超级账本 Fabric v1.0 多节点集群的部署

张海宁、陈家豪
VMware 中国研发中心

一、概述

在千呼万唤之后， 犹抱琵琶的超级账本 1.0 GA 版即将揭开面纱， 翘首以待的社区用户将广泛使用这个版本。本文将介绍如何使用 Docker 容器技术来建立起一个多节点 Fabric 集群，并且描述在集群上如何进行基本的操作，如 chaincode 的生命周期维护等。文中采用 Fabric 1.0 beta 的端到端（e2e_cli）示例作为基础来说明原理 。本文提供是为手动配置的方法，今后将介绍利用容器平台（如 K8s 等）自动部署超级账本的方式。

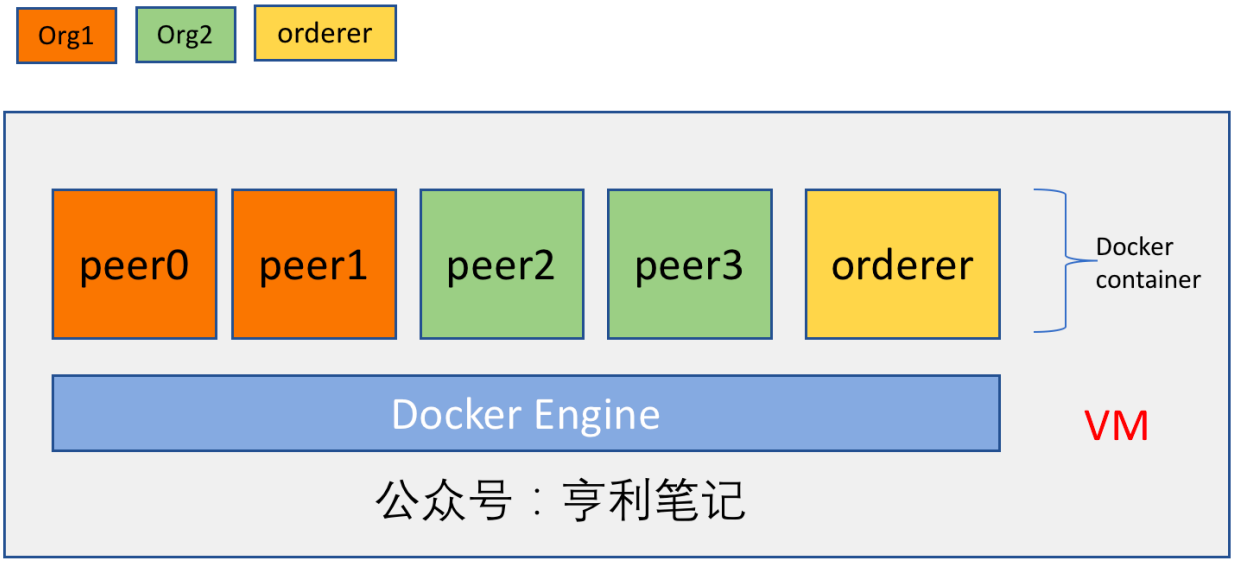


图 1.1 单节点下的 Fabric 网络结构图

Fabric 源码中包含一个简单的 e2e_cli 单机部署示例，方便用户理解、研究和开发应用。如图 1.1 所示， 在单个机器节点上通过 docker-compose 建立了 5 个节点 的 Fabric 网络，每个节点都是由单独的 Docker 容器来模拟。其中 peer0 和 peer1 是同属于 org1 的节点，peer2 和 peer3 是同属于 org2 的节点，它们都加入了相同的 channel 中，并在该 channel 中进行交易，而 orderer 则为该 channel 中的交易提供排序服务。

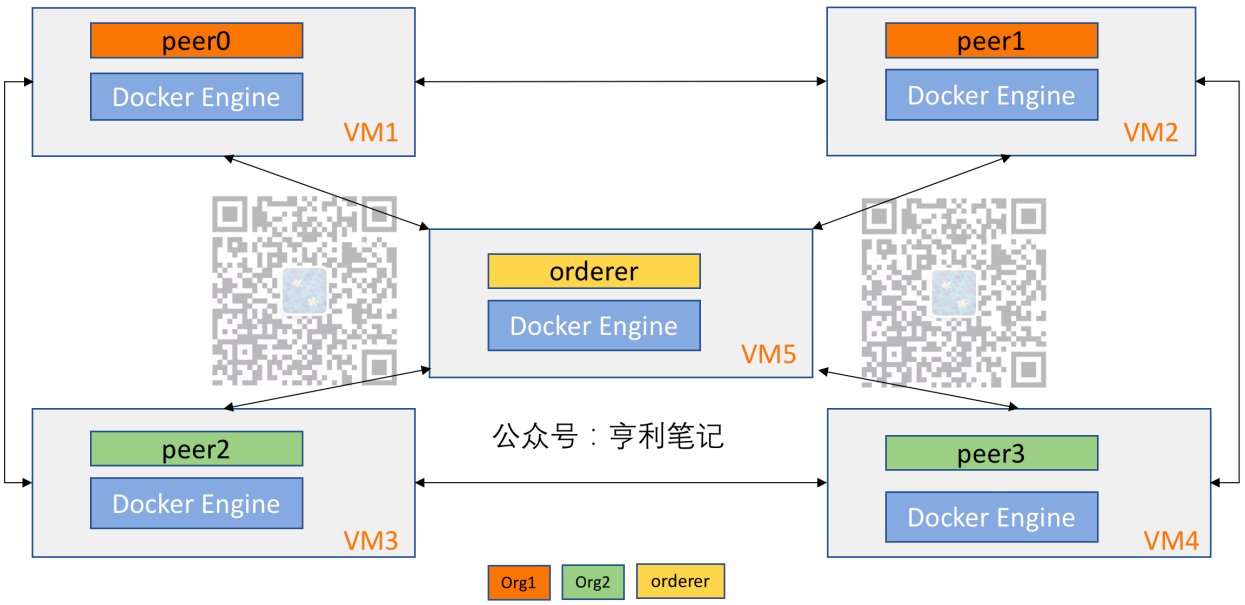


图 1.2 多节点下的 Fabric 网络结构图

e2e_cli 的示例虽然简单，但它把多个节点混合部署在一起，无法区分哪些配置对应哪个节点。另外，在实际场景中，Fabric 节点可能会由不同的组织分别拥有和维护，peers 和 orderer 必然会分布在不同的物理节点上，因此多节点的 Fabric 部署成为需要解决的问题，图 1.2 是多节点 Fabric 集群拓扑图。

下面是把单节点 e2e_cli 示例改为多节点的大致步骤：

1. 准备环境

   运行 Fabric 节点需要依赖以下工具：

   a. Docker ：用于管理 Fabric 镜像以及运行 peer 和 orderer 等组件
   b. Docker-compose： 用于配置 Fabric 容器
   c. Fabric 源码：源码提供了用于生成证书和配置 channel 的工具和测试代码
   d. Go 语言开发环境：源码的工具编译依赖于 Go 语言

2. 配置多节点 Fabric 集群

   在单节点 e2e_cli 示例中，所有节点部署在同一个 docker-compose 的内部网络中，通过容器的 7051 端口进行通信。但是在多节点的情况下，容器之间不能进行直接通讯，因此需要把容器的 7051 端口映射到宿主机上，通过各个宿主机的 7051 端口来实现节点间通信。 我们在每个节点中修改 docker-compose.yaml 中的 service 定义，在不同节点只启动需要的 service。例如，在节点 1 中只启动 peer0 的 service，在节点 5 中仅启动 orderer 等。

3. 启动多节点 Fabric 集群

在各个节点上配置好 Fabric 的启动环境后，需要依次登录到节点上通过 docker-compose up 的方式启动 Fabric 节点。由于启动环境有依赖关系，如 peer1 以 peer0 作为发现节点，因此需要先启动 peer0 再启动 peer1。

4. 配置 channel

在 Fabric 中，channel 代表了一个私有的广播通道，保证了消息的隔离性和私密性，它由 orderer 来管理。channel 中的成员共享该 channel 的账本，并且只有通过验证的用户才能在 channel 中进行交易，与一个 channel 相关的属性记录在该 channel 的初始区块中，可通过 reconfiguration 交易进行更改。 channel 的初始区块由 create channel 交易生成，peer 向 orderer 发送该交易时会带有的 config.tx 文件，该文件定义 channel 的相关属性。

5. 发布 chaincode

chaincode 是开发人员按照特定接口编写的智能合约，通过 SDK 或者 CLI 在 Fabric 的网络上安装并且初始化后，该应用就能访问网络中的共享账本。

chaincode 的生命周期如下:

a. install （安装）

chaincode 要在 Fabric 网络上运行，必须要先安装在网络中的 peer 上，安装同时注明版本号保证应用的版本控制。

b. instantiate （实例化）

在 peer 上安装 chaincode 后，还需要实例化才能真正激活该 chaincode。在实例化的过程中，chaincode 就会被编译并打包成容器镜像，然后启动运行。若 chaincode 在实例化的过程中更新了数据状态，如给某个变量赋予初始值，则该状态变化会被记录在共享账本中。每个应用只能被实例化一次，实例化可在任意一个已安装该 chaincode 的 peer 上进行。

c. invoke 和 query （调用和查询）

chaincode 在实例化后，用户就能与它进行交互，其中 query 查询与应用相关的状态（即只读），而 invoke 则可能会改变其状态。

d. upgrade（升级）

在 chaincode 添加新功能或出现 bug 需要升级时，可以通过 upgrade 交易来实现。这时需要把新的代码通过 install 交易安装到正在运行该 chaincode
的 peer 上，安装时需注明比先前版本更高的版本号，接下来只需要向任意一个安装了新代码的 peer 发送 upgrade 交易就能更新 chaincode，chaincode 在更新前的状态也会得到保留。

二、操作步骤

1、 环境构建与测试

在本文中用到的宿主机环境是 Ubuntu ,版本为 Ubuntu 14.04.5 LTS，通过 Docker 容器来运行 Fabric 的节点, 版本为 v1.0 beta。因此，启动 Fabric 网络中的节点需要先安装 Docker、Docker-compose 和 Go 语言环境，然后在网上拉取相关的 Docker 镜像，再通过配置 compose 文件来启动各个节点。

1.1、Docker 与 Docker-compose 安装

首先切换到 root 用户

```
su root
```

Docker 通过官方提供的脚本可以很方便的安装，先运行以下命令安装 curl：

```
apt-get update && apt-get install curl
```

然后用 wget 来下载脚本并且安装：

```
wget -qO- https://get.docker.com/ | sh
```

当安装完成后，可以通过 docker version 命令来查看 docker 的版本信息：

```
Client:
 Version:      17.05.0-ce
 API version:  1.29
 Go version:   go1.7.5
 Git commit:   89658be
 Built:        Thu May  4 22:10:54 2017
 OS/Arch:      linux/amd64

Server:
 Version:      17.05.0-ce
 API version:  1.29 (minimum version 1.12)
 Go version:   go1.7.5
 Git commit:   89658be
 Built:        Thu May  4 22:10:54 2017
 OS/Arch:      linux/amd64
 Experimental: false
```

接下来就要下载 docker-compose 了，它可以通过 curl 来下载：

```
curl -L https://github.com/docker/compose/releases/download/1.12.0/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose
```

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

执行完毕后可以通过 docker-compose version 来查看 docker-compose 信息：

```
docker-compose version 1.12.0, build b31ff33
docker-py version: 2.2.1
CPython version: 2.7.12
OpenSSL version: OpenSSL 1.0.2g  1 Mar 2016
```

docker 和 docker-compose 安装完毕。

1.2、Go 1.8.3 安装：

1). 通过以下命令下载 go1.8.3：

```
curl -O https://storage.googleapis.com/golang/go1.8.3.linux-amd64.tar.gz
```

2). 解压 go1.8.3.linux-amd64.tar.gz 到/usr/local：

```
tar -C /usr/local -xzf go1.8.3linux-adm64.tar.gz
```

3). 在~/.profile 中添加$GOPATH 环境变量，并把 Go 加到$PATH 环境变量，编辑~/.profile 在最后添加两行：

```
export PATH=$PATH:/usr/local/go/bin
export GOPATH=/opt/gopath
```

4). 在终端运行 source ~/.profile 后用 go version 可查看 Go 语言的版本，返回如下信息说明安装成功：

```
go1.8.3 linux/amd64
```

1.3、下载 Fabric 源码：

下载 Fabric 源码是因为要用到源码中提供的例子和工具，工具的编译需要用到 Go 环境，因此需要把源码目录放到$GOPATH 下。通过上面 Go 的安装配置，$GOPATH 设置为/opt/gopath，用以下命令创建并且进入到 hyperledger 目录中：

mkdir -p /opt/gopath/github.com/hyperledger/ && cd /opt/gopath/github.com/hyperledger

从 github 上下载 Fabric 源码：

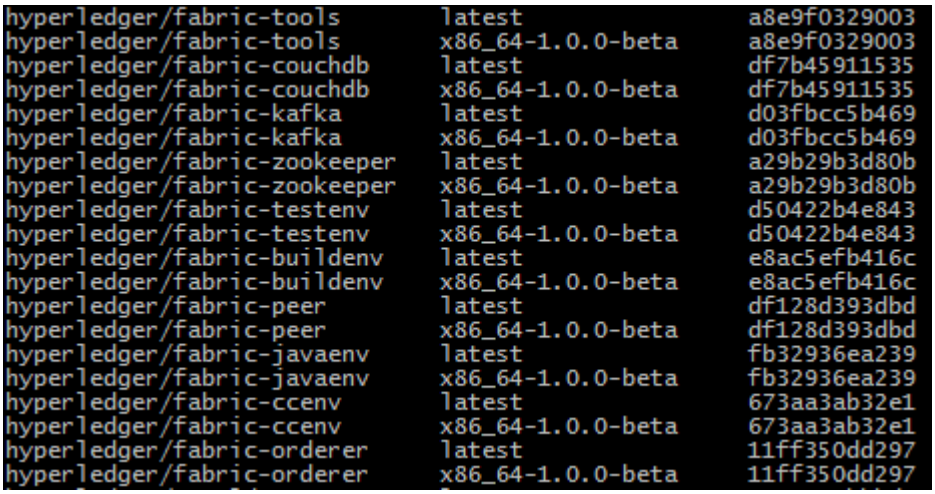
git clone https://github.com/hyperledger/fabric.git

因为镜像使用的是 beta 版本，因此需要把 Fabric 切换到 v1.0.0 beta 源码分支，以兼容 fabric/example/e2e 中的配置:

git checkout v1.0.0-beta

1.4、docker 镜像下载

构建 Fabric 网络所需要用到的 docker 镜像如下：



hyperledger/fabric-tools	latest	a8e9f0329003
hyperledger/fabric-tools	x86_64-1.0.0-beta	a8e9f0329003
hyperledger/fabric-couchdb	latest	df7b45911535
hyperledger/fabric-couchdb	x86_64-1.0.0-beta	df7b45911535
hyperledger/fabric-kafka	latest	d03fbcc5b469
hyperledger/fabric-kafka	x86_64-1.0.0-beta	d03fbcc5b469
hyperledger/fabric-zookeeper	latest	a29b29b3d80b
hyperledger/fabric-zookeeper	x86_64-1.0.0-beta	a29b29b3d80b
hyperledger/fabric-testenv	latest	d50422b4e843
hyperledger/fabric-testenv	x86_64-1.0.0-beta	d50422b4e843
hyperledger/fabric-buildenv	latest	e8ac5efb416c
hyperledger/fabric-buildenv	x86_64-1.0.0-beta	e8ac5efb416c
hyperledger/fabric-peer	latest	df128d393dbd
hyperledger/fabric-peer	x86_64-1.0.0-beta	df128d393dbd
hyperledger/fabric-javaenv	latest	fb32936ea239
hyperledger/fabric-javaenv	x86_64-1.0.0-beta	fb32936ea239
hyperledger/fabric-ccenv	latest	673aa3ab32e1
hyperledger/fabric-ccenv	x86_64-1.0.0-beta	673aa3ab32e1
hyperledger/fabric-orderer	latest	11ff350dd297
hyperledger/fabric-orderer	x86_64-1.0.0-beta	11ff350dd297

进入到 fabric/examples/e2e_cli 目录下，运行./download-dockerimages.sh 来下载必要镜像，镜像下载完成后，就可以通过 docker save 命令把镜像打包成压缩文件，传送到各个 VM。当 VM 接收到压缩文件后，可以通过 docker load 来解压和导入镜像。如果有私有的容器 registry，如 Harbor 等，也可以把镜像推送到私有 registry，再从各个机器中拉取。

通过以下命令来保存所有 tag 含有 beta 标识的镜像到名字为 images 的压缩文件中：

docker save $(docker images | grep beta | awk {‘print $1’}) -o images

生成 images 文件后，就可以通过 scp 把它拷贝到还没有镜像的其他节点中，例如，地址为 10.112.122.6 的节点需要安装以上镜像，可以通过以下命令把 images 远程拷贝到 10.112.122.6 的 home 目录下：

scp images root@10.112.122.6:~

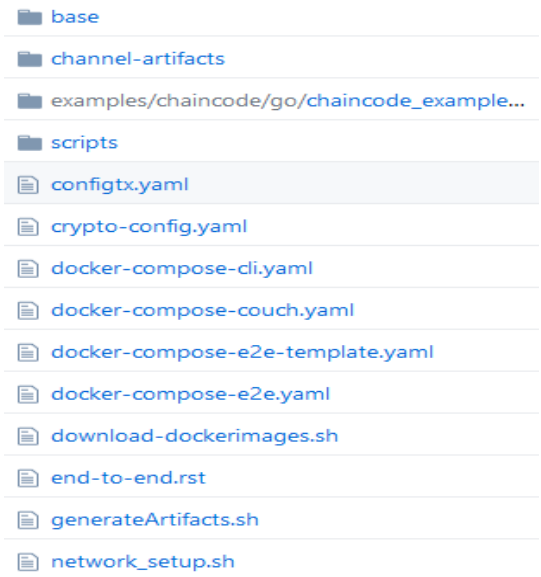
然后 在 10.112.112.6 这台主机的 home 目录上运行：

```
docker load -i images
```

等待一段时间后通过 docker images 命令就能查看到相关镜像的信息。

1.5、运行测试

进入到 fabric/example/e2e_cli 文件夹，文件结构如下：



network_setup.sh 是一键测试脚本，该脚本启动了 6 个 docker 容器，其中有 4 个容器运行 peer 节点和 1 个容器运行 orderer 节点，它们组成一个 Fabric 集群。另外，还有一个 cli 容器用于执行创建 channel、加入 channel、安装和执行 chaincode 等操作。测试用的 chaincode 定义了两个变量，在实例化的时候给这两个变量赋予了初值，通过 invoke 操作可以使两个变量的值发生变化。
通过以下命令执行测试：

```
bash network_setup.sh up
```

接下来会有许多的调试信息，具体可参考 e2e_cli 目录下的 script/script.sh 文件，当终端出现以下信息时说明测试通过，所有部件工作正常：

```
=== All GOOD, End-2-End execution completed ===
```

至此，环境配置工作完毕，通过 docker ps -a 命令可以查看各容器的状态。chaincode 会在独立的容器中运行，因此会出现 3 个以 dev 开头的容器，它们与各自的 peer 对应，记录了 peer 对 chaincode 的操作。

```
CONTAINER ID        IMAGE                                  COMMAND                  CREATED             STATUS
817b897d64ef        dev-peer1.org2.example.com-mycc-1.0    "chaincode -peer.a..."   3 minutes ago       Up 3 minutes
5665f47c9e24        dev-peer0.org1.example.com-mycc-1.0    "chaincode -peer.a..."   4 minutes ago       Up 4 minutes
28ce9a1dba00        dev-peer0.org2.example.com-mycc-1.0    "chaincode -peer.a..."   4 minutes ago       Up 4 minutes
18615e039f45        hyperledger/fabric-tools               "/bin/bash -c './s..."   8 minutes ago       Up 8 minutes
625df2554690        hyperledger/fabric-peer                "peer node start"        8 minutes ago       Up 8 minutes
bcbce1a18d60        hyperledger/fabric-peer                "peer node start"        8 minutes ago       Up 8 minutes
e0b5f88298a4        hyperledger/fabric-peer                "peer node start"        8 minutes ago       Up 8 minutes
505c8cfd03dc        hyperledger/fabric-peer                "peer node start"        8 minutes ago       Up 8 minutes
4c24182b0004        hyperledger/fabric-orderer             "orderer"                8 minutes ago       Up 8 minutes
```

2、 创建 Fabric 多节点集群

2.1 前期准备

我们将重现 Fabric 自带的 e2e_cli 例子中的集群，不同的是把容器分配到不同的虚拟机上，彼此之间通过网络来进行通信，网络构建完成后则进行相关的 channel 和 chaincode 操作。

先准备 5 台虚拟机（VM），所有虚拟机均按照上述环境构建与测试步骤配置，当然也可安装一个虚拟机模板，然后克隆出其他虚拟机。其中 4 台虚拟机运行 peer 节点，另外一台运行 orderer 节点，为其他的四个节点提供 order 服务。

在本文中，虚拟机的参数如下：

Name	IP	节点标识	节点 Hostname	Organization
VM1	10.112.122.144	peer0	peer0.org1.example.com	Org1
VM2	10.112.122.46	peer1	peer1.org1.example.com	Org1
VM3	10.112.122.12	peer2	peer0.org2.example.com	Org2
VM4	10.112.122.45	peer3	peer1.org2.example.com	Org2
VM5	10.112.122.69	orderer	orderer.example.com	Orderer

2.2 使用 generateArtifacts.sh 生成证书和 config.tx

在任意 VM 上运行 fabric/examples/e2e_cli 目录下的 generateArtifacts.sh 脚本，可生成两个目录，它们分别为 channel-artifacts 和 crypto-config，两个目录的结构分别如下：

```
-channel-artifacts
   -channel.tx
    -genesis.block
   -Org1MSPanchors.tx
   -Org2MSPanchors.tx
```

上述目录里的文件用于 orderer 创建 channel，它们根据 configtx.yaml 的配置生成。

```
-crypto-config
  -ordererOrganizations
  -peerOrganizations
```

上述目录里面有 orderer 和 peer 的证书、私钥和以及用于通信加密的 tls 证书等文件，它通过 configtx.yaml 配置文件生成。

（未完待续）

关注公众号：**亨利笔记**，获取更多区块链和云计算方面科技文章。https://github.com/hainingzhang/articles

扫码关注公众号：**亨利笔记**，获取更多区块链和云计算等方面科技文章。

https://github.com/hainingzhang/articles

VMware 公司招聘区块链实习生和外包开发工程师

VMware 公司为超级账本 Hyperledger 项目创始成员，中国研发中心现在招募区块链方向实习生和外包开发工程师，地点：北京知春里。

外包软件开发工程师：3 年以上软件开发经验，熟悉 Java 或 Go 开发语言，熟悉分布式系统、Docker，了解区块链技术优先。

实习生：要求在读研究生，计算机相关专业，懂 Java 或 Go 开发语言，能够实习 3 个月以上，熟悉区块链技术优先。欢迎自荐或推荐。

有兴趣者发简历到：harbor@vmware.com