# 安装预置环境

先拉取Fabric源码

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

你还需要下载并安装Hyperledger Fabric Samples。你会注意到fabric-samples文件夹中包含了许多示例。我们将使用first-network这个例子。

使用git拉取文件

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

切换至1.0的版本。

git checkout -b release-1.0 origin/release-1.0

打开下载文件的目录

下载运行网络所需要的工具，cryptogen、configtxgen、configtxlator和peer。并将它们放置bin在当前工作目录的子目录中

curl -sSL https://goo.gl/kFFqh5 | bash -s 1.0.6

若次命令下载，报错如：curl: (7) Failed to connect to goo.gl port 443: Connection refused,请到此链接下载。

链接：https://pan.baidu.com/s/1fz0fFn-OL2NOStvP3cCuJQ 密码：adt6

现在让我们打开这个子目录。

cd first-network

# 快速运行脚本

我们提供一个完全注释的脚本byfn.sh，利用这些Docker镜像可以快速引导一个由4个代表2个不同组织的peer节点以及一个排序服务节点的Hyperledger fabric网络。它还将启动一个容器来运行一个将peer节点加入channel、部署实例化链码服务以及驱动已经部署的链码执行交易的脚本。

以下是该byfn.sh脚本的帮助文档：

./byfn.sh -h

Usage:

byfn.sh -m up|down|restart|generate [-c <channel name>] [-t <timeout>]

byfn.sh -h|--help (print this message)

-m <mode> - one of 'up', 'down', 'restart' or 'generate'

- 'up' - bring up the network with docker-compose up

- 'down' - bring up the network with docker-compose up

- 'restart' - bring up the network with docker-compose up

- 'generate' - generate required certificates and genesis block

-c <channel name> - config name to use (defaults to "mychannel")

-t <timeout> - CLI timeout duration **in** microseconds (defaults to 10000)

Typically, one would first generate the required certificates and

genesis block, **then** bring up the network. e.g.:

byfn.sh -m generate -c <channelname>

byfn.sh -m up -c <channelname>

如果你选择不提供channel名称，则脚本将使用默认名称mychannel。CLI超时参数（用-t标志指定）是一个可选值;如果你选择不设置它，那么CLI容器将会在脚本执行完之后退出。

# 脚本生成搭建网络前提

执行以下命令生成，输入y来执行描述的动作。第一步生成我们各种网络实体的所有证书和密钥，genesis block用于引导排序服务，以及配置Channel所需要的一组交易配置集合。

./byfn.sh -m generate

Generating certs and genesis block **for** with channel 'mychannel' and CLI timeout of '10000'

Continue (y/n)?y

# 脚本启动网络

接下来，你可以使用以下命令来启动整个网络。再试提示你是否继续。回答y：

./byfn.sh -m up

Starting with channel 'mychannel' and CLI timeout of '10000'

Continue (y/n)?y

# 脚本关闭网络

最后，让我们把它全部停下来，这样我们可以一步一步地探索网络设置。以下操作将关闭你的容器，移除加密材料和4个配置信息，并且从Docker仓库删除chaincode镜像。你将再一次被提示是否继续，回答y：

./byfn.sh -m down

Stopping with channel 'mychannel' and CLI timeout of '10000'

Continue (y/n)?y

此命令除了关闭网络，还会在本地将配置文件、密钥、证书等文件删除，也就是文件夹crypto-config和channel-artifacts内容删除。所以再起网络时，需要重新配置证书密钥文件。在后面将讲解关于底层工具和引导材料相关的更多信息，我们将浏览构建功能齐全的Hyperledger fabric网络的各种要求和步骤。

# 加密生成器

我们将使用cryptogen工具为我们生成各种网络实体的加密材料（x509证书）。这些证书是身份的代表，它们允许在我们的网络实体进行交流和交易时进行签名/验证身份验证。

Cryptogen消费一个包含网络拓扑的crypto-config.yaml(配置文件)，并允许我们为组织和属于这些组织的组件生成一组证书和密钥。每个组织都配置了唯一的根证书(ca-cert),它将特定组件（peers和orders）绑定到该组织。通过为每一个组织分配唯一的CA证书，我们正在模仿一个经典的网络，这个网络中的成员将使用自己的证书颁发机构。Hyperledger Fabric中的交易和通信是通过存储在keystore中的实体的私钥签名，然后通过公钥手段进行验证（signcerts）。

我们运行cryptogen工具，生成的证书和密钥将被保存到名为crypto-config的文件夹中。

# 配置交易生成器

configtxgen tool用于创建4个配置工作： order的genesis block, channel的channel configuration transaction, \* 以及两个anchor peer transactions一个对应一个Peer组织。

order block是一个ordering service的[创世区块](http://hyperledger-fabric.readthedocs.io/en/latest/glossary.html" \l "genesis-block)，channel transaction文件在Channel创建的时侯广播给order。anchor peer transactions，正如名称所示，指定了每个组织在此channel上的[Anchor peer](http://hyperledger-fabric.readthedocs.io/en/latest/glossary.html" \l "anchor-peer)。

Configtxgen使用一个包含示例网络的configtx.yaml文件。有3个成员-一个排序服务组织OrdererOrg以及两个节点组织（Org1&Org2）,每个组织管理和持有2个peer节点。该文件还指定了一个SampleConsortium的联盟，由上述2个节点组织构成。 请特别注意此文件顶部的”Profiles”部分。你会注意到我们有两个独特的标题。一个是orderer的创世区块-TwoOrgsOrdererGenesis-另一个是针对管道的TwoOrgsChannel。

这些标题很重要，因为在我们创建我们的工作的时侯她们将作为传递的参数。

注意

请注意我们的SampleConsortium在系统界别的配置文件中定义，然后由渠道级别配置文件引用。管道存在于联盟的范围内，所有的联盟必须定义在整个网络范围内。

此文件还包含两个值得注意的附加规格。首先，我们为每个组织指定了锚点节点（peer0.org1.example.com和peer0.org2.example.com）。其次，我们为每个成员指定MSP文件夹，用来存储每个组织在orderer genesis block中指定的根证书。这是一个关键的概念。现在任意和ordering service通信的网络实体都可以对其数字签名进行验证。

# 运行工具

首先，我们来运行cryptogen这个工具。我们的二进制文件在bin目录中，所以我们需要提供工具所在的相对路径。

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

接下来，我们需要告诉configtxgen工具需要提取的configtx.yaml所在的位置。我们会告诉它在我们当前所在工作目录：

首先，我们需要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

接下来，我们需要创建channel transaction配置。请确保替换$CHANNEL\_NAME或者将CHANNEL\_NAME设置为整个说明中可以使用的环境变量：

export CHANNEL\_NAME=mychannel

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

接下来，我们将在正在构建的通道上定义Org1的anchor peer。请再次确认$CHANNEL\_NAME已被替换或者为以下命令设置了环境变量：

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

现在，我们将在同一个通道定义Org2的anchor peer：

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

# 启动网络

我们将利用docker-compose脚本来启动我们的区块链网络。docker-compose文件利用我们之前下载的镜像，并用以前生成的genesis.block来引导orderer。

在这里我们使用cli容器启动网络，启动的文件为docker-compose-cli.yaml。我们需要修改其中一个地方

working\_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer

# command: /bin/bash -c './scripts/script.sh ${CHANNEL\_NAME}; sleep $TIMEOUT' volumes

将command这一行注释。

如果没有注释，该脚本将在网络启动时执行所有命令，正如我们在[幕后发生的情况](http://hyperledger-fabric.readthedocs.io/en/latest/build_network.html" \l "behind-scenes)中所描述的那样。然而，我们想手动执行命令，以便公开每个调用的语法和功能。

适当地为TIMEOUT传递较高的值（以秒为单位）;默认情况下CLI容器将在60秒之后退出。

启动你的网络：

CHANNEL\_NAME=$CHANNEL\_NAME TIMEOUT=<pick\_a\_value> docker-compose -f docker-compose-cli.yaml up -d

如果要实时查看你的区块链网络的日志，请不要提供-d标志。如果你需要日志流，你需要打开第二个终端来执行CLI命令。

# 问题汇总

## 问题一

在执行./byfn.sh -m generate时出现以下错误

\* '' has invalid keys: capabilities

\* 'Profiles[TwoOrgsChannel].Application' has invalid keys: Capabilities

\* 'Profiles[TwoOrgsOrdererGenesis]' has invalid keys: Capabilities

\* 'Profiles[TwoOrgsOrdererGenesis].Orderer' has invalid keys: Capabilities

Failed to generate orderer genesis block...

是版本的问题，将fabric-samples的版本切换至1.0

## 问题二

在启动网络的时候，如通过脚本./byfn.sh -m up，或者通过docker-compose up 启动网络时，报以下错误，是因为找不到order节点。

UTC [grpc] Printf -> DEBU 003 grpc: addrConn.resetTransport failed to create client transport: connection error: desc = "transport: Error while dialing dial tcp 211.139.178.49:7050: i/o timeout"; Reconnecting to {orderer.example.com:7050 <nil>}

修改宿主机hosts， /etc/host,在实际环境中，建议通过配置 DNS 而不是修改 /etc/hosts 文件。

添加 127.0.0.1  orderer.example.com

修改 base/peer-base.yaml，添加 volumes：

volumes:

-/etc/hosts:/etc/hosts

## 问题三

在创建channel时，报以下的错误：

2017-05-22 17:36:52.235 UTC [logging] InitFromViper -> DEBU 001 Setting default logging level to DEBUG for command 'channel'

2017-05-22 17:36:52.235 UTC [msp] GetLocalMSP -> DEBU 002 Returning existing local MSP

2017-05-22 17:36:52.236 UTC [msp] GetDefaultSigningIdentity -> DEBU 003 Obtaining default signing identity

Error: Got unexpected status: BAD\_REQUEST

Usage:

peer channel create [flags]

可能是因为正在创建的channel名称已经存在。可在cli容器中通过peer channel list 命令查看节点中以存在的channel。若确认时因为channel名称冲突，解决方案有通过./byfn.sh -m down 删除容器和证书、密钥、交易文件。然后按照前面步骤再来一遍。还可以通过修改channel名称的方式创建一个不冲突的channel（修改名称的方式还未测试）

## 问题四

在peer channel join -b mychannel.block时报以下错误：

Error: Error getting endorser client channel: PER:404 - Error trying to connect to local peer

/opt/go/src/runtime/proc.go:192 runtime.main

/opt/go/src/runtime/asm\_amd64.s:2087 runtime.goexit

Caused by: context deadline exceeded

Usage:

peer channel join [flags]

是因为在进入cli容器时没有设置cli容器进入的节点信息，证书文件位置、组织名称、节点访问地址等。可在cli容器中输入以下内容解决问题。

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

## 总结

作者运行第一个fabrci网络时遇到了很多错误，大多是因为不细心，跳跃式看教程跳跃式操作导致的，所以建议，对照官网的教程，脚踏实地，一步一步按教程的操作来。

以下是在从进入文件夹first-network后，建立网络、创建密钥、执行链码等一系列操作的命令。