**Hyperledger Fabric 搭建流程**

**--- 基于solo共识**

**一、环境搭建**

各个主机的配置情况：

|  |  |
| --- | --- |
| 10.10.44.120 | orderer0.example.com |
| 10.10.44.122 | peer0.org1.example.com |
| 10.10.44.123 | peer1.org1.example.com |
| 10.10.44.124 | peer0.org2.example.com |
| 10.10.44.129 | peer1.org2.example.com |

**每台主机都需要先进行单机的区块链部署，所以以下步骤适用于所有主机。**

**1、Docker-CE 安装**

**1）安装docker-ce**

**step 1: 安装必要的一些系统工具**

#sudo yum install -y yum-utils device-mapper-persistent-data lvm2

**Step 2: 添加软件源信息**

#sudo yum-config-manager --add-repo <http://mirrors.aliyun.com/docker->ce/linux/centos/docker-ce.repo

**Step 3: 更新并安装 Docker-CE**

#sudo yum makecache fast

#sudo yum -y install docker-ce

**2) 是否安装成功**

#docker --version

**3）docker 启动**

#service docker start

**4）docker 开机自启动**

#chkconfig docker on

**2、Docker-compose 安装**

**1）Compose 安装**

#curl https://get.daocloud.io/docker/compose/releases/download/1.19.0/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose

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

**2）是否安装成功**

#docker-compose --version

**3、Go语言安装**

**1）下载解压**

#cd /opt

#mkdir golang

#cd golang

#curl -O https://www.golangtc.com/static/go/1.9.2/go1.9.2.linux-amd64.tar.gz

#tar -zxvf go1.9.2.linux-amd64.tar.gz

**2) 配置Go语言环境变量**

#vi /etc/profile

export GOROOT=/opt/golang/go

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

**以上内容添加到/etc/profile后面**

#source /etc/profile

**3） 查看版本**

#go version

**4）查看环境变量**

#go env

**4、git 安装**

# yum install git

**5、Fabric的安装**

**1）Fabric的下载**

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

#cd $GOPATH/src/github.com/hyperledger/

#git clone https://github.com/hyperledger/fabric.git

#cd fabric

#git checkout v1.1.0

**6、加速镜像**

#sudo mkdir -p /etc/docker

#sudo tee /etc/docker/daemon.json <<-'EOF'

{

"registry-mirrors": ["https://8w1wqmsz.mirror.aliyuncs.com"]

}

EOF

#sudo systemctl daemon-reload

#sudo systemctl restart docker

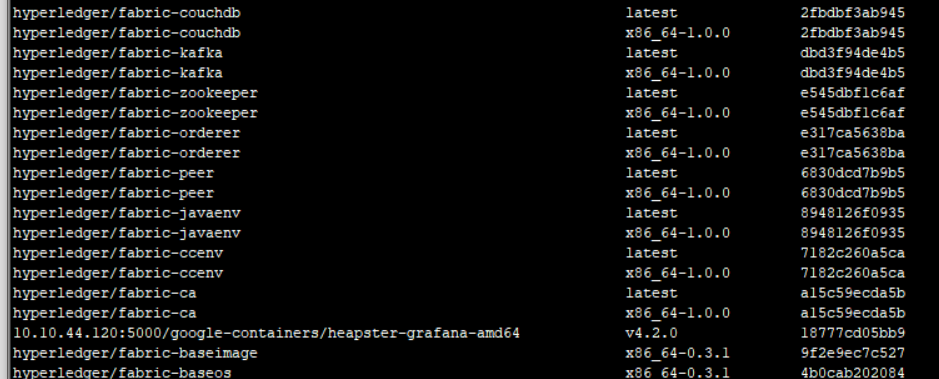
**7、下载镜像**

#cd /opt/gopath/src/github.com/hyperledger/fabric/example/e2e\_cli

#./download-dockerimages.sh

#docker images

**部镜像截图如下：**



**8、安装gcc**

#yum install gcc

**9、运行e2e\_cli**

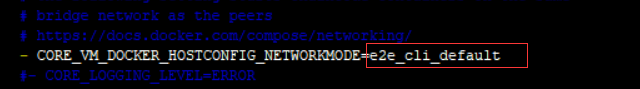
#cd /opt/gopath/src/github.com/hyperledger/fabric

#make release

#cd /opt/gopath/src/github.com/hyperledger/fabric/examples/e2e\_cli

#vim base/peer-base.yaml

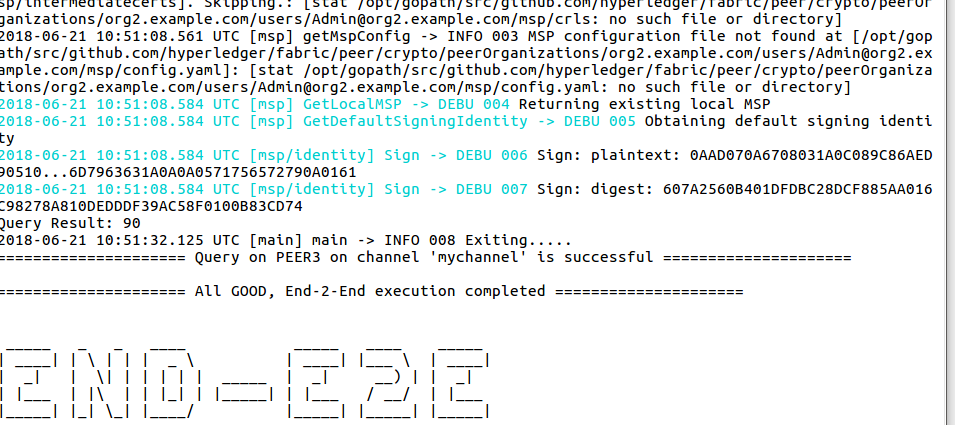
**(将圈住的部分改为图中所显示的名称)**



# bash generateArtifacts.sh

#./network\_setup.sh up

(出现ALL GOOD 即单机部署成功)



**关闭网络**

#./network\_setup.sh down

**二、多机部署**

|  |  |
| --- | --- |
| 10.10.44.120 | orderer0.example.com |
| 10.10.44.122 | peer0.org1.example.com |
| 10.10.44.123 | peer1.org1.example.com |
| 10.10.44.124 | peer0.org2.example.com |
| 10.10.44.129 | peer1.org2.example.com |

**（1）生成公私钥、证书、创世区块。**

**1）** **切换到orderer.example.com这台服务器上**

#cd /opt/gopath/src/github.com/hyperledger/fabric/examples/e2e\_cli

# ./generateArtifacts.sh mychannel

**2）生成的crypto-config和channel-artifacts使用scp命令拷贝到其它四台电脑上的e2e\_cli的目录中，如：**

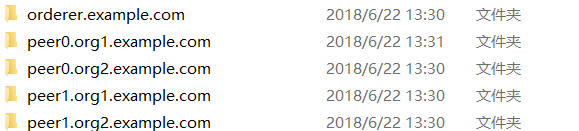
#scp -r crypto-config root@10.10.44.122 /opt/gopath/src/github.com/hyperledger/fabric/examples/e2e\_cli

#scp -r channel-artifacts root@10.10.44.122 /opt/gopath/src/github.com/hyperledger/fabric/examples/e2e\_cli

**（2）编写对应的配置文件**

**注：配置文件的编写流程已经独立出来，可在区块链的根目录下查看。**

**配置文件已经打包在该目录下。**

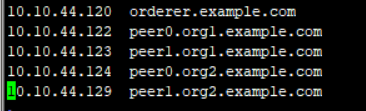


**（3）将对应的配置文件发送到对应的主机上**

**（4）修改hosts文件。**

#vim /etc/hosts

将图片中的数据添加到Centos7的/etc/hosts文件内



**三、多点集群的启动**

**1）orderer 排序服务启动**

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

**分别在每台机器上启动 -d 代表后台启动**

**2）peer 组织节点的启动**

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

**分别在每个节点上启动**

**四、运行脚本**

**启动脚本已在文档中备份**

**进入cli客户端容器**

#docker exec -it cli bash

#./script001.sh

**升级链码：**

**首先在cli 下把需要升级的链码进行安装，安装需要注意链码的路径,和版本号的变化，具体命令在多机启动脚本里详见。然后是升级命令:**

#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

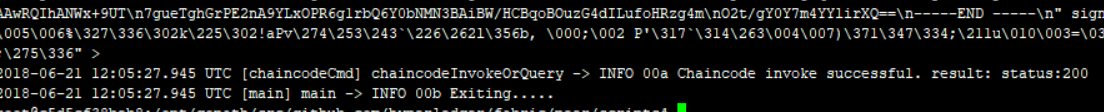
#CORE\_PEER\_TLS\_ENABLED=true

#peer chaincode upgrade -o orderer.example.com:7050 --tls true --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 mychannel -n mycc -v 1.2 -c '{"Args":["init"]}' -P "OR ('Org1MSP.member','Org2MSP.member')"

**验证：**

**链上存储：**

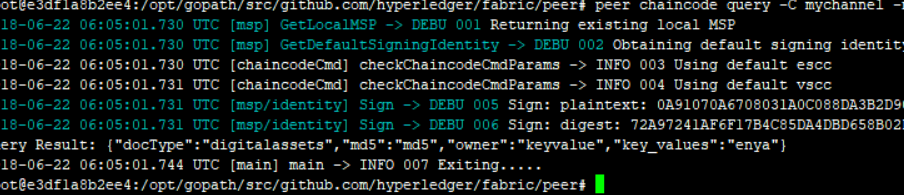
#peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc -v 1.2 -c'{"Args":["setdigital","md5","keyvalue","enya"]}'



**链上查询：**

#peer chaincode query -C mychannel -n mycc -v 1.2 -c '{"Args":["readdisital","md5"]}'

**如图所示：**



多机部署成功。