Deploying A JS Smart Contract (HLF 2.2)

End users interact with the blockchain ledger by invoking smart contracts. In Hyperledger Fabric, smart contracts are deployed in packages referred to as chaincode. A chaincode is deployed to a channel using a process known as the Fabric chaincode lifecycle.

Steps:

1. **Start the network**
   1. First traverse to the Network

cd fabric-samples/test-network

* 1. Kill any active or stale docker containers and remove previously generated artifacts

./network.sh down

* 1. Start the network

./network.sh up createChannel

1. **Package the Smart Contract**

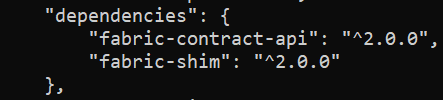
We need to package the chaincode before it can be installed on our peers. And before we package the chaincode, we need to install the chaincode dependences.

* 1. Navigate to the folder that contains the chaincode

cd fabric-samples/ct-chaincode-js

* 1. [Optional] View the dependencies

cat package.json



* 1. Install the smart contract dependencies

npm install

* 1. Check whether npm\_modules or node\_modules folder is created

ls

1. **Create the Chaincode Package**
   1. Navigate back to our working directory

cd ../test-network

First, create Chaincode package in the format specified by Peer CLI available in bin folder.

* 1. Add the binaries to your CLI path

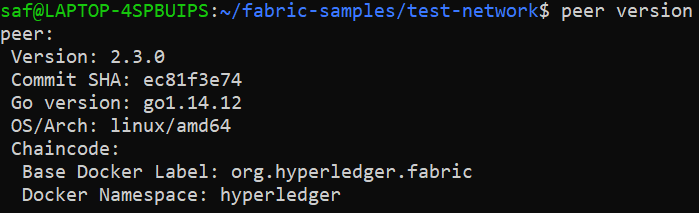
export PATH=${PWD}/../bin:$PATH

* 1. Also, set the FABRIC\_CFG\_PATH to point to the core.yaml file

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

* 1. To confirm, chech the version of binaries. (Should be 2.0.0 or Later)

peer version



* 1. Create the chaincode package using the “peer lifecycle chaincode package” command

peer lifecycle chaincode package ctjscc.tar.gz --path ../ct-chaincode-js/ --lang node --label basic\_1.0

* 1. Check your directory, if ctjscc.tar.gz file is created

ls

1. **Install the Chaincode Package on the Peers**

The chaincode needs to be installed on every peer that will endorse a transaction.

* 1. Let us install on the Org1 Peer first. Set following Env variables to operate the peer as Admin.

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.example.com/users/Admin@org1.example.com/msp

export CORE\_PEER\_ADDRESS=localhost:7051

* 1. Now install the chaincode on this peer

peer lifecycle chaincode install ctjscc.tar.gz

This command will return a Package ID which will be similar for all peers.

* 1. Set similar Env variables for the Org2 peer

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\_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.example.com/users/Admin@org2.example.com/msp

export CORE\_PEER\_ADDRESS=localhost:9051

* 1. Install the chaincode on this peer as well

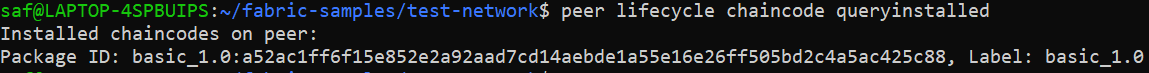
peer lifecycle chaincode install ctjscc.tar.gz

1. **Approve Chaincode Definition**

After installing the chaincode, approve the chaincode definition at each organization.

* 1. Query your Package Id

peer lifecycle chaincode queryinstalled



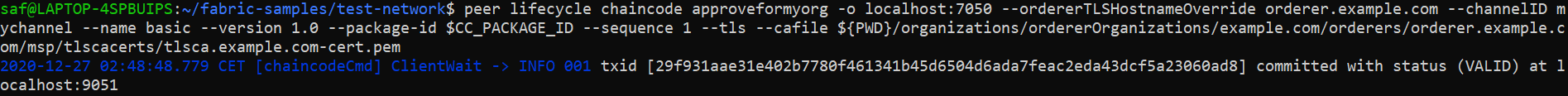
The package ID is the combination of the chaincode label and a hash of the chaincode binaries.

* 1. To use the package ID to approve the chaincode, save it as an Env variable.

export CC\_PACKAGE\_ID=basic\_1.0:a52ac1ff6f15e852e2a92aad7cd14aebde1a55e16e26ff505bd2c4a5ac425c88

* 1. Since we are on Org2 peer, let us approve the chaincode from this peer.

peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name basic --version 1.0 --package-id $CC\_PACKAGE\_ID --sequence 1 --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem



* 1. Now, in order to approve the chaincode as Org1 peer, set following Env variables

export CORE\_PEER\_LOCALMSPID="Org1MSP"

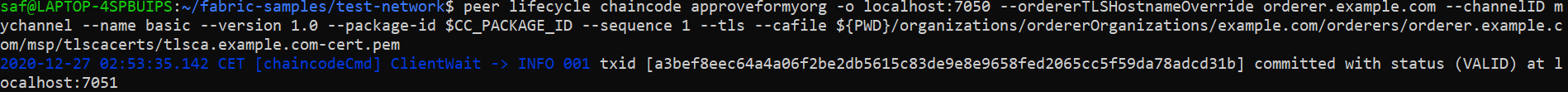
export CORE\_PEER\_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp

export CORE\_PEER\_TLS\_ROOTCERT\_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt

export CORE\_PEER\_ADDRESS=localhost:7051

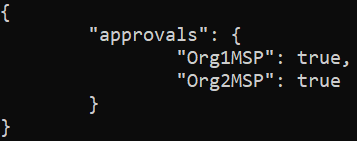
* 1. Approve as Org1 peer

peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name basic --version 1.0 --package-id $CC\_PACKAGE\_ID --sequence 1 --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem



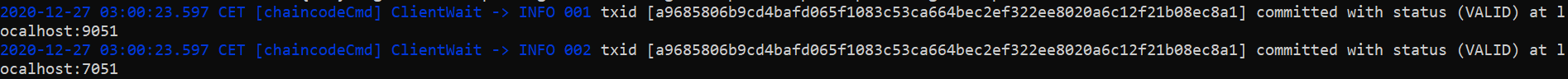
1. **Commit the Chaincode**
   1. Check whether the members have approved the Chaincode definition

peer lifecycle chaincode checkcommitreadiness --channelID mychannel --name basic --version 1.0 --sequence 1 --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --output json



* 1. Commit the Chaincode definition to the channel (Should be submitted by an Admin)

peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --channelID mychannel --name basic --version 1.0 --sequence 1 --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem --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



The chaincode definition endorsements by channel members are submitted to the ordering service to be added to a block and distributed to the channel. The peers on the channel then validate whether a sufficient number of organizations have approved the chaincode definition. Above command will wait for the validations from the peers before returning a response.

* 1. To confirm whether the chaincode definition has been committed to the channel

peer lifecycle chaincode querycommitted --channelID mychannel --name basic --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem



1. **Invoke the Chaincode**

The chaincode is now ready to be invoked by client applications.

* 1. Set initial state of assets on the Ledger

peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/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":[]}'



* 1. Query a function to get all Assets

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