多通道搭建

参考文件：1》[http://www.cnblogs.com/studyzy/p/7237287.html](http://www.cnblogs.com/studyzy/p/7237287.html" \t "https://tower.im/projects/a66e4ebecc8e49a2a27fb21e5485644d/docs/cf23dcde3c9e47e6b70715589868557e/_blank)

<https://tower.im/projects/a66e4ebecc8e49a2a27fb21e5485644d/uploads/48619d89b7924f4c99df75025a11c593?version=1>

：：：最开始用的第一个文档来进行环境的搭建，做到后面发现多机连接后在创建通道，加入通道部署chaincode会有问题，又参考第二个文档最终完成的多节点的部署

1. 前期准备：(可以直接参考第一个文件第二个文件准备即可）
2. 环境构建与测试  
    1>安装go

2>安装docker和docker-compose

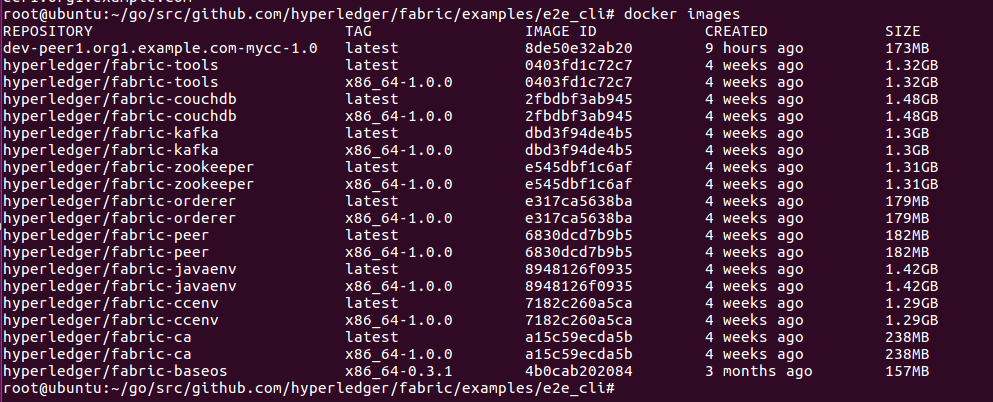
3>下载fabric代码mkdir -p /opt/gopath/github.com/hyperledger/ && cd /opt/gopath/github.com/hyperledge，

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

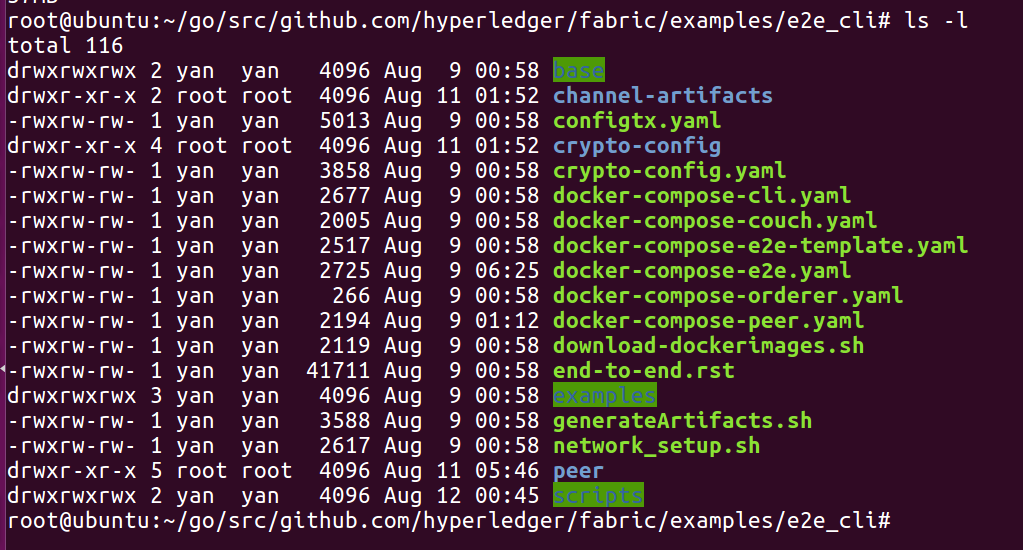
4>git checkout v1.0.0-beta 镜像使用的是beta版本，因此需要把Fabric切换到v1.0.0beta源码分支，以兼容fabric/example/e2e中的配置

5>下载docker 镜像

进入到fabric/examples/e2e\_cli目录下执行./doenload-dockerimages.sh下载如下镜像



6>运行测试进入到fabric/examples/e2e\_cli文件夹：



执行bash network\_setup.sh up 一键搭建网络

最后出现=== All GOOD, End-2-End execution completed ===

Bash network\_setup.sh down删除所有容器和证书

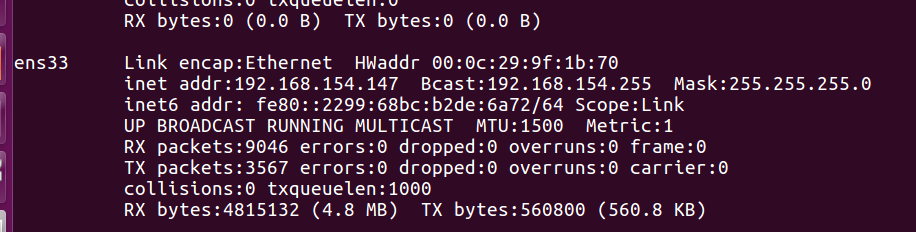
说明环境搭建成功！这个命令可以在本机启动4+1的Fabric网络并且进行测试，跑Example02这个ChainCode。我们可以看到每一步的操作，最后确认单机没有问题。确认我们的镜像和脚本都是正常的，我们就可以关闭Fabric网络，继续我们的多机Fabric网络设置工作。关闭Fabric命令

可以以此虚拟机为标准准备另外4台虚拟机或者实体机或者服务器！！

1. 准备5台电脑：（可以是虚拟机服务器之类的）

|  |  |  |  |
| --- | --- | --- | --- |
| VM1 | orderer | orderer.example.com | 192.168.154.143 |
| VM2 | Peer0 | peer0.org1.example.com | 192.168.154.148 |
| VM3 | Peer1 | Peer1.org1.example.com | 192.168.154.147 |
| VM4 | Peer2 | Peer2.org2.example.com | 192.168.154.146 |
| VM5 | Peer3 | Peer3.org2.example.com | 192.168.154.145 |

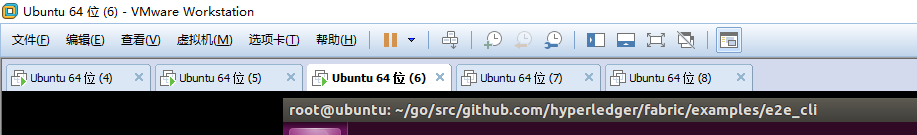
在终端内执行ifconfig察看极其ip地址



地址为inet addr:192.168.154.147

1>准备5台实体机或服务器每台均按第一台一样搭建

2>最简单的方法：用虚拟机直接克隆出与第一台一模一样的其他4台机器



如图：ubuntu5.6.7.8均为4的克隆版，所以4搭建好之后直接就可以使用克隆，5678就不需搭建

1. 配置多节点网络：
2. 生成公私钥、证书、创世区块等

公私钥和证书是用于Server和Server之间的安全通信，另外要创建Channel并让其他节点加入Channel就需要创世区块，这些必备文件都可以一个命令生成，官方已经给出了脚本：

./generateArtifacts.sh mychannel（在一个上面执行发送到其他节点）

生成cryptoconfig文件夹，以及channelartifacts（执行前会有一个空的文件夹）文件夹，五台机器需要用同一个证书，所以这个这俩个文件夹可以备份下来，每次出错需要重新生成时，无需在执行生成证书命令，可直接把这俩个文件夹复制到该目录里面！避免之后出现的很多由于证书不同而产生的各种错误

发送到其他机器的命令

scp -r 要发送的文件夹 主机名@主机ip：目的目录

（新创建的虚拟机会出现无法联通的问题：一般都是没有安装sshopenserver 安装即可连接成功！sudo apt install openssh-server）

1. 设置peer 节点的docker-compose 文件

(http://files.cnblogs.com/files/studyzy/fabric4peer1orderer.zip配置好的文件，已上传tower临时文件夹,更改ip即可)

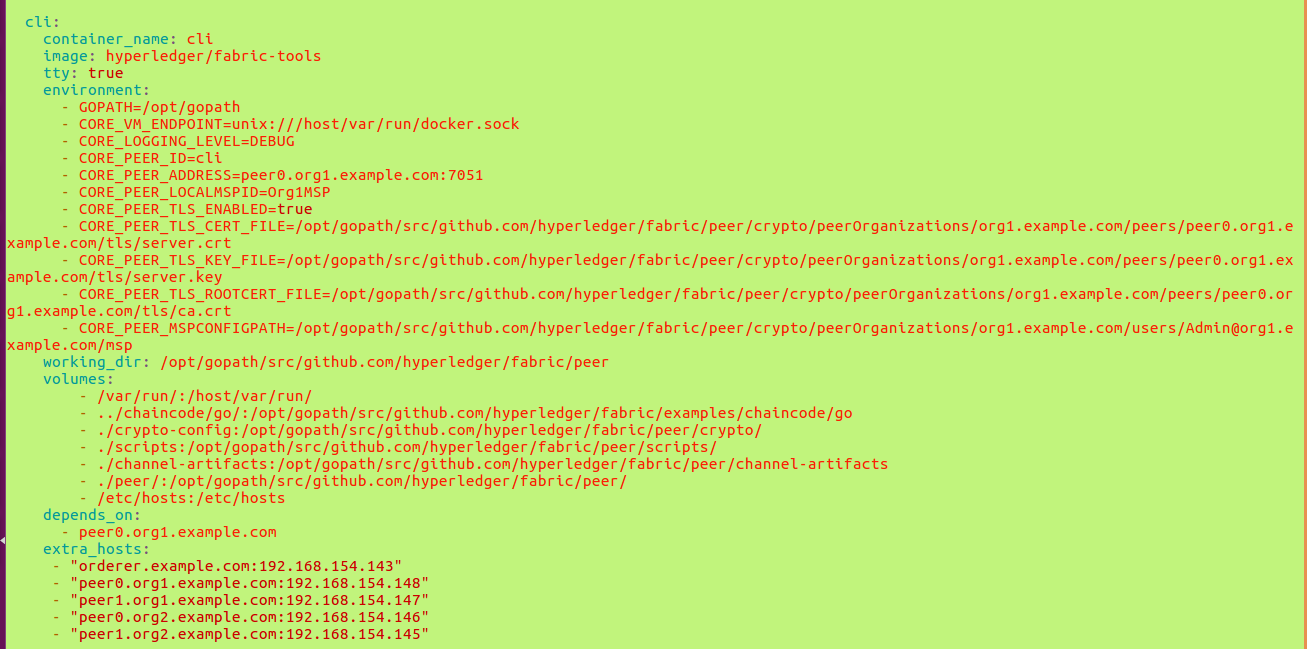
可以直接修改docker-compose-cli文件里面的配置，最后改名为docker-compose-peer

也可以：cp docker-compose-cli.yaml docker-compose-peer.yaml复制出一个，然后对yaml文件进行修改：去掉orderer的配置，只保留一个peer和cli，因为我们要多级部署，节点与节点之前又是通过主机名通讯，所以需要修改容器中的host文件，也就是extra\_hosts设置，修改后的peer配置如下：

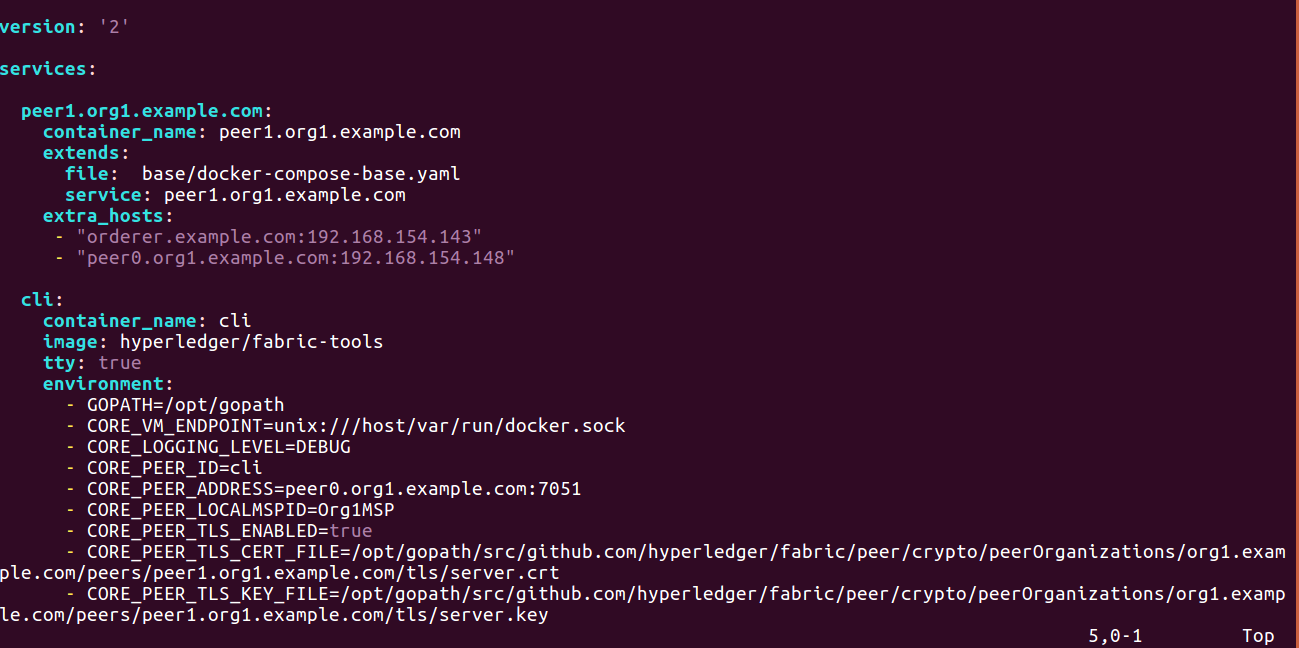


增加了一个extra\_hosts:

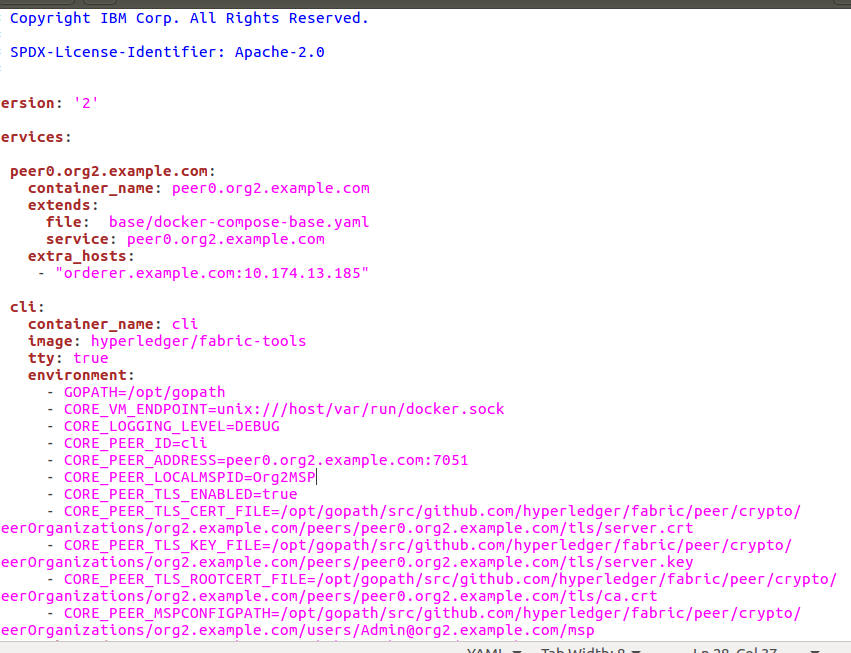
同样，cli也需要能够和各个节点通讯，所以cli下面也需要添加extra\_hosts设置，去掉无效的依赖，并且去掉command这一行，因为我们是每个peer都会有个对应的客户端，也就是cli，所以我只需要去手动执行一次命令，而不是自动运行。修改后的cli配置如下：



修改peer1节点的时候只需将对应为peer0的文件全部修改为peer1的 此外！：我们需要对docker-compose-peer.yaml做一个小小的修改，将启动的容器改为peer1.org1.example.com，并且添加peer0.org1.example.com的IP映射，对应的cli中也改成对peer1.org1.example.com的依赖。这是修改后的peer1.org1.example.com上的配置文件



org2的peer0与peer1同上！

如下图：

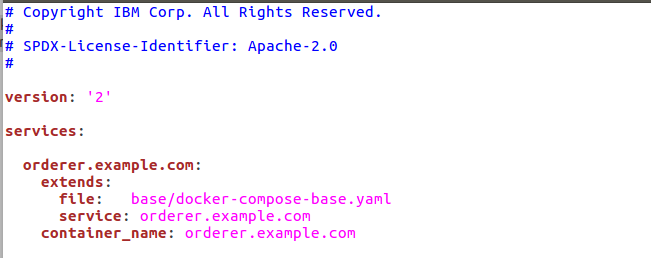
在单击模式下，4个peer会映射主机不同的端口，但是我们在多机部署的时候是不需要映射不同端口的，所以需要修改base/docker-compose-base.yaml文件，将所有peer的端口映射都改为相同的：

ports:

- 7051:7051

- 7052:7052

- 7053:7053

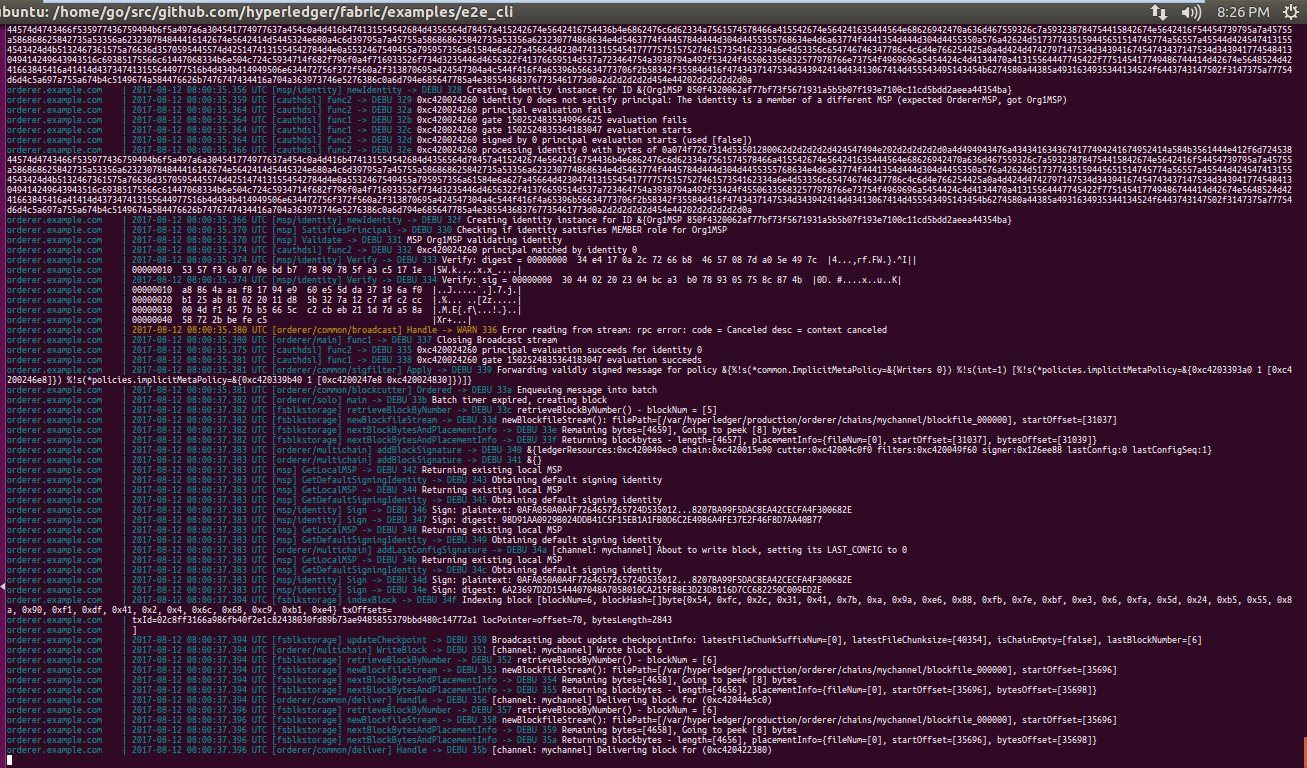
1. 设置orderer的docker-compose-orderer.yaml文件：orderer服务器上我们只需要保留order设置，其他peer和cli设置都可以删除。orderer可以不设置extra\_hosts。

此时文件均配置准备成功，接下来启动fabric节点！！！

1. 多节点的启动：

先启动orderer 节点：

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

或ocker-compose -f docker-compose-orderer.yaml up peer节点的每一个交易均会出现在日志中

启动peer0节点：

进入peer0对应所在机器的目录:执行docker-compose -f docker-compose-peer.yaml up –d

运行完毕后我们使用docker ps应该可以看到2个正在运行的容器

进入cli容器中：docker exec -it cli [bash进入到cli容器后会自动跳转到root@eae7587e7a9b:/opt/gopath/src/github.com/hyperledger/fabric/peer](mailto:bash进入到cli容器后会自动跳转到root@eae7587e7a9b:/opt/gopath/src/github.com/hyperledger/fabric/peer)

该目录映射到宿主机的e2e\_cli/peer/目录

创建通道：

察看脚本文件、scripts/script.sh发现每一步的命令：

1. 给变量赋值

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 （此变量为证书文件，也可以不用此变量直接也能用目录的文件，但用这个时每次都需赋值）

1. 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

channel 创建成功后， 会在当前目录下生成 mychannel.block 文件。 每个peer在向 orderer发送加入通道交易的时候，需要提供这个文件才能加入到mychannel 中，因此运行在其他 VM 上的peer

需要得到mychannel.block文件来加入到channel 中。 由于之前的文件映射关系，mychannel.block 文件可在宿主机的 e2e\_cli/peer/ 目录下获取， 这时可以通过宿主机把mychannel.block拷贝到其他3个几点所在机器的e2e\_cli/peer/ 目录下。

1. 把peer0加入到通道中：peer channel join -b mychannel.block
2. 安装chaincode:peer chaincode install -n mycc -v 1.0 -p github.com/hyperledger/fabric/examples/chaincode/go/chaincode\_example02 (这里装的是example02的chaincode,也可以装其他的）这时候peer0的chaincode已经装好，但并未实例化，接下来先配置其他3个节点。
3. peer1：启动peer1和cli：docker-compose -f docker-compose-peer.yaml up -d

进入到cli容器，发现容器里面并没有，mychannel.block文件，exit,将e2e\_cli/peer/里面的mychannel.block(之前从peer0拷贝来的）复制到cli容器里面，

然后再进去cli里面，加入通道，安装chaincode，此时各个节点的证书，创世块都是相同的再有报错可能就是环境变量的问题，执行一下环境变量的赋值命令即可，接下来2个节点的操作与peer1相同！

1. 所有节点加入通道并且安装chaincode 之后，我们来实例化chaincoode :

登录到4个节点的任意一个：peer chaincode instantiate -o orderer.example.com:7050 --tls --cafile $ORDERER\_CA -C mychannel -n mycc -v

1.0 -c '{"Args":["init","a","100","b","200"]}' -P "OR ('Org1MSP.member','Org2MSP.member')" （此时各个节点的账本已记录chaincode的world state）

实例化后，docker ps 发现多了一个dev-peer的容器，再在任意一个节点查询，peer chaincode query -C mychannel -n mycc -c '{"Args":["query","a"]}'

查询之后，chaincode也被实例化，docker ps 多了一个dev-peer...容器

到此！fabric多节点的部署已经完成！最重要的是证书等文件一定要一致！