Test Network Overview

In this guide, we will see how to set up the Hyperledger Fabric test network and how to interact with the network using the fabric binaries.

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

Download the script:

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

Execute the script:

```
./install-fabric.sh -f '2.5.12' -c '1.5.15'
```

It performs the following steps:

- Clones the hyperledger/fabric-samples repository.
- Downloads the latest/specified Hyperledger Fabric Docker images and tags them as latest
- Downloads the following platform-specific Hyperledger Fabric CLI tool binaries and config files into the fabric-samples /bin and /config directories.

Copy the binaries to usr/local/bin:

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

Before we move on to an advanced view of how the network is being set up, let's brief the steps involved in the process.

- Starting the network - We will be using the network script here to start a network quickly

    - Crypto material generation (CA/cryptogen)
    - Generate the genesis block
    - Bring up the components (2 peers and 1 orderer)
    - Create channel and join the orderer
    - Join 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)

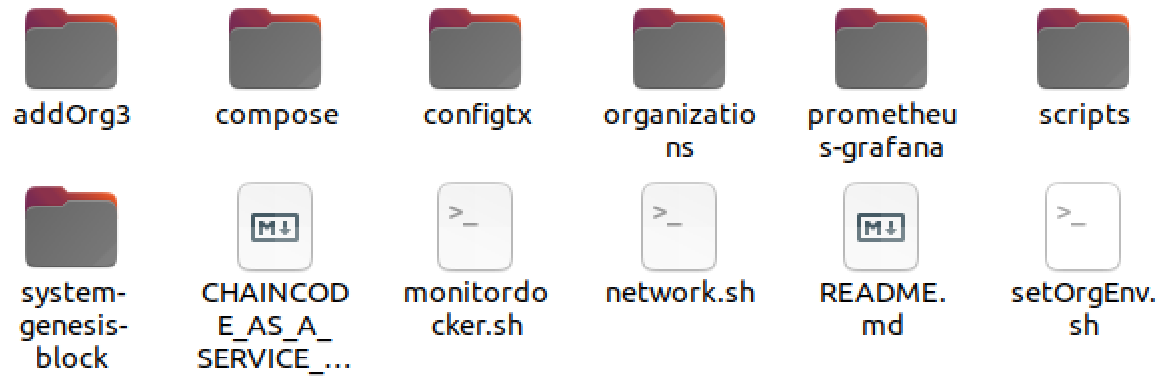
Ordering Service : Raft (Single Node - orderer.example.com)

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

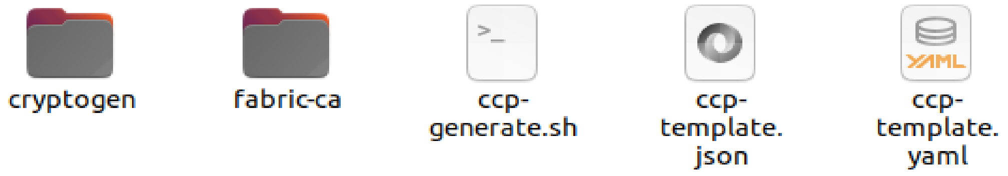
fabric-samples : fabric-samples consist of a collection of tested examples of Hyperledger Fabric sample applications.

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

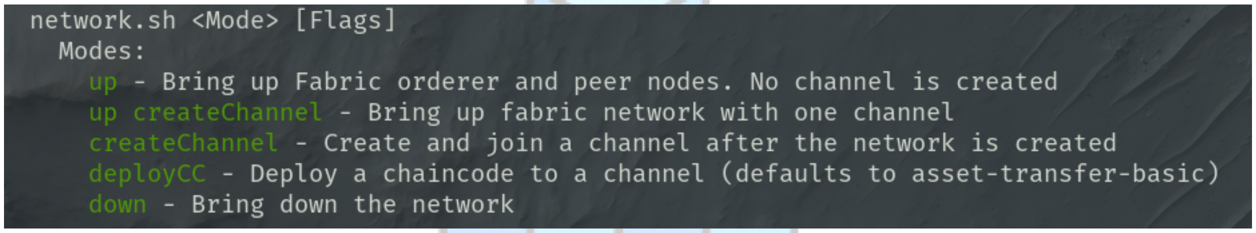
Before getting deep into a test network's working, let's familiarize ourselves with the folders inside the test-network.



- **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 to 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.

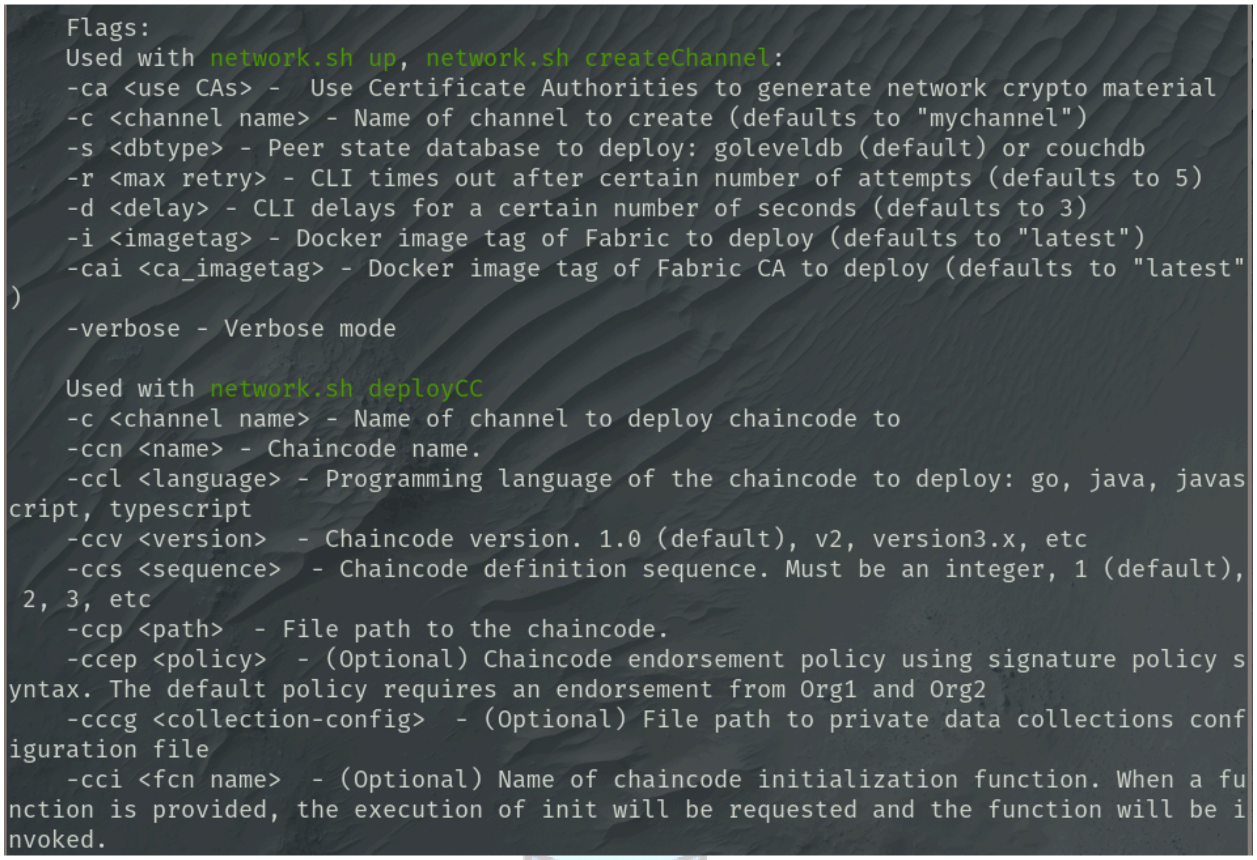


- **prometheus-grafana** -Prometheus is an open-source monitoring system for which Grafana provides data visualization.
- **scripts** - The scripts folder consists of the complete scripts that the test-network uses to modify and control the test network
- **system-genesis-block** - Once the configtxgen command is executed, the resulting genesis block will be stored inside the system-genesis-block folder.
- **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, createChannel, deployCC and down etc.



```
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,

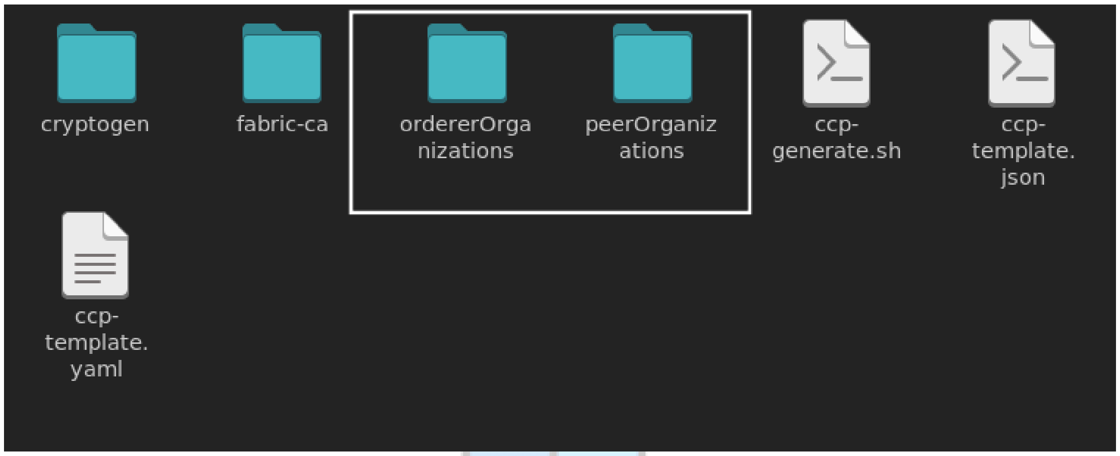
```
./network.sh up createChannel
```

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 in 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

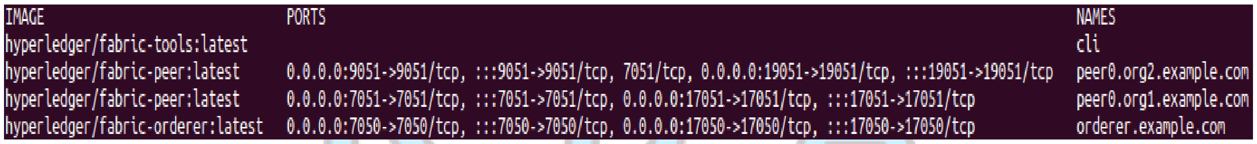
organizations - new folders having crypto materials generated.



You can check the docker containers created using the command

docker ps --format "table {{.Image}}\t{{.Ports}}\t{{.Names}}"

This will give a complete picture of the network



```
IMAGE                              PORTS                                                                                    NAMES
hyperledger/fabric-tools:latest                                                                                             cli
hyperledger/fabric-peer:latest     0.0.0.0:9051->9051/tcp, :::9051->9051/tcp, 7051/tcp, 0.0.0.0:19051->19051/tcp, :::19051->19051/tcp   peer0.org2.example.com
hyperledger/fabric-peer:latest     0.0.0.0:7051->7051/tcp, :::7051->7051/tcp, 0.0.0.0:17051->17051/tcp, :::17051->17051/tcp         peer0.org1.example.com
hyperledger/fabric-orderer:latest  0.0.0.0:7050->7050/tcp, :::7050->7050/tcp, 0.0.0.0:17050->17050/tcp, :::17050->17050/tcp         orderer.example.com
```

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

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

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

Chaincode name (ccn) as basic

Chaincode path (ccp) as chaincode-go location

Chaincode language (ccl) as go

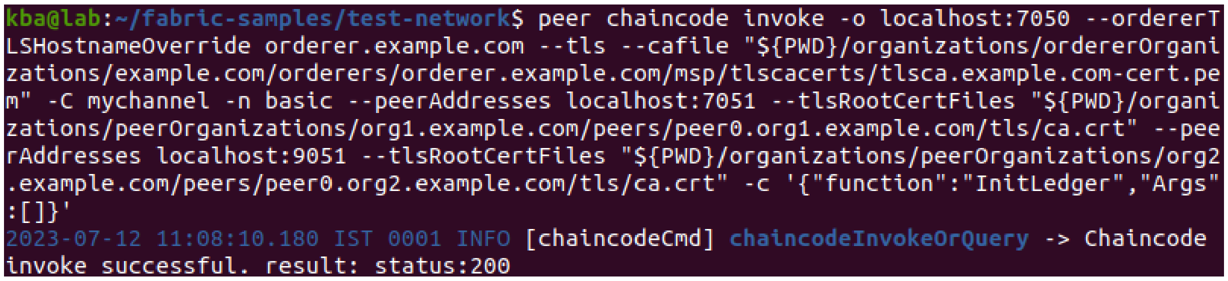
Once the chaincode is deployed, we can finally try interacting with the deployed chaincode.

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/or
g1.example.com/peers/peer0.org1.example.com/tls/ca.crt

export
CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.e
xample.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,

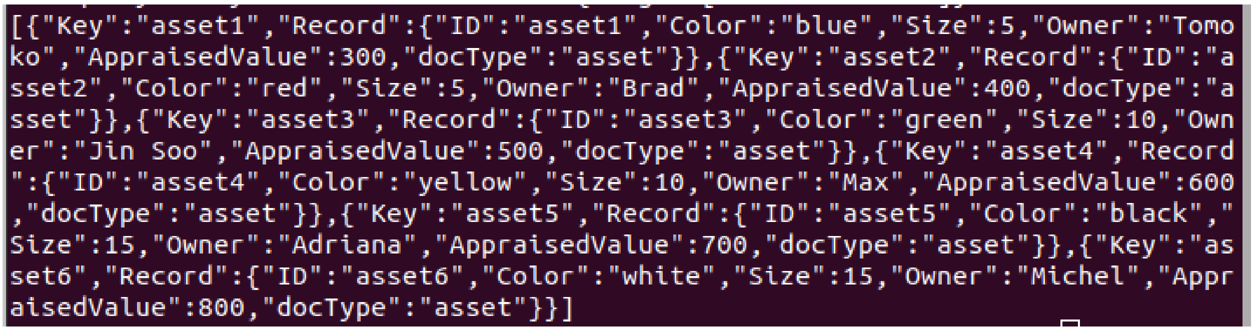
```
peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls --cafile
"${PWD}/organizations/ordererOrganizations/example.com/orderers/order
er.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C
mychannel -n basic --peerAddresses localhost:7051 --tlsRootCertFiles
"${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 assets available, use the following command to query the complete assets.

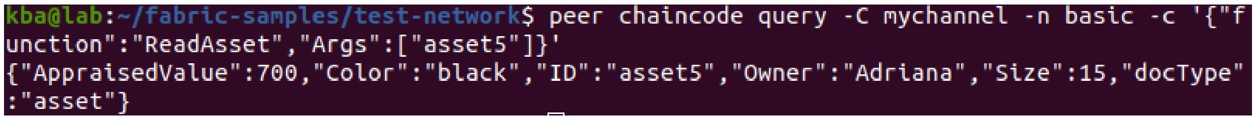
```
peer chaincode query -C mychannel -n basic -c
'{"Args":["GetAllAssets"]}'
```



```
[{"Key":"asset1","Record":{"ID":"asset1","Color":"blue","Size":5,"Owner":"Tomo
ko","AppraisedValue":300,"docType":"asset"}},{"Key":"asset2","Record":{"ID":"a
sset2","Color":"red","Size":5,"Owner":"Brad","AppraisedValue":400,"docType":"a
sset"}},{"Key":"asset3","Record":{"ID":"asset3","Color":"green","Size":10,"Own
er":"Jin Soo","AppraisedValue":500,"docType":"asset"}},{"Key":"asset4","Record
":{"ID":"asset4","Color":"yellow","Size":10,"Owner":"Max","AppraisedValue":600
,"docType":"asset"}},{"Key":"asset5","Record":{"ID":"asset5","Color":"black","
Size":15,"Owner":"Adriana","AppraisedValue":700,"docType":"asset"}},{"Key":"as
set6","Record":{"ID":"asset6","Color":"white","Size":15,"Owner":"Michel","Appr
aisedValue":800,"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"}
```

Stopping the network

To stop the network, you can use the network.sh script

```
./network.sh down
```