一。docker 安装

1.卸载可能存在旧版本的 docker:

sudo apt-get remove docker docker-engine docker-ce docker.io

2.更新 apt 包索引:

sudo apt-get update

3.为了使 apt 可以通过 https 使用 Repository,先安装以下包:

sudo apt-get install -y apt-transport-https ca-certificates curl software-properties-common

4.添加 Docker 官方 GPG 密钥:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key ad

5.检查 GPG Key 信息是否正确:

sudo apt-key fingerprint 0EBFCD88

GPG Key 信息如下:

pub

4096R/0EBFCD88 2017-02-22

Key fingerprint = 9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88uid

Docker Release (CE deb) <docker@docker.com>sub

4096R/F273FCD8 2017-02-22

6.添加源信息:

将源信息直接写入/etc/apt/sources.list

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

7.再更新下 apt 包索引:

sudo apt-get update

8.确认 Docker 的源信息是否正确，新的源是否添加成功:

sudo apt-cache madison docker-ce

```
 docker-ce | 18.03.1~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable amd64 Pack
ages
 docker-ce | 18.03.0~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable amd64 Pack
ages
 docker-ce | 17.12.1~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable amd64 Pack
ages
 docker-ce | 17.12.0~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable amd64 Pack
ages
 docker-ce | 17.09.1~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable amd64 Pack
ages
 docker-ce | 17.09.0~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable amd64 Pack
ages
 docker-ce | 17.06.2~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable amd64 Pack
ages
 docker-ce | 17.06.1~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable amd64 Pack
ages
 docker-ce | 17.06.0~ce-0~ubuntu | https://download.docker.com/linux/ubuntu xenial/stable amd64 Pack
ages
 docker-ce | 17.03.2~ce-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu xenial/stable amd
64 Packages
 docker-ce | 17.03.1~ce-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu xenial/stable amd
64 Packages
 docker-ce | 17.03.0~ce-0~ubuntu-xenial | https://download.docker.com/linux/ubuntu xenial/stable amd
64 Packages
```

https://blog.csdn.net/yx_222

9.安装docker

sudo apt install docker.io

10.Docker 安装验证

sudo docker -v

11.查看 Docker 安装版本详细信息

sudo docker version

12.启动 Docker

13.查看 docker 服务是否启动:

$ sudo systemctl status docker

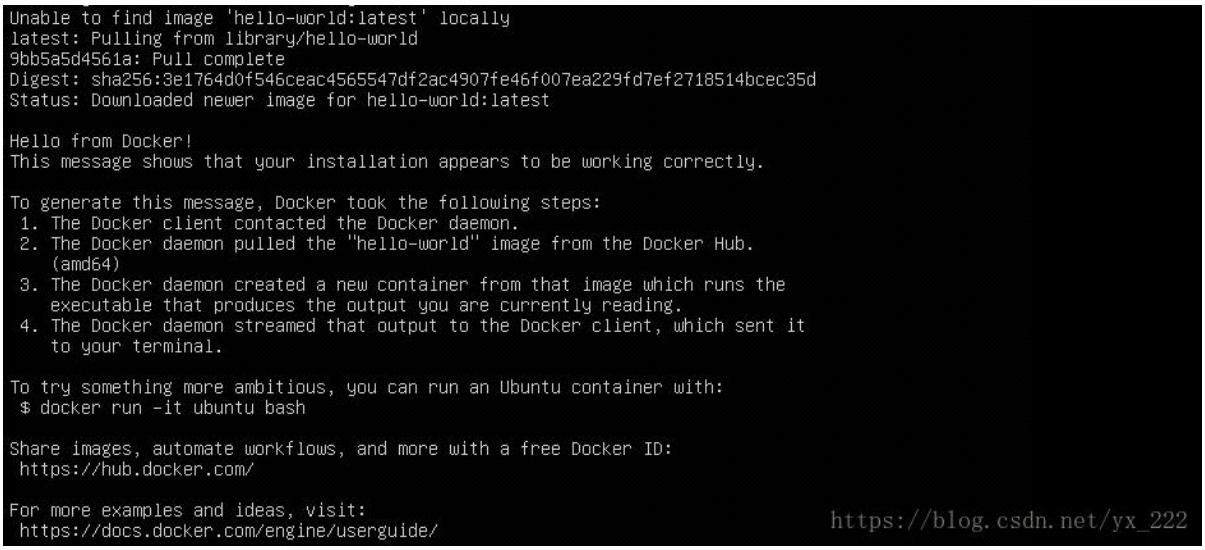
14.如果 Docker 未启动,则启动 Docker:

$ sudo systemctl start docke

15.运行 Hello World,校验 Docker 是否安装成功:

$ sudo docker run hello-world

运行这个命令会下载一个 hello-world 的镜像，并且运行在一个容器中。该容器运行时会有如下输出:



16. python3 安装

为了保证之后的运行不会出错,需要保证 python3 的版本至少是 3.6。使用以下命令查看 python3 版本。

$ python3 --version

Python 3.6.9

如果 python3 版本不够,更新方法如下:

下载 Gzipped source tarball(3.6.9 为例)

到 python 官网 https://www.python.org/downloads/release/python-369/

得到压缩包 Python-3.6.9.tgz

解压:

tar xfz Python-3.6.9.tgz

cd Python-3.6.9

./configure --with-ssl

make

sudo make install

会自动安装到/usr/local/bin 下

构建 docker 映像

检索正确的代码版本

在安装新的 epc-fed 之前,先清除老的版本

sudo docker ps -a

$ sudo docker rm prod-oai-spgwu-tiny prod-oai-spgwc prod-oai-mme prod-oai-hss

$ prod-cassandra -f

$ sudo rm -r openair-epc-fed/

克隆源码:

$ git clone https://github.com/OPENAIRINTERFACE/openair-epc-fed.git

$ cd openair-epc-fed

$ git checkout master

$ git pull origin master

$ sudo ./scripts/syncComponents.sh

OAI-HSS component branch : develop
OAI-MME component branch : develop
OAI-SPGW-C component branch : develop
OAI-SPGW-U component branch : develop
二。下载与编译

Build hss Image　　On a Ubuntu 18.04 Host
cd 到 openair-epc-fed/component/oai-hss 目录下
sudo docker build --target oai-hss --tag oai-hss:production --file
docker/Dockerfile.ubuntu18.04 .

Build mme Image
cd 到 openair-epc-fed/component/oai-mme 目录下,
$ sudo docker build --target oai-mme --tag oai-mme:production –file
docker/Dockerfile.ubuntu18.04 .
Build SPGW-C Image
cd 到 openair-epc-fed/component/oai-spgwc 目录下,
$ sudo docker build --target oai-spgwc --tag oai-spgwc:production –file
docker/Dockerfile.ubuntu18.04 .
Build SPGW-U Image
cd 到 openair-epc-fed/component/oai-spgwu-tiny 目录下,
$ sudo docker build --target oai-spgwu-tiny --tag oai-spgwu-tiny:production --file
docker/Dockerfile.ubuntu18.04

四个部分全部 build 完以后
$ sudo docker image prune --force
配置容器

部署容器
sudo docker run --name prod-cassandra -d -e CASSANDRA_CLUSTER_NAME="OAI HSS Cluster" -e CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch cassandra:2.
docker run --privileged --name prod-oai-hss -d --entrypoint /bin/bash oai-hss:production -c "sleep infini
docker network connect prod-oai-public-net prod-oa
docker run --privileged --name prod-oai-mme --network prod-oai-public-net \ -d --entrypoint /bin/bash oai-mme:production -c "sleep infin
docker run --privileged --name prod-oai-spgwc --network prod-oai-public-net \ -d --entrypoint /bin/bash oai-spgwc:production -c "sleep infinity"

docker run --privileged --name prod-oai-spgwu-tiny --network prod-oai-public-net \ -d --entrypoint /bin/bash oai-spgwu-tiny:production -c "sleep infinity"
*如果遇到 The container name "xxx" is already in use by container "xxx". You have to remove (or rename) that container to be able to reuse that name. 的问题
先运行sudo docker ps -a
找到找到对应的 container ID，
执行 $
sudo docker rm -f container ID 将该 container 先删去,再运行以上命令即可。 配置容器
我使用以下配置，命令中需要根据自己的配置进行设置。 PLMN MCC = 208 MNC = 93 TAC 5 <--- Primary TAC 6 7 APN1 = apn1.carrier.com APN2 = apn2.carrier.com 10 Users with the 1st IMSI =208930000000005 LTE-KEY = 8baf473f2f8fd09487cccbd7097c6862 OP = e734f8734007d6c5ce7a0508809e7e9c
Cassandra
$ sudo docker cp component/oai-hss/src/hss_rel14/db/oai_db.cql prod-cassandra:/home
$ sudo docker exec -it prod-cassandra /bin/bash -c "nodetool status" Cassandra_IP=`sudo docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" prod-cassandra`
$ sudo docker exec -it prod-cassandra /bin/bash -c "cqlsh --file /home/oai_db.cql ${Cassandra_IP}"

HSS
HSS_IP=`sudo docker exec -it prod-oai-hss /bin/bash -c "ifconfig eth1 | grep inet" | sed -f ./ci-scripts/convertIpAddrFromIfconfig.sed`
python3 component/oai-hss/ci-scripts/generateConfigFiles.py --kind=HSS --cassandra=${Cassandra_IP} --hss_s6a=${HSS_IP} --apn1=apn1.carrier.com --apn2=apn2.carrier.com --users=10 --imsi=208930000000005 --ltek=8baf473f2f8fd09487cccbd7097c6862 --op=e734f8734007d6c5ce7a0508809e7e9c --nb_mmes=1 --from_docker_fil
确认这一步会生成 hss-cfg.sh 文件，确认配置的 apn1，apn2，users，imsi，ltek，op 与 生成的 hss-cfg.sh 文件中的一致。
$ sudo docker cp ./hss-cfg.sh prod-oai-hss:/openair-hss/scripts $ sudo docker exec -it prod-oai-hss /bin/bash -c "cd /openair-hss/scripts && chmod 777 hss-cfg.sh && ./hss-cfg.sh"
MME
MME_IP=`docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" prod-oai-mme`
SPGW0_IP=`docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" prod-oai-spgwc`
python3 component/oai-mme/ci-scripts/generateConfigFiles.py --kind=MME \ --hss_s6a=${HSS_IP} --mme_s6a=${MME_IP} \ --mme_s1c_IP=${MME_IP} --mme_s1c_name=eth0 \ --mme_s10_IP=${MME_IP} --mme_s10_name=eth0 \ --mme_s11_IP=${MME_IP} --mme_s11_name=eth0 --spgwc0_s11_IP=${SPGW0_IP} \ --mcc=208

--mnc=93 --tac_list="5 6 7" --from_docker_file *同样这一步会生成 mme-cfg.sh 文件，需配置 mcc 和 mnc。

docker cp ./mme-cfg.sh prod-oai-mme:/openair-mme/scripts $ docker exec -it prod-oai-mme /bin/bash -c "cd /openair-mme/scripts && chmod 777 mme-cfg.sh && ./mme-cfg.sh"

SPGW-C

$ python3 component/oai-spgwc/ci-scripts/generateConfigFiles.py --kind=SPGW-C --s11c=eth0 --sxc=eth0 --apn=apn1.carrier.com --dns1_ip=8.8.8.8 --dns2_ip=8.8.4.4 --from_docker_file *生成 spgwc-cfg.sh 文件，不需更改。

$ sudo docker cp ./spgwc-cfg.sh prod-oai-spgwc:/openair-spgwc $ sudo docker exec -it prod-oai-spgwc /bin/bash -c "cd /openair-spgwc && chmod 777 spgwc-cfg.sh && ./spgwc-cfg.sh"

SPGW-U

$ python3 component/oai-spgwu-tiny/ci-scripts/generateConfigFiles.py --kind=SPGW-U --sxc_ip_addr=${SPGW0_IP} --sxu=eth0 --s1u=eth0 --from_docker_file *生成 spgwu-cfg.sh，不需更改。

$ sudo docker cp ./spgwu-cfg.sh prod-oai-spgwu-tiny:/openair-spgwu-tiny $ sudo docker exec -it prod-oai-spgwu-tiny /bin/bash -c "cd /openair-spgwu-tiny && chmod 777 spgwu-cfg.sh && ./spgwu-cfg.sh"