네트워크 구축 실습3-1 TLS

Org1 에 peer 2개, Org2에 peer 2개를 가진 네트워크 구축
couchdb 사용
TLS 사용
gossip 프로토콜 사용
전체 스크립트는 basic-network3-1-TLS.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

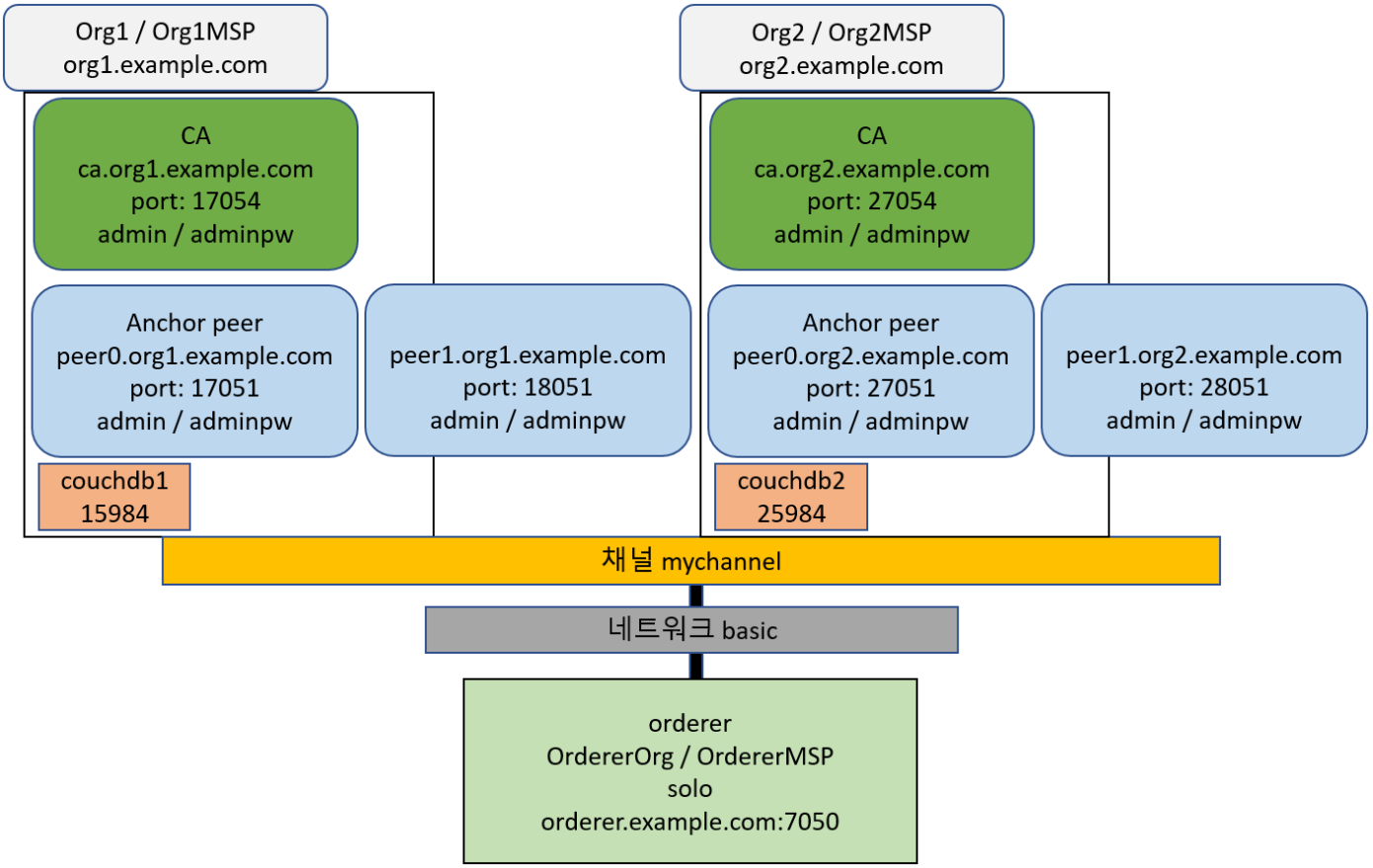
주소 및 포트:

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-network3 을 basic-network3-1-TLS 로 복사한다.

cp -r basic-network3 basic-network3-1-TLS

cd basic-network3-1-TLS

2)docker-compose.yaml 수정

a. ca-base 섹션에서 TLS 사용하도록 설정

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

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

command : 란에 --ca.certfile, --ca.keyfile 의 경로를 설정

--ca.certfile : 고정, --ca.keyfile : generate.sh 실행하면 crypto-config 이 변경되므로 수정 필요

```
ca-base:
  image: hyperledger/fabric-ca
  environment:
    - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
    - FABRIC_CA_SERVER_TLS_ENABLED=true
  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/63d78d3391935effc6e6e678d7da9cc4358112b15926422cf2fd16cf6f4de4ae_sk
  ports:
    - "17054:7054"
  command: sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.org1.example.com-cert.pem --ca.keyfile
/etc/hyperledger/fabric-ca-server-config/63d78d3391935effc6e6e678d7da9cc4358112b15926422cf2fd16cf6f4de4ae_sk -b admin:adminpw -d'
  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/3b0bc2a09e7081cfa2c05504b9fc98fc99194f71f81a1054f3444480815d95ac_sk
  ports:
    - "27054:7054"
  command: sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.org2.example.com-cert.pem --ca.keyfile
/etc/hyperledger/fabric-ca-server-config/3b0bc2a09e7081cfa2c05504b9fc98fc99194f71f81a1054f3444480815d95ac_sk -b admin:adminpw -d'
  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

g. TLS 관련 키를 추가해줘야 함 : TLS_CERT_FILE, TLS_KEY_FILE, TLS_ROOTCERT_FILE

   peer-base의 volume 과 각 peer 들의 volume 에 경로 추가 필요

```
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_PEER_TLS_ENABLED=true
      - CORE_PEER_PROFILE_ENABLED=true
      - CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt
      - CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key
      - CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt

      - 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
      - ./crypto-config:/etc/hyperledger/crypto-config
    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/peers/peer0.org1.example.com/tls:/etc/hyperledger/fabric/tls
        - ./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/peers/peer1.org1.example.com/tls:/etc/hyperledger/fabric/tls
      - ./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/org1.example.com/peers/peer0.org2.example.com/tls:/etc/hyperledger/fabric/tls
      - ./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/org1.example.com/peers/peer1.org2.example.com/tls:/etc/hyperledger/fabric/tls
      - ./crypto-config/peerOrganizations/org2.example.com/users:/etc/hyperledger/msp/users
  depends_on:
    - orderer.example.com
    - couchdb2
```

h. cli 수정하고 복사

i. 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

TLS 관련 키를 추가해줘야 함 : TLS_CERT_FILE, TLS_KEY_FILE, TLS_ROOTCERT_FILE

cli-base의 volume 에 경로 추가 필요

```
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:/etc/hyperledger/crypto-config
      - ./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/Ad min@org1.exampl
e.com/msp
      # Enable TLS
      - CORE_PEER_TLS_ENABLED=true
      -
CORE_PEER_TLS_CERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/pe er0.org1.example.com/tl
s/server.crt
      -
CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/pee r0.org1.example.com/tls
/server.key
      -
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peer s/peer0.org1.example.
com/tls/ca.crt

  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

     # Enable TLS
     - CORE_PEER_TLS_ENABLED=true
     -
CORE_PEER_TLS_CERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/server.crt
     -
CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/server.key
     -
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt
    depends_on:
     - orderer.example.com
     - peer0.org2.example.com
     - couchdb2
```

3) 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
```

TLS 통신 위해 --tls --cafile 설정이 필요한데 이를 환경변수로 지정해둔다.

체인코드 생성에 관련된 명령에는 --tls --cafile "$ORDERER_CA" 를 추가해줘야 에러 없이 TLS통신이 이뤄진다.

```
ORDERER_CA=/etc/hyperledger/crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
```

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 "$CHANNEL_NAME" -f /etc/hyperledger/configtx/Org1MSPanchors.tx --tls --cafile "$ORDERER_CA"
```

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 --tls --cafile "$ORDERER_CA"
```

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 "$CHANNEL_NAME".block --tls --cafile "$ORDERER_CA"
```

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 --tls --cafile "$ORDERER_CA"

# 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 "$CHANNEL_NAME".block --tls --cafile "$ORDERER_CA"

# 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 "$CHANNEL_NAME" -f /etc/hyperledger/configtx/Org2MSPanchors.tx --tls --cafile "$ORDERER_CA"

#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 --tls --cafile "$ORDERER_CA"

# 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 "$CHANNEL_NAME".block --tls --cafile "$ORDERER_CA"
```

4) 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
```

5) 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

6)체인코드 설치 및 실행

TLS 통신 위해 --tls --cafile 설정이 필요한데 이를 변수로 지정해둔다.

체인코드 생성에 관련된 명령에는 --tls --cafile "$ORDERER_CA" 를 추가해줘야 에러 없이 TLS통신이 이뤄진다.

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
ORDERER_CA=/etc/hyperledger/crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

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')"  --tls --cafile "$ORDERER_CA"
sleep 5
docker exec  peer0.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'
docker exec  peer0.org1.example.com peer chaincode invoke  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["set","a","130"]}'  --tls --cafile
"$ORDERER_CA"
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"]}'  --tls --cafile "$ORDERER_CA"
sleep 5
docker exec  peer1.org1.example.com peer chaincode query  -C "$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"]}'  --tls --cafile "$ORDERER_CA"
sleep 5
docker exec  peer0.org2.example.com peer chaincode query  -C "$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 "$CC_NAME" -c '{"Args":["set","a","170"]}'  --tls --cafile "$ORDERER_CA"
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
```

7)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=$(date +%s)
CHANNEL_NAME=mychannel
CC_RUNTIME_LANGUAGE=golang
CC_SRC_PATH=github.com/chaincode_example02/go
CC_NAME=example02
CC_VERSION=1.0
ORDERER_CA=/etc/hyperledger/crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
docker-compose -f ./docker-compose.yml up -d cli_org1 cli_org2
docker ps -a
# Org1
echo ===================================================================
echo        Org1
echo ===================================================================
echo  install chaincode to peer0.org1.example.com
#install chaincode to peer0.org1.example.com - 각 endoser peer에 모두 설치
docker exec  cli_org1 peer chaincode install -n "$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 "$CC_VERSION" -c '{"Args":["init","a","100","b","200"]}' -P "OR ('Org1MSP.member','Org2MSP.member')"  --tls --cafile "$ORDERER_CA"
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"]}'  --tls --cafile "$ORDERER_CA"
sleep 5
docker exec  peer0.org1.example.com peer chaincode query  -C "$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"]}'  --tls --cafile
"$ORDERER_CA"
sleep 5
docker exec  peer1.org1.example.com peer chaincode query  -C "$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"]}'  --tls --cafile "$
ORDERER_CA
"
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"]}'  --tls --cafile
"$ORDERER_CA"
sleep 5
docker exec  peer1.org2.example.com peer chaincode query  -C "$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
```