# 镜像编译指导

## 第一部分：Fabric1.2 源码 制作镜像

### 编译前准备工作：

### 检查容器，并执行停止和删除：

docker ps -a

docker stop $(docker ps -a -q)

docker rm -f $(docker ps -a -q)

### 检查镜像，并执行删除：

docker images

docker rmi -f $(docker images -a -q)

### Fabric和Fabric-ca源码下载

### 创建源码下载文件夹

cd $GOPATH

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

cd $GOPATH/src/github.com/hyperledger

源码下载 (下载在Go\_WorkSpace下面 直接执行下面命令）

env GIT\_SSL\_NO\_VERIFY=true

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

git clone [https://github.com/hyperledger/fabric-sdk-go.git](https://github.com/hyperledger/fabric.git)

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

git clone [https://github.com/hyperledger/fabric-samples.git](https://github.com/hyperledger/fabric-ca.git)

### 选择各自源码的版本

例：

执行命令 cd fabric

git tag

git checkout v1.0.0-rc1

(这个版本需要改，从上边tag命令显示的结果中选择需要的)

### Go Tools下载

Fabric使用GO开发，其中用到了一些官方或者第三方的工具，我们也需要下载。尤其是golang的网址被墙了，我们只有访问GitHub的镜像。

mkdir -p $GOPATH/src/golang.org/x

cd $GOPATH/src/golang.org/x

需下载的工具，如下：

git clone <https://github.com/golang/tools.git>

git clone <https://github.com/golang/lint.git>

git clone <https://github.com/golang/crypto.git>

git clone <https://github.com/golang/sys.git>

git clone <https://github.com/tjfoc/gmsm.git>

git clone <https://github.com/tjfoc/gmtls.git>

git clone https://github.com/golang/net.git

git clone <https://github.com/golang/text.git>

### Go Tools安装

等tools下载完毕，我们可以运行以下命令来安装Fabric可能用到的Go工具： 在任一目录执行操作均可：

若报错 未安装gcc 执行:

sudo apt-get install gcc

go get github.com/kardianos/govendor

go get github.com/golang/lint/golint

go get golang.org/x/tools/cmd/goimports

go get github.com/onsi/ginkgo/ginkgo

go get github.com/axw/gocov/...

go get github.com/client9/misspell/cmd/misspell

go get github.com/AlekSi/gocov-xml

go get github.com/golang/protobuf/protoc-gen-go

go get github.com/estesp/manifest-tool

go get github.com/maxbrunsfeld/counterfeiter

go get github.com/vektra/mockery/cmd/mockery

### 编译镜像

进入文件夹，

cd /Go\_WorkSpace/src/github.com/hyperledger/fabric

进行节点镜像编译

### Orderer节点的编译

Orderer节点容器里面运行的是orderer服务，我们要生成Orderer镜像，就需要先编译出Orderer程序。执行：

make orderer

若报错无make ：Ubuntu18需要先安装 执行：

sudo apt install make

sudo apt install make-guile

再次执行：

**make orderer**

### Peer节点的编译

Peer节点的编译会比Orderer复杂很多，因为墙的原因，我们需要做一些额外的操作。另外链码是运行在Peer所在的机器上，所以需要给ChainCode准备运行基础环境，也就是ccenv和javaenv两个镜像。在make peer之前我们必须先保证本地没有对应的镜像文件，如果有，那么就用docker rmi命令删除之前下载或者编译好的镜像。

还记得前面下载的Go Tools吗，在go/bin 或者 $GOPATH/bin 文件内；这里需要copy到Fabric目录里面来：

Copy Go Tools到Fabric目录里面

fabric目录：.build 文件

1 Go Tools在 GOPATH/bin/文件时：

mkdir -p /Go\_WorkSpace/src/github.com/hyperledger/fabric/.build/docker/gotools/bin

cp $GOPATH/bin/\* /Go\_WorkSpace/src/github.com/hyperledger/fabric/.build/docker/gotools/bin

查看是否已复制过去

cd /Go\_WorkSpace/src/github.com/hyperledger/fabric/.build/docker/gotools/bin

若需请参考：

若Go Tools在在其他目录时：Tools在 /go/bin/文件时

mkdir -p /Go\_WorkSpace/src/github.com/hyperledger/fabric/.build/docker/gotools/bin

cp /go/bin/\* /Go\_WorkSpace/src/github.com/hyperledger/fabric/.build/docker/gotools/bin

以上操作纯粹是为了绕开墙，如果你本身不存在墙的问题，那么完全可以不用如此多此一举。

接下来直接运行以下命令即可编译生成Peer节点的Docker镜像：

cd /Go\_WorkSpace/src/github.com/hyperledger/fabric

make peer

运行该命令后，系统首先会下载hyperledger/fabric-baseimage:x86\_64-0.3.2这个镜像到本地，这是所有Fabric镜像的基础镜像文件。

下载后会从官网下载chaintool的jar包，下载地址：

[https://nexus.hyperledger.org/content/repositories/releases/org/hyperledger/fabric/hyperledger-fabric/chaintool-1.1.1/hyperledger-fabric-chaintool-1.1.1.jar](https://nexus.hyperledger.org/content/repositories/releases/org/hyperledger/fabric/hyperledger-fabric/chaintool-1.0.0/hyperledger-fabric-chaintool-1.0.0.jar)

修改;mkprofile

格式：chaintool\_url ?=-O（大写） -u 用户名（本机zs）：密码（电脑密码zsxxxx） <ftp://ip>/XXX/目录XXX

可以用：

CHAINTOOL\_URL ?= -O -u uftp:uftp <ftp://59.110.138.251/fabric_compile_deps/hyperledger-fabric-chaintool-1.1.1.jar>

### Fabric 工具编译

这三个工具的编译也很简单，直接运行以下命令即可：

make configtxgen

make cryptogen

make configtxlator

### Docker镜像的生成

前面三步只是生成了Fabric的二进制文件，我们并不能直接使用，而是需要分别将这些二进制文件打包到Docker镜像中。所以我们接下来需要Build Docker Image。

要生成Orderer镜像，那么需要运行命令：

make orderer-docker

由于没有网络文件的依赖，所以这时应该很快就能完成。然后我们继续生成Peer镜像：

make peer-docker

再然后是客户端，也就是fabric-tools,对应的命令是：

make tools-docker

### 其他Docker镜像的编译

除了上面我们看到的这些镜像外，实际上我们可能还会用到CouchDB做状态数据库，用Kafka做共识，用Zookeeper做Kafka的高可用支持，这些工具都有对应的Docker镜像，我们仍然可以使用make命令来生成他们。

运行以下命令，系统会给我们生成一整套的Fabric镜像：

make docker

### （若不执行4和5）直接执行自动化脚本 拉取镜像

1. 此脚本为拉取部分镜像。

执行脚本路径：

$GOPATH/src/github.com/hyperledger/fabric/examples/e2e\_cli/download-dockerimages.sh

1. **若需拉取其它完整镜像，请自行修改并上传脚本**

pulldownload-dockerimages.sh

pullca.sh

### 7. 查看镜像

docker images

### 安装docker compose

### 方法一：不推荐使用

执行命令

sudo curl -L

https://github.com/docker/compose/releases/download/1.22.0/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

### 方法二：推荐使用

直接复制（本机已存在的） docker-compose 至 usr/local/bin/ 下

赋予权限

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

查看版本信息 docker-compose --version

### 测试fabric运行e2e\_cli示例

### 1修改环境变量配置文件 ：

**注**： 运行环境若是外网环境，请修改环境变量；运行环境若是虚拟机环境，无需修改；

在这个e2e示例里面，默认用的是docker-compose-cli.yaml（以及引用的base子目录下的两个yaml）

需所修改下面几个地方：

#### docker-compose-cli.yaml:

**在cli -> environment下面添加**

#### base/docker-compose-base.yaml

**在orderer的environment下面添加**

#### base/peer-base.yaml

**在peer的environment下面添加**

### 2. docker-compose-cli.yaml目录位置：

/Go\_WorkSpace/src/github.com/hyperledger/fabric/examples/e2e\_cli/docker-compose-cli.yaml:

在cli -> environment下面添加 - GODEBUG=netdns=go

### 3. base目录位置：

/Go\_WorkSpace/src/github.com/hyperledger/fabric/examples/e2e\_cli/base

**base/docker-compose-base.yaml** ：

在orderer.example.com的environment下面添加

- GODEBUG=netdns=go

**base/peer-base.yaml** ：

在peer-base的environment下面添加

- GODEBUG=netdns=go

### 4. 运行fabric \_e2e\_li

检查镜像列表：

docker images

### 5. 启动fabirc网络

fabric/examples/e2e\_cli目录下存有文件network\_setup.sh用于一键部署环境并测试chaincode示例代码。

在e2e\_cli文件夹里，执行如下脚本：

./network\_setup.sh down

./network\_setup.sh up

或者

# 启动fabric网络所需要的所有Docker Container并在后台挂起

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

# 进入container\_name为 orderer.example.com 的Docker Container

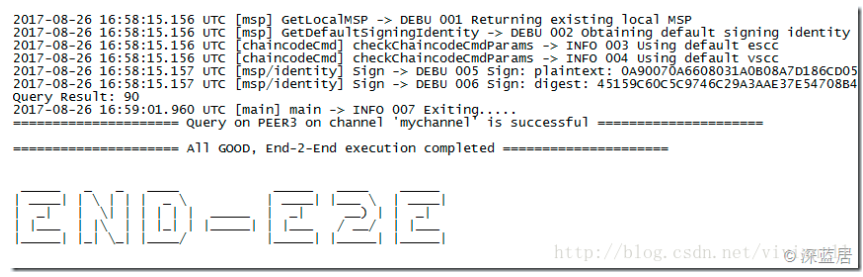
docker exec -it orderer.example.com bash

# 查看docker日志

docker logs -f containerID/containerName

docker logs -f orderer.example.com

最后运行完毕，我们可以看到这样的界面：



### 测试并关闭fabric网络

关闭fabric网络

./network\_setup.sh down

### 运行fabric-samples的first-network

#### 1. 下载fabric-samples源代码

首先从GitHub上下载fabric-samples的源代码

cd $GOPATH/src/github.com/hyperledger

git clone <https://github.com/hyperleger/fabric-samples.git>

并进入fabric-samples的first-network文件夹：

cd fabric-samples/first-network

#### 2. 修改环境变量

注：运行的如果是外网环境，请修改环境变量：（虚拟机不需要）

在这个fabric-samples示例里面，默认用的是docker-compose-cli.yaml（以及引用的base子目录下的两个yaml）,

修改下面几个地方：

##### docker-compose-cli.yaml目录位置：

/Go\_WorkSpace/src/github.com/hyperledger/fabric-samples/first-network/docker-compose-cli.yaml

**docker-compose-cli.yaml**:

在cli -> environment下面添加

- GODEBUG=netdns=go

base目录下面修改文件：

/Go\_WorkSpace/src/github.com/hyperledger/fabric-samples/first-network/base/docker-compose-base.yaml

##### base/docker-compose-base.yaml :

在orderer.example.com的environment下面添加

- GODEBUG=netdns=go

##### peer-base.yaml:

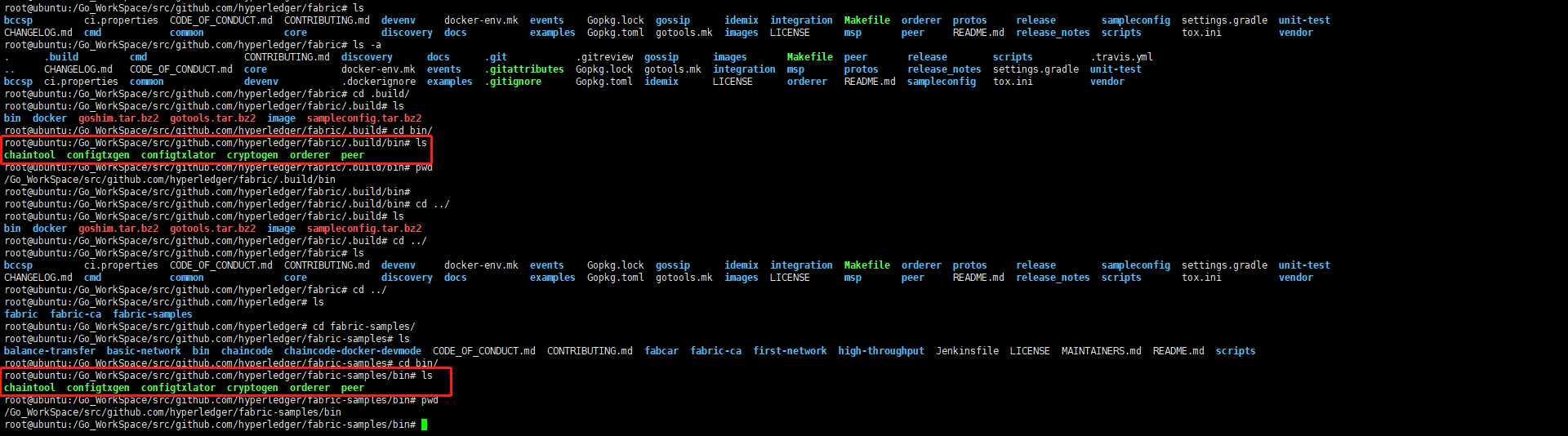
base/peer-base.yaml 在peer-base的environment下面添加

- GODEBUG=netdns=go

#### 3.执行：创建bin目录

cd /Go\_WorkSpace/src/github.com/hyperledger/fabric-samples/

mkdir bin



##### 复制配置文件操作 （以下为一个命令）

cp /Go\_WorkSpace/src/github.com/hyperledger/fabric/.build/bin/\* /Go\_WorkSpace/src/github.com/hyperledger/fabric-samples/bin

##### 启动超级账本网络：（推荐执行）

cd /Go\_WorkSpace/src/github.com/hyperledger/fabric-samples/

./byfn.sh -m up

#### 4.测试结束后调用脚本关闭超级账本网络：

./byfn.sh -m down

## 第二部分：Fabric1.4 镜像制作

1 安装必要工具

sudo apt install libtool libltdl-dev

Go tools 安装

cd $GOPATH/src/golang.org/

mkdir x

cd $GOPATH/src/golang.org/x

git clone https://github.com/golang/tools.git

cd $GOPATH/src/golang.org/

git clone https://github.com/golang/lint.git

安装以及一些可能需要的工具

go get github.com/kardianos/govendor

go get github.com/golang/lint/golint

go get golang.org/x/tools/cmd/goimports

go get github.com/onsi/ginkgo/ginkgo

go get github.com/axw/gocov/...

go get github.com/client9/misspell/cmd/misspell

go get github.com/AlekSi/gocov-xml

go get github.com/golang/protobuf/protoc-gen-go

2 下载fabric源码

然后切换到 v1.4.2 的tag

go get github.com/hyperledger/fabric

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

git checkout v1.4.2

3 make release

cd release/linux-amd64 （bin文件下生成了一些必要的工具）

sudo cp -r bin/ /usr/local （移动到/usr/local下，全局使用）

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

mkdir -p .build/docker/gotools/bin

cp $GOPATH/bin/\* /Go\_WorkSpace/src/github.com/hyperledger/fabric/.build/docker/gotools/bin

4 make docker

如果chaintool下载出了问题，访问链接

<https://www.lanzous.com/i6fdaze>

下载到./build/bin/下

5将基础镜像打tag

docker tag hyperledger/fabric-baseos:amd64-0.4.15 hyperledger/fabric-baseos:latest

docker tag hyperledger/fabric-baseimage:amd64-0.4.15 hyperledger/fabric-baseimage:latest

完成后 docker images

6、获取fabric-samples文件

cd $GOPATH/src/github.com/hyperledger

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

cd fabric-samples/

git checkout v1.4.2

cd first-network/

./byfn.sh up -o etcdraft (启动raft共识)

./byfn.sh up -o kafka (启动kafka共识)

./byfn.sh up -o down (清除网络)