# Department of Computer Engineering Academic Term: Jan-May 23-24

**Class: B.E Computer Sem -VII Subject: Blockchain Technology Lab Subject 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 :**

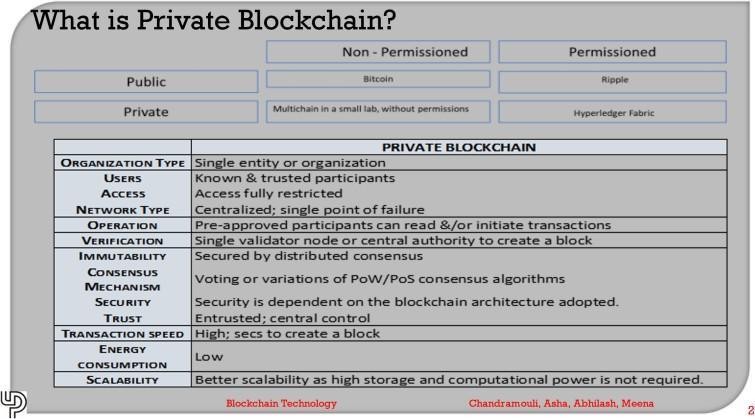
# Experiment No.8

**Case Study on Hyperledger**

**Aim:** Case Study on Hyperledger

# Theory:

Private Blockchain



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:

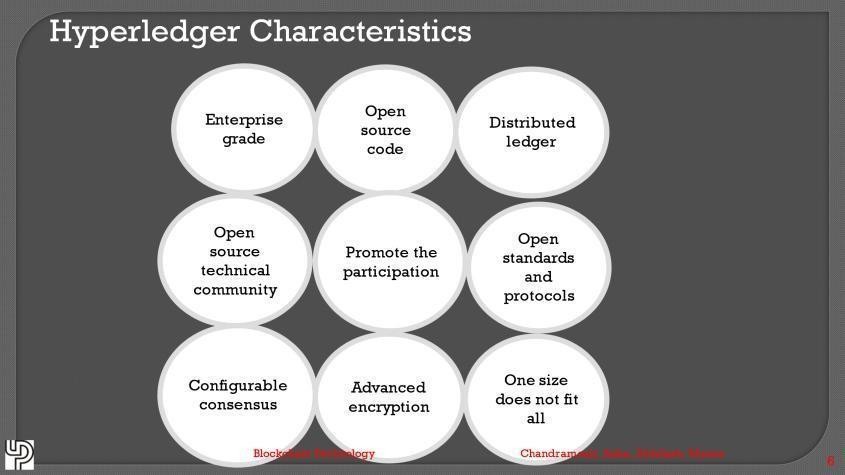
The Linux Foundation took up the challenge for an open-source enterprise-grade distributed ledger technology

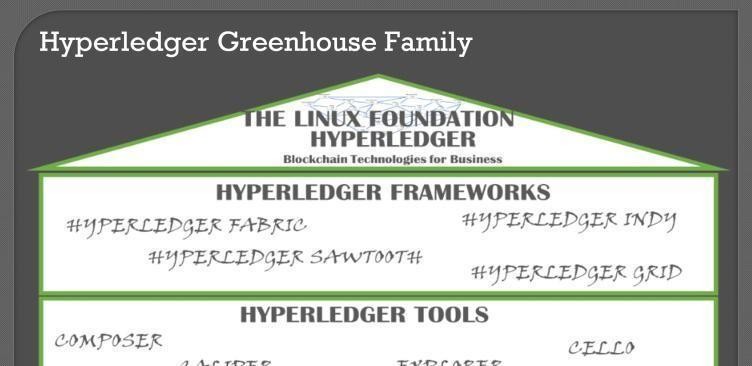
and announced the Hyperledger Project in December 2015.

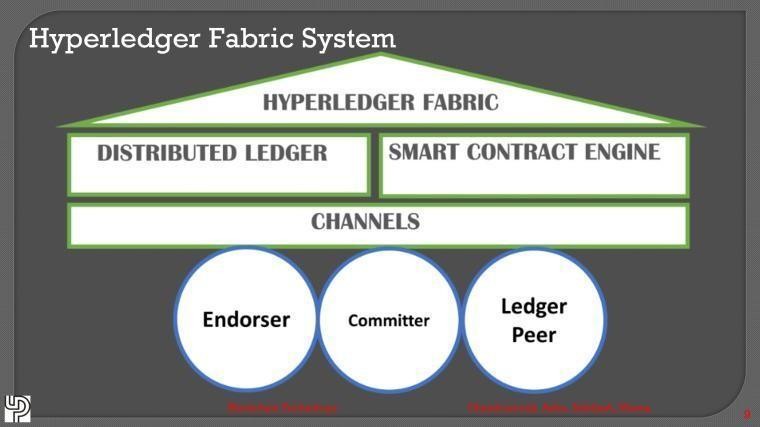
* + The approach was to ensure that the best practices of computer science related to distributed computing are used in blockchain for enterprise solutions.
  + It needs to be noted that Hyperledger has stated they will not be issuing its cryptocurrency

# 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 Indy:

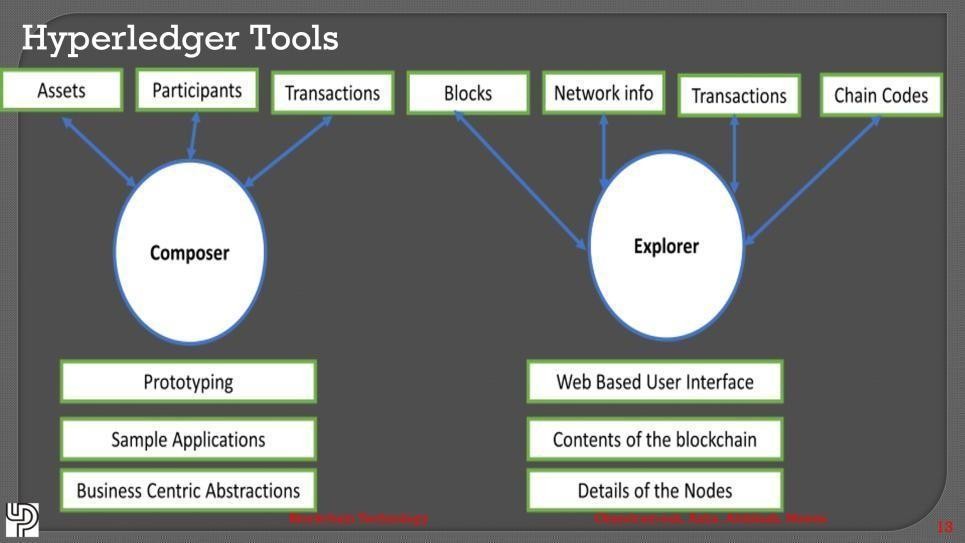
* + - Hyperledger INDY, a part of the hyperledger framework, is a blockchain tool for digital identity. An organization called sovrin.org donated the code base for INDY.
    - INDY is known for providing digital identities in a decentralized environment. INDY provides tools, libraries, and reusable components 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:

# Explorer

* 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:

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](https://download.docker.com/linux/ubuntu/dists/xenial/pool/stable/amd64/docker-ce_18.06.3~ce~3-0~ubuntu_amd64.deb)

$ sudodpkg -i[docker-ce\_18.06.3~ce~3-0~ubuntu\_amd64.deb](https://download.docker.com/linux/ubuntu/dists/xenial/pool/stable/amd64/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

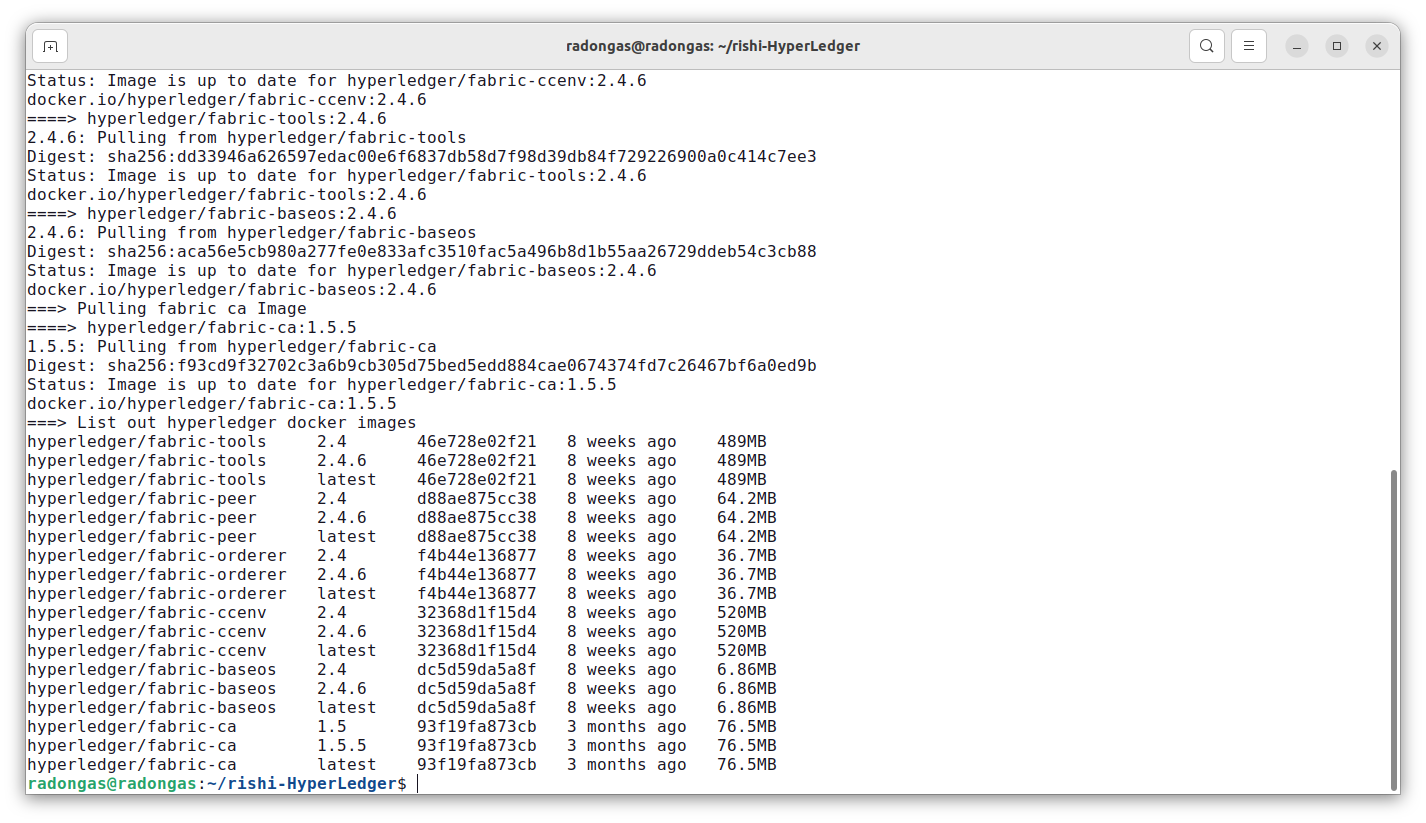
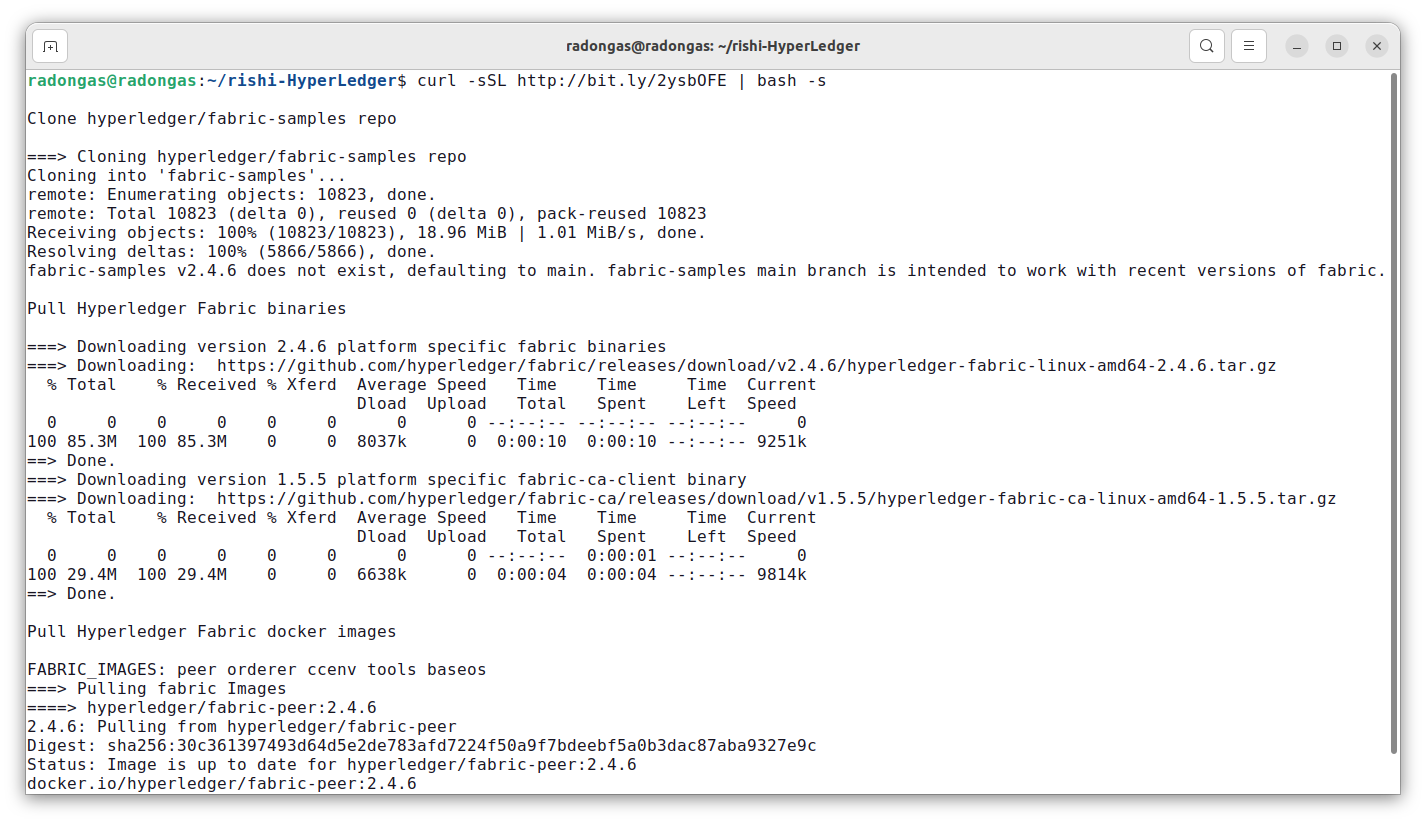
Verify the installation and check the version from below command.

$ docker-compose version

## Hyperledger Installation:

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

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



## Running Hyperleder Fabric Testnetwork:

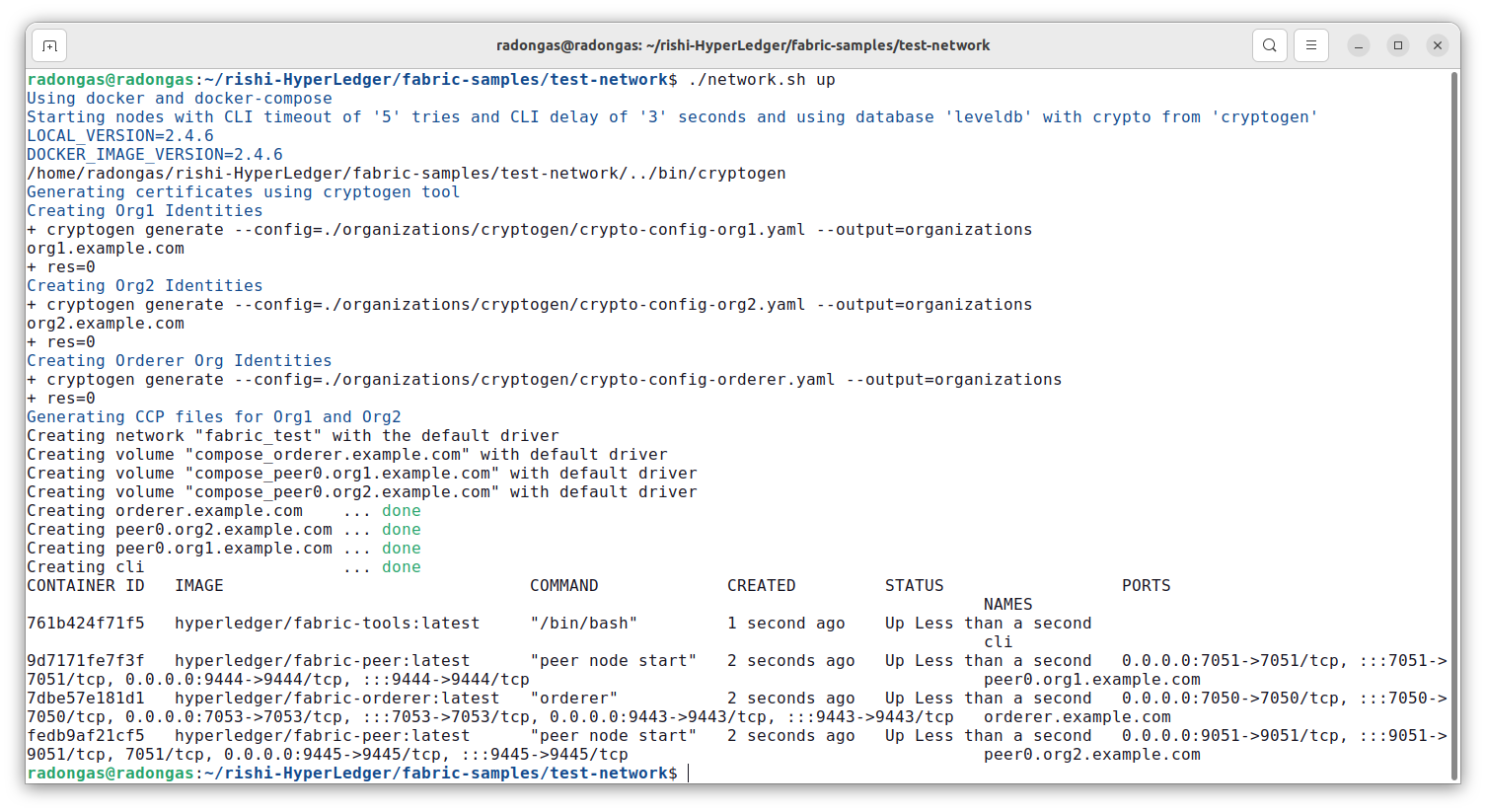
Step 1: Go to fabric-samples folder by using below command.

$ cd fabric-samples

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

$ cd test-network

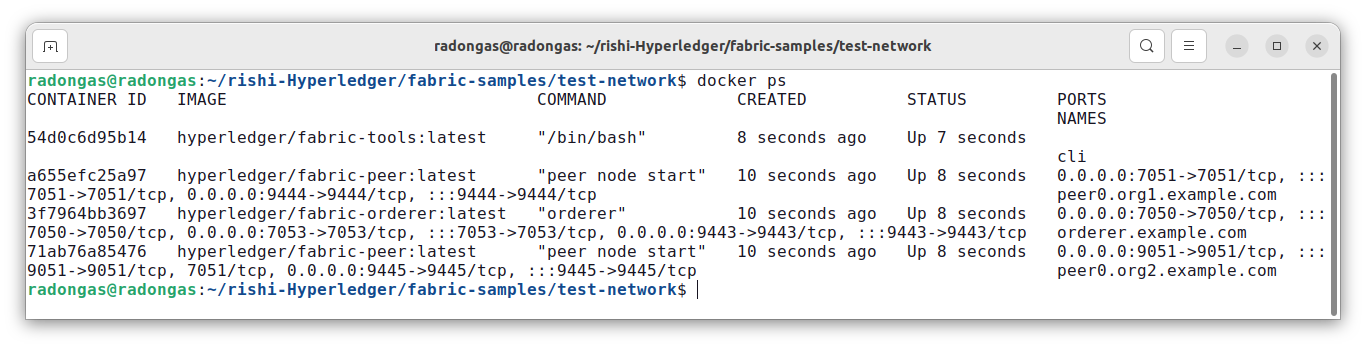
Step 3: Run below command to start your test-network

$ sudo ./network.sh up

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

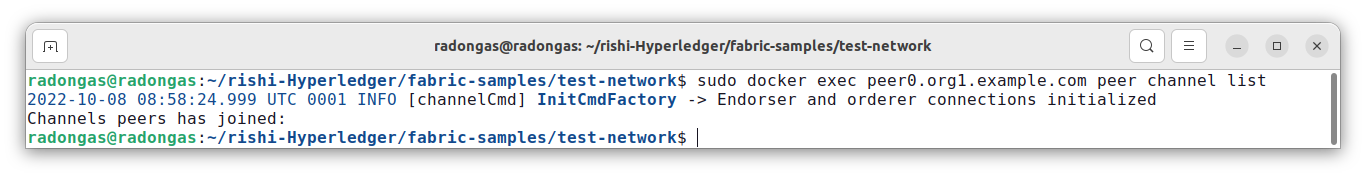
$ sudo docker ps

This shows you three docker containers

* One for Org1 peer node
* One for Org2 peer node
* One for Orderer

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.

Step 4: Create new channel by using below command.

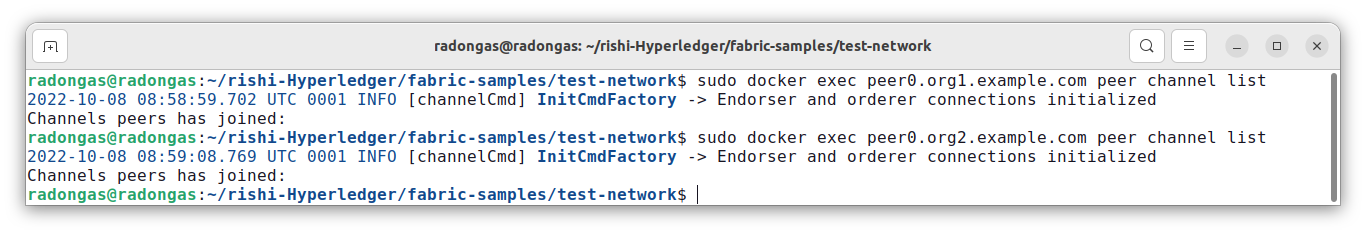
$ sudo ./network.sh createChannel -c testchannel

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



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.

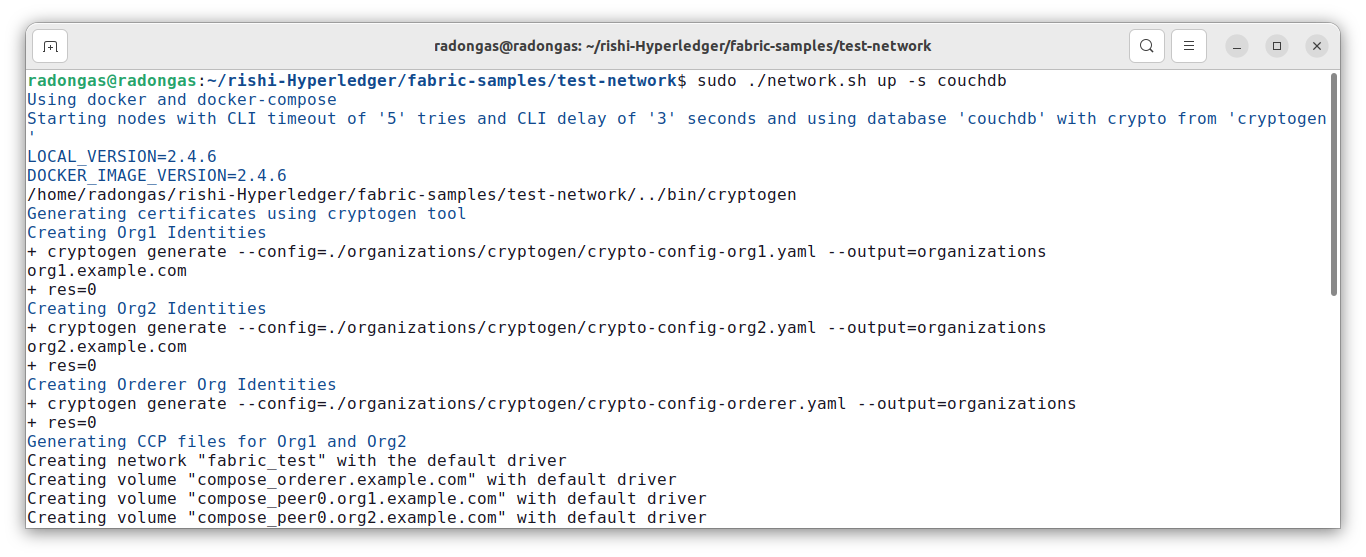
$ cd fabric-samples

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

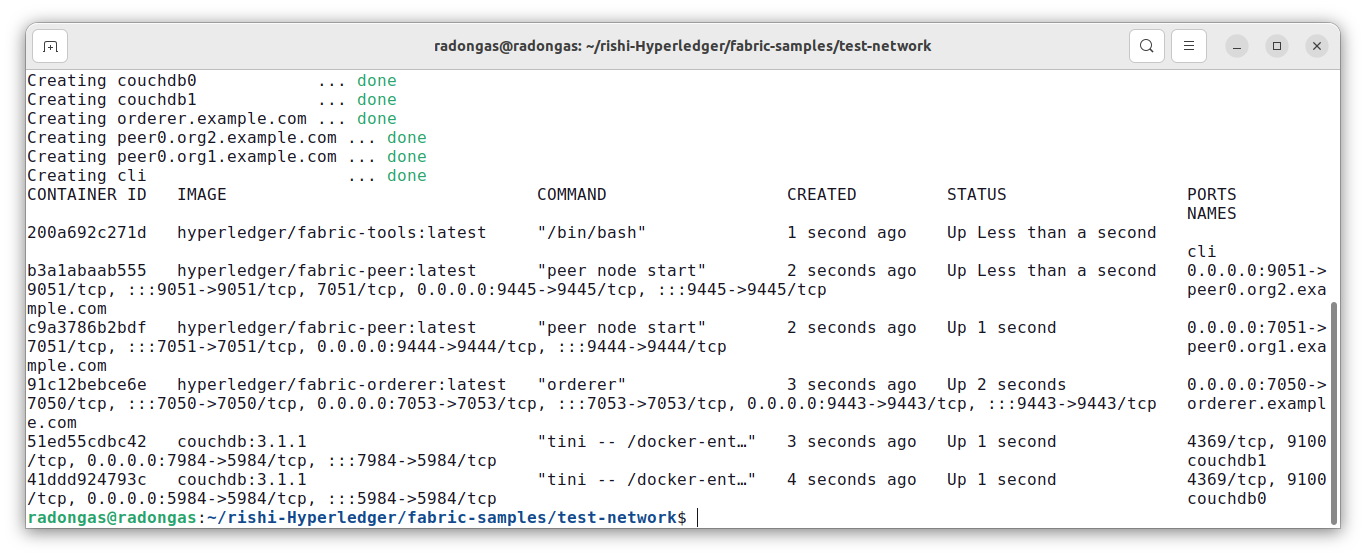
$ cd test-network

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

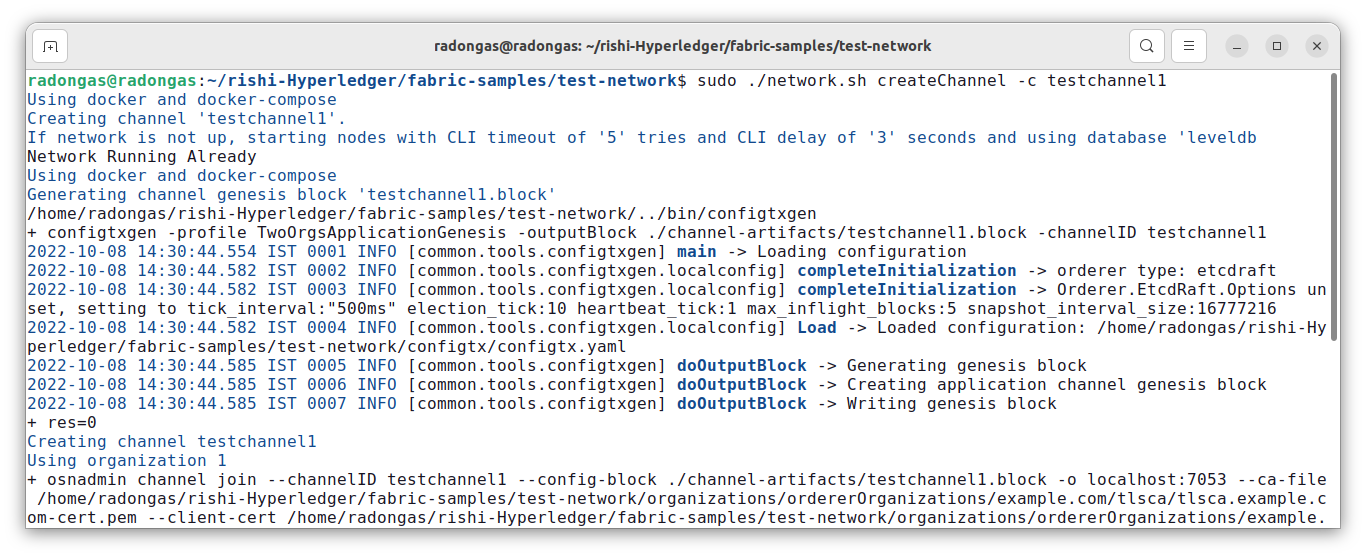
$ sudo ./network.sh up -s couchdb



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



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

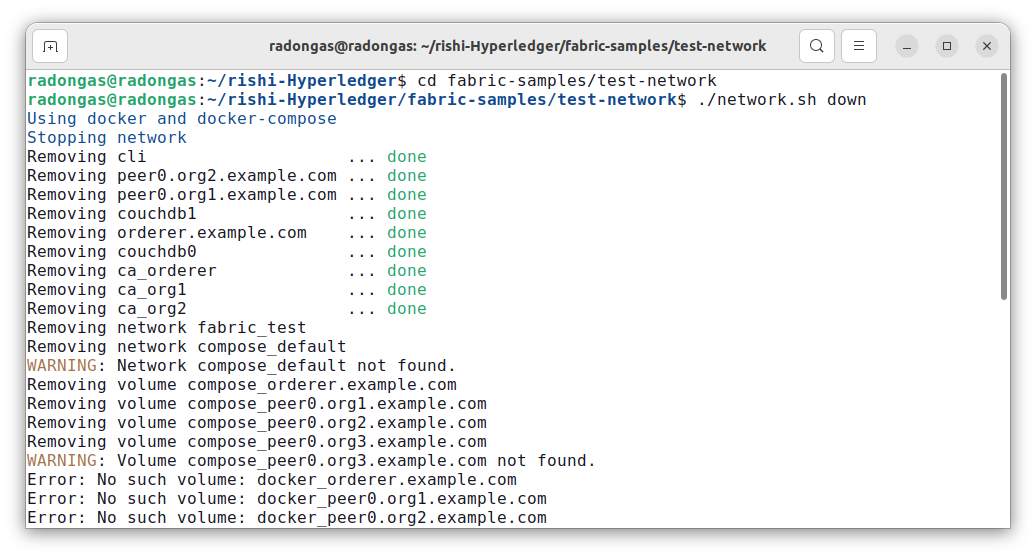
**SET UP THE BLOCKCHAIN NETWORK:**

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.

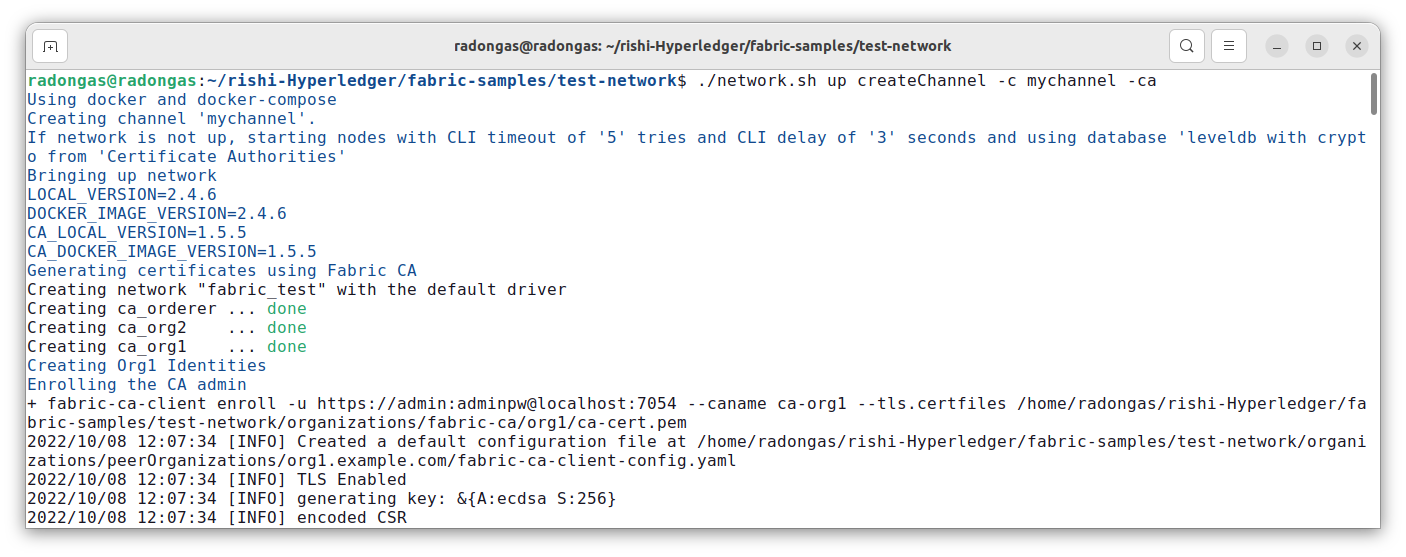
$ cdfabric**-**samples**/**test**-**network

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

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

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

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.

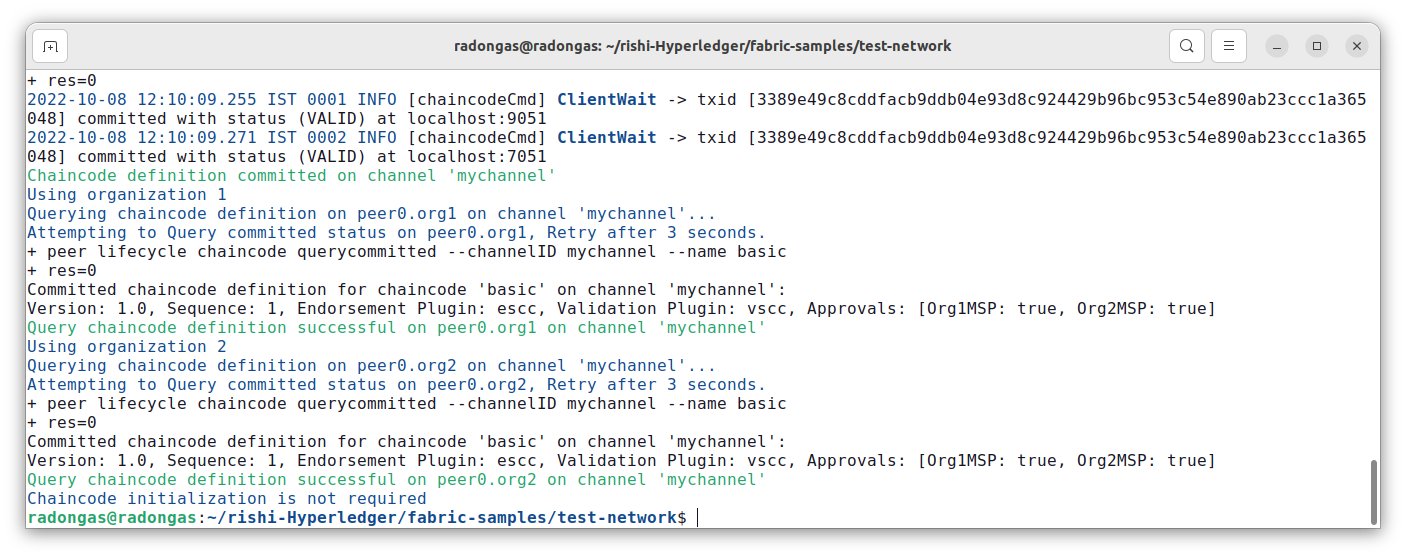
Next, let’s deploy the chaincode by calling the ./network.sh script with the chaincode name and language options.

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



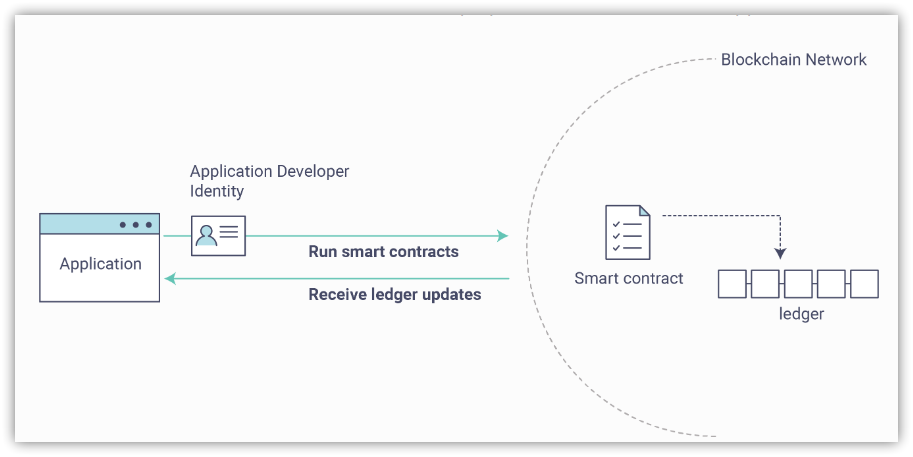
Behind the scenes, this script uses the chaincode lifecycle to package, install, query installed chaincode, approve chaincode for both Org1 and Org2, and finally commit the chaincode.

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



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

* + [JavaScript applicatio](https://github.com/hyperledger/fabric-samples/blob/main/asset-transfer-basic/application-javascript)n



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

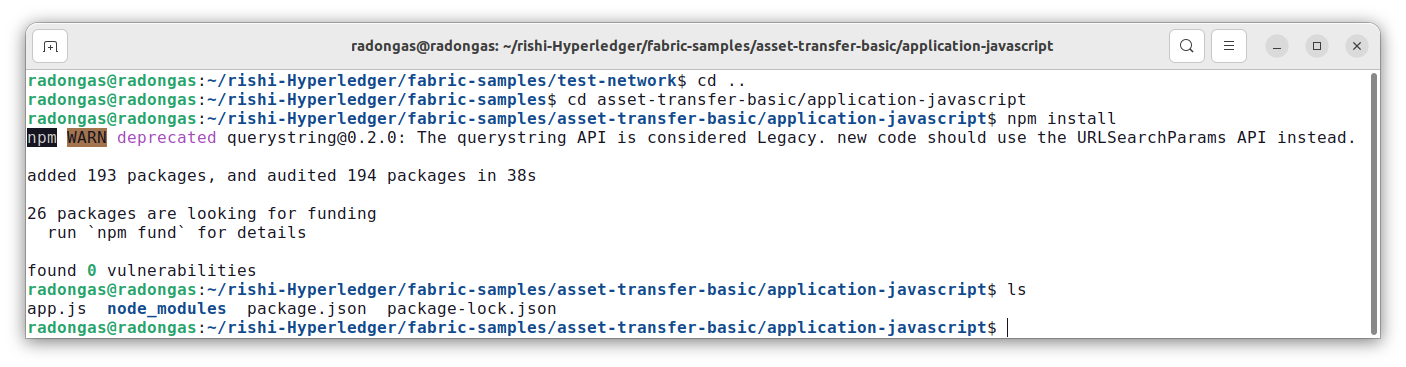
This process is installing the key application dependencies defined in the application’s package.json. The

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

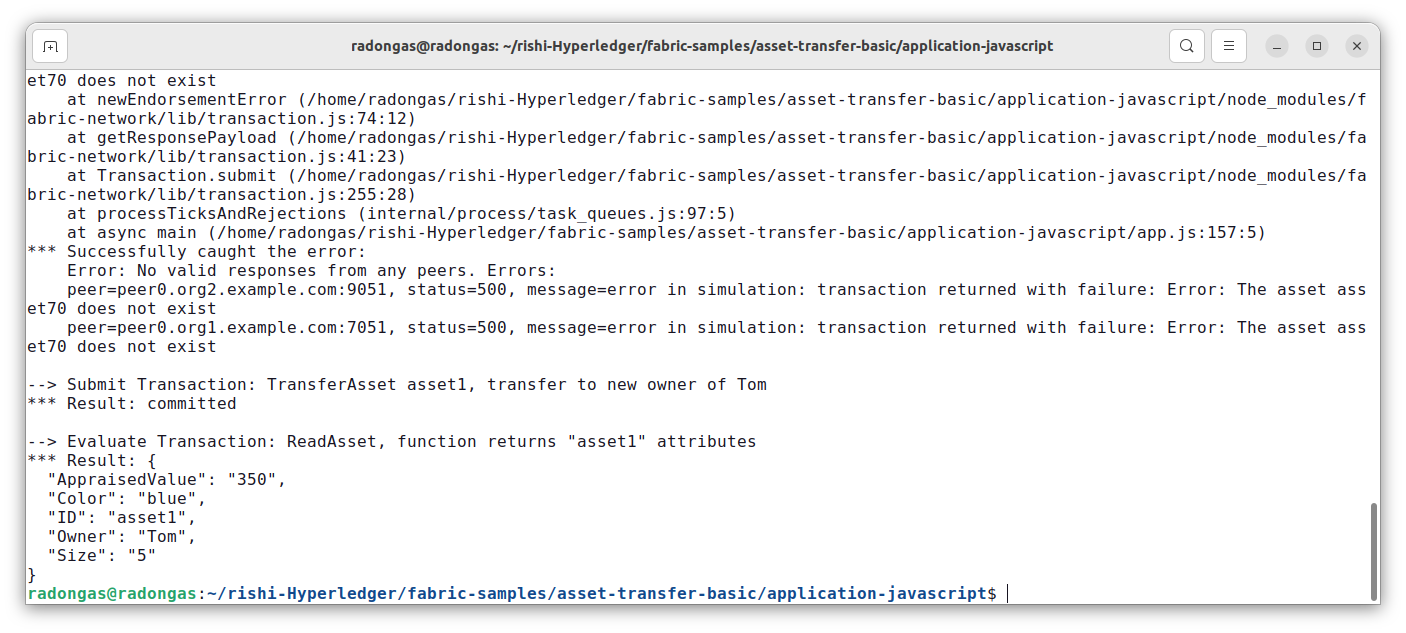
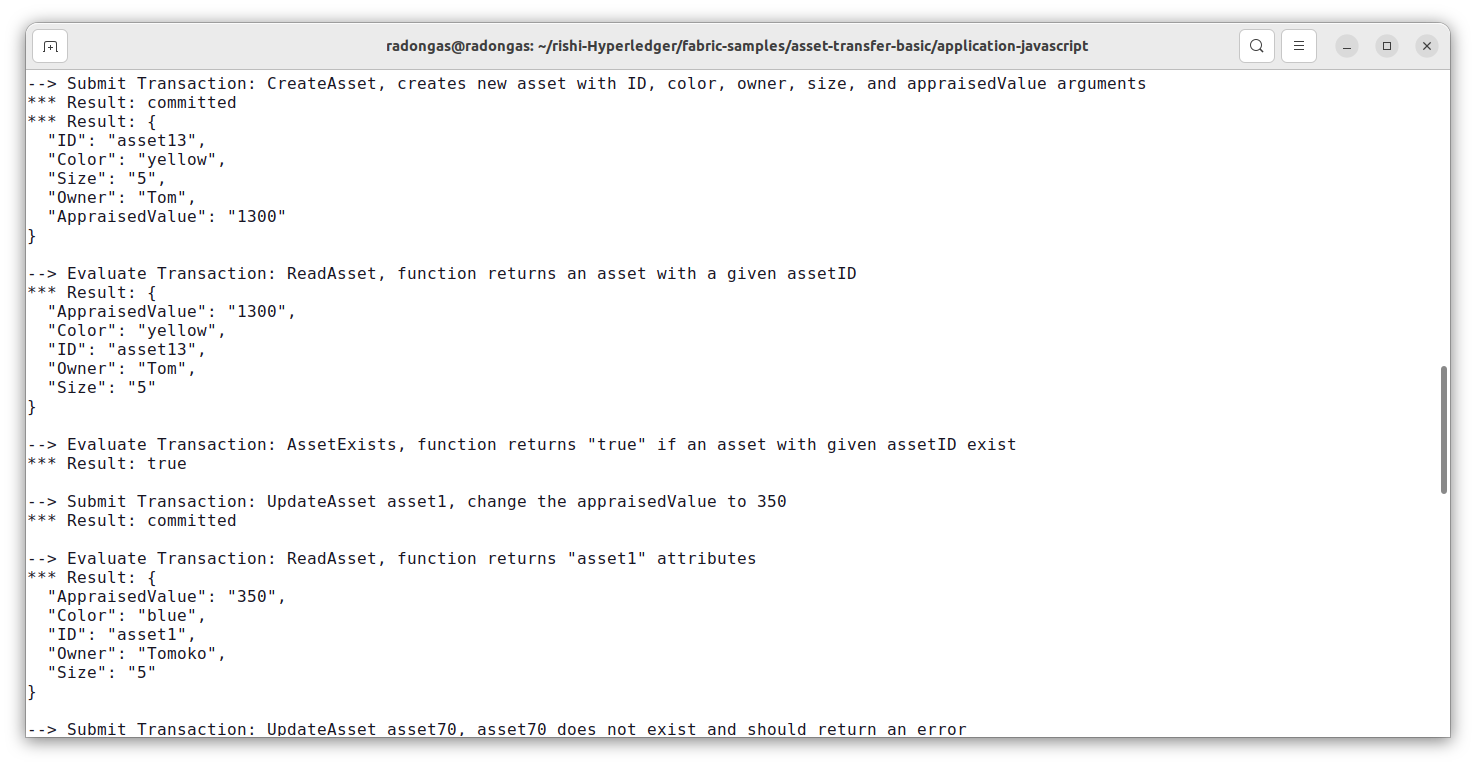
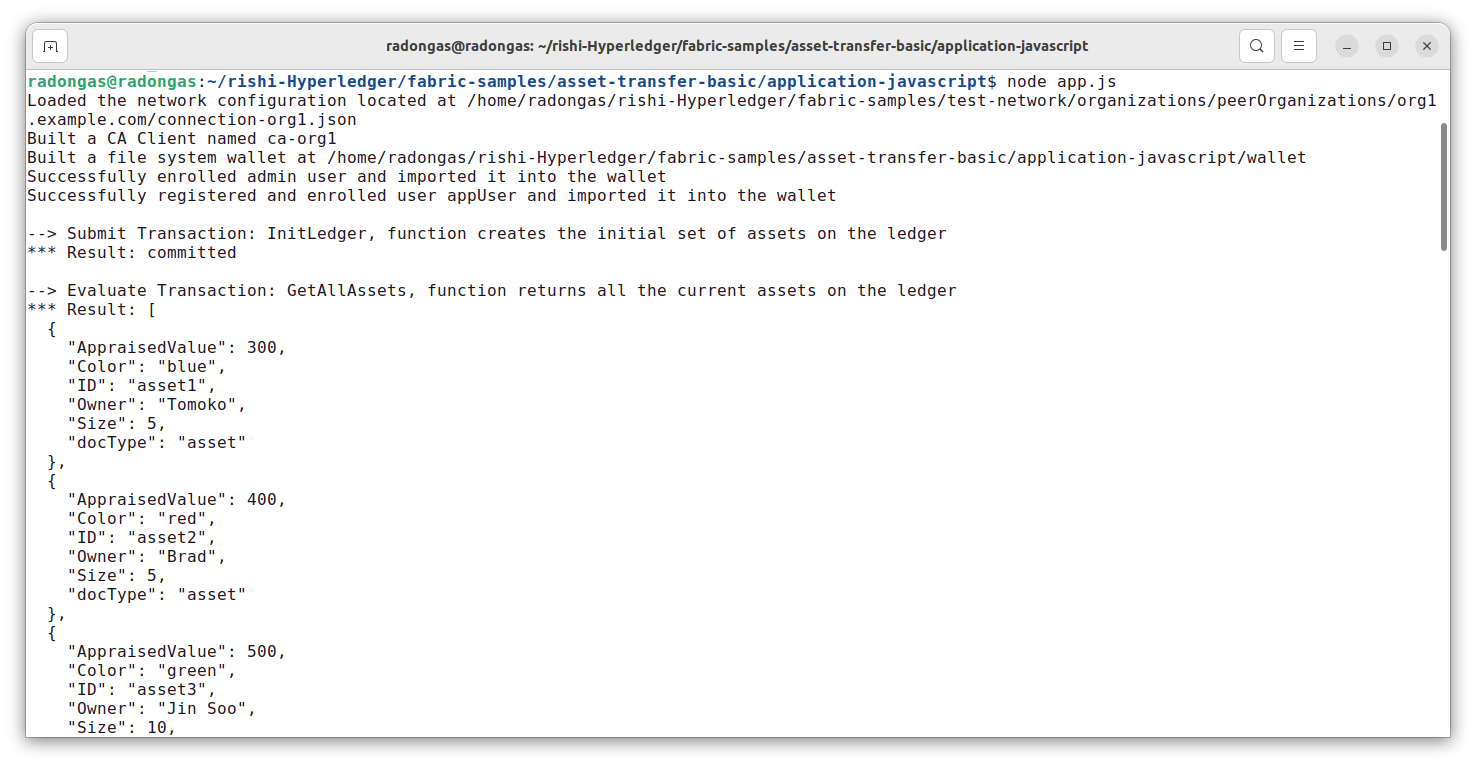


You should see the following:

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.



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:

## WhatsApp Image 2023-10-12 at 9.22.37 PM