네트워크 구축 실습3

Org1 에 peer 2개, Org2에 peer 2개를 가진 네트워크 구축
couchdb 사용
gossip 프로토콜 사용
전체 스크립트는 basic-network3.tar 참조

1. 네트워크 개요 정리

Organization수: 2

Channel

채널수: 1

채널이름: mychannel

Orderer

Orderer수 : 1 Concensus 방식: solo

주소 및 포트: orderer.example.com:7050

Ca

Ca수 : 2

주소 및 포트: ca1.example.com:17054 ca2.example.com:27054

Peer

Organization 별 peer수: Org1 : 2 / Org2 : 2

주소 및 포트:

Org1 : peer0.org1.example.com:17051

peer1.org1.example.com:18051

Org2 : peer0.org2.example.com:27051

peer1.org2.example.com:28051

Cli

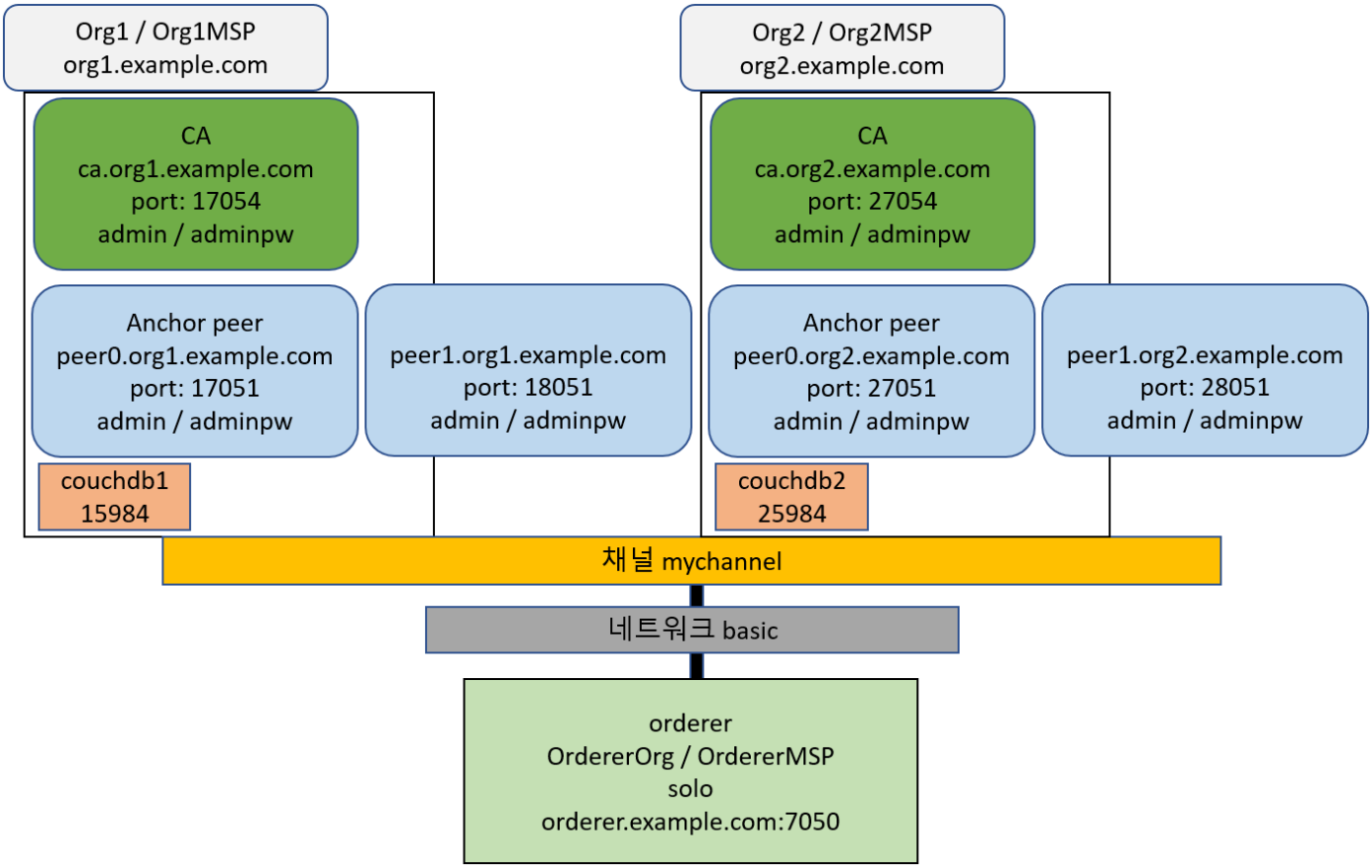
주소 및 포트:

Org1 : cli1.example.com / Org2 : cli2.example.com

counchdb

주소 및 포트: couchdb1 : 15984 / couchdb2 : 25984

1. 네트워크 스펙 정리



3. 네트워크 작성하기

1) basic-network을 basic-network3 로 복사한다.

```
cp -r basic-network basic-network3
cd basic-network3
```

2) configtx.yaml 수정

&Org1 단락 복사해서 2로 수정

```
- &Org1
    # DefaultOrg defines the organization which is used in the sampleconfig
    # of the fabric.git development environment
    Name: Org1MSP

    # ID to load the MSP definition as
    ID: Org1MSP

    MSPDir: crypto-config/peerOrganizations/org1.example.com/msp

    AnchorPeers:
        # AnchorPeers defines the location of peers which can be used
        # for cross org gossip communication.  Note, this value is only
        # encoded in the genesis block in the Application section context
        - Host: peer0.org1.example.com
          Port: 17051

- &Org2
    # DefaultOrg defines the organization which is used in the sampleconfig
    # of the fabric.git development environment
    Name: Org2MSP

    # ID to load the MSP definition as
```

```
        ID: Org2MSP

        MSPDir: crypto-config/peerOrganizations/org2.example.com/msp

        AnchorPeers:
            # AnchorPeers defines the location of peers which can be used
            # for cross org gossip communication.  Note, this value is only
            # encoded in the genesis block in the Application section context
            - Host: peer0.org2.example.com
              Port: 27051
```

아래처럼 OrdererGenesis 와 OrgChannel 이름 변경, Org2 추가

```
Profiles:

    TwoOrgOrdererGenesis:
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
        Consortiums:
            SampleConsortium:
                Organizations:
                    - *Org1
                    - *Org2
    TwoOrgChannel:
        Consortium: SampleConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
                - *Org2
```

3)crypto-config.yaml 수정

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

Org1 단락을 복사하여 Org2로 수정

```
  - Name: Org2
    Domain: org2.example.com
    Template:
      Count: 2
      # Start: 5
    Users:
      Count: 1
```

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 - Org1

```
# 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
```

하단에 추가 #peer가 2개 이상이 되면 1개가 anchor peer가 되어야 하므로 설정 필요 - Org2

```
# generate anchor peer transaction
configtxgen -profile TwoOrgChannel -outputAnchorPeersUpdate ./config/Org2MSPanchors.tx -channelID $CHANNEL_NAME -asOrg Org2MSP
if [ "$?" -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 Org1 peer0 peer1 Org2 peer0 peer1 관련 폴더 생성 확인

6)docker-compose.yaml 수정

a. ca가 2개이므로 공통부분을 ca-base 섹션 만들어서 저장하고 활용

각 Org마다 ca 1개씩 생성

ca-base 섹션 사용하려면 extends: 로 선언해줘야 함

```
  ca-base:
    image: hyperledger/fabric-ca
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
    networks:
      - basic

  # ORG1 CA
  ca.org1.example.com:
    extends:
      service: ca-base
```

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

crypto-config/peerOrganizations/org1.example.com/ca , crypto-config/peerOrganizations/org2.example.com/ca 폴더에서 _sk 로 끝나는 파일명 으로 대체

각 ca마다 port 번호 다르게 변경

```
  ca-base:
    image: hyperledger/fabric-ca
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
    networks:
      - basic

  # ORG1 CA
  ca.org1.example.com:
    extends:
      service: ca-base
    environment:
      - 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
      - FABRIC_CA_SERVER_CA_KEYFILE=/etc/hyperledger/fabric-ca-server-config/931060b9927958909b111e87f76fba95327035ce23c95056fae831e05aa3ac83_sk
    ports:
      - "17054:7054"
    command: sh -c 'fabric-ca-server start -b admin:adminpw'
```

```
    volumes:
      - ./crypto-config/peerOrganizations/org1.example.com/ca/:/etc/hyperledger/fabric-ca-server-config
    container_name: ca.org1.example.com
    networks:
      - basic

  # ORG2 CA
  ca.org2.example.com:
    extends:
      service: ca-base
    environment:
      - FABRIC_CA_SERVER_CA_NAME=ca.org2.example.com
      - FABRIC_CA_SERVER_CA_CERTFILE=/etc/hyperledger/fabric-ca-server-config/ca.org2.example.com-cert.pem
      - FABRIC_CA_SERVER_CA_KEYFILE=/etc/hyperledger/fabric-ca-server-
config/d06dc6a324470b9650a1063e8f3b58734df4cb0eb65f069e345475e9373c318c_sk
    ports:
      - "27054:7054"
    command: sh -c 'fabric-ca-server start -b admin:adminpw'
    volumes:
      - ./crypto-config/peerOrganizations/org2.example.com/ca/:/etc/hyperledger/fabric-ca-server-config
    container_name: ca.org2.example.com
    networks:
      - basic
```

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

d. org가 2개, 각 org당 peer가 2개씩이므로 공통부분을 peer-base 섹션 만들어서 저장하고 활용

   각 Org마다 peer 2개씩 생성

   peer-base 섹션 사용하려면 extends: 로 선언해줘야 함

d. 각 Org, peer마다 port 번호 다르게 변경

#CORE_PEER_ADDRESS=0.0.0.0:17051 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

f. 각 Org마다 couchdb 1개씩 배치하므로 환경설정과 depends_on 에 추가해줘야 함

```
  peer-base:
    image: hyperledger/fabric-peer
    environment:
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - FABRIC_LOGGING_SPEC=info
      - CORE_CHAINCODE_LOGGING_LEVEL=info
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=${COMPOSE_PROJECT_NAME}_basic
      - CORE_PEER_GOSSIP_USELEADERELECTION=true
      - CORE_PEER_GOSSIP_ORGLEADER=false
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      # 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
  volumes:
    - /var/run/:/host/var/run/
    - ./config:/etc/hyperledger/configtx
  networks:
    - basic

peer0.org1.example.com:
  extends:
    service: peer-base
  container_name: peer0.org1.example.com
  environment:
    - CORE_PEER_ID=peer0.org1.example.com
    - CORE_PEER_LOCALMSPID=Org1MSP
    - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp
    - CORE_PEER_ADDRESS=peer0.org1.example.com:17051
    - CORE_PEER_LISTENADDRESS=0.0.0.0:17051
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb1:5984
    - CORE_PEER_CHAINCODEADDRESS=peer0.org1.example.com:17053
    - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:17053
    - CORE_PEER_GOSSIP_BOOTSTRAP=peer1.org1.example.com:18051 peer0.org1.example.com:17051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:17051
  ports:
    - 17051:17051
    - 17053:17053
  volumes:
      - ./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
  depends_on:
    - orderer.example.com
    - couchdb1

peer1.org1.example.com:
  extends:
    service: peer-base
  container_name: peer1.org1.example.com
  environment:
    - CORE_PEER_ID=peer1.org1.example.com
    - CORE_PEER_LOCALMSPID=Org1MSP
    - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp
    - CORE_PEER_ADDRESS=peer1.org1.example.com:18051
    - CORE_PEER_LISTENADDRESS=0.0.0.0:18051
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb1:5984
    - CORE_PEER_CHAINCODEADDRESS=peer1.org1.example.com:18053
    - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:18053
    - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org1.example.com:17051 peer1.org1.example.com:18051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org1.example.com:18051
  ports:
    - 18051:18051
    - 18053:18053
  volumes:
      - ./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
  depends_on:
    - orderer.example.com
    - couchdb1

peer0.org2.example.com:
```

```
  extends:
    service: peer-base
  container_name: peer0.org2.example.com
  environment:
    - CORE_PEER_ID=peer0.org2.example.com
    - CORE_PEER_LOCALMSPID=Org2MSP
    - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org2.example.com/msp
    - CORE_PEER_ADDRESS=peer0.org2.example.com:27051
    - CORE_PEER_LISTENADDRESS=0.0.0.0:27051
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb2:5984
    - CORE_PEER_CHAINCODEADDRESS=peer0.org2.example.com:27053
    - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:27053
    - CORE_PEER_GOSSIP_BOOTSTRAP=peer1.org2.example.com:28051 peer0.org2.example.com:27051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org2.example.com:27051
  ports:
    - 27051:27051
    - 27053:27053
  volumes:
      - ./crypto-config/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/msp:/etc/hyperledger/msp/peer
      - ./crypto-config/peerOrganizations/org2.example.com/users:/etc/hyperledger/msp/users
  depends_on:
    - orderer.example.com
    - couchdb2

peer1.org2.example.com:
  extends:
    service: peer-base
  container_name: peer1.org2.example.com
  environment:
    - CORE_PEER_ID=peer1.org2.example.com
    - CORE_PEER_LOCALMSPID=Org2MSP
    - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org2.example.com/msp
    - CORE_PEER_ADDRESS=peer1.org2.example.com:28051
    - CORE_PEER_LISTENADDRESS=0.0.0.0:28051
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb2:5984
    - CORE_PEER_CHAINCODEADDRESS=peer1.org2.example.com:28053
    - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:28053
    - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org2.example.com:27051 peer1.org2.example.com:28051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org2.example.com:28051
  ports:
    - 28051:28051
    - 28053:28053
  volumes:
      - ./crypto-config/peerOrganizations/org2.example.com/peers/peer1.org2.example.com/msp:/etc/hyperledger/msp/peer
      - ./crypto-config/peerOrganizations/org2.example.com/users:/etc/hyperledger/msp/users
  depends_on:
    - orderer.example.com
    - couchdb2
```

g. couchdb 수정하고 복사

h. org가 2개, 각 org당 couchdb가 1개씩이므로 공통부분을 couchdb-base 섹션 만들어서 저장하고 활용
   각 Org마다 couchdb 1개씩 생성
   couchdb-base 섹션 사용하려면 extends: 로 선언해줘야 함

i. 각 couchdb 마다 port 번호 다르게 변경

```
couchdb-base:
  image: hyperledger/fabric-couchdb
  # Populate the COUCHDB_USER and COUCHDB_PASSWORD to set an admin user and password
```

```
    # for CouchDB.  This will prevent CouchDB from operating in an "Admin Party" mode.
    environment:
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=
    networks:
      - basic

  couchdb1:
    extends:
      service: couchdb-base
    container_name: couchdb1
    ports:
      - 15984:5984

  couchdb2:
    extends:
      service: couchdb-base
    container_name: couchdb2
    ports:
      - 25984:5984
```

j. cli 수정하고 복사

h. org가 2개, 각 org당 cli가 1개씩이므로 공통부분을 cli-base 섹션 만들어서 저장하고 활용

각 Org마다 cli 1개씩 생성

cli-base 섹션 사용하려면 extends: 로 선언해줘야 함

```
  cli-base:
    image: hyperledger/fabric-tools
    tty: true
    environment:
      - GOPATH=/opt/gopath
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - FABRIC_LOGGING_SPEC=info
      - CORE_CHAINCODE_KEEPALIVE=10
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
    command: /bin/bash
    volumes:
        - /var/run/:/host/var/run/
        - ./config/:/etc/hyperledger/configtx
        - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
        - ./../chaincode/:/opt/gopath/src/github.com/
    networks:
      - basic

  cli_org1:
    extends:
      service: cli-base
    container_name: cli_org1
    environment:
      - CORE_PEER_ID=cli_org1
      - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
      - CORE_PEER_LOCALMSPID=Org1MSP
      -
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.
example.com/msp
    depends_on:
     - orderer.example.com
     - peer0.org1.example.com
     - couchdb1

  cli_org2:
    extends:
      service: cli-base
    container_name: cli_org2
```

```
  environment:
    - CORE_PEER_ID=cli_org2
    - CORE_PEER_ADDRESS=peer0.org2.example.com:7051
    - CORE_PEER_LOCALMSPID=Org2MSP
    -
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/users/Admin@org2.
example.com/msp
  depends_on:
    - orderer.example.com
    - peer0.org2.example.com
    - couchdb2
```

7) start.sh 수정

상단 수정 - ca.org1.example.com ca.org2.example.com peer1.org1.example.com peer0.org2.example.com peer1.org2.example.com couchdb1 couchdb2 추가

```
docker-compose -f docker-compose.yml up -d ca.org1.example.com ca.org2.example.com orderer.example.com peer0.org1.example.com
peer1.org1.example.com couchdb1 peer0.org2.example.com peer1.org2.example.com couchdb2
```

Org1에 peer가 2개 이상이므로 peer0를 anchor 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
```

Org2 도 동일하게 update , fetch, join 진행

```
# Org2
#fetch
docker exec -e "CORE_PEER_LOCALMSPID=Org2MSP" -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org2.example.com/msp"
peer0.org2.example.com peer channel fetch 0 "$CHANNEL_NAME".block --channelID "$CHANNEL_NAME" --orderer orderer.example.com:7050

# Join peer0.org2.example.com to the channel.
docker exec -e "CORE_PEER_LOCALMSPID=Org2MSP" -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org2.example.com/msp"
peer0.org2.example.com peer channel join -b "$CHANNEL_NAME".block

# update mychannel1
docker exec -e "CORE_PEER_LOCALMSPID=Org2MSP" -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org2.example.com/msp"
peer0.org2.example.com peer channel update -o orderer.example.com:7050 -c "$CHANNEL_NAME" -f /etc/hyperledger/configtx/Org2MSPanchors.tx

#fetch
docker exec -e "CORE_PEER_LOCALMSPID=Org2MSP" -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org2.example.com/msp"
peer1.org2.example.com peer channel fetch 0 "$CHANNEL_NAME".block --channelID "$CHANNEL_NAME" --orderer orderer.example.com:7050

# Join peer1.org2.example.com to the channel.
docker exec -e "CORE_PEER_LOCALMSPID=Org2MSP" -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org2.example.com/msp"
peer1.org2.example.com peer channel join -b "$CHANNEL_NAME".block
```

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

```
ca.org1.example.com ca.org2.example.com orderer.example.com peer0.org1.example.com peer1.org1.example.com couchdb1 peer0.org2.example.com
peer1.org2.example.com couchdb2
```

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.org2.example.com peer channel list
docker exec peer1.org2.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 peer0.org2.example.com peer node status
docker exec peer1.org2.example.com peer node status
```

결과 확인 - 각 피어가 설계된대로 채널에 가입되었는지와 상태 확인

10)체인코드 설치 및 실행

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

#Org1

cli_org1 에서 sacc 체인코드 설치

cli_org1 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

#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

#Org2

cli_org2 에서 sacc 체인코드 설치

cli_org2 에서 sacc 체인코드 인스턴스화

peer0.org2.example.com 에서 query 로 a 값 읽어오기 150

peer0.org2.example.com 에서 invoke 로 a 값 변경하기 => 160

peer0.org2.example.com 에서 query 로 a 값 다시 읽어오기 160

#peer1.org2.example.com 에서도 동일한 값을 읽어올 수 있어야 하고, 값도 변경할 수 있어야 한다.

peer1.org2.example.com 에서 query 로 a 값 읽어오기 160

peer1.org2.example.com 에서 invoke 로 a 값 변경하기 =>170

peer1.org2.example.com 에서 query 로 a 값 다시 읽어오기 170

peer0.org2.example.com 에서 query 로 a 값 다시 읽어오기 170

peer0.org1.example.com 에서 query 로 a 값 다시 읽어오기 170

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_org1  cli_org2
docker ps -a
# Org1
echo ===================================================================
echo        Org1
echo ===================================================================
echo install chaincode to peer0.org1.example.com
#install chaincode to peer0.org1.example.com - 각 endoser peer에 모두 설치
docker exec  cli_org1 peer chaincode install -n "$CC_NAME" -v "$CC_VERSION" -p "$CC_SRC_PATH" -l "$CC_RUNTIME_LANGUAGE"
echo instantiate chaincode to mychannel
#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_org1 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','Org2MSP.member')"
sleep 5
# 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"]}'
echo ===================================================================
echo install chaincode to peer1.org1.example.com
#install chaincode to peer1.org1.example.com - 각 endoser peer에 모두 설치
docker exec -e CORE_PEER_ADDRESS=peer1.org1.example.com:18051  cli_org1 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"]}'

# Org2
echo ===================================================================
echo        Org2
echo ===================================================================
echo install chaincode to peer0.org2.example.com
#install chaincode to peer0.org2.example.com - 각 endoser peer에 모두 설치
docker exec -e CORE_PEER_ADDRESS=peer0.org2.example.com:27051  cli_org2 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-peer0.org2.example.com-sacc-1.0-xxxx 식의 컨테이너 생성됨
docker exec  peer0.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'
sleep 5
docker exec  peer0.org2.example.com peer chaincode invoke  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["set","a","160"]}'
sleep 5
docker exec  peer0.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'
echo ===================================================================
echo install chaincode to peer1.org2.example.com
#install chaincode to peer1.org2.example.com - 각 endoser peer에 모두 설치
docker exec -e CORE_PEER_ADDRESS=peer1.org2.example.com:28051  cli_org2 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.org2.example.com-sacc-1.0-xxxx 식의 컨테이너 생성됨
docker exec  peer1.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'
sleep 5
docker exec  peer1.org2.example.com peer chaincode invoke  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["set","a","170"]}'
sleep 5
docker exec  peer1.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'
docker exec  peer0.org2.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

#Org1

cli_org1 에서 example02 체인코드 설치 -> peer0.org1.example.com

cli_org1 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

#Org2

cli_org2 에서 example02 체인코드 설치 -> peer0.org2.example.com

cli_org2 에서 example02 체인코드 인스턴스화

peer0.org2.example.com 에서 query 로 a,b 값 읽어오기 85 215

peer0.org2.example.com 에서 invoke 로 a,b 값 변경하기 =>a,b,5

peer0.org2.example.com 에서 query 로 a,b 값 다시 읽어오기 80 220

cli 에서 example02 체인코드 설치 -> peer1.org2.example.com

#peer1.org2.example.com 에서도 동일한 값을 읽어올 수 있어야 하고, 값도 변경할 수 있어야 한다.

peer1.org2example.com 에서 query 로 a,b 값 읽어오기 80 220

peer1.org2.example.com 에서 invoke 로 a,b 값 변경하기 => a,b,5

peer1.org2.example.com 에서 query 로 a,b 값 다시 읽어오기 75, 225

peer0.org2.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_org1 cli_org2
docker ps -a
# Org1
echo ===================================================================
echo         Org1
echo ===================================================================
echo  install chaincode to peer0.org1.example.com
#install chaincode to peer0.org1.example.com - 각 endoser peer에 모두 설치
docker exec  cli_org1 peer chaincode install -n "$CC_NAME" -v "$CC_VERSION" -p "$CC_SRC_PATH" -l "$CC_RUNTIME_LANGUAGE"
echo  instantiate chaincode to mychannel
#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_org1 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','Org2MSP.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"]}'
echo ===================================================================
echo  install chaincode to peer1.org1.example.com
#install chaincode to peer1.org1.example.com - 각 endoser peer에 모두 설치
docker exec -e CORE_PEER_ADDRESS=peer1.org1.example.com:18051  cli_org1 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-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"]}'
sleep 5
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"]}'

# Org2
echo ===================================================================
echo         Org2
echo ===================================================================
echo  install chaincode to peer0.org2.example.com
#install chaincode to peer0.org2.example.com - 각 endoser peer에 모두 설치
docker exec -e CORE_PEER_ADDRESS=peer0.org2.example.com:27051  cli_org2   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-peer0.org2.example.com-example02-1.0-xxxx 식의 컨테이너 생성됨
# docker exec  -e CORE_PEER_ADDRESS=peer0.org2.example.com:7051  cli_org2   peer chaincode query -o orderer.example.com:7050 -C
"$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
```

```
docker exec  peer0.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
sleep 5
docker exec  peer0.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'
# docker exec  -e CORE_PEER_ADDRESS=peer0.org2.example.com:7051  cli_org2  peer chaincode invoke -o orderer.example.com:7050  -C
"$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["invoke","a","b","5"]}'
docker exec  peer0.org2.example.com peer chaincode invoke  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["invoke","a","b","5"]}'
sleep 5
docker exec  peer0.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
docker exec  peer0.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'
echo ========================================================================
echo  install chaincode to peer1.org2.example.com
#install chaincode to peer1.org2.example.com - 각 endoser peer에 모두 설치
docker exec -e CORE_PEER_ADDRESS=peer1.org2.example.com:28051  cli_org2 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.org2.example.com-example02-1.0-xxxx 식의 컨테이너 생성됨
docker exec  peer1.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
sleep 5
docker exec  peer1.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'
docker exec  peer1.org2.example.com peer chaincode invoke  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["invoke","a","b","5"]}'
sleep 5
docker exec  peer1.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
docker exec  peer1.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'

docker exec  peer0.org2.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","a"]}'
docker exec  peer0.org2.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

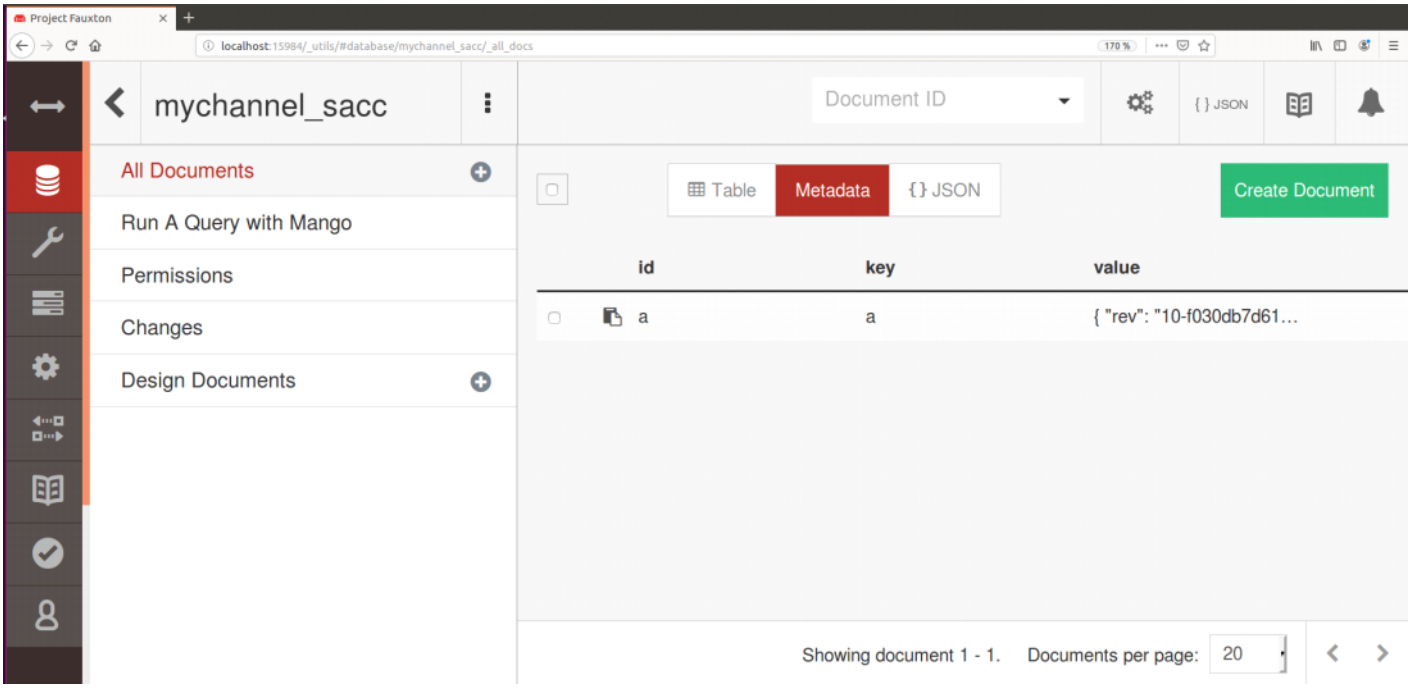
채널명과 chaincode 명으로 경로를 지정하면 내용을 확인할 수 있다.

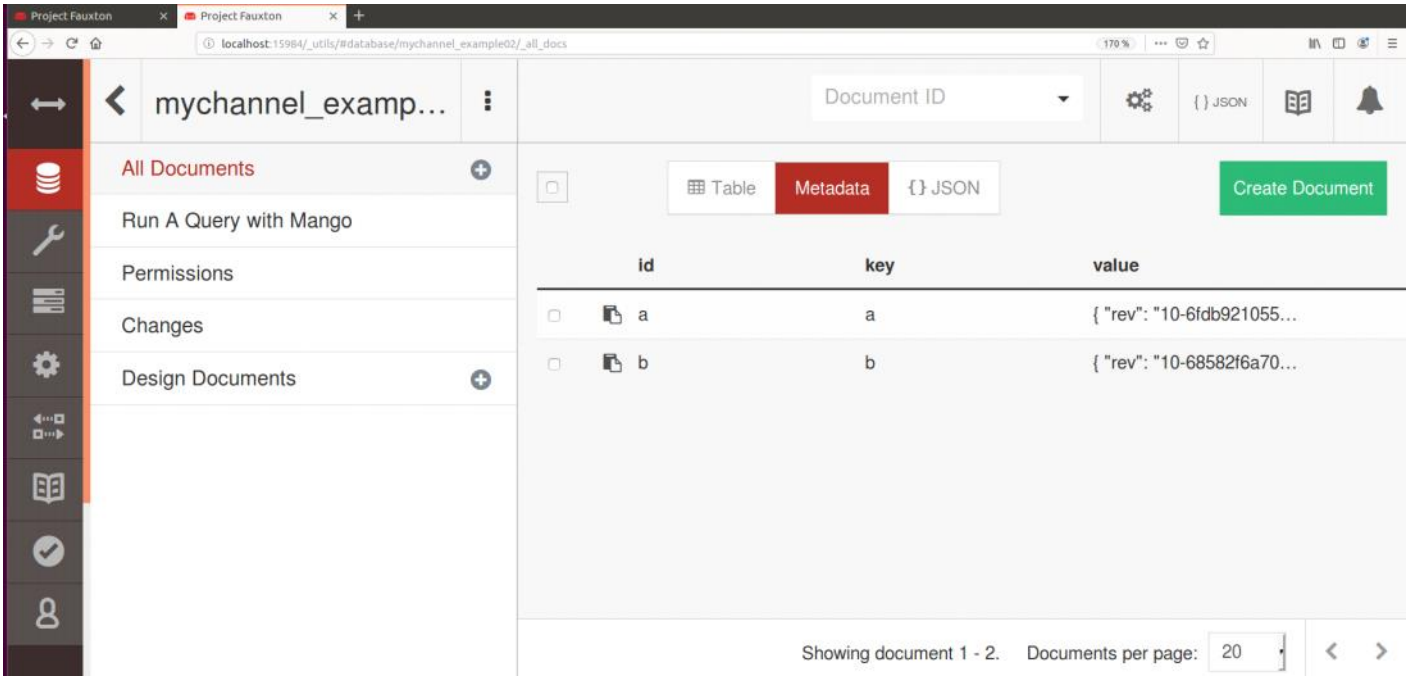
http://localhost:15984/_utils/#database/mychannel_sacc/_all_docs

http://localhost:25984/_utils/#database/mychannel_sacc/_all_docs

http://localhost:15984/_utils/#database/mychannel_example02/_all_docs

http://localhost:25984/_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)
```