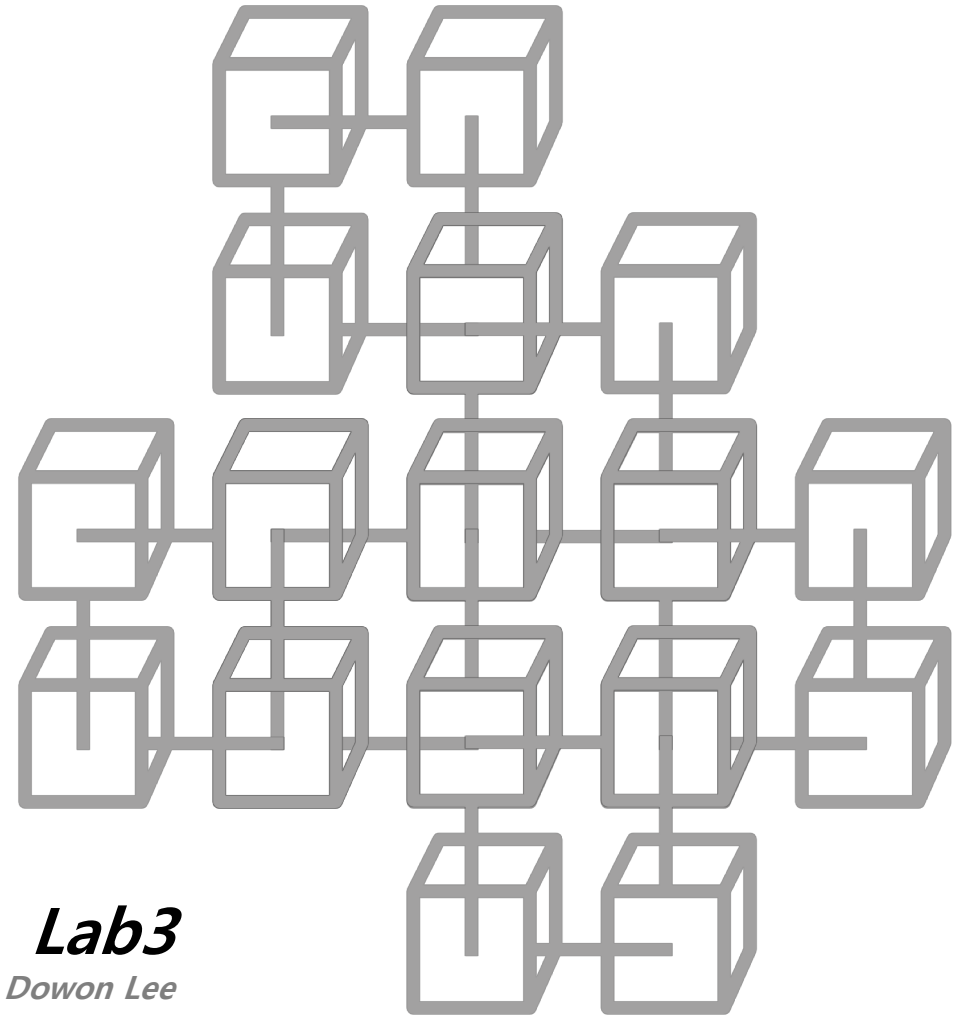


HYPERLEDGER FABRIC

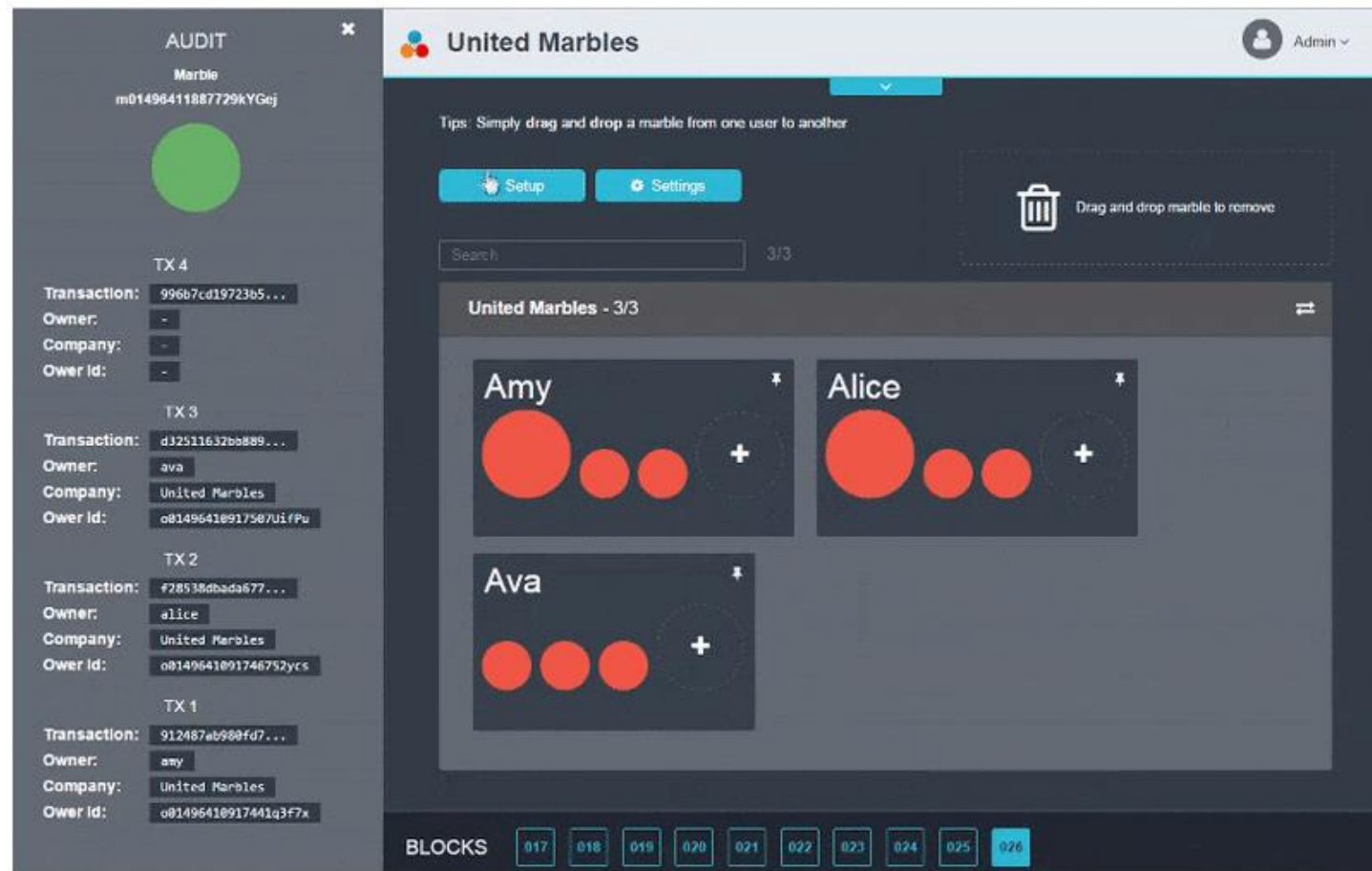


Lab3

written by Dowon Lee

Marbles Demo

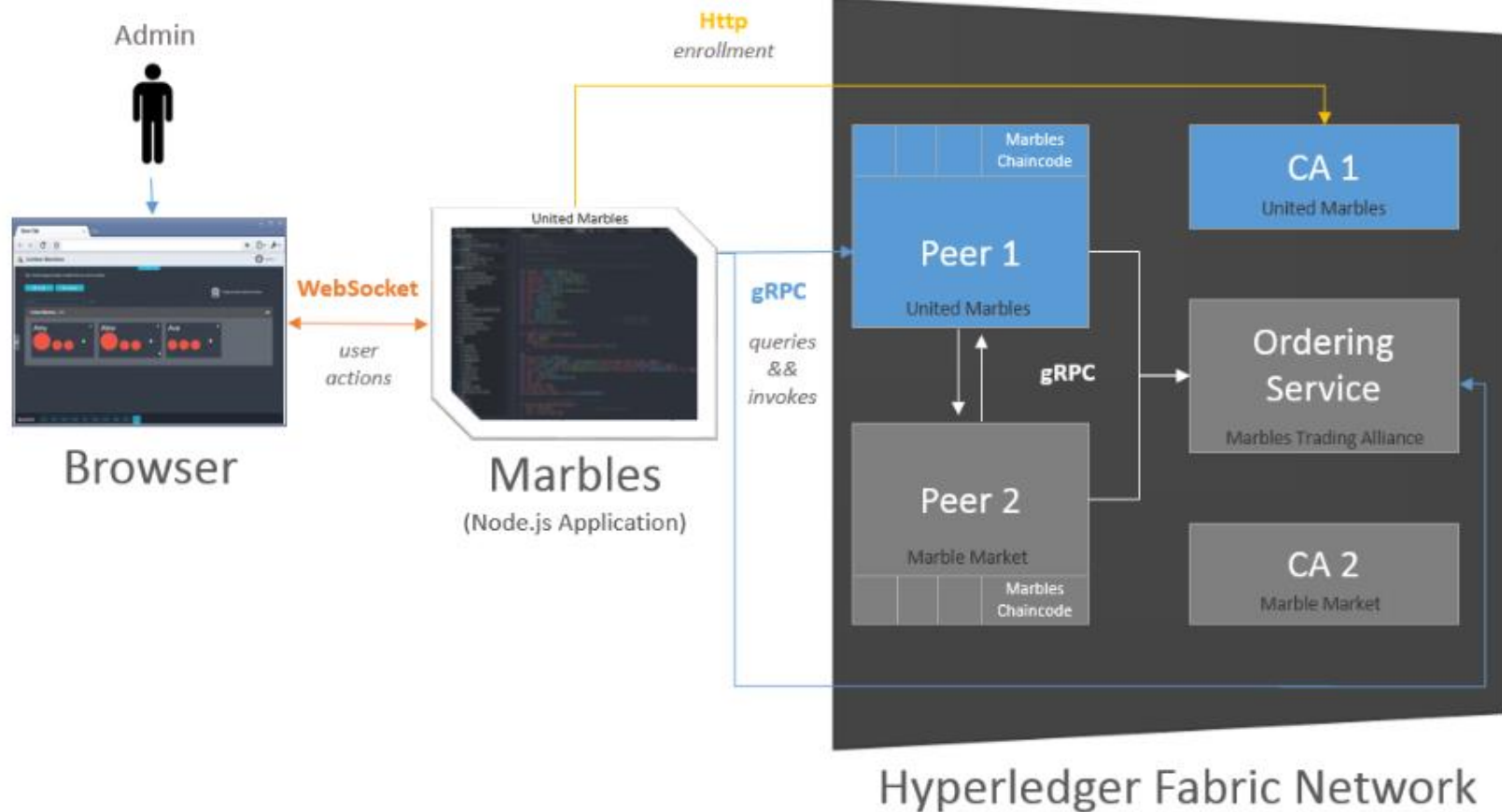
- 본 애플리케이션은 Linux Foundation 의 Hyperledger Fabric 네트워크를 이용한 예제로써, Hyperledger Fabric 에 대한 구조 및 작동 원리에 대해 이해할 수 있다.
- 이 데모를 통해 개발자가 패브릭 네트워크로 Chaincode 및 앱 개발의 기본 사항을 익힐 수 있다.
- 여러 사용자가 Marble 을 만들고 다른 사용자에게 전송할 수 있다.



Marbles Demo

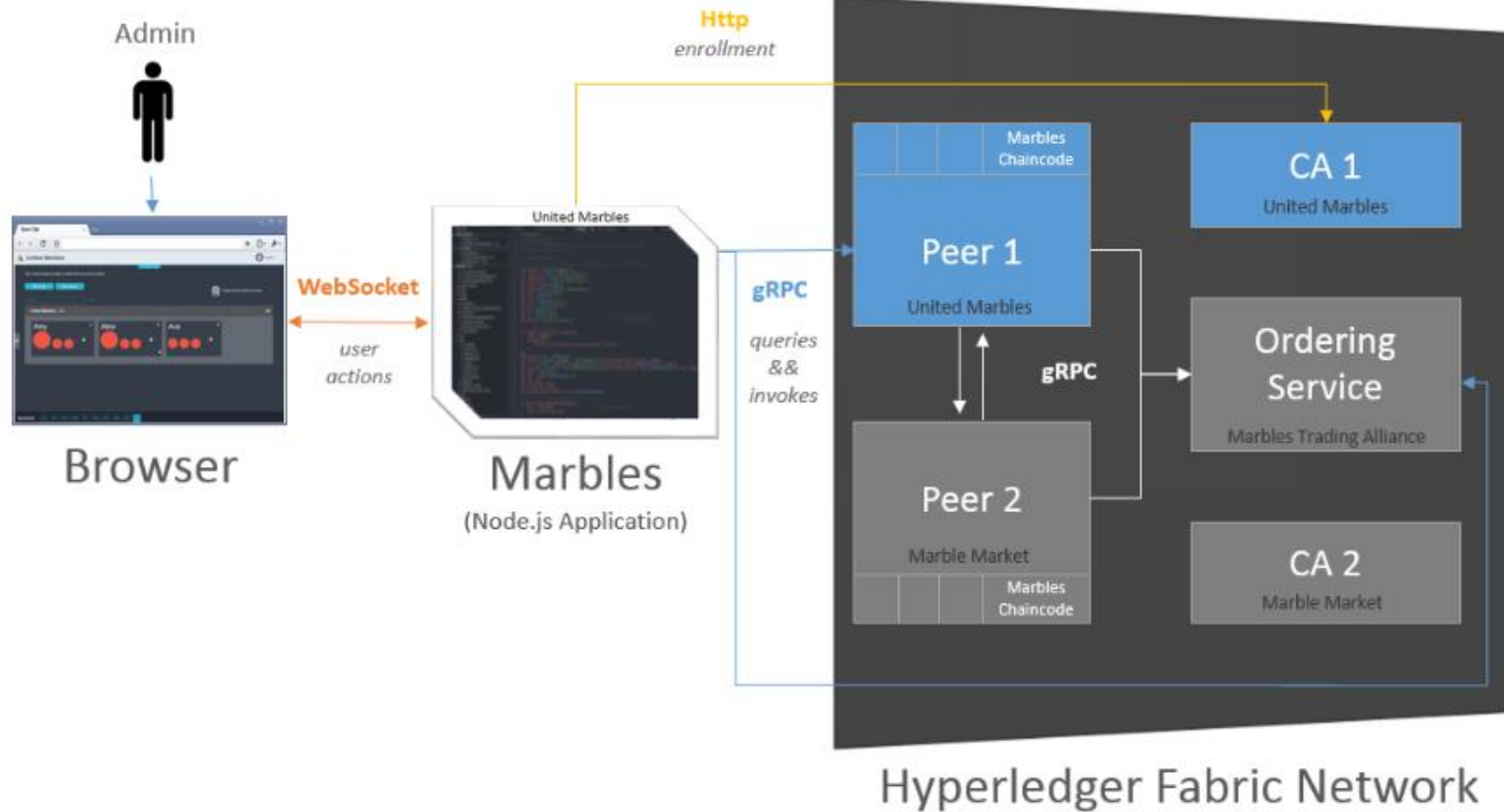
- Marble의 속성
 - 1) id (고유 문자열, 키로 사용됨)
 - 2) 색상 (문자열, CSS 색상 이름)
 - 3) 크기 (정수, 크기 (mm))
 - 4) 소유자 (문자열)
- Marble의 특징
 - 웹 기반 UI
 - {Marble ID, Marble의 속성} → JSON → Blockchain에 저장
 - CC의 상호 작용은 네트워크의 피어(Peer)에게 gRPC 프로토콜로 전달
 - gRPC의 세부 사항은 Hyperledger Fabric Client SDK에 의해 처리

Marbles Demo



- 1) 관리자는 브라우저에서 Node.js 애플리케이션 인 Marbles와 상호 작용한다.
- 2) 이 클라이언트 측 JS 코드는 백엔드 Node.js 애플리케이션에 대한 websocket을 연다. 클라이언트 JS는 관리자가 사이트와 상호 작용할 때 백엔드에 메시지를 보낸다.
- 3) 장부를 읽거나 쓰는 것은 제안 (Proposal)이라고 한다. 이 제안은 Marbles에서 SDK를 통해 작성한 다음 블록 체인 동료에게 전송된다.

Marbles Demo



- 4) 피어는 Marble Chaincode 컨테이너와 통신한다. Chaincode는 트랜잭션을 실행 / 시뮬레이션한다. 문제가 없으면 트랜잭션을 승인하고 Marbles 애플리케이션으로 다시 보낸다.
- 5) Marbles (SDK를 통해)는 승인된 제안서를 Ordering service에 보낸다. Orderer는 전체 네트워크에서 전송된 모든 제안(Proposal)을 블록으로 묶는다. 그런 다음 새로운 블록을 네트워크의 동료(Peer)에게 브로드 캐스트한다.
- 6) 마지막으로 피어는 블록의 유효성을 검사하여 장부에 기록한다. 거래가 효력을 발휘하고 이후의 모든 읽기는 이 변경 사항을 반영한다.

Marbles Demo

- Marble의 구성

- 1) Chaincode Part

- 블록 체인 네트워크상의 피어 상에서 함께 실행되는 GoLang 코드이다.
- 모든 Marble / 블록 체인 상호 작용이 궁극적으로 여기서 발생한다.
- /chaincode 밑에 저장

- 2) 클라이언트 측 JS 부분

- 이것은 사용자의 브라우저에서 실행되는 JavaScript 코드이다.
- 사용자 인터페이스 상호 작용이 발생한다.
- /public/js 밑에 저장

- 3) 서버 측 JS 부분

- 애플리케이션의 백엔드를 실행하는 JavaScript 코드이다.
- Marbles의 핵심 인 Node.js 코드로써, Node 또는 서버 코드라고도 한다.
- Marble 관리자와 블록 체인 사이의 연결하는 게이트웨이(Gateway) 역할을 한다.
- /utils 및 /routes 밑에 저장

Marbles 설치

- Marble 다운로드

```
$ git clone https://github.com/IBM-Blockchain/marbles.git --depth 1
$ cd marbles
$ npm install
```

- Marble 네트워크 구성
 - fabcar 애플리케이션을 이용해서 네트워크 시작

```
$ cd $HOME/fabric-samples/fabcar
$ ./startFabric.sh
```

- admin, user1 사용자 생성

```
$ npm install
$ node enrollAdmin.js
$ node registerUser.js
```

```
Total setup execution time : 32 secs ...
```

```
Start by installing required packages run 'npm install'
Then run 'node enrollAdmin.js', then 'node registerUser'
```

```
The 'node invoke.js' will fail until it has been updated with valid arguments
The 'node query.js' may be run at anytime once the user has been registered
```

```
bcadmin@hlf03:~/fabric-samples/fabcar$
```

Marbles 설치

- Chaincode 설치 및 인스턴스화

```
$ cd $HOME/marbles/scripts
$ node install_chaincode.js
$ node instantiate_chaincode.js
```

```
-----
debug: [fcw] Installing Chaincode
debug: [fcw] Sending install req targets=[grpc.http2.keepalive_time=300, grpc.keepalive_time_ms=300000, grpc.http2.keepalive_timeo
ut=35, grpc.keepalive_timeout_ms=3500, grpc.max_receive_message_length=-1, grpc.max_send_message_length=-1, grpc.primary_user_agen
t=grpc-node/1.10.1, _url=grpc://localhost:7051, addr=localhost:7051, , _request_timeout=90000, , _name=null], chaincodePath=marble
s, chaincodeId=marbles, chaincodeVersion=v4
info: [packager/Golang.js]: packaging GOLANG from marbles
debug: [fcw] Successfully obtained transaction endorsement
-----
```

```
info: Install done. Errors: nope
-----
```

```
info: Now we instantiate
-----
debug: [fcw] Instantiating Chaincode peer_urls=[grpc://localhost:7051], channel_id=mychannel, chaincode_id=marbles, chaincode_vers
ion=v4, cc_args=[12345], ssl-target-name-override=null, pem=null, grpc.http2.keepalive_time=300, grpc.keepalive_time_ms=300000, gr
pc.http2.keepalive_timeout=35, grpc.keepalive_timeout_ms=3500
debug: [fcw] Sending instantiate req targets=[grpc.http2.keepalive_time=300, grpc.keepalive_time_ms=300000, grpc.http2.keepalive_t
imeout=35, grpc.keepalive_timeout_ms=3500, grpc.max_receive_message_length=-1, grpc.max_send_message_length=-1, grpc.primary_user_
agent=grpc-node/1.10.1, _url=grpc://localhost:7051, addr=localhost:7051, , _request_timeout=90000, , _name=null], chaincodeId=marb
les, chaincodeVersion=v4, fcn=init, args=[12345], 0=101, 1=48, 2=228, 3=252, 4=100, 5=66, 6=0, 7=104, 8=173, 9=236, 10=234, 11=171
, 12=54, 13=95, 14=223, 15=242, 16=81, 17=136, 18=232, 19=161, 20=35, 21=50, 22=217, 23=177, _transaction_id=16e8aea35fd0c59aa205d
bfe06d4136ff34bd97414a9f46b61355f228d308272
debug: [fcw] Successfully obtained transaction endorsement
debug: [fcw] Successfully ordered instantiate endorsement.
-----
info: Instantiate done. Errors: nope
-----
```


Marbles 설치

- Chaincode 설치 및 인스턴스화

```
$ cd $HOME/marbles/scripts
$ node install_chaincode.js
$ node instantiate_chaincode.js
```

```
-----
debug: [fcw] Installing Chaincode
debug: [fcw] Sending install req targets=[grpc.http2.keepalive_time=300, grpc.keepalive_time_ms=300000, grpc.http2.keepalive_timeo
ut=35, grpc.keepalive_timeout_ms=3500, grpc.max_receive_message_length=-1, grpc.max_send_message_length=-1, grpc.primary_user_agen
t=grpc-node/1.10.1, _url=grpc://localhost:7051, addr=localhost:7051, , _request_timeout=90000, , _name=null], chaincodePath=marble
s, chaincodeId=marbles, chaincodeVersion=v4
info: [packager/Golang.js]: packaging GOLANG from marbles
debug: [fcw] Successfully obtained transaction endorsement
-----
```

```
info: Install done. Errors: nope
-----
```

```
info: Now we instantiate
-----
debug: [fcw] Instantiating Chaincode peer_urls=[grpc://localhost:7051], channel_id=mychannel, chaincode_id=marbles, chaincode_vers
ion=v4, cc_args=[12345], ssl-target-name-override=null, pem=null, grpc.http2.keepalive_time=300, grpc.keepalive_time_ms=300000, gr
pc.http2.keepalive_timeout=35, grpc.keepalive_timeout_ms=3500
debug: [fcw] Sending instantiate req targets=[grpc.http2.keepalive_time=300, grpc.keepalive_time_ms=300000, grpc.http2.keepalive_t
imeout=35, grpc.keepalive_timeout_ms=3500, grpc.max_receive_message_length=-1, grpc.max_send_message_length=-1, grpc.primary_user_
agent=grpc-node/1.10.1, _url=grpc://localhost:7051, addr=localhost:7051, , _request_timeout=90000, , _name=null], chaincodeId=marb
les, chaincodeVersion=v4, fcn=init, args=[12345], 0=101, 1=48, 2=228, 3=252, 4=100, 5=66, 6=0, 7=104, 8=173, 9=236, 10=234, 11=171
, 12=54, 13=95, 14=223, 15=242, 16=81, 17=136, 18=232, 19=161, 20=35, 21=50, 22=217, 23=177, _transaction_id=16e8aea35fd0c59aa205d
bfe06d4136ff34bd97414a9f46b61355f228d308272
debug: [fcw] Successfully obtained transaction endorsement
debug: [fcw] Successfully ordered instantiate endorsement.
-----
info: Instantiate done. Errors: nope
-----
```

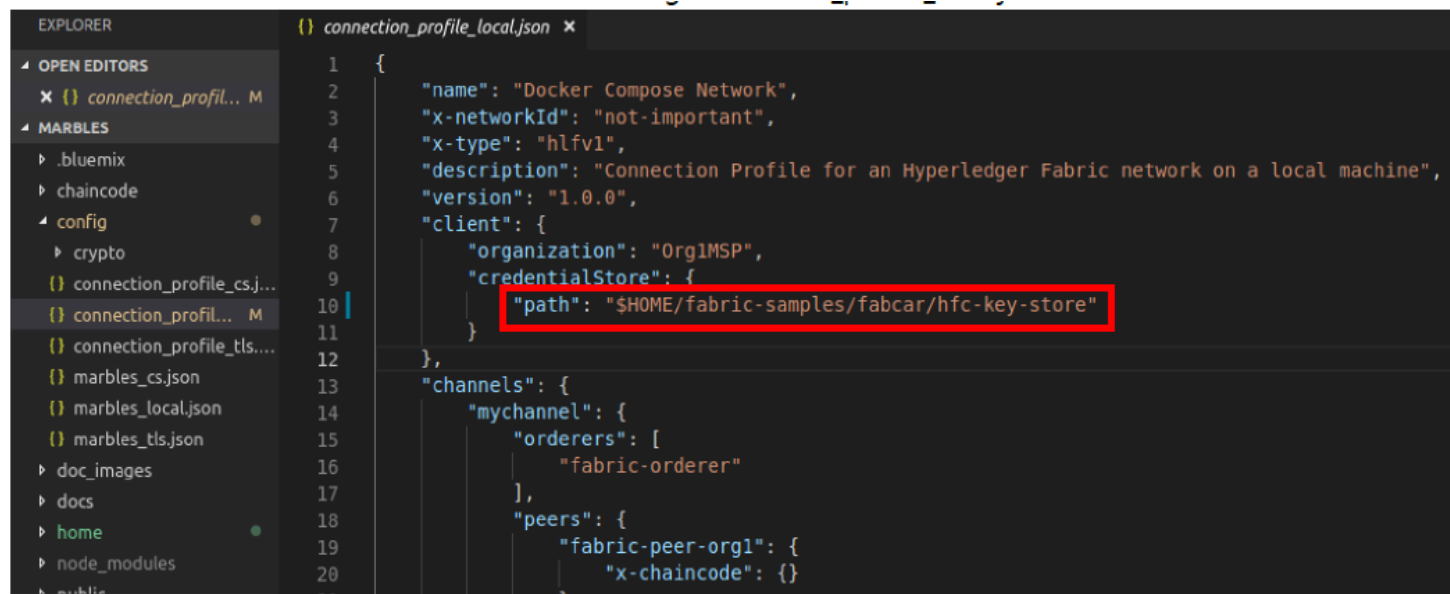
Marbles 설치

- 애플리케이션 실행
 - Marble을 작동하기 위한 서버 필요
 - 자동화 빌드 패키지 설치

```
$ cd $HOME/marbles  
$ sudo npm install gulp -g  
$ npm install
```

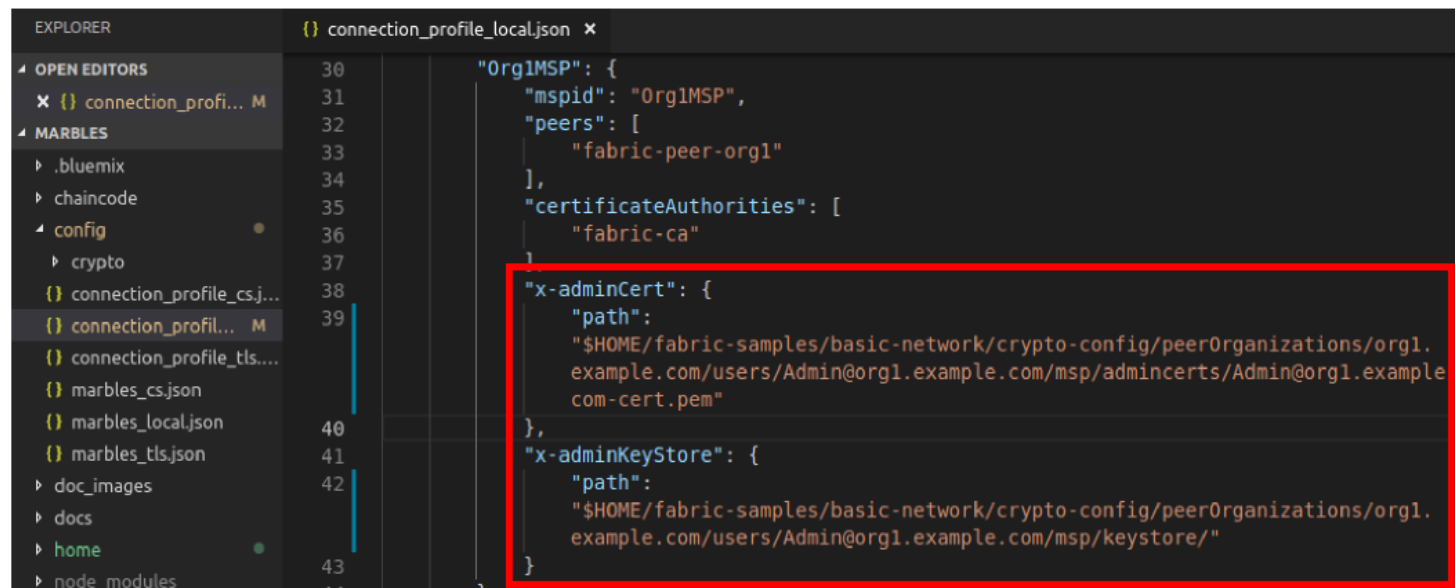
- \$HOME/marbles/config/connection_profile_location.json 수정/확인

Marbles 설치



```
EXPLORER
  OPEN EDITORS
    {} connection_profil... M
  MARBLES
    .bluemix
    chaincode
    config
      crypto
    {} connection_profile_cs.j...
    {} connection_profil... M
    {} connection_profile_tls....
    {} marbles_cs.json
    {} marbles_local.json
    {} marbles_tls.json
    doc_images
    docs
    home
    node_modules
    public

1 {} connection_profile_local.json x
2 {
3   "name": "Docker Compose Network",
4   "x-networkId": "not-important",
5   "x-type": "hlfv1",
6   "description": "Connection Profile for an Hyperledger Fabric network on a local machine",
7   "version": "1.0.0",
8   "client": {
9     "organization": "Org1MSP",
10    "credentialStore": {
11      "path": "$HOME/fabric-samples/fabcar/hfc-key-store"
12    }
13  },
14  "channels": {
15    "mychannel": {
16      "orderers": [
17        "fabric-orderer"
18      ],
19      "peers": {
20        "fabric-peer-org1": {
21          "x-chaincode": {}
22        }
23      }
24    }
25  }
26 }
```



```
EXPLORER
  OPEN EDITORS
    {} connection_profi... M
  MARBLES
    .bluemix
    chaincode
    config
      crypto
    {} connection_profile_cs.j...
    {} connection_profil... M
    {} connection_profile_tls....
    {} marbles_cs.json
    {} marbles_local.json
    {} marbles_tls.json
    doc_images
    docs
    home
    node_modules

30 "Org1MSP": {
31   "mspid": "Org1MSP",
32   "peers": [
33     "fabric-peer-org1"
34   ],
35   "certificateAuthorities": [
36     "fabric-ca"
37   ]
38 }
39 "x-adminCert": {
40   "path":
41     "$HOME/fabric-samples/basic-network/crypto-config/peerOrganizations/org1.
42     example.com/users/Admin@org1.example.com/msp/admincerts/Admin@org1.example
43     com-cert.pem"
44 },
45 "x-adminKeyStore": {
46   "path":
47     "$HOME/fabric-samples/basic-network/crypto-config/peerOrganizations/org1.
48     example.com/users/Admin@org1.example.com/msp/keystore/"
49 }
50 }
```

Marbles 실행

- 애플리케이션 실행

```
$ gulp marbles_local
```

- 서버 기동 시 오류 발생할 경우

- ***\$HOME/.hfc-key-store/*** 디렉토리의 모든 파일을 삭제
- fabcar 디렉토리의 ***hfc-key-store/*** 디렉토리의 모든 파일을 ***\$HOME/.hfc-key-store/***에 복사

```
$ rm -rf $HOME/.hfc-key-store/
```

```
$ cp $HOME/fabric-samples/fabcar/.hfc-key-store/* $HOME/.hfc-key-store/
```

- *Guest OS Port forwarding (Vagrant)*

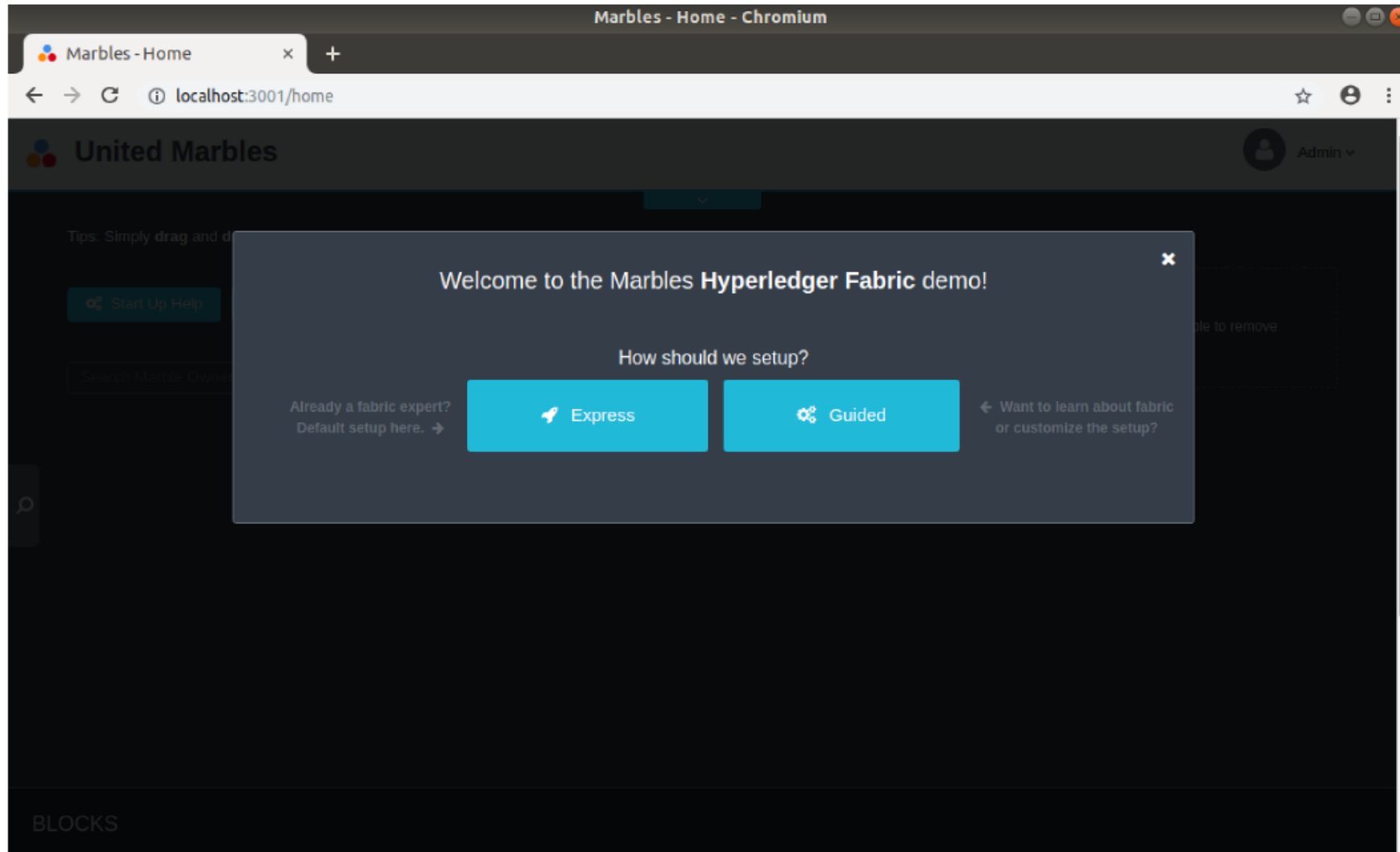
```
> vagrant halt  
> vi Vagrantfile
```

```
22 # Create a forwarded port mapping which allows access to a specific port  
23 # within the machine from a port on the host machine. In the example below,  
24 # accessing "localhost:8080" will access port 80 on the guest machine.  
25 # NOTE: This will enable public access to the opened port  
26 config.vm.network "forwarded_port", guest: 7050, host: 7050  
27 config.vm.network "forwarded_port", guest: 5984, host: 5984  
28 config.vm.network "forwarded_port", guest: 3001, host: 3001  
29
```

```
> vagrant up  
> vagrant ssh
```

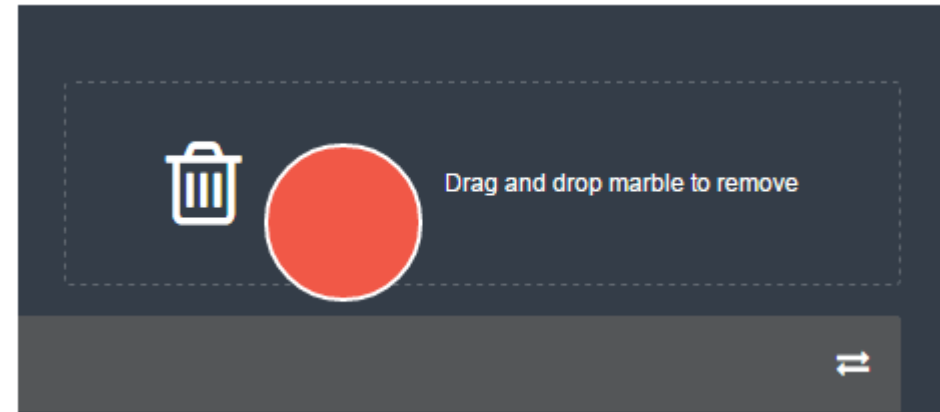
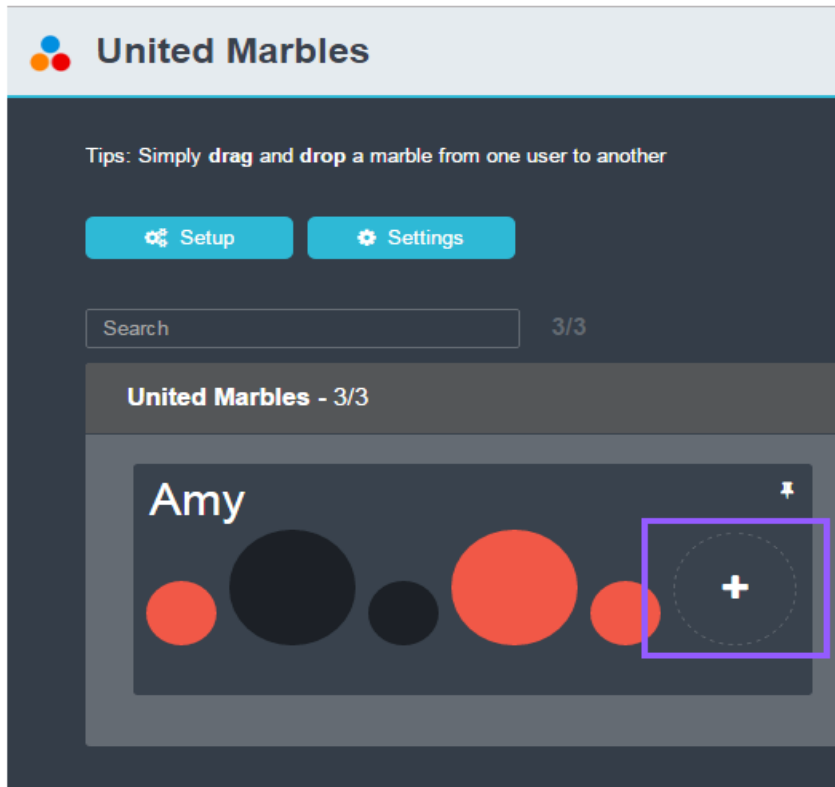
Marbles 사용

- <http://localhost:3001>



Marbles 사용

- 애플리케이션 테스트
 - Marble 생성
 - Marble 교환
 - Marble 소유권 이전
 - Marble 삭제



Marbles 사용

- Hyperledger Fabric CLI로 자산활동 확인

```
$ docker exec peer0.org1.example.com peer chaincode query ₩  
  -C mychannel ₩  
  -n marbles ₩  
  -c '{"Args": [""read_everything"]}'
```

- JSON 데이터 정렬하기

```
$ sudo apt -y install jq
```

```
$ docker exec peer0.org1.example.com peer chaincode query ₩  
  -C mychannel -n marbles -c '{"Args": [""read_everything"]}' ₩  
  | grep 'Query Result' | sed -e 's/Query Result://g' ₩  
  | jq '.marbles[] | ""₩(owner.username) ₩(color) ₩(size) ₩ (.id)""' | sort
```