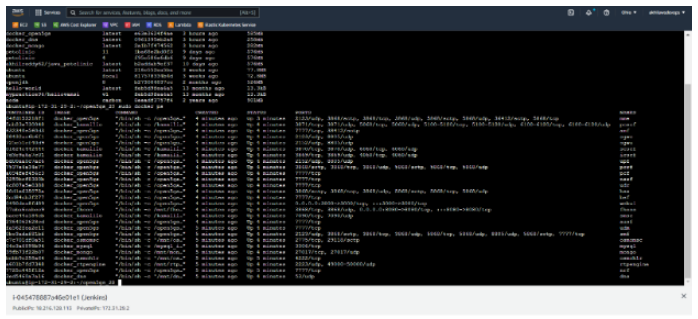
- Clone the code from GitHub Repository
- Stop the previous container that were build using docker compose
- Build base images of open5gs, kamailio, ueransim srslte
- Build and Run using docker-compose.

VII. Kubernetes

Kubernetes is an open-source container orchestration platform. Kubernetes allows you to deploy cloud-native applications anywhere and manage them exactly as you like everywhere. Kubernetes operates at the container level rather hardware level. Kubernetes streamlines application administration by automating operational activities associated with container management and providing built-in commands for application deployment, rollout of updates, scaling up and down to accommodate changing requirements, monitoring, and more.

VIII. AWS

AWS Management console will provide multiple ways for navigating to individual service console. Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing



```
[illegible]
```

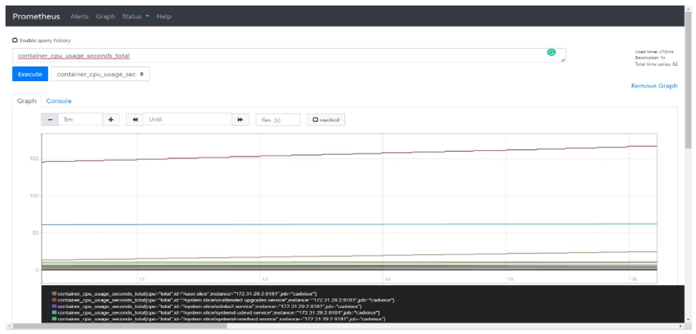
i-045478887a46e01e1 (Jenkins)

Fig. 4. Docker Container

capacity in the Amazon Web Services (AWS) Cloud. Block storage It is a technology which stores the data and divides the data into blocks for fast access. Security Policy for inbound value of ports access. IAM It is a web service which gives the secure access to AWS resources.

IX. Docker container and images

A Docker Container is a box with the ability to run Docker image templates. When you construct a container using one of those stable images, you basically create a read-write clone of that filesystem (a Docker image) within the selected container. Sudo docker images: list all the images build with given docker file docker compose Sudo docker ps : list all the container which are in active state Sudo docker ps -a : list all the container which are in active exited state



```
container_cpu_usage_seconds_total
```

Load time: 210ms
Resolution: 1s
Total time series: 62

```
container_cpu_usage_seconds_total{cpu="total",id="/user.slice",instance="172.31.29.2:8181",job="cadvisor"}
container_cpu_usage_seconds_total{cpu="total",id="/system.slice/unattended-upgrades.service",instance="172.31.29.2:8181",job="cadvisor"}
container_cpu_usage_seconds_total{cpu="total",id="/system.slice/udisks2.service",instance="172.31.29.2:8181",job="cadvisor"}
container_cpu_usage_seconds_total{cpu="total",id="/system.slice/systemd-udevd.service",instance="172.31.29.2:8181",job="cadvisor"}
container_cpu_usage_seconds_total{cpu="total",id="/system.slice/systemd-resolved.service",instance="172.31.29.2:8181",job="cadvisor"}
```

Fig. 5. Prometheus Dashboard

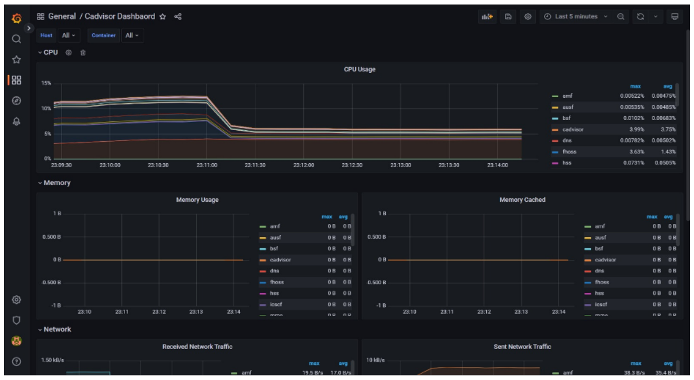


Fig. 6. Grafana Dashboard

X. Monitoring

Prometheus, one of the monitoring solutions promoted by the CNCF, as the monitoring framework to collect the status of the whole system. Prometheus is an open-source tool that provides monitoring and alerting for cloud native systems, such as Kubernetes. It can collect and store measurements as time-series data, including a timestamp for each entry. Labels, which are optional key-value pairs, can also be collected and recorded. We will deploy the cloud-native version of Prometheus in the same Kubernetes cluster used for the Open5Gs so that we could have a consistent picture of the present state of the mobile network in a single platform. It is open-source tool that provides monitoring and alerting for cloud-native systems. Time-series data, including a timestamp for each entry. Key features of Prometheus are following:

- Multidimensional Data Model
- Pushing Time-series Data
- Visualization
- Pull model
- Monitoring target discovery

XI. Grafana

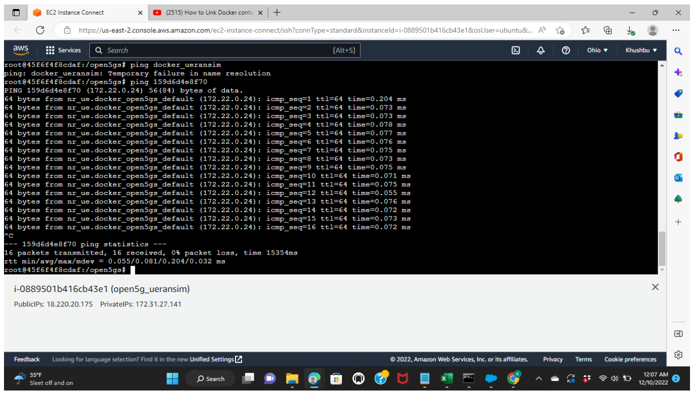
A visualizing tool with a dashboard that provides a lot of options with graphs. Open-source tool to view the metrics, do the queries, and get alerts of logs being generated. Charts and graphs are generated from data sources and available through a web browser. Using interactive query builders, end users may design sophisticated monitoring dashboards. A common addition to monitoring stacks is Grafana, which is frequently combined with time series databases like Influx DB, Prometheus, and Graphite.

Following are key features of grafana

- Dashboard templating
- Provisioning
- Annotations
- Custom Plugins
- SQL datasources

XII. Interaction

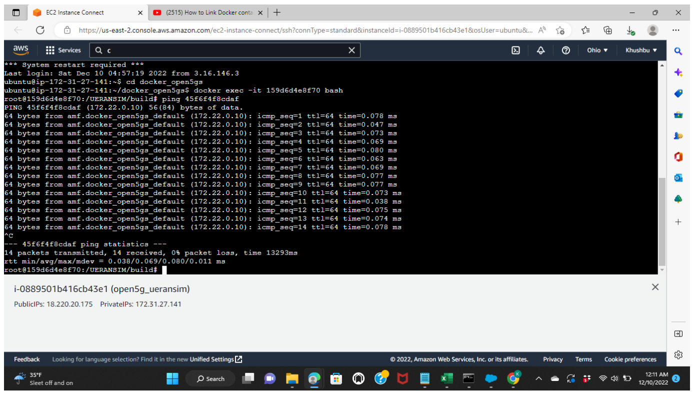
We tried to communicate the two different components of this open5gs and ueransim system



```
root@45f6f4f8cdaf:/open5gs# ping docker_ueransim
ping: docker_ueransim: Temporary failure in name resolution
root@45f6f4f8cdaf:/open5gs# ping 159d6d4e8f70
PING 159d6d4e8f70 (172.22.0.24) 56(84) bytes of data.
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=1 ttl=64 time=0.204 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=2 ttl=64 time=0.073 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=3 ttl=64 time=0.073 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=4 ttl=64 time=0.078 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=5 ttl=64 time=0.077 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=6 ttl=64 time=0.076 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=7 ttl=64 time=0.075 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=8 ttl=64 time=0.073 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=9 ttl=64 time=0.075 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=10 ttl=64 time=0.071 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=11 ttl=64 time=0.075 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=12 ttl=64 time=0.055 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=13 ttl=64 time=0.076 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=14 ttl=64 time=0.072 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=15 ttl=64 time=0.073 ms
64 bytes from nr_ue.docker_open5gs_default (172.22.0.24): icmp_seq=16 ttl=64 time=0.072 ms
^C
--- 159d6d4e8f70 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 15354ms
rtt min/avg/max/mdev = 0.055/0.081/0.204/0.032 ms
root@45f6f4f8cdaf:/open5gs#
```

i-0889501b416cb43e1 (open5g_ueransim)
PublicIPs: 18.220.20.175 PrivateIPs: 172.31.27.141

Fig. 7. Interaction between components Open5gs and Ueransim



```
*** System restart required ***
Last login: Sat Dec 10 04:57:19 2022 from 3.16.146.3
ubuntu@ip-172-31-27-141:~$ cd docker_open5gs
ubuntu@ip-172-31-27-141:~/docker_open5gs$ docker exec -it 159d6d4e8f70 bash
root@159d6d4e8f70:/UERANSIM/build# ping 45f6f4f8cdaf
PING 45f6f4f8cdaf (172.22.0.10) 56(84) bytes of data.
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=1 ttl=64 time=0.078 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=2 ttl=64 time=0.047 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=3 ttl=64 time=0.073 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=4 ttl=64 time=0.069 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=5 ttl=64 time=0.080 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=6 ttl=64 time=0.063 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=7 ttl=64 time=0.069 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=8 ttl=64 time=0.077 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=9 ttl=64 time=0.077 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=10 ttl=64 time=0.073 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=11 ttl=64 time=0.038 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=12 ttl=64 time=0.075 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=13 ttl=64 time=0.074 ms
64 bytes from amf.docker_open5gs_default (172.22.0.10): icmp_seq=14 ttl=64 time=0.078 ms
^C
--- 45f6f4f8cdaf ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13293ms
rtt min/avg/max/mdev = 0.038/0.069/0.080/0.011 ms
root@159d6d4e8f70:/UERANSIM/build#
```

i-0889501b416cb43e1 (open5g_ueransim)
PublicIPs: 18.220.20.175 PrivateIPs: 172.31.27.141

Fig. 8. Interaction between components Open5gs and Ueransim

XIII. Conclusion

With the adoption of these novel ideas, networking researchers need to be able to quickly and affordably establish end-to-end mobile networks for testing and development. In this project, we planned and created an end-to-end cloud-native open-source 5G and Ueransim system by utilizing already existing open-source software tools and creating new modules.

References

[1] https://github.com/open5gs/open5gs
[2] Nikolaos Apostolakis, Marco Gramaglia, and Pablo Serrano - Design and Validation of an Open-Source Cloud Native Mobile Network
[3] https://medium.com/rahasak/5g-core-network-setup-with-open5gs-and-ueransim-cd0e77025fd7
[4] https://open5gs.org/open5gs