Fabric 测试环境搭建

1：电脑。 CPU-G4560 , 威刚4G内存，硬盘SSD 64G，铭瑄MS-B250.

2：系统。 中国味道的ubuntu kylin. 其实ubuntu 16.04 64位。https://www.ubuntukylin.com/

3: **go version**

go version go1.9.2 linux/amd64

**docker version**

18.06.0-ce

**docker-compose version**

docker-compose version 1.12.0, build b31ff33

4：下载Fabric

可以从github下载fabric源码，地址为：<https://github.com/hyperledger/fabric>

先在GOPATH目录下，依次建立文件夹src->github.com->hyperledger,然后cd到hyperledger

1:cd $GOPATH/src/github.com/hyperledger

使用git 命令下载源码到当前目录(如果不会git,请去网易云课堂上学习git)

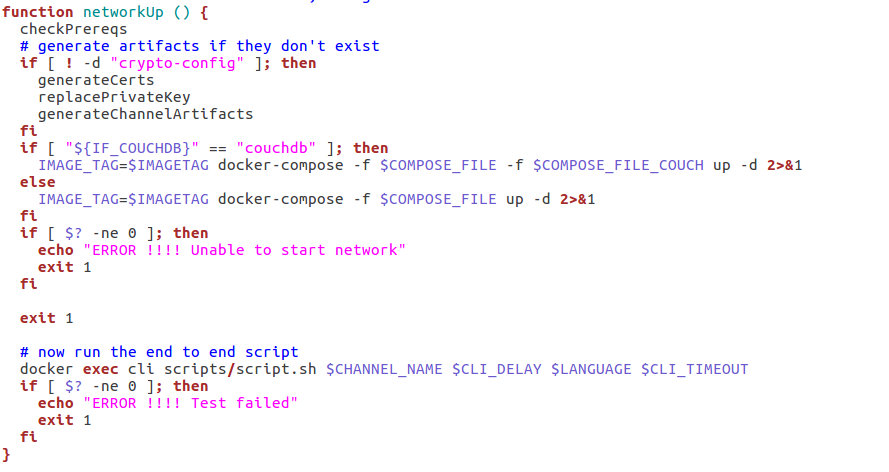
2: git clone <https://github.com/hyperledger/fabric>

3: fabric-samples 学习研究。　在执行bynf.sh 文件跑通了之后，我花了相当长的时间研究了bynf.sh这个脚本文件，看懂这个脚本文件，基本上就算入了fabric的门了。此文件干了如下的事情(网上也有总结，我这里简单的说下我的总结)：

１：生成了证书和创世区块，通道文件等。

２：启动了１个orderer节点容器，４个peer节点容器，１个cli终端容器。

３：创建了通道，部署链码，查询，转账。

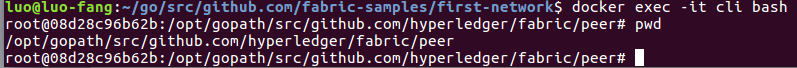


相关代码如上，可以看到最后一部分”#now run the end to end script” 其实就是：创建通道，节点加入通道，部署合约，查询，转账的脚本。这一系列的操作都是在终端cli完成的。

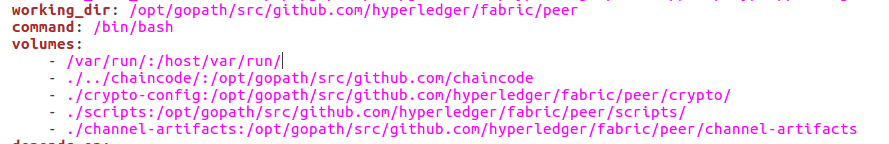
现在我在上面加入了 exit 1 代码，在容器生成运行后，我自己手动操作　创建通道，链码部署。

3.1　承上。６个容器创建运行后，手动进行通道和链码部署操作。

输入：　docker exec -it cli bash 即可进入cli终端交互，如下图。



为什么一进入cli 就到了peer 目录下呢？因为在docker-compose-cli.yaml(编排容器启动运行的配置文件)文件中有个working\_dir选项，如下图。



从上图还可以看到本机的./crypto-config和./scripts和./channel-artifacts 挂接到了cli容器的peer目录下。

下面word字体格式和书写问题，不方便粘贴复制代码，此处可以参考目录下的<手动创建通道－链码－交易>文件。

**3.2.1 先设置环境变量，如下。**

export ORDERER\_CA=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/or

derer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

export CHANNEL\_NAME="mychannel"

export CORE\_PEER\_LOCALMSPID="Org1MSP"

export 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

export CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp

export

CORE\_PEER\_ADDRESS=peer0.org1.example.com:7051

**3.2.2 创建通道**

peer channel create -o orderer.example.com:7050 -c $CHANNEL\_NAME -f ./channel-artifacts/channel.tx --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA >&log.txt

创建信息写入到了log.txt 中，同时peer目录下也会多出mychannel.block 文件。

**3.2.3 加入通道**

peer channel join -b $CHANNEL\_NAME.block >&log.txt

**3.2.４安装链码**

CC\_SRC\_PATH="github.com/chaincode/chaincode\_example02/go/"

peer chaincode install -n mycc -v 1.0 -l "golang" -p ${CC\_SRC\_PATH} >&log.txt

**3.2.5实例化链码-部署链码**

peer chaincode instantiate -o orderer.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C $CHANNEL\_NAME -n mycc -v 1.0 -c '{"Args":["init","a","100","b","200"]}' -P "OR ('Org1MSP.peer','Org2MSP.peer')"

**3.3.6 查询链码**

peer chaincode query -C $CHANNEL\_NAME -n mycc -c '{"Args":["query","a"]}' >&log.txt

#　-c 指定通道名字　　-n 链码的名字　　>&log.txt 即将本条命令执行后返回的信息存入log.txt中。