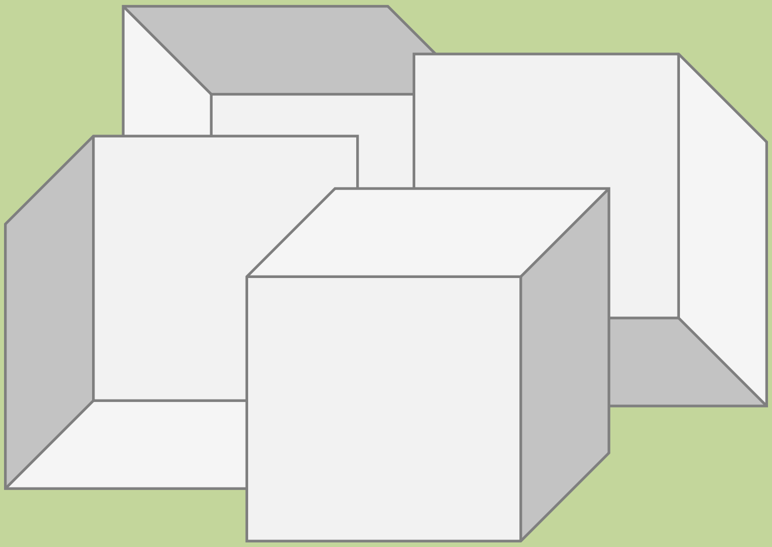

Blockchain Course

Land Record System

- Configure and Test Network

(Detailed)

Raghunandan Mishra

1. CREATE CRYPTOGRAPHIC MATERIAL AND CHANNEL CONFIGURATION.

i. Create **MSP** and **cryptographic material.**

Set environment variable $PATH `export PATH=~/bin:$PATH`

Change directory: `cd ~/fabric-samples/lrsnetwork`

ii. Run **cryptogen tool** to create orgs. This will create "crypto-config" directory that contains MSP material.

Input is the “crypto-config.yaml” file that contains the Org definition.

```
~/bin/cryptogen generate --config=./crypto-config.yaml
```

```
##### Generate certificates using cryptogen tool #########
============= Citizen.org =============
============= Registrar.org =============
============= Endorser.org =============
```

iii. Declare the path of **configtx.yaml** for **configtxgen** tool.

```
export FABRIC_CFG_PATH=$PWD
```

iv. Invoke **configtxgen** tool. This tool creates channel configuration. Output of this and later commands can be found in "channel-artifacts" directory.

Input is the profile name: “ThreeOrgsOrdererGenesis” defined within the “configtx.yaml” file.

```
~/bin/configtxgen -profile ThreeOrgsOrdererGenesis -outputBlock ./channel-
artifacts/genesis.block
```

```
######### Generating Orderer Genesis block ##############
```

v. **Create channel configuration** transaction. The **channel.tx** artifact contains the definitions for our sample channel. This command generates new channel and writes new **channel.tx** (basically introduce orgs to the channel created). The channel name is hard-coded to “mychannel”.

Input is the profile name: “ThreeOrgsChannel” defined within the “configtx.yaml” file.

```
export CHANNEL_NAME=mychannel && ~/bin/configtxgen -profile ThreeOrgsChannel -
outputCreateChannelTx ./channel-artifacts/channel.tx -channelID $CHANNEL_NAME
```

```
### Generating channel configuration transaction 'channel.tx' ###
```

vi. Define **anchor peers** from the org on the channel.

Input: we have defined the organization MSP IDs as: “CitizenMSP”, “RegistrarMSP” and “EndorserMSP” within the “configtx.yaml” file.

Input: The corresponding anchors are identified as: “Org1MSPanchors”, “Org2MSPanchors” and “Org3MSPanchors”. Anchor peer is defined as peer0 node for each of these organizations. These definitions are a part of “configtx.yaml” file.

```
~/bin/configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID $CHANNEL_NAME -asOrg CitizenMSP

~/bin/configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID $CHANNEL_NAME -asOrg RegistrarMSP

~/bin/configtxgen -profile ThreeOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org3MSPanchors.tx -channelID $CHANNEL_NAME -asOrg EndorserMSP
```

```
#######    Generating anchor peer update for CitizenMSP   ##########
#######    Generating anchor peer update for RegistrarMSP   ##########
#######    Generating anchor peer update for EndorserMSP   ##########
```

2. SET ENVIRONMENT VARIABLES MANUALLY AS WE ARE IN DEVELOPMENT ENVIRONMENT.

These variables are going to be used multiple times within the test. For each of the orgs, we will define four variable commands below and reference them by a single name whenever we want to use it. Please note that you will have to enter the full command while running the tests.

i. Environment variables for **"Citizen Org Env."**.

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/citizen.org/users/Admin@citizen.org/msp

CORE_PEER_ADDRESS=peer0.citizen.org:7051

CORE_PEER_LOCALMSPID="CitizenMSP"

CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/citizen.org/peers/peer0.citizen.org/tls/ca.crt
```

ii. Environment variables for **"Registrar Org Env."**.

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/registrar.org/users/Admin@registrar.org/msp

CORE_PEER_ADDRESS=peer0.registrar.org:7051

CORE_PEER_LOCALMSPID="RegistrarMSP"

CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/registrar.org/peers/peer0.registrar.org/tls/ca.crt
```

iii. Environment variables for **"Endorser Org Env."**.

```
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/endorser.org/users/Admin@registrar.org/msp

CORE_PEER_ADDRESS=peer0.endorser.org:7051

CORE_PEER_LOCALMSPID="EndorserMSP"

CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/endorser.org/peers/peer0.endorser.org/tls/ca.crt
```

3. CREATE AND START DOCKER CONTAINER. ENTER THE CONTAINER CLI.

i. Create and start **docker container “cli”.**

Input: Two variable names: IMAGE_TAG and PROJECT_NAME

Input: docker file that contains configurations to create a container: “docker-compose-cli.yaml”

```
IMAGE_TAG=x86_64-1.1.0 COMPOSE_PROJECT_NAME=lrsnetwork docker-compose -f docker-compose-cli.yaml up -d
```

```
Creating network "lrsnetwork_lrsfn" with the default driver
Creating volume "lrsnetwork_peer0.endorser.org" with default driver
Creating volume "lrsnetwork_orderer.example.com" with default driver
Creating volume "lrsnetwork_peer0.citizen.org" with default driver
Creating volume "lrsnetwork_peer0.registrar.org" with default driver
Creating volume "lrsnetwork_peer2.citizen.org" with default driver
Creating volume "lrsnetwork_peer1.citizen.org" with default driver
Creating volume "lrsnetwork_peer3.citizen.org" with default driver
Creating peer0.citizen.org ... done
Creating cli ... done
Creating orderer.example.com ...
Creating peer0.endorser.org ...
Creating peer1.registrar.org ...
Creating peer1.citizen.org ...
Creating peer0.registrar.org ...
Creating peer0.citizen.org ...
Creating peer3.citizen.org ...
Creating cli …
```

4A. CREATE A CHANNEL AND ADD PEER0 OF CITIZEN.ORG TO THE CHANNEL.

i. Set **"Citizen Org Env. – peer0"** (copy commands from section 2.i with peer0)

ii. Enter the **command line interface** of the container

Input: container name i.e., "cli"

```
docker exec –it cli bash
```

If successful, we can see:
root@0d78bb69300d:/opt/gopath/src/github.com/hyperledger/fabric/peer#

iii. Create **Channel** by passing channel.tx file. The channel.tx file is mounted in the channel-artifacts directory within your CLI container. As a result, we pass the full path for the file. We also pass the path for the orderer ca-cert in order to verify the TLS handshake.

Input: channel name "mychannel" and channel.tx that was previously created

```
export CHANNEL_NAME=mychannel
```

Run the command to create channel

```
peer channel create -o orderer.example.com:7050 -c $CHANNEL_NAME -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
```

A new block called **"mychannel.block"** is added to the **"genesis.block"**.

```
================= Channel "mychannel" is created successfully =================
```

iv. Join **peer0 of citizen.org** to the channel.

Input: channel name i.e., "mychannel"

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

peer0 of citizen.org joins the channel successfully.

```
================= peer0.org1 joined on the channel "mychannel" =================
```

v. Update the channel definition to **define anchor peer** for citizen org (peer0 is the anchor peer).

Input: MSP anchor definition file: Org1MSPanchor.tx

```
peer channel update -o orderer.example.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/Org1MSPanchors.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
```

peer0 of citizen.org is defined as anchor peer on the channel.

```
====== Anchor peers for org "CitizenMSP" on "mychannel" is updated successfully =======
```

4B. JOIN REMAINING PEERS OF CITIZEN.ORG TO THE CHANNEL.

i. Set **“Citizen Org Env. – peer1”** (copy commands from section 2.i with peer1)

ii. Join **peer1 of citizen.org** to the channel.

Input: channel name i.e., “mychannel”

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

peer1 of citizen.org joins the channel successfully.

```
================= peer1.org1 joined on the channel "mychannel" =================
```

iii. Run the above two steps by changing the environment variables in section 2.i with **peer2** and **peer3**

```
================= peer2.org1 joined on the channel "mychannel" =================

================= peer3.org1 joined on the channel "mychannel" =================
```

5. UPDATE THE CHANNEL BY ADDING PEERS OF REGISTRAR.ORG TO THE CHANNEL.

i. Set **“Registrar Org Env. – peer0”** (copy commands from section 2.ii with peer0)

ii. Join **peer0 of registrar.org** to the channel.

Input: channel name i.e., “mychannel”

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

peer0 of registrar.org joins the channel successfully.

```
================ peer0.org2 joined on the channel "mychannel" ================
```

iii. Update the channel definition to **define anchor peer** for registrar org (peer0 is the anchor peer).

Input: MSP anchor definition file: Org2MSPanchor.tx

```
peer channel update -o orderer.example.com:7050 -c $CHANNEL_NAME -f ./channel-
artifacts/Org2MSPanchors.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
```

peer0 of registrar.org is defined as anchor peer on the channel.

```
====== Anchor peers for org "RegistrarMSP" on "mychannel" is updated successfully =======
```

iv. Set **“Registrar Org Env. – peer1”** (copy commands from section 2.ii with peer1)

v. Join **peer1 of registrar.org** to the channel.

Input: channel name i.e., “mychannel”

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

peer1 of registrar.org joins the channel successfully.

```
================ peer1.org2 joined on the channel "mychannel" ================
```

6. UPDATE THE CHANNEL BY ADDING PEERS OF ENDORSER.ORG TO THE CHANNEL.

i. Set **"Endorser Org Env. – peer0"** (copy commands from section 2.iii with peer0)

ii. Join **peer0 of endorser.org** to the channel.

Input: channel name i.e., "mychannel"

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

peer0 of endorser.org joins the channel successfully.

```
================ peer0.org2 joined on the channel "mychannel" =================
```

iii. Update the channel definition to **define anchor peer** for registrar org (peer0 is the anchor peer).

Input: MSP anchor definition file: Org3MSPanchor.tx

```
peer channel update -o orderer.example.com:7050 -c $CHANNEL_NAME -f ./channel-
artifacts/Org3MSPanchors.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
```

peer0 of endorser.org is defined as anchor peer on the channel.

```
====== Anchor peers for org "EndorserMSP" on "mychannel" is updated successfully =======
```

7. INSTALL CHAINCODE ON ALL THE PEER0 (ANCHOR) NODES OF THE ORGS ON CHANNEL.

i. Set **“Citizen Org Env. – peer0”** (copy commands from section 2.i with peer0)

ii. Install **chaincode on peer0 of citizen.org** to the channel.

Input: channel name i.e., “mychannel”

```
peer chaincode install -n mycc -v 1.0 -l go -p
/opt/gopath/src/github.com/chaincode/chaincode_example02/go/
```

peer0 of citizen.org has installed chaincode successfully.

```
================== Chaincode is installed on peer0.org1 ===================
```

iii. Set **“Registrar Org Env. – peer0”** (copy commands from section 2.ii with peer0)

iv. Install **chaincode on peer0 of registrar.org** to the channel.

Input: channel name i.e., “mychannel”

```
peer chaincode install -n mycc -v 1.0 -l go -p
/opt/gopath/src/github.com/chaincode/chaincode_example02/go/
```

peer0 of registrar.org has installed chaincode successfully.

```
================== Chaincode is installed on peer0.org2 ===================
```

v. Set **“Endorser Org Env. – peer0”** (copy commands from section 2.iii with peer0)

vi. Install **chaincode on peer0 of endorser.org** to the channel.

Input: channel name i.e., “mychannel”

```
peer chaincode install -n mycc -v 1.0 -l go -p
/opt/gopath/src/github.com/chaincode/chaincode_example02/go/
```

peer0 of endorser.org has installed chaincode successfully.

```
================== Chaincode is installed on peer0.org3 ===================
```

8. INSTANTIATE CHAINCODE ON PEER0 (ANCHOR) NODE OF CITIZEN ORGS.

i. Set **"Endorser Org Env. – peer0"** (copy commands from section 2.iii with peer0)

ii. Instantiate **chaincode on peer0 of endorser.org** to the channel.

Input: channel name and chaincode name: "mychannel", "mycc" respectively

Input: Endorsement policy → either the CitizenMSP peer or RegistrarMSP peer can endorse

Input: a = 100; b = 200;

```
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
$CHANNEL_NAME -n mycc -l node -v 1.0 -c '{"Args":["init","a", "100", "b","200"]}' -P 'OR
('\''CitizenMSP.peer'\'', '\''RegistrarMSP.peer'\'')'
```

The values will be assigned to a & b and recorded in blockchain.

```
====== Chaincode Instantiation on peer0.org3 on channel 'mychannel' is successful =======
```

9. QUERY CHAINCODE ON PEER1 (ANCHOR) NODE OF CITIZEN ORGS.

i. Set **"Citizen Org Env. – peer1"** (copy commands from section 2.i with peer1)

ii. Query **chaincode on peer1 of citizen.org** to the channel.

Input: channel name and chaincode name: "mychannel", "mycc" respectively

Input: 'a';

```
peer chaincode query -C mychannel -n mycc -c '{"Args":["query","a"]}'
```

The values of a = 100 would be displayed.

```
======= Query on peer1.org1 on channel 'mychannel' is successful ======
```

10. INVOKE CHAINCODE TRANSACTION ON PEER0 (ANCHOR) NODE OF REGISTRAR ORGS.

i. Set **“Registrar Org Env. – peer0”** (copy commands from section 2.i with peer0)

ii. Invoke **chaincode on peer0 of registrar.org** to the channel.

Input: channel name and chaincode name: “mychannel”, “mycc” respectively

Input: ‘a’;

```
peer chaincode invoke -o orderer.example.com:7050 --tls true --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
mycc -c '{"Args":["invoke","a","b","10"]}'
```

The value 10 should be transferred from ‘a’ to ‘b’.

```
========= Invoke transaction on peer0.org1 on channel 'mychannel' is successful =========
```

10. QUERY CHAINCODE TRANSACTION ON PEER1 NODE OF CITIZEN ORGS.

i. Set **“Citizen Org Env. – peer1”** (copy commands from section 2.i with peer1)

ii. Install **chaincode on peer1 of citizen.org** to the channel.

Input: channel name i.e., “mychannel”

```
peer chaincode install -n mycc -v 1.0 -l go -p
/opt/gopath/src/github.com/chaincode/chaincode_example02/go/
```

peer1 of citizen.org has installed chaincode successfully.

```
=================== Chaincode is installed on peer1.org1 ===================
```

iii. Query **chaincode on peer1 of citizen.org** to the channel.

Input: channel name and chaincode name: “mychannel”, “mycc” respectively

Input: ‘a’;

```
peer chaincode query -C mychannel -n mycc -c '{"Args":["query","a"]}'
```

The value of a = 90 should be shown.

```
========= Query transaction on peer1.org1 on channel 'mychannel' is successful =========
```

NOTES:

1. Note that if we do not bring down the network, the container will remain and use the docker images that we had downloaded. Next time we try to work on the network, there may be a conflict in managing and orchestrating the docker container.
2. This is a development environment and hence all the nodes were installed, instantiated and executed on the same virtual machines using docker container capabilities. In production, peer nodes would interact with the container through REST APIs.
3. The above single script can be broken into unique commands to simulate each step one-by-one. Refer to document **9. LRS – Business Network Detailed.docx**.

CONGRATULATIONS! YOU NOW UNDERSTAND HOW TO CREATE, DEPLOY AND TEST A STANDALONE BLOCKCHAIN APPLICATION ON HYPERLEDGER FABRIC IN DETAIL. YOU SHOULD BE ABLE TO NOW CREATE YOUR OWN NETWORKS AND CHAINCODE IN DEV ENVIRONMENT AND TEST THEM SUCCESSFULLY !!!