Hyperledger Fabric SDK 构建应用程序

介绍Hyperledger Fabric Go SDK，基于区块链构建一个应用程序。

1. 先决条件
2. Hyperledger Fabric简介
3. 安装指南
    1. Docker
    2. Docker-compose
    3. Go
    4. Fabric SDK Go
4. 制作第一个区块链网络
    1. 准备环境
    2. 测试
5. 使用Fabric SDK Go
6. 链码开发及安装与实例化
7. 链码调用
8. 在Web应用程序中进行设置

1.先决条件

在**Ubuntu 16.04**上发布, 使用Go语言来设计应用程序，因为Hyperledger Fabric也是在Go中构建的，Fabric SDK Go的使用非常简单

Hyperledger Fabric使用Docker轻松部署区块链网络。 另外，一些组件（同级）也部署docker容器来分离数据（通道）。 所以请确保所使用的平台支持这种虚拟化。

2. Hyperledger Fabric介绍

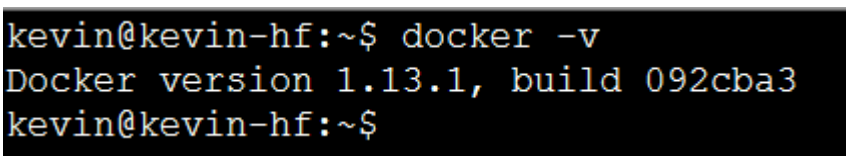
> Hyperledger Fabric是分布式账本解决方案的平台，支持模块化架构，提供高度的机密性，弹性，灵活性和可扩展性。 它旨在支持不同组件的可插拔实现，并适应经济生态系统中存在的复杂性。

3.安装指南

a. Docker

需要Docker版本1.12或更高版本。

```
docker -v
```

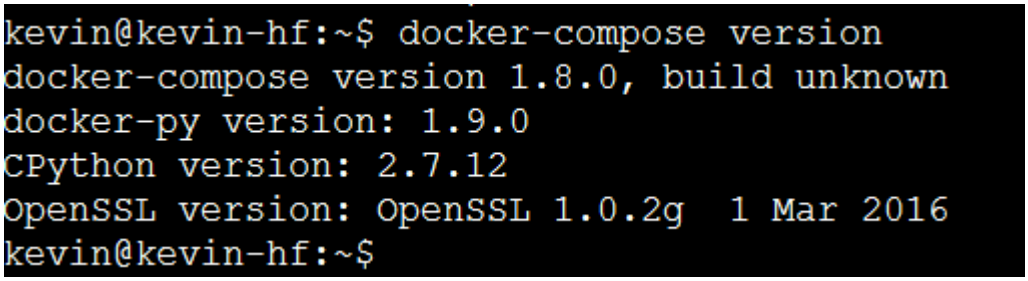


b. Docker Compose

Docker-compose 1.8或更高版本是必需的。

我们目前无法一次性轻松管理多个容器。 为了解决这个问题，需要**docker-compose** 。

```
docker-compose version
```

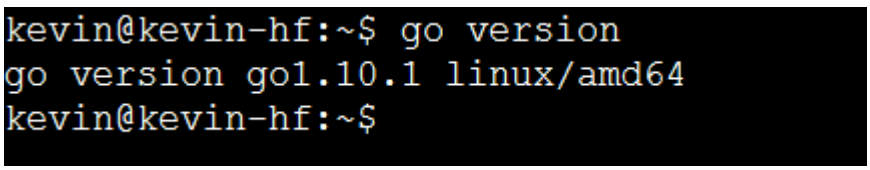


C. Go

需要版本1.10.x或更高版本。

```
go version
```

设置GOPATH & GOROOT环境变量, 通过 `go env` 查看GOPATH路径



d. Fabric SDK Go

安装软件包`libltdl-dev`

```
$ sudo apt update
$ sudo apt install libltdl-dev
```

将当前用户添加到Docker组

否则在执行make命令时会造成错误: `ERROR: Couldn't connect to Docker daemon at http+docker://localunixsocket - is it running?`

```
$ sudo usermod -aG docker kevin
```

添加成功后**必须注销/退出并重新登录**(退出终端重新连接即可)

执行以下命令请确保网络通畅及稳定

安装Hyperledger Fabric SDK Go，可以很容易的与Fabric的组件进行通信。 不需要安装Fabric或Fabric CA框架，因为SDK会自动在本地处理。

```
$ go get -u --tags nopkcs11
github.com/hyperledger/fabric/core/chaincode/shim
```

> **将fabric-sdk-go目录上传并解压至$GOPATH/src/github.com/hyperledger目录下**
>
> ```
> $ unzip -d $GOPATH/src/github.com/hyperledger fabric-sdk-go.zip
> ```

将如下环境变量设置到用户的环境文件中(.bashrc)中

```
export PATH=$PATH:$GOPATH/bin
```

确保有所需的所有依赖关系：

```
$ cd $GOPATH/src/github.com/hyperledger/fabric-sdk-go
$ make depend-install
```

在执行 `make depend-install` 的时候很有可能出现多种错误，这些错误基本上都是由网络原因造成

如出现以下错误:

```
Please replace the Apache license header comment text with:
SPDX-License-Identifier: Apache-2.0

Checking committed files for traditional Apache License headers ...
The following files are missing traditional Apache 2.0 headers:
ci.properties
Fatal Error - All files must have a license header
Makefile:185: recipe for target 'license' failed
make: *** [license] Error 1
```

将 `test/scripts/check_license.sh` 文件中的内容替换为:

```
#!/bin/bash
#
# Copyright IBM Corp, SecureKey Technologies Inc. All Rights
Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#

function filterExcludedFiles {
  CHECK=`echo "$CHECK" | grep -v .png$ | grep -v .rst$ | grep -v
^.git/ \
  | grep -v .pem$ | grep -v .block$ | grep -v .tx$ | grep -v
^LICENSE$ | grep -v _sk$ \
  | grep -v .key$ | grep -v .crt$ | grep -v \\.gen.go$ | grep -v
\\.json$ | grep -v Gopkg.lock$ \
  | grep -v .md$ | grep -v ^vendor/ | grep -v ^build/ | grep -v
.pb.go$ | grep -v ci.properties$ | sort -u`
}

CHECK=$(git diff --name-only --diff-filter=ACMRTUXB HEAD)
filterExcludedFiles
if [[ -z "$CHECK" ]]; then
  LAST_COMMITS=($(git log -2 --pretty=format:"%h"))
  CHECK=$(git diff-tree --no-commit-id --name-only --diff-
filter=ACMRTUXB -r ${LAST_COMMITS[1]} ${LAST_COMMITS[0]})
  filterExcludedFiles
```

```
fi

if [[ -z "$CHECK" ]]; then
   echo "All files are excluded from having license headers"
   exit 0
fi

missing=`echo "$CHECK" | xargs ls -d 2>/dev/null | xargs grep -L
"SPDX-License-Identifier"`
if [[ -z "$missing" ]]; then
   echo "All files have SPDX-License-Identifier headers"
   exit 0
fi
echo "The following files are missing SPDX-License-Identifier
headers:"
echo "$missing"
echo
echo "Please replace the Apache license header comment text with:"
echo "SPDX-License-Identifier: Apache-2.0"

echo
echo "Checking committed files for traditional Apache License
headers ..."
missing=`echo "$missing" | xargs ls -d 2>/dev/null | xargs grep -L
"http://www.apache.org/licenses/LICENSE-2.0"`
if [[ -z "$missing" ]]; then
   echo "All remaining files have Apache 2.0 headers"
   exit 0
fi
echo "The following files are missing traditional Apache 2.0
headers:"
echo "$missing"
echo "Fatal Error - All files must have a license header"
exit 1
```

来源: https://github.com/hyperledger/fabric-sdk-go/blob/master/test/scripts/check_license.sh

修改配置文件

```
$ vim Makefile
fabric-sdk-go/Makefile文件中
    FABRIC_STABLE_VERSION           := 1.2.0
    FABRIC_STABLE_VERSION_MINOR     := 1.2
    FABRIC_STABLE_VERSION_MAJOR     := 1
    FABRIC_BASEIMAGE_STABLE_VERSION := 0.4.10

    FABRIC_PRERELEASE_VERSION       := 1.2.0-alpha
    FABRIC_PREV_VERSION             := 1.2.0
    FABRIC_DEVSTABLE_VERSION_MINOR  := 1.1
    FABRIC_DEVSTABLE_VERSION_MAJOR  := 1


$ vim test/fixtures/dockerenv/.evn
fabric-sdk-go/test/fixtures/dockerenv/.env 文件中
    FABRIC_FIXTURE_VERSION=v1.2
    FABRIC_CRYPTOCONFIG_VERSION=v1

    FABRIC_CA_FIXTURE_TAG=1.2.0
    FABRIC_ORDERER_FIXTURE_TAG=1.2.0
    FABRIC_PEER_FIXTURE_TAG=1.2.0
    FABRIC_COUCHDB_FIXTURE_TAG=1.2.0
    FABRIC_BUILDER_FIXTURE_TAG=1.2.0
    FABRIC_BASEOS_FIXTURE_TAG=0.4.10
    FABRIC_BASEIMAGE_FIXTURE_TAG=0.4.10
```

4.创建第一个区块链网络

a. 准备环境

为了构建区块链网络，使用`docker`构建处理不同角色的虚拟计算机。 在这里我们将尽可能保持简单。 Hyperledger Fabric需要大量证书来确保在整个端到端流程（TSL，身份验证，签名块……）期间进行加密。 创建这些文件需要一点时间，为了直接了解问题的核心，我们已经在此存储库的文件夹中为您准备了所有相关内容。

在`GOPATH`的`src`文件夹中新建一个目录如下：

```
$ mkdir -p $GOPATH/src/github.com/kongyixueyuan.com/bill
$ cd $GOPATH/src/github.com/kongyixueyuan.com/bill
```

新建`fixtures`文件夹

```
$ mkdir fixtures
```

将`channel-artifacts`及`crypto-config`两个文件夹复制到`fixture`目录中

```
$ cd fixtures
$ sudo cp -r ~/hyfa/fabric-samples/first-network/channel-artifacts
./
$ sudo cp -r ~/hyfa/fabric-samples/first-network/crypto-config ./
```

将`channel-artifacts`文件夹名称修改为`artifacts`

```
$ mv channel-artifacts/ artifacts
```

移除无用的文件

```
$ sudo rm -f artifacts/.gitkeep
```

将`fabric-samples/basic-network/docker-compose.yml`文件复制至当前的`fixtures`目录下, 进行编辑

```
$ sudo cp ~/hyfa/fabric-samples/basic-network/docker-compose.yml ./
$ sudo vim docker-compose.yml
```

1. 将 `network下的basic` 修改为 `default`

```
version: '2'

networks:
  default:

services:
```

2. 编辑 orderer 部分

```
 orderer.example.com:
    container_name: orderer.example.com
    image: hyperledger/fabric-orderer
    environment:
      - ORDERER_GENERAL_LOGLEVEL=debug
      - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
      - ORDERER_GENERAL_GENESISMETHOD=file
      -
ORDERER_GENERAL_GENESISFILE=/var/hyperledger/orderer/orderer.gene
sis.block
      - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
      - ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/msp
      - ORDERER_GENERAL_LISTENPORT=7050
      # enabled TLS
      - ORDERER_GENERAL_TLS_ENABLED=true
      -
ORDERER_GENERAL_TLS_PRIVATEKEY=/var/hyperledger/orderer/tls/serve
r.key
      -
ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/serv
er.crt
      - ORDERER_GENERAL_TLS_ROOTCAS=
[/var/hyperledger/orderer/tls/ca.crt,
/var/hyperledger/peerOrg1/tls/ca.crt,
/var/hyperledger/peerOrg2/tls/ca.crt]
```

```
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric
    command: orderer
    ports:
      - 7050:7050
    volumes:
      -
./artifacts/genesis.block:/var/hyperledger/orderer/orderer.genesi
s.block
      - ./crypto-
config/ordererOrganizations/example.com/orderers/orderer.example.
com/msp:/var/hyperledger/orderer/msp
      - ./crypto-
config/ordererOrganizations/example.com/orderers/orderer.example.
com/tls:/var/hyperledger/orderer/tls
      - ./crypto-
config/peerOrganizations/org1.example.com/peers/peer0.org1.exampl
e.com/:/var/hyperledger/peerOrg1
      - ./crypto-
config/peerOrganizations/org2.example.com/peers/peer0.org2.exampl
e.com/:/var/hyperledger/peerOrg2
    networks:
      default:
        aliases:
          - orderer.example.com
```

3. 编辑 ca 部分

```
  ca.org1.example.com:
    image: hyperledger/fabric-ca
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca.org1.example.com
      - FABRIC_CA_SERVER_CA_CERTFILE=/etc/hyperledger/fabric-ca-
server-config/ca.org1.example.com-cert.pem
      # path: crypto-
config\peerOrganizations\org1.example.com\ca\..._sk
```

```
      - FABRIC_CA_SERVER_CA_KEYFILE=/etc/hyperledger/fabric-ca-
server-
config/e9d74f61229b0b6b12c113940cf77de80996cace5911134cc9425c5232
d9234d_sk
      - FABRIC_CA_SERVER_TLS_ENABLED=true
      - FABRIC_CA_SERVER_TLS_CERTFILE=/etc/hyperledger/fabric-ca-
server-config/ca.org1.example.com-cert.pem
      - FABRIC_CA_SERVER_TLS_KEYFILE=/etc/hyperledger/fabric-ca-
server-
config/e9d74f61229b0b6b12c113940cf77de80996cace5911134cc9425c5232
d9234d_sk
    ports:
      - "7054:7054"
    command: sh -c 'fabric-ca-server start -b admin:adminpw -d'
    volumes:
      - ./crypto-
config/peerOrganizations/org1.example.com/ca/:/etc/hyperledger/fa
bric-ca-server-config
    container_name: ca.org1.example.com
    networks:
      default:
        aliases:
          - ca.org1.example.com
```

4. 编辑Peer部分
    1. `peer0.org1.example.com` 内容如下

```
   peer0.org1.example.com:
     image: hyperledger/fabric-peer
     container_name: peer0.org1.example.com
     environment:
       - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
       - CORE_VM_DOCKER_ATTACHSTDOUT=true
       - CORE_LOGGING_LEVEL=DEBUG
       - CORE_PEER_NETWORKID=bill
       - CORE_PEER_PROFILE_ENABLED=true
```

```
      - CORE_PEER_TLS_ENABLED=true
      -
CORE_PEER_TLS_CERT_FILE=/var/hyperledger/tls/server.crt
      -
CORE_PEER_TLS_KEY_FILE=/var/hyperledger/tls/server.key
      -
CORE_PEER_TLS_ROOTCERT_FILE=/var/hyperledger/tls/ca.crt
      - CORE_PEER_ID=peer0.org1.example.com
      - CORE_PEER_ADDRESSAUTODETECT=true
      - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
      -
CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:7051
      - CORE_PEER_GOSSIP_USELEADERELECTION=true
      - CORE_PEER_GOSSIP_ORGLEADER=false
      - CORE_PEER_GOSSIP_SKIPHANDSHAKE=true
      - CORE_PEER_LOCALMSPID=Org1MSP
      - CORE_PEER_MSPCONFIGPATH=/var/hyperledger/msp
      -
CORE_PEER_TLS_SERVERHOSTOVERRIDE=peer0.org1.example.com
    working_dir:
/opt/gopath/src/github.com/hyperledger/fabric/peer
    command: peer node start
    volumes:
      - /var/run/:/host/var/run/
      - ./crypto-
config/peerOrganizations/org1.example.com/peers/peer0.org1.ex
ample.com/msp:/var/hyperledger/msp
      - ./crypto-
config/peerOrganizations/org1.example.com/peers/peer0.org1.ex
ample.com/tls:/var/hyperledger/tls
    ports:
      - 7051:7051
      - 7053:7053
    depends_on:
      - orderer.example.com
    links:
      - orderer.example.com
    networks:
```

```
      default:
        aliases:
          - peer0.org1.example.com
```

2. peer1.org1.example.com 内容如下

```
  peer1.org1.example.com:
    image: hyperledger/fabric-peer
    container_name: peer1.org1.example.com
    environment:
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - CORE_VM_DOCKER_ATTACHSTDOUT=true
      - CORE_LOGGING_LEVEL=DEBUG
      - CORE_PEER_NETWORKID=bill
      - CORE_PEER_PROFILE_ENABLED=true
      - CORE_PEER_TLS_ENABLED=true
      -
CORE_PEER_TLS_CERT_FILE=/var/hyperledger/tls/server.crt
      -
CORE_PEER_TLS_KEY_FILE=/var/hyperledger/tls/server.key
      -
CORE_PEER_TLS_ROOTCERT_FILE=/var/hyperledger/tls/ca.crt
      - CORE_PEER_ID=peer1.org1.example.com
      - CORE_PEER_ADDRESSAUTODETECT=true
      - CORE_PEER_ADDRESS=peer1.org1.example.com:7051
      -
CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org1.example.com:7051
      - CORE_PEER_GOSSIP_USELEADERELECTION=true
      - CORE_PEER_GOSSIP_ORGLEADER=false
      - CORE_PEER_GOSSIP_SKIPHANDSHAKE=true
      - CORE_PEER_LOCALMSPID=Org1MSP
      - CORE_PEER_MSPCONFIGPATH=/var/hyperledger/msp
      -
CORE_PEER_TLS_SERVERHOSTOVERRIDE=peer1.org1.example.com
    working_dir:
/opt/gopath/src/github.com/hyperledger/fabric/peer
    command: peer node start
    volumes:
```

```
      - /var/run/:/host/var/run/
      - ./crypto-
config/peerOrganizations/org1.example.com/peers/peer1.org1.ex
ample.com/msp:/var/hyperledger/msp
      - ./crypto-
config/peerOrganizations/org1.example.com/peers/peer1.org1.ex
ample.com/tls:/var/hyperledger/tls
    ports:
      - 8051:7051
      - 8053:7053
    depends_on:
      - orderer.example.com
    links:
      - orderer.example.com
    networks:
      default:
        aliases:
          - peer1.org1.example.com
```

将 `fixtures` 文件的所属修改为当前用户及组

```
$ sudo chown -R kevin:kevin ../fixtures
```

b. 测试

为了检查网络是否正常工作，使用`docker-compose`同时启动或停止所有容器。 进入`fixtures`文件夹，运行：

```
$ cd $GOPATH/src/github.com/kongyixueyuan.com/bill/fixtures
$ docker-compose up -d
```

控制台会输出很多不同颜色的日志（红色不等于错误）。

```
 Module 'gossip/discovery' logger enabled for log level 'WARNING'
peer1.org1.example.com    | 2018-06-09 04:32:54.704 UTC [flogging] setModuleLevel -> DEBU 1b1
 Module 'gossip/election' logger enabled for log level 'WARNING'
peer1.org1.example.com    | 2018-06-09 04:32:54.704 UTC [flogging] setModuleLevel -> DEBU 1b2
 Module 'gossip/pull' logger enabled for log level 'WARNING'
peer1.org1.example.com    | 2018-06-09 04:32:54.705 UTC [flogging] setModuleLevel -> DEBU 1b3
 Module 'gossip/gossip' logger enabled for log level 'WARNING'
peer1.org1.example.com    | 2018-06-09 04:32:54.705 UTC [flogging] setModuleLevel -> DEBU 1b4
 Module 'ledgermgmt' logger enabled for log level 'INFO'
peer1.org1.example.com    | 2018-06-09 04:32:54.705 UTC [flogging] setModuleLevel -> DEBU 1b5
 Module 'cauthdsl' logger enabled for log level 'WARNING'
peer1.org1.example.com    | 2018-06-09 04:32:54.705 UTC [flogging] setModuleLevel -> DEBU 1b6
 Module 'policies' logger enabled for log level 'WARNING'
peer1.org1.example.com    | 2018-06-09 04:32:54.705 UTC [flogging] setModuleLevel -> DEBU 1b7
 Module 'grpc' logger enabled for log level 'ERROR'
peer1.org1.example.com    | 2018-06-09 04:32:54.705 UTC [flogging] setModuleLevel -> DEBU 1b8
 Module 'peer/gossip/mcs' logger enabled for log level 'WARNING'
peer1.org1.example.com    | 2018-06-09 04:32:54.705 UTC [flogging] setModuleLevel -> DEBU 1b9
 Module 'peer/gossip/sa' logger enabled for log level 'WARNING'
peer1.org1.example.com    | 2018-06-09 04:32:54.706 UTC [nodeCmd] func7 -> INFO 1ba Starting
profiling server with listenAddress = 0.0.0.0:6060
```

打开一个新终端并运行：

```
$ docker ps
```

```
kevin@kevin-hf:~$ docker ps
CONTAINER ID        IMAGE                      COMMAND                  CREATED
 STATUS              PORTS                                           NAMES
ea8a366994b5        hyperledger/fabric-peer    "peer node start"        13 seconds ago
 Up 11 seconds       0.0.0.0:7051->7051/tcp, 0.0.0.0:7053->7053/tcp  peer0.org1.example.com
f3488787566b        hyperledger/fabric-peer    "peer node start"        13 seconds ago
 Up 11 seconds       0.0.0.0:8051->7051/tcp, 0.0.0.0:8053->7053/tcp  peer1.org1.example.com
10f2be6c099d        hyperledger/fabric-orderer "orderer"                13 seconds ago
 Up 12 seconds       0.0.0.0:7050->7050/tcp                          orderer.example.com
da88565be84a        hyperledger/fabric-ca      "sh -c 'fabric-ca-..."   13 seconds ago
 Up 12 seconds       0.0.0.0:7054->7054/tcp                          ca.org1.example.com
kevin@kevin-hf:~$
```

将看到：两个peer，orderer和一个CA容器。 代表已成功创建了一个新的网络，可以随SDK一起使用。 要停止网络，请返回到上一个终端，按`Ctrl+C`并等待所有容器都停止。

> **提示** ：当网络停止时，所有使用的容器都可以访问。 例如，这对检查日志非常有用。 可以用`docker ps -a`来看它们。 为了清理这些容器，需要使用`docker rm $(docker ps -aq)`将其删除，或者如果使用了`docker-compose`文件，请转至此文件的位置并运行`docker-compose down`

> **提示** ：可以在后台运行`docker-compose`命令以保持提示。 为此，请使用参数`-d` ，如下所示： `docker-compose up -d` 。 要停止容器，请在`docker-compose.yaml`所在的文件夹中运行命令： `docker-compose stop` （或`docker-compose down`进行清理停止所有容器）。

最后执行命令

```
$ cd $GOPATH/src/github.com/kongyixueyuan.com/bill/fixtures
$ docker-compose down
```

```
^CGracefully stopping... (press Ctrl+C again to force)
Stopping peer0.org1.example.com ... done
Stopping peer1.org1.example.com ... done
Stopping orderer.example.com ... done
Stopping ca.org1.example.com ... done
kevin@kevin-hf:~/go/src/github.com/kongyixueyuan.com/bill/fixtures$ docker-compose down
Removing peer0.org1.example.com ... done
Removing peer1.org1.example.com ... done
Removing orderer.example.com ... done
Removing ca.org1.example.com ... done
Removing network fixtures_default
```