IOS MCN v0.1.0 Agartala Release

Installation Guide: SD-Core v0.1

# Introduction

The core software for the India Open Source for Mobile Communication Network (IOS-MCN) is based on open-source SD-Core. In this document, includes the installation and configurations for making the platform up and running for the integration with RAN and UPF. This includes emulated RAN test with built in tool.

# Purpose and Audience

This document is for installing and configuring the IOS-MCN core software on the server. It can be connected to RAN, attach UEs and integrate with UPF.

# Installation methods

The installation can be proceeded with this documentation and the core can get from IOS-MCN git repository. The details of each step are given on next section.

# Installation

## Hardware Requirements

The SD-Core installation requires the following prerequisites

* 4 Cores
* 16 GB RAM
* 40 GB Disk Storage

## Software Prerequisites

* Ubuntu 20.04 LTS Operating system
* Git
* Curl
* Make
* Net tools
* Netplan
* Ansible
* Python3
* Docker.io
* Ethtool

## Prerequisite Installation

Login to the Ubuntu 22.04.5 LTS desktop and take a terminal. The internet access is required for the machine.

Update the repository by the command

*sudo apt update*

After the successful completion of repository update start installing the required tools.

*sudo apt install git curl make net-tools pipx python3-venv docker.io*

*pipx install --include-deps ansible*

*pipx ensurepath*

*sudo apt install sshpass netplan.io iptables ethtool*

## Prerequisite Environment

### Firewall status

Verify the firewall is inactive on the system using the command

*sudo ufw status*

If the firewall is active, disable the firewall using the command

*sudo ufw disable*

### Networkd status

Verify the *systemd-networkd* status for the network configuration using the command

*systemctl status systemd-networkd.service*

4.4.3 Disable Generic receive offload (GRO)

Find the interface from the *ifconfig* command and input the interface name to the following command

*sudo ethtool -K <core-interface> gro off*

## Installation of IOS-MCN Core

### Clone aether onramp repo

create a directory for IOS-MCN in home directory

*cd*

*mkdir SD-Core*

*cd SD-Core*

*git clone --recursive https://github.com/5GC-DEV/IOSMCN-CoreDpm.git*

This brings up a Kubernetes cluster, deploy a 5G version of SD-Core on that cluster, and then connect that SD-Core to either an emulated 5G RAN or physical RAN.

*cd IOSMCN-CoreDpm/*

### Target Parameter Settings

Open the hosts.ini file

*vi hosts.ini*

Update IP address, username and password of the system

eg.,

*node1 ansible\_host=10.176.26.86 ansible\_user=ios5gn ansible\_password=ios5gn ansible\_sudo\_pass=ios5gn*

Open vars/main.yml

*vi vars/main.yml*

Update data\_iface name on *core*:

eg.,

*data\_iface: ens3*

Update data\_iface name on *gnbsim:router:*

*gnbsim:*

*router:*

*data\_iface: ens3*

Update IP address of the system

*amf:*

*ip: "10.176.26.86"*

Open *locale* file

*vi /etc/default/locale*

eg.,

*# File generated by update-locale*

*LANG="en\_IN.UTF-8"*

*LANGUAGE="en\_IN:en"*

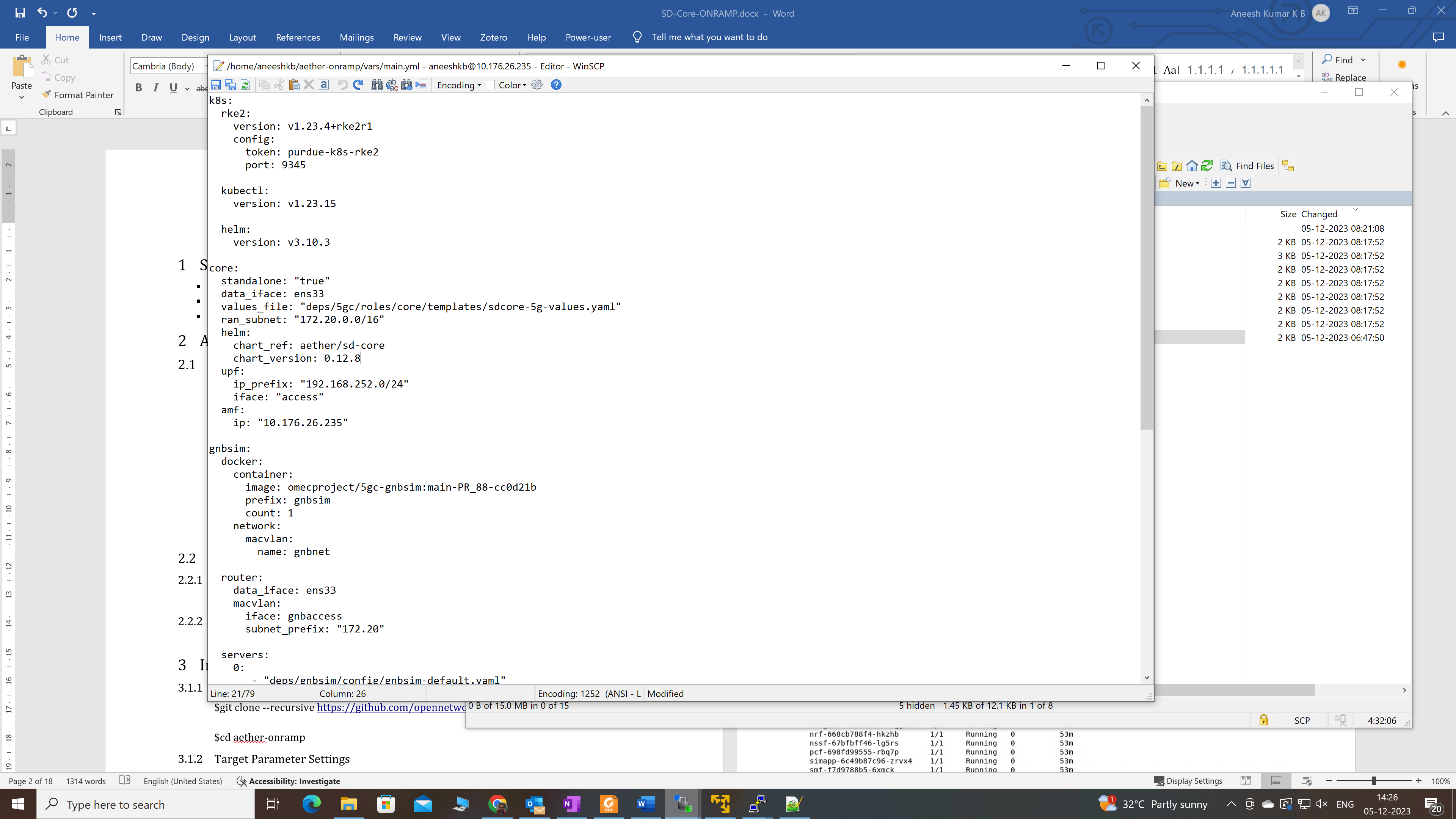


Figure 1: Parameter settings in var/main.yml

### Install Kubernetes

Start installation with the command

*make aether-k8s-install*

This may take several minutes to complete the installation. On its completion, verify the installation status by the command

*kubectl get pods --all-namespaces*

The successful output looks like

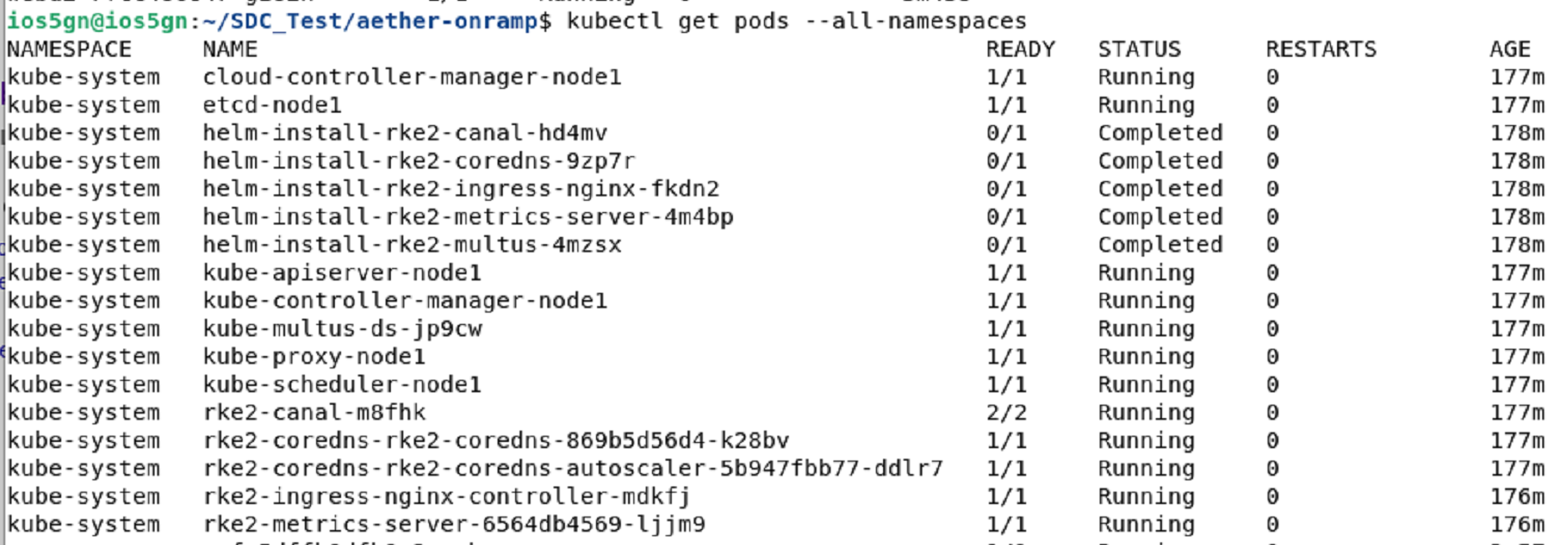


Figure 2:Output of Kubernetes installation

### Pre-Configuration for SD-Core

Verify the netpan is configured with ip, gateway and dns address.

eg.,

*network:*

*version: 2*

*ethernets:*

*ens3:*

*addresses:*

*- 10.176.26.86/23*

*gateway4: 10.176.26.1*

*nameservers:*

*addresses:*

*- 10.176.0.11*

If any change on the configuration, execute the command

*sudo netplan apply*

Login to the docker registry using github username and access token

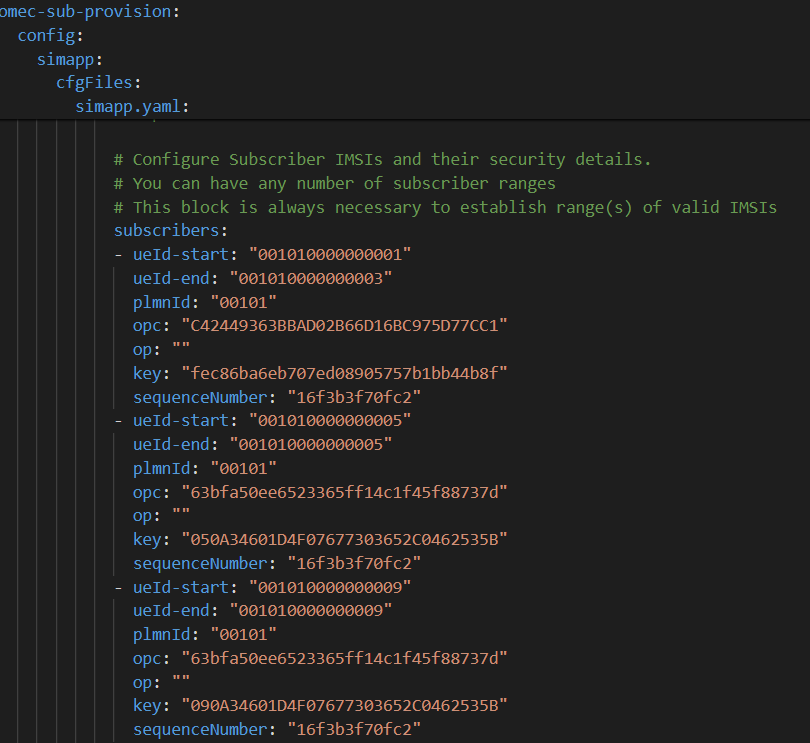
*sudo docker login ghcr.io -u <github-id> --password-stdin <<< <access-token>*

*Eg.,*

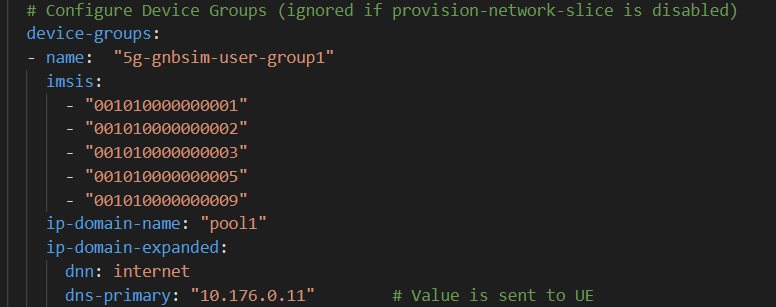
*sudo docker login ghcr.io -u ios5gn --password-stdin <<< ghp\_Sb0OuvbtSu8DRGouzz3aRcJDPaVPQ83DVAqm*

### Configure SD-Core

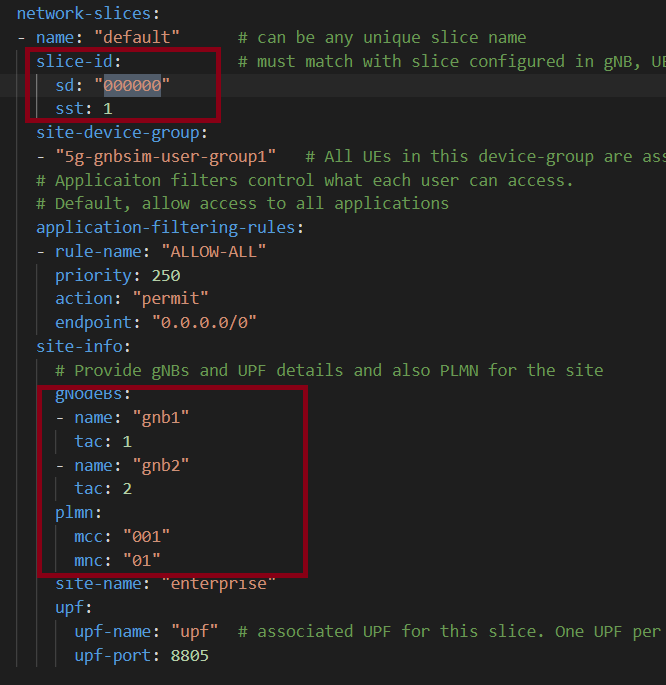
Modify the subscribers block of the omec-sub-provision section in file deps/5gc/roles/core/templates/ sdcore-5g-values.yaml to record the IMSI, OPc, and Key values configured onto your SIM cards. For example, the following code block adds IMSIs between 001010000000001 and 001010000000003



Further down in the same omec-sub-provision section you will find two other blocks that also need to be edited. The first, device-groups, assigns IMSIs to Device Groups. You will need to reenter the individual IMSIs from the subscribers block that will be part of the device-group. Also, update the DNS.



The second block, network-slices, sets various parameters associated with the Slices that connect device groups to applications. Here, you will need to reenter the PLMN information



### Install SD Core

Initiate the installation by the command

*make aether-5gc-install*

The successful outcome shall be verified using the following command

*kubectl get pods -n ios-mcn*

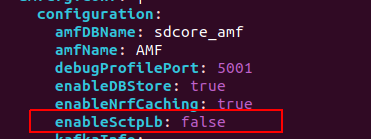


*Fig.3 Success state of SD-Core installation*

## Disable Load Balancer

$ kubectl edit cm amf -n ios~~-~~mcn

Update Sctplb to false and change mcc,mnc,sst and sd based on the sdcore-5g-values.yaml configuration



Delete amf pod for update

$kubectl delete pod <pod-name> -n ios-mcn

## Install Management Application

### Install Aether AMP

*make aether-amp-install*

If an error occurs, try to install again

### Access Management Interface

Access the dashboards for Aether's Runtime Control system and Aether's Monitoring system using the following URL:

http://<server\_ip>:31194

Access the Grafana using the URL

http://<server\_ip>:30950

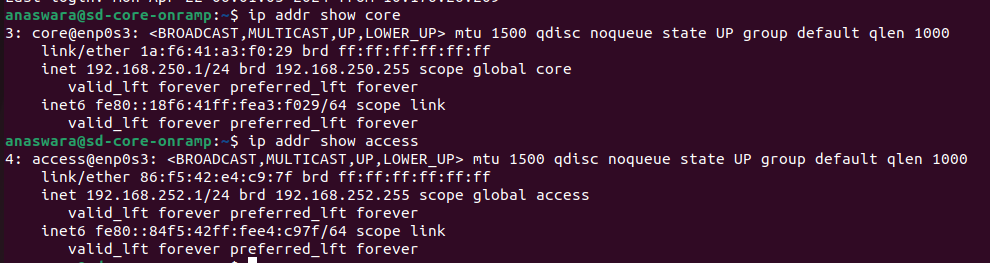
## Add Route in gNB

*sudo ip route add 192.168.252.0/24 via <core\_ip>*

## Verify Network

### Check core and access are properly configured outside the container

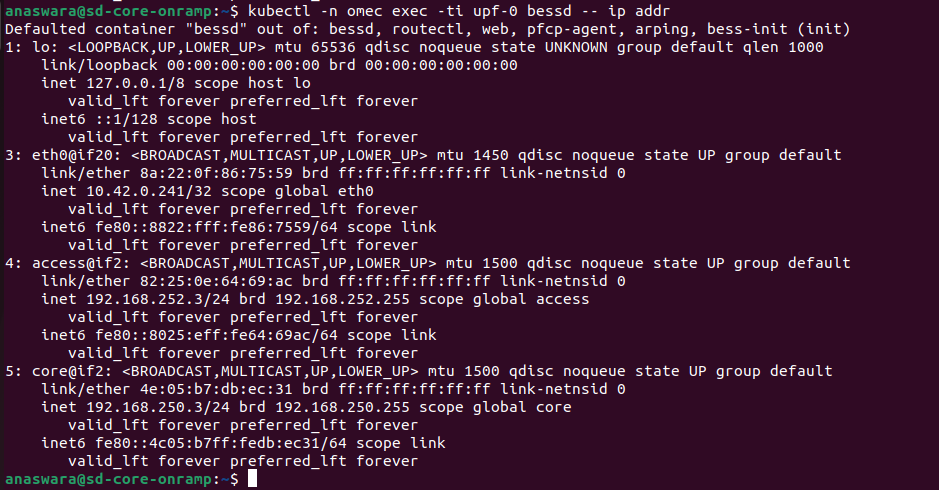
There are two bridges that connect the physical interface with the UPF container. The access bridge connects the UPF downstream to the RAN (this corresponds to 3GPP’s N3 interface) and is assigned IP subnet 192.168.252.0/24 . The core bridge connects the UPF upstream to the Internet (this corresponds to 3GPP’s N6 interface) and is assigned IP subnet 192.168.250.0/24 .

The above output from ip shows the two interfaces visible to the server, but running outside

the container.

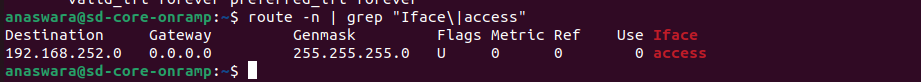
### Check core and access are properly configured inside the container

kubectl can be used to see what’s running inside the UPF, where bessd is the name of the container image that implements the UPF, and access and core are the last two interfaces shown below:



When packets flowing upstream from the gNB arrive on the server’s physical interface, they need to be forwarded over the access interface. This is done by having the following kernel route installed, which should be the case if your Aether installation was successful.

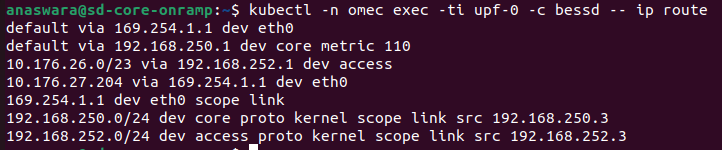
*route -n | grep "Iface\|access”*

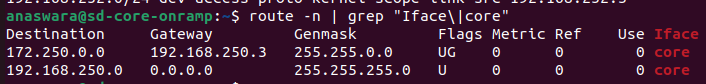


Within the UPF, the correct behavior is to forward packets between the access and core

interfaces. Upstream packets arriving on the access interface have their GTP headers

removed and the raw IP packets are forwarded to the core interface. The routes inside the UPF’s bessd container will look something like this:





The first rule above matches packets to the UEs on the 172.250.0.0/16 subnet. The next hop for these packets is the core IP address inside the UPF. The second rule says that next hop address is reachable on the core interface outside the UPF. As a result, the downstream packets arrive in the UPF where they are GTP-encapsulated with the IP address of the gNB.

Note that if you are not finding access and core interfaces outside the UPF, the following commands can be used to create these two interfaces manually (again using our running example for the physical ethernet interface):

*ip link add core link ens3 type macvlan mode bridge 192.168.250.3*

*ip link add access link ens3 type macvlan mode bridge 192.168.252.3*

Note that if you are find access and core interfaces outside the UPF, but route not generated the following commands can be used to create these two routes manually (again using our running example for the physical ethernet interface):

*sudo ip route add 192.168.252.0/24 dev access*

*sudo ip route add 192.168.250.0/24 dev core*

### Packet Traces

Packet traces are the best way to diagnose your deployment, and the most helpful traces you can capture are shown in the following commands. You can run these on the Aether server, where we use our example ens18 interface for illustrative purposes:

*sudo tcpdump -i any sctp -w sctp-test.pcap*

*sudo tcpdump -i ens18 port 2152 -w gtp-outside.pcap*

*sudo tcpdump -i access port 2152 -w gtp-inside.pcap*

*sudo tcpdump -i core net 172.250.0.0/16 -w n6-inside.pcap*

*sudo tcpdump -i ens18 net 172.250.0.0/16 -w n6-outside.pcap*

If the gtp-outside.pcap has packets and the gtp-inside.pcap is empty (no packets captured), you may run the following commands to make sure packets are forwarded from the ens18 interface to the access interface and vice versa:

*sudo iptables -A FORWARD -i ens18 -o access -j ACCEPT*

*sudo iptables -A FORWARD -i access -o ens18 -j ACCEPT*

## Routing Configuration inside UPF pod

### Enter the UPF Pod

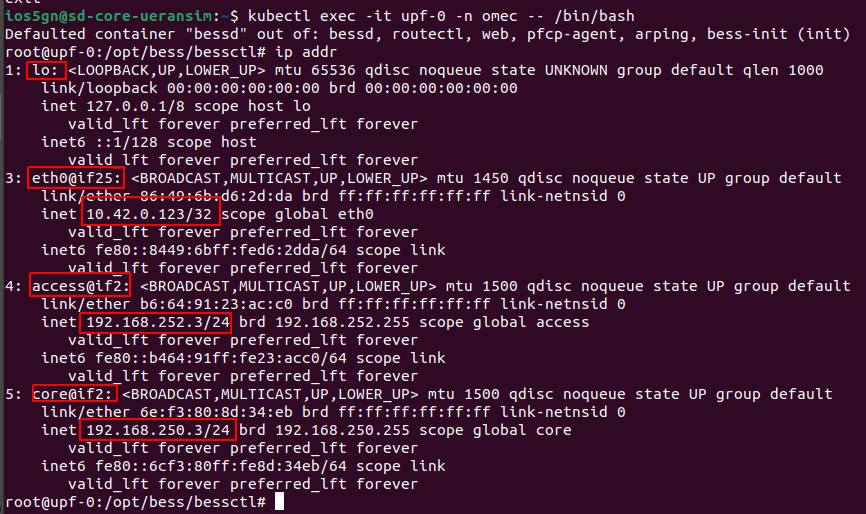
To enter the UPF pod, execute the following commands:

*kubectl exec -it upf-0 -n ios-mcn bash*

Once inside the pod, verify the network interfaces:

*ip addr*

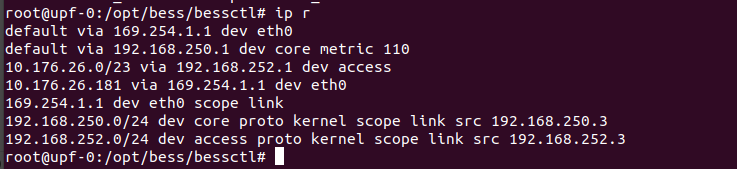
Ensure the following four interfaces are created: lo, eth0, access, and core.



#### Verify Routing Table

Check the routing table to ensure it matches the expected configuration

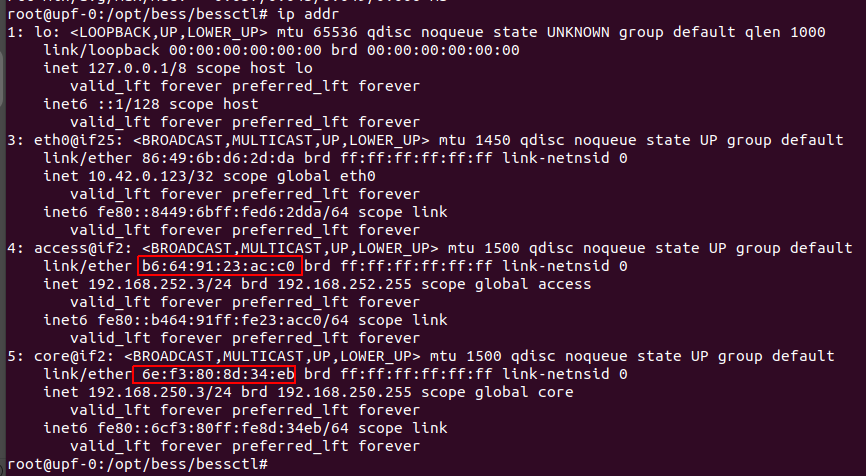
*ip r*



#### Ping test

### 

#### Check and Update ARP Entries



Ensure that the ARP entries for access and core interfaces have the correct MAC addresses. If the MAC addresses do not match, update them as follows:

*sudo arp -s 192.168.252.3 <mac-address-access>*

*sudo arp -s 192.168.250.3 <mac-address-core>*

Replace <mac-access> and <mac-core> with the actual MAC addresses of the access and core interfaces respectively [Inside UPF-POD].

1. Related Artifacts & links

|  |  |  |
| --- | --- | --- |
| **Document Name** | **Purpose** | **Link** |
| User Guide | Quick start quide | https |
| Installation Guide | Installation of SD-Core |  |
| Developer Guide | For SD-Core developers |  |
| Troubleshooting guide | For troubleshooting |  |
|  |  |  |
|  |  |  |