Quick Installation

- We are using VM on Google Cloud with 16 core processors and 64GB of RAM
- Better to set up your VM with Max capacity
- Better to do ‘sudo –i’ and then redirect to /home/user directory

First please install VNC Viewer on your OS

Config VNC port on your Google Cloud and set the port 5901and tag

```
sudo -i
```

go to the root home

```
git clone https://github.com/openaicellular/main-file-repo.git
cd main-file-repo
chmod +x setup-vnc.sh
sudo ./setup-vnc.sh
```

enter a password for your vnc server

```
gnome-terminal
chmod +x ricinstallation.sh
sudo ./ricinstallation.sh
sudo kubectl get pods -A
cp -f -r srsrandeploy.sh oaic
cd oaic
chmod +x srsrandeploy.sh
./srsrandeploy.sh
cd..
```

Run the Network before Deploying the xApp

srsEPC:

Before deploying xApp make sure your network is up, check to run EPC, en-gNB, near-RT RIC, and UE:

```
sudo ip netns add ue1
```

```
sudo ip netns list
```

Now, in a new command window on Machine 1 run srsRAN EPC:

```
sudo srsepc
```

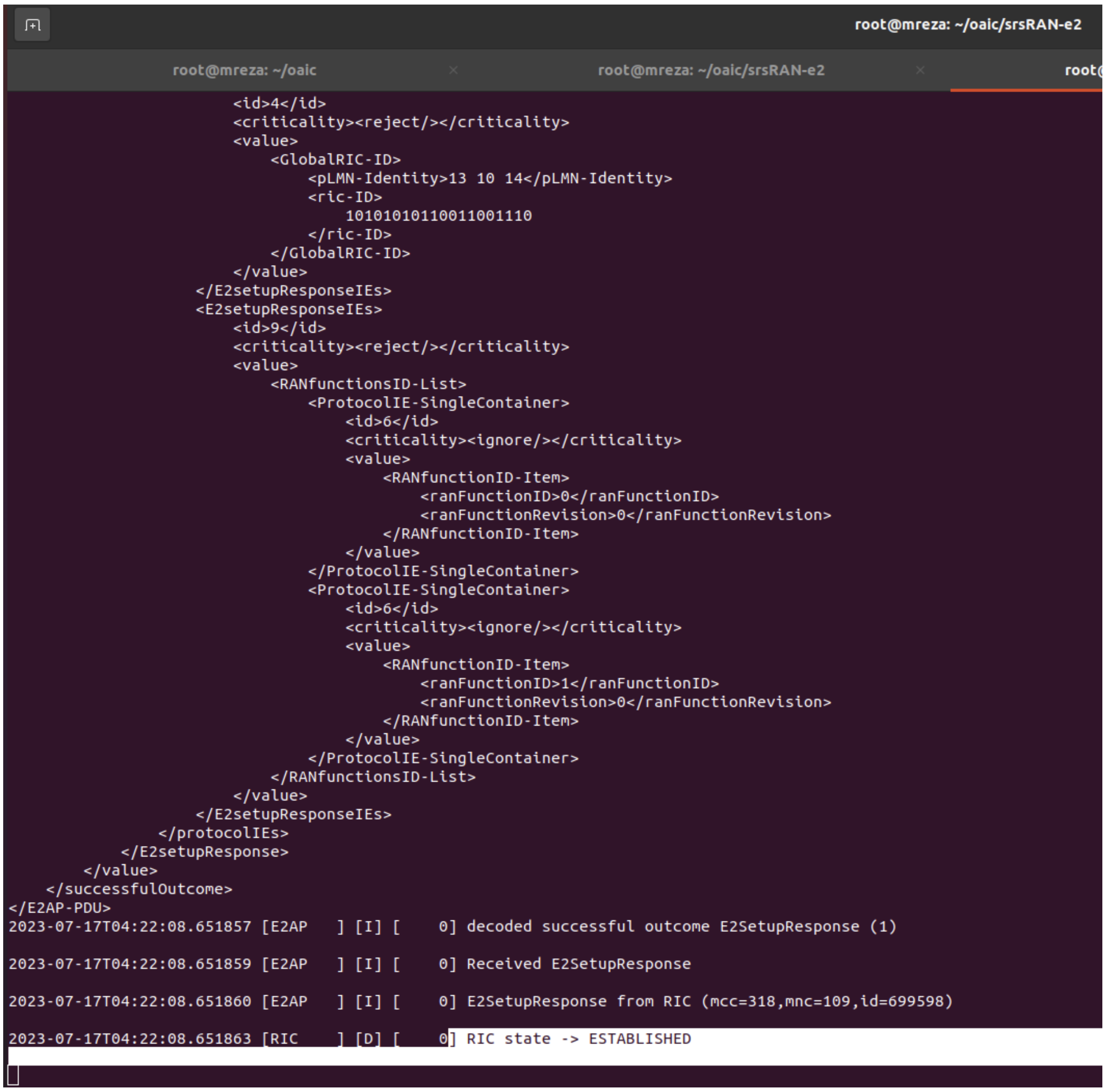
In a new command window on Machine 1 run srsRAN en-gNB.But before we start the en-gNB, we need to get the current machine's IP address and the IP address of the E2 Termination service at the near-RT RIC.

```
export E2NODE_IP=`hostname  -I | cut -f1 -d' '`
export E2NODE_PORT=5006
export E2TERM_IP=`sudo kubectl get svc -n ricplt --field-selector
metadata.name=service-ricplt-e2term-sctp-alpha -o
jsonpath='{.items[0].spec.clusterIP}'`
```

Run srsENB

```
sudo srsenb --enb.n_prb=50 --enb.name=enb1 --enb.enb_id=0x19B \
--rf.device_name=zmq --
rf.device_args="fail_on_disconnect=true,tx_port0=tcp://*:2000,rx_port0=tcp://localhost
:2001,tx_port1=tcp://*:2100,rx_port1=tcp://localhost:2101,id=enb,base_srate=23.04e6
" \
--ric.agent.remote_ipv4_addr=${E2TERM_IP} --log.all_level=warn --
ric.agent.log_level=debug --log.filename=stdout --
ric.agent.local_ipv4_addr=${E2NODE_IP} --ric.agent.local_port=${E2NODE_PORT}
```

Once the en-gNB is up and successfully connected to the near-RT RIC, you will see **E2 Setup** and **E2 Response** messages on the console. You will also see RIC Connection Initialized and RIC state established messages.



```
                        <id>4</id>
                        <criticality><reject/></criticality>
                        <value>
                            <GlobalRIC-ID>
                                <pLMN-Identity>13 10 14</pLMN-Identity>
                                <ric-ID>
                                    10101010110011001110
                                </ric-ID>
                            </GlobalRIC-ID>
                        </value>
                    </E2setupResponseIEs>
                    <E2setupResponseIEs>
                        <id>9</id>
                        <criticality><reject/></criticality>
                        <value>
                            <RANfunctionsID-List>
                                <ProtocolIE-SingleContainer>
                                    <id>6</id>
                                    <criticality><ignore/></criticality>
                                    <value>
                                        <RANfunctionID-Item>
                                            <ranFunctionID>0</ranFunctionID>
                                            <ranFunctionRevision>0</ranFunctionRevision>
                                        </RANfunctionID-Item>
                                    </value>
                                </ProtocolIE-SingleContainer>
                                <ProtocolIE-SingleContainer>
                                    <id>6</id>
                                    <criticality><ignore/></criticality>
                                    <value>
                                        <RANfunctionID-Item>
                                            <ranFunctionID>1</ranFunctionID>
                                            <ranFunctionRevision>0</ranFunctionRevision>
                                        </RANfunctionID-Item>
                                    </value>
                                </ProtocolIE-SingleContainer>
                            </RANfunctionsID-List>
                        </value>
                    </E2setupResponseIEs>
                </protocolIEs>
            </E2setupResponse>
        </value>
    </successfulOutcome>
</E2AP-PDU>
2023-07-17T04:22:08.651857 [E2AP   ] [I] [    0] decoded successful outcome E2SetupResponse (1)

2023-07-17T04:22:08.651859 [E2AP   ] [I] [    0] Received E2SetupResponse

2023-07-17T04:22:08.651860 [E2AP   ] [I] [    0] E2SetupResponse from RIC (mcc=318,mnc=109,id=699598)

2023-07-17T04:22:08.651863 [RIC    ] [D] [    0] RIC state -> ESTABLISHED
```

In the new terminal:

sudo srsue --gw.netns=ue1

PING:

This is the simplest way to test the network. This will test whether or not the UE and core can successfully communicate.

The ping command should be executed on a new terminal from the UE's network space

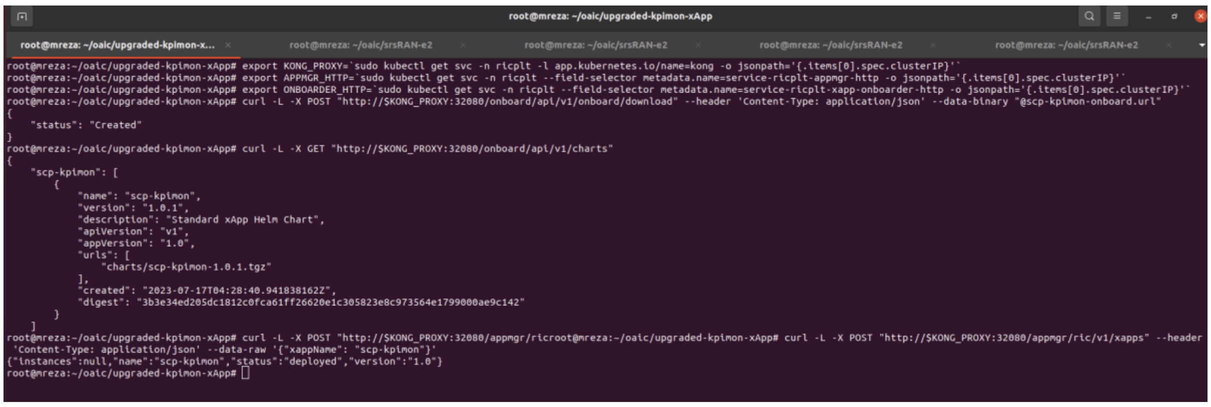
sudo ip netns exec ue1 ping 172.16.0.1

cp -f -r kpimondeploy.sh oaic

cd oaic

chmod +x kpimondeploy.sh

./kpimondeploy.sh


root@mreza: ~/oaic/upgraded-kpimon-xApp
root@mreza: ~/oaic/upgraded-kpimon-x...
root@mreza: ~/oaic/srsRAN-e2
root@mreza: ~/oaic/srsRAN-e2
root@mreza: ~/oaic/srsRAN-e2
root@mreza: ~/oaic/srsRAN-e2

```
root@mreza:~/oaic/upgraded-kpimon-xApp# export KONG_PROXY=`sudo kubectl get svc -n ricplt -l app.kubernetes.io/name=kong -o jsonpath='{.items[0].spec.clusterIP}'`
root@mreza:~/oaic/upgraded-kpimon-xApp# export APPMGR_HTTP=`sudo kubectl get svc -n ricplt --field-selector metadata.name=service-ricplt-appmgr-http -o jsonpath='{.items[0].spec.clusterIP}'`
root@mreza:~/oaic/upgraded-kpimon-xApp# export ONBOARDER_HTTP=`sudo kubectl get svc -n ricplt --field-selector metadata.name=service-ricplt-xapp-onboarder-http -o jsonpath='{.items[0].spec.clusterIP}'`
root@mreza:~/oaic/upgraded-kpimon-xApp# curl -L -X POST "http://$KONG_PROXY:32080/onboard/api/v1/onboard/download" --header 'Content-Type: application/json' --data-binary "@scp-kpimon-onboard.url"
{
    "status": "Created"
}
root@mreza:~/oaic/upgraded-kpimon-xApp# curl -L -X GET "http://$KONG_PROXY:32080/onboard/api/v1/charts"
{
    "scp-kpimon": [
        {
            "name": "scp-kpimon",
            "version": "1.0.1",
            "description": "Standard xApp Helm Chart",
            "apiVersion": "v1",
            "appVersion": "1.0",
            "urls": [
                "charts/scp-kpimon-1.0.1.tgz"
            ],
            "created": "2023-07-17T04:28:40.941838162Z",
            "digest": "3b3e34ed205dc1812c0fca61ff26620e1c305823e8c973564e1799000ae9c142"
        }
    ]
root@mreza:~/oaic/upgraded-kpimon-xApp# curl -L -X POST "http://$KONG_PROXY:32080/appmgr/ricroot@mreza:~/oaic/upgraded-kpimon-xApp# curl -L -X POST "http://$KONG_PROXY:32080/appmgr/ric/v1/xapps" --header
 'Content-Type: application/json' --data-raw '{"xappName": "scp-kpimon"}'
{"instances":null,"name":"scp-kpimon","status":"deployed","version":"1.0"}
root@mreza:~/oaic/upgraded-kpimon-xApp# 
```

Verifying xApp Deployment

There should be a ricxapp-scp-kpimon pod in ricxapp namespace

sudo kubectl get pods -A

We can check the xApp logs using

sudo kubectl logs -f -n ricxapp -l app=ricxapp-scp-kpimon

Since the E2 Node is already up and running and the Key Performance Metrics (KPM) RAN function is enabled by default, the xApp will be able to subscribe to the E2 Node and start getting INDICATION messages. The decoded message containing information about the metrics is stored in the kpimon.log within the pod. This can be viewed by,

> sudo kubectl exec -it -n ricxapp `sudo kubectl get pod -n ricxapp -l app=ricxapp-scp-kpimon -o jsonpath='{.items[0].metadata.name}'` -- tail -F /opt/kpimon.log

cp -f -r oaictdeploy.sh oaic

chmod +x oaictdeploy.sh

sudo ./ oaictdeploy.sh

sudo kubectl logs -f -n ricxapp -l app=ricxapp-SC3

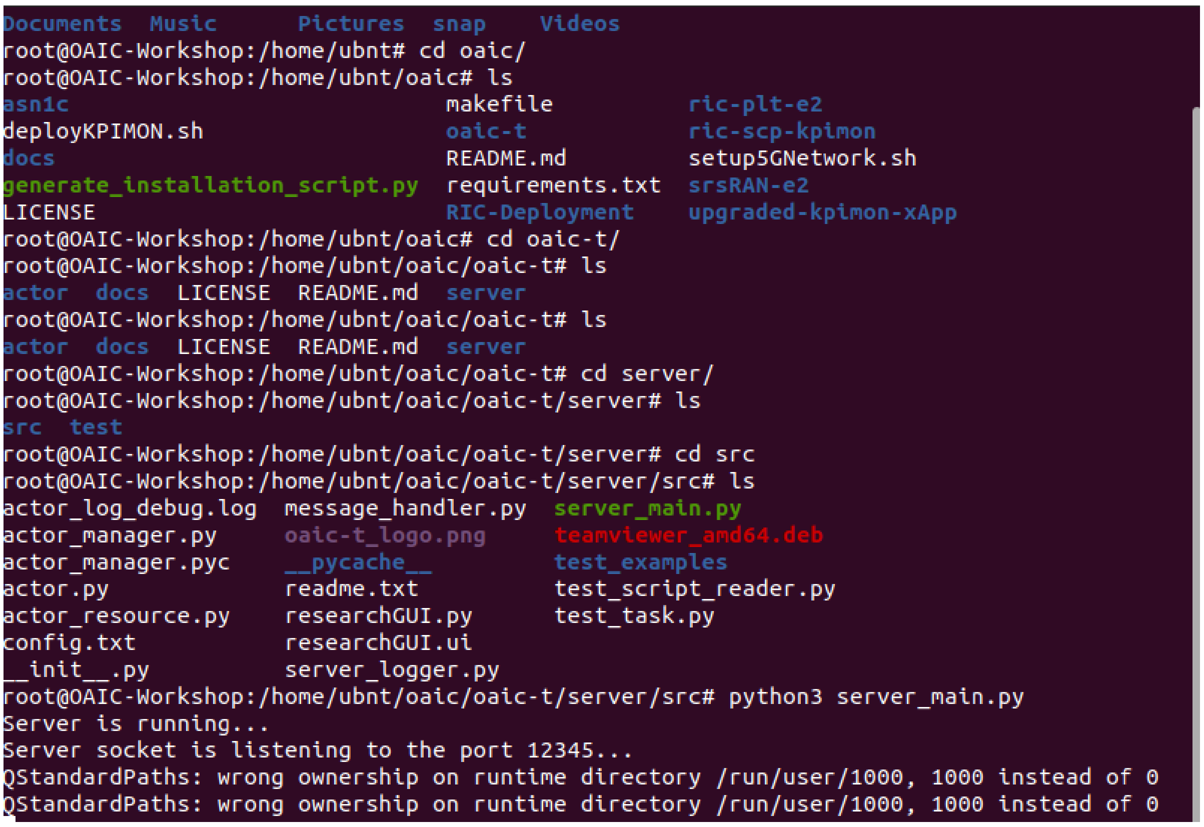
Copy IP from log to the order xml

Go to the vnc server

Step 1. Run the OAIC-T Server:

cd server/src

python3 server_main.py

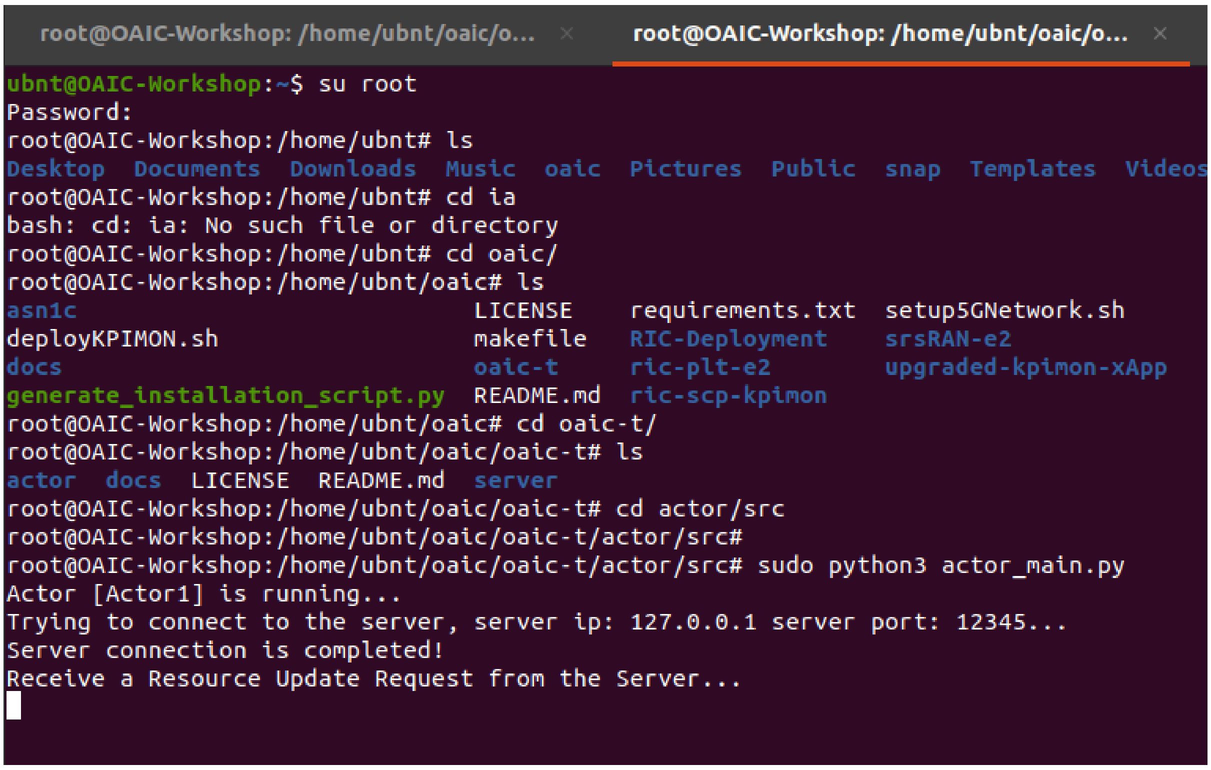


```
Documents  Music      Pictures  snap    Videos
root@OAIC-Workshop:/home/ubnt# cd oaic/
root@OAIC-Workshop:/home/ubnt/oaic# ls
asn1c                            makefile          ric-plt-e2
deployKPIMON.sh                  oaic-t            ric-scp-kpimon
docs                             README.md         setup5GNetwork.sh
generate_installation_script.py  requirements.txt  srsRAN-e2
LICENSE                          RIC-Deployment    upgraded-kpimon-xApp
root@OAIC-Workshop:/home/ubnt/oaic# cd oaic-t/
root@OAIC-Workshop:/home/ubnt/oaic/oaic-t# ls
actor  docs  LICENSE  README.md  server
root@OAIC-Workshop:/home/ubnt/oaic/oaic-t# ls
actor  docs  LICENSE  README.md  server
root@OAIC-Workshop:/home/ubnt/oaic/oaic-t# cd server/
root@OAIC-Workshop:/home/ubnt/oaic/oaic-t/server# ls
src  test
root@OAIC-Workshop:/home/ubnt/oaic/oaic-t/server# cd src
root@OAIC-Workshop:/home/ubnt/oaic/oaic-t/server/src# ls
actor_log_debug.log  message_handler.py  server_main.py
actor_manager.py     oaic-t_logo.png     teamviewer_amd64.deb
actor_manager.pyc    __pycache__         test_examples
actor.py             readme.txt          test_script_reader.py
actor_resource.py    researchGUI.py      test_task.py
config.txt           researchGUI.ui
__init__.py          server_logger.py
root@OAIC-Workshop:/home/ubnt/oaic/oaic-t/server/src# python3 server_main.py
Server is running...
Server socket is listening to the port 12345...
QStandardPaths: wrong ownership on runtime directory /run/user/1000, 1000 instead of 0
QStandardPaths: wrong ownership on runtime directory /run/user/1000, 1000 instead of 0
```

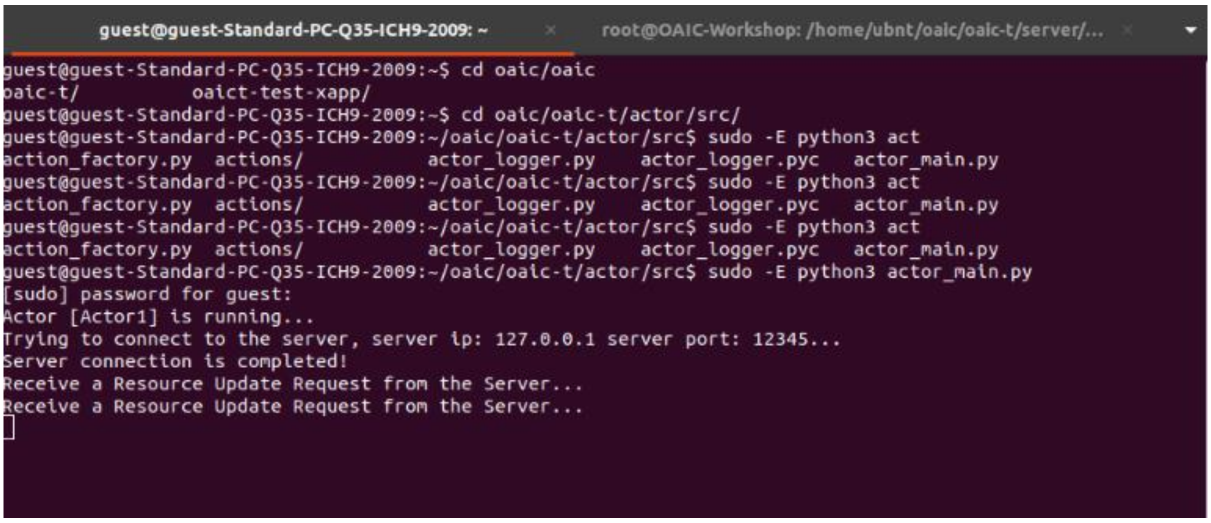
Step 2. Run the OAIC-T Actor(s):

cd actor/src

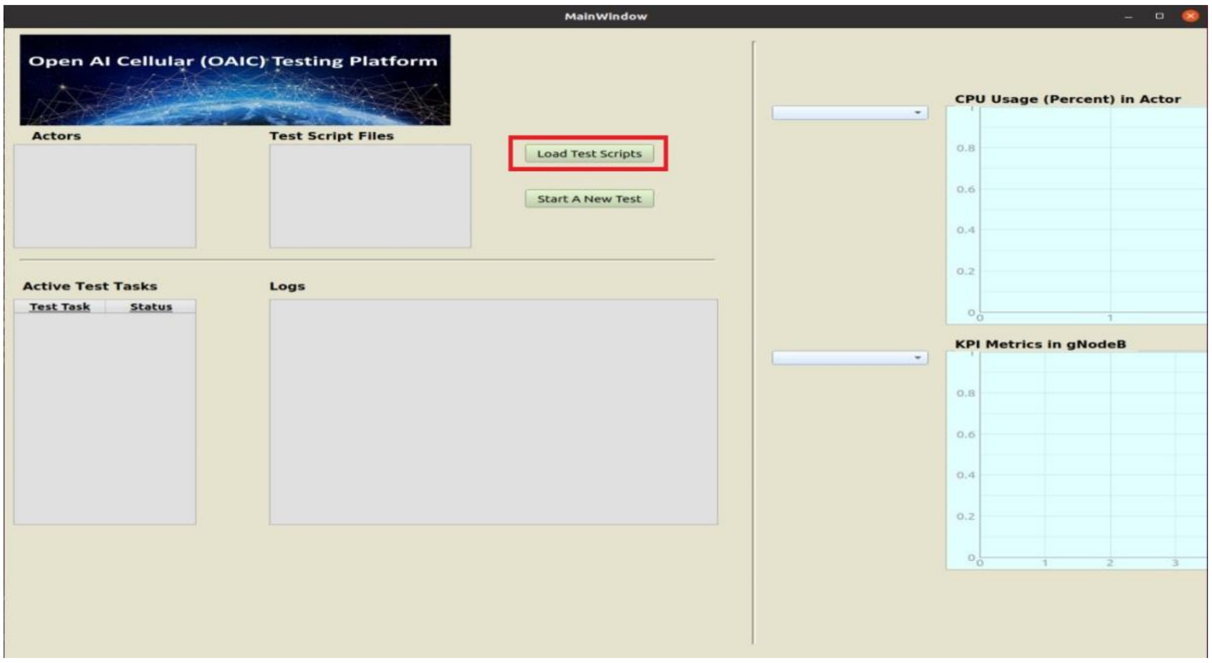
sudo python3 actor_main.py

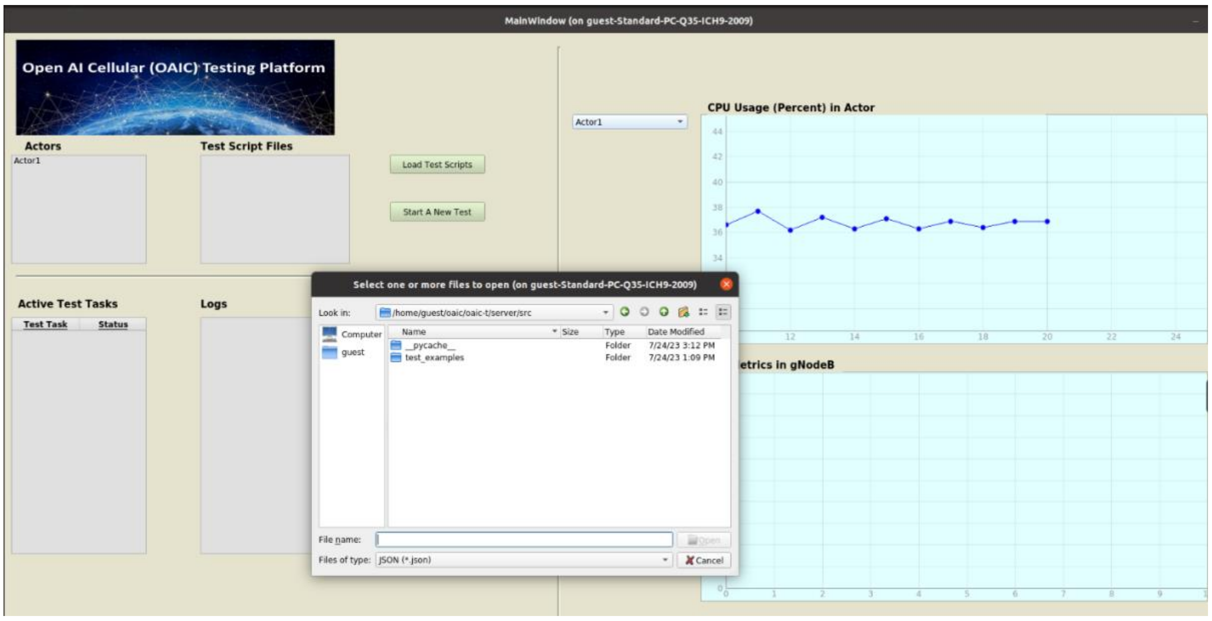

root@OAIC-Workshop: /home/ubnt/oaic/o...
root@OAIC-Workshop: /home/ubnt/oaic/o...

```
ubnt@OAIC-Workshop:~$ su root
Password:
root@OAIC-Workshop:/home/ubnt# ls
Desktop  Documents  Downloads  Music  oaic  Pictures  Public  snap  Templates  Videos
root@OAIC-Workshop:/home/ubnt# cd ia
bash: cd: ia: No such file or directory
root@OAIC-Workshop:/home/ubnt# cd oaic/
root@OAIC-Workshop:/home/ubnt/oaic# ls
asn1c                           LICENSE    requirements.txt  setup5GNetwork.sh
deployKPIMON.sh                 makefile   RIC-Deployment    srsRAN-e2
docs                            oaic-t     ric-plt-e2        upgraded-kpimon-xApp
generate_installation_script.py README.md  ric-scp-kpimon
root@OAIC-Workshop:/home/ubnt/oaic# cd oaic-t/
root@OAIC-Workshop:/home/ubnt/oaic/oaic-t# ls
actor  docs  LICENSE  README.md  server
root@OAIC-Workshop:/home/ubnt/oaic/oaic-t# cd actor/src
root@OAIC-Workshop:/home/ubnt/oaic/oaic-t/actor/src#
root@OAIC-Workshop:/home/ubnt/oaic/oaic-t/actor/src# sudo python3 actor_main.py
Actor [Actor1] is running...
Trying to connect to the server, server ip: 127.0.0.1 server port: 12345...
Server connection is completed!
Receive a Resource Update Request from the Server...
```


guest@guest-Standard-PC-Q35-ICH9-2009: ~
root@OAIC-Workshop: /home/ubnt/oaic/oaic-t/server/...

```
guest@guest-Standard-PC-Q35-ICH9-2009:~$ cd oaic/oaic
oaic-t/          oaict-test-xapp/
guest@guest-Standard-PC-Q35-ICH9-2009:~$ cd oaic/oaic-t/actor/src/
guest@guest-Standard-PC-Q35-ICH9-2009:~/oaic/oaic-t/actor/src$ sudo -E python3 act
action_factory.py  actions/          actor_logger.py    actor_logger.pyc    actor_main.py
guest@guest-Standard-PC-Q35-ICH9-2009:~/oaic/oaic-t/actor/src$ sudo -E python3 act
action_factory.py  actions/          actor_logger.py    actor_logger.pyc    actor_main.py
guest@guest-Standard-PC-Q35-ICH9-2009:~/oaic/oaic-t/actor/src$ sudo -E python3 act
action_factory.py  actions/          actor_logger.py    actor_logger.pyc    actor_main.py
guest@guest-Standard-PC-Q35-ICH9-2009:~/oaic/oaic-t/actor/src$ sudo -E python3 actor_main.py
[sudo] password for guest:
Actor [Actor1] is running...
Trying to connect to the server, server ip: 127.0.0.1 server port: 12345...
Server connection is completed!
Receive a Resource Update Request from the Server...
Receive a Resource Update Request from the Server...
```





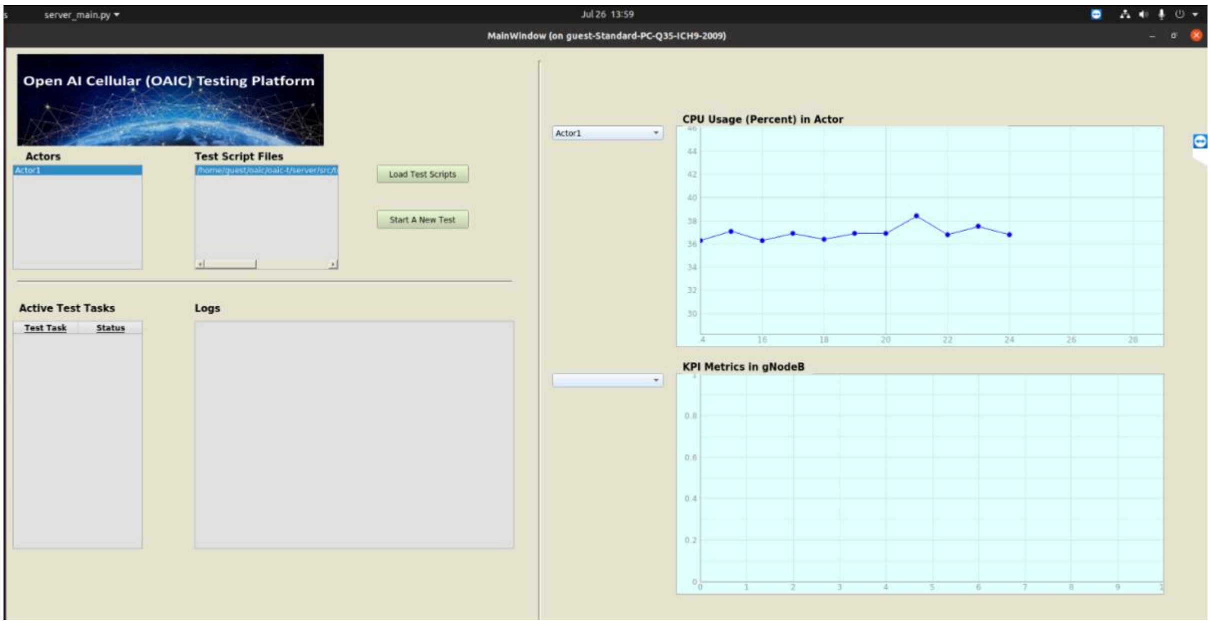
Select one or more files to open (on guest-Standard-PC-Q35-ICH9-2009)

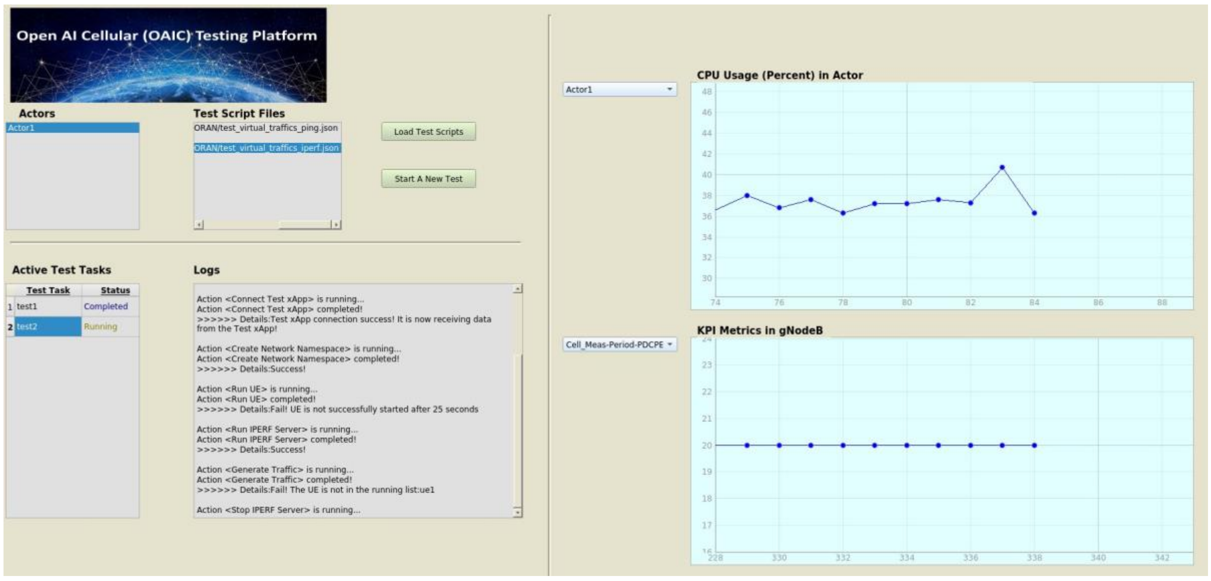
Look in: /home/guest/oaic/oaic-t/server/src

Name	Size	Type	Date Modified
__pycache__		Folder	7/24/23 3:12 PM
test_examples		Folder	7/24/23 1:09 PM

File name:

Files of type: JSON (*.json)


server_main.py
Jul 26 13:59
MainWindow (on guest-Standard-PC-Q35-ICH9-2009)
Open AI Cellular (OAIC) Testing Platform
Actors
Actor1
Test Script Files
/home/guest/oaic/oaic-t/server/src/t
Load Test Scripts
Start A New Test
Active Test Tasks
Test Task
Status
Logs
Actor1
CPU Usage (Percent) in Actor
KPI Metrics in gNodeB


Open AI Cellular (OAIC) Testing Platform
Actors
Actor1
Test Script Files
ORAN/test_virtual_traffics_ping.json
ORAN/test_virtual_traffics_iperf.json
Load Test Scripts
Start A New Test
Active Test Tasks
Logs
Actor1
CPU Usage (Percent) in Actor
Cell_Meas-Period-PDCPE
KPI Metrics in gNodeB

	Test Task	Status
1	test1	Completed
2	test2	Running

```
Action <Connect Test xApp> is running...
Action <Connect Test xApp> completed!
>>>>>> Details:Test xApp connection success! It is now receiving data
from the Test xApp!

Action <Create Network Namespace> is running...
Action <Create Network Namespace> completed!
>>>>>> Details:Success!

Action <Run UE> is running...
Action <Run UE> completed!
>>>>>> Details:Fail! UE is not successfully started after 25 seconds

Action <Run IPERF Server> is running...
Action <Run IPERF Server> completed!
>>>>>> Details:Success!

Action <Generate Traffic> is running...
Action <Generate Traffic> completed!
>>>>>> Details:Fail! The UE is not in the running list:ue1

Action <Stop IPERF Server> is running...
```