< 하이퍼레져 페브릭 실습1 >

1. 하이퍼레져 페브릭 실습준비

- VM의 우분투 들어가서 로그인(guru / work)해서 서버 기동
  → VisualCode에서 $ ssh 52698:localhost:52698 guru@127.0.0.1로 로그인

```
19  # set PATH so it includes user's private bin directories
20  export GOROOT=/usr/local/go
21  export GOPATH=$HOME/gowork
22  PATH="$HOME/bin:$HOME/.local/bin:$PATH:$GOROOT/bin:"
guru@guru:~$ go get github.com/hyperledger/fabric
```

- 하고 fabric-ca해서 한 번 더! 설치!

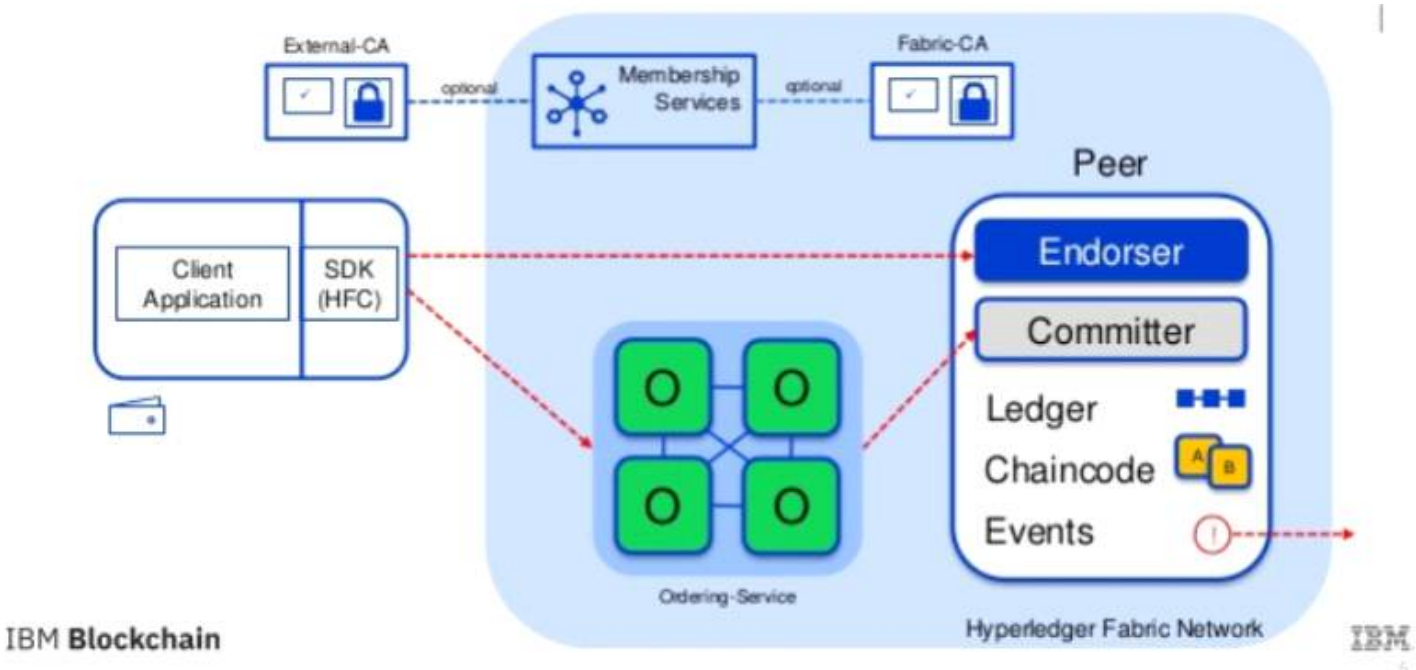
- $ cd $GOPATH → ls해서 src가 있는 것을 확인 →
  $ cd src/github.com/hyperledger/fabric 위치이동 → $ git branch로 'release-1.3' 확인 →
  $ make native docker 실행

- $ cd src/github.com/hyperledger/fabric-ca 위치이동 → $ make docker 실행

2. 블록체인 직업

- 블록체인 개발자 : 어플리케이션, 스마트컨트렉트의 개발.(피어 등에는 관심 ㄴ)
- 블록체인 운영자 : 동작구조에 대해 상세히 파악해야 함.(peer, consensus, security)
  (반드시 블록체인 개발을 할 수 있어야 하는 것은 아님 / 코딩 ㄴ)
- 블록체인 아키텍쳐 : 비즈니스 디자인 등 큰 설계를 해야 함.

3. Hyperledger Fabric V1 Architecture



: Fabric-CA(인증서 발급 기관 : Certificate Authority)
$ cd fabric-samples/fabcar/hfc-key-store해서 들어가 보면 private과 public key가 생성됨
(외부 인증기관이던, 페브릭 인증기관이던 승인을 받아 시스템에 들어오게되면? KEY가 생성됨!)

: Endorser는 승인을 해주는 역할
SDK가 proposal(어떤 이벤트) 보낸 것에 대해 ChainCode를 실행해보고, 인증서를 검증해서 맞는지 확인(World State 이용)

: Endorser가 검증 후 SDK에 다시 보내주는 이유?
합의를 위해서(엔도서만 아는 것이 아니라 사용하는 참가자 모두에게 내용을 알려주기 위함)

: Ordering-Service에는 여러 개의 채널이 들어올 수 있음
SDK로부터 받은 내용들을 채널별, 시간별 정리 후 Leader에게 보내고 그게 Committer에게 전달 됨!

: Committer가 최종 받은 데이터를 Endorser에게 받은 결과와 같은지 비교를 통해 블록에 기록!

: Committer가 기록하는 것은 원장(Ledger)인데 이는 world state + blockchain으로 이루어짐
'get', 'put', 'delete' 'recorded'

: World State는 levelDB와 couchDB로 구성되어 있음
key:value값만 쓰여있는 DB NoSQL의 종류 중 하나로 테이블이란 개념이 없는 DB(실제 최종 값이 저장됨)
모든 데이터가 아니라 마지막 값(모든 데이터는 어짜피 블록에 쌓임)

4. Fabric Ledger

- blockchain + world state로 구성
- 채널 : peer(committer)와 node(orderer)를 연결해주는 가상공간
  채널을 나누는 등의 작업은 최초 설계 시에 해야 함.(채널을 나누는 이유? 각 체인 별 privacy)

5. Ordering Service

- 트랜잭션들을 블록으로 패키징하여 피어에게 전달하는 역할을 담당.
- 채널을 통해 Ordering Service와 통신
  서로 다른 원장 간 프라이버시를 제공함(채널 별로 또 다른 수첩이 있다고 생각하면 됨), peer는 멀티채널에 참여 할 수 있음
  orderor에 들어올 때는 채널 구분 없이 순서대로 쌓이고 → 최종 원장에 쓰여 질 때 다시 채널별로 원장이 생김
- 설정옵션(SOLO / Kafka)
  개발을 위한 하나의 노드 최소 3개의 노드가 필요(홀수노드 권장)

6. Fabric Peer

- 한 개 이상의 채널에 연결됨(각 채널에 대해서 한 개 이상의 원장을 관리)
- 체인코드는 도커 컨테이너로 분리되어 인스턴스화 되며, 채널을 통해 공유됨

7. Client Application

- 모든 클라이언트는 패브릭 SDK를 사용
- 개인키는 login할 때 쓰이는 정보고, channel은 이 생태계에 들어온 이후 쓰이는 방 번호 같은 개념

8. Transaction Flow(유통업계)

- 어부가 참치를 잡아서 app(웹사이트, 스마트폰)에 로그를 기록하고 디지털 서명을 함 → Endorser peer에게 데이터가 넘어감 → 서명에 대한 검증과 Chain code에 의한 데이터 검증 → 다시 app에 Endorser Transactions 보냄 → app은 O-service에 트랜잭션 보냄 → Orderer는 시간 순으로 순차적 정리를 통해 leader에게 보냄 → leader는 Committer에게 전송 → Committer가 최종 확인 후 채널별 블록에 기록(원장 기입)

9. 실습!!!!!!

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

: 인증서를 새로 만드는데, crypto-config.yaml로 config(구성)해서 만들어라

: 이 명령어를 실행 후 $ ll을 해보면, crypto-config라는 디렉토리가 생성됨!!(내부에는 orderer, peer 생성)

: 생성된 파일 내부로 더 들어가보면 ca/, msp/, peers/, tlsca/ 등 필요 정보들이 전부 생성된 것을 확인 가능!

```
guru@guru:~/fabric-samples/first-network/crypto-config$ ll
total 16
drwxr-xr-x 4 guru guru 4096 Nov 15 18:03 ./
drwxrwxr-x 7 guru guru 4096 Nov 15 18:03 ../
drwxr-xr-x 3 guru guru 4096 Nov 15 18:03 ordererOrganizations/
drwxr-xr-x 4 guru guru 4096 Nov 15 18:03 peerOrganizations/
```

$ FABRIC_CFG_PATH=$PWD(print working directory)

: CFG(ConFigGen)의 경로를 현재경로로 지정

$ ../bin/configtxgen -profile TwoOrgsOrdererGenesis -channelID byfn-sys-channel –outputBlock ./channel-artifacts/genesis.block

: Orderer genesis block 생성(-outputBlock : 제네시스블록이 생성된 경로와 이름)

$ export CHANNEL_NAME=mychannel

: export(리눅스에서 사용하는 환경변수 명령어)를 사용해 CHALLEL_NAME을 쓰면 자동 mychannel을 지정하도록

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

: Channel Transaction artifacts 생성

```
guru@guru:~/fabric-samples/first-network$ ../bin/cryptogen generate --config=./crypto-config.yaml
org1.example.com
org2.example.com
guru@guru:~/fabric-samples/first-network$
guru@guru:~/fabric-samples/first-network$
guru@guru:~/fabric-samples/first-network$
guru@guru:~/fabric-samples/first-network$ ../bin/configtxgen -profile TwoOrgsOrdererGenesis -channelID byfn-sys-channel -outputBlock ./channel-artifacts/genesis.block
2018-11-15 15:56:22.047 KST [common/tools/configtxgen] main -> INFO 001 Loading configuration
2018-11-15 15:56:22.233 KST [common/tools/configtxgen] doOutputBlock -> INFO 002 Generating genesis block
2018-11-15 15:56:22.237 KST [common/tools/configtxgen] doOutputBlock -> INFO 003 Writing genesis block
guru@guru:~/fabric-samples/first-network$
guru@guru:~/fabric-samples/first-network$
guru@guru:~/fabric-samples/first-network$
guru@guru:~/fabric-samples/first-network$ export CHANNEL_NAME=mychannel
guru@guru:~/fabric-samples/first-network$ ../bin/configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/channel.tx -channelID $CHANNEL_NAME
2018-11-15 16:00:26.900 KST [common/tools/configtxgen] main -> INFO 001 Loading configuration
2018-11-15 16:00:27.103 KST [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 002 Generating new channel configtx
2018-11-15 16:00:27.110 KST [common/tools/configtxgen] doOutputChannelCreateTx -> INFO 003 Writing new channel tx
```

$ cd ./channel-artifacts/ 에 가보면 파일 2개가 생성됨(channel.tx와 genesis.block)

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

```
guru@guru:~/fabric-samples/first-network$ ../bin/configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID $CHANNEL_NAME -asOrg Org1MSP
2018-11-15 16:10:21.630 KST [common/tools/configtxgen] main -> INFO 001 Loading configuration
2018-11-15 16:10:21.842 KST [common/tools/configtxgen] doOutputAnchorPeersUpdate -> INFO 002 Generating anchor peer update
2018-11-15 16:10:21.845 KST [common/tools/configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Writing anchor peer update
guru@guru:~/fabric-samples/first-network$
guru@guru:~/fabric-samples/first-network$ ../bin/configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID $CHANNEL_NAME -asOrg Org2MSP
2018-11-15 16:10:51.112 KST [common/tools/configtxgen] main -> INFO 001 Loading configuration
2018-11-15 16:10:51.315 KST [common/tools/configtxgen] doOutputAnchorPeersUpdate -> INFO 002 Generating anchor peer update
2018-11-15 16:10:51.318 KST [common/tools/configtxgen] doOutputAnchorPeersUpdate -> INFO 003 Writing anchor peer update
```

: 두 가지 조직(Org1, 2)에 각각 anchor를 지정하는 작업(peer 두 개 중 하나를 앵커로 지정)

$ docker-compose -f docker-compose-cli.yaml up –d

: Network 기동하면 6개(피어4개, orderer, cli가 켜짐)

```
guru@guru:~/fabric-samples/first-network$ docker-compose -f docker-compose-cli.yaml up -d
Starting peer0.org2.example.com
Starting peer1.org1.example.com
Starting peer0.org1.example.com
Starting orderer.example.com
Starting peer1.org2.example.com
Starting cli
```

docker-compose-cli.yaml파일을 실행하는데 up이라는 명령어를 사용해서 –d(데몬 / 백에서 돌리겠다)로 실행

$ docker ps → 해보면 총 6개의 process가 기동중(peer2개 / orderer / cli)

```
guru@guru:~/fabric-samples/first-network$ docker ps
CONTAINER ID        IMAGE                               COMMAND             CREATED             STATUS              PORTS                                            NAMES
e85aa81f3251        hyperledger/fabric-tools:latest     "/bin/bash"         2 hours ago         Up 18 seconds                                                        cli
66c0e0ba7baa        hyperledger/fabric-orderer:latest   "orderer"           2 hours ago         Up 19 seconds       0.0.0.0:7050->7050/tcp                           orderer.example.com
0cca1bebcb0d        hyperledger/fabric-peer:latest      "peer node start"   2 hours ago         Up 21 seconds       0.0.0.0:7051->7051/tcp, 0.0.0.0:7053->7053/tcp   peer0.org1.example.com
ae91525618bf        hyperledger/fabric-peer:latest      "peer node start"   2 hours ago         Up 20 seconds       0.0.0.0:10051->7051/tcp, 0.0.0.0:10053->7053/tcp peer1.org2.example.com
c4732d1c5129        hyperledger/fabric-peer:latest      "peer node start"   2 hours ago         Up 19 seconds       0.0.0.0:9051->7051/tcp, 0.0.0.0:9053->7053/tcp   peer0.org2.example.com
cd1dca3df30e        hyperledger/fabric-peer:latest      "peer node start"   2 hours ago         Up 20 seconds       0.0.0.0:8051->7051/tcp, 0.0.0.0:8053->7053/tcp   peer1.org1.example.com
```

$ docker exec -it cli bash

: Docker 실행(CLI) why? 전부 도커에서 돌아가므로 ubuntu에서 docker로 이동(guru → peer로 바뀐 것 확인)

```
guru@guru:~/fabric-samples/first-network$ docker exec -it cli bash
root@e85aa81f3251:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

cd crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem 이 경로 밑에 인증서 파일이 존재!!(cert 파일)

$ export CHANNEL_NAME=mychannel

: docker와 ubuntu는 다르기 때문에 docker에서도 한 번 더 환경설정을 해줘야 함!
: 안 해주면 아래 명령어 작성 시 오류 발생!! 꼭!!!!!써주기

$ peer channel create -o orderer.example.com:7050 -c $CHANNEL_NAME –f ./channel-artifacts/channel.tx --tls $CORE_PEER_TSL_ENABLED —cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem

: 채널 생성해주는 코드 인증서 이름을 설정하고 해당 경로를 지정해줌(--tls는 대표적인 인증서를 나타내는 옵션)

$ ls 해서 mychnnel.block이 나오면 성공!!

안되면? 2개 지우고 다시

* crypto-config는 rm –rf로 디렉토리 자체를 날려버리고
* channel은 cd로 파일 안에 이동해서 내부 파일만 rm –rf *