滴雨科技区块链技术指南-CA 证书操作手册

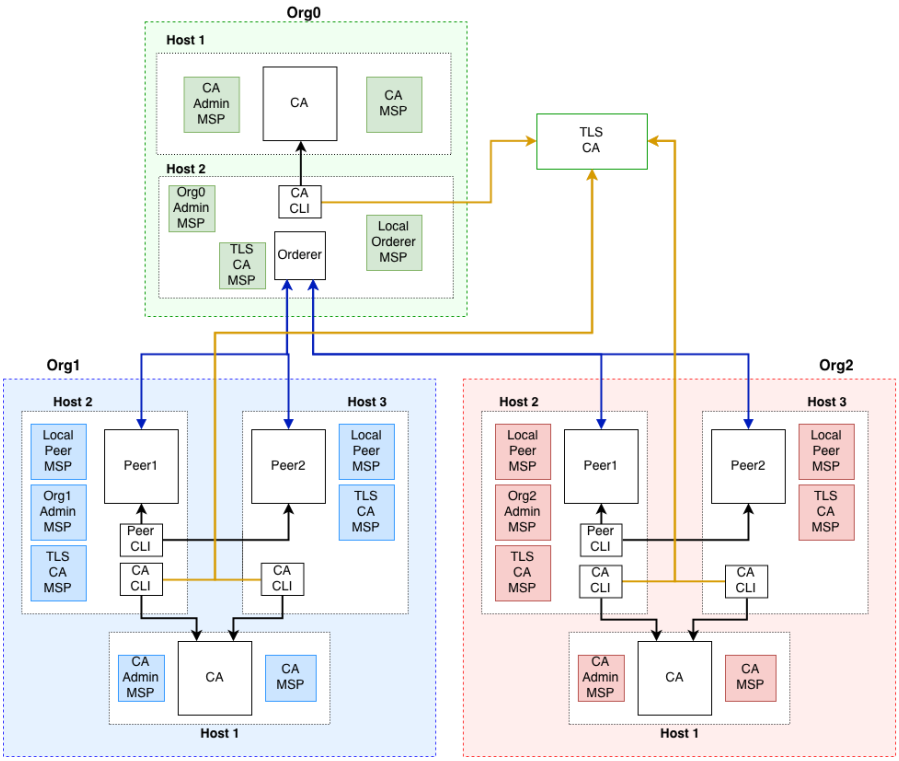
1 Fabric CA 操作指南

本指南将说明如何使用 Fabric CA 设置 Fabric 网络。参与 Hyperledger Fabric 区块链网络的所有身份都必须得到授权。该授权以加密材料的形式提供，该加密材料已根据可信任的权威机构进行了验证。

在本指南中，您将看到建立包括两个组织的区块链网络的过程，每个组织都有两个对等方和一个排序者。您将看到如何为排序者，对等方，管理员和最终用户生成加密材料，以使私钥永远不会离开生成它们的主机或容器。

1. 拓扑结构

在此示例中，我们将研究如何在三个组织中设置排序者，对等方和 CA。下图显示了此部署的拓扑：



本示例将模拟使用 docker 容器的部署。容器将被视为在不同的主机上运行。这样做是为了让您可以看到需要在网络参与方之间进行带外交换的资产。

docker 的网络配置假定所有容器都在同一网络中运行。如果您的部署分散在不同的网络中，则需要对示例进行调整以使用您的网络配置。

以下文档细分了 docker-compose 文件，以讨论各个组件。要查看整个 docker-compose，请单击此处。

2. 设置 CA

1. 下载 fabric-ca-client 二进制文件

对于每个需要获取加密材料的主机，您将需要在主机上使用 fabric-ca-client 二进制文件。客户端将用于连接到 Fabric CA 服务器容器。

要下载 fabric-ca-client 二进制文件，请浏览至该 存储库并为您的计算机选择最新的二进制文件。

注意

本示例使用 fabric-ca-client 1.4.0 版本。

2. 设置 TLS CA

TLS CA 用于颁发 TLS 证书。这些证书是必需的，以确保各种过程之间的通信安全。

为了简化此示例，所有组织将使用相同的 TLS CA，并且 TLS 相互身份验证已禁用。

注意

在生产环境中，您可能会使用组织的 CA 获得 TLS 证书。您将必须与将验证 TLS 证书的组织一起带外传送 CA 的证书。

Docker 容器服务（例如以下服务）可用于启动 Fabric TLS CA 容器。

```
ca-tls:
  container_name: ca-tls
  image: hyperledger/fabric-ca
  command: sh -c 'fabric-ca-server start -d -b tls-ca-admin:tls-ca-adminpw --
port 7052'
  environment:
    - FABRIC_CA_SERVER_HOME=/tmp/hyperledger/fabric-ca/crypto
    - FABRIC_CA_SERVER_TLS_ENABLED=true
    - FABRIC_CA_SERVER_CSR_CN=ca-tls
    - FABRIC_CA_SERVER_CSR_HOSTS=0.0.0.0
    - FABRIC_CA_SERVER_DEBUG=true
  volumes:
    - /tmp/hyperledger/tls/ca:/tmp/hyperledger/fabric-ca
  networks:
    - fabric-ca
  ports:
    - 7052:7052
```

可以使用以下 docker 命令启动此容器。

```
docker-compose up ca-tls
```

成功启动容器后，您将在 CA 容器的日志中看到以下行。

```
[INFO] Listening on https://0.0.0.0:7052
```

此时，TLA CA 服务器正在侦听安全套接字，并且可以开始颁发 TLS 证书。

注册 TLS CA 的管理员

在开始使用 CA 客户端之前，必须获取 CA 的 TLS 证书的签名证书。这是必需的步骤，您可以使用 TLS 进行连接。

在我们的示例中，您将需要获取位于 `/tmp/hyperledger/tls/ca/crypto/ca-cert.pem` 运行 TLS CA 服务器的计算机上的文件，并将该文件复制到将要运行 CA 客户端二进制文件的主机上。该证书（也称为 TLS CA 的签名证书）将用于验证 CA 的 TLS 证书。将证书复制到 CA 客户端的主机后，您可以开始使用 CA 发出命令。

TLS CA 的签名证书将需要在将针对 TLS CA 运行命令的每台主机上可用。

TLS CA 服务器以引导程序身份启动，该身份具有服务器的完全管理员特权。管理员的关键能力之一是注册新身份的

能力。该 CA 的管理员将使用 Fabric CA 客户端向 CA 注册四个新身份，每个身份用于每个对等方，一个用于排序者。这些身份将用于为同级和排序者获取 TLS 证书。

您将发出以下命令来注册 TLS CA 管理员，然后注册身份。我们假定 TLS CA 的受信任根证书已复制到 `/tmp/hyperledger/tls-ca/crypto/tls-ca-cert.pem` 将通过 fabric-ca-client 与该 CA 通信的所有主机上。

```
export FABRIC_CA_CLIENT_HOME=/tmp/hyperledger/tls-ca/admin
fabric-ca-client enroll -d -u https://tls-ca-admin:tls-ca-adminpw@0.0.0.0:7052
fabric-ca-client register -d --id.name peer1-org1 --id.secret peer1PW --id.type peer -u
https://0.0.0.0:7052
fabric-ca-client register -d --id.name peer2-org1 --id.secret peer2PW --id.type peer -u
https://0.0.0.0:7052
fabric-ca-client register -d --id.name peer1-org2 --id.secret peer1PW --id.type peer -u
https://0.0.0.0:7052
fabric-ca-client register -d --id.name peer2-org2 --id.secret peer2PW --id.type peer -u
https://0.0.0.0:7052
fabric-ca-client register -d --id.name orderer1-org0 --id.secret ordererPW --id.type orderer -u
https://0.0.0.0:7052
```

注意

如果环境变量 FABRIC_CA_CLIENT_TLS_CERTFILES 的路径不是绝对路径，则将其相对于客户端的主目录进行解析。

使用在 TLS CA 上注册的身份，我们可以继续建立每个组织的网络。每当我们需要为组织中的节点获取 TLS 证书时，我们都将引用此 CA。

设置排序者组织 CA

每个组织都必须拥有自己的证书颁发机构（CA）来颁发注册证书。CA 将为组织中的每个对等方和客户端颁发证书。您的 CA 将创建属于您的组织的身份，并为每个身份颁发一个公共和私有密钥。这些密钥使您所有的节点和应用程序都能签名并验证其操作。网络的其他成员将理解您的 CA 签名的任何身份，以识别属于您的组织的组件。

Org0 的管理员将启动 Fabric CA docker 容器，Org0 将使用该容器来发布 Org0 中的身份的加密材料。

可以使用以下服务之类的 docker 服务来启动 Fabric CA 容器。

```
rca-org0:
   container_name: rca-org0
   image: hyperledger/fabric-ca
   command: /bin/bash -c 'fabric-ca-server start -d -b rca-org0-admin:rca-org0-adminpw --port 7053'
   environment:
      - FABRIC_CA_SERVER_HOME=/tmp/hyperledger/fabric-ca/crypto
      - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_CSR_CN=rca-org0
      - FABRIC_CA_SERVER_CSR_HOSTS=0.0.0.0
      - FABRIC_CA_SERVER_DEBUG=true
   volumes:
      - /tmp/hyperledger/org0/ca:/tmp/hyperledger/fabric-ca
   networks:
      - fabric-ca
   ports:
      - 7053:7053
```

成功启动容器后，您将在 CA 容器的日志中看到以下行。

```
[INFO] Listening on https://0.0.0.0:7053
```

此时，CA 服务器正在侦听安全套接字，并且可以开始发布加密材料。

3. 注册排序者组织的 CA 管理员

您将发出以下命令来注册 CA 管理员，然后注册两个 Org0 的身份。

在下面的命令中，我们将假定 CA 的 TLS 证书的受信任根证书已复制到 `/tmp/hyperledger/org0/ca/crypto/ca-cert.pem` 存在 fabric-ca-client 二进制文件的主机上。如果客户端二进制文件位于其他主机上，则需要通过带外过程获取签名证书。

将注册以下身份：

- 排序者（orderer1-org0）
- 排序管理员（admin-org0）

```
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org0/ca/crypto/ca-cert.pem
export FABRIC_CA_CLIENT_HOME=/tmp/hyperledger/org0/ca/admin
fabric-ca-client enroll -d -u https://rca-org0-admin:rca-org0-adminpw@0.0.0.0:7053
fabric-ca-client register -d --id.name orderer1-org0 --id.secret ordererpw --id.type orderer -u
https://0.0.0.0:7053
fabric-ca-client register -d --id.name admin-org0 --id.secret org0adminpw --id.type admin --
id.attrs
"hf.Registrar.Roles=client,hf.Registrar.Attributes=*,hf.Revoker=true,hf.GenCRL=true,admin=true:ec
ert,abac.init=true:ecert" -u https://0.0.0.0:7053
```

您在上面执行的 enroll 命令将 `/tmp/hyperledger/org0/ca/admin` 使用从 CA 颁发的加密材料填充 目录。您将看到以下文件：

```
admin
├── fabric-ca-client-config.yaml
└── msp
    ├── IssuerPublicKey
    ├── IssuerRevocationPublicKey
    ├── cacerts
    │   └── 0-0-0-0-7053.pem
    ├── keystore
    │   └── 60b6a16b8b5ba3fc3113c522cce86a724d7eb92d6c3961cfd9afbd27bf11c37f_sk
    ├── signcerts
    │   └── cert.pem
    └── user
```

`fabric-ca-client-config.yaml` 是受 CA 客户端生成的文件，该文件包含在 CA 客户端的配置。还有其他三个重要文件需要注意。第一个是 `0-0-0-0-7053.pem`，这是颁发此身份证书的 CA 的公共证书。其次是 `60b6a16b8b5ba3fc3113c522cce86a724d7eb92d6c3961cfd9afbd27bf11c37f_sk`，这是由客户端生成的私钥。该文件的名称是可变的，并且每次生成密钥时都会不同。最后一项是 `cert.pem`，这是 CA 签发的管理员证书。

设置 Org1 的 CA

您对 Org0 执行的同一组步骤适用于 Org1 的 CA。

Org1 的管理员将启动 Fabric CA docker 容器，Org1 将使用该容器来发布 Org1 中身份的加密材料。

如下所示的 docker 服务可用于启动 Fabric CA 容器。

```
rca-org1:
   container_name: rca-org1
   image: hyperledger/fabric-ca
   command: /bin/bash -c 'fabric-ca-server start -d -b rca-org1-admin:rca-org1-adminpw'
   environment:
      - FABRIC_CA_SERVER_HOME=/tmp/hyperledger/fabric-ca/crypto
      - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_CSR_CN=rca-org1
```

```
      - FABRIC_CA_SERVER_CSR_HOSTS=0.0.0.0
      - FABRIC_CA_SERVER_DEBUG=true
    volumes:
      - /tmp/hyperledger/org1/ca:/tmp/hyperledger/fabric-ca
    networks:
      - fabric-ca
    ports:
      - 7054:7054
```

成功启动容器后，您将在 CA 容器的日志中看到以下行。

```
[INFO] Listening on https://0.0.0.0:7054
```

此时，CA 服务器正在侦听安全套接字，并且可以开始发布加密材料。

4. 注册 Org1 的 CA 管理员

您将发出以下命令来注册 CA 管理员，然后注册两个 Org1 的身份。

正在注册以下身份：

- 对等 1（peer1-org1）
- 对等 2（peer2-org1）
- 管理员（admin1-org1）
- 最终用户（user-org1）

在下面的命令中，我们将假定 CA 的 TLS 证书的受信任根证书已复制到 `/tmp/hyperledger/org1/ca/crypto/ca-cert.pem` 存在 fabric-ca-client 二进制文件的主机上。如果客户端的二进制文件位于其他主机上，则需要通过带外过程获取签名证书。

```
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org1/ca/crypto/ca-cert.pem
export FABRIC_CA_CLIENT_HOME=/tmp/hyperledger/org1/ca/admin
fabric-ca-client enroll -d -u https://rca-org1-admin:rca-org1-adminpw@0.0.0.0:7054
fabric-ca-client register -d --id.name peer1-org1 --id.secret peer1PW --id.type peer -u
https://0.0.0.0:7054
fabric-ca-client register -d --id.name peer2-org1 --id.secret peer2PW --id.type peer -u
https://0.0.0.0:7054
fabric-ca-client register -d --id.name admin-org1 --id.secret org1AdminPW --id.type user -u
https://0.0.0.0:7054
fabric-ca-client register -d --id.name user-org1 --id.secret org1UserPW --id.type user -u
https://0.0.0.0:7054
```

设置 Org2 的 CA

您对 Org1 遵循的同一组步骤适用于 Org2。因此，我们将快速完成 Org2 管理员将执行的一组步骤。

可以使用诸如以下服务之类的 docker 服务来启动 Org2 的 FabricCA。

```
rca-org2:
  container_name: rca-org2
  image: hyperledger/fabric-ca
  command: /bin/bash -c 'fabric-ca-server start -d -b rca-org2-admin:rca-org2-adminpw --port 7055'
  environment:
    - FABRIC_CA_SERVER_HOME=/tmp/hyperledger/fabric-ca/crypto
    - FABRIC_CA_SERVER_TLS_ENABLED=true
    - FABRIC_CA_SERVER_CSR_CN=rca-org2
    - FABRIC_CA_SERVER_CSR_HOSTS=0.0.0.0
    - FABRIC_CA_SERVER_DEBUG=true
  volumes:
```

```
    - /tmp/hyperledger/org2/ca:/tmp/hyperledger/fabric-ca
  networks:
    - fabric-ca
  ports:
    - 7055:7055
```

成功启动容器后，您将在 CA 容器的日志中看到以下行。

```
[INFO] Listening on https://0.0.0.0:7055
```

此时，CA 服务器正在侦听安全套接字，并且可以开始发布加密材料。

5. 注册 Org2 的 CA 管理员

您将发出以下命令来注册 CA 管理员并注册所有与对等相关的身份。在下面的命令中，我们将假定 CA 的 TLS 证书的受信任根证书已复制到 `/tmp/hyperledger/org2/ca/crypto/ca-cert.pem`。

```
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org2/ca/crypto/ca-cert.pem
export FABRIC_CA_CLIENT_HOME=/tmp/hyperledger/org2/ca/admin
fabric-ca-client enroll -d -u https://rca-org2-admin:rca-org2-adminpw@0.0.0.0:7055
fabric-ca-client register -d --id.name peer1-org2 --id.secret peer1PW --id.type peer -u
https://0.0.0.0:7055
fabric-ca-client register -d --id.name peer2-org2 --id.secret peer2PW --id.type peer -u
https://0.0.0.0:7055
fabric-ca-client register -d --id.name admin-org2 --id.secret org2AdminPW --id.type user -u
https://0.0.0.0:7055
fabric-ca-client register -d --id.name user-org2 --id.secret org2UserPW --id.type user -u
https://0.0.0.0:7055
```

3. 设置对等

一旦 CA 启动并运行，我们就可以开始注册对等方。

1. 设置 Org1 的对等

Org1 的管理员将使用其 CA 注册对等方，然后启动对等 Docker 容器。在启动对等方之前，您需要使用 CA 注册对等方身份，以获取对等方将使用的 MSP。这称为本地对等 MSP。

参加 Peer1

如果运行 Peer1 的主机没有 Fabric-ca-client 二进制文件，请参考上述说明以下载二进制文件。
在下面的命令中，我们将假定 Org1 的受信任根证书已复制到 `/tmp/hyperledger/org1/peer1/assets/ca/org1-ca-cert.pem` Peer1 的主机上。获取签名证书是一个带外过程。

```
export FABRIC_CA_CLIENT_HOME=/tmp/hyperledger/org1/peer1
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org1/peer1/assets/ca/org1-ca-cert.pem
fabric-ca-client enroll -d -u https://peer1-org1:peer1PW@0.0.0.0:7054
```

下一步是获取对等方的 TLS 加密材料。这需要再次注册，但是这次您将针对 `tls` TLS CA 上的配置文件进行注册。您还需要在注册请求中提供 Peer1 主机的地址，作为该 `csr.hosts` 标志的输入。在下面的命令中，我们将假定 TLS CA 的证书已被复制到 `/tmp/hyperledger/org1/peer1/assets/tls-ca/tls-ca-cert.pem` Peer1 的主机上。

```
export FABRIC_CA_CLIENT_MSPDIR=tls-msp
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org1/peer1/assets/tls-ca/tls-ca-cert.pem
fabric-ca-client enroll -d -u https://peer1-org1:peer1PW@0.0.0.0:7052 --enrollment.profile tls --
csr.hosts peer1-org1
```

转到路径 `/tmp/hyperledger/org1/peer1/tls-msp/keystore` 并将密钥名称更改为 `key.pem` 。这将使以后的步骤中易于引用。
此时，您将拥有两个 MSP 目录。一个 MSP 包含对等方的注册证书，另一个 MSP 包含对等方的 TLS 证书。但是，在注册 MSP 目录中需要添加一个额外的文件夹，这就是该 `admincerts` 文件夹。该文件夹将包含 Org1 管理员的证书。当我们更进一步地注册 Org1 的管理员时，我们将详细讨论这一点。

参加 Peer2

您将对 Peer2 执行类似的命令。在下面的命令中，我们将假定 Org1 的受信任根证书已复制到 `/tmp/hyperledger/org1/peer2/assets/ca/org1-ca-cert.pem` Peer2 的主机上。

```
export FABRIC_CA_CLIENT_HOME=/tmp/hyperledger/org1/peer2
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org1/peer2/assets/ca/org1-ca-cert.pem
fabric-ca-client enroll -d -u https://peer2-org1:peer2PW@0.0.0.0:7054
```

下一步是获取对等方的 TLS 加密材料。这需要再次注册，但是这次您将针对 `tls` TLS CA 上的配置文件进行注册。您还需要在注册请求中提供 Peer2 主机的地址，作为该 `csr.hosts` 标志的输入。在下面的命令中，我们将假定 TLS CA

的证书已复制到 `/tmp/hyperledger/org1/peer2/assets/tls-ca/tls-ca-cert.pem` Peer2 的主机上。

```
export FABRIC_CA_CLIENT_MSPDIR=tls-msp
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org1/peer2/assets/tls-ca/tls-ca-cert.pem
fabric-ca-client enroll -d -u https://peer2-org1:peer2PW@0.0.0.0:7052 --enrollment.profile tls --
csr.hosts peer2-org1
```

转到路径 `/tmp/hyperledger/org1/peer2/tls-msp/keystore` 并将密钥名称更改为 `key.pem` 。这将使以后的步骤中易于引用。

此时，您将拥有两个 MSP 目录。一个 MSP 包含对等方的注册证书，另一个 MSP 包含对等方的 TLS 证书。`admincerts` 一旦管理员被注册，您便会将文件夹添加 到注册 MSP。

注册 Org1 的管理员

至此，两个同伴都已注册。现在，您将注册 Org1 的管理员身份。管理员身份负责诸如安装和实例化 chaincode 的活动。以下步骤将注册管理员。以下命令假定此操作正在 Peer1 的主机上执行。

```
export FABRIC_CA_CLIENT_HOME=/tmp/hyperledger/org1/admin
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org1/peer1/assets/ca/org1-ca-cert.pem
export FABRIC_CA_CLIENT_MSPDIR=msp
fabric-ca-client enroll -d -u https://admin-org1:org1AdminPW@0.0.0.0:7054
```

注册后，您应该具有管理员 MSP。您将从该 MSP 复制证书，并将其移动到 `admincerts` 文件夹中 Peer1 的 MSP 中。您将需要将此管理证书分发给组织中的其他对等方，并且需要进入 `admincerts` 每个对等方 MSP 的文件夹中。下面的命令仅用于 Peer1，将管理证书交换到 Peer2 将在带外进行。

```
mkdir /tmp/hyperledger/org1/peer1/msp/admincerts
cp /tmp/hyperledger/org1/admin/msp/signcerts/cert.pem
/tmp/hyperledger/org1/peer1/msp/admincerts/org1-admin-cert.pem
```

如果 `admincerts` 对等方的本地 MSP 中缺少该文件夹，则对等方将无法启动。

开始 Org1 的对等节点

一旦我们注册了所有对等方和组织管理员，便拥有了启动对等方所需的 MSP。

可以使用以下服务之类的 Docker 服务来启动 Peer1 的容器。

```
peer1-org1:
  container_name: peer1-org1
  image: hyperledger/fabric-peer
  environment:
    - CORE_PEER_ID=peer1-org1
    - CORE_PEER_ADDRESS=peer1-org1:7051
    - CORE_PEER_LOCALMSPID=org1MSP
    - CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org1/peer1/msp
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=guide_fabric-ca
    - FABRIC_LOGGING_SPEC=debug
    - CORE_PEER_TLS_ENABLED=true
    - CORE_PEER_TLS_CERT_FILE=/tmp/hyperledger/org1/peer1/tls-msp/signcerts/cert.pem
    - CORE_PEER_TLS_KEY_FILE=/tmp/hyperledger/org1/peer1/tls-msp/keystore/key.pem
    - CORE_PEER_TLS_ROOTCERT_FILE=/tmp/hyperledger/org1/peer1/tls-msp/tlscacerts/tls-0-0-0-0-
7052.pem
    - CORE_PEER_GOSSIP_USELEADERELECTION=true
    - CORE_PEER_GOSSIP_ORGLEADER=false
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1-org1:7051
    - CORE_PEER_GOSSIP_SKIPHANDSHAKE=true
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric/org1/peer1
```

```
  volumes:
    - /var/run:/host/var/run
    - /tmp/hyperledger/org1/peer1:/tmp/hyperledger/org1/peer1
  networks:
    - fabric-ca
```

启动对等服务将启动一个对等容器，并且在日志中，您将看到以下行：

```
serve -> INFO 020 Started peer with ID=[name:"peer1-org1" ], network ID=[dev], address=[peer1-org1:7051]
```

可以使用以下服务之类的 Docker 服务来启动 Peer2 的容器。

```
peer2-org1:
  container_name: peer2-org1
  image: hyperledger/fabric-peer
  environment:
    - CORE_PEER_ID=peer2-org1
    - CORE_PEER_ADDRESS=peer2-org1:7051
    - CORE_PEER_LOCALMSPID=org1MSP
    - CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org1/peer2/msp
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=guide_fabric-ca
    - FABRIC_LOGGING_SPEC=grpc=debug:info
    - CORE_PEER_TLS_ENABLED=true
    - CORE_PEER_TLS_CERT_FILE=/tmp/hyperledger/org1/peer2/tls-msp/signcerts/cert.pem
    - CORE_PEER_TLS_KEY_FILE=/tmp/hyperledger/org1/peer2/tls-msp/keystore/key.pem
    - CORE_PEER_TLS_ROOTCERT_FILE=/tmp/hyperledger/org1/peer2/tls-msp/tlscacerts/tls-0-0-0-0-7052.pem
    - CORE_PEER_GOSSIP_USELEADERELECTION=true
    - CORE_PEER_GOSSIP_ORGLEADER=false
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer2-org1:7051
    - CORE_PEER_GOSSIP_SKIPHANDSHAKE=true
    - CORE_PEER_GOSSIP_BOOTSTRAP=peer1-org1:7051
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric/org1/peer2
  volumes:
    - /var/run:/host/var/run
    - /tmp/hyperledger/org1/peer2:/tmp/hyperledger/org1/peer2
  networks:
    - fabric-ca
```

启动对等服务将启动一个对等容器，并且在日志中，您将看到以下行：

```
serve -> INFO 020 Started peer with ID=[name:"peer2-org1" ], network ID=[dev], address=[peer2-org1:7051]
```

2. 设置 Org2 的对等

Org2 的管理员将使用 CA 的引导程序身份向 CA 注册对等方，然后启动对等 Docker 容器。

参加 Peer1

您将发出以下命令来注册 Peer1。在下面的命令中，我们假定 Org2 的受信任根证书 `/tmp/hyperledger/org2/peer1/assets/ca/org2-ca-cert.pem` 在 Peer1 的主机上可用。

```
export FABRIC_CA_CLIENT_HOME=/tmp/hyperledger/org2/peer1
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org2/peer1/assets/ca/org2-ca-cert.pem
fabric-ca-client enroll -d -u https://peer1-org2:peer1PW@0.0.0.0:7055
```

接下来，您将获得 TLS 证书。在下面的命令中，我们将假定 TLS CA 的证书已被复制到 `/tmp/hyperledger/org2/peer1/assets/tls-ca/tls-ca-cert.pem` Peer1 的主机上。

```
export FABRIC_CA_CLIENT_MSPDIR=tls-msp
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org2/peer1/assets/tls-ca/tls-ca-cert.pem
```

```
fabric-ca-client enroll -d -u https://peer1-org2:peer1PW@0.0.0.0:7052 --enrollment.profile tls --
csr.hosts peer1-org2
```

转到路径`/tmp/hyperledger/org2/peer1/tls-msp/keystore`并将密钥名称更改为`key.pem`。

参加 Peer2

您将发出以下命令来注册 Peer2。在下面的命令中，我们假定 Org2 的受信任根证书`/tmp/hyperledger/org2/peer2/tls/org2-ca-cert.pem`在 Peer2 的主机上可用。

```
export FABRIC_CA_CLIENT_HOME=/tmp/hyperledger/org2/peer2
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org2/peer2/assets/ca/org2-ca-cert.pem
fabric-ca-client enroll -d -u https://peer2-org2:peer2PW@0.0.0.0:7055
```

接下来，您将获得 TLS 证书。在下面的命令中，我们将假定 TLS CA 的证书已复制到`/tmp/hyperledger/org2/peer2/assets/tls-ca/tls-ca-cert.pem` Peer2 的主机上。

```
export FABRIC_CA_CLIENT_MSPDIR=tls-msp
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org2/peer2/assets/tls-ca/tls-ca-cert.pem
fabric-ca-client enroll -d -u https://peer2-org2:peer2PW@0.0.0.0:7052 --enrollment.profile tls --
csr.hosts peer2-org2
```

转到路径`/tmp/hyperledger/org2/peer2/tls-msp/keystore`并将密钥名称更改为`key.pem`。

注册 Org2 的管理员

此时，您将拥有两个 MSP 目录。一个 MSP 包含您的注册证书，另一个 MSP 包含您的 TLS 证书。但是，在注册 MSP 目录中需要再添加一个文件夹，这就是该`admincerts`文件夹。该文件夹将包含 Org2 管理员的证书。以下步骤将注册管理员。以下命令假定此操作正在 Peer1 的主机上执行。

```
export FABRIC_CA_CLIENT_HOME=/tmp/hyperledger/org2/admin
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org2/peer1/assets/ca/org2-ca-cert.pem
export FABRIC_CA_CLIENT_MSPDIR=msp
fabric-ca-client enroll -d -u https://admin-org2:org2AdminPW@0.0.0.0:7055
```

注册后，您应该具有管理员 MSP。您将从该 MSP 复制证书并将其移动到该`admincerts`文件夹下的对等 MSP。下面的命令仅用于 Peer1，将 admin cert 交换到 peer2 将在带外进行。

```
mkdir /tmp/hyperledger/org2/peer1/msp/admincerts
cp /tmp/hyperledger/org2/admin/msp/signcerts/cert.pem
/tmp/hyperledger/org2/peer1/msp/admincerts/org2-admin-cert.pem
```

如果`admincerts`对等方的本地 MSP 中缺少该文件夹，则对等方将无法启动。

开始 Org2 的 Peer

一旦我们注册了所有对等方和 admin，我们就有了必要的 MSP 来启动对等方。
Docker 容器服务（例如以下服务）可用于为 peer1 启动容器。

```
peer1-org2:
  container_name: peer1-org2
  image: hyperledger/fabric-peer
  environment:
```

```
    - CORE_PEER_ID=peer1-org2
    - CORE_PEER_ADDRESS=peer1-org2:7051
    - CORE_PEER_LOCALMSPID=org2MSP
    - CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org2/peer1/msp
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=guide_fabric-ca
    - FABRIC_LOGGING_SPEC=debug
    - CORE_PEER_TLS_ENABLED=true
    - CORE_PEER_TLS_CERT_FILE=/tmp/hyperledger/org2/peer1/tls-msp/signcerts/cert.pem
    - CORE_PEER_TLS_KEY_FILE=/tmp/hyperledger/org2/peer1/tls-msp/keystore/key.pem
    - CORE_PEER_TLS_ROOTCERT_FILE=/tmp/hyperledger/org2/peer1/tls-msp/tlscacerts/tls-0-0-0-0-
7052.pem
    - CORE_PEER_GOSSIP_USELEADERELECTION=true
    - CORE_PEER_GOSSIP_ORGLEADER=false
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1-org2:7051
    - CORE_PEER_GOSSIP_SKIPHANDSHAKE=true
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric/org2/peer1
  volumes:
    - /var/run:/host/var/run
    - /tmp/hyperledger/org2/peer1:/tmp/hyperledger/org2/peer1
  networks:
    - fabric-ca
```

启动对等服务将启动一个对等容器，并且在日志中，您将看到以下行：

```
serve -> INFO 020 Started peer with ID=[name:"peer1-org2" ], network ID=[dev], address=[peer1-
org2:7051]
```

Docker 容器服务（例如以下服务）可用于为 peer1 启动容器。

```
peer2-org2:
  container_name: peer2-org2
  image: hyperledger/fabric-peer
  environment:
    - CORE_PEER_ID=peer2-org2
    - CORE_PEER_ADDRESS=peer2-org2:7051
    - CORE_PEER_LOCALMSPID=org2MSP
    - CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org2/peer2/msp
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=guide_fabric-ca
    - FABRIC_LOGGING_SPEC=debug
    - CORE_PEER_TLS_ENABLED=true
    - CORE_PEER_TLS_CERT_FILE=/tmp/hyperledger/org2/peer2/tls-msp/signcerts/cert.pem
    - CORE_PEER_TLS_KEY_FILE=/tmp/hyperledger/org2/peer2/tls-msp/keystore/key.pem
    - CORE_PEER_TLS_ROOTCERT_FILE=/tmp/hyperledger/org2/peer2/tls-msp/tlscacerts/tls-0-0-0-0-
7052.pem
    - CORE_PEER_GOSSIP_USELEADERELECTION=true
    - CORE_PEER_GOSSIP_ORGLEADER=false
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer2-org2:7051
    - CORE_PEER_GOSSIP_SKIPHANDSHAKE=true
    - CORE_PEER_GOSSIP_BOOTSTRAP=peer1-org2:7051
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric/org2/peer2
  volumes:
    - /var/run:/host/var/run
    - /tmp/hyperledger/org2/peer2:/tmp/hyperledger/org2/peer2
  networks:
    - fabric-ca
```

启动对等服务将启动一个对等容器，并且在日志中，您将看到以下行：

```
serve -> INFO 020 Started peer with ID=[name:"peer2-org2" ], network ID=[dev], address=[peer2-
org2:7052]
```

4. 设置排序者

我们需要设置的最后一件事是排序者。在启动排序器之前，我们需要采取一些措施。

3. 注册排序者

在启动排序者之前，您将需要使用 CA 注册排序者的身份以获取排序者将使用的 MSP。这称为本地排序者 MSP。
如果主机没有 Fabric-ca-client 二进制文件，请参考上面的说明以下载二进制文件。
您将发出以下命令来注册排序者。在下面的命令中，我们假定 Org0 的受信任根证书在排序者的主机上可用 `/tmp/hyperledger/org0/orderer/assets/ca/org0-ca-cert.pem` 。

```
export FABRIC_CA_CLIENT_HOME=/tmp/hyperledger/org0/orderer
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org0/orderer/assets/ca/org0-ca-cert.pem
fabric-ca-client enroll -d -u https://orderer-org0:ordererPW@0.0.0.0:7053
```

接下来，您将获得 TLS 证书。在下面的命令中，我们将假定 TLS CA 的证书已复制到 `/tmp/hyperledger/org0/orderer/assets/tls-ca/tls-ca-cert.pem` 排序者的主机上。

```
export FABRIC_CA_CLIENT_MSPDIR=tls-msp
export FABRIC_CA_CLIENT_TLS_CERTFILES=/tmp/hyperledger/org0/orderer/assets/tls-ca/tls-ca-cert.pem
fabric-ca-client enroll -d -u https://orderer-org0:ordererPW@0.0.0.0:7052 --enrollment.profile tls --csr.hosts orderer1-org0
```

转到路径 `/tmp/hyperledger/org0/orderer/tls-msp/keystore` 并将密钥名称更改为 `key.pem` 。
此时，您将拥有两个 MSP 目录。一个 MSP 包含您的注册证书，另一个 MSP 包含您的 TLS 证书。但是，在注册 MSP 目录中需要添加一个额外的文件夹，该 `admincerts` 文件夹。该文件夹将包含对等 1 管理员的证书。现在，您将通过发出以下命令来注册 Org0 的管理员身份。

4. 注册 Org0 的管理员

下面的命令假定此命令正在排序者的主机上执行。

```
export FABRIC_CA_CLIENT_HOME=/tmp/hyperledger/org0/admin
export FABRIC_CA_CLIENT_MSPDIR=msp
fabric-ca-client enroll -d -u https://orderer-org0-admin:ordererAdminPW@0.0.0.0:7053
```

注册后，您应该在拥有一个 msp 文件夹 `/tmp/hyperledger/org0/admin` 。您将从该 MSP 复制证书，并将其移动到 `admincerts` 文件夹下的排序者的 MSP 。

```
mkdir /tmp/hyperledger/org0/orderer/msp/admincerts
cp /tmp/hyperledger/org0/admin/msp/signcerts/cert.pem
/tmp/hyperledger/org0/orderer/msp/admincerts/orderer-admin-cert.pem
```

5. 创建创世块和渠道交易

定购者需要一个用于自举的创世块。您可以在 Hyperledger Fabric 文档中找到更多信息。
在下面的文档中，您将找到 `configtx.yaml` 为该特定部署编写的摘要。有关完整信息 `configtx.yaml` ，请单击此处。
在排序者的主机上，我们需要收集所有组织的 MSP。该 `organization` 部分的 `configtx.yaml` 外观如下：

```
Organizations:

- &org0
```

```
    Name: org0

    ID: org0MSP

    MSPDir: /tmp/hyperledger/org0/msp

- &org1

    Name: org1

    ID: org1MSP

    MSPDir: /tmp/hyperledger/org1/msp

    AnchorPeers:
        - Host: peer1-org1
          Port: 7051

- &org2

    Name: org2

    ID: org2MSP

    MSPDir: /tmp/hyperledger/org2/msp

    AnchorPeers:
        - Host: peer1-org2
          Port: 7051
```

用于 Org0 的 MSP 将包含 Org0 的受信任的根证书，Org0 的管理员身份的证书以及 TLS CA 的受信任的根证书。MSP 文件夹结构如下所示。

```
/tmp/hyperledger/org0/msp
├── admincerts
│   └── admin-org0-cert.pem
├── cacerts
│   └── org0-ca-cert.pem
├── tlscacerts
│   └── tls-ca-cert.pem
└── users
```

所有组织的模式都是相同的。Org1 的 MSP 文件夹结构如下：

```
/tmp/hyperledger/org1/msp
├── admincerts
│   └── admin-org1-cert.pem
├── cacerts
│   └── org1-ca-cert.pem
├── tlscacerts
│   └── tls-ca-cert.pem
└── users
```

Org2 的 MSP 文件夹结构如下：

```
/tmp/hyperledger/org2/msp
├── admincerts
│   └── admin-org2-cert.pem
├── cacerts
│   └── org2-ca-cert.pem
├── tlscacerts
│   └── tls-ca-cert.pem
└── users
```

一旦所有这些 MSP 都存在于排序者的主机上，您将从 `configtx.yaml` 存在的目录中执行以下命令：

```
configtxgen -profile OrgsOrdererGenesis -outputBlock /tmp/hyperledger/org0/orderer/genesis.block
configtxgen -profile OrgsChannel -outputCreateChannelTx /tmp/hyperledger/org0/orderer/channel.tx -
channelID mychannel
```

这将生成两个工件，`genesis.block` 和 `channel.tx`，将在以后的步骤中使用。

收集证书的命令

Fabric CA 客户端有几个命令，这些命令可用于获取排序者生成和对等 MSP 设置的证书。
第一个命令是 *fabric-ca-client certificate* 命令。此命令可用于获取 admincers 文件夹的证书。有关如何使用此命令的更多信息，请参考：列出证书信息
第二个命令是 *fabric-ca-client getcainfo* 命令。此命令可用于收集 *cacerts* 和 *tlscacerts* 文件夹的证书。该 *getcainfo* 命令返回该 CA 证书

6. 相互 TLS

也可以使用 Mutual TLS 保护端点。如果 CA，Peer 或 Orderer 使用双向 TLS，则客户端还必须提供将由服务器验证的 TLS 证书。
相互 TLS 要求客户端获取它将提供给服务器的 TLS 证书。可以通过启用了双向 TLS 的 TLS 证书颁发机构来完成 TLS 证书的获取。客户端获取 TLS 证书后，只要服务器上受信任的 TLS 权限与客户端 TLS 证书的颁发权限相同，它就可以开始与启用了 TLS 的相互通信。

7. 开始排序节点

一旦创建了创世块和渠道交易，就可以定义一个指向上面创建的 genesis.block 的排序服务。

```
orderer1-org0:
  container_name: orderer1-org0
  image: hyperledger/fabric-orderer
  environment:
    - ORDERER_HOME=/tmp/hyperledger/orderer
    - ORDERER_HOST=orderer1-org0
    - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
    - ORDERER_GENERAL_GENESISMETHOD=file
    - ORDERER_GENERAL_GENESISFILE=/tmp/hyperledger/org0/orderer/genesis.block
    - ORDERER_GENERAL_LOCALMSPID=org0MSP
    - ORDERER_GENERAL_LOCALMSPDIR=/tmp/hyperledger/org0/orderer/msp
    - ORDERER_GENERAL_TLS_ENABLED=true
    - ORDERER_GENERAL_TLS_CERTIFICATE=/tmp/hyperledger/org0/orderer/tls-msp/signcerts/cert.pem
    - ORDERER_GENERAL_TLS_PRIVATEKEY=/tmp/hyperledger/org0/orderer/tls-msp/keystore/key.pem
    - ORDERER_GENERAL_TLS_ROOTCAS=[/tmp/hyperledger/org0/orderer/tls-msp/tlscacerts/tls-0-0-0-0-
7052.pem]
    - ORDERER_GENERAL_LOGLEVEL=debug
    - ORDERER_DEBUG_BROADCASTTRACEDIR=data/logs
  volumes:
    - /tmp/hyperledger/org0/orderer:/tmp/hyperledger/org0/orderer/
  networks:
    - fabric-ca
```

启动排序者服务将调出排序者容器，并且在日志中，您将看到以下行：

```
UTC [orderer/common/server] Start -> INFO 0b8 Beginning to serve requests
```

5. 创建 CLI 容器

与对等方通信需要一个 CLI 容器，该容器包含适当的二进制文件，使您可以发出与对等方相关的命令。您将为每个组织创建一个 CLI 容器。在此示例中，我们为每个组织在与 Peer1 相同的主机中启动 CLI 容器。

1. 启动 Org1 的 CLI

```
cli-org1:
   container_name: cli-org1
   image: hyperledger/fabric-tools
   tty: true
   stdin_open: true
   environment:
     - GOPATH=/opt/gopath
     - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
     - FABRIC_LOGGING_SPEC=DEBUG
     - CORE_PEER_ID=cli-org1
     - CORE_PEER_ADDRESS=peer1-org1:7051
     - CORE_PEER_LOCALMSPID=org1MSP
     - CORE_PEER_TLS_ENABLED=true
     - CORE_PEER_TLS_ROOTCERT_FILE=/tmp/hyperledger/org1/peer1/tls-msp/tlscacerts/tls-0-0-0-0-
7052.pem
     - CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org1/peer1/msp
   working_dir: /opt/gopath/src/github.com/hyperledger/fabric/org1
   command: sh
   volumes:
     - /tmp/hyperledger/org1/peer1:/tmp/hyperledger/org1/peer1
     - /tmp/hyperledger/org1/peer1/assets/chaincode:/opt/gopath/src/github.com/hyperledger/fabric-
samples/chaincode
     - /tmp/hyperledger/org1/admin:/tmp/hyperledger/org1/admin
   networks:
     - fabric-ca
```

2. 启动 Org2 的 CLI

```
cli-org2:
   container_name: cli-org2
   image: hyperledger/fabric-tools
   tty: true
   stdin_open: true
   environment:
     - GOPATH=/opt/gopath
     - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
     - FABRIC_LOGGING_SPEC=DEBUG
     - CORE_PEER_ID=cli-org2
     - CORE_PEER_ADDRESS=peer1-org2:7051
     - CORE_PEER_LOCALMSPID=org2MSP
     - CORE_PEER_TLS_ENABLED=true
     - CORE_PEER_TLS_ROOTCERT_FILE=/tmp/hyperledger/org2/peer1/tls-msp/tlscacerts/tls-0-0-0-0-
7052.pem
     - CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org2/peer1/msp
   working_dir: /opt/gopath/src/github.com/hyperledger/fabric/org2
   command: sh
   volumes:
     - /tmp/hyperledger/org2/peer1:/tmp/hyperledger/org2/peer1
     - /tmp/hyperledger/org1/peer1/assets/chaincode:/opt/gopath/src/github.com/hyperledger/fabric-
samples/chaincode
     - /tmp/hyperledger/org2/admin:/tmp/hyperledger/org2/admin
   networks:
     - fabric-ca
```

6. 建立并加入频道

1. 组织 1

随着 CLI 容器的启动和运行，您现在可以发出命令来创建和加入通道。我们将使用 Peer1 创建通道。在 Peer1 的主机中，您将执行：

```
docker exec -it cli-org1 sh
```

此命令将带您进入 CLI 容器并打开终端。从这里，您将使用 admin MSP 执行以下命令：

```
export CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org1/admin/msp
peer channel create -c mychannel -f /tmp/hyperledger/org1/peer1/assets/channel.tx -o orderer1-
org0:7050 --outputBlock /tmp/hyperledger/org1/peer1/assets/mychannel.block --tls --cafile
/tmp/hyperledger/org1/peer1/tls-msp/tlscacerts/tls-0-0-0-0-7052.pem
```

其中 `channel.tx` 是通过运行所产生的伪像 `configtxgen` 的排序者命令。该工件需要从排序者带外传输到 Peer1 的主机。上面的命令将 `mychannel.block` 在 Peer1 的指定输出路径上生成，`/tmp/hyperledger/org1/peer1/assets/mychannel.block` 网络中所有希望加入该通道的对等点都将使用该命令。这 `mychannel.block` 将需要转移到 Org1 和 Org2 带外的所有对等方。

接下来要运行的命令是将 Peer1 和 Peer2 加入通道。

```
export CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org1/admin/msp
export CORE_PEER_ADDRESS=peer1-org1:7051
peer channel join -b /tmp/hyperledger/org1/peer1/assets/mychannel.block

export CORE_PEER_ADDRESS=peer2-org1:7051
peer channel join -b /tmp/hyperledger/org1/peer1/assets/mychannel.block
```

2. 组织 2

运行以下命令以输入 CLI docker 容器。

```
docker exec -it cli-org2 sh
```

在 Org2 中，您只需要让对等方加入频道即可。Org2 中的对等方不需要创建通道，这已由 Org1 完成。在 Org2 CLI 容器内部，您将使用 admin MSP 执行以下命令：

```
export CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org2/admin/msp
export CORE_PEER_ADDRESS=peer1-org2:7051
peer channel join -b /tmp/hyperledger/org2/peer1/assets/mychannel.block

export CORE_PEER_ADDRESS=peer2-org2:7051
peer channel join -b /tmp/hyperledger/org2/peer1/assets/mychannel.block
```

7. 安装并实例化 Chaincode

将这个链码 从 Github 下载到两个组织中 Peer1 上的本地文件系统。

1. 组织 1

在 Peer1 上，您将安装 chaincode。该命令假定 GOPATH 内有需要安装的链码。在这个例子中，我们将假定 chaincode 位于 `/opt/gopath/src/github.com/hyperledger/fabric-samples/chaincode/abac/go` 与 GOPATH 之中 `/opt/gopath`。在 Org1 的 CLI 容器中，您将执行以下命令：

```
export CORE_PEER_ADDRESS=peer1-org1:7051
export CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org1/admin/msp
peer chaincode install -n mycc -v 1.0 -p github.com/hyperledger/fabric-samples/chaincode/abac/go
```

对等方 2 将遵循相同的步骤。

```
export CORE_PEER_ADDRESS=peer2-org1:7051
export CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org1/admin/msp
peer chaincode install -n mycc -v 1.0 -p github.com/hyperledger/fabric-samples/chaincode/abac/go
```

2. 组织 2

在 Peer1 上，您将执行与 Org1 相同的步骤。该命令假定需要安装的链码位于 `/opt/gopath/src/github.com/hyperledger/org2/peer1/assets/chaincode/abac/go`。在 Org2 的 CLI 容器中，您将执行以下命令：

```
export CORE_PEER_ADDRESS=peer1-org2:7051
export CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org2/admin/msp
peer chaincode install -n mycc -v 1.0 -p github.com/hyperledger/fabric-samples/chaincode/abac/go
```

对等方 2 将遵循相同的步骤。

```
export CORE_PEER_ADDRESS=peer2-org2:7051
export CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org2/admin/msp
peer chaincode install -n mycc -v 1.0 -p github.com/hyperledger/fabric-samples/chaincode/abac/go
```

下一步将是实例化链码。通过执行以下操作来完成：

```
peer chaincode instantiate -C mychannel -n mycc -v 1.0 -c '{"Args":["init","a","100","b","200"]}' -o
orderer1-org0:7050 --tls --cafile /tmp/hyperledger/org2/peer1/tls-msp/tlscacerts/tls-0-0-0-0-
7052.pem
```

8. 调用和查询链码

在 Org1 的 CLI 容器中，执行：

```
export CORE_PEER_ADDRESS=peer1-org1:7051
export CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org1/admin/msp
peer chaincode query -C mychannel -n mycc -c '{"Args":["query","a"]}'
```

这应返回的值 `100`。

在 Org2 的 CLI 容器中，执行：

```
export CORE_PEER_ADDRESS=peer1-org2:7051
export CORE_PEER_MSPCONFIGPATH=/tmp/hyperledger/org2/admin/msp
peer chaincode invoke -C mychannel -n mycc -c '{"Args":["invoke","a","b","10"]}' --tls --cafile
/tmp/hyperledger/org2/peer1/tls-msp/tlscacerts/tls-0-0-0-0-7052.pem
```

这将从的值中减去 10 `a` 并将其移至 `b`。现在，如果您通过运行查询：

```
peer chaincode query -C mychannel -n mycc -c '{"Args":["query","a"]}'
```

这应返回的值 `90`。

以上是所有的《 Fabric CA 操作指南》