# Hyperledger Explorer部署

## 一、Hyperledger Fabric环境搭建

该环境基于1.2版本，在ubuntu16.04版本上搭建,具体搭建环境如下：

1、安装curl

sudo apt install curl

2、安装docker-ce：本次安装使用脚本安装

1）使用curl下载docker 安装脚本：

curl -fsSL get.docker.com -o get-docker.sh

下载完成后会出现一个get-docker.sh的脚本

2）启动脚本

sudo sh get-docker.sh

3）验证

使用docker version查看安装版本

3、安装docker-compose

1）下载

sudo curl -L https://github.com/docker/compose/releases/download/1.24.0/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

2）设置权限

sudo chmod +x /usr/local/bin/docker-compose

3）验证安装

docker-compose version

4、安装go环境

1）下载golang安装包，并解压。

wget https://dl.google.com/go/go1.10.3.linux-amd64.tar.gz

sudo tar -C /usr/local -xzf go1.10.3.linux-amd64.tar.gz

2）创建go path目录

sudo mkdir /opt/gopath

3）设置环境变量

sudo vim /etc/profile

添加环境变量：

export GOROOT=/usr/local/go

export GOPATH=/opt/gopath

export PATH=$PATH:$GOROOT/bin:$GOPATH/bin

执行命令使文件生效：

source /etc/profile

5、下载fabric的引导脚本bootstrap.sh

1）创建文件夹：/opt/gopath/src/github.com/hyperledger

2）进入创建的文件夹，下载脚本：

curl -sSL [https://raw.githubusercontent.com/hyperledger/fabric/release-1.2/scripts/bootstrap.sh -o bootstrap.sh](https://raw.githubusercontent.com/hyperledger/fabric/release-1.2/scripts/bootstrap.sh%20-o%20bootstrap.sh)

3）安装git

apt install git

4）执行下载的脚本：

chmod +x bootstrap.sh

sudo ./bootstrap.sh 1.2.0 1.2.0 0.4.10

6、启动fabric1.2网络

1）生成证书。

进入fabric-sample文件夹下的first network执行以下脚本。（first network文件夹用提供的示例代码替换）。

./byfn.sh generate （sudo chmod -R 777 fabric-samples）

该命令会生成网络所需要的证书。

2）启动网络

./byfn.sh up （sudo chmod -R 777 /var/run/docker.sock）

docker-compose -f docker-compose-cli.yaml start (如果docker容器存在)

./byfn.sh down (会删除所有docker容器和数据）

7、链码部署

将链码文件夹financialApp/chaincode/turing拷贝到fabric-sample/chaincode文件夹下

1）进入cli容器

docker exec -it cli bash

2）设置环境参数（链码部署的节点，在cli容器中执行命令，连接的是环境变量中设置的节点，节点peer的docker容器不能执行指令）

export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp（设置签名证书位置）

export CORE\_PEER\_ADDRESS=peer0.org1.example.com:7051（设置节点）

export CORE\_PEER\_LOCALMSPID="Org1MSP" （设置签名组织）

3）初始化链码

在/opt/gopath/src/github.com/hyperledger/fabric-samples/chaincode（该文件夹下有很多链码示例）下新建文件夹（例如turing），再把链码文件（例如turing.go）放在turing下。

peer chaincode install -n turing-v 1.0 -p github.com/chaincode/turing

每次初始化之前都先要设置环境参数，要在4个节点都安装链码。

4）实例化链码

以下过程在任意一个节点执行一次就行。

peer chaincode instantiate -o orderer.example.com:7050 -C mychannel -n turingcc -v 1.0 -c '{"Args":["init","a","10000","b","20000"]}' -P "AND ('Org1MSP.member','Org2MSP.member')"

5）升级链码

每个节点安装

peer chaincode install -n turingcc -v 1.1 -p github.com/chaincode/turing

初始化

peer chaincode upgrade -o orderer.example.com:7050 -C mychannel -n turingcc -v 1.1 -c '{"Args":["init","a","20000","b","10000"]}' -P "AND ('Org1MSP.member','Org2MSP.member')"

6）调用链码

peer chaincode query -C mychannel -n turingcc -c '{"Args":["query","a"]}'

peer chaincode invoke -C mychannel -n turingcc -o orderer.example.com:7050 --peerAddresses peer0.org1.example.com:7051 --peerAddresses peer0.org2.example.com:9051 --peerAddresses peer0.org3.example.com:11051 -c '{"Args":["invoke","a","b","20"]}'

## 二、Hyperledger Expolorer环境搭建

1、安装项目运行所依赖的环境

1）安装node.js环境（版本必须是8.11.\*）

在/home/demo目录下：

wget https://nodejs.org/dist/v10.11.0/node-v10.11.0-linux-x64.tar.xz

tar xf node-v10.11.0-linux-x64.tar.xz

ln -s /home/demo/node-v10.11.0-linux-x64/bin/node /usr/local/bin/

ln -s /home/demo/node-v10.11.0-linux-x64/bin/npm /usr/local/bin/

2）安装PostgreSQL 9.5或更高版本

apt-get install postgresql

3）安装Jq (json工具)

apt-get install jq

2、获取源码（**后面所有操作都不要在root用户下操作**）

git clone <https://github.com/hyperledger/blockchain-explorer.git>

3、设置数据库

1）进入blockchain-explorer/app下

explorerconfig.json为配置文件。

2）生成项目所需要的表和库

运行create database脚本。

cd blockchain-explorer/app/persistence/fabric/postgreSQL/db

./createdb.sh

(如在root用户下操作，会提示psql: FATAL: database "root" does not exist

需要sudo -u postgres psql进入psql命令行执行：

CREATE USER root SUPERUSER;

CREATE DATABASE root WITH OWNER root;

)

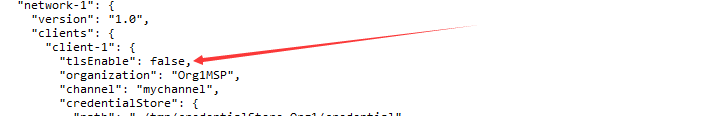
使用sudo -u postgres psql命令连接数据库，其他数据操作命令需要参考官网

区块链环境如果重新搭建，需要清空数据库，使用命令drop database fabricexplorer; 删除postgre数据库后需要cd app/persistence/fabric/postgreSQL/db，./createdb.sh

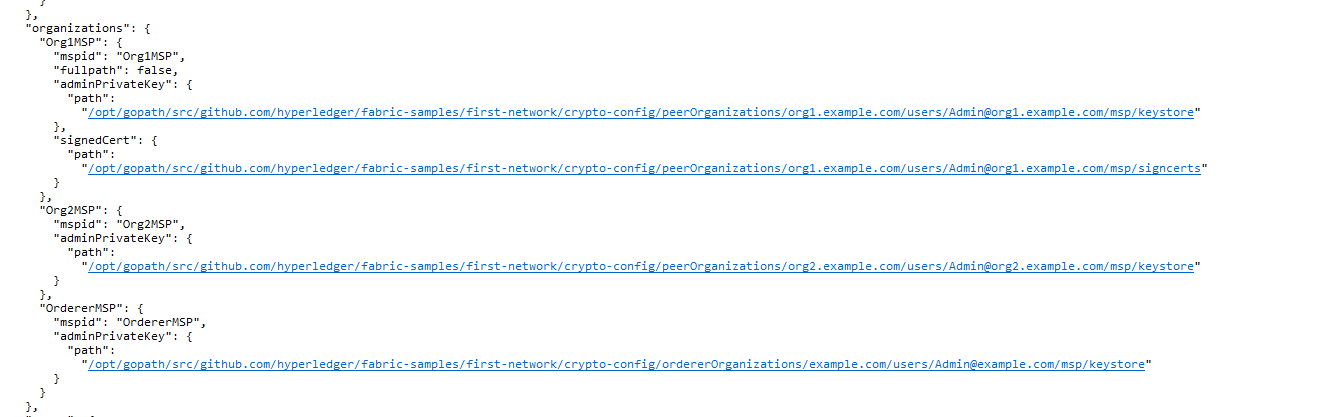
4、修改配置文件

cd blockchain-explorer/app/platform/fabric

修改config.json如下地方，本环境未启用TLS



修改 path部分，路径为证书地址



修改各个节点的ip



5、编译项目：

cd blockchain-explorer

npm install

cd blockchain-explorer/app/test

npm install

npm run test

cd blockchain-explorer/client

npm install

npm test -- -u --coverage

npm run build

6、运行项目：

cd blockchain-explorer/

执行start.sh脚本或者运行node main.js命令，stop.sh停止服务。

访问：<http://ip:port/8080>

注意，如果要修改代码，每次修改完，都需要到clinet下面去编译，运行npm run build 命令进行编译。

如果启动失败，查看是否是权限问题，需要执行下面命令：

sudo chmod 644 /opt/gopath/src/github.com/hyperledger/fabric-samples/first-network/crypto-config/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp/keystore/b45278dadce16c8bf8c09350809ffc2ec4337bc76fd5464642cd75c90ccb9f45\_sk

sudo chmod 644 /opt/gopath/src/github.com/hyperledger/fabric-samples/first-network/crypto-config/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp/keystore/124d9be555f6d078f542b2f92277ca81ba22973f1798d19479706119d346712c\_sk

sudo chmod 644 /opt/gopath/src/github.com/hyperledger/fabric-samples/first-network/crypto-config/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp/keystore/77a0f7a0ddab9346aa99809a11a091128ab3df6b671bb4e785c29c9fd5dc6987\_sk

sudo chmod 644 /opt/gopath/src/github.com/hyperledger/fabric-samples/first-network/crypto-config/ordererOrganizations/example.com/users/Admin@example.com/msp/keystore/b96cd4119bc8a9b4f827b9989fec02509602108206f85e1185a9018b5e57853d\_sk

644后文件根据实际文件名而定

## 三、SDK项目（接口）环境搭建

SDK项目为Java springboot maven项目，只需要编译jar包，编译运行就可以。fabric/financialApp/fabricTsg与链码turing.go对应。

SDK项目在8883端口起HTTP服务，如下两个HTTP消息会触发链码的调用

GET <http://192.168.66.137:8883/v1/accounts/a/balance> 等效 peer chaincode query -C mychannel -n turingcc -c '{"Args":["query","a"]}'

GET <http://192.168.66.137:8883/v1/accounts/a/b/20/createTransaction> 等效 peer chaincode invoke -C mychannel -n turingcc -o orderer.example.com:7050 --peerAddresses peer0.org1.example.com:7051 --peerAddresses peer0.org2.example.com:9051 --peerAddresses peer0.org3.example.com:11051 -c '{"Args":["invoke","a","b","20"]}'

SDK代码中的getChannel函数需要根据背书策略，将相应的节点加入。

public static Channel getChannel() throws Exception{

init();

Channel channel = client.newChannel(C.CHANNELID);

channel.addPeer(client.newPeer(C.CONNFIG\_Peer0Org1Name, C.CONNFIG\_Peer0Org1));

channel.addPeer(client.newPeer(C.CONNFIG\_Peer0Org2Name, C.CONNFIG\_Peer0Org2));

channel.addPeer(client.newPeer(C.CONNFIG\_Peer0Org3Name, C.CONNFIG\_Peer0Org3));

channel.addOrderer(client.newOrderer(C.CONNFIG\_OrdererName, C.CONNFIG\_Orderer));

channel.initialize();

return channel;

}

SDK代码中的证书路径需要根据具体证书位置调整

public String CERTDIR = "/opt/gopath/src/github.com/hyperledger/fabric-samples/three-org-network";

如果需要在官方first-network/.byfn.sh up单机部署的fabric示例上运行，需要将linux的/etc/hosts中设置的peer0.org1.example.com等注释掉，SDK中所有的grpc://orderer.example.com:7050地址中的域名都替换成127.0.0.1即可。

参考文档：

kafka多节点部署：https://www.cnblogs.com/llongst/p/9608886.html

动态增删节点：https://www.jianshu.com/p/fdbd7581b0fc

链码API：https://godoc.org/github.com/hyperledger/fabric/core/chaincode/shim

Fabric CLI命令手册：http://cw.hubwiz.com/card/c/fabric-command-manual/1/1/5/