目录

一.Fabric 简介

在介绍我们的主角Fabric之前，我们先来介绍一下它的父项目:Hyperledger-超级账本

Hyperledger 项目是首个面向企业的开放区块链技术的重要探索。在 Linux 基金会的支持下，吸引了包括 IBM、Intel、摩根等在内的众多科技和金融巨头的参与。

1.发展历史

区块链已经成为当下最受人关注的开源技术，有人说它将颠覆金融行业的未来。然而对很多人来说，区块链技术难以理解和实现，而且缺乏统一的规范。

2015 年 12 月，开源世界的旗舰——Linux 基金 会牵头，联合 30 家初始企业成员（包括 IBM、Accenture、Intel、J.P.Morgan、R3、DAH、DTCC、FUJITSU、HITACHI、SWIFT、Cisco 等），共同宣告 了 Hyperledger 项目的成立。该项目试图打造一个透明、公开、去中心化的分布式账本项目，作为区块链技术的开源规范和标准，让更多的应用能更容易的建立在区块链技术之上。项目官方信息网站在 hyperledger.org,

目前已经有超过 120 家全球知名企业和机构（大部分均为各自行业的领导者）宣布加入 Hyperledger 项目，其中包括 30 家来自中国本土的企业，包括艾亿新融旗下的艾亿数融科技公司（2016.05.19）、Onchain（2016.06.22）、比邻共赢（Belink）信息技术有限公司（2016.06.22）、BitSE（2016.06.22）、布比（2016.07.27）、三一重工（2016.08.30）、万达金融（2016.09.08）、华为（2016.10.24）等。 如果说以比特币为代表的货币区块链技术为 1.0，以以太坊为代表的合同区块链技术为 2.0，那么实现了完备的权限控制和安全保障的 Hyperledger 项目毫无疑问代表着 3.0 时代的到来。

IBM 贡献了数万行已有的 Open Blockchain 代码，Digital Asset 则贡献了企业和开发者相关资源，R3 贡献了新的金融交易架构，Intel 也刚贡献了跟分布式账本相关的代码。

Hyperledger 社区由技术委员会（Technical Steering Committee，TSC）指导，首任主席由来自 IBM 开源技术部 CTO 的 Chris Ferris 担任，管理组主席则由来自 Digital Asset Holdings 的 CEO Blythe Masters 担任。另外，自 2016 年 5 月起，Apache 基金会创始人 Brian Behlendorf 担任超级账本项目的首位执行董事。2016 年 12 月，中国技术工作组 正式成立，负责本土社区组织和技术引导工作。官方网站也提供了十分详细的 组织结构信息。

该项目的出现，实际上宣布区块链技术已经不再是仅面向“社会实验”性质的应用场景，它已经正式被主流机构和企业市场认可；同时，Hyperledger 首次提出和实现的完备权限管理、创新的一致性算法和可拔插、可扩展的框架，对于区块链相关技术和产业的发展都将产生深远的影响。

2.项目组成

目前主要包括三大账本平台项目和若干其它项目。

账本平台项目：

- **Fabric：包括 Fabric、Fabric CA、Fabric SDK（包括 Node.Js、Python 和 Java 等语言）和 fabric-api、fabric-sdk-node、fabric-sdk-py 等，目标是区块链的基础核心平台，支持 pbft 等新的 consensus 机制，支持权限管理，最早由 IBM 和 DAH 发起；https://github.com/hyperledger/fabric
- SawToothLake：包括 arcade、core、dev-tools、validator、mktplace 等。是 Intel 主要发起和贡献的区块链平台，支持全新的基于硬件芯片的共识机制 Proof of Elapsed Time（PoET）。
- Iroha：账本平台项目，基于 C++ 实现，带有不少面向 Web 和 Mobile 的特性，主要由 Soramitsu 发起和贡献。

其它项目：

- Blockchain Explorer：提供 Web 操作界面，通过界面快速查看查询绑定区块链的状态（区块个数、交易历史）信息等。
- Cello：提供"Blockchain as a Service" 功能，使用 Cello，管理员可以轻松获取和管理多条区块链；应用开发者可以无需关心如何搭建和维护区块链。

项目约定共同遵守的 基本原则 为：

- 重视模块化设计，包括交易、合同、一致性、身份、存储等技术场景；
- 代码可读性，保障新功能和模块都可以很容易添加和扩展；
- 演化路线，随着需求的深入和更多的应用场景，不断增加和演化新的项目。

3.联盟链VS公链

公链： 针对所有用户 激励机制->代币

联盟链 针对特定的组织用户 没有激励机制 应用程序 数据共享 每一个行业都可以组成一个联盟

二.Fabric环境搭建

1.安装环境

推荐在 Linux（如 Ubuntu 16.04+）或 MacOS 环境中开发代码，并安装如下工具。

git：用来获取代码。

vim: 用来进行文本编辑

curl: 部署脚本中会使用curl命令

```
$ sudo apt update
$ sudo apt install git vim curl -y
```

Docker 1.12+：用来支持容器环境，注意 MacOS 下要用 Docker for Mac。

```
$ sudo apt install docker.io docker-compose -y
```

golang 1.10+安装成功后需要配置 $GOPATH 等环境变量。

- Fabric1.1.0版本要求Go1.9+
- Fabric1.0.0版本要求Go1.7+

```
$ tar -zxvf go1.10.3.linux-amd64.tar.gz -C /usr/local/
```

2.配置环境变量

```
$ sudo vim /etc/profile
```

添加如下内容

```
export GOPATH=$HOME/go
export GOROOT=/usr/local/go
export PATH=$GOROOT/bin:$PATH
```

让配置生效

```
$ source /etc/profile
$ go version
```

3.下载源码和镜像

3.1 创建存放目录

```
$ mkdir hyfa ;cd hyfa
```

3.2 下载源码与镜像

https://github.com/hyperledger/fabric/blob/master/scripts/bootstrap.sh

bootstrap.sh 脚本内容，默认版本为最新的版本:1.2.0

如果需要下载不同的版本号，$1=指定版本号即可，如./bootstrap.sh 1.1.0

```
#!/bin/bash
#
# Copyright IBM Corp. All Rights Reserved.
#
```

```
# SPDX-License-Identifier: Apache-2.0
#

# if version not passed in, default to latest released version
export VERSION=1.2.0
# if ca version not passed in, default to latest released version
export CA_VERSION=$VERSION
# current version of thirdparty images (couchdb, kafka and zookeeper)
released
export THIRDPARTY_IMAGE_VERSION=0.4.10
export ARCH=$(echo "$(uname -s|tr '[:upper:]' '[:lower:]'|sed
's/mingw64_nt.*/windows/')-$(uname -m | sed 's/x86_64/amd64/g')")
export MARCH=$(uname -m)

printHelp() {
  echo "Usage: bootstrap.sh [<version>] [<ca_version>]
[<thirdparty_version>][-d -s -b]"
  echo
  echo "-d - bypass docker image download"
  echo "-s - bypass fabric-samples repo clone"
  echo "-b - bypass download of platform-specific binaries"
  echo
  echo "e.g. bootstrap.sh 1.2.0 -s"
  echo "would download docker images and binaries for version 1.2.0"
}

dockerFabricPull() {
  local FABRIC_TAG=$1
  for IMAGES in peer orderer ccenv tools; do
      echo "==> FABRIC IMAGE: $IMAGES"
      echo
      docker pull hyperledger/fabric-$IMAGES:$FABRIC_TAG
      docker tag hyperledger/fabric-$IMAGES:$FABRIC_TAG hyperledger/fabric-
$IMAGES
  done
}

dockerThirdPartyImagesPull() {
  local THIRDPARTY_TAG=$1
  for IMAGES in couchdb kafka zookeeper; do
      echo "==> THIRDPARTY DOCKER IMAGE: $IMAGES"
      echo
      docker pull hyperledger/fabric-$IMAGES:$THIRDPARTY_TAG
      docker tag hyperledger/fabric-$IMAGES:$THIRDPARTY_TAG
hyperledger/fabric-$IMAGES
  done
}

dockerCaPull() {
      local CA_TAG=$1
      echo "==> FABRIC CA IMAGE"
      echo
      docker pull hyperledger/fabric-ca:$CA_TAG
      docker tag hyperledger/fabric-ca:$CA_TAG hyperledger/fabric-ca
}

samplesInstall() {
  # clone (if needed) hyperledger/fabric-samples and checkout corresponding
  # version to the binaries and docker images to be downloaded
  if [ -d first-network ]; then
    # if we are in the fabric-samples repo, checkout corresponding version
    echo "===> Checking out v${VERSION} branch of hyperledger/fabric-
samples"
    git checkout v${VERSION}
  elif [ -d fabric-samples ]; then
    # if fabric-samples repo already cloned and in current directory,
    # cd fabric-samples and checkout corresponding version
    echo "===> Checking out v${VERSION} branch of hyperledger/fabric-
samples"
    cd fabric-samples && git checkout v${VERSION}
  else
    echo "===> Cloning hyperledger/fabric-samples repo and checkout
v${VERSION}"
    git clone -b master https://github.com/hyperledger/fabric-samples.git
&& cd fabric-samples && git checkout v${VERSION}
  fi
}

# Incrementally downloads the .tar.gz file locally first, only
decompressing it
# after the download is complete. This is slower than binaryDownload() but
# allows the download to be resumed.
binaryIncrementalDownload() {
      local BINARY_FILE=$1
      local URL=$2
      curl -f -s -C - ${URL} -o ${BINARY_FILE} || rc=$?
      # Due to limitations in the current Nexus repo:
      # curl returns 33 when there's a resume attempt with no more bytes to
download
      # curl returns 2 after finishing a resumed download
      # with -f curl returns 22 on a 404
      if [ "$rc" = 22 ]; then
      # looks like the requested file doesn't actually exist so stop here
      return 22
      fi
      if [ -z "$rc" ] || [ $rc -eq 33 ] || [ $rc -eq 2 ]; then
          # The checksum validates that RC 33 or 2 are not real failures
          echo "==> File downloaded. Verifying the md5sum..."
```

```
            localMd5sum=$(md5sum ${BINARY_FILE} | awk '{print $1}')
            remoteMd5sum=$(curl -s ${URL}.md5)
            if [ "$localMd5sum" == "$remoteMd5sum" ]; then
                echo "==> Extracting ${BINARY_FILE}..."
                tar xzf ./${BINARY_FILE} --overwrite
            echo "==> Done."
                rm -f ${BINARY_FILE} ${BINARY_FILE}.md5
            else
                echo "Download failed: the local md5sum is different from the
remote md5sum. Please try again."
                rm -f ${BINARY_FILE} ${BINARY_FILE}.md5
                exit 1
            fi
        else
            echo "Failure downloading binaries (curl RC=$rc). Please try
again and the download will resume from where it stopped."
            exit 1
        fi
}

# This will attempt to download the .tar.gz all at once, but will trigger
the
# binaryIncrementalDownload() function upon a failure, allowing for resume
# if there are network failures.
binaryDownload() {
        local BINARY_FILE=$1
        local URL=$2
        echo "===> Downloading: " ${URL}
        # Check if a previous failure occurred and the file was partially
downloaded
        if [ -e ${BINARY_FILE} ]; then
            echo "==> Partial binary file found. Resuming download..."
            binaryIncrementalDownload ${BINARY_FILE} ${URL}
        else
            curl ${URL} | tar xz || rc=$?
            if [ ! -z "$rc" ]; then
                echo "==> There was an error downloading the binary file.
Switching to incremental download."
                echo "==> Downloading file..."
                binaryIncrementalDownload ${BINARY_FILE} ${URL}
        else
            echo "==> Done."
            fi
        fi
}

binariesInstall() {
  echo "===> Downloading version ${FABRIC_TAG} platform specific fabric
binaries"
```

```
  binaryDownload ${BINARY_FILE}
https://nexus.hyperledger.org/content/repositories/releases/org/hyperledger
/fabric/hyperledger-fabric/${ARCH}-${VERSION}/${BINARY_FILE}
  if [ $? -eq 22 ]; then
     echo
     echo "------> ${FABRIC_TAG} platform specific fabric binary is not
available to download <----"
     echo
   fi

  echo "===> Downloading version ${CA_TAG} platform specific fabric-ca-
client binary"
  binaryDownload ${CA_BINARY_FILE}
https://nexus.hyperledger.org/content/repositories/releases/org/hyperledger
/fabric-ca/hyperledger-fabric-ca/${ARCH}-${CA_VERSION}/${CA_BINARY_FILE}
  if [ $? -eq 22 ]; then
     echo
     echo "------> ${CA_TAG} fabric-ca-client binary is not available to
download  (Available from 1.1.0-rc1) <----"
     echo
   fi
}

dockerInstall() {
  which docker >& /dev/null
  NODOCKER=$?
  if [ "${NODOCKER}" == 0 ]; then
      echo "===> Pulling fabric Images"
      dockerFabricPull ${FABRIC_TAG}
      echo "===> Pulling fabric ca Image"
      dockerCaPull ${CA_TAG}
      echo "===> Pulling thirdparty docker images"
      dockerThirdPartyImagesPull ${THIRDPARTY_TAG}
      echo
      echo "===> List out hyperledger docker images"
      docker images | grep hyperledger*
  else
    echo "========================================================="
    echo "Docker not installed, bypassing download of Fabric images"
    echo "========================================================="
  fi
}

DOCKER=true
SAMPLES=true
BINARIES=true

# Parse commandline args pull out
# version and/or ca-version strings first
```

```
if [ ! -z $1 ]; then
  VERSION=$1;shift
  if [ ! -z $1 ]; then
    CA_VERSION=$1;shift
    if [ ! -z $1 ]; then
      THIRDPARTY_IMAGE_VERSION=$1;shift
    fi
  fi
fi

# prior to 1.2.0 architecture was determined by uname -m
if [[ $VERSION =~ ^1\.[0-1]\.* ]]; then
  export FABRIC_TAG=${MARCH}-${VERSION}
  export CA_TAG=${MARCH}-${CA_VERSION}
  export THIRDPARTY_TAG=${MARCH}-${THIRDPARTY_IMAGE_VERSION}
else
  # starting with 1.2.0, multi-arch images will be default
  : ${CA_TAG:="$CA_VERSION"}
  : ${FABRIC_TAG:="$VERSION"}
  : ${THIRDPARTY_TAG:="$THIRDPARTY_IMAGE_VERSION"}
fi

BINARY_FILE=hyperledger-fabric-${ARCH}-${VERSION}.tar.gz
CA_BINARY_FILE=hyperledger-fabric-ca-${ARCH}-${CA_VERSION}.tar.gz

# then parse opts
while getopts "h?dsb" opt; do
  case "$opt" in
    h|\?)
      printHelp
      exit 0
    ;;
    d)  DOCKER=false
    ;;
    s)  SAMPLES=false
    ;;
    b)  BINARIES=false
    ;;
  esac
done

if [ "$SAMPLES" == "true" ]; then
  echo
  echo "Installing hyperledger/fabric-samples repo"
  echo
  samplesInstall
fi
if [ "$BINARIES" == "true" ]; then
  echo
  echo "Installing Hyperledger Fabric binaries"
  echo
  binariesInstall
fi
if [ "$DOCKER" == "true" ]; then
  echo
  echo "Installing Hyperledger Fabric docker images"
  echo
  dockerInstall
fi
```

3.3 执行脚本

```
$chmod +x bootstrap.sh   ; ./bootstrap.sh
```

(1) 下载源码包

- 下载fabric压缩包

```
===> Downloading version 1.2.0 platform specific fabric binaries
===> Downloading:
https://nexus.hyperledger.org/content/repositories/releases/org/hyperledger
/fabric/hyperledger-fabric/linux-amd64-1.2.0/hyperledger-fabric-linux-
amd64-1.2.0.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100 39.0M  100 39.0M    0     0   482k      0  0:01:22  0:01:22 --:--:--
450k
==> Done.
```

- 下载fabric-ca-client压缩包

```
===> Downloading version 1.2.0 platform specific fabric-ca-client binary
===> Downloading:
https://nexus.hyperledger.org/content/repositories/releases/org/hyperledger
/fabric-ca/hyperledger-fabric-ca/linux-amd64-1.2.0/hyperledger-fabric-ca-
linux-amd64-1.2.0.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100 4940k  100 4940k    0     0   435k      0  0:00:11  0:00:11 --:--:--
538k
==> Done.
```

(2)下载Fabric核心模块镜像

- peer： fabric环境中的peer容器, 主要用来背书, 记账

  peer模块是主节点模块，负责存储区块链数据，运行维护链码

```
==> FABRIC IMAGE: peer

1.2.0: Pulling from hyperledger/fabric-peer
b234f539f7a1: Pull complete
55172d420b43: Pull complete
5ba5bbeb6b91: Pull complete
43ae2841ad7a: Pull complete
f6c9c6de4190: Pull complete
c6af77e36488: Pull complete
964f7f4f22f3: Pull complete
d4a3f4cfba3d: Pull complete
73782018d902: Pull complete
039eb34e730e: Pull complete
Digest:
sha256:949b38bad9496d7694b54d30b90b72653804d761a44d721c4dc7a16a5cbcabe8
Status: Downloaded newer image for hyperledger/fabric-peer:1.2.0
```

- order：主要用于对交易排序且生成区块

orderer模块负责对交易进行排序，并将排序好的交易打包成区块。

```
==> FABRIC IMAGE: orderer

1.2.0: Pulling from hyperledger/fabric-orderer
b234f539f7a1: Already exists
55172d420b43: Already exists
5ba5bbeb6b91: Already exists
43ae2841ad7a: Already exists
f6c9c6de4190: Already exists
c6af77e36488: Already exists
964f7f4f22f3: Already exists
d4a3f4cfba3d: Already exists
8d0e11f5a0f7: Pull complete
0bda0f004d88: Pull complete
Digest:
sha256:1a8cbe6abef245432730035d08ea1d1aa54a50717136fa3be58f8af4570ad57e
Status: Downloaded newer image for hyperledger/fabric-orderer:1.2.0
```

- ccenv：针对Go语言的容器

```
==> FABRIC IMAGE: ccenv
```

```
1.2.0: Pulling from hyperledger/fabric-ccenv
b234f539f7a1: Already exists
55172d420b43: Already exists
5ba5bbeb6b91: Already exists
43ae2841ad7a: Already exists
f6c9c6de4190: Already exists
c6af77e36488: Already exists
964f7f4f22f3: Already exists
13cd31405e09: Pull complete
e03b35c19d96: Pull complete
96c2920985e3: Pull complete
e91461be8304: Pull complete
950c3368692b: Pull complete
c5de8d20c137: Pull complete
5536b64bc67b: Pull complete
Digest:
sha256:0a7fb37111cafce79cf89ca8d1af5ca6f721e60a8bd1b2b93521e671e3348af2
Status: Downloaded newer image for hyperledger/fabric-ccenv:1.2.0
```

- tools：fabric环境中的工具容器, 主要用来测试客户端

```
==> FABRIC IMAGE: tools

1.2.0: Pulling from hyperledger/fabric-tools
b234f539f7a1: Already exists
55172d420b43: Already exists
5ba5bbeb6b91: Already exists
43ae2841ad7a: Already exists
f6c9c6de4190: Already exists
c6af77e36488: Already exists
964f7f4f22f3: Already exists
13cd31405e09: Already exists
e03b35c19d96: Already exists
96c2920985e3: Already exists
e91461be8304: Already exists
314928def9dd: Pull complete
d5b68ae13f8d: Pull complete
dde25187799d: Pull complete
Digest:
sha256:284f997b33d6745b52d378f8c7ba1a208b8c13633f3ef63e68377b1986077cb6
Status: Downloaded newer image for hyperledger/fabric-tools:1.2.0
```

(3)下载Fabric的CA镜像

Fabric环境中的CA容器(fabric-ca: 用于认证, 根据配置文件中指定的路径检查当前用户是否合法)

```
===> Pulling fabric ca Image
==> FABRIC CA IMAGE
```

```
1.2.0: Pulling from hyperledger/fabric-ca
b234f539f7a1: Already exists
55172d420b43: Already exists
5ba5bbeb6b91: Already exists
43ae2841ad7a: Already exists
f6c9c6de4190: Already exists
c6af77e36488: Already exists
964f7f4f22f3: Already exists
f7a6370a6f7f: Pull complete
37cc94e973b2: Pull complete
a80e45d2f608: Pull complete
8feb19f73d3a: Pull complete
5f3ea048e0c0: Pull complete
Digest:
sha256:0a6f8b3af8537fa725dc40d09565c77b1b284f848f653c32cb4125b3134a8726
Status: Downloaded newer image for hyperledger/fabric-ca:1.2.0
```

(4) 下载第三方镜像

- couchdb：是一个可选DB，可选, 可将Fabric中默认的DB替代为couchDB

  Apache CouchDB是一个开源数据库，它是一个使用JSON作为存储格式，JavaScript作为查询语言，MapReduce和HTTP作为API的NoSQL数据库。

```
===> Pulling thirdparty docker images
==> THIRDPARTY DOCKER IMAGE: couchdb

0.4.10: Pulling from hyperledger/fabric-couchdb
b234f539f7a1: Already exists
55172d420b43: Already exists
5ba5bbeb6b91: Already exists
43ae2841ad7a: Already exists
f6c9c6de4190: Already exists
c6af77e36488: Already exists
964f7f4f22f3: Already exists
13cd31405e09: Already exists
e03b35c19d96: Already exists
96c2920985e3: Already exists
e91461be8304: Already exists
6a752ce8f7fe: Pull complete
a49e2cb854b0: Pull complete
493b25e70e6d: Pull complete
2721753a3e7c: Pull complete
adede0f2a5f1: Pull complete
9eb593f76305: Pull complete
bb49a3450e11: Pull complete
929b9bb5d788: Pull complete
```

```
Digest:
sha256:d7eb3fd24acafaeaaae94a44659409270b89bd599d017cf9d5b75d8f21438b51
Status: Downloaded newer image for hyperledger/fabric-couchdb:0.4.10
```

- kafka：

  Kafka是一个消息系统，原本开发自LinkedIn，用作LinkedIn的活动流（Activity Stream）和运营数据处理管道（Pipeline）的基础。现在它已被多家不同类型的公司 作为多种类型的数据管道和消息系统使用。

```
==> THIRDPARTY DOCKER IMAGE: kafka

0.4.10: Pulling from hyperledger/fabric-kafka
b234f539f7a1: Already exists
55172d420b43: Already exists
5ba5bbeb6b91: Already exists
43ae2841ad7a: Already exists
f6c9c6de4190: Already exists
c6af77e36488: Already exists
964f7f4f22f3: Already exists
13cd31405e09: Already exists
e03b35c19d96: Already exists
96c2920985e3: Already exists
e91461be8304: Already exists
146aa6695f33: Pull complete
954e41d8cd46: Pull complete
9d750070047f: Pull complete
Digest:
sha256:7c07db5b38ca3259528b3e53691ecd273e44d1521218aa8f3a5dc34ab6947ff5
Status: Downloaded newer image for hyperledger/fabric-kafka:0.4.10
```

- zookeeper：主要用来作共识，在开发测试环境下可选, 一般在开发测试环境下为单点模式

ZooKeeper曾是Hadoop的正式子项目，后发展成为Apache顶级项目，与Hadoop密切相关但却没有任何依赖。它是一个针对大型应用提供高可用的数据管理、应用程序协调服务的分布式服务框架，基于对Paxos算法的实现，使该框架保证了分布式环境中数据的强一致性，提供的功能包括：配置维护、统一命名服务、状态同步服务、集群管理等。

```
==> THIRDPARTY DOCKER IMAGE: zookeeper

0.4.10: Pulling from hyperledger/fabric-zookeeper
b234f539f7a1: Already exists
55172d420b43: Already exists
5ba5bbeb6b91: Already exists
43ae2841ad7a: Already exists
f6c9c6de4190: Already exists
c6af77e36488: Already exists
964f7f4f22f3: Already exists
```

```
13cd31405e09: Already exists
e03b35c19d96: Already exists
96c2920985e3: Already exists
e91461be8304: Already exists
c335e6e59168: Pull complete
016e000b0cc8: Pull complete
e716b0c9790c: Pull complete
390f47e71470: Pull complete
Digest:
sha256:151fe67421663fe860c1aba4b80877a22b6b8fa18cbc97990c309c603cd6f5f1
Status: Downloaded newer image for hyperledger/fabric-zookeeper:0.4.10
```

3.4 脚本执行结果

(1) 源码包文件结构

```
bruce@ubuntu:~$ls hyfa/fabric-samples/ -l
```

返回结果

```
total 76
drwxrwxr-x 5 bruce bruce  4096 Jul 10 00:59 balance-transfer
drwxrwxr-x 4 bruce bruce  4096 Jul 10 00:59 basic-network
drwxrwxr-x 2  1001  1001  4096 Jul  3 13:41 bin
drwxrwxr-x 8 bruce bruce  4096 Jul 10 00:59 chaincode
drwxrwxr-x 3 bruce bruce  4096 Jul 10 00:59 chaincode-docker-devmode
-rw-rw-r-- 1 bruce bruce   597 Jul 10 00:59 CODE_OF_CONDUCT.md
drwxrwxr-x 2  1001  1001  4096 Jul  3 12:04 config
-rw-rw-r-- 1 bruce bruce   961 Jul 10 00:59 CONTRIBUTING.md
drwxrwxr-x 2 bruce bruce  4096 Jul 10 00:59 fabcar
drwxrwxr-x 3 bruce bruce  4096 Jul 10 00:59 fabric-ca
drwxrwxr-x 6 bruce bruce  4096 Jul 10 00:59 first-network
drwxrwxr-x 4 bruce bruce  4096 Jul 10 00:59 high-throughput
-rw-rw-r-- 1 bruce bruce  3095 Jul 10 00:59 Jenkinsfile
-rw-rw-r-- 1 bruce bruce 11358 Jul 10 00:59 LICENSE
-rw-rw-r-- 1 bruce bruce   470 Jul 10 00:59 MAINTAINERS.md
-rw-rw-r-- 1 bruce bruce  1229 Jul 10 00:59 README.md
drwxrwxr-x 3 bruce bruce  4096 Jul 10 00:59 scripts
```

(2) 查看docker镜像

```
bruce@ubuntu:~$ sudo docker images
```

返回结果

```
REPOSITORY                      TAG                 IMAGE ID
CREATED             SIZE
hyperledger/fabric-ca           1.2.0               66cc132bd09c        6
days ago           252MB
hyperledger/fabric-ca           latest              66cc132bd09c        6
days ago           252MB
hyperledger/fabric-tools        1.2.0               379602873003        6
days ago           1.51GB
hyperledger/fabric-tools        latest              379602873003        6
days ago           1.51GB
hyperledger/fabric-ccenv        1.2.0               6acf31e2d9a4        6
days ago           1.43GB
hyperledger/fabric-ccenv        latest              6acf31e2d9a4        6
days ago           1.43GB
hyperledger/fabric-orderer      1.2.0               4baf7789a8ec        6
days ago           152MB
hyperledger/fabric-orderer      latest              4baf7789a8ec        6
days ago           152MB
hyperledger/fabric-peer         1.2.0               82c262e65984        6
days ago           159MB
hyperledger/fabric-peer         latest              82c262e65984        6
days ago           159MB
hyperledger/fabric-zookeeper    0.4.10              2b51158f3898        11
days ago          1.44GB
hyperledger/fabric-zookeeper    latest              2b51158f3898        11
days ago          1.44GB
hyperledger/fabric-kafka        0.4.10              936aef6db0e6        11
days ago          1.45GB
hyperledger/fabric-kafka        latest              936aef6db0e6        11
days ago          1.45GB
hyperledger/fabric-couchdb      0.4.10              3092eca241fc        11
days ago          1.61GB
hyperledger/fabric-couchdb      latest              3092eca241fc        11
days ago          1.61GB
```

4.再次配置环境变量

为fabric的相关命令配置环境变量

```
$ sudo vim /etc/profile
```

添加如下内容：

```
FabricSampleDir="/home/bruce/hyfa/fabric-samples"
export PATH=${FabricSampleDir}/bin:$PATH
```

让配置生效

```
$ source /etc/profile
```

三.Fabric目录结构

1.三大核心工具

```
fabric-samples/bin/cryptogen
        cryptogen: 根据指定的配置文件,生成组织结构及身份证书的工具

fabric-samples/bin/configtxgen
        configtxgen: 主要生成三种配置文件
            1. 生成Orderer初始区块
            2. 生成应用通道交易配置文件
            3. 锚节点更新配置文件
                锚节点: 数据交换
fabric-samples/bin/configtxlator
        configtxlator: 用来在正在运行的网络联盟链中添加一个新的组织
```

2.关键目录

```
chaincode/
        Fabric示例链码所在目录
chaincode-docker-devmode/
        开启开发测试模式
        在开发测试模式下链码的存放目录
config/
        关于orderer及peer配置信息的文件存放目录
        configtx.yaml: 生成初始区块及应用通道交易配置文件的参考
        core.yaml: peer配置信息的参考
        orderer.yaml: orderer配置信息的参考
fabcar/
        测试Node环境的所在目录
        有一个小汽车的应用示例
fabric-ca/
        fabric基础环境提供的一个简单fabric-ca
first-network/
        与Fabric网络相关的所有内容
```

3.其他概念

```
联盟:
    org1:
        peer0.org1.example.com
        peer1.org1.example.com

    org2:
        peer0.org2.example.com
        peer1.org2.example.com
应用通道:
    用于隔离不同联盟之间交易

Peer节点角色:
    1. 背书节点
        -- 在fabric中只能部分节点是背书节点
    2. 记账节点(commiter)
        -- 在fabric中所有的Peer节点都是记账节点
    3. 锚节点
        -- 通过配置文件指定
        -- 作用: 用于跨组织交换数据

背书: 在fabric中, 就是签名的概念
```

四.Node环境搭建

后期我们需要使用Fabric SDK做应用程序的开发，官方虽然提供了Node.JS，Java，Go，Python等多种语言的SDK，但是由于整个Fabric太新了，很多SDK还不成熟和完善，所以一般采用Node JS的SDK，毕竟这个是功能毕竟齐全，而且也是官方示例的时候使用的SDK。

1.安装nvm

由于Node版本的迭代速度很快，版本很多，不同项目对node的依赖不同，故需要切换不同node版本目前有n和nvm这两个工具可以对Node进行平滑升级，n 命令是作为一个 node 的模块而存在，而 nvm 是一个独立于 node/npm 的外部 shell 脚本，因此 nvm 命令相比 n更加全面，n更有局限性。

nvm 是 node 管理工具，有点类似管理 Ruby 的 rvm，如果是需要管理 Windows 下的 node，官方推荐是使用 nvmw 或 nvm-windows 。

```
$ sudo apt update
$ curl -o-
https://raw.githubusercontent.com/creationix/nvm/v0.33.10/install.sh | bash

$ export NVM_DIR="$HOME/.nvm"
$ [ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"
```

2.安装Node与npm

NPM是随同NodeJS一起安装的**包管理工具**，能解决NodeJS代码部署上的很多问题，常见的使用场景有以下几种：

- 允许用户从NPM服务器下载别人编写的第三方包到本地使用。
- 允许用户从NPM服务器下载并安装别人编写的命令行程序到本地使用。
- 允许用户将自己编写的包或命令行程序上传到NPM服务器供别人使用。

```
$ nvm install v8.11.1
```

检查Node版本

```
$ node -v
v8.11.1
```

检查npm版本

```
$ npm -v
5.6.0
```