

HYPERLEDGER FABRIC v1.0

개발자 가이드

HYPERLEDGER FABRIC 을 이용한 애플리케이션 개발
MOO JE KONG

1.	Fabric SDK 로 트랜잭션 테스트 하기	3
1.1.	테스트를 위한 블록체인 네트워크 실행하기	4
1.2.	트랜잭션 테스트	5
1.3.	이벤트 리스너 사용하기	Error! Bookmark not defined.
2.	스마트 컨트랙트 (체인코드) 개발	3
3.	Hyperledger Composer 를 이용한 비즈니스 네트워크 개발	Error! Bookmark not defined.
3.1.	Hyperledger Composer 설치	Error! Bookmark not defined.
3.2.	Hyperledger Fabric 시작	Error! Bookmark not defined.
3.3.	Hyperledger Composer 를 이용한 비즈니스 네트워크 생성	Error! Bookmark not defined.
3.4.	Hyperledger Fabric 런타임에 비즈니스 네트워크 아카이브 디플로이	Error! Bookmark not defined.
3.5.	REST API 생성 및 테스트	Error! Bookmark not defined.
3.6.	스켈레톤 웹 어플리케이션 생성	Error! Bookmark not defined.

1. 스마트 컨트랙트 (체인코드) 개발

이번 장에서는 개발모드에서 체인코드를 개발하고 테스트 하는 방법에 대해서 설명합니다. fabric-samples/chaincode-docker-devmode 에서 테스트합니다.

샘플 파일이 있는 위치로 이동하고 현재 실행중인 컨테이너가 있다면 모두 중지합니다.

```
cd $HOME/fabric-samples/chaincode-docker-devmode
docker stop $(docker ps -qa) && docker rm $(docker ps -qa)
```

블록체인 네트워크를 실행하고 채널 생성을 합니다.

```
docker-compose -f docker-compose-simple.yaml up -d
```

다음으로는 체인코드를 빌드해서 실행합니다. 빌드를 하기 위해서는 체인코드 빌드를 위한 환경을 제공하고 있는 hyperledger/fabric-ccenv 컨테이너에서 빌드를 하면되고 현재 실행되고 있는 컨테이너명은 chaincode 입니다. 다음의 명령을 통해서 chaincode 컨테이너에 접속하고 sacc 체인코드를 빌드합니다.

```
docker exec -it chaincode bash
cd sacc
go build
```

정상적으로 빌드를 완료하였으면 다음의 명령을 통해서 체인코드를 실행합니다.

CORE_PEER_ADDRESS 는 체인코드가 붙어서 통신해야하는 Peer 의 접속 정보를 입력해주면 되고, CORE_CHAINCODE_ID_NAME 은 체인코드의 이름과 버전입니다.

현재는 체인코드의 프로세스만 실행을 시킨 상태이고 채널에 등록이 된 상태는 아닙니다.

```
CORE_PEER_ADDRESS=peer:7052 CORE_CHAINCODE_ID_NAME=mycc:0 ./sacc
```

그리고는 채널에 체인코드를 등록하고 초기화 시키기 위해서 **새로운 터미널을 실행하여** cli 컨테이너에 접속합니다.

```
docker exec -it cli bash
```

다음의 명령을 통해서 체인코드를 채널에 등록하고 초기화합니다.

```
peer chaincode install -p chaincodedev/chaincode/sacc -n mycc -v 0
peer chaincode instantiate -n mycc -v 0 -c '{"Args":["a","10"]}' -C myc
```

다음 코드를 통해서 트랜잭션을 실행해서 정상 동작을 확인합니다.

```
peer chaincode invoke -n mycc -c '{"Args":["set", "a", "20"]}' -C myc
peer chaincode query -n mycc -c '{"Args":["query","a"]}' -C myc
```

수정된 체인코드의 빌드 → 채널의 등록(install, instantiate) → 체인코드 호출 (invoke, query) 등을 반복해서 작업합니다.

2. Fabric SDK(Node SDK) 로 트랜잭션 테스트 하기

Node.js 기반의 SDK 동작 테스트를 해봅니다. 앞서 블록체인 네트워크 테스트를 했던 fabric-samples 에서 fabcar 샘플을 통해 Fabric SDK 를 이용한 어플리케이션에서의 트랜잭션 테스트를 해보겠습니다. fabcar 샘플은 자동차에 대한 기본정보(모델명, 제조사, 색깔)와 소유자 정보를 블록체인으로 관리하는 샘플이며, 자동차 리스트 조회, 자동차 정보 생성, 소유자 변경등의 동작으로 테스트 해 볼 수 있습니다.

2.1. 테스트를 위한 블록체인 네트워크 실행하기

다음의 명령을 통해서 fabcar 디렉토리로 이동합니다.

```
cd $HOME/fabric-samples/fabcar
```

테스트를 위한 블록체인 네트워크를 실행하기 전에 이미 실행되고 있는 Hyperledger Fabric 컨테이너가 있다면 모두 중지 및 삭제 합니다.

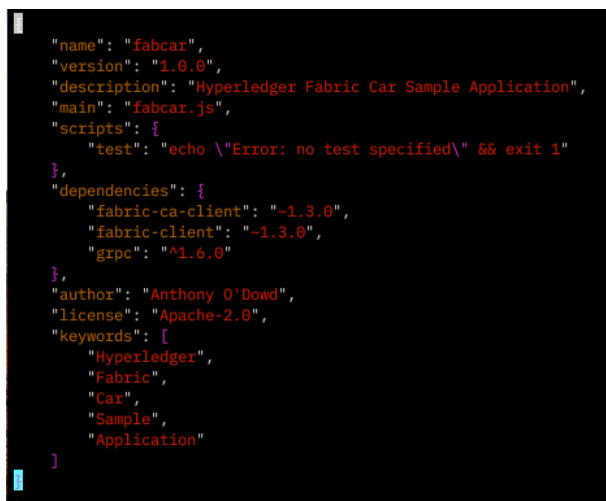
```
docker stop $(docker ps -aq) && docker rm $(docker ps -qa)
```

그리고는 블록체인 네트워크를 실행합니다.

```
./startFabric.sh
```

그런 다음 다음 명령을 통해서 Fabric SDK 를 설치합니다. 이 명령어는 아래 그림과 같이 두개의 Fabric SDK(fabric-ca-client, fabric-client)를 설치합니다.

```
npm install
```



```
{
  "name": "fabcar",
  "version": "1.0.0",
  "description": "Hyperledger Fabric Car Sample Application",
  "main": "fabcar.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "dependencies": {
    "fabric-ca-client": "~1.3.0",
    "fabric-client": "~1.3.0",
    "grpc": "^1.6.0"
  },
  "author": "Anthony O'Dowd",
  "license": "Apache-2.0",
  "keywords": [
    "Hyperledger",
    "Fabric",
    "Car",
    "Sample",
    "Application"
  ]
}
```

만약 "npm install" 실행 시 make, g++ 관련 에러가 발생할 경우 다음의 명령을 통해서 관련 패키지를 설치합니다.

```
sudo apt install -y make gcc g++
```

2.2. 테스트를 위한 사용자 등록

블록체인 네트워크 관리자 역할을 담당할 admin 사용자 등록을 다음의 명령을 통해 진행합니다.

```
node enrollAdmin.js
```

```
bcadmin@hlf01:~/fabric-samples/fabcar$ node enrollAdmin.js
Store path:/home/bcadmin/fabric-samples/fabcar/hfc-key-store
(node:16631) DeprecationWarning: grpc.load: Use the @grpc/proto-loader module with grpc.loadPackageDefinition instead
Successfully enrolled admin user "admin"
Assigned the admin user to the fabric client :{"name":"admin","mspid":"Org1MSP","roles":null,"affiliation":"","enrollmentSecret":"","enrollment":{"signingIdentity":"4adfdc208ed5ea5f08d836e727cf3f1cbcb680c9836d6b4a1c4b2e6efa5acf791"},"identity":{"certificate":"-----BEGIN CERTIFICATE-----\nMIICAjCCAaigAwIBAgIUAnHxGvXm0yaQfVgVM0cWJ4ibJEe8wCgYIKoZIzj0EAwIw\ncnzELMAkGA1UEBhMCVVMMxEzARBgNVBAgTCKNhbGlmb3JuawExFjAUBGZNVBAcTDVNh\nbnbiBGcmFuY2l2Y28xGTAXBGNVBAAoTEG9yZzEuZXhhbXBsZS5jb20xHDAaBgNVBAMT\nN2NhNm9yZzEuZXhhbXBsZS5jb20wHhcNMjgxMDU0MDc0ODAwWHcnMTkxMDU0MDc1\nnmZAwwWjAhMQ8wDQYDVQQLEwZjbGllbnRxdjAMBGNVBAMTBWFkbWluMFkwEwYHKoZI\nnzj0CAQYIKoZIzj0DAQcDQGAErWm3HrYKjxe55g7Dk2KN7s2qxQy/8vCXHYhv6r\nXhT\n/n0/iSQak60vY9jBq2KPURrrXe2t2UDDhb/ngB6TCkBL33q6NsMGowDgYDVR0PAQH/\n\nnBAQDAgeAMAwwGA1UdEWEB/wQCMAAwHQYDVR0OBBYEFivHFyXmzv5WV16c4e5W1n6U\n\nncQnwMcSGA1UdIwQkMCKAIEI5qg3NdtruuloM2nAYUdFFBNMarRst3dusa\nlc2Xkl8\n\nnMAoGCCqGSM49BAMCA0gAMEUCIQCbjiGEQE06Vy5xek5zNd+t9nEDqYmJgPUKQE1R\n\nnEr14dwIgLGabC4EH3MZxB5mZF7+M1FTu694IRey/NFPafh784j0=\n\n-----END CERTIFICATE-----\n"}}}
```

admin 사용자의 등록이 완료되었으면 트랜잭션을 발생시키기 위한 사용자(user1)을 추가하고 등록합니다.

```
node registerUser.js
```

```

bcadmin@hl1f01:~/fabric-samples/fabcar$ node registerUser.js
Store path:/home/bcadmin/fabric-samples/fabcar/hfc-key-store
(node:16662) DeprecationWarning: grpc.load: Use the @grpc/proto-loader module with grpc.loadPackageDefinition instead
Successfully loaded admin from persistence
Successfully registered user1 - secret:qzCZdDUWkBiB
Successfully enrolled member user "user1"
User1 was successfully registered and enrolled and is ready to interact with the fabric network

```

2.3. 트랜잭션 테스트

이번에는 query.js, invoke.js 를 통해서 조회인 데이터 입력 및 변경에 대한 테스트를 해보겠습니다.

먼저 다음 명령을 통해서 블록체인에서 현재 등록된 자동차 리스트를 검색해 보겠습니다.

```
node query.js
```

```

bcadmin@hlf01:~/fabric-samples/fabcar$ node query.js
Store path:/home/bcadmin/fabric-samples/fabcar/hfc-key-store
(node:16687) DeprecationWarning: grpc.load: Use the @grpc/proto-loader module with grpc.loadPackageDefinition instead
Successfully loaded user1 from persistence
Query has completed, checking results
Response is [{"Key":"CAR0", "Record":{"colour":"blue","make":"Toyota","model":"Prius","owner":"Tomoko"}}, {"Key":"CAR1", "Record":{"colour":"red","make":"Ford","model":"Mustang","owner":"Brad"}}, {"Key":"CAR2", "Record":{"colour":"green","make":"Hyundai","model":"Tucson","owner":"Jin Soo"}}, {"Key":"CAR3", "Record":{"colour":"yellow","make":"Volkswagen","model":"Passat","owner":"Max"}}, {"Key":"CAR4", "Record":{"colour":"black","make":"Tesla","model":"S","owner":"Adriana"}}, {"Key":"CAR5", "Record":{"colour":"purple","make":"Peugeot","model":"205","owner":"Michel"}}, {"Key":"CAR6", "Record":{"colour":"white","make":"Chery","model":"S22L","owner":"Aarav"}}, {"Key":"CAR7", "Record":{"colour":"violet","make":"Fiat","model":"Punto","owner":"Pari"}}, {"Key":"CAR8", "Record":{"colour":"indigo","make":"Tata","model":"Nano","owner":"Valeria"}}, {"Key":"CAR9", "Record":{"colour":"brown","make":"Holden","model":"Barina","owner":"Shotaro"}}]

```

다음으로는 새로운 자동차 정보를 등록해보겠습니다. invoke.js 파일을 열어서 다음과 같이 64,65 라인을 수정하고 다음의 명령을 실행합니다.

```

61     var request = {
62         //targets: let default to the peer assigned to the client
63         chaincodeId: 'fabcar',
64         fcn: 'createCar',
65         args: ['CAR12', 'Honda', 'Accord', 'Black', 'Tom'],
66         chainId: 'mychannel',
67         txId: tx_id
68     };
69
70     // send the transaction proposal to the peers

```

```
node invoke.js
```

```

bcadmin@hlf01:~/fabric-samples/fabcar$ node invoke.js
Store path:/home/bcadmin/fabric-samples/fabcar/hfc-key-store
(node:16759) DeprecationWarning: grpc.load: Use the @grpc/proto-loader module with grpc.loadPackageDefinition instead
Successfully loaded user1 from persistence
Assigning transaction_id: 66861c430f530be826cafb1cc81f0337ae3fde7950ce7836aee45b799cb73008
Transaction proposal was good
Successfully sent Proposal and received ProposalResponse: Status - 200, message - ""
The transaction has been committed on peer localhost:7051
Send transaction promise and event listener promise have completed
Successfully sent transaction to the orderer.
Successfully committed the change to the ledger by the peer

```

정상적으로 처리되었으면 query.js 파일을 열어서 다음과 같이 57,58 라인을 수정하고 명령을 실행해서 정상적으로 CAR12 이 검색되는지 확인합니다.

```

51
52     // queryCar chaincode function - requir
53     // queryAllCars chaincode function - re
54     const request = {
55         //targets : --- letting this de
56         chaincodeId: 'fabcar',
57         fcn: 'queryCar',
58         args: ['CAR12']
59     };
60

```

node query.js

```

Successfully committed the change to the ledger by the peer
bcadmin@hlf01:~/fabric-samples/fabcar$ node query.js
Store path:/home/bcadmin/fabric-samples/fabcar/hfc-key-store
(node:19640) DeprecationWarning: grpc.load: Use the @grpc/proto-loader module with grpc.loadPackageDefinition instead
Successfully loaded user1 from persistence
Query has completed, checking results
Response is  {"colour":"Black","make":"Honda","model":"Accord","owner":"Tom"}

```

다음으로는 조금 전 등록했던 CAR12에 대해서 소유권을 변경해보겠습니다.. invoke.js 파일의 64,65 라인을 다음과 같이 수정합니다. 기존에 "Tom"의 소유였던 CAR12을 "Barry"로 변경하는 내용입니다. 수정 후 다음의 명령을 실행하여 블록체인에 트랜잭션을 실행합니다.

```

61     var request = {
62         //targets: let default to the peer
63         chaincodeId: 'fabcar',
64         fcn: 'changeCarOwner',
65         args: ['CAR12', 'Barry'],
66         chainId: 'mychannel',
67         txId: tx_id
68     };
69

```

node invoke.js

```

bcadmin@hlf01:~/fabric-samples/fabcar$ node invoke.js
Store path:/home/bcadmin/fabric-samples/fabcar/hfc-key-store
(node:19673) DeprecationWarning: grpc.load: Use the @grpc/proto-loader module with grpc.loadPackageDefinition instead
Successfully loaded user1 from persistence
Assigning transaction_id: 610d3e688f0944e01195b1c04ce1d8b88362dde5676ffeecd131c60f5e468b4a
Transaction proposal was good
Successfully sent Proposal and received ProposalResponse: Status - 200, message - ""
The transaction has been committed on peer localhost:7051
Send transaction promise and event listener promise have completed
Successfully sent transaction to the orderer.
Successfully committed the change to the ledger by the peer

```

다음과 같이 정상적으로 처리되었으면 다시 쿼리를 해서 정상적으로 소유권 변경이 되었는지 확인해봅니다.

```
node query.js
```

```
bcadmin@hlf01:~/fabric-samples/fabcar$ node query.js
Store path:/home/bcadmin/fabric-samples/fabcar/hfc-key-store
(node:19688) DeprecationWarning: grpc.load: Use the @grpc/proto-loader module with grpc.loadPackageDefinition instead
Successfully loaded user1 from persistence
Query has completed, checking results
Response is  {"colour":"Black","make":"Honda","model":"Accord","owner":"Barry"}
```