

< multi node raft blockchain 네트워크 구성 >

1. 1VM에서 네트워크 구성

① hyperledger fabric 2.0.0 first-network 참고

```
cryptojeu@cryptojeu:~/fabric-samples/first-network$ tree
.
├── base
│   ├── docker-compose-base.yaml
│   └── peer-base.yaml
├── byfn.sh
├── ccp-generate.sh
├── ccp-template.json
├── ccp-template.yaml
├── channel-artifacts
├── configtx.yaml
├── crypto-config.yaml
├── docker-compose-ca.yaml
├── docker-compose-cli.yaml
├── docker-compose-couch-org3.yaml
├── docker-compose-couch.yaml
├── docker-compose-e2e-template.yaml
├── docker-compose-etcdraft2.yaml
├── docker-compose-org3.yaml
├── eyfn.sh
├── org3-artifacts
│   ├── configtx.yaml
│   └── org3-crypto.yaml
├── README.md
├── scripts
│   ├── script.sh
│   ├── step1org3.sh
│   ├── step2org3.sh
│   ├── testorg3.sh
│   ├── upgrade_to_v14.sh
│   └── utils.sh
└── 4 directories, 25 files
```

- * base 내부 2개의 파일을 참조하는
docker-compose-ca, cli, couch.yaml 파일 참고(기본 구성)
- * 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 -a -s couchdb
→ ordering service는 raft가 default이기 때문에 따로 명시하지 않음.
→ CA를 생성하고(-a), 저장공간(-s)은 couchdb를 사용 하겠다.
- * 네트워크를 끌 때는 docker rm 이용하지 말고 아래
\$./byfn.sh down
명령어 사용(container 및 image와 network 삭제)

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
dev-peer1.org1.cryptojeyu.joy-wallet-1-905aa228f3164078f86da08c4c3ecaf145cd7fa58c355aa22ab3738afca-ade0f6c2270b3f3580b17d0d54549a6e3c47a0241373746639651433935a	dev-peer1.org1.cryptojeyu.joy-wallet-1-905aa228f3164078f86da08c4c3ecaf145cd7fa58c355aa22ab3738afca	"chaincode-peer add."	3 hours ago	Up 3 hours		dev-peer1.org1.cryptojeyu.joy
dev-peer1.org2.cryptojeyu.joy-wallet-1-905aa228f3164078f86da08c4c3ecaf145cd7fa58c355aa22ab3738afca-ade0f6c2270b3f3580b17d0d54549a6e3c47a0241373746639651433935a	dev-peer1.org2.cryptojeyu.joy-wallet-1-905aa228f3164078f86da08c4c3ecaf145cd7fa58c355aa22ab3738afca	"chaincode-peer add."	3 hours ago	Up 3 hours		dev-peer1.org2.cryptojeyu.joy
dev-peer1.org3.cryptojeyu.joy-wallet-1-905aa228f3164078f86da08c4c3ecaf145cd7fa58c355aa22ab3738afca-ade0f6c2270b3f3580b17d0d54549a6e3c47a0241373746639651433935a	dev-peer1.org3.cryptojeyu.joy-wallet-1-905aa228f3164078f86da08c4c3ecaf145cd7fa58c355aa22ab3738afca	"chaincode-peer add."	3 hours ago	Up 3 hours		dev-peer1.org3.cryptojeyu.joy
dev-peer0.org1.cryptojeyu.joy-wallet-1-905aa228f3164078f86da08c4c3ecaf145cd7fa58c355aa22ab3738afca-ade0f6c2270b3f3580b17d0d54549a6e3c47a0241373746639651433935a	dev-peer0.org1.cryptojeyu.joy-wallet-1-905aa228f3164078f86da08c4c3ecaf145cd7fa58c355aa22ab3738afca	"chaincode-peer add."	3 hours ago	Up 3 hours		dev-peer0.org1.cryptojeyu.joy
dev-peer0.org2.cryptojeyu.joy-wallet-1-905aa228f3164078f86da08c4c3ecaf145cd7fa58c355aa22ab3738afca-ade0f6c2270b3f3580b17d0d54549a6e3c47a0241373746639651433935a	dev-peer0.org2.cryptojeyu.joy-wallet-1-905aa228f3164078f86da08c4c3ecaf145cd7fa58c355aa22ab3738afca	"chaincode-peer add."	3 hours ago	Up 3 hours		dev-peer0.org2.cryptojeyu.joy
dev-peer0.org3.cryptojeyu.joy-wallet-1-905aa228f3164078f86da08c4c3ecaf145cd7fa58c355aa22ab3738afca-ade0f6c2270b3f3580b17d0d54549a6e3c47a0241373746639651433935a	dev-peer0.org3.cryptojeyu.joy-wallet-1-905aa228f3164078f86da08c4c3ecaf145cd7fa58c355aa22ab3738afca	"chaincode-peer add."	3 hours ago	Up 3 hours		dev-peer0.org3.cryptojeyu.joy
hyperledger/fabric-tools:latest	hyperledger/fabric-tools:latest	"/bin/bash"	3 hours ago	Up 3 hours		cli
hyperledger/fabric-peer:latest	hyperledger/fabric-peer:latest	"peer node start"	3 hours ago	Up 3 hours	0.0.0.0:7051->7051/tcp	peer0.org1.cryptojeyu.joy
hyperledger/fabric-peer:latest	hyperledger/fabric-peer:latest	"peer node start"	3 hours ago	Up 3 hours	7051/tcp, 0.0.0.0:8051->8051/tcp	peer1.org1.cryptojeyu.joy
hyperledger/fabric-peer:latest	hyperledger/fabric-peer:latest	"peer node start"	3 hours ago	Up 3 hours	7051/tcp, 0.0.0.0:8051->8051/tcp	peer2.org1.cryptojeyu.joy
hyperledger/fabric-peer:latest	hyperledger/fabric-peer:latest	"peer node start"	3 hours ago	Up 3 hours	7051/tcp, 0.0.0.0:8051->8051/tcp	peer0.org2.cryptojeyu.joy
hyperledger/fabric-peer:latest	hyperledger/fabric-peer:latest	"peer node start"	3 hours ago	Up 3 hours	7051/tcp, 0.0.0.0:8051->8051/tcp	peer1.org2.cryptojeyu.joy
hyperledger/fabric-peer:latest	hyperledger/fabric-peer:latest	"peer node start"	3 hours ago	Up 3 hours	7051/tcp, 0.0.0.0:8051->8051/tcp	peer2.org2.cryptojeyu.joy
hyperledger/fabric-orderer:latest	hyperledger/fabric-orderer:latest	"tini -- /docker-ent."	3 hours ago	Up 3 hours	4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp	orderer4.cryptojeyu.joy
hyperledger/fabric-orderer:latest	hyperledger/fabric-orderer:latest	"tini -- /docker-ent."	3 hours ago	Up 3 hours	7050/tcp, 0.0.0.0:8050->8050/tcp	orderer5.cryptojeyu.joy
hyperledger/fabric-ca:latest	hyperledger/fabric-ca:latest	"sh -c 'fabric-ca-se."	3 hours ago	Up 3 hours	7054/tcp, 0.0.0.0:8054->8054/tcp	ca-peer0.org1
couchdb:2.3	couchdb:2.3	"tini -- /docker-ent."	3 hours ago	Up 3 hours	4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp	couchdb0
hyperledger/fabric-orderer:latest	hyperledger/fabric-orderer:latest	"orderer"	3 hours ago	Up 3 hours	0.0.0.0:7050->7050/tcp	orderer.cryptojeyu.joy
hyperledger/fabric-orderer:latest	hyperledger/fabric-orderer:latest	"orderer"	3 hours ago	Up 3 hours	7050/tcp, 0.0.0.0:11050->11050/tcp	orderer5.cryptojeyu.joy
hyperledger/fabric-ca:latest	hyperledger/fabric-ca:latest	"sh -c 'fabric-ca-se."	3 hours ago	Up 3 hours	0.0.0.0:7054->7054/tcp	ca-peer1.org1
couchdb:2.3	couchdb:2.3	"tini -- /docker-ent."	3 hours ago	Up 3 hours	4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp	couchdb1
hyperledger/fabric-orderer:latest	hyperledger/fabric-orderer:latest	"orderer"	3 hours ago	Up 3 hours	4369/tcp, 9100/tcp, 0.0.0.0:8050->8050/tcp	couchdb3
hyperledger/fabric-orderer:latest	hyperledger/fabric-orderer:latest	"orderer"	3 hours ago	Up 3 hours	4369/tcp, 9100/tcp, 0.0.0.0:8050->8050/tcp	orderer2.cryptojeyu.joy

* 네트워크 구동을 완료하면 최종적으로 20개의 컨테이너 생성
peer 4개, couchdb 4개, orderer 5개, cli, ca 2개, chaincode 4개

② hyperledger fabric 2.0.0 test-network 참고

- * docker-compose-test-net.yaml 파일을 기본으로 하는 네트워크 생성
(ca, cli, couchdb는 별도의 파일 존재)
- * shell script를 실행하는 최종 파일은 network.sh이며, scripts 디렉토리에 createChannel.sh과
deployCC.sh 파일을 별도로 구성해 단계별 테스트 가능
- * 네트워크를 구동하는 명령어는
\$./network.sh up -ca -s couchdb
→ CA를 생성하고(-a), 저장공간(-s)은 couchdb를 사용 하겠다.
- * 네트워크를 끌 때는 docker rm 이용하지 말고 아래
\$./network.sh down
명령어 사용(container 및 image와 network 삭제)

CONTAINER ID	IMAGE	STATUS	PORTS	NAMES
dev-peer0.org2.example.com-fabcar_1-4754f10303c1a4c734ad33adc554b27237b	dev-peer0.org2.example.com-fabcar_1-4754f10303c1a4c734ad33adc554b27237b	Up 44 seconds		chaincode-peer.a
dev-peer0.org1.example.com-fabcar_1-4754f10303c1a4c734ad33adc554b27237b	dev-peer0.org1.example.com-fabcar_1-4754f10303c1a4c734ad33adc554b27237b	Up 44 seconds		chaincode-peer.a
hyperledger/fabric-peer:latest	hyperledger/fabric-peer:latest	Up 43 seconds		peer node start*
hyperledger/fabric-peer:latest	hyperledger/fabric-peer:latest	Up 43 seconds		peer node start*
hyperledger/fabric-orderer:latest	hyperledger/fabric-orderer:latest	Up 9 minutes	7051/tcp, 0.0.0.0:9051->9051/tcp	orderer*
hyperledger/fabric-couchdb	hyperledger/fabric-couchdb	Up 9 minutes	0.0.0.0:7050->7050/tcp	tini -- /docker-e
hyperledger/fabric-couchdb	hyperledger/fabric-couchdb	Up 9 minutes	4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp	tini -- /docker-e
hyperledger/fabric-ca:latest	hyperledger/fabric-ca:latest	Up 9 minutes	4369/tcp, 9100/tcp, 0.0.0.0:7984->7984/tcp	sh -c 'fabric-ca-
hyperledger/fabric-ca:latest	hyperledger/fabric-ca:latest	Up 9 minutes	0.0.0.0:7054->7054/tcp	sh -c 'fabric-ca-
hyperledger/fabric-ca:latest	hyperledger/fabric-ca:latest	Up 10 minutes	7054/tcp, 0.0.0.0:9054->9054/tcp	sh -c 'fabric-ca-
hyperledger/fabric-ca:latest	hyperledger/fabric-ca:latest	Up 10 minutes	7054/tcp, 0.0.0.0:8054->8054/tcp	

* 네트워크 구동을 완료하면 최종적으로 10개의 컨테이너 생성
peer 2개, couchdb 2개, orderer 1개, ca 3개, chaincode 2개

2. node sdk 사용

① controller, server, routes 사용

```
cryptojeju@cryptojeju:~/jeju_tumbler/sdks$ ll
total 10084
drwxr-xr-x  5 cryptojeju cryptojeju  4096 Feb  5 04:46 ./
drwxrwxr-x  9 cryptojeju cryptojeju  4096 Feb  5 04:45 ../
-rw-r--r--  1 cryptojeju cryptojeju  7264 Feb  5 04:46 connection.json
-rw-r--r--  1 cryptojeju cryptojeju  41057 Feb  5 04:45 controller.js
-rw-r--r--  1 cryptojeju cryptojeju   1848 Feb  5 04:45 enrollAdmin.js
drwxr-xr-x 382 cryptojeju cryptojeju 16384 Feb  5 04:46 node_modules/
-rw-r----- 1 cryptojeju cryptojeju 10042902 Feb  5 04:45 nohup.out
-rw-r--r--  1 cryptojeju cryptojeju   1297 Feb  5 04:45 package.json
-rw-r--r--  1 cryptojeju cryptojeju 175360 Feb  5 04:45 package-lock.json
drwxrwxr-x  2 cryptojeju cryptojeju  4096 Feb  5 04:46 public/
-rw-r--r--  1 cryptojeju cryptojeju   2513 Feb  5 04:46 registerUser.js
-rw-r--r--  1 cryptojeju cryptojeju   2177 Feb  5 04:45 routes.js
-rw-r--r--  1 cryptojeju cryptojeju   2512 Feb  5 04:45 server.js
drwxrwxr-x  4 cryptojeju cryptojeju  4096 Feb  5 04:45 wallet/
```

* 1번에서 네트워크 구동이 완료되었으면

\$ node enrollAdmin.js

\$ node registerUser.js

위 두 파일을 이용해 wallet에 신원 등록
(동시에 ~/.hfc-key-store에도 admin 키 등록)

```
kim@boms:~/kismi_blockchain/sdk/javascript$ tree wallet/
wallet/
├── admin
│   ├── 1c1a7ceb3c5b7019ac628048de35e21036ef8a4e1ce4aa318f0134e171959944-priv
│   └── 1c1a7ceb3c5b7019ac628048de35e21036ef8a4e1ce4aa318f0134e171959944-pub
├── admin
│   ├── d2cae29bae1470f9e2194437055f64aad024ccb85bfdba9dea81082ef8cc6c2b-priv
│   └── d2cae29bae1470f9e2194437055f64aad024ccb85bfdba9dea81082ef8cc6c2b-pub
└── user1
    ├── d2cae29bae1470f9e2194437055f64aad024ccb85bfdba9dea81082ef8cc6c2b-priv
    └── d2cae29bae1470f9e2194437055f64aad024ccb85bfdba9dea81082ef8cc6c2b-pub

kim@boms:~/kismi_blockchain/sdk/javascript$ ls ~/.hfc-key-store/
1c1a7ceb3c5b7019ac628048de35e21036ef8a4e1ce4aa318f0134e171959944-priv
```

* 최종적으로 node 서버를 구동시켜 진행

\$ sudo npm install pm2 -g

\$ pm2 start server.js

\$ pm2 monit

3. 7VM에서 네트워크 구성

① 각 node들이 서로를 인식할 수 있도록 /etc/hosts 파일에 등록

* 클라우드 peering이 되었다는
전제 하에 진행(VM간 ping test 완료)

\$ sudo vi /etc/hosts → 수정 후 저장

```
10.1.0.4 cryptojeju
10.2.0.4 jejudo
10.3.0.4 orderer1
10.3.0.9 orderer2
10.3.0.6 orderer3
10.3.0.7 orderer4
10.3.0.8 orderer5
```

```
127.0.0.1 localhost

10.1.0.4 cryptojeju
10.2.0.4 jejudo
10.3.0.4 orderer1
10.3.0.9 orderer2
10.3.0.6 orderer3
10.3.0.7 orderer4
10.3.0.8 orderer5
```

node이름으로 ping test (ex - ping cryptojeju 등)

② solo에서 구성한 chaincode가 포함된 git repository 다운로드

\$ git clone https://dheo@bitbucket.org/dheo/cryptojeju_blockchain.git
※ .env 및 .gitignore 파일 존재여부 확인

③ docker swarm 연결하기
(초기화)

<https://tech.osci.kr/2019/02/13/59736201/> 참고 사이트

(cryptojeju PC)\$ **docker swarm init --advertise-addr 10.1.0.4**

(cryptojeju PC)\$ **docker swarm join-token manager**

```
cryptojeju@cryptojeju:~/jeju_tumbler$ docker swarm init --advertise-addr 10.1.0.4
Swarm initialized: current node (8h0wflzlm234llhwhjicvasl) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-5ahxjnt45gigbl1iidyxyls1rgfscxc6xp10gferfypzz2ijs-7rxjniidadj9i2ocywycs8eo 10.1.0.4:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

cryptojeju@cryptojeju:~/jeju_tumbler$ docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-5ahxjnt45gigbl1iidyxyls1rgfscxc6xp10gferfypzz2ijs-d6l1fkht5a3hj1kq2s8bjlibw6 10.1.0.4:2377
```

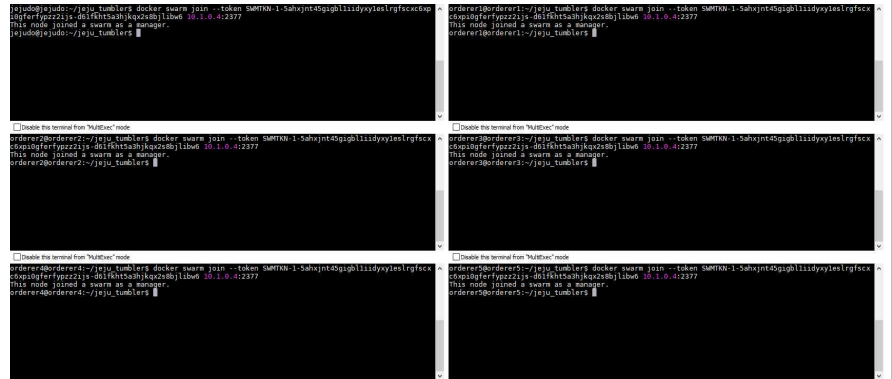
※ 이전 기록이 있다면 **\$ docker swarm leave --force** 로 삭제

3. 7VM에서 네트워크 구성(이어서)

④ docker swarm 연결하기 (다른 노드들 manager로 join)

(나머지 PC)\$ docker swarm join --token SWMTKN-1-5ahxjnt45gigbl1iidyxy1eslrgfscxc6xpi0gferfypzz2ijs-d61fkht5a3hjkqx2s8bjlibw6 10.1.0.4:2377

→ This node joined a swarm as a manager. 확인



\$ docker node ls → 7개 node가 잘 연결되었는지 확인

```
cryptojeju@cryptojeju:~/jeju_tumblers$ docker node ls
```

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
8h0wwflzlmz34llhwhjicvas1 *	cryptojeju	Ready	Active	Leader	19.03.5
0al880aa6wlnxm25kxdlfher	jejudo	Ready	Active	Reachable	19.03.5
sadumibcczy9n9xa5qyn4le4ns	orderer1	Ready	Active	Reachable	19.03.5
t2biuc469arhpc5ihpoh5mtjt	orderer2	Ready	Active	Reachable	19.03.5
4yj9n3cw2sj4jttvvh1o8pju8	orderer3	Ready	Active	Reachable	19.03.5
vwuwt0a5zbwc5mpcj8281s3im	orderer4	Ready	Active	Reachable	19.03.5
j7yx6aone0mc9b0ua4i2lvee8	orderer5	Ready	Active	Reachable	19.03.5

⑤ docker swarm 연결하기 (network 생성하기)

(cryptojeju PC)\$ docker network create --attachable --driver overlay kismi-cryptojeju

```
cryptojeju@cryptojeju:~/jeju_tumblers$ docker network create --attachable --driver overlay kismi-cryptojeju
le8hmf7d3zlwlsq7ongikbci
```

(cryptojeju PC)\$ docker network inspect kismi-cryptojeju

```
cryptojeju@cryptojeju:~/jeju_tumblers$ docker network inspect kismi-cryptojeju
[
  {
    "Name": "kismi-cryptojeju",
    "Id": "le8hmf7d3zlwlsq7ongikbci",
    "Created": "2020-02-05T06:44:33.395775398Z",
    "Scope": "swarm",
    "Driver": "overlay",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "10.0.1.0/24",
          "Gateway": "10.0.1.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": true,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": null,
    "Options": {
      "com.docker.network.driver.overlay.vxlanid_list": "4097"
    },
    "Labels": null
  }
]
```

※ 다른 manager 노드에서 \$ docker network ls 명령어 실행시 모두 동일하게 'kismi-cryptojeju' network 보유

⑥~⑦ 한 번에 실행하기

\$./network.sh up

```
cryptojeju@cryptojeju:~/jeju_tumblers$ ./network.sh up
===== MSP 생성 시작 =====
1. cert file 생성
2. genesis.block 생성
3. channel.tx 생성
4. OrgMSP 별 anchors.tx 생성
=====
```


3. 7VM에서 네트워크 구성(이어서)

⑥ Cert 관련 파일들 생성하기

\$ cryptogen generage --config=./crypto-config.yaml

```
===== 1. cert file 생성 =====  
  
org1.cryptoteju.joy  
org2.cryptoteju.joy
```

⑦ ChannelArtifacts 생성하기

- genesis.block
- channel.tx
- anchorpeer_org1
- anchorpeer_org2

\$ configtxgen -profile SampleMultiNodeEtcdRaft --channelID byfn-sys-channel -outputBlock ./channel-artifacts/genesis.block

```
===== 2. genesis.block 생성 =====  
  
2020-02-05 08:18:14.799 UTC [common.tools.configtxgen] main -> INFO 001 Loading configuration  
2020-02-05 08:18:14.851 UTC [common.tools.configtxgen.localconfig] completeInitialization -> INFO 002 orderer type: etcdraft  
2020-02-05 08:18:14.851 UTC [common.tools.configtxgen.localconfig] completeInitialization -> INFO 003 Orderer.Etcdraft.Options unset, setting to tick_interval:"500ms" election_tick:10 heartbeat_tick:1 max_inflight_blocks:500 poll_interval_size:1077234  
2020-02-05 08:18:14.854 UTC [common.tools.configtxgen.localconfig] load -> INFO 004 Loaded configuration: /home/cryptoteju/jeju_tumbler/configtx.yaml  
2020-02-05 08:18:14.854 UTC [common.tools.configtxgen] doOutputBlock -> INFO 005 Generating genesis block  
2020-02-05 08:18:14.855 UTC [common.tools.configtxgen] doOutputBlock -> INFO 006 Writing genesis block
```

\$ configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx -channelID \$CHANNEL_NAME

```
===== 3. channel.tx 생성 =====  
  
2020-02-05 08:18:14.882 UTC [common.tools.configtxgen] main -> INFO 001 Loading configuration  
2020-02-05 08:18:14.921 UTC [common.tools.configtxgen.localconfig] load -> INFO 002 Loaded configuration: /home/cryptoteju/jeju_tumbler/configtx.yaml  
2020-02-05 08:18:14.921 UTC [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 003 Generating new channel configtx  
2020-02-05 08:18:14.923 UTC [common.tools.configtxgen] doOutputChannelCreateTx -> INFO 004 Writing new channel tx
```

\$ configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID \$CHANNEL_NAME -asOrg Org1MSP

\$ configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID \$CHANNEL_NAME -asOrg Org2MSP

```
===== 4. anchors.tx 생성 =====  
  
2020-02-05 08:18:14.948 UTC [common.tools.configtxgen] main -> INFO 001 Loading configuration  
2020-02-05 08:18:14.979 UTC [common.tools.configtxgen.localconfig] load -> INFO 002 Loaded configuration: /home/cryptoteju/jeju_tumbler/configtx.yaml  
2020-02-05 08:18:14.981 UTC [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update  
2020-02-05 08:18:14.981 UTC [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update  
2020-02-05 08:18:15.005 UTC [common.tools.configtxgen] main -> INFO 001 Loading configuration  
2020-02-05 08:18:15.037 UTC [common.tools.configtxgen.localconfig] load -> INFO 002 Loaded configuration: /home/cryptoteju/jeju_tumbler/configtx.yaml  
2020-02-05 08:18:15.037 UTC [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Generating anchor peer update  
2020-02-05 08:18:15.038 UTC [common.tools.configtxgen] doOutputAnchorPeersUpdate -> INFO 004 Writing anchor peer update
```

⑧ node별 docker container 생성하기(shell script 활용)

docker run을 활용하여 환경변수(-e)와 볼륨(-v), working directory(-w) 등을 설정한 shell script를 동시에 시작
\$./run_<tap키>/>

```
cryptoteju@cryptoteju:~/jeju_tumbler$
```

```
[c]reate the terminal from "tumbler" mode
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
jejud081judo	hyperledger/fabric-peer:latest	peer node start	4 minutes ago	up
jejud082judo	hyperledger/fabric-peer:latest	peer node start	4 minutes ago	up
jejud083judo	hyperledger/fabric-ca:latest	sh -c 'fabric-ca-ssl'	4 minutes ago	up

```
cryptoteju@cryptoteju:~/jeju_tumbler$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	NAMES	CREATED
jejud081judo	hyperledger/fabric-peer:latest	peer node start	peer0.org1.cryptoteju.joy	4 minutes ago
jejud082judo	hyperledger/fabric-peer:latest	peer node start	peer0.org2.cryptoteju.joy	4 minutes ago
jejud083judo	hyperledger/fabric-ca:latest	sh -c 'fabric-ca-ssl'	ca.org1.cryptoteju.joy	4 minutes ago

```
[c]reate the terminal from "tumbler" mode
```

```
orderer@orderer1:~/jeju_tumbler$
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
orderer1orderer1	hyperledger/fabric-orderer:latest	orderer	4 minutes ago	up

```
orderer@orderer1:~/jeju_tumbler$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	NAMES	CREATED	STATUS
orderer1orderer1	hyperledger/fabric-orderer:latest	orderer	orderer1.cryptoteju.joy	4 minutes ago	up

```
[c]reate the terminal from "tumbler" mode
```

```
orderer@orderer2:~/jeju_tumbler$
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
orderer2orderer2	hyperledger/fabric-orderer:latest	orderer	4 minutes ago	up

```
orderer@orderer2:~/jeju_tumbler$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	NAMES	CREATED	STATUS
orderer2orderer2	hyperledger/fabric-orderer:latest	orderer	orderer2.cryptoteju.joy	4 minutes ago	up

```
[c]reate the terminal from "tumbler" mode
```

```
orderer@orderer3:~/jeju_tumbler$
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
orderer3orderer3	hyperledger/fabric-orderer:latest	orderer	4 minutes ago	up

```
orderer@orderer3:~/jeju_tumbler$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	NAMES	CREATED	STATUS
orderer3orderer3	hyperledger/fabric-orderer:latest	orderer	orderer3.cryptoteju.joy	4 minutes ago	up

각 VM 별로 알맞은 docker container들이 작동 시작!!

- ⑨ cli에서 network 최종 세팅
 - channel 생성
 - peer들 channel에 join시키기
 - org별 anchor peer 등록
 - chaincode package화
 - install 및 commit

[illegible][illegible]

```
$ sudo npm install pm2 -g → 전역에 pm2 설치
$ pm2 start server.js → node server start!
$ pm2 show server → server 관련 정보보기(로그위치 등)
$ pm2 monit → 모니터링 페이지
```

[illegible]

\$ tail -f ~/.pm2/logs/server-out.log → 로그만 따로 볼 수도 있음

4. 트러블슈팅

① shell script 만들 때 Syntax error

```
cryptojeju@cryptojeju:~/jeju_tumbler$ ./network.sh up
./network.sh: 13: ./network.sh: Syntax error: "(" unexpected
```

→ shell script 맨 첫줄 #!/bin/sh에서 #!/bin/bash로 변경 (<http://pchero21.com/?p=678> 참고)

② Admin 등록 시 오류발생(end point failed error write EPROTO / socket hang up)

```
cryptojeju@cryptojeju:~/jeju_tumbler/sdk$ node enrollAdmin.js
Wallet path: /home/cryptojeju/jeju_tumbler/sdk/wallet
2020-02-06T01:58:54.592Z - error: [FabricClientService.js]: Failed to enroll admin, error:so message=Calling enrollment endpoint failed with error [Error: write EPROTO 140359042647872:error:1408F108:SSL routines:ssl3_get_record:wrong version number:../deps/openssl/openssl/ssl/record/ssl3_re
cord.c:332:
], stack=Error: Calling enrollment endpoint failed with error [Error: write EPROTO 140359042647872:error:1408F108:SSL routines:ssl3_get_record:wrong version number:../deps/openssl/openssl/ssl/record/ssl3_re
cord.c:332:
]
at ClientRequest.request.on (/home/cryptojeju/jeju_tumbler/sdk/node_modules/fabric-ca-client/lib/FabricCAClient.js:487:12)
at ClientRequest.emit (events.js:198:13)
at TLSSocket.socketErrorListener (/http_client.js:392:0)
at TLSSocket.emit (events.js:198:13)
at errorOrDestroy (internal/stream_base.js:107:12)
at onwriteError (/stream_writable.js:436:5)
at onwrite (/stream_writable.js:461:5)
at _destroy (internal/stream_base.js:49:7)
at TLSSocket.destroy (/net.js:614:3)
at TLSSocket.destroy (internal/stream_base.js:37:8)
Failed to enroll admin user "admin": Error: Calling enrollment endpoint failed with error [Error: write EPROTO 140359042647872:error:1408F108:SSL routines:ssl3_get_record:wrong version number:../deps/openssl/openssl/ssl/record/ssl3_re
cord.c:332:
]
```

→ [peer1] https-proxy 때문에 발생하는 오류 (https를 http로 변경하면 해결)

```
jejud@jejud:~/jeju_tumbler/sdk$ node enrollAdmin.js
Wallet path: /home/jejud/jeju_tumbler/sdk/wallet
2020-02-06T01:55:25.461Z - error: [FabricClientService.js]: Failed to enroll admin, error:so message=Calling enrollment endpoint failed with error [Error: socket hang up], stack=Error: Calling enrollment
endpoint failed with error [Error: socket hang up]
at ClientRequest.request.on (/home/jejud/jeju_tumbler/sdk/node_modules/fabric-ca-client/lib/FabricCAClient.js:487:12)
at ClientRequest.emit (events.js:198:13)
at Socket.socketOnEnd (/http_client.js:426:9)
at Socket.emit (events.js:203:15)
at endReadableNT (/stream_readable.js:1145:12)
at process._tickCallback (internal/process/next_tick.js:63:19)
Failed to enroll admin user "admin": Error: Calling enrollment endpoint failed with error [Error: socket hang up]
```

→ [peer2] CA docker container 구동 시 port번호를 8054:8054에서 8054:7054로 변경
CA 자체가 7054로 listen하기 때문에 8054로 들어오는 데이터를 7054로 연결해주면 문제 해결

③ npm install 시 오류 발생(<https://helloinyong.tistory.com/191>)

```
npm ERR! path /home/jejud/jeju_tumbler/sdk/node_modules
npm ERR! code EACCES
npm ERR! errno -13
npm ERR! syscall access
npm ERR! Error: EACCES: permission denied, access '/home/jejud/jeju_tumbler/sdk/node_modules'
npm ERR! { [Error: EACCES: permission denied, access '/home/jejud/jeju_tumbler/sdk/node_modules']
npm ERR!   stack:
npm ERR!     'Error: EACCES: permission denied, access \'/home/jejud/jeju_tumbler/sdk/node_modules\'',
npm ERR!     errno: -13,
npm ERR!     code: 'EACCES',
npm ERR!     syscall: 'access',
npm ERR!     path: '/home/jejud/jeju_tumbler/sdk/node_modules' }
npm ERR! Please try running this command again as root/Administrator.

npm ERR! A complete log of this run can be found in:
npm ERR! /home/jejud/.npm/_logs/2020-02-06T01_31_44_759Z-debug.log
```

→ 현재 접속중인 local 계정이 npm 설치 경로에 대한 권한을 가지고 있지 않아서 발생하는 문제

→ 해결 방법은 1) root로 되어있는 디렉토리 권한을 User 그룹의 권한으로 변경하던가,

2) npm install -g로 설치되는 디렉토리 경로를 home 디렉토리로 변경하는 것.

3) 기타 등등

1) \$ sudo chown -R \$USER:\$GROUP ~/.npm

```
1 | mkdir ~/.npm-global
2 |
3 | npm config set prefix '~/.npm-global'
4 |
5 | # vi ~/.profile (아래 문장을 추가)
6 | export PATH=~/.npm-global/bin:$PATH
7 |
8 | source ~/.profile
```

(<https://en.dbookwordpress.com/2016/03/05/npm-permission-%EB%A8%BC%A0%8C-%ED%85%B4%EA%B2%B0%ED%85%98%EA%B8%B0/>)

3) \$ sudo npm install --unsafe-perm=true --allow-root

(<https://stackoverflow.com/questions/46439403/ubuntu-nodejs-npm-install-g-error-eaccess-permission-denied-mkdir>)

④ node server.js 이후 try~catch 관련 에러

```
Wallet path: /home/cryptojeju/jeju_tumbler/sdk/wallet
(node:27629) UnhandledPromiseRejectionWarning: ReferenceError: error is not defined
    at Object.query_block (/home/cryptojeju/jeju_tumbler/sdk/controller.js:811:64)
(node:27629) UnhandledPromiseRejectionWarning: Unhandled promise rejection. This error originated either by throwing inside of an async function without a catch block, or by rejecting a promise which was not
handled with .catch(). (rejection id: 1)
```

→ 'error' 라는 변수가 없어서 발생하는 문제(선언되지 않은 변수 - 선언해주면 해결)

(https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Errors/Not_defined 참고)