目录

一.链码概念

1.基本概念

Fabric的智能合约称为链码（chaincode），分为系统链码和用户链码。系统链码用来实现系统层面的功能，用户链码实现用户的应用功能。链码被编译成一个独立的应用程序，运行于隔离的Docker容器中。

和以太坊相比，Fabric链码和底层账本是分开的，升级链码时并不需要迁移账本数据到新链码当中，真正实现了逻辑与数据的分离，同时，链码采用Go、Java、Nodejs语言编写。

2.数据流向

Fabric链码通过gprc与peer节点交互

(1)当peer节点收到客户端请求的输入(propsal)后，会通过发送一个链码消息对象（带输入信息，调用者信息）给对应的链码。

(2)链码调用ChaincodeBase里面的invoke方法，通过发送获取数据（getState）和写入数据（putState）消息，向peer节点获取账本状态信息和发送预提交状态。

(3)链码发送最终输出结果给peer节点，节点对输入(propsal)和 输出(propsalreponse)进行背书签名，完成第一段签名提交。

(4)之后客户端收集所有peer节点的第一段提交信息，组装事务（transaction）并签名，发送事务到orderer节点排队，最终orderer产生区块，并发送到各个peer节点，把输入和输出落到账本上，完成第二段提交过程。

3.链码类型

3.1 用户链码

由应用开发人员使用Go(Java/JS)语言编写基于区块链分布式账本的状态及处理逻辑，运行在链码容器中, 通过Fabric提供的接口与账本平台进行交互

3.2系统链码

负责Fabric节点自身的处理逻辑, 包括系统配置、背书、校验等工作

系统链码仅支持Go语言, 在Peer节点启动时会自动完成注册和部署

系统链码共有五种类型

配置系统链码(CSCC) Configuration System Chaincode

-- 负责账本和链的配置管理

背书管理系统链码(ESCC) Endorsement System Chaincode

-- 负责背书(签名)过程, 并可以支持对背书策略进行管理

生命周期系统链码(LSCC) Lifecycle System Chaincode

-- 负责对用户链码的生命周期进行管理

-- 链码生命周期包括安装、部署、升级、权限管理、获取信息等环节.

查询系统链码(QSCC) Query System Chaincode

-- 负责提供账本和链的信息查询功能

验证系统链码(VSCC) Verification System Chaincode

交易提交前根据背书策略进行检查 验证过程:

(1) 首先解析出交易结构, 并对交易结构格式进行校验

(2) 检查交易的读集合中元素版本跟本地账本中版本一致

(3) 检查带有合法的背书信息(主要是检查签名信息)

(4) 通过则返回正确, 否则返回错误消息

二.链码操作

管理Chaincode的生命周期四个命令	
安装	install
实例化	instantiate
升级	upgrade
打包	package
签名	signpackage

> 未来还会支持 stop 和 start 命令, 来禁用和重新启用链代码

链代码成功安装和实例化后，链代码处于活动状态(正在运行)，可通过 invoke 命令调用处理事务

链代码可以在安装后随时升级

1.启动开发模式

(1) 下载链码API

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

参数说明:

go get: 根据要求和实际情况从互联网上下载或更新指定的代码包及其依赖包，并进行编译和安装

-u: 利用网络来更新已有代码包及其依赖包。默认情况下，该命令只会从网络上下 载本地不存在的代码包，而不会更新已有的代码包

go build: 加上可编译的go源文件可以得到一个可执行文件

(2) 进入开发模式

正常情况下chaincode由对等体启动和维护。然而，在“开发模式”下，链码由用户构建并启动 ,前提保证前期通过`docker-compose -f docker-compose-cli.yaml up -d`创建的网络处于关闭状态

```
cd /home/bruce/hyfa/fabric-samples/chaincode-docker-devmode
```

```
$ docker-compose  -f docker-compose-simple.yaml  up -d //启动开发环境
```

> 查看启动的容器

```
CONTAINER ID        IMAGE                          COMMAND
CREATED             STATUS              PORTS
             NAMES
1bf418435b45        hyperledger/fabric-ccenv       "/bin/bash -c 'sleep…"   6
seconds ago       Up 4 seconds
            chaincode
96aa47625135        hyperledger/fabric-tools       "/bin/bash -c ./scri…"   6
seconds ago       Up 4 seconds
            cli
e88a7faa98bc        hyperledger/fabric-peer        "peer node start --p…"   7
seconds ago       Up 6 seconds        0.0.0.0:7051->7051/tcp, 0.0.0.0:7053-
>7053/tcp   peer
e946e0472e7b        hyperledger/fabric-orderer     "orderer"                8
seconds ago       Up 7 seconds        0.0.0.0:7050->7050/tcp
            orderer
```

2.启动链码

```
docker exec -it chaincode bash  #进入chaincode容器
```

- 编译链码

```
root@1bf418435b45:/opt/gopath/src/chaincode# cd sacc/
root@1bf418435b45:/opt/gopath/src/chaincode/sacc# go build  //编译sacc.go
```

- 运行链码

```
CORE_PEER_ADDRESS=peer:7052 CORE_CHAINCODE_ID_NAME=devcc1:0 ./sacc
```

-- 链码名称：devcc1

-- 链码初始版本号： 0

3.安装并实例化

```
$ docker exec  -it cli  bash  #进入cli容器
```

- 安装链码

```
$ peer chaincode install -p chaincodedev/chaincode/sacc -n devcc1 -v 0
```

- 链码实例化

```
$ peer chaincode  instantiate -n devcc1 -v 0 -c '{"Args":
["brucefeng","100"]}' -C myc
```

- 查询bruce账户的余额

```
peer chaincode query -n devcc1 -c '{"Args":["query","brucefeng"]}' -C myc
```

命令返回

```
2018-07-13 09:22:12.753 UTC [msp/identity] Sign -> DEBU 044 Sign:
plaintext:
0ACB070A6308031A0C08C4DEA1DA0510...71756572790A09627275636566656E67
2018-07-13 09:22:12.753 UTC [msp/identity] Sign -> DEBU 045 Sign: digest:
3DBE7633D80A2F903926D8A72CDFF8F88594434892B661B2D0999A67F11571ED
100
```

- 调用链码，修改账户余额

```
$ peer chaincode invoke -n devcc1 -c '{"Args":["set","brucefeng","200"]}' -
C myc
```

命令返回

```
2018-07-13 09:24:18.538 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO
0a8 Chaincode invoke successful. result: status:200 payload:"200"
```

- 查询最新余额

```
peer chaincode query -n devcc1 -c '{"Args":["query","brucefeng"]}' -C myc
```

命令返回

```
2018-07-13 09:25:40.418 UTC [msp/identity] Sign -> DEBU 044 Sign:
plaintext:
0ACB070A6308031A0C0894E0A1DA0510...71756572790A09627275636566656E67
2018-07-13 09:25:40.418 UTC [msp/identity] Sign -> DEBU 045 Sign: digest:
EE8F9C3A592373E9DACB9BD69C08C46343A49EA3EA49DDD26545221642323C94
200
```

4.链码打包与签名

通过将链码相关数据进行封装, 可以实现对其进行打包和签名操作

- 链码打包

```
peer chaincode  package -n devcc1 -p chaincodedev/chaincode/sacc -v 0 -s -S
-i "AND('OrgA.admin')" devcc1.out
```

参数说明:

-s: 创建角色支持的CC部署规范包, 而不是原始的CC部署规范 -S: 如果创建CC部署规范方案角色支持,也与本地MSP签名 -i: 指定实例化策略

命令返回

```
2018-07-13 09:28:07.353 UTC [msp/identity] Sign -> DEBU 040 Sign:
plaintext:
0A2E080112280A1B636861696E636F64...4552544946494341544 52D2D2D2D2D0A
2018-07-13 09:28:07.353 UTC [msp/identity] Sign -> DEBU 041 Sign: digest:
337C515704E41942E87B914B8C06C5D7AC17A0FC9C4F9CD01D8A76602A6D1680
2018-07-13 09:28:07.353 UTC [chaincodeCmd] chaincodePackage -> DEBU 042
Packaged chaincode into deployment spec of size <2209>, with args =
[devcc1.out]
```

文件查看

```
root@96aa47625135:/opt/gopath/src/chaincodedev# ls -l  devcc1.out
-rwx------ 1 root root 2209 Jul 13 09:28 devcc1.out
```

- 签名

对一个打包文件进行签名操作(添加当前MSP签名到签名列表中)

```
peer chaincode signpackage devcc1.out signeddevcc1.out
```

命令返回

```
2018-07-13 09:36:07.115 UTC [msp/identity] Sign -> DEBU 037 Sign:
plaintext:
0A2E080112280A1B636861696E636F64...4552544946494341544 52D2D2D2D2D0A
2018-07-13 09:36:07.115 UTC [msp/identity] Sign -> DEBU 038 Sign: digest:
337C515704E41942E87B914B8C06C5D7AC17A0FC9C4F9CD01D8A76602A6D1680
Wrote signed package to signeddevcc1.out successfully
```

5.升级链码

- 重新启动链码(容器:chaincode)

> 参考2.启动链码的编译链码与启动链码

```
CORE_PEER_ADDRESS=peer:7052 CORE_CHAINCODE_ID_NAME=devcc1:1 ./sacc
```

-- 链码版本号修改为: 1

- 重新安装链码(容器:cli)

```
$ peer chaincode install -p chaincodedev/chaincode/sacc -n devcc1 -v 1
```

注意版本号必须一致

在对某链码代码升级前, 推荐先将所有该链码的容器停止, 并从Peer上备份并移除旧 链码部署文件. 之后先在个别Peer节点上部署新链码, 对原有数据进行测试, 成功后再在其它节点上进行升级操作

- 升级链码(容器:cli)

```
$ peer chaincode upgrade -n devcc1 -v 1 -c '{"Args":["bruce", "800"]}' -C
myc
```

命令返回

```
2018-07-13 09:47:12.397 UTC [msp/identity] Sign -> DEBU 0aa Sign: digest:
4D430EB0D48AD5235F21F39D473B9C18C6B199F41289E9DBEA24E6BFF18A2CC1
2018-07-13 09:47:12.397 UTC [chaincodeCmd] upgrade -> DEBU 0ab Get Signed
envelope
2018-07-13 09:47:12.397 UTC [chaincodeCmd] chaincodeUpgrade -> DEBU 0ac
Send signed envelope to orderer
```

- 查询用户余额

```
$ peer chaincode query -n devcc1 -c '{"Args":["query","bruce"]}' -C myc  #新
用户bruce
```

命令返回

```
2018-07-13 09:49:53.445 UTC [msp/identity] Sign -> DEBU 044 Sign:
plaintext:
0ACB070A6308031A0C08C1EBA1DA0510...1A0E0A0571756572790A056272756365
2018-07-13 09:49:53.445 UTC [msp/identity] Sign -> DEBU 045 Sign: digest:
229B5B1B788011E2C539D63122E2403E88432B9F8304018D607085EFFCF9188F
800
```

```
$peer chaincode query -n devcc1 -c '{"Args":["query","brucefeng"]}' -C myc
#老用户brucefeng
```

命令返回

```
2018-07-13 09:52:31.230 UTC [msp/identity] Sign -> DEBU 044 Sign:
plaintext:
0ACA070A6208031A0B08DFECA1DA0510...71756572790A09627275636566656E67
2018-07-13 09:52:31.230 UTC [msp/identity] Sign -> DEBU 045 Sign: digest:
F6D314D991E18ED26E8B62A1D9338FF37318951ABA777BC015FCFE6D53BB53E9
200
```