联盟链基础平台部署手册

# 介绍

Hyperledger Fabric是一个开源的企业级许可分布式账本技术平台，即联盟链，专为在企业环境下使用。Fabric的重要特性包括高度模块化和可配置架构、通用智能合约编程、可拔插共识和无需原生加密货币的共识协议等。基于上述特性，Fabric被广泛应用于银行、金融、保险、供应链等领域。如蚂蚁链的智慧金融服务，华为的智慧物流解决方案等，“区块链+网络货运”平台借鉴了一些优秀项目的思路，也采用了Fabric作为底层的联盟链平台。因此，“区块链+网络货运”平台是一个高度可扩展的系统，该平台是支持灵活的信任假设的联盟链平台。

本文档由华东师范大学软件学院以陈志立教授为代表的区块链项目研究团队整理编写，当前对应的Fabric版本为v2.2，默认使用的共识算法为Raft共识方法。

# 运行环境

在开始之前，我们需要检查以下所有先决条件是否已满足在您将运行Hyperledger Fabric的平台上。

## 1.1 安装docker

采用镜像的方式安装fabric，可以避免环境差异问题，也是fabric推荐的安装方式。Docker的版本无特殊要求，安装最新版即可。执行脚本如下：

1. # 卸载旧版本
2. sudo yum remove docker
3. # 安装基础依赖
4. sudo yum install -y yum-utils device-mapper-persistent-data lvm2
5. # 添加源
6. sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
7. # 更新yum包索引
8. yum makecache fast
9. # 安装dockerCE
10. sudo yum install -y docker-ce
11. # 将当前用户加入docker用户组以获得权限
12. sudo groupadd docker     # 添加docker用户组
13. sudo gpasswd -a $USER docker     # 将登陆用户加入到docker用户组中
14. newgrp docker     # 更新用户组
15. # 启停docker服务
16. sudo systemctl start docker.service
17. sudo systemctl stop docker.service
18. # 卸载
19. sudo yum remove docker-ce
20. # 删除image、容器、卷
21. sudo rm -rf /var/lib/docker
22. sudo systemctl restart docker

## 1.2 安装docker-compose

通过compose，可以使用指定配置文件配置较为复杂的服务，可以一键启动fabric各节点，便于开发调试。版本无特殊要求，安装最新版本即可。

1. # 下载compose，如果找不到，可以先执行 sudo yum install -y epel-release
2. sudo yum install -y docker-compose

## 1.3 下载fabric镜像

首先下载fabric源码，源码内含有拉取镜像的脚本，采用fabric2.4版本。

1. git clone git://github.com/hyperledger/fabric.git
2. git checkout -b release-2.4 remotes/origin/release-2.4

镜像脚本位于scripts目录下，修改脚本，更改其中的控制变量，使其只下载镜像。

1. vi $GOPATH/src/github.com/hyperledger/fabric/scripts/bootstrap.sh
2. DOCKER=**true**
3. SAMPLES=**false**
4. BINARIES=**false**

执行镜像脚本，拉取fabric2.4相关镜像

1. sh bootstrap.sh

## 1.4 配置网络

本节的目的是配置网络使得不同主机上的容器可以互相访问，此处采用修改hosts文件的方法绑定ip地址，实际生产中根据需要自行配置

1. sudo vi /etc/hosts
2. 192.168.1.61 orderer1.blockchain.com
3. 192.168.1.62 orderer2.blockchain.com
4. 192.168.1.63 orderer3.blockchain.com
5. 192.168.1.61 peer0.org1.blockchain.com
6. 192.168.1.62 peer0.org2.blockchain.com
7. 192.168.1.63 peer0.org3.blockchain.com

启动网卡转发

1. sudo sysctl -w net.ipv4.ip\_forward=1

使网络修改生效

1. sudo /etc/init.d/network restart

# 2 获取fabric生成工具

本节通过编译fabric源码获得生成工具，fabric生成工具可以根据对应配置文件生成创世区块、通道、节点证书等。

## 2.1 安装git

版本控制工具

1. sudo yum -y install git

## 2.2 安装go

版本无特殊要求，安装最新版本即可

1. sudo yum install -y epel-release
2. sudo yum install -y golang

配置环境变量

1. vi ~/.bashrc
2. # 安装目录
3. export GOROOT=/usr/lib/golang
4. # 工作区
5. export GOPATH=/home/ccccit/golang
6. # 禁止强制使用go mod
7. export GO111MODULE=auto

使生效

1. source ~/.bashrc

## 2.3 下载fabric源码

设置下载目录，除非熟悉fabric配置，否则不要改路径

1. mkdir -p $GOPATH/src/github.com/hyperledger

下载

1. cd $GOPATH/src/github.com/hyperledger
2. git clone git://github.com/hyperledger/fabric.git

## 2.4 编译生成fabric生成工具

本节要生成的fabric生成工具共有两个，其一是cryptogen，用来生成证书，其二是configtxgen，用来生成创世块和通道等

安装编译工具

1. sudo yum install libtool
2. sudo yum install libtool-ltdl-devel

生成cryptogen

1. cd $GOPATH/src/github.com/hyperledger/fabric
2. make cryptogen
3. # 生成位置 build/bin/

生成configtxgen

1. cd $GOPATH/src/github.com/hyperledger/fabric
2. make configtxgen
3. # 生成位置 build/bin/

# 3 编写fabric配置文件

fabric配置文件有专门的语法，可以在其官网上查询。如希望快速开发测试，可以找一个示例文件进行调整

1. # 调整bootstrap.sh变量，令SAMPLES=**true**，其余为**false**
2. sh bootstrap.sh
3. cd fabric-samples
4. git checkout -b release-2.2 remotes/origin/release-2.2
5. cd fabric-samples/interest\_rate\_swaps/network
6. # 拷贝crypto-config.yaml、configtx.yaml

crypto-config.yaml定义组织与节点。具体内容见附件。

configtx.yaml主要定义创世块配置信息，其中包括：共识规则、组织通信协议等。具体内容见附件。

完成配置文件的编写后，开始生成fabric相关文件并分发到各节点对应机器上。

1. # 生成证书文件
2. ./cryptogen generate --config=./crypto-config.yaml
4. # 生成创世区块文件，并指定创世块通道id
5. # 将configtxgen与configtx.yaml放到同一目录下
6. ./configtxgen -profile BeginGenesis -channelID channel0 -outputBlock ./channel-artifacts/genesis.block
8. # 生成通道
9. ./configtxgen -profile BeginChannel -outputCreateChannelTx ./channel-artifacts/channel1.tx -channelID channel1
11. # 分发
12. tar zcvf all.tar.gz channel-artifacts/\* crypto-config/\* # 正式环境下，这里只应该拷贝各自节点的文件
13. # 通过文件管理传递all.tar.gz
14. # 解压 tar zxvf all.tar.gz

# 4 启动并执行链码

## 4.1 启动

由于采用了docker-compose，这里只需要执行对应配置文件即可，此处列出了其中一个机器上的节点启动和停止的指令。配置文件具体内容见附件

1. # docker-compose-orderer.yaml
2. docker-compose -f ./docker-compose-orderer61.yaml up -d
3. docker-compose -f ./docker-compose-orderer61.yaml down -v
4. # docker-compose-cli.yaml
5. docker-compose -f ./docker-compose-cli61.yaml up -d
6. docker-compose -f ./docker-compose-cli61.yaml down -v

## 4.2 创建通道

根据配置文件创建新通道，在其中一个机器上执行即可

1. docker exec -it cli bash # 进入cli容器
2. peer channel create -o orderer1.blockchain.com:7050 -c channel1 -f ./channel-artifacts/channel1.tx --tls **true** --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/blockchain.com/orderers/orderer1.blockchain.com/msp/tlscacerts/tlsca.blockchain.com-cert.pem

加入通道

1. peer channel join -b channel1.block
2. peer channel list

拷贝通道文件到其他机器下

1. # 拷出容器
2. sudo docker cp 容器id:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel1.block ./
3. # 拷入容器。或者直接把block文件放到channel-artifacts文件夹里，因为docker配置文件中配置了路径映射，容器里也可以从这个文件夹拿到
4. sudo docker cp ./channel1.block c2220953c219:/opt/gopath/src/github.com/hyperledger/fabric/peer/

其他节点按照之前指令加入通道即可

## 4.3 添加锚节点

生成并添加锚节点

1. ./configtxgen -profile BeginChannel -outputAnchorPeersUpdate ./channel-artifacts/org1anchors.tx -channelID channel1 -asOrg org1
2. ./configtxgen -profile BeginChannel -outputAnchorPeersUpdate ./channel-artifacts/org2anchors.tx -channelID channel1 -asOrg org2
3. ./configtxgen -profile BeginChannel -outputAnchorPeersUpdate ./channel-artifacts/org3anchors.tx -channelID channel1 -asOrg org3
4. peer channel update -o orderer1.blockchain.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/blockchain.com/orderers/orderer1.blockchain.com/msp/tlscacerts/tlsca.blockchain.com-cert.pem -c channel1 -f ./channel-artifacts/org1anchors.tx
5. peer channel update -o orderer2.blockchain.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/blockchain.com/orderers/orderer2.blockchain.com/msp/tlscacerts/tlsca.blockchain.com-cert.pem -c channel1 -f ./channel-artifacts/org2anchors.tx
6. peer channel update -o orderer3.blockchain.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/blockchain.com/orderers/orderer3.blockchain.com/msp/tlscacerts/tlsca.blockchain.com-cert.pem -c channel1 -f ./channel-artifacts/org3anchors.tx

## 4.4 安装链码

链码编写可以使用多种语言，此处以samples提供的go编写的fabcar为例

1. # 将链码源码传入容器，并打包
2. peer lifecycle chaincode **package** fabcar.tar.gz --path ./fabric-chaincode-go-main --lang golang --label chaincode1 >&log.txt
3. # 如果报下载失败，可以考虑更换代理
4. go env -w GOPROXY=https://goproxy.io,direct
5. go env -w GO111MODULE=on

安装链码，每个组织都要安装，并获得返回id

1. peer lifecycle chaincode install fabcar.tar.gz >&log.txt
2. # 查询已安装链码
3. peer lifecycle chaincode queryinstalled

审批链码。每个组织都要审批

1. peer lifecycle chaincode approveformyorg --tls **true** --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/blockchain.com/orderers/orderer1.blockchain.com/msp/tlscacerts/tlsca.blockchain.com-cert.pem --channelID channel1 --name transport07 --version 1 --**package**-id transport07:a51752047b8b17436862f7ddb0649904f14a45eb1a7cc8ae0aa8e9688cbad9b6 --sequence 1 --waitForEvent --signature-policy "OR('org1.member','org2.member','org3.member')" >&log.txt
2. # 查看批准情况
3. peer lifecycle chaincode checkcommitreadiness --channelID channel1 --name transport07 --version 1 --sequence 1 -o -orderer1.blockchain.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/blockchain.com/orderers/orderer1.blockchain.com/msp/tlscacerts/tlsca.blockchain.com-cert.pem

提交链码。任意一个组织执行即可

1. peer lifecycle chaincode commit -o orderer1.blockchain.com:7050 --ordererTLSHostnameOverride orderer1.blockchain.com --channelID channel1 --name transport07 --version 1 --sequence 1 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/blockchain.com/orderers/orderer1.blockchain.com/msp/tlscacerts/tlsca.blockchain.com-cert.pem --peerAddresses peer0.org1.blockchain.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.blockchain.com/peers/peer0.org1.blockchain.com/tls/ca.crt --peerAddresses peer0.org2.blockchain.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.blockchain.com/peers/peer0.org2.blockchain.com/tls/ca.crt --signature-policy "OR('org1.member','org2.member','org3.member')" >&log.txt
2. # 查看提交情况
3. peer lifecycle chaincode querycommitted --channelID channel1 --name transport07

链码业务执行：查询

1. peer chaincode invoke -o orderer1.blockchain.com:7050 --ordererTLSHostnameOverride orderer1.blockchain.com --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/blockchain.com/orderers/orderer1.blockchain.com/msp/tlscacerts/tlsca.blockchain.com-cert.pem --channelID channel1 --name transport07 --peerAddresses peer0.org1.blockchain.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.blockchain.com/peers/peer0.org1.blockchain.com/tls/ca.crt --peerAddresses peer0.org2.blockchain.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.blockchain.com/peers/peer0.org2.blockchain.com/tls/ca.crt --peerAddresses peer0.org3.blockchain.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.blockchain.com/peers/peer0.org3.blockchain.com/tls/ca.crt  -c '{"function":"GetAllTmsOrder","Args":[]}'

链码业务执行：增加

1. export TMS\_ORDER\_INFO=$(echo -n "{\"object\_type\":\"tms\_order\_info\",\"consignorPrice\":\"100\",\"salt\":\"c94a8fe5ccb19ba61c4c0873d391e987982fbbd4\"}" | base64 | tr -d \\n)
2. peer chaincode invoke -o orderer1.blockchain.com:7050 --ordererTLSHostnameOverride orderer1.blockchain.com --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/blockchain.com/orderers/orderer1.blockchain.com/msp/tlscacerts/tlsca.blockchain.com-cert.pem --channelID channel1 --name transport07 --peerAddresses peer0.org1.blockchain.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.blockchain.com/peers/peer0.org1.blockchain.com/tls/ca.crt --peerAddresses peer0.org2.blockchain.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.blockchain.com/peers/peer0.org2.blockchain.com/tls/ca.crt --peerAddresses peer0.org3.blockchain.com:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org3.blockchain.com/peers/peer0.org3.blockchain.com/tls/ca.crt  -c '{"function":"createTmsOrder","Args":["ZJZY","yanghao001","weijian","shanghai","shenzhen","20220909","1000"]}' --**transient** "{\"tms\_order\_info\":\"$TMS\_ORDER\_INFO\"}"

# 5 使用Java调用链码

加载密钥用于权限认证

1. // 加载本地密钥用于权限认证
2. **public** **static** **void** populateWallet(Wallet wallet) **throws** InvalidKeyException, CertificateException, IOException {
3. String certPathAdmin = "/home/ccccit/crypto-config/peerOrganizations/org1.blockchain.com/ca/ca.org1.blockchain.com-cert.pem";
4. String keyPathAdmin = "/home/ccccit/crypto-config/peerOrganizations/org1.blockchain.com/ca/priv\_sk";
5. X509Certificate certAdmin = Identities.readX509Certificate(**new** PemReader(**new** FileReader(certPathAdmin)));
6. PrivateKey keyAdmin = Identities.readPrivateKey(**new** PemReader(**new** FileReader(keyPathAdmin)));
7. X509Identity identityAdmin = Identities.newX509Identity("org1", certAdmin, keyAdmin);
8. wallet.put("admin", identityAdmin);
10. String certPath = "/home/ccccit/crypto-config/peerOrganizations/org1.blockchain.com/users/User1@org1.blockchain.com/msp/signcerts/User1@org1.blockchain.com-cert.pem";
11. String keyPath = "/home/ccccit/crypto-config/peerOrganizations/org1.blockchain.com/users/User1@org1.blockchain.com/msp/keystore/priv\_sk";
12. X509Certificate cert = Identities.readX509Certificate(**new** PemReader(**new** FileReader(certPath)));
13. PrivateKey key = Identities.readPrivateKey(**new** PemReader(**new** FileReader(keyPath)));
14. X509Identity identity = Identities.newX509Identity("org1", cert, key);
15. wallet.put("appUser", identity);
16. }

建立连接，用于执行链码

1. // 建立连接，用于执行链码
2. **public** **static** Gateway connect() **throws** Exception {
3. // 创建钱包，加载用户密钥
4. Path walletPath = Paths.get("/home", "ccccit", "crypto-config", "wallet");
5. Wallet wallet = Wallets.newFileSystemWallet(walletPath);
6. populateWallet(wallet);
7. // 网络连接，加载fabric信息与peer节点密钥
8. Path networkConfigPath = Paths.get("/home", "ccccit", "crypto-config", "connection.yaml");
9. // 连接建立，后续可用于执行链码
10. Gateway.Builder builder = Gateway.createBuilder();
11. builder.identity(wallet, "appUser").networkConfig(networkConfigPath).discovery(**true**);
12. **return** builder.connect();
13. }

调用示例

1. // connect to the network and invoke the smart contract
2. **try** (Gateway gateway = connect()) {
3. ChaincodeService service = **new** ChaincodeService();
5. Map<String, String> orderMap = **new** HashMap<>();
6. orderMap.put("carrierOrgName", "carrierOrgName5");
7. orderMap.put("consigneeName", "consigneeName1");
8. orderMap.put("consignorName", "consignorName1");
9. orderMap.put("loadAddr", "loadAddr1");
10. orderMap.put("unloadAddr", "unloadAddr1");
11. orderMap.put("arrivePlanTime", "arrivePlanTime1");
12. orderMap.put("transDistance", "1");
13. orderMap.put("price", "1");
15. System.out.println("\n");
16. String all = service.queryAllTmsOrder();
17. System.out.println("Evaluate Transaction: GetAllTmsOrder, result: " + all);
19. String add = service.addTmsOrder(orderMap);
20. System.out.println("Evaluate Transaction: CreateTmsOrder, result: " + add);
21. } **catch** (Exception e) {
22. System.err.println(e);
23. }