**多机多组织fabric操作文档**

**一、初始化fabric环境**

* 1. **启动Fabric环境的容器**

首先启动orderer节点，在orderer服务器上运行：

docker-compose -f docker-compose-orderer.yaml up –d

运行完毕后可以使用docker ps看到运行了一个名字为orderer.example.com的节点。 然后切换到peer0.org1.example.com服务器，启动本服务器的peer节点和cli，命令为：

docker-compose -f docker-compose-peer.yaml up –d

运行完毕后使用docker ps应该可以看到2个正在运行的容器。 接下来依次在另外3台服务器运行启动peer节点容器的命令：

docker-compose -f docker-compose-peer.yaml up –d

现在整个Fabric4+1服务器网络已经成型。

* 1. **创建channel**

切换到peer0.org1.example.com服务器上，使用该服务器上的cli来运行创建Channel和运行ChainCode的操作。首先进入cli容器：

docker exec -it cli bash

进入容器后我们可以看到命令提示变为： root@b41e67d40583:/opt/gopath/src/github.com/hyperledger/fabric/peer#

说明我们已经以root的身份进入到cli容器内部。 创建Channel的命令是peer channel create，我们前面创建Channel的配置区块时，指定了Channel的名字是mychannel，那么这里我们必须创建同样名字的Channel。

ORDERER\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem peer channel create -o orderer.example.com:7050 -c mychannel -f ./channel-artifacts/channel.tx --tls true --cafile $ORDERER\_CA

执行该命令后，系统会提示：

2017-08-29 20:36:47.486 UTC [channelCmd] readBlock -> DEBU 020 Received block:0

系统会在cli内部的当前目录创建一个mychannel.block文件，这个文件非常重要，接下来其他节点要加入这个Channel就必须使用这个文件。

* 1. **各个peer加入channel**

#peer1加入channel

peer channel join -b mychannel.block

#修改环境变量，使peer2加入channel

CORE\_PEER\_LOCALMSPID="Org2MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp CORE\_PEER\_ADDRESS=peer0.org2.example.com:7051 peer channel join -b mychannel.block

#修改环境变量，使peer3加入channel

CORE\_PEER\_LOCALMSPID="Org3MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp CORE\_PEER\_ADDRESS=peer0.org3.example.com:7051 peer channel join -b mychannel.block

#修改环境变量，使peer4加入channel

CORE\_PEER\_LOCALMSPID="Org4MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/peers/peer0.org4.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/users/Admin@org4.example.com/msp CORE\_PEER\_ADDRESS=peer0.org4.example.com:7051 peer channel join -b mychannel.block

* 1. **更新锚节点**

对于Org1来说，peer0.org1是锚节点，我们需要连接上它并更新锚节点，其他组织同理，peer0为锚节点：

#更新org1的锚节点

CORE\_PEER\_LOCALMSPID="Org1MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp CORE\_PEER\_ADDRESS=peer0.org1.example.com:7051 peer channel update -o orderer.example.com:7050 -c mychannel -f ./channel-artifacts/Org1MSPanchors.tx --tls true --cafile $ORDERER\_CA

#更新org2的锚节点

CORE\_PEER\_LOCALMSPID="Org2MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp CORE\_PEER\_ADDRESS=peer0.org2.example.com:7051 peer channel update -o orderer.example.com:7050 -c mychannel -f ./channel-artifacts/Org2MSPanchors.tx --tls true --cafile $ORDERER\_CA

#更新org3的锚节点

CORE\_PEER\_LOCALMSPID="Org3MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp CORE\_PEER\_ADDRESS=peer0.org3.example.com:7051 peer channel update -o orderer.example.com:7050 -c mychannel -f ./channel-artifacts/Org3MSPanchors.tx --tls true --cafile $ORDERER\_CA

#更新org4的锚节点

CORE\_PEER\_LOCALMSPID="Org4MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/peers/peer0.org4.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/users/Admin@org4.example.com/msp CORE\_PEER\_ADDRESS=peer0.org4.example.com:7051 peer channel update -o orderer.example.com:7050 -c mychannel -f ./channel-artifacts/Org4MSPanchors.tx --tls true --cafile $ORDERER\_CA

**二、链上代码的安装与运行**

以上，整个Fabric网络和Channel都准备完毕，接下来安装和运行ChainCode。链码放在

`/opt/gopath/src/github.com/hyperledger/fabric/examples/chaincode/go/mychaincode`

目录下。

**2.1 Install ChainCode安装链上代码**

#在peer1上安装链码

CORE\_PEER\_LOCALMSPID="Org1MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp CORE\_PEER\_ADDRESS=peer0.org1.example.com:7051 peer chaincode install -n mycc -v 1.0 -p github.com/hyperledger/fabric/examples/chaincode/go/mychaincode

#在peer2上安装链码

CORE\_PEER\_LOCALMSPID="Org2MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp CORE\_PEER\_ADDRESS=peer0.org2.example.com:7051 peer chaincode install -n mycc -v 1.0 -p github.com/hyperledger/fabric/examples/chaincode/go/mychaincode

#在peer3上安装链码

CORE\_PEER\_LOCALMSPID="Org3MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp CORE\_PEER\_ADDRESS=peer0.org3.example.com:7051 peer chaincode install -n mycc -v 1.0 -p github.com/hyperledger/fabric/examples/chaincode/go/mychaincode

#在peer4上安装链码

CORE\_PEER\_LOCALMSPID="Org4MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/peers/peer0.org4.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/users/Admin@org4.example.com/msp CORE\_PEER\_ADDRESS=peer0.org4.example.com:7051 peer chaincode install -n mycc -v 1.0 -p github.com/hyperledger/fabric/examples/chaincode/go/mychaincode

**2.2 Instantiate ChainCode 实例化链上代码**

运行以下命令完成实例化：

peer chaincode instantiate -o orderer.example.com:7050 --tls true --cafile $ORDERER\_CA -C mychannel -n mycc -v 1.0 -c '{"Args":[]}' -P "OR ('Org1MSP.member','Org2MSP.member','Org3MSP.member','Org4MSP.member')"

**2.3升级链码**

当需要对链码进行修改时，可以安装新的链码并进行升级。 注意：如果直接修改存放链码的目录或者修改链码文件本身，可能运行docker时链码并不会被修改，本人也很困惑。这种方法目前是可行的修改链码的方式。

**2.3.1 安装新版本的链码**

下面的命令跟之前的安装链码的命令相比，修改了-p后链码的路径，和-v后链码的版本，这里也说明1.0版本和1是不同的。

#在peer1上安装新版本的链码

CORE\_PEER\_LOCALMSPID="Org1MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp CORE\_PEER\_ADDRESS=peer0.org1.example.com:7051 peer chaincode install -n mycc -v 1 -p github.com/hyperledger/fabric/examples/chaincode/go/algoblu

#在peer2上安装新版本的链码

CORE\_PEER\_LOCALMSPID="Org2MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp CORE\_PEER\_ADDRESS=peer0.org2.example.com:7051 peer chaincode install -n mycc -v 1 -p github.com/hyperledger/fabric/examples/chaincode/go/algoblu

#在peer3上安装新版本的链码

CORE\_PEER\_LOCALMSPID="Org3MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp CORE\_PEER\_ADDRESS=peer0.org3.example.com:7051 peer chaincode install -n mycc -v 1 -p github.com/hyperledger/fabric/examples/chaincode/go/algoblu

#在peer4上安装新版本的链码

CORE\_PEER\_LOCALMSPID="Org4MSP" CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/peers/peer0.org4.example.com/tls/ca.crt CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org4.example.com/users/Admin@org4.example.com/msp CORE\_PEER\_ADDRESS=peer0.org4.example.com:7051 peer chaincode install -n mycc -v 1 -p github.com/hyperledger/fabric/examples/chaincode/go/algoblu

**2.3.2 升级链码**

升级链码的命令和初始化类似，同样需要指定背书策略。

peer chaincode upgrade -o orderer.example.com:7050 --tls true --cafile $ORDERER\_CA -C mychannel -n mycc -v 1 -c '{"Args":[]}' -P "OR ('Org1MSP.member','Org2MSP.member','Org3MSP.member','Org4MSP.member')"

**2.4 调用链码**

接下来按顺序发布合约、响应合约、达成合约、关闭合约、查询合约等即可。

示例如下：

发布合约：

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile $ORDERER\_CA -C mychannel -n mycc -v 1 -c '{"Args":["link\_contract\_create","{\"task\_id\": \"1522112000\",\"user\_code\": \"CMCC\",\"contract\_code\":\"lc1522512008\",\"purchaser\_user\_code\":\"CMCC\",\"biding\_start\_time\": \"1525104000\",\"biding\_end\_time\":\"1525881600\",\"deal\_time\":\"\",\"close\_time\":\"\",\"contract\_start\_time\": \"1527782400\",\"contract\_end\_time\":\"1559318400\",\"link\_start\": \"Beijing\",\"link\_end\": \"NewYork\",\"path\_list\":[{\"path\_start\": \"Beijing\",\"path\_end\":\"Tokyo\",\"path\_bandwidth\":\"50Mbps\",\"path\_packet\_loss\_rate\": \"5%\",\"path\_avg\_rtt\": \"200ms\",\"path\_resource\_list\":[]},{\"path\_start\": \"Tokyo\",\"path\_end\":\"NewYork\",\"path\_bandwidth\":\"50Mbps\",\"path\_packet\_loss\_rate\": \"5%\",\"path\_avg\_rtt\": \"200ms\",\"path\_resource\_list\":[]}],\"contract\_status\":\"yes\"}"]}' -C mychannel

响应合约：

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile $ORDERER\_CA -n mycc -v 1 -c '{"Args":["link\_contract\_biding","{\"task\_id\": \"1522112006\",\"user\_code\": \"CTCC\",\"contract\_code\": \"lc1522512008\",\"path\_list\": [{\"path\_start\": \"Beijing\",\"path\_end\": \"Tokyo\",\"path\_bandwidth\": \"50Mbps\",\"path\_packet\_loss\_rate\": \"5%\",\"path\_avg\_rtt\": \"200ms\",\"path\_resource\_list\": [{\"supplyer\_user\_code\": \"CTCC\",\"path\_bandwidth\": \"100Mbps\",\"path\_packet\_loss\_rate\": \"5%\",\"path\_avg\_rtt\": \"200ms\",\"path\_bandwidth\_unit\_price\": \"200yuan\",\"path\_deal\_status\": \"\"}]},{\"path\_start\": \"Tokyo\",\"path\_end\": \"NewYork\",\"path\_bandwidth\": \"50Mbps\",\"path\_packet\_loss\_rate\": \"5%\",\"path\_avg\_rtt\": \"200ms\",\"path\_resource\_list\": [{\"supplyer\_user\_code\": \"CTCC\",\"path\_bandwidth\": \"100Mbps\",\"path\_packet\_loss\_rate\": \"5%\",\"path\_avg\_rtt\": \"200ms\",\"path\_bandwidth\_unit\_price\": \"200yuan\",\"path\_deal\_status\": \"\"}]}],\"contract\_status\": \"\"}"]}' -C mychannel

达成合约：

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile $ORDERER\_CA -C mychannel -n mycc -v 1.0 -c'{"Args":["link\_contract\_deal","{\"task\_id\": \"1522112000\",\"user\_code\": \"CMCC\",\"contract\_code\":\"lc1522512008\",\"deal\_time\":\"\",\"path\_list\":[{\"path\_start\": \"Beijing\",\"path\_end\":\"Tokyo\",\"path\_resource\_list\":[{\"supplyer\_user\_code\": \"CTCC\",\"path\_deal\_status\": \"yes\"}]},{\"path\_start\": \"Tokyo\",\"path\_end\":\"NewYork\",\"path\_resource\_list\":[{\"supplyer\_user\_code\": \"algoblu\",\"path\_deal\_status\": \"yes\"}]}],\"contract\_status\":\"deal\"}"]}'

关闭合约：

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile $ORDERER\_CA -C mychannel -n mycc -v 1.0 -c'{"Args":["link\_contract\_close","{\"task\_id\": \"1522112000\",\"user\_code\": \"CMCC\",\"contract\_code\":\"lc1522512008\",\"close\_time\": \"1525881600\",\"contract\_status\":\"close\"}"]}'

查询合约：

peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile $ORDERER\_CA -C mychannel -n mycc -v 1 -c '{"Args":["query","{\"task\_id\": \"1527228705614\",\"user\_code\": \"CMCC\",\"contract\_code\": \"lc1522512008\",\"version\_type\": \"last\"}"]}'