



Experiment No. 7

Aim:

Case Study on Hyperledger

Theory:

Private Blockchain

What is Private Blockchain?		
Public	Non - Permissioned	Permissioned
Private	Bitcoin Multichain in a small lab, without permissions	Ripple Hyperledger Fabric
PRIVATE BLOCKCHAIN		
ORGANIZATION TYPE	Single entity or organization	
USERS ACCESS	Known & trusted participants 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.	

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 via process, and user rights are created, changed, or deleted by an authorized user. These needs arose 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.

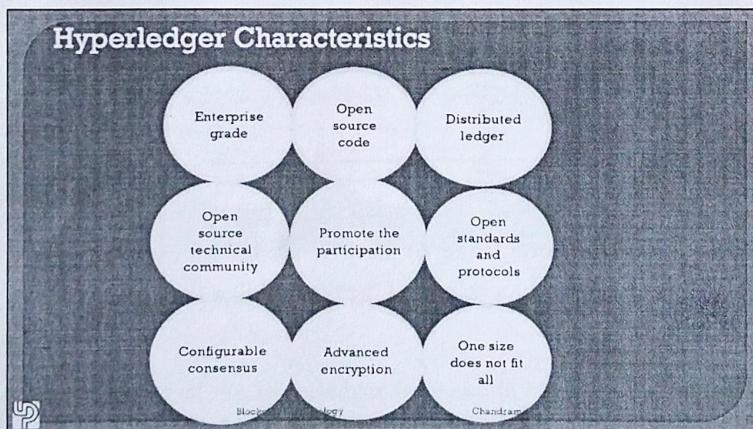
The approach was to ensure that the best practices of computer science related to distributed computing are used in blockchain for enterprise solutions.

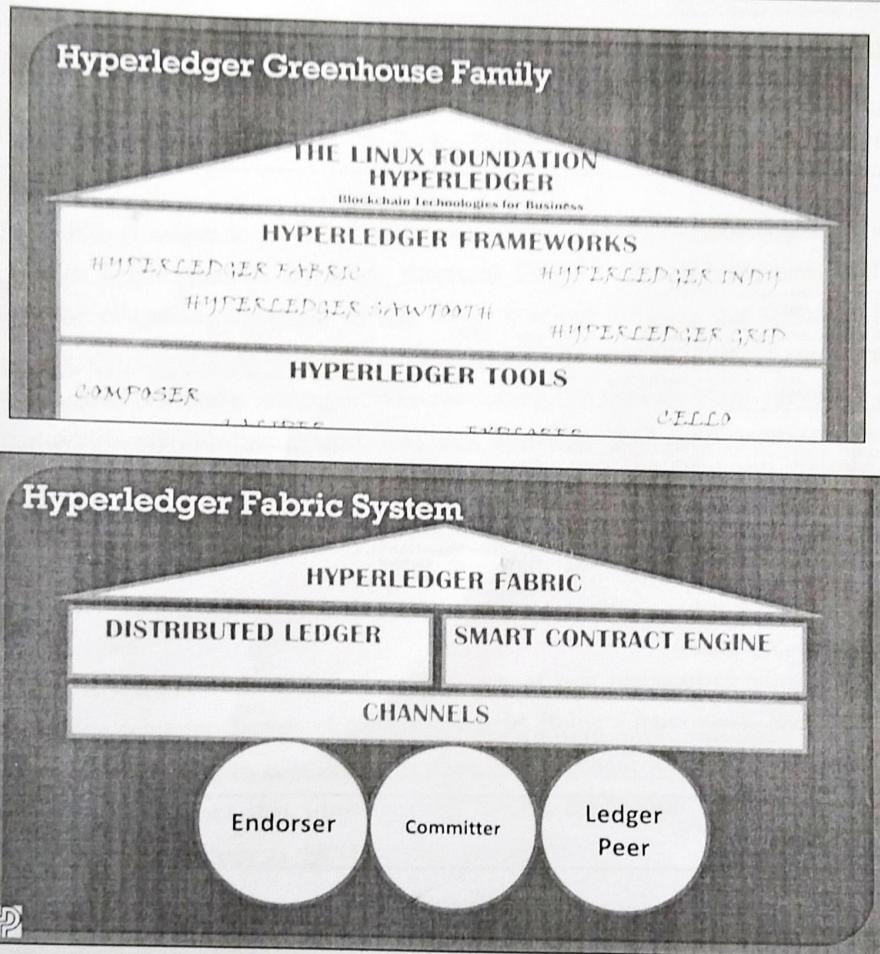
It needs to be noted that Hyperledger has stated they will not be issuing its cryptocurrency.

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 LedgerProject) solution
- promote the participation of leading members of the ecosystem, including developers, service and solution providers and end-users.





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 sample implementation 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:



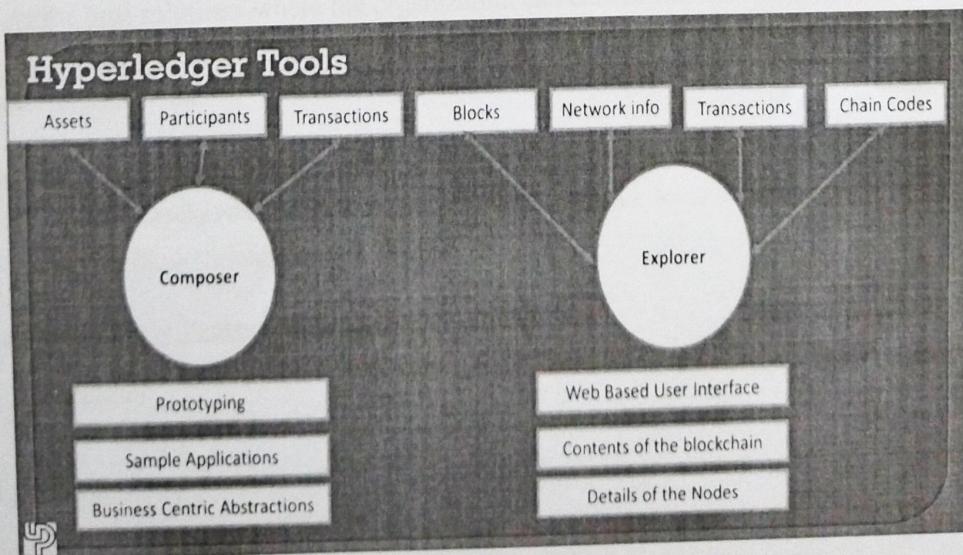
- Privacy by design
- Privacy-preserving technologies

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: 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 : 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>.

Hyperledger 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/stable/amd64/_docker-ce_18.06.3~ce~3-0~ubuntu_amd64.deb
```

```
$ sudo dpkg -i _docker-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
```

```

radongas@radongas:~/rishi-HyperLedger
radongas@radongas:~/rishi-HyperLedgers curl -sSL http://bit.ly/2ysb0FE | 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  Current
          Dload  Upload Total Spent   Left Speed
  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  Current
          Dload  Upload Total Spent   Left Speed
  0     0     0     0      0      0      0:--:--:--:0:00:01 0:--:--: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: sha256:30c361397493d64d5e2de783af7224f50a9f7bdeebf5a0b3dac87aba9327e9c
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-HyperLedgers |

```



Running Hyperledger 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
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
+ resm0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ resm0
Creating Orderer Org identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ resm0
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 ...
Creating peer0.org2.example.com ...
Creating peer0.org1.example.com ...
Creating cli ...
CONTAINER ID IMAGE COMMAND CREATED STATUS NAMES PORTS
761b424f71f5 hyperledger/fabric-tools:latest "/bin/bash" 1 second ago Up Less than a second cli
9d717fe7f37 hyperledger/fabric-peer:latest "peer node start" 2 seconds ago Up Less than a second 0.0.0.0:7051->7051/tcp, ::1:7051->7051/tcp, 0.0.0.0:9444->9444/tcp, ::1:9444->9444/tcp
7dbe57e181d1 hyperledger/fabric-orderer:latest "orderer" 2 seconds ago Up Less than a second 0.0.0.0:7050->7050/tcp, ::1:7050->7050/tcp, 0.0.0.0:7053->7053/tcp, ::1:7053->7053/tcp, 0.0.0.0:9443->9443/tcp, ::1:9443->9443/tcp
fed9af21cf5 hyperledger/fabric-peer:latest "peer node start" 2 seconds ago Up Less than a second 0.0.0.0:9051->9051/tcp, ::1:9051->9051/tcp, 0.0.0.0:9445->9445/tcp, ::1:9445->9445/tcp
radongas@radongas:~/rishi-HyperLedger/fabric-samples/test-networks |
```

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
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
540c6d95b1a 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, ::1:7051->7051/tcp, 0.0.0.0:9444->9444/tcp, ::1:9444->9444/tcp
3f7964ab3697 hyperledger/fabric-orderer:latest "orderer" 10 seconds ago Up 8 seconds 0.0.0.0:7050->7050/tcp, ::1:7050->7050/tcp, 0.0.0.0:7053->7053/tcp, ::1:7053->7053/tcp, 0.0.0.0:9443->9443/tcp, ::1:9443->9443/tcp
71ab76a85476 hyperledger/fabric-peer:latest "peer node start" 10 seconds ago Up 8 seconds 0.0.0.0:9051->9051/tcp, ::1:9051->9051/tcp, 0.0.0.0:9445->9445/tcp, ::1:9445->9445/tcp
radongas@radongas:~/rishi-HyperLedger/fabric-samples/test-networks |
```



Vidyavardhini's College of Engineering and Technology, Vasai

Department of Computer Science & Engineering (Data Science)

to

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
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-networks |
```

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
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-networks | sudo docker exec peer0.org1.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-networks |
```

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

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

Working with State DataBase (Couch DB):

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$ sudo ./network.sh createChannel -c testchannel
Using docker and docker-compose
Creating channel 'testchannel'.
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 'testchannel.block'
/home/radongas/rishi-Hyperledger/fabric-samples/test-network/../bin/configtxgen
+ configtxgen -profile TwoOrgsApplicationGenesis -outputBlock ./channel-artifacts/testchannel.block -channelID testchannel
2022-10-08 14:30:44.584 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 unset, 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-Hyperledger/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 testchannel
Using organization 1
+ osnadmin channel join --channelID testchannel --config-block ./channel-artifacts/testchannel.block -o localhost:7053 --ca-file /home/radongas/rishi-Hyperledger/fabric-samples/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --client-cert /home/radongas/rishi-Hyperledger/fabric-samples/test-network/organizations/ordererOrganizations/example.com
```

```
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						
COMMAND	CREATED	STATUS	PORTS NAMES			
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						
200a692c271d hyperledger/fabric-tools:latest	/bin/bash	1 second ago	Up Less than a second			
b3alabaab555 hyperledger/fabric-peer:latest	"peer node start"	2 seconds ago	Up Less than a second			
9051/tcp, ::::9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp, ::::9445->9445/tcp						
mple.com						
c9a3786b2bdf hyperledger/fabric-peer:latest	"peer node start"	2 seconds ago	Up 1 second			
7051/tcp, ::::7051->7051/tcp, 0.0.0.0:9444->9444/tcp, ::::9444->9444/tcp						
mple.com						
91c12bebce6e hyperledger/fabric-orderer:latest	"orderer"	3 seconds ago	Up 2 seconds			
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						
e.com						
51ed55cdbc42 couchdb:3.1.1	"tini -- /docker-ent_"	3 seconds ago	Up 1 second			
/tcp, 0.0.0.0:7984->5984/tcp, ::::7984->5984/tcp						
41ddd924793c couchdb:3.1.1	"tini -- /docker-ent_"	4 seconds ago	Up 1 second			
/tcp, 0.0.0.0:5984->5984/tcp, ::::5984->5984/tcp						
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-networks						

Step 4: Create new channel by using below command

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

This will create a new channel with the name testchannel1.

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.

```
$ cd fabric-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.sh down
```

```
radongas@radongas: ~/rishi-Hyperledger/fabric-samples/test-network
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network$ cd fabric-samples/test-network
Using docker and docker-compose
Stopping network
Stopping 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.



Vidyavardhini's College of Engineering and Technology, Vasai

Department of Computer Science & Engineering (Data Science)

14

```
$ ./network.sh up createChannel -c mychannel -ca
```

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

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

```
radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network
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, approvechaincode for both Org1 and Org2, and finally commit the chaincode.

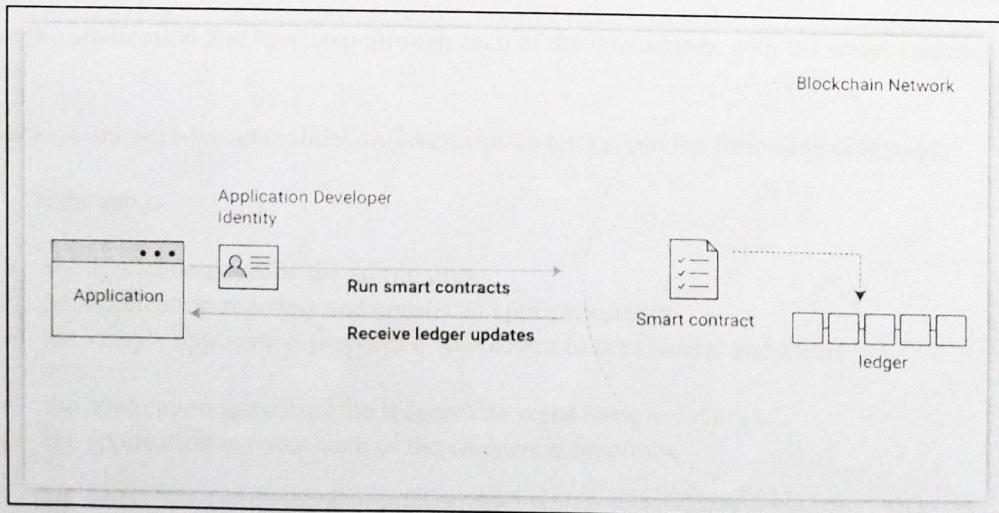


If the chaincode is successfully deployed, the end of the output in your terminal should look like below:

```
[root@radongas ~]# radongas@radongas:~/rishi-Hyperledger/fabric-samples/test-network
+ res=0
2022-10-08 12:10:09.255 IST 0001 INFO [chaincodeCmd] ClientWait -> txid [3389e49c8cddfacb9ddb04e93d8c924429b96bc953c54e890ab23cc1a365
048] committed with status [VALID] at localhost:7051
2022-10-08 12:10:09.271 IST 0002 INFO [chaincodeCmd] ClientWait -> txid [3389e49c8cddfacb9ddb04e93d8c924429b96bc953c54e890ab23cc1a365
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]
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
[root@radongas ~]#
```

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.



```
$ cd asset-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:

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@redongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript$ cd ..
radongas@redongas:~/rishi-Hyperledger/fabric-samples$ cd asset-transfer-basic/application-javascript
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@redongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript$ ls
app.js  node_modules  package.json  package-lock.json
radongas@redongas:~/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:

```
$ node app.js
```

- the application enrolls the admin user.
- the application registers and enrolls an application user.
- the sample application prepares a connection to the channel and smart contract.
- the application initializes the ledger with some sample data.
- the application invokes each of the chaincode functions.



Loaded the network configuration located at /home/radongas/rishi-Hyperledger/fabric-samples/test-network/organizations/peerOrganizations/o

Built a `l1s` smart contract at `home` using Hyperledger `IJab1c-samples/assets/asset-transfer-basics/applications/javascript`. It was deployed successfully and imported into the wallet.

- Submit Transaction: `InitLedger` , function creates the initial set of assets on the ledger.

| evaluate Transaction:

```
"ID": "asset1",  
"Owner": "Tomoko",  
"Size": 5
```

"Size": 5

"AppraisedValue": 500

13

'Size': 10

```
[2] radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/wallet$ radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/wallet$ cd wallet/
[2] radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/wallet$ cat admin_id.js && echo ""
[2] {"credentials": {"certificate": "..."}, "privateKey": "..."}, "END CERTIFICATE", "..."}, "END PRIVATE KEY", "..."}, "mspld": "orgJMS", "version": "1"}]
[2] radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/wallet$ cat admin_id.js && echo ""
[2] {"credentials": {"certificate": "..."}, "privateKey": "..."}, "END CERTIFICATE", "..."}, "END PRIVATE KEY", "..."}, "mspld": "orgJMS", "version": "1"}]
[2] radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/wallet$ cat user_id.js && echo ""
[2] {"credentials": {"certificate": "..."}, "privateKey": "..."}, "END CERTIFICATE", "..."}, "END PRIVATE KEY", "..."}, "mspld": "orgJMS", "version": "1"}]
[2] radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/wallet$ cat user_id.js && echo ""
[2] {"credentials": {"certificate": "..."}, "privateKey": "..."}, "END CERTIFICATE", "..."}, "END PRIVATE KEY", "..."}, "mspld": "orgJMS", "version": "1"}]
[2] radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/wallet$ cat user_id.js && echo ""
[2] {"credentials": {"certificate": "..."}, "privateKey": "..."}, "END CERTIFICATE", "..."}, "END PRIVATE KEY", "..."}, "mspld": "orgJMS", "version": "1"}]
[2] radongas@radongas:~/rishi-Hyperledger/fabric-samples/asset-transfer-basic/application-javascript/wallet$ cat user_id.js && echo ""
[2] {"credentials": {"certificate": "..."}, "privateKey": "..."}, "END CERTIFICATE", "..."}, "END PRIVATE KEY", "..."}, "mspld": "orgJMS", "version": "1"}]
```

When you are finished using the asset-transfer sample, you can bring down the test network by running the command:

Using network.sh script

Conclusion

Q. Can you provide a real-world case study of how a company implemented Hyperledger technology in its business operations, and what benefits did it achieve?