**srsRAN Project Installation**

Steps of Installation:

1. UHD Installation

2. srsRAN 4G

3. ZeroMQ

4. srsRAN 5G

5. Open5GS

**## UHD Installation**

**Source:** [**https://kb.ettus.com/Building\_and\_Installing\_the\_USRP\_Open-Source\_Toolchain\_(UHD\_and\_GNU\_Radio)\_on\_Linux**](https://kb.ettus.com/Building_and_Installing_the_USRP_Open-Source_Toolchain_(UHD_and_GNU_Radio)_on_Linux)

**Install the required dependencies for UHD and GNU Radio:**

sudo apt-get -y install autoconf automake build-essential ccache cmake cpufrequtils doxygen ethtool fort77 g++ gir1.2-gtk-3.0 git gobject-introspection gpsd gpsd-clients inetutils-tools libasound2-dev libboost-all-dev libcomedi-dev libcppunit-dev libfftw3-bin libfftw3-dev libfftw3-doc libfontconfig1-dev libgmp-dev libgps-dev libgsl-dev liblog4cpp5-dev libncurses5 libncurses5-dev libpulse-dev libqt5opengl5-dev libqwt-qt5-dev libsdl1.2-dev libtool libudev-dev libusb-1.0-0 libusb-1.0-0-dev libusb-dev libxi-dev libxrender-dev libzmq3-dev libzmq5 ncurses-bin python3-cheetah python3-click python3-click-plugins python3-click-threading python3-dev python3-docutils python3-gi python3-gi-cairo python3-gps python3-lxml python3-mako python3-numpy python3-opengl python3-pyqt5 python3-requests python3-scipy python3-setuptools python3-six python3-sphinx python3-yaml python3-zmq python3-ruamel.yaml swig wget

**Building and installing UHD from source code**

git clone https://github.com/EttusResearch/uhd

cd uhd

Next, checkout the desired UHD version

# Example: For UHD 3.15.0.0

git checkout v3.15.0.0

Next, create a build folder within the repository.

cd host

mkdir build

cd build

Next, invoke CMake.

cmake ..

Once the cmake command succeeds without errors, build UHD.

make

Next, install UHD, using the default install prefix, which will install UHD under the /usr/local/lib folder. You need to run this as root due to the permissions on that folder.

sudo make install

Next, update the system's shared library cache.

sudo ldconfig

Finally, make sure that the LD\_LIBRARY\_PATH environment variable is defined and includes the folder under which UHD was installed. Most commonly, you can add the line below to the end of your $HOME/.bashrc file:

export LD\_LIBRARY\_PATH=/usr/local/lib

If the LD\_LIBRARY\_PATH environment variable is already defined with other folders in your $HOME/.bashrc file, then add the line below to the end of your $HOME/.bashrc file to preserve the current settings.

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/usr/local/lib

**Downloading the UHD FPGA Images**

You can now download the UHD FPGA Images for this installation. This can be done by running the command uhd\_images\_downloader.

$ sudo uhd\_images\_downloader

Example output for UHD 3.15:

$ sudo uhd\_images\_downloader

[INFO] Images destination: /usr/local/share/uhd/images

[INFO] No inventory file found at /usr/local/share/uhd/images/inventory.json. Creating an empty one.

00006 kB / 00006 kB (100%) usrp1\_b100\_fw\_default-g6bea23d.zip

19484 kB / 19484 kB (100%) x3xx\_x310\_fpga\_default-g494ae8bb.zip

02757 kB / 02757 kB (100%) usrp2\_n210\_fpga\_default-g6bea23d.zip

02109 kB / 02109 kB (100%) n230\_n230\_fpga\_default-g494ae8bb.zip

00522 kB / 00522 kB (100%) usrp1\_b100\_fpga\_default-g6bea23d.zip

00474 kB / 00474 kB (100%) b2xx\_b200\_fpga\_default-g494ae8bb.zip

02415 kB / 02415 kB (100%) usrp2\_n200\_fpga\_default-g6bea23d.zip

05920 kB / 05920 kB (100%) e3xx\_e320\_fpga\_default-g494ae8bb.zip

15883 kB / 15883 kB (100%) n3xx\_n310\_fpga\_default-g494ae8bb.zip

00506 kB / 00506 kB (100%) b2xx\_b205mini\_fpga\_default-g494ae8bb.zip

18676 kB / 18676 kB (100%) x3xx\_x300\_fpga\_default-g494ae8bb.zip

00017 kB / 00017 kB (100%) octoclock\_octoclock\_fw\_default-g14000041.zip

04839 kB / 04839 kB (100%) usb\_common\_windrv\_default-g14000041.zip

00007 kB / 00007 kB (100%) usrp2\_usrp2\_fw\_default-g6bea23d.zip

00009 kB / 00009 kB (100%) usrp2\_n200\_fw\_default-g6bea23d.zip

00450 kB / 00450 kB (100%) usrp2\_usrp2\_fpga\_default-g6bea23d.zip

00142 kB / 00142 kB (100%) b2xx\_common\_fw\_default-g3ff4186b.zip

00460 kB / 00460 kB (100%) b2xx\_b200mini\_fpga\_default-g494ae8bb.zip

00319 kB / 00319 kB (100%) usrp1\_usrp1\_fpga\_default-g6bea23d.zip

00009 kB / 00009 kB (100%) usrp2\_n210\_fw\_default-g6bea23d.zip

11537 kB / 11537 kB (100%) n3xx\_n300\_fpga\_default-g494ae8bb.zip

05349 kB / 05349 kB (100%) e3xx\_e310\_fpga\_default-g494ae8bb.zip

00866 kB / 00866 kB (100%) b2xx\_b210\_fpga\_default-g494ae8bb.zip

[INFO] Images download complete.

At this point, UHD should be installed and ready to use. You can test this, with USRP device attached, by running

uhd\_find\_devices

**## srsRAN 4G**

**Source:** [**https://docs.srsran.com/projects/4g/en/latest/general/source/1\_installation.html**](https://docs.srsran.com/projects/4g/en/latest/general/source/1_installation.html)

**Package Installation**

The srsRAN 4G software suite can be installed using packages on Ubuntu:

sudo add-apt-repository ppa:softwareradiosystems/srsran

sudo apt-get update

sudo apt-get install srsran -y

**Installation from Source**

* Mandatory requirements:
  + Common:
    - [cmake](https://cmake.org/)
    - [libfftw](http://www.fftw.org/)
    - [mbedTLS](https://tls.mbed.org/)
  + srsUE:
    - [Boost](http://www.boost.org/)
  + srsENB:
    - [Boost](http://www.boost.org/)
    - [lksctp](http://lksctp.sourceforge.net/)
    - [config](http://www.hyperrealm.com/libconfig/)
  + srsEPC:
    - [Boost](http://www.boost.org/)
    - [lksctp](http://lksctp.sourceforge.net/)
    - [config](http://www.hyperrealm.com/libconfig/)

For example, on Ubuntu, one can install the required libraries with:

sudo apt-get install build-essential cmake libfftw3-dev libmbedtls-dev libboost-program-options-dev libconfig++-dev libsctp-dev

Download and build srsRAN 4G:

git clone https://github.com/srsRAN/srsRAN\_4G.git

cd srsRAN\_4G

mkdir build

cd build

cmake ../

make

make test

Install srsRAN 4G:

sudo make install

srsran\_install\_configs.sh user

This installs srsRAN 4G and also copies the default srsRAN 4G config files to *~/.config/srsran*.

**## ZeroMQ Installation**

**Source:** [**https://docs.srsran.com/projects/4g/en/latest/app\_notes/source/zeromq/source/index.html**](https://docs.srsran.com/projects/4g/en/latest/app_notes/source/zeromq/source/index.html)

First thing is to install ZeroMQ and build srsRAN 4G. On Ubuntu, ZeroMQ development libraries can be installed with:

sudo apt-get install libzmq3-dev

Alternatively, installing from sources can also be done.

First, one needs to install libzmq:

git clone https://github.com/zeromq/libzmq.git

cd libzmq

./autogen.sh

./configure

make

sudo make install

sudo ldconfig

Second, install czmq:

git clone https://github.com/zeromq/czmq.git

cd czmq

./autogen.sh

./configure

make

sudo make install

sudo ldconfig

Finally, you need to compile srsRAN 4G (assuming you have already installed all the required dependencies). Note, if you have already built and installed srsRAN 4G prior to installing ZMQ and other dependencies you will have to re-run the make command to ensure srsRAN 4G recognizes the addition of ZMQ:

git clone https://github.com/srsRAN/srsRAN\_4G.git

cd srsRAN\_4G

mkdir build

cd build

cmake ../

make

Put extra attention in the cmake console output. Make sure you read the following line:

...

-- FINDING ZEROMQ.

-- Checking **for** module 'ZeroMQ'

-- No package 'ZeroMQ' found

-- Found libZEROMQ: /usr/local/include, /usr/local/lib/libzmq.so

...

**## srsRAN 5G**

**Source:** [**https://docs.srsran.com/projects/project/en/latest/user\_manuals/source/installation.html**](https://docs.srsran.com/projects/project/en/latest/user_manuals/source/installation.html)

**Build Tools and Dependencies**[**ℑ**](https://docs.srsran.com/projects/project/en/latest/user_manuals/source/installation.html#build-tools-and-dependencies)

The srsRAN Project uses CMake and C++14. We recommend the following build tools:

* [cmake](https://cmake.org/)
* [gcc](https://gcc.gnu.org/) (v9.4.0 or later) **OR** [Clang](https://clang.llvm.org/) (v10.0.0 or later)

The srsRAN Project has the following necessary dependencies:

* [libfftw](https://www.fftw.org/)
* [libsctp](https://github.com/sctp/lksctp-tools)
* [yaml-cpp](https://github.com/jbeder/yaml-cpp)
* [PolarSSL/mbedTLS](https://www.trustedfirmware.org/projects/mbed-tls/)
* [googletest](https://github.com/google/googletest/)

You can install the required build tools and dependencies for various distributions as follows:

sudo apt-get install cmake make gcc g++ pkg-config libfftw3-dev libmbedtls-dev libsctp-dev libyaml-cpp-dev libgtest-dev

**Clone and Build**[**ℑ**](https://docs.srsran.com/projects/project/en/latest/user_manuals/source/installation.html#clone-and-build)

First, clone the srsRAN Project repository:

git clone https://github.com/srsRAN/srsRAN\_Project.git

Then build the code-base:

cd srsRAN\_Project

mkdir build

cd build

cmake ../

make -j **$(**nproc**)**

make test -j **$(**nproc**)**

You can now run the gNB from srsRAN\_Project/build/apps/gnb/. If you wish to install the srsRAN Project gNB, you can use the following command:

sudo make install

**Packages**[**ℑ**](https://docs.srsran.com/projects/project/en/latest/user_manuals/source/installation.html#packages)

download the srsRAN Project packages using the following commands:

sudo add-apt-repository ppa:softwareradiosystems/srsran-project

sudo apt-get update

sudo apt-get install srsran-project -y

**## Open5GS**

**Source:** [**https://open5gs.org/open5gs/docs/guide/02-building-open5gs-from-sources/**](https://open5gs.org/open5gs/docs/guide/02-building-open5gs-from-sources/)

**Building** **Open5GS from Sources:**

Getting MongoDB

Import the public key used by the package management system.

$ sudo apt update

$ sudo apt install gnupg

$ curl -fsSL https://pgp.mongodb.com/server-6.0.asc | sudo gpg -o /usr/share/keyrings/mongodb-server-6.0.gpg --dearmor

Create the list file /etc/apt/sources.list.d/mongodb-org-6.0.list for your version of Ubuntu.

$ echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-6.0.gpg] https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list

Install the MongoDB packages.

$ sudo apt update

$ sudo apt install -y mongodb-org

$ sudo systemctl start mongod **(if** '/usr/bin/mongod' is not running**)**

$ sudo systemctl enable mongod **(**ensure to automatically start it on system boot**)**

**\*\*NOTE :**  At command “ $ sudo apt install -y mongodb-org ” we get broken packages error so we skip some steps and directly jump to Git Clone open5gs.

**Building Open5GS:**

Git clone

$ git clone https://github.com/open5gs/open5gs

To compile with meson:

$ cd open5gs

$ meson build --prefix**=**`pwd`/install

$ ninja -C build

Check whether the compilation is correct.

$ ./build/tests/attach/attach *## EPC Only*

$ ./build/tests/registration/registration *## 5G Core Only*

Run all test programs as below.

$ cd build

$ meson test -v

You need to perform the **installation process**.

$ cd build

$ ninja install

$ cd ../

------------------------##################----------------------

**## srsRAN 4G Connection**

**Running a full end-to-end LTE network on a single computer :**

**Source:** [**https://docs.srsran.com/projects/4g/en/latest/app\_notes/source/zeromq/source/index.html**](https://docs.srsran.com/projects/4g/en/latest/app_notes/source/zeromq/source/index.html)

**Steps:**

* 1. **Network Namespace Creation**
  2. **Running the EPC**
  3. **Running the eNodeB**
  4. **Running the UE**

**Network Namespace Creation[ℑ](https://docs.srsran.com/projects/4g/en/latest/app_notes/source/zeromq/source/index.html" \l "network-namespace-creation" \o "Link to this heading)**

Let’s start with creating a new network namespace called “ue1” for the (first) UE:

sudo ip netns add ue1

To verify the new “ue1” netns exists, run:

sudo ip netns list

**Running the EPC[ℑ](https://docs.srsran.com/projects/4g/en/latest/app_notes/source/zeromq/source/index.html" \l "running-the-epc" \o "Link to this heading)**

Now let’s start the EPC. This will create a TUN device in the default network namespace and therefore needs root permissions.

sudo ./srsepc/src/srsepc

**\*\*Note:** We need to make a small change in this command as we are directly opening terminal from [srcepc] folder

devika@swift:~/Documents/SRS-RAN/srsRAN\_4G/srsepc$ sudo srsepc

**\*\*** nowcore network is ready to connect with base station

Output:

Built in Release mode using commit ec29b0c1f on branch master.

--- Software Radio Systems EPC ---

Couldn't open , trying /root/.config/srsran/epc.conf

Reading configuration file /root/.config/srsran/epc.conf...

Couldn't open user\_db.csv, trying /root/.config/srsran/user\_db.csv

HSS Initialized.

MME S11 Initialized

MME GTP-C Initialized

MME Initialized. MCC: 0xf001, MNC: 0xff01

SPGW GTP-U Initialized.

SPGW S11 Initialized.

SP-GW Initialized.

**Running the eNodeB[ℑ](https://docs.srsran.com/projects/4g/en/latest/app_notes/source/zeromq/source/index.html" \l "running-the-enodeb" \o "Link to this heading)**

./srsenb/src/srsenb --rf.device\_name=zmq --rf.device\_args="fail\_on\_disconnect=true,tx\_port=tcp://\*:2000,rx\_port=tcp://localhost:2001,id=enb,base\_srate=23.04e6"

\*\*Note: We need to make a small change in this command as we are directly opening terminal from [src] folder

devika@swift:~/Documents/SRS-RAN/srsRAN\_4G/srsenb/src$ sudo srsenb --rf.device\_name=zmq --rf.device\_args="fail\_on\_disconnect=true,tx\_port=tcp://\*:2000,rx\_port=tcp://localhost:2001,id=enb,base\_srate=23.04e6"

Output:

Active RF plugins: libsrsran\_rf\_uhd.so libsrsran\_rf\_zmq.so

Inactive RF plugins:

--- Software Radio Systems LTE eNodeB ---

Couldn't open , trying /root/.config/srsran/enb.conf

Reading configuration file /root/.config/srsran/enb.conf...

Couldn't open sib.conf, trying /root/.config/srsran/sib.conf

Couldn't open rr.conf, trying /root/.config/srsran/rr.conf

Couldn't open rb.conf, trying /root/.config/srsran/rb.conf

Built in Release mode using commit ec29b0c1f on branch master.

Opening 1 channels in RF device=zmq with args=fail\_on\_disconnect=true,tx\_port=tcp://\*:2000,rx\_port=tcp://localhost:2001,id=enb,base\_srate=23.04e6

Supported RF device list: UHD zmq file

CHx base\_srate=23.04e6

CHx id=enb

Current sample rate is 1.92 MHz with a base rate of 23.04 MHz (x12 decimation)

CH0 rx\_port=tcp://localhost:2001

CH0 tx\_port=tcp://\*:2000

CH0 fail\_on\_disconnect=true

==== eNodeB started ===

Type <t> to view trace

Current sample rate is 11.52 MHz with a base rate of 23.04 MHz (x2 decimation)

Current sample rate is 11.52 MHz with a base rate of 23.04 MHz (x2 decimation)

Setting frequency: DL=2680.0 Mhz, UL=2560.0 MHz for cc\_idx=0 nof\_prb=50

After running eNodeB, core network (EPC) output gets updated:

Built in Release mode using commit ec29b0c1f on branch master.

--- Software Radio Systems EPC ---

Couldn't open , trying /root/.config/srsran/epc.conf

Reading configuration file /root/.config/srsran/epc.conf...

Couldn't open user\_db.csv, trying /root/.config/srsran/user\_db.csv

HSS Initialized.

MME S11 Initialized

MME GTP-C Initialized

MME Initialized. MCC: 0xf001, MNC: 0xff01

SPGW GTP-U Initialized.

SPGW S11 Initialized.

SP-GW Initialized.

Received S1 Setup Request.

S1 Setup Request - eNB Name: srsenb01, eNB id: 0x19b

S1 Setup Request - MCC:001, MNC:01

S1 Setup Request - TAC 7, B-PLMN 0xf110

S1 Setup Request - Paging DRX v128

Sending S1 Setup Response

**Running the UE[ℑ](https://docs.srsran.com/projects/4g/en/latest/app_notes/source/zeromq/source/index.html" \l "running-the-ue" \o "Link to this heading)**

Lastly we can launch the UE, again with root permissions to create the TUN device.

sudo ./srsue/src/srsue --rf.device\_name=zmq --rf.device\_args="tx\_port=tcp://\*:2001,rx\_port=tcp://localhost:2000,id=ue,base\_srate=23.04e6" --gw.netns=ue1

The last command should start the UE and attach it to the core network. The UE will be assigned an IP address in the configured range (e.g. 172.16.0.2).

\*\*Note: We need to make a small change in this command as we are directly opening terminal from [src] folder

devika@swift:~/Documents/SRS-RAN/srsRAN\_4G/srsue/src$ sudo srsue --rf.device\_name=zmq --rf.device\_args="tx\_port=tcp://\*:2001,rx\_port=tcp://localhost:2000,id=ue,base\_srate=23.04e6" --gw.netns=ue1

Output:

Active RF plugins: libsrsran\_rf\_uhd.so libsrsran\_rf\_zmq.so

Inactive RF plugins:

Couldn't open , trying /root/.config/srsran/ue.conf

Reading configuration file /root/.config/srsran/ue.conf...

Built in Release mode using commit ec29b0c1f on branch master.

srsLog error - Unable to create log file "/tmp/ue.log": Permission denied

Opening 1 channels in RF device=zmq with args=tx\_port=tcp://\*:2001,rx\_port=tcp://localhost:2000,id=ue,base\_srate=23.04e6

Supported RF device list: UHD zmq file

CHx base\_srate=23.04e6

CHx id=ue

Current sample rate is 1.92 MHz with a base rate of 23.04 MHz (x12 decimation)

CH0 rx\_port=tcp://localhost:2000

CH0 tx\_port=tcp://\*:2001

Waiting PHY to initialize ... done!

Attaching UE...

Current sample rate is 1.92 MHz with a base rate of 23.04 MHz (x12 decimation)

Current sample rate is 1.92 MHz with a base rate of 23.04 MHz (x12 decimation)

.

Found Cell: Mode=FDD, PCI=1, PRB=50, Ports=1, CP=Normal, CFO=-0.2 KHz

Current sample rate is 11.52 MHz with a base rate of 23.04 MHz (x2 decimation)

Current sample rate is 11.52 MHz with a base rate of 23.04 MHz (x2 decimation)

Found PLMN: Id=00101, TAC=7

Random Access Transmission: seq=8, tti=341, ra-rnti=0x2

RRC Connected

Random Access Complete. c-rnti=0x46, ta=0

Network attach successful. IP: 172.16.0.2

Software Radio Systems RAN (srsRAN) 26/6/2024 5:36:28 TZ:0

Received RRC Connection Release (releaseCause: other)

RRC IDLE

After getting Connection, the eNodeB output gets updated like:

RACH: tti=4981, cc=0, pci=1, preamble=22, offset=0, temp\_crnti=0x47

User 0x47 connected

**## srsRAN 5G Connection**

**Source:** <https://docs.srsran.com/projects/project/en/latest/tutorials/source/srsUE/source/index.html>

<https://open5gs.org/open5gs/docs/guide/02-building-open5gs-from-sources/>

**Steps:**

1. Building the WebUI of Open5GS
2. Open5GS Core
3. gNB
4. srsUE

### Building the WebUI of Open5GS :

### Source: <https://open5gs.org/open5gs/docs/guide/02-building-open5gs-from-sources/>

Install the dependencies to run WebUI

Path in Terminal: ~/Documents/SRS-RAN/open5gs/webui$

$ cd webui

$ npm ci

The WebUI runs as an [npm](https://www.npmjs.com/) script.

$ npm run dev

Server listening can be changed by setting the environment variable HOSTNAME or PORT as below.

$ HOSTNAME**=**192.168.0.11 npm run dev

$ PORT**=**7777 npm run dev

### OutPut of $ PORT=7777 npm run dev

waxwing@waxwing:~/open5gs/webui$ PORT=7777 npm run dev

> open5gs@2.7.0 dev

> node server/index.js

> Using external babel configuration

> Location: "/home/waxwing/open5gs/webui/.babelrc"

DONE Compiled successfully in 2453ms 3:27:32 pm

Mongoose: subscribers.ensureIndex({ imsi: 1 }, { unique: true, background: true })

Mongoose: accounts.ensureIndex({ username: 1 }, { unique: true, background: true })

(node:3000181) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.

(Use `node --trace-deprecation ...` to show where the warning was created)

Mongoose: accounts.count({}, {})

(node:3000181) DeprecationWarning: collection.count is deprecated, and will be removed in a future version. Use Collection.countDocuments or Collection.estimatedDocumentCount instead

> Ready on http://localhost:7777

### Now copy the link <http://localhost:7777> in Browser and below window will showup:

### 

After login in webpage, we’ll get this terminal output add on:

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

> Building page: /

DONE Compiled successfully in 6794ms 3:30:34 pm

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

WAIT Compiling... 3:30:35 pm

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

Mongoose: subscribers.find({}, { projection: {} })

Mongoose: profiles.find({}, { projection: {} })

Mongoose: accounts.findOne({ '$or': [ { username: 'admin' } ] }, { projection: { hash: 0, salt: 0 } })

DONE Compiled successfully in 492ms 3:30:36 pm

………………………………….

\*\*Note: For some systems Webpage creation commands don’t work so first we start with running open5gs (the 2nd step) and there, in console output, we get the link for webpage.

**Running the Network**[**ℑ**](https://docs.srsran.com/projects/project/en/latest/tutorials/source/srsUE/source/index.html#running-the-network)

The following order should be used when running the network:

1. 5GC
2. gNB
3. UE

Source: <https://docs.srsran.com/projects/project/en/latest/tutorials/source/srsUE/source/index.html>

1. **Open5GS Core**[**ℑ**](https://docs.srsran.com/projects/project/en/latest/tutorials/source/srsUE/source/index.html#open5gs-core)

Path: ~/srsran/srsRAN\_Project/docker$ sudo docker-compose up --build 5gc

cd ./srsRAN\_Project/docker

docker compose up --build 5gc

Note: here docker-compose.yml file should be there in ~/srsran/srsRAN\_Project/docker path.

Here we are building the Core Network.

1. **gNB**[ℑ](https://docs.srsran.com/projects/project/en/latest/tutorials/source/srsUE/source/index.html#id4)

Path: ~/srsran/srsRAN\_Project/build/apps/gnb$ sudo gnb -c gnb\_zmq.yaml

We run gNB directly from the build folder (the config file is also located there) with the following command:

sudo gnb -c gnb\_zmq.yaml

Output:

[sudo] password for waxwing:

The PRACH detector will not meet the performance requirements with the configuration {Format 0, ZCZ 0, SCS 1.25kHz, Rx ports 1}.

Lower PHY in executor blocking mode.

--== srsRAN gNB (commit 78238fd15) ==--

Connecting to AMF on 10.53.1.2:38412

Available radio types: uhd and zmq.

Cell pci=1, bw=20 MHz, 1T1R, dl\_arfcn=368500 (n3), dl\_freq=1842.5 MHz, dl\_ssb\_arfcn=368410, ul\_freq=1747.5 MHz

==== gNodeB started ===

Type <t> to view trace

After running gNB, updated output in open5GS:

open5gs\_5gc | 07/15 10:15:58.215: [amf] INFO: gNB-N2 accepted[10.53.1.1]:46210 in ng-path module (../src/amf/ngap-sctp.c:113)

open5gs\_5gc | 07/15 10:15:58.215: [amf] INFO: gNB-N2 accepted[10.53.1.1] in master\_sm module (../src/amf/amf-sm.c:741)

open5gs\_5gc | 07/15 10:15:58.222: [amf] INFO: [Added] Number of gNBs is now 1 (../src/amf/context.c:1231)

open5gs\_5gc | 07/15 10:15:58.222: [amf] INFO: gNB-N2[10.53.1.1] max\_num\_of\_ostreams : 30 (../src/amf/amf-sm.c:780)

1. **srsUE**[ℑ](https://docs.srsran.com/projects/project/en/latest/tutorials/source/srsUE/source/index.html#id5)

Path: ~/srsran/srsRAN\_4G/srsue$ sudo srsue ue\_zmq.conf

We will suse srsUE in 4G as srsUE in 5G is not functioning properly .

sudo srsue ue\_zmq.conf

Terminal Output:

waxwing@waxwing:~/srsran/srsRAN\_4G/srsue$ sudo srsue ue\_zmq.conf

[sudo] password for waxwing:

Active RF plugins: libsrsran\_rf\_uhd.so libsrsran\_rf\_blade.so libsrsran\_rf\_zmq.so

Inactive RF plugins:

Reading configuration file ue\_zmq.conf...

Built in Release mode using commit ec29b0c1f on branch master.

Opening 1 channels in RF device=zmq with args=tx\_port=tcp://127.0.0.1:2001,rx\_port=tcp://127.0.0.1:2000,base\_srate=23.04e6

Supported RF device list: UHD bladeRF zmq file

CHx base\_srate=23.04e6

Current sample rate is 1.92 MHz with a base rate of 23.04 MHz (x12 decimation)

CH0 rx\_port=tcp://127.0.0.1:2000

CH0 tx\_port=tcp://127.0.0.1:2001

Current sample rate is 23.04 MHz with a base rate of 23.04 MHz (x1 decimation)

Current sample rate is 23.04 MHz with a base rate of 23.04 MHz (x1 decimation)

Waiting PHY to initialize ... done!

Attaching UE...

Random Access Transmission: prach\_occasion=0, preamble\_index=0, ra-rnti=0x39, tti=174

Random Access Complete. c-rnti=0x4601, ta=0

RRC Connected

PDU Session Establishment successful. IP: 10.45.1.2

RRC NR reconfiguration successful.

After running UE , Core Network output will show mote details, like:

pen5gs\_5gc | 07/15 10:29:08.669: [amf] INFO: InitialUEMessage (../src/amf/ngap-handler.c:401)

open5gs\_5gc | 07/15 10:29:08.669: [amf] INFO: [Added] Number of gNB-UEs is now 1 (../src/amf/context.c:2550)

open5gs\_5gc | 07/15 10:29:08.670: [amf] INFO: RAN\_UE\_NGAP\_ID[0] AMF\_UE\_NGAP\_ID[1] TAC[7] CellID[0x66c000] (../src/amf/ngap-handler.c:562)

open5gs\_5gc | 07/15 10:29:08.670: [amf] INFO: [suci-0-001-01-0000-0-0-0123456780] Unknown UE by SUCI (../src/amf/context.c:1835)

open5gs\_5gc | 07/15 10:29:08.670: [amf] INFO: [Added] Number of AMF-UEs is now 1 (../src/amf/context.c:1616)

open5gs\_5gc | 07/15 10:29:08.670: [gmm] INFO: Registration request (../src/amf/gmm-sm.c:1165)

open5gs\_5gc | 07/15 10:29:08.670: [gmm] INFO: [suci-0-001-01-0000-0-0-0123456780] SUCI (../src/amf/gmm-handler.c:166)

open5gs\_5gc | 07/15 10:29:08.909: [gmm] INFO: [imsi-001010123456780] Registration complete (../src/amf/gmm-sm.c:2146)

open5gs\_5gc | 07/15 10:29:08.910: [amf] INFO: [imsi-001010123456780] Configuration update command (../src/amf/nas-path.c:612)

open5gs\_5gc | 07/15 10:29:08.910: [gmm] INFO: UTC [2024-07-15T10:29:08] Timezone[0]/DST[0] (../src/amf/gmm-build.c:559)

open5gs\_5gc | 07/15 10:29:08.910: [gmm] INFO: LOCAL [2024-07-15T10:29:08] Timezone[0]/DST[0] (../src/amf/gmm-build.c:564)

open5gs\_5gc | 07/15 10:29:08.911: [amf] INFO: [Added] Number of AMF-Sessions is now 1 (../src/amf/context.c:2571)

open5gs\_5gc | 07/15 10:29:08.911: [gmm] INFO: UE SUPI[imsi-001010123456780] DNN[srsapn] S\_NSSAI[SST:1 SD:0xffffff] smContextRef [NULL] (../src/amf/gmm-handler.c:1241)

open5gs\_5gc | 07/15 10:29:08.911: [gmm] INFO: SMF Instance [fcc2c53a-4294-41ef-8f0f-9b0cb304f524] (../src/amf/gmm-handler.c:1280)

open5gs\_5gc | 07/15 10:29:08.912: [smf] INFO: [Added] Number of SMF-UEs is now 1 (../src/smf/context.c:1019)

open5gs\_5gc | 07/15 10:29:08.912: [smf] INFO: [Added] Number of SMF-Sessions is now 1 (../src/smf/context.c:3068)

open5gs\_5gc | 07/15 10:29:08.926: [smf] INFO: UE SUPI[imsi-001010123456780] DNN[srsapn] IPv4[10.45.1.2] IPv6[] (../src/smf/npcf-handler.c:539)

open5gs\_5gc | 07/15 10:29:08.926: [upf] INFO: [Added] Number of UPF-Sessions is now 1 (../src/upf/context.c:208)

open5gs\_5gc | 07/15 10:29:08.926: [gtp] INFO: gtp\_connect() [127.0.0.4]:2152 (../lib/gtp/path.c:60)

open5gs\_5gc | 07/15 10:29:08.926: [upf] INFO: UE F-SEID[UP:0x73a CP:0x972] APN[srsapn] PDN-Type[1] IPv4[10.45.1.2] IPv6[] (../src/upf/context.c:485)

open5gs\_5gc | 07/15 10:29:08.926: [upf] INFO: UE F-SEID[UP:0x73a CP:0x972] APN[srsapn] PDN-Type[1] IPv4[10.45.1.2] IPv6[] (../src/upf/context.c:485)

open5gs\_5gc | 07/15 10:29:08.927: [gtp] INFO: gtp\_connect() [10.53.1.2]:2152 (../lib/gtp/path.c:60)

open5gs\_5gc | 07/15 10:29:09.017: [gtp] INFO: gtp\_connect() [10.53.1.1]:2152 (../lib/gtp/path.c:60)

open5gs\_5gc | 07/15 10:29:09.021: [amf] INFO: [imsi-001010123456780:1:11][0:0:NULL] /nsmf-pdusession/v1/sm-contexts/{smContextRef}/modify (../src/amf/nsmf-handler.c:837)

open5gs\_5gc | 07/15 10:29:09.138: [gmm] INFO: [imsi-001010123456780] No GUTI allocated (../src/amf/gmm-sm.c:1443)