# Hyperledger Fabric

## Fabric Test Network

### Prerequisites installation and environment setup

First of all, we need to download the latest version of Git, cURL and Go Language (optional but highly recommended because Go is helpful when we start to use chaincode) which can be found in the following websites:

<https://git-scm.com/downloads>

Next step is installing Docker which provides an operating platform for Hyperledger Fabric. For Mac OS, \*nix, or Windows 10 users, you can download it at the first link. For older versions of Windows users, you can find the instruction to download Docker Toolbox at the second link.

<https://www.docker.com/get-started> <https://docs.docker.com/toolbox/toolbox_install_windows/>

Installing Docker will also automatically install Docker Compose. We should check the version of Docker (version 17.06.2-ce or greater is required) and Docker Compose (version 1.14.0 or greater is required) with the following command from a terminal prompt:

$ docker --version

$ docker-compose --version

Be aware that if you are Window 10 users, you need some extra configuration steps. Before running any 'git clone' commands, you should run following commands:

$ git config --global core.autocrlf false

$ git config --global core.longpaths true

You can check the state of these parameters by entering:

$ git config --get core.autocrlf

$ git config --get core.longpaths

These need to be 'false' and 'true' respectively.

After downloading Docker and Docker Compose, we need to determine a location or create a folder where we want to put 'fabric-samples' repository in. Entering that folder from a terminal, we are going to run the following command:

$ curl -sSL https://bit.ly/2ysbOFE | bash -s

Previous command includes downloading the Hyperledger Fabric Docker images for the latest version and installing the Hyperledger Fabric platform-specific binaries and config files for the latest production release.

The setup is finished.

### Bringing up the test network

Now we have installed all prerequisites, binaries and images. We can officially start to interact with Hyperledger Fabric test network. By typing in the following command, we can enter the 'test- network' directory in 'fabric-samples' folder.

$ cd fabric-samples/test-network

Before bringing up the network, printing the script help text gives us the first impression about this network, its modes and its flags.

$ ./network.sh -h

'./network.sh' represents the Hyperledger Fabric network which are using Docker on a local machine. Also inside this directory, we need to run the following command in order to check and remove all possible Docker containers or other artifacts from any previous runs.

$ ./network.sh down

Now it is time to officially bring up the test network.

$ ./network.sh up

If successful, we will see the similar information to this:

Creating network "net\_test" with the default driver

Creating volume "net\_orderer.example.com" with default driver Creating volume "net\_peer0.org1.example.com" with default driver Creating volume "net\_peer0.org2.example.com" with default driver

Creating peer0.org1.example.com ... done

Creating orderer.example.com ... done Creating peer0.org2.example.com ... done

CONTAINER ID IMAGE COMMAND

CREATED STATUS PORTS NAMES

26e05653a9dd hyperledger/fabric-peer:latest "peer node start" Less than a second ago Up Less than a second 7051/tcp, 0.0.0.0:9051->9051/tcp peer0.org2.example.com

e1bdf3a2efdc hyperledger/fabric-orderer:latest "orderer" Less than a second ago Up Less than a second 0.0.0.0:7050->7050/tcp orderer.example.com

1f79093369e9 hyperledger/fabric-peer:latest "peer node start" Less than a second ago Up Less than a second 0.0.0.0:7051->7051/tcp peer0.org1.example.com

There are three nodes in the test network: two peers and one orderer. We have talked about their definitions and functions in both Lecture 9 and previous introduction section. The following command displays the list of components in this network:

$ docker ps -a

### Creating a channel

Next step is creating a channel between two peers. This channel is a private layer for the communication and transaction between Org1 and Org2 who are invited into this channel. We can create the channel by running following command:

$ ./network.sh createChannel

The default name of this channel is 'mychannel'. If you want to create a channel with a specific name, you can try the following command:

$ ./network.sh createChannel -c channel1

### Starting a chaincode on the channel

The lab explained the definition of chaincode in the previous introduction section. The following command will install the asset-transfer (basic) chaincode on Org1 and Org2 and then deploy the chaincode on default channel or a specified channel.

$ ./network.sh deployCC

### Interacting with the network

To interact with our network, now we are able to use 'peer' CLI (command-line interface) which gives us an access to use deployed contract and update the channel. The following command will help us to add 'peer' binaries, which is in the 'bin' folder of 'fabric-samples' repository, to our CLI path.

$ export PATH=${PWD}/../bin:$PATH

At same time, we need to set 'FABRIC\_CFG\_PATH' to the 'core.yaml', which is in the 'config' folder of 'fabric-samples' repository.

$ export FABRIC\_CFG\_PATH=$PWD/../config/

We can now set the environment variables for Org1 with the following commands:

$ export CORE\_PEER\_TLS\_ENABLED=true

$ export CORE\_PEER\_LOCALMSPID="Org1MSP"

$ export CORE\_PEER\_TLS\_ROOTCERT\_FILE=${PWD}/organizations/peerOrganizations/org1.

example.com/peers/peer0.org1.example.com/tls/ca.crt

$ export CORE\_PEER\_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.exa mple.com/users/Admin@org1.example.com/msp

$ export CORE\_PEER\_ADDRESS=localhost:7051

Then we need to initialize the ledger with the assets.

$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile

${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.co m/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles

${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example. com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles

${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example. com/tls/ca.crt -c '{"function":"InitLedger","Args":[]}'

If successful, we can see the result which are similar to this:

2020-08-14 18:17:41.809 EST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200

Now we can query the assets in our ledger with the following command:

$ peer chaincode query -C mychannel -n basic -c '{"Args":["GetAllAssets"]}'

If successful, we can see the output below:

[

{"ID": "asset1", "color": "blue", "size": 5, "owner": "Tomoko", "appraisedValue": 300},

{"ID": "asset2", "color": "red", "size": 5, "owner": "Brad", "appraisedValue": 400},

{"ID": "asset3", "color": "green", "size": 10, "owner": "Jin Soo", "appraisedValue": 500},

{"ID": "asset4", "color": "yellow", "size": 10, "owner": "Max", "appraisedValue": 600},

{"ID": "asset5", "color": "black", "size": 15, "owner": "Adriana", "appraisedValue": 700},

{"ID": "asset6", "color": "white", "size": 15, "owner": "Michel", "appraisedValue": 800}

]

With the assets in our ledger, we are able to make the transaction between two peers. Before doing that, we need to invoke the asset-transfer (basic) chaincode by running the following command:

$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile

${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.co m/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles

${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example. com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles

${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example. com/tls/ca.crt -c '{"function":"TransferAsset","Args":["asset6","Christopher"]}'

If successful, we can see a similar output to this:

2020-08-14 18:43:18.626 EST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200

In order to query the asset of Org2, we need to set the environment variables by typing in the following commands:

$ export CORE\_PEER\_TLS\_ENABLED=true

$ export CORE\_PEER\_LOCALMSPID="Org2MSP"

$ export

CORE\_PEER\_TLS\_ROOTCERT\_FILE=${PWD}/organizations/peerOrganizations/org2. example.com/peers/peer0.org2.example.com/tls/ca.crt

$ export CORE\_PEER\_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2.exa mple.com/users/Admin@org2.example.com/msp

$ export CORE\_PEER\_ADDRESS=localhost:9051

Now we can query a specific asset stored in Org2 by running:

$ peer chaincode query -C mychannel -n basic -c '{"Args":["ReadAsset","asset6"]}'

If successful, we can see the output below:

{"ID":"asset6","color":"white","size":15,"owner":"Christopher","appraisedValue":800}

### Bringing down the network

At this point, we finished the interaction with Fabric test network and we can now use the following command to bring down the network as well as stop and remove the nodes and containers.

$./network.sh down