1. **Install Samples, Binaries, and Docker Images**
   1. Create a new directory  
        
      > mkdir hlf
   2. Run the below command to download docker images  
        
      > curl **-**sSL https:**//**bit**.**ly**/**2ysbOFE **|** bash **-**s
   3. Since the Below chaincode we are using will be of JavaScript type the u should have node installed in your local machine. U can visit the given page to download node  
      https://nodejs.org/en/download/
2. **Bring up the test network**https://hyperledger-fabric.readthedocs.io/en/release-2.3/test\_network.html
   1. Navigate to the test network directory

> cd fabric-samples/test-network

* 1. Run the following command to see the help

> ./network.sh -h

* 1. Remove any containers or artefacts that are existing:

> ./network.sh down

* 1. Bring up the network. This command creates a Fabric network that consists of two peer nodes, one ordering node and CA for each node

> ./network.sh up -ca

* 1. Create a channel  
       
     > **./**network**.**sh createChannel -c mychannel
  2. Set some Environment parameters   
       
     > export PATH=${PWD}/../bin:$PATH  
     > export FABRIC\_CFG\_PATH=$PWD/../config/
  3. To confirm that you are able to use the peer CLI, check the version of the binaries. The binaries need to be version 2.0.0 or later to run this tutorial.  
       
     > peer version

1. **Packaging the Chaincode for Deployment**
   1. Open a New Terminal under Asset-Transfer-Basic/Chaincode-javascript  
        
      > cd fabric-samples/asset-transfer-basic/chaincode-javascript
   2. Install required dependencies for the javascript project specified in package.json file  
        
      > npm install
   3. Switch Back to the Terminal in which test network was opened and You can now create the chaincode package using the peer lifecycle chaincode package command  
        
      > peer lifecycle chaincode package basic.tar.gz --path ../asset-transfer-basic/chaincode-javascript/ --lang node --label basic\_1.0
2. **Installing Chaincode on peer0.org1.example.com and peer0.org2.example.com  
   (**<https://hyperledger-fabric.readthedocs.io/en/release-2.3/deploy_chaincode.html>**)**
   1. For peer0.org1 setting the environment variables   
      > 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
   2. Installing the Chaincode in peer0.org1  
        
      > peer lifecycle chaincode install basic.tar.gz  
        
      output :  
      > 2020-07-16 10:09:57.534 CDT [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remotely: response:<status:200 payload:"\nJbasic\_1.0:e2db7f693d4aa6156e652741d5606e9c5f0de9ebb88c5721cb8248c3aead8123\022\tbasic\_1.0" >  
      > 2020-07-16 10:09:57.534 CDT [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode code package identifier: basic\_1.0:e2db7f693d4aa6156e652741d5606e9c5f0de9ebb88c5721cb8248c3aead8123
   3. Checking Installed chaincode  
        
      > peer lifecycle chaincode queryinstalled  
        
        
        
        
        
      output:  
      > Installed chaincodes on peer:  
      Package ID: basic\_1.0:69de748301770f6ef64b42aa6bb6cb291df20aa39542c3ef94008615704007f3, Label: basic\_1.0
   4. Setting Environment Variable to point to this package(this value is unique so copy from your previous query)  
        
      > export CC\_PACKAGE\_ID=basic\_1.0:69de748301770f6ef64b42aa6bb6cb291df20aa39542c3ef94008615704007f3
   5. Approve the chaincode from peer0.Org1  
        
      > 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 ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem
   6. Following the Same Process for peer0.Org2 ,   
      For peer0.org1 setting the environment variables   
      > export CORE\_PEER\_TLS\_ENABLED=true  
      > export CORE\_PEER\_LOCALMSPID="Org2MSP"  
      > export CORE\_PEER\_TLS\_ROOTCERT\_FILE=${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt  
      > export CORE\_PEER\_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2.example.com/users/Admin@org2.example.com/msp  
      > export CORE\_PEER\_ADDRESS=localhost:9051
   7. Installing the Chaincode in peer0.org2  
        
      > peer lifecycle chaincode install basic.tar.gz  
        
      output :  
      > 2020-07-16 10:09:57.534 CDT [cli.lifecycle.chaincode] submitInstallProposal -> INFO 001 Installed remotely: response:<status:200 payload:"\nJbasic\_1.0:e2db7f693d4aa6156e652741d5606e9c5f0de9ebb88c5721cb8248c3aead8123\022\tbasic\_1.0" >  
      > 2020-07-16 10:09:57.534 CDT [cli.lifecycle.chaincode] submitInstallProposal -> INFO 002 Chaincode code package identifier: basic\_1.0:e2db7f693d4aa6156e652741d5606e9c5f0de9ebb88c5721cb8248c3aead8123
   8. Approve the chaincode from peer0.Org1, we need not set the package id because we already set it as environment variable during approving peer0.org1  
        
      > 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
   9. Now u can check the status of approvals of the installed chaincode by running   
        
      > 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  
        
      output:   
       > {  
       "Approvals": {  
       "Org1MSP": true,  
       "Org2MSP": true  
       }  
       }
3. **Committing the chaincode to the fabric test network**
   1. Once u have the approval of all the endorsing members (all in our present case) we can commit the chaincode to the network  
        
      > 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
   2. Commiting code to the Network  
        
      > 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

* 1. U can query the chaincode too see the approval status and state  
       
     > 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  
       
       
       
     output:  
     > Committed chaincode definition for chaincode 'basic' on channel 'mychannel': Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2MSP: true]

1. **Involving the Committed Chaincode :**
   1. U can communicate with the chaincode either with cli or by writing a dedicated application. Here we are invoking with the cli and once successful invoke following output will be shown(status 200) and on any error status 500 will be thrown(in the chaincode code a init function is declared to initialse the chaincode with dummy data)  
        
      > 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":[]}'  
        
      output:  
      > 2020-02-12 18:22:20.576 EST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
   2. And to query the updated status u can query the chaincode  
        
      > peer chaincode query -C mychannel -n basic -c '{"Args":["GetAllAssets"]}'  
        
      output:  
      > [{"Key":"asset1","Record":{"ID":"asset1","color":"blue","size":5,"owner":"Tomoko","appraisedValue":300}},{"Key":"asset2","Record":{"ID":"asset2","color":"red","size":5,"owner":"Brad","appraisedValue":400}},{"Key":"asset3","Record":{"ID":"asset3","color":"green","size":10,"owner":"JinSoo","appraisedValue":500}},{"Key":"asset4","Record":{"ID":"asset4","color":"yellow","size":10,"owner":"Max","appraisedValue":600}},{"Key":"asset5","Record":{"ID":"asset5","color":"black","size":15,"owner":"Adriana","appraisedValue":700}},{"Key":"asset6","Record":{"ID":"asset6","color":"white","size":15,"owner":"Michel","appraisedValue":800}}]
   3. To Update the present Assets u can invoke the chaincode again and query (there is function CreateAsset is initialized in the code)  
        
        
      > 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":"CreateAsset","Args":["asset8","blue","16","Kelley","750"]}'  
      > peer chaincode query -C mychannel -n basic -c '{"Args":["GetAllAssets"]}'  
        
      output(u can see new asset8 created at the end):  
      > [{"Key":"asset1","Record":{"ID":"asset1","color":"blue","size":5,"owner":"Tomoko","appraisedValue":300}},{"Key":"asset2","Record":{"ID":"asset2","color":"red","size":5,"owner":"Brad","appraisedValue":400}},{"Key":"asset3","Record":{"ID":"asset3","color":"green","size":10,"owner":"JinSoo","appraisedValue":500}},{"Key":"asset4","Record":{"ID":"asset4","color":"yellow","size":10,"owner":"Max","appraisedValue":600}},{"Key":"asset5","Record":{"ID":"asset5","color":"black","size":15,"owner":"Adriana","appraisedValue":700}},{"Key":"asset6","Record":{"ID":"asset6","color":"white","size":15,"owner":"Michel","appraisedValue":800}},{"Key":"asset8","Record":{"ID":"asset8","color":"blue","size":16,"owner":"Kelley","appraisedValue":750}}]
2. **Running Application to communicate with the chaincode(the functions are predefined in the application u can modify as u like or u can observe the output and see how they behave)**<https://hyperledger-fabric.readthedocs.io/en/release-2.3/write_first_app.html>
   1. There are some prerequisites applications present to be able to run this application. Open your powershell and run the below command as administrator   
        
      > npm install --global windows-build-tools
   2. Open a new terminal on asset-transfer-basic/application-javascript and run the below command to bring all dependencies  
        
      > npm install
   3. Once this is successful without any errors uu can run the below command to start the application  
        
      > node app.js
3. **Configuration Info:**<https://hyperledger-fabric.readthedocs.io/en/release-2.3/channel_update_tutorial.html>  
   https://hyperledger-fabric.readthedocs.io/en/release-2.3/config\_update.html
   1. The Configuration file for the topology of the network can be found under configtx folder (configtx.yaml) so any additional peers or organizations can be added from here and her we specify the required certificates.  
      we specify the role of the peer in this file whether he is a admin or just a observer or endorsing peer etc.
   2. Under Organization we can find 2 folders peerOrganization and orderOrganozation which contains all the connection configuration and keys and certificates of the respective peer.