Deep Dive

Instructions:

1. Make sure you watch over the logs to check what is happening with various docker instances when the commands are executed
2. Try and connect to the theory we discussed in the initial days
3. Always refer to the network.sh and the corresponding shell scripts to understand from the comments what is happening
4. **To watch for logs in the console, make changes to attach the docker logs to console**

In the test network folder do the following:

1. Set the Config Path

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

>> export FABRIC\_CFG\_PATH=${PWD}/configtx

1. Create the Certificates for Org1

>> cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output="organizations"

1. Create the certificates for Org2

>> cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output="organizations"

1. Create Certificates for OrdererOrg

>> cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output="organizations"

1. Generate the Genesis Block

>> configtxgen -profile TwoOrgsOrdererGenesis -channelID system-channel -outputBlock ./system-genesis-block/genesis.block

1. Start the docker containers

>> IMAGE\_TAG=latest docker-compose -f docker/docker-compose-test-net.yaml up

### Create Channel

1. Open a new Terminal
2. Set the Config Path

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

>> export FABRIC\_CFG\_PATH=${PWD}/configtx

>> export CHANNEL\_NAME=mychannel

1. Copy some pre-requisites

>> cp ../config/core.yaml ./configtx/.

1. Create the Channel Transaction

>> configtxgen -profile TwoOrgsChannel -outputCreateChannelTx ./channel-artifacts/$CHANNEL\_NAME.tx -channelID $CHANNEL\_NAME

1. Create the anchor peer transaction for Org1

>> configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org1MSPanchors.tx -channelID $CHANNEL\_NAME -asOrg Org1MSP

1. Create the anchor peer transaction for Org2

>> configtxgen -profile TwoOrgsChannel -outputAnchorPeersUpdate ./channel-artifacts/Org2MSPanchors.tx -channelID $CHANNEL\_NAME -asOrg Org2MSP

1. Set the context: to run commands as Org1 use 1 as argument

>> source ./scripts/setChannelContext.sh 1

1. Create the channel

>> peer channel create -o localhost:7050 -c $CHANNEL\_NAME --ordererTLSHostnameOverride orderer.example.com -f ./channel-artifacts/${CHANNEL\_NAME}.tx --outputBlock ./channel-artifacts/${CHANNEL\_NAME}.block --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA

1. Join Org1 to channel

>> peer channel join -b ./channel-artifacts/$CHANNEL\_NAME.block

1. Join Org2 to channel

>> source ./scripts/setChannelContext.sh 2

>> peer channel join -b ./channel-artifacts/$CHANNEL\_NAME.block

1. Update Anchor peer for Org1

>> source ./scripts/setChannelContext.sh 1

>> peer channel update -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c $CHANNEL\_NAME -f ./channel-artifacts/${CORE\_PEER\_LOCALMSPID}anchors.tx --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA

1. Update Anchor peer for Org2

>> source ./scripts/setChannelContext.sh 2

>> peer channel update -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c $CHANNEL\_NAME -f ./channel-artifacts/${CORE\_PEER\_LOCALMSPID}anchors.tx --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA

### Deploy ChainCode

1. Update the environment variable to configure use of GoLang Chaincode

>> source ./scripts/setGoLangContext.sh

>> export FABRIC\_CFG\_PATH=${PWD}/configtx

>> export CHANNEL\_NAME=mychannel

1. Package the chaincode

>> source ./scripts/setChannelContext.sh 1

>> peer lifecycle chaincode package fabcar.tar.gz --path ${CC\_SRC\_PATH} --lang ${CC\_RUNTIME\_LANGUAGE} --label fabcar\_${VERSION}

1. Check if the package is created; fabcar.tar.gz file should be seen.

>> ls

1. Install the chaincode on peer of Org1

>> peer lifecycle chaincode install fabcar.tar.gz

1. Install the chaincode on peer of Org2

>> source ./scripts/setChannelContext.sh 2

>> peer lifecycle chaincode install fabcar.tar.gz

1. Query for Installed package

>> peer lifecycle chaincode queryinstalled 2>&1 | tee outfile

1. Set the PACKAGE\_ID value

>> source ./scripts/setPackageID.sh outfile

1. Approve the Chaincode as Org1

>> source ./scripts/setChannelContext.sh 1

>> peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA --channelID $CHANNEL\_NAME --name fabcar --version ${VERSION} --init-required --package-id ${PACKAGE\_ID} --sequence ${VERSION}

1. Check for commitrediness as Org1

>> peer lifecycle chaincode checkcommitreadiness --channelID $CHANNEL\_NAME --name fabcar --version ${VERSION} --sequence ${VERSION} --output json --init-required

1. Check for commitrediness as Org2

>> source ./scripts/setChannelContext.sh 2

>> peer lifecycle chaincode checkcommitreadiness --channelID $CHANNEL\_NAME --name fabcar --version ${VERSION} --sequence ${VERSION} --output json --init-required

1. Approve the Chaincode as Org2

>> peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA --channelID $CHANNEL\_NAME --name fabcar --version ${VERSION} --init-required --package-id ${PACKAGE\_ID} --sequence ${VERSION}

1. Check for commitrediness as Org2

>> peer lifecycle chaincode checkcommitreadiness --channelID $CHANNEL\_NAME --name fabcar --version ${VERSION} --sequence ${VERSION} --output json --init-required

1. Check for commitrediness as Org1

>> source ./scripts/setChannelContext.sh 1

>> peer lifecycle chaincode checkcommitreadiness --channelID $CHANNEL\_NAME --name fabcar --version ${VERSION} --sequence ${VERSION} --output json --init-required

1. Set the peer address for identifying the endorsing peers

>> source ./scripts/setPeerConnectionParam.sh 1 2

1. Commit the chaincode definition to Channel

>> peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA --channelID $CHANNEL\_NAME --name fabcar $PEER\_CONN\_PARMS --version ${VERSION} --sequence ${VERSION} --init-required

1. Query chaincode commit as Org2

>> peer lifecycle chaincode querycommitted --channelID $CHANNEL\_NAME --name fabcar

1. Query chaincode commit as Org1

>> source ./scripts/setChannelContext.sh 1

>> peer lifecycle chaincode querycommitted --channelID $CHANNEL\_NAME --name fabcar

### Chaincode Invocation

1. This chaincode Fabcar, has to be initialized before executing other transactions

>> peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C $CHANNEL\_NAME -n fabcar $PEER\_CONN\_PARMS --isInit -c '{"function":"initLedger","Args":[]}'

1. Query Status of the state after init

>> peer chaincode query -C $CHANNEL\_NAME -n fabcar -c '{"Args":["queryAllCars"]}'

1. Change the ownership of CAR9 to dave:

>> peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls $CORE\_PEER\_TLS\_ENABLED --cafile $ORDERER\_CA -C $CHANNEL\_NAME -n fabcar $PEER\_CONN\_PARMS -c '{"function":"changeCarOwner","Args":["CAR9","Dave"]}'

1. As Org2 query to check status as ownership of CAR9

>> source ./scripts/setChannelContext.sh 2

>> peer chaincode query **-**C mychannel **-**n fabcar **-**c '{"Args":["queryCar","CAR9"]}'

1. You did complete your experiment: Bring down the network

>> ./network.sh down=${PWD}/configtx