**Fabric CA基本操作**

目录

[CA基本操作流程图 2](#_Toc518811136)

[1.通过容器的方式启动ca-server 2](#_Toc518811137)

[2.查看生成的内容 3](#_Toc518811138)

[2.1该ca-server的证书（根证书）内容： 3](#_Toc518811139)

[2.2查看ca-server当前数据库内容（数据库内置为sqlite3） 4](#_Toc518811140)

[3.CA管理员通过fabric-ca-client获得证书(Ecert)和私钥 5](#_Toc518811141)

[4.通过CA管理员来register新的identity。 7](#_Toc518811142)

[4.1 admin（CA管理员）的属性 7](#_Toc518811143)

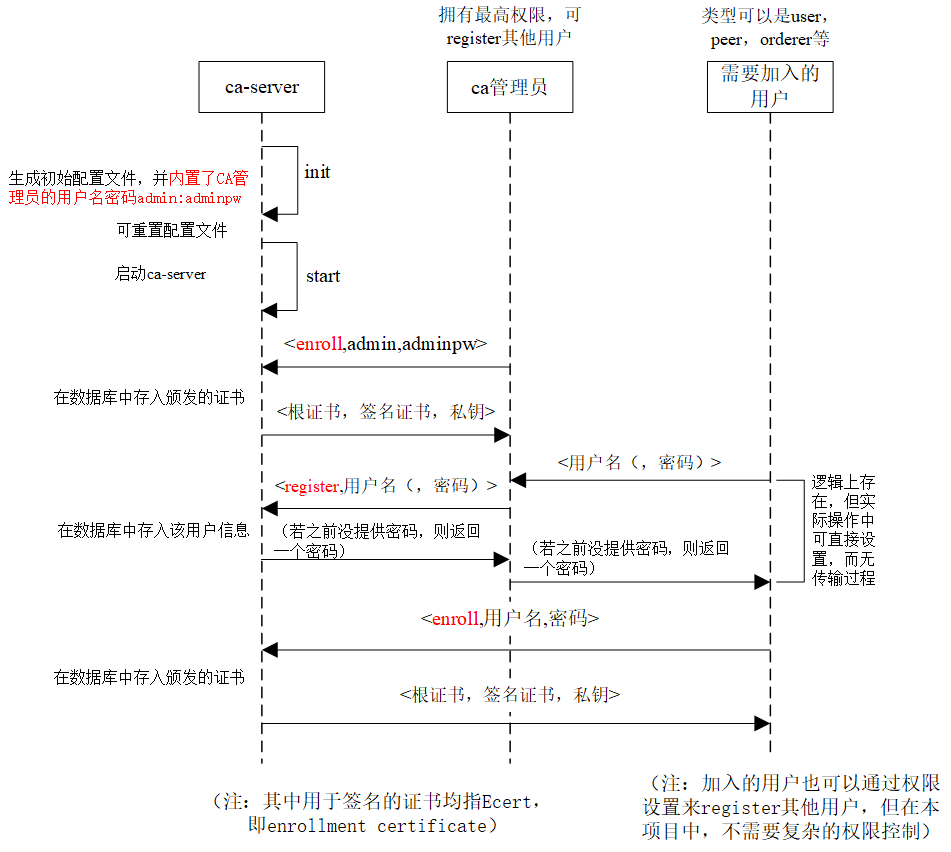
[4.2 Register时涉及到的参数 8](#_Toc518811144)

[4.3注册peer 9](#_Toc518811145)

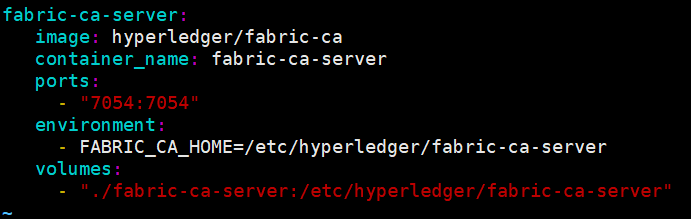
[5.已注册的identity进行enroll操作 10](#_Toc518811146)

[6.查看ca-server的日志记录 13](#_Toc518811147)

# CA基本操作流程图



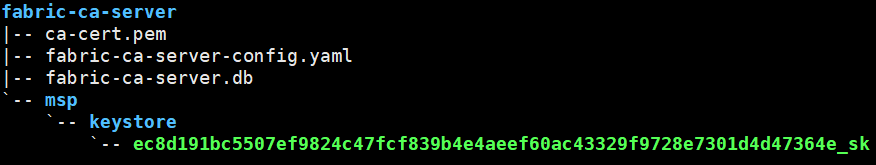
# 1.通过容器的方式启动ca-server



然后执行docker-compose up -d 启动这个ca-server容器。

执行sudo docker exec -it fabric-ca-server /bin/bash进入这个容器，在容器中执行命令fabric-ca-server init -b admin:adminpw来初始化ca-server，-b参数提供了节点登记过程的引导节点，后面登记节点的时候需要用到，admin和adminpw是默认的用户名和密码。

此时会在/etc/hyperledger文件夹下生成文件夹fabric-ca-server，内容如下：



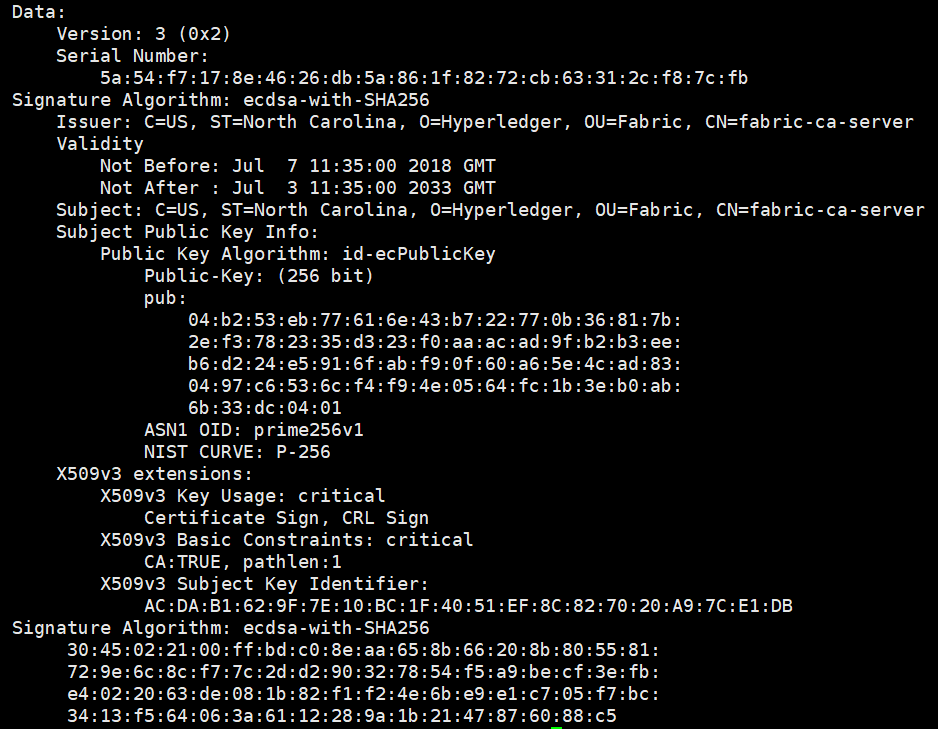
其中fabric-ca-server-config.yaml是配置文件，可以根据需要修改配置文件内容并删除ca-cert.pem和msp文件，之后启动ca-server时便可根据修改过后的配置文件生成相应的证书及私钥。

执行fabric-ca-server start -b admin:adminpw命令来启动ca-server。

# 2.查看生成的内容

## 2.1该ca-server的证书（根证书）内容：

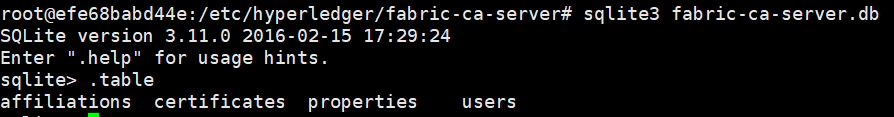
使用命令openssl x509 -in ca-cert.pem -inform pem -noout -text可以查



可以看到key usage包括证书签名及证书吊销列表（CRL）的签名。

## 2.2查看ca-server当前数据库内容（数据库内置为sqlite3）

查看当前表有4个



可以使用命令select\* from 表名; 查看表中数据。可以看到certificates表中为空，因为此时还没有颁发证书。users表中有admin这个用户。

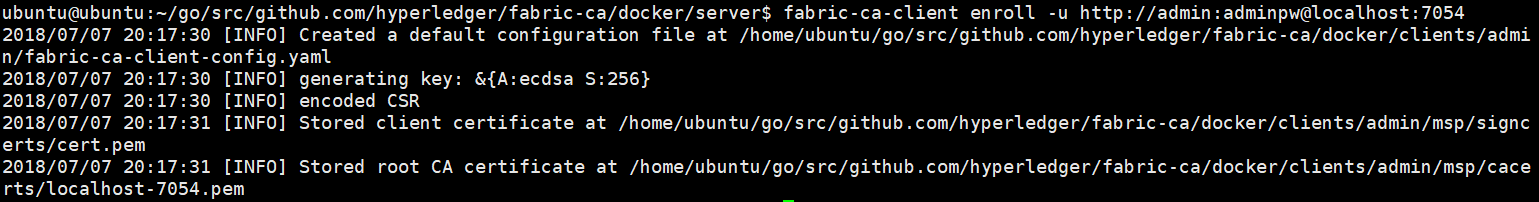
# 3.CA管理员通过fabric-ca-client获得证书(Ecert)和私钥

Fabric-ca-server在启动时已将CA管理员用户的用户名密码内置于其中，即admin：adminpw，所以此时需要CA管理员用户通过enroll获得其证书（Ecert）和私钥。

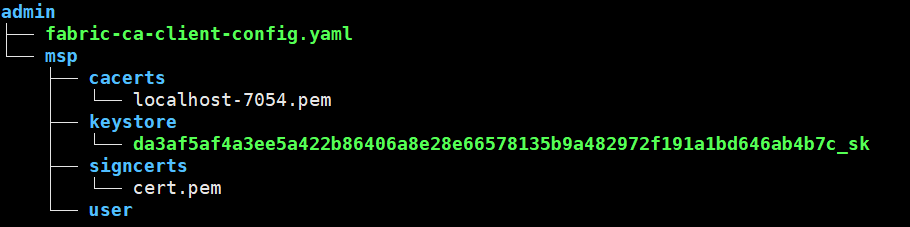
首先需要设置环境变量：

export FABRIC\_CA\_CLIENT\_HOME=~/go/src/github.com/hyperledger/fabric-ca/docker/clients/admin，即生成的内容会存放于这个目录下。

然后执行fabric-ca-client enroll -u <http://admin:adminpw@localhost:7054>



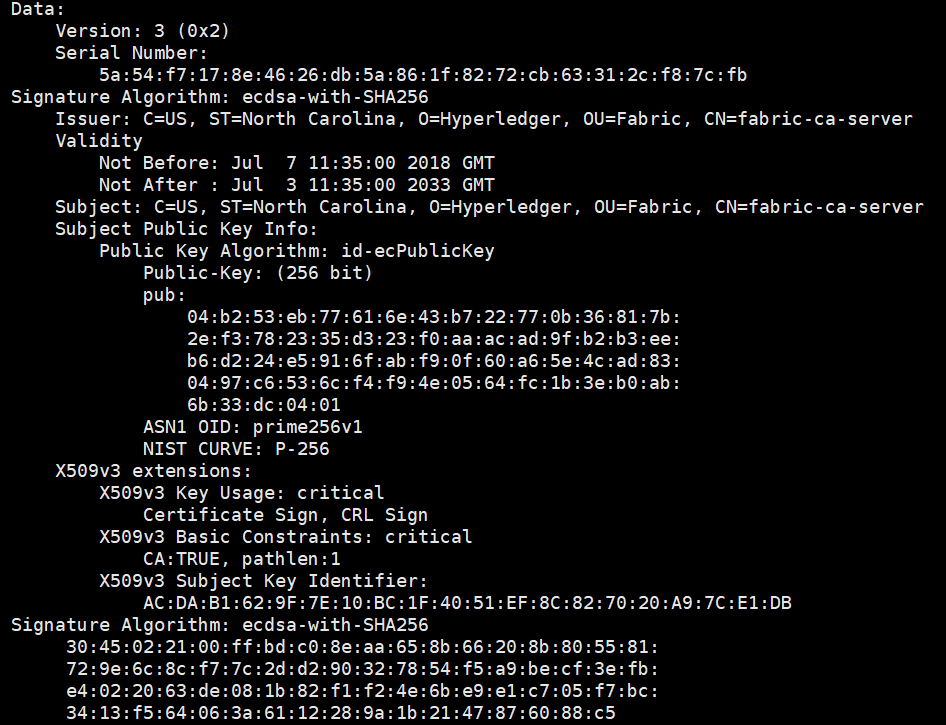
查看admin目录：



其中fabric-ca-client-config.yaml是配置文件；

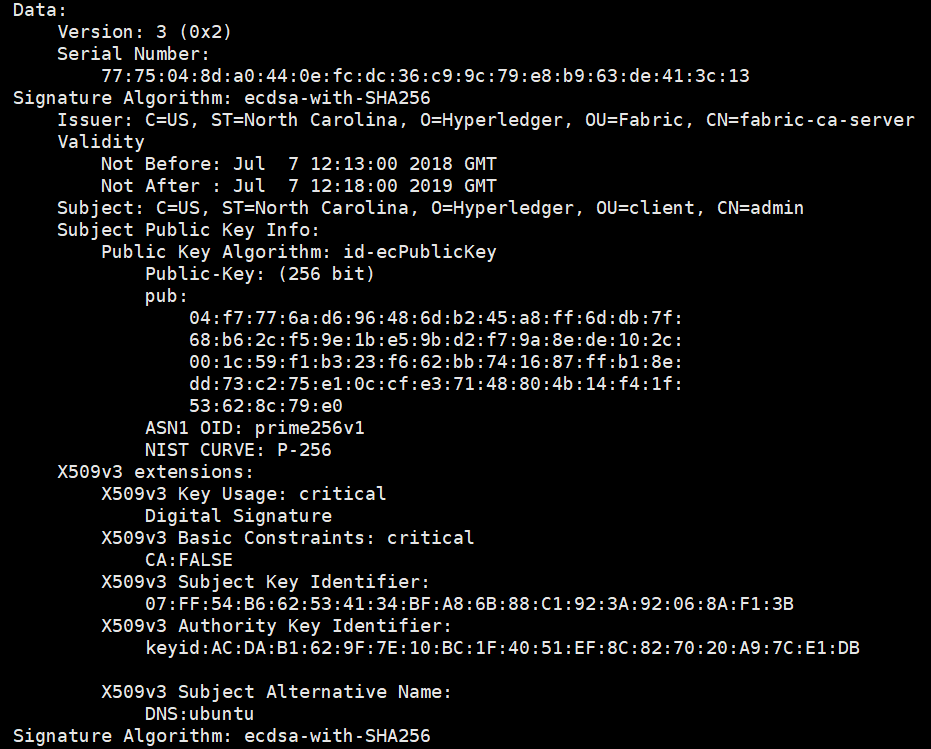
cacerts中的localhost-7054.pem是根证书也就是上文中ca-server中的证书，查看对比及可验证；

localhost-7054.pem:



keystore中是用于签名的私钥，其公钥对应于signcerts中的证书cert.pem。

cert.pem:可以看到key usage 只有 digital signature



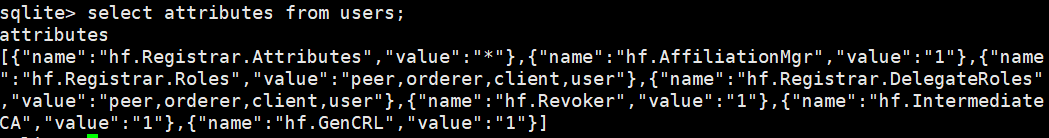
进入ca-server 的docker中，再次查看数据库中的表certificates，可以看到此时多出来一个证书，代表其签发的证书，根据序列号可以看出该证书就是cert.pem，且根据authority\_key\_identifier可以看出该证书是由根证书签发的。



# 4.通过CA管理员来register新的identity。

## 4.1 admin（CA管理员）的属性

我们可以通过查看ca-server中生成的数据库中的users表的attributes列来查看admin（CA管理员）的属性。（或者在server的配置文件中查看）



具体内容如下：可见管理员有最高权限。

[{"name":"hf.Registrar.Attributes","value":"\*"},

{"name":"hf.AffiliationMgr","value":"1"},

{"name":"hf.Registrar.Roles","value":"peer,orderer,client,user"},

{"name":"hf.Registrar.DelegateRoles","value":"peer,orderer,client,user"},

{"name":"hf.Revoker","value":"1"},

{"name":"hf.IntermediateCA","value":"1"},

{"name":"hf.GenCRL","value":"1"}]

hf.Registrar.Attributes：值为\*，可以为任意部门关系的identity注册；

hf.AffiliationMgr：值为true，Identity is able to manage affiliations if attribute value is true；

hf.Registrar.Roles：可以为peer,orderer,client,user这四个role注册；

hf.Registrar.DelegateRoles：List of roles that the registrar is allowed to give to a registree for its ‘hf.Registrar.Roles’ attribute；

hf.Revoker：Identity is able to revoke a user and/or certificates if attribute value is true；

hf.IntermediateCA：值为true，能够enroll一个身份作为中间CA，在attribute属性允许的情况下；

hf.GenCRL：值为true，能够生成CRL, 在attribute属性允许的情况下.

在attribute属性允许的情况下，是指hf.Registrar.Attributes的值是被注册人attribute前缀或相等。Attribute具体指什么？

## 4.2 Register时涉及到的参数

如下图所示（和数据库中users的字段一致）：

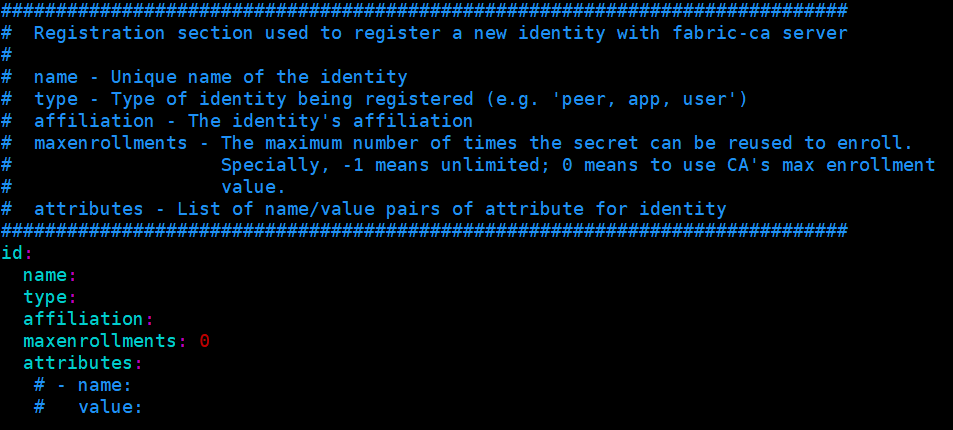
Name：注册的名字，唯一的

Type：peer, user, orderer, client等

Affiliation：从属关系，机构，部门，team等

Maxenrollments：当前密码最多被enroll的次数，若设为-1，则代表无限，若设为0，则代表用CA的该值。

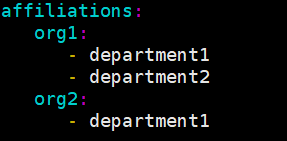
Attributes：上述内容。



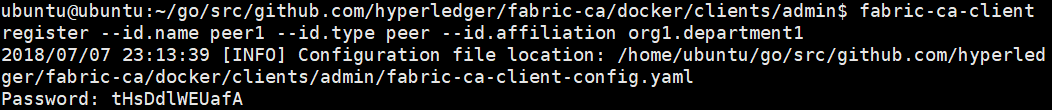
## 4.3注册peer

注：此时环境变量FABRIC\_CA\_CLIENT\_HOME确定是在admin文件夹下，因为该文件夹包含了admin的相关内容。

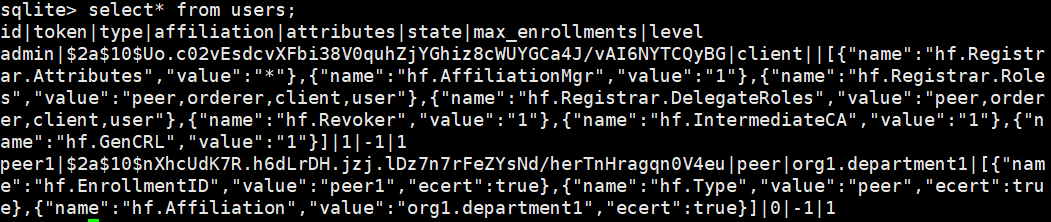
1. 执行命令fabric-ca-client register --id.name peer1 --id.type peer --id.affiliation org1.department1，代表注册了一个名为peer1的peer类型的用户，其从属机构为org1.department1（从属机构的内容在ca-server的配置文件中定义）。



可以看到执行完这个命令后，会生成该用户对应的密码tHsDdlWEUafA，用于之后的enroll。

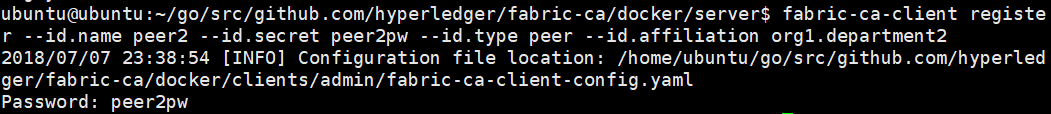


查看ca-server的数据库中的users表 可以看到此时多了id为peer1的条目。

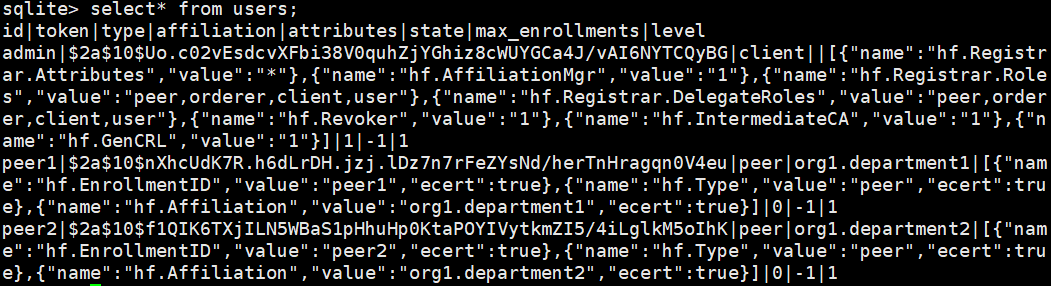


1. 通过指定密码来注册一个名为peer2的peer类型的用户，所属于org1.department2。

执行命令fabric-ca-client register --id.name peer2 --id.secret peer2pw --id.type peer --id.affiliation org1.department2



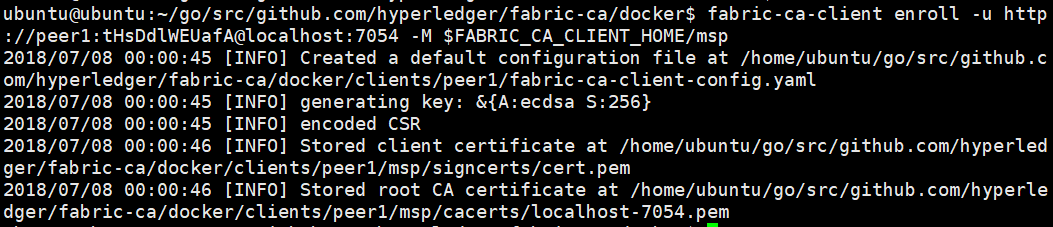
查看ca-server的数据库中的users表 可以看到此时多了id为peer2的条目。



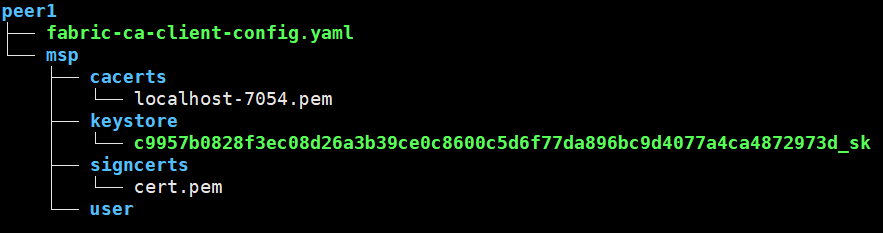
# 5.已注册的identity进行enroll操作

首先设置环境变量，使得之后生成的内容存放于该环境变量下，export FABRIC\_CA\_CLIENT\_HOME=~/go/src/github.com/hyperledger/fabric-ca/docker/clients/peer1

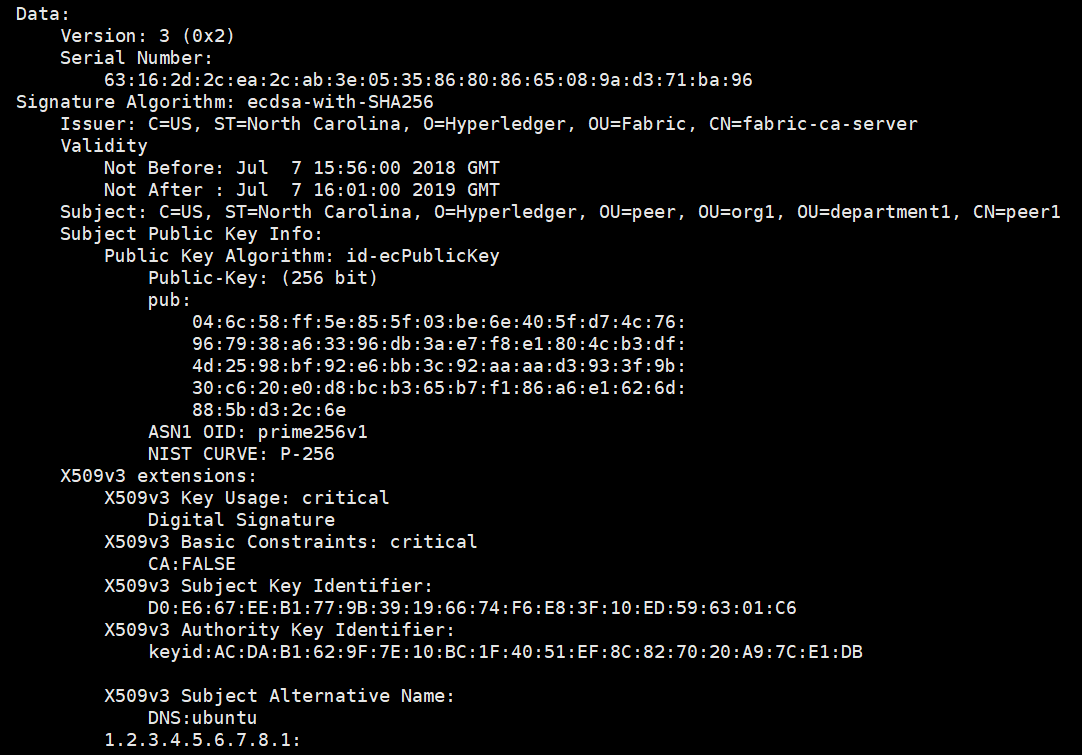
执行命令fabric-ca-client enroll -u http://peer1:tHsDdlWEUafA@localhost:7054 -M $FABRIC\_CA\_CLIENT\_HOME/msp，进行enroll，其中-M代表生成的msp的存储路径。



查看文件夹peer1的结构：

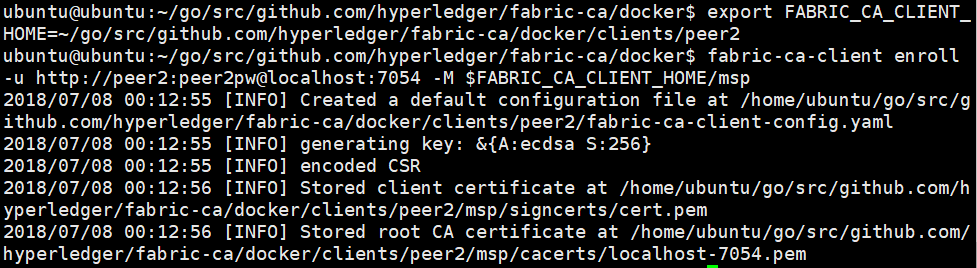


可以查看得知cacerts中存放的依然是上文的根证书。Signcerts中的证书cert.pem如下图所示：

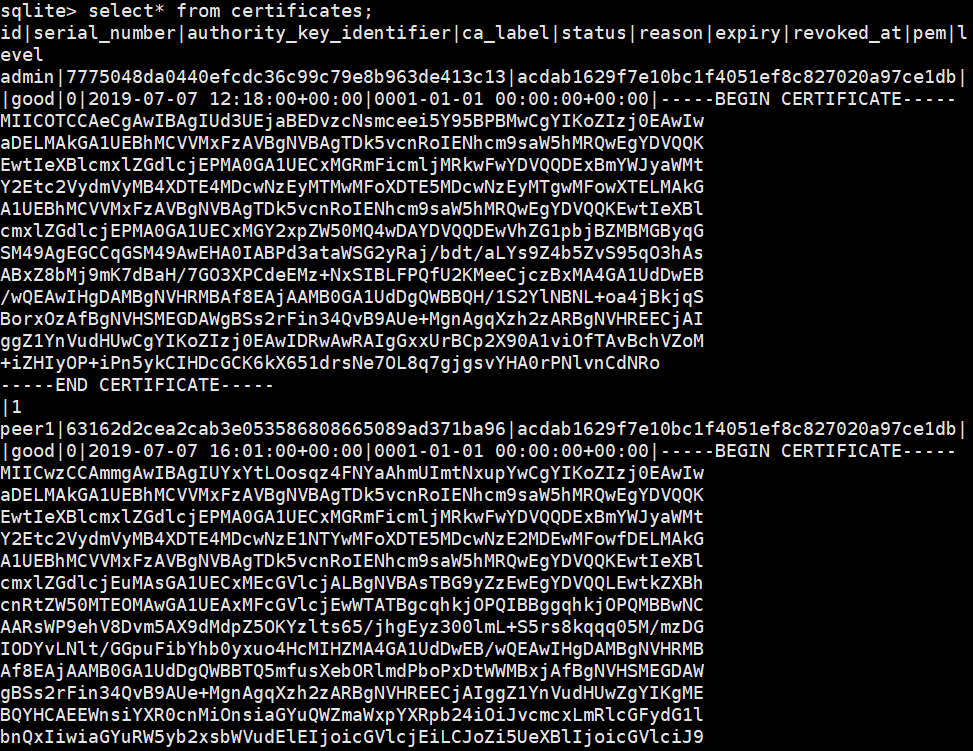


可以看到认证者依然是根证书。Organizational unit（OU）由type和affiliation定义，依次是peer, org1, department1.

同理enroll peer2.



然后查看ca-server数据库中的certificates表，可以看到此时有三个证书，而且可以看到他们的认证密钥都是根CA的密钥。





# 6.查看ca-server的日志记录

执行命令docker logs -f fabric-ca-server

