Hyperledger-Fabric-CA

1. 基础环境

角色	IP地址	依赖环境
orderer1	192.168.3.15	go1.21.4+Docker-Composev2.23.3+Docker20.10.10+jq-1.7+git1.8.3.1+Hyperledger-FabricV2.5.4
orderer2	192.168.3.16	go1.21.4+Docker-Composev2.23.3+Docker20.10.10+jq-1.7+git1.8.3.1+Hyperledger-FabricV2.5.4
orderer3	192.168.3.17	go1.21.4+Docker-Composev2.23.3+Docker20.10.10+jq-1.7+git1.8.3.1+Hyperledger-FabricV2.5.4
peer1	192.168.3.18	go1.21.4+Docker-Composev2.23.3+Docker20.10.10+jq-1.7+git1.8.3.1+Hyperledger-FabricV2.5.4
peer2	192.168.3.19	go1.21.4+Docker-Composev2.23.3+Docker20.10.10+jq-1.7+git1.8.3.1+Hyperledger-FabricV2.5.4
peer3	192.168.3.21	go1.21.4+Docker-Composev2.23.3+Docker20.10.10+jq-1.7+git1.8.3.1+Hyperledger-FabricV2.5.4

1.1 安装git

```
yum -y install git
```

1.2 安装go

```
cd /usr/local/ && wget https://studygolang.com/dl/golang/go1.21.4.linux-amd64.tar.gz && tar -zxvf go1.21.4.linux-amd64.tar.gz && echo "export PATH=$PATH:/usr/local/go/bin" >>/etc/profile && echo "export GOROOT=/usr/local/go" >>/etc/profile && echo "export GOPATH=/root/go/" >>/etc/profile && source /etc/profile
```

1.3 安装docker

```
sudo yum install -y yum-utils
sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
sudo yum install docker-ce docker-ce-cli containerd.io
sudo systemctl start docker
```

1.4 安装docker-compose

```
 curl -L https://github.com/docker/compose/releases/download/v2.23.3/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose && chmod +x /usr/local/bin/docker-compose
```

1.5 安装jq

```
wget https://github.com/jqlang/jq/releases/download/jq-1.7/jq-linux-amd64 && mkdir jq && mv jq-linux-amd64 jq && mv jq/jq-linux-amd64 jq/jq && chmod +x jq/jq && echo "export PATH=$PATH:/usr/local/jq" >>/etc/profile
```

1.6 设置hosts

```
echo "192.168.3.15 orderer.example.com" >>/etc/hosts
echo "192.168.3.16 orderer2.example.com" >>/etc/hosts
echo "192.168.3.17 orderer3.example.com" >>/etc/hosts
echo "192.168.3.18 peer0.org1.example.com" >>/etc/hosts
echo "192.168.3.19 peer0.org2.example.com" >>/etc/hosts
scp -r /etc/hosts orderer2.example.com:/etc/hosts
scp -r /etc/hosts orderer3.example.com:/etc/hosts
scp -r /etc/hosts peer0.org1.example.com:/etc/hosts
scp -r /etc/hosts peer0.org1.example.com:/etc/hosts
```

2. 上传部署包

> 下载地址：https://deploystore.insightone.cn/middleware/Hyperledger-Fabric/
>
> 统一上传到/data/fabric

2.1 orderer1

https://deploystore.insightone.cn/middleware/Hyperledger-Fabric/fabric-ca-orderer1.zip

2.2 orderer2

https://deploystore.insightone.cn/middleware/Hyperledger-Fabric/fabric-ca-orderer2.zip

2.3 orderer3

https://deploystore.insightone.cn/middleware/Hyperledger-Fabric/fabric-ca-orderer3.zip

2.4 peer1

https://deploystore.insightone.cn/middleware/Hyperledger-Fabric/fabric-ca-peer1.zip

2.5 peer2

https://deploystore.insightone.cn/middleware/Hyperledger-Fabric/fabric-ca-peer2.zip

3. 导入镜像

> 下载地址：https://deploystore.insightone.cn/middleware/Hyperledger-Fabric/images.zip
>
> 所有角色节点

```
unzip -d /data images.zip
cd /data/ && sh load.sh
```

4. 启动网络

4.1 orderer1

```
cd /data/fabric/fabric-orderer1/test-network
[root@orderer1 test-network]# ./network.sh up -ca
```

将organizations传输给其余四台节点

```
scp -rq organizations orderer2.example.com:/data/fabric/fabric-orderer2/test-network
```

```
2 scp -rq organizations orderer3.example.com:/data/fabric/fabric-orderer3/test-
  network
3 scp -rq organizations peer0.org1.example.com:/data/fabric/fabric-peer1/test-
  network
4 scp -rq organizations peer0.org2.example.com:/data/fabric/fabric-peer2/test-
  network
```

4.2 orderer2

```
1 [root@orderer2 test-network]# ./network.sh up -ca
```

4.3 orderer3

```
1 [root@orderer3 test-network]# ./network.sh up -ca
```

4.4 peer1

```
1 [root@peer1 test-network]# ./network.sh up -ca
```

4.5 peer2

```
1 [root@peer2 test-network]# ./network.sh up -ca
```

5. Creating a channel

5.1 orderer1

```
1 ./network.sh createChannel -c carbonchain
```

将channel-artifacts传输给peer节点

```
scp -r channel-artifacts/ peer0.org1.example.com:/data/fabric/fabric-peer1/test-network
scp -r channel-artifacts/ peer0.org2.example.com:/data/fabric/fabric-peer2/test-network
```

5.2 peer1

```
./network.sh createChannel -c carbonchain
```

5.3 peer2

```
./network.sh createChannel -c carbonchain
```

6. Starting a chaincode on the channel

> 在任意peer节点执行

6.1 peer1

```
 ./network.sh deployCC -ccn carbon_exchange -ccp ../asset-transfer-basic/chaincode-go -ccl go -c carbonchain
```

7. Interacting with the network

7.1 peer1

```
export PATH=${PWD}/../bin:$PATH
export FABRIC_CFG_PATH=$PWD/../config/
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com
```

```
/users/Admin@org1.example.com/msp
export CORE_PEER_ADDRESS=localhost:7051
```

运行以下命令用一些资产来初始化账本：

```
peer chaincode invoke -o orderer.example.com:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C  carbonchain  -n carbon_exchange --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"function":"InitLedger","Args":[]}'
```

如果命令成功，您将观察到类似以下的输出：

```
-->INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
```

现在你可以用你的 CLI 工具来查询账本。运行以下指令来获取添加到通道账本的资产列表：

```
peer chaincode query -C carbonchain -n carbon_exchange -c '{"Args":["GetAllAssets"]}'
```

如果命令成功，您应该看到以下响应：

```
[{"AppraisedValue":300,"Color":"blue","ID":"asset1","Owner":"Tomoko","Size":5},{"AppraisedValue":400,"Color":"red","ID":"asset2","Owner":"Brad","Size":5},{"AppraisedValue":500,"Color":"green","ID":"asset3","Owner":"Jin Soo","Size":10},{"AppraisedValue":600,"Color":"yellow","ID":"asset4","Owner":"Max","Size":10},{"AppraisedValue":700,"Color":"black","ID":"asset5","Owner":"Adriana","Size":15},{"AppraisedValue":800,"Color":"white","ID":"asset6","Owner":"Michel","Size":15}]
```

当一个网络成员希望在账本上转一些或者改变一些资产，链码会被调用。使用以下的指令来通过调用 asset-transfer (basic) 链码改变账本上的资产所有者：

```
peer chaincode invoke -o orderer.example.com:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C  carbonchain  -n carbon_exchange  --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"function":"TransferAsset","Args":["asset6","Christopher"]}'
```

如果命令成功，您应该看到以下响应：

```
INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200 payload:"Michel"
```

7.2 peer2

调用链码之后，我们可以使用另一个查询来查看调用如何改变了区块链账本的资产。因为我们已经查询了 Org1 的 peer，我们可以把这个查询链码的机会通过 Org2 的 peer 来运行。设置以下的环境变量来操作 Org2：

```
export PATH=${PWD}/../bin:$PATH
export FABRIC_CFG_PATH=$PWD/../config/
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org2MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
export CORE_PEER_ADDRESS=localhost:9051
```

你可以查询运行在 `peer0.org2.example.com` asset-transfer (basic) 链码：

```
peer chaincode query -C  carbonchain  -n carbon_exchange  -c '{"Args":["ReadAsset","asset6"]}'
```

结果显示 `"asset6"` 转给了 Christopher:

```
{"AppraisedValue":800,"Color":"white","ID":"asset6","Owner":"Christopher","Size":15}
```

8. 关停网络

```
./network.sh down
docker system prune --volumes
```

该命令将停止并删除节点和链码容器，删除组织加密材料，并从Docker Registry移除链码镜像。 该命令还删除之前运行的通道项目和docker卷。如果您遇到任何问题，还允许您再次运行 `./network.sh up` 。

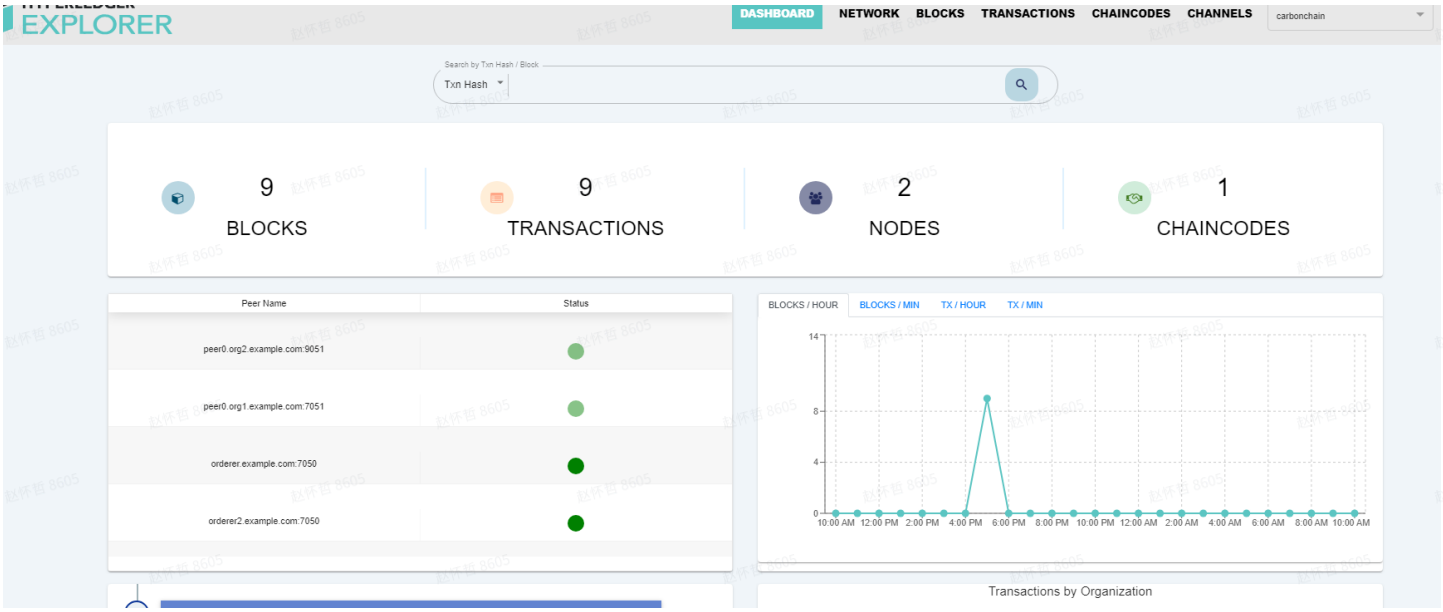
9. Hyperledger explorer

```
[root@orderer1 test-network]# cd explorer/
[root@orderer1 explorer]# cp -r ../organizations/ .
修改adminPrivateKey处
                "path": "/tmp/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp/keystore/每一次的都不一样_sk"
[root@orderer1 explorer]# docker-compose up -d
```

用户：exploreradmin

密码：exploreradminpw

访问地址：http://IP:8080/



10. addOrg3

10.1 上传部署包

10.1.1.1 peer3

https://deploystore.insightone.cn/middleware/Hyperledger-Fabric/fabric-ca-peer3.zip

10.2 Orderer1

```
scp -rq channel-artifacts/ organizations/
peer0.org3.example.com:/data/fabric/fabric-peer3/test-network
```

10.3 Peer3

10.3.1 Join

```
cd /data/fabric/fabric-peer3/test-network/addOrg3
./addOrg3.sh up -ca -c carbonchain
```

10.3.2 Set environment

```
cd /data/fabric/fabric-peer3/test-network
export PATH=${PWD}/../bin:$PATH
export FABRIC_CFG_PATH=$PWD/../config/
export CORE_PEER_TLS_ENABLED=true
```

```
export CORE_PEER_LOCALMSPID="Org3MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp
export CORE_PEER_ADDRESS=localhost:11051
```

10.3.3 打包 Basic 链码

```
peer lifecycle chaincode package basic.tar.gz --path ../asset-transfer-basic/chaincode-go/ --lang golang --label basic_1
scp -r basic.tar.gz peer0.org1.example.com:/data/fabric/fabric-peer1/test-network
scp -r basic.tar.gz peer0.org2.example.com:/data/fabric/fabric-peer2/test-network
```

10.3.4 安装链代码包

> 所有peer

```
[root@peer3 test-network]# peer lifecycle chaincode install basic.tar.gz
2024-03-13 11:29:56.985 CST 0001 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response:<status:200 payload:"\nHbasic_1:ef2394600055b69053a488d0ea2ac66bd544e93bb1b272a68d8860df5ac82c8c\022\007basic_1" >
2024-03-13 11:29:56.986 CST 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: basic_1:ef2394600055b69053a488d0ea2ac66bd544e93bb1b272a68d8860df5ac82c8c
```

10.3.5 批准 Basic 的链码定义为 Org3

```
[root@peer3 test-network]# peer lifecycle chaincode queryinstalled
Installed chaincodes on peer:
Package ID: basic_1:ef2394600055b69053a488d0ea2ac66bd544e93bb1b272a68d8860df5ac82c8c, Label: basic_1

[root@peer3 test-network]# export CC_PACKAGE_ID=basic_1:ef2394600055b69053a488d0ea2ac66bd544e93bb1b272a68d8860df5
```

```
ac82c8c
```

10.3.6 批准 Org3 基本链码的定义

```
peer lifecycle chaincode approveformyorg -o orderer.example.com:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" --channelID carbonchain --name basic --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1
```

10.4 peer1 批准

```
export PATH=${PWD}/../bin:$PATH
export FABRIC_CFG_PATH=$PWD/../config/
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
export CORE_PEER_ADDRESS=localhost:7051
peer lifecycle chaincode install basic.tar.gz
export CC_PACKAGE_ID=basic_1:ef2394600055b69053a488d0ea2ac66bd544e93bb1b272a68d8860df5ac82c8c
peer lifecycle chaincode approveformyorg -o orderer.example.com:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" --channelID carbonchain --name basic --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1
```

10.5 peer2 批准

```
export PATH=${PWD}/../bin:$PATH
export FABRIC_CFG_PATH=$PWD/../config/
export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org2MSP"
```

```
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
export CORE_PEER_ADDRESS=localhost:9051
peer lifecycle chaincode install basic.tar.gz
export CC_PACKAGE_ID=basic_1:ef2394600055b69053a488d0ea2ac66bd544e93bb1b272a68d8860df5ac82c8c
peer lifecycle chaincode approveformyorg -o orderer.example.com:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" --channelID carbonchain --name basic --version 1.0 --package-id $CC_PACKAGE_ID --sequence 1
```

10.6 peer3

将链码定义提交到通道，并完成实例化过程。确保在执行该命令时，指定了正确的链码名称、版本和序列号，并等待链码在通道上成功实例化。

```
peer lifecycle chaincode commit -o orderer.example.com:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" --channelID carbonchain --name basic --version 1.0 --sequence 1 --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt"
```

10.6.1 检查您批准的链码定义是否已提交到通道

```
[root@peer3 test-network]# peer lifecycle chaincode querycommitted --channelID carbonchain --name basic --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem"

Committed chaincode definition for chaincode 'basic' on channel 'carbonchain':
```

```
4 Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc,
  Approvals: [Org1MSP: true, Org2MSP: true, Org3MSP: true]
```

10.6.2 Org3 在批准提交给通道的链码定义后可以使用基本链码

> 链码定义使用默认的`背书(endorsement )策略`，这需要通道上的大多数组织对交易进行赞同。这意味着如果一个组织被添加到通道或从通道中删除，背书策略将自动更新。我们之前需要 Org1 和 Org2 的认可（二选二）。现在我们需要 Org1、Org2 和 Org3 中的两个组织的认可（3 个中的 2 个）。 查询账本确保链码已经在 Org3 节点运行。注意我们此时需要链码容器启动

用一些示例资产填充分类帐。我们将从Org2对等体和新的Org3对等体获得endorsements，以使endorsement策略得到满足:

```
1 [root@peer3 test-network]# peer chaincode invoke -o orderer.example.com:7050 --
  ordererTLSHostnameOverride orderer.example.com --tls --cafile
  "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example
  .com/msp/tlscacerts/tlsca.example.com-cert.pem" -C carbonchain  -n basic --
  peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles
  "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.examp
  le.com/tls/ca.crt" --peerAddresses peer0.org3.example.com:11051 --
  tlsRootCertFiles
  "${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.examp
  le.com/tls/ca.crt" -c '{"function":"InitLedger","Args":[]}'
2
3 2024-03-13 11:33:43.446 CST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery ->
  Chaincode invoke successful. result: status:200
```

10.6.3 您可以查询链码以确保 Org3 对等方已提交数据

```
1 peer chaincode query -C carbonchain -n basic -c '{"Args":["GetAllAssets"]}'
```