实验三 Fabric 搭建 peer 并加入通道

龚小航 PB18151866

【实验目的及要求】

在 Fabric 中，根据提供的服务不同，可以把服务节点分为三类：CA、Orderer 和 Peer。

CA：用于提供 Fabric 中组织成员的身份注册和证书颁发

Orderer：排序节点，搜集交易并排序出块，广播给其他组织的主节点

Peer：背书、验证和存储节点

本次实验需要使用 Fabric 搭建一个 peer 节点，并用这个 peer 节点加入已经创建的通道之中。

【实验原理】

根据实验文档配置

【实验平台】

- 远程服务器 222.195.70.190，Ubuntu 18.04.4 LTS
- terminal.icu 远程连接管理软件

【实验步骤】

1、 在 Fabric 客户端注册身份，申请证书：

使用虚拟机环境下预先配置好的 fabric-ca-client 工具，与 CA 服务交互得到身份证书。

其中 222.195.70.186 服务器上已经搭建了 CA 服务器和 Orderer 服务器,CA 服务端口号为 7054，orderer 服务端口号为 7050。

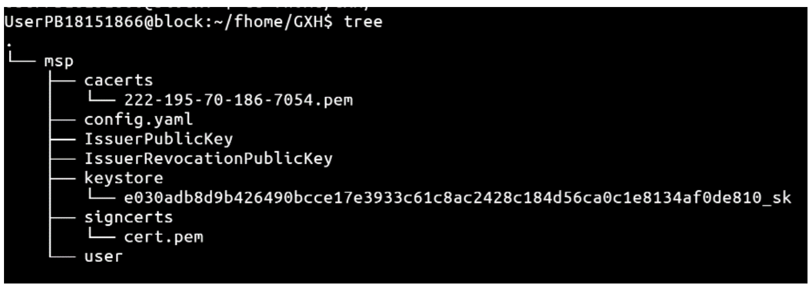
由于实验已经预先注册了一个用户 kecheng，它的 home 目录下有 CA 组织的 admin，直接将用 scp -r 指令将 ca-admin-msp 文件夹拷入本地 ～/ fhome 目录下，作为 CA 组织的 admin 证书，执行注册指令，用这个 CA 的 admin 证书注册在 fabric 组织中的 peer 证书，注意此时 peer 的身份必须是 admin，不能是普通的 peer：

```
fabric-ca-client  register  --id.name  GXH  --id.secret GXH  --id.type admin
                        -u  http://222.195.70.186: 7054   --mspdir  ~/ fhome / ca-admin-msp
```

再将得到的证书 enroll 到本地，完成证书的颁发：

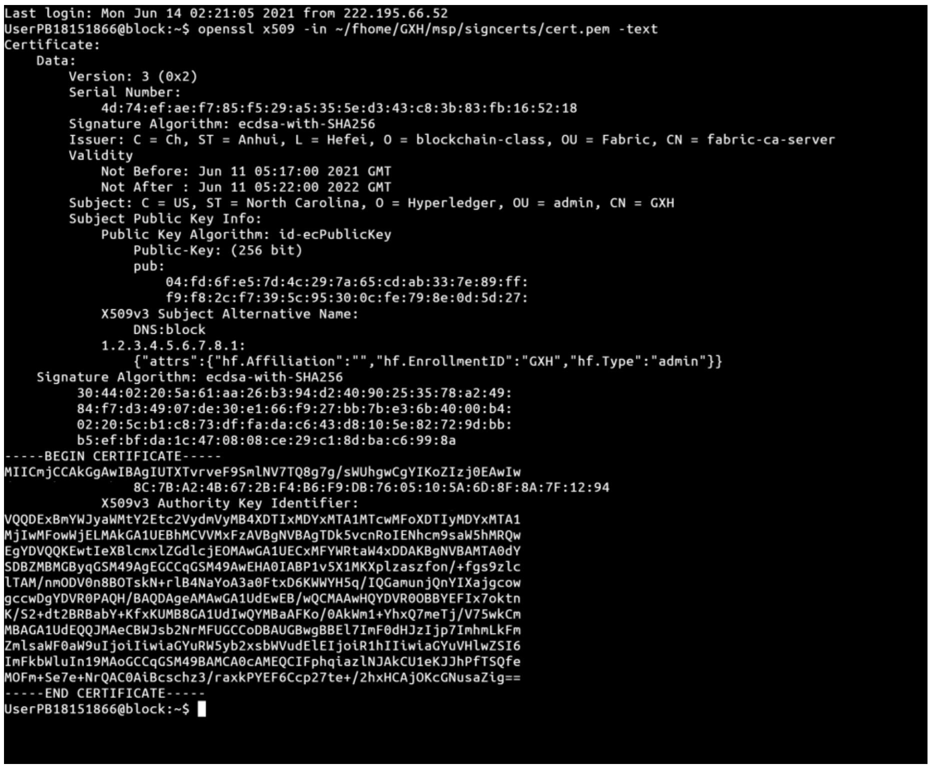
```
fabric - ca - client enroll  - u http://GXH: GXH@222.195.70.186: 7054  -- mspdir ~/fhome/GXH/
```

GXH 文件夹下的结构：



```
UserPB18151866@block:~/fhome/GXH$ tree
.
└── msp
    ├── cacerts
    │   └── 222-195-70-186-7054.pem
    ├── config.yaml
    ├── IssuerPublicKey
    ├── IssuerRevocationPublicKey
    ├── keystore
    │   └── e030adb8d9b426490bcce17e3933c61c8ac2428c184d56ca0c1e8134af0de810_sk
    ├── signcerts
    │   └── cert.pem
    └── user
```

查看注册完成的证书信息：



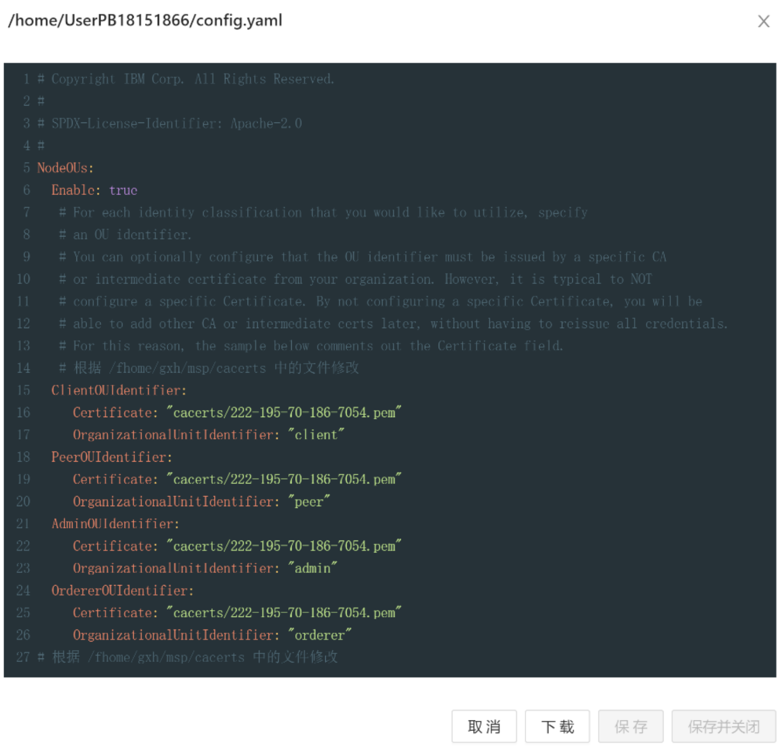
```
Last login: Mon Jun 14 02:21:05 2021 from 222.195.66.52
UserPB18151866@block:~$ openssl x509 -in ~/fhome/GXH/msp/signcerts/cert.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            4d:74:ef:ae:f7:85:f5:29:a5:35:5e:d3:43:c8:3b:83:fb:16:52:18
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = Ch, ST = Anhui, L = Hefei, O = blockchain-class, OU = Fabric, CN = fabric-ca-server
        Validity
            Not Before: Jun 11 05:17:00 2021 GMT
            Not After : Jun 11 05:22:00 2022 GMT
        Subject: C = US, ST = North Carolina, O = Hyperledger, OU = admin, CN = GXH
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:fd:6f:e5:7d:4c:29:7a:65:cd:ab:33:7e:89:ff:
                    f9:f8:2c:f7:39:5c:95:30:0c:fe:79:8e:0d:5d:27:
            X509v3 Subject Alternative Name:
                DNS:block
            1.2.3.4.5.6.7.8.1:
                {"attrs":{"hf.Affiliation":"","hf.EnrollmentID":"GXH","hf.Type":"admin"}}
    Signature Algorithm: ecdsa-with-SHA256
         30:44:02:20:5a:61:aa:26:b3:94:d2:40:90:25:35:78:a2:49:
         84:f7:d3:49:07:de:30:e1:66:f9:27:bb:7b:e3:6b:40:00:b4:
         02:20:5c:b1:c8:73:df:fa:da:c6:43:d8:10:5e:82:72:9d:bb:
         b5:ef:bf:da:1c:47:08:08:ce:29:c1:8d:ba:c6:99:8a
-----BEGIN CERTIFICATE-----
MIICmjCCAkGgAwIBAgIUTXTvrveF9SmlNV7TQ8g7g/sWUhgwCgYIKoZIzj0EAwIw
                8C:7B:A2:4B:67:2B:F4:B6:F9:DB:76:05:10:5A:6D:8F:8A:7F:12:94
            X509v3 Authority Key Identifier:
VQQDExBmYWJyaWMtY2Etc2VydmVyMB4XDTIxMDYxMTA1MTcwMFoXDTIyMDYxMTA1
MjIwMFowWjELMAkGA1UEBhMCVVMxFzAVBgNVBAgTDk5vcnRoIENhcm9saW5hMRQw
EgYDVQQKEwtIeXBlcmxlZGdlcjEOMAwGA1UECxMFYWRtaW4xDDAKBgNVBAMTA0dY
SDBZMBMGByqGSM49AgEGCCqGSM49AwEHA0IABP1v5X1MKXplzaszfon/+fgs9zlc
lTAM/nmODV0n8BOTskN+rlB4NaYoA3a0FtxD6KWWYH5q/IQGamunjQnYIXajgcow
gccwDgYDVR0PAQH/BAQDAgeAMAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFIx7oktn
K/S2+dt2BRBabY+KfxKUMB8GA1UdIwQYMBaAFKo/0AkWm1+YhxQ7meTj/V75wkCm
MBAGA1UdEQQJMAeCBWJsb2NrMFUGCCoDBAUGBwgBBEl7ImF0dHJzIjp7ImhmLkFm
ZmlsaWF0aW9uIjoiIiwiaGYuRW5yb2xsbWVudElEIjoiR1hIIiwiaGYuVHlwZSI6
ImFkbWluIn19MAoGCCqGSM49BAMCA0cAMEQCIFphqiazlNJAkCU1eKJJhPfTSQfe
MOFm+Se7e+NrQAC0AiBcschz3/raxkPYEF6Ccp27te+/2hxHCAjOKcGNusaZig==
-----END CERTIFICATE-----
UserPB18151866@block:~$ 
```

2、配置 peer 节点以启动

先将环境变量配置如下，设置 FABRIC_CFG_PATH 指向 core.yaml 所在文件夹：

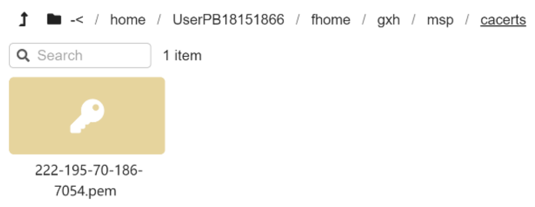
```
export FABRIC_CFG_PATH =/home/UserPB18151866
```

Msp 在 1.4.3 版本以上对身份的判断通过 OU 进行，OU 的设置在 config.yaml 文件内。先将实验框架提供的此文件复制至~目录下，将其内容修改如下：



```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
NodeOUs:
  Enable: true
   # For each identity classification that you would like to utilize, specify
   # an OU identifier.
   # You can optionally configure that the OU identifier must be issued by a specific CA
   # or intermediate certificate from your organization. However, it is typical to NOT
   # configure a specific Certificate. By not configuring a specific Certificate, you will be
   # able to add other CA or intermediate certs later, without having to reissue all credentials.
   # For this reason, the sample below comments out the Certificate field.
   # 根据 /fhome/gxh/msp/cacerts 中的文件修改
  ClientOUIdentifier:
    Certificate: "cacerts/222-195-70-186-7054.pem"
    OrganizationalUnitIdentifier: "client"
  PeerOUIdentifier:
    Certificate: "cacerts/222-195-70-186-7054.pem"
    OrganizationalUnitIdentifier: "peer"
  AdminOUIdentifier:
    Certificate: "cacerts/222-195-70-186-7054.pem"
    OrganizationalUnitIdentifier: "admin"
  OrdererOUIdentifier:
    Certificate: "cacerts/222-195-70-186-7054.pem"
    OrganizationalUnitIdentifier: "orderer"
# 根据 /fhome/gxh/msp/cacerts 中的文件修改
```

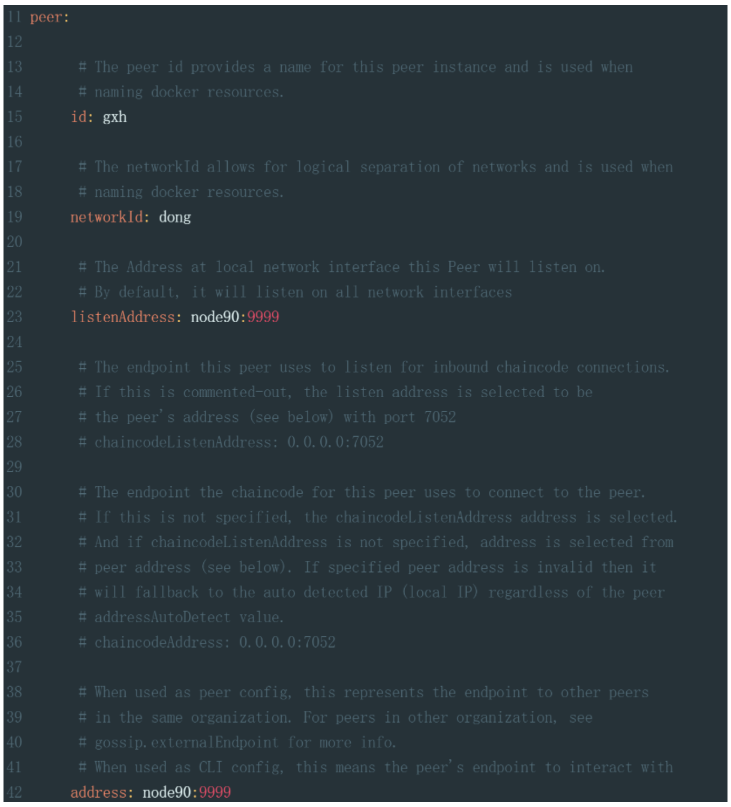
根据注册时生成的文件进行身份证书路径修改：



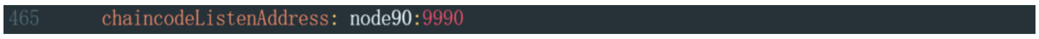
再将这个配置文件放在 ~/fhome/gxh/msp 目录下。

最后配置 core.yaml 文件：

首先将节点 id，侦听地址与端口更改为合适的值：



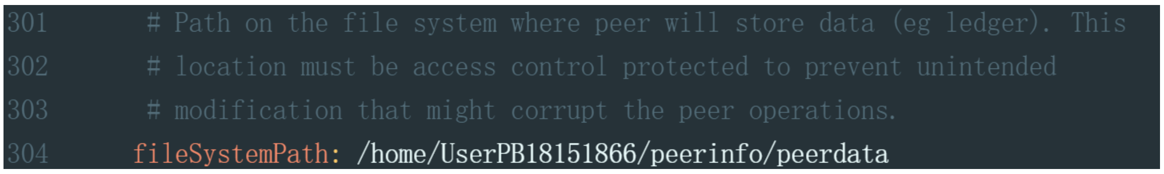
```
peer:

    # The peer id provides a name for this peer instance and is used when
    # naming docker resources.
    id: gxh

    # The networkId allows for logical separation of networks and is used when
    # naming docker resources.
    networkId: dong

    # The Address at local network interface this Peer will listen on.
    # By default, it will listen on all network interfaces
    listenAddress: node90:9999

    # The endpoint this peer uses to listen for inbound chaincode connections.
    # If this is commented-out, the listen address is selected to be
    # the peer's address (see below) with port 7052
    # chaincodeListenAddress: 0.0.0.0:7052

    # The endpoint the chaincode for this peer uses to connect to the peer.
    # If this is not specified, the chaincodeListenAddress address is selected.
    # And if chaincodeListenAddress is not specified, address is selected from
    # peer address (see below). If specified peer address is invalid then it
    # will fallback to the auto detected IP (local IP) regardless of the peer
    # addressAutoDetect value.
    # chaincodeAddress: 0.0.0.0:7052

    # When used as peer config, this represents the endpoint to other peers
    # in the same organization. For peers in other organization, see
    # gossip.externalEndpoint for more info.
    # When used as CLI config, this means the peer's endpoint to interact with
    address: node90:9999
```



```
    chaincodeListenAddress: node90:9990
```

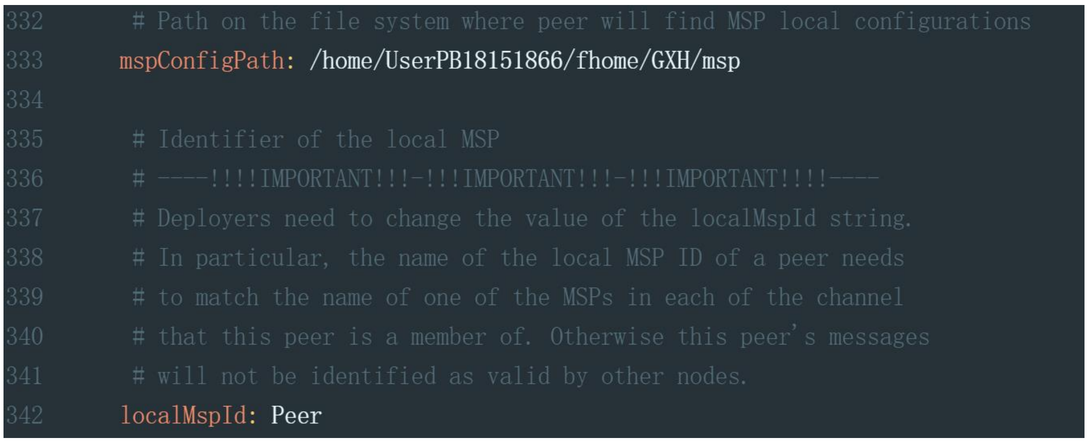
由于账号地址位于服务器 222.195.70.190 上，因此侦听地址为 222.195.70.190，等效于 node90;侦听端口只需要选择一个未被占用的端口即可，此处选择 9999 端口。

更改 peer 文件存储路径：



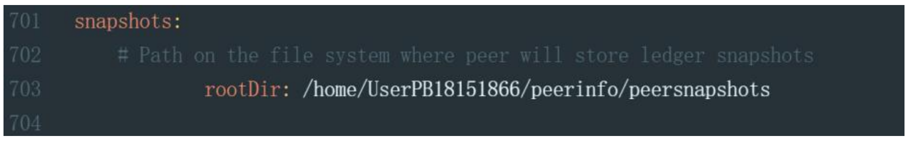
```
    # Path on the file system where peer will store data (eg ledger). This
    # location must be access control protected to prevent unintended
    # modification that might corrupt the peer operations.
   fileSystemPath: /home/UserPB18151866/peerinfo/peerdata
```

更改 msp 路径，指向刚注册出的 GXH 文件夹下的 msp 文件夹，同时更改 peer 节点组织名：



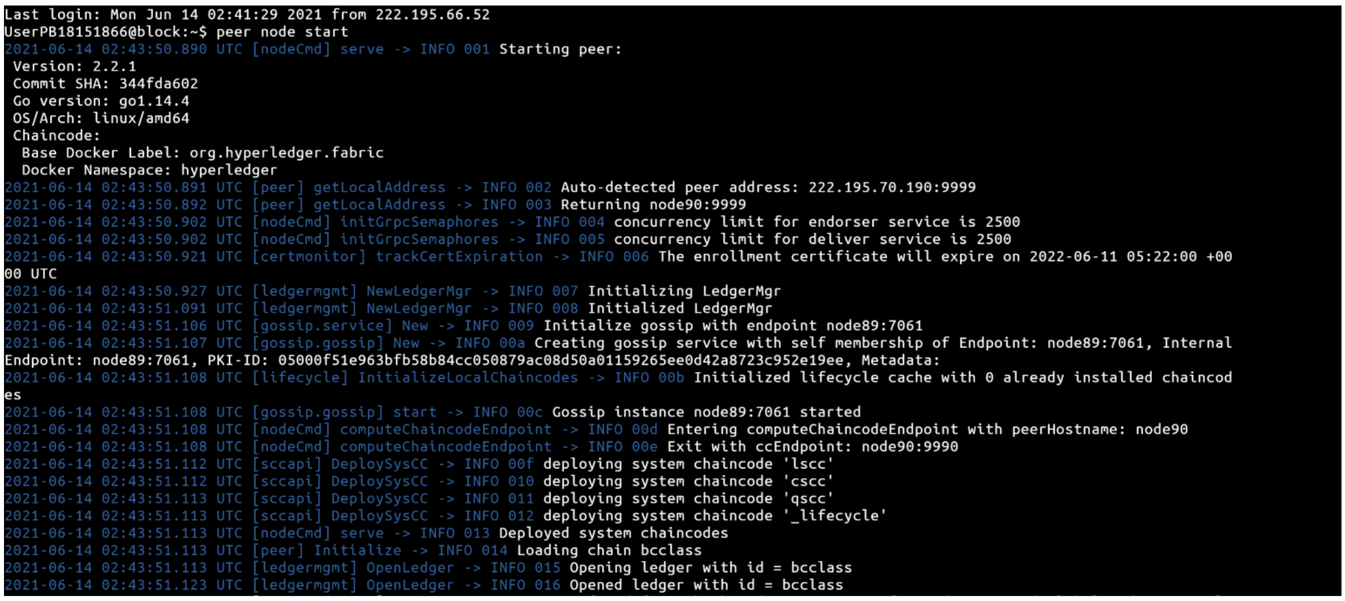
```
    # Path on the file system where peer will find MSP local configurations
   mspConfigPath: /home/UserPB18151866/fhome/GXH/msp

    # Identifier of the local MSP
    # ----!!!!IMPORTANT!!!-!!!IMPORTANT!!!-!!!IMPORTANT!!!!----
    # Deployers need to change the value of the localMspId string.
    # In particular, the name of the local MSP ID of a peer needs
    # to match the name of one of the MSPs in each of the channel
    # that this peer is a member of. Otherwise this peer's messages
    # will not be identified as valid by other nodes.
   localMspId: Peer
```

更改快照存储路径：



```
snapshots:
    # Path on the file system where peer will store ledger snapshots
        rootDir: /home/UserPB18151866/peerinfo/peersnapshots

```

全部更改完毕后，即可启动 peer 节点。



```
Last login: Mon Jun 14 02:41:29 2021 from 222.195.66.52
UserPB18151866@block:~$ peer node start
2021-06-14 02:43:50.890 UTC [nodeCmd] serve -> INFO 001 Starting peer:
 Version: 2.2.1
 Commit SHA: 344fda602
 Go version: go1.14.4
 OS/Arch: linux/amd64
 Chaincode:
  Base Docker Label: org.hyperledger.fabric
  Docker Namespace: hyperledger
2021-06-14 02:43:50.891 UTC [peer] getLocalAddress -> INFO 002 Auto-detected peer address: 222.195.70.190:9999
2021-06-14 02:43:50.892 UTC [peer] getLocalAddress -> INFO 003 Returning node90:9999
2021-06-14 02:43:50.902 UTC [nodeCmd] initGrpcSemaphores -> INFO 004 concurrency limit for endorser service is 2500
2021-06-14 02:43:50.902 UTC [nodeCmd] initGrpcSemaphores -> INFO 005 concurrency limit for deliver service is 2500
2021-06-14 02:43:50.921 UTC [certmonitor] trackCertExpiration -> INFO 006 The enrollment certificate will expire on 2022-06-11 05:22:00 +00
00 UTC
2021-06-14 02:43:50.927 UTC [ledgermgmt] NewLedgerMgr -> INFO 007 Initializing LedgerMgr
2021-06-14 02:43:51.091 UTC [ledgermgmt] NewLedgerMgr -> INFO 008 Initialized LedgerMgr
2021-06-14 02:43:51.106 UTC [gossip.service] New -> INFO 009 Initialize gossip with endpoint node89:7061
2021-06-14 02:43:51.107 UTC [gossip.gossip] New -> INFO 00a Creating gossip service with self membership of Endpoint: node89:7061, Internal
Endpoint: node89:7061, PKI-ID: 05000f51e963bfb58b84cc050879ac08d50a01159265ee0d42a8723c952e19ee, Metadata:
2021-06-14 02:43:51.108 UTC [lifecycle] InitializeLocalChaincodes -> INFO 00b Initialized lifecycle cache with 0 already installed chaincod
es
2021-06-14 02:43:51.108 UTC [gossip.gossip] start -> INFO 00c Gossip instance node89:7061 started
2021-06-14 02:43:51.108 UTC [nodeCmd] computeChaincodeEndpoint -> INFO 00d Entering computeChaincodeEndpoint with peerHostname: node90
2021-06-14 02:43:51.108 UTC [nodeCmd] computeChaincodeEndpoint -> INFO 00e Exit with ccEndpoint: node90:9990
2021-06-14 02:43:51.112 UTC [sccapi] DeploySysCC -> INFO 00f deploying system chaincode 'lscc'
2021-06-14 02:43:51.112 UTC [sccapi] DeploySysCC -> INFO 010 deploying system chaincode 'cscc'
2021-06-14 02:43:51.113 UTC [sccapi] DeploySysCC -> INFO 011 deploying system chaincode 'qscc'
2021-06-14 02:43:51.113 UTC [sccapi] DeploySysCC -> INFO 012 deploying system chaincode '_lifecycle'
2021-06-14 02:43:51.113 UTC [nodeCmd] serve -> INFO 013 Deployed system chaincodes
2021-06-14 02:43:51.113 UTC [peer] Initialize -> INFO 014 Loading chain bcclass
2021-06-14 02:43:51.113 UTC [ledgermgmt] OpenLedger -> INFO 015 Opening ledger with id = bcclass
2021-06-14 02:43:51.123 UTC [ledgermgmt] OpenLedger -> INFO 016 Opened ledger with id = bcclass
```

最后将 peer 加入通道：

首先获取配置区块：

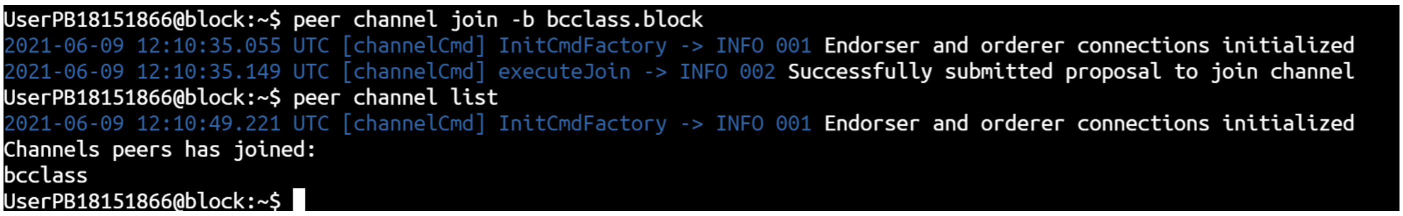
$$\text{peer channel fetch config bcclass.block} - \text{c bcclass} - -\text{orderer } 222.195.70.186{:}\ 7050$$

再将 peer 加入通道：

$$\text{peer channel join} - \text{b bcclass.block}$$

【实验结果】

最后加入通道成功后的截图：



```
UserPB18151866@block:~$ peer channel join -b bcclass.block
2021-06-09 12:10:35.055 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2021-06-09 12:10:35.149 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
UserPB18151866@block:~$ peer channel list
2021-06-09 12:10:49.221 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
Channels peers has joined:
bcclass
UserPB18151866@block:~$ 
```