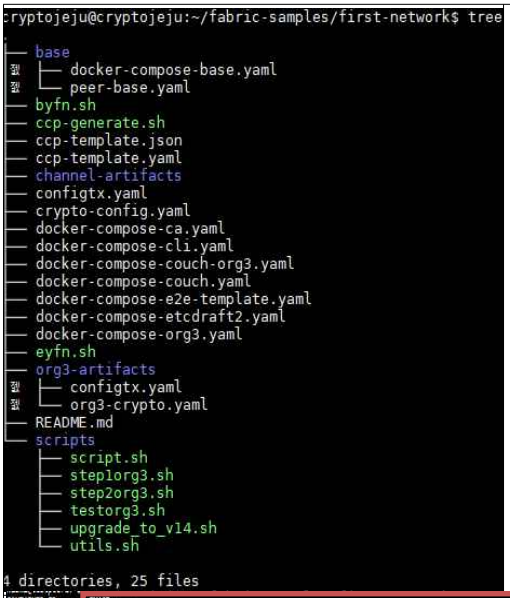
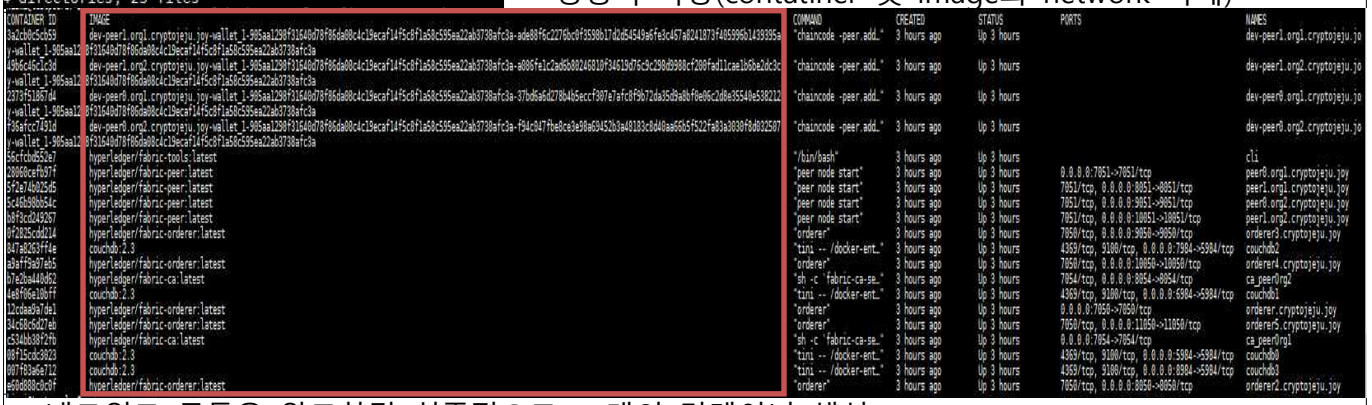
# 〈 multi node raft blockchain 네트워크 구성 〉

1. 1VM에서 네트워크 구성

① hyperledger fabric 2.0.0 first-network 참고



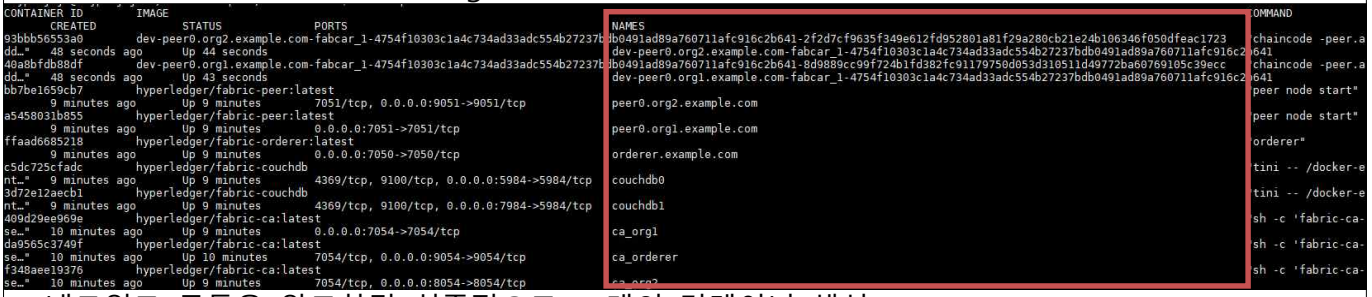

* base 내부 2개의 파일을 참조하는
  docker-compose-ca, cli, couch.yaml 파일 참고(기본 구성)
* orderer를 5개 구성하는
  raft관련 파일은 docker-compose-etcdraft2.yaml
* shell script를 실행하는 최종 파일은 byfn.sh이지만,
  내부에는 scripts 디렉토리의 script.sh과 utils.sh 파일을 참조
  script와 utils에서 chaincode 관련 경로 및 이름, 버전 수정
* 네트워크를 구동하는 명령어는
  **$ ./byfn.sh 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(contatiner 및 image와 network 삭제)




* 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(contatiner 및 image와 network 삭제)




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

kim@bdms:~/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

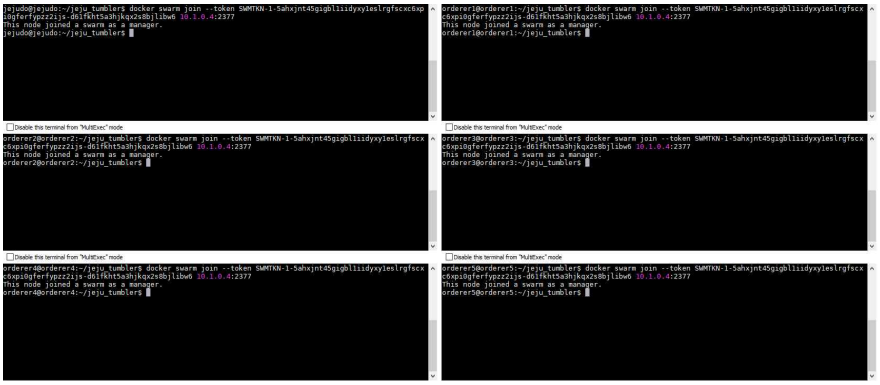
| | |
|---|---|
| ① 각 node들이 서로를 인식할 수 있도록 /etc/hosts 파일에 등록<br>* 클라우드 peeering이 되었다는 전제 하에 진행(VM간 ping test 완료) | $ sudo vi /etc/hosts → 수정 후 저장<br><br>10.1.0.4 cryptojeju<br>10.2.0.4 jejudo<br>10.3.0.4 orderer1<br>10.3.0.9 orderer2<br>10.3.0.6 orderer3<br>10.3.0.7 orderer4<br>10.3.0.8 orderer5<br><br>127.0.0.1 localhost<br><br>10.1.0.4 cryptojeju<br>10.2.0.4 jejudo<br>10.3.0.4 orderer1<br>10.3.0.9 orderer2<br>10.3.0.6 orderer3<br>10.3.0.7 orderer4<br>10.3.0.8 orderer5<br><br>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<br>※ .env 및 .gitignore 파일 존재여부 확인 |
| ③ docker swarm 연결하기 (초기화) | https://tech.osci.kr/2019/02/13/59736201/ 참고 사이트<br>(cryptojeju PC)$ docker swarm init --advertise-addr 10.1.0.4<br>(cryptojeju PC)$ docker swarm join-token manager<br><br>※ 이전 기록이 있다면 $ docker swarm leave --force 로 삭제 |

```
cryptojeju@cryptojeju:~/jeju_tumbler$ docker swarm init --advertise-addr 10.1.0.4
Swarm initialized: current node (8h0wwflz1mz341lhhwjicvas1) is now a manager.

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

    docker swarm join --token SWMTKN-1-5ahxjnt45gigbliidyxy1eslrgfscxc6xpi0gferfypzz2ijs-7rxjniidadju9i2ocyywcs8eo 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-1-5ahxjnt45gigbliidyxy1eslrgfscxc6xpi0gferfypzz2ijs-d61fkht5a3hjkqx2s8bjlibw6 10.1.0.4:2377
```

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_tumbler$ docker node ls
ID                            HOSTNAME            STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
8h0wwflz1mz341lhhwjicvas1 *   cryptojeju          Ready               Active              Leader              19.03.5
0al88oaa6wlwxnm25kxd1fher     jejudo              Ready               Active              Reachable           19.03.5
sadum1bczy9n9xa5qyn4le4ns     orderer1            Ready               Active              Reachable           19.03.5
t2biuc469arhpc5ihpoh5mtjt     orderer2            Ready               Active              Reachable           19.03.5
4yj9n3cw2sj4jjtvvh1o8pju8     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_tumbler$ docker network create --attachable --driver overlay kismi-cryptojeju
le8hmfb7d3zlwisq7ong1kbci
```

(cryptojeju PC)$ docker network inspect kismi-cryptojeju

```
cryptojeju@cryptojeju:~/jeju_tumbler$ docker network inspect kismi-cryptojeju
[
    {
        "Name": "kismi-cryptojeju",
        "Id": "le8hmfb7d3zlwisq7ong1kbci",
        "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
    }
]
```

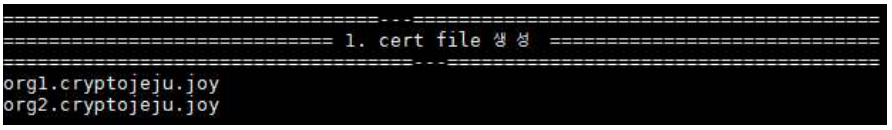
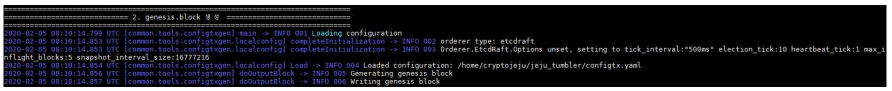
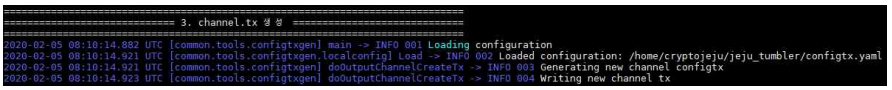
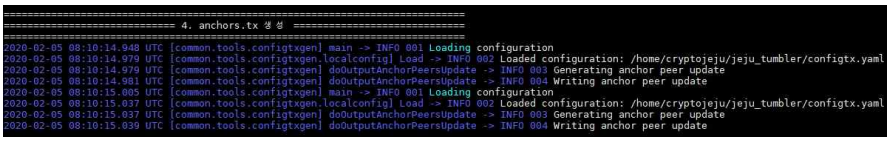
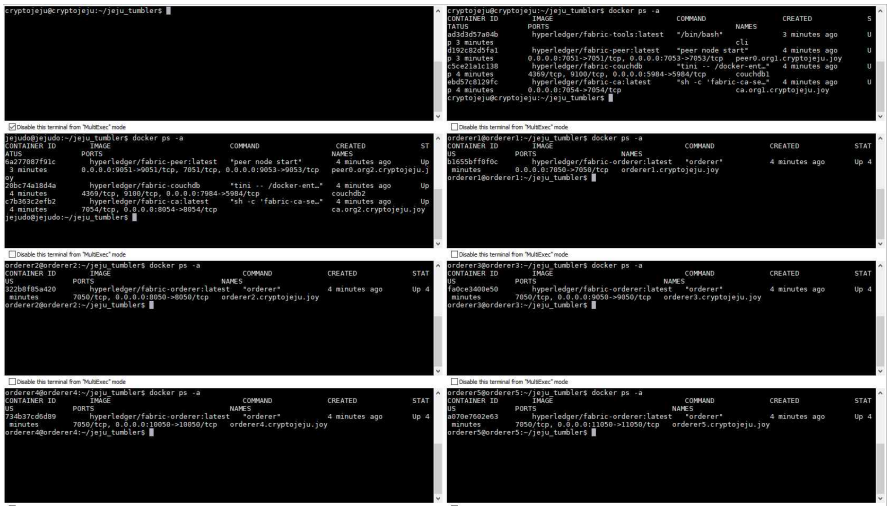
※ 다른 manager 노드에서 $ docker network ls 명령어 실행시 모두 동일하게 'kismi-cryptojeju' network 보유

⑥~⑦ 한 번에 실행하기

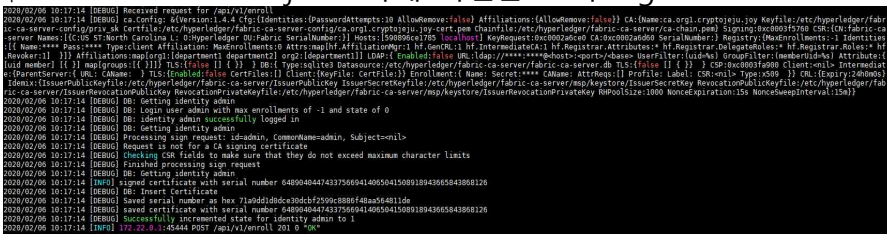
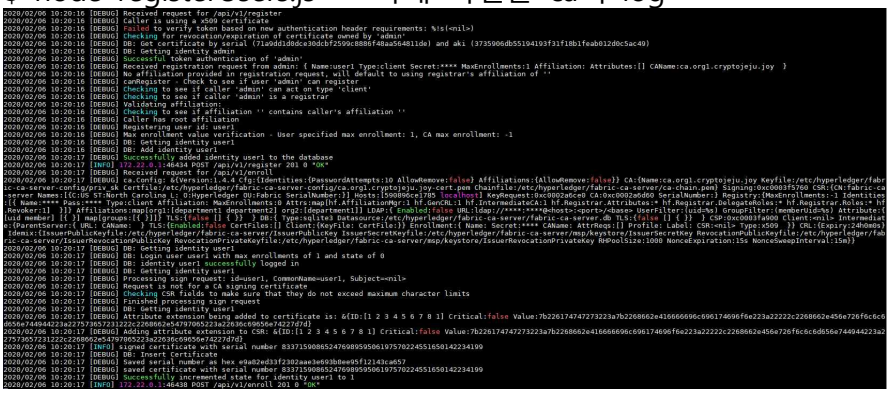
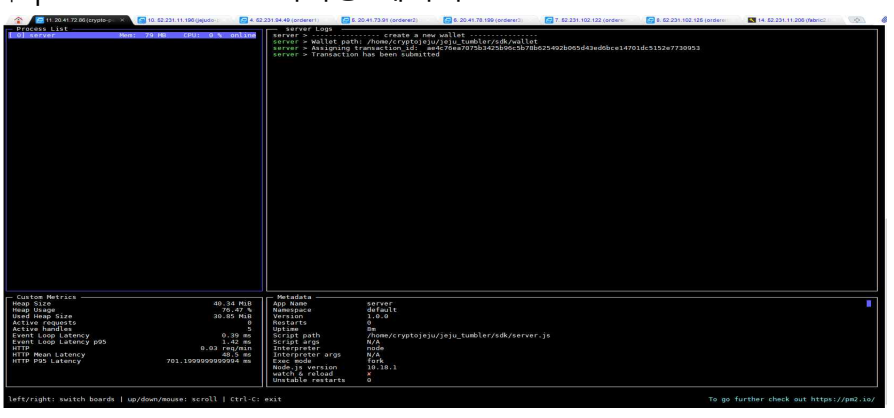
$ ./network.sh up

```
cryptojeju@cryptojeju:~/jeju_tumbler$ ./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<br>  |
| ⑦ ChannelArtifacts 생성하기<br>- genesis.block<br>- channel.tx<br>- anchorpeer_org1<br>- anchorpeer_org2 | $ configtxgen -profile SampleMultiNodeEtcdRaft --channelID byfn-sys-channel -outputBlock ./channel-artifacts/genesis.block<br>  <br>$ configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx -channelID $CHANNEL_NAME<br>  <br>$ configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID $CHANNEL_NAME -asOrg Org1MSP<br>$ configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID $CHANNEL_NAME -asOrg Org2MSP<br>  |
| ⑧ node별 docker container 생성하기(shell script 활용) | docker run을 활용하여 환경변수(-e)와 볼륨(-v), working directory(-w) 등을 설정한 shell script를 동시에 시작<br>$ ./run_*<tap키>*<br>  <br>각 VM 별로 알맞은 docker container들이 작동 시작!! |

3. 7VM에서 네트워크 구성(이어서)

<table>
<tr>
<td>⑨ cli에서 network 최종 세팅<br>- channel 생성<br>- peer들 channel에 join시키기<br>- org별 anchor peer 등록<br>- chaincode package화<br>- install 및 commit</td>
<td>$ ./cli_run.sh<br># ./scripts/script.sh<br>  <br># peer chaincode invoke -o orderer1.cryptojeju.joy:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/cryptojeju.joy/orderers/orderer1.cryptojeju.joy/msp/tlscacerts/tlsca.cryptojeju.joy-cert.pem -C cafechannel -n wallet --peerAddresses peer0.org1.cryptojeju.joy:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.cryptojeju.joy/peers/peer0.org1.cryptojeju.joy/tls/ca.crt --peerAddresses peer0.org2.cryptojeju.joy:9051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.cryptojeju.joy/peers/peer0.org2.cryptojeju.joy/tls/ca.crt -c '{"Args":["create_wallet","cryptojeju:00001"]}'<br>  </td>
</tr>
<tr>
<td>⑩ enrollAdmin & registerUsers</td>
<td>Org1MSP와 Org2MSP 별 connection.json 파일 작성 후<br>$ node enrollAdmin.js → 아래 화면은 ca의 log<br>  <br>$ node registerUsers.js → 아래 화면은 ca의 log<br>  </td>
</tr>
<tr>
<td>⑪ API 서버 키고 작동<br>* nodejs의 프로세스 관리도구인 PM2 활용</td>
<td>$ sudo npm install pm2 -g → 전역에 pm2 설치<br>$ pm2 start server.js → node server start!<br>$ pm2 show server → server 관련 정보보기(로그위치 등)<br>$ pm2 monit → 모니터링 페이지<br>  <br>$ tail -f ~/.pm2/logs/server-out.log → 로그만 따로 볼 수도 있음</td>
</tr>
</table>

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
2020-02-05T12:58:54.592Z - error: [FabricCAClientService.js]: Failed to enroll admin, error:%o message=Calling enrollment endpoint failed with error [Error: write EPROTO 140359042647872:error:1408F10B:SSL routines:ssl3_get_record:wrong version number:../deps/openssl/openssl/ssl/record/ssl3_record.c:332:
], stack=Error: Calling enrollment endpoint failed with error [Error: write EPROTO 140359042647872:error:1408F10B:SSL routines:ssl3_get_record:wrong version number:../deps/openssl/openssl/ssl/record/ssl3_record.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:392:9)
    at TLSSocket.emit (events.js:198:13)
    at errorOrDestroy (internal/streams/destroy.js:107:12)
    at onwriteError (_stream_writable.js:436:5)
    at onwrite (_stream_writable.js:461:5)
    at _destroy (internal/streams/destroy.js:49:7)
    at TLSSocket.Socket._destroy (net.js:614:3)
    at TLSSocket.destroy (internal/streams/destroy.js:37:8)
Failed to enroll admin user "admin": Error: Calling enrollment endpoint failed with error [Error: write EPROTO 140359042647872:error:1408F10B:SSL routines:ssl3_get_record:wrong version number:../deps/openssl/openssl/ssl/record/ssl3_record.c:332:
]
```

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

```
jejudo@jejudo:~/jeju_tumbler/sdk$ node enrollAdmin.js
wallet path: /home/jejudo/jeju_tumbler/sdk/wallet
2020-02-06T01:55:25.461Z - error: [FabricCAClientService.js]: Failed to enroll admin, error:%o 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/jejudo/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/jejudo/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/jejudo/jeju_tumbler/sdk/node_modules'
npm ERR!  { [Error: EACCES: permission denied, access '/home/jejudo/jeju_tumbler/sdk/node_modules']
npm ERR!   stack:
npm ERR!    'Error: EACCES: permission denied, access \'/home/jejudo/jeju_tumbler/sdk/node_modules\'',
npm ERR!   errno: -13,
npm ERR!   code: 'EACCES',
npm ERR!   syscall: 'access',
npm ERR!   path: '/home/jejudo/jeju_tumbler/sdk/node_modules' }
npm ERR!
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/jejudo/.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

2)
```
mkdir ~/.npm-global

npm config set prefix '~/.npm-global'

# vi ~/.profile (아래 문장을 추가)
export PATH=~/.npm-global/bin:$PATH

source ~/.profile
```
(https://enydoorkwordpress.com/2016/08/05/npm-permission-%EB%AC%B8%EC%A0%9C-%ED%95%B4%EA%B2%B0%ED%95%98%EA%B8%B0/)

3) $ sudo npm install --unsafe-perm=true --alow-root
(https://stackoverflow.com/questions/46439403/ubuntu-nodejs-npm-install-g-error-eacces-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:611:64)
(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)