# Fabric安装环境构建：

<https://www.cnblogs.com/llongst/p/9482038.html> 参考博客

## 安装docker

**1卸载原有的docker,**

yum remove docker

yum remove docker docker-common

yum remove docker docker-selinux

yum remove docker docker-engine

**2安装docker ce**

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

#yum-utils提供yum-config-manager工具，设备映射存储驱动需要device-mapper-persistent-data和lvm2工具

yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo #设置稳定存储库

# yum-config-manager --enable docker-ce-edge #开启edge，待理解

# yum-config-manager --enable docker-ce-test #开启test，待理解

yum install docker-ce

docker –version #安装顺利将看到ce版本

systemctl start docker

systemctl enable docker

docker images

docker kill $(docker ps -a -q) 杀死所有正在运行的容器

docker rm $(docker ps -a -q) 删除所有已经停止的容器

docker rmi $(docker images -q) 删除所有镜像  
docker rmi -f $(docker images –q 强制删除所有镜像

**3安装docker-compose**

curl -L https://github.com/docker/compose/releases/download/1.15.0/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose #下载最1.15.0版本docker-compose

chmod +x /usr/local/bin/docker-compose #赋予执行权限

cp /usr/local/bin/docker-compose /usr/bin #

docker-compose –version #安装顺利将看到版本

## go语言安装：

1. 参照Go官网，找到linux版本下载路径，执行以下操作下载最新版Go语言包

wget https://studygolang.com/dl/golang/go1.10.3.linux-amd64.tar.gz #1.10.3版本视情况而定

tar -xzf go1.8.3.linux-amd64.tar.gz –C /usr/local #解压到指定目录

vim /etc/profile #修改profile文件，最后添加如下两行

export PATH=$PATH:/usr/local/go/bin

export GOPATH=/opt/gopath

source profile #修改后执行

echo $PATH #正常会多个/usr/local/go/bin

echo $GOPATH #正常会显示/opt/gopath

## 安装gcc

yum -y install gcc #如果二进制文件要编译就需要安装gcc

# 下载fabric源码与相关镜像：

方式一：一般不成功

go get github.com/hyperledger/fabric

方式二：git clone

yum install git

git clone https://github.com/hyperledger/fabric/ #git clone下来最新的fabric源码（本次为1.4.1版本）

# cd fabric #待验证

# git checkout v1.4.0 #待验证

下载相关镜像：

1. 可根据git 下来的 /fabric/scripts/bootstrap.sh 脚本文件下载相关镜像，
2. bootstrap.sh脚本是个自动化下载相关版本二进制文件和镜像文件的脚本。

配置加速镜像

复制代码

# mkdir -p /etc/docker

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

{

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

}

EOF

# systemctl daemon-reload

# systemctl restart docker

docker镜像：

dcker pull hyperledger/fabric-ca:1.4.1 #先下载镜像，以ca为例

dcker tag hyperledger/fabric-ca:1.4.1 hyperledger/fabric-ca:latest #再将下载的改TAG（标签）

[root@fabric1 fabric-samples]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

hyperledger/fabric-ca 1.4.1 3a1799cda5d7 2 months ago 252MB

hyperledger/fabric-ca latest 3a1799cda5d7 2 months ago 252MB

hyperledger/fabric-tools 1.4.1 432c24764fbb 2 months ago 1.55GB

hyperledger/fabric-tools latest 432c24764fbb 2 months ago 1.55GB

hyperledger/fabric-ccenv 1.4.1 d7433c4b2a1c 2 months ago 1.43GB

hyperledger/fabric-ccenv latest d7433c4b2a1c 2 months ago 1.43GB

hyperledger/fabric-orderer 1.4.1 ec4ca236d3d4 2 months ago 173MB

hyperledger/fabric-orderer latest ec4ca236d3d4 2 months ago 173MB

hyperledger/fabric-peer 1.4.1 a1e3874f338b 2 months ago 178MB

hyperledger/fabric-peer latest a1e3874f338b 2 months ago 178MB

hyperledger/fabric-zookeeper 0.4.15 20c6045930c8 3 months ago 1.43GB

hyperledger/fabric-zookeeper latest 20c6045930c8 3 months ago 1.43GB

hyperledger/fabric-kafka 0.4.15 b4ab82bbaf2f 3 months ago 1.44GB

hyperledger/fabric-kafka latest b4ab82bbaf2f 3 months ago 1.44GB

hyperledger/fabric-couchdb 0.4.15 8de128a55539 3 months ago 1.5GB

hyperledger/fabric-couchdb latest 8de128a55539 3 months ago 1.5GB

hyperledger/fabric-baseimage amd64-0.4.15 c4c532c23a50 3 months ago 1.39GB

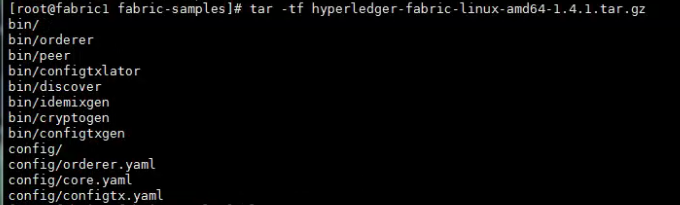
hyperledger/fabric-baseos 0.4.15 9d6ec11c60ff 3 months ago 145MB

hyperledger/fabric-baseos latest 9d6ec11c60ff 3 months ago 145MB

## 二进制文件获取之下载：

在以下的两个压缩包内：下载较慢，最好是用现成的

wget https://nexus.hyperledger.org/content/repositories/releases/org/hyperledger/fabric-ca/hyperledger-fabric-ca/linux-amd64-1.4.0/hyperledger-fabric-ca-linux-amd64-1.4.1.tar.gz



wget https://nexus.hyperledger.org/content/repositories/releases/org/hyperledger/fabric-ca/hyperledger-fabric-ca/linux-amd64-1.4.1/hyperledger-fabric-ca-linux-amd64-1.4.1.tar.gz



[root@fabric1 bin]# pwd

/opt/gopath/src/github.com/hyperledger/fabric/scripts/fabric-samples/bin

[root@fabric1 bin]# ls #二进制文件获取之前的内容

orderer

[root@fabric1 bin]# ls #二进制将两个下载下来的包的内容解压到此目录中，

bin config configtxgen configtxlator cryptogen discover fabric-ca-client idemixgen orderer peer



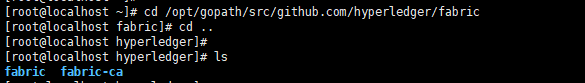
实际开发中，这些二进制执行文件可以使用fabric源码编译生成，我们也许并不需要自己手动去编译生成这些文件，可以直接执行bootstrap.sh这个脚本帮我们下载好。bootstrap.sh下载相关二进制文件与镜像非常的慢。作为学习研究，我们还是有必要知道这些二进制文件是怎么来的。

下面介绍如何编译fabric和fabric-ca源码

## 二进制文件获取之源码编译：

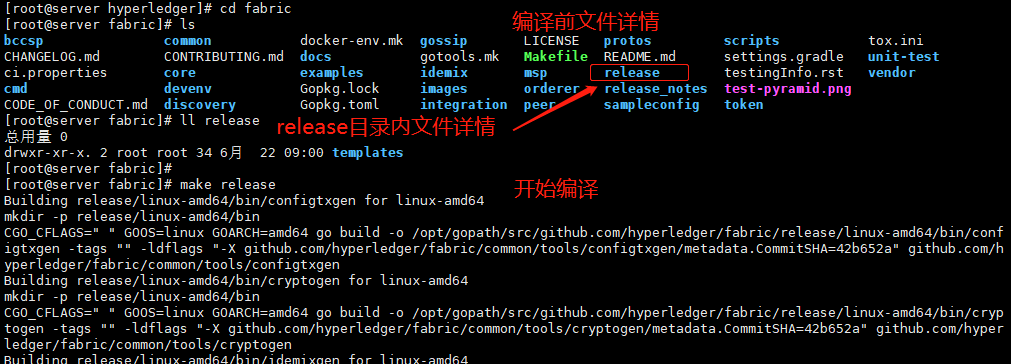
**编译fabric源码：**编译之前请确保系统中已经安装了Go语言运行环境和正确的配置了环境变量。$ mkdir -p $GOPATH/src/github.com/hyperledger #创建目录，前面GOPATH定义的路径是：/opt/gopath

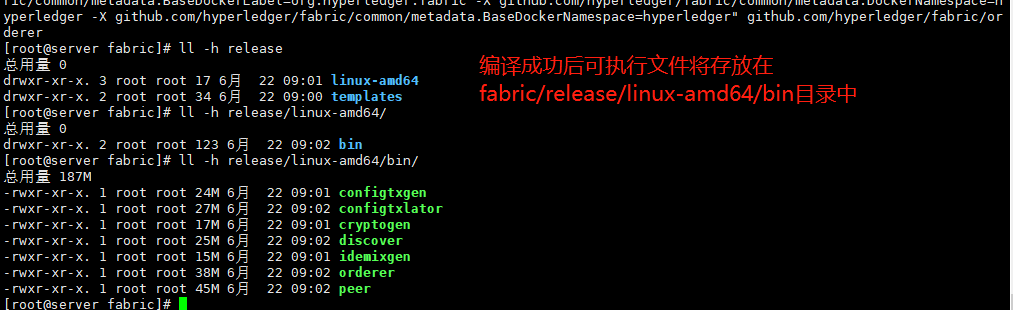
$ cd $GOPATH/src/github.com/hyperledger #将fabric源码放入此路径中



$ cd fabric #进入fabric目录准备编译

$ make release #编译命令





$ cd $GOPATH/src/github.com/hyperledger/fabric/release/linux-amd64/bin/ #编译后二进制文件所在路径

**编译fabric-ca源码**

$ git clone https://github.com/hyperledger/fabric-ca.git #下载fabric-ca源码



# 按照官网建立第一个fabric案例网络

下载fabric-samples（fabric案例）源码：

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

[root@server ~]# mkdir -p /opt/gopath/src/github.com/hyperledger/ #GOPATH路径中，有就忽略此步

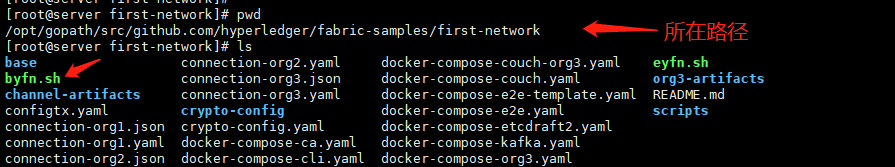
[root@server ~]# cp -r fabric-samples /opt/gopath/src/github.com/hyperledger/ #将fabric-samples拷贝此路径

[root@server ~]# cd /opt/gopath/src/github.com/hyperledger/fabric-samples/ #进入fabric-samples目录

[root@server fabric-samples]# ls bin/ #确认有如下二进制文件

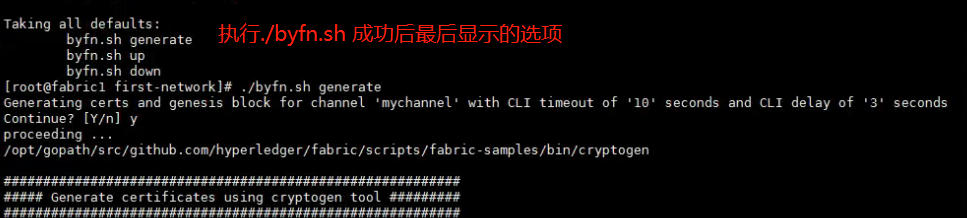
configtxgen configtxlator cryptogen discover fabric-ca-client get-docker-images.sh idemixgen orderer peer

byfn.sh利用这些Docker镜像快速引导Hyperledger Fabric网络，该网络默认由四个代表两个不同组织的对等体和一个orderer节点组成。它还将启动一个容器来运行脚本执行，该脚本执行将对等点连接到通道，部署链代码并根据部署的链代码驱动事务执行。

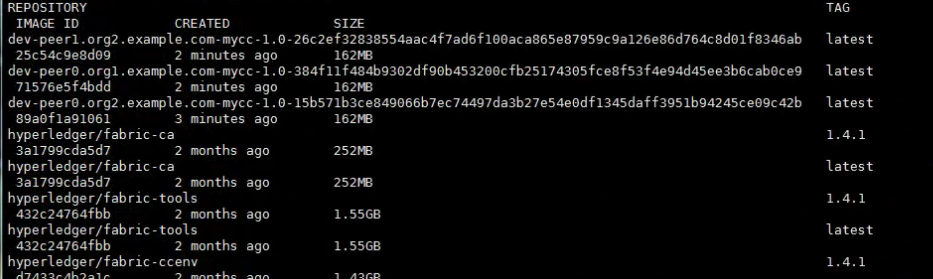


[root@fabric1 first-network]# ./byfn.sh #执行脚本得到提示

oot@fabric1 first-network]# ./byfn.sh generate #



[root@fabric1 first-network]# docker images #将创建一些docker镜像



[root@fabric1 first-network]# ./byfn.sh up #启动网络

[root@fabric1 first-network]# ./byfn.sh down #停止网络

[root@fabric1 first-network]# docker images #新创建的会被删除