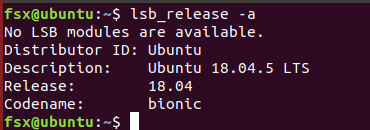
实验环境

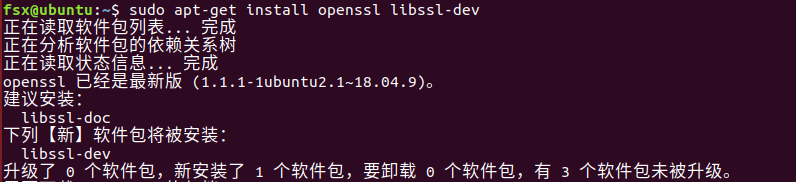
Ubuntu 18.04 LTS 64位



### 实验环境搭建

安装openssl

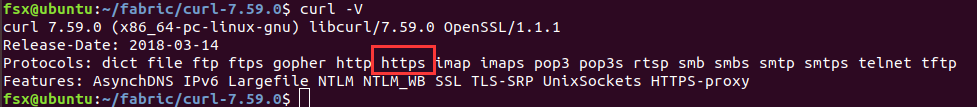
sudo apt-get install openssl libssl-dev



安装 curl

sudo apt-get install curl

curl -V



支持Https协议

安装docker和docker-compose

下载相关工具

sudo apt-get install apt-transport-https ca-certificates software-properties-common

添加docker官方GPG密钥

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –

设立仓库

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

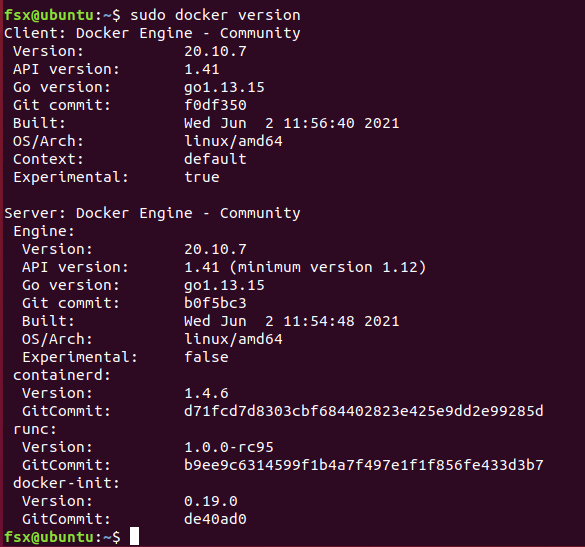
安装docker

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io

检验是否安装完成

sudo docker version



安装docker-compose

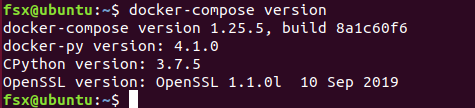
sudo curl -L "https://github.com/docker/compose/releases/download/1.25.5/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

给相应的二进制文件可执行权限

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

查看docker-compose 的版本

docker-compose version

*   
  Go programming language环境搭建

tar -xzvf go1.16.5.linux-amd64.tar.gz

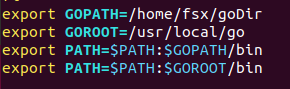
sudo cp -r go /usr/local

在主目录下建立goDir文件夹，做为GOPATH路径

mkdir /home/fsx/goDir

打开~/.bashrc,并在最后添加以下类似语句：

vim ~/.bashrc



使配置生效

source ~/.bashrc

检测go是否配置成功

go version



修改GOPROXY为国内golang代理地址

go env -w GOPROXY=https://goproxy.cn

配置编写chaincode安装必要环境

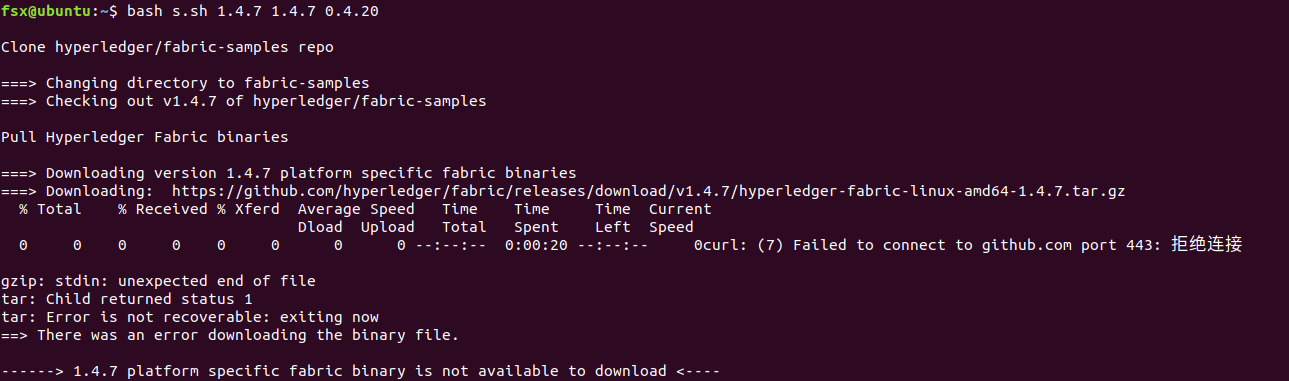
go get -u --tags nopkcs11 github.com/hyperledger/fabric/core/chaincode/shim

sudo apt install libtool libltdl-dev

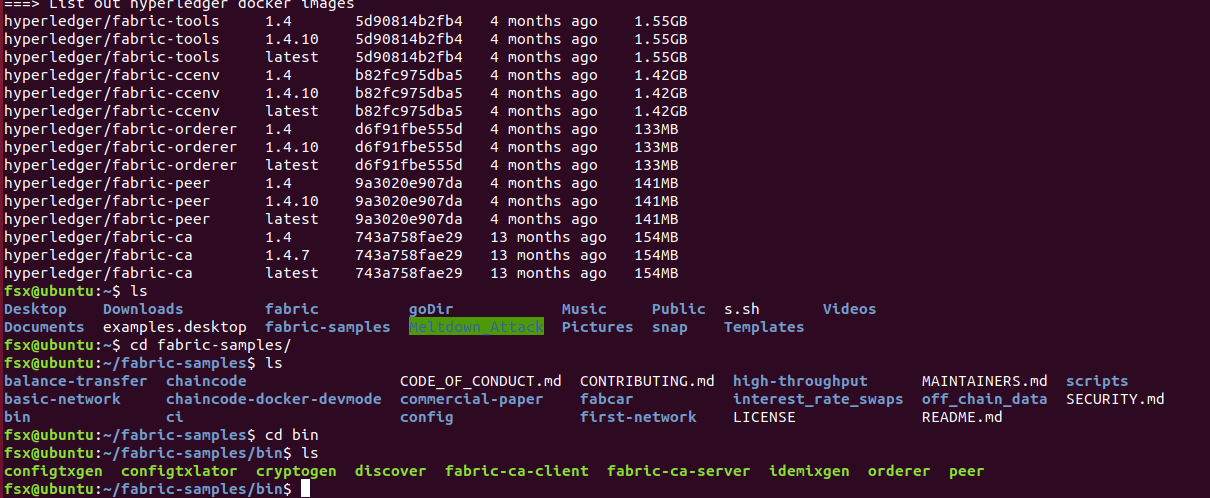
下载docker镜像和fabric的可执行文件

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

bash s.sh 1.4.7 1.4.7 0.4.20



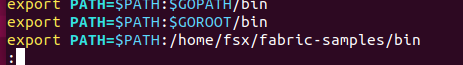
总是失败，检查发现是因为github上最老的fabric的releases版本是v1.4.10，因此修改命令为sudo bash s.sh 1.4.10 1.4.7 0.4.20 ，运行成功。



脚本后的三个参数分别对应fabric、fabric-ca和第三方镜像的版本。若仍不能成功。依照：<https://blog.csdn.net/ltt1st/article/details/81332752>和<https://blog.csdn.net/tiwolf/article/details/104822431/> 分别手动从github上下载hyperledger-fabric-linux-amd64-1.4.10.tar.gz、hyperledger-fabric-ca-linux-amd64-1.4.7.tar.gz

将bin目录的路径加到系统变量$PATH中，

vim ~/.bashrc



使配置生效

source ~/.bashrc

虚拟机最好打个快照

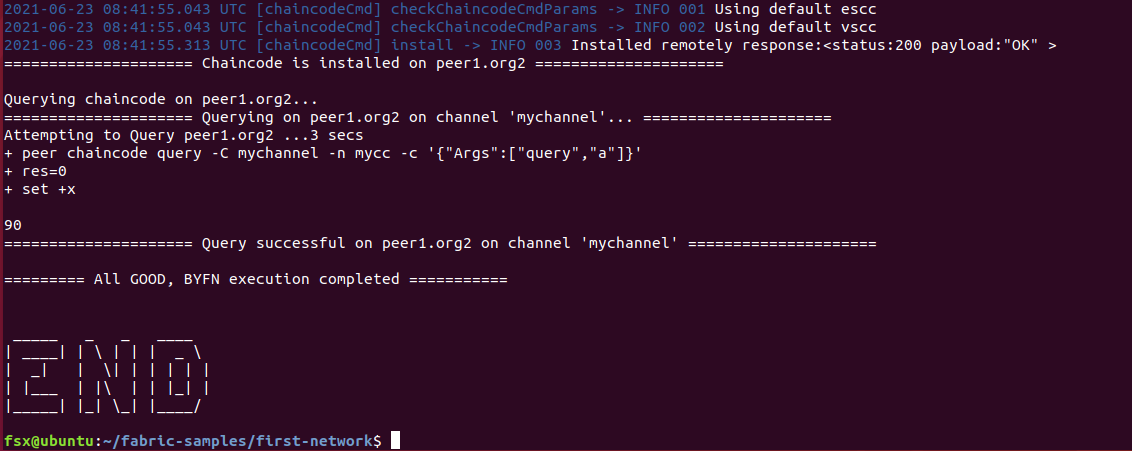
#### 实验任务

#### 任务1: Building Your First Network

cd fabric-samples/first-network/

首先运行官方提供的脚本：

sudo ./byfn.sh up



使用sudo ./byfn.sh -m down关闭网络，要不行就恢复快照

**不使用官方bash脚本，使用现有工具的前提的情况下，使用终端逐步完成部署、链码安装、初始化、调用、查询等相关操作。以下均在root权限进行。**

1.8. 运行工具

1.8.1. 手动生成配置文件

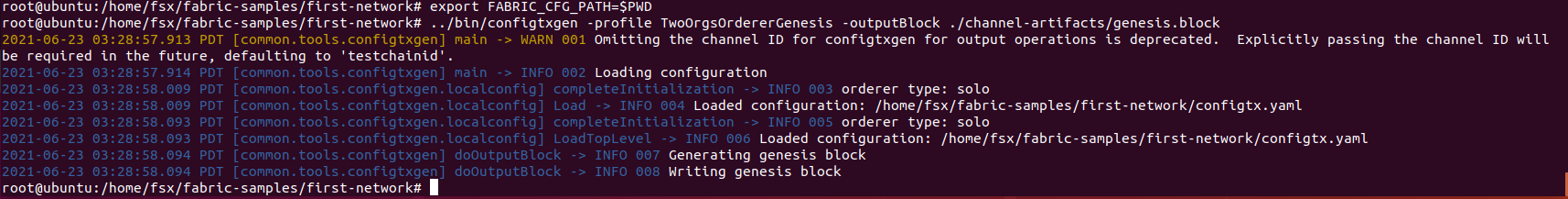
../bin/cryptogen generate --config=./crypto-config.yaml



设置一个环境变量来告诉configtxgen哪里去寻找configtx.yaml。然后，我们将调用configtxgen工具去创建orderer genesis block

export FABRIC\_CFG\_PATH=$PWD

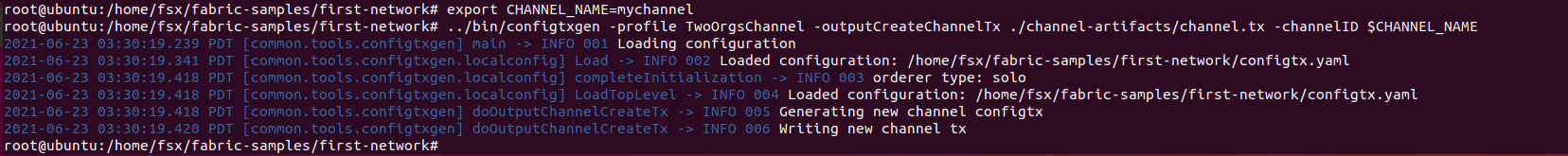
../bin/configtxgen -profile TwoOrgsOrdererGenesis -outputBlock ./channel-artifacts/genesis.block



创建channel transaction配置

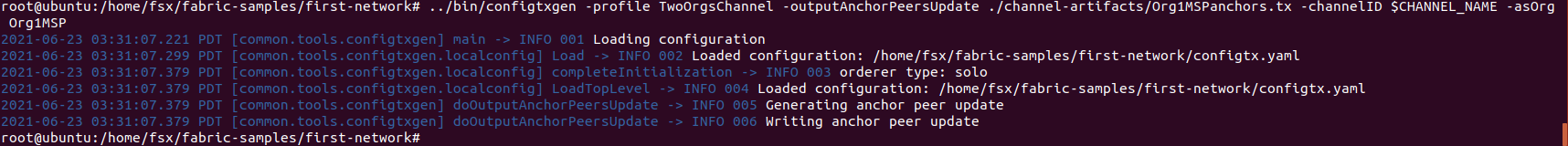
export CHANNEL\_NAME=mychannel

../bin/configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx -channelID $CHANNEL\_NAME



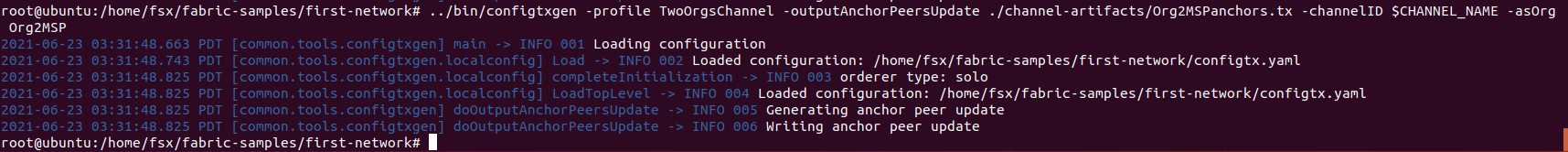
在正在构建的通道上定义Org1的anchor peer

../bin/configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID $CHANNEL\_NAME -asOrg Org1MSP



在同一个通道定义Org2的anchor peer

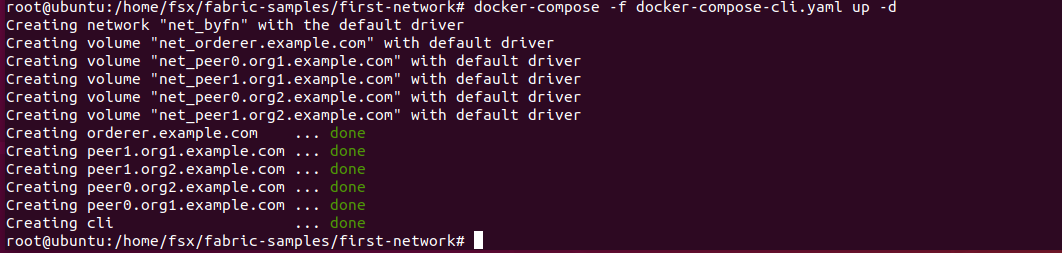
../bin/configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID $CHANNEL\_NAME -asOrg Org2MSP



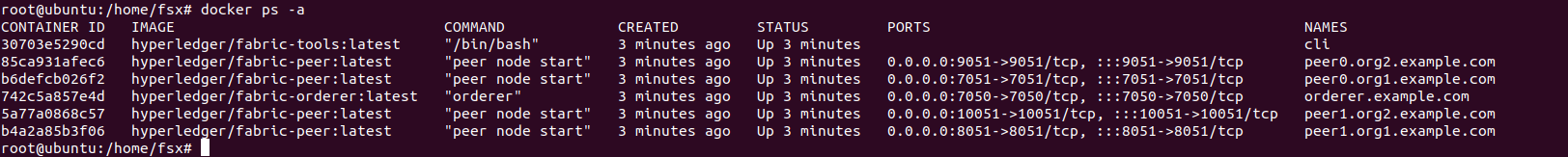
1.9. 启动网络

启动网络：

docker-compose -f docker-compose-cli.yaml up -d



用docker ps -a查看各docker



1.9.2. 创建&加入信道

我们将使用docker exec命令进入CLI容器：

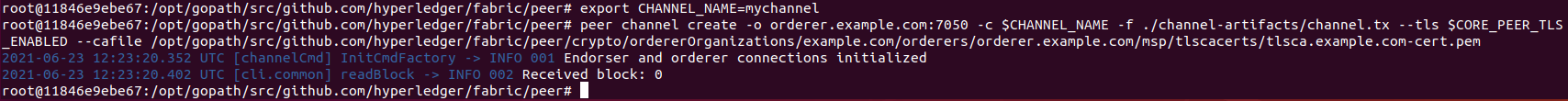
docker exec -it cli bash



创建通道

export CHANNEL\_NAME=mychannel

peer channel create -o orderer.example.com:7050 -c $CHANNEL\_NAME -f ./channel-artifacts/channel.tx --tls $CORE\_PEER\_TLS\_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem



此命令返回一个创世区块-<channel-ID.block>-我们将使用它加入信道。它包含了channel.tx中的配置信息。

设置环境变量，将peer0.org1.example.com加入通道

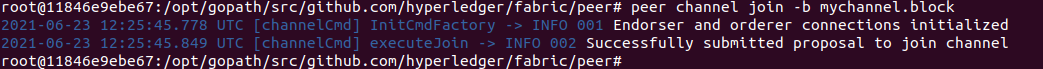
CORE\_PEER\_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp

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

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

peer channel join -b mychannel.block



修改环境变量，将peer0.org2.example.com加入通道

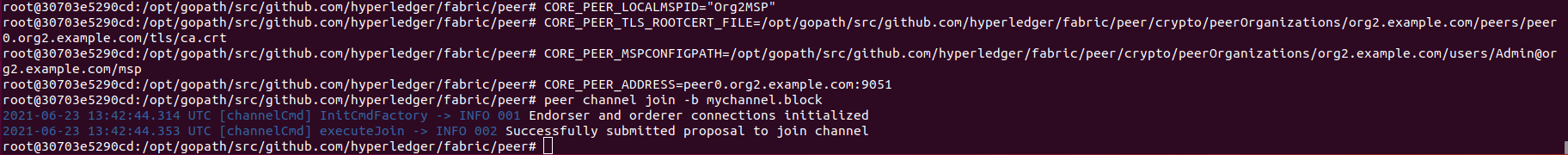
CORE\_PEER\_LOCALMSPID="Org2MSP"

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:9051

peer channel join -b mychannel.block



修改环境变量，将peer1.org1.example.com加入通道

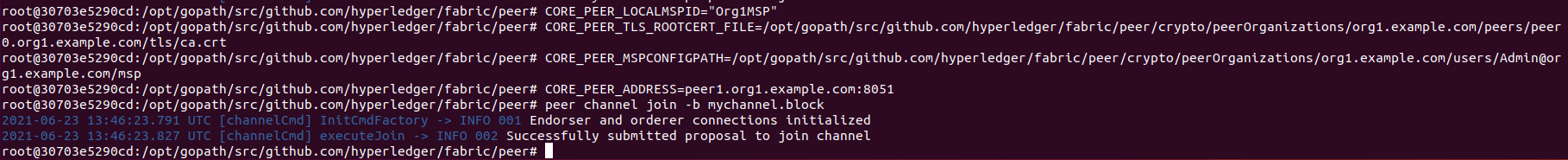
CORE\_PEER\_LOCALMSPID="Org1MSP"

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=peer1.org1.example.com:8051

peer channel join -b mychannel.block



修改环境变量，将peer1.org2.example.com加入通道

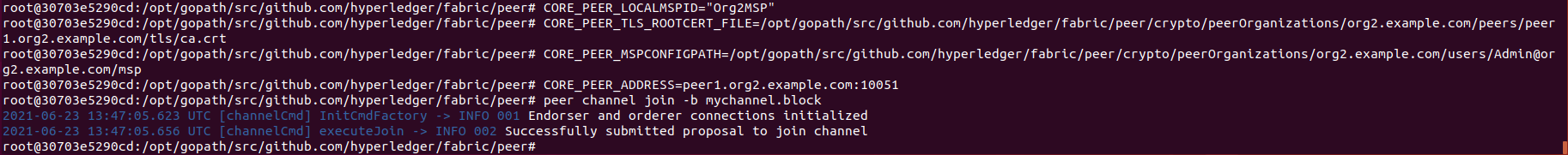
CORE\_PEER\_LOCALMSPID="Org2MSP"

CORE\_PEER\_TLS\_ROOTCERT\_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer1.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

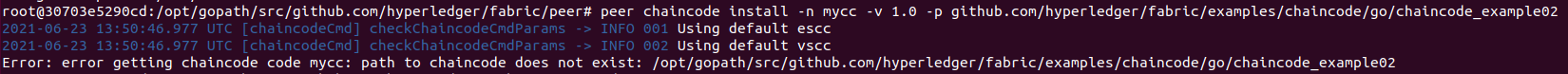
CORE\_PEER\_ADDRESS=peer1.org2.example.com:10051

peer channel join -b mychannel.block



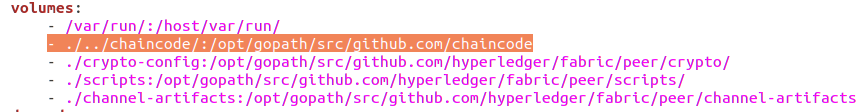
1.9.3. 安装和实例化链码

peer chaincode install -n mycc -v 1.0 -p github.com/hyperledger/fabric/examples/chaincode/go/chaincode\_example02

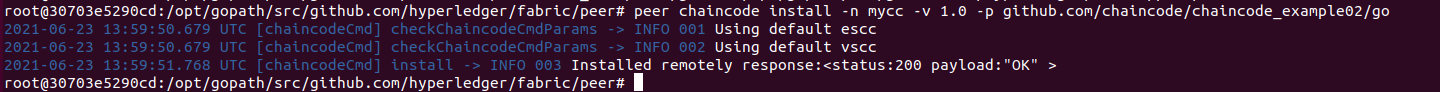


出现错误Error: error getting chaincode code mycc: path to chaincode does not exist: /opt/gopath/src/github.com/hyperledger/fabric/examples/chaincode/go/chaincode\_example02，是由于命令中指定的路径不对所导致的。

在docker-compose-cli.yaml会看到chaincode这一行，指定了chaincode的路径。



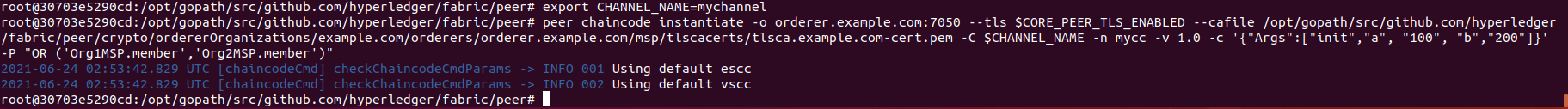
修改命令为peer chaincode install -n mycc -v 1.0 -p github.com/chaincode/chaincode\_example02/go



在信道上实例化chaincode，需要Org1或者Org2组织中的其中一个的节点的背书即可（即只有一个背书）：

export CHANNEL\_NAME=mychannel

peer chaincode instantiate -o orderer.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL\_NAME -n mycc -v 1.0 -c '{"Args":["init","a", "100", "b","200"]}' -P "OR ('Org1MSP.member','Org2MSP.member')"



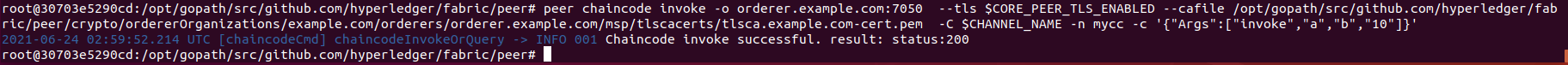
1.9.4. 查询，查询一下a的值，以确保链码被正确实例化，state DB被填充。

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



1.9.5. 调用，现在让我们从a账户转10到b账户。这个交易将创建一个新的区块并更新state DB。调用语法如下：

peer chaincode invoke -o orderer.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL\_NAME -n mycc -c '{"Args":["invoke","a","b","10"]}'



1.9.6. 查询，让我们确认下我们之前的调用被正确地执行了。我们初始化了a的值为100，在上一次调用的时侯转移了10给b。因此，查询a应该展示90。

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

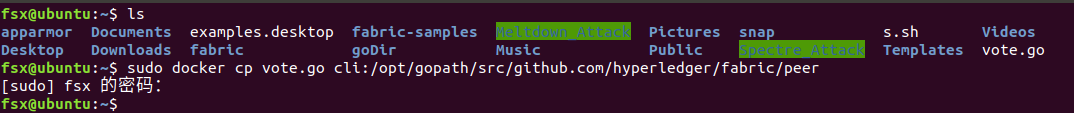


#### 任务2: chaincode 编写

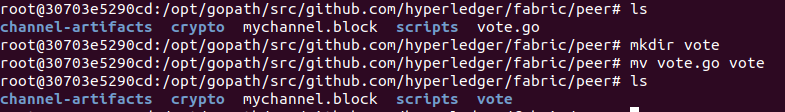
参考https://blog.csdn.net/u010857052/article/details/83931554

新开一个终端，将编写的chaincode拷贝入客户端docker中。

sudo docker cp vote.go cli:/opt/gopath/src/github.com/hyperledger/fabric/peer

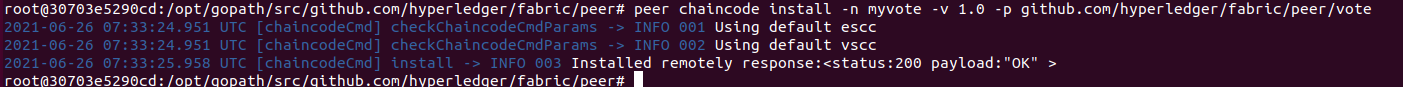


在客户端docker中创建目录vote，将编写的vote.go移至vote文件夹内。

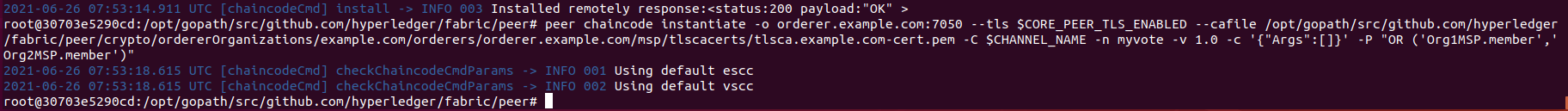


安装和实例化链码（myvote为链码名称，可自定义）

peer chaincode install -n myvote -v 1.0 -p github.com/hyperledger/fabric/peer/vote

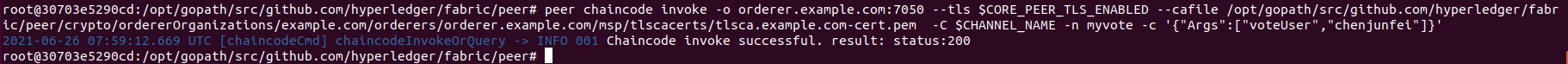


peer chaincode instantiate -o orderer.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL\_NAME -n myvote -v 1.0 -c '{"Args":[]}' -P "OR ('Org1MSP.member','Org2MSP.member')"



创建投票人信息，初始创建票数加一

peer chaincode invoke -o orderer.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL\_NAME -n myvote -c '{"Args":["voteUser","chenjunfei"]}'



查询投票人信息，可见成功创建投票人信息，正确计数

peer chaincode query -C $CHANNEL\_NAME -n myvote -c '{"Args":["getUserVote"]}'

