네트워크 구축 실습2

Org1 에 peer를 추가하여 총 2개 peer를 가진 네트워크 구축
couchdb 사용 (기본값은 leveldb)
전체 스크립트는 basic-network2.tar 참조

1. 네트워크 개요 정리

Organization수: 1

Channel

채널수: 1

채널이름: mychannel

Orderer

Orderer수 : 1 Concensus 방식: solo

주소 및 포트: orderer.example.com:7050

Ca

Ca수 : 1

주소 및 포트: ca.example.com:7054

Peer

Organization 별 peer수:

Org1 : 2

주소 및 포트:

Org1 : peer0.org1.example.com:7051

peer1.org1.example.com:8051

Cli

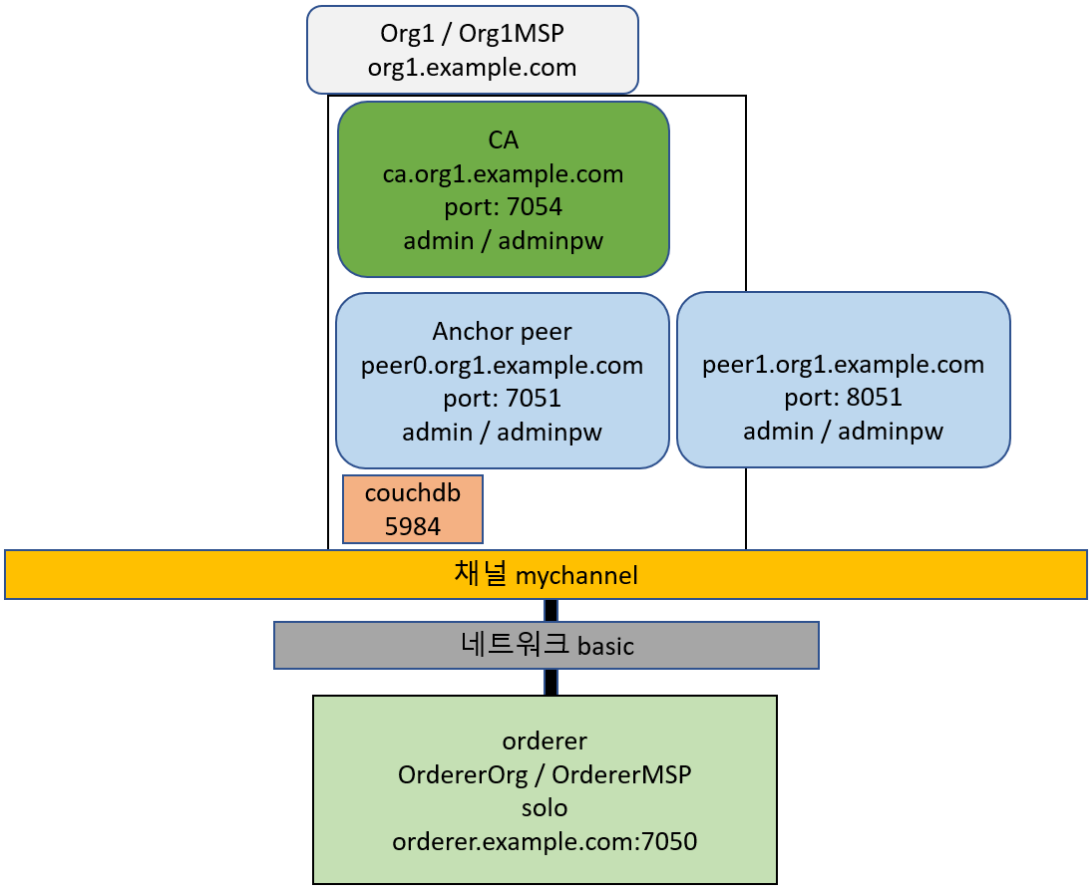
주소 및 포트:

Org1 : cli.example.com

counchdb

주소 및 포트: couchdb : 5984

2. 네트워크 스펙 정리



3. 네트워크 작성하기

1) basic-network을 basic-network2 로 복사한다.

cp -r basic-network basic-network2
cd basic-network2

2) configtx.yaml 수정

수정 사항 없음

3)crypto-config.yaml 수정

Org1 -> Template -> Count : 1 => 2

4) generate.sh 수정

상단에 추가 #향후 채널 추가에 대비하여 변수로 지정

CHANNEL_NAME=mychannel

수정

```
# generate channel configuration transaction
configtxgen -profile OneOrgChannel -outputCreateChannelTx ./config/"$CHANNEL_NAME".tx -channelID $CHANNEL_NAME
if [ "$?" -ne 0 ]; then
  echo "Failed to generate channel configuration transaction..."
  exit 1
fi
```

하단에 추가 #peer가 2개 이상이 되면 1개가 anchor peer가 되어야 하므로 설정 필요

```
# generate anchor peer transaction
configtxgen -profile OneOrgChannel -outputAnchorPeersUpdate ./config/Org1MSPanchors.tx -channelID $CHANNEL_NAME -asOrg Org1MSP
if [ "$?" -ne 0 ]; then
  echo "Failed to generate anchor peer update for Org1MSP..."
  exit 1
```

```
fi
```

5)실행 ./generate.sh

config 와 crypto-config 폴더 생성 확인, tree 명령으로 peer1 관련 폴더 생성 확인

6)docker-compose.yaml 수정

a. ca의 FABRIC_CA_SERVER_CA_KEYFILE 값 변경 - generate.sh 실행하면 crypto-config 이 변경됨

crypto-config/peerOrganizations/org1.example.com/ca 폴더에서 _sk 로 끝나는 파일명 으로 대체

```
  ca.example.com:
    image: hyperledger/fabric-ca
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca.example.com
      - FABRIC_CA_SERVER_CA_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org1.example.com-cert.pem
      - FABRIC_CA_SERVER_CA_KEYFILE=/etc/hyperledger/fabric-ca-server-
config/8e2c0651e3d27fec24ec10773b2ee58fca161ffaeac0354dfd9abc07e75e5574_sk
```

b. peer0.org1.example.com: 단락의 내용을 복사하여 수정

각 peer마다 port 번호 다르게 변경 , CORE_PEER_ADDRESS=0.0.0.0:8051 형식으로 port에서 지정한 대로 설정

```
  peer0.org1.example.com:
    container_name: peer0.org1.example.com
    image: hyperledger/fabric-peer
    environment:
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - CORE_PEER_ID=peer0.org1.example.com
      - FABRIC_LOGGING_SPEC=info
      - CORE_CHAINCODE_LOGGING_LEVEL=info
      - CORE_PEER_LOCALMSPID=Org1MSP
      # - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/peer/
      - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp
      - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
      - CORE_PEER_LISTENADDRESS=0.0.0.0:7051
      # the following setting starts chaincode containers on the same
      # bridge network as the peers
      # https://docs.docker.com/compose/networking/
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=${COMPOSE_PROJECT_NAME}_basic
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb:5984
      # The CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME and CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD
      # provide the credentials for ledger to connect to CouchDB.  The username and password must
      # match the username and password set for the associated CouchDB.
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric
    command: peer node start
    # command: peer node start --peer-chaincodedev=true
    ports:
      - 7051:7051
      - 7053:7053
    volumes:
        - /var/run/:/host/var/run/
        - ./crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp:/etc/hyperledger/msp/peer
        - ./crypto-config/peerOrganizations/org1.example.com/users:/etc/hyperledger/msp/users
        - ./config:/etc/hyperledger/configtx
    depends_on:
```

```
    - orderer.example.com
    - couchdb
  networks:
    - basic

peer1.org1.example.com:
  container_name: peer1.org1.example.com
  image: hyperledger/fabric-peer
  environment:
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - CORE_PEER_ID=peer1.org1.example.com
    - FABRIC_LOGGING_SPEC=info
    - CORE_CHAINCODE_LOGGING_LEVEL=info
    - CORE_PEER_LOCALMSPID=Org1MSP
    # - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/peer/
    - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp
    - CORE_PEER_ADDRESS=peer1.org1.example.com:8051
    - CORE_PEER_LISTENADDRESS=0.0.0.0:8051
    # # the following setting starts chaincode containers on the same
    # # bridge network as the peers
    # # https://docs.docker.com/compose/networking/
    - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=${COMPOSE_PROJECT_NAME}_basic
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb:5984
    # The CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME and CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD
    # provide the credentials for ledger to connect to CouchDB.  The username and password must
    # match the username and password set for the associated CouchDB.
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric
  command: peer node start
  # command: peer node start --peer-chaincodedev=true
  ports:
    - 8051:8051
    - 8053:8053
  volumes:
      - /var/run/:/host/var/run/
      - ./crypto-config/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/msp:/etc/hyperledger/msp/peer
      - ./crypto-config/peerOrganizations/org1.example.com/users:/etc/hyperledger/msp/users
      - ./config:/etc/hyperledger/configtx
  depends_on:
    - orderer.example.com
    - couchdb
  networks:
    - basic
```

7) start.sh 수정

상단에 추가 #향후 채널 추가에 대비하여 변수로 지정

CHANNEL_NAME=mychannel

상단 수정 - peer1.org1.example.com 추가

```
docker-compose -f docker-compose.yml up -d ca.example.com orderer.example.com peer0.org1.example.com peer1.org1.example.com couchdb
```

Org에 peer가 2개 이상이므로 peer0를 anker peer로 지정

```
# update  mychannel1
docker exec  -e "CORE_PEER_LOCALMSPID=Org1MSP" -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp"
peer0.org1.example.com peer channel update -o orderer.example.com:7050 -c "$CHANNEL_NAME" -f /etc/hyperledger/configtx/Org1MSPanchors.tx
```

블록체인을 fetch

```
#fetch
```

```
docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp"
peer1.org1.example.com peer channel fetch 0 "$CHANNEL_NAME".block --channelID "$CHANNEL_NAME" --orderer orderer.example.com:7050
```

peer1.org1.example.com 을 mychannel에 join

```
# Join peer1.org1.example.com to the channel.
docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp"
peer1.org1.example.com peer channel join -b "$CHANNEL_NAME".block
```

8) 컨테이너가 모두 잘 실행되었는지 확인 - 위의 docker ps -a 결과 확인

```
ca.example.com orderer.example.com peer0.org1.example.com peer1.org1.example.com couchdb
```

9) 피어가 채널에 조인되어 있는지 확인 / 피어 노드가 실행되고 있는지 확인

```
docker exec peer0.org1.example.com peer channel list
docker exec peer1.org1.example.com peer channel list
docker exec peer0.org1.example.com peer node status
docker exec peer1.org1.example.com peer node status
```

```
bstudent@block-VM:~/fabric-samples/basic-network2$ docker exec peer0.org1.example.com peer channel list
2019-06-20 01:39:43.606 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
Channels peers has joined:
mychannel
bstudent@block-VM:~/fabric-samples/basic-network2$ docker exec peer1.org1.example.com peer channel list
2019-06-20 01:39:44.300 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
Channels peers has joined:
mychannel
bstudent@block-VM:~/fabric-samples/basic-network2$ docker exec peer0.org1.example.com peer node status
status:STARTED
bstudent@block-VM:~/fabric-samples/basic-network2$ docker exec peer1.org1.example.com peer node status
status:STARTED
```

가입된 채널(mychannel)을 확인할 수 있고, 각 피어의 상태를 알 수 있다.(STARTED가 정상임)

10) 체인코드 설치 및 실행

chaincode = sacc : chaincode install & instantiate & invoke & query

cli 에서 sacc 체인코드 설치-> peer0.org1.example.com

cli 에서 sacc 체인코드 인스턴스화

peer0.org1.example.com 에서 query 로 a 값 읽어오기 15

peer0.org1.example.com 에서 invoke 로 a 값 변경하기 => 130

peer0.org1.example.com 에서 query 로 a 값 다시 읽어오기 130

cli 에서 sacc 체인코드 설치-> peer1.org1.example.com

#peer1.org1.example.com 에서도 동일한 값을 읽어올 수 있어야 하고, 값도 변경할 수 있어야 한다.

peer1.org1.example.com 에서 query 로 a 값 읽어오기 130

peer1.org1.example.com 에서 invoke 로 a 값 변경하기 =>150

peer1.org1.example.com 에서 query 로 a 값 다시 읽어오기 150

peer0.org1.example.com 에서 query 로 a 값 다시 읽어오기 150

cc_start_sacc.sh

```
#!/bin/bash
# Exit on first error
set -e
starttime=$(date +%s)
CHANNEL_NAME=mychannel
CC_RUNTIME_LANGUAGE=golang
CC_SRC_PATH=github.com/sacc
CC_NAME=sacc
```

```
CC_VERSION=1.0

docker-compose -f ./docker-compose.yml up -d cli
docker ps -a

#install chaincode to peer0.org1.example.com - 각 endoser peer에 모두 설치
docker exec  cli peer chaincode install -n "$CC_NAME" -v "$CC_VERSION" -p "$CC_SRC_PATH" -l "$CC_RUNTIME_LANGUAGE"
#instantiate chaincode - 채널 당 한번만 실행
# 인스턴스 생성 docker ps -a 해보면
# dev-peer0.org1.example.com-sacc-1.0-xxxx 식의 컨테이너 생성됨
docker exec  cli peer chaincode instantiate -o orderer.example.com:7050 -C "$CHANNEL_NAME" -n "$CC_NAME" -l "$CC_RUNTIME_LANGUAGE" -v "$CC_VERSION" -c '{"Args":["a","15"]}' -P "OR ('Org1MSP.member')"
sleep 5
# cli로 실행하면 현재 peer0.org1.example.com 이 CORE_PEER_ADDRESS로 설정되어
# peer0.org1.example.com에서 실행한 것과 동일함
# docker exec  cli peer chaincode invoke -o orderer.example.com:7050 -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'
# docker exec  cli peer chaincode invoke -o orderer.example.com:7050 -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["set","a","110"]}'
# sleep 5
# docker exec  cli peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'

docker exec  peer0.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'
docker exec  peer0.org1.example.com peer chaincode invoke  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["set","a","130"]}'
sleep 5
docker exec  peer0.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'

#install chaincode to peer1.org1.example.com - 각 endoser peer에 모두 설치
docker exec -e CORE_PEER_ADDRESS=peer1.org1.example.com:8051  cli peer chaincode install -n "$CC_NAME" -v "$CC_VERSION" -p "$CC_SRC_PATH" -l "$CC_RUNTIME_LANGUAGE"
# endoser peer에서 처음 query 수행하면 인스턴스 생성됨 docker ps -a 해보면
# dev-peer1.org1.example.com-sacc-1.0-xxxx 식의 컨테이너 생성됨
docker exec  peer1.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'
docker exec  peer1.org1.example.com peer chaincode invoke  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["set","a","150"]}'
sleep 5
docker exec  peer1.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'
docker exec  peer0.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'
cat <<EOF
Total setup execution time : $(($(date +%s) - starttime)) secs ...
EOF
```

11)체인코드 설치 및 실행

chaincode = example02 : chaincode install & instantiate & invoke & query

cli 에서 example02 체인코드 설치 -> peer0.org1.example.com

cli 에서 example02 체인코드 인스턴스화

peer0.org1.example.com 에서 query 로 a,b 값 읽어오기 100 200

peer0.org1.example.com 에서 invoke 로 a,b 값 변경하기 =>a,b,10

peer0.org1.example.com 에서 query 로 a,b 값 다시 읽어오기 90 210

cli 에서 eample02 체인코드 설치 -> peer1.org1.example.com

#peer1.org1.example.com 에서도 동일한 값을 읽어올 수 있어야 하고, 값도 변경할 수 있어야 한다.

peer1.org1.example.com 에서 query 로 a,b 값 읽어오기 90 210

peer1.org1.example.com 에서 invoke 로 a,b 값 변경하기 => a,b,5

peer1.org1.example.com 에서 query 로 a,b 값 다시 읽어오기 85, 215

peer0.org1.example.com 에서 query 로 a,b 값 다시 읽어오기 85, 215

cc_start.example02.sh

```
#!/bin/bash
# Exit on first error
set -e
starttime=$(date +%s)
CHANNEL_NAME=mychannel
CC_RUNTIME_LANGUAGE=golang
CC_SRC_PATH=github.com/chaincode_example02/go
CC_NAME=example02
CC_VERSION=1.0

docker-compose -f ./docker-compose.yml up -d cli
docker ps -a

#install chaincode to peer0.org1.example.com - 각 endoser peer에 모두 설치
docker exec  cli peer chaincode install -n "$CC_NAME" -v "$CC_VERSION" -p "$CC_SRC_PATH" -l "$CC_RUNTIME_LANGUAGE"
#instantiate chaincode - 채널 당 한번만 실행
# 인스턴스 생성 docker ps -a 해보면
# dev-peer0.org1.example.com-example02-1.0-xxxx 식의 컨테이너 생성됨
docker exec  cli peer chaincode instantiate -o orderer.example.com:7050 -C "$CHANNEL_NAME" -n "$CC_NAME" -l "$CC_RUNTIME_LANGUAGE" -v
"$CC_VERSION" -c '{"Args":["init","a","100","b","200"]}' -P "OR ('Org1MSP.member')"
sleep 5

docker exec  peer0.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
docker exec  peer0.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'
docker exec  peer0.org1.example.com peer chaincode invoke  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["invoke","a","b","10"]}'
sleep 5
docker exec  peer0.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
docker exec  peer0.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'

#install chaincode to peer1.org1.example.com - 각 endoser peer에 모두 설치
docker exec -e CORE_PEER_ADDRESS=peer1.org1.example.com:8051  cli peer chaincode install -n "$CC_NAME" -v 1.0 -p "$CC_SRC_PATH" -l
"$CC_RUNTIME_LANGUAGE"
# endoser peer에서 처음 query 수행하면 인스턴스 생성됨 docker ps -a 해보면
# dev-peer1.org1.example.com-example02-1.0-xxxx 식의 컨테이너 생성됨
docker exec  peer1.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
docker exec  peer1.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'
docker exec  peer1.org1.example.com peer chaincode invoke  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["invoke","a","b","5"]}'
sleep 5
docker exec  peer1.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
docker exec  peer1.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'

docker exec  peer0.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
docker exec  peer0.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'

cat <<EOF
Total setup execution time : $(($(date +%s) - starttime)) secs ...
EOF
```

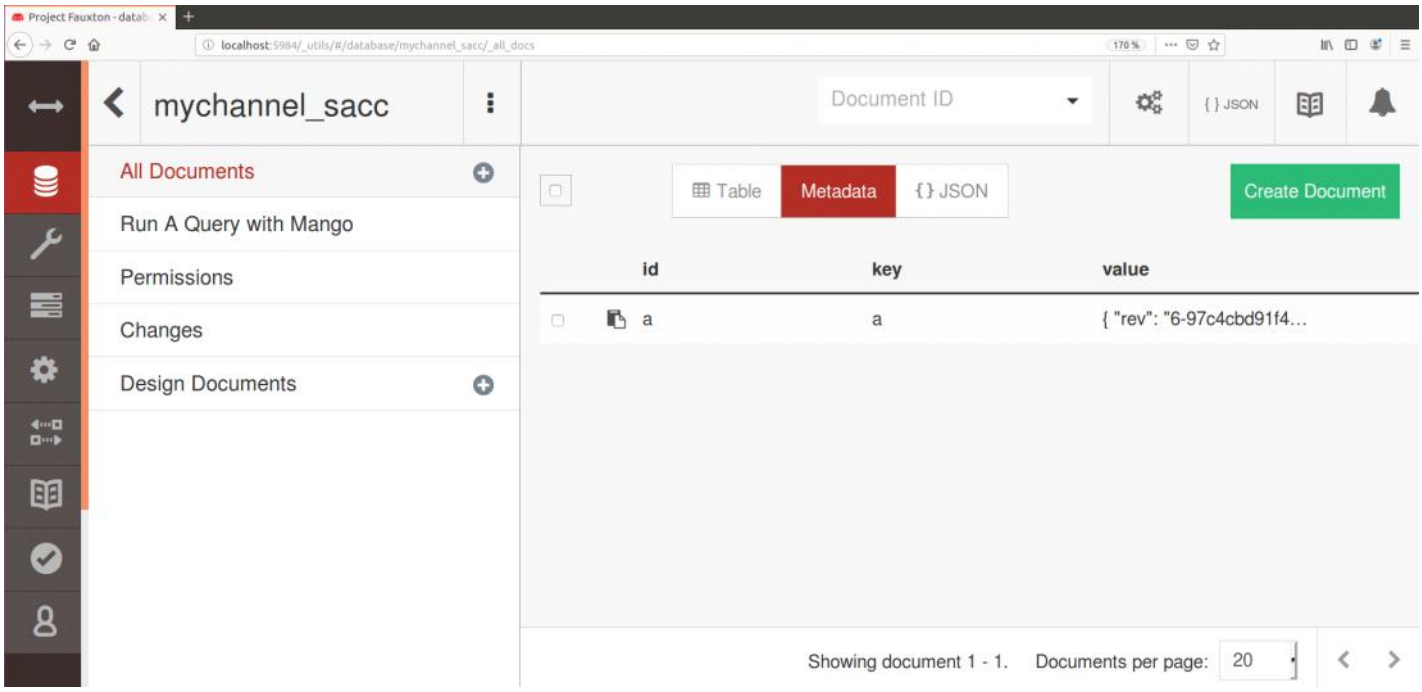
12)couchdb 접속해서 확인해보기

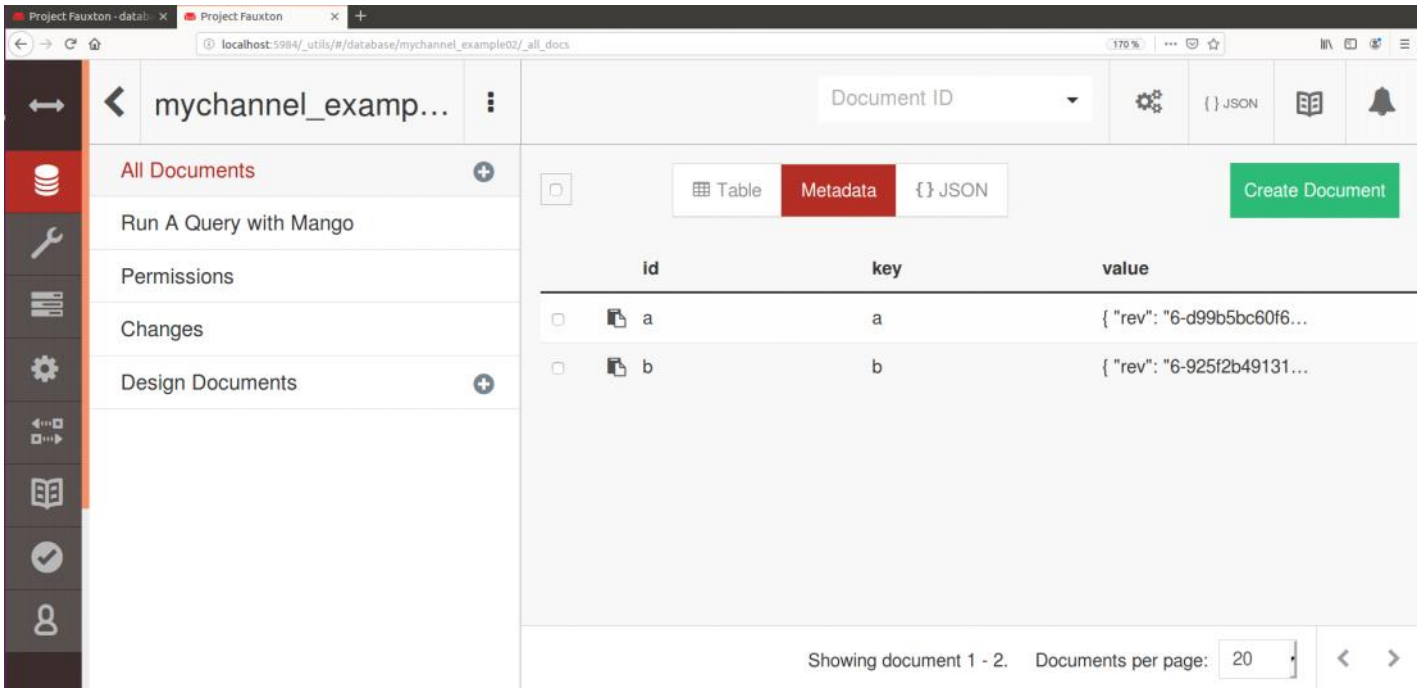
웹브라우저에 localhost 지정된 포트번호로 접속하여

채널명과 chaincode 명으로 경로를 지정하면 내용을 확인할 수 있다.

http://localhost:5984/_utils/#database/mychannel_sacc/_all_docs

http://localhost:5984/_utils/#database/mychannel_example02/_all_docs





13)teardown.sh 수정

기동중인 네트워크를 정지할 때 사용. chaincode가 인스턴트화되면 컨테이너가 추가되므로 다음과 같이 수정

```
# docker rm $(docker ps -aq)
# docker rmi $(docker images dev-* -q)
docker rm $(docker ps -aq -f 'name=dev-*') || true
docker rmi $(docker images dev-* -q)
```