< 하이퍼레져 페브릭 실습2 >

1. 합의 알고리즘

- execute → order → validate → update state

kafka(Apache 서비스 중의 하나로 메시지 큐 서비스)로 순차적으로 발생한 event(Tx)에 대해서 시간 순으로 순차적 정렬
합의 관련 내용은 이미 execute 단계에서 endorser가 실시하기 때문에 order는 별도의 검증 과정을 행하지 않음

- https://hyperledger-fabric.readthedocs.io/en/release-1.3/command_ref.html
  위 사이트에서 명령어 관련 옵션 정보들 확인 가능!!(모르면 들어가서 읽어보자)

2. 어제 실습에 이어서(channel join부터)

```
## peer0.org1(4줄 +1)
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 join –b mychannel.block

## 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:7051
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 mychannel.block

## 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:7051
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 mychannel.block

## 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:7051
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 mychannel.block
```

위 명령어를 문단별로 총 4번 입력해보면?

아래와 같이 endorser와 orderer의 연결이 초기화 되고, peer가 성공적으로 채널에 들어가짐!

```
root@f831468920b3:/opt/gopath/src/github.com/hyperledger/fabric/peer# CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
root@f831468920b3:/opt/gopath/src/github.com/hyperledger/fabric/peer# CORE_PEER_ADDRESS=peer1.org1.example.com:7051
root@f831468920b3:/opt/gopath/src/github.com/hyperledger/fabric/peer# CORE_PEER_LOCALMSPID="Org1MSP"
root@f831468920b3:/opt/gopath/src/github.com/hyperledger/fabric/peer# 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.c
rt
root@f831468920b3:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer channel join -b mychannel.block
2018-11-16 00:55:45.772 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2018-11-16 00:55:45.917 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel
```

$ peer channel getinfo –c mychannel

```
root@f831468920b3:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer channel getinfo -c mychannel
2018-11-16 00:57:40.757 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
Blockchain info: {"height":1,"currentBlockHash":"Qpw1UXOa6hxFSmhg9KKUJk7APgEl1okZcoDac7Eml0c="}
```

: 현재 채널의 정보를 출력해보면 내용이 나옴(height라고 하는 이유는 peer들을 쌓기 때문)

$ peer channel update -o orderer.example.com:7050 -c mychannel –f ./channel-artifacts/Org1MSPanchors.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

: anchor peer의 정보를 update해주는 작업(peer0.org1, peer0.org2를 update)
(Org1MSPanchors.tx 파일이 필요한 정보를 가지고 있으므로 이를 anchor-peer와 연결해주는 작업)
(만약 에러가 난다면? 환경변수 설정이 잘못 되서 나는 것이기 때문에 위에 채널에 peer를 join해주는 4줄짜리 코드를 한 번 더 써서 지정해주고 다시 한 번 업데이트!!)

```
guru@guru:~/fabric-samples/first-network/channel-artifacts$ ll
total 36
drwxrwxr-x 2 guru guru  4096 Nov 15 18:05 ./
drwxrwxr-x 7 guru guru  4096 Nov 15 18:03 ../
-rw-r--r-- 1 guru guru   346 Nov 15 18:04 channel.tx
-rw-r--r-- 1 guru guru 12778 Nov 15 18:03 genesis.block
-rw-rw-r-- 1 guru guru     0 Nov 14 13:42 .gitkeep
-rw-r--r-- 1 guru guru   284 Nov 15 18:04 Org1MSPanchors.tx
-rw-r--r-- 1 guru guru   284 Nov 15 18:05 Org2MSPanchors.tx
```

```
root@bfae1121bd9d:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer channel getinfo -c mychannel
2018-11-16 01:57:03.345 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
Blockchain info: {"height":3,"currentBlockHash":"E88XgSvJd0N4Fg3gHpbnrh2w+NNMGSwwN6+uME90+qw=","previousBlockHash":"sPLOrR8qvXeP4OXfBZACUdR52xbHZPhYC1GPJ0y5uIM="}
```

: 다시 해보면 anchor peer 두 개가 추가되어 height가 3이 됨을 볼 수 있음!!

$ sudo apt-get install tree

```
guru@guru:~/fabric-samples/chaincode/chaincode_example02/node$ rmate -p 52698 chaincode_example02.js
```

: 파일 구조를 tree형식으로 보기 위해 설치

: 이렇게 해서 chaincode_example02.js 파일을 열어보면 init, invoke, delete, query, main 등의 함수 안에 이체 정보(누구에게 +, 누구에게 -, 금액받기 등), set, get 등이 담겨있음.

```
guru@guru:~/fabric-samples/chaincode/chaincode_example02$ ll
total 20
drwxrwxr-x 5 guru guru 4096 Nov 14 13:42 ./
drwxrwxr-x 9 guru guru 4096 Nov 14 16:45 ../
drwxrwxr-x 2 guru guru 4096 Nov 14 13:42 go/
drwxrwxr-x 3 guru guru 4096 Nov 14 13:42 java/
drwxrwxr-x 2 guru guru 4096 Nov 14 13:42 node/
```

: chaincode 파일은 chaincode_example02에 다 들어있는데 총 3개 모드(nodejs, JAVA, golan)
: 각각의 디렉토리로 들어가서 $ rmate –p 52698 [해당경로] 로 확인가능!!(아래는 JAVA 코드)

```
public class SimpleChaincode extends ChaincodeBase {

    private static Log _logger = LogFactory.getLog(SimpleChaincode.class);

    @Override
    public Response init(ChaincodeStub stub) {…
    }

    @Override
    public Response invoke(ChaincodeStub stub) {…
    }

    private Response invoke(ChaincodeStub stub, List<String> args) {…
    }

    // Deletes an entity from state
    private Response delete(ChaincodeStub stub, List<String> args) {…
    }

    // query callback representing the query of a chaincode
    private Response query(ChaincodeStub stub, List<String> args) {…
    }

    public static void main(String[] args) {…
    }

}
```

```
root@bfae1121bd9d:/opt/gopath/src/github.com/chaincode# peer
Usage:
  peer [command]

Available Commands:
  chaincode   Operate a chaincode: install|instantiate|invoke|package|query|signpackage|upgrade|list.
  channel     Operate a channel: create|fetch|join|list|update|signconfigtx|getinfo.
  help        Help about any command
  logging     Log levels: getlevel|setlevel|revertlevels.
  node        Operate a peer node: start|status.
  version     Print fabric peer version.

Flags:
  -h, --help                    help for peer
      --logging-level string    Default logging level and overrides, see core.yaml for full syntax

Use "peer [command] --help" for more information about a command.
```

: docker로 들어가서 github.com/chaincode로 이동하면 ubuntu와 같은 파일들이 나와 있음.

: peer라는 명령어를 쳐보면 아래 사용할 수 있는 옵션들이 쭉 나옴(주로 사용하는 것은 네모박스 3가지)

$ peer chaincode install -n mycc_go -v 1.0 -p /opt/gopath/src/github.com/chaincode/chaincode_example02/go

```
root@bfae1121bd9d:/opt/gopath/src/github.com/chaincode# peer chaincode install -n mycc_go -v 1.0 -p github.com/chaincode/chaincode_example02/go
2018-11-16 02:45:58.491 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2018-11-16 02:45:58.493 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2018-11-16 02:45:59.769 UTC [chaincodeCmd] install -> INFO 003 Installed remotely response:<status:200 payload:"OK" >
```

: peer에 chaincode를 install해주는 작업(이후 instance도 만들어 줄꺼!)

: chaincode는 같은 이름, 버전으로 올리면 ERROR가 발생하므로 반드시 이름 바꿔줘야 함!!

: go는 기본값이므로 그냥 작성하되, nodejs나 java의 경우 –l 옵션을 활용해 언어를 명시해주고 체인코드 이름과 마지막 경로를 바꿔주고 install 할 것!(마지막에 200 뜨면 성공)

$ peer chaincode list --installed

```
root@bfae1121bd9d:/opt/gopath/src/github.com/chaincode# peer chaincode list --installed
Get installed chaincodes on peer:
Name: mycc_go, Version: 1.0, Path: github.com/chaincode/chaincode_example02/go, Id: 8e46fdccc656ee04213f3f6adec74f15285a068f7e090add6262f0970a697bd8
Name: mycc_java, Version: 1.0, Path: /opt/gopath/src/github.com/chaincode/chaincode_example02/java, Id: d3c12621c4b531bb3e9c5599c3b563116d3a7ff99ec5d2e0f43de420ef664efe
Name: mycc_node, Version: 1.0, Path: /opt/gopath/src/github.com/chaincode/chaincode_example02/node, Id: aeea614a690ffee63bbfbcc4ea23c2120a1bd44a65ae5f4728c69b36e171cf6c
```

: 설치된 chaincode list를 출력하면 3개가 정상 설치 된 것을 알 수 있다.

$ peer chaincode instantiate –o orderer.example.com:7050 --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 –C mychannel –n mycc_go –v 1.0 –c '{"Args":["init", "a", "100", "b", "200"]}' -P "AND('Org1MSP.member', 'Org2MSP.member')"

: 앞에서 install 한 golang 체인코드에 instance를 만들어주는 작업(-c는 생성자에 전달되는 초기화 값)

: 사전에 반드시!!! peer의 환경변수를 지정해주고 실행할 것(peer1~4까지 환경변수 써 준 이후 코드 작성)

```
root@745948a77b75:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode instantiate -o orderer.example.com:7050 --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 -C mychannel -n mycc_go -v 1.0 -c '{"Args":["init", "a", "100", "b", "200"]}' -P "AND ('Org1MSP.member', 'Org2MSP.member')"
2018-11-16 05:18:44.637 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2018-11-16 05:18:44.637 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
root@745948a77b75:/opt/gopath/src/github.com/hyperledger/fabric/peer#
root@745948a77b75:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode list --installed
Get installed chaincodes on peer:
Name: mycc_go, Version: 1.0, Path: github.com/chaincode/chaincode_example02/go, Id: 8e46fdccc656ee0421373f6adec74f15285a0687fe090add6262f0970a697bd8
root@745948a77b75:/opt/gopath/src/github.com/hyperledger/fabric/peer#
root@745948a77b75:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode list --instantiated -C mychannel
Get instantiated chaincodes on channel mychannel:
Name: mycc_go, Version: 1.0, Path: github.com/chaincode/chaincode_example02/go, Escc: escc, Vscc: vscc
```

: escc, vscc가 default로하여 instance가 생성되었으며, **$ peer chaincode list —installed**로 확인해보면 mycc_go 있음.

: **$ peer chaincode list —instantiated –C mychannel**로 확인해보면 mychannel에 생성된 instance가 보임.

: **ESCC(Endorser System Chaincode) : 그림에서 2~3번 과정 수행 시 호출되는 CC**

* 트랜잭션을 실행한 후 트랜잭션 실행 결과(트랜잭션 상태, 체인 코드 이벤트, reat/write set 등)를 포함하는 트랜잭션 응답 메시지에 서명을 넣기 위해 엔도싱 피어에 의해 호출
* invoke 함수는 5 - 7 개의 매개 변수를 수용 / 이 매개 변수는 Header, ChaincodeProposalPayload, ChaincodeID, Response, 시뮬레이션 결과, 이벤트 및 페이로드 가시성(payload visibility)

: **VSCC(Validator System Chaincode) : 그림에서 7번 과정 수행 시 호출되는 CC**

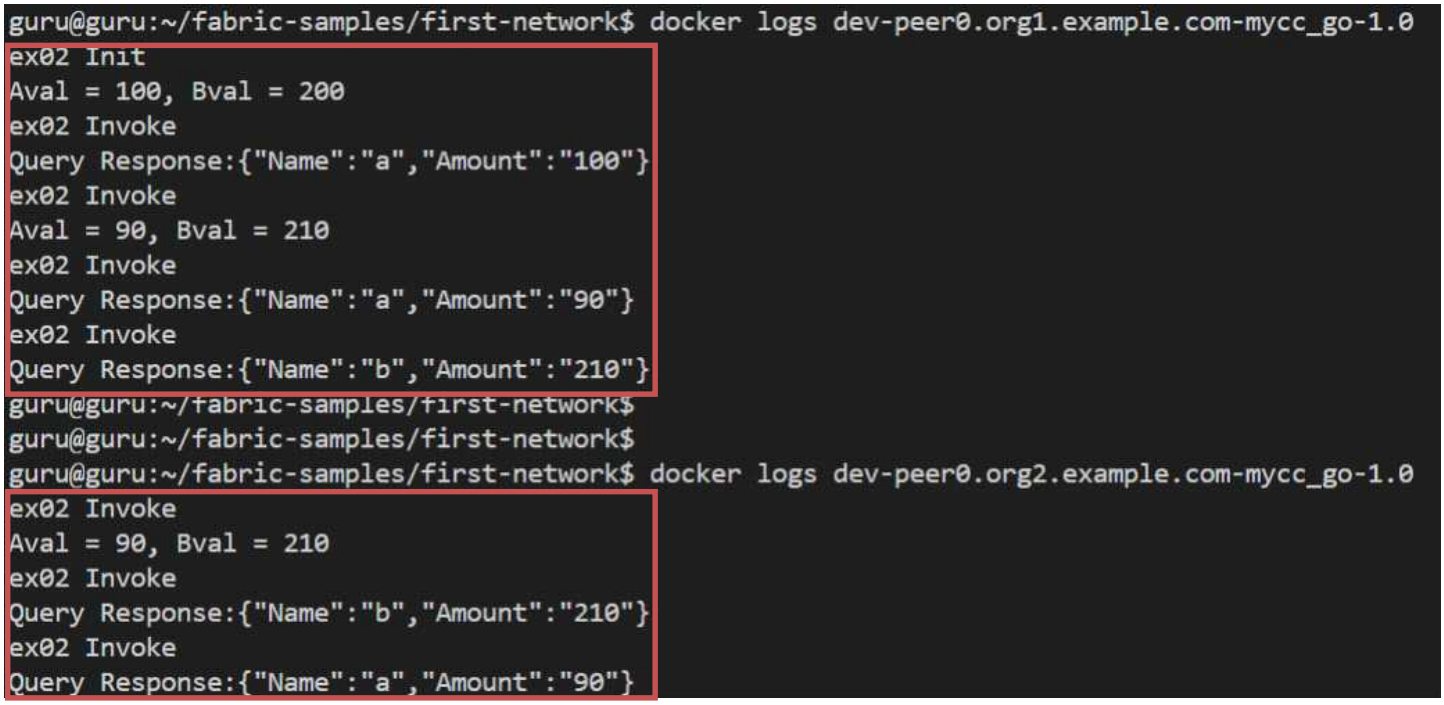
* committing 피어 (core/committer/txvalidator/validator.go)에 의해 호출되어 각 트랜잭션의 서명 세트를 체인 코드의 보증 정책에 대해 유효성을 검사

$ peer chaincode invoke -o orderer.example.com:7050 --tls true —cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME –n mycc_go --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:7051 —tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt –c '{"Args":["invoke","a","b","10"]}'

```
root@745948a77b75:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode invoke -o orderer.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C $CHANNEL_NAME -n mycc_go --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:7051 --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"Args":["invoke","a","b","10"]}'
2018-11-16 05:48:44.858 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
root@745948a77b75:/opt/gopath/src/github.com/hyperledger/fabric/peer#
root@745948a77b75:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode query -C mychannel -n mycc_go -c '{"Args":["query","a"]}'
90
root@745948a77b75:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode query -C mychannel -n mycc_go -c '{"Args":["query","b"]}'
210
```

: a가 b에게 10을 보내는 invoke 함수를 실행하면 status : 200과 함께 완료되었다는 메시지 띄움.

: **$ peer chaincode query –C mychannel –n mycc_go –c '{"Args":["query", "a"]}'와 "b"**로 확인해보면 각각 90과 210나와야함(금액 전송결과) → peer를 바꿔서 실행해 봐도 다 똑같음(broadcast 완료!!)



: 잘 전송이 되었는지 log정보를 통해 확인해보기(**$ docker logs dev-peer0.org1.example.com-mycc_go-1.0**)

: peer0에서 실행했기 때문에 Init부터 Invoke까지 모든 정보가 뜸!

: peer3에서 확인해보면 Invoke만 뜸!(앵커피어로 peer0에서 해당 정보를 전달받은 이후부터 출력)

3. 교재 실습!!

- p214(오픈이지/하이퍼레져 패브릭으로 구현하는 블록체인)에 나와 있는 https://github.com/IBM-Blockchain/marbles 과제 해보기!