GoLedger Challenge

Now you will create your own permissioned blockchain. For that, you will bring up a network, create a channel, join a peer to that channel, install and instantiate a smart contract (aka chaincode in Hyperledger Fabric). After that, you will issue a transaction in the chaincode, change the source code of the chaincode and upgrade the network with the new smart contract. In **Part I** we will provide step-by-step instructions and in **Part II** you must complete the instructions on your own.

To accomplish that, we recommend you use a UNIX-like machine (Linux/macOS). Besides that, we will need to install cURL and Docker.

Install the prerequisites

- Install cURL (https://curl.haxx.se/download.html)
- Install Docker and Docker Compose (https://www.docker.com/)
- Install Go (https://golang.org/dl/)
- Fork the repository https://github.com/goledgerdev/goledger-challenge
  - Fork it, do **NOT** clone it, since you will need to send us your forked repository
  - If you cannot fork it, create a private repository and give access to `samuelvenzi` and `vieiramanoel`
  - Make sure you the repository is in your `$GOPATH`
- Download the Docker images of Hyperledger

```
curl -sSL http://bit.ly/2ysbOFE | bash -s -- 1.4.4 1.4.4 0.4.18
```

  - Make sure that the Fabric binaries are in your `$PATH`

PART I

Set up the network

Generate certificates and Create channel artifacts

Go inside the folder /goledger-network. Now, we need to generate the certificates that will be used by the peer and the orderer so that they can sign and validate transactions.

```
./bgldgr.sh generate
```

This command will create the needed certificates and the channel artifacts.

Bring up the network containers

Assuming that you have installed Docker and Docker Compose, the following command will bring up the network elements.

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

You should see the following output:

```
Creating orderer.example.com    ... done
Creating peer0.org1.example.com ... done
Creating cli                    ... done
```

Create and Join Channel

When deploying a local network, we can use the `cli` container to do operations in the network. For that, we need to be inside the container. Use the following command to get inside it.

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

Create the channel.

```
peer channel create
    -o orderer.example.com:7050
    -c mychannel
    -f ./channel-artifacts/channel.tx
    --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
```

Join the channel.

```
peer channel join -b mychannel.block
```

Install and Instantiate Chaincode

Still inside the `cli` container, install the chaincode.

```
peer chaincode install -n fabcar -v 1.0 -p github.com/chaincode/fabcar/go/
```

Instantiate the chaincode.

```
peer chaincode instantiate
    -o orderer.example.com:7050
    --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
    -C mychannel
    -n fabcar
    -v 1.0
    -c '{"Args":["init"]}'
```

Make sure everything is in order by checking the instantiated chaincodes on the channel with the command:

```
peer chaincode list --instantiated -C mychannel
```

Print your terminal and save it as `inst0.jpg`

FABCAR Chaincode

The chaincode you just instantiated in your newly created channel is called FABCAR and is one of the sample chaincodes provided by Hyperledger. This chaincodes takes care of registering and managing cars and their characteristics like (manufacturer, model, color and owner). The source code is under `/chaincode/fabcar/go` in the root folder of the repository. **Make sure you take a look at the code so that you know what it's about.**

Initialize ledger

Your chaincode is instantiated clean, so let's initialize the ledger with a few cars. The FABCAR chaincode provides a operation so that you can do that with ease. To invoke a transaction you can run the following command (inside the `cli` container):

```
peer chaincode invoke
    -o orderer.example.com:7050
    -C mychannel
    -n fabcar
    -c '{"function":"initLedger","Args":[]}'
    --waitForEvent
    --tls
    --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganization
s/example.com/orderers/   orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
    --peerAddresses peer0.org1.example.com:7051
    --tlsRootCertFiles /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrgan
izations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
```

Now check the cars in the ledger:

```
peer chaincode query
    -C mychannel
    -n fabcar
    -c '{"Args":["queryAllCars"]}'
```

Print your terminal and save it as `query0.jpg`

PART II

Now that you have your network up and running with the FABCAR chaincode, you must accomplish the following tasks:

- Query only the car identified by key `CAR0`
  - print your terminal, save it as `query1.jpg`
- Change the owner of that car to anyone you want
  - query that car again, print your terminal, save it as `query2.jpg`
- Change the source code of the chaincode and add the optional attribute “Year” to the cars’ attributes
- Install and upgrade your new chaincode in the network
  - check the instantiated chaincodes, print your terminal, save it as `inst1.jpg`
- Create one new car
  - query your newly created car, print your terminal, save it as `query3.jpg`
- Create a folder `prints` in the root of the repository and put the images you generated inside it
- **Commit** your changes (`prints` folder and source code) to your forked repository

To complete the challenge, you must send us the link to your forked repository with the alterations you made.