**Annexure 1**

**Hyperledger Fabric Test Network Implementation Documentation**

# **INDEX**

[**INDEX 2**](#_43dsmuyhp4o2)

[**INTRODUCTION 5**](#_3amyw0mlir43)

[Update & Upgrade Ubuntu 6](#_3moq85942o7c)

[Installing git, curl & docker-compose 6](#_qbjq5br2q9u8)

[Start & Enable Docker 6](#_b75ka6uh3o8j)

[● Verify the installation 6](#_ahzfvlqw76to)

[● Installing jq: 6](#_d260972n83qd)

[● Installing python 6](#_eg15cnm3va01)

[● Install Go 7](#_gbzvmqarsfpo)

[Hyperledger fabric Installation 7](#_strcszpnitrh)

[● Configure Git\* 7](#_syivcmm3vyf8)

[● Clone the repository. 7](#_dlbbwiqpfyhr)

[Hyperledger fabric Test Network Implementation 9](#_sdc14290n7yn)

[● Navigate to the test-network directory 9](#_47zwpf7pvpsu)

[● Starting your Test-Network 9](#_mmtnmkglum3n)

[● Check Docker Containers 9](#_lc8qyd6eyndr)

[● Channels 10](#_5zsqzunl8op9)

[Chain-Code Deployment 11](#_iklx1mly14t7)

[Interacting with the network 11](#_qzucubja0vaq)

[Organization 1 11](#_bgwj7sbny6jd)

[Setting up environment variables (global variables) 11](#_n13we6hipj3s)

[Initialize the ledger with assets 11](#_d2mgf8guxdcr)

[List the Assets 12](#_4vhv3dnkabrs)

[Invoking the Chain-codes 13](#_gdsdws1dft2z)

[Organization 2 14](#_8ld2v9h1hz9p)

[Setting up environment variables (global variables) 14](#_hux56fegwkx0)

[Querying the Ledger 14](#_dv9flxaov0mn)

[Bringing Down the Network 14](#_ttyqa3ucq7iq)

[**Adding Org3 to the existing network 15**](#_xora4i226hu2)

[Starting Network 15](#_7egcbd5aj67)

[Starting the network and creating certificates for the nodes, then creating a channel named channel1 15](#_ffwoue3u0cmk)

[Generate the Org3 Crypto Material 15](#_blkn5zy72jjw)

[Deploying Chain code 15](#_t8a4x34h9l2j)

[Invoking and Query Chain Code 16](#_lmn7w9bulp4t)

[Invoke 16](#_ew3ufqynq74a)

[Query 16](#_5fz2yz22yw7)

[Org3 Setup 16](#_3zqmqlza5aw7)

[AddOrg3 directory 16](#_l8h7zxoyf97z)

[Generate ca certificates for Org3 16](#_5sb1567sxbv1)

[Generate org config 16](#_9xvdxi4q71ka)

[Starting docker container 17](#_zpmj3skbceo)

[setup ENV for Org3 17](#_em0mrksfgwgj)

[Fetch config 17](#_d2udg7hu96al)

[Decoding config block and trimming it 17](#_s29lkqb2161q)

[Adding config 18](#_k7pxmy5cap8s)

[Translate config.json back into a protobuf called config.pb 18](#_1csjo8lhwdg6)

[Encode modified\_config.json to modified\_config.pb 18](#_bl1o10qaq7ix)

[Calculate the delta between these two config protobufs 18](#_c7525ts4i5xt)

[Decode this object into editable JSON format and call it org3\_update.json 18](#_g8z1539xfii7)

[Adding headers back 19](#_eah8mnaqg2sd)

[convert it into the fully fledged protobuf format 19](#_gd3qy4ilobby)

[Sign the tx 19](#_we84jaakc6dk)

[Sign from Org1 19](#_x5ipx5fv7xh2)

[Env for org2 19](#_4rrdf6jbduq9)

[Sign from Org2 19](#_mr5jvxw2u6b7)

[Join Org3 to the Channel 20](#_3o6p1uuwvcje)

[Env 20](#_1rg0ki386yvu)

[Fetch block 0 20](#_8dnlckg8vaju)

[Join channel 20](#_b8t1cbpy5f69)

[Chaincode setup 20](#_8zx9ko4z0ivs)

[Install CC 20](#_d5xtz61gn48r)

[Approve CC 21](#_w5zatg1yeoyt)

[Query Commited 21](#_hbf8ublk6cdf)

[Invoke CC 21](#_mnk3xk8sqkvd)

[Query CC 21](#_6qt8cfc9ni56)

[**Adding a peer to the existing network 21**](#_y4usr1k3e6qe)

[Add the necessary files 22](#_6542ijsj9kbn)

[Add the Registerpeer1.sh file (Appendix 1.1) in the following directory. 22](#_f8hlt1s7z7jx)

[Add the docker-compose-peer1org1.sh file (Appendix 1.2) in the following directory. 22](#_wf7yw0t287m2)

[Setting up the new peer 22](#_3dwpkoli6gni)

[Creating MSP Identities for peer1.org1.example.com 22](#_btgjg3o8fj4z)

[Starting up the peer container 22](#_m6zzl9x2appa)

[Joining existing channel 22](#_sb8mbdnzvhji)

[Query channel on peer0.org1.example.com 22](#_fblhk77jz92s)

[Query channel on peer1.org1.example.com 23](#_8qet1n9x3rvk)

[Join channel 23](#_mnw4iwl19lvj)

[Query channel on peer1.org1.example.com 23](#_92avepevo20n)

[Chaincode Setup 23](#_un66qjdt09he)

[Install CC 23](#_to3cxvexfir7)

[Query installed CC 23](#_998uvf20gc8o)

[Invoke CC 23](#_3tgjlcb402t2)

[Query CC 24](#_j7vp41x81q6a)

[**Appendices 24**](#_lwg5kik2jln8)

[1.1 Registerpeer1.sh 24](#_ebhiqdri5hyc)

[1.2 docker-compose-peer1org1.yaml 25](#_d6w3coip1wx9)

# 

# **INTRODUCTION**

*This is a comprehensive documentation on setting up a test Hyperledger Fabric network on your device. Hyperledger Fabric stands as a robust and versatile blockchain framework, and this guide aims to provide you with a clear and step-by-step process to establish a functional test network. Whether you are a developer, system administrator, or blockchain enthusiast, this documentation is designed to assist you in navigating through the essential steps involved in configuring and deploying a Hyperledger Fabric network locally.*

*From installing the necessary prerequisites to initializing the network, creating channels, and deploying smart contracts, each section is crafted to provide detailed insights and instructions. By the end of this guide, you will have a fully operational Hyperledger Fabric network running on your device, offering you hands-on experience with this powerful blockchain technology.*

1. **References**

* <https://hyperledger-fabric.readthedocs.io/en/latest/test_network.html>
* Github repo link

### 

### Update & Upgrade Ubuntu

| sudo apt update  sudo apt ugrade |
| --- |

### Installing git, curl & docker-compose

| sudo apt-get install git curl docker-compose -y |
| --- |

### Start & Enable Docker

| sudo systemctl start docker sudo usermod -a -G docker sarthak docker --version  docker-compose --version  sudo systemctl enable docker |
| --- |

Note: The test network has been successfully verified with Docker Desktop **version 2.5.0.1** and is the recommended version at this time. Higher versions may not work.

### Verify the installation

| curl --version  git —version |
| --- |

### Installing jq:

* Install the latest version of jq

| sudo apt-get install -y jq |
| --- |

Verify the installation and check the version of jq using the below command.

| jq --version |
| --- |

### Installing python

| sudo apt-get install python3 |
| --- |

Verify the installation and check the version of Git using the below command.

| python3 --version |
| --- |

### Install Go

* Install the latest version of go
* # https://go.dev/doc/install

| cd ~ wget https://go.dev/dl/go1.21.1.linux-amd64.tar.gz  sudo rm -rf /usr/local/go && tar -C ~ -xzf go1.21.1.linux-amd64.tar.gz |
| --- |

# Add ~/go/bin to the PATH environment variable.

# You can do this by adding the following line to your $HOME**/.profile** or /etc/profile (for a system-wide installation):

| export PATH  =$PATH:~/go/bin |
| --- |

# Note: Changes made to a profile file may not apply until the next time you log into your computer. To apply the changes immediately, just run the shell commands directly or execute them from the profile using a command such as source $HOME/.profile.

Verify the installation and check the version of Git using the below command.

| go version |
| --- |

## 

## Hyperledger fabric Installation

### Configure Git\*

if you do decide to install Git on Windows and manage the Fabric repositories natively (as opposed to within WSL2 and its Git installation), then make sure you configure Git as follows:

| git config --global core.autocrlf false  git config --global core.longpaths true |
| --- |

### Clone the repository.

| curl -sSLO https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-fabric.sh && chmod +x install-fabric.sh  ./install-fabric.sh -h  ./install-fabric.sh docker samples binary |
| --- |

**NOTE:**

1) You may encounter the below issue when you run the above command.

| failed to get default registry endpoint from daemon (Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: |
| --- |

To fix this you need to run the command below.

| sudo chmod 666 /var/run/docker.sock |
| --- |

**NOTE:** 1) You may receive the following error while installing the fabrics

| Cannot connect to the Docker daemon at unix:///var/run/docker.sock. Is the docker daemon running? |
| --- |

To fix the error, install the latest version of Docker. Then clear the docker container and install again.

| docker rm -f $(docker ps -aq)  docker rmi -f $(docker images -q)  docker network prune |
| --- |

## 

## Hyperledger fabric Test Network Implementation

### Navigate to the test-network directory

| cd fabric-samples/test-network |
| --- |

### Starting your Test-Network

| ./network.sh -h // to print the script help text ./network.sh down ./network.sh up |
| --- |

If any error occurs regarding that the docker images are already present, Clear any existing docker images

| ./network.sh down docker rm -f $(docker ps -aq)  docker rmi -f $(docker images -q)  docker network prune |
| --- |

This will re-set up the docker container and delete all the pre-existing images or networks in the container.

After you do this, try starting your network again.

For any more errors refer to [Troubleshooting](https://hyperledger-fabric.readthedocs.io/en/latest/test_network.html#troubleshooting)

### Check Docker Containers

| docker ps -a |
| --- |

| output:  peer0.org2.example.com  peer0.org1.example.com  orderer.example.com |
| --- |

Note:

# Add ~/fabric-samples/bin to the PATH environment variable.

# You also need to set the FABRIC\_CFG\_PATH to point to the core.yaml file in the fabric-samples repository.

# You can do this by adding the following lines to your $HOME/.profile or /etc/profile (for a system-wide installation):

| export PATH=$PATH:~/fabric-samples/bin  export FABRIC\_CFG\_PATH=~/fabric-samples/config/ |
| --- |

# Note: Changes made to a profile file may not apply until the next time you log into your computer. To apply the changes immediately, just run the shell commands directly or execute them from the profile using a command such as source $HOME/.profile.

### Channels

* Creating new channels

| ./network.sh createChannel  ./network.sh createChannel -c channel1  ./network.sh createChannel -c channel2 |
| --- |

## 

## 

## 

## 

## 

## 

## 

## 

## 

## Chain-Code Deployment

* Deploy the chaincode package containing the smart contract by calling the ./network.sh script with the chaincode name and language options.

| ./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl go |
| --- |

This script uses the chaincode lifecycle to package, install, query installed chaincode, approve chaincode for both Org1 and Org2, and finally commit the chaincode.

## Interacting with the network

| PWD=~/sarthak/fabric-samples/test-network |
| --- |

### Organization 1

#### Setting up environment variables (global variables)

Execute on console.

Set the environment variables that allow you to operate the peer CLI as Org1:

| 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 |
| --- |

#### Initialize the ledger with assets

Run the following command to initialize the ledger with assets. (Note the CLI does not access the Fabric Gateway peer, so each endorsing peer must be specified.)

:

| 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:  2023-09-19 05:15:04.640 UTC 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200 |
| --- |

#### List the Assets

You can now query the ledger from your CLI. Run the following command to get the list of assets that were added to your channel ledger:

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

| output (formatted using jsonformatter.com):  [  {  "AppraisedValue": 300,  "Color": "blue",  "ID": "asset1",  "Owner": "Tomoko",  "Size": 5  },  {  "AppraisedValue": 400,  "Color": "red",  "ID": "asset2",  "Owner": "Brad",  "Size": 5  },  {  "AppraisedValue": 500,  "Color": "green",  "ID": "asset3",  "Owner": "Jin Soo",  "Size": 10  },  {  "AppraisedValue": 600,  "Color": "yellow",  "ID": "asset4",  "Owner": "Max",  "Size": 10  },  {  "AppraisedValue": 700,  "Color": "black",  "ID": "asset5",  "Owner": "Adriana",  "Size": 15  },  {  "AppraisedValue": 800,  "Color": "white",  "ID": "asset6",  "Owner": "Michel",  "Size": 15  }  ] |
| --- |

#### Invoking the Chain-codes

Chain Codes are invoked when a network member wants to transfer or change an asset on the ledger. Use the following command to change the owner of an asset on the ledger by invoking the asset-transfer (basic) chaincode:

| 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":"TransferAsset","Args":["asset6","Christopher"]}' |
| --- |

This command changes the owner of an asset on the ledger by invoking the asset-transfer (basic) chaincode

| output:  2023-09-19 05:24:01.738 UTC 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200 payload:"Michel" |
| --- |

As the change should be visible to each node of each organization, let’s check the ledger of node1 of organization 2.

### Organization 2

#### Setting up environment variables (global 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 |
| --- |

As the ledger is already initialized on the network, let’s just query the ledger for asset info.

#### Querying the Ledger

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

| output:  {"AppraisedValue":800,"Color":"white","ID":"asset6","Owner":"Christopher","Size":15} |
| --- |

We can see that the Owner of asset6 changed from Michel to Christopher

## Bringing Down the Network

| ./network.sh down |
| --- |

# 

# **Adding Org3 to the existing network**

## Starting Network

### Starting the network and creating certificates for the nodes, then creating a channel named channel1

| ./network.sh up createChannel -c channel1 |
| --- |

## Generate the Org3 Crypto Material

| cd addOrg3  ../../bin/cryptogen generate --config=org3-crypto.yaml --output="../organizations" |
| --- |

### Deploying Chain code

| ./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-go -ccl go  export PATH=${PWD}/../bin:$PATH  export FABRIC\_CFG\_PATH=${PWD}/../config  export CORE\_PEER\_TLS\_ENABLED=true  export ORDERER\_CA=${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem  export CORE\_PEER\_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp  export CORE\_PEER\_ADDRESS=localhost:7051  export CORE\_PEER\_TLS\_ROOTCERT\_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt  export CORE\_PEER\_LOCALMSPID=Org1MSP |
| --- |

## Invoking and Query Chain Code

### Invoke

| 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 '{"Args":["CreateAsset","501","red","20","aditya","100"]}' |
| --- |

### Query

| peer chaincode query -C mychannel -n basic -c '{"Args":["ReadAsset","501"]}' |
| --- |

## Org3 Setup

### AddOrg3 directory

| cd addOrg3/ |
| --- |

### Generate ca certificates for Org3

| ./addOrg3.sh generate -ca |
| --- |

### Generate org config

| export FABRIC\_CFG\_PATH=$PWD  ../../bin/configtxgen -printOrg Org3MSP > ../organizations/peerOrganizations/org3.example.com/org3.json |
| --- |

It converts the config.yaml to a json file which is stored in the directory mentioned.

### Starting docker container

| docker-compose -f compose/docker/docker-compose-org3.yaml up -d |
| --- |

### setup ENV for Org3

| cd ..  export PATH=${PWD}/../bin:$PATH  export FABRIC\_CFG\_PATH=${PWD}/../config/  export CORE\_PEER\_TLS\_ENABLED=true  export CORE\_PEER\_LOCALMSPID="Org1MSP"  export CORE\_PEER\_TLS\_ROOTCERT\_FILE=${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 |
| --- |

### Fetch config

| peer channel fetch config channel-artifacts/config\_block.pb -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c mychannel --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem |
| --- |

We fetch the config block and storing is as an output in channel-artifacts/config\_block.pb.  
We also specify the channel name, orderer, and tls certs.

### Decoding config block and trimming it

| configtxlator proto\_decode --input ./channel-artifacts/config\_block.pb --type common.Block --output ./channel-artifacts/config\_block.json  jq ".data.data[0].payload.data.config" ./channel-artifacts/config\_block.json > ./channel-artifacts/config.json |
| --- |

We decode the config\_block.pb file into json format and store it in the location channel-artifacts/config\_block.json

This file contains the configuration for the whole block

A file name config.json is also created containing just the config part of the config\_block.json

Now we will add the Org3.json file to the config file so that we have the configuration of Org3 in config\_block.json

### Adding config

| jq -s '.[0] \* {"channel\_group":{"groups":{"Application":{"groups": {"Org3MSP":.[1]}}}}}' ./channel-artifacts/config.json ./organizations/peerOrganizations/org3.example.com/org3.json > ./channel-artifacts/modified\_config.json |
| --- |

This command creates a modified config.json file named modified\_config.json in the channel-artifacts directory which contains the configuration of Org3 as well.

### Translate config.json back into a protobuf called config.pb

| configtxlator proto\_encode --input ./channel-artifacts/config.json --type common.Config --output ./channel-artifacts/config.pb |
| --- |

We encode the config.json file to protobuf format to calculate the delta.

### Encode modified\_config.json to modified\_config.pb

| configtxlator proto\_encode --input ./channel-artifacts/modified\_config.json --type common.Config --output ./channel-artifacts/modified\_config.pb |
| --- |

We encode the modified\_config.json file to protobuf format to calculate the delta.

### Calculate the delta between these two config protobufs

In the context of the configtxlator compute\_update command in Hyperledger Fabric, the term "delta" refers to the difference or change between two channel configurations. It represents the set of modifications made to the original configuration to transform it into the updated configuration.

| configtxlator compute\_update --channel\_id mychannel --original ./channel-artifacts/config.pb --updated ./channel-artifacts/modified\_config.pb --output ./channel-artifacts/org3\_update.pb |
| --- |

Org3\_update.pb is created which contains the difference in configuration of two files namely –original file i.e config.pb and the updated file modified\_config.pb

### Decode this object into editable JSON format and call it org3\_update.json

| configtxlator proto\_decode --input ./channel-artifacts/org3\_update.pb --type common.ConfigUpdate --output ./channel-artifacts/org3\_update.json |
| --- |

Converting org3\_update.pb to org3\_update.json

### Adding headers back

Now as we initially trimmed some part of the config-block file to add the org3, we would have to add the headers back to the org3.json file to submit the transaction.

| echo '{"payload":{"header":{"channel\_header":{"channel\_id":"'mychannel'", "type":2}},"data":{"config\_update":'$(cat ./channel-artifacts/org3\_update.json)'}}}' | jq . > ./channel-artifacts/org3\_update\_in\_envelope.json |
| --- |

### convert it into the fully fledged protobuf format

| configtxlator proto\_encode --input ./channel-artifacts/org3\_update\_in\_envelope.json --type common.Envelope --output ./channel-artifacts/org3\_update\_in\_envelope.pb |
| --- |

## Sign the tx

### Sign from Org1

| peer channel signconfigtx -f channel-artifacts/org3\_update\_in\_envelope.pb |
| --- |

Signing the transaction by org1, peer0

### Env for org2

| 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 |
| --- |

### Sign from Org2

| peer channel update -f channel-artifacts/org3\_update\_in\_envelope.pb -c mychannel -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 |
| --- |

Signing and updating the channel with the latest channel configuration

## Join Org3 to the Channel

### Env

| export CORE\_PEER\_TLS\_ENABLED=true  export CORE\_PEER\_LOCALMSPID="Org3MSP"  export CORE\_PEER\_TLS\_ROOTCERT\_FILE=${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt  export CORE\_PEER\_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp  export CORE\_PEER\_ADDRESS=localhost:11051 |
| --- |

### Fetch block 0

| peer channel fetch 0 channel-artifacts/mychannel.block -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c mychannel --tls --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem |
| --- |

### Join channel

| peer channel list //will give the list of channels connected to thispeer  peer channel join -b channel-artifacts/mychannel.block  peer channel getinfo -c mychannel |
| --- |

## Chaincode setup

### Install CC

| peer lifecycle chaincode install basic.tar.gz  peer lifecycle chaincode queryinstalled |
| --- |

### Approve CC

| export CC\_PACKAGE\_ID=  peer lifecycle chaincode approveformyorg -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 --channelID mychannel --name basic --version 1.0 --package-id $CC\_PACKAGE\_ID --sequence 1 |
| --- |

### Query Commited

| 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 |
| --- |

### Invoke CC

| 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:9051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt --peerAddresses localhost:11051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org3.example.com/peers/peer0.org3.example.com/tls/ca.crt -c '{"function":"InitLedger","Args":[]}' |
| --- |

### Query CC

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

# **Adding a peer to the existing network**

## Add the necessary files

### Add the Registerpeer1.sh file ([Appendix 1.1](#_ebhiqdri5hyc)) in the following directory.

| ./test-network/organizations/fabric-ca/ |
| --- |

### Add the docker-compose-peer1org1.sh file ([Appendix 1.2](#_d6w3coip1wx9)) in the following directory.

| ./test-network/compose/docker/ |
| --- |

## 

## Setting up the new peer

### Creating MSP Identities for peer1.org1.example.com

| ./organizations/fabric-ca/registerPeer1.sh |
| --- |

### Starting up the peer container

| docker-compose -f compose/docker/docker-compose-peer1.yaml up -d |
| --- |

## Joining existing channel

### Query channel on peer0.org1.example.com

| peer channel list  peer channel fetch -c mychannel newest  CORE\_PEER\_ADDRESS=localhost:8051 peer channel getinfo -c mychannel |
| --- |

### Query channel on peer1.org1.example.com

| CORE\_PEER\_ADDRESS=localhost:8051 peer channel list  CORE\_PEER\_ADDRESS=localhost:8051 peer channel fetch -c mychannel newest  CORE\_PEER\_ADDRESS=localhost:8051 peer channel getinfo -c mychannel |
| --- |

### Join channel

| CORE\_PEER\_ADDRESS=localhost:8051 peer channel join -b ./channel-artifacts/mychannel.block |
| --- |

### Query channel on peer1.org1.example.com

| CORE\_PEER\_ADDRESS=localhost:8051 peer channel list  CORE\_PEER\_ADDRESS=localhost:8051 peer channel fetch -c mychannel newest  CORE\_PEER\_ADDRESS=localhost:8051 peer channel getinfo -c mychannel |
| --- |

## Chaincode Setup

### Install CC

| export CC\_NAME=basic  CORE\_PEER\_ADDRESS=localhost:8051 peer lifecycle chaincode install ${CC\_NAME}.tar.gz |
| --- |

### Query installed CC

| CORE\_PEER\_ADDRESS=localhost:8051 peer lifecycle chaincode queryinstalled |
| --- |

### Invoke CC

| CORE\_PEER\_ADDRESS=localhost:8051 peer chaincode invoke -n basic -C mychannel -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "$ORDERER\_CA" --peerAddresses localhost:8051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer1.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 '{"Args":["CreateAsset", "200","red", "20","aditya","100"]}' |
| --- |

### Query CC

| CORE\_PEER\_ADDRESS=localhost:8051 peer chaincode query -n basic -C mychannel -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com -c '{"Args":["ReadAsset", "200"]}' |
| --- |

### 

# **Appendices**

## 1.1 Registerpeer1.sh

| export PATH=$PATH:${PWD}/../bin  export FABRIC\_CA\_CLIENT\_HOME=${PWD}/organizations/peerOrganizations/org1.example.com/  fabric-ca-client register --caname ca-org1 --id.name peer1 --id.secret peer1pw --id.type peer --tls.certfiles ${PWD}/organizations/fabric-ca/org1/tls-cert.pem  mkdir -p organizations/peerOrganizations/org1.example.com/peers/peer1.org1.example.com  fabric-ca-client enroll -u https://peer1:peer1pw@localhost:7054 --caname ca-org1 -M ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/msp --csr.hosts peer1.org1.example.com --tls.certfiles ${PWD}/organizations/fabric-ca/org1/tls-cert.pem  cp ${PWD}/organizations/peerOrganizations/org1.example.com/msp/config.yaml ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/msp/config.yaml  fabric-ca-client enroll -u https://peer1:peer1pw@localhost:7054 --caname ca-org1 -M ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls --enrollment.profile tls --csr.hosts peer1.org1.example.com --csr.hosts localhost --tls.certfiles ${PWD}/organizations/fabric-ca/org1/tls-cert.pem  cp ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls/tlscacerts/\* ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls/ca.crt  cp ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls/signcerts/\* ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls/server.crt  cp ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls/keystore/\* ${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls/server.key |
| --- |

## 

## 1.2 docker-compose-peer1org1.yaml

| # Copyright IBM Corp. All Rights Reserved.  #  # SPDX-License-Identifier: Apache-2.0  #  version: "3.7"  volumes:  peer1.org1.example.com:  networks:  test:  name: fabric\_test  services:  peer1.org1.example.com:  container\_name: peer1.org1.example.com  image: hyperledger/fabric-peer:latest  labels:  service: hyperledger-fabric  environment:  #Generic peer variables  - CORE\_VM\_ENDPOINT=unix:///host/var/run/docker.sock  - CORE\_VM\_DOCKER\_HOSTCONFIG\_NETWORKMODE=fabric\_test  - FABRIC\_LOGGING\_SPEC=INFO  #- FABRIC\_LOGGING\_SPEC=DEBUG  - CORE\_PEER\_TLS\_ENABLED=true  - CORE\_PEER\_PROFILE\_ENABLED=true  - CORE\_PEER\_TLS\_CERT\_FILE=/etc/hyperledger/fabric/tls/server.crt  - CORE\_PEER\_TLS\_KEY\_FILE=/etc/hyperledger/fabric/tls/server.key  - CORE\_PEER\_TLS\_ROOTCERT\_FILE=/etc/hyperledger/fabric/tls/ca.crt  # Peer specific variabes  - CORE\_PEER\_ID=peer1.org1.example.com  - CORE\_PEER\_ADDRESS=peer1.org1.example.com:8051  - CORE\_PEER\_LISTENADDRESS=0.0.0.0:8051  - CORE\_PEER\_CHAINCODEADDRESS=peer1.org1.example.com:8052  - CORE\_PEER\_CHAINCODELISTENADDRESS=0.0.0.0:8052  - CORE\_PEER\_GOSSIP\_BOOTSTRAP=peer1.org1.example.com:8051  - CORE\_PEER\_GOSSIP\_EXTERNALENDPOINT=peer1.org1.example.com:8051  - CORE\_PEER\_LOCALMSPID=Org1MSP  - CORE\_OPERATIONS\_LISTENADDRESS=0.0.0.0:18051  volumes:  - /var/run/docker.sock:/host/var/run/docker.sock  - ../organizations/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/msp:/etc/hyperledger/fabric/msp  - ../organizations/peerOrganizations/org1.example.com/peers/peer1.org1.example.com/tls:/etc/hyperledger/fabric/tls  - peer1.org1.example.com:/var/hyperledger/production  working\_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer  command: peer node start  ports:  - 8051:8051  - 18051:18051  networks:  - test |
| --- |

# 