

Guía completa desde cero: FREE5GC + Docker + Network Slicing

Esta guía describe paso a paso la instalación y configuración de un simulador 5G basado en FREE5GC y UERANSIM utilizando Docker en Ubuntu 20.04 (bare metal). Incluye configuración de network slicing (eMBB y uRLLC), NAT, rutas, medición de KPIs y traffic control.

1. Requisitos previos

- Sistema operativo: Ubuntu 20.04 LTS instalado en hardware físico (NO máquina virtual)
- Mínimo 8 GB de RAM (16 GB recomendado)
- Usuario con permisos sudo
- Conexión a Internet

2. Actualización del sistema

Actualizar el sistema operativo antes de comenzar:

```
sudo apt update && sudo apt upgrade -y
sudo reboot
```

3. Instalación de dependencias básicas

Instalar herramientas y librerías necesarias:

```
sudo apt install -y \
    git curl wget net-tools iproute2 \
    build-essential make gcc \
    iptables iptables-persistent \
    linux-headers-$(uname -r)
```

4. Instalación de Docker

Eliminar versiones antiguas de Docker (si existen):

```
sudo apt remove -y docker docker-engine docker.io containerd runc
```

Instalar Docker usando el método oficial:

```
curl -fsSL https://get.docker.com | sudo bash
```

Habilitar y arrancar Docker:

```
sudo systemctl enable docker  
sudo systemctl start docker
```

Dar permisos al usuario actual:

```
sudo usermod -aG docker $USER  
newgrp docker
```

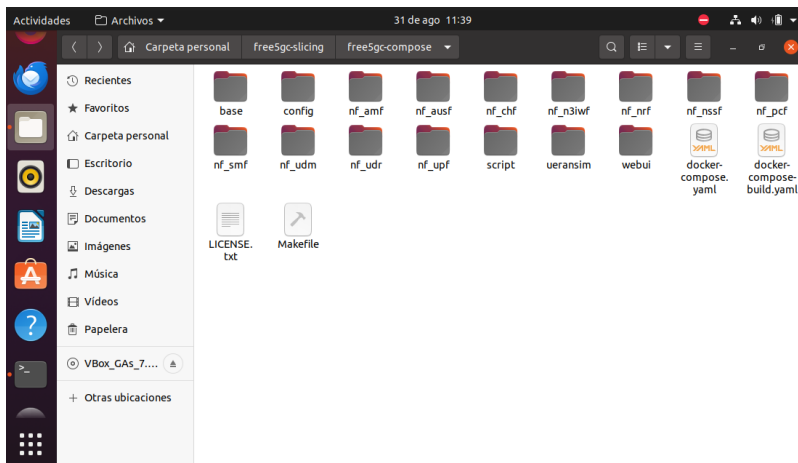
5. Forzar iptables legacy (PASO CRÍTICO)

FREE5GC y Docker utilizan iptables-legacy. Este paso evita problemas de NAT y conectividad.

```
sudo update-alternatives --set iptables /usr/sbin/iptables-legacy  
sudo update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy  
iptables --version
```

6. Clonación de repositorios

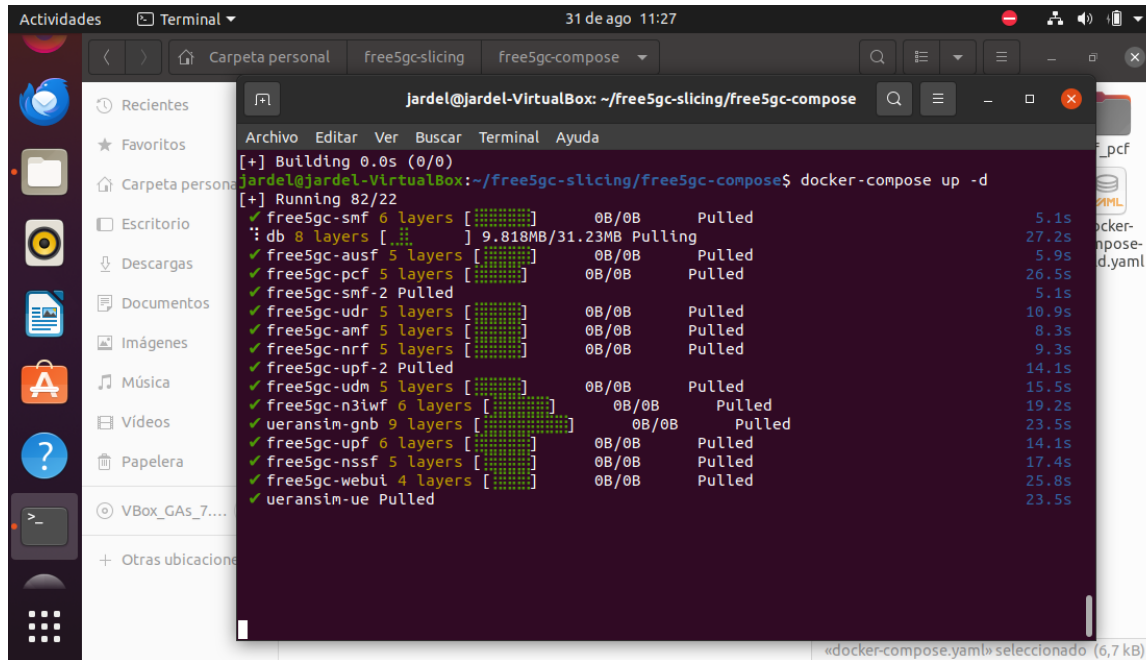
```
cd ~  
git clone https://github.com/abousselmi/docker-free5gc.git  
git clone https://github.com/lasseufpa/free5gc-slicing.git
```



7. Despliegue de FREE5GC con slicing

Levantar todos los contenedores del core 5G:

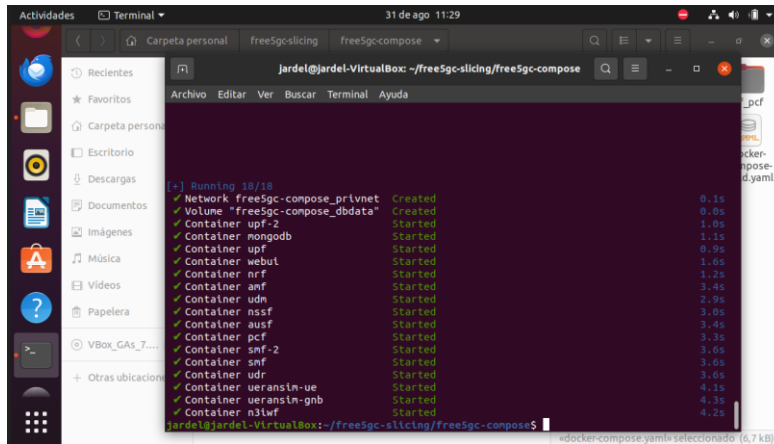
```
cd ~/free5gc-slicing/free5gc-compose
docker compose up -d
```



```
Archivo Editar Ver Buscar Terminal Ayuda
Jardel@jardel-VirtualBox: ~/free5gc-slicing/free5gc-compose
[+] Building 0.0s (0/0)
Jardel@jardel-VirtualBox:~/free5gc-slicing/free5gc-compose$ docker-compose up -d
[+] Running 82/22
 ✓ free5gc-smf 6 layers [ ] 0B/0B Pulled 5.1s
 ✓ db 8 layers [ ] 9.818MB/31.23MB Pulling 27.2s
 ✓ free5gc-ausf 5 layers [ ] 0B/0B Pulled 5.9s
 ✓ free5gc-pcf 5 layers [ ] 0B/0B Pulled 26.5s
 ✓ free5gc-smf-2 Pulled 5.1s
 ✓ free5gc-udr 5 layers [ ] 0B/0B Pulled 10.9s
 ✓ free5gc-amf 5 layers [ ] 0B/0B Pulled 8.3s
 ✓ free5gc-nrf 5 layers [ ] 0B/0B Pulled 9.3s
 ✓ free5gc-upf-2 Pulled 14.1s
 ✓ free5gc-udm 5 layers [ ] 0B/0B Pulled 15.5s
 ✓ free5gc-n3lwf 6 layers [ ] 0B/0B Pulled 19.2s
 ✓ ueransim-gnb 9 layers [ ] 0B/0B Pulled 23.5s
 ✓ free5gc-upf 6 layers [ ] 0B/0B Pulled 14.1s
 ✓ free5gc-nssf 5 layers [ ] 0B/0B Pulled 17.4s
 ✓ free5gc-webui 4 layers [ ] 0B/0B Pulled 25.8s
 ✓ ueransim-ue Pulled 23.5s
```

Verificar que todos los contenedores estén activos:

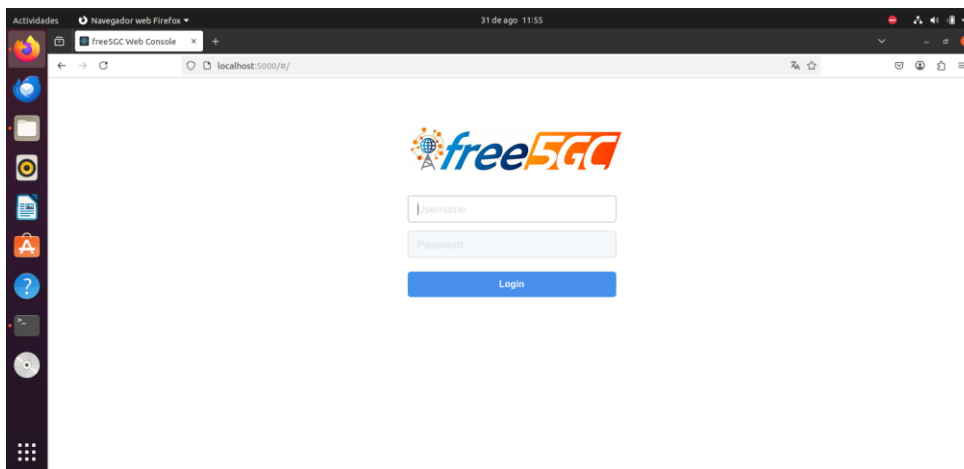
```
docker ps
```



8. Configuración de slices y UEs

Ingresar a la WebUI de Free5GC la cual se puede ingresar mediante cualquier navegador con:

<http://localhost:5000>



Login: admin

Password: free5gc

Luego generar los UEs conforme a las siguientes especificaciones:

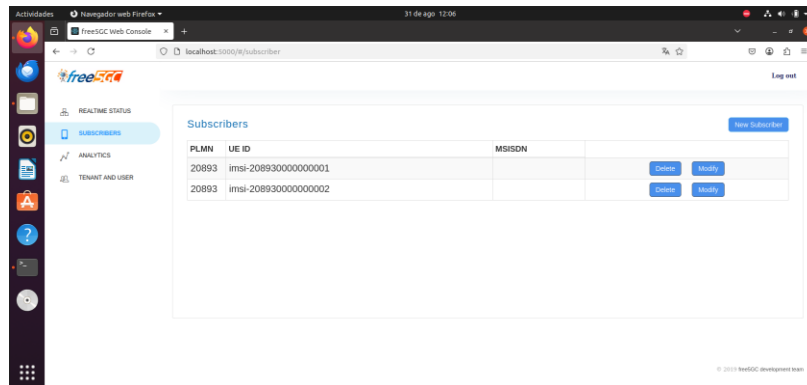
Slices configurados:

- Slice eMBB: SST=1, SD=010203, UE IP=10.60.0.0/16, UPF=upf
- Slice uRLLC: SST=2, SD=112233, UE IP=10.62.0.0/16, UPF=upf-2

Valores de SQN:

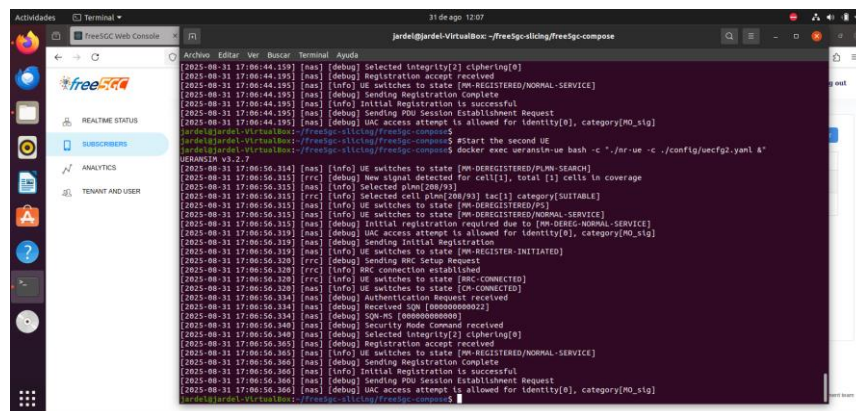
- UE1: 00000000000001

- UE2: 000000000000



Arrancar los UEs:

```
docker exec ueransim-ue bash -lc "./nr-ue -c config/uecfg.yaml &"
docker exec ueransim-ue bash -lc "./nr-ue -c config/uecfg2.yaml &"
```



9. Configuración de NAT en los UPF

NAT para slice eMBB:

```
docker exec upf bash -lc \
"iptables-legacy -t nat -A POSTROUTING -s 10.60.0.0/16 -o eth0 -j
MASQUERADE"
```

NAT para slice uRLLC:

```
docker exec upf-2 bash -lc \
"iptables-legacy -t nat -A POSTROUTING -s 10.62.0.0/16 -o eth0 -j
MASQUERADE"
```

10. Rutas de retorno en DN server

Agregar rutas estáticas para retorno de tráfico:

```
UPF1_IP=$(docker inspect -f '{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' upf)
UPF2_IP=$(docker inspect -f '{{range
.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' upf-2)
docker exec dn-server bash -lc "ip route add 10.60.0.0/16 via
$UPF1_IP"
docker exec dn-server bash -lc "ip route add 10.62.0.0/16 via
$UPF2_IP"
```

11. Verificación de conectividad

```
docker exec ueransim-ue ping -c 3 -I uesimtun0 10.100.200.1
docker exec ueransim-ue ping -c 3 -I uesimtun1 10.100.200.1
```

12. Instalación de herramientas de medición (KPIs)

```
sudo apt install -y iperf3 python3 python3-pip
pip3 install pandas matplotlib
docker exec dn-server apt install -y iperf3 iputils-ping
docker exec -d dn-server iperf3 -s
```

13. Traffic Control para diferenciar slices

Ejemplo de degradación eMBB y priorización uRLLC:

```
docker exec upf tc qdisc add dev eth0 root netem delay 80ms loss 2%
docker exec upf-2 tc qdisc add dev eth0 root netem delay 5ms loss 0.1%
```

13. Errores que presenta la simulación:

Los paquetes no pueden salir a internet (8.8.8.8) debido a incompatibilidad entre el kernel de gtp5g con el firmware del Linux. (interfaces físicas se confunden al momento de direccionar a interfaces lógicas).

Debido a que el desarrollo de free5gc no obtuvo mas actualizaciones el problema no tiene solución debiendo necesariamente recurrir al uso de Open5GS como solución de core 5G.