네트워크 구축 실습1

Org1 에 peer1개를 가진 네트워크 구축
couchdb 사용 (기본값은 leveldb)
전체 스크립트는 basic-network1.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 : 1

주소 및 포트:

Org1 : peer0.org1.example.com:7051

Cli

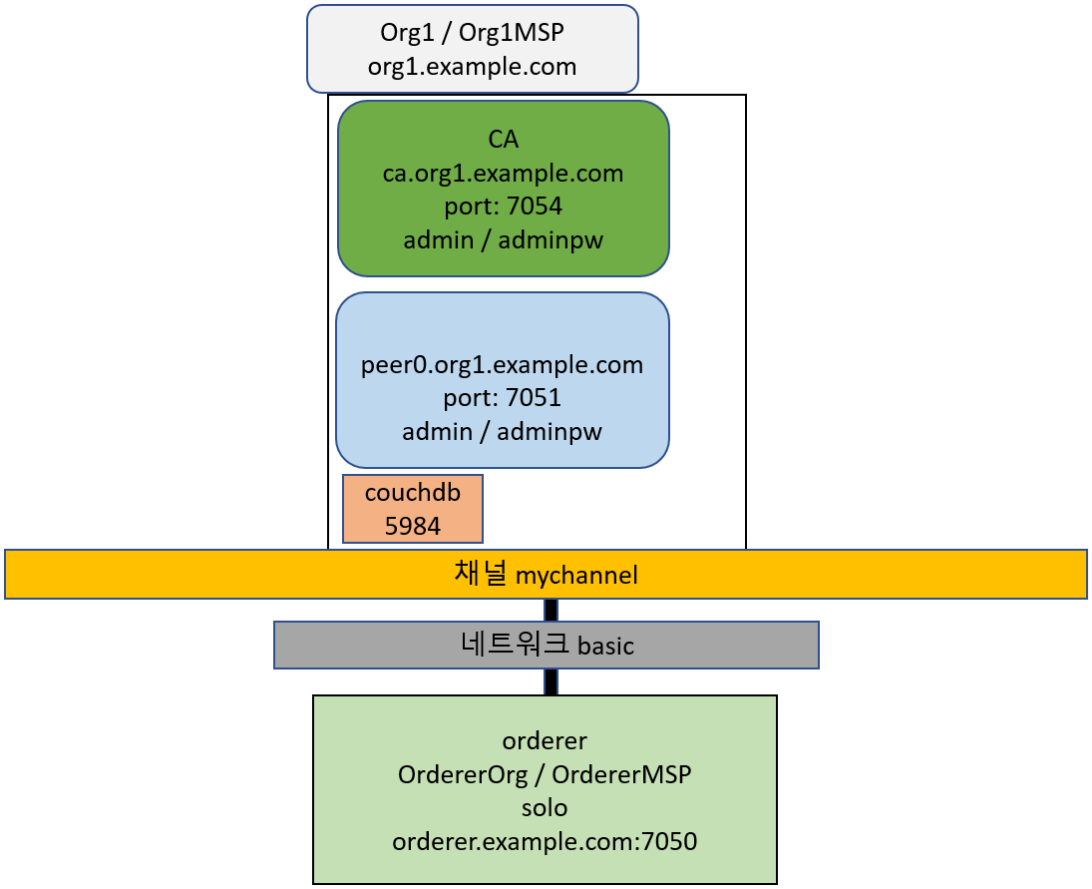
주소 및 포트:

Org1 : cli.example.com

counchdb

주소 및 포트: couchdb : 5984

1. 네트워크 스펙 정리



3. 네트워크 작성하기

1) basic-network을 basic-network1 로 복사한다.

cp -r basic-network basic-network1
cd basic-network1

2) configtx.yaml

Profiles 섹션에 이름 지정 하고 Org 1개에 peer 1개 지정

OneOrgOrdererGenesis OneOrgChannel

&Org1 섹션 내용 입력하고, anchor peer 지정

```
# Copyright IBM Corp. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
##############################################################################
#   Section: Organizations
#   - This section defines the different organizational identities which will
#   be referenced later in the configuration.
##############################################################################
Organizations:
    # SampleOrg defines an MSP using the sampleconfig.  It should never be used
    # in production but may be used as a template for other definitions
    - &OrdererOrg
        # DefaultOrg defines the organization which is used in the sampleconfig
        # of the fabric.git development environment
        Name: OrdererOrg
        # ID to load the MSP definition as
        ID: OrdererMSP
        # MSPDir is the filesystem path which contains the MSP configuration
        MSPDir: crypto-config/ordererOrganizations/example.com/msp
    - &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: 7051
################################################################################
#   SECTION: Application
#   - This section defines the values to encode into a config transaction or
#   genesis block for application related parameters
################################################################################
Application: &ApplicationDefaults
    # Organizations is the list of orgs which are defined as participants on
    # the application side of the network
    Organizations:
################################################################################
#   SECTION: Orderer
#   - This section defines the values to encode into a config transaction or
#   genesis block for orderer related parameters
################################################################################
Orderer: &OrdererDefaults
    # Orderer Type: The orderer implementation to start
    # Available types are "solo" and "kafka"
    OrdererType: solo
    Addresses:
        - orderer.example.com:7050
    # Batch Timeout: The amount of time to wait before creating a batch
    BatchTimeout: 2s
    # Batch Size: Controls the number of messages batched into a block
    BatchSize:
        # Max Message Count: The maximum number of messages to permit in a batch
        MaxMessageCount: 10
        # Absolute Max Bytes: The absolute maximum number of bytes allowed for
        # the serialized messages in a batch.
        AbsoluteMaxBytes: 99 MB
        # Preferred Max Bytes: The preferred maximum number of bytes allowed for
        # the serialized messages in a batch. A message larger than the preferred
        # max bytes will result in a batch larger than preferred max bytes.
        PreferredMaxBytes: 512 KB
    Kafka
        # Brokers: A list of Kafka brokers to which the orderer connects
        # NOTE: Use IP:port notation
        Brokers:
            - 127.0.0.1:9092
    # Organizations is the list of orgs which are defined as participants on
    # the orderer side of the network
    Organizations:
################################################################################
#   Profile
#   - Different configuration profiles may be encoded here to be specified
#   as parameters to the configtxgen tool
################################################################################
Profiles:
    OneOrgOrdererGenesis:
```

```
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
        Consortiums:
            SampleConsortium:
                Organizations:
                    - *Org1
    OneOrgChannel:
        Consortium: SampleConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
```

3)crypto-config.yaml

Org1 -> Template -> Count : 1 <= peer 개수

```
# Copyright IBM Corp. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
# ---------------------------------------------------------------------------
# "OrdererOrgs" - Definition of organizations managing orderer nodes
# ---------------------------------------------------------------------------
OrdererOrgs:
  # ---------------------------------------------------------------------------
  # Orderer
  # ---------------------------------------------------------------------------
  - Name: Orderer
    Domain: example.com
    # ---------------------------------------------------------------------------
    # "Specs" - See PeerOrgs below for complete description
    # ---------------------------------------------------------------------------
    Specs:
      - Hostname: orderer
# ---------------------------------------------------------------------------
# "PeerOrgs" - Definition of organizations managing peer nodes
# ---------------------------------------------------------------------------
PeerOrgs:
  # ---------------------------------------------------------------------------
  # Org1
  # ---------------------------------------------------------------------------
  - Name: Org1
    Domain: org1.example.com
    # ---------------------------------------------------------------------------
    # "Specs"
    # ---------------------------------------------------------------------------
    # Uncomment this section to enable the explicit definition of hosts in your
    # configuration.  Most users will want to use Template, below
    # Specs is an array of Spec entries.  Each Spec entry consists of two fields:
    #   - Hostname:   (Required) The desired hostname, sans the domain.
    #   - CommonName: (Optional) Specifies the template or explicit override for
    #                 the CN.  By default, this is the template:
    #                                 "{{.Hostname}}.{{.Domain}}"
    #                 which obtains its values from the Spec.Hostname and
    #                 Org.Domain, respectively.
    # ---------------------------------------------------------------------------
    # Specs:
    #   - Hostname: foo # implicitly "foo.org1.example.com"
    #     CommonName: foo27.org5.example.com # overrides Hostname-based FQDN set above
```

```
    #   - Hostname: bar
    #   - Hostname: baz
    # ---------------------------------------------------------------------------
    # "Template"
    # ---------------------------------------------------------------------------
    # Allows for the definition of 1 or more hosts that are created sequentially
    # from a template. By default, this looks like "peer%d" from 0 to Count-1.
    # You may override the number of nodes (Count), the starting index (Start)
    # or the template used to construct the name (Hostname).
    # Note: Template and Specs are not mutually exclusive.  You may define both
    # sections and the aggregate nodes will be created for you.  Take care with
    # name collisions
    # ---------------------------------------------------------------------------
    Template:
      Count: 1
      # Start: 5
      # Hostname: {{.Prefix}}{{.Index}} # default
    # ---------------------------------------------------------------------------
    # "Users"
    # ---------------------------------------------------------------------------
    # Count: The number of user accounts _in addition_ to Admin
    # ---------------------------------------------------------------------------
    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
```

ancker peer 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 peer0 관련 폴더 생성 확인

조직 변경이 없으면 1 -5 번 과정은 수행할 필요 없음

6)docker-compose.yaml

a. ca

ca의 FABRIC_CA_SERVER_CA_KEYFILE 값 - generate.sh 실행하면 crypto-config 이 변경되므로

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

```
  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
  ports:
    - "7054:7054"
  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.example.com
  networks:
    - basic
```

b. orderer

```
orderer.example.com:
  container_name: orderer.example.com
  image: hyperledger/fabric-orderer
  environment:
    - FABRIC_LOGGING_SPEC=info
    - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
    - ORDERER_GENERAL_GENESISMETHOD=file
    - ORDERER_GENERAL_GENESISFILE=/etc/hyperledger/configtx/genesis.block
    - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
    - ORDERER_GENERAL_LOCALMSPDIR=/etc/hyperledger/msp/orderer/msp
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric/orderer
  command: orderer
  ports:
    - 7050:7050
  volumes:
      - ./config/:/etc/hyperledger/configtx
      - ./crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/:/etc/hyperledger/msp/orderer
      - ./crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/:/etc/hyperledger/msp/peerOrg1
  networks:
    - basic
```

c. peer

```
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
    # # 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
```

d. couchdb

```
couchdb:
  container_name: couchdb
  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=
  ports:
    - 5984:5984
  networks:
    - basic
```

e. cli

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

7) start.sh

상단에 추가 #향후 채널 추가에 대비하여 변수로 지정

CHANNEL_NAME=mychannel

상단 - 실행할 컨테이너 지정

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

create channel

```
CHANNEL_NAME=mychanne
# Create the channel
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 create -o orderer.example.com:7050 -c "$CHANNEL_NAME"  -f /etc/hyperledger/configtx/"$CHANNEL_NAME".tx
```

peer0.org1.example.com 을 mychannel에 join

```
# Join peer0.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"
peer0.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 couchdb
```

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

```
docker exec peer0.org1.example.com peer channel list
docker exec peer0.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 peer0.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

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

docker-compose -f ./docker-compose.yml up -d cli
```

```
docker ps -a

docker exec  cli peer chaincode install -n "$CC_NAME" -v 1.0 -p "$CC_SRC_PATH" -l "$CC_RUNTIME_LANGUAGE"
docker exec  cli peer chaincode instantiate -o orderer.example.com:7050 -C "$CHANNEL_NAME" -n "$CC_NAME" -l "$CC_RUNTIME_LANGUAGE" -v 1.0 -c '{"Args":["a","15"]}' -P "OR ('Org1MSP.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 "$CHANNEL_NAME" -n sacc -c '{"Args":["get","a"]}'
docker exec  peer0.org1.example.com peer chaincode invoke  -C "$CHANNEL_NAME" -n sacc -c '{"Args":["set","a","130"]}'
sleep 5
docker exec  peer0.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n sacc -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

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

docker exec  cli peer chaincode install -n "$CC_NAME" -v "$CC_VERSION" -p "$CC_SRC_PATH" -l "$CC_RUNTIME_LANGUAGE"
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  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":["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"]}'

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

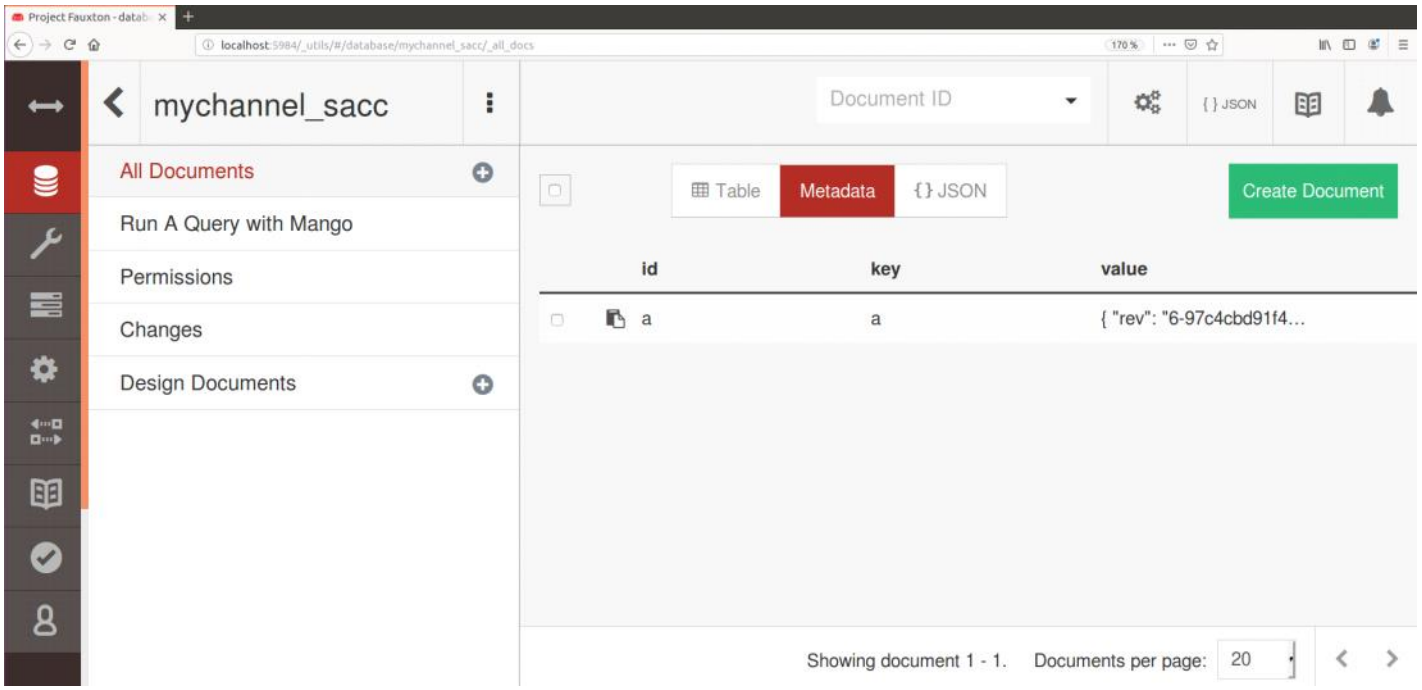
12)couchdb 접속해서 확인해보기

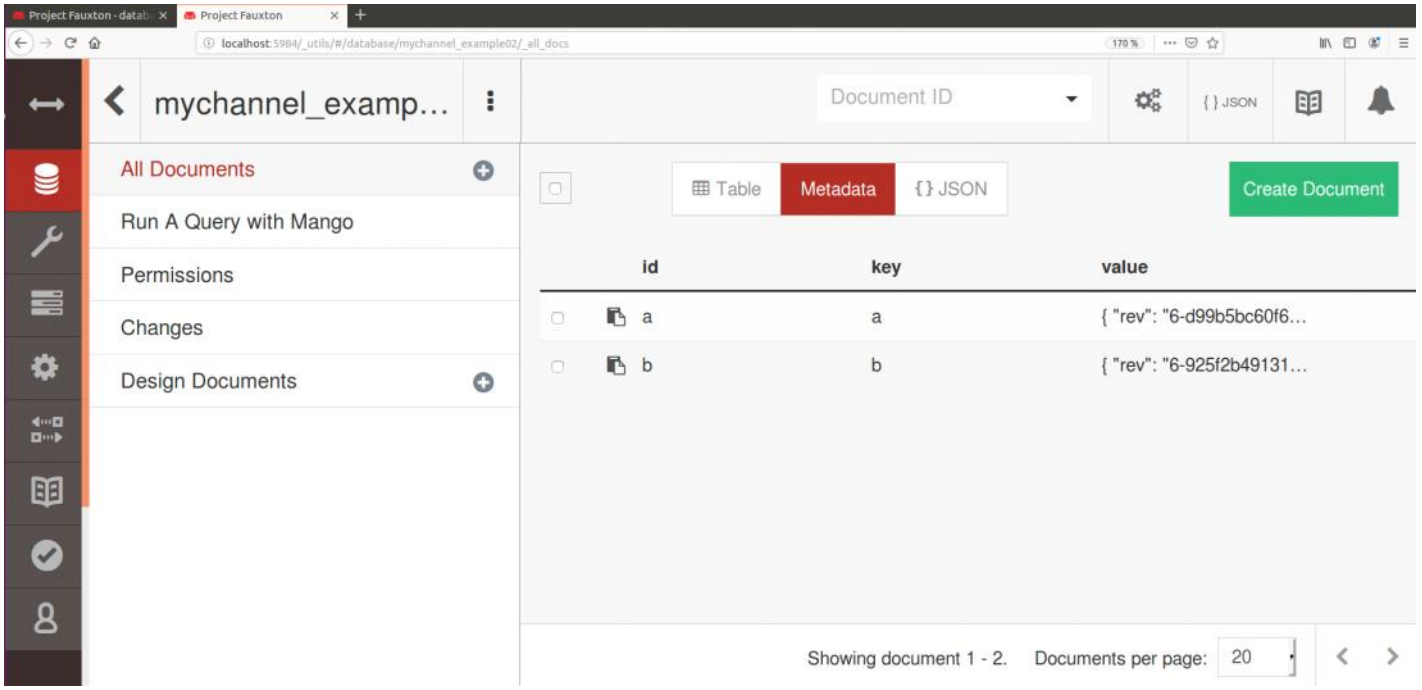
웹브라우저에 localhost 지정된 포트번호로 접속하여

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)
```