Case1

✓ 참고

구성

gcp host	docker container 구성
org1-a	orderer(x3), kafka(x4), zookepper(x3)
org1-b	org1(peer0, peer1), org2(peer0, peer1)
공통적용	docker ip, docker-compose ip 변경 peer data 백업

- kafka backup 볼륨 설정? → "/tmp/kafka-logs kafka" 컨테이너 내부에 이경로 맞나??
- 왜 오더러 컨테이너에는 아무것도 없지? → "docker exec -it orderer0.example.com //bin/sh" 명령어로 접속

설정파일 정의

기본 설정

- 폴더 생성

```
$ cd ~/fabric-samples
$ mkdir case1
$ cd case1
```

- docker compose파일에서 활용할 .env 파일 작성

```
$ vi .env
```

.env

```
ORG1A=35.243.100.147
ORG1B=35.243.117.220
```

crypto-config.yaml 파일

- cryptoconfig

```
$ vi crypto-config.yaml
```

crypto-config.yaml

```
OrdererOrgs:
  # ---------------------------------------------------------------------------
  # Orderer
  # ---------------------------------------------------------------------------
  - Name: Orderer
    Domain: example.com
    Template:
      Count: 3
PeerOrgs:
  - Name: Org1
    Domain: org1.example.com
    EnableNodeOUs: true
    Template:
      Count: 2
    Users:
      Count: 1
  - Name: Org2
    Domain: org2.example.com
    EnableNodeOUs: true
    Template:
      Count: 2
    Users:
      Count: 1
```

Configtx.yaml 파일

- configtx

```
$ vi configtx.yaml
```

configtx.yaml

```
Organizations:
    - &OrdererOrg
        Name: OrdererOrg
        ID: OrdererMSP
        MSPDir: crypto-config/ordererOrganizations/example.com/msp
        Policies:
            Readers:
                Type: Signature
                Rule: "OR('OrdererMSP.member')"
            Writers:
                Type: Signature
                Rule: "OR('OrdererMSP.member')"
            Admins:
                Type: Signature
                Rule: "OR('OrdererMSP.admin')"

    - &Org1
        Name: Org1MSP
        ID: Org1MSP
        MSPDir: crypto-config/peerOrganizations/org1.example.com/msp
        Policies:
            Readers:
                Type: Signature
                Rule: "OR('Org1MSP.admin', 'Org1MSP.peer', 'Org1MSP.client')"
            Writers:
                Type: Signature
                Rule: "OR('Org1MSP.admin', 'Org1MSP.client')"
            Admins:
                Type: Signature
                Rule: "OR('Org1MSP.admin')"
            Endorsement:
                Type: Signature
                Rule: "OR('Org1MSP.peer')"

        AnchorPeers:
            - Host: peer0.org1.example.com
              Port: 7051

    - &Org2
        Name: Org2MSP
        ID: Org2MSP
        MSPDir: crypto-config/peerOrganizations/org2.example.com/msp
        Policies:
            Readers:
                Type: Signature
                Rule: "OR('Org2MSP.admin', 'Org2MSP.peer', 'Org2MSP.client')"
            Writers:
                Type: Signature
                Rule: "OR('Org2MSP.admin', 'Org2MSP.client')"
            Admins:
                Type: Signature
                Rule: "OR('Org2MSP.admin')"
            Endorsement:
                Type: Signature
                Rule: "OR('Org2MSP.peer')"

        AnchorPeers:
            - Host: peer0.org2.example.com
              Port: 7051

Capabilities:
    Channel: &ChannelCapabilities
        V1_3: true
    Orderer: &OrdererCapabilities
        V1_1: true
    Application: &ApplicationCapabilities
        V2_0: true
        V1_3: false
        V1_2: false
        V1_1: false

Application: &ApplicationDefaults
    Organizations:
    Policies:
        Readers:
            Type: ImplicitMeta
            Rule: "ANY Readers"
        Writers:
            Type: ImplicitMeta
            Rule: "ANY Writers"
        Admins:
            Type: ImplicitMeta
            Rule: "MAJORITY Admins"
        LifecycleEndorsement:
            Type: ImplicitMeta
            Rule: "MAJORITY Endorsement"
        Endorsement:
            Type: ImplicitMeta
            Rule: "MAJORITY Endorsement"

    Capabilities:
        <<: *ApplicationCapabilities

Orderer: &OrdererDefaults
```

```
    OrdererType: kafka
    Addresses:
        - orderer0.example.com:7050
        - orderer1.example.com:7050
        - orderer2.example.com:7050
    BatchTimeout: 2s
    BatchSize:
        MaxMessageCount: 10
        AbsoluteMaxBytes: 99 MB
        PreferredMaxBytes: 512 KB
    Kafka:
        Brokers:
            - kafka0:9092
            - kafka1:9092
            - kafka2:9092
            - kafka3:9092
    Organizations:
    Policies:
        Readers:
            Type: ImplicitMeta
            Rule: "ANY Readers"
        Writers:
            Type: ImplicitMeta
            Rule: "ANY Writers"
        Admins:
            Type: ImplicitMeta
            Rule: "MAJORITY Admins"
        BlockValidation:
            Type: ImplicitMeta
            Rule: "ANY Writers"

Channel: &ChannelDefaults
    Policies:
        Readers:
            Type: ImplicitMeta
            Rule: "ANY Readers"
        Writers:
            Type: ImplicitMeta
            Rule: "ANY Writers"
        Admins:
            Type: ImplicitMeta
            Rule: "MAJORITY Admins"
    Capabilities:
        <<: *ChannelCapabilities

Profiles:

    TwoOrgsOrdererGenesis:
        <<: *ChannelDefaults
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
            Capabilities:
                <<: *OrdererCapabilities
        Consortiums:
            SampleConsortium:
                Organizations:
                    - *Org1
                    - *Org2
    OneOrgsChannel:
        Consortium: SampleConsortium
        <<: *ChannelDefaults
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
            Capabilities:
                <<: *ApplicationCapabilities

    TwoOrgsChannel:
        Consortium: SampleConsortium
        <<: *ChannelDefaults
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org2
            Capabilities:
                <<: *ApplicationCapabilities

    AllOrgsChannel:
        Consortium: SampleConsortium
        <<: *ChannelDefaults
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
                - *Org2
            Capabilities:
                <<: *ApplicationCapabilities
```

Docker-compose 파일

- docker-compse base파일 작성

```
$ vi docker-compose-base.yaml
```

docker-compose-base.yaml

```
version: "2"

services:
  zookeeper:
    image: hyperledger/fabric-zookeeper

  kafka:
    image: hyperledger/fabric-kafka
    environment:
      - KAFKA_LOG_RETENTION_MS=-1
      - KAFKA_MESSAGE_MAX_BYTES=103809024
      - KAFKA_REPLICA_FETCH_MAX_BYTES=103809024
      - KAFKA_UNCLEAN_LEADER_ELECTION_ENABLE=false
      - KAFKA_DEFAULT_REPLICATION_FACTOR=3
      - KAFKA_MIN_INSYNC_REPLICAS=2

  orderer:
    image: hyperledger/fabric-orderer
    environment:
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=case1_fabric
      - ORDERER_HOME=/var/hyperledger/orderer
      - ORDERER_GENERAL_LOGLEVEL=debug
      - ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/msp
      - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
      - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
      - ORDERER_GENERAL_LISTENPORT=7050
      - ORDERER_GENERAL_LEDGERTYPE=ram
      - ORDERER_GENERAL_GENESISMETHOD=file
      - ORDERER_GENERAL_GENESISFILE=/var/hyperledger/configs/genesis.block
      - CONFIGTX_ORDERER_ORDERERTYPE=solo
      - CONFIGTX_ORDERER_BATCHSIZE_MAXMESSAGECOUNT=10
      - CONFIGTX_ORDERER_BATCHTIMEOUT=2s
      - CONFIGTX_ORDERER_ADDRESSES=[127.0.0.1:7050]
      # TLS settings
      - ORDERER_GENERAL_TLS_ENABLED=true
      - ORDERER_GENERAL_TLS_PRIVATEKEY=/var/hyperledger/tls/server.key
      - ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/tls/server.crt
      - ORDERER_GENERAL_TLS_ROOTCAS=[/var/hyperledger/tls/ca.crt]
      - ORDERER_TLS_CLIENTAUTHREQUIRED=true
      - ORDERER_TLS_CLIENTROOTCAS_FILES=/var/hyperledger/users/Admin@example.com/tls/ca.crt
      - ORDERER_TLS_CLIENTCERT_FILE=/var/hyperledger/users/Admin@example.com/tls/client.crt
      - ORDERER_TLS_CLIENTKEY_FILE=/var/hyperledger/users/Admin@example.com/tls/client.key
    volumes:
      - ./channel-artifacts/:/var/hyperledger/configs
      - ./crypto-config/ordererOrganizations/example.com/users:/var/hyperledger/users
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/orderer
    command: orderer

  peer:
    image: hyperledger/fabric-peer
    environment:
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      # 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=case1_fabric
      - FABRIC_LOGGING_SPEC=INFO
      #- FABRIC_LOGGING_SPEC=DEBUG
      - CORE_PEER_TLS_ENABLED=true
      - CORE_PEER_GOSSIP_USELEADERELECTION=true
      - CORE_PEER_GOSSIP_ORGLEADER=false
      - 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
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
    command: peer node start
```

- org1, org2 peer docker compose 파일 작성

```
$ vi docker-compose-node.yaml
```

docker-compose-node.yaml

```
version: "2"

volumes:
  peer0.org1.example.com:
  peer1.org1.example.com:
  peer0.org2.example.com:
  peer1.org2.example.com:

networks:
  fabric:
    ipam:
      driver: default
      config:
        - subnet: 192.168.10.1/24

services:
  peer0.org1.example.com:
    container_name: peer0.org1.example.com
    extends:
      file: docker-compose-base.yaml
      service: peer
    environment:
      - CORE_PEER_ID=peer0.org1.example.com
      - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
      - CORE_PEER_LISTENADDRESS=0.0.0.0:7051
      - CORE_PEER_CHAINCODEADDRESS=peer0.org1.example.com:7052
      - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:7052
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer1.org1.example.com:8051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org1.example.com:7051
      - CORE_PEER_LOCALMSPID=Org1MSP
    extra_hosts:
      - "orderer0.example.com:${ORG1A}"
      - "orderer1.example.com:${ORG1A}"
      - "orderer2.example.com:${ORG1A}"

    volumes:
      - /var/run/:/host/var/run/
      - ./crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/msp:/etc
/hyperledger/fabric/msp
      - ./crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls:/etc
/hyperledger/fabric/tls
      - /home/sjpark/fabric-samples/case1/backup-org1-peer0:/var/hyperledger/production
    ports:
      - 7051:7051
    networks:
      - fabric

  peer1.org1.example.com:
    container_name: peer1.org1.example.com
    extends:
      file: docker-compose-base.yaml
      service: peer
    environment:
      - CORE_PEER_ID=peer1.org1.example.com
      - CORE_PEER_ADDRESS=peer1.org1.example.com:8051
      - CORE_PEER_LISTENADDRESS=0.0.0.0:8051
      - CORE_PEER_CHAINCODEADDRESS=peer1.org1.example.com:8052
      - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:8052
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org1.example.com:8051
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org1.example.com:7051
      - CORE_PEER_LOCALMSPID=Org1MSP
    extra_hosts:
      - "orderer0.example.com:${ORG1A}"
      - "orderer1.example.com:${ORG1A}"
      - "orderer2.example.com:${ORG1A}"
    volumes:
      - /var/run/:/host/var/run/
      - ./crypto-config/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/msp:/etc
/hyperledger/fabric/msp
      - ./crypto-config/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls:/etc
/hyperledger/fabric/tls
      - /home/sjpark/fabric-samples/case1/backup-org1-peer1:/var/hyperledger/production
    ports:
      - 8051:8051
    networks:
      - fabric

  peer0.org2.example.com:
    container_name: peer0.org2.example.com
    extends:
      file: docker-compose-base.yaml
      service: peer
    environment:
      - CORE_PEER_ID=peer0.org2.example.com
      - CORE_PEER_ADDRESS=peer0.org2.example.com:9051
      - CORE_PEER_LISTENADDRESS=0.0.0.0:9051
      - CORE_PEER_CHAINCODEADDRESS=peer0.org2.example.com:9052
      - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:9052
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer1.org2.example.com:10051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer0.org2.example.com:9051
      - CORE_PEER_LOCALMSPID=Org2MSP
    extra_hosts:
      - "orderer0.example.com:${ORG1A}"
      - "orderer1.example.com:${ORG1A}"
      - "orderer2.example.com:${ORG1A}"
    volumes:
      - /var/run/:/host/var/run/
      - ./crypto-config/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/msp:/etc
/hyperledger/fabric/msp
      - ./crypto-config/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls:/etc
/hyperledger/fabric/tls
      - /home/sjpark/fabric-samples/case1/backup-org2-peer0:/var/hyperledger/production
    ports:
      - 9051:9051
    networks:
      - fabric

  peer1.org2.example.com:
    container_name: peer1.org2.example.com
    extends:
      file: docker-compose-base.yaml
      service: peer
    environment:
      - CORE_PEER_ID=peer1.org2.example.com
      - CORE_PEER_ADDRESS=peer1.org2.example.com:10051
      - CORE_PEER_LISTENADDRESS=0.0.0.0:10051
      - CORE_PEER_CHAINCODEADDRESS=peer1.org2.example.com:10052
      - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:10052
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org2.example.com:10051
      - CORE_PEER_GOSSIP_BOOTSTRAP=peer0.org2.example.com:9051
      - CORE_PEER_LOCALMSPID=Org2MSP
    extra_hosts:
      - "orderer0.example.com:${ORG1A}"
      - "orderer1.example.com:${ORG1A}"
      - "orderer2.example.com:${ORG1A}"
    volumes:
      - /var/run/:/host/var/run/
      - ./crypto-config/peerOrganizations/org2.example.com/peers/peer1.org2.example.com/msp:/etc
/hyperledger/fabric/msp
      - ./crypto-config/peerOrganizations/org2.example.com/peers/peer1.org2.example.com/tls:/etc
/hyperledger/fabric/tls
      - /home/sjpark/fabric-samples/case1/backup-org2-peer1:/var/hyperledger/production
    ports:
      - 10051:10051
    networks:
      - fabric

  cli:
    container_name: cli
    image: hyperledger/fabric-tools
    tty: true
    stdin_open: true
    environment:
      - GOPATH=/opt/gopath
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      #- FABRIC_LOGGING_SPEC=DEBUG
      - FABRIC_LOGGING_SPEC=INFO
      - CORE_PEER_ID=cli
      - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
      - CORE_PEER_LOCALMSPID=Org1MSP
      - 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
      - CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
    extra_hosts:
      - "orderer0.example.com:${ORG1A}"
      - "orderer1.example.com:${ORG1A}"
      - "orderer2.example.com:${ORG1A}"
    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/chaincode
      - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
      - ./scripts:/opt/gopath/src/github.com/hyperledger/fabric/peer/scripts/
      - ./channel-artifacts:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel-artifacts
    depends_on:
      - peer0.org1.example.com
      - peer1.org1.example.com
      - peer0.org2.example.com
      - peer1.org2.example.com
    networks:
      - fabric
```

- orderer, kafka, zookeeper를 위한 docker compose 파일 작성

```
$ vi docker-compose-orderer.yaml
```

docker-compose-orderer.yaml

```
version: '2'

volumes:
    orderer0.example.com:
    orderer1.example.com:
    orderer2.example.com:

networks:
    fabric:
        ipam:
            driver: default
            config:
                - subnet: 192.168.10.1/24

services:
    zookeeper0:
        extends:
            file: docker-compose-base.yaml
            service: zookeeper
        container_name: zookeeper0
        environment:
            - ZOO_MY_ID=1
            - ZOO_SERVERS=server.1=zookeeper0:2888:3888 server.2=zookeeper1:2888:3888 server.
3=zookeeper2:2888:3888
        networks:
            - fabric

    zookeeper1:
        extends:
            file: docker-compose-base.yaml
            service: zookeeper
        container_name: zookeeper1
        environment:
            - ZOO_MY_ID=2
            - ZOO_SERVERS=server.1=zookeeper0:2888:3888 server.2=zookeeper1:2888:3888 server.
3=zookeeper2:2888:3888
        networks:
            - fabric

    zookeeper2:
        extends:
            file: docker-compose-base.yaml
            service: zookeeper
        container_name: zookeeper2
        environment:
            - ZOO_MY_ID=3
            - ZOO_SERVERS=server.1=zookeeper0:2888:3888 server.2=zookeeper1:2888:3888 server.
3=zookeeper2:2888:3888
        networks:
            - fabric

    kafka0:
        extends:
            file: docker-compose-base.yaml
            service: kafka
        container_name: kafka0
        environment:
            - KAFKA_BROKER_ID=0
            - KAFKA_ZOOKEEPER_CONNECT=zookeeper0:2181,zookeeper1:2181,zookeeper2:2181
            - KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://kafka0:9092
            - KAFKA_MESSAGE_MAX_BYTES=1000012 B
            - KAFKA_REPLICA_FETCH_MAX_BYTES=1048576 B
            - KAFKA_REPLICA_FETCH_RESPONSE_MAX_BYTES=10485760 B
        depends_on:
            - zookeeper0
            - zookeeper1
            - zookeeper2
        networks:
            - fabric

    kafka1:
        extends:
            file: docker-compose-base.yaml
            service: kafka
        container_name: kafka1
        environment:
            - KAFKA_BROKER_ID=1
            - KAFKA_ZOOKEEPER_CONNECT=zookeeper0:2181,zookeeper1:2181,zookeeper2:2181
            - KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://kafka1:9092
            - KAFKA_MESSAGE_MAX_BYTES=1000012 B
            - KAFKA_REPLICA_FETCH_MAX_BYTES=1048576 B
            - KAFKA_REPLICA_FETCH_RESPONSE_MAX_BYTES=10485760 B
        depends_on:
            - zookeeper0
            - zookeeper1
            - zookeeper2
        networks:
```

```
        - fabric

kafka2:
    extends:
        file: docker-compose-base.yaml
        service: kafka
    container_name: kafka2
    environment:
        - KAFKA_BROKER_ID=2
        - KAFKA_ZOOKEEPER_CONNECT=zookeeper0:2181,zookeeper1:2181,zookeeper2:2181
        - KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://kafka2:9092
        - KAFKA_MESSAGE_MAX_BYTES=1000012 B
        - KAFKA_REPLICA_FETCH_MAX_BYTES=1048576 B
        - KAFKA_REPLICA_FETCH_RESPONSE_MAX_BYTES=10485760 B
    depends_on:
        - zookeeper0
        - zookeeper1
        - zookeeper2
    networks:
        - fabric

kafka3:
    extends:
        file: docker-compose-base.yaml
        service: kafka
    container_name: kafka3
    environment:
        - KAFKA_BROKER_ID=3
        - KAFKA_ZOOKEEPER_CONNECT=zookeeper0:2181,zookeeper1:2181,zookeeper2:2181
        - KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://kafka3:9092
        - KAFKA_MESSAGE_MAX_BYTES=1000012 B
        - KAFKA_REPLICA_FETCH_MAX_BYTES=1048576 B
        - KAFKA_REPLICA_FETCH_RESPONSE_MAX_BYTES=10485760 B
    depends_on:
        - zookeeper0
        - zookeeper1
        - zookeeper2
    networks:
        - fabric


orderer0.example.com:
    extends:
        file: docker-compose-base.yaml
        service: orderer
    container_name: orderer0.example.com
    environment:
        - ORDERER_HOST=orderer0.example.com
        - CONFIGTX_ORDERER_ORDERERTYPE=kafka
        - CONFIGTX_ORDERER_KAFKA_BROKERS=[kafka0:9092,kafka1:9092,kafka2:9092,kafka3:9092]
        - ORDERER_KAFKA_RETRY_SHORTINTERVAL=1s
        - ORDERER_KAFKA_RETRY_SHORTTOTAL=30s
        - ORDERER_KAFKA_VERBOSE=true
        - ORDERER_GENERAL_GENESISPROFILE=SampleInsecureKafka
        - ORDERER_ABSOLUTEMAXBYTES=10 MB
        - ORDERER_PREFERREDMAXBYTES=512 KB
    extra_hosts:
        - "peer0.org1.example.com:${ORG1B}"
        - "peer1.org1.example.com:${ORG1B}"
        - "peer0.org2.example.com:${ORG1B}"
        - "peer1.org2.example.com:${ORG1B}"
    depends_on:
        - zookeeper0
        - zookeeper1
        - zookeeper2
        - kafka0
        - kafka1
        - kafka2
        - kafka3
    volumes:
        - ./crypto-config/ordererOrganizations/example.com/orderers/orderer0.example.com/msp:/var
```

```
/hyperledger/msp
            - ./crypto-config/ordererOrganizations/example.com/orderers/orderer0.example.com/tls:/var
/hyperledger/tls
            - ./channel-artifacts/:/var/hyperledger/configs
            - /home/sjpark/fabric-samples/case1/backup-orderer0:/var/hyperledger/production
        networks:
            - fabric
        ports:
          - 7050:7050

    orderer1.example.com:
        extends:
            file: docker-compose-base.yaml
            service: orderer
        container_name: orderer1.example.com
        environment:
            - ORDERER_HOST=orderer1.example.com
            - CONFIGTX_ORDERER_ORDERERTYPE=kafka
            - CONFIGTX_ORDERER_KAFKA_BROKERS=[kafka0:9092,kafka1:9092,kafka2:9092,kafka3:9092]
            - ORDERER_KAFKA_RETRY_SHORTINTERVAL=1s
            - ORDERER_KAFKA_RETRY_SHORTTOTAL=30s
            - ORDERER_KAFKA_VERBOSE=true
            - ORDERER_GENERAL_GENESISPROFILE=SampleInsecureKafka
            - ORDERER_ABSOLUTEMAXBYTES=10 MB
            - ORDERER_PREFERREDMAXBYTES=512 KB
        extra_hosts:
            - "peer0.org1.example.com:${ORG1B}"
            - "peer1.org1.example.com:${ORG1B}"
            - "peer0.org2.example.com:${ORG1B}"
            - "peer1.org2.example.com:${ORG1B}"
        depends_on:
            - zookeeper0
            - zookeeper1
            - zookeeper2
            - kafka0
            - kafka1
            - kafka2
            - kafka3
        volumes:
            - ./crypto-config/ordererOrganizations/example.com/orderers/orderer1.example.com/msp:/var
/hyperledger/msp
            - ./crypto-config/ordererOrganizations/example.com/orderers/orderer1.example.com/tls:/var
/hyperledger/tls
            - ./channel-artifacts/:/var/hyperledger/configs
            - /home/sjpark/fabric-samples/case1/backup-orderer1:/var/hyperledger/production
        networks:
            - fabric
        ports:
          - 8050:7050

    orderer2.example.com:
        extends:
            file: docker-compose-base.yaml
            service: orderer
        container_name: orderer2.example.com
        environment:
            - ORDERER_HOST=orderer2.example.com
            - CONFIGTX_ORDERER_ORDERERTYPE=kafka
            - CONFIGTX_ORDERER_KAFKA_BROKERS=[kafka0:9092,kafka1:9092,kafka2:9092,kafka3:9092]
            - ORDERER_KAFKA_RETRY_SHORTINTERVAL=1s
            - ORDERER_KAFKA_RETRY_SHORTTOTAL=30s
            - ORDERER_KAFKA_VERBOSE=true
            - ORDERER_GENERAL_GENESISPROFILE=SampleInsecureKafka
            - ORDERER_ABSOLUTEMAXBYTES=10 MB
            - ORDERER_PREFERREDMAXBYTES=512 KB
        extra_hosts:
            - "peer0.org1.example.com:${ORG1B}"
            - "peer1.org1.example.com:${ORG1B}"
            - "peer0.org2.example.com:${ORG1B}"
            - "peer1.org2.example.com:${ORG1B}"
        depends_on:
            - zookeeper0
            - zookeeper1
            - zookeeper2
            - kafka0
            - kafka1
            - kafka2
            - kafka3
        volumes:
            - ./crypto-config/ordererOrganizations/example.com/orderers/orderer2.example.com/msp:/var
/hyperledger/msp
            - ./crypto-config/ordererOrganizations/example.com/orderers/orderer2.example.com/tls:/var
/hyperledger/tls
            - ./channel-artifacts/:/var/hyperledger/configs
            - /home/sjpark/fabric-samples/case1/backup-orderer2:/var/hyperledger/production
        networks:
            - fabric
        ports:
          - 9050:7050
```

인증서 및 channel-artifacts 생성

- crypto-config 생성

```
$ ../bin/cryptogen generate --config=./crypto-config.yaml
```

- genesis.block 생성

```
$ mkdir channel-artifacts


$ ../bin/configtxgen -profile TwoOrgsOrdererGenesis -channelID test-channel -outputBlock ./channel-
artifacts/genesis.block
```

- 채널 트랜잭션 생성

```
$ ../bin/configtxgen -profile OneOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel1.tx -
channelID channel1

$ ../bin/configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel2.tx -
channelID channel2

$ ../bin/configtxgen -profile AllOrgsChannel -outputCreateChannelTx ./channel-artifacts/channelall.tx -
channelID channelall
```

- channelall에 대한 앵커피어 트랜잭션

```
$ ../bin/configtxgen -profile AllOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts
/Org1MSPanchors_channelall.tx -channelID channelall -asOrg Org1MSP

$ ../bin/configtxgen -profile AllOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts
/Org2MSPanchors_channelall.tx -channelID channelall -asOrg Org2MSP
```

생성 파일 전송 및 실행

준비

1. 압축

```
$ cd ~/fabric-samples

$ tar cf case1.tar case1/
```

2. 전송(scp or 파일질라 등 이용)
3. 압축 해제

```
$ tar xf case1.tar


$ cd case1
```

4. docker-compose 파일 volume 부분에 명시된 backup 폴더를 생성해준다
   - org1-a

   ```
   $ cd ~/fabric-samples/case1


   $ mkdir backup-orderer0 && mkdir backup-orderer1 && mkdir backup-orderer2
   ```

   - org1-b

   ```
   $ cd ~/fabric-samples/case1


   $ mkdir backup-org1-peer0 && mkdir backup-org1-peer1 && mkdir backup-org2-peer0 && mkdir backup-
   org2-peer1
   ```

5. docker up
   - org1-a

   ```
   $ docker-compose -f docker-compose-orderer.yaml up -d


   $ docker ps
   ```

   - org1-b

   ```
   $ docker-compose -f docker-compose-node.yaml up -d

   $ docker ps
   ```

channel1 생성 및 조인

1. 채널1 생성

```
$ docker exec -it cli bash

# peer channel create -o orderer0.example.com:7050 -c channel1 -f ./channel-artifacts/channel1.tx --tls
--cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com
/orderers/orderer0.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
```

2. 채널1 참여(peer0.org1)

```
# peer channel join -b channel1.block
```

3. 채널1 참여(peer1.org1)

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/users/Admin@org1.example.com/msp
CORE_PEER_ADDRESS=peer1.org1.example.com:8051
CORE_PEER_LOCALMSPID="Org1MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/peers/peer1.org1.example.com/tls/ca.crt

# peer channel join -b channel1.block
```

channel2 생성 및 조인

1. 채널2 생성

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer0.org2.example.com:9051
CORE_PEER_LOCALMSPID="Org2MSP"
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

# peer channel create -o orderer0.example.com:7050 -c channel2 -f ./channel-artifacts/channel2.tx --tls
--cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com
/orderers/orderer0.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
```

2. 채널2 참여(peer0.org2)

```
# peer channel join -b channel2.block
```

3. 채널2 참여(peer1.org2)

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer1.org2.example.com:10051
CORE_PEER_LOCALMSPID="Org2MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org2.example.com/peers/peer1.org2.example.com/tls/ca.crt

# peer channel join -b channel2.block
```

channelall 생성 및 조인

1. 채널all 생성

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/users/Admin@org1.example.com/msp
CORE_PEER_ADDRESS=peer0.org1.example.com:7051
CORE_PEER_LOCALMSPID="Org1MSP"
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

# peer channel create -o orderer0.example.com:7050 -c channelall -f ./channel-artifacts/channelall.tx --
tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com
/orderers/orderer0.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
```

2. 채널all 참여(peer0.org1)

```
# peer channel join -b channelall.block
```

3. 채널all 참여(peer1.org1)

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/users/Admin@org1.example.com/msp
CORE_PEER_ADDRESS=peer1.org1.example.com:8051
CORE_PEER_LOCALMSPID="Org1MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/peers/peer1.org1.example.com/tls/ca.crt

# peer channel join -b channelall.block
```

4. 채널all 참여(peer0.org2)

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer0.org2.example.com:9051
CORE_PEER_LOCALMSPID="Org2MSP"
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

# peer channel join -b channelall.block
```

5. 채널all 참여(peer1.org2)

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer1.org2.example.com:10051
CORE_PEER_LOCALMSPID="Org2MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org2.example.com/peers/peer1.org2.example.com/tls/ca.crt


# peer channel join -b channelall.block
```

앵커피어 업데이트(channelall)

1. peer0.org1

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/users/Admin@org1.example.com/msp
CORE_PEER_ADDRESS=peer0.org1.example.com:7051
CORE_PEER_LOCALMSPID="Org1MSP"
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

# peer channel update -o orderer0.example.com:7050 -c channelall -f ./channel-artifacts
/Org1MSPanchors_channelall.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/ordererOrganizations/example.com/orderers/orderer0.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
```

2. peer0.org2

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer0.org2.example.com:9051
CORE_PEER_LOCALMSPID="Org2MSP"
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

# peer channel update -o orderer0.example.com:7050 -c channelall -f ./channel-artifacts
/Org2MSPanchors_channelall.tx --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/ordererOrganizations/example.com/orderers/orderer0.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
```

체인코드 lifecycle

1. packaging chaincode

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/users/Admin@org1.example.com/msp
CORE_PEER_ADDRESS=peer0.org1.example.com:7051
CORE_PEER_LOCALMSPID="Org1MSP"
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

# peer lifecycle chaincode package mycc.tar.gz --path github.com/chaincode/abstore/go/ --lang golang --
label mycc_1
```

2. install chaincode

   - peer0.org1

```
# peer lifecycle chaincode install mycc.tar.gz

# peer lifecycle chaincode queryinstalled
Installed chaincodes on peer:
Package ID: mycc_1:1a5cee241429a822f8b7282a9d196217e54efcc49e122d8675dfca2e20ef82ca, Label: mycc_1

# CC_PACKAGE_ID=mycc_1:1a5cee241429a822f8b7282a9d196217e54efcc49e122d8675dfca2e20ef82ca
```

   - peer1.org1

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
CORE_PEER_ADDRESS=peer1.org1.example.com:8051
CORE_PEER_LOCALMSPID="Org1MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls/ca.crt

# peer lifecycle chaincode install mycc.tar.gz
```

   - peer0.org2

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer0.org2.example.com:9051
CORE_PEER_LOCALMSPID="Org2MSP"
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

# peer lifecycle chaincode install mycc.tar.gz
```

   - peer1.org2

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer1.org2.example.com:10051
CORE_PEER_LOCALMSPID="Org2MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/peerOrganizations/org2.example.com/peers/peer1.org2.example.com/tls/ca.crt

# peer lifecycle chaincode install mycc.tar.gz
```

3. Approve chaincode by each org

   - peer0.org1

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
CORE_PEER_ADDRESS=peer0.org1.example.com:7051
CORE_PEER_LOCALMSPID="Org1MSP"
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

# peer lifecycle chaincode approveformyorg --channelID channelall --name mycc --version 1.0 --init-
required --package-id $CC_PACKAGE_ID --sequence 1 --tls true --cafile /opt/gopath/src/github.com
/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer0.example.com/msp
/tlscacerts/tlsca.example.com-cert.pem
```

   - peer1.org1

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer0.org2.example.com:9051
CORE_PEER_LOCALMSPID="Org2MSP"
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

# peer lifecycle chaincode approveformyorg --channelID channelall --name mycc --version 1.0 --init-
required --package-id $CC_PACKAGE_ID --sequence 1 --tls true --cafile /opt/gopath/src/github.com
/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer0.example.com/msp
/tlscacerts/tlsca.example.com-cert.pem
```

4. Check approval status

```
# peer lifecycle chaincode queryapprovalstatus --channelID channelall --name mycc --version 1.0 --init-
required --sequence 1 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/ordererOrganizations/example.com/orderers/orderer0.example.com/msp/tlscacerts/tlsca.example.com-cert.pem



{
  "Approved": {
    "Org1MSP": true,
    "Org2MSP": true
  }
}
```

5. Commit chaincode

```
# peer lifecycle chaincode commit -o orderer0.example.com:7050 --channelID channelall --name mycc --
version 1.0 --sequence 1 --init-required --tls true --cafile /opt/gopath/src/github.com/hyperledger
/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer0.example.com/msp/tlscacerts/tlsca.
example.com-cert.pem --peerAddresses peer0.org1.example.com:7051 --tlsRootCertFiles /opt/gopath/src
/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.
com/tls/ca.crt --peerAddresses peer0.org2.example.com:9051 --tlsRootCertFiles /opt/gopath/src/github.com
/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.
crt
2019-08-04 15:22:26.531 UTC [chaincodeCmd] ClientWait -> INFO 001 txid
[644c9af077618db04de192c19520cbdd09f7e28b3202f2f7b123cb903c4441d6] committed with status (VALID) at
peer0.org2.example.com:9051

2019-08-04 15:22:26.532 UTC [chaincodeCmd] ClientWait -> INFO 002 txid
[644c9af077618db04de192c19520cbdd09f7e28b3202f2f7b123cb903c4441d6] committed with status (VALID) at
peer0.org1.example.com:7051
```

6. Invoke chaincode

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/users/Admin@org1.example.com/msp
CORE_PEER_ADDRESS=peer0.org1.example.com:7051
CORE_PEER_LOCALMSPID="Org1MSP"
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

# peer chaincode invoke -o orderer0.example.com:7050 --isInit --tls true --cafile /opt/gopath/src/github.
com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer0.example.com/msp
/tlscacerts/tlsca.example.com-cert.pem -C channelall -n mycc --peerAddresses peer0.org1.example.com:7051
--tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.
example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses peer0.org2.example.com:9051 --
tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.
example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"Args":["Init","a","100","b","500"]}' --
waitForEvent
```

7. 현재 까지 진행후 host 에서 docker ps → 체인코드 컨테이너 2개 떠있음

```
$ docker ps


CONTAINER ID
IMAGE
COMMAND                  CREATED              STATUS              PORTS
NAMES
86bd995ea6f1        dev-peer0.org2.example.com-mycc_1-
2e0c5cc7b69c44f57e492c42e87f2fa716b2de4870592cc3dcab267156f741d4-
3d561a680024c67c144d226ccc9eefbb197294c5aa86816d519efc4c481da5ff   "chaincode -peer.add…"   41 seconds
ago       Up 40 seconds                                            dev-peer0.org2.example.com-mycc_1-
2e0c5cc7b69c44f57e492c42e87f2fa716b2de4870592cc3dcab267156f741d4
2d8af6895971        dev-peer0.org1.example.com-mycc_1-
2e0c5cc7b69c44f57e492c42e87f2fa716b2de4870592cc3dcab267156f741d4-
aa451222e4c35eaa0f97de0ac9b553c9d7f3826d65a67869a27f1ec9bbc84ec9   "chaincode -peer.add…"   About a
minute ago   Up About a minute                                        dev-peer0.org1.example.com-mycc_1-
2e0c5cc7b69c44f57e492c42e87f2fa716b2de4870592cc3dcab267156f741d4
b4ea9e408108        hyperledger/fabric-
tools
"/bin/bash"              38 minutes ago       Up 38 minutes
cli
83d2e71727c8        hyperledger/fabric-
peer
"peer node start"        38 minutes ago       Up 38 minutes       7051/tcp, 0.0.0.0:9051->9051/tcp
peer0.org2.example.com
1042ac497c76        hyperledger/fabric-
peer
"peer node start"        38 minutes ago       Up 38 minutes       7051/tcp, 0.0.0.0:8051->8051/tcp
peer1.org1.example.com
48a57ed70232        hyperledger/fabric-
peer
"peer node start"        38 minutes ago       Up 38 minutes       0.0.0.0:7051->7051/tcp
peer0.org1.example.com
6c8bf43eabc3        hyperledger/fabric-
peer
"peer node start"        38 minutes ago       Up 38 minutes       7051/tcp, 0.0.0.0:10051->10051/tcp
peer1.org2.example.com
```

8. 다시 cli로 접속

```
$ docker exec -it cli bash
```

9. Query chaincode(peer0.org1)

```
# peer chaincode query -C channelall -n mycc -c '{"Args":["query","a"]}'
```

10. Query chaincode(peer1.org1) → 쿼리 하면 체인코드 인스턴스화 함

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/users/Admin@org1.example.com/msp
CORE_PEER_ADDRESS=peer1.org1.example.com:8051
CORE_PEER_LOCALMSPID="Org1MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org1.example.com/peers/peer1.org1.example.com/tls/ca.crt

# peer chaincode query -C channelall -n mycc -c '{"Args":["query","a"]}'
```

11. Query chaincode(peer0.org2)

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer0.org2.example.com:9051
CORE_PEER_LOCALMSPID="Org2MSP"
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



# peer chaincode query -C channelall -n mycc -c '{"Args":["query","a"]}'
```

12. Query chaincode(peer1.org2) → 쿼리 하면 체인코드 인스턴스화 함

```
# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org2.example.com/users/Admin@org2.example.com/msp
CORE_PEER_ADDRESS=peer1.org2.example.com:10051
CORE_PEER_LOCALMSPID="Org2MSP"
CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations
/org2.example.com/peers/peer1.org2.example.com/tls/ca.crt



# peer chaincode query -C channelall -n mycc -c '{"Args":["query","a"]}'
```

13. 다시 호스트에서 docker 컨테이너 보면 → 체인코드 4개 떠있음

```
$ docker ps


CONTAINER ID
IMAGE
COMMAND                  CREATED             STATUS              PORTS
NAMES
f58687cd1382        dev-peer1.org2.example.com-mycc_1-
2e0c5cc7b69c44f57e492c42e87f2fa716b2de4870592cc3dcab267156f741d4-
d9a4865dd26ba99d09a40dff9b9bf5e52e7af08747399545cf50fc9212435803   "chaincode -peer.add…"   19 seconds
ago      Up 18 seconds                                            dev-peer1.org2.example.com-mycc_1-
2e0c5cc7b69c44f57e492c42e87f2fa716b2de4870592cc3dcab267156f741d4
2df06b66857e        dev-peer1.org1.example.com-mycc_1-
2e0c5cc7b69c44f57e492c42e87f2fa716b2de4870592cc3dcab267156f741d4-
8bf09a1550fc71c9927d15930e724dd685eb2dff72633adfeca230da32a2c47a   "chaincode -peer.add…"   3 minutes
ago       Up 3 minutes                                             dev-peer1.org1.example.com-mycc_1-
2e0c5cc7b69c44f57e492c42e87f2fa716b2de4870592cc3dcab267156f741d4
86bd995ea6f1        dev-peer0.org2.example.com-mycc_1-
2e0c5cc7b69c44f57e492c42e87f2fa716b2de4870592cc3dcab267156f741d4-
3d561a680024c67c144d226ccc9eefbb197294c5aa86816d519efc4c481da5ff   "chaincode -peer.add…"   5 minutes
ago       Up 5 minutes                                             dev-peer0.org2.example.com-mycc_1-
2e0c5cc7b69c44f57e492c42e87f2fa716b2de4870592cc3dcab267156f741d4
2d8af6895971        dev-peer0.org1.example.com-mycc_1-
2e0c5cc7b69c44f57e492c42e87f2fa716b2de4870592cc3dcab267156f741d4-
aa451222e4c35eaa0f97de0ac9b553c9d7f3826d65a67869a27f1ec9bbc84ec9   "chaincode -peer.add…"   5 minutes
ago       Up 5 minutes                                             dev-peer0.org1.example.com-mycc_1-
2e0c5cc7b69c44f57e492c42e87f2fa716b2de4870592cc3dcab267156f741d4
b4ea9e408108        hyperledger/fabric-
tools
"/bin/bash"              43 minutes ago      Up 43 minutes                                           cli
83d2e71727c8        hyperledger/fabric-
peer
"peer node start"        43 minutes ago      Up 43 minutes       7051/tcp, 0.0.0.0:9051->9051/tcp
peer0.org2.example.com
1042ac497c76        hyperledger/fabric-
peer
"peer node start"        43 minutes ago      Up 43 minutes       7051/tcp, 0.0.0.0:8051->8051/tcp
peer1.org1.example.com
48a57ed70232        hyperledger/fabric-
peer
"peer node start"        43 minutes ago      Up 43 minutes       0.0.0.0:7051->7051/tcp
peer0.org1.example.com
6c8bf43eabc3        hyperledger/fabric-
peer
"peer node start"        43 minutes ago      Up 43 minutes       7051/tcp, 0.0.0.0:10051->10051/tcp
peer1.org2.example.com
```

초기화

- org1-a
    1. docker compose down

    ```
    $ cd ~/fabric-samples/case1


    $ docker-compose -f docker-compose-orderer.yaml down
    ```

    2. 폴더 삭제는 생략
- org1-b
    1. docker compose down

```
$ cd ~/fabric-samples/case1


$ docker-compose -f docker-compose-node.yaml down


$ docker rm $(docker ps -aq)
```

2. 폴더 삭제는 생략