Department of Computer Engineering

Academic Term: Jan-May 23-24

Class: B.E Computer Sem -VII Subject:

Blockchain Technology LabSubject

Code : CSDL7022

Practical No:	8
Title:	Case Study on Hyperledger
Date of Performance:	19/09/2023
Date of Submission:	26/09/2023
Roll No:	9427
Name of the Student:	Atharva Prashant Pawar

Evaluation:

Sr. No	**Rubric**	**Grade**
1	Time Line (2)	
2	Output (3)	
3	Code optimization (2)	
4	Post lab (3)	

Signature of the Teacher :

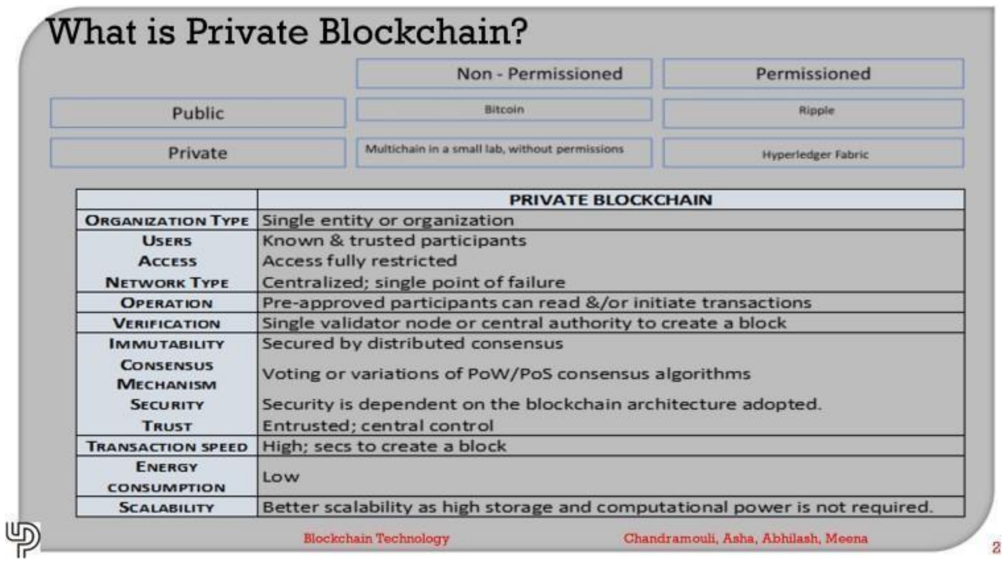
Experiment No.8

Case Study on Hyperledger

Aim: Case Study on Hyperledger

Theory:

Private Blockchain



What is Private Blockchain?

	Non - Permissioned	Permissioned
Public	Bitcoin	Ripple
Private	Multichain in a small lab, without permissions	Hyperledger Fabric

	PRIVATE BLOCKCHAIN
ORGANIZATION TYPE	Single entity or organization
USERS	Known & trusted participants
ACCESS	Access fully restricted
NETWORK TYPE	Centralized; single point of failure
OPERATION	Pre-approved participants can read &/or initiate transactions
VERIFICATION	Single validator node or central authority to create a block
IMMUTABILITY	Secured by distributed consensus
CONSENSUS MECHANISM	Voting or variations of PoW/PoS consensus algorithms
SECURITY	Security is dependent on the blockchain architecture adopted.
TRUST	Entrusted; central control
TRANSACTION SPEED	High; secs to create a block
ENERGY CONSUMPTION	Low
SCALABILITY	Better scalability as high storage and computational power is not required.

Blockchain Technology Chandramouli, Asha, Abhilash, Meena 2

Private blockchains are used by individual hobbyists or by private enterprises or organizations with a specific purpose (e.g., an NGO may like to keep a record of money spent in various schools). Organizations prefer using a private blockchain, if they would like to control:

- Who can use the system
- Who can write to the system
- Who can read the system?

Besides, organizations need a solution with a mechanism to ensure users are added viaprocess, and user rights are created, changed, or deleted by an authorized user. These needsarose and gave birth to a need for private blockchain. In addition, the solution needs faster transactions, proper audit trail, interactions with the organization’s existing IT systems.

Hyperledger:

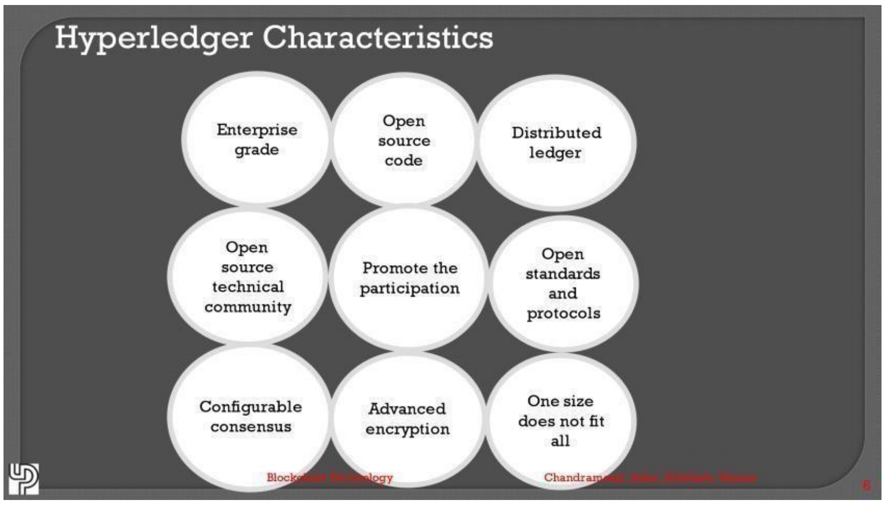
The Linux Foundation took up the challenge for an open-source enterprise-grade distributed ledger technology and announced the Hyperledger Project in December 2015.

- The approach was to ensure that the best practices of computer science related to distributed computing are used in blockchain for enterprise solutions.
- It needs to be noted that Hyperledger has stated they will not be issuing its cryptocurrency

Mision of hyperledger:

As per hyperledger.org, the mission of Hyperledger Project (HLP) is to

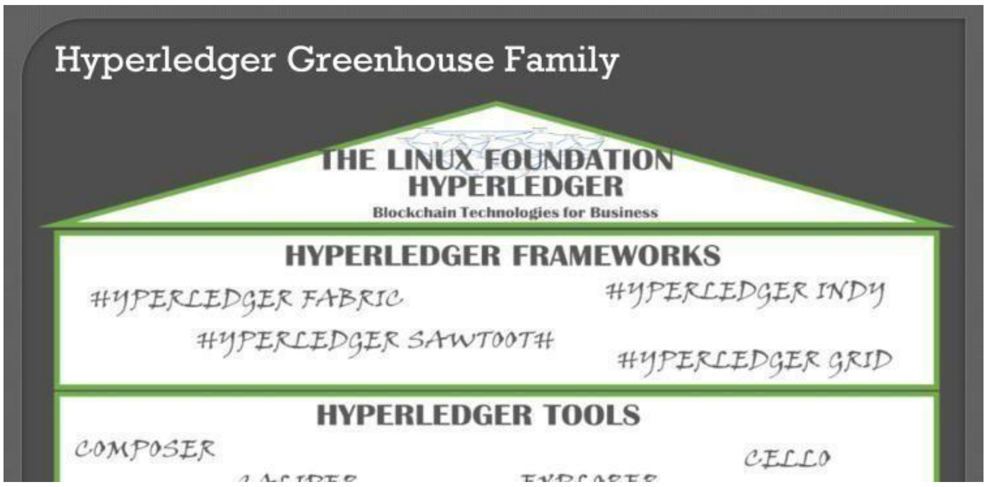
- create an enterprise-grade, open-source distributed ledger framework and codebase
- create an open-source, technical community to benefit the ecosystem of the HLP (Hyper Ledger Project) solution
- promote the participation of leading members of the ecosystem, including developers, service and solution providers and end-users.



Hyperledger Characteristics

Enterprise grade
Open source code
Distributed ledger
Open source technical community
Promote the participation
Open standards and protocols
Configurable consensus
Advanced encryption
One size does not fit all

Blockchain Technology
Chandramouli

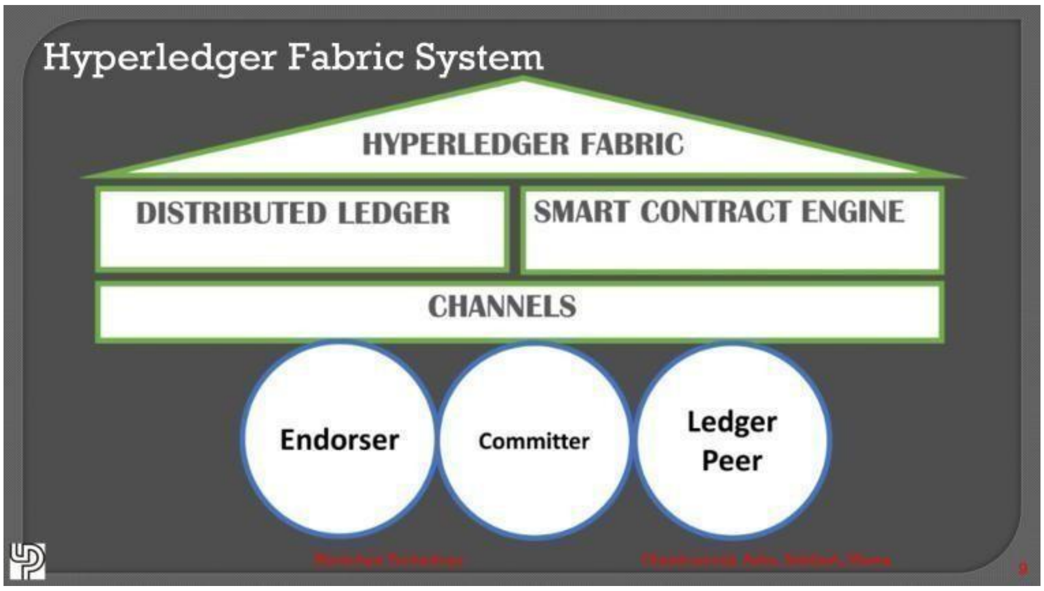


Hyperledger Greenhouse Family

THE LINUX FOUNDATION
HYPERLEDGER
Blockchain Technologies for Business

HYPERLEDGER FRAMEWORKS
HYPERLEDGER FABRIC
HYPERLEDGER INDY
HYPERLEDGER SAWTOOTH
HYPERLEDGER GRID

HYPERLEDGER TOOLS
COMPOSER
CELLO



Hyperledger Indy:

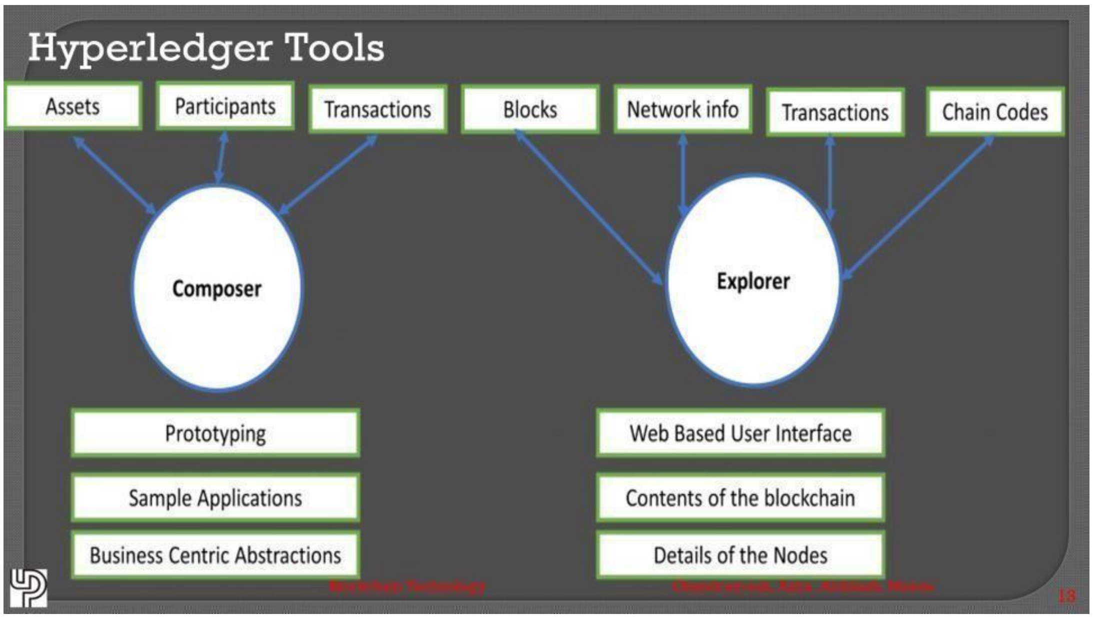
- Hyperledger INDY, a part of the hyperledger framework, is a blockchain tool for digital identity. An organization called sovrin.org donated the code base for INDY.
- INDY is known for providing digital identities in a decentralized environment. INDY provides tools, libraries, and reusable components for the creation of digital identities. INDY's status is incubation, although it has documented specifications for identity along with a sample implementation are available.
- As Hyperledger, INDY is related to identity, and being a blockchain, an identity created once cannot be altered. Designers and administrators of INDY are requested to have proper training for foundational concepts, which includes the following:
    - Privacy by design
    - Privacy-preserving technologies

Hyperledger Sawtooth:

Sawtooth is a distributed ledger technology and powered with a smart contract engine. The sawtooth project started with a contribution via Intel. Sawtooth offers a robust runtime environment, even allowing change of consensus approach in run time. Sawtooth being a permissioned layer bring restrictions via Access Control Lists, nodes are put into these restrictions: Who can connect to the network? Who can send consensus messages? Who can submit transactions to the network? Sawtooth is the only project within Hyperledger Project: that uses Ethereum: and smart contracts are written via solidity, as it is written in Ethereum. Even the smart contracts can be deployed on the fly to the sawtooth network.

Hyperledger Grid:

Hyperledger Grid is a framework (a framework is a set of best practices to achieve a task). Grid is focussed on only one segment: Supply chain Management. Being a framework, Hyperledger grid does not contain rules; instead it is an ecosystem detailing a blockchain for supply chain management. It includes data sets, frameworks that work together, letting application developers choose the best-suited technology or methodology as per company's requirement.



Hyperledger tools:

Composer:
Composer offers business-centric abstractions as well as sample apps, which are used to test or replicate business problems. Composer is handy when you have to build an application part of Proof-of-Concepts (PoC), and you have a concise timeline. Composer operates as a rapid prototyping tool for user-facing solutions. Hyperledger fabric is the underlying mechanism to operate a blockchain for the composer. Composer allows for creation/modification of the following:

- Assets
- Participants
- Transactions

Explorer

Explorer is another tool hosted in the Hyperledger greenhouse, which provides a web-based user interface. Hyperledger Explorer allows the user to view contents in the blockchain, and list the nodes Hyperledger Explorer can view, invoke, deploy or query. These nodes include the following:

- Blocks

- Transactions
- Network information (name, status, list of nodes)
- Chain codes.

Hyperledger Fabric:

It is the most popular of the Hyperledger project. All blockchains strive to solve the problem of trust and time, and blockchain Hyperledger fabric is no different. Hyperledger fabric is amongst the best solutions where the organization can choose the trust mechanism it needs to use. The issue of trust has a profound impact on Supply Chain Management, which is the most, talked and piloted use-case of blockchain.
Before you install Hyperledger Fabric, you must first download and install the prerequisites that are required to run a Docker-based Fabric test network on your local machine from

https://hyperledger-fabric.readthedocs.io/en/latest/prereqs.html

Hyperleder Fabric Prerequisites Setup:

Curl Installation

Run below command to install Curl.

```
$  sudo apt-get install curl
```

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

```
$  curl –version
```

NodeJs Installation

Open the terminal window and run below command to download and execute the nodejs file.

```
$  curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash –
```

Then run below command.

```
$  sudo apt-get update
```

Run below command to start the installation for NodeJs.

```
$  sudo apt-get install nodejs
```

Run below command to check if Nodejs is successfully installed or not. This should return the version of NodeJs.

```
$ node –version
```

Git Installation

Open the terminal window and run below command. This will start the installation for Git.

```
$ sudo apt-get install git
```

Run below command to check if Git is successfully installed or not. This should return the version of Git.

```
$ git –version
```

Python Installation

In the terminal window, run below command to install Python.

```
$ sudo apt-get install python
```

Verify the installation by running below command and that should return the version of Python.

```
$ python –version
```

Lib Tools Installation

Install Lib tools using below command.

```
$ sudo apt-get install libltdl-dev
```

Install Docker CE (Community Edition)

First download and then install it using below commands.

```
$ wget
  https://download.docker.com/linux/ubuntu/dists/xenial/pool/stable/amd64/
  docker-ce_18.06.3~ce~3-0~ubuntu_amd64.deb
```

```
$ sudodpkg -idocker-ce_18.06.3~ce~3-0~ubuntu_amd64.deb
```

Check the version of docker using below command and this should return the version of docker.

```
$ docker –version
```

Install Docker Compose

Run below commands to setup Docker compose.

```
$ sudo apt-get install python-pip
```

```
$ pip --version
```

```
$ sudo pip install docker-compose
```

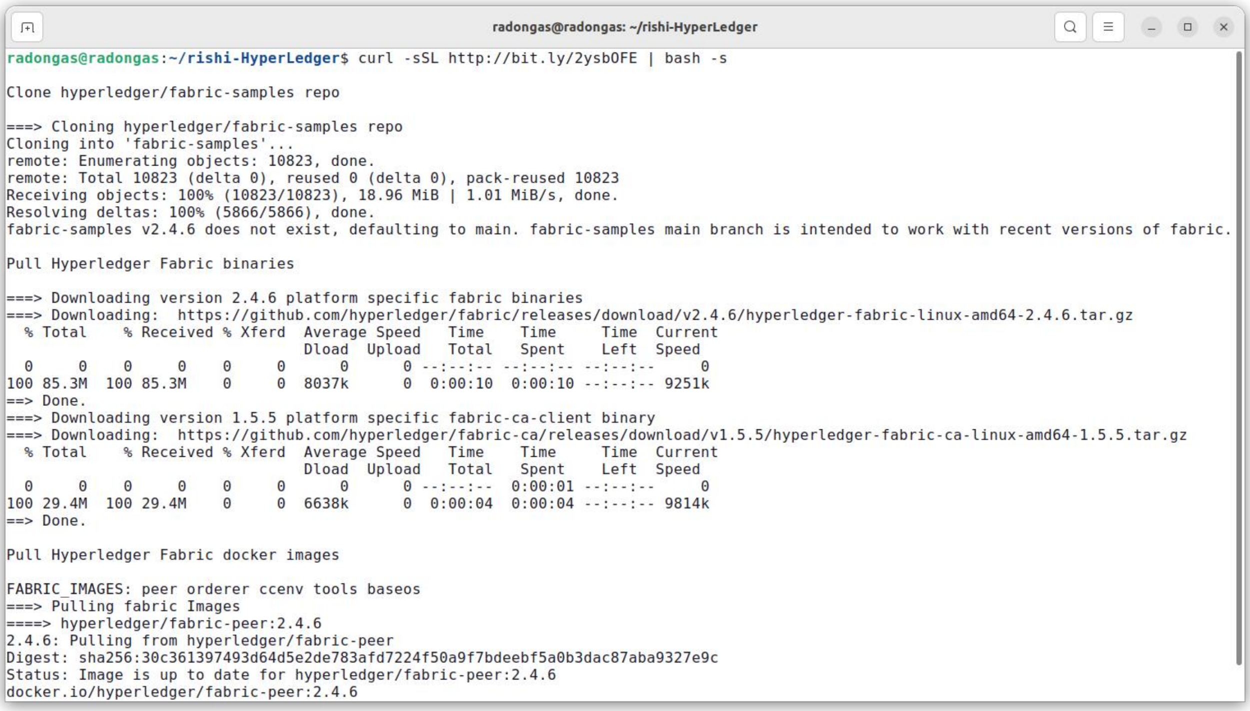
Verify the installation and check the version from below command.

```
$  docker-compose version
```

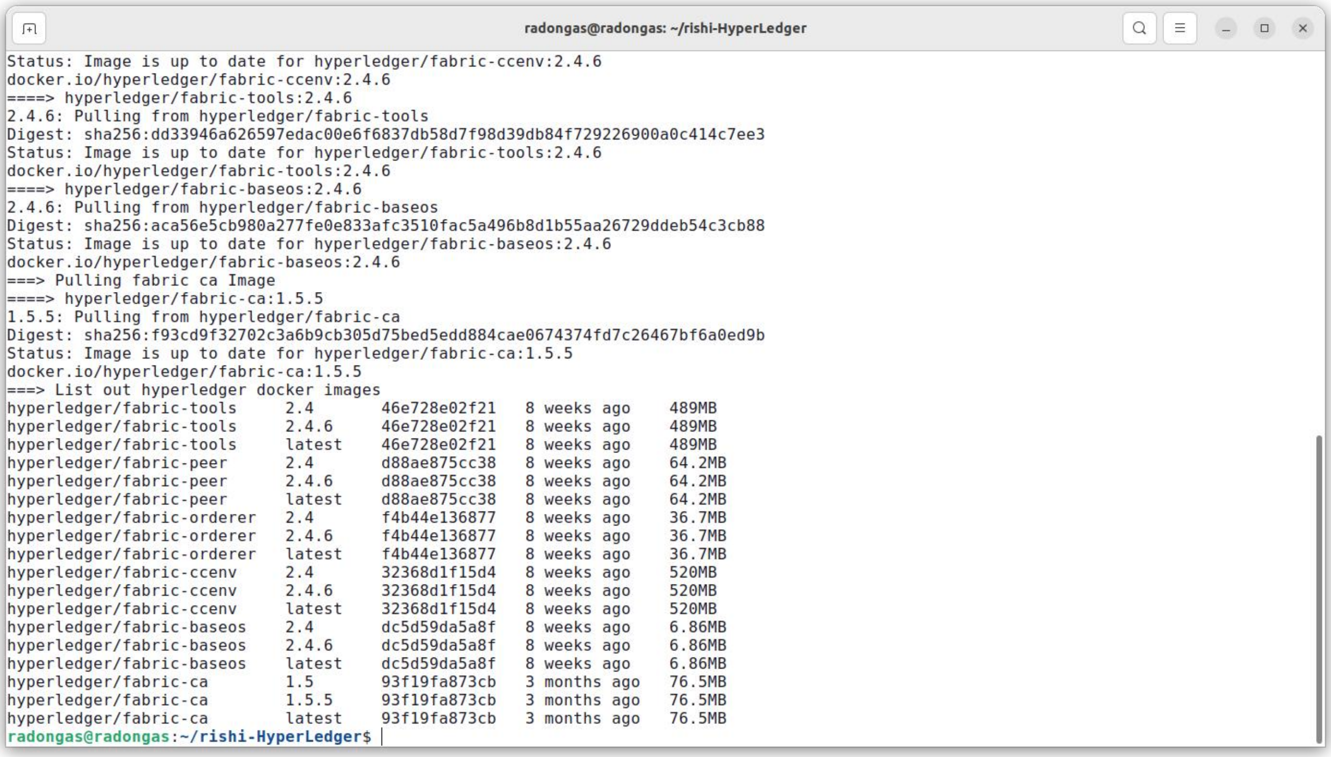
Hyperledger Installation:

Step 1: Run below command to download and setup Fabric.

```
$  curl -sSL https://bit.ly/2ysbOFE | bash -s
```



```
radongas@radongas: ~/rishi-HyperLedger
radongas@radongas:~/rishi-HyperLedger$ curl -sSL http://bit.ly/2ysbOFE | bash -s

Clone hyperledger/fabric-samples repo

===> Cloning hyperledger/fabric-samples repo
Cloning into 'fabric-samples'...
remote: Enumerating objects: 10823, done.
remote: Total 10823 (delta 0), reused 0 (delta 0), pack-reused 10823
Receiving objects: 100% (10823/10823), 18.96 MiB | 1.01 MiB/s, done.
Resolving deltas: 100% (5866/5866), done.
fabric-samples v2.4.6 does not exist, defaulting to main. fabric-samples main branch is intended to work with recent versions of fabric.

Pull Hyperledger Fabric binaries

===> Downloading version 2.4.6 platform specific fabric binaries
===> Downloading:  https://github.com/hyperledger/fabric/releases/download/v2.4.6/hyperledger-fabric-linux-amd64-2.4.6.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 85.3M  100 85.3M    0     0  8037k      0  0:00:10  0:00:10 --:--:-- 9251k
==> Done.
===> Downloading version 1.5.5 platform specific fabric-ca-client binary
===> Downloading:  https://github.com/hyperledger/fabric-ca/releases/download/v1.5.5/hyperledger-fabric-ca-linux-amd64-1.5.5.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:--     0
100 29.4M  100 29.4M    0     0  6638k      0  0:00:04  0:00:04 --:--:-- 9814k
==> Done.

Pull Hyperledger Fabric docker images

FABRIC_IMAGES: peer orderer ccenv tools baseos
===> Pulling fabric Images
====> hyperledger/fabric-peer:2.4.6
2.4.6: Pulling from hyperledger/fabric-peer
Digest: sha256:30c361397493d64d5e2de783afd7224f50a9f7bdeebf5a0b3dac87aba9327e9c
Status: Image is up to date for hyperledger/fabric-peer:2.4.6
docker.io/hyperledger/fabric-peer:2.4.6
```



```
radongas@radongas: ~/rishi-HyperLedger
Status: Image is up to date for hyperledger/fabric-ccenv:2.4.6
docker.io/hyperledger/fabric-ccenv:2.4.6
====> hyperledger/fabric-tools:2.4.6
2.4.6: Pulling from hyperledger/fabric-tools
Digest: sha256:dd33946a626597edac00e6f6837db58d7f98d39db84f729226900a0c414c7ee3
Status: Image is up to date for hyperledger/fabric-tools:2.4.6
docker.io/hyperledger/fabric-tools:2.4.6
====> hyperledger/fabric-baseos:2.4.6
2.4.6: Pulling from hyperledger/fabric-baseos
Digest: sha256:aca56e5cb980a277fe0e833afc3510fac5a496b8d1b55aa26729ddeb54c3cb88
Status: Image is up to date for hyperledger/fabric-baseos:2.4.6
docker.io/hyperledger/fabric-baseos:2.4.6
===> Pulling fabric ca Image
====> hyperledger/fabric-ca:1.5.5
1.5.5: Pulling from hyperledger/fabric-ca
Digest: sha256:f93cd9f32702c3a6b9cb305d75bed5edd884cae0674374fd7c26467bf6a0ed9b
Status: Image is up to date for hyperledger/fabric-ca:1.5.5
docker.io/hyperledger/fabric-ca:1.5.5
===> List out hyperledger docker images
hyperledger/fabric-tools     2.4       46e728e02f21   8 weeks ago    489MB
hyperledger/fabric-tools     2.4.6     46e728e02f21   8 weeks ago    489MB
hyperledger/fabric-tools     latest    46e728e02f21   8 weeks ago    489MB
hyperledger/fabric-peer      2.4       d88ae875cc38   8 weeks ago    64.2MB
hyperledger/fabric-peer      2.4.6     d88ae875cc38   8 weeks ago    64.2MB
hyperledger/fabric-peer      latest    d88ae875cc38   8 weeks ago    64.2MB
hyperledger/fabric-orderer   2.4       f4b44e136877   8 weeks ago    36.7MB
hyperledger/fabric-orderer   2.4.6     f4b44e136877   8 weeks ago    36.7MB
hyperledger/fabric-orderer   latest    f4b44e136877   8 weeks ago    36.7MB
hyperledger/fabric-ccenv     2.4       32368d1f15d4   8 weeks ago    520MB
hyperledger/fabric-ccenv     2.4.6     32368d1f15d4   8 weeks ago    520MB
hyperledger/fabric-ccenv     latest    32368d1f15d4   8 weeks ago    520MB
hyperledger/fabric-baseos    2.4       dc5d59da5a8f   8 weeks ago    6.86MB
hyperledger/fabric-baseos    2.4.6     dc5d59da5a8f   8 weeks ago    6.86MB
hyperledger/fabric-baseos    latest    dc5d59da5a8f   8 weeks ago    6.86MB
hyperledger/fabric-ca        1.5       93f19fa873cb   3 months ago   76.5MB
hyperledger/fabric-ca        1.5.5     93f19fa873cb   3 months ago   76.5MB
hyperledger/fabric-ca        latest    93f19fa873cb   3 months ago   76.5MB
radongas@radongas:~/rishi-HyperLedger$
```

Running Hyperleder Fabric Testnetwork:

Step 1: Go to fabric-samples folder by using below command.

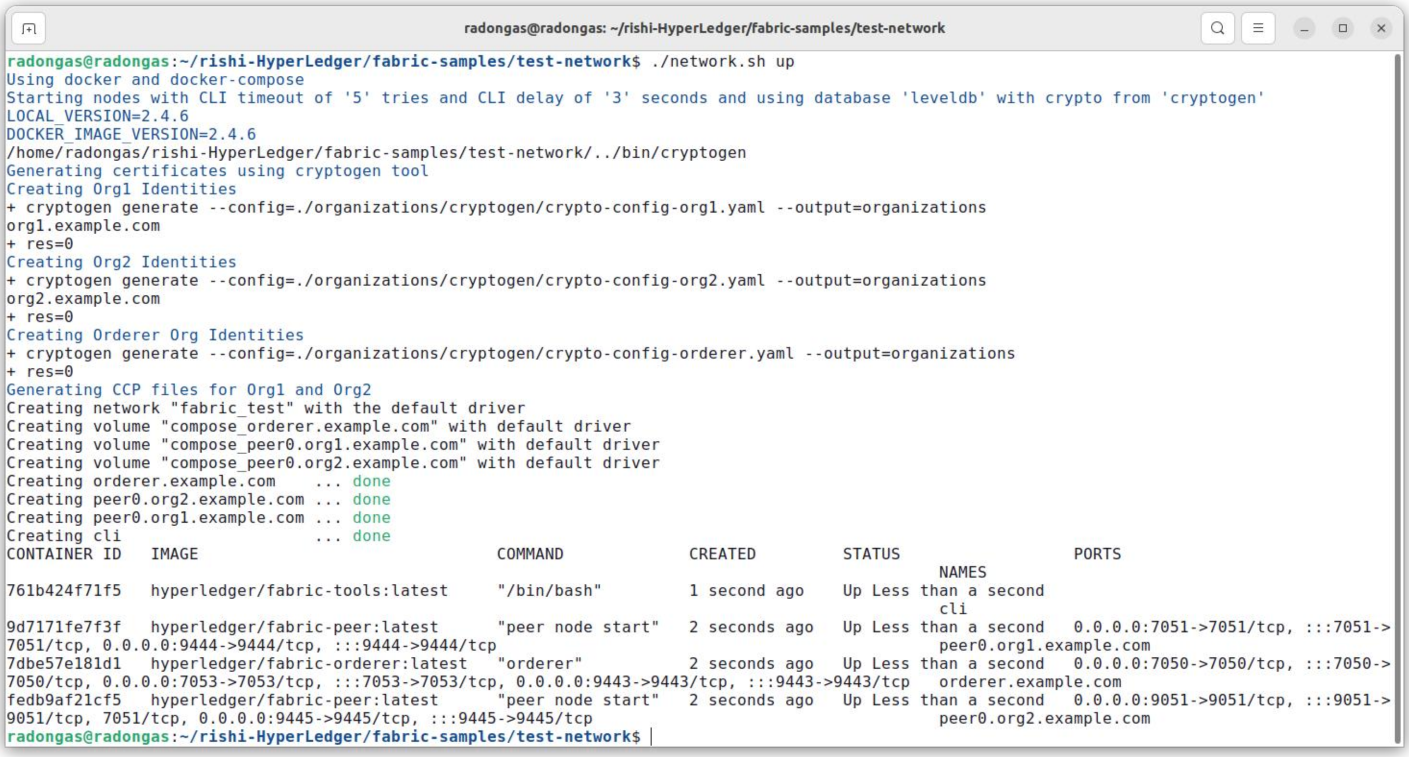
```
$  cd fabric-samples
```

Step 2: Go to test-network folder by using below command.

```
$  cd test-network
```

Step 3: Run below command to start your test-network

```
$  sudo ./network.sh up
```


radongas@radongas: ~/rishi-HyperLedger/fabric-samples/test-network

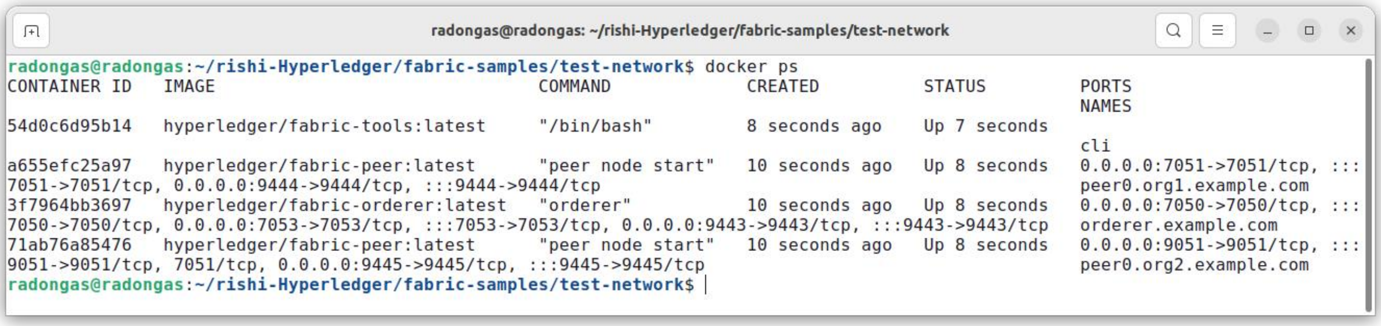
```
radongas@radongas:~/rishi-HyperLedger/fabric-samples/test-network$ ./network.sh up
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto from 'cryptogen'
LOCAL_VERSION=2.4.6
DOCKER_IMAGE_VERSION=2.4.6
/home/radongas/rishi-HyperLedger/fabric-samples/test-network/../bin/cryptogen
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Creating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
Creating network "fabric_test" with the default driver
Creating volume "compose_orderer.example.com" with default driver
Creating volume "compose_peer0.org1.example.com" with default driver
Creating volume "compose_peer0.org2.example.com" with default driver
Creating orderer.example.com    ... done
Creating peer0.org2.example.com ... done
Creating peer0.org1.example.com ... done
Creating cli                    ... done
CONTAINER ID   IMAGE                               COMMAND             CREATED         STATUS                  PORTS
                                                                                                  NAMES
761b424f71f5   hyperledger/fabric-tools:latest     "/bin/bash"         1 second ago    Up Less than a second
                                                                                                  cli
9d7171fe7f3f   hyperledger/fabric-peer:latest      "peer node start"   2 seconds ago   Up Less than a second   0.0.0.0:7051->7051/tcp, :::7051->
7051/tcp, 0.0.0.0:9444->9444/tcp, :::9444->9444/tcp                                               peer0.org1.example.com
7dbe57e181d1   hyperledger/fabric-orderer:latest   "orderer"           2 seconds ago   Up Less than a second   0.0.0.0:7050->7050/tcp, :::7050->
7050/tcp, 0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp   orderer.example.com
fedb9af21cf5   hyperledger/fabric-peer:latest      "peer node start"   2 seconds ago   Up Less than a second   0.0.0.0:9051->9051/tcp, :::9051->
9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp, :::9445->9445/tcp                                     peer0.org2.example.com
radongas@radongas:~/rishi-HyperLedger/fabric-samples/test-network$ |
```

This start the network, you can run below command to check docker containers.

```
$  sudo docker ps
```

This shows you three docker containers

- One for Org1 peer node
- One for Org2 peer node
- One for Orderer

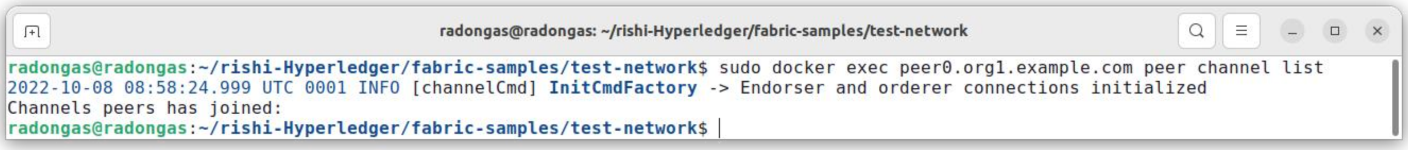

radongas@radongas: ~/rishi-Hyperledger/fabric-samples/test-network

```
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ docker ps
CONTAINER ID   IMAGE                               COMMAND             CREATED          STATUS         PORTS
                                                                                                       NAMES
54d0c6d95b14   hyperledger/fabric-tools:latest     "/bin/bash"         8 seconds ago    Up 7 seconds
                                                                                                       cli
a655efc25a97   hyperledger/fabric-peer:latest      "peer node start"   10 seconds ago   Up 8 seconds   0.0.0.0:7051->7051/tcp, :::
7051->7051/tcp, 0.0.0.0:9444->9444/tcp, :::9444->9444/tcp                                              peer0.org1.example.com
3f7964bb3697   hyperledger/fabric-orderer:latest   "orderer"           10 seconds ago   Up 8 seconds   0.0.0.0:7050->7050/tcp, :::
7050->7050/tcp, 0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp   orderer.example.com
71ab76a85476   hyperledger/fabric-peer:latest      "peer node start"   10 seconds ago   Up 8 seconds   0.0.0.0:9051->9051/tcp, :::
9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp, :::9445->9445/tcp                                    peer0.org2.example.com
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ |
```

When you start the network, you will also not get any channel by default. You can check the channel by using below command.

```
$  sudo docker exec peer0.org1.example.com peer channel list
```

This command shows you that, you don't have any channel created.



radongas@radongas: ~/rishi-Hyperledger/fabric-samples/test-network

```
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ sudo docker exec peer0.org1.example.com peer channel list
2022-10-08 08:58:24.999 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Channels peers has joined:
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ |
```

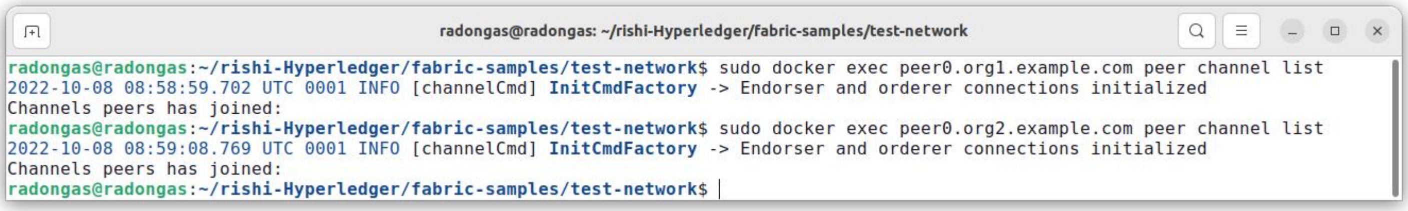
Step 4: Create new channel by using below command.

```
$  sudo ./network.sh createChannel -c testchannel
```

This will create a new channel with the name test channel.

To verify this channel creation, run below command on both the peers.

```
$  sudo docker exec peer0.org1.example.com peer channel list

$  sudo docker exec peer0.org2.example.com peer channel list
```



radongas@radongas: ~/rishi-Hyperledger/fabric-samples/test-network

```
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ sudo docker exec peer0.org1.example.com peer channel list
2022-10-08 08:58:59.702 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Channels peers has joined:
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ sudo docker exec peer0.org2.example.com peer channel list
2022-10-08 08:59:08.769 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
Channels peers has joined:
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ |
```

Step 5: To stop the network, you need to run below command.

```
   $  sudo ./network.sh down
```

<u>**Working with State DataBase (Couch DB):**</u>

Step 1: Go to fabric-samples folder by using below command.

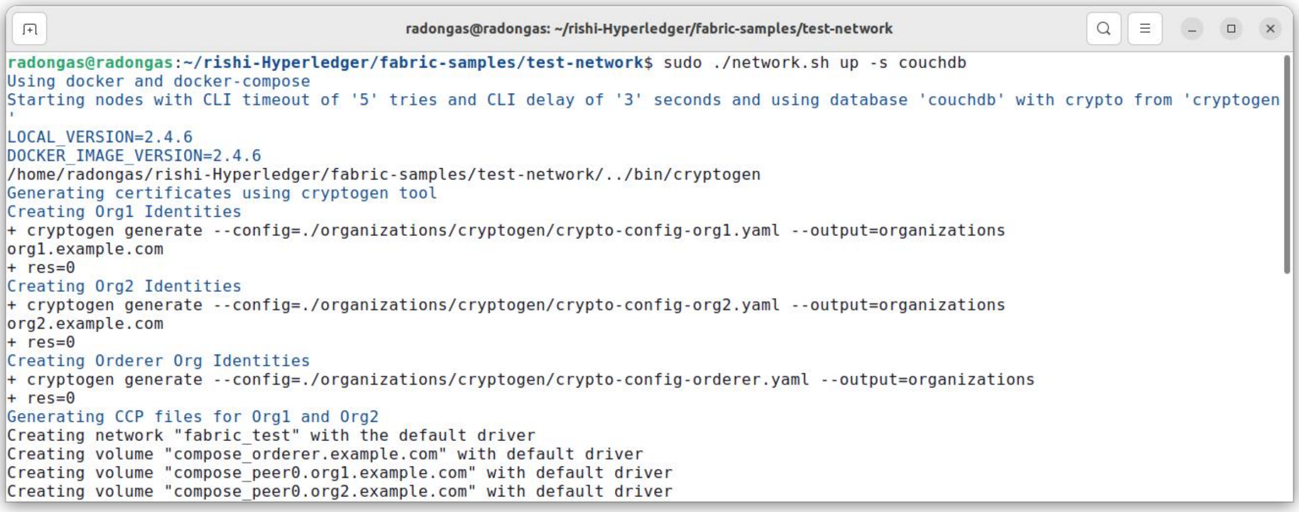
```
   $  cd fabric-samples
```

Step 2: Go to test-network folder by using below command.

```
   $  cd test-network
```

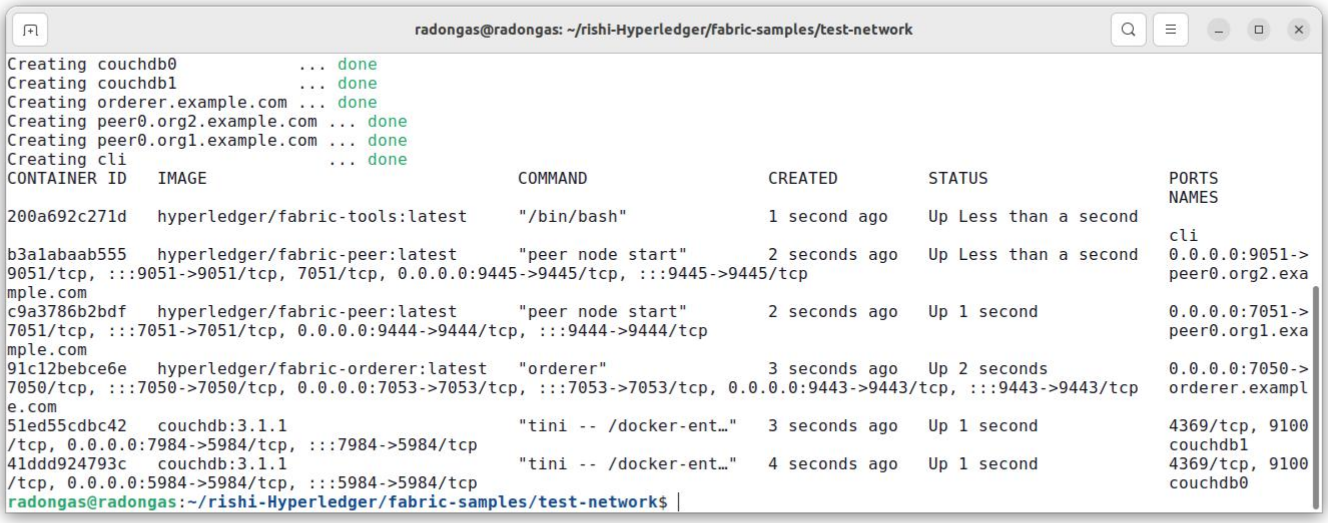
Step 3: Run below command to start the network and create couchDB containers as well.

```
   $  sudo ./network.sh up -s couchdb
```


radongas@radongas: ~/rishi-Hyperledger/fabric-samples/test-network

```
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ sudo ./network.sh up -s couchdb
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'couchdb' with crypto from 'cryptogen
'
LOCAL_VERSION=2.4.6
DOCKER_IMAGE_VERSION=2.4.6
/home/radongas/rishi-Hyperledger/fabric-samples/test-network/../bin/cryptogen
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Creating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
Creating network "fabric_test" with the default driver
Creating volume "compose_orderer.example.com" with default driver
Creating volume "compose_peer0.org1.example.com" with default driver
Creating volume "compose_peer0.org2.example.com" with default driver
```

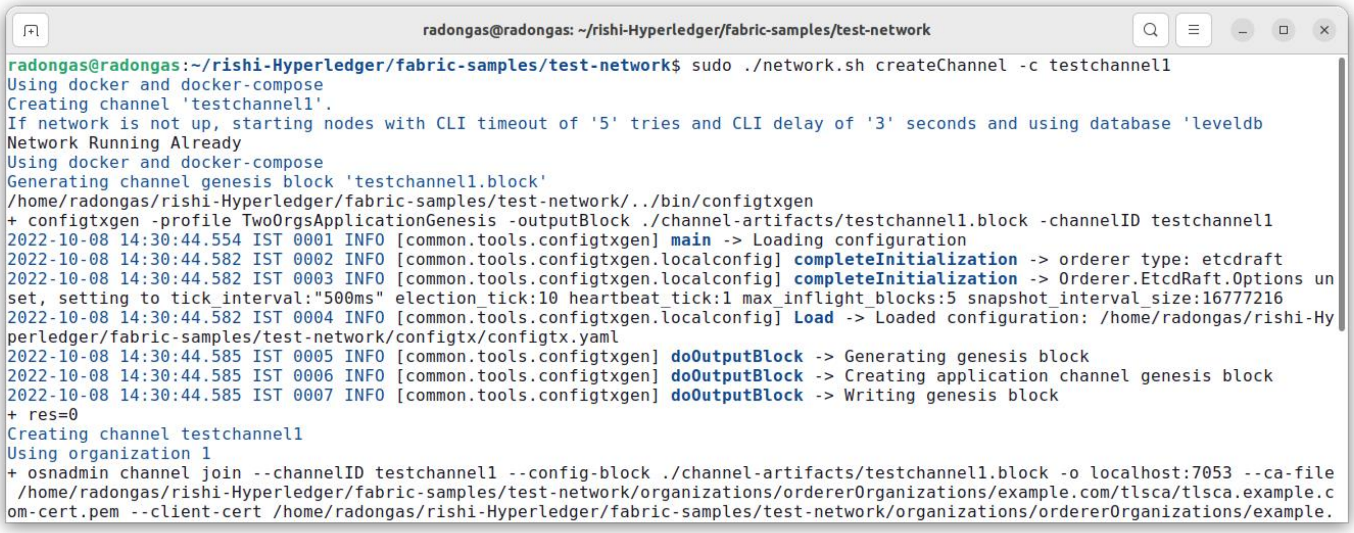
This command starts your network and create couchdb container for each peer as well.


radongas@radongas: ~/rishi-Hyperledger/fabric-samples/test-network

```
Creating couchdb0            ... done
Creating couchdb1            ... done
Creating orderer.example.com ... done
Creating peer0.org2.example.com ... done
Creating peer0.org1.example.com ... done
Creating cli                    ... done
CONTAINER ID   IMAGE                               COMMAND                  CREATED          STATUS                    PORTS
                                                                                                                       NAMES
200a692c271d   hyperledger/fabric-tools:latest     "/bin/bash"              1 second ago     Up Less than a second
                                                                                                                       cli
b3a1abaab555   hyperledger/fabric-peer:latest      "peer node start"        2 seconds ago    Up Less than a second     0.0.0.0:9051->
9051/tcp, :::9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp, :::9445->9445/tcp                                       peer0.org2.exa
mple.com
c9a3786b2bdf   hyperledger/fabric-peer:latest      "peer node start"        2 seconds ago    Up 1 second               0.0.0.0:7051->
7051/tcp, :::7051->7051/tcp, 0.0.0.0:9444->9444/tcp, :::9444->9444/tcp                                                 peer0.org1.exa
mple.com
91c12bebce6e   hyperledger/fabric-orderer:latest   "orderer"                3 seconds ago    Up 2 seconds              0.0.0.0:7050->
7050/tcp, :::7050->7050/tcp, 0.0.0.0:7053->7053/tcp, :::7053->7053/tcp, 0.0.0.0:9443->9443/tcp, :::9443->9443/tcp      orderer.exampl
e.com
51ed55cdbc42   couchdb:3.1.1                       "tini -- /docker-ent…"   3 seconds ago    Up 1 second               4369/tcp, 9100
/tcp, 0.0.0.0:7984->5984/tcp, :::7984->5984/tcp                                                                        couchdb1
41ddd924793c   couchdb:3.1.1                       "tini -- /docker-ent…"   4 seconds ago    Up 1 second               4369/tcp, 9100
/tcp, 0.0.0.0:5984->5984/tcp, :::5984->5984/tcp                                                                        couchdb0
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ |
```

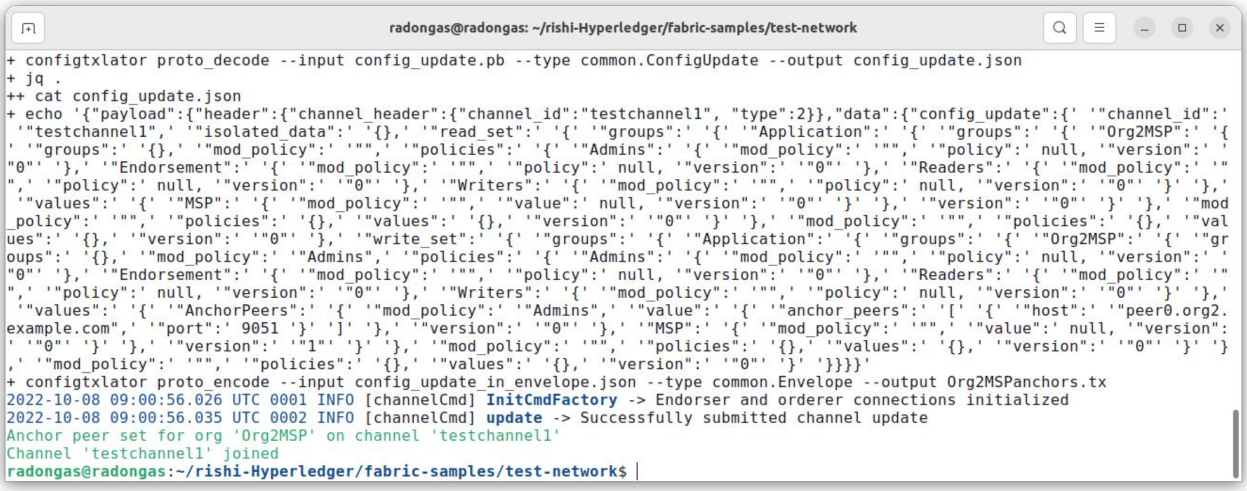
Step 4: Create new channel by using below command.

```
$  sudo ./network.sh createChannel -c testchannel1
```


radongas@radongas: ~/rishi-Hyperledger/fabric-samples/test-network

```
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ sudo ./network.sh createChannel -c testchannel1
Using docker and docker-compose
Creating channel 'testchannel1'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb
Network Running Already
Using docker and docker-compose
Generating channel genesis block 'testchannel1.block'
/home/radongas/rishi-Hyperledger/fabric-samples/test-network/../bin/configtxgen
+ configtxgen -profile TwoOrgsApplicationGenesis -outputBlock ./channel-artifacts/testchannel1.block -channelID testchannel1
2022-10-08 14:30:44.554 IST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2022-10-08 14:30:44.582 IST 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> orderer type: etcdraft
2022-10-08 14:30:44.582 IST 0003 INFO [common.tools.configtxgen.localconfig] completeInitialization -> Orderer.EtcdRaft.Options un
set, setting to tick_interval:"500ms" election_tick:10 heartbeat_tick:1 max_inflight_blocks:5 snapshot_interval_size:16777216
2022-10-08 14:30:44.582 IST 0004 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: /home/radongas/rishi-Hy
perledger/fabric-samples/test-network/configtx/configtx.yaml
2022-10-08 14:30:44.585 IST 0005 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
2022-10-08 14:30:44.585 IST 0006 INFO [common.tools.configtxgen] doOutputBlock -> Creating application channel genesis block
2022-10-08 14:30:44.585 IST 0007 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
+ res=0
Creating channel testchannel1
Using organization 1
+ osnadmin channel join --channelID testchannel1 --config-block ./channel-artifacts/testchannel1.block -o localhost:7053 --ca-file
 /home/radongas/rishi-Hyperledger/fabric-samples/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.c
om-cert.pem --client-cert /home/radongas/rishi-Hyperledger/fabric-samples/test-network/organizations/ordererOrganizations/example.
```

This will create a new channel with the name testchannel1.


```
+ 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":"testchannel1", "type":2}},"data":{"config_update":{' '"channel_id":'
 '"testchannel1",' '"isolated_data":' '{},' '"read_set":' '{' '"groups":' '{' '"Application":' '{' '"groups":' '{' '"Org2MSP":' '{
' '"groups":' '{},' '"mod_policy":' '"",' '"policies":' '{' '"Admins":' '{' '"mod_policy":' '"",' '"policy":' 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":' '{' '"MSP":' '{' '"mod_policy":' '"",' '"value":' null, '"version":' '"0"' '}' '},' '"version":' '"0"' '}' '},' '"mod
_policy":' '"",' '"policies":' '{},' '"values":' '{},' '"version":' '"0"' '}' '},' '"mod_policy":' '"",' '"policies":' '{},' '"val
ues":' '{},' '"version":' '"0"' '},' '"write_set":' '{' '"groups":' '{' '"Application":' '{' '"groups":' '{' '"Org2MSP":' '{' '"gr
oups":' '{},' '"mod_policy":' '"Admins",' '"policies":' '{' '"Admins":' '{' '"mod_policy":' '"",' '"policy":' 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"' '}' '},' '"version":' '"1"' '}' '},' '"mod_policy":' '"",' '"policies":' '{},' '"values":' '{},' '"version":' '"0"' '}' '}
,' '"mod_policy":' '"",' '"policies":' '{},' '"values":' '{},' '"version":' '"0"' '}' '}}}}'
+ configtxlator proto_encode --input config_update_in_envelope.json --type common.Envelope --output Org2MSPanchors.tx
2022-10-08 09:00:56.026 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2022-10-08 09:00:56.035 UTC 0002 INFO [channelCmd] update -> Successfully submitted channel update
Anchor peer set for org 'Org2MSP' on channel 'testchannel1'
Channel 'testchannel1' joined
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ |
```

Step 5: To stop the network, you need to run below command.

```
$  sudo ./network.sh down
```

SET UP THE BLOCKCHAIN NETWORK:

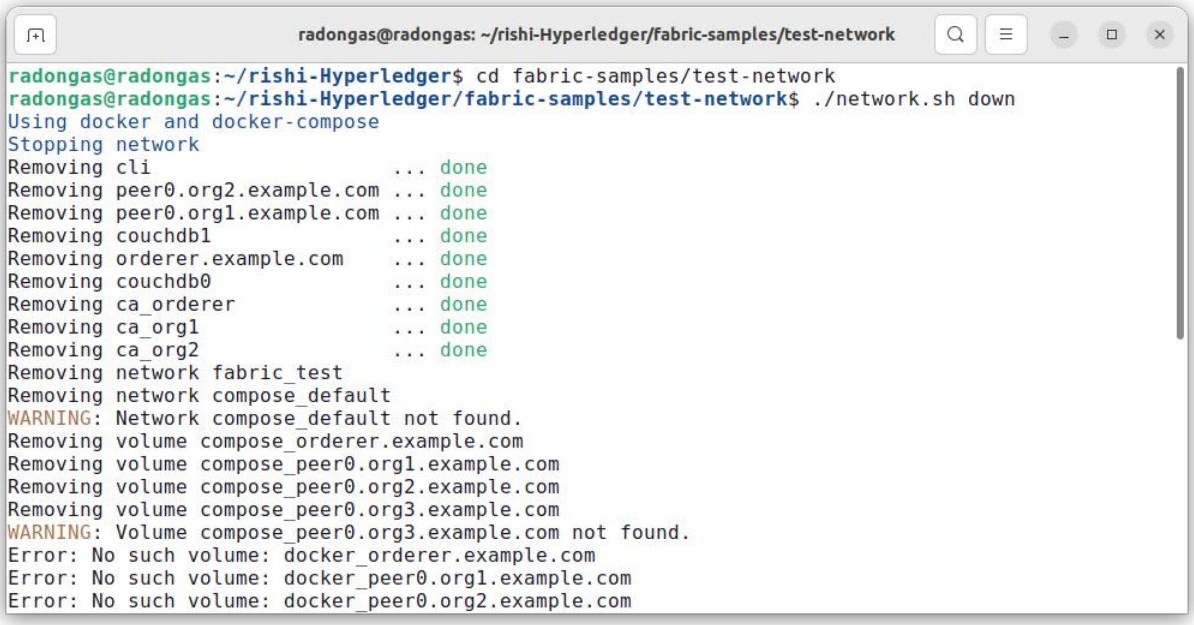
If you've already run through Using the Fabric test network tutorial and have a network up and running, this tutorial will bring down your running network before bringing up a new one.

```
$  cdfabric-samples/test-network
```

Navigate to the test-network subdirectory within your local clone of the fabric-samples repository.

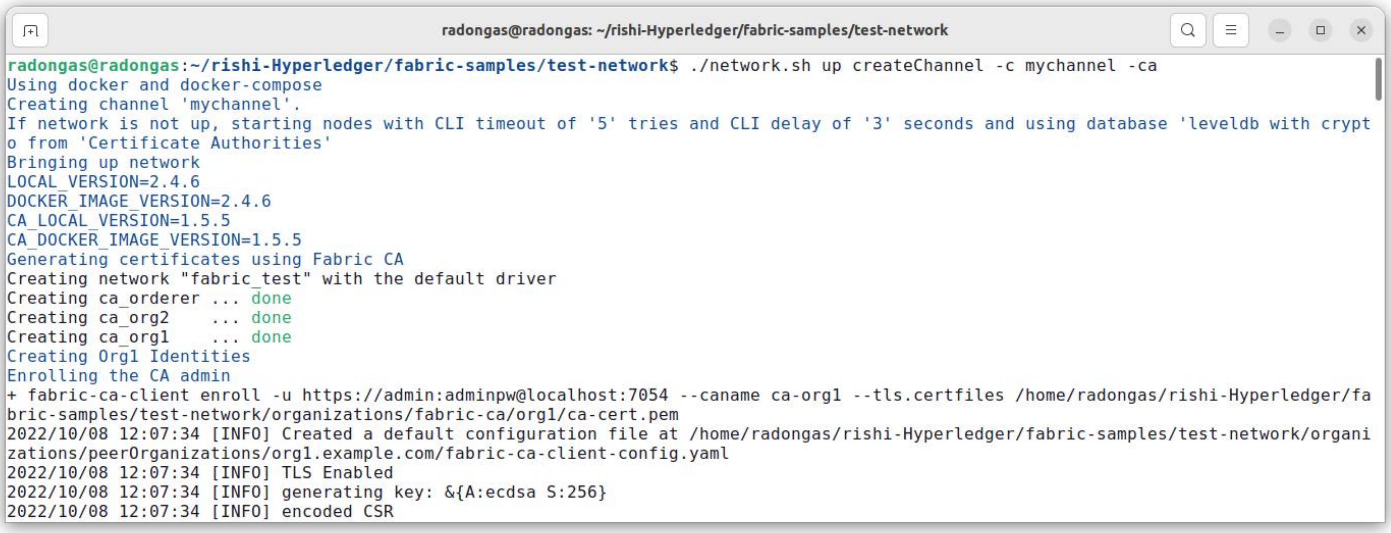
If you already have a test network running, bring it down to ensure the environment is clean.

```
$  ./network.shdown
```


```
radongas@radongas:~/rishi-Hyperledger$ cd fabric-samples/test-network
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ ./network.sh down
Using docker and docker-compose
Stopping network
Removing cli                    ... done
Removing peer0.org2.example.com ... done
Removing peer0.org1.example.com ... done
Removing couchdb1               ... done
Removing orderer.example.com    ... done
Removing couchdb0               ... done
Removing ca_orderer             ... done
Removing ca_org1                ... done
Removing ca_org2                ... 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: No such volume: docker_orderer.example.com
Error: No such volume: docker_peer0.org1.example.com
Error: No such volume: docker_peer0.org2.example.com
```

Launch the Fabric test network using the network.sh shell script.

```
$ ./network.shupcreateChannel-cmychannel-ca
```



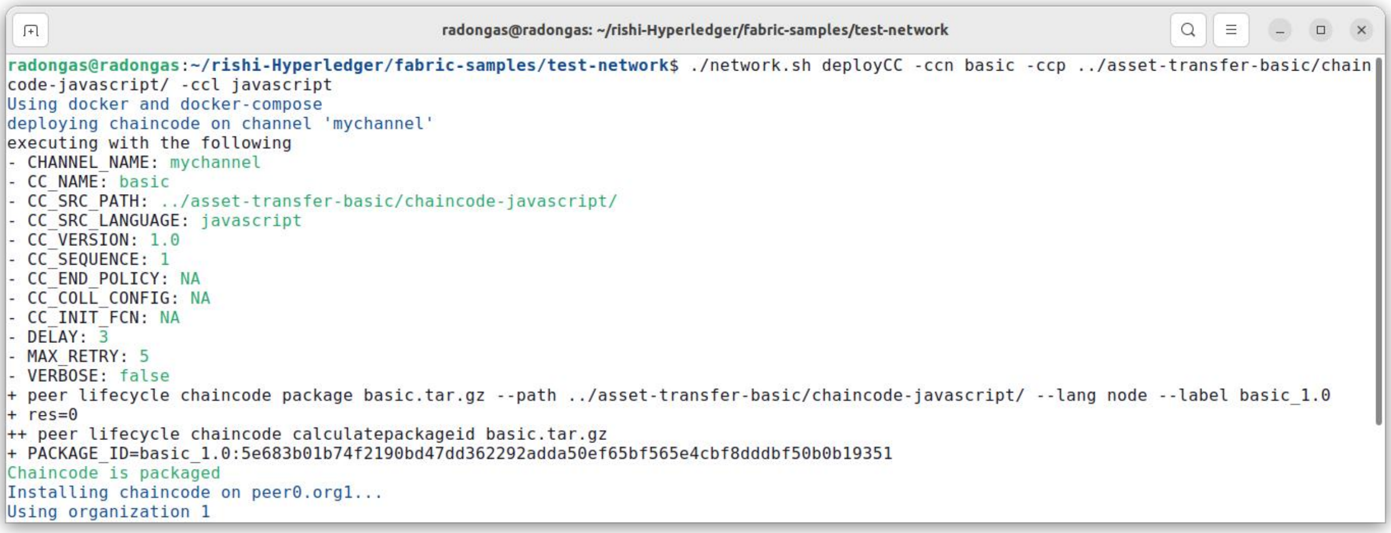
radongas@radongas: ~/rishi-Hyperledger/fabric-samples/test-network

```
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ ./network.sh up createChannel -c mychannel -ca
Using docker and docker-compose
Creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb with crypt
o from 'Certificate Authorities'
Bringing up network
LOCAL_VERSION=2.4.6
DOCKER_IMAGE_VERSION=2.4.6
CA_LOCAL_VERSION=1.5.5
CA_DOCKER_IMAGE_VERSION=1.5.5
Generating certificates using Fabric CA
Creating network "fabric_test" with the default driver
Creating ca_orderer ... done
Creating ca_org2    ... done
Creating ca_org1    ... done
Creating Org1 Identities
Enrolling the CA admin
+ fabric-ca-client enroll -u https://admin:adminpw@localhost:7054 --caname ca-org1 --tls.certfiles /home/radongas/rishi-Hyperledger/fa
bric-samples/test-network/organizations/fabric-ca/org1/ca-cert.pem
2022/10/08 12:07:34 [INFO] Created a default configuration file at /home/radongas/rishi-Hyperledger/fabric-samples/test-network/organi
zations/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
2022/10/08 12:07:34 [INFO] TLS Enabled
2022/10/08 12:07:34 [INFO] generating key: &{A:ecdsa S:256}
2022/10/08 12:07:34 [INFO] encoded CSR
```

This command will deploy the Fabric test network with two peers, an ordering service, and three certificate authorities (Orderer, Org1, Org2). Instead of using the cryptogen tool, we bring up the test network using Certificate Authorities, hence the -ca flag. Additionally, the org admin user registration is bootstrapped when the Certificate Authority is started. In a later step, we will show how the sample application completes the admin enrollment.

Next, let's deploy the chaincode by calling the ./network.sh script with the chaincode name and language options.

```
$ ./network.shdeployCC-ccnbasic-ccp../asset-transfer-basic/chaincode-ja
  vascript/-ccljavascript
```

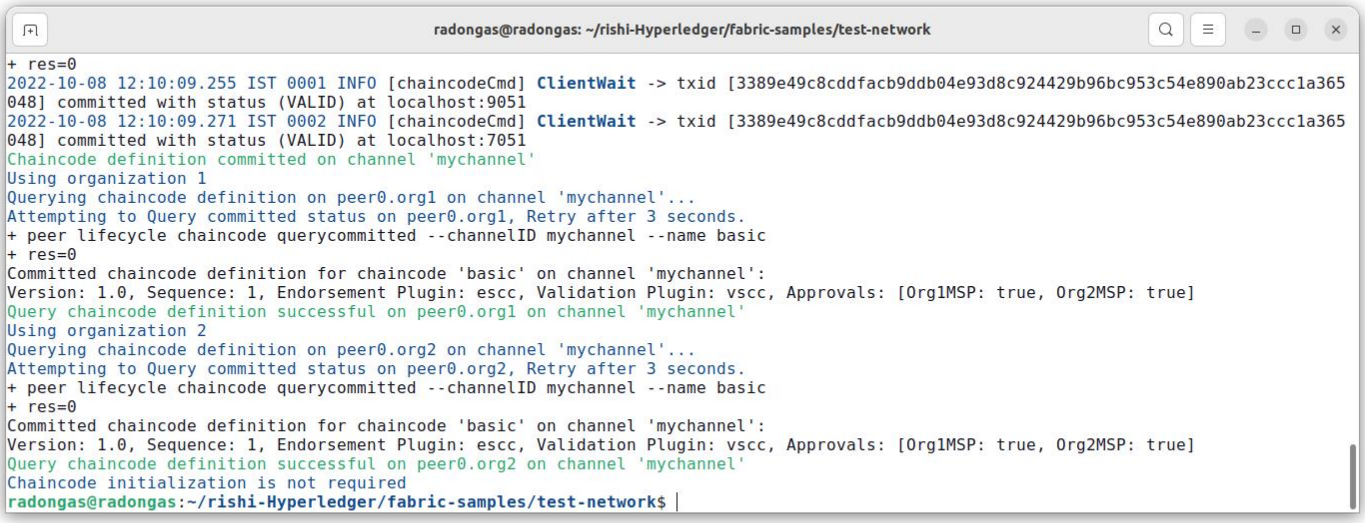


radongas@radongas: ~/rishi-Hyperledger/fabric-samples/test-network

```
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ ./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chain
code-javascript/ -ccl javascript
Using docker and docker-compose
deploying chaincode on channel 'mychannel'
executing with the following
- CHANNEL_NAME: mychannel
- CC_NAME: basic
- CC_SRC_PATH: ../asset-transfer-basic/chaincode-javascript/
- CC_SRC_LANGUAGE: javascript
- CC_VERSION: 1.0
- CC_SEQUENCE: 1
- CC_END_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FCN: NA
- DELAY: 3
- MAX_RETRY: 5
- VERBOSE: false
+ peer lifecycle chaincode package basic.tar.gz --path ../asset-transfer-basic/chaincode-javascript/ --lang node --label basic_1.0
+ res=0
++ peer lifecycle chaincode calculatepackageid basic.tar.gz
+ PACKAGE_ID=basic_1.0:5e683b01b74f2190bd47dd362292adda50ef65bf565e4cbf8dddbf50b0b19351
Chaincode is packaged
Installing chaincode on peer0.org1...
Using organization 1
```

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

If the chaincode is successfully deployed, the end of the output in your terminal should look like below:

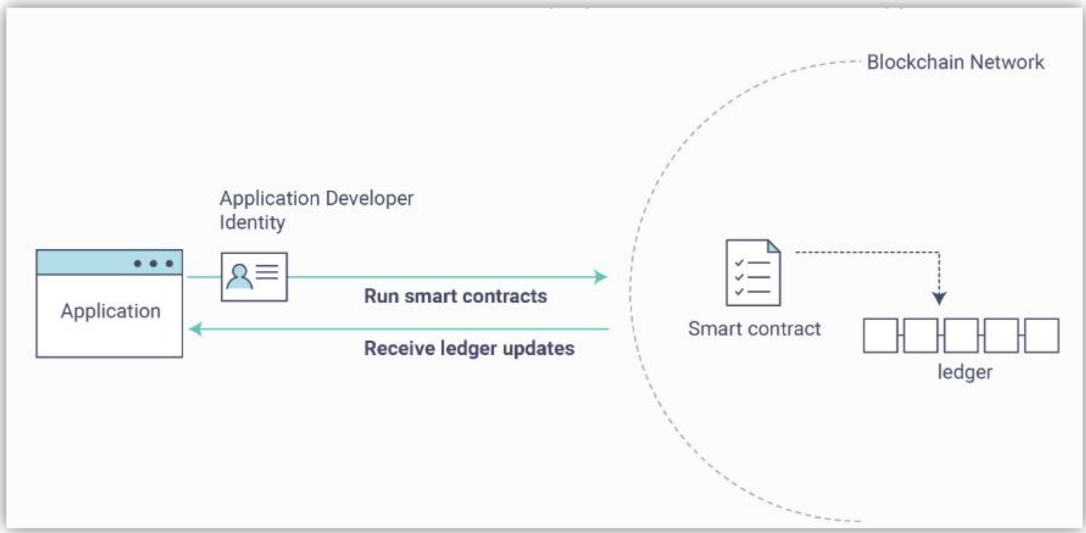


radongas@radongas: ~/rishi-Hyperledger/fabric-samples/test-network

```
+ res=0
2022-10-08 12:10:09.255 IST 0001 INFO [chaincodeCmd] ClientWait -> txid [3389e49c8cddfacb9ddb04e93d8c924429b96bc953c54e890ab23ccc1a365
048] committed with status (VALID) at localhost:9051
2022-10-08 12:10:09.271 IST 0002 INFO [chaincodeCmd] ClientWait -> txid [3389e49c8cddfacb9ddb04e93d8c924429b96bc953c54e890ab23ccc1a365
048] committed 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
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ |
```

Next, let's prepare the sample Asset Transfer Javascript application that will be used to interact with the deployed chaincode.

- JavaScript application



Open a new terminal, and navigate to the application-javascript folder.

```
$ cdasset-transfer-basic/application-javascript
```

This directory contains sample programs that were developed using the Fabric SDK for Node.js. Run the following command to install the application dependencies. It may take up to a minute to complete:

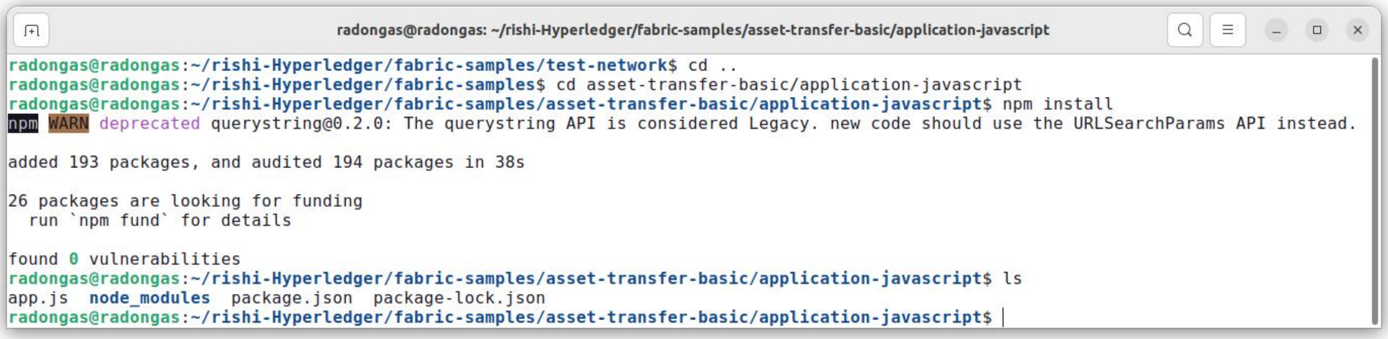
```
$ npminstall
```

This process is installing the key application dependencies defined in the application's package.json. The most important of which is the fabric-network Node.js module; it enables an application to use identities, wallets, and gateways to connect to channels, submit transactions, and wait for notifications. This tutorial also uses the fabric-ca-client module to enroll users with their respective certificate authorities, generating a valid identity which is then used by the fabric-network module to interact with the blockchain network.

Once npm install completes, everything is in place to run the application. Let's take a look at the sample JavaScript application files we will be using in this tutorial. Run the following command to list the files in this directory:

```
$ ls
```

You should see the following:

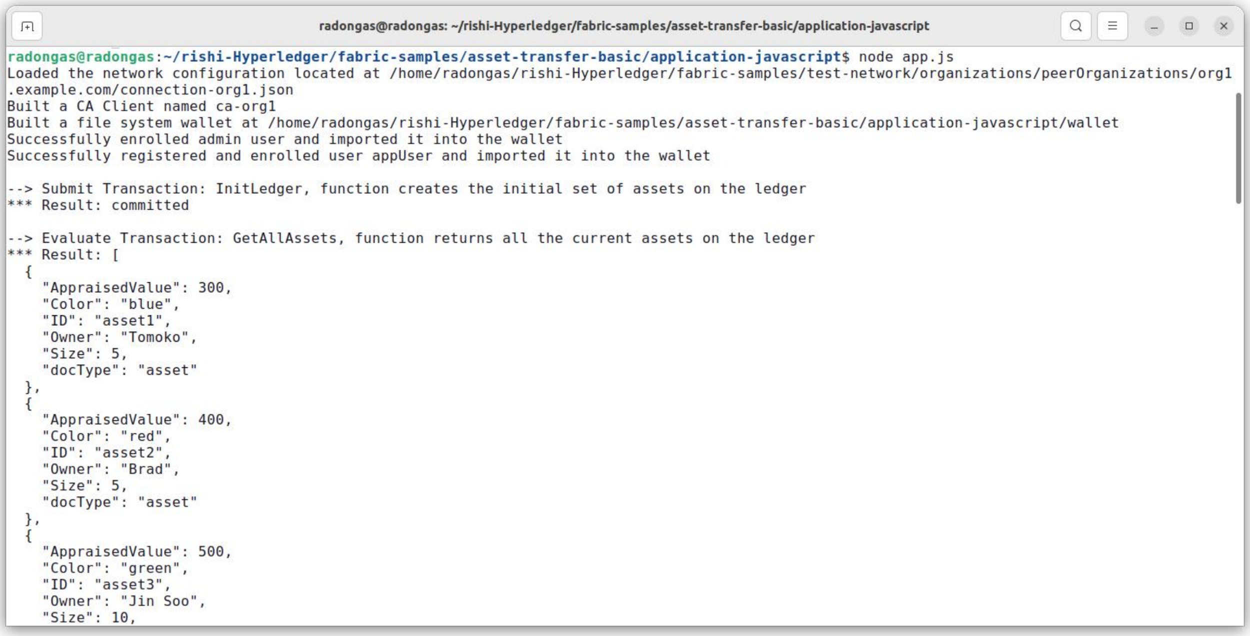


```
radongas@radongas: ~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ cd ..
radongas@radongas:~/rishi-Hyperledger/fabric-samples$ cd asset-transfer-basic/application-javascript
radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript$ npm install
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.

added 193 packages, and audited 194 packages in 38s

26 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript$ ls
app.js  node_modules  package.json  package-lock.json
radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript$ |
```

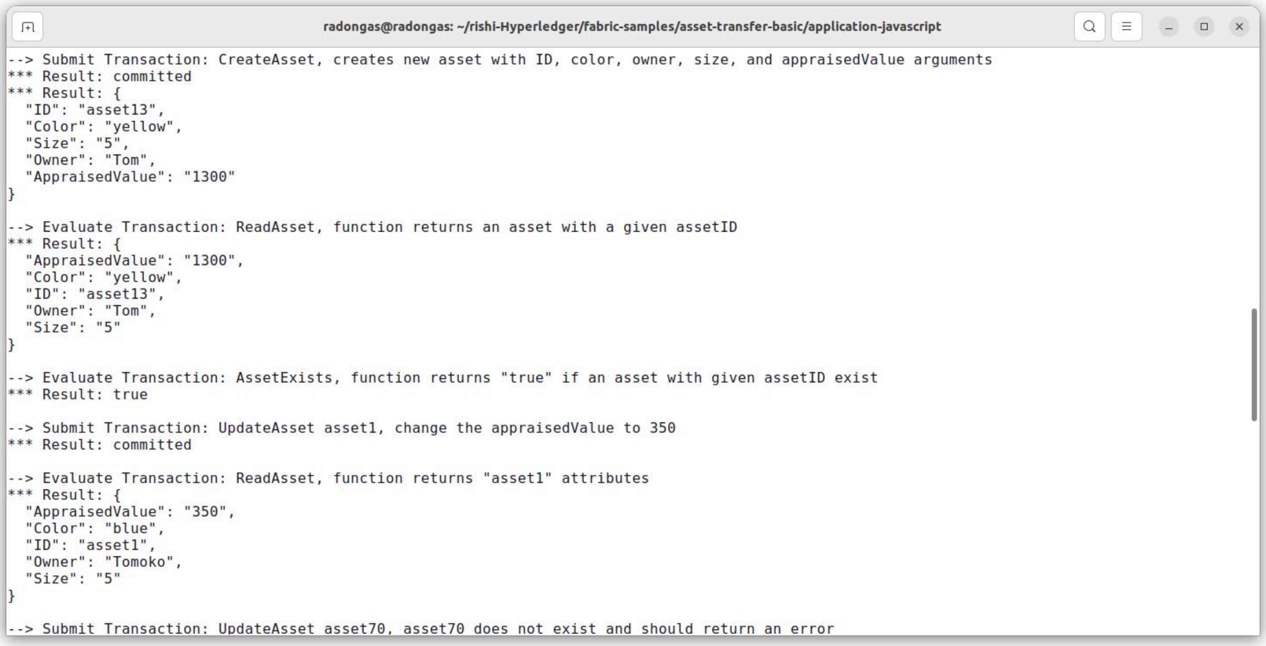
Let's run the application and then step through each of the interactions with the smart contract functions. From the asset-transfer-basic/application-javascript directory, run the following command:

```
$ nodeapp.js
```

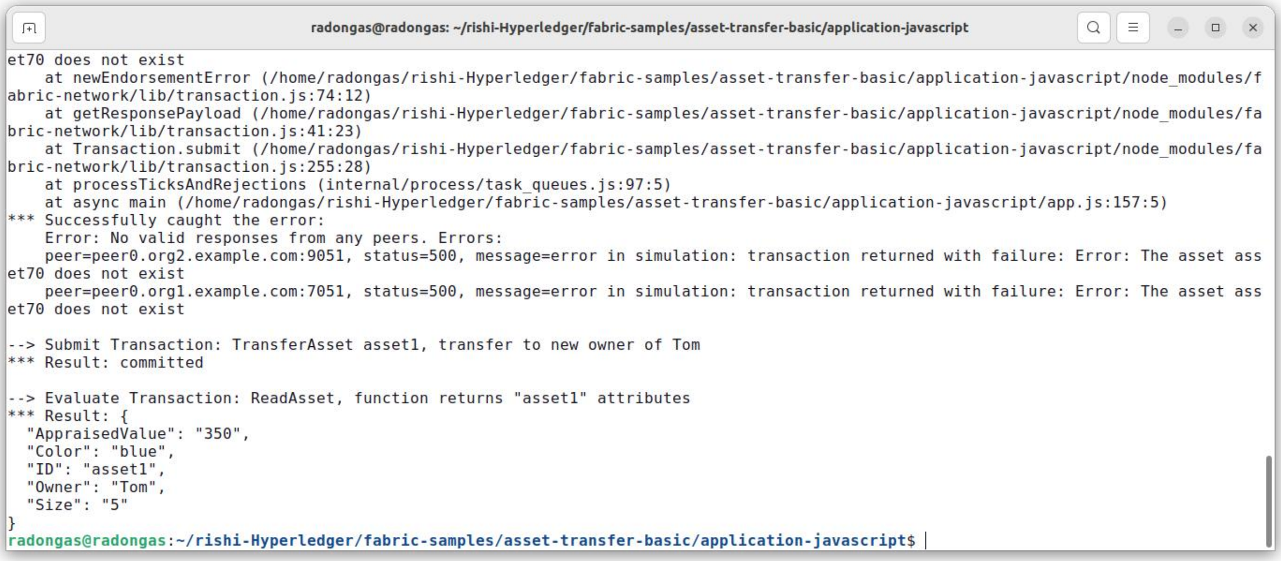
1. First, the application enrolls the admin user.
2. Second, the application registers and enrolls an application user.
3. Third, the sample application prepares a connection to the channel and smart contract.
4. Fourth, the application initializes the ledger with some sample data.
5. Fifth, the application invokes each of the chaincode functions.


radongas@radongas: ~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript

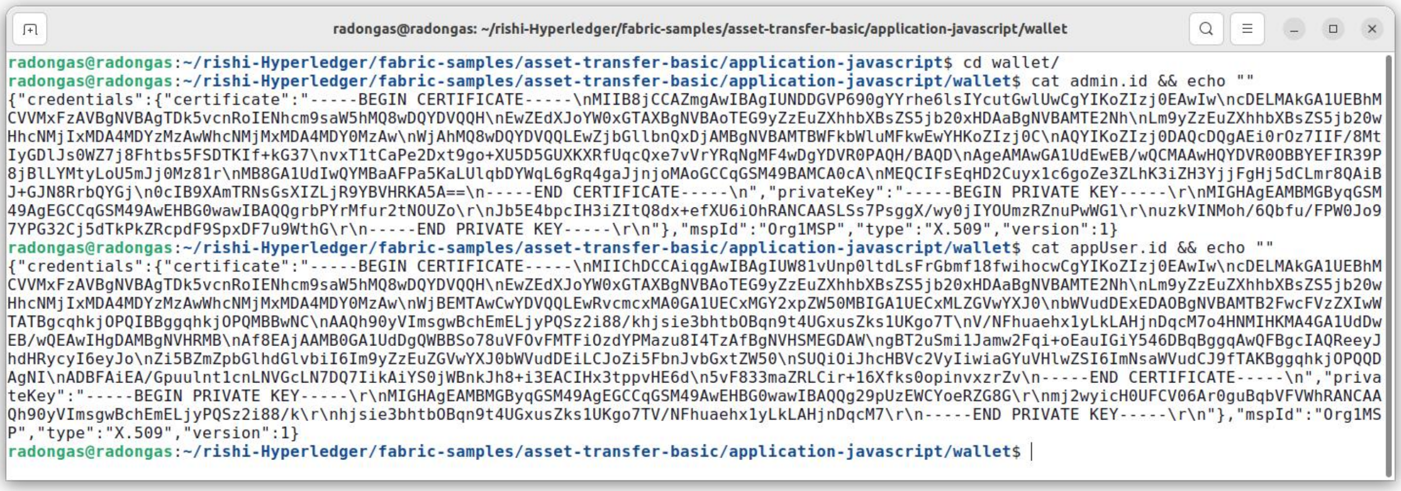
```
radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript$ node app.js
Loaded the network configuration located at /home/radongas/rishi-Hyperledger/fabric-samples/test-network/organizations/peerOrganizations/org1
.example.com/connection-org1.json
Built a CA Client named ca-org1
Built a file system wallet at /home/radongas/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/wallet
Successfully enrolled admin user and imported it into the wallet
Successfully registered and enrolled user appUser and imported it into the wallet

--> Submit Transaction: InitLedger, function creates the initial set of assets on the ledger
*** Result: committed

--> Evaluate Transaction: GetAllAssets, function returns all the current assets on the ledger
*** Result: [
  {
    "AppraisedValue": 300,
    "Color": "blue",
    "ID": "asset1",
    "Owner": "Tomoko",
    "Size": 5,
    "docType": "asset"
  },
  {
    "AppraisedValue": 400,
    "Color": "red",
    "ID": "asset2",
    "Owner": "Brad",
    "Size": 5,
    "docType": "asset"
  },
  {
    "AppraisedValue": 500,
    "Color": "green",
    "ID": "asset3",
    "Owner": "Jin Soo",
    "Size": 10,
```


radongas@radongas: ~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript

```
--> Submit Transaction: CreateAsset, creates new asset with ID, color, owner, size, and appraisedValue arguments
*** Result: committed
*** Result: {
  "ID": "asset13",
  "Color": "yellow",
  "Size": "5",
  "Owner": "Tom",
  "AppraisedValue": "1300"
}

--> Evaluate Transaction: ReadAsset, function returns an asset with a given assetID
*** Result: {
  "AppraisedValue": "1300",
  "Color": "yellow",
  "ID": "asset13",
  "Owner": "Tom",
  "Size": "5"
}

--> Evaluate Transaction: AssetExists, function returns "true" if an asset with given assetID exist
*** Result: true

--> Submit Transaction: UpdateAsset asset1, change the appraisedValue to 350
*** Result: committed

--> Evaluate Transaction: ReadAsset, function returns "asset1" attributes
*** Result: {
  "AppraisedValue": "350",
  "Color": "blue",
  "ID": "asset1",
  "Owner": "Tomoko",
  "Size": "5"
}

--> Submit Transaction: UpdateAsset asset70, asset70 does not exist and should return an error
```



```
et70 does not exist
    at newEndorsementError (/home/radongas/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/node_modules/f
abric-network/lib/transaction.js:74:12)
    at getResponsePayload (/home/radongas/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/node_modules/fa
bric-network/lib/transaction.js:41:23)
    at Transaction.submit (/home/radongas/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/node_modules/fa
bric-network/lib/transaction.js:255:28)
    at processTicksAndRejections (internal/process/task_queues.js:97:5)
    at async main (/home/radongas/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/app.js:157:5)
*** Successfully caught the error:
    Error: No valid responses from any peers. Errors:
    peer=peer0.org2.example.com:9051, status=500, message=error in simulation: transaction returned with failure: Error: The asset ass
et70 does not exist
    peer=peer0.org1.example.com:7051, status=500, message=error in simulation: transaction returned with failure: Error: The asset ass
et70 does not exist

--> Submit Transaction: TransferAsset asset1, transfer to new owner of Tom
*** Result: committed

--> Evaluate Transaction: ReadAsset, function returns "asset1" attributes
*** Result: {
  "AppraisedValue": "350",
  "Color": "blue",
  "ID": "asset1",
  "Owner": "Tom",
  "Size": "5"
}
radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript$ 
```



```
radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript$ cd wallet/
radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/wallet$ cat admin.id && echo ""
{"credentials":{"certificate":"-----BEGIN CERTIFICATE-----\nMIIB8jCCAZmgAwIBAgIUNDDGVP690gYYrhe6lsIYcutGwlUwCgYIKoZIzj0EAwIw\ncDELMAkGA1UEBhM
CVVMxFzAVBgNVBAgTDk5vcnRoIENhcm9saW5hMQ8wDQYDVQQH\nEwZEdXJoYW0xGTAXBgNVBAoTEG9yZzEuZXhhbXBsZS5jb20xHDAaBgNVBAMTE2Nh\nLm9yZzEuZXhhbXBsZS5jb20w
HhcNMjIxMDA4MDYzMzAwWhcNMjMxMDA4MDY0MzAw\nWjAhMQ8wDQYDVQQLEwZjbGllbnQxDjAMBgNVBAMTBWFkbWluMFkwEwYHKoZIzj0C\nAQYIKoZIzj0DAQcDQgAEi0rOz7IIF/8Mt
IyGDlJs0WZ7j8Fhtbs5FSDTKIf+kG37\nvxT1tCaPe2Dxt9go+XU5D5GUXKXRfUqcQxe7vVrYRqNgMF4wDgYDVR0PAQH/BAQD\nAgeAMAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFIR39P
8jBlLYMtyLoU5mJj0Mz81r\nMB8GA1UdIwQYMBaAFPa5KaLUlqbDYWqL6gRq4gaJjnjoMAoGCCqGSM49BAMCA0cA\nMEQCIFsEqHD2Cuyx1c6goZe3ZLhK3iZH3YjjFgHj5dCLmr8QAiB
J+GJN8RrbQYGj\n0cIB9XAmTRNsGsXIZLjR9YBVHRKA5A==\n-----END CERTIFICATE-----\n","privateKey":"-----BEGIN PRIVATE KEY-----\r\nMIGHAgEAMBMGByqGSM
49AgEGCCqGSM49AwEHBG0wawIBAQQgrbPYrMfur2tNOUZo\r\nJb5E4bpcIH3iZItQ8dx+efXU6iOhRANCAASLSs7PsggX/wy0jIYOUmzRZnuPwWG1\r\nuzkVINMoh/6Qbfu/FPW0Jo9
7YPG32Cj5dTkPkZRcpdF9SpxDF7u9WthG\r\n-----END PRIVATE KEY-----\r\n"},"mspId":"Org1MSP","type":"X.509","version":1}
radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/wallet$ cat appUser.id && echo ""
{"credentials":{"certificate":"-----BEGIN CERTIFICATE-----\nMIIChDCCAiqgAwIBAgIUW81vUnp0ltdLsFrGbmf18fwihocwCgYIKoZIzj0EAwIw\ncDELMAkGA1UEBhM
CVVMxFzAVBgNVBAgTDk5vcnRoIENhcm9saW5hMQ8wDQYDVQQH\nEwZEdXJoYW0xGTAXBgNVBAoTEG9yZzEuZXhhbXBsZS5jb20xHDAaBgNVBAMTE2Nh\nLm9yZzEuZXhhbXBsZS5jb20w
HhcNMjIxMDA4MDYzMzAwWhcNMjMxMDA4MDY0MzAw\nWjBEMTAwCwYDVQQLEwRvcmcxMA0GA1UECxMGY2xpZW50MBIGA1UECxMLZGVwYXJ0\nbWVudDExEDAOBgNVBAMTB2FwcFVzZXIwW
TATBgcqhkjOPQIBBggqhkjOPQMBBwNC\nAAQh90yVImsgwBchEmELjyPQSz2i88/khjsie3bhtbOBqn9t4UGxusZks1UKgo7T\nV/NFhuaehx1yLkLAHjnDqcM7o4HNMIHKMA4GA1UdDw
EB/wQEAwIHgDAMBgNVHRMB\nAf8EAjAAMB0GA1UdDgQWBBSo78uVFOvFMTFiOzdYPMazu8I4TzAfBgNVHSMEGDAW\ngBT2uSmi1Jamw2Fqi+oEauIGiY546DBqBggqAwQFBgcIAQReeyJ
hdHRycyI6eyJo\nZi5BZmZpbGlhdGlvbiI6Im9yZzEuZGVwYXJ0bWVudDEiLCJoZi5FbnJvbGxtZW50\nSUQiOiJhcHBVc2VyIiwiaGYuVHlwZSI6ImNsaWVudCJ9fTAKBggqhkjOPQQD
AgNI\nADBFAiEA/Gpuulnt1cnLNVGcLN7DQ7IikAiYS0jWBnkJh8+i3EACIHx3tppvHE6d\n5vF833maZRLCir+16Xfks0opinvxzrZv\n-----END CERTIFICATE-----\n","priva
teKey":"-----BEGIN PRIVATE KEY-----\r\nMIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQg29pUzEWCYoeRZG8G\r\nmj2wyicH0UFCV06Ar0guBqbVFVWhRANCAA
Qh90yVImsgwBchEmELjyPQSz2i88/k\r\nhjsie3bhtbOBqn9t4UGxusZks1UKgo7TV/NFhuaehx1yLkLAHjnDqcM7\r\n-----END PRIVATE KEY-----\r\n"},"mspId":"Org1MS
P","type":"X.509","version":1}
radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/wallet$ 
```

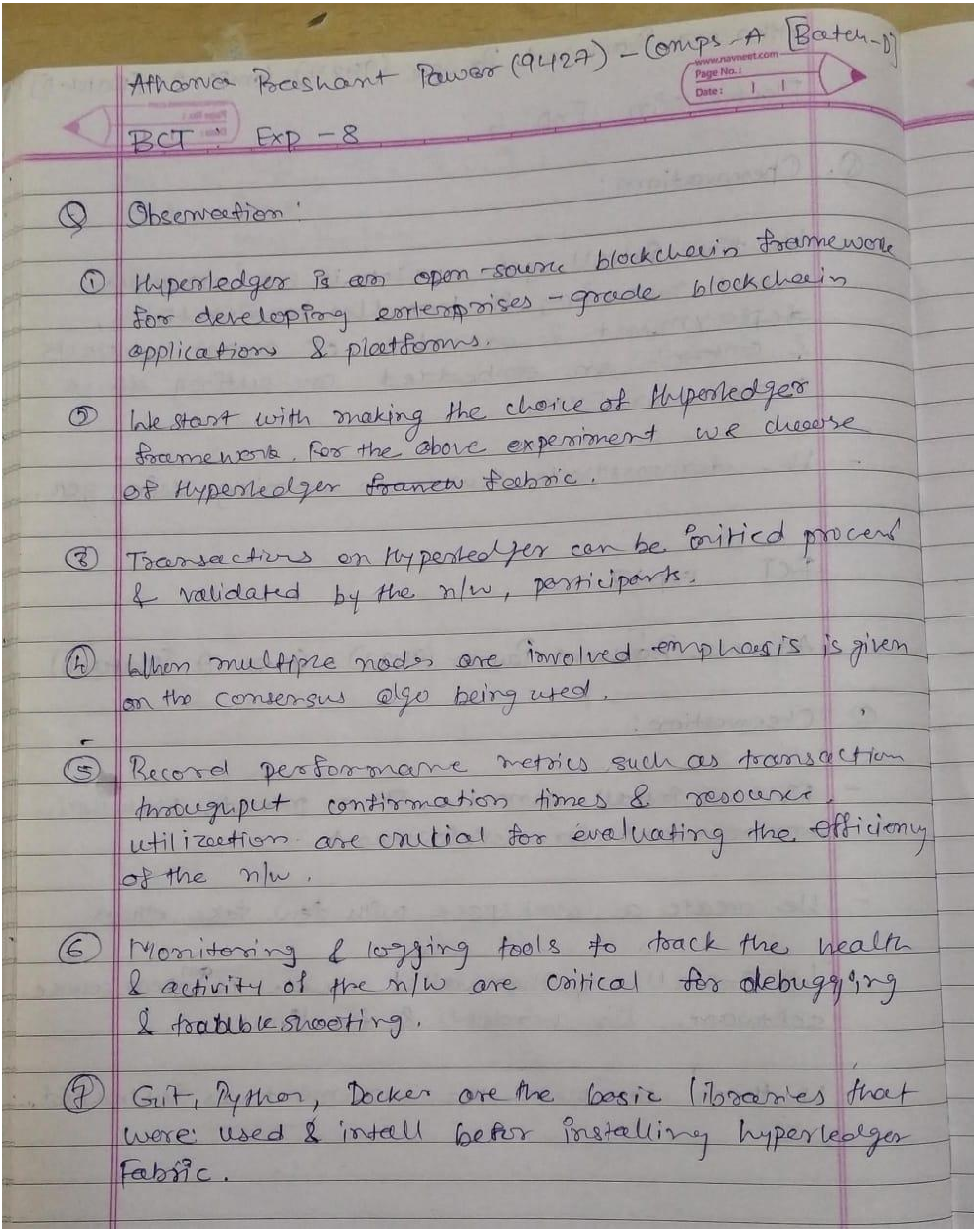
When you are finished using the asset-transfer sample, you can bring down the test network using network.sh script.

```
$ ./network.shdown
```

Conclusion:

Now that we've seen how the sample application and chain code are written and how they interact with each other, we have a pretty good sense of how applications interact with a blockchain network using a smart contract to query or update the ledger. We have seen the basics of the roles smart contracts, APIs, and the SDK play in queries and updates and we should have a feel for how different kinds of applications could be used to perform other business tasks and operations. Thus, we have successfully implemented a sample application on Hyperledger Fabric.

Post Lab:



Atharva Prashant Pawar (9427) - Comps - A [Batch-D]

BCT : Exp - 8

Q Observation:

1. Hyperledger is an open-source blockchain framework for developing enterprises - grade blockchain applications & platforms.

2. We start with making the choice of Hyperledger framework. For the above experiment we choose of Hyperledger ~~framew~~ Fabric.

3. Transactions on Hyperledger can be critical process & validated by the n/w, participants.

4. When multiple nodes are involved emphasis is given on the consensus algo being used.

5. Record performance metrics such as transaction throughput confirmation times & resource utilization are critical for evaluating the efficiency of the n/w.

6. Monitoring & logging tools to track the health & activity of the n/w are critical for debugging & troubleshooting.

7. Git, Python, Docker are the basic libraries that were used & intall befor installing hyperledger Fabric.