

< 하이퍼레저 페브릭4 >

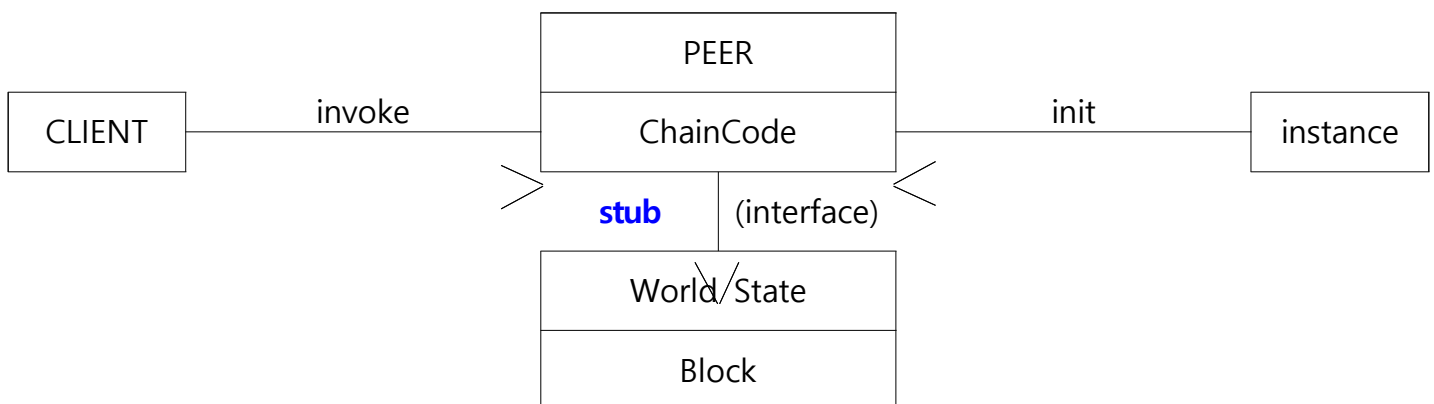
1. go lang의 문법

```
package main

import (
    "fmt"
    "strconv"
    "github.com/hyperledger/fabric/core/chaincode/shim"
    "github.com/hyperledger/fabric/protos/peer"
)

// SimpleAsset implements a simple chaincode to manage an asset
type SimpleAsset struct {
}
```

- package main과 import는 필수적으로 써줘야 하는 것(just 문법)
 - : fmt는 c언어의 <stdio.h>와 같은 역할(print 등을 쓰게 해주는 표준 라이브러리 - debug, logging)
- shim은 연결해주는 interface(약속) 역할 => client와 chaincode를 연결해주는 역할
 - : shim은 spacer로 위,아래 공간을 마련해주기 위한 장치(=즉, 체인코드를 블록체인 네트워크와 연결해주는 역할)
- peer는 원장에 최종 작성을 하기 위해 작성해주는 것
- type SimpleAsset struct는 class의 개념으로 생각(일종의 상자 같은 개념)
 - : 내가 필요한 자료형을 만드는 것(실습 간 예제에서는 내용을 채우지 않았지만, marbles에서 쓰이는 원형 같은 것들이 이 안에 쓰임)
- **\$ func (t *SimpleAsset) Init(stub shim.ChaincodeStubInterface) peer.Response**
 - python의 self같은 역할(SimpleAsset 구조체 안에서 쓰겠다) 반환해주는 함수(Response로 return)
- stub은 대리 역할을 해주는 agent라고 생각
 - : client가 invoke를 통해 CC를 호출하면, CC가 블록(World State)에 접근하기 위해 쓰이는 매개체
 - : instance를 만들기 위해 init을 호출할 때도 마찬가지(문을 여는 열쇠 같은 개념으로 생각)
 - : 밖에서(client) 쓰인 변수를 안으로(world state) 가져와 쓰기 위해서 거쳐야 할 interface!!



2. ChainCode 수정 후 버전 업그레이드

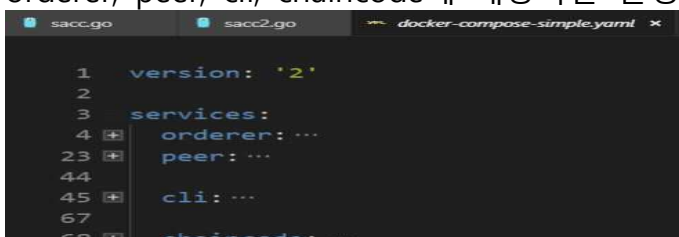
- fabric-samples에서는 기본으로 제공해주는 개발자용 네트워크가 있음(chaincode-docker-devmode)

```
guru@guru:~/fabric-samples$ ll
total 96
drwxrwxr-x 14 guru guru 4096 Nov 15 13:20 ./
drwxr-xr-x 9 guru guru 4096 Nov 15 14:39 ../
drwxrwxr-x 5 guru guru 4096 Nov 14 13:42 balance-transfer/
drwxrwxr-x 4 guru guru 4096 Nov 14 13:42 basic-network/
drwxrwxr-x 2 guru guru 4096 Oct 10 23:55 bin/
drwxrwxr-x 10 guru guru 4096 Nov 19 14:54 chaincode/
drwxrwxr-x 4 guru guru 4096 Nov 19 14:32 chaincode-docker-devmode/
-rw-rw-r-- 1 guru guru 597 Nov 14 13:42 CODE_OF_CONDUCT.md
drwxrwxr-x 2 guru guru 4096 Oct 10 23:41 config/
-rw-rw-r-- 1 guru guru 961 Nov 14 13:42 CONTRIBUTING.md
drwxrwxr-x 4 guru guru 4096 Nov 15 14:04 fabcar/
drwxrwxr-x 3 guru guru 4096 Nov 14 13:42 fabric-ca/
drwxrwxr-x 8 guru guru 4096 Nov 16 16:24 first-network/
drwxrwxr-x 8 guru guru 4096 Nov 14 13:42 .git/
-rw-rw-r-- 1 guru guru 130 Nov 14 13:42 .gitignore
-rw-rw-r-- 1 guru guru 109 Nov 14 13:42 .gitreview
drwxrwxr-x 4 guru guru 4096 Nov 14 13:42 high-throughput/
-rw-rw-r-- 1 guru guru 3193 Nov 14 13:42 Jenkinsfile
-rw-rw-r-- 1 guru guru 11358 Nov 14 13:42 LICENSE
-rw-rw-r-- 1 guru guru 470 Nov 14 13:42 MAINTAINERS.md
-rw-rw-r-- 1 guru guru 1230 Nov 14 13:42 README.md
drwxrwxr-x 3 guru guru 4096 Nov 14 13:42 scripts/
```

- 이 안에 들어가 보면 myc.block, tx, orderer 등 기본 설정해줘야 할 파일들이 이미 생성되어 들어가 있음.

```
guru@guru:~/fabric-samples/chaincode-docker-devmode$ ll
total 60
drwxrwxr-x 4 guru guru 4096 Nov 19 14:32 ./
drwxrwxr-x 14 guru guru 4096 Nov 15 13:20 ../
drwxr-xr-x 2 root root 4096 Nov 19 14:32 chaincode/
-rw-rw-r-- 1 guru guru 2702 Nov 14 13:42 docker-compose-simple.yaml
-rw-rw-r-- 1 guru guru 83 Nov 14 13:42 .gitignore
drwxrwxr-x 8 guru guru 4096 Nov 14 13:42 msp/
-rw-r--r-- 1 root root 10876 Nov 19 14:32 myc.block
-rw-rw-r-- 1 guru guru 274 Nov 14 13:42 myc.tx
-rw-rw-r-- 1 guru guru 7902 Nov 14 13:42 orderer.block
-rw-rw-r-- 1 guru guru 5270 Nov 14 13:42 README.rst
-rwxrwxr-x 1 guru guru 1019 Nov 14 13:42 script.sh*
```

- \$ rmate -p 52698 docker-compose-simple.yaml로 띄워서 보면
orderer, peer, cli, chaincode에 해당하는 환경변수 값, 포트번호 등이 다 작성되어 있음



```
sacc.go sacc2.go docker-compose-simple.yaml x
1 version: '2'
2
3 services:
4   orderer: ...
23   peer: ...
44
45   cli: ...
67
68   chaincode: ...
```

- \$ **docker-compose -f docker-compose-simple.yaml up -d**로 네트워크 기동(up)
- \$ **docker exec -it chaincode bash**로 체인코드 컨테이너로 이동!
- \$ **CORE_PEER_ADDRESS=peer:7052 CORE_CHAINCODE_ID_NAME=mycc:2 ./sacc2**
: 네트워크 기본값들을 설정(주소는 7052포트를 이용하고, 체인코드 이름은 mycc, 버전은 2, ./sacc2파일로 실행하겠다)
라고 엔터를 치면 'starting up'으로 시작이 됨

```
root@a9f8766e7797:/opt/gopath/src/chaincode/sacc2# CORE_PEER_ADDRESS=peer:7052 CORE_CHAINCODE_ID_NAME=mycc:2 ./sacc2
2018-11-20 01:44:00.114 UTC [shim] setupChaincodeLogging -> INFO 001 Chaincode log level not provided; defaulting to: INFO
2018-11-20 01:44:00.117 UTC [shim] setupChaincodeLogging -> INFO 002 Chaincode (build level: ) starting up ...
```

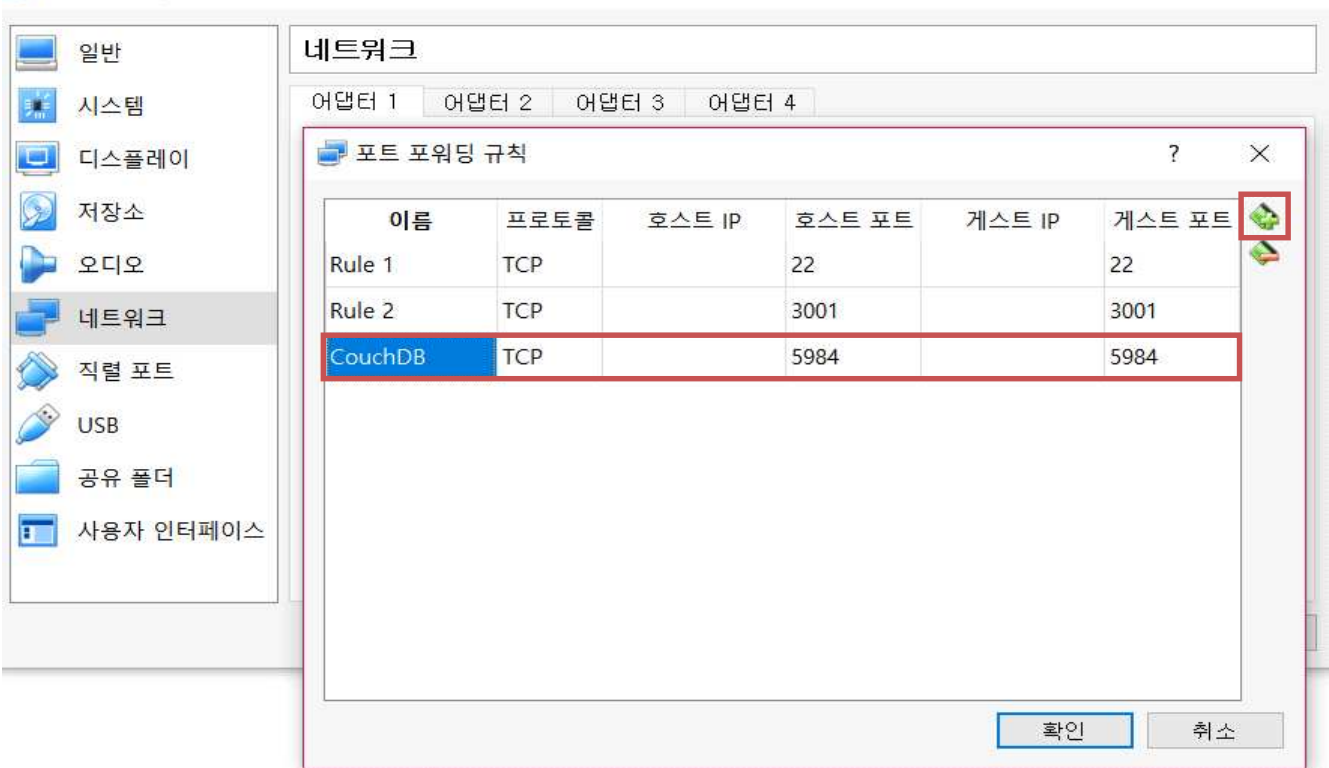
- \$ **peer chaincode install -p chaincodedev/chaincode/sacc2 -n mycc -v 2**로 설치
- \$ **peer chaincode upgrade -n mycc -v 2 -C myc -c '{"Args":["eo","10"]}'**로 업그레이드
: instantiate해도 되긴하는데 기존에 각 node별로 전파된 것을 다 수정해줘야하므로 upgrade 추천
(굳이 새로 instantiate 하지 않아도 됨)
- \$ **peer chaincode list —installed / peer chaincode list —instantiated -C myc**
: 기존에 설치된 모든 내용과 현재 instance화 된 최근 정보가 뜸
- 이후에는 **invoke(set, get, transfer, delete)**를 이용해 잘 작동하는지 실습!!

2. Couch DB로 실습하기

- 최초 ubuntu server의 설정 변경(port번호 추가)

: 맨 마지막에 localhost:5984/_utils로 들어가보면 couchDB로 만든 서버 확인 가능!

ubuntu01_up_1 - 설정



- \$ **docker ps -a**로 아무것도 없는 것 확인
- \$ **rmate -p 52698 marbles_chaincode.go**를 통해 해당 파일 띄우기
- CRUD(insert / select / update / delete)
put state get state del state
- 실습 전에 코드 분석 먼저(marbles_chaincode.go)

```
// === Save marble to state ===
err = stub.PutState(marbleName, marbleJSONasBytes)
if err != nil {
    return shim.Error(err.Error())
}
```

: 뒤에 넣는 값은 JSON형식의 Byte 타입을 그대로 넣겠다

```
marbleJSONasBytes, err := json.Marshal(marble)
```

: Marshal이란 함수는 JSON형식의 format을 Byte 배열로 바꿔주는 함수(↔UnMarshal)

: 이렇게하면 CouchDB에 그대로 document가 들어갈 수 있음!

이는 굉장히 많은 의미를 가짐

(데이터를 검색할 수 있는 많은 방법을 제공하고, data를 중첩적으로 관리할 수 있음)

```
err := stub.PutState(args[0], []byte(args[1]))
if err != nil {
    return shim.Error(fmt.Sprintf("Failed to create asset: %s", args[0]))
}
```

: 기존 예제(sacc2)에서는 이렇게 PutState 뒤에 []byte(args[1]) 값을 넣었음!! -> levelDB 사용한 것

```
err = json.Unmarshal([]byte(valAsBytes), &marbleJSON)
```

: GetState로 받아온 값을 Unmarshal을 통해 byte배열로 바꿔줌

- fabric-samples/first-network 디렉토리 안에는 다음과 같은 파일들이 있고, 네트워크 기동에 이용!

```
guru@guru:~/fabric-samples/first-network$ ll
total 120
drwxrwxr-x 8 guru guru 4096 Nov 16 16:24 ./
drwxrwxr-x 14 guru guru 4096 Nov 15 13:20 ../
drwxrwxr-x 2 guru guru 4096 Nov 14 13:42 base/
-rwxrwxr-x 1 guru guru 20636 Nov 14 13:42 byfn.sh*
drwxrwxr-x 2 guru guru 4096 Nov 16 14:14 channel-artifacts/
-rw-rw-r-- 1 guru guru 12265 Nov 14 13:42 configtx.yaml
drwxr-xr-x 4 guru guru 4096 Nov 16 14:14 crypto-config/
-rw-rw-r-- 1 guru guru 3906 Nov 14 13:42 crypto-config.yaml
-rw-rw-r-- 1 guru guru 2971 Nov 14 13:42 docker-compose-cli.yaml
-rw-rw-r-- 1 guru guru 2345 Nov 14 13:42 docker-compose-couch-org3.yaml
-rw-rw-r-- 1 guru guru 4560 Nov 14 13:42 docker-compose-couch.yaml
-rw-rw-r-- 1 guru guru 2883 Nov 14 13:42 docker-compose-e2e-template.yaml
-rw-rw-r-- 1 guru guru 3801 Nov 14 13:42 docker-compose-org3.yaml
-rw-rw-r-- 1 guru guru 42 Nov 14 13:42 .env
-rwxrwxr-x 1 guru guru 10409 Nov 14 13:42 eyfn.sh*
drwxrwxr-x 14 guru guru 4096 Nov 16 16:25 fabric-samples/
-rw-rw-r-- 1 guru guru 118 Nov 14 13:42 .gitignore
drwxrwxr-x 2 guru guru 4096 Nov 14 13:42 org3-artifacts/
-rw-rw-r-- 1 guru guru 335 Nov 14 13:42 README.md
drwxrwxr-x 2 guru guru 4096 Nov 14 13:42 scripts/
```

- 네트워크를 기동하기 전에 인증서, genesis블록, 채널생성, 앵커피어 등록의 작업을 실시. 완료 이후 네트워크 기동(아래)

- **docker-compose -f docker-compose-cli.yaml -f docker-compose-couch.yaml up -d**로 네트워크 기동

: 네트워크 기동은 cli를 이용해서 하고 DB는 couch를 사용하겠다!(그래서 2개 / default는 levelDB)

: 가장 쉬운 네트워크 기동 : byfn.sh(levelDB)이지만, peer를 추가하고 channel에 peer들을 가입시키기

위해서 docker-compose-couch.yaml을 사용해 네트워크 기동! **교재 p146 참고해서 실습**

: 네트워크 기동하면 최초에는 create ~~ 뜨지만 내용 수정 후 다시 기동하면 up-to-date 뜸!

```
guru@guru:~/fabric-samples/first-network$ docker-compose -f docker-compose-cli.yaml -f docker-compose-couch.yaml up -d
Creating network "net_byfn" with the default driver
Creating volume "net_peer0.org2.example.com" with default driver
Creating volume "net_peer1.org2.example.com" with default driver
Creating volume "net_peer1.org1.example.com" with default driver
Creating volume "net_peer0.org1.example.com" with default driver
Creating volume "net_orderer.example.com" with default driver
Creating couchdb0
Creating couchdb1
Creating orderer.example.com
Creating couchdb2
Creating couchdb3
Creating peer0.org2.example.com
Creating peer1.org2.example.com
Creating peer1.org1.example.com
Creating peer0.org1.example.com
Creating cli
```

- 만약 잘못했을 경우??

\$ **rm -rf crypto-config**로 인증서 파일을 모두 지우고, \$ **cd channel-artifacts/**로 이동해

\$ **rm -rf *** 내용을 모두 지움(디렉토리는 반드시! 남겨둘 것)

또는, first_network에서 \$ **./byfn.sh -m down** 하고 재시작!(CA생성~)해도 됨.

- \$**docker exec -it cli bash**이용 docker로 들어가서 peer에 install 및 instance화 작업 실시.

: github 참고(https://github.com/joneconsulting/bc/blob/master/BYFN_couchdb_cmd.txt#L7)

The screenshot shows the CouchDB Databases interface. The table lists the following databases:

Name	Size	# of Docs	Actions
_replicator	2.3 KB	1	[Icons]
_users	2.1 KB	1	[Icons]
mychannel_	13.7 KB	2	[Icons]
mychannel_iscc	0.6 KB	1	[Icons]
mychannel_marbles	3.4 KB	7	[Icons]

The 'mychannel_marbles' row is highlighted with a red box. The interface also shows a sidebar with navigation icons and a top bar with the database name and a 'Create Database' button.