Practical 5

Aim: Car auction network: A Hello World example with Hyperledger Fabric Node SDK and IBM Block chain Starter Plan. Use Hyperledger Fabric to invoke chaincode while storing results and data in the starter plan

Setup the blockchain network

Open a new terminal and make sure the network is down before starting

```
cd test-network
./network.sh down
```

```
→ fabric-samples git:(v2.2.0) cd test-network
→ test-network git:(v2.2.0) ./network.sh down
Stopping network
```

Remove any existing/running docker containers

```
docker rm -f $(docker ps -aq)
docker rmi -f $(docker images | grep fabcar | awk '{print $3}')
```

```
→ test-network git:(v2.2.0) docker rm -f $(docker ps -aq)
docker rmi -f $(docker images | grep fabcar | awk '{print $3}')

"docker rm" requires at least 1 argument.
See 'docker rm --help'.

Usage:  docker rm [OPTIONS] CONTAINER [CONTAINER...]

Remove one or more containers
"docker rmi" requires at least 1 argument.
See 'docker rmi --help'.

Usage:  docker rmi [OPTIONS] IMAGE [IMAGE...]

Remove one or more images
```

Starting the network

For this Network we will be using Javascript

```
cd fabcar
./startFabric.sh javascript
```

```
→ fabric-samples git:(v2.2.0) cd fabcar
→ fabcar git:(v2.2.0) ./startFabric.sh javascript
~/web3/fabric-samples/test-network ~/web3/fabric-samples/fabcar
Stopping network
Removing network fabric_test
WARNING: Network fabric_test not found.
```

You need to be in the javascript directory
Run the following command to install the Fabric dependencies for the applications. It will take about a minute to complete:

```
cd javascript
npm i
```

```
→ fabcar git:(v2.2.0) cd javascript
→ javascript git:(v2.2.0) npm i
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.
th.random() in certain circumstances, which is known to be problematic.
ath-random for details.
npm WARN deprecated mkdirp@0.5.1: Legacy versions of mkdirp are no longe
 to mkdirp 1.x. (Note that the API surface has changed to use Promises i
npm WARN deprecated sinon@7.5.0: 16.1.1

added 358 packages, and audited 359 packages in 29s
```

Once **npm i** completes, everything is in place to run the application. For this tutorial, you'll primarily be using the application JavaScript files in the **`fabcar/javascript`** directory. Let's take a look at what's inside:

```
ls
```

```
→ javascript git:(v2.2.0) ls
enrollAdmin.js  node_modules  package-lock.json  registerUser.js
invoke.js       package.json  query.js           wallet
→ javascript git:(v2.2.0)
```

Enrolling the admin user

We will subsequently register and enroll a new application user which will be used by our application to interact with the blockchain.

```
node enrollAdmin.js
```

```
→ javascript git:(v2.2.0) node enrollAdmin.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Successfully enrolled admin user "admin" and imported it into the wallet
→ javascript git:(v2.2.0)
```

Register and Enroll

Now that we have the administrator's credentials in a wallet, we can enroll a new user **user1** which will be used to query and update the ledger:

```
node registerUser.js
```

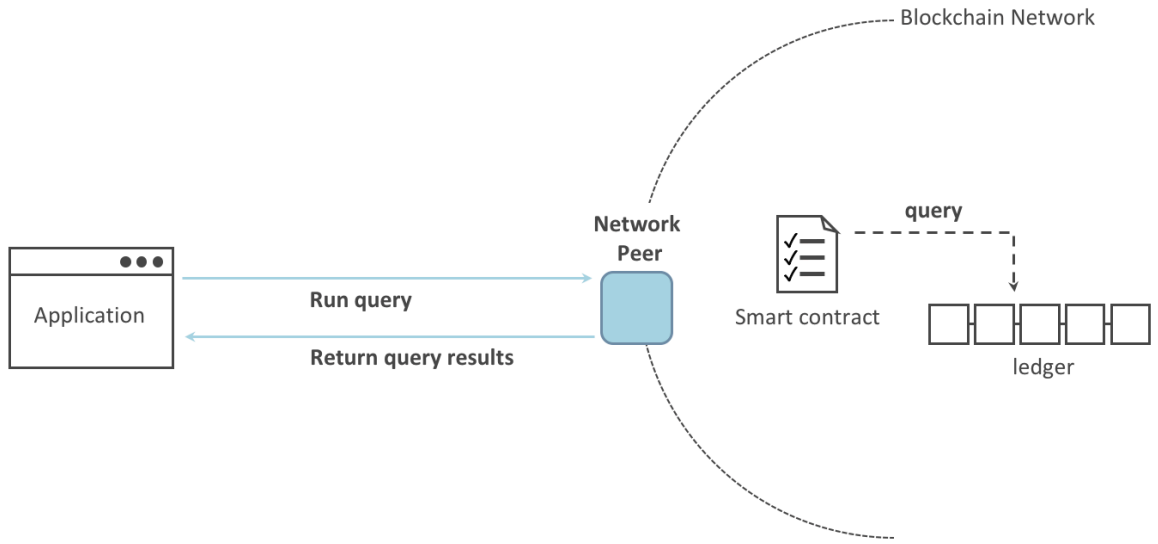
```
→ javascript git:(v2.2.0) node registerUser.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Successfully registered and enrolled admin user "appUser" and imported it into the wallet
→ javascript git:(v2.2.0)
```

Similar to the admin enrollment, this program uses a CSR to enroll **user1** and store its credentials alongside those of **admin** in the wallet. We now have identities for two separate users **admin** and **user1** and these are used by our application.

Querying the ledger

Each peer in a blockchain network hosts a copy of the ledger, and an application program can query the ledger by invoking a smart contract which queries the most recent value of the ledger and returns it to the application.

Here is a simplified representation of how a query works:



First, let's run our **query.js** program to return a listing of all the cars on the ledger. This program uses our second identity – **user1** – to access the ledger:

```
node query.js
```

```
→ javascript git:(v2.2.0) node query.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: [{"Key":"CAR0","Record":{"color":"blue","docType":"car"
,"make":"Toyota","model":"Prius","owner":"Tomoko"}},{"Key":"CAR1","Record":{"color":"red","docType
":"car","make":"Ford","model":"Mustang","owner":"Brad"}},{"Key":"CAR2","Record":{"color":"green","
docType":"car","make":"Hyundai","model":"Tucson","owner":"Jin Soo"}},{"Key":"CAR3","Record":{"colo
r":"yellow","docType":"car","make":"Volkswagen","model":"Passat","owner":"Max"}},{"Key":"CAR4","Re
cord":{"color":"black","docType":"car","make":"Tesla","model":"S","owner":"Adriana"}},{"Key":"CAR5
","Record":{"color":"purple","docType":"car","make":"Peugeot","model":"205","owner":"Michel"}},{"K
ey":"CAR6","Record":{"color":"white","docType":"car","make":"Chery","model":"S22L","owner":"Aarav"
}},{"Key":"CAR7","Record":{"color":"violet","docType":"car","make":"Fiat","model":"Punto","owner":
"Pari"}},{"Key":"CAR8","Record":{"color":"indigo","docType":"car","make":"Tata","model":"Nano","ow
ner":"Valeria"}},{"Key":"CAR9","Record":{"color":"brown","docType":"car","make":"Holden","model":"
Barina","owner":"Shotaro"}}]
```

Changing query.js code

change the **evaluateTransaction** request to query **CAR4**.The **query** program should now look like this:

```
const result = await contract.evaluateTransaction('queryCar',
'CAR4');
```

```
const result = await contract.evaluateTransaction('queryCar', 'CAR4');
console.log(`Transaction has been evaluated, result is: ${result.toString()}`);
```

Save the program and run the **query** program again:

```
node query.js
```

```
→ javascript git:(v2.2.0) ✗ node query.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: {"color":"black","docType":"car","make":"Tesla","model":"S","owner":"Adriana"}
→ javascript git:(v2.2.0) ✗
```

Updating the ledger

Our first update to the ledger will create a new car. We have a separate program called **invoke.js** that we will use to make updates to the ledger. Just as with queries, use an editor to open the program and navigate to the code block where we construct our transaction and submit it to the network:

```
await contract.submitTransaction('createCar', 'CAR12', 'Honda', 'Accord', 'Black', 'Tom');
console.log('Transaction has been submitted');
```

Run the program

```
node invoke.js
```

```
→ javascript git:(v2.2.0) ✗ node invoke.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Transaction has been submitted
→ javascript git:(v2.2.0) ✗
```

Changing the query.js to fetch the new transaction

```
const result = await contract.evaluateTransaction('queryCar', 'CAR12');
console.log(`Transaction has been evaluated, result is: ${result.toString()}`);
```

Save and run

```
node query.js
```

```
→ javascript git:(v2.2.0) ✗ node query.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: {"color":"Black","docType":"car","make":"Honda","model":"Accord","owner":"Tom"}
→ javascript git:(v2.2.0) ✗
```

Changing the ownership of car by making changes in invoke.js

To do this, go back to **invoke.js** and change the smart contract transaction from **createCar** to **changeCarOwner** with a corresponding change in input arguments

```
await contract.submitTransaction('changeCarOwner', 'CAR12',
'Dave');
```

```
// await contract.submitTransaction('createCar', 'CAR12', 'Honda', 'Accord', 'Black', 'Tom');
await contract.submitTransaction('changeCarOwner', 'CAR12', 'Dave');
console.log('Transaction has been submitted');
```

Save and run

```
node invoke.js
```

```
→ javascript git:(v2.2.0) ✗ node invoke.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Transaction has been submitted
→ javascript git:(v2.2.0) ✗
```

Verifying if the Owner for Car12 is updated

node query.js

```
→ javascript git:(v2.2.0) ✗ node query.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: {"color":"Black","docType":"car","make":"Honda","model":"Accord","owner":"Dave"}
→ javascript git:(v2.2.0) ✗
```

The ownership of **CAR12** has been changed from Tom to Dave.