5G NRF开发环境搭建 v1.1

系统版本最好使用ubuntu18.04

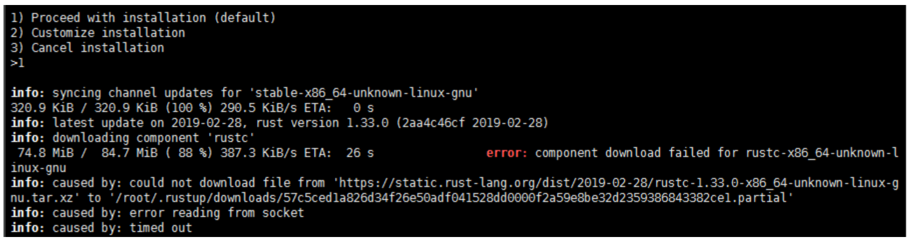
#安装rustup，利用rustup安装rust

```
$ curl https://sh.rustup.rs -sSf | sh
```

看到 “ Rust is installed now. Great! ” 即为安装成功，查看rust版本

```
$ rustc --version
```

如果安装过程如下错误，则重复多安装几遍，这种属于网络问题导致下载出错：



```
1) Proceed with installation (default)
2) Customize installation
3) Cancel installation
>1

info: syncing channel updates for 'stable-x86_64-unknown-linux-gnu'
320.9 KiB / 320.9 KiB (100 %) 290.5 KiB/s ETA:   0 s
info: latest update on 2019-02-28, rust version 1.33.0 (2aa4c46cf 2019-02-28)
info: downloading component 'rustc'
 74.8 MiB /  84.7 MiB ( 88 %) 387.3 KiB/s ETA:  26 s              error: component download failed for rustc-x86_64-unknown-l
inux-gnu
info: caused by: could not download file from 'https://static.rust-lang.org/dist/2019-02-28/rustc-1.33.0-x86_64-unknown-linux-g
nu.tar.xz' to '/root/.rustup/downloads/57c5ced1a826d34f26e50adf041528dd0000f2a59e8be32d2359386843382ce1.partial'
info: caused by: error reading from socket
info: caused by: timed out
```

#安装完成注意要修改环境变量

```
$ source $HOME/.cargo/env
```

#一般安装完成rust会有cargo，没有的话安装cargo

```
$ apt-get install cargo
```

可以尝试创建hello_world测试cargo的可用性

```
$cargo new hello_world --bin
$ls
hello_world
```

#安装git

```
$ apt-get install git
$ apt-get install gitk
```

最好把gitk一并装了，方便后面使用，有了gitk可以在git仓库中执行gitk命令查看分支和提交等

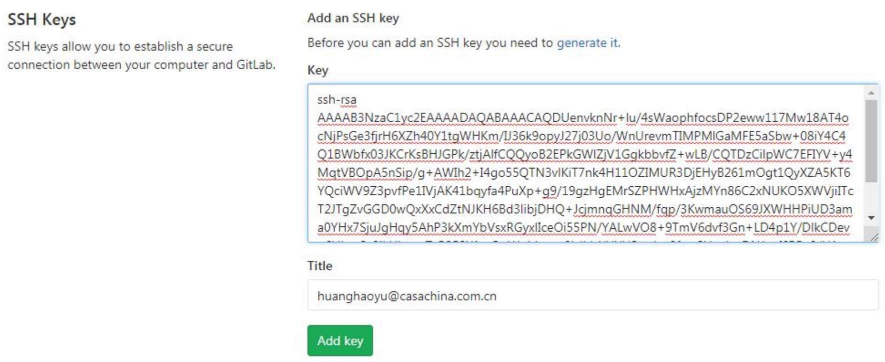
#设置本地git用户

```
$ git config --global user.name "huanghaoyu"
$ git config --global user.email "huanghaoyu@casachina.com.cn"
```

#使用ssh-keygen生成公钥，指定长度为4096，并确保本机的ssh能用

```
$ ssh-keygen -t rsa -C "huanghaoyu@casachina.com.cn" -b 4096
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDUenvknNr+lu/4sWaophfocsDP2eww117
Mw18AT4ocNjPsGe3fjrH6XZh40Y1tgWHKm/IJ36k9opyJ27j03Uo/WnUrevmTIMPMlGaMFE
5aSbw+08iY4C4Q1BWbfx03JKCrKsBHJGPk/ztjAlfCQQyoB2EPkGWlZjV1GgkbbvfZ+wLB/
CQTDzCilpWC7EFIYV+y4MqtVBOpA5nSip/g+AWIh2+I4go55QTN3vlKiT7nk4H11OZIMUR3
DjEHyB261mOgt1QyXZA5KT6YQciWV9Z3pvfPe1IVjAK41bqyfa4PuXp+g9/19gzHgEMrSZP
HWHxAjzMYn86C2xNUKO5XWVjiITcT2JTgZvGGD0wQxXxCdZtNJKH6Bd3libjDHQ+JcjmnqG
HNM/fqp/3KwmauOS69JXWHHPiUD3ama0YHx7SjuJgHqy5AhP3kXmYbVsxRGyxlIceOi55PN
 /YALwVO8+9TmV6dvf3Gn+LD4p1Y/DlkCDevn6Hhxy8x3jkULu+xZsB359Y1mOoWqidwym9
 hJhlzYUYUSqnJps0f+oSUqvlqsDNJuzf6P5p3/H1pyl4ldttSUj4G9pkv67JJUsaYpNhn6
 1OvB9H92HXzCe6jDi0DK481RS1Q5NySXfpAqDnPX5+fBcUMPSBJ0OHJJ7SYz9J1rNtcTng
 zIeQvTzhEul30+w== huanghaoyu@casachina.com.cn
```

将下面cat到的一长串内容拷贝到git的网站http://gitlab.casa-systems.com/profile/keys


SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

Before you can add an SSH key you need to generate it.

Key

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDUenvknNr+lu/4sWaophfocsDP2eww117Mw18AT4o
cNjPsGe3fjrH6XZh40Y1tgWHKm/IJ36k9opyJ27j03Uo/WnUrevmTIMPMIGaMFE5aSbw+08iY4C4
Q1BWbfx03JKCrKsBHJGPk/ztjAlfCQQyoB2EPkGWIZjV1GgkbbvfZ+wLB/CQTDzCilpWC7EFIYV+y4
MqtVBOpA5nSip/g+AWIh2+I4go55QTN3vIKiT7nk4H11OZIMUR3DjEHyB261mOgt1QyXZA5KT6
YQciWV9Z3pvfPe1IVjAK41bqyfa4PuXp+g9/19gzHgEMrSZPHWHxAjzMYn86C2xNUKO5XWVjiITc
T2JTgZvGGD0wQxXxCdZtNJKH6Bd3libjDHQ+JcjmnqGHNM/fqp/3KwmauOS69JXWHHPiUD3am
a0YHx7SjuJgHqy5AhP3kXmYbVsxRGyxlIceOi55PN/YALwVO8+9TmV6dvf3Gn+LD4p1Y/DIkCDev
```

Title

huanghaoyu@casachina.com.cn

点击Add key，成功加入之后会发邮件通知。如果遇到网站打不开的问题，先确定电脑主机加入了gitlab的域名解析文件

****

有遇到某些机子ssh-keygen使用时报错如下，暂时无解

```
$ ssh-keygen
OpenSSL version mismatch. Built against 1000204f, you have 1000106f
```

即使使用apt-get --purge remove openssh-server和apt-get --purge remove openssl彻底删除再重装也得不到解决

#如果遇到本机没有安装ssh，先安装ssh并且设置为可root用户登录，方便xshell开终端

```
$ apt-get install ssh
$ vim /etc/ssh/sshd_config
```

把这几行改为如下：

```
# Authentication:
LoginGraceTime 120
PermitRootLogin yes
#PermitRootLogin without-password
StrictModes yes
```

重启ssh服务即可

```
$ service ssh restart
```

#用git拷贝代码仓库

```
$ git clone git@gitlab.casa-systems.com:mobility/sim-nrf.git
```

#进去仓库里先尝试编译

```
$ cd sim-nrf
$ cargo build
```

默认会尝试去https//:crates.io拉取所需要的crate，要保证网速够好，crate源在国外，目前只能在物理机上编译

****

发生过在虚拟机里无论如何都下载不下来的情况，虚拟机曾使用过iperf测过吞吐量，也用过scp拷大文件对比与host主机的网速，已经基本持平，然后还是下载不了，原因不明。即使更换crate源为国内的中科大源也不管用，网速一开始会很快，但是后面会卡住连不上，可能中科大的源中缺少某些所需的crate

#安装docker-ce，可以参考https://docs.docker.com/install/linux/docker-ce/ubuntu/，大概的安装步骤如下：

```
$ apt-get remove docker docker-engine docker.io containerd runc
$ apt-get install \
apt-transport-https \
ca-certificates \
curl \
software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) \
 stable"
```

```
$ apt-get install docker-ce
```

#手动更改docker源

```
$ touch /etc/docker/daemon.json
$ vim /etc/docker/daemon.json
```

加入以下内容，这样docker创建镜像会优先使用国内的源

```
{
 "registry-mirrors":["https://registry.docker-cn.com"]
}
```

#安装docker-compose，docker-compose是一个多容器管理和部署工具

```
$ env GIT_SSL_NO_VERIFY=true
$ git config http.sslVerify "false"
$ curl -L "https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
$ chmod 777 /usr/local/bin/docker-compose
```

由于国内网络环境原因，几乎是下不下来的，这个时候用uname -m和uname -s查看本机的系统版本，然后在windows上根据此版本去https://github.com/docker/compose/releases/download/1.22.0/这个网址下载即可，一般在windows上也是下载不了的，需要先下载到美国的服务器，再拉回来，然后把下载好的文件，比如docker-compose-Linux-x86_64，把它放在/usr/local/bin目录下并修改名字为docker-compose，chmod 777开放所有权限

关于docker-compose更详细的资料可以参考https://www.cnblogs.com/neptunemoon/p/6512121.html

#由于后面需要去gitlab上拉去docker的镜像，所以要先添加gitlab的域名

```
$ vim /etc/hosts
```

然后加入以下域名：

```
50.206.125.231 gitlab.casa-systems.com
50.206.125.231 registry.gitlab.casa-systems.com
```

#要在gitlab上拉取镜像还需要证书，把证书解压后放到docker的目录下即可

```
$ cp -r cert.d /etc/docker/cert.d/
```

#使用docker-compose拉取镜像部署服务

```
docker-compose -f docker-compose-sqa.yml up
```

查看docker-compose-sqa.yml后内容如下：

```
$ cat docker-compose-sqa.yml
version: '3'
services:
 sim-nrf:
 image: registry.gitlab.casa-systems.com/mobility/sim-nrf:latest

 pytest-server:
 image: registry.gitlab.casa-systems.com/mobility/sim-nrf/pytest-server:latest

  dev:
  image: registry.gitlab.casa-systems.com/mobility/sim-nrf/docker-dev
  volumes:
  - "/var/run/docker.sock:/var/run/docker.sock"
  - "/usr/bin/docker:/usr/bin/docker"
  - ./:/work
  depends_on:
  - sim-nrf
  - pytest-server
```

版本为3，一共有三个服务，image为所要拉取的镜像地址，volume为文件映射，比如"/usr/bin/docker:/usr/bin/docker"是把上下文（context）目录下的/usr/bin/docker映射到到镜像的/usr/bin/docker，depends_on是说要运行dev需要依赖sim-nrf和pytest-server这两个服务，所以会先运行这两个
成功运行后情况如下：

```
Starting sim-nrf_pytest-server_1 ... done
Creating sim-nrf_sim-nrf_1 ... done
Recreating sim-nrf_dev_1 ... done
Attaching to sim-nrf_pytest-server_1, sim-nrf_sim-nrf_1, sim-nrf_dev_1
sim-nrf_1 | 8:C 10 Jan 06:27:33.487 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
sim-nrf_1 | 8:C 10 Jan 06:27:33.487 # Redis version=4.0.11, bits=64, commit=bca38d14, modified=0, pid=8, just started
sim-nrf_1 | 8:C 10 Jan 06:27:33.487 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
sim-nrf_1 | ERROR 2019-01-10T06:27:33Z: r2d2: address not available: Address not available (os error 99)
sim-nrf_1 | ERROR 2019-01-10T06:27:33Z: r2d2: address not available: Address not available (os error 99)
 sim-nrf_1 | ERROR 2019-01-10T06:27:33Z: r2d2: address not available: Address not available (os error 99)
 sim-nrf_1 | ERROR 2019-01-10T06:27:33Z: r2d2: address not available: Address not available (os error 99)
 sim-nrf_1 | ERROR 2019-01-10T06:27:33Z: r2d2: address not available: Address not available (os error 99)
 sim-nrf_1 | ERROR 2019-01-10T06:27:33Z: r2d2: address not available: Address not available (os error 99)
 sim-nrf_1 | ERROR 2019-01-10T06:27:33Z: r2d2: address not available: Address not available (os error 99)
 sim-nrf_1 | ERROR 2019-01-10T06:27:33Z: r2d2: address not available: Address not available (os error 99)
 sim-nrf_1 | ERROR 2019-01-10T06:27:33Z: r2d2: address not available: Address not available (os error 99)
 sim-nrf_1 | ERROR 2019-01-10T06:27:33Z: r2d2: address not available: Address not available (os error 99)
 sim-nrf_1 | 8:M 10 Jan 06:27:33.490 * Running mode=standalone, port=6379.
 sim-nrf_1 | 8:M 10 Jan 06:27:33.490 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
 sim-nrf_1 | 8:M 10 Jan 06:27:33.490 # Server initialized
 sim-nrf_1 | 8:M 10 Jan 06:27:33.490 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
 sim-nrf_1 | 8:M 10 Jan 06:27:33.490 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency and memory usage issues with Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
 sim-nrf_1 | 8:M 10 Jan 06:27:33.490 * Ready to accept connections
 sim-nrf_1 | Started sim-nrf: 0.0.0.0:80 with DB at localhost:6379
```

这个用于docker-compose-sqa.yml用于QA测试，不知道这个docker用的sim-nrf可执行文件是哪来的

#研发使用docker-compose-dev.yml这个yaml配置文件

```
$ docker-compose -f docker-compose-dev.yml up
```

查看当前运行的docker container

```
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
c5880c75b1b8 registry.gitlab.casa-systems.com/mobility/sim-nrf/docker-dev "tail -f /dev/null" 6 minutes ago Up 6 minutes sim-nrf_dev_1
5d821703c81a redis:alpine "docker-entrypoint.s…" 3 weeks ago Up 6 minutes 6379/tcp sim-nrf_redis_1
```

或者

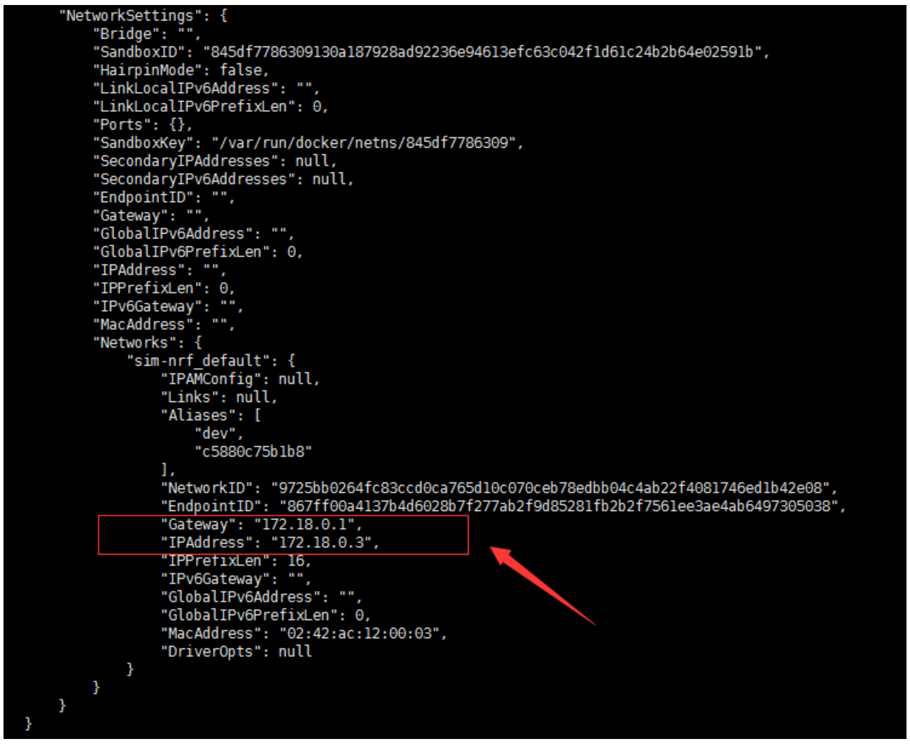
```
$ docker-compose -f docker-compose-dev.yml ps
 Name Command State Ports
--------------------------------------------------------------------
sim-nrf_dev_1 tail -f /dev/null Up
sim-nrf_redis_1 docker-entrypoint.sh redis ... Up 6379/tcp
```

停止docker的命令：

```
$ docker stop sim-nrf_dev_1
$ docker rm sim-nrf_dev_1
```

查看docker ip

```
$ docker inspect sim-nrf_dev_1
```

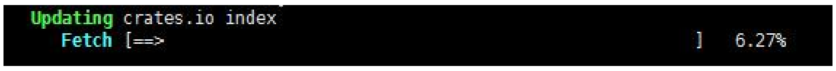


```
        "NetworkSettings": {
            "Bridge": "",
            "SandboxID": "845df7786309130a187928ad92236e94613efc63c042f1d61c24b2b64e02591b",
            "HairpinMode": false,
            "LinkLocalIPv6Address": "",
            "LinkLocalIPv6PrefixLen": 0,
            "Ports": {},
            "SandboxKey": "/var/run/docker/netns/845df7786309",
            "SecondaryIPAddresses": null,
            "SecondaryIPv6Addresses": null,
            "EndpointID": "",
            "Gateway": "",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "",
            "IPPrefixLen": 0,
            "IPv6Gateway": "",
            "MacAddress": "",
            "Networks": {
                "sim-nrf_default": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": [
                        "dev",
                        "c5880c75b1b8"
                    ],
                    "NetworkID": "9725bb0264fc83ccd0ca765d10c070ceb78edbb04c4ab22f4081746ed1b42e08",
                    "EndpointID": "867ff00a4137b4d6028b7f277ab2f9d85281fb2b2f7561ee3ae4ab6497305038",
                    "Gateway": "172.18.0.1",
                    "IPAddress": "172.18.0.3",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "02:42:ac:12:00:03",
                    "DriverOpts": null
                }
            }
        }
    }
```

#用exec命令进入docker容器sim-nrf_dev_1里面做开发，docker里面已经映射了本地的NRF仓库到/work目录下

```
$ docker exec -it sim-nrf_dev_1 bash
root@c5880c75b1b8:/# cd work
root@c5880c75b1b8:/work# cargo run
```

运行cargo run之后会根据cargo.toml生成cargo.lock，他会到github上去检索所有所需crate的路径，然后开始下载所需要的crate



```
  Updating crates.io index
     Fetch [==>                                                   ]  6.27%
```

****

每次carg.toml被修改都需要重新updating crates.io index，这个过程有可能会因为网络问题而失败，表现为fetch进度条不动然后很久之后就提示ssh远程连接失败，然后重新连接，如果尝试三次没成功就结束

目前没有较好的解决办法，只能到美国的设备去编译：

①ssh 192.168.2.44 （hhr/casawlan）

②ssh casa@10.180.2.10 （pwd：casa）

③把虚拟机跑起来

```
ssh -i ~/Downloads/lli-wag1.pem ubuntu@10.180.70.37
```

④配置git的user.name和user.email

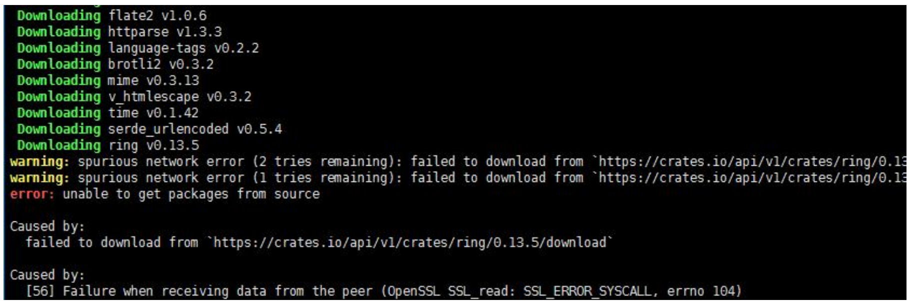
⑤运行以下命令把docker跑起来

```
$ docker pull registry.gitlab.casa-systems.com \
/mobility/sim-nrf/docker-dev:latest
$ docker run -d --name dev-lli -v \
/var/run/docker.sock:/var/run/docker.sock -v \
/usr/bin/docker:/usr/bin/docker -e DISPLAY=$DISPLAY -v \
/tmp/.X11-unix:/tmp/.X11-unix --mount type=bind,source="$(pwd)", \
target=/work --rm registry.gitlab.casa-systems.com/mobility/sim-nrf/ \
docker-dev:latest
```

⑥exec进入到dev-lli这个容器中运行cargo run，网速很快，会很快编译完成并运行起来

或者直接下载docker-compose然后用docker-compose-dev.yml把sim-nrf_dev_1运行起来然后用exec进到这个容器里

如果update index成功则会很快下载完全部的crates，但是编译的时候有可能会下不到ring这个库



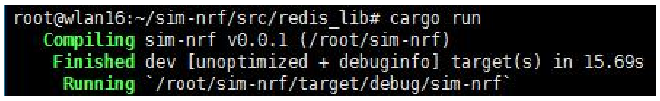
```
  Downloading flate2 v1.0.6
  Downloading httparse v1.3.3
  Downloading language-tags v0.2.2
  Downloading brotli2 v0.3.2
  Downloading mime v0.3.13
  Downloading v_htmlescape v0.3.2
  Downloading time v0.1.42
  Downloading serde_urlencoded v0.5.4
  Downloading ring v0.13.5
warning: spurious network error (2 tries remaining): failed to download from `https://crates.io/api/v1/crates/ring/0.13
warning: spurious network error (1 tries remaining): failed to download from `https://crates.io/api/v1/crates/ring/0.13
error: unable to get packages from source

Caused by:
  failed to download from `https://crates.io/api/v1/crates/ring/0.13.5/download`

Caused by:
  [56] Failure when receiving data from the peer (OpenSSL SSL_read: SSL_ERROR_SYSCALL, errno 104)
```

这时候需要去提示中所描述的https://crates.io/api/v1/crates/ring/0.13.5/download手动下载crate并放置到~/.cargo/registry/cache/github.com-1ecc6299db9ec823/中。

编译完成后就可以看到NRF正确运行



```
root@wlan16:~/sim-nrf/src/redis_lib# cargo run
   Compiling sim-nrf v0.0.1 (/root/sim-nrf)
    Finished dev [unoptimized + debuginfo] target(s) in 15.69s
     Running `/root/sim-nrf/target/debug/sim-nrf`
```

****

这里不确定是否拷贝一个已经编译好的cargo.lock可以省去updating crates.io index的过程

#也可以不在docker container 里做开发，直接在本地运行，不过NRF的运行以来于数据库redis，需要先安装数据库并把数据库跑起来

```
$ apt-get install redis
$ redis-server &
```

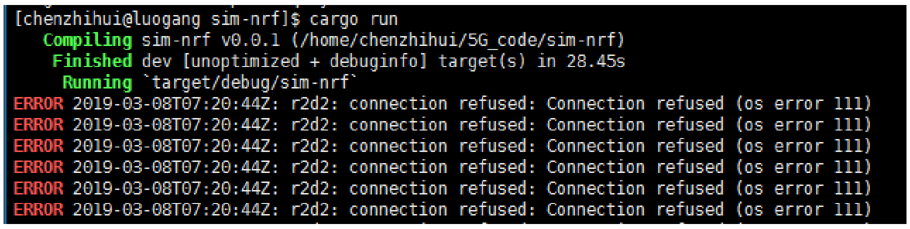
&指定为后台运行

如果需要清除数据库的数据可以用redis的客户端命令

```
$ redis-cli
127.0.0.1:6379> flushdb
OK
127.0.0.1:6379> flushall
OK
127.0.0.1:6379> exit
```

exit退出客户端

#执行cargo run运行NRF，NRF的ip地址分配需要依赖于redis，其正常的运行数据存储也需要依赖于redis，如果没安装redis会出现如下问题



```
[chenzhihui@luogang sim-nrf]$ cargo run
   Compiling sim-nrf v0.0.1 (/home/chenzhihui/5G_code/sim-nrf)
    Finished dev [unoptimized + debuginfo] target(s) in 28.45s
     Running `target/debug/sim-nrf`
ERROR 2019-03-08T07:20:44Z: r2d2: connection refused: Connection refused (os error 111)
ERROR 2019-03-08T07:20:44Z: r2d2: connection refused: Connection refused (os error 111)
ERROR 2019-03-08T07:20:44Z: r2d2: connection refused: Connection refused (os error 111)
ERROR 2019-03-08T07:20:44Z: r2d2: connection refused: Connection refused (os error 111)
ERROR 2019-03-08T07:20:44Z: r2d2: connection refused: Connection refused (os error 111)
ERROR 2019-03-08T07:20:44Z: r2d2: connection refused: Connection refused (os error 111)
```