Test Network Overview

In this guide, we will see how to set up the Hyperledger Fabric test network and how to run a fabric sample application in the network using the fabric binaries.

First , installing all the prerequisites are given.

HARDWARE PREREQUISITES

- Operating System: Ubuntu 20.04 or 22.04
- RAM: 8GB or higher
- Free disk space: 40 GB
- High-speed internet connectivity

SOFTWARE PREREQUISITES

The following prerequisites have to be installed for building Hyperledger Fabric application:

- cURL
- Docker
- Docker-Compose
- Node.js
- JQ
- VS-code

The fabric-samples can be downloaded using the following steps:

To get the install script:

```
curl -sSLO https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-fabric.sh && chmod +x install-fabric.sh
```

Run the script using following command

```
./install-fabric.sh -f '2.5.3' -c '1.5.6'
```

Copy the binaries to usr/local/bin

```
sudo cp fabric-samples/bin/* /usr/local/bin
```

- Starting the network - We will be using the network script here for starting a network quickly
  - Crypto material generation (CA/cryptogen)
  - Bringing up the components (2 peers and 1 orderer)
  - Generate the genesis block
  - Creating channel & joining the orderer
  - Joining the peers to the created channel
- Deploying a chaincode - The chaincode is deployed using the deployCC mode available on the test-network
  - Package the chaincode
  - Install the packaged chaincode to selected peers
  - Approve chaincode with chaincode definition
  - Commit the chaincode to the channel

These are the very high level steps involved while one sets up the network.

The test network is a sample network provided by Hyperledger fabric, It comes with a 2 org single peer consortium with a single orderer.

Test network Architecture

No of Orgs : 2

Org1 - 1 peer (peer0.org1.example.com)

Org2 - 1 peer (peer0.org2.example.com)

Ordering Service : etcdraft (Single Node - example.com)

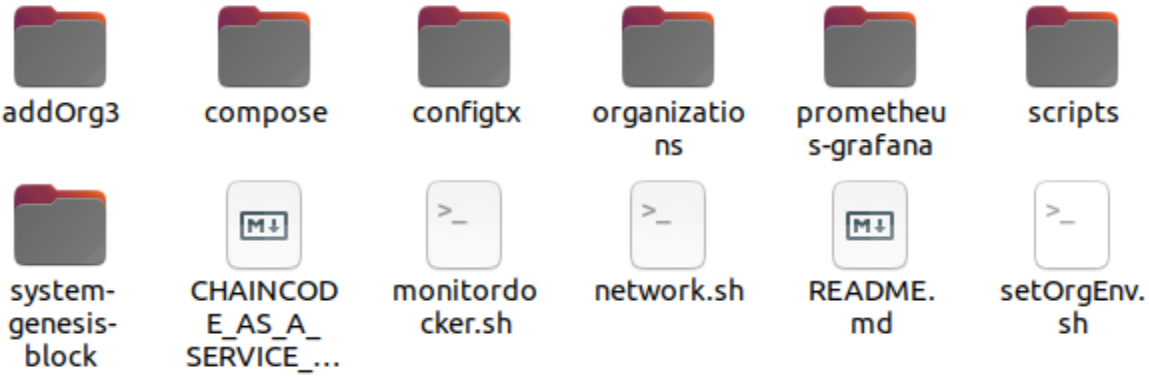
Database Type - CouchDB

Certificate Authority : Separate CA for org1, org2 and orderer.

fabric-samples : fabric samples consist of a collection of tested examples of Hyperledger fabric sample applications.

Test network provides a network.sh script which helps us to simplify the steps used to bootstrap the Hyperledger fabric network.

Before getting deep into the working of a test network, let's first familiarize ourselves with the folders inside the test network.



addOrg3 - It includes all the files related to adding a new organization to an already existing network

configtx - This folder includes the configtx.yaml file which has the detailed configuration relating the consortium. configtx helps to create the genesis block.

compose - This folder has the complete docker-compose files for bootstrapping the network.

organizations - This folder contains the configurations and folders related to crypto material generation for the organization.

prometheus-grafana -Prometheus is an open source monitoring system for which Grafana provides out-of-the-box support.Prometheus focuses on data acquisition, allowing users to select and aggregate time series data in real time. Grafana, on the other hand, focuses on data visualization.

scripts - The scripts folder consists of the complete scripts that the test-network uses to modify and control the test network

network.sh - This is the script we will be using frequently to control the network. It comes with a lot of handy modes like **up**, **up createChannel**, **createChannel**, **deployCC** and **down.**

```
network.sh <Mode> [Flags]
  Modes:
    up - Bring up Fabric orderer and peer nodes. No channel is created
    up createChannel - Bring up fabric network with one channel
    createChannel - Create and join a channel after the network is created
    deployCC - Deploy a chaincode to a channel (defaults to asset-transfer-basic)
    down - Bring down the network
```

Some other flags and options are

```
    Flags:
    Used with network.sh up, network.sh createChannel:
    -ca <use CAs> -  Use Certificate Authorities to generate network crypto material
    -c <channel name> - Name of channel to create (defaults to "mychannel")
    -s <dbtype> - Peer state database to deploy: goleveldb (default) or couchdb
    -r <max retry> - CLI times out after certain number of attempts (defaults to 5)
    -d <delay> - CLI delays for a certain number of seconds (defaults to 3)
    -i <imagetag> - Docker image tag of Fabric to deploy (defaults to "latest")
    -cai <ca_imagetag> - Docker image tag of Fabric CA to deploy (defaults to "latest"
)
    -verbose - Verbose mode

    Used with network.sh deployCC
    -c <channel name> - Name of channel to deploy chaincode to
    -ccn <name> - Chaincode name.
    -ccl <language> - Programming language of the chaincode to deploy: go, java, javas
cript, typescript
    -ccv <version>  - Chaincode version. 1.0 (default), v2, version3.x, etc
    -ccs <sequence>  - Chaincode definition sequence. Must be an integer, 1 (default),
 2, 3, etc
    -ccp <path>  - File path to the chaincode.
    -ccep <policy>  - (Optional) Chaincode endorsement policy using signature policy s
yntax. The default policy requires an endorsement from Org1 and Org2
    -cccg <collection-config>  - (Optional) File path to private data collections conf
iguration file
    -cci <fcn name>  - (Optional) Name of chaincode initialization function. When a fu
nction is provided, the execution of init will be requested and the function will be i
nvoked.
```

To use the script navigate to test-network folder inside the fabric-samples folder,

```
cd fabric-samples/test-network/
```

To print the available modes and flags available in the script use the command

./network.sh -h

Let's try starting a fabric network using the network script,

./network.sh up createChannel

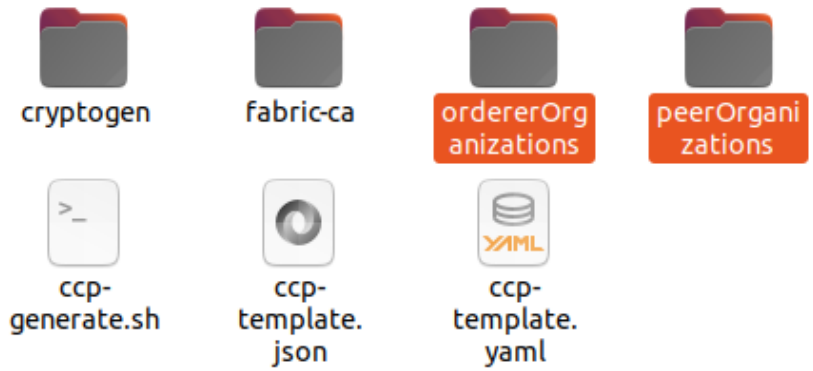
```
2023-11-16 07:22:49.436 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2023-11-16 07:22:49.439 UTC 0002 INFO [cli.common] readBlock -> Received block: 0
2023-11-16 07:22:49.439 UTC 0003 INFO [channelCmd] fetch -> Retrieving last config block: 0
2023-11-16 07:22:49.443 UTC 0004 INFO [cli.common] readBlock -> Received block: 0
Decoding config block to JSON and isolating config to Org2MSPconfig.json
+ configtxlator proto_decode --input config_block.pb --type common.Block --output config_block.json
+ jq '.data.data[0].payload.data.config' config_block.json
Generating anchor peer update transaction for Org2 on channel mychannel
+ jq '.channel_group.groups.Application.groups.Org2MSP.values += {"AnchorPeers":{"mod_policy": "Admins","value":{"anchor_peers": [{"host": "pe
er0.org2.example.com","port": 9051}]},"version": "0"}}' Org2MSPconfig.json
+ configtxlator proto_encode --input Org2MSPconfig.json --type common.Config --output original_config.pb
+ configtxlator proto_encode --input Org2MSPmodified_config.json --type common.Config --output modified_config.pb
+ configtxlator compute_update --channel_id mychannel --original original_config.pb --updated modified_config.pb --output config_update.pb
+ configtxlator proto_decode --input config_update.pb --type common.ConfigUpdate --output config_update.json
+ jq .
++ cat config_update.json
+ echo '{"payload":{"header":{"channel_header":{"channel_id":"mychannel", "type":2}},"data":{"config_update":{' '"channel_id":' '"mychannel",'
 '"isolated_data":' '{},' '"read_set":' '{' '"groups":' '{' '"Application":' '{' '"groups":' '{' '"Org2MSP":' '{' '"groups":' '{},' '"mod_poli
cy":' '"",' '"policies":' '{' '"Admins":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '},' '"Endorsement":' '{' '"mod_polic
y":' '"",' '"policy":' null, '"version":' '"0"' '},' '"Readers":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '},' '"Writer
s":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '}' '},' '"values":' '{' '"MSP":' '{' '"mod_policy":' '"",' '"value":' nul
l, '"version":' '"0"' '}' '},' '"version":' '"0"' '}' '},' '"mod_policy":' '"",' '"policies":' '{},' '"values":' '{},' '"version":' '"0"' '}'
'},' '"mod_policy":' '"",' '"policies":' '{},' '"values":' '{},' '"version":' '"0"' '},' '"write_set":' '{' '"groups":' '{' '"Application":' '
{' '"groups":' '{' '"Org2MSP":' '{' '"groups":' '{},' '"mod_policy":' '"Admins",' '"policies":' '{' '"Admins":' '{' '"mod_policy":' '"",' '"po
licy":' null, '"version":' '"0"' '},' '"Endorsement":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '},' '"Readers":' '{' '"
mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '},' '"Writers":' '{' '"mod_policy":' '"",' '"policy":' null, '"version":' '"0"' '}'
'},' '"values":' '{' '"AnchorPeers":' '{' '"mod_policy":' '"Admins",' '"value":' '{' '"anchor_peers":' '[' '{' '"host":' '"peer0.org2.example.
com",' '"port":' 9051 '}' ']' '},' '"version":' '"0"' '},' '"MSP":' '{' '"mod_policy":' '"",' '"value":' null, '"version":' '"0"' '}' '},' '"v
ersion":' '"1"' '}' '},' '"mod_policy":' '"",' '"policies":' '{},' '"values":' '{},' '"version":' '"0"' '}' '},' '"mod_policy":' '"",' '"polic
ies":' '{},' '"values":' '{},' '"version":' '"0"' '}' '}}}}'
+ configtxlator proto_encode --input config_update_in_envelope.json --type common.Envelope --output Org2MSPanchors.tx
2023-11-16 07:22:50.230 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2023-11-16 07:22:50.242 UTC 0002 INFO [channelCmd] update -> Successfully submitted channel update
Anchor peer set for org 'Org2MSP' on channel 'mychannel'
Channel 'mychannel' joined
```

The up mode in the network script will start the network using the configurations. We will be looking deep into the configurations at a later point of time.

After the execution of the script we can see some new files and folders created, let's see what those are.

channel-artifacts - mychannel.block will be added inside this folder

organizations - new folders having crypto materials is generated.



You can check the docker containers created using the command

docker ps -a

```
kba@lab:~/fabric-samples/test-network$ docker ps --format "table {{.Image}}\t{{.Ports}}\t
{{.Names}}"
IMAGE                               PORTS
                                                                                NAMES
hyperledger/fabric-tools:latest
                                                                                cli
hyperledger/fabric-peer:latest      0.0.0.0:9051->9051/tcp, :::9051->9051/tcp, 7051/tcp,
0.0.0.0:9445->9445/tcp, :::9445->9445/tcp                                       peer0.org2.e
xample.com
hyperledger/fabric-orderer:latest   0.0.0.0:7050->7050/tcp, :::7050->7050/tcp, 0.0.0.0:70
53->7053/tcp, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp   orderer.exam
ple.com
hyperledger/fabric-peer:latest      0.0.0.0:7051->7051/tcp, :::7051->7051/tcp, 0.0.0.0:94
44->9444/tcp, :::9444->9444/tcp                                                 peer0.org1.e
xample.com
```

Once the network is bootstrapped, we can now try deploying a chaincode. We will be trying to deploy the asset transfer-basic javascript chaincode.

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

To deploy the chaincode we will be using the deployCC mode available on the network script, which basically automates the chaincode installation and instantiation. We are passing the several flags such as

Chaincode name (ccn) as basic

Chaincode path (ccp) as basic javascript location

Chaincode language (ccl) as javascript

```
{
        "approvals": {
                "Org1MSP": true,
                "Org2MSP": true
        }
}
Checking the commit readiness of the chaincode definition successful on peer0.org2 on channel 'mychannel'
Using organization 1
Using organization 2
+ peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /home/kba/fabric-samples/t
est-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --channelID mychannel --name basic --peerAddresses
 localhost:7051 --tlsRootCertFiles /home/kba/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/tlsca/tlsca.org1.exa
mple.com-cert.pem --peerAddresses localhost:9051 --tlsRootCertFiles /home/kba/fabric-samples/test-network/organizations/peerOrganizations/org2
.example.com/tlsca/tlsca.org2.example.com-cert.pem --version 1.0 --sequence 1
+ res=0
2023-11-16 13:06:31.351 IST 0001 INFO [chaincodeCmd] ClientWait -> txid [ff0383a4d186f5e2e091d6999e7b9a050e975573b7c880eec3c0603d6e290e22] com
mitted with status (VALID) at localhost:9051
2023-11-16 13:06:31.505 IST 0002 INFO [chaincodeCmd] ClientWait -> txid [ff0383a4d186f5e2e091d6999e7b9a050e975573b7c880eec3c0603d6e290e22] com
mitted with status (VALID) at localhost:7051
Chaincode definition committed on channel 'mychannel'
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to Query committed status on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
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]
Query chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to Query committed status on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name basic
+ res=0
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]
Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Chaincode initialization is not required
kba@lab:~/fabric-samples/test-network$
```

Once the chaincode is deployed we can finally try interacting with the deployed chaincode.

To interact with the deployed chaincode we will need to set up the proper environment for the peer binary. Provide the following environments,

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

Once the environment is set now it's time to use the peer binary to invoke a function "initLedger" in the chaincode.

To invoke the chaincode use the following command,

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":[]}'

```
kba@lab:~/fabric-samples/test-network$ peer chaincode invoke -o localhost:7050 --ordererT
LSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrgani
zations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pe
m" -C mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organi
zations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --pee
rAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2
.example.com/peers/peer0.org2.example.com/tls/ca.crt" -c '{"function":"InitLedger","Args"
:[]}'
2023-07-12 11:08:10.180 IST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode
invoke successful. result: status:200
```

Now let's query the asset available, use the following command to query the complete assets.

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

```
kba@lab:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n basic -c '{"A
rgs":["GetAllAssets"]}'
[{"AppraisedValue":300,"Color":"blue","ID":"asset1","Owner":"Tomoko","Size":5,"docType":"
asset"},{"AppraisedValue":400,"Color":"red","ID":"asset2","Owner":"Brad","Size":5,"docTyp
e":"asset"},{"AppraisedValue":500,"Color":"green","ID":"asset3","Owner":"Jin Soo","Size":
10,"docType":"asset"},{"AppraisedValue":600,"Color":"yellow","ID":"asset4","Owner":"Max",
"Size":10,"docType":"asset"},{"AppraisedValue":700,"Color":"black","ID":"asset5","Owner":
"Adriana","Size":15,"docType":"asset"},{"AppraisedValue":800,"Color":"white","ID":"asset6
","Owner":"Michel","Size":15,"docType":"asset"}]
```

Read a single asset

peer chaincode query -C mychannel -n basic -c '{"function":"ReadAsset","Args":["asset5"]}'

```
kba@lab:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n basic -c '{"f
unction":"ReadAsset","Args":["asset5"]}'
{"AppraisedValue":700,"Color":"black","ID":"asset5","Owner":"Adriana","Size":15,"docType"
:"asset"}
```

Transfer a asset to a Particular Owner.

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":["asset5","Bob"]}'

```
kba@lab:~/fabric-samples/test-network$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.
 "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.co
n basic --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/p
m/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example
ple.com/tls/ca.crt" -c '{"function":"TransferAsset","Args":["asset5","Bob"]}'
2023-11-16 13:57:03.354 IST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result:
a"
kba@lab:~/fabric-samples/test-network$
```

Read updated asset

peer chaincode query -C mychannel -n basic -c '{"function":"ReadAsset","Args":["asset5"]}'

```
kba@lab:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n basic -c '{"function":"ReadAsset","Args":["asset5"]}'
{"AppraisedValue":700,"Color":"black","ID":"asset5","Owner":"Bob","Size":15,"docType":"asset"}
kba@lab:~/fabric-samples/test-network$
```

Bring down the network

When you are finished using the test network, you can bring down the network with the following command:

./network.sh down

```
kba@lab:~/fabric-samples/test-network$ ./network.sh down
Using docker and docker-compose
Stopping network
Stopping cli                    ... done
Stopping peer0.org2.example.com ... done
Stopping orderer.example.com    ... done
Stopping peer0.org1.example.com ... done
Removing cli                    ... done
Removing peer0.org2.example.com ... done
Removing orderer.example.com    ... done
Removing peer0.org1.example.com ... done
Removing network fabric_test
Removing network compose_default
WARNING: Network compose_default not found.
Removing volume compose_orderer.example.com
Removing volume compose_peer0.org1.example.com
Removing volume compose_peer0.org2.example.com
Removing volume compose_peer0.org3.example.com
WARNING: Volume compose_peer0.org3.example.com not found.
Error response from daemon: get docker_orderer.example.com: no such volume
Error response from daemon: get docker_peer0.org1.example.com: no such volume
Error response from daemon: get docker_peer0.org2.example.com: no such volume
Removing remaining containers
Removing generated chaincode docker images
Untagged: dev-peer0.org2.example.com-basic_1.0-5f82377ca15c97a6a71fa154084f1ced34801bc33a2976250904f8b98c9353ce-081e5bfb130264da8b9aff8b8f19c5
16e7e29a947a0ff8735005b80bd5b0e8fe:latest
Deleted: sha256:79bc21a2f33d5d5d66aa37cdffbdb4c224cf8f7c99d7f71ba819a2fb7948d86c
Deleted: sha256:c6341c9038d690ba62af0de79328dcf150b7492509b1c6898f4f5f15fa4c5972
Deleted: sha256:d1efad3c586c538e45c2aab5b38a2a18880e5583021daf68cc82ed73a6db4b6c
Deleted: sha256:674c91552507be80eaee3ec2e0b209d69757ea0f4605461f684c7a87806f217a
Untagged: dev-peer0.org1.example.com-basic_1.0-5f82377ca15c97a6a71fa154084f1ced34801bc33a2976250904f8b98c9353ce-ba4efd3eec35c3e7cf603a8dc0b2f8
bf5368211e0482d0e34d245c3c8066761d:latest
Deleted: sha256:b1c4a5f8aaf5267a17a1ef23ad6ef69bec1fba98b9ccc149d59b8d670e719306
Deleted: sha256:f616aa4f24b0c89fea9fcab6ecc9f9c127bfbb9ad099db3185237e95692235ac
Deleted: sha256:bc4e0f81230fa8d4f896d83fee78217525667c4055c1ef8787ded9cab60ae4bd
Deleted: sha256:a86f501cdf1fc904c7bcb4841225d692fa45c0e078f150b1029fe707ff116693
kba@lab:~/fabric-samples/test-network$
```

Clearing the orphaned containers

docker container prune
docker volume prune
docker system prune