**该文档已经默认fabric1.1已经搭建完成**

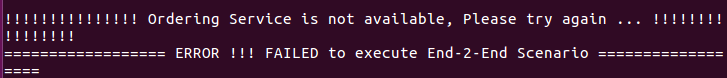
**建议**：对于任何文件的修改和删除都应该予以备份，以备日后遇错恢复，对于配置的修改在docker模式下必须先关闭fabric后进行修改，请将文档中的IP 通道名，合约等信息替换成当前环境和部署的名称。

对于fabric中的couchdb的启动需要配置fabric环境，

对于初始搭建的fabric不牵扯到本地IP的设置是可以进行迁移的

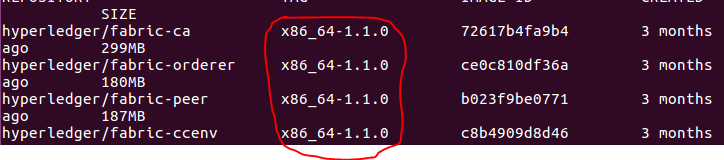
**fabric 启动后的可能的错误以及解决方案**：

error1：



需要查看docker的镜像

**docker images**



原因：镜像缺少latest标签

修改tag的命令：

**docker tag hyperledger/fabric-xxx:x86\_64-1.1.0 docker.io/hyperledger/fabric-xxx:latest**

将fabric-xxx替换成没有latest标签的镜像名称

error2：

Creating peer0.org2.example.com ... error

原因该容器以及被占用：

解决方案：停止该容器的运行，后删除或者重命名该容器

docker stop <name\_of\_container>

docker rm <name\_of\_container>

对于其他错误可以尝试

1、恢复原始配置（修改前可以运行的配置）

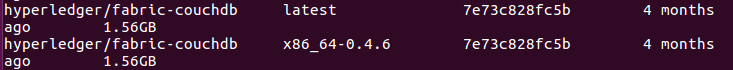
2、停止所有容器的运行

3、删除所有容器

4、重新启动fabric

**Fabric CouchDB配置**

**前提**：在fabric中需要拥有couchdb的镜像



首先修改docker-compose-cli.yaml文件

**cd $GOPATH/src/github.com/hyperledger/fabric/examples/e2e\_cli**

修改docker-compose-cli.yaml 往里面添加couchdb容器（里面IP换成本地主机的IP）

#couchdb container

couchdb0:

container\_name: couchdb0

image: hyperledger/fabric-couchdb

ports:

- "5984:5984"

environment:

- CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB

- CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=172.16.1.135:5984

couchdb1:

container\_name: couchdb1

image: hyperledger/fabric-couchdb

ports:

- "6984:5984"

environment:

- CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB

- CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=172.16.1.135:5984

couchdb2:

container\_name: couchdb2

image: hyperledger/fabric-couchdb

ports:

- "7984:5984"

environment:

- CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB

- CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=172.16.1.135:5984

couchdb3:

container\_name: couchdb3

image: hyperledger/fabric-couchdb

ports:

- "8984:5984"

environment:

- CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB

- CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=172.16.1.135:5984

说明:（以couchdb0为例）

couchdb0:

container\_name: couchdb0 #容器名称，对应于fabric启动时生成的容器名称不可以重复

image: hyperledger/fabric-couchdb #对应于docker images中的第一列REPOSITORY

ports:

- "5984:5984" #couchdb的访问端口通常为5984

environment:

- CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB

- CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=172.16.1.135:5984 #couchdb的访问地址，如果couchdb在有设置用户名和密码还需添加

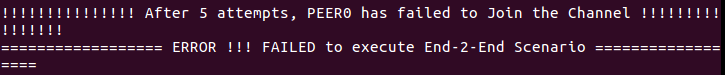
- CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=对应的用户名   
- CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=对应的密码

浏览器访问：

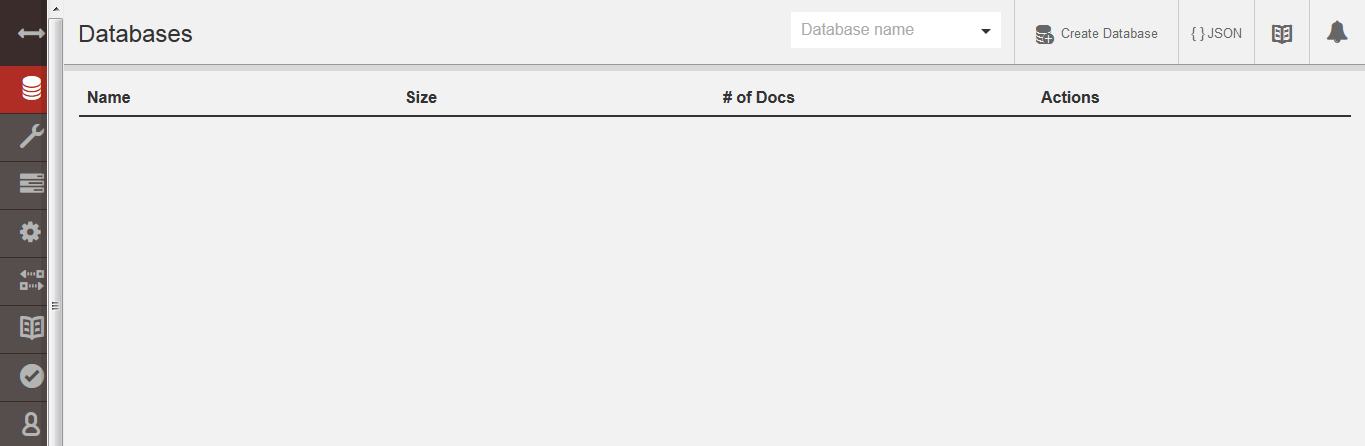
**http://172.16.1.135:5984/\_utils**

**这步在完成迁移后必须检查，因为docker 容器起来的时候并不检查对于couchdb地址的连接性**

**couchdb如果加入到peer下，启动时候会检查couchdb地址的连通性**

****

**此时需要删除容器，重新启动**



再在docker-compose-cli.yaml中的peer0.org1.example.com容器添加（红色字体部分）

peer0.org1.example.com:

container\_name: peer0.org1.example.com

**environment:**

**- CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB**

**- CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=172.16.1.135:5984**

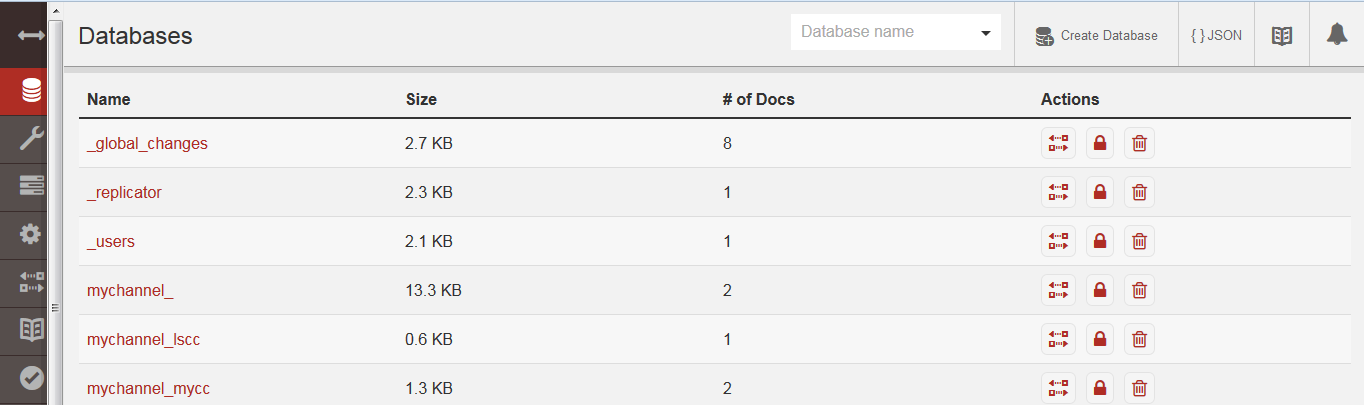
extends:

file: base/docker-compose-base.yaml

service: peer0.org1.example.com

重启fabric

刷新http://172.16.1.135:5984/\_utils



Note：**CouchDB需要将其添加到所有peer中去，这样才可以在合约编译运行时达成对于数据的更新。**

---------------------------------------------------------------------------------------------------------------------------

**手动拉取外部couchdb的操作演示（此操作可以跳过）**：

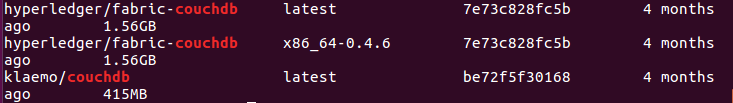
couchdb并不是fabric所必须的组件，可以从外部手动拉取couchdb镜像后生成容器

手动拉取外部镜像

**$ docker pull klaemo/couchdb**

查看docker镜像

**$ docker images | grep couchdb**



创建一个couchdb容器（实例）

**$ docker run -p 5984:5984 -d --name my-couchdb -e COUCHDB\_USER=admin -e COUCHDB\_PASSWORD=password -v ~/couchdb:/opt/couchdb/data klaemo/couchdb**

查看容器：

**$ docker ps -a**

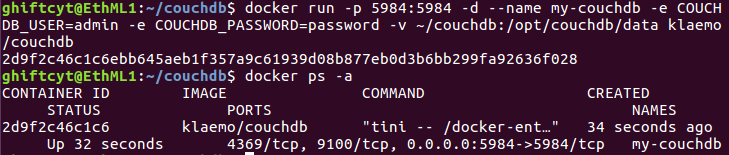
其中 -p：为couchdb的开放端口号

--name：couchdb容器的名称

COUCHDB\_USER：为couchdb数据库建立用户名

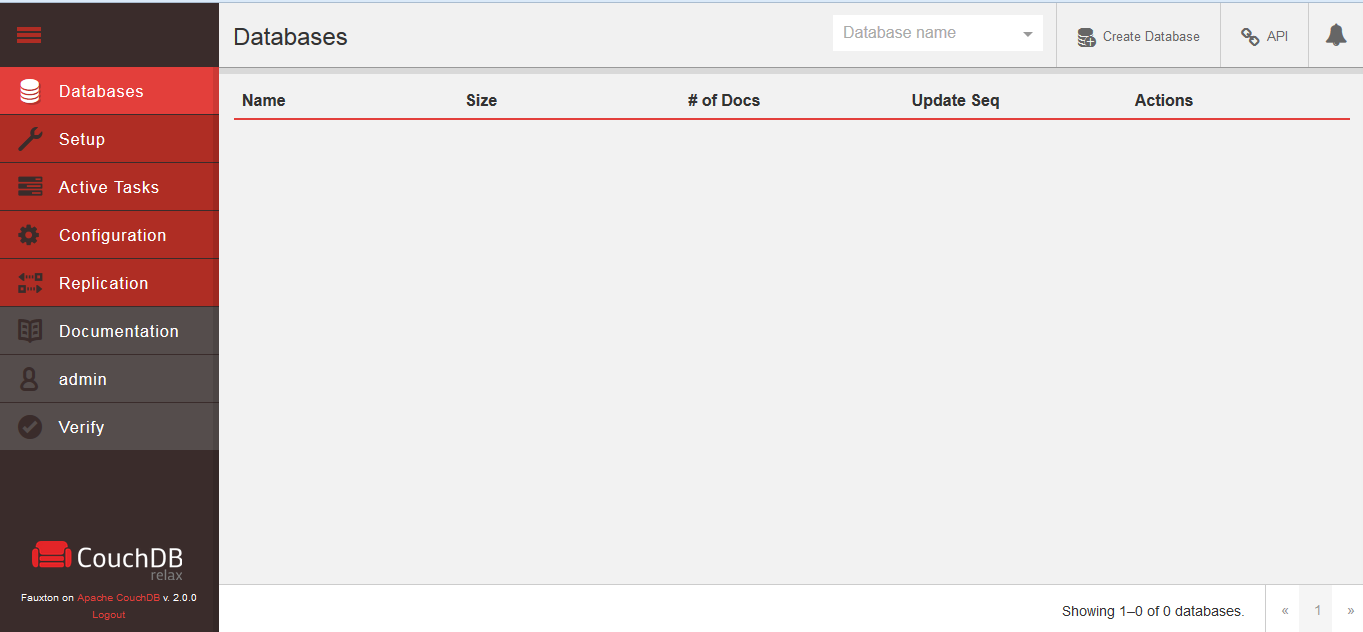
COUCHDB\_PASSWORD：为couchdb数据库建立用户对应的密码

klaemo/couchdb ：couchdb容器对应的镜像

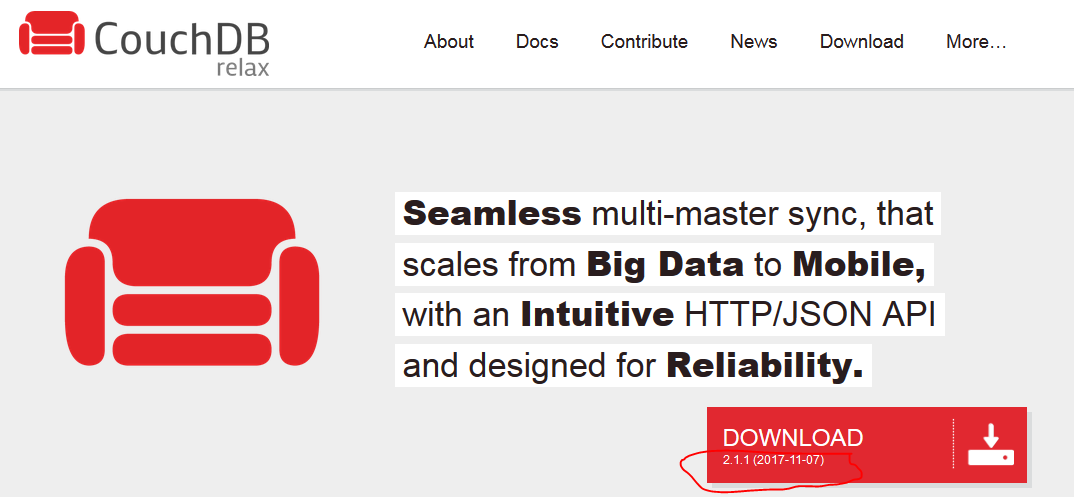


登录couchdb的web页面检查

**http://172.16.1.136:5984/\_utils**



注意：通过**klaemo/couchdb 拉取的官方image最后生成的版本为2.0.0，fabric-couchdb的版本为2.1.1与官方最新版本保持一致（截止2018年6月29日）**



由于4个Peer+1Orderer的模式，所以我们先创建4个CouchDB数据库

**$ cd**

**$ mkdir couchdb0**

**$ mkdir couchdb1**

**$ mkdir couchdb2**

**$ mkdir couchdb3**

**$ docker run -p 5984:5984 -d --name couchdb0 -e COUCHDB\_USER=admin -e COUCHDB\_PASSWORD=password -v ~/couchdb0:/opt/couchdb/data klaemo/couchdb**

**$ docker run -p 6984:5984 -d --name couchdb1 -e COUCHDB\_USER=admin -e COUCHDB\_PASSWORD=password -v ~/couchdb1:/opt/couchdb/data klaemo/couchdb**

**$ docker run -p 7984:5984 -d --name couchdb2 -e COUCHDB\_USER=admin -e COUCHDB\_PASSWORD=password -v ~/couchdb2:/opt/couchdb/data klaemo/couchdb**

**$ docker run -p 8984:5984 -d --name couchdb3 -e COUCHDB\_USER=admin -e COUCHDB\_PASSWORD=password -v ~/couchdb3:/opt/couchdb/data klaemo/couchdb**

官方已经提供了多个Docker-compose文件，如果我们使用的是./network\_setup.sh up命令，那么启用的就是docker-compose-cli.yaml这个文件。如果要基于这个yaml文件启用CouchDB的Peer，则打开该文件，并编辑其中的Peer节点，改为如下的形式（红色部分为添加部分）：

peer0.org1.example.com:

container\_name: peer0.org1.example.com

**environment:**

**- CORE\_LEDGER\_STATE\_STATEDATABASE=CouchDB**

**- CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_COUCHDBADDRESS=172.16.1.136:5984**

**- CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_USERNAME=admin**

**- CORE\_LEDGER\_STATE\_COUCHDBCONFIG\_PASSWORD=password**

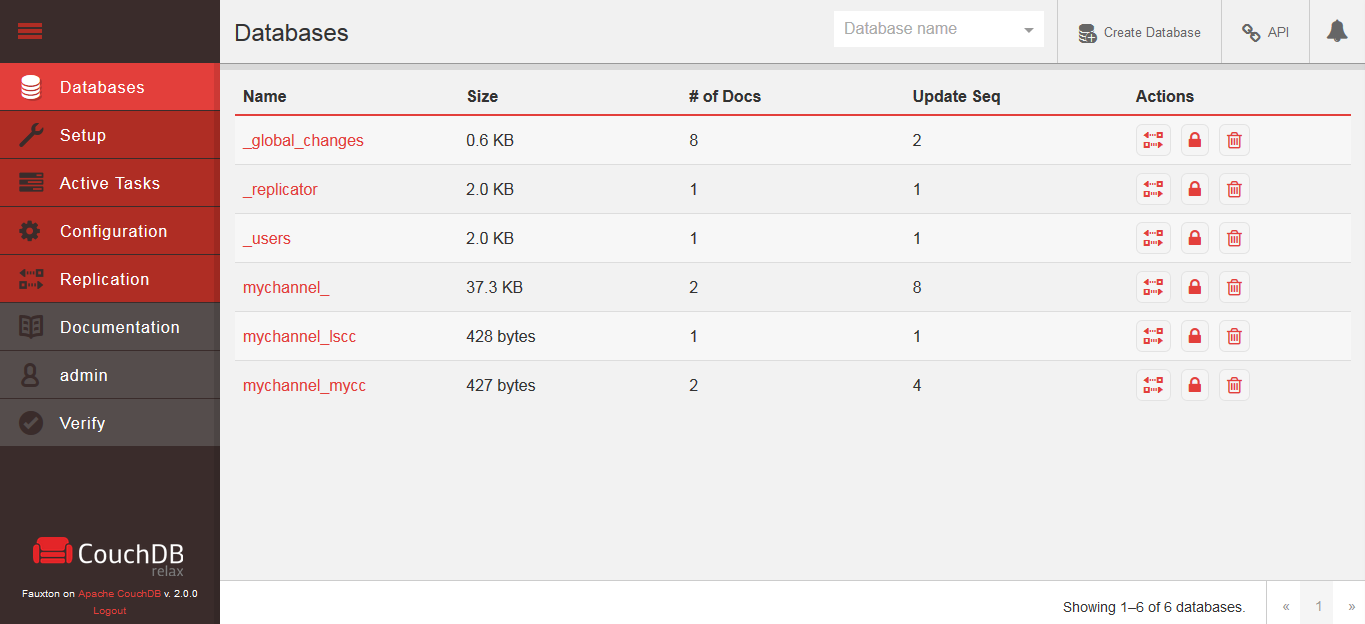
extends:

file: base/docker-compose-base.yaml

service: peer0.org1.example.com

登录：

**http://172.16.1.136:5984/\_utils**



---------------------------------------------------------------------------------------------------------------------

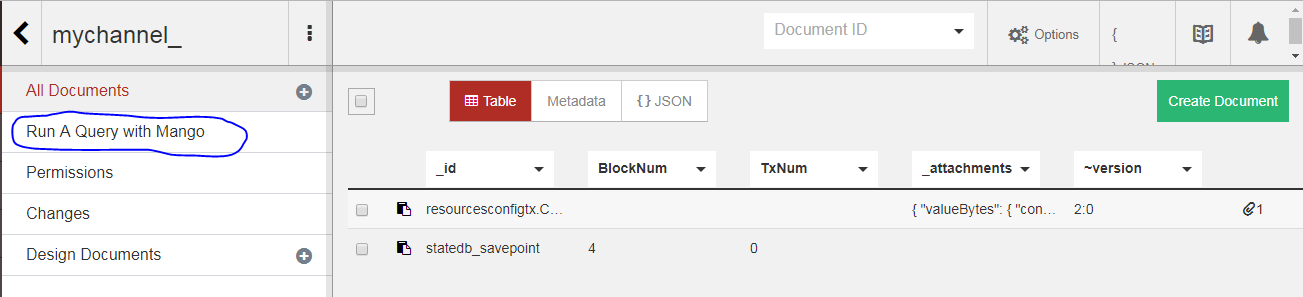
**fabric-couchdb数据库的增删改查**：

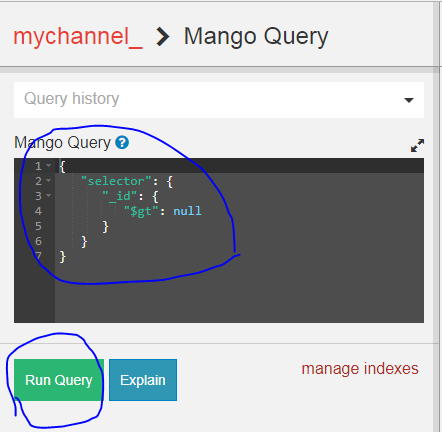
**（1）查询过程**：

鼠标点入一个数据库如[mychannel\_](http://172.16.1.136:5984/_utils/database/mychannel_/_all_docs)

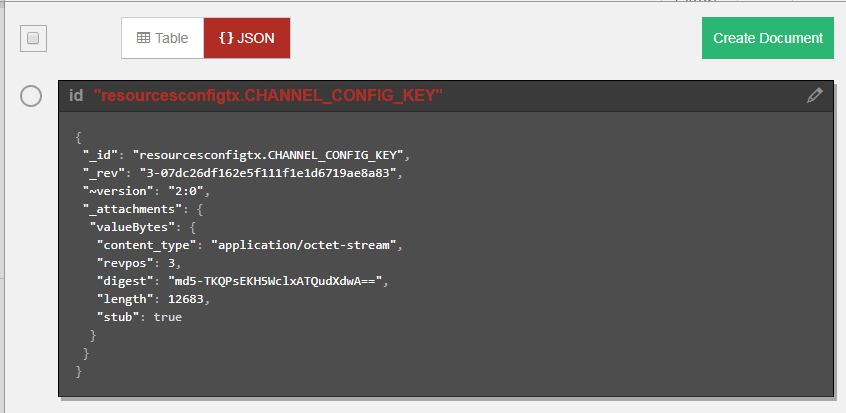


点击Run A Query with Mango可以进行查询



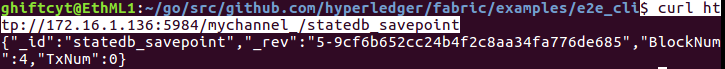


查询结果：可以以Table形式或者JSON形式



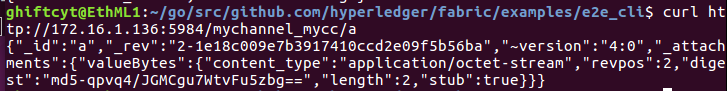
**在Ubuntu上查询**

**$ curl http://172.16.1.136:5984/mychannel\_/statedb\_savepoint**



**业务数据查询**：

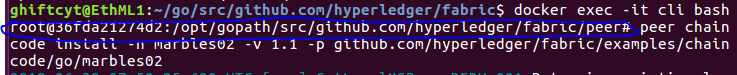
**$ curl http://172.16.1.136:5984/mychannel\_mycc/a**

**（2）对chouchdb的增加、修改、删除**：

为了完整验证CouchDB的功能，使用**marbles02**示例来做演示.。marbles02 合约放置在 fabric/examples/chaincode/go 目录.

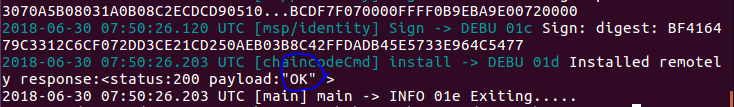
1/进入cli bash

**$ docker exec -it cli bash**

****

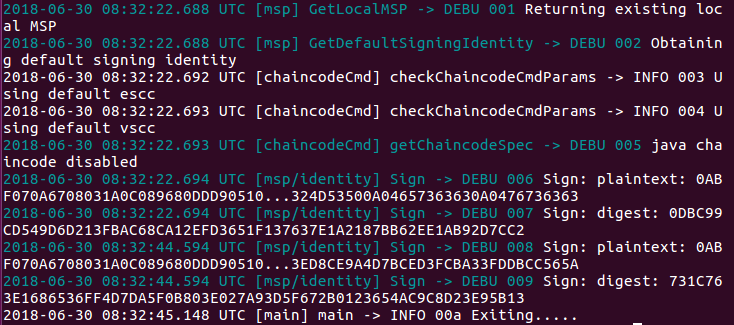
2/安装链码

**# peer chaincode install -n marbles02 -v 1.1 -p github.com/hyperledger/fabric/examples/chaincode/go/marbles02**



3/初始化数据

**# peer chaincode instantiate -o orderer.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n marbles02 -v 1.1 -c '{"Args":["init"]}' -P "OR ('Org1MSP.member','Org2MSP.member')"**

****

可能产生的错误——证书未知



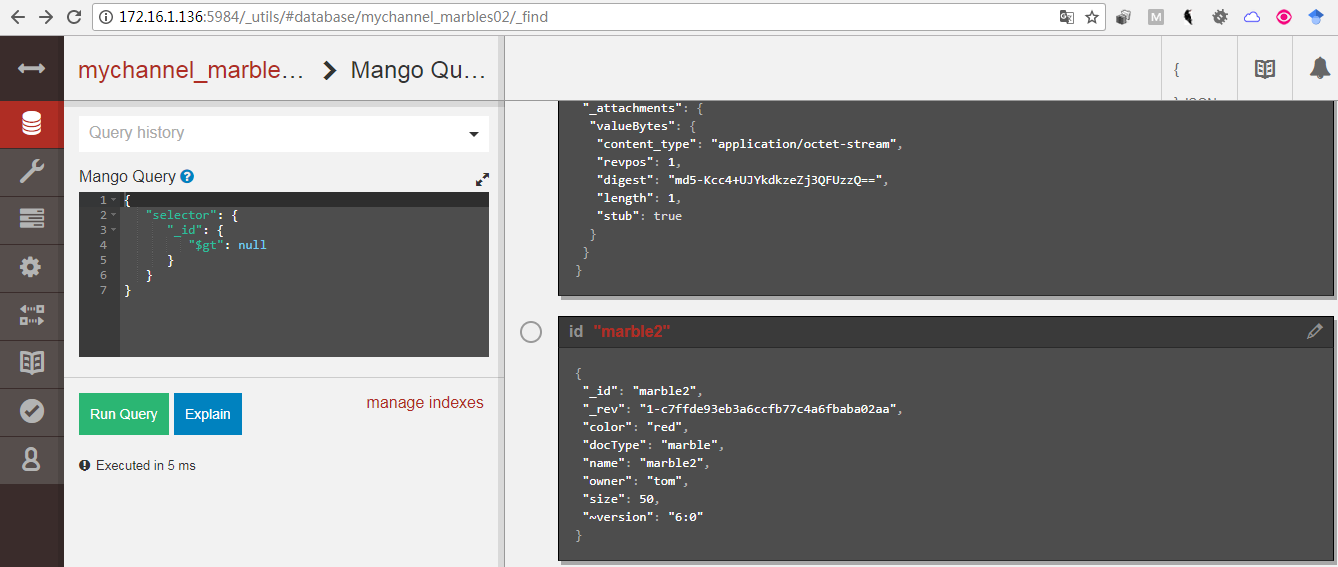
需要替换证书：修改/ cacerts/ ca.example. com-cert.pem 下证书路径为/tlscacerts/tlsca.example.com-cert.pem，可成功

4/创建弹珠并移动它们：

**# peer chaincode invoke -o orderer.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n marbles02 -c '{"Args":["initMarble","marble2","red","50","tom"]}'**



**通过couchdb的web查询**



**命令行查询**：

**$ curl http://172.16.1.136:5984/mychannel\_marbles02/marble2**



**Cli容器查询**：

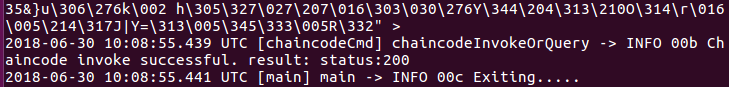
**# peer chaincode query -C mychannel -n marbles02 -c '{"Args":["readMarble","marble2"]}'**



**转账操作：**

1/添加弹珠（不是必须，是为了转账测试所用）

**# peer chaincode invoke -o orderer.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n marbles02 -c '{"Args":["initMarble","marble1","blue","35","tom"]}'**



**# peer chaincode invoke -o orderer.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n marbles02 -c '{"Args":["initMarble","marble3","blue","70","tom"]}'**



此时我们可以看到web 页面中有三个marble



2/进行转账（将marble1由tom转给jerry）

**# peer chaincode invoke -o orderer.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n marbles02 -c '{"Args":["transferMarble","marble1","jerry"]}'**

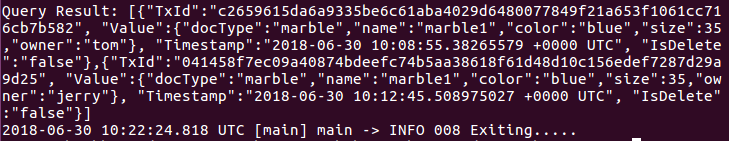


可以看到owner已经转成jerry

3/查看marble1的历史数据

**# peer chaincode query -C mychannel -n marbles02 -c '{"Args":["getHistoryForMarble","marble1"]}'**

查询结果：



4/从web页面中修改数据（将marble2的color: red，size: 50 改成color: green size: 10）



通过CLi查询

**# peer chaincode query -C mychannel -n marbles02 -c '{"Args":["readMarble","marble2"]}'**



5/在web页面进行修改后查看是否能转账成功（将marble2转给jerry）：

**# peer chaincode invoke -o orderer.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n marbles02 -c '{"Args":["transferMarble","marble2","jerry"]}'**



CLi查询：

**# peer chaincode query -C mychannel -n marbles02 -c '{"Args":["readMarble","marble2"]}'**

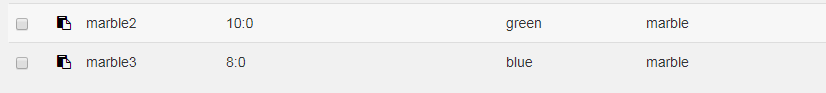


我们可以看到在web页面数据修改后，转账是成功的。

**可能的问题1**：我们对CouchDB数据库的更改都是有效的，在Fabric看来似乎并不知道我们改了CouchDB的内容？？

6/删除marble1

**# peer chaincode invoke -o orderer.example.com:7050 --tls $CORE\_PEER\_TLS\_ENABLED --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n marbles02 -c '{"Args":["delete","marble1"]}'**



这边可以看到marble1已经被删除了

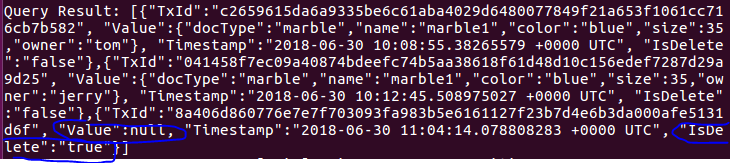
通过CLI查看

**# peer chaincode query -C mychannel -n marbles02 -c '{"Args":["readMarble","marble1"]}'**



但是仍可以查看marble1的历史数据：

**# peer chaincode query -C mychannel -n marbles02 -c '{"Args":["getHistoryForMarble","marble1"]}'**



可以看到marble1已经被删除了

7/对数据进行rich查询

**# peer chaincode query -C mychannel -n marbles02 -c '{"Args":["queryMarblesByOwner","jerry"]}'**

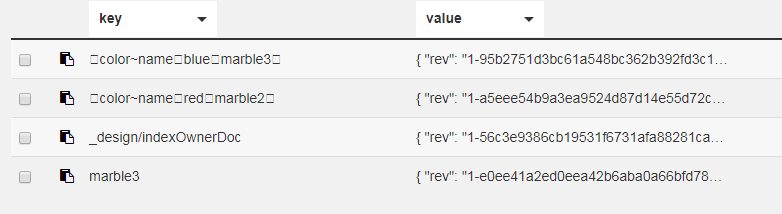


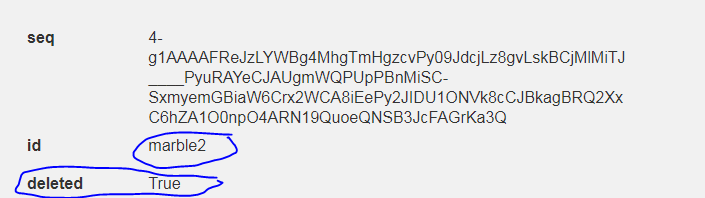
8/按字段owner、值为jerry查询marble信息

**# peer chaincode query -C mychannel -n marbles02 -c '{"Args":["queryMarbles","{\"selector\":{\"owner\":\"jerry\"}}"]}'**

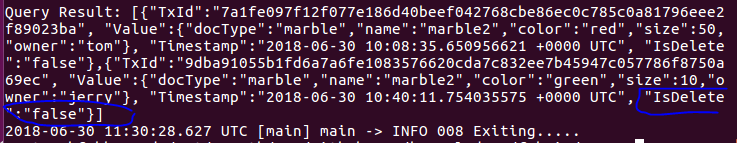


可能的问题2：在web界面删除了marble2





而在CLI查看marble2的历史数据：



但是在CLI查询marble当前状态时

