네트워크 구축 실습2-2-gossip

Org1 에 peer를 추가하여 총 3개 peer를 가진 네트워크 구축
couchdb 사용
gossip 프로토콜 사용
전체 스크립트는 basic-network2-2-gossip.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

peer2.org1.example.com:9051

Cli

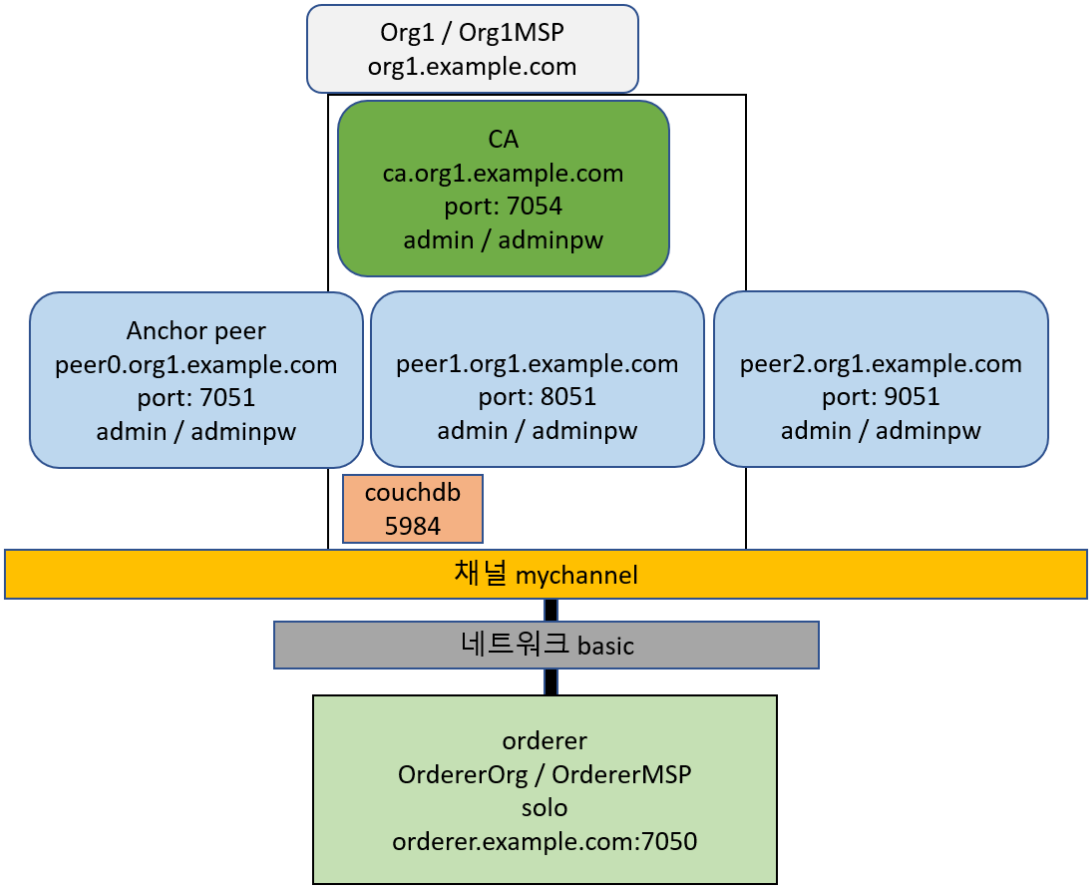
주소 및 포트:

Org1 : cli.example.com

counchdb

주소 및 포트: couchdb : 5984

2. 네트워크 스펙 정리



3. 네트워크 작성하기

1) basic-network2을 basic-network2-2-gossip 로 복사한다.

cp -r basic-network2 basic-network2-2-gossip

cd basic-network2-2-gossip

2) configtx.yaml 수정

수정 사항 없음

3)crypto-config.yaml 수정

Org1 -> Template -> Count : 2 => 3

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

#CORE_PEER_CHAINCODEADDRESS 와 CORE_PEER_CHAINCODELISTENADDRESS 를 설정

#CORE_PEER_GOSSIP_USELEADERELECTION=true : 리더를 투표로 선출

#CORE_PEER_GOSSIP_ORGLEADER=false : 리더를 그룹별로 지정

=> 위 2가지 설정은 서로 exclusive여야 함 (true, false / false, true)

#CORE_PEER_GOSSIP_BOOTSTRAP : 그룹 내의 다른 피어들을 적어주면 됨

#CORE_PEER_GOSSIP_EXTERNALENDPOINT : 다른 그룹에 노출됨

```
  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
      - CORE_PEER_CHAINCODEADDRESS=peer0.org1.example.com:7053
      - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:7053
      - CORE_PEER_GOSSIP_USELEADERELECTION=true
      - CORE_PEER_GOSSIP_ORGLEADER=false
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer1.org1.example.com:8051 peer2.org1.example.com:9051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com: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
    - CORE_PEER_CHAINCODEADDRESS=peer1.org1.example.com:8053
    - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:8053
    - CORE_PEER_GOSSIP_USELEADERELECTION=true
    - CORE_PEER_GOSSIP_ORGLEADER=tfalse
    - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org1.example.com:7051 peer2.org1.example.com:9051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org1.example.com: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

peer2.org1.example.com:
  container_name: peer2.org1.example.com
  image: hyperledger/fabric-peer
  environment:
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - CORE_PEER_ID=peer2.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=peer2.org1.example.com:9051
    - CORE_PEER_LISTENADDRESS=0.0.0.0:9051
    - CORE_PEER_CHAINCODEADDRESS=peer2.org1.example.com:9053
    - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:9053
    - CORE_PEER_GOSSIP_USELEADERELECTION=true
    - CORE_PEER_GOSSIP_ORGLEADER=false
    - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org1.example.com:7051  peer1.org1.example.com:8051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer2.org1.example.com:9051
    # # 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:
    - 9051:9051
    - 9053:9053
  volumes:
      - /var/run/:/host/var/run/
      - ./crypto-config/peerOrganizations/org1.example.com/peers/peer2.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

상단 수정 - peer2.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 peer2.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
```

블록체인을 fetch

```
#fetch
docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp" peer2.org1.example.com peer channel fetch 0 "$CHANNEL_NAME".block --channelID "$CHANNEL_NAME" --orderer orderer.example.com:7050
```

peer2.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" peer2.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 peer2.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 peer2.org1.example.com peer channel list
docker exec peer0.org1.example.com peer node status
docker exec peer1.org1.example.com peer node status
docker exec peer2.org1.example.com peer node status
```

```
bstudent@block-VM:~/fabric-samples$ docker exec peer0.org1.example.com peer channel list
2019-06-20 02:21:48.111 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
Channels peers has joined:
mychannel
bstudent@block-VM:~/fabric-samples$ docker exec peer1.org1.example.com peer channel list
2019-06-20 02:21:49.048 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
Channels peers has joined:
mychannel
bstudent@block-VM:~/fabric-samples$ docker exec peer2.org1.example.com peer channel list
2019-06-20 02:21:49.936 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
Channels peers has joined:
mychannel
bstudent@block-VM:~/fabric-samples$ docker exec peer0.org1.example.com peer node status
status:STARTED
bstudent@block-VM:~/fabric-samples$ docker exec peer1.org1.example.com peer node status
status:STARTED
bstudent@block-VM:~/fabric-samples$ docker exec peer2.org1.example.com peer node status
status:STARTED
```

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

docker logs peer0.org1.example.com 명령으로 각 피어들의 로그를 확인할 수 있다.
이 로그에 gossip 관련 내용이 많이 보입니다.

```
bstudent@block-VM:~/fabric-samples/basic-network2-gossip$ docker logs peer0.org1.example.com
2019-06-20 02:18:38.314 UTC [nodeCmd] serve -> INFO 001 Starting peer:
 Version: 1.4.1
 Commit SHA: 87074a7
 Go version: go1.11.5
 OS/Arch: linux/amd64
 Chaincode:
  Base Image Version: 0.4.15
  Base Docker Namespace: hyperledger
  Base Docker Label: org.hyperledger.fabric
  Docker Namespace: hyperledger
2019-06-20 02:18:38.315 UTC [ledgermgmt] initialize -> INFO 002 Initializing ledger mgmt
2019-06-20 02:18:38.315 UTC [kvledger] NewProvider -> INFO 003 Initializing ledger provider
2019-06-20 02:18:38.439 UTC [kvledger] NewProvider -> INFO 004 ledger provider Initialized
2019-06-20 02:18:38.485 UTC [couchdb] handleRequest -> WARN 005 Retrying couchdb request in 125ms. Attempt:1  Error:Get http://couchdb:5984/: dial tcp 172.31.0.4:5984: connect: connection refused
2019-06-20 02:18:38.612 UTC [couchdb] handleRequest -> WARN 006 Retrying couchdb request in 250ms. Attempt:2  Error:Get http://couchdb:5984/: dial tcp 172.31.0.4:5984: connect: connection refused
2019-06-20 02:18:38.862 UTC [couchdb] handleRequest -> WARN 007 Retrying couchdb request in 500ms. Attempt:3  Error:Get http://couchdb:5984/: dial tcp 172.31.0.4:5984: connect: connection refused
2019-06-20 02:18:40.075 UTC [couchdb] CreateDatabaseIfNotExist -> INFO 008 [_users] Created state database
2019-06-20 02:18:40.119 UTC [couchdb] CreateDatabaseIfNotExist -> INFO 009 Created state database _replicator
2019-06-20 02:18:40.120 UTC [ledgermgmt] initialize -> INFO 00a ledger mgmt initialized
2019-06-20 02:18:40.120 UTC [peer] func1 -> INFO 00b Auto-detected peer address: 172.31.0.6:7051
2019-06-20 02:18:40.121 UTC [peer] func1 -> INFO 00c Returning peer0.org1.example.com:7051
2019-06-20 02:18:40.121 UTC [peer] func1 -> INFO 00d Auto-detected peer address: 172.31.0.6:7051
2019-06-20 02:18:40.121 UTC [peer] func1 -> INFO 00e Returning peer0.org1.example.com:7051
2019-06-20 02:18:40.124 UTC [nodeCmd] computeChaincodeEndpoint -> INFO 00f Entering computeChaincodeEndpoint with peerHostname: peer0.org1.example.com
2019-06-20 02:18:40.124 UTC [nodeCmd] computeChaincodeEndpoint -> INFO 010 Exit with ccEndpoint: peer0.org1.example.com:7053
2019-06-20 02:18:40.127 UTC [sccapi] registerSysCC -> INFO 011 system chaincode lscc(github.com/hyperledger/fabric/core/scc/lscc) registered
2019-06-20 02:18:40.127 UTC [sccapi] registerSysCC -> INFO 012 system chaincode cscc(github.com/hyperledger/fabric/core/scc/cscc) registered
2019-06-20 02:18:40.128 UTC [sccapi] registerSysCC -> INFO 013 system chaincode qscc(github.com/hyperledger/fabric/core/scc/qscc) registered
2019-06-20 02:18:40.128 UTC [sccapi] registerSysCC -> INFO 014 system chaincode (+lifecycle,github.com/hyperledger/fabric/core/chaincode/lifecycle,true) disabled
2019-06-20 02:18:40.132 UTC [gossip.service] func1 -> INFO 015 Initialize gossip with endpoint peer0.org1.example.com:7051 and bootstrap set [peer1.org1.example.com:8051 peer2.org1.example.com:9051]
2019-06-20 02:18:40.141 UTC [gossip.gossip] NewGossipService -> INFO 016 Creating gossip service with self membership of Endpoint: peer0.org1.example.com:7051, InternalEndpoint: peer0.org1.example.com:7051, PKI-ID: 0fdfd94d0e45f399785db1ef0a7e573f3ebbe7b700c135d76083c47b09ed9a61, Metadata:
2019-06-20 02:18:40.142 UTC [gossip.gossip] start -> INFO 017 Gossip instance peer0.org1.example.com:7051 started
```

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

cli 에서 sacc 체인코드 설치-> peer2.org1.example.com

#peer2.org1.example.com 에서도 동일한 값을 읽어올 수 있어야 하고, 값도 변경할 수 있어야 한다.

peer2.org1.example.com 에서 query 로 a 값 읽어오기 150

peer2.org1.example.com 에서 invoke 로 a 값 변경하기 =>160

peer2.org1.example.com 에서 query 로 a 값 다시 읽어오기 160
peer1.org1.example.com 에서 query 로 a 값 다시 읽어오기 160
peer0.org1.example.com 에서 query 로 a 값 다시 읽어오기 160

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"]}'

#install chaincode to peer2.org1.example.com - 각 endoser peer에 모두 설치
docker exec -e CORE_PEER_ADDRESS=peer2.org1.example.com:9051  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-peer2.org1.example.com-sacc-1.0-xxxx 식의 컨테이너 생성됨
docker exec  peer2.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'
docker exec  peer2.org1.example.com peer chaincode invoke  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["set","a","160"]}'
```

```
sleep 5
docker exec  peer2.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME"  -c '{"Args":["get","a"]}'
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 example02 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

cli 에서 example02 체인코드 설치 -> peer2.org1.example.com

#peer2.org1.example.com 에서도 동일한 값을 읽어올 수 있어야 하고, 값도 변경할 수 있어야 한다.

peer2.org1.example.com 에서 query 로 a,b 값 읽어오기 85 215

peer2.org1.example.com 에서 invoke 로 a,b 값 변경하기 => a,b,10

peer2.org1.example.com 에서 query 로 a,b 값 다시 읽어오기 75, 225

peer1.org1.example.com 에서 query 로 a,b 값 다시 읽어오기 75, 225

peer0.org1.example.com 에서 query 로 a,b 값 다시 읽어오기 75, 225

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"]}'

#install chaincode to peer2.org1.example.com - 각 endoser peer에 모두 설치
docker exec -e CORE_PEER_ADDRESS=peer2.org1.example.com:9051  cli peer chaincode install -n "$CC_NAME" -v 1.0 -p "$CC_SRC_PATH" -l
"$CC_RUNTIME_LANGUAGE"
# endoser peer에서 처음 query 수행하면 인스턴스 생성됨 docker ps -a 해보면
# dev-peer2.org1.example.com-example02-1.0-xxxx 식의 컨테이너 생성됨
docker exec  peer2.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
docker exec  peer2.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'
docker exec  peer2.org1.example.com peer chaincode invoke  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["invoke","a","b","10"]}'
sleep 5
docker exec  peer2.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
docker exec  peer2.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'

docker exec  peer2.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
docker exec  peer2.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'
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
```

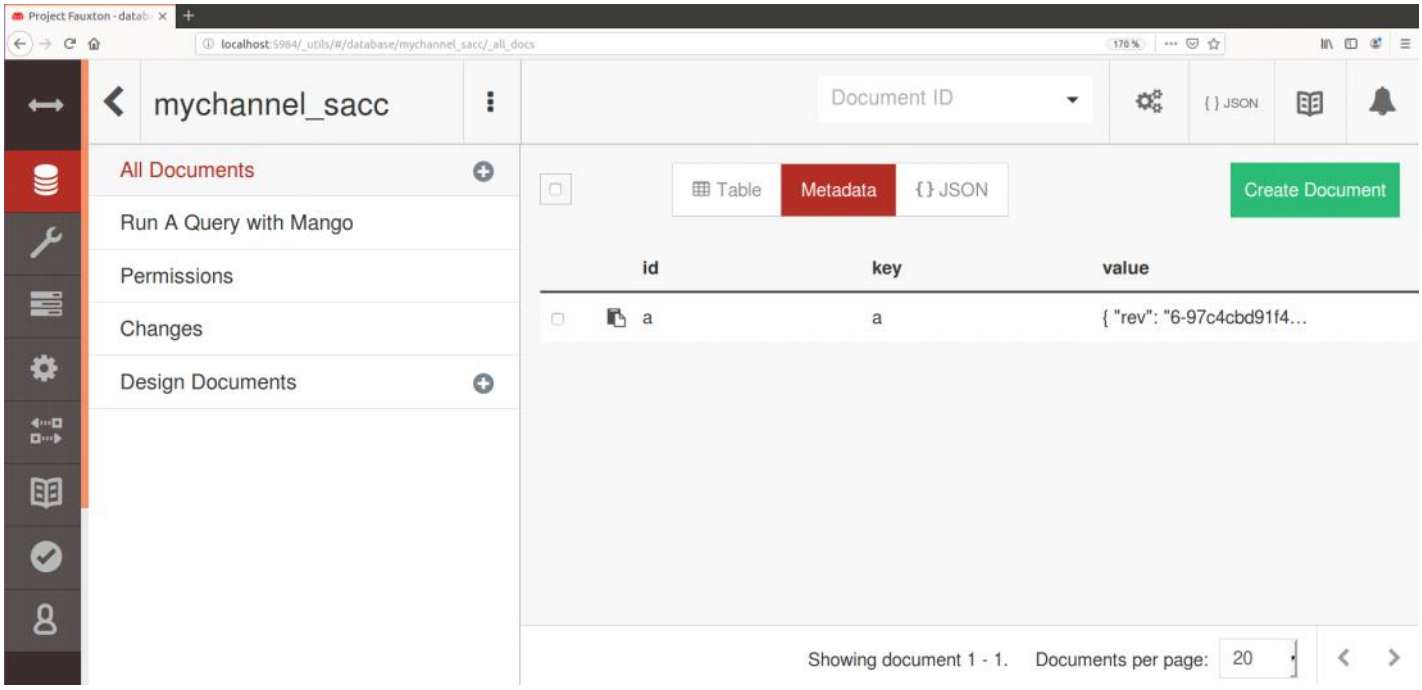
12)couchdb 접속해서 확인해보기

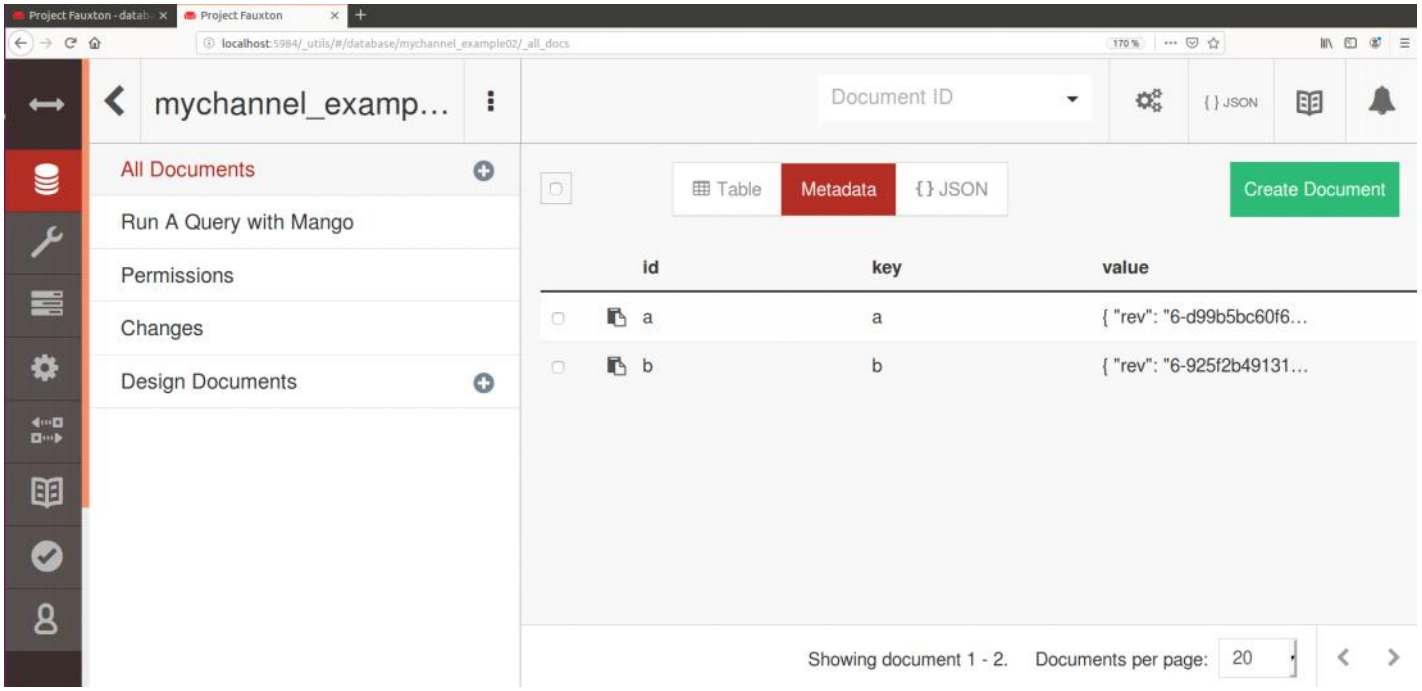
웹브라우저에 localhost 지정된 포트번호로 접속하여

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)
```