3일차 실습

- **패브릭 네트워크 확장**

① 설정파일 업데이트

- configtx.yaml
- crypto-config.yaml
- docker-compose.yml

② 체인코드 관리

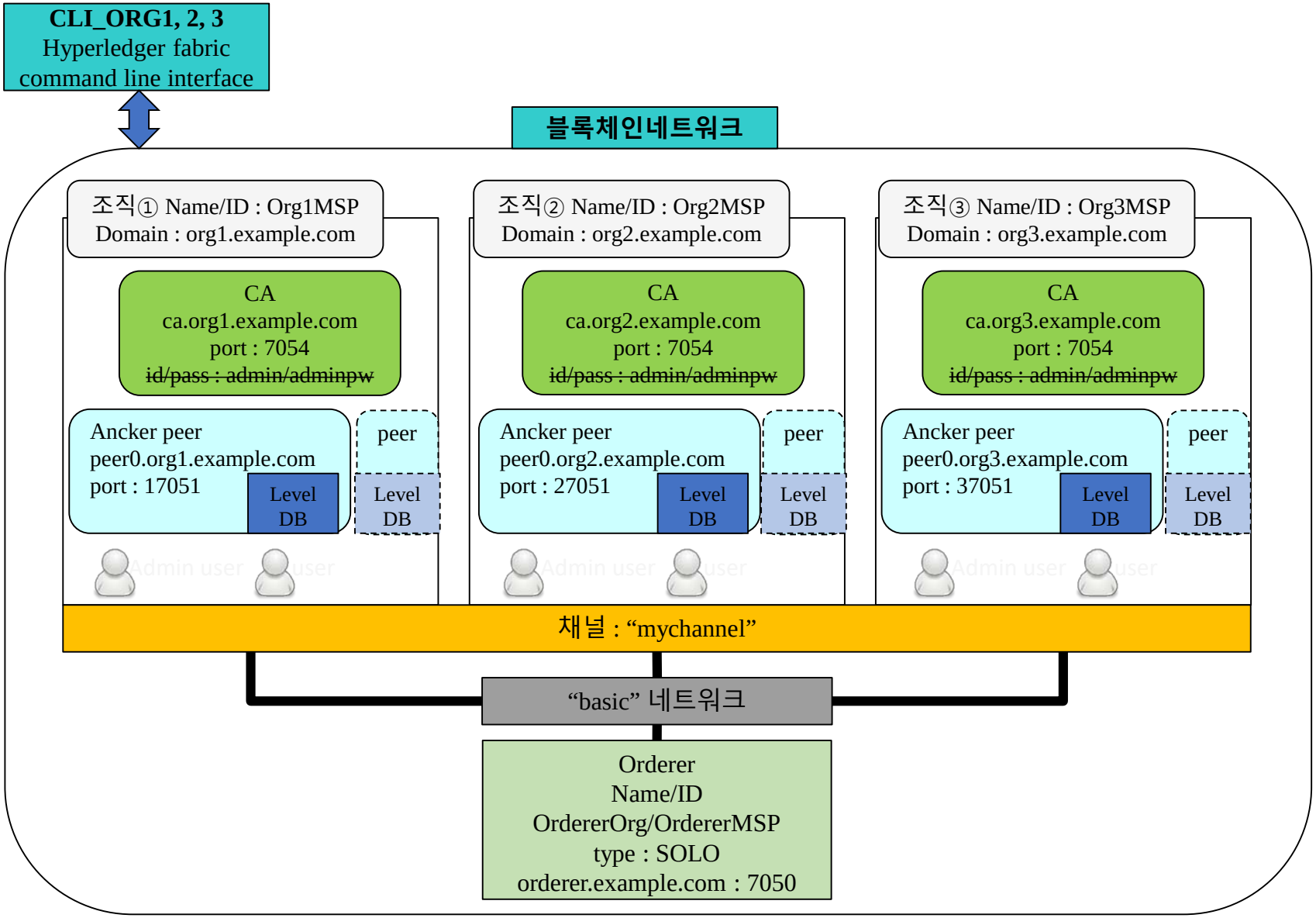
① 설정파일 업데이트

- configtx.yaml
- crypto-config.yaml
- docker-compose.yml

네트워크구성 확장 개요

- 목표
  - Basic-network를 수정하여 다른 형태의 네트워크 구성
    - 3개의 Organization
    - 각 Organization에 CA 구성
    - 각 Organization에 ancker peer 구성
    - TLS 암호화 통신 구현
    - CouchDB대신 LevelDB사용

네트워크구성 확장 개요


CLI_ORG1, 2, 3
Hyperledger fabric
command line interface
블록체인네트워크
조직① Name/ID : Org1MSP
Domain : org1.example.com
조직② Name/ID : Org2MSP
Domain : org2.example.com
조직③ Name/ID : Org3MSP
Domain : org3.example.com
CA
ca.org1.example.com
port : 7054
id/pass : admin/adminpw
CA
ca.org2.example.com
port : 7054
id/pass : admin/adminpw
CA
ca.org3.example.com
port : 7054
id/pass : admin/adminpw
Ancker peer
peer0.org1.example.com
port : 17051
Level DB
peer
Level DB
Ancker peer
peer0.org2.example.com
port : 27051
Level DB
peer
Level DB
Ancker peer
peer0.org3.example.com
port : 37051
Level DB
peer
Level DB
Admin user
user
Admin user
user
Admin user
user
채널 : “mychannel”
“basic” 네트워크
Orderer
Name/ID
OrdererOrg/OrdererMSP
type : SOLO
orderer.example.com : 7050

네트워크구성 확장 개요

- 양식 복사 및 수정
  - 새로운 네트워크 생성의 양식인 'basic-network'복사
  - 홈 디렉토리에 'fabricbook'디렉토리를 생성하여 복사
    ~$cp -r fabric-samples/basic-network/ fabricbook
    ~$cd fabricbook/
    ~/fabricbook$ls

```
bstudent@bstudent-VirtualBox:~/fabricbook$ ls
README.md  config  configtx.yaml  crypto-config  crypto-config.yaml  docker-compose.yml
  generate.sh  init.sh  start.sh  stop.sh  teardown.sh
```

네트워크구성 확장 개요

- 네트워크 확장시 작성 필요한 파일
  - configtx.yaml
  - crypto-config.yaml
  - generate.sh
  - docker-compose.yml
  - start.sh
  - teardown.sh

실습

- ~ 디렉토리 내 새로운 디렉토리 만들기
  - cd ~
  - mkdir fabbook
  - cd fabbook
- 새로 생성된 디렉토리에 초기 파일들 복사
  - cd ~/fabric-samples/basic-network
  - cp configtx.yaml crypto-config.yaml docker-compose.yml .env *.sh ~/fabricbook

```
bstudent@bstudent-VirtualBox:~/fabricbook$ ls
configtx.yaml       docker-compose.yml  init.sh   stop.sh
crypto-config.yaml  generate.sh         start.sh  teardown.sh
bstudent@bstudent-VirtualBox:~/fabricbook$
```

네트워크구성 확장

- crypto-config.yaml
  - cryptogen 툴이 crypto-config.yaml 파일 사용
  - crypto-config.yaml 파일을 이용해서 organization과 그 구성원들에게 인증서 발급
  - peer와 user수 설정

```
- Name: Org2
  Domain: org2.example.com
  Template:
    Count: 1
  Users:
    Count: 1
```

네트워크구성 확장

- configtx.yaml
  - configtxgen 툴이 configtx.yaml파일 사용
  - 네트워크의 channel과 genesis block 생성을 위한 설정
  - anchor peer 설정
  - orderer 설정
  - 네트워크 전체의 구조 및 설정 내용 포함
  - Organization 생성 (3개의 Organization 생성)

```
Profiles:

    ThreeOrgOrdererGenesis:
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
        Consortiums:
            SampleConsortium:
                Organizations:
                    - *Org1
                    - *Org2
                    - *Org3
```

```
    ThreeOrgChannel:
        Consortium: SampleConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
                - *Org2
                - *Org3
```

```
AnchorPeers:
   # AnchorPeers defines the location of peers which can be used
   # for cross org gossip communication.  Note, this value is only
   # encoded in the genesis block in the Application section context
   - Host: peer0.org1.example.com
     Port: 7051
```

네트워크구성 확장

- generate.sh
  - cryptogen, configtxgen 툴을 사용하여 블록체인 네트워크를 위한 요소들 자동 생성 스크립트
    - 이전 crypto material and config transactions 삭제
    - crypto material 생성
    - genesis block for orderer 생성
    - channel configuration transaction 생성
    - anchor peer transaction 생성

네트워크구성 확장

- generate.sh

```
# remove previous crypto material and config transactions
rm -fr config/*
rm -fr crypto-config/*


# generate crypto material
cryptogen generate --config=./crypto-config.yaml
if [ "$?" -ne 0 ]; then
  echo "Failed to generate crypto material..."
  exit 1
fi


# generate genesis block for orderer
configtxgen -profile OneOrgOrdererGenesis -outputBlock ./config/genesis.block
if [ "$?" -ne 0 ]; then
  echo "Failed to generate orderer genesis block..."
  exit 1
fi


# generate channel configuration transaction
configtxgen -profile OneOrgChannel -outputCreateChannelTx ./config/channel.tx -channelID $CHANNEL_NAME
if [ "$?" -ne 0 ]; then
  echo "Failed to generate channel configuration transaction..."
  exit 1
fi
```

ThreeOrgOrdererGenesis

ThreeOrgOrdererChannel

실습

- crypto-config.yaml 수정
  - Org1을 복사하여 Org2, Org3 생성
- configtx.yaml 수정
  - OneOrgOrdererGenesis-> ThreeOrgOrdererGenesis 프로파일명 수정 및 Org2, Org3 맴버 추가
  - OneOrgChannel-> ThreeOrgChannel 프로파일명 수정 및 Org2, Org3 맴버 추가
- generate.sh 수정
  - configtxgen 명령에 포함된 프로파일명 업데이트
- mkdir config
- ./generate.sh 실행

네트워크구성 확장

- docker-compose.yml
  - 여러 컨테이너를 일괄 관리할 수 있는 “docker compose”의 구성 관리 파일
  - docker-compose 파일 수정 요소
    - CA 컨테이너와 peer 컨테이너의 정의 – Org2, Org3 추가
    - CA 관리자(Admin) 패스워드 변경
    - TLS 암호통신을 위한 환경변수 설정(TLS옵션을 원할 시에)
    - CLI 컨테이너 구성
    - 옵션 : CouchDB 컨테이너 기동 제거 (피어 컨테이너의 LevelDB 사용 시)

네트워크구성 확장

- start.sh
  - docker-compose 실행을 통한 컨테이너 및 네트워크 구동

```
docker-compose -f docker-compose.yml down
docker-compose -f docker-compose.yml up -d ca.example.com
        orderer.example.com peer0.org1.example.com couchdb
```

  - 채널 생성

```
docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" -e
"CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp" peer0.org1.example.com peer
channel create -o orderer.example.com:7050 -c mychannel -f /etc/hyperledger/configtx/channel.tx
```

  - peer의 채널 참가

```
docker exec -e "CORE_PEER_LOCALMSPID=Org1MSP" -e
"CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp" peer0.org1.example.com peer
channel join -b mychannel.block
```

네트워크구성 확장

- teardown.sh
  - 실행 시 Docker 컨테이너 삭제

```
# remove chaincode docker images
docker rm $(docker ps -aq)
docker rmi $(docker images dev-* -q)
```

실습

- docker-compose-yml 수정
  - ca.example.com 에 포함된 FABRIC_CA_SERVER_CA_KEYFILE파일이름 수정
  - peer0.0rg1.example.com 포트번호 수정, couchdb 종속성 제거, Ledger DB 를 LevelDB로 전환
  - peer0.org2.example.com 서비스추가
  - peer0.org3.example.com 서비스추가
  - cli working 디렉토리 수정, configtx 볼륨추가

- start.sh 수정
  - docker-compose up 에 포함되는 서비스 추가
  - Peer0.0rg2 과 Peer0.0rg3를 mychannel에 조인
- start.sh 실행
- teardown.sh 실행

```yaml
peer0.org1.example.com:
  container_name: peer0.org1.example.com
  image: hyperledger/fabric-peer
  environment:
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - CORE_PEER_ID=peer0.org1.example.com
    - FABRIC_LOGGING_SPEC=info
    - CORE_CHAINCODE_LOGGING_LEVEL=info
    - CORE_PEER_LOCALMSPID=Org1MSP
    - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/peer/
    - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
    - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=${COMPOSE_PROJECT_NAME}_basic
    - CORE_LEDGER_STATE_STATEDATABASE=LevelDB
    #- CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb:5984
    #- CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    #- CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric
  command: peer node start
  ports:
    - 17051:7051
  volumes:
      - /var/run/:/host/var/run/
      - ./crypto-config/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/
      msp:/etc/hyperledger/msp/peer
      - ./crypto-config/peerOrganizations/org1.example.com/users:/etc/hyperledger/msp/
      users
      - ./config:/etc/hyperledger/configtx
  depends_on:
    - orderer.example.com
  networks:
    - basic
```

```
cli:
  container_name: cli
  image: hyperledger/fabric-tools
  tty: true
  environment:
    - GOPATH=/opt/gopath
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - FABRIC_LOGGING_SPEC=info
    - CORE_PEER_ID=cli
    - CORE_PEER_ADDRESS=peer0.org1.example.com:7051
    - CORE_PEER_LOCALMSPID=Org1MSP
    - CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
    peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
    - CORE_CHAINCODE_KEEPALIVE=10
  working_dir: /etc/hyperledger/configtx
  command: /bin/bash
  volumes:
      - /var/run/:/host/var/run/
      - ./chaincode/:/opt/gopath/src/github.com/
      - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
      - ./config:/etc/hyperledger/configtx
  networks:
      - basic
```

```
13 docker-compose -f docker-compose.yml down
14
15 docker-compose -f docker-compose.yml up -d ca.example.com orderer.example.com
   peer0.org1.example.com peer0.org2.example.com peer0.org3.example.com cli
16 docker ps -a
17
18 # wait for Hyperledger Fabric to start
19 # incase of errors when running later commands, issue export FABRIC_START_TIMEOUT=<larger
   number>
20 export FABRIC_START_TIMEOUT=10
21 #echo ${FABRIC_START_TIMEOUT}
22 sleep ${FABRIC_START_TIMEOUT}
23
24 # Create the channel
25 docker exec cli peer channel create -o orderer.example.com:7050 -c mychannel -f /etc/
   hyperledger/configtx/channel.tx
26
27 sleep 5
28
29 # Join peer0.org1.example.com to the channel.
30 docker exec -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/
   msp" peer0.org1.example.com peer channel join -b /etc/hyperledger/configtx/mychannel.block
31
32 sleep 5
33
34 docker exec -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org2.example.com/
   msp" peer0.org2.example.com peer channel join -b /etc/hyperledger/configtx/mychannel.block
35
36 sleep 5
37
38 docker exec -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org3.example.com/
   msp" peer0.org3.example.com peer channel join -b /etc/hyperledger/configtx/mychannel.block
39
40 sleep 5
```

```
bstudent@bstudent-VirtualBox: ~/fabricbook
Creating peer0.org1.example.com
Creating peer0.org2.example.com
docker ps -a
CONTAINER ID        IMAGE                      COMMAND                  CREATED             STATUS                  PORTS
                    NAMES
15042e458582        hyperledger/fabric-peer    "peer node start"        5 seconds ago       Up Less than a second   0.0.0.
0:27051->7051/tcp   peer0.org2.example.com
5a099fae8cc6        hyperledger/fabric-peer    "peer node start"        5 seconds ago       Up 1 second             0.0.0.
0:17051->7051/tcp   peer0.org1.example.com
83770a20ab89        hyperledger/fabric-peer    "peer node start"        5 seconds ago       Up 2 seconds            0.0.0.
0:37051->7051/tcp   peer0.org3.example.com
0efdb50cfa1e        hyperledger/fabric-ca      "sh -c 'fabric-ca-se…"   6 seconds ago       Up 4 seconds            0.0.0.
0:7054->7054/tcp    ca.example.com
958e8555632f        hyperledger/fabric-tools   "/bin/bash"              6 seconds ago       Up 3 seconds
                    cli
0dcbd22f642f        hyperledger/fabric-orderer "orderer"                6 seconds ago       Up 5 seconds            0.0.0.
0:7050->7050/tcp    orderer.example.com

# wait for Hyperledger Fabric to start
# incase of errors when running later commands, issue export FABRIC_START_TIMEOUT=<larger number>
export FABRIC_START_TIMEOUT=10
#echo ${FABRIC_START_TIMEOUT}
sleep ${FABRIC_START_TIMEOUT}

# Create the channel
docker exec cli peer channel create -o orderer.example.com:7050 -c mychannel -f /etc/hyperledger/configtx/channel.tx
2019-07-08 08:29:51.394 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2019-07-08 08:29:51.431 UTC [cli.common] readBlock -> INFO 002 Received block: 0

sleep 5

# Join peer0.org1.example.com to the channel.
docker exec -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org1.example.com/msp" peer0.org1.example.com peer c
hannel join -b /etc/hyperledger/configtx/mychannel.block
2019-07-08 08:29:56.802 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2019-07-08 08:29:56.889 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel

sleep 5

docker exec -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org2.example.com/msp" peer0.org2.example.com peer c
hannel join -b /etc/hyperledger/configtx/mychannel.block
2019-07-08 08:30:02.266 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2019-07-08 08:30:02.349 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel

sleep 5

docker exec -e "CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/users/Admin@org3.example.com/msp" peer0.org3.example.com peer c
hannel join -b /etc/hyperledger/configtx/mychannel.block
2019-07-08 08:30:07.696 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
2019-07-08 08:30:07.797 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel

sleep 5
bstudent@bstudent-VirtualBox:~/fabricbook$
```

② 체인코드 관리

체인코드 라이프사이클

체인코드 구현

체인코드 설치 (체인코드이름, 버전)

체인코드 객체화 (체인코드이름, 버전, 채널명)

QUERY, INVOKE (체인코드이름, 버전, 채널명)

체인코드 버전업 구현

체인코드 설치 (체인코드이름, **버전**)

체인코드 업그레이드 (체인코드이름, **버전**, 채널명)

QUERY, INVOKE (체인코드이름, **버전**, 채널명)

체인코드 개발

- 체인코드 개발 언어
  - JAVA
  - Node.js
  - GO language

Every chaincode program must implement the `Chaincode` interface:

- Go
- node.js
- Java

whose methods are called in response to received transactions. In particular the `Init` method is called when a chaincode receives an `instantiate` or `upgrade` transaction so that the chaincode may perform any necessary initialization, including initialization of application state. The `Invoke` method is called in response to receiving an `invoke` transaction to process transaction proposals.

The other interface in the chaincode "shim" APIs is the `ChaincodeStubInterface`:

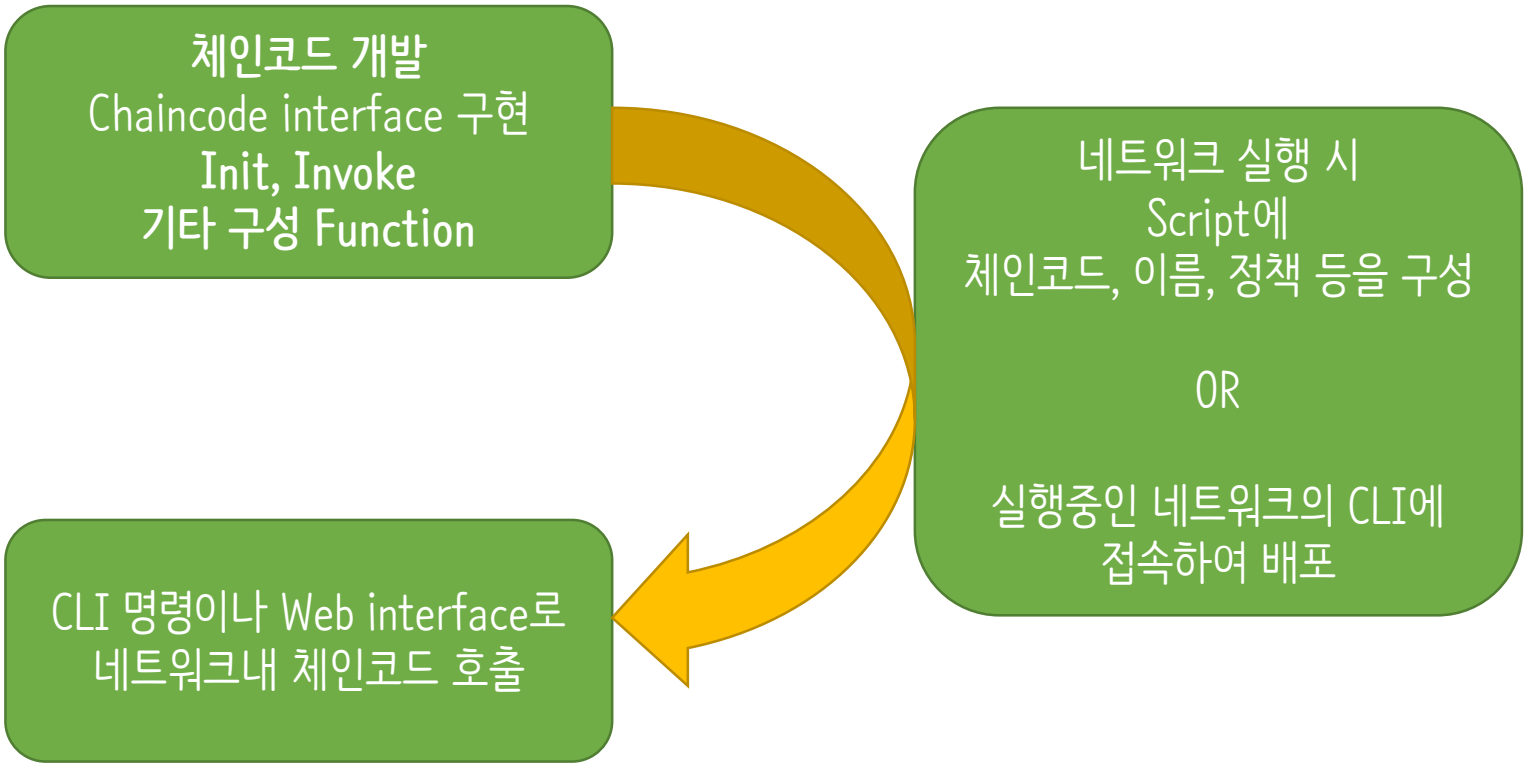
- Go
- node.js
- Java

which is used to access and modify the ledger, and to make invocations between chaincodes.

In this tutorial using Go chaincode, we will demonstrate the use of these APIs by implementing a simple chaincode application that manages simple "assets".

체인코드 개발 및 배포, 실행구조

- Go언어, java, node.js로 개발가능



체인코드 인터페이스 기능 (자산, 함수 등)

- Every chaincode program must **implement** the Chaincode interface:
  - **Init method** called WHEN chaincode receives an instantiate or upgrade transaction
  - perform any necessary initialization
  - **Invoke method** an invoke transaction
- The other interface "shim" APIs is the ChaincodeStubInterface interface:
  - used to access and modify the ledger,
  - to make invocations between chaincodes.

SHIM interface

- a lower level API for "Smart Contracts".
- to support communication with Hyperledger Fabric peers for Smart Contracts
- written using the fabric-contract-api
- together with the fabric-chaincode-node cli to launch Chaincode or Smart Contracts.

```
package main

import (
    "fmt"

    "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 {
}
```

실습

- fabricbook디렉토리에 chaincode 디렉토리 생성
  - cd ~/fabricbook
  - mkdir chaincode
- sacc.go 파일 복사
  - cp -r ~/fabric-samples/chaincode/sacc ./sacc
- cli 명령 환경으로 접속
  - docker exec -it cli bash
  - peer chaincode install -n sacc -v 1.0 -p github.com/sacc
  - peer chaincode instantiate -n sacc -v 1.0 -C mychannel -P "OR ('Org1MSP.member')" -c '{"Args":["a","100"]}'
  - peer chaincode query -n sacc -C mychannel -c '{"Args":["get","a"]}'

수행결과

```
bstudent@bstudent-VirtualBox:~/fabricbook$ docker exec  -it cli bash
Error: No such container:  -it
bstudent@bstudent-VirtualBox:~/fabricbook$ docker exec -it cli bash
root@d5d6346b9984:/etc/hyperledger/configtx# peer chaincode install -n sacc -v 1.0 -p github.com/sacc
2019-07-08 09:12:47.697 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2019-07-08 09:12:47.698 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
2019-07-08 09:12:47.858 UTC [chaincodeCmd] install -> INFO 003 Installed remotely response:<status:200 payload:"OK" >
root@d5d6346b9984:/etc/hyperledger/configtx# peer chaincode instantiate -n sacc -v 1.0 -C mychannel -P "OR ('Org1MSP.member'
)" -c '{"Args":["a","100"]}'
2019-07-08 09:13:01.840 UTC [chaincodeCmd] InitCmdFactory -> INFO 001 Retrieved channel (mychannel) orderer endpoint: ordere
r.example.com:7050
2019-07-08 09:13:01.841 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default escc
2019-07-08 09:13:01.841 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 003 Using default vscc
root@d5d6346b9984:/etc/hyperledger/configtx# peer chaincode query -n sacc -C mychannel -c '{"Args":["get","a"]}'
100
root@d5d6346b9984:/etc/hyperledger/configtx#
```