< raft blockchain 네트워크 구성 및 node-sdk 연결 간 배운 점 >

1. 네트워크 구성

  - hyperledger fabric 1.4.1 first-network 참고

```
kim@bdms:~/kismi_blockchain/sdk/javascript$
../../first-network/
├── base
│   ├── docker-compose-base.yaml
│   └── peer-base.yaml
├── byfn.sh
├── channel-artifacts
│   ├── channel.tx
│   ├── genesis.block
│   ├── Org1MSPanchors.tx
│   └── Org2MSPanchors.tx
├── configtx.yaml
├── crypto-config
│   ├── ordererOrganizations
│   │   └── example.com
│   └── peerOrganizations
│       ├── org1.example.com
│       └── org2.example.com
├── crypto-config.yaml
├── docker-compose-cli.yaml
├── docker-compose-couch-org3.yaml
├── docker-compose-couch.yaml
├── docker-compose-e2e-template.yaml
├── docker-compose-e2e.yaml
├── docker-compose-etcdraft2.yaml
├── docker-compose-kafka.yaml
├── docker-compose-org3.yaml
├── eyfn.sh
├── org3-artifacts
│   ├── configtx.yaml
│   └── org3-crypto.yaml
├── README.md
└── scripts
    ├── script.sh
    ├── step1org3.sh
    ├── step2org3.sh
    ├── step3org3.sh
    ├── testorg3.sh
    ├── upgrade_to_v14.sh
    └── utils.sh

10 directories, 28 files
```

* base 내부 2개의 파일을 docker-compose-cli.yaml이 참조
  이를 활용해 기본 네트워크 구성(CA는 만들어줌)

* orderer를 5개 구성하는
  raft관련 파일은 docker-compose-etcdraft2.yaml

* shell script를 실행하는 최종 파일은 byfn.sh이지만,
  내부에는 scripts 디렉토리의 script.sh과 utils.sh 파일을 참조
  script와 utils에서 chaincode 관련 경로 및 이름, 버전 수정

* 네트워크를 구동하는 명령어는
  $ ./byfn.sh up -o etcdraft -s couchdb -l golang
  → ordering service는 raft를 사용하고 저장공간은 couchdb를 사용 하겠다.

* 네트워크를 끌 때는 docker rm 이용하지 말고 반드시 아래
  $ ./byfn.sh down
  명령어 사용해야 함(contatiner삭제 뿐 아니라 초기화 시켜줌)

```
CONTAINER ID        IMAGE                                                                                                     COMMAND                  CREATED             STATUS              PORTS
AMES
c7299b967c34        dev-peer1.org1.example.com-bdms-1.0-fa1d313db8e2ea465ef957f3eb3ddd519be20dc9dede64081f08ff96a5690bb6   "chaincode -peer.add…"   About an hour ago   Up About an hour
ev-peer1.org1.example.com-bdms-1.0
9534aa0a165b        dev-peer1.org2.example.com-bdms-1.0-bf3cfc3946e6d83510a23b3a6700b16317c83af45ec8a025c43d4b1a2dc30638   "chaincode -peer.add…"   About an hour ago   Up About an hour
ev-peer1.org2.example.com-bdms-1.0
f364ef155867        dev-peer0.org2.example.com-bdms-1.0-b217cc476f5ae0ee3dc2e2950de12a2190322cd31b7d0a98aeaf55e2abc81694   "chaincode -peer.add…"   About an hour ago   Up About an hour
ev-peer0.org2.example.com-bdms-1.0
84063821781b        dev-peer0.org1.example.com-bdms-1.0-4b174220d89f6a1f0b886df75c0dae79c7319b7d9b360ce4cc5faaff920755de   "chaincode -peer.add…"   About an hour ago   Up About an hour
ev-peer0.org1.example.com-bdms-1.0
71e738368ebf        hyperledger/fabric-tools:latest       "/bin/bash"              About an hour ago   Up About an hour
li
6edd8bb93066        hyperledger/fabric-peer:latest        "peer node start"        About an hour ago   Up About an hour    0.0.0.0:9051->9051/tcp
eer0.org2.example.com
4c3a6353dcbf        hyperledger/fabric-peer:latest        "peer node start"        About an hour ago   Up About an hour    0.0.0.0:8051->8051/tcp
eer1.org1.example.com
196c6bda466a        hyperledger/fabric-peer:latest        "peer node start"        About an hour ago   Up About an hour    0.0.0.0:7051->7051/tcp
eer0.org1.example.com
8500c9535a22        hyperledger/fabric-peer:latest        "peer node start"        About an hour ago   Up About an hour    0.0.0.0:10051->10051/tcp
eer1.org2.example.com
7ca3f9383b44        hyperledger/fabric-orderer:latest     "orderer"                About an hour ago   Up 28 minutes       0.0.0.0:9050->7050/tcp
rderer3.example.com
51c6e35bf39b        hyperledger/fabric-orderer:latest     "orderer"                About an hour ago   Up 14 minutes       0.0.0.0:10050->7050/tcp
rderer4.example.com
ac8e59dc4793        hyperledger/fabric-orderer:latest     "orderer"                About an hour ago   Up About an hour    0.0.0.0:8050->7050/tcp
rderer2.example.com
04f7ca6405a0        hyperledger/fabric-couchdb            "tini -- /docker-ent…"   About an hour ago   Up About an hour    4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp
ouchdb0
90056a949138        hyperledger/fabric-couchdb            "tini -- /docker-ent…"   About an hour ago   Up About an hour    4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp
ouchdb2
60dc21da2769        hyperledger/fabric-couchdb            "tini -- /docker-ent…"   About an hour ago   Up About an hour    4369/tcp, 9100/tcp, 0.0.0.0:6984->5984/tcp
ouchdb1
3795e493c232        hyperledger/fabric-couchdb            "tini -- /docker-ent…"   About an hour ago   Up About an hour    4369/tcp, 9100/tcp, 0.0.0.0:8984->5984/tcp
ouchdb3
a917294a797c        hyperledger/fabric-ca                 "sh -c 'fabric-ca-se…"   About an hour ago   Up About an hour    0.0.0.0:7054->7054/tcp
a.example.com
d507cce39f52        hyperledger/fabric-orderer:latest     "orderer"                About an hour ago   Up About an hour    0.0.0.0:11050->7050/tcp
rderer5.example.com
509f85aa561d        hyperledger/fabric-orderer:latest     "orderer"                About an hour ago   Up 28 minutes       0.0.0.0:7050->7050/tcp
rderer.example.com
```

* 네트워크 구동을 완료하면 최종적으로 16개의 컨테이너 생성
  peer 4개, couchdb 4개, orderer 5개, cli, ca, chaincode

2. node sdk를 사용하는 방법

- controller, server, routes 사용

```
kim@bdms:~/kismi_blockchain/sdk/javascript$ ll
total 292
drwxrwxr-x   4 kim kim   4096  6월  19 17:19 ./
drwxrwxr-x   5 kim kim   4096  6월  18 16:00 ../
-rw-rw-r--   1 kim kim   6890  6월  19 17:37 connection.json
-rw-rw-r--   1 kim kim   4266  6월  19 15:19 connection.yaml
-rw-rw-r--   1 kim kim  34415  6월  19 17:52 controller.js
-rwxrwxr-x   1 kim kim    231  6월  10 17:02 .editorconfig*
-rw-rw-r--   1 kim kim   1851  6월  19 15:24 enrollAdmin.js
-rw-rw-r--   1 kim kim     52  6월  10 17:02 .eslintignore
-rw-rw-r--   1 kim kim    945  6월  10 17:02 .eslintrc.js
-rw-rw-r--   1 kim kim   1145  6월  10 17:02 .gitignore
-rw-rw-r--   1 kim kim   2041  6월  13 11:35 invoke.js
drwxrwxr-x 375 kim kim  16384  6월  18 17:53 node_modules/
-rw-rw-r--   1 kim kim   1127  6월  18 17:53 package.json
-rw-rw-r--   1 kim kim 171373  6월  18 17:53 package-lock.json
-rw-rw-r--   1 kim kim   1923  6월  10 17:02 query.js
-rw-rw-r--   1 kim kim   2540  6월  19 15:23 registerUser.js
-rw-rw-r--   1 kim kim   2545  6월  18 14:23 routes.js
-rw-rw-r--   1 kim kim   1677  6월  18 18:03 server.js
drwxrwxr-x   4 kim kim   4096  6월  19 17:19 wallet/
```

* node sdk에 새로 생긴 fabric-network 모듈을 사용하기 위해서는 connection.json(yaml) 파일을 잘 다뤄야 함(연결의 뽀인트!)
* 1번에서 네트워크 구동이 완료되었으면

  $ node enrollAdmin.js

  $ node registerUser.js

  위 두 파일을 이용해 wallet에 신원을 등록해줘야 함 (동시에 ~/.hfc-key-store에도 admin 키 등록됨)

```
kim@bdms:~/kismi_blockchain/sdk/javascript$ tree wallet/
wallet/
├── admin
│   ├── 1c1a7ceb3c5b7019ac628048de35e21036ef8a4e1ce4aa318f0134e171959944-priv
│   ├── 1c1a7ceb3c5b7019ac628048de35e21036ef8a4e1ce4aa318f0134e171959944-pub
│   ├── admin
│   └── d2cae29bae1470f9e2194437055f64aad024ccb85bfdba9dea81082ef0cc6c2b-priv
└── user1
    ├── d2cae29bae1470f9e2194437055f64aad024ccb85bfdba9dea81082ef0cc6c2b-priv
    ├── d2cae29bae1470f9e2194437055f64aad024ccb85bfdba9dea81082ef0cc6c2b-pub
    └── user1

2 directories, 7 files
kim@bdms:~/kismi_blockchain/sdk/javascript$ ls ~/.hfc-key-store/
1c1a7ceb3c5b7019ac628048de35e21036ef8a4e1ce4aa318f0134e171959944-priv
```

* 최종적으로 node 서버를 구동시켜 진행

  $ node server.js

3. 구현 간 발생한 오류

- 서버 구동 후 invoke 실행 시 channel이 org1을 생성 할 수 없다는 오류

  → CA를 잘못 생성하여 발생한 문제(docker-compose-cli.yaml 파일 수정 후 완료)

① KEY 매칭 에러

Error: Failed to find private key for certificate in '/etc/hyperledger/fabric-ca-server-config/ca.example.com-cert.pem':Could not find matching private key for SKI
위와 같은 에러코드가 발생 할 경우, docker-compose-cli.yaml에 있는 ca 생성 코드 확인 volumes에서 지정한 바와 같이 FABRIC_CA_SERVER_CA_CERTFILE과 KEYFILE을 잘 써줘야 함. 변수는 byfn.sh에서 지정해줬기 때문에 네트워크 구동시마다 바뀌는 key값을 하드코딩 할 필요 없이 자동 반영. 계속 문제가 됐던 부분은 **CA를 구동하는 command에서 암호키가 빠진채로 구동이되어 올바르지 못한 CA로 인식했기 때문에 다양한 에러 발생**시킴. --ca.certfile과 --ca.keyfile 옵션을 통해 지정해주면 해결!!

```
ca.example.com:
  container_name: ca.example.com
  image: hyperledger/fabric-ca
  environment:
    - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
    - FABRIC_CA_SERVER_CA_NAME=ca.example.com
    - FABRIC_CA_SERVER_TLS_ENABLED=true
    - 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/${BYFN_CA1_PRIVATE_KEY}
  ports:
    - "7054:7054"
  command: sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.org1.example.com-cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-config/${BYFN_CA1_PRIVATE_KEY} -b admin:adminpw -d'
  volumes:
    - ./crypto-config/peerOrganizations/org1.example.com/ca/:/etc/hyperledger/fabric-ca-server-config
  networks:
    - byfn
```

② 네트워크 관련 오류

2019-06-19T06:16:21.960Z - error: [Channel.js]: Error: 2 UNKNOWN: access denied: channel [mychannel] creator org [Org1MSP]
channel [mychannel]: MSP error: the supplied identity is not valid: x509: certificate signed by unknown authority
맨 위 오류코드는 invoke 실행 시 발생한 코드이며, 마지막 오류코드는 $ docker logs peer0.org1.example.com을 통해 확인한 코드 **해결방법은 ①과 동일!(모두 CA가 잘못 등록되어 발생한 오류!!)**

- 각종 인증서, TLS, endorsering peer 관련 문제

→ blockchain network와 sdk 환경이 연동되지 않아 발생한 문제(connection.json 파일 수정 후 완료)

① TLS 관련 오류

error: [Network]: _initializeInternalChannel: Unable to initialize channel. Attempted to contact 2 Peers. Last error was Error: 2 UNKNOWN: Stream removed Failed to evaluate transaction: Error: Unable to initialize channel. Attempted to contact 2 Peers. Last error was Error: 2 UNKNOWN: Stream removed
위와 같은 에러코드가 발생 할 경우, connection.json 파일 확인 byfn은 통신을 보호하기 위해 모든 node에서 TLS를 활성화 함. **connection.json 파일에서 grpc://를 grpcs://로 바꿔주어야 TLS CA 인증서를 바르게 인식**함. 예제 파일은 orderer만 가져왔지만, peer에도 모두 s를 붙여 올바른 주소를 지정 해줘야 함.


```
"orderers": {
    "orderer.example.com": {
        "url": "grpcs://localhost:7050",
        "grpcOptions": {
            "ssl-target-name-override": "orderer.example.com"
        },
        "tlsCACerts": {
            "path": "../../first-network/crypto-config/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem"
        }
    },
    "orderer2.example.com": {
        "url": "grpcs://localhost:8050",
        "grpcOptions": {
            "ssl-target-name-override": "orderer2.example.com"
        },
        "tlsCACerts": {
            "path": "../../first-network/crypto-config/ordererOrganizations/example.com/orderers/orderer2.example.com/msp/tlscacerts/tlsca.example.com-cert.pem"
        }
    },
    "orderer3.example.com": {
        "url": "grpcs://localhost:9050",
        "grpcOptions": {
            "ssl-target-name-override": "orderer3.example.com"
        },
        "tlsCACerts": {
            "path": "../../first-network/crypto-config/ordererOrganizations/example.com/orderers/orderer3.example.com/msp/tlscacerts/tlsca.example.com-cert.pem"
        }
    },
    "orderer4.example.com": {
        "url": "grpcs://localhost:10050",
        "grpcOptions": {
            "ssl-target-name-override": "orderer4.example.com"
        },
        "tlsCACerts": {
            "path": "../../first-network/crypto-config/ordererOrganizations/example.com/orderers/orderer4.example.com/msp/tlscacerts/tlsca.example.com-cert.pem"
        }
    },
    "orderer5.example.com": {
        "url": "grpcs://localhost:11050",
        "grpcOptions": {
            "ssl-target-name-override": "orderer5.example.com"
        },
        "tlsCACerts": {
            "path": "../../first-network/crypto-config/ordererOrganizations/example.com/orderers/orderer5.example.com/msp/tlscacerts/tlsca.example.com-cert.pem"
```

② endorsingPeer 지정 및 PEM 인증서 문제

TypeError: Cannot read property 'endorsingPeer' of null
Error: PEM encoded certificate is required.

```
"peers": {
    "peer0.org1.example.com": {
        "endorsingPeer": true,
        "chaincodeQuery": true,
        "ledgerQuery": true,
        "eventSource": true
    },
    "peer1.org1.example.com": {
        "endorsingPeer": true,
        "chaincodeQuery": true,
        "ledgerQuery": true,
        "eventSource": true
    },
    "peer0.org2.example.com": {
        "endorsingPeer": true,
        "chaincodeQuery": true,
        "ledgerQuery": true,
        "eventSource": true
    },
    "peer1.org2.example.com": {
        "endorsingPeer": true,
        "chaincodeQuery": true,
        "ledgerQuery": true,
        "eventSource": true
    }
```

위와 같은 에러코드가 발생 할 경우,
connection.json 파일 확인

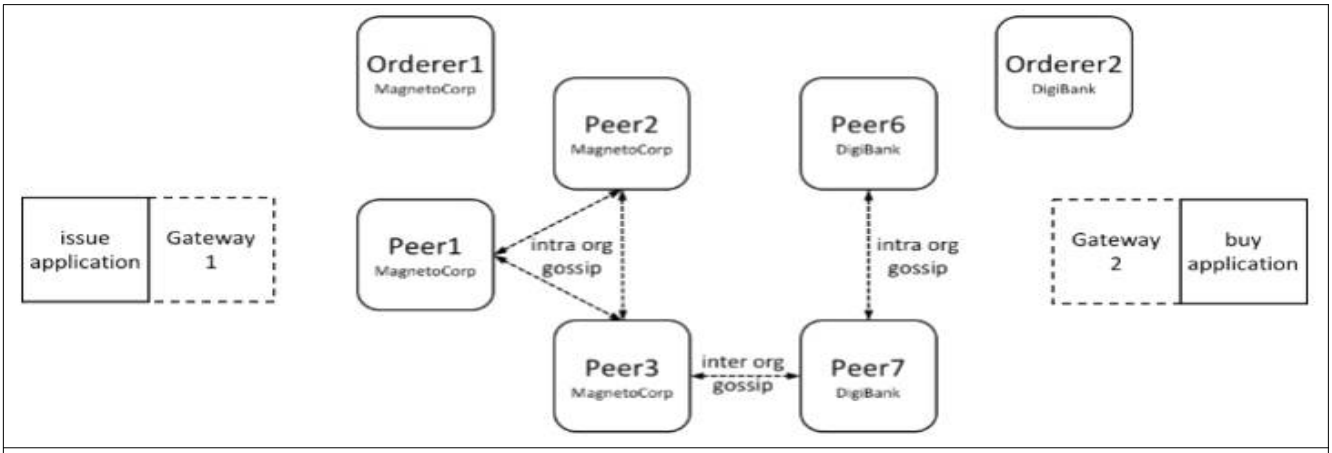
생성된 blockchain network와 sdk 환경이 매칭되지 않아 발생하는 오류로 옵션들 잘 확인해야 함.

endorsingPeer는 본래 default값으로 생성이 되지만, peer가 2개 이상 되는 경우에는 코딩 해줄 것.(그림참고)

PEM 오류는 tlsCACerts가 지정되지 않아 발생.
orderer와 peer의 설정 하는 부분에 모두 추가해주면 해결!

- node sdk에 새로 만들어진 fabric-network 모듈 사용 방법
→ https://fabric-sdk-node.github.io/release-1.4/module-fabric-network.html 참고
다양한 class가 존재하는데 기존 controller에서 작성되어 있던 것들을 모듈화 한 것.

① wallet에서 신분을 확인하고 하는 것은 gateway 클래스 참고



게이트웨이는 네트워크와 어플리케이션이 쉽게 연결될 수 있도록 해줌.

② gateway 연결 시 사용하는 옵션 문제

TypeError: this._createTxEventHandler is not a function
DefaultEventHandlerStrategies 문제 controller에서 gateway와 연결 간 함부로 옵션 지정하지 말 것! 아래 gateway.connect에서 옵션 지정하지 말 것

③ Transaction을 보내기 위해서는 Contract와 Transaction 클래스 참고

```
// Create a new file system based wallet for managing identities.
const walletPath = path.join(process.cwd(), 'wallet');
const wallet = new FileSystemWallet(walletPath);
console.log(`Wallet path: ${walletPath}`);

// Check to see if we've already enrolled the user.
const userExists = await wallet.exists('user1');
if (!userExists) {
    console.log('An identity for the user "user1" does not exist in the wallet');
    console.log('Run the registerUser.js application before retrying');
    return;
}

// Create a new gateway for connecting to our peer node.
const gateway = new Gateway();
await gateway.connect(ccp, { wallet, identity: 'user1', discovery: { enabled: false } });

// Get the network (channel) our contract is deployed to.
const network = await gateway.getNetwork('mychannel');
```

지갑에서 신원을 확인(enrollAdmin, registerUser의 결과) 후
블록체인에 접근 할 수 있는 문(gateway)을 생성
connection.json에서 작성해준 속성에 따라 user와 blockchain network가 연결!

```
// Get the contract from the network.
const contract = network.getContract('bdms');
const tx_id = contract.createTransaction('Fabric_modify_doc');
console.log("Assigning transaction_id: ", tx_id.getTransactionID()._transaction_id);

// Submit the specified transaction.
await tx_id.submit(key, title, content, tail_binary, address, state);
console.log('Transaction has been submitted');

res.status(200).json(tx_id.getTransactionID()._transaction_id); // string으로 들어감
```

연결된 blockchain network에서 bdms라는 체인코드를 찾아 contract 변수에 지정
Contract 클래스에 있는 createTransaction이란 함수를 이용해 체인코드(bdms)에서
Fabric_modify_doc라는 함수를 찾아 tx_id 생성
Transaction 클래스에 있는 submit 함수를 이용해 Fabric_modify_doc 함수에 필요한
인자값들을 전달(위에서 생성한 tx_id와 매칭)
최종적으로 ledger에 기록이 완료되고 웹에 tx_id가 반환되며 user는 다음 작업 진행