

Free5GC部署指南-BUPT NIRC

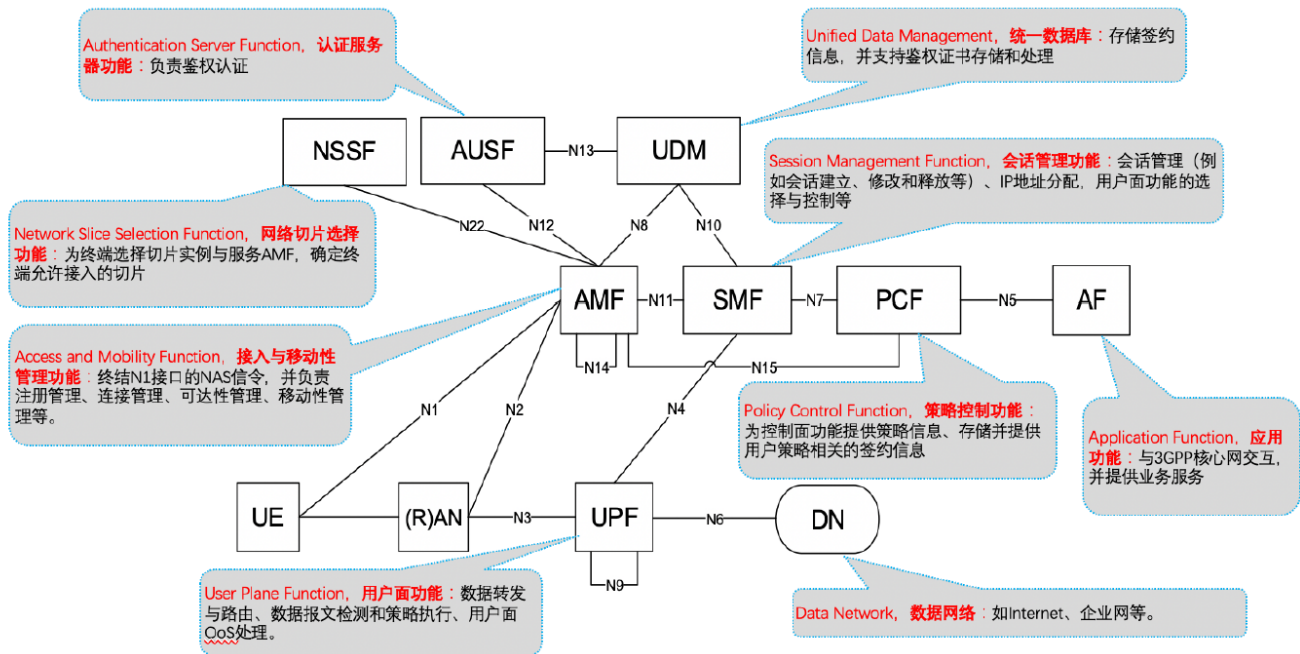
MADE BY 北京邮电大学-网络智能研究中心

- 1. 简要介绍
- 2. 说明
 - 2.1. 多种部署 Free5GC 的方法
 - 2.2. 服务器上的5GC部署情况
 - 2.3. 参考资料与网站
- 3. 系统镜像文件
- 4. KVM虚拟机的搭建方法
 - 4.1. 安装KVM
 - 4.2. 创建并配置网桥
 - 4.3. 安装虚拟机
 - 4.4. 配置虚拟机
 - 4.4.1. 配置hostname和密码
 - 4.4.2. 配置虚拟机的SSH
- 5. Free5GC-compose部署方法
 - 5.1. 5GC-COMMON-IMAGE.qcow2 镜像内容
 - 5.2. Free5gc-compose具体部署的步骤
 - 5.3. UERANSIM模拟设备安装
 - 5.4. Free5gc-compose完整环境运行
 - 5.5. 运行测试结果
- 6. Free5GC多虚拟机部署方法
 - 6.1. 在vm1中配置基本环境
 - 6.2. 利用vm1修改后的镜像创建vm2-v6
 - 6.3. 配置并编译5GC各个模块
 - 6.3.1. vm1虚拟机
 - 6.3.2. vm2虚拟机
 - 6.3.3. vm3虚拟机
 - 6.3.4. vm4虚拟机
 - 6.3.5. vm5虚拟机
 - 6.3.6. vm6虚拟机
 - 6.4. 配置部分配置
 - 6.5. 启动各个网元模块
 - 6.6. 问题与解决方案
 - 6.7. 如何关闭各网元并重启
 - 6.8. 所有网元后台运行（不挂断）

1. 简要介绍

文档围绕 Free5GC 的部署进行说明，主要记录了 Free5GC-compose 项目（利用docker-compose部署 Free5GC）和 多虚拟机部署Free5GC 两种方法。

5GC核心网的基础架构如下图所示



2. 说明

2.1. 多种部署 Free5GC 的方法

首先需要说明的是 Free5GC有多种部署方法，包括：

1. 官网给的 exe 文件（看到过但没用过，不知道能不能用）
2. 知乎上的 K8S 部署方法：<https://zhuanlan.zhihu.com/p/138629674>
3. 官网的 多合一 方法：利用其GitHub项目，将5GC很多模块放在一个虚拟机里面，再单独建一个UPF虚拟机装配UPF进行测试，官网网址：<https://www.free5gc.org/>
4. docker-compose 方法：利用 docker-compose 利用 单宿主机多容器 方法进行部署，github有参考项目：<https://github.com/free5gc/free5gc-compose>
5. 多虚拟机部署 方法（我们所使用的）：将不同组件放在不同虚拟机中，实现跨主机通信，便于QUIC改造，没有完整的官方参考网址，陈嘉楠师兄在部署时简要记录了部署过程：《5GC系统搭建说明》

我们最终需要实现的是 多机虚拟机 配置方法，与官方的单机配置方法 不同，可以多安装一些虚拟机，不同的组件放在不同的虚拟机中。因为容器化方法（docker-compose）会共用宿主机内核，在 QUIC改造并不方便，且会调用共用的宿主机内核协议栈，没有真正跨主机通信。而 K8S 方法并没有官方项目，参考内容较少，

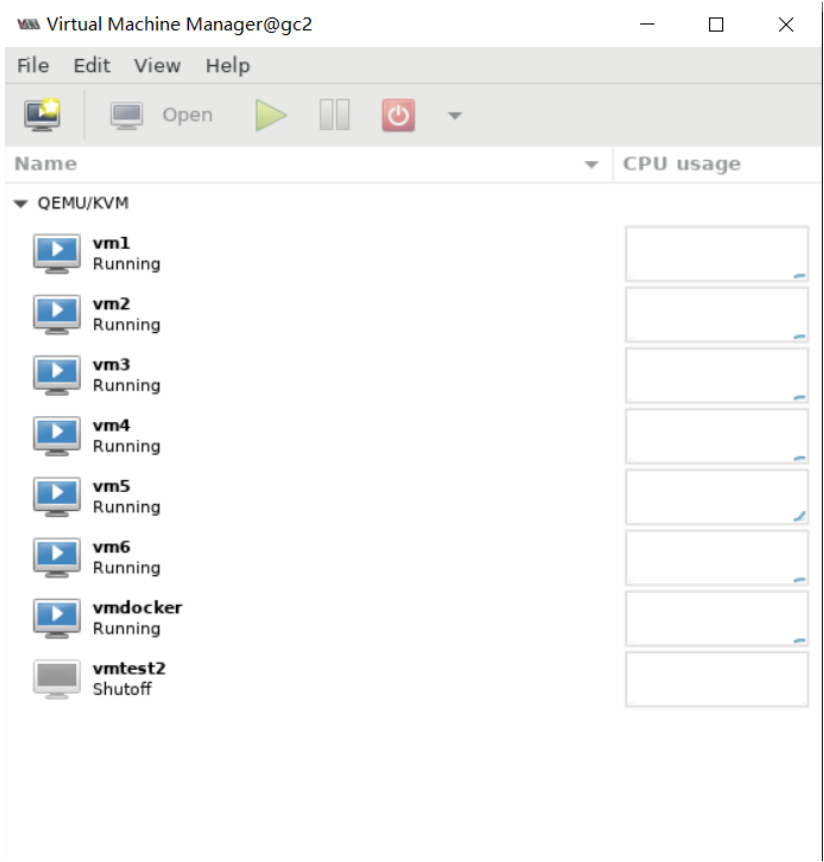
预计会有大坑。所以采用在官方单虚拟机方法的基础上进行改造的 **多虚拟机部署** 方法。

在部署的过程中，我首先实现了 **Free5GC-compose** 的部署，因为它有官方的github项目，但是部署过程中仍然有很多问题需要注意。建议先通过 **Free5GC-compose** 的部署熟悉整个 **Free5GC** 的架构，并且后续使用 **多虚拟机方法** 进行部署时，模块间的配置可以参考 **Free5GC-compose** 中的思路。

2.2. 服务器上的5GC部署情况

通过指令 `sudo virt-manager` 可以查看，第二台服务器的Free5GC部署情况如下

1. **vm1** 中部署 **gnb** [即为 (R)AN]
2. **vm2** 中部署 **mongodb**、**nrf**、**udr**、**udm**、**ausf**、**nssf**、**pcf**、**webconsole**
3. **vm3** 中部署 **amf**
4. **vm4** 中部署 **smf**
5. **vm5** 中部署 **upf**
6. **vm6** 中部署 **UE**
7. **vmdocker** 中部署了 **Free5GC-compose** 和 **UERANSIM**
8. **vmtest2** 为测试机器，基于我修改后的镜像建立，可以直接用它的镜像新建虚拟机



2.3. 参考资料与网站

由于Free5GC部署的资料很乱且很分散，故在此进行记录

1. Free5GC官网，网址为：<https://www.free5gc.org/>

需要 **注意** 的是：注意关注 **installation** 下的 **stage3:Free5GC** 的参考文件，因为这是它2020年更新后的最新版本。同时，这里给出的部署方式是 **多合一部署** 方式，即将所有模块部署在同一台虚拟机中

2. **Free5GC** 的GitHub官方项目集合：<https://github.com/free5gc>，主要参考以下项目

2.1. **Free5GC-compose** 官方项目网址：<https://github.com/free5gc/free5gc-compose> (docker-compose部署方式)

2.2. **Free5GC** 官方项目网址：<https://github.com/free5gc/free5gc> (多合一部署方式)

2.3. **UERANSIM** 官方项目网址：<https://github.com/aligungr/UERANSIM> (可以理解为 **5G手机和基站** 项目，可用于 **测试5G核心网** 和 **研究5G系统**)

3. 系统镜像文件

基于在Ubuntu18.04 (kernel version 5.0.0-23-generic) 镜像的基础上进行5GC所需环境的配置，主要修改了包括了 **已配置GO**、**已下载free5GC**、**已部分配置SSH** 等等内容，封装得到了 **5GC-COMMON-IMAGE.qcow2** 镜像，此后建立虚拟机都基于此镜像，避免了多次重复配置的麻烦。

4. KVM虚拟机的搭建方法

虚拟机部署的首要问题是虚拟机的配置和安装。需要在实验室的ubuntu服务器上 **安装kvm**。虚拟机之间的网络通信方式有两种：NAT模式、网桥模式。我们使用 **网桥模式**，这样可以为每个虚拟机分配独立的IP，并且自己搭建网桥，通过学校的DHCP动态分配给每个虚拟机一个校园局域网的IP，这样可以不通过宿主机而是直接 **ssh登录虚拟机**，方便登录和管理。坏处是每当学校的网络出现故障或者停电导致 **服务器重启** 时，机器的局域网IP便会改变。

4.1. 安装KVM

参考网上相关教程在服务器中安装 **KVM及其相关组件**，建议使用 **MobaXterm** 软件终端进行登陆，在利用KVM安装虚拟机时可以图形化。

4.2. 创建并配置网桥

1. 创建网桥，在后续会将宿主机和虚拟机全部绑到网桥上面

```
sudo brctl addbr br0
```

2. 修改配置文件 **/etc/netplan/00-installer**，其中红色圈内改为宿主机对应的网卡名，然后重启网络配置。这一步将网卡绑定到网桥上。可以先使用 **sudo netplan try** 测试是否内容无误，然后使用 **sudo netplan apply** 启用新的设定

```
ebupt@gc:/etc/netplan$ ls
00-installer-config.yaml  00-installer-config.yaml.back
ebupt@gc:/etc/netplan$ cat 00-installer-config.yaml
# This is the network config written by 's
#
network:
  ethernets:
    enp4s0f3:
      dhcp4: no
      dhcp6: no
      version: 2
  bridges:
    br0:
      interfaces: [enp4s0f3]
      dhcp4: yes
      #dhcp6: no
      #addresses: [10.108.124.212/22]
      # gateway4: 10.108.124.1
      #nameservers:
        # addresses: [10.3.9.4,10.3.9.5]
```

3. 开启网桥相关的内核配置，这一步是个大坑，一定要开启iptables的配置,文件位置在 `/etc/sysctl.conf`，需要添加的代码为

```
net.bridge.bridge-nf-call-ip6tables=0
net.bridge.bridge-nf-call-iptables=0
net.bridge.bridge-nf-call-arptables=0
```

4. 激活修改后的配置

```
sysctl -p /etc/sysctl.conf
```

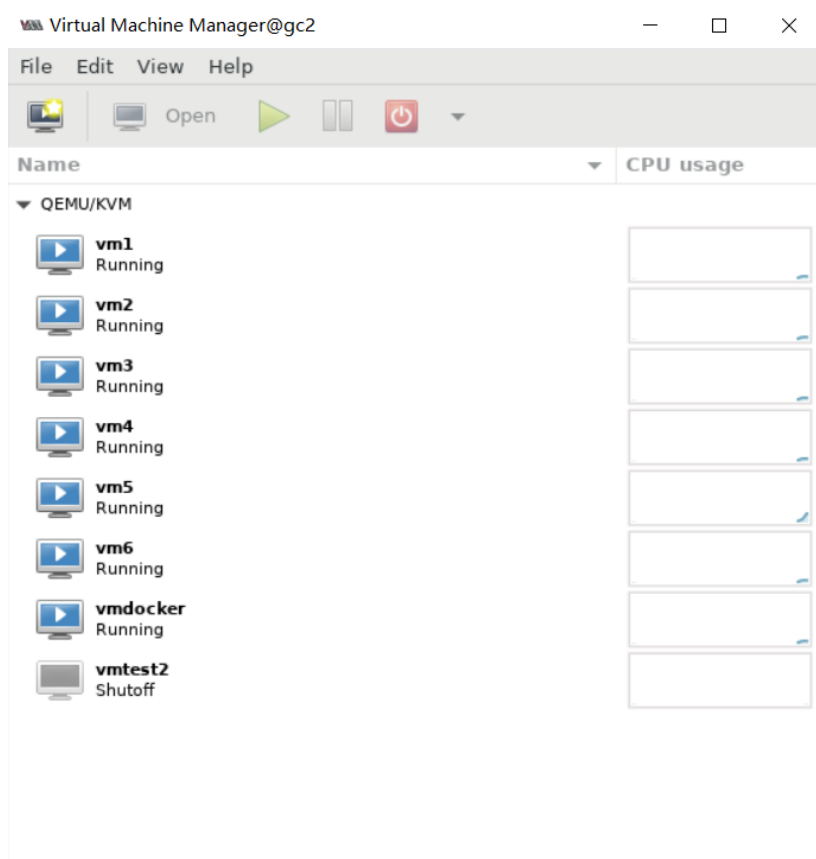
5. 后续所有虚拟机都挂到网桥上面之后，可以通过命令查看

```
brctl show
```

```
ebupt@gc2:/home/cjn/images$ brctl show
bridge name      bridge id        STP enabled     interfaces
br0               8000.b6d7825502be no               enp4s0f3
                  vnet0
                  vnet1
                  vnet2
                  vnet3
                  vnet4
```

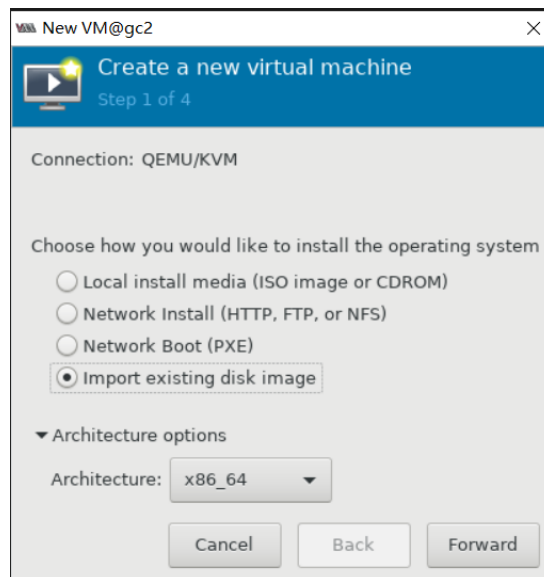
4.3. 安装虚拟机

1. 利用 `sudo virt-manager` 命令打开KVM控制界面。下面我以安装 `vmtest2` 为例，进行虚拟机的安装与配置

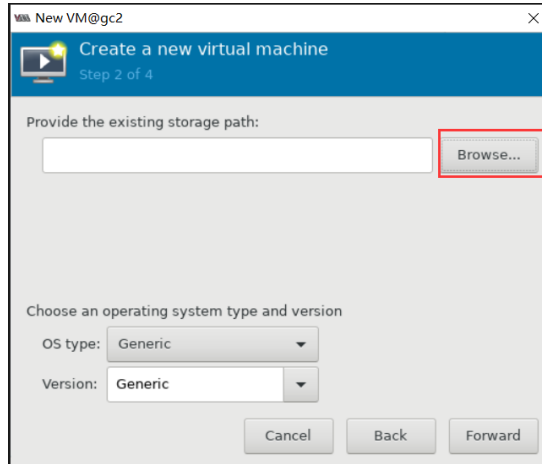


2. 首先需要复制一份 `5GC-COMMON-IMAGE.qcow2` 镜像到合适的位置。

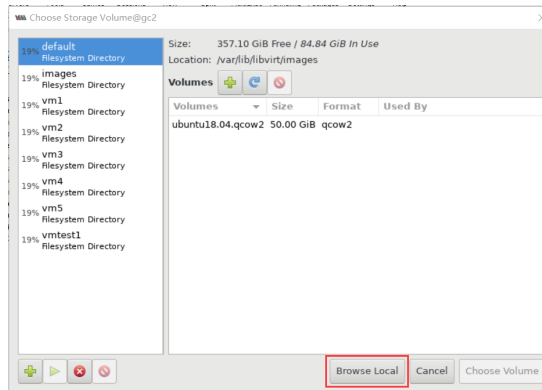
3. 点击 virt-manager 界面左上角的新建虚拟机，进入 **新建虚拟机** 界面。注意选择 **Import existing disk image** 方式新建虚拟机。我尝试了第一种方式，但是没有成功。



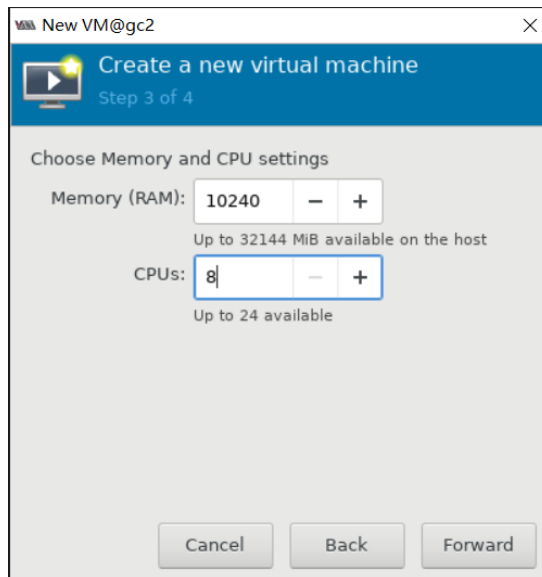
4. 点击 **Forward** 进入下一步的选择镜像的界面，点击 **Browse** 选择路径



5. 点击 **Browse Local** 选择刚刚复制的那个镜像



6. 配置虚拟机的内存和CPU

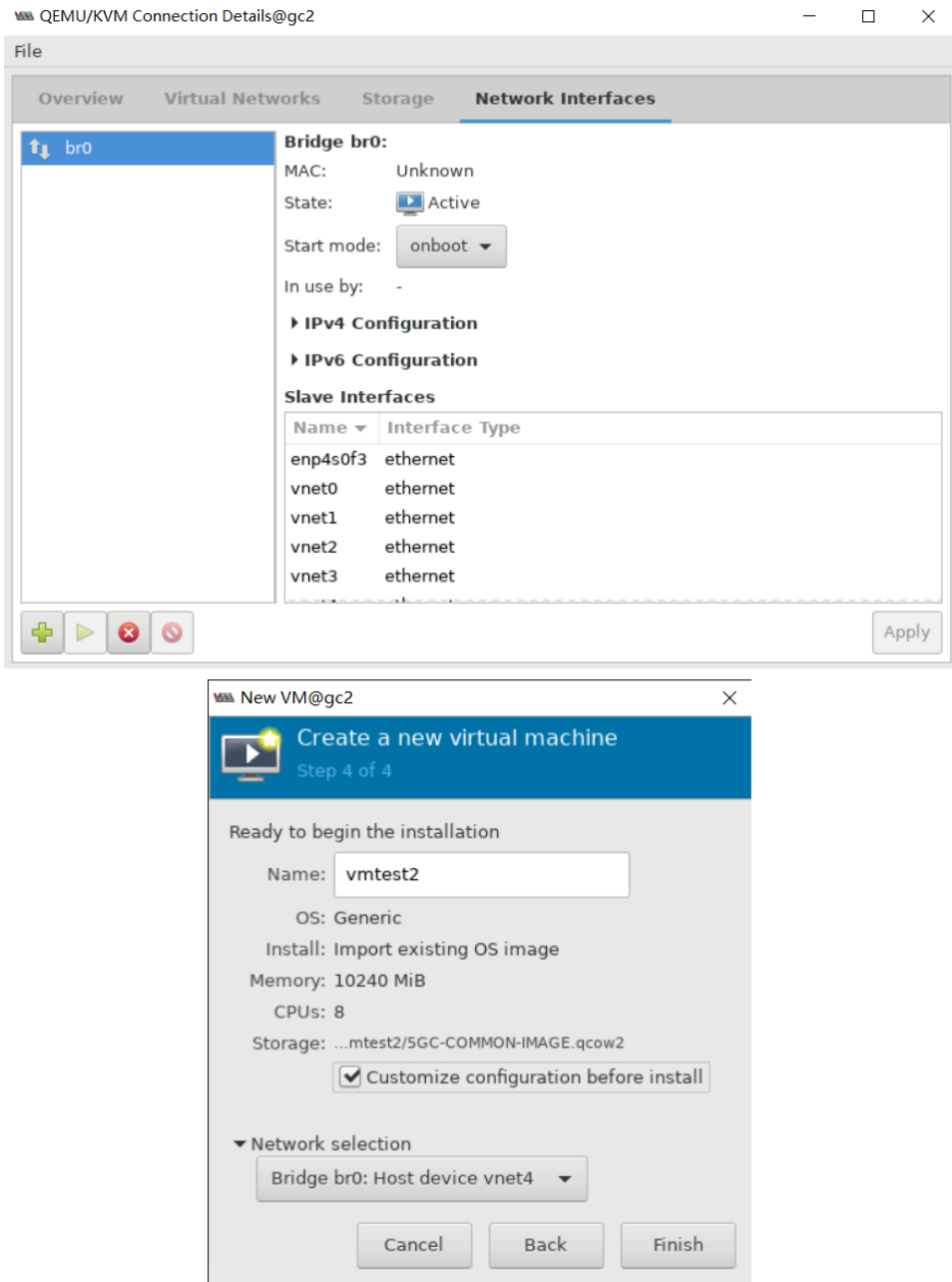


7. 设置 **虚拟机的名字**，注意 选择 **Customize configuration before install**，在安装前先进行参数的配置。在 **Network selection** 选择时有两种方法：

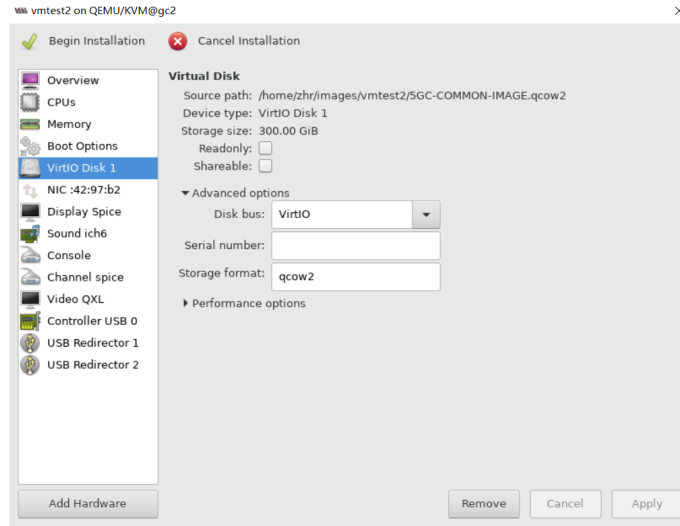
7.1. 直接指定网桥的名称



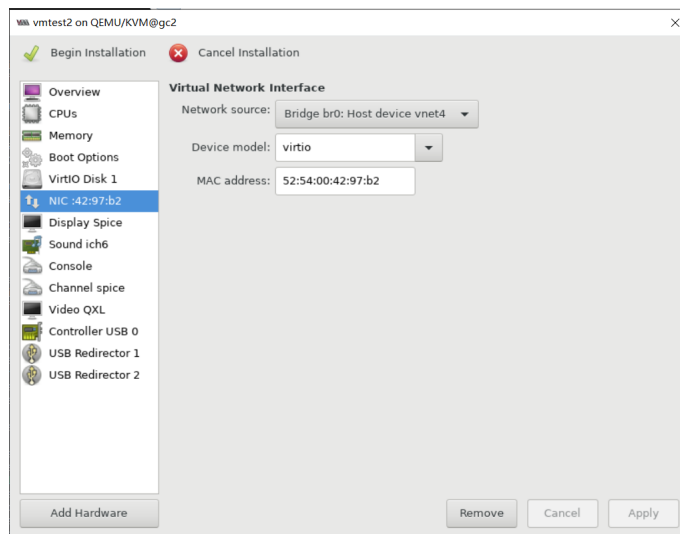
7.2. 在 virt-manager 中加入 **br0**网桥，之后直接选择 **网桥br0** 进行挂载（有时候不好使）



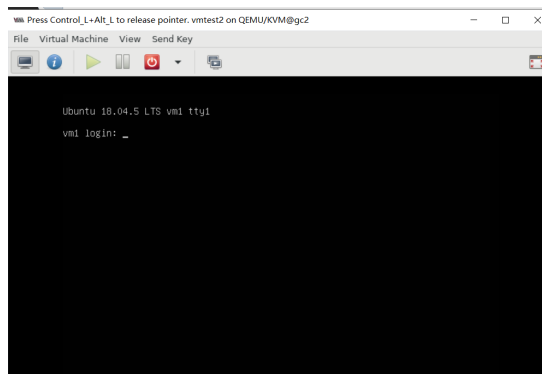
8. 配置 **Disk bus** 为 **VirtIO**



9. 选择 Network source 为 br0 , 并且 Device model 为 virtio



10. 点击左上角 begin installation ,安装完成后, 进入虚拟机界面



4.4. 配置虚拟机

4.4.1. 配置hostname和密码

1. 可以修改虚拟机的hostname, 主要修改以下两处:

```
sudo vim /etc/hostname
sudo vim /etc/hosts
```

2. 可以修改用户的密码，输入 `passwd` 即可

4.4.2. 配置虚拟机的SSH

因为 `virt-manager` 界面太古老，用起来很难受，还是建议使用ssh直接连接虚拟机进行管理

1. 基于 `5GC-COMMON-IMAGE.qcow2` 镜像建立的虚拟机已经安装ssh-server服务

```
cjn@vm1:~$ dpkg -l | grep ssh
ii  openssh-client                    1:7.6p1-4ubuntu0.3
ii  openssh-server                    1:7.6p1-4ubuntu0.3
ii  openssh-sftp-server               1:7.6p1-4ubuntu0.3
ii  ssh-import-id                     5.7-0ubuntu1.1
```

2. 修改 `/etc/ssh/sshd_config` 配置中 端口号

```
# Example of overriding settings of a specific user group
#Match User anoncvs
#    X11Forwarding no
#    AllowTcpForwarding no
#    PermitTTY no
#    ForceCommand cvs server
PasswordAuthentication yes
Port 246
```

3. 重启ssh服务，完成ssh配置，可以使用ssh直接登陆虚拟机

```
sudo /etc/init.d/ssh stop
sudo /etc/init.d/ssh start
```

```
cjn@vm1:~$ sudo /etc/init.d/ssh stop
[sudo] password for cjn:
[ ok ] Stopping ssh (via systemctl): ssh.service.
cjn@vm1:~$ sudo /etc/init.d/ssh start
[ ok ] Starting ssh (via systemctl): ssh.service.
cjn@vm1:~$
```

5. Free5GC-compose部署方法

GitHub项目地址: <https://github.com/free5gc/free5gc-compose>

建议先使用docker-compose部署5GC环境，后续多虚拟机部署时可以参考这其中的很多思路

利用 SSH 登入基于 `5GC-COMMON-IMAGE.qcow2` 镜像部署的虚拟机，进行 `Free5GC-compose` 的配置

5.1. 5GC-COMMON-IMAGE.qcow2 镜像内容

`5GC-COMMON-IMAGE.qcow2` 镜像当中配置了很多5GC所需的环境，在此进行探索与记录

1. 已经 `git clone` 了GitHub上的相关文件（注意：由于这里面一些项目版本过旧，需要删除重新下载，尤其是 `free5gc-compose`）

```
cjn@vm1:~$ ls
free5gc  free5gc-compose  go  gtp5g
cjn@vm1:~$
```

2. Free5GC要求内核版本为 `5.0.0-23-generic` 或 `5.4.0`及之后的版本，镜像中已经安装了 `5.0.0-23-generic` 版本

```
cjn@vm1:~$ uname -a
Linux vm1 5.0.0-23-generic #24~18.04.1-Ubuntu SMP Mon Jul 29 16:12:28 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
cjn@vm1:~$
```

3. 已经安装 `GO语言`

```
cjn@vm1:~$ go version
go version go1.14.4 linux/amd64
```

4. 已经在 `.bashrc` 中配置好了 `GO语言` 环境变量

```
export GOPATH=$HOME/go
export GOROOT=/usr/local/go
export PATH=$PATH:$GOPATH/bin:$GOROOT/bin
```

5. 已经安装了很多所需要的组件，但是有个 **大坑**：`cmake` 虽然安装，但是版本需要修改，在后续步骤中会提到

5.2. Free5gc-compose具体部署的步骤

1. 先连接校园网，具体指令为：

```
curl 'http://校园网地址/login' --data 'user=学号&pass=校园网登陆密码'
```

2. 为了避免github不稳定的问题，替换github网站为国内镜像源，执行以下命令：

```
git config --global url."https://hub.fastgit.org".insteadOf https://github.com
```

3. 配置GO语言模块下载代理，默认的下载地址由于国内网络原因，并不稳定，具体代码为：

```
export GOPROXY=https://goproxy.io
export GO111MODULE=on
go env -w GOPROXY=https://goproxy.io
```

4. 更换docker源，默认源下载速度过慢，会导致之后部署时出现下载超时错误。所以修改相关的配置文件，并且重启docker完成修改，具体代码为：

```
sudo vim /etc/docker/daemon.json
{"registry-mirrors": ["https://docker.mirrors.ustc.edu.cn"]}
systemctl restart docker
```

5. 补充安装依赖:

```
sudo apt update
sudo apt install make
```

6. 大坑：需要将当前普通用户加入docker用户组，不然在后续对free5gc-compose进行make的时候，会出现 `permission denied` 的错误，具体代码为：

```
sudo gpasswd -a $USER docker    #将当前普通用户加入docker用户组
newgrp docker                  #更新docker用户组
```

7. 大坑：cmake需要更新版本，否则后续UERANSIM环境搭建会报错 无法找到makefile，首先 删掉旧的cmake，具体命令为

```
sudo apt remove cmake
```

注意：如果过程中出现类似于 `dpkg: error: dpkg frontend is locked by another process` 的错误，可以参考 <https://blog.csdn.net/shimadear/article/details/90598646>，如果这个里面所有的都不作用，直接使用 `reboot` 大法

8. 然后 安装指定版本的新的cmake，具体命令为

```
wget https://cmake.org/files/v3.20/cmake-3.20.0-rc3-linux-x86_64.tar.gz
tar zxvf cmake-3.20.0-rc3-linux-x86_64.tar.gz
sudo mv cmake-3.20.0-rc3-linux-x86_64 /opt/cmake-3.20.0
sudo ln -sf /opt/cmake-3.20.0/bin/* /usr/bin/
```

9. 安装yarn，具体命令（三行命令）为

```
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee
/etc/apt/sources.list.d/yarn.list
sudo apt install yarn
```

10. 构建GTP5G模块，首先，删除 原来镜像环境中的 GTP5G文件，接着通过git clone下载项目代码：

```
cd ~
git clone https://github.com/free5gc/gtp5g.git
```

编译gtp5g代码:

```
cd gtp5g
sudo make
sudo make install
```

11. 首先, 删除 原来镜像环境中的 `docker-compose`文件, 下载 `free5gc-compose` 项目:

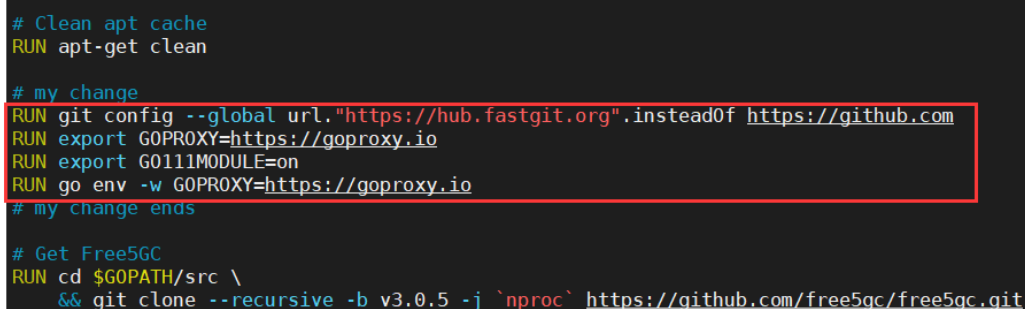
```
cd ~
git clone https://github.com/free5gc/free5gc-compose.git
```

12. 大坑: 由于国内网络问题, 在执行make base时, 会导致脚本中git clone还有go get安装模块时出现超时错误, 所以这里需要修改make base时执行的脚本, 添加设置代理步骤:

```
cd ~/free5gc-compose
cd base
vim Dockerfile
```

找到 **# Get Free5GC**一行, 在此行之前添加以下四行代码:

```
RUN git config --global url."https://hub.fastgit.org".insteadOf https://github.com
RUN export GOPROXY=https://goproxy.io
RUN export GO111MODULE=on
RUN go env -w GOPROXY=https://goproxy.io
```



```
# Clean apt cache
RUN apt-get clean

# my change
RUN git config --global url."https://hub.fastgit.org".insteadOf https://github.com
RUN export GOPROXY=https://goproxy.io
RUN export GO111MODULE=on
RUN go env -w GOPROXY=https://goproxy.io
# my change ends

# Get Free5GC
RUN cd $GOPATH/src \
    && git clone --recursive -b v3.0.5 -j `nproc` https://github.com/free5gc/free5gc.git
```

13. 大大大坑: 由于 `iptables` 的设置, 直接配置的Free5gc项目启动后, UE会无法连接到互联网, 所以需要进行以下配置:

① 在free5gc-compose项目的 `config`文件夹 下面新建一个shell脚本 `upf-iptables.sh`, 用于修改防火墙设置, 这个脚本的内容为:

```
#!/bin/sh

### UPF IPtables forwarding rules configuration

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -I FORWARD 1 -j ACCEPT
```

② 修改 `docker-compose.yaml` 中 `upf1` 代码如下:

```
services:
  free5gc-upf-1:
    container_name: upf1
    build:
      context: ./nf_upf
      args:
        DEBUG_TOOLS: "false"
    command: sh -c "chmod +x upf-iptables.sh && ./upf-iptables.sh && ./free5gc-upfd -f ../config/upfcfg.yaml"
    volumes:
      - ./config/upfcfg1.yaml:/free5gc/config/upfcfg.yaml
      - ./config/upf-iptables.sh:/free5gc/free5gc-upfd/upf-iptables.sh
    cap_add:
      - NET_ADMIN
    networks:
      privnet:
        aliases:
          - upf1.free5gc.org
```

③ 修改 `docker-compose.yaml` 中 `upf2` 代码如下:

```
services:
  free5gc-upf-2:
    container_name: upf2
    build:
      context: ./nf_upf
      args:
        DEBUG_TOOLS: "false"
    command: sh -c "chmod +x upf-iptables.sh && ./upf-iptables.sh && ./free5gc-upfd -f ../config/upfcfg.yaml"
    volumes:
      - ./config/upfcfg2.yaml:/free5gc/config/upfcfg.yaml
      - ./config/upf-iptables.sh:/free5gc/free5gc-upfd/upf-iptables.sh
    cap_add:
      - NET_ADMIN
    networks:
      privnet:
```

```
aliases:
  - upf1.free5gc.org
```

④ 注意：通过将 iptables 设置写入sh脚本并在 build 的时候自动运行，就不需要每次重启容器都自己手动输入iptables规则了，更为方便。否则，如果选择自己手动输入第①点中的iptables指令，则每次重启容器，都需要输入。

14. 大大大坑：由于第13点中修改了 upf1和upf2 的配置文件，需要使用 iptables，所以需要 在利用cmake进行容器构建时即安装iptables，顺便安装 ping 组件，便于后续测试容器联网。

① 利用 cd free5gc-compose/nf_upf/ 进入 upf容器 的配置文件夹，利用 vim Dockerfile 编辑配置文件。

② 修改 # Install UPF dependencies 部分如下图所示。

```
# Install UPF dependencies
RUN apt-get update \
    && apt-get install -y libmnl0 libyaml-0-2 iproute2 iptables inetutils-ping \
    && apt-get clean
```

15. 大大大坑：修改mongodb容器的版本为4.4.6!!!，当时我进行测试的时候，使用最新版本 latest或者5.0，都会出现 EXIT 132 的情况，紧接着由于mongodb无法运行，依赖其的nrf和weibu都会停止运行，但是官网并没有给出解决方案，build之后如下图所示

Name	Command	State	Ports
amf	./amf -amfcfg ../config/am ...	Up	8000/tcp
ausf	./ausf -ausfcfg ../config/ ...	Up	8000/tcp
mongodb	docker-entrypoint.sh --nam ...	Exit 132	
n3iwf	sh -c ./n3iwf-ipsec.sh && ...	Up	
nrf	./nrf -nrfcfg ../config/nr ...	Exit 1	
nssf	./nssf -nssfcfg ../config/ ...	Up	8000/tcp
pcf	./pcf -pcfcfg ../config/pc ...	Up	8000/tcp
smf	./smf -smfcfg ../config/sm ...	Up	8000/tcp
udm	./udm -udmcfg ../config/ud ...	Up	8000/tcp
udr	./udr -udrcfg ../config/ud ...	Up	8000/tcp
upf1	./free5gc-upfd -f ../confi ...	Up	
upf2	./free5gc-upfd -f ../confi ...	Up	
upfb	./free5gc-upfd -f ../confi ...	Up	
webui	./webui	Exit 1	

测试发现 4.4.6版本 不会出现这个问题，所以指定 mongo容器版本为4.4.6

首先 cd ~/free5gc-compose/，接着 vim docker-compose.yaml 文件，指定 mongo容器的版本

本

```
db:
  container_name: mongodb
  image: mongo:4.4.6
  command: mongod --port 27017
  expose:
    - "27017"
  volumes:
    - dbdata:/data/db
  networks:
    privnet:
      aliases:
        - db
```

16. 此外，我还修改了宿主机的 `iptables` 的设置（`sudo iptables -S`可以查看防火墙设置）：

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A POSTROUTING -o ens3 -j MASQUERADE
sudo systemctl stop ufw
sudo iptables -I FORWARD 1 -j ACCEPT
```

17. 部署 free5gc-compose 项目，输入以下指令：

```
cd ~/free5gc-compose
sudo make base
docker-compose build
```

5.3. UERANSIM模拟设备安装

1. 项目下载

```
cd ~
git clone https://github.com/aligungr/UERANSIM
```

2. 安装依赖

```
sudo apt install make
sudo apt install libsctp-dev
sudo apt install lksctp-tools
sudo apt install iproute2
```

3. 编译源代码


```
cd ~/UERANSIM
sudo make
```

5.4. Free5gc-compose完整环境运行

1. 启动free5gc环境

```
cd ~/free5gc-compose
docker-compose up -d
```

2. UERANSIM配置设置

① 查看并记录虚拟机网卡地址 (ifconfig)

```
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.112.34.229 netmask 255.255.0.0 broadcast 10.112.255.255
    inet6 fe80::5054:ff:fe10:e70e prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:10:e7:0e txqueuelen 1000 (Ethernet)
    RX packets 1198209 bytes 741985126 (741.9 MB)
    RX errors 0 dropped 22 overruns 0 frame 0
    TX packets 351094 bytes 29885831 (29.8 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

② 查看并记录amf网元的ip地址 (docker inspect amf)

```
"NetworkID": "868c8db7311219353961b7c830b4faaa47e0476c86f7bf76f73b4cf52ef1c6f9",
"EndpointID": "8386e30208b75d629bfae985d63a2d8fa7a218ed130e7d5e4d3e265bf86033cb",
"Gateway": "10.100.200.1",
"IPAddress": "10.100.200.14",
"IPPrefixLen": 24,
"IPv6Gateway": "",
"GlobalIPv6Address": "",
"GlobalIPv6PrefixLen": 0,
"MacAddress": "02:42:0a:64:c8:0e",
"DriverOpts": null
```

③ 对UERANSIM中gnb进行配置, 修改free5gc-gnb.yaml配置文件

```
cd ~/UERANSIM/config/
vim free5gc-gnb.yaml
```

注意: 需要修改其中的ngapIp、gtpIp为本机ip

注意: 修改其中的amfconfig一项下的address为amf的ip

```
mc: '208' # Mobile Country Code value
mnc: '93' # Mobile Network Code value (2 or 3 digits)
nci: '0x000000010' # NR Cell Identity (36-bit)
idLength: 32 # NR gNB ID length in bits [22...32]
tac: 1 # Tracking Area Code
linkIp: 127.0.0.1 # gNB's local IP address for Radio Link Simulation (Usually same with local IP)
ngapIp: 10.112.34.229 # gNB's local IP address for N2 Interface (Usually same with local IP)
gtpIp: 10.112.34.229 # gNB's local IP address for N3 Interface (Usually same with local IP)
# List of AMF address information
amfConfigs:
  - address: 10.100.200.14
    port: 38412
# List of supported S-NSSAIs by this gNB
slices:
  - sst: 0x1
    sd: 0x010203
# Indicates whether or not SCTP stream number errors should be ignored.
ignoreStreamIds: true
```

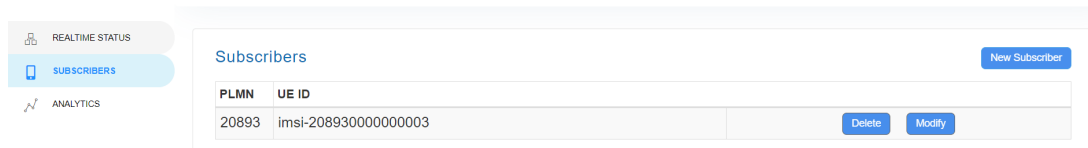
3. **大坑**：需要注意到，每次重启容器，amf容器的地址都会改变，所以重启后需要修改amfconfig一项下的address

4. 在free5gc中注册UERANSIM的UE部分 访问地址 `http://localhost:5000/` 可进入到free5gc的webui处 (`localhost`为虚拟机ip)

用户名：admin

密码：free5gc

进入网页之后，点击左边 **SUBSCRIBERS**，然后点击右上角 **NEW Subscriber**，由于我们配置的参数和网站弹出来的默认参数一致，所以 **不需要修改任何东西**，直接点击 **SUBMIT** 即可。之后界面如下方所示



5. 运行UE进行注册

利用SSH **新建** 一个 **窗口1**，输入以下指令：

```
cd ~/UERANSIM/build
./nr-gnb -c ../config/free5gc-gnb.yaml
```

利用SSH **新建** 一个 **窗口2**，输入以下指令：

```
cd ~/UERANSIM/build
sudo ./nr-ue -c ../config/free5gc-ue.yaml
```

5.5. 运行测试结果

将原来的窗口视为窗口0，在上一步新建了窗口1和窗口2，下面观察运行结果

1. 在窗口0输入 `ifconfig`，可以看到容器虚拟出很多虚拟网卡

```

veth6025781: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::2cc2:9fff:fe4f:8fcf prefixlen 64 scopeid 0x20<link>
    ether 2e:c2:9f:4f:8f:cf txqueuelen 0 (Ethernet)
    RX packets 3 bytes 175 (175.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 49 bytes 2624 (2.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

veth05861af: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::703e:94ff:fe95:affb prefixlen 64 scopeid 0x20<link>
    ether 72:3e:94:95:af:fb txqueuelen 0 (Ethernet)
    RX packets 733 bytes 66977 (66.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 617 bytes 144104 (144.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

veth11b0337: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet6 fe80::6cf5:d6ff:fe56:2346 prefixlen 64 scopeid 0x20<link>
    ether 6e:f5:d6:56:23:46 txqueuelen 0 (Ethernet)
    RX packets 21 bytes 1975 (1.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 67 bytes 4922 (4.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

2. 在 `free5gc-compose` 文件夹下输入 `docker-compose ps` 可以观察到容器的运行情况

```

cjin@vmdocker:~/free5gc-compose$ docker-compose ps

```

Name	Command	State	Ports
amf	./amf -amfcfg ../config/am ...	Up	8000/tcp
ausf	./ausf -ausfcfg ../config/ ...	Up	8000/tcp
mongodb	docker-entrypoint.sh mongo ...	Up	27017/tcp
n3iwf	sh -c ./n3iwf-ipsec.sh && ...	Up	
nrf	./nrf -nrfcfg ../config/nr ...	Up	8000/tcp
nssf	./nssf -nssfcfg ../config/ ...	Up	8000/tcp
pcf	./pcf -pcfcfg ../config/pc ...	Up	8000/tcp
smf	./smf -smfcfg ../config/sm ...	Up	8000/tcp
udm	./udm -udmcfg ../config/ud ...	Up	8000/tcp
udr	./udr -udrcfg ../config/ud ...	Up	8000/tcp
upf1	sh -c chmod +x upf-iptables ...	Up	
upf2	sh -c chmod +x upf-iptables ...	Up	
upfb	./free5gc-upfd -f ../confi ...	Up	
webui	./webui	Up	0.0.0.0:5000->5000/tcp

3. 输入 `docker exec -it upf1 /bin/bash` 进入 `upf1` 容器，测试发现可以成功联网，`upf2` 容器同理

```

cjin@vmdocker:~$ docker exec -it upf1 /bin/bash
root@95cac9bffe6:/free5gc/free5gc-upfd# ping www.baidu.com
PING www.a.shifen.com (39.156.66.18): 56 data bytes
64 bytes from 39.156.66.18: icmp_seq=0 ttl=47 time=5.317 ms
64 bytes from 39.156.66.18: icmp_seq=1 ttl=47 time=5.368 ms
64 bytes from 39.156.66.18: icmp_seq=2 ttl=47 time=5.195 ms
64 bytes from 39.156.66.18: icmp_seq=3 ttl=47 time=5.614 ms
^C--- www.a.shifen.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 5.195/5.373/5.614/0.152 ms
root@95cac9bffe6:/free5gc/free5gc-upfd# exit
exit

```

4. 在窗口1输入指令之后，如下图所示

```

cjin@vmdocker:~$ cd ~/UERANSIM/build
cjin@vmdocker:~/UERANSIM/build$ ./nr-gnb -c ../config/free5gc-gnb.yaml
UERANSIM v3.2.2
[2021-07-18 10:12:19.832] [sctp] [info] Trying to establish SCTP connection... (10.100.200.14:38412)
[2021-07-18 10:12:19.840] [sctp] [info] SCTP connection established (10.100.200.14:38412)
[2021-07-18 10:12:19.840] [sctp] [debug] SCTP association setup ascid[17]
[2021-07-18 10:12:19.841] [ngap] [debug] Sending NG Setup Request
[2021-07-18 10:12:19.853] [ngap] [debug] NG Setup Response received
[2021-07-18 10:12:19.853] [ngap] [info] NG Setup procedure is successful
[2021-07-18 10:12:30.855] [rrc] [debug] UE[1] new signal detected
[2021-07-18 10:12:30.861] [rrc] [info] RRC Setup for UE[1]
[2021-07-18 10:12:30.864] [ngap] [debug] Initial NAS message received from UE[1]
[2021-07-18 10:12:31.268] [ngap] [debug] Initial Context Setup Request received
[2021-07-18 10:12:31.701] [ngap] [info] PDU session resource(s) setup for UE[1] count[1]

```

5. 在窗口2输入指令之后，如下图所示

```

cjin@vmdocker:~$ cd ~/UERANSIM/build
cjin@vmdocker:~/UERANSIM/build$ sudo ./nr-ue -c ../config/free5gc-ue.yaml
[sudo] password for cjin:
UERANSIM v3.2.2
[2021-07-18 10:12:30.854] [nas] [info] UE switches to state [MM-DEREGISTERED/PLMN-SEARCH]
[2021-07-18 10:12:30.856] [rrc] [debug] New signal detected for cell[1], total [1] cells in coverage
[2021-07-18 10:12:30.857] [nas] [info] Selected plmn[208/03]
[2021-07-18 10:12:30.857] [rrc] [info] Selected cell plmn[208/93] tac[1] category[SUITABLE]
[2021-07-18 10:12:30.857] [nas] [info] UE switches to state [MM-DEREGISTERED/PS]
[2021-07-18 10:12:30.858] [nas] [info] UE switches to state [MM-DEREGISTERED/NORMAL-SERVICE]
[2021-07-18 10:12:30.858] [nas] [debug] Initial registration required due to [MM-DEREG-NORMAL-SERVICE]
[2021-07-18 10:12:30.858] [nas] [debug] UAC access attempt is allowed for identity[0], category[M0_sig]
[2021-07-18 10:12:30.859] [nas] [debug] Sending Initial Registration
[2021-07-18 10:12:30.860] [nas] [info] UE switches to state [MM-REGISTER-INITIATED]
[2021-07-18 10:12:30.860] [rrc] [debug] Sending RRC Setup Request
[2021-07-18 10:12:30.862] [rrc] [info] RRC connection established
[2021-07-18 10:12:30.862] [rrc] [info] UE switches to state [RRC-CONNECTED]
[2021-07-18 10:12:30.863] [nas] [info] UE switches to state [CM-CONNECTED]
[2021-07-18 10:12:31.043] [nas] [debug] Authentication Request received
[2021-07-18 10:12:31.083] [nas] [debug] Security Mode Command received
[2021-07-18 10:12:31.084] [nas] [debug] Selected integrity[2] ciphering[0]
[2021-07-18 10:12:31.269] [nas] [debug] Registration accept received
[2021-07-18 10:12:31.270] [nas] [info] UE switches to state [MM-REGISTERED/NORMAL-SERVICE]
[2021-07-18 10:12:31.270] [nas] [debug] Sending Registration Complete
[2021-07-18 10:12:31.270] [nas] [info] Initial Registration is successful
[2021-07-18 10:12:31.270] [nas] [debug] Sending PDU Session Establishment Request
[2021-07-18 10:12:31.270] [nas] [debug] UAC access attempt is allowed for identity[0], category[M0_sig]
[2021-07-18 10:12:31.703] [nas] [debug] PDU Session Establishment Accept received
[2021-07-18 10:12:31.703] [nas] [warning] SM cause received in PduSessionEstablishmentAccept [PDU_SESSION_TYPE_IPV4_ONLY_ALLOWED]
[2021-07-18 10:12:31.704] [nas] [info] PDU Session establishment is successful PSI[1]
[2021-07-18 10:12:31.746] [app] [info] Connection setup for PDU session[1] is successful, TUN interface[uesimtun0, 60.60.0.2] is up.

```

6. 在窗口0输入指令 `ifconfig`，发现成功虚拟出 UE 的虚拟网卡 `uesimtun0`。由于我之前已经测试过，所以这次的 `uesimtun0` 的 ip 是 60.60.0.2，第一次进行测试应该为 60.60.0.1，只要位于 60.60.0.0/16 均正常

```

uesimtun0: flags=369<UP,POINTOPOINT,NOTRAILERS,RUNNING,PROMISC> mtu 1400
inet 60.60.0.2 netmask 255.255.255.255 destination 60.60.0.2
inet6 fe80::9b4a:c85c:5444:c90f prefixlen 64 scopeid 0x20<link>
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 9 bytes 544 (544.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

7. 测试能否通过 UE 正常 5G 上网，输入指令 `ping www.baidu.com -I uesimtun0`

```

cjin@vmdocker:~$ ping www.baidu.com -I uesimtun0
PING www.a.shifen.com (39.156.66.14) from 60.60.0.2 uesimtun0: 56(84) bytes of data.
64 bytes from 39.156.66.14 (39.156.66.14): icmp_seq=1 ttl=45 time=11.6 ms
64 bytes from 39.156.66.14 (39.156.66.14): icmp_seq=2 ttl=45 time=10.9 ms
64 bytes from 39.156.66.14 (39.156.66.14): icmp_seq=3 ttl=45 time=11.1 ms
64 bytes from 39.156.66.14 (39.156.66.14): icmp_seq=4 ttl=45 time=12.3 ms
64 bytes from 39.156.66.14 (39.156.66.14): icmp_seq=5 ttl=45 time=9.97 ms
64 bytes from 39.156.66.14 (39.156.66.14): icmp_seq=6 ttl=45 time=11.3 ms
^C
--- www.a.shifen.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5009ms
rtt min/avg/max/mdev = 9.975/11.250/12.367/0.724 ms
cjin@vmdocker:~$

```

8. 至此，证明 Free5GC-compose 搭建成功！

6. Free5GC 多虚拟机部署方法

GitHub项目地址: <https://github.com/free5gc/free5gc>

利用 SSH 登入基于 5GC-COMMON-IMAGE.qcow2 镜像部署的多台虚拟机，进行多虚拟部署Free5GC 的配置

使用 镜像中的free5gc版本 进行 多虚拟机部署 , 需要 注意 的是, 使用的版本为 Free5GC-v3.0.4

6.1. 在vm1中配置基本环境

1. 先连接校园网, 具体指令为:

```
curl 'http://校园网/login' --data 'user=学号&pass=校园网登陆密码'
```

2. 为了避免github不稳定的问题, 替换github网站为国内镜像源, 执行以下命令:

```
git config --global url."https://hub.fastgit.org".insteadOf https://github.com
```

3. 配置GO语言模块下载代理, 默认的下载地址由于国内网络原因, 并不稳定, 具体代码为:

```
export GOPROXY=https://goproxy.io
export GO111MODULE=on
go env -w GOPROXY=https://goproxy.io
```

4. 更换docker源, 默认源下载速度过慢, 会导致之后部署时出现下载超时错误。所以修改相关的配置文件, 并且重启docker完成修改, 具体代码为:

```
sudo vim /etc/docker/daemon.json
{"registry-mirrors": ["https://docker.mirrors.ustc.edu.cn"]}
systemctl restart docker
```

5. 补充安装依赖:

```
sudo apt update
sudo apt install make
sudo apt -y install git gcc g++ cmake autoconf libtool pkg-config libmnl-dev
libyaml-dev
sudo apt install libsctp-dev lksctp-tools
sudo apt install iproute2
go get -u github.com/sirupsen/logrus
```

6. 大坑: 需要将当前普通用户加入docker用户组, 不然在后续对free5gc-compose进行make的时候, 会出现 `permission denied` 的错误, 具体代码为:

```
sudo gpasswd -a $USER docker #将当前普通用户加入docker用户组
newgrp docker #更新docker用户组
```

7. 大坑：cmake需要更新版本，否则后续UERANSIM环境搭建会报错 无法找到makefile，首先删掉旧的cmake，具体命令为

```
sudo apt remove cmake
```

如果过程中出现类似于 dpkg: error: dpkg frontend is locked by another process 的错误，可以参考 <https://blog.csdn.net/shimadear/article/details/90598646>，如果这个里面所有的都不作用，直接使用 reboot 大法

8. 然后 安装指定版本的新的cmake，具体命令为

```
wget https://cmake.org/files/v3.20/cmake-3.20.0-rc3-linux-x86_64.tar.gz
tar zxvf cmake-3.20.0-rc3-linux-x86_64.tar.gz
sudo mv cmake-3.20.0-rc3-linux-x86_64 /opt/cmake-3.20.0
sudo ln -sf /opt/cmake-3.20.0/bin/* /usr/bin/
```

9. 安装yarn，具体命令（三行）为

```
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee
/etc/apt/sources.list.d/yarn.list
sudo apt install yarn
```

10. 构建GTP5G模块，编译gtp5g代码：

```
cd gtp5g
sudo make
sudo make install
```

11. 更改防火墙设置（sudo iptables -S可以查看防火墙设置）：

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A POSTROUTING -o ens3 -j MASQUERADE
sudo systemctl stop ufw
sudo iptables -I FORWARD 1 -j ACCEPT
```

12. 安装GO相关依赖

```
cd ~/free5gc
go mod download
```

6.2. 利用vm1修改后的镜像创建vm2-vm6

1. 在第二台服务器上利用 KVM 基于 vm1 修改后的镜像建立虚拟机 vm2-vm6

2. 修改主机名，主要修改以下两处

```
sudo vim /etc/hostname
sudo vim /etc/hosts
```

3. 多虚拟机部署内容

```
vm1 中部署 gnb [即为 (R)AN]
vm2 中部署 mongodb、nrf、udr、udm、ausf、nssf、pcf、webconsole
vm3 中部署 amf
vm4 中部署 smf
vm5 中部署 upf
vm6 中部署 UE
```

4. 由于不需要使用 N3IWF 模块，故未进行部署

6.3. 配置并编译5GC各个模块

6.3.1. vm1虚拟机

1. 联网后 下载 UERANSIM 模拟设备项目

```
cd ~
git clone https://github.com/aligungr/UERANSIM
```

2. 编译源代码

```
cd ~/UERANSIM
sudo make
```

3. 对UERANSIM中 gnb 进行配置，修改free5gc-gnb.yaml配置文件

```
cd ~/UERANSIM/config/
vim free5gc-gnb.yaml
```

注意：需要修改其中的 linkIP、ngapIp、gtpIp 为 gnb所在虚拟机的ip

注意：修改其中的 amfconfig 一项下的 address 为 安装AMF模块的虚拟机的IP

```

UERANSIM-master > config > ! free5gc-gnb.yaml
1  mcc: '208'           # Mobile Country Code value
2  mnc: '93'           # Mobile Network Code value (2 or 3 digits)
3
4  nci: '0x000000010'  # NR Cell Identity (36-bit)
5  idLength: 32        # NR gNB ID length in bits [22...32]
6  tac: 1              # Tracking Area Code
7
8  linkIp: 10.112.62.28 # 此处的IP为 gNB 所在虚拟机的IP (即本机的IP)
9  ngapIp: 10.112.62.28 # 此处的IP为 gNB 所在虚拟机的IP (即本机的IP)
10 gtpIp: 10.112.62.28 # 此处的IP为 gNB 所在虚拟机的IP (即本机的IP)
11 # List of AMF address information
12 amfConfigs:
13 | - address: 10.112.162.222 # 此处的IP为 AMF 所在虚拟机的IP
14 |   port: 38412
15
16 # List of supported S-NSSAIs by this gNB
17 slices:
18 | - sst: 0x1
19 |   sd: 0x010203
20
21 # Indicates whether or not SCTP stream number errors should be ignored.
22 ignoreStreamIds: true
23

```

6.3.2. vm2虚拟机

1. 安装 `mongodb`

先连接校园网，然后安装并启动mongodb，具体指令为

```

curl 'http://校园网地址/login' --data 'user=学号&pass=校园网登陆密码'
sudo apt update
sudo apt -y install mongodb
sudo systemctl start mongodb

```

2. 配置并编译 `NRF`

```

cd ~/free5gc/config
sudo vim nrfcfg.conf

```

修改完模块的配置 之后，进行编译

```

cd ~/free5gc
sudo make nrf

```

修改文件配置如下图所示：


```
free5gc > config > nrfcfg.conf
1  info:
2    version: 1.0.0
3    description: NRF initial local configuration
4
5  configuration:
6    MongoDBName: "free5gc"
7    MongoDBUrl: "mongodb://10.112.106.106:27017" # 此处的IP为 MongoDB 所在虚拟机的IP
8    DefaultServiceIP: "10.112.106.106" # 此处的IP为 NRF 所在虚拟机的IP
9    sbi:
10     scheme: http
11     ipv4Addr: 10.112.106.106 # 此处的IP为 NRF 所在虚拟机的IP
12     port: 29510
13     DefaultPlmnId:
14     mcc: "208"
15     mnc: "93"
16     serviceNameList:
17     - nrf-nfm
18     - nrf-disc
```

3. 配置并编译 UDR

```
cd ~/free5gc/config
sudo vim udrcfg.conf
```

修改完模块的配置 之后，进行编译

```
cd ~/free5gc
sudo make udr
```

修改文件配置如下图所示：

```
free5gc > config > udrcfg.conf
1  info:
2    version: 1.0.0
3    description: UDR initial local configuration
4
5  configuration:
6    sbi:
7     scheme: http
8     registerIPv4: 10.112.106.106 # 此处的IP为 UDR 所在虚拟机的IP
9     bindingIPv4: 10.112.106.106 # 此处的IP为 UDR 所在虚拟机的IP
10    port: 29504
11    mongodb:
12     name: free5gc
13     url: mongodb://10.112.106.106:27017 # 此处的IP为 MongoDB 所在虚拟机的IP
14     nrfUri: http://10.112.106.106:29510 # 此处的IP为 NRF 所在虚拟机的IP
15
```

4. 配置并编译 UDM

```
cd ~/free5gc/config
sudo vim udmcfg.conf
```

修改完模块的配置 之后，进行编译

```
cd ~/free5gc
sudo make udm
```

修改文件配置如下图所示：

```
free5gc > config > ⚙ udmcfg.conf
1  info:
2    version: 1.0.0
3    description: UDM initial local configuration
4
5  configuration:
6    serviceNameList:
7      - nudm-sdm
8      - nudm-uecm
9      - nudm-ueau
10     - nudm-ee
11     - nudm-pp
12  sbi:
13    scheme: http
14    registerIPv4: 10.112.106.106 # 此处的IP为 UDM 所在虚拟机的IP
15    bindingIPv4: 10.112.106.106 # 此处的IP为 UDM 所在虚拟机的IP
16    port: 29503
17    tls:
18      log: free5gc/udmsslkey.log
19      pem: free5gc/support/TLS/udm.pem
20      key: free5gc/support/TLS/udm.key
21
22  udrclient:
23    scheme: http
24    ipv4Addr: 10.112.106.106 # 此处的IP为 UDR 所在虚拟机的IP
25    port: 29504
26
27  nrfclient:
28    scheme: http
29    ipv4Addr: 10.112.106.106 # 此处的IP为 NRF 所在虚拟机的IP
30    port: 29510
31    nrfUri: 10.112.106.106 # 此处的IP为 NRF 所在虚拟机的IP
32
```

5. 配置并编译 AUSF

```
cd ~/free5gc/config
sudo vim ausfcfg.conf
```

修改完模块的配置 之后，进行编译

```
cd ~/free5gc
sudo make ausf
```

修改文件配置如下图所示：

```
free5gc > config > ⚙️ ausfcfg.conf
1  info:
2    version: 1.0.0
3    description: AUSF initial local configuration
4
5  configuration:
6    sbi:
7      scheme: http
8      registerIPv4: 10.112.106.106 # 此处的IP为 AUSF 所在虚拟机的IP
9      bindingIPv4: 10.112.106.106 # 此处的IP为 AUSF 所在虚拟机的IP
10     port: 29509
11     serviceNameList:
12       - nausf-auth
13     nrfUri: http://10.112.106.106:29510 # 此处的IP为 NRF 所在虚拟机的IP
14     plmnSupportList:
15       - mcc: 208
16         mnc: 93
17       - mcc: 123
18         mnc: 45
19     groupId: ausfGroup001
20
```

6. 配置并编译 NSSF

```
cd ~/free5gc/config
sudo vim nssfcfg.conf
```

修改完模块的配置 之后，进行编译

```
cd ~/free5gc
sudo make nssf
```

修改文件配置如下图所示：

```

free5gc > config > nssfcfg.conf
9      registerIPv4: 10.112.106.106 # 此处的IP为 NRF 所在虚拟机的IP
10     bindingIPv4: 10.112.106.106 # 此处的IP为 NRF 所在虚拟机的IP
11     port: 29531
12     serviceNameList:
13     - nnssf-nselecion
14     - nnssf-nssaiavailability
15     nrfUri: http://10.112.106.106:29510 # 此处的IP为 NRF 所在虚拟机的IP
16     supportedPlmnList:
17     - mcc: 208
18       mnc: 93
19     supportedNssaiInPlmnList:
20     - plmnId:
21       mcc: 208
22       mnc: 93
23       supportedSnssaiList:
24       - sst: 1
25         sd: 010203
26       - sst: 1
27         sd: 112233
28       - sst: 1
29         sd: 3
30       - sst: 2
31         sd: 1
32       - sst: 2
33         sd: 2
34     nsiList:
35     - snssai:
36       sst: 1
37       nsiInformationList:
38       - nrfId: http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances # 此处的IP为 NRF 所在虚拟机的IP
39         nsiId: 10
40     - snssai:
41       sst: 1
42       sd: 1
43       nsiInformationList:
44       - nrfId: http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances # 此处的IP为 NRF 所在虚拟机的IP
45         nsiId: 11

```

```

free5gc > config > nssfcfg.conf
48     sd: 2
49     nsiInformationList:
50     - nrfId: http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances # 此处的IP为 NRF 所在虚拟机的IP
51       nsiId: 12
52     - nrfId: http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances # 此处的IP为 NRF 所在虚拟机的IP
53       nsiId: 12
54     - snssai:
55       sst: 1
56       sd: 3
57     nsiInformationList:
58     - nrfId: http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances # 此处的IP为 NRF 所在虚拟机的IP
59       nsiId: 13
60     - snssai:
61       sst: 2
62     nsiInformationList:
63     - nrfId: http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances # 此处的IP为 NRF 所在虚拟机的IP
64       nsiId: 20
65     - snssai:
66       sst: 2
67       sd: 1
68     nsiInformationList:
69     - nrfId: http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances # 此处的IP为 NRF 所在虚拟机的IP
70       nsiId: 21
71     - snssai:
72       sst: 1
73       sd: 010203
74     nsiInformationList:
75     - nrfId: http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances # 此处的IP为 NRF 所在虚拟机的IP
76       nsiId: 22
77     amfSetList:
78     - amfSetId: 1
79     amfList:
80     - ffa2e8d7-3275-49c7-8631-6af1df1d9d26
81     - 0e8831c3-6286-4689-ab27-1e2161e15cb1
82     - a1fba9ba-2e39-4e22-9c74-f749da571d0d
83     nrfAmfSet: http://10.112.106.106:8081/nnrf-nfm/v1/nf-instances # 此处的IP为 NRF 所在虚拟机的IP
84     supportedNssaiAvailabilityData:

```

```

- tai:
  plmnId:
    mcc: 466
    mnc: 92
    tac: 33456
  supportedSnssaiList:
    - sst: 1
      sd: 1
    - sst: 1
      sd: 2
    - sst: 2
      sd: 1
- tai:
  plmnId:
    mcc: 466
    mnc: 92
    tac: 33457
  supportedSnssaiList:
    - sst: 1
      sd: 1
    - sst: 1
      sd: 1
    - sst: 1
      sd: 2
- amfSetId: 2
nrfAmfSet: http://10.112.106.106:8084/nrf-nfm/v1/nf-instances # 此处的IP为 NRF 所在虚拟机的IP
supportedNssaiAvailabilityData:
  - tai:

```

7. 配置并编译 PCF

```

cd ~/free5gc/config
sudo vim pcfcfg.conf

```

修改完模块的配置 之后，进行编译

```

cd ~/free5gc
sudo make pcf

```

修改文件配置如下图所示：

```

free5gc-change > config > ⚙️ pcfcfg.conf
1  info:
2    version: 1.0.0
3    description: PCF initial local configuration
4
5  configuration:
6    pcfName: PCF
7    sbi:
8      scheme: http
9      registerIPv4: 10.112.106.106 # 此处的IP为 PCF 所在虚拟机的IP
10     bindingIPv4: 10.112.106.106 # 此处的IP为 PCF 所在虚拟机的IP
11     port: 29507
12   timeFormat: 2019-01-02 15:04:05
13   defaultBdtRefId: BdtPolicyId-
14   nrfUri: http://10.112.106.106:29510 # 此处的IP为 NRF 所在虚拟机的IP
15   serviceList:
16     - serviceName: npcfc-am-policy-control
17     - serviceName: npcfc-smpolicycontrol
18       suppFeat: 3fff
19     - serviceName: npcfc-bdtpolicycontrol
20     - serviceName: npcfc-policyauthorization
21       suppFeat: 3
22     - serviceName: npcfc-eventexposure
23     - serviceName: npcfc-ue-policy-control
24

```

8. 修改 webconsole 配置文件

```

cd ~/free5gc/config
sudo vim webuicfg.conf

```

修改完模块的配置 之后，进行编译

```

cd ~/free5gc
sudo make webconsole

```

修改文件配置如下图所示：

```

free5gc > config > ⚙️ webuicfg.conf
1  info:
2    version: 1.0.0
3    description: WebUI initial local configuration
4
5  configuration:
6    mongodb:
7      name: free5gc
8      url: mongodb://10.112.106.106:27017 # 此处的IP为 MongoDB 所在虚拟机的IP
9

```

6.3.3. vm3虚拟机

1. 配置并编译 AMF

```
cd ~/free5gc/config
sudo vim amfcfg.conf
```

修改完模块的配置 之后，进行编译

```
cd ~/free5gc
sudo make amf
```

修改文件配置如下图所示：

```
free5gc > config > amfcfg.conf
> configuration:
6   amfName: AMF
7   ngapIpList:
8     - 10.112.162.222 # 此处的IP为 AMF 所在虚拟机的IP
9   sbi:
10    scheme: http
11    registerIPv4: 10.112.162.222 # 此处的IP为 AMF 所在虚拟机的IP
12    bindingIPv4: 10.112.162.222 # 此处的IP为 AMF 所在虚拟机的IP
13    port: 29518
14    serviceNameList:
15      - namf-comm
16      - namf-evts
17      - namf-mt
18      - namf-loc
19      - namf-oam
20    servedGuamiList:
21      - plmnId:
22          mcc: 208
23          mnc: 93
24          amfId: cafe00
25    supportTaiList:
26      - plmnId:
27          mcc: 208
28          mnc: 93
29          tac: 1
30    plmnSupportList:
31      - plmnId:
32          mcc: 208
33          mnc: 93
34          snssaiList:
35            - sst: 1
36              sd: 010203
37            - sst: 1
38              sd: 112233
39    supportDnnList:
40      - internet
41    nrfUri: http://10.112.106.106:29510 # 此处的IP为 NRF 所在虚拟机的IP
42    security:
```

6.3.4. vm4虚拟机

1. 配置并编译 SMF

```
cd ~/free5gc/config
sudo vim smfcfg.conf
```

修改完模块的配置 之后，进行编译

```
cd ~/free5gc
sudo make smf
```

修改文件配置如下图所示：

```
free5gc > config > ⚙ smfcfg.conf
1  info:
2    version: 1.0.0
3    description: SMF initial local configuration
4
5  configuration:
6    smfName: SMF
7    sbi:
8      scheme: http
9      registerIPv4: 10.112.252.200 # 此处的IP为 SMF 所在虚拟机的IP
10     bindingIPv4: 10.112.252.200 # 此处的IP为 SMF 所在虚拟机的IP
11     port: 29502
12     tls:
13       key: free5gc/support/TLS/smf.key
14       pem: free5gc/support/TLS/smf.pem
15     serviceNameList:
16       - nsmf-pdusession
17       - nsmf-event-exposure
18       - nsmf-oam
19     snssai_info:
20       - sNssai:
21         sst: 1
22         sd: 010203
23         dnnSmfInfoList:
24           - dnn: internet
25       - sNssai:
26         sst: 1
27         sd: 112233
28         dnnSmfInfoList:
29           - dnn: internet
30     pfcf:
31       addr: 10.112.252.200 # 此处的IP为 SMF 所在虚拟机的IP
32     userplane_information:
33
```



```

free5gc > config > smfcfg.conf
24     - dnn: internet
25     - sNssai:
26         sst: 1
27         sd: 112233
28         dnnSmfInfoList:
29             - dnn: internet
30     pfcfg:
31         addr: 10.112.252.200 # 此处的IP为 SMF 所在虚拟机的IP
32     userplane_information:
33         up_nodes:
34             gNB1:
35                 type: AN
36                 an_ip: 127.0.0.100
37             UPF:
38                 type: UPF
39                 node_id: 10.112.157.84 # 此处的IP为 UPF 所在虚拟机的IP
40
41     links:
42         - A: gNB1
43           B: UPF
44     ue_subnet: 60.60.0.0/16
45     dnn:
46         internet:
47             dns:
48                 ipv4: 8.8.8.8
49                 ipv6: 2001:4860:4860::8888
50         internet2:
51             dns:
52                 ipv4: 8.8.4.4
53                 ipv6: 2001:4860:4860::8844
54     nrfUri: http://10.112.106.106:29510 # 此处的IP为 NRF 所在虚拟机的IP
55

```

6.3.5. vm5虚拟机

1. 注意：安装 UPF 模块的虚拟机 vm5 需要联网，联网指令为：

```
curl 'http://校园网地址/login' --data 'user=学号&pass=校园网登陆密码'
```

2. 注意：安装 UPF 模块的虚拟机 vm5 需要配置防火墙：

```

sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -I INPUT --source 60.60.0.0/16 -j ACCEPT
sudo iptables -t nat -A POSTROUTING -s 60.60.0.0/16 ! -o upfgtp -j MASQUERADE
sudo iptables -I FORWARD --in-interface ens3 --out-interface upfgtp -j ACCEPT
sudo iptables -I FORWARD --in-interface upfgtp --out-interface ens3 -j ACCEPT

```

3. 大坑：注意 不要使用官网给出的直接编译命令 `sudo make upf`，否则会出现以下错误：

```

[ 3%] Building C object lib/prep/test/Makefiles/testprep.dir/test.c.o
[ 4%] Building libgtp5gnl
/bin/sh: 1: go: not found
lib/utlt/CMakeFiles/utlt_logger.dir/build.make:60: recipe for target 'utlt_logger' failed
make[3]: *** [utlt_logger] Error 127
make[3]: Leaving directory '/home/cjn/free5gc/src/upf/build'
CMakeFiles/Makefile2:644: recipe for target 'lib/utlt/CMakeFiles/utlt_logger.dir/all' failed
make[2]: *** [lib/utlt/CMakeFiles/utlt_logger.dir/all] Error 2

```

```
make[2]: Leaving directory '/home/cjn/free5gc/src/upf/build'
[ 8%] Built target libgtp5gnl
make[2]: Leaving directory '/home/cjn/free5gc/src/upf/build'
Makefile:83: recipe for target 'all' failed
make[1]: *** [all] Error 2
make[1]: Leaving directory '/home/cjn/free5gc/src/upf/build'
Makefile:35: recipe for target 'upf' failed
make: *** [upf] Error 2
```

4. 使用 **自行编译** 的方式编译UPF模块，具体方式如下：

- 1) 首先进入以下文件夹：`cd ~/free5gc/src/upf`
- 2) 利用 `ls` 指令查看当前文件夹下 **是否存在build文件夹**。若 **存在**，则利用 `sudo rm -rf build` 删除该文件夹。
- 3) 接下来使用以下指令进行手动编译：

```
mkdir build
cd build
cmake ..
make -j`nproc`
```

4) 编译成功后，如下图所示

```
[ 96%] Building C object src/CMakeFiles/free5GC_UPF_main.dir/n4/n4_pfcpath.c.o
[ 97%] Building C object src/CMakeFiles/free5GC_UPF_main.dir/up/up_match.c.o
[ 98%] Building C object src/CMakeFiles/free5GC_UPF_main.dir/up/up_path.c.o
[100%] Linking C executable /home/cjn/free5gc/src/upf/build/bin/free5gc-upfd
[100%] Built target free5GC_UPF_main
cjin@vm5:~/free5gc/src/upf/build$
```

5. 配置 UPF

```
cd free5gc/src/upf/build/config/
sudo vim upfcfg.yaml
```

修改文件配置如下图所示：

```
free5gc > src > upf > build > config > ! upfcfg.yaml
1  info:
2    version: 1.0.0
3    description: UPF configuration
4
5  configuration:
6    # debugLevel: panic|fatal|error|warn|info|debug|trace
7    debugLevel: info
8    # ReportCaller: true|false
9    ReportCaller: false
10
11  pfcfp:
12    - addr: 10.112.157.84 # 此处的IP为 UPF 所在虚拟机的IP
13
14  gtpu:
15    - addr: 10.112.157.84 # 此处的IP为 UPF 所在虚拟机的IP
16    # [optional] gtpu.name
17    # - name: upf.5gc.nctu.me
18    # [optional] gtpu.ifname
19    # - ifname: gtpif
20
21  dnn_list:
22    - dnn: internet
23      cidr: 60.60.0.0/24
24    # [optional] dnn_list[*].natifname
25    # natifname: eth0
26
```

6.3.6. vm6虚拟机

1. 联网后 下载 UERANSIM 模拟设备项目

```
cd ~
git clone https://github.com/aligungr/UERANSIM
```

2. 编译源代码

```
cd ~/UERANSIM
sudo make
```

3. 对UERANSIM中 ue 进行配置，修改free5gc-ye.yaml配置文件

```
cd ~/UERANSIM/config/
vim free5gc-ye.yaml
```

配置如下图所示：

```

UERANSIM-master > config > ! free5gc-ue.yaml
1  # IMSI number of the UE. IMSI = [MCC|MNC|MSISDN] (In total 15 or 16 digits)
2  supi: 'imsi-208930000000003'
3  # Mobile Country Code value of HPLMN
4  mcc: '208'
5  # Mobile Network Code value of HPLMN (2 or 3 digits)
6  mnc: '93'
7
8  # Permanent subscription key
9  key: '8baf473f2f8fd09487cccbd7097c6862'
10 # Operator code (OP or OPC) of the UE
11 op: '8e27b6af0e692e750f32667a3b14605d'
12 # This value specifies the OP type and it can be either 'OP' or 'OPC'
13 opType: 'OPC'
14 # Authentication Management Field (AMF) value
15 amf: '8000'
16 # IMEI number of the device. It is used if no SUPI is provided
17 imei: '356938035643803'
18 # IMEISV number of the device. It is used if no SUPI and IMEI is provided
19 imeiSv: '4370816125816151'
20
21 # List of gNB IP addresses for Radio Link Simulation
22 gnbSearchList:
23   - 10.112.62.28 # 此处的IP为 gNB 所在虚拟机的IP
24
25 # UAC Access Identities Configuration
26 uacAic:
27   mps: false
28   mcs: false
29
30 # UAC Access Control Class
31 uacAcc:

```

6.4. 配置部分配置

1. 注意：在6.5中启动完网元之后，可以通过以下方式查看是否存在报错：

- 1) 启动后会在 `~/free5gc` 下出现 `log` 文件夹，可以查看 `free5gc.log` 来观察是否存在报错
- 2) 对于使用 `nohup` 启动的模块，在 `~/free5gc/bin` 路径下会生成 `nohup.out` 文件，可以观察是否报错

2. 修改mongodb的配置。由于mongodb默认对访问权限有一定限制，如果不进行修改，会在后续启动各个模块时无法连接mongodb，会出现NRF模块无法连接mongodb数据库，导致NRF无法建立，紧接着其余各个模块无法与NRF进行通信，从而出现以下报错

```

time="2021-07-22T08:25:26Z" level=fatal msg="server selection error: server selection timeout\ncurrent topology: Type: Unknown\nServers:\nAddr: 10.112.106.106:27017, Type: Unknown, State: Connected, Average RTT: 0, Last error: dial tcp 10.112.106.106:27017: connect: connection refused\n" category=MongoDB component=LIB
UDM register to NRF Error[Put "http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances/5fe5b638-1127-4bb2-9d35-d30d0db9960e": dial tcp 10.112.106.106:29510: connection refused]
NSSF register to NRF Error[Put "http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances/a6c994da-e6cc-451a-9239-487354947645": dial tcp 10.112.106.106:29510: connection refused]
AUSF register to NRF Error[Put "http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances/c9c05c14-353d-402e-a2ec-6a7fc99d5047": dial tcp 10.112.106.106:29510: connection refused]
NSSF register to NRF Error[Put "http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances/a6c994da-e6cc-451a-9239-487354947645": dial tcp 10.112.106.106:29510: connection refused]
AUSF register to NRF Error[Put "http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances/c9c05c14-353d-402e-a2ec-6a7fc99d5047": dial tcp 10.112.106.106:29510: connection refused]
NSSF register to NRF Error[Put "http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances/a6c994da-e6cc-451a-9239-487354947645": dial tcp 10.112.106.106:29510: connection refused]

2021-07-22T09:00:24Z [INFO][NRF][Management] Handle NFRegisterRequest
2021-07-22T09:00:26Z [INFO][NRF][Management] Create NF Profile
2021-07-22T09:00:34Z [INFO][NRF][Management] urilist create
2021-07-22T09:00:56Z [FATAL][LIB][MongoDB] server selection error: server selection timeout
current topology: Type: Unknown
Servers:
Addr: 10.112.106.106:27017, Type: Unknown, State: Connected, Average RTT: 0, Last error: dial tcp 10.112.106.106:27017: connect: connection refused
UDM register to NRF Error[Put "http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances/539770a6-d83b-4e3a-9fb3-190c3f073ecd": unexpected EOF]
NSSF register to NRF Error[Put "http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances/fd6e8b36-4675-493c-999b-82e98d72497a": unexpected EOF]
UDR register to NRF Error[Put "http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances/b17f5a46-e46c-4955-b789-ea88cbe5c90b": unexpected EOF]
UDR register to NRF Error[Put "http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances/b17f5a46-e46c-4955-b789-ea88cbe5c90b": dial tcp 10.112.106.106:29510: connection refused]
UDM register to NRF Error[Put "http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances/539770a6-d83b-4e3a-9fb3-190c3f073ecd": dial tcp 10.112.106.106:29510: connection refused]
NSSF register to NRF Error[Put "http://10.112.106.106:29510/nnrf-nfm/v1/nf-instances/fd6e8b36-4675-493c-999b-82e98d72497a": dial tcp 10.112.106.106:29510: connection refused]

```

所以需要修改mongodb的配置：

- 1) 进入以下文件夹 `cd /etc`

- 2) 修改 `mongodb.conf` 中 `bind_ip = 0.0.0.0` , 从而允许所有地址访问mongodb数据库
- 3) 更新mongodb配置, 并重启数据库

```
mongod --config mongodb.conf
sudo systemctl stop mongod
sudo systemctl start mongod
```

3. 安装upf模块的vm5 虚拟机需要联网并配置iptables转发设置

- 1) 注意： 安装 UPF模块 的虚拟机vm5需要联网, 联网指令为:

```
curl 'http://校园网地址/login' --data 'user=学号&pass=校园网登陆密码'
```

- 2) 注意： 安装 UPF模块 的虚拟机vm5需要 配置防火墙：

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -I INPUT --source 60.60.0.0/16 -j ACCEPT
sudo iptables -t nat -A POSTROUTING -s 60.60.0.0/16 ! -o upfgtp -j MASQUERADE
sudo iptables -I FORWARD --in-interface ens3 --out-interface upfgtp -j ACCEPT
sudo iptables -I FORWARD --in-interface upfgtp --out-interface ens3 -j ACCEPT
```

6.5. 启动各个网元模块

1. 一定注意 网元启动顺序为: `mongodb > NRF > UDR > UDM > AUSF > NSSF > AMF > PCF > UPF > SMF > webconsole > gNB > UE`

2. 后台挂起运行指令 `nohup ... &` , 将指令 放在...位置 , 则可以在后台运行该指令, 输入完成之后, 中断该次SSH连接, 指令仍在后台运行

3. 各个网元的启动方式 (不包括webconsole) 如下表格所示:

虚拟机	运行的服务	启动方式
vm1	RAN (gNB)	cd ~/UERANSIM/build sudo ./nr-gnb -c ../config/free5gc-gnb.yaml
vm2	nrf udr udm ausf nssf pcf	cd ~/free5gc/bin nohup ./... & (...为网元名称)
vm3	amf	cd ~/free5gc/bin ./amf
vm4	smf	cd ~/free5gc/bin ./smf
vm5	upf	cd ~/free5gc/src/upf/build sudo ./bin/free5gc-upfd -f ../config/upfcfg.yaml

虚拟机	运行的服务	启动方式
vm6	UE	<pre>d ~/UERANSIM/build sudo ./nr-ue -c ../config/free5gc-ue.yaml</pre>

4. vm2中的 webconsole 的启动方式为

```
cd ~/free5gc/webconsole
go run server.go
```

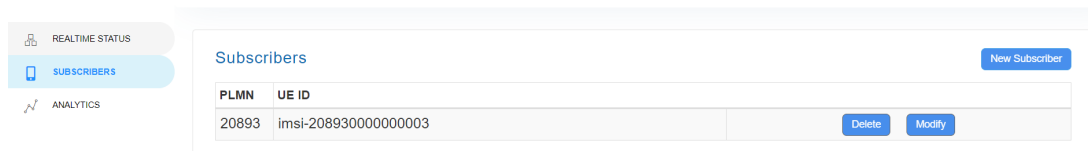
接着在free5gc中注册UERANSIM的UE部分

访问地址 <http://localhost:5000/> 可进入到free5gc的webui处 ([localhost为虚拟机ip](#))

用户名: [admin](#)

密码: [free5gc](#)

进入网页之后, 点击左边 [SUBSCRIBERS](#), 然后点击右上角 [NEW Subscriber](#), 由于我们配置的参数和网站弹出来的默认参数一致, 所以 [不需要修改任何东西](#), 直接点击 [SUBMIT](#) 即可。之后界面如下方所示



5. 完整的启动流程

启动顺序: [mongodb](#) > [NRF](#) > [UDR](#) > [UDM](#) > [AUSF](#) > [NSSF](#) > [AMF](#) > [PCF](#) > [UPF](#) > [SMF](#) > [webconsole](#) > [gNB](#) > [UE](#)

1) [mongodb](#)已经默认启动, 可以输入 [mongo](#) 查看

2) SSH进入vm2, 利用 [nohup ./nrf &](#) 启动nrf网元。关闭该窗口, 新建ssh窗口进入vm2, 查看 [~/free5gc/log/free5gc.log](#) 是否存在 [error](#)

3) SSH进入vm2, 利用 [nohup ./udr &](#) 启动udr网元。关闭该窗口, 新建ssh窗口进入vm2, 查看 [~/free5gc/log/free5gc.log](#) 是否存在 [error](#)

4) SSH进入vm2, 利用 [nohup ./udm &](#) 启动udm网元。关闭该窗口, 新建ssh窗口进入vm2, 查看 [~/free5gc/log/free5gc.log](#) 是否存在 [error](#)

5) SSH进入vm2, 利用 [nohup ./ausf &](#) 启动ausf网元。关闭该窗口, 新建ssh窗口进入vm2, 查看 [~/free5gc/log/free5gc.log](#) 是否存在 [error](#)

6) SSH进入vm2, 利用 [nohup ./nssf &](#) 启动nssf网元。关闭该窗口, 新建ssh窗口进入vm2, 查看 [~/free5gc/log/free5gc.log](#) 是否存在 [error](#)

7) SSH进入vm3, 利用 [./amf](#) 启动amf网元, [不关闭该窗口](#), 观察是否存在 [error](#)

8) SSH进入vm2, 利用 [nohup ./pcf &](#) 启动pcf网元。关闭该窗口, 新建ssh窗口进入vm2, 查看 [~/free5gc/log/free5gc.log](#) 是否存在 [error](#)

9) SSH进入vm5, 利用 [表格中的指令](#) 启动upf网元, [不关闭该窗口](#), 观察是否存在 [error](#)

10) SSH进入vm4, 利用 [./smf](#) 启动smf网元, [不关闭该窗口](#), 观察是否存在 [error](#)

11) SSH进入vm2, 启动 [webconsole](#) 并 [完成网页端注册](#), [不关闭该窗口](#), 观察是否存在 [error](#)

12) SSH进入vm1, 启动 [gNB](#) 服务, [不关闭该窗口](#), 观察是否存在 [error](#)

13) SSH进入vm6, 启动 [UE](#) 服务, [不关闭该窗口](#), 观察是否存在 [error](#)

6. 启动完成所有网元后，新建窗口SSH进入vm6，输入 `ifconfig` 可以查看到 属于UE的网卡 `uesimtun0`，然后测试UE可以成功上网，如下图所示

```
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.112.78.164 netmask 255.255.0.0 broadcast 10.112.255.255
    inet6 fe80::5054:ff:fe29:214f prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:29:21:4f txqueuelen 1000 (Ethernet)
    RX packets 1386301 bytes 184495521 (184.4 MB)
    RX errors 0 dropped 7 overruns 0 frame 0
    TX packets 13495 bytes 1337227 (1.3 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 768 bytes 66924 (66.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 768 bytes 66924 (66.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

uesimtun0: flags=369<UP,POINTOPOINT,NOTRAILERS,RUNNING,PROMISC> mtu 1400
    inet 60.60.0.3 netmask 255.255.255.255 destination 60.60.0.3
    inet6 fe80::84b6:e80c:1e42:6ffa prefixlen 64 scopeid 0x20<link>
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 7 bytes 448 (448.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

cjn@vm6:~$ ping www.baidu.com -I uesimtun0
PING www.a.shifen.com (39.156.66.14) from 60.60.0.3 uesimtun0: 56(84) bytes of data.
64 bytes from 39.156.66.14 (39.156.66.14): icmp_seq=1 ttl=46 time=10.5 ms
64 bytes from 39.156.66.14 (39.156.66.14): icmp_seq=2 ttl=46 time=8.70 ms
64 bytes from 39.156.66.14 (39.156.66.14): icmp_seq=3 ttl=46 time=8.17 ms
64 bytes from 39.156.66.14 (39.156.66.14): icmp_seq=4 ttl=46 time=8.49 ms
64 bytes from 39.156.66.14 (39.156.66.14): icmp_seq=5 ttl=46 time=8.19 ms
^C
--- www.a.shifen.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 8.171/8.826/10.574/0.903 ms
cjn@vm6:~$
```

6.6. 问题与解决方案

1. 查看 `~/free5gc/log/free5gc.log` 和 `~/free5gc/bin/nohup.out` 文件观察究竟出现什么错误。如果为 NRF无法连接 则参考 6.4mongodb配置方法；如果 UE和PDU成功建立，但UE无法联网，查看UPF虚拟机 是否联网以及iptables是否配置

2. 如果在启动模块的过程中报错 某网元的端口以及被占用，如下图所示

```
time="2021-07-22T10:16:00Z" level=info msg="| 201 | 10.112.106.106 | PUT | /nnrf-nfm/v1/nf-instances/c359180f-c03d-4acd-940a-14434e417102 | " category=GIN component=NRF
time="2021-07-22T10:16:00Z" level=fatal msg="HTTP server setup failed: listen tcp 10.112.106.106:29509: bind: address already in use" category=Init component=AUSF

time="2021-07-22T10:16:57Z" level=info msg="| 200 | 10.112.106.106 | GET | /nnrf-disc/v1/nf-instances?requester-nf-type=PCF&service-names=nudr-dr&target-nf-type=UDR | " category=GIN component=NRF
time="2021-07-22T10:16:57Z" level=fatal msg="HTTP server setup failed: listen tcp 10.112.106.106:29507: bind: address already in use" category=Init component=PCF

time="2021-07-22T10:16:57Z" level=info msg="| 200 | 10.112.106.106 | GET | /nnrf-disc/v1/nf-instances?requester-nf-type=PCF&service-names=nudr-dr&target-nf-type=UDR | " category=GIN component=NRF
```

说明 启动网元的顺序出现问题，严格按照以下顺序启动网元： `mongodb > NRF > UDR > UDM > AUSF > NSSF > AMF > PCF > UPF > SMF > webconsole > gNB > UE`

3. 如果在 启动UPF 的过程中报错和 `gtp5g` 模块相关，如下图所示

```
time="2021-07-22T14:03:21Z" level=error msg="gtp5g device named upfgtp created fail" category=Util component=UPF
time="2021-07-22T14:03:21Z" level=error msg="Gtp5gDeviceInit failed" category=Util component=UPF
time="2021-07-22T14:03:21Z" level=error msg="Epoll Wait Error : Invalid argument" category=Util component=UPF
time="2021-07-22T14:03:21Z" level=error msg="Epoll Wait error : Invalid argument" category=Util component=UPF
```

重新编译gtp5g模块，并且 重启该虚拟机


```
cd ~/gtp5g
sudo make
sudo make install
sudo reboot
```

6.7. 如何关闭各网元并重启

1. 对于在前端运行的模块，例如 **SMF**、**AMF** 等网元，直接 **ctrl+C** 退出运行
2. 对于使用 **nohup ./*** &** 在后台挂起运行的网元，使用 **ps -aux|grep ***|grep -v grep** 查看对应 **进程号**，******* 为对应的网元名称，例如 **nrf**，如何使用 **kill -9 进程号** 删除对应网元的进程
3. 对于 **webconsole**，由于已经在 **mongodb** 数据库中进行存储，所以需要清空对应的数据库

```
mongo free5gc
db.dropDatabase()
exit
```

```
cjn@vm2:~/free5gc/log$ mongo free5gc
MongoDB shell version v3.6.3
connecting to: mongodb://127.0.0.1:27017/free5gc
MongoDB server version: 3.6.3
Server has startup warnings:
2021-07-22T10:07:20.295+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
See http://dochub.mongodb.org/core/prodnotes-filesystem
2021-07-22T10:07:20.295+0000 I STORAGE [initandlisten] **
2021-07-22T10:07:22.305+0000 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
Read and write access to data and configuration is unrestricted.
2021-07-22T10:07:22.305+0000 I CONTROL [initandlisten] **
> db.dropDatabase()
{ "dropped" : "free5gc", "ok" : 1 }
> exit
bye
cjn@vm2:~/free5gc/log$
```

4. 然后按照 6.5 中的方法启动各个网元

6.8. 所有网元后台运行（不挂断）

1. 网元的启动顺序：**mongodb > NRF > UDR > UDM > AUSF > NSSF > AMF > PCF > UPF > SMF > webconsole > gNB > UE**
2. 对于 **mongodb**、**NRF**、**UDR**、**UDM**、**AUSF**、**NSSF**、**PCF** 仍然按照 **nohup ./... &** 方式启动
3. 对于 **gNB** 和 **UE**，由于是用来模拟基站和5G手机，如果没有测试多UE的需求，也不需要使用后台运行
4. 对于 **AMF** 和 **SMF**，进入 **~/free5gc/bin** 之后，使用 **nohup ./... &** 运行对应模块
5. 对于 **UPD**，SSH进入虚拟机之后，利用 **su root** 切换为root权限用户，然后输入以下指令

```
cd ~/free5gc/src/upf/build
nohup ./bin/free5gc-upfd -f ./config/upfcfg.yaml &
```


6. 对于 `webconsole` , SSH进入虚拟机之后, 输入以下指令

```
cd ~/free5gc/webconsole  
nohup go run server.go &
```

7. 注意：使用这种在后台运行的方式, 停止所有的网元都需要使用 `ps -aux|grep ***|grep -v grep` 查看进程号, 然后使用 `kill -9` 杀死进程