네트워크 구축 실습3-2 Kafka

Org1 에 peer 2개, Org2에 peer 2개를 가진 네트워크 구축
couchdb 사용
TLS 사용
orderer 2, kakfa 4 , zookeeper3 사용
gossip 프로토콜 사용
전체 스크립트는 basic-network3-2-Kafka.tar 참조

1. 네트워크 개요 정리

Organization수: 2

Channel

채널수: 1

채널이름: mychannel

Orderer

Orderer수 : 1 Concensus 방식: kafka

주소 및 포트: orderer0.example.com:7050 orderer1.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

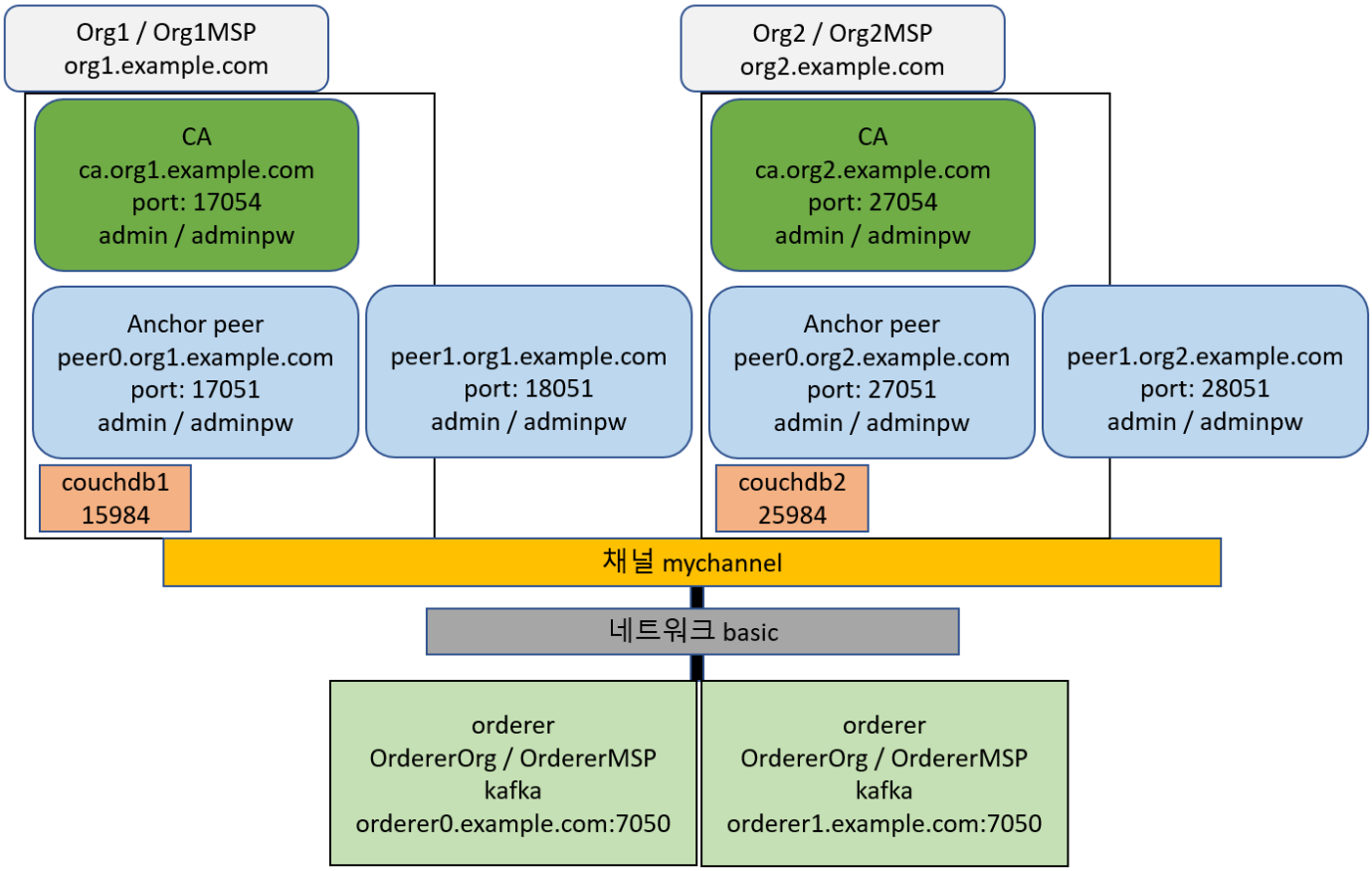
주소 및 포트:

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

counchdb

주소 및 포트: couchdb1 : 15984 / couchdb2 : 25984

1. 네트워크 스펙 정리



2. 네트워크 작성하기

1) basic-network3-1 을 basic-network3-2-Kafka 로 복사한다.

cp -r basic-network3-1 basic-network3-2-Kafka

cd basic-network3-2-Kafka

2) configtx.yaml 수정

Oredere SECTION 에서 원하는 만큼의 orderer와 kakfa 를 추가해준다.

일반적으로 kakfa 는 4.6.8.10 식으로 짝수에서 선택한다고 합니다.

```
Orderer: &OrdererDefaults

    # Orderer Type: The orderer implementation to start
    # Available types are "solo" and "kafka"
    OrdererType: kafka

    Addresses:
        - orderer0.example.com:7050
        - orderer1.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:
            - kafka0.example.com:9092
            - kafka1.example.com:9092
            - kafka2.example.com:9092
            - kafka3.example.com:9092

    # Organizations is the list of orgs which are defined as participants on
    # the orderer side of the network
    Organizations:
```

3)crypto-config.yaml 수정

OrdererOrgs의 specs 에 orderer 원하는 만큼 추가해준다

```
OrdererOrgs:
  # ---------------------------------------------------------------------------
  # Orderer
  # ---------------------------------------------------------------------------
  - Name: Orderer
    Domain: example.com
    # ---------------------------------------------------------------------------
    # "Specs" - See PeerOrgs below for complete description
    # ---------------------------------------------------------------------------
    Specs:
      - Hostname: orderer0
      - Hostname: orderer1
```

4)docker-compose.yaml 수정

a. kafka 정보를 4개 입력

```
  kafka0.example.com:
    container_name: kafka0.example.com
    image: hyperledger/fabric-kafka:latest
    restart: always
    environment:
      - KAFKA_BROKER_ID=1
      - KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://kafka0.example.com:9092
      - KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1
      - KAFKA_MESSAGE_MAX_BYTES=1048576 # 1 * 1024 * 1024 B
      - KAFKA_REPLICA_FETCH_MAX_BYTES=1048576 # 1 * 1024 * 1024 B
      - KAFKA_UNCLEAN_LEADER_ELECTION_ENABLE=false
      - KAFKA_LOG_RETENTION_MS=-1
      - KAFKA_MIN_INSYNC_REPLICAS=1
      - KAFKA_DEFAULT_REPLICATION_FACTOR=1
      - KAFKA_ZOOKEEPER_CONNECT=zookeeper0:2181,zookeeper1:2181,
    ports:
      - 19092:9092
    depends_on:
      - zookeeper0
      - zookeeper1
      - zookeeper2
    networks:
      - basic

  kafka1.example.com:
    container_name: kafka1.example.com
    image: hyperledger/fabric-kafka:latest
    restart: always
    environment:
      - KAFKA_BROKER_ID=2
      - KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://kafka1.example.com:9092
      - KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1
      - KAFKA_MESSAGE_MAX_BYTES=1048576 # 1 * 1024 * 1024 B
      - KAFKA_REPLICA_FETCH_MAX_BYTES=1048576 # 1 * 1024 * 1024 B
      - KAFKA_UNCLEAN_LEADER_ELECTION_ENABLE=false
      - KAFKA_LOG_RETENTION_MS=-1
      - KAFKA_MIN_INSYNC_REPLICAS=1
      - KAFKA_DEFAULT_REPLICATION_FACTOR=1
      - KAFKA_ZOOKEEPER_CONNECT=zookeeper0:2181,zookeeper1:2181,
    ports:
      - 29092:9092
    depends_on:
      - zookeeper0
```

```
    - zookeeper1
    - zookeeper2
  networks:
    - basic

kafka2.example.com:
  container_name: kafka2.example.com
  image: hyperledger/fabric-kafka:latest
  restart: always
  environment:
    - KAFKA_BROKER_ID=3
    - KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://kafka2.example.com:9092
    - KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1
    - KAFKA_MESSAGE_MAX_BYTES=1048576 # 1 * 1024 * 1024 B
    - KAFKA_REPLICA_FETCH_MAX_BYTES=1048576 # 1 * 1024 * 1024 B
    - KAFKA_UNCLEAN_LEADER_ELECTION_ENABLE=false
    - KAFKA_LOG_RETENTION_MS=-1
    - KAFKA_MIN_INSYNC_REPLICAS=1
    - KAFKA_DEFAULT_REPLICATION_FACTOR=1
    - KAFKA_ZOOKEEPER_CONNECT=zookeeper0:2181,zookeeper1:2181,
  ports:
    - 39092:9092
  depends_on:
    - zookeeper0
    - zookeeper1
    - zookeeper2
  networks:
    - basic

kafka3.example.com:
  container_name: kafka3.example.com
  image: hyperledger/fabric-kafka:latest
  restart: always
  environment:
    - KAFKA_BROKER_ID=4
    - KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://kafka3.example.com:9092
    - KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1
    - KAFKA_MESSAGE_MAX_BYTES=1048576 # 1 * 1024 * 1024 B
    - KAFKA_REPLICA_FETCH_MAX_BYTES=1048576 # 1 * 1024 * 1024 B
    - KAFKA_UNCLEAN_LEADER_ELECTION_ENABLE=false
    - KAFKA_LOG_RETENTION_MS=-1
    - KAFKA_MIN_INSYNC_REPLICAS=1
    - KAFKA_DEFAULT_REPLICATION_FACTOR=1
    - KAFKA_ZOOKEEPER_CONNECT=zookeeper0:2181,zookeeper1:2181,
  ports:
    - 49092:9092
  depends_on:
    - zookeeper0
    - zookeeper1
    - zookeeper2
  networks:
    - basic
```

b. zookeeper 정보를 3개 입력

일반적으로 zookeeper 는 3,5,7,9 식으로 홀수에서 선택한다고 합니다.

```
zookeeper0:
  container_name: zookeeper0
  image: hyperledger/fabric-zookeeper:latest
  restart: always
  environment:
    ZOOKEEPER_CLIENT_PORT: 32181
    ZOOKEEPER_TICK_TIME: 2000
  ports:
    - 12181:2181
  networks:
    - basic

zookeeper1:
  container_name: zookeeper1
  image: hyperledger/fabric-zookeeper:latest
  restart: always
  environment:
    ZOOKEEPER_CLIENT_PORT: 32181
    ZOOKEEPER_TICK_TIME: 2000
  ports:
    - 22181:2181
  networks:
    - basic
```

```
  zookeeper2:
    container_name: zookeeper2
    image: hyperledger/fabric-zookeeper:latest
    restart: always
    environment:
      ZOOKEEPER_CLIENT_PORT: 32181
      ZOOKEEPER_TICK_TIME: 2000
    ports:
      - 32181:2181
    networks:
      - basic
```

c. orderer 정보를 2개 입력

```
  orderer0.example.com:
    container_name: orderer0.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
      # Enable TLS
      - ORDERER_GENERAL_TLS_ENABLED=true
      - ORDERER_GENERAL_TLS_PRIVATEKEY=/etc/hyperledger/msp/orderer/tls/server.key
      - ORDERER_GENERAL_TLS_CERTIFICATE=/etc/hyperledger/msp/orderer/tls/server.crt
      - ORDERER_GENERAL_TLS_ROOTCAS=[/etc/hyperledger/msp/orderer/tls/ca.crt]
      - ORDERER_KAFKA_RETRY_LONGINTERVAL=10s
      - ORDERER_KAFKA_RETRY_LONGTOTAL=100s
      - ORDERER_KAFKA_RETRY_SHORTINTERVAL=1s
      - ORDERER_KAFKA_RETRY_SHORTTOTAL=30s
      - ORDERER_KAFKA_VERBOSE=false
    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/orderer0.example.com/:/etc/hyperledger/msp/orderer
        - ./crypto-
config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/:/etc/hyperledger/msp/peerOrg1
    networks:
      - basic

  orderer1.example.com:
    container_name: orderer1.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
      # Enable TLS
      - ORDERER_GENERAL_TLS_ENABLED=true
      - ORDERER_GENERAL_TLS_PRIVATEKEY=/etc/hyperledger/msp/orderer/tls/server.key
      - ORDERER_GENERAL_TLS_CERTIFICATE=/etc/hyperledger/msp/orderer/tls/server.crt
      - ORDERER_GENERAL_TLS_ROOTCAS=[/etc/hyperledger/msp/orderer/tls/ca.crt]
      - ORDERER_KAFKA_RETRY_LONGINTERVAL=10s
      - ORDERER_KAFKA_RETRY_LONGTOTAL=100s
      - ORDERER_KAFKA_RETRY_SHORTINTERVAL=1s
      - ORDERER_KAFKA_RETRY_SHORTTOTAL=30s
      - ORDERER_KAFKA_VERBOSE=false
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/orderer
    command: orderer
    ports:
      - 8050:7050
    volumes:
        - ./config/:/etc/hyperledger/configtx
        - ./crypto-config/ordererOrganizations/example.com/orderers/orderer1.example.com/:/etc/hyperledger/msp/orderer
        - ./crypto-
config/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/:/etc/hyperledger/msp/peerOrg2
    networks:
      - basic
```

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

--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, peer0.org1.example.com: depends_on 에 orderer0.example.com으로 수정
peer0.org2.example.com, peer0.org2.example.com: depends_on 에 orderer1.example.com으로 수정

```
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:
    - orderer0.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:
    - orderer0.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/peers/peer0.org2.example.com/tls:/etc/hyperledger/fabric/tls
      - ./crypto-config/peerOrganizations/org2.example.com/users:/etc/hyperledger/msp/users
  depends_on:
    - orderer1.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/peers/peer1.org2.example.com/tls:/etc/hyperledger/fabric/tls
        - ./crypto-config/peerOrganizations/org2.example.com/users:/etc/hyperledger/msp/users
    depends_on:
      - orderer1.example.com
      - couchdb2
```

d. cli.org1: depends_on 에 orderer0.example.com으로 수정

cli.org2: depends_on 에 orderer1.example.com으로 수정

```
 cli_org1:
   extends:
     service: cli-base
   container_name: cli_org1
   environment:
     - CORE_PEER_ID=cli_org1
     - CORE_PEER_ADDRESS=peer0.org1.example.com:17051
     - 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
     # 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/peer0.org1.example.com/tls/server.crt
     -
CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/server.key
     -
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
   depends_on:
     - orderer0.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:27051
     - 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:
     - orderer1.example.com
     - peer0.org2.example.com
     - couchdb2
```

3) start.sh 수정

상단 수정 - orderer0.example.com orderer1.example.com kafka0.example.com kafka1.example.com

kafka2.example.com kafka3.example.com zookeeper0 zookeeper1 zookeeper2 추가

```
docker-compose -f docker-compose.yml up -d ₩
ca.org1.example.com ca.org2.example.com ₩
orderer0.example.com orderer1.example.com ₩
peer0.org1.example.com peer1.org1.example.com couchdb1 ₩
peer0.org2.example.com peer1.org2.example.com couchdb2 ₩
kafka0.example.com kafka1.example.com ₩
kafka2.example.com kafka3.example.com ₩
zookeeper0 zookeeper1 zookeeper2
```

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 orderer0.example.com orderer1.example.com kafka0.example.com kafka1.example.com kafka2.example.com
kafka3.example.com zookeeper0 zookeeper1 zookeeper2
```

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

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/orderer0.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 orderer0.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 "$ORDERER_CA"
sleep 3
# docker exec  cli peer chaincode invoke -o orderer0.example.com:7050 -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["get","a"]}'
# docker exec  cli peer chaincode invoke -o orderer0.example.com:7050 -C "$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 "$CC_NAME" -c '{"Args":["get","a"]}'
docker exec  peer0.org1.example.com peer chaincode invoke -o orderer0.example.com:7050  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["set","a","130"]}'  --tls --cafile "$ORDERER_CA"
sleep 3
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"]}'
sleep 3
docker exec  peer1.org1.example.com peer chaincode invoke -o orderer0.example.com:7050  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["set","a","150"]}'   --tls --cafile "$ORDERER_CA"
sleep 3
docker exec  peer1.org1.example.com peer chaincode query  -C "$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 3
docker exec  peer0.org2.example.com peer chaincode invoke -o orderer0.example.com:7050   -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["set","a","160"]}'  --tls --cafile "$ORDERER_CA"
sleep 3
docker exec  peer0.org2.example.com peer chaincode query  -C "$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 3
docker exec  peer1.org2.example.com peer chaincode invoke -o orderer0.example.com:7050  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["set","a","170"]}'  --tls --cafile "$ORDERER_CA"
sleep 3
docker exec  peer1.org2.example.com peer chaincode query  -C "$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/orderer0.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 orderer0.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 3

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  -o orderer0.example.com:7050  -C "$CHANNEL_NAME" -n "$CC_NAME" -c
'{"Args":["invoke","a","b","10"]}'  --tls --cafile "$ORDERER_CA"
sleep 3
docker exec  peer0.org1.example.com peer chaincode query  -C "$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 "$CC_NAME" -c '{"Args":["query","a"]}'
sleep 3
docker exec  peer1.org1.example.com peer chaincode query  -C "$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["query","b"]}'
docker exec  peer1.org1.example.com peer chaincode invoke  -o orderer0.example.com:7050  -C "$CHANNEL_NAME" -n "$CC_NAME" -c
'{"Args":["invoke","a","b","5"]}'  --tls --cafile "$ORDERER_CA"
sleep 3
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  -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 3
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 orderer0.example.com:7050  -C
"$CHANNEL_NAME" -n "$CC_NAME" -c '{"Args":["invoke","a","b","5"]}'
docker exec  peer0.org2.example.com peer chaincode invoke  -o orderer0.example.com:7050  -C "$CHANNEL_NAME" -n "$CC_NAME" -c
'{"Args":["invoke","a","b","5"]}'  --tls --cafile "$ORDERER_CA"
sleep 3
docker exec  peer0.org2.example.com peer chaincode query  -C "$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 3
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  -o orderer0.example.com:7050  -C "$CHANNEL_NAME" -n "$CC_NAME" -c
'{"Args":["invoke","a","b","5"]}'  --tls --cafile "$ORDERER_CA"
sleep 3
docker exec  peer1.org2.example.com peer chaincode query  -C "$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
```