

Web3 Technologies

Practical



R.D.&S.H NATIONAL COLLEGE & S.W.A. SCIENCE COLLEGE

Bandra, Mumbai - 400050

DEPARTMENT OF COMPUTER SCIENCE

M.Sc. Computer Science – Semester III

Web3 Technologies Practical

JOURNAL 2025-2026

Roll No. CS24002

Seat No. 1293021



R.D. & S.H. NATIONAL COLLEGE & S.W.A. SCIENCE COLLEGE,

Bandra, Mumbai – 400050.

Department of Computer Science

CERTIFICATE

This is to certify that **Mr. Mohd Bilal Shakir Hasan** of **M.Sc Part II (Sem III)** class has satisfactorily completed **VIII** Practicals in the subject of **Web3 Technologies Practical** as a part of M.Sc. Degree Course in Computer Science during the academic year 2025 – 2026.

Date of Submission: 26/01/2026

Faculty Incharge

Co-ordinator,

Department of Computer Science

Signature of External Examiner

INDEX

Sr. No.	Practical List	Pg. Nos	Date	Sign
1	Install and understand Docker container, Node.js, Java and Hyperledger Fabric, Ethereum and perform necessary software installation on local machine/create instance on Cloud to run.	1	04-12-2025	
2	Create and deploy a block chain network using Hyperledger Fabric SDK for Java.	9	04-12-2025	
3	Interact with a block chain network. Execute transactions and requests against a block chain network by creating an app to test the network and its rules	12	31-12-2025	
4	Deploy an asset-transfer app using block chain. Learn app development within a Hyperledger Fabric network.	19	23-12-2025	
5	Use block chain to track fitness club rewards.	26	06-12-2026	
6	Build a web app that uses Hyperledger Fabric to track and trace member rewards.	28	12-12-2025	
7	Car auction network: A Hello World example with Hyperledger Fabric Node SDK and IBM Block chain Starter Plan. Use Hyperledger Fabric to invoke chaincode while storing results and data in the starter plan.	34	09-12-2025	
8	Develop a voting application using Hyperledger and Ethereum.	38	06-12-2025	

Practical No. 1

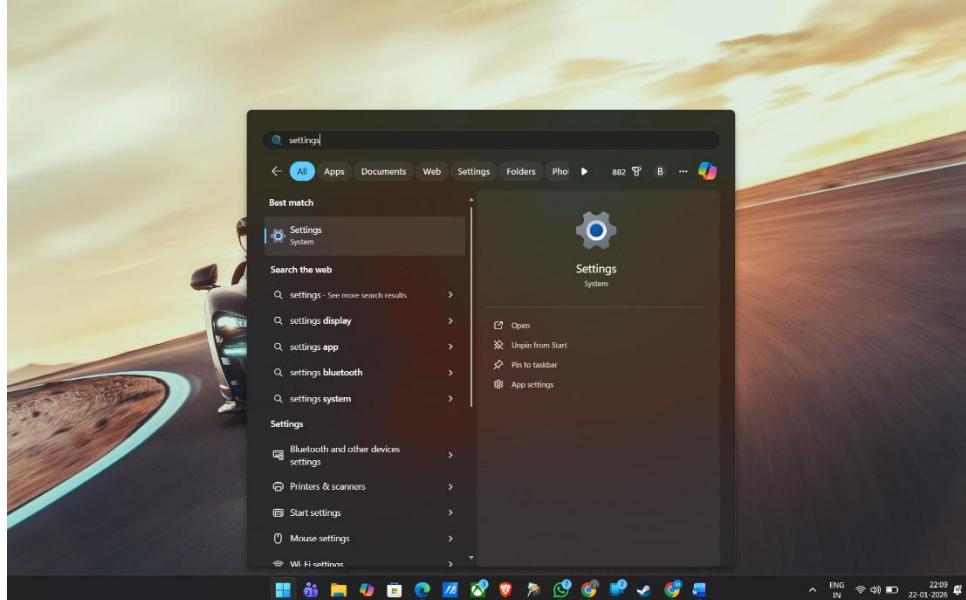
Aim - Install and understand Docker container, Node.js, Java and Hyperledger Fabric, Ethereum and perform necessary software installation on local machine/create instance on Cloud to run.

Theory – This experiment establishes the development environment required for both permissioned (enterprise) and permissionless (public) blockchain applications.

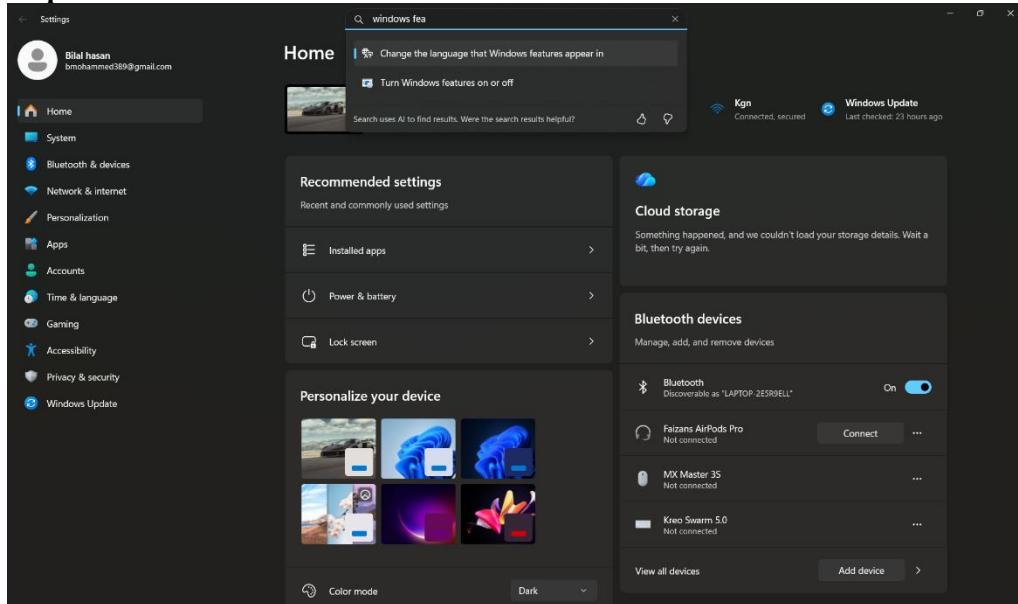
- Docker: A containerization platform used to run applications in isolated environments. In blockchain, it is critical for running multiple distinct network nodes (Peers, Orderers) on a single local machine to mimic a distributed network.
- Node.js: A JavaScript runtime environment. It is essential for running client-side SDKs (like the Hyperledger Fabric SDK or Ethereum's web3.js) that allow external applications to communicate with the blockchain ledger.
- Java: A robust, object-oriented language. It is a primary language used for writing Chaincode (smart contracts) in Hyperledger Fabric due to its stability and strong typing.
- Hyperledger Fabric: A permissioned, enterprise-grade blockchain framework. It uses a modular architecture where components like consensus and membership services run as separate Docker containers.
- Ethereum: A public blockchain platform featuring the Ethereum Virtual Machine (EVM). Development involves simulating the EVM locally to test smart contracts before deploying them to the live network.

Steps:

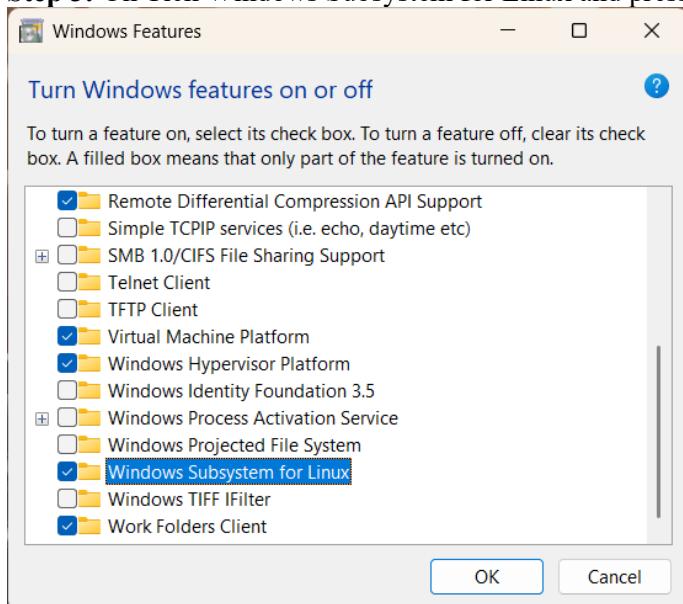
Step 1: Go to Device Setting



Step 2: Go to Windows Features



Step 3: On Tick Windows Subsystem for Linux and press OK.



Step 4: Install following things. (using below command)

```
bilalhasan@LAPTOP-2E5R9ELL:~$ # Update system  
sudo apt-get update  
sudo apt-get install -y ca-certificates curl  
sudo install -m 0755 -d /etc/apt/keyrings  
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc  
sudo chmod a+r /etc/apt/keyrings/docker.asc  
echo \  
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] \  
https://download.docker.com/linux/ubuntu \  
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \  
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null  
sudo apt-get update
```

```

sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-
plugin docker-compose
sudo usermod -aG docker $USER
newgrp docker

```

```

Selecting previously unselected package libslirp0:amd64.
Preparing to unpack .../15-libslirp0_0.6.1-1build1_amd64.deb ...
Unpacking libslirp0:amd64 (0.6.1-1build1) ...
Selecting previously unselected package slirp4netns.
Preparing to unpack .../16-slirp4netns_1.0.1-2_amd64.deb ...
Unpacking slirp4netns (1.0.1-2) ...
Setting up python3-datetime (0.19.2-1) ...
Setting up python3-texttable (1.6.4-1) ...
Setting up python3-dcopclient (0.6.2-4) ...
Setting up dbus-user-session (1.12.20-2ubuntu4.1) ...
Setting up docker-buildx-plugin (0.39.1-1~ubuntu.22.04~jammy) ...
Setting up containerd.io (2.2.1-1~ubuntu.22.04~jammy) ...
Created symlink /etc/systemd/system/multi-user.target.wants/containerd.service → /lib/systemd/system/containerd.service.
Setting up docker-compose-plugin (5.0.2-1~ubuntu.22.04~jammy) ...
Setting up docker-ce-clients (5.29.1.5-1~ubuntu.22.04~jammy) ...
Setting up libslirp0:amd64 (0.6.1-1build1) ...
Setting up pigz (2.6-1) ...
Setting up docker-ce-rootless-extras (5.29.1.5-1~ubuntu.22.04~jammy) ...
Setting up python3-websocket (1.2.3-1) ...
Setting up python3-dockerproxy (0.4.1-2) ...
Setting up slirp4netns (1.0.1-2) ...
Setting up python3-docker (5.0.3-1) ...
Setting up docker-ce (5.29.1.5-1~ubuntu.22.04~jammy) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Setting up docker-compose (1.29.2-1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
bilalhasan@LAPTOP-2E5R9ELL:~$ |

```

Step 5: Verifying the Installed Version and Program.

```

bilalhasan@LAPTOP-2E5R9ELL:~$ docker --version
docker run hello-world

```

```

bilalhasan@LAPTOP-2E5R9ELL:~$ docker --version
docker run hello-world
Docker version 29.1.5, build 0e6fee6
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
ea52d2000f90: Download complete
Digest: sha256:05813aedc15fb7b4d732e1be879d3252c1c9c25d885824f6295cab4538cb85cd
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

```

Step 6: Verify if docker is working

```

bilalhasan@LAPTOP-2E5R9ELL:~$ sudo docker pull busybox

```

```

bilalhasan@LAPTOP-2E5R9ELL:~$ sudo docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
61dfb50712f5: Pull complete
96cfb76e59bd: Download complete
Digest: sha256:e226d63288690dbe282443c8c7e57365c96b5228f0fe7f40731b5d84d37a06839
Status: Downloaded newer image for busybox:latest
dockers.io/library/busybox:latest
bilalhasan@LAPTOP-2E5R9ELL:~$ |

```

```

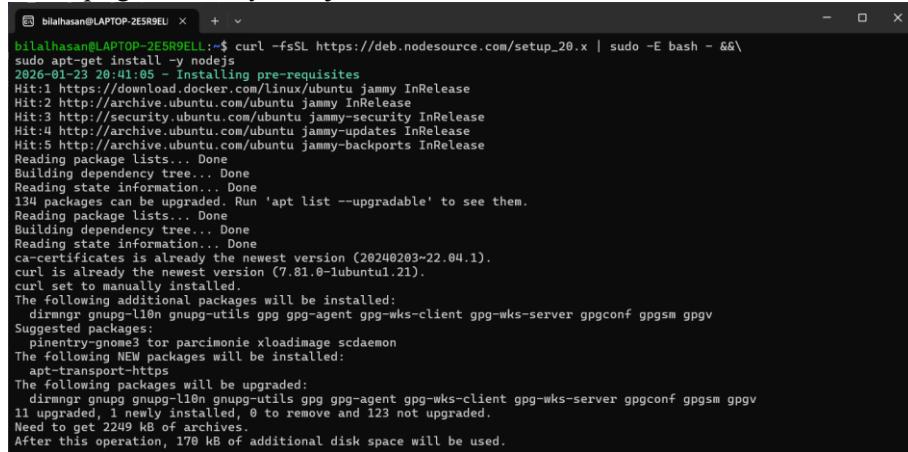
bilalhasan@LAPTOP-2E5R9ELL:~$ sudo docker images

```

IMAGE	ID	DISK USAGE	CONTENT SIZE	EXTRA
busybox:latest	e226d6308690	6.77MB	2.22MB	
hello-world:latest	05813aedc15f	25.9kB	9.52kB	U
ubuntu:latest	c01dbaa51b30	119MB	31.7MB	U

Step 7(a): Installation of Node.js

```
bilalhasan@LAPTOP-2E5R9ELL:~$ curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E  
bash - &&\  
sudo apt-get install -y nodejs
```



```
bilalhasan@LAPTOP-2E5R9ELL:~$ curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash - &&\  
sudo apt-get install -y nodejs  
2026-01-23 20:41:05 - Installing pre-requisites  
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease  
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease  
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease  
Hit:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease  
Hit:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
134 packages can be upgraded. Run 'apt list --upgradable' to see them.  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
ca-certificates is already the newest version (20240203-22.04.1).  
curl is already the newest version (7.81.0-lubuntu1.21).  
curl set to manually installed.  
The following additional packages will be installed:  
  dirmngr gnupg-l10n gnupg-utils gpg gpg-agent gpg-wks-client gpg-wks-server gpgconf gpgsm gpgv  
Suggested packages:  
  pinentry-gnome3 tor parcellite xloadimage scdaemon  
The following NEW packages will be installed:  
  apt-transport-https  
The following packages will be upgraded:  
  dirmngr gnupg gnupg-l10n gnupg-utils gpg gpg-agent gpg-wks-client gpg-wks-server gpgconf gpgsm gpgv  
11 upgraded, 1 newly installed, 0 to remove and 123 not upgraded.  
Need to get 2249 kB of archives.  
After this operation, 170 kB of additional disk space will be used.
```

Verifying the version of node.js

```
bilalhasan@LAPTOP-2E5R9ELL:~$ node -v
```



```
bilalhasan@LAPTOP-2E5R9ELL:~$ node -v  
v20.20.0  
bilalhasan@LAPTOP-2E5R9ELL:~$ |
```

Step 7(b): Create Your Project Folder

```
bilalhasan@LAPTOP-2E5R9ELL:~$ mkdir ~/node-server  
bilalhasan@LAPTOP-2E5R9ELL:~$ cd ~/node-server
```

Step 7(c): Create server.js

```
bilalhasan@LAPTOP-2E5R9ELL:~/node-server$ nano server.js
```

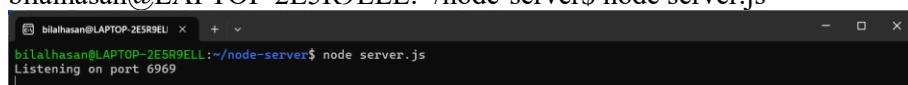
Paste this code:

```
const http = require("http")  
  
http.createServer((req, res) => {  
  res.writeHead(200, {"content-type": "text/html"});  
  res.end("Hello Node.js")  
}).listen(6969, () => console.log("Listening on port 6969"))
```

Save and exit (Ctrl+O, Enter, Ctrl+X in nano).

Step 7(d): Run the Server

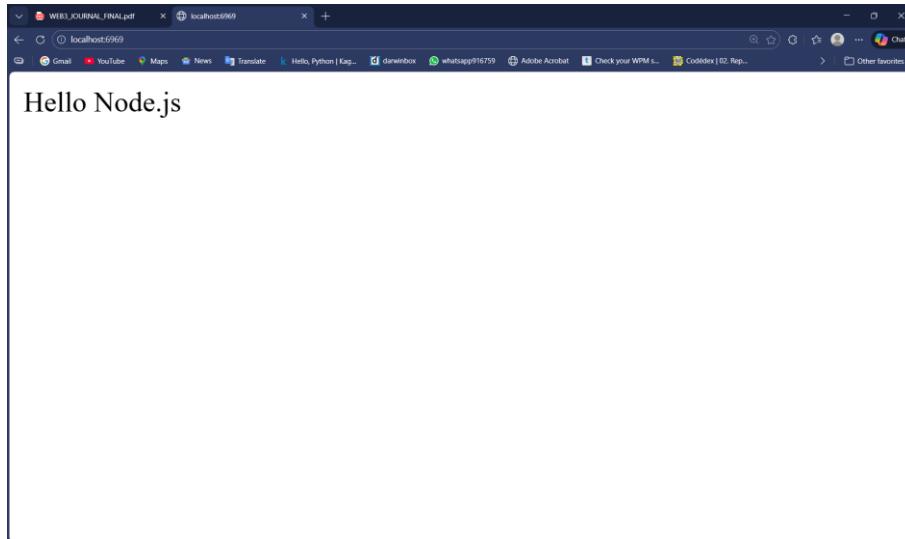
```
bilalhasan@LAPTOP-2E5R9ELL:~/node-server$ node server.js
```



```
bilalhasan@LAPTOP-2E5R9ELL:~/node-server$ node server.js  
Listening on port 6969  
|
```

Step 7(e): View in Browser

Open your Windows browser and go to:
<http://localhost:6969>



Step 8(a): Installing Java

bilalhasan@LAPTOP-2E5R9ELL:~\$ sudo apt install default-jdk

```
bilalhasan@LAPTOP-2E5R9ELL:~$ sudo apt install default-jdk
[sudo] password for bilalhasan:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
alsa-topology-conf alsacard ca-certificates-java default-jdk-headless default-jre default-jre-headless
fonts-dejavu-extra java-common libasound2 libasound2-data libatk-wrapper-java libatk-wrapper-java-jni libfontenc1
libglf7 libice-dev libice6 libnss3 libpcre1 libpthread-stubs0-dev libsm-dev libsm6 libx11-dev
libxau-dev libxaw7 libxcb-shape0 libxcb1-dev libxdmcp-dev libxft2 libxkbfile1 libxmu6 libxpm4 libxt-dev libxt6
libxf86-video-fbdev libxf86gal openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre openjdk-11-jre-headless x11-utils
x11proto-dev xorg-sgml-doctools xtrans-dev
Suggested packages:
libasound2-plugins alsalibs libice-doc pccsd libsm-doc libx11-doc libxcb-doc libxt-doc openjdk-11-demo
openjdk-11-source visualvm libnss-mdns fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei
| fonts-wqy-zhenhei fonts-indic mesa-utils
The following NEW packages will be installed:
alsa-topology-conf alsacard ca-certificates-java default-jdk default-jdk-headless default-jre
default-jre-headless fonts-dejavu-extra java-common libasound2 libasound2-data libatk-wrapper-java
libatk-wrapper-java-jni libfontenc1 libice6 libnss3 libpcre1 libpthread-stubs0-dev
libsm-dev libsm6 libx11-dev libxaw7 libxcb-shape0 libxcb1-dev libxdmcp-dev libxft2 libxkbfile1 libxmu6
libxf86-video-fbdev libxf86gal openjdk-11-jdk openjdk-11-jdk-headless openjdk-11-jre
openjdk-11-jre-headless x11-utils x11proto-dev xorg-sgml-doctools xtrans-dev
0 upgraded, 45 newly installed, 0 to remove and 123 not upgraded.
Need to get 126 MB of archives.
After this operation, 287 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Step 8(b): Verifying Java Installation.

```
bilalhasan@LAPTOP-2E5R9ELL:~$ java --version
openjdk 11.0.29 2025-10-21
OpenJDK Runtime Environment (build 11.0.29+7-post-Ubuntu-1ubuntu12.04)
OpenJDK 64-Bit Server VM (build 11.0.29+7-post-Ubuntu-1ubuntu12.04, mixed mode, sharing)
```

Step 9(a): Installing GoLang

bilalhasan@LAPTOP-2E5R9ELL:~\$ wget https://go.dev/dl/go1.25.5.linux-amd64.tar.gz

```
bilalhasan@LAPTOP-2E5R9ELL:~$ wget https://go.dev/dl/go1.25.5.linux-amd64.tar.gz
--2026-01-23 21:14:09-- https://go.dev/dl/go1.25.5.linux-amd64.tar.gz
Resolving go.dev (go.dev) ... 216.239.34.21, 216.239.36.21, 216.239.38.21, ...
Connecting to go.dev (go.dev)|216.239.34.21|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://dl.google.com/go/go1.25.5.linux-amd64.tar.gz [following]
--2026-01-23 21:14:09-- https://dl.google.com/go/go1.25.5.linux-amd64.tar.gz
Resolving dl.google.com (dl.google.com)... 216.58.203.14, 2404:6800:4009:803::200e
Connecting to dl.google.com (dl.google.com)|216.58.203.14|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 59768009 (57M) [application/x-gzip]
Saving to: 'go1.25.5.linux-amd64.tar.gz'

go1.25.5.linux-amd64.tar.gz 100%[=====] 57.00M 1.16MB/s in 24s
```

```

bilalhasan@LAPTOP-2E5R9ELL:~$ sudo rm -rf /usr/local/go
bilalhasan@LAPTOP-2E5R9ELL:~$ sudo tar -C /usr/local -xzf go1.25.5.linux-amd64.tar.gz
bilalhasan@LAPTOP-2E5R9ELL:~$ echo 'export PATH=$PATH:/usr/local/go/bin' >> ~/.bashrc
source ~/.bashrc

```

A terminal window titled 'bilalhasan@LAPTOP-2E5R9ELL'. It contains the following command sequence:

```

bilalhasan@LAPTOP-2E5R9ELL:~$ sudo rm -rf /usr/local/go
[sudo] password for bilalhasan:
bilalhasan@LAPTOP-2E5R9ELL:~$ sudo tar -C /usr/local -xzf go1.25.5.linux-amd64.tar.gz
bilalhasan@LAPTOP-2E5R9ELL:~$ echo 'export PATH=$PATH:/usr/local/go/bin' >> ~/.bashrc
source ~/.bashrc

```

Step 9(b): Verifying go version

```

bilalhasan@LAPTOP-2E5R9ELL:~$ go version

```

A terminal window titled 'bilalhasan@LAPTOP-2E5R9ELL'. It shows the output of the 'go version' command:

```

bilalhasan@LAPTOP-2E5R9ELL:~$ go version
go version go1.25.5 linux/amd64
bilalhasan@LAPTOP-2E5R9ELL:~$ 

```

Step 10(a): Installing Git

```

bilalhasan@LAPTOP-2E5R9ELL:~$ sudo apt install git

```

A terminal window titled 'bilalhasan@LAPTOP-2E5R9ELL'. It shows the output of the 'sudo apt install git' command, which installs version 2.34.1-1ubuntu1.15:

```

bilalhasan@LAPTOP-2E5R9ELL:~$ sudo apt install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following packages will be upgraded:
  git
1 upgraded, 0 newly installed, 0 to remove and 122 not upgraded.
Need to get 3166 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get: http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.15 [3166 kB]
Fetched 3166 kB in 3s (1214 kB/s)
(Reading database ... 50394 files and directories currently installed.)
Preparing to unpack .../git_1%3a2.34.1-1ubuntu1.15_amd64.deb ...
Unpacking git (1:2.34.1-1ubuntu1.15) over (1:2.34.1-1ubuntu1.11) ...
Setting up git (1:2.34.1-1ubuntu1.15) ...

```

Step 10(b): Verifying Git version

```

bilalhasan@LAPTOP-2E5R9ELL:~$ git --version

```

A terminal window titled 'bilalhasan@LAPTOP-2E5R9ELL'. It shows the output of the 'git --version' command:

```

bilalhasan@LAPTOP-2E5R9ELL:~$ git --version
git version 2.34.1
bilalhasan@LAPTOP-2E5R9ELL:~$ 

```

Step 10(c): Installing Hyperledger Fabric

Before installing hyper ledger make sure docker is up and running (sudo systemctl status docker)

Login into the Docker

```

bilalhasan@LAPTOP-2E5R9ELL:~$ docker login

```

A terminal window titled 'bilalhasan@LAPTOP-2E5R9ELL'. It shows the Docker login process, including a one-time device confirmation code (ZCG-WWSQ) and a warning about unencrypted credentials:

```

bilalhasan@LAPTOP-2E5R9ELL:~$ docker login
USING WEB-BASED LOGIN
Info → To sign in with credentials on the command line, use 'docker login -u <username>'

Your one-time device confirmation code is: ZCG-WWSQ
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate
Waiting for authentication in the browser...

WARNING! Your credentials are stored unencrypted in '/home/bilalhasan/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
bilalhasan@LAPTOP-2E5R9ELL:~$ 

```

Verify Docker Pull Request

```

bilalhasan@LAPTOP-2E5R9ELL:~$ docker pull hello-world

```

A terminal window titled 'bilalhasan@LAPTOP-2E5R9ELL'. It shows the output of the 'docker pull hello-world' command:

```

bilalhasan@LAPTOP-2E5R9ELL:~$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
Digest: sha256:05813aedc15fb7b4d732e1be879d3252c1c9c25d885824f6295cab4538cb85cd
Status: Image is up to date for hello-world:latest
docker.io/library/hello-world:latest

```

```
bilalhasan@LAPTOP-2E5R9ELL:~$ curl -sSL http://bit.ly/2ysbOFE | bash -s 2.2.0
```

```
bilalhasan@LAPTOP-2E5R9ELL:~$ curl -sSL http://bit.ly/2ysbOFE | bash -s 2.2.0
Clone hyperledger/fabric-samples repo
====> Changing directory to fabric-samples
====> Checking out v2.2.0 of hyperledger/fabric-samples
Pull Hyperledger Fabric binaries
====> Downloading version 2.2.0 platform specific fabric binaries
====> Downloading: https://github.com/hyperledger/fabric/releases/download/v2.2.0/hyperledger-fabric-linux-amd64-2.2.0.t
ar.gz
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
0  0     0  0     0  0      0  --::-- --::-- --::-- 0
100 72.7M 100 72.7M 0  0  235k  0  0:05:16 0:05:16 3977k
==> Done.
====> Downloading version 1.5.15 platform specific fabric-ca-client binary
====> Downloading: https://github.com/hyperledger/fabric-ca/releases/download/v1.5.15/hyperledger-fabric-ca-linux-amd64-1.5.15.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 30.8M 100 30.8M 0  0  203k  0  0:02:31 0:02:31 3552k
==> Done.

Pull Hyperledger Fabric docker images
FABRIC_IMAGES: peer orderer ccenv baseos
====> Pulling fabric Images
====> docker.io/hyperledger/fabric-peer:2.2.0
2.2.0: Pulling from hyperledger/fabric-peer
Digest: sha256:81efaab943387820815c9b9bd3e3bec9d262365e635bac6f500a287fb3e873e7
Status: Image is up to date for hyperledger/fabric-peer:2.2.0
docker.io/hyperledger/fabric-peer:2.2.0
====> docker.io/hyperledger/fabric-orderer:2.2.0
2.2.0: Pulling from hyperledger/fabric-orderer
Digest: sha256:bb97a8e80b53d9f32c3b856ae3b55bab57bf3cc9483978a471a5bee0ac49728e
Status: Image is up to date for hyperledger/fabric-orderer:2.2.0
docker.io/hyperledger/fabric-orderer:2.2.0
====> docker.io/hyperledger/fabric-ccenv:2.2.0
2.2.0: Pulling from hyperledger/fabric-ccenv
```

```
bilalhasan@LAPTOP-2E5R9ELL:~$ curl -sSL http://bit.ly/2ysbOFE | bash -s 2.2.0
1.5.15: Pulling from hyperledger/fabric-ca
Digest: sha256:1b64dc702bd408e14c84f452dedc0359b7dc06679b751b08937eb2bf2865fed40
Status: Image is up to date for hyperledger/fabric-ca:1.5.15
docker.io/hyperledger/fabric-ca:1.5.15
====> List out hyperledger docker images
WARNING: This output is designed for human readability. For machine-readable output, please use --format.
hyperledger/fabric-baseos:2.2 6584eb96bb3 13MB 3.18MB
hyperledger/fabric-baseos:2.2.0 6584eb96bb3 13MB 3.18MB
hyperledger/fabric-baseos:latest 6584eb96bb3 13MB 3.18MB
hyperledger/fabric-ca:1.5 1b64dc702bd4 351MB 112MB
hyperledger/fabric-ca:1.5.15 1b64dc702bd4 351MB 112MB
hyperledger/fabric-ca:latest 1b64dc702bd4 351MB 112MB
hyperledger/fabric-ccenv:2.2 4ec2f71fbcd5 838MB 214MB
hyperledger/fabric-ccenv:2.2.0 4ec2f71fbcd5 838MB 214MB
hyperledger/fabric-ccenv:latest 4ec2f71fbcd5 838MB 214MB
hyperledger/fabric-orderer:2.2 bb97a8e80b53 59.8MB 18.3MB
hyperledger/fabric-orderer:2.2.0 bb97a8e80b53 59.8MB 18.3MB
hyperledger/fabric-orderer:latest bb97a8e80b53 59.8MB 18.3MB
hyperledger/fabric-peer:2.2 81efaab94338 83.3MB 25.3MB
hyperledger/fabric-peer:2.2.0 81efaab94338 83.3MB 25.3MB
hyperledger/fabric-peer:latest 81efaab94338 83.3MB 25.3MB
bilalhasan@LAPTOP-2E5R9ELL:~$
```

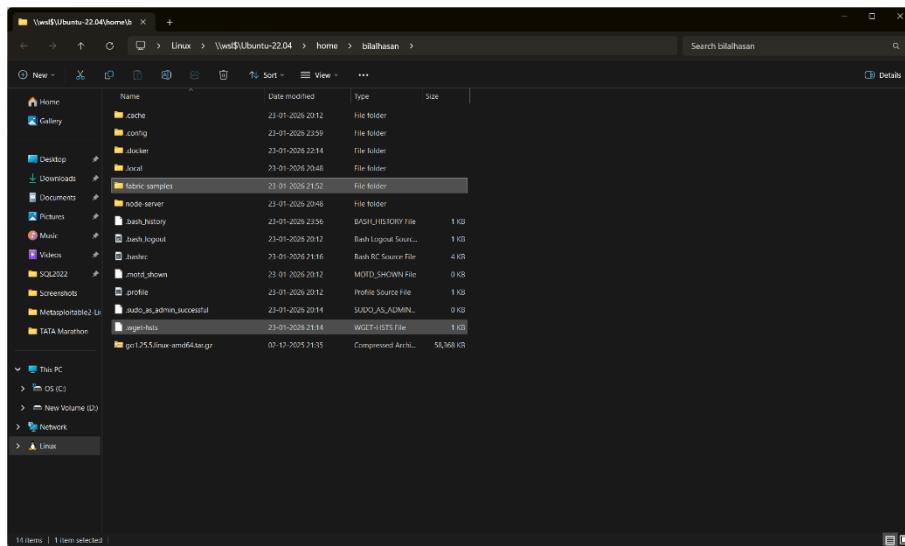
Step 10(d): Install wslu for viewing the library where Hyperledger installed.

```
bilalhasan@LAPTOP-2E5R9ELL:~$ sudo apt-get install wslu
```

```
bilalhasan@LAPTOP-2E5R9ELL:~$ sudo apt-get install wslu
[sudo] password for bilalhasan:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
desktop-file-utils libasynccn8 libflac8 libogg0 libopus0 libpulse0 libpulsedsp libsndfile1 libvorbis0a libvorbisenc2
pulseaudio-utils
Suggested packages:
opus-tools pulseaudio avahi-daemon imagemagick
The following NEW packages will be installed:
desktop-file-utils libasynccn8 libflac8 libogg0 libopus0 libpulse0 libpulsedsp libsndfile1 libvorbis0a libvorbisenc2
pulseaudio-utils wslu
0 upgraded, 12 newly installed, 0 to remove and 122 not upgraded.
Need to get 1270 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 desktop-file-utils amd64 0.26-1ubuntu3 [55.9 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 libasynccn8 amd64 0.8-6build2 [12.8 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 libflac8 amd64 1.3.5-1ubuntu2 [22.9 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/main amd64 libflac8 amd64 1.3.3-2ubuntu0.2 [111 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 libogg0 amd64 1.3.1-9.1build2 [203 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/main amd64 libopus0 amd64 1.3.7-1build2 [99.2 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/main amd64 libvorbisenc2 amd64 1.3.7-1build2 [82.6 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsndfile1 amd64 1.0.31-2ubuntu0.2 [196 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpulse0 amd64 1:15.99.1+dfsg1-1ubuntu2.2 [298 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpulsedsp amd64 1:15.99.1+dfsg1-1ubuntu2.2 [23.3 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 pulseaudio-utils amd64 1:15.99.1+dfsg1-1ubuntu2.2 [76.1 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/universe amd64 wslu amd64 3.2.3-0ubuntu3 [89.7 kB]
Fetched 1270 kB in 2s (592 kB/s)
```

For Viewing the Library:

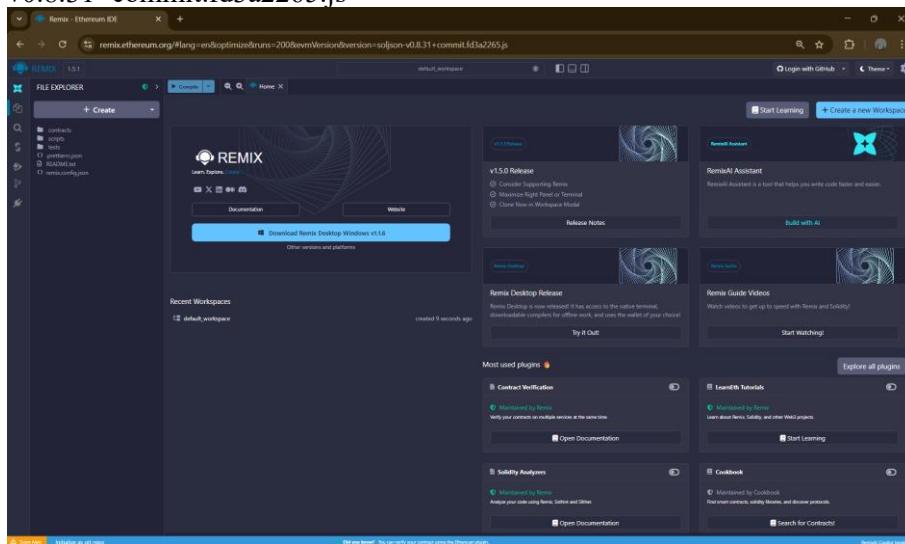
bilalhasan@LAPTOP-2E5R9ELL:~\$ wslview .



Step 12: Ethereum:

Running Ethereum IDE online (Remix IDE)

link: <https://remix.ethereum.org/#lang=en&optimize=false&runs=200&evmVersion=&version=soljson-v0.8.31+commit.fd3a2265.js>



Practical No. 2

Aim - Create and deploy a block chain network using Hyperledger Fabric SDK for Java.

Theory: This experiment focuses on the application layer, demonstrating how a Java client application interacts with a running Hyperledger Fabric blockchain.

- Hyperledger Fabric SDK for Java: This is a library that enables Java applications to communicate with the Fabric network. It abstracts the complex low-level gRPC protocols, allowing developers to easily manage the network and invoke smart contracts.
- Connection Profile: The SDK connects to the blockchain using a Connection Profile (usually connection.yaml or json). This file describes the network topology, specifying the addresses and ports of the Peers, Orderers, and Certificate Authorities.
- Identity Management (Wallet): Fabric is a permissioned network. The SDK uses a Wallet to store digital identities (X.509 certificates and private keys). The Java application retrieves these credentials from the wallet to sign transactions, proving the user's identity.
- The Gateway Class: This is the primary entry point for the application. It acts as a bridge, managing the transaction lifecycle—from proposal and endorsement to submission—simplifying the code required to talk to the network.
- Transaction Types:
 - Submit Transaction: Used to write data to the ledger. This process involves the consensus mechanism and updating the world state.
 - Evaluate Transaction: Used to read data from the ledger. This is a quick query that executes on a peer but does not change the state or require consensus.

Steps:

Step 1: Install wslu for viewing the library where Hyperledger installed.

```
bilalhasan@LAPTOP-2E5R9ELL:~$ sudo apt-get install wslu
```

```
bilalhasan@LAPTOP-2E5R9ELL:~$ sudo apt-get install wslu
[sudo] password for bilalhasan:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
desktop-file-utils libasynncns0 libflac8 libogg0 libopus0 libpulse0 libpulsedsp libsndfile1 libvorbis0a libvorbisenc2
pulseaudio-utils
Suggested packages:
opus-tools pulseaudio avahi-daemon imagemagick
The following NEW packages will be installed:
desktop-file-utils libasynncns0 libflac8 libogg0 libopus0 libpulse0 libpulsedsp libsndfile1 libvorbis0a libvorbisenc2
pulseaudio-utils wslu
0 upgraded, 12 newly installed, 0 to remove and 122 not upgraded.
Need to get 1270 kB of archives.
After this operation, 4385 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 desktop-file-utils amd64 0.26-1ubuntu3 [55.9 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 libasynncns0 amd64 0.8-6build2 [12.8 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 libogg0 amd64 1.3.5-0ubuntu3 [22.9 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libflac8 amd64 1.3.3-2ubuntu0.2 [111 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 libopus0 amd64 1.3.1-0.1build2 [203 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/main amd64 libvorbis0a amd64 1.3.7-1build2 [99.2 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/main amd64 libvorbisenc2 amd64 1.3.7-1build2 [82.6 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsndfile1 amd64 1.0.31-2ubuntu0.2 [196 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpulse0 amd64 1:15.99.1+dfsg1-1ubuntu2.2 [298 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpulsedsp amd64 1:15.99.1+dfsg1-1ubuntu2.2 [23.3 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 pulseaudio-utils amd64 1:15.99.1+dfsg1-1ubuntu2.2 [76.1 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/universe amd64 wslu amd64 3.2.3-0ubuntu3 [89.7 kB]
Fetched 1270 kB in 2s (592 kB/s)
```

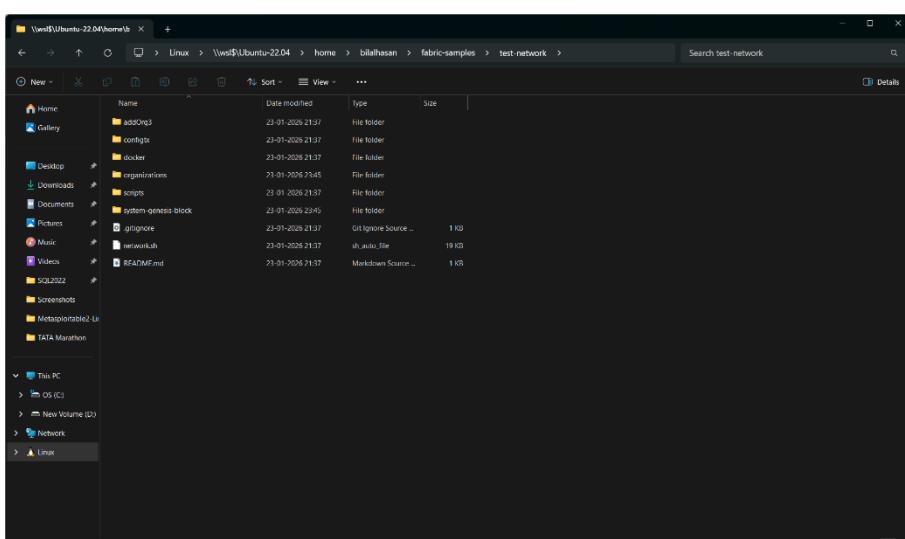
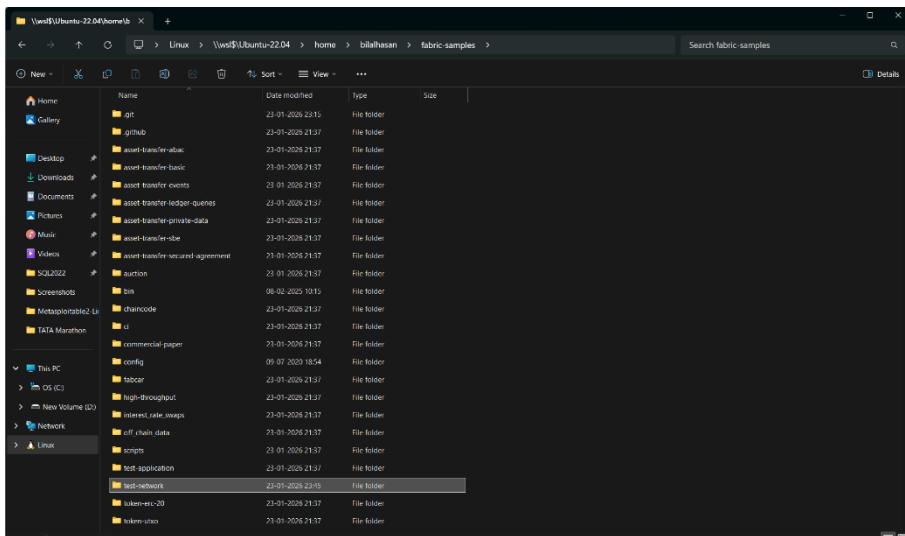
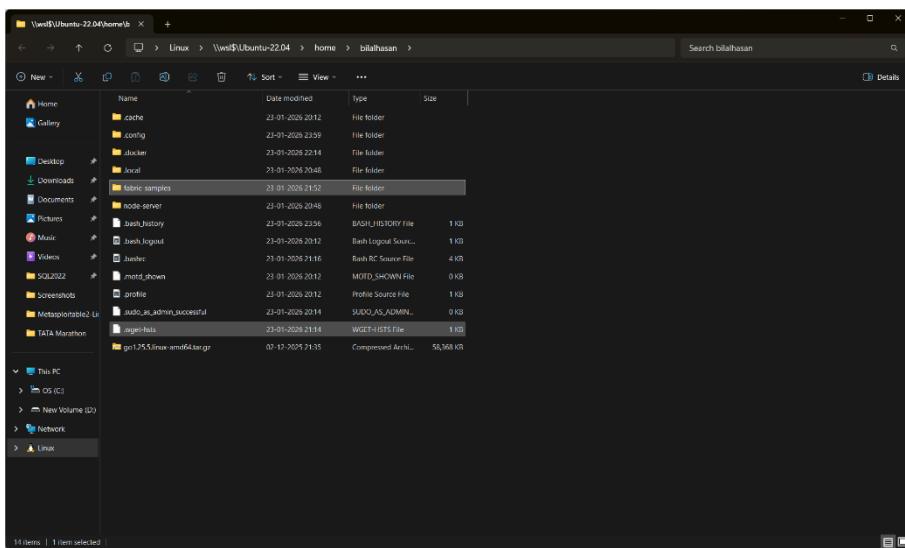
Step 2: Install Hyperledger (If not already done)

Once its installed there will be a directory named fabric-samples

```
bilalhasan@LAPTOP-2E5R9ELL:~$ ls
fabric-samples go1.25.5.linux-amd64.tar.gz node-server
bilalhasan@LAPTOP-2E5R9ELL:~$ cd fabric-samples
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples$ cd test-network
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ |
```

For Viewing the Library:

bilalhasan@LAPTOP-2E5R9ELL:~\$ wslview .



Step 3: Test the network.sh

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ cd fabric-samples/test-network
```

```
_size:16777216
2025-01-24 00:03:29.266 IST [common_tools.configtxgen.localconfig] Load -> INFO 004 Loaded configuration: /home/bilalhasan/fabric-samples/test-network/configtx/configtx.yaml
2025-01-24 00:03:29.267 IST [common_tools.configtxgen] doOutputBlock --> INFO 005 Generating genesis block
2025-01-24 00:03:29.267 IST [common_tools.configtxgen] doOutputBlock --> INFO 006 Writing genesis block
+ res=0
Creating network "fabric_test" with the default driver
Creating volume "docker_orderer.example.com" with default driver
Creating volume "docker_peer0.org1.example.com" with default driver
Creating volume "docker_peer0.org2.example.com" with default driver
Creating peer0.org2.example.com ... done
Creating orderer.example.com ... done
Creating peer0.org1.example.com ... done
Creating cli ... done
CONTAINER ID IMAGE COMMAND CREATED STATUS NAMES
PORTS
d8a951281fcf hyperledger/fabric-tools:latest "/bin/bash" Less than a second ago Up Less than a second cli
38f0f847ea52 hyperledger/fabric-orderer:latest "orderer" 1 second ago Up Less than a second orderer.example.com
84d46b03321b hyperledger/fabric-peer:latest "peer node start" 1 second ago Up Less than a second peer0.org1.example.com
e9aff76a0245 hyperledger/fabric-peer:latest "peer node start" 1 second ago Up Less than a second peer0.org2.example.com
bafef96d13b1c ubuntu "bash" 4 hours ago Exited (0) 4 hours ago lucid.liskov
66978b728d21 hello-world "/hello" 4 hours ago Exited (0) 4 hours ago pensive_hoover
```

Step 4: Create Channel on network.sh

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ ./network.sh createChannel
```

```
+ jq '.data.data[0].payload.data.config'
+ jq '.channel_group.groups.Org2MSP.values += [{"AnchorPeers":{"mod_policy": "Admins", "value": {"anchor_peers": [{"host": "peer0.org2.example.com", "port": 9051}], "version": "0"}}, "Org2MSPconfig.json"
Generating anchor peer update transaction for Org2 on channel mychannel
+ configtxlator proto_encode --input Org2MSPConfig.json --type common.Config
+ configtxlator proto_encode --input Org2MSPmodified_config.json --type common.Config
+ configtxlator compute_update --channel_id mychannel --original original_config.pb --updated modified_config.pb
+ configtxlator decode --input config_update.pb --type common.ConfigUpdate
+ jq .
++ cat config_update.json
+ echo ${payload}
{
  "header": {
    "channel_header": {
      "channel_id": "mychannel",
      "type": 2
    },
    "data": {
      "config_update": [
        {
          "channel_id": "mychannel",
          "isolated_data": {},
          "read_set": {
            "groups": [
              {
                "Application": {
                  "groups": [
                    {
                      "Org2MSP": {
                        "groups": [
                          {
                            "mod_policy": "Admins",
                            "policies": [
                              {
                                "mod_policy": "Admins",
                                "policy": null,
                                "version": "0"
                              }
                            ],
                            "Endorsement": {
                              "mod_policy": "Admins",
                              "policy": null,
                              "version": "0"
                            }
                          }
                        ],
                        "Reader": {
                          "mod_policy": "Admins",
                          "policy": null,
                          "version": "0"
                        },
                        "Writer": {
                          "mod_policy": "Admins",
                          "policy": null,
                          "version": "0"
                        }
                      }
                    ]
                  }
                }
              }
            ]
          },
          "mod_policy": "Admins",
          "policies": [
            {
              "mod_policy": "Admins",
              "value": null
            }
          ],
          "version": "0"
        }
      ]
    }
  }
}
+ configtxlator proto_encode --input config_update_in_envelope.json --type common.Envelope
2025-01-23 18:34:06.562 UTC 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2025-01-23 18:34:06.589 UTC 0002 INFO [channelCmd] update -> Successfully submitted channel update
Anchor peer set for org 'Org2MSP' on channel 'mychannel'
Channel 'mychannel' joined
```

Step 5: Down network.sh

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ ./network.sh down
```

```
Stopping network
Stopping cli ... done
Stopping orderer.example.com ... done
Stopping peer0.org1.example.com ... done
Stopping peer0.org2.example.com ... done
Removing cli ... done
Removing orderer.example.com ... done
Removing peer0.org1.example.com ... done
Removing peer0.org2.example.com ... done
Removing network fabric_test
Removing volume docker_orderer.example.com
Removing volume docker_peer0.org1.example.com
Removing volume docker_peer0.org2.example.com
Removing network fabric_test
WARNING: Network fabric_test not found.
Removing volume docker_peer0.org3.example.com
WARNING: Volume docker_peer0.org3.example.com not found.
No containers available for deletion
WARNING: This output is designed for human readability. For machine-readable output, please use --format.
No images available for deletion
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ |
```

Practical No. 3

Aim - Interact with a block chain network. Execute transactions and requests against a block chain network by creating an app to test the network and its rules.

Theory:

To interact with a permissioned blockchain like Hyperledger Fabric, a Client Application is used. This application acts as a bridge between the end-user and the blockchain network, utilizing the Fabric SDK (Software Development Kit) to invoke smart contracts.

1. The Application Gateway: The application connects to the network using a "Gateway." The Gateway manages the complexity of the network interactions, such as service discovery (finding available peers) and transaction submission.
2. Identity Management (Wallet): All interactions are authenticated. The application uses a Wallet to store digital identities (X.509 certificates). Every transaction request is cryptographically signed by the user's private key to prove identity and authority.
3. Transaction Types:
 - a. Submit Transaction (Invoke): Used to modify the ledger (e.g., creating or transferring an asset). This process is complex: the transaction is proposed to peers for endorsement, sent to the Orderer for packaging into a block, and finally validated and committed by all peers.
 - b. Evaluate Transaction (Query): Used to read data from the ledger. This is lightweight; the client queries a peer directly, and the peer returns the result immediately without involving the ordering service or consensus.
4. Testing Network Rules: The application tests the "business rules" defined in the Chaincode. For example, if the logic states "Asset ownership cannot be negative," the app sends a request violating this to verify that the network correctly rejects it.

Steps:

Step 1: Installing dependencies

```
bilalhasan@LAPTOP-2E5R9ELL:~$ sudo apt install make
```

```
bilalhasan@LAPTOP-2E5R9ELL:~$ sudo apt install make
[sudo] password for bilalhasan:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  make-doc
The following NEW packages will be installed:
  make
0 upgraded, 1 newly installed, 0 to remove and 122 not upgraded.
Need to get 188 kB of archives.
After this operation, 426 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 make amd64 4.3-4.1build1 [188 kB]
Fetched 188 kB in 2s (107 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (Dialog frontend requires a screen at least 13 lines tall and 31 columns wide.)
debconf: falling back to frontend: Readline
Selecting previously unselected package make.
(Reading database ... (unpackaged files and directories currently installed.))
Preparing to unpack .../make_4.3-4.1build1_amd64.deb ...
Unpacking make (4.3-4.1build1) ...
Setting up make (4.3-4.1build1) ...
Processing triggers for man-db (2.10.2-1) ...
bilalhasan@LAPTOP-2E5R9ELL:~$ |
```

(Use this command if Golang is not installed in the PC)

```
sudo apt update
```

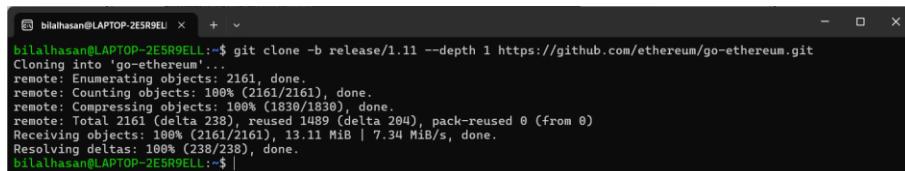
```
sudo apt install golang-go
```

Step 2: Install GEth (Go Ethereum)

Geth is a CLI with some resources to connect you to the Ethereum network. It will be used to start our private network in the local environment.

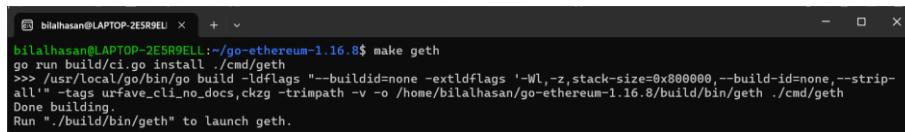
To install Geth in Ubuntu/Debian follow the following steps:

```
bilalhasan@LAPTOP-2E5R9ELL:~$ git clone -b release/1.11 --depth 1  
https://github.com/ethereum/go-ethereum.git
```



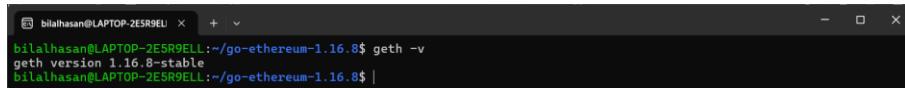
```
bilalhasan@LAPTOP-2E5R9ELL:~$ git clone -b release/1.11 --depth 1 https://github.com/ethereum/go-ethereum.git  
Cloning into 'go-ethereum'...  
remote: Enumerating objects: 2161, done.  
remote: Counting objects: 100% (2161/2161), done.  
remote: Compressing objects: 100% (1830/1830), done.  
remote: Total 2161 (delta 238), reused 1889 (delta 204), pack-reused 0 (from 0)  
Receiving objects: 100% (2161/2161), 13.11 MiB | 7.34 MiB/s, done.  
Resolving deltas: 100% (238/238), done.  
bilalhasan@LAPTOP-2E5R9ELL:~$ |
```

```
bilalhasan@LAPTOP-2E5R9ELL:~/go-ethereum-1.16.8$ make geth
```



```
bilalhasan@LAPTOP-2E5R9ELL:~/go-ethereum-1.16.8$ make geth  
go run build/ci.go install ./cmd/geth  
->> ./usr/local/go/bin/go build -ldflags "--buildid=None -extldflags '-Wl,-z,stack-size=0x800000,--build-id=None,--strip-all'" -tags urfave_cli_no_docs,ckzg -trimpath -v -o /home/bilalhasan/go-ethereum-1.16.8/build/bin/geth ./cmd/geth  
Done building.  
Run "./build/bin/geth" to launch geth.
```

```
bilalhasan@LAPTOP-2E5R9ELL:~/go-ethereum-1.16.8$ geth -v
```



```
bilalhasan@LAPTOP-2E5R9ELL:~/go-ethereum-1.16.8$ geth -v  
geth version 1.16.8-stable  
bilalhasan@LAPTOP-2E5R9ELL:~/go-ethereum-1.16.8$ |
```

The Genesis Block

All blockchains start with the genesis block. This block defines the initial configuration to a blockchain. The configuration to genesis block is defined in genesis.json file.

So, let's create a folder inside the web3 folder to start the private network and create the genesis.json file.

Start Database (The Bootnode)

The bootnode is the first node created when the blockchain is started. To start the database to first node, execute:

The folder node1 will be created with the database to bootnode.

The terminal result must be something like this:

```
bilalhasan@LAPTOP-2E5R9ELL:~$ cd blockchain
```



```
bilalhasan@LAPTOP-2E5R9ELL:~$ cd blockchain  
bilalhasan@LAPTOP-2E5R9ELL:~/blockchain$
```

Config genesis.json

```
# 1. Clean up old broken data  
pkill -f geth  
rm -rf node1 genesis.json start.sh  
# 2. Create password  
echo "bilal2000" > password.txt  
# 3. Create NEW account  
echo "Generating new account..."  
ADDRESS=$(geth account new --datadir node1 --password password.txt | grep "Public address" | awk '{print $6}')
```

```

# 4. Generate Genesis (Safe Mode: No Cancun)
python3 -c "
import json
address = '$ADDRESS'[2:]
extra_data = '0x' + ('0' * 64) + address.lower() + ('0' * 130)
genesis = {
    'config': { 'chainId': 6969, 'homesteadBlock': 0, 'eip150Block': 0, 'eip155Block': 0, 'eip158Block': 0,
    'byzantiumBlock': 0, 'constantinopleBlock': 0, 'petersburgBlock': 0, 'istanbulBlock': 0, 'berlinBlock': 0,
    'londonBlock': 0, 'shanghaiTime': 0, 'clique': { 'period': 5, 'epoch': 30000 } },
    'difficulty': '1', 'gasLimit': '30000000', 'alloc': { '$ADDRESS': { 'balance':
    '10000000000000000000000000000000' } }, 'extraData': extra_data
}
with open('genesis.json', 'w') as f: json.dump(genesis, f, indent=2)
"

# 5. Initialize
geth init --datadir node1 genesis.json
# 6. Create the Shortcut Script (start.sh)
echo "geth --datadir node1 --networkid 6969 --http --http.api 'eth,net,web3,personal,miner,clique' --
http.corsdomain '*' --allow-insecure-unlock --mine --miner.etherbase $ADDRESS --unlock
$ADDRESS --password password.txt console" > start.sh
chmod +x start.sh
echo "-----"
echo "  SETUP COMPLETE! Your address is: $ADDRESS"
echo "-----"

```

```

bilalhasan@LAPTOP-2E5R9ELL: ~ + 
INFO [01-24|20:56:04.830] Smartcard socket not found, disabling err="stat /run/pcscd/pcscd.comm: no such file or directory"
INFO [01-24|20:56:04.832] Set global gas cap cap=50,000,000
INFO [01-24|20:56:04.832] Initializing the KZG library backend=gokzg
INFO [01-24|20:56:04.847] Defaulting to pebble as the backing database database=/home/bilalhasan/node1/geth/chaindata cache=16.00MiB handles=16
INFO [01-24|20:56:04.847] Allocated cache and file handles database=/home/bilalhasan/node1/geth/chaindata/ancient scheme=hash
INFO [01-24|20:56:04.873] Opened ancient database nodes=1 size=149.00B time=1.056739ms gcnodes=0 gcsizes=0.00B gctime=0s livenodes=0 livesize=0.00B
INFO [01-24|20:56:04.874] State schema set to default database=chaindata hash=27e525..60a651
INFO [01-24|20:56:04.874] Writing custom genesis block database=chaindata hash=27e525..60a651
INFO [01-24|20:56:04.875] Persisted trie from memory database nodes=1 size=149.00B time=1.056739ms gcnodes=0 gcsizes=0.00B gctime=0s livenodes=0 livesize=0.00B
INFO [01-24|20:56:04.875] SuccessFully wrote genesis state database=chaindata hash=27e525..60a651
INFO [01-24|20:56:04.887] Defaulting to pebble as the backing database database=chaindata hash=27e525..60a651
INFO [01-24|20:56:04.887] Allocated cache and file handles database=/home/bilalhasan/node1/geth/lightchaindata cache=16.00MiB handles=16
INFO [01-24|20:56:04.915] Opened ancient database database=/home/bilalhasan/node1/geth/lightchaindata/ancient scheme=hash
INFO [01-24|20:56:04.915] State schema set to default database=lightchaindata hash=27e525..60a651
INFO [01-24|20:56:04.915] Writing custom genesis block database=lightchaindata hash=27e525..60a651
INFO [01-24|20:56:04.917] Persisted trie from memory database nodes=1 size=149.00B time=1.631827ms gcnodes=0 gcsizes=0.00B gctime=0s livenodes=0 livesize=0.00B
INFO [01-24|20:56:04.940] Successfully wrote genesis state database=lightchaindata hash=27e525..60a651
-----  

REPAIR COMPLETE! Run this command to start:  

geth --datadir node1 --networkid 6969 --http --http.api 'eth,net,web3,personal,miner,clique' --http.corsdomain '*' --allow-insecure-unlock --mine --miner.etherbase 0xc5Ac08783996a0994F3FB2045Eb9f8ED1438e54b --unlock 0xc5Ac08783996a0994F3FB2045Eb9f8ED1438e54b --password password.txt console  

bilalhasan@LAPTOP-2E5R9ELL:~$ |

```

Start Node

- This command will start the node in the private network with id 6969.
- The flag --http is used to enable Web App Access, we will use this in next post to connect Metamask with that private network.
- The flag --allow-insecure-unlock is used to permit execute transfers without a web application, we will use this in tests to this network.
- The flag --nodiscover is used to prevent node from trying to connect to others automatically.

The terminal result must be something like this:

```
bilalhasan@LAPTOP-2E5R9ELL:~/
```

```

bilalhasan@LAPTOP-2E5R9ELL:~/Desktop$ ./start.sh
INFO [01-24|21:18:46.220] Maximum peer count
INFO [01-24|21:18:46.221] Smartcard socket not found, disabling
category"
INFO [01-24|21:18:46.223] Set global gas cap
INFO [01-24|21:18:46.223] Initializing the KZG library
INFO [01-24|21:18:46.243] Allocated trie memory caches
INFO [01-24|21:18:46.243] Using pebble as the backing database
INFO [01-24|21:18:46.243] Allocated cache and file handles
512.0MiB handles=524,288
INFO [01-24|21:18:46.262] Opened ancient database
tchain readonly=false
INFO [01-24|21:18:46.262] State scheme set to already existing
INFO [01-24|21:18:46.263] Initialising Ethereum protocol
INFO [01-24|21:18:46.264]
INFO [01-24|21:18:46.264]

```

Done!! The blockchain in the private network is running.

Let's go do some tests to verify this. Open another terminal window in the same folder my blockchain and execute the following command:

This opens an interactive javascript terminal to execute some tasks to this node. The terminal result should be something like this:

```

Welcome to the Geth JavaScript console!

instance: Geth/v1.13.13-stable/linux-amd64/go1.21.13
coinbase: 0x72b386713c064c0fd0db6e48f8063152579eeff19
at block: 1 (Sat Jan 24 2026 21:18:46 GMT+0530 (IST))
datadir: /home/bilalhasan/node1
modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit press ctrl-d or type exit
> INFO [01-24|21:18:49.126] New local node record
14.71 udp=51505 tcp=30303
INFO [01-24|21:18:51.006] Successfully sealed new block
eLapsed=4.146s
INFO [01-24|21:18:51.007] Commit new sealing work
lapsed="427.998μs"
INFO [01-24|21:18:56.004] Successfully sealed new block
eLapsed=4.996s
INFO [01-24|21:18:56.004] Commit new sealing work
lapsed="232.754μs"
INFO [01-24|21:18:56.368] Looking for peers
INFO [01-24|21:19:01.006] Successfully sealed new block
eLapsed=5.001s
INFO [01-24|21:19:01.006] Commit new sealing work
lapsed="295.762μs"
INFO [01-24|21:19:06.007] Successfully sealed new block
eLapsed=5.000s
INFO [01-24|21:19:06.007] Commit new sealing work
lapsed="475.472μs"
INFO [01-24|21:19:06.845] Looking for peers
INFO [01-24|21:19:11.008] Successfully sealed new block
eLapsed=5.000s
INFO [01-24|21:19:11.009] Commit new sealing work
lapsed=1.031ms

```

Adding member peers

Let's start the second node to our blockchain. To do this execute the following commands in new terminal inside the same directory:

Make directory Node2 for adding member in peer

mkdir node2

create geth for node2

geth init --datadir node2 genesis.json

make start2.sh command for starting the second peer.

echo "geth --datadir node2 --networkid 6969 --port 30304 --http --http.addr 127.0.0.1 --http.port 8546 --http.api 'eth,net,web3,personal,miner,clique' --http.corsdomain '*' --authrpc.port 8552 console" > start2.sh

bilalhasan@LAPTOP-2E5R9ELL:~/Desktop\$./start2.sh

Node1

```

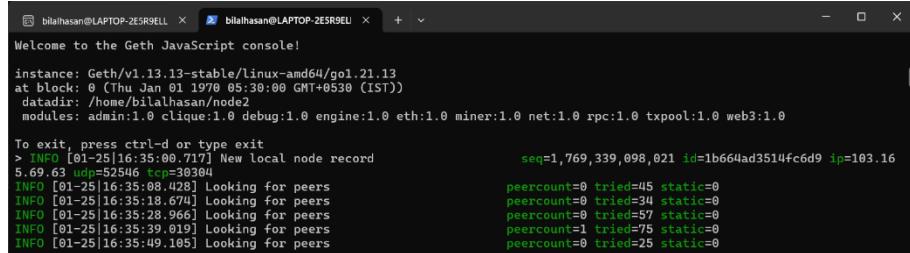
Welcome to the Geth JavaScript console!

instance: Geth/v1.13.13-stable/linux-amd64/go1.21.13
coinbase: 0x892321f4bd341a37e4be3fc530d48d241b254e8b
at block: 1 (Sun Jan 25 2026 16:34:30 GMT+0530 (IST))
datadir: /home/bilalhasan/node1
modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit press ctrl-d or type exit
> INFO [01-25|16:34:32.803] New local node record
5.69.63 udp=52543 tcp=30303
INFO [01-25|16:34:35.008] Successfully sealed new block
eLapsed=4.078s
INFO [01-25|16:34:35.008] Commit new sealing work
lapsed="478.722μs"

```

Node2



```
bilalhasan@LAPTOP-2E5R9ELI ~ bilalhasan@LAPTOP-2E5R9ELI ~ + 
Welcome to the Geth JavaScript console!

instance: Geth/v1.13.13-stable/linux-amd64/go1.21.13
at block: 0 (Thu Jan 01 1970 05:38:00 GMT+0530 (IST))
datadir: /home/bilalhasan/node2
modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> INFO [01-25|16:35:00.717] New local node record
5.69.63 udp=52946 tcp=30304
INFO [01-25|16:35:08.428] Looking for peers
INFO [01-25|16:35:18.674] Looking for peers
INFO [01-25|16:35:28.966] Looking for peers
INFO [01-25|16:35:39.019] Looking for peers
INFO [01-25|16:35:49.105] Looking for peers
peercount=0 tried=45 static=0
peercount=0 tried=34 static=0
peercount=0 tried=57 static=0
peercount=1 tried=75 static=0
peercount=0 tried=25 static=0
```

Connecting Peers:

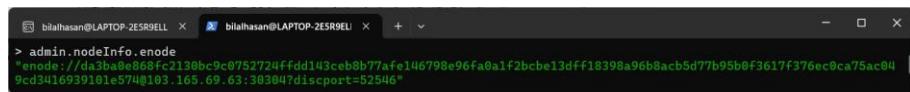
Now, let's create the peer-to-peer connection between the two nodes in blockchain.

Copy the enode from the nodeInfo of the second node, then execute the following command in javascript terminal of the first node

For Connecting both node:

In Node2 terminal get the enode using below command:

```
> admin.nodeInfo.enode
```



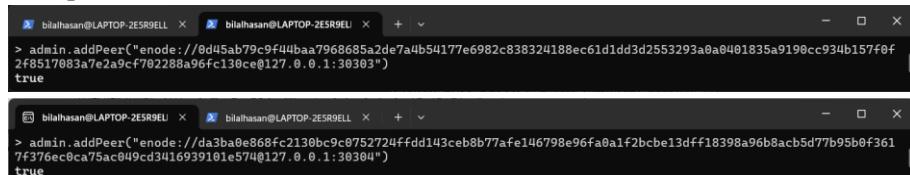
```
> admin.nodeInfo.enode
"enode://da3ba0e868fc2130bc9c0752724ffdd143ceb8b77afe146798e96fa0alf2bcbe13dff18398a96b8acb5d77b95b0f3617f376ec0ca75ac04
9cd3416939101e574@103.165.69.63:30304?discport=52946"
```

Copy the enode from the nodeInfo of the first node, then execute the following command in javascript terminal of the second node:

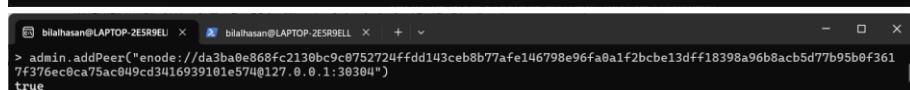
In Node1 terminal put the below command to link both node:

```
>admin.addPeer("enode://0d45ab79c9f44baa7968685a2de7a4b54177e6982c838324188ec61d1dd3d2
553293a0a0401835a9190cc934b157f0f2f8517083a7e2a9cf702288a96fc130ce@127.0.0.1:30303")
```

```
>net.peerCount
```



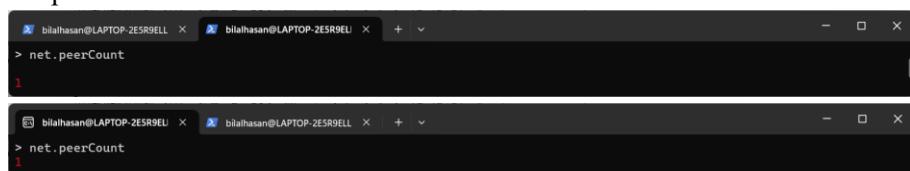
```
> admin.addPeer("enode://0d45ab79c9f44baa7968685a2de7a4b54177e6982c838324188ec61d1dd3d2553293a0a0401835a9190cc934b157f0f
2f8517083a7e2a9cf702288a96fc130ce@127.0.0.1:30303")
true
```

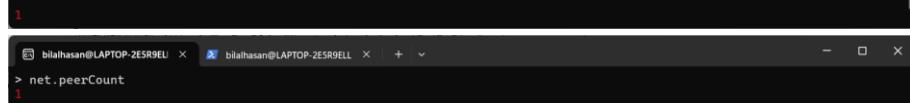
```
> admin.addPeer("enode://da3ba0e868fc2130bc9c0752724ffdd143ceb8b77afe146798e96fa0alf2bcbe13dff18398a96b8acb5d77b95b0f361
7f376ec0ca75ac049cd3416939101e574@127.0.0.1:30304")
true
```

For Verifying the both connected on the same block chain the Output get from the below command should be 1.

```
net.peerCount
```

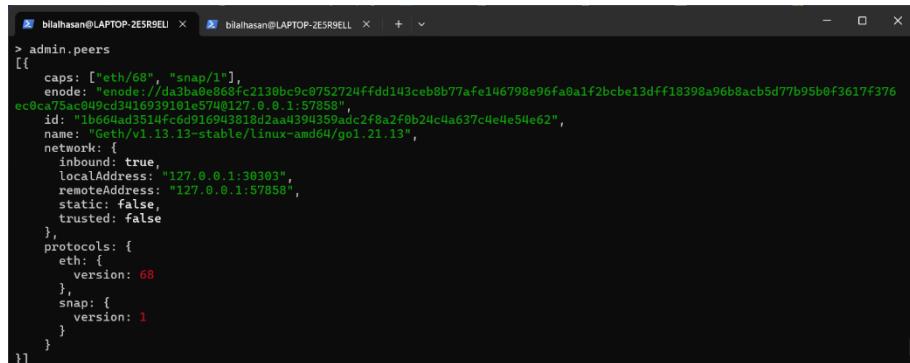


```
> net.peerCount
1
```

```
> net.peerCount
1
```

Node1:



```
> admin.peers
[{
  caps: ["eth/68", "snap/1"],
  enode: "enode://da3ba0e868fc2130bc9c0752724ffdd143ceb8b77afe146798e96fa0alf2bcbe13dff18398a96b8acb5d77b95b0f3617f376
ec0ca75ac049cd3416939101e574@127.0.0.1:30303",
  id: "1b664ad3514fc6d916943818d2aa394359adc2f8a2f0b24c4a637c4e4e54e62",
  name: "Geth/v1.13.13-stable/linux-amd64/go1.21.13",
  network: {
    inbound: true,
    localAddress: "127.0.0.1:30303",
    remoteAddress: "127.0.0.1:57858",
    static: false,
    trusted: false
  },
  protocols: {
    eth: {
      version: 68
    },
    snap: {
      version: 1
    }
  }
}]
```

Node2:

```
bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ + 
> admin.peers
[{
  caps: ["eth/68", "snap/1"],
  enode: "enode://0d45ab79cf4bba7968685a2de7a4b54177e6982c838324188ec61d1dd3d2553293a0a0401835a9190cc934b157f0f2f8517083a7e2a9c7f02288a96fc130ce@127.0.0.1:30303",
  id: "6b321ca0bcfb504d5c39483882f098b16dd092f805966d9e9c792594076aeef6f",
  name: "Geth/v1.13.13-stable/linux-amd64/go1.21.13",
  network: {
    inbound: false,
    localAddress: "127.0.0.1:57858",
    remoteAddress: "127.0.0.1:30303",
    static: true,
    trusted: false
  },
  protocols: {
    eth: {
      version: 68
    },
    snap: {
      version: 1
    }
  }
}]
```

Done!! Your blockchain is created with multiple nodes running and connected.

Let's create accounts and start mining to see data updated in both nodes.

Mining

To mine blocks it's necessary to have a base account. Let's do this. In the javascript terminal of the first node, execute the command to create a new account and define the password required to this account, the public address from the created account should be presented.

Open New Terminal for WSL

```
> bilalhasan@LAPTOP-2E5R9ELL:~geth account new --datadir node1e1
```

```
bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ + 
> bilalhasan@LAPTOP-2E5R9ELL:~geth account new --datadir node1e1
INFO [01-25|19:06:02.654] Maximum peer count          ETH=50 total=50
INFO [01-25|19:06:02.656] Smartcard socket not found, disabling   err="stat /run/pcscd/pcscd.comm: no such file or directory"
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:
Your new key was generated

Public address of the key: 0x62f66f95de6d717e8b27fd159b145Dba1F99a2c
Path of the secret key file: node1/keystore/UTC--2026-01-25T13-36-11.162478013Z--62f66f95de6d717e8b27fdc159b145dba1f99a2c

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

```
bilalhasan@LAPTOP-2E5R9ELL:~$ geth attach node1/geth.ipc
```

```
bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ + 
bilalhasan@LAPTOP-2E5R9ELL:~$ geth attach node1/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.13.13-stable/linux-amd64/go1.21.13
coinbase: 0x992321f4bd341a37e4be3fc530d48d241b254e8b
at block: 1243 (Sun Jan 25 2026 19:06:18 GMT+0530 (IST))
datadir: /home/bilalhasan/node1
modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
```

In the “>” function type this command:

```
> miner.setEtherbase(eth.accounts[0])
```

```
bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ + 
> miner.setEtherbase(eth.accounts[0])
true
```

```
> miner.start()
```

```
bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ + 
> miner.start()
null
```

```
>miner.stop()
```

```
bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ + 
> miner.stop()
null
```

Eth.blockNumber

```
bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELI ~ + - □ ×  
> eth.blockNumber  
1255  
  
> web3.fromWei(eth.getBalance(eth.accounts[0]), "ether")  
bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELL ~ bilalhasan@LAPTOP-2E5R9ELI ~ + - □ ×  
> web3.fromWei(eth.getBalance(eth.accounts[0]), "ether")  
1000
```

Practical No. 4

Aim - Deploy an asset-transfer app using block chain. Learn app development within a Hyperledger Fabric network.

Objective: To understand the fundamental operations of a ledger (Create, Read, Update, Delete) within a Hyperledger Fabric network.

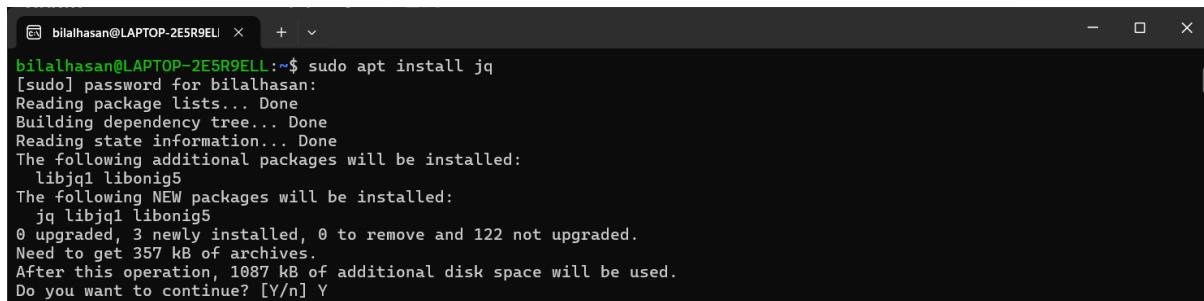
Theory:

- Hyperledger Fabric Architecture: Unlike public blockchains (like Bitcoin), Fabric is a permissioned network. All participants are known and authenticated via an MSP (Membership Service Provider).
- Asset Transfer Concept: In blockchain, an "asset" can be tangible (a house, car) or intangible (intellectual property, stocks). The "Transfer" logic is handled by Chaincode (Smart Contracts).
- The Ledger: The Fabric ledger consists of two parts:
 1. World State: A database (typically CouchDB or LevelDB) that holds the current value of an asset.
 2. Blockchain (Transaction Log): An immutable history of all transactions that resulted in the current world state.
- Smart Contract Role: The application logic defines the asset (e.g., AssetID, Owner, Color, Size) and the rules for changing ownership.

Steps:

Installing dependencies

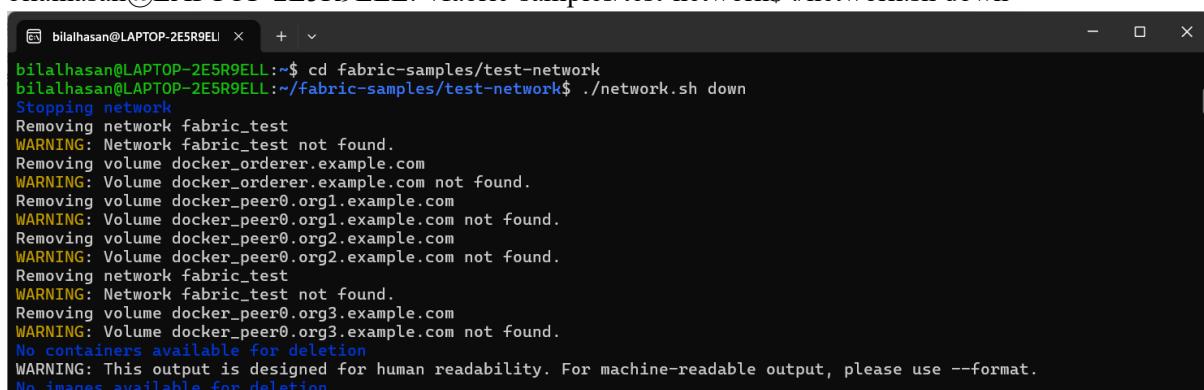
```
bilalhasan@LAPTOP-2E5R9ELL:~$ sudo apt install jq
```



```
bilalhasan@LAPTOP-2E5R9ELL:~$ sudo apt install jq
[sudo] password for bilalhasan:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libjq1 libonig5
The following NEW packages will be installed:
  jq libjq1 libonig5
0 upgraded, 3 newly installed, 0 to remove and 122 not upgraded.
Need to get 357 kB of archives.
After this operation, 1087 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

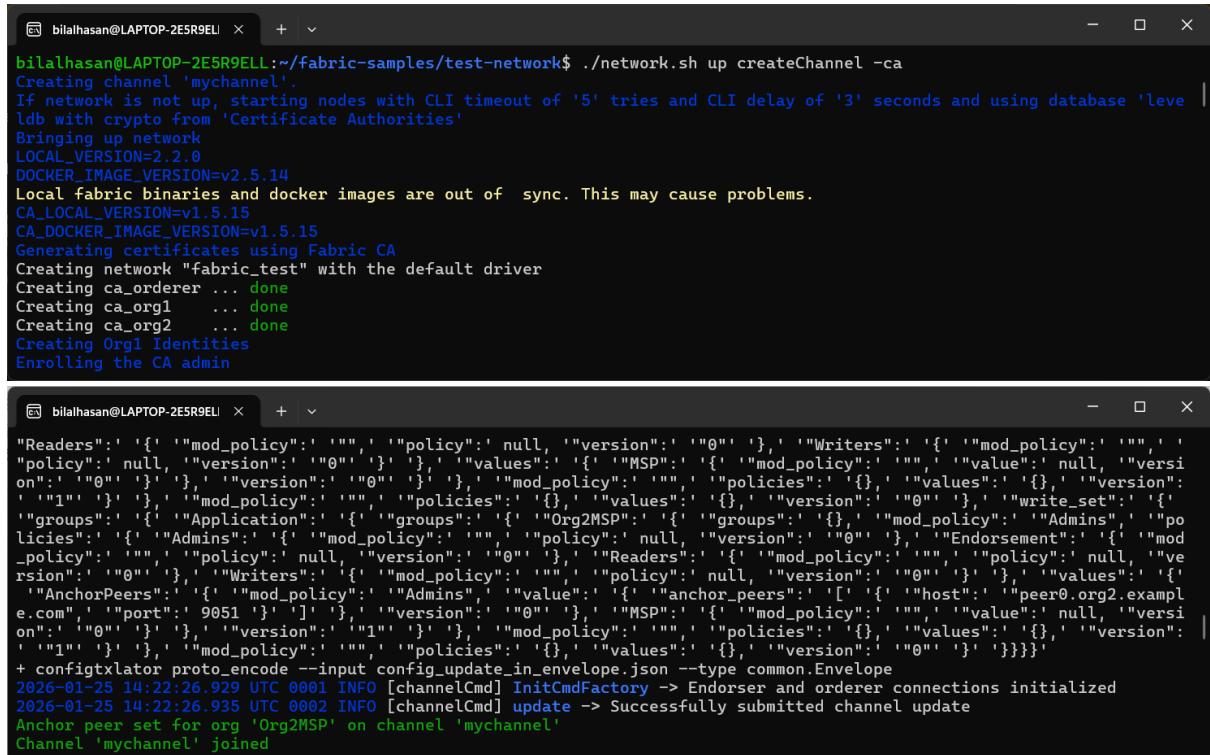
Start the network and deploy the smart contract

```
bilalhasan@LAPTOP-2E5R9ELL:~$ cd fabric-samples/test-network
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ ./network.sh down
```



```
bilalhasan@LAPTOP-2E5R9ELL:~$ cd fabric-samples/test-network
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ ./network.sh down
Stopping network
Removing network fabric_test
WARNING: Network fabric_test not found.
Removing volume docker_orderer.example.com
WARNING: Volume docker_orderer.example.com not found.
Removing volume docker_peer0.org1.example.com
WARNING: Volume docker_peer0.org1.example.com not found.
Removing volume docker_peer0.org2.example.com
WARNING: Volume docker_peer0.org2.example.com not found.
Removing network fabric_test
WARNING: Network fabric_test not found.
Removing volume docker_peer0.org3.example.com
WARNING: Volume docker_peer0.org3.example.com not found.
No containers available for deletion
WARNING: This output is designed for human readability. For machine-readable output, please use --format.
No images available for deletion
```

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ ./network.sh up createChannel -ca
```



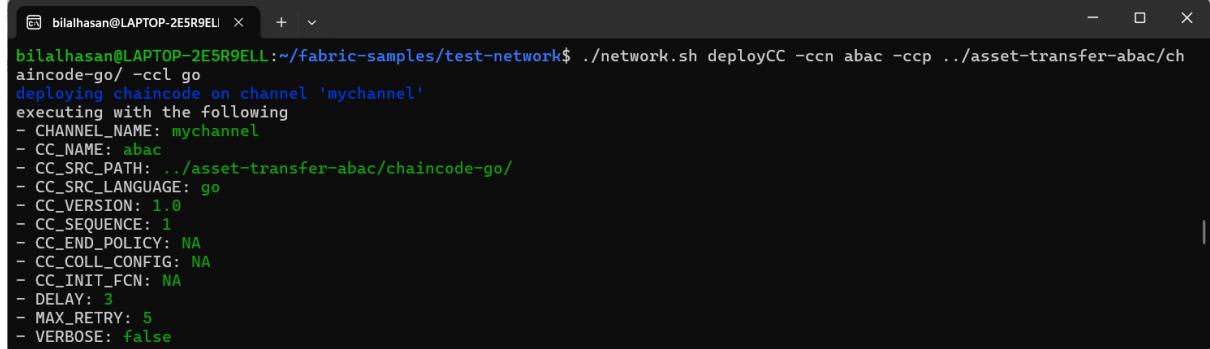
```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ ./network.sh up createChannel -ca
Creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb with crypto from 'Certificate Authorities'
Bringing up network
LOCAL_VERSION=2.2.0
DOCKER_IMAGE_VERSION=v2.5.14
Local fabric binaries and docker images are out of sync. This may cause problems.
CA_LOCAL_VERSION=v1.5.15
CA_DOCKER_IMAGE_VERSION=v1.5.15
Generating certificates using Fabric CA
Creating network "fabric_test" with the default driver
Creating ca_orderer ... done
Creating ca_org1 ... done
Creating ca_org2 ... done
Creating Org1 Identities
Enrolling the CA admin
```



```
"Readers": [{"mod_policy": {"policy": null, "version": "0"}, "Writers": [{"mod_policy": {"value": null, "version": "0"}}, {"values": [{"MSP": {"mod_policy": {"value": null, "version": "0"}, "policies": [{"values": [{"version": "1"}]}], "mod_policy": {"policies": [{"values": [{"version": "0"}]}], "write_set": [{"groups": [{"Application": {"groups": [{"Org2MSP": {"groups": [{"Admins": [{"mod_policy": {"policy": null, "version": "0"}, "Endorsement": [{"mod_policy": {"policy": null, "version": "0"}, "Readers": [{"mod_policy": {"policy": null, "version": "0"}, "Writers": [{"mod_policy": {"value": null, "version": "0"}, "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"}, "policies": [{"values": [{"version": "1"}]}], "mod_policy": {"policies": [{"values": [{"version": "0"}]}]}}, {"configtxlator proto_encode --input config_update_in_envelope.json --type common.Envelope
2026-01-25 14:22:26.929 UTC 0001 [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2026-01-25 14:22:26.935 UTC 0002 [channelCmd] update -> Successfully submitted channel update
Anchor peer set for org 'Org2MSP' on channel 'mychannel'
Channel 'mychannel' joined
```

You can then use the test network script to deploy the asset-transfer-abac smart contract to a channel on the network:

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ ./network.sh deployCC -ccn abac -ccp .../asset-transfer-abac/chaincode-go/ -ccl go
```



```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ ./network.sh deployCC -ccn abac -ccp .../asset-transfer-abac/chaincode-go/ -ccl go
deploying chaincode on channel 'mychannel'
executing with the following
- CHANNEL_NAME: mychannel
- CC_NAME: abac
- CC_SRC_PATH: .../asset-transfer-abac/chaincode-go/
- CC_SRC_LANGUAGE: go
- 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
```

Register identities with attributes

We can use the one of the test network Certificate Authorities to register and enroll identities with the attribute of abac.creator=true. First, we need to set the following environment variables in order to use the Fabric CA client.

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$
```

```
export PATH=${PWD}/..bin:${PWD}:$PATH
```

```
export FABRIC_CFG_PATH=$PWD/..config/
```

We will create the identities using the Org1 CA. Set the Fabric CA client home to the MSP of the Org1 CA admin:

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ export
```

```
FABRIC_CA_CLIENT_HOME=${PWD}/organizations/peerOrganizations/org1.example.com/
```

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ export PATH=$PWD/../bin:$PWD:$PATH
export FABRIC_CFG_PATH=$PWD/../config/
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ export FABRIC_CA_CLIENT_HOME=${PWD}/organizations/peerOrganizations/org1.example.com/

```

There are two ways to generate certificates with attributes added. We will use both methods and create two identities in the process. The first method is to specify that the attribute be added to the certificate by default when the identity is registered. The following command will register an identity named creator1 with the attribute of abac.creator=true.

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ fabric-ca-client register --id.name creator1 --id.secret creator1pw --id.type client --id.affiliation org1 --id.attrs 'abac.creator=true:ecert' --tls.certfiles "${PWD}/organizations/fabric-ca/org1/tls-cert.pem"

```

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ fabric-ca-client enroll -u https://creator1:creator1pw@localhost:7054 --caname ca-org1 -M "${PWD}/organizations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp" --tls.certfiles "${PWD}/organizations/fabric-ca/org1/tls-cert.pem"

```

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ cp "${PWD}/organizations/peerOrganizations/org1.example.com/msp/config.yaml" "${PWD}/organizations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp/config.yaml"

```

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ fabric-ca-client register --id.name creator1 --id.secret creator1pw --id.type client --id.affiliation org1 --id.attrs 'abac.creator=true:ecert' --tls.certfiles "${PWD}/organizations/fabric-ca/org1/tls-cert.pem"
2026/01/25 19:54:04 [INFO] Configuration file location: /home/bilalhasan/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
2026/01/25 19:54:04 [INFO] TLS Enabled
2026/01/25 19:54:04 [INFO] TLS Enabled
Password: creator1pw
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ fabric-ca-client enroll -u https://creator1:creator1pw@localhost:7054 --caname ca-org1 -M "${PWD}/organizations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp" --tls.certfiles "${PWD}/organizations/fabric-ca/org1/tls-cert.pem"
2026/01/25 19:54:41 [INFO] TLS Enabled
2026/01/25 19:54:41 [INFO] generating key: &{A:ecdsa S:256}
2026/01/25 19:54:41 [INFO] encoded CSR
2026/01/25 19:54:41 [INFO] Stored client certificate at /home/bilalhasan/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp/signcerts/cert.pem
2026/01/25 19:54:41 [INFO] Stored root CA certificate at /home/bilalhasan/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
2026/01/25 19:54:41 [INFO] Stored Issuer public key at /home/bilalhasan/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp/IssuerPublicKey
2026/01/25 19:54:41 [INFO] Stored Issuer revocation public key at /home/bilalhasan/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp/IssuerRevocationPublicKey
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ cp "${PWD}/organizations/peerOrganizations/org1.example.com/msp/config.yaml" "${PWD}/organizations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp/config.yaml"

```

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ fabric-ca-client register --id.name creator2 --id.secret creator2pw --id.type client --id.affiliation org1 --id.attrs 'abac.creator=true:' --tls.certfiles "${PWD}/organizations/fabric-ca/org1/tls-cert.pem"

```

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ fabric-ca-client enroll -u https://creator2:creator2pw@localhost:7054 --caname ca-org1 --enrollment.attrs "abac.creator" -M "${PWD}/organizations/peerOrganizations/org1.example.com/users/creator2@org1.example.com/msp" --tls.certfiles "${PWD}/organizations/fabric-ca/org1/tls-cert.pem"

```

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ cp "${PWD}/organizations/peerOrganizations/org1.example.com/msp/config.yaml" "${PWD}/organizations/peerOrganizations/org1.example.com/users/creator2@org1.example.com/msp/config.yaml"

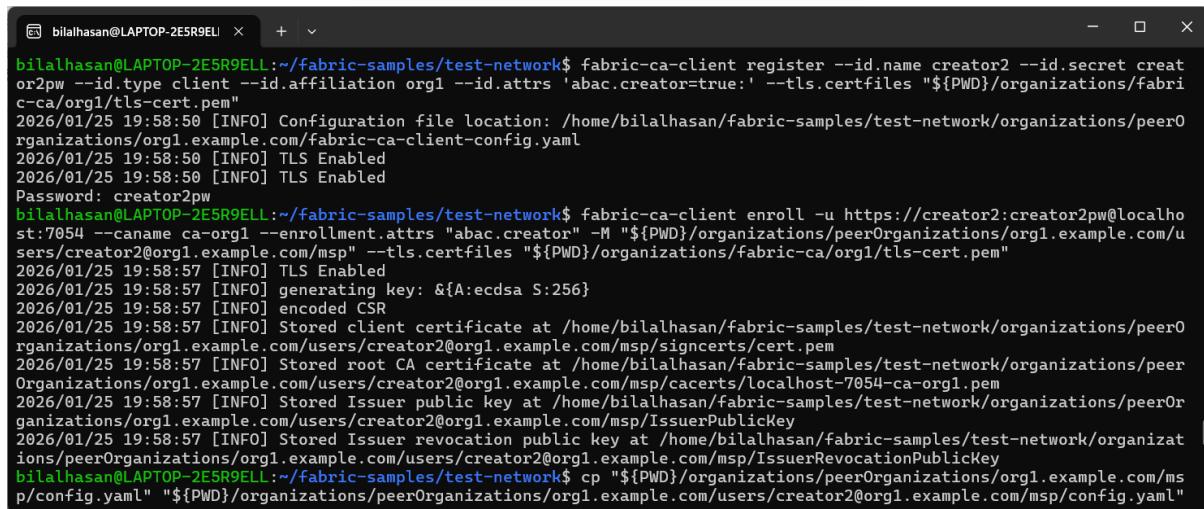
```

The ecert suffix adds the attribute to the certificate automatically when the identity is enrolled. As a result, the following enroll command will contain the attribute that was provided in the registration command.

The second method is to request that the attribute be added upon enrollment. The following command will register an identity named creator2 with the same abac.creator attribute.

The following enroll command will add the attribute to the certificate:

Run the command below to copy the Node OU configuration file into the creator2 MSP folder.



```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ fabric-ca-client register --id.name creator2 --id.secret creat or2pw --id.type client --id.affiliation org1 --id.attrs 'abac.creator=true:' --tls.certfiles "${PWD}/organizations/fabri c-ca/org1/tls-cert.pem"
2026/01/25 19:58:50 [INFO] Configuration file location: /home/bilalhasan/fabric-samples/test-network/organizations/peer0/organizations/org1.example.com/fabric-ca-client-config.yaml
2026/01/25 19:58:50 [INFO] TLS Enabled
2026/01/25 19:58:50 [INFO] TLS Enabled
Password: creator2pw
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ fabric-ca-client enroll -u https://creator2:creator2pw@localhost:7054 --caname ca-org1 --enrollment.attrs "abac.creator" -M "${PWD}/organizations/peerOrganizations/org1.example.com/users/creator2@org1.example.com/msp" --tls.certfiles "${PWD}/organizations/fabric-ca/org1/tls-cert.pem"
2026/01/25 19:58:57 [INFO] TLS Enabled
2026/01/25 19:58:57 [INFO] generating key: &{A:ecdsa S:256}
2026/01/25 19:58:57 [INFO] encoded CSR
2026/01/25 19:58:57 [INFO] Stored client certificate at /home/bilalhasan/fabric-samples/test-network/organizations/peer0/organizations/org1.example.com/users/creator2@org1.example.com/msp/signcerts/cert.pem
2026/01/25 19:58:57 [INFO] Stored root CA certificate at /home/bilalhasan/fabric-samples/test-network/organizations/peer0/organizations/org1.example.com/users/creator2@org1.example.com/msp/cacerts/localhost-7054-ca-org1.pem
2026/01/25 19:58:57 [INFO] Stored Issuer public key at /home/bilalhasan/fabric-samples/test-network/organizations/peer0/organizations/org1.example.com/users/creator2@org1.example.com/msp/IssuerPublicKey
2026/01/25 19:58:57 [INFO] Stored Issuer revocation public key at /home/bilalhasan/fabric-samples/test-network/organizations/peer0/organizations/org1.example.com/users/creator2@org1.example.com/msp/IssuerRevocationPublicKey
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ cp "${PWD}/organizations/peerOrganizations/org1.example.com/msp/config.yaml" "${PWD}/organizations/peerOrganizations/org1.example.com/users/creator2@org1.example.com/msp/config.yaml"
```

Create an asset:

You can use either identity with the abac.creator=true attribute to create an asset using the asset-transfer-abac smart contract. We will set the following environment variables to use the first identity that was generated, creator1:

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ export
CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org1MSP"
export
CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp
export
CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER_ADDRESS=localhost:7051
export TARGET_TLS_OPTIONS=(-o localhost:7050 --ordererTLSHostnameOverride
orderer.example.com --tls --cafile
"${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" --peerAddresses localhost:7051 --tlsRootCertFiles
"${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles
"${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt")
```

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ export CORE_PEER_TLS_ENABLED=true
export CORE_PEER_LOCALMSPID="Org1MSP"
export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp
export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER_ADDRESS=localhost:7051
export TARGET_TLS_OPTIONS=(-o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscerts/tlsca.example.com-cert.pem" --peerAddresses localhost:7051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt" --peerAddresses localhost:9051 --tlsRootCertFiles "${PWD}/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt")

```

Run the following command to create Asset1:

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode invoke
"${TARGET_TLS_OPTIONS[@]}" -C mychannel -n abac -c
'{"function":"CreateAsset","Args":["Asset1","blue","20","100"]}'

```

The result will list the creator1 identity as the asset owner.

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode invoke "${TARGET_TLS_OPTIONS[@]}" -C mychannel -n abac -c '{"function":"CreateAsset","Args":["Asset1","blue","20","100"]}'
2026-01-25 19:59:28.367 IST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status | s:200

```

Transfer the asset:

As the owner of Asset1, the creator1 identity has the ability to transfer the asset to another owner. In order to transfer the asset, the owner needs to provide the name and issuer of the new owner to the TransferAsset function. The asset-transfer-abac smart contract has a GetSubmittingClientIdentity function that allows users to retrieve their certificate information and provide it to the asset owner out of band (we omit this step). Issue the command below to transfer Asset1 to the user1 identity from Org1 that was created when the test network was deployed:

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ export
RECIPIENT="x509::CN=user1,OU=client,O=Hyperledger,ST=North
Carolina,C=US::CN=ca.org1.example.com,O=org1.example.com,L=Durham,ST=North
Carolina,C=US"

```

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode invoke
"${TARGET_TLS_OPTIONS[@]}" -C mychannel -n abac -c
'{"function":"TransferAsset","Args":["Asset1","$$RECIPIENT"]}'"

```

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ export RECIPIENT="x509::CN=user1,OU=client,O=Hyperledger,ST=North
Carolina,C=US::CN=ca.org1.example.com,O=org1.example.com,L=Durham,ST=North Carolina,C=US"
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode invoke "${TARGET_TLS_OPTIONS[@]}" -C mychannel -n abac -c '{"function":"TransferAsset","Args":["Asset1","$$RECIPIENT"]}'"
2026-01-25 20:01:14.808 IST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status | s:200

```

Query the ledger to verify that the asset has a new owner:

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n abac -c '{"function":"ReadAsset","Args":["Asset1"]}'"

```

We can see that Asset1 with is now owned by User1:

```

bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n abac -c '{"function":"ReadAsset","Args":["Asset1"]}'"
{"ID":"Asset1","color":"blue","size":20,"owner":"x509::CN=user1,OU=client,O=Hyperledger,ST=North Carolina,C=US::CN=ca.org1.example.com,O=org1.example.com,L=Durham,ST=North Carolina,C=US","appraisedValue":100}

```

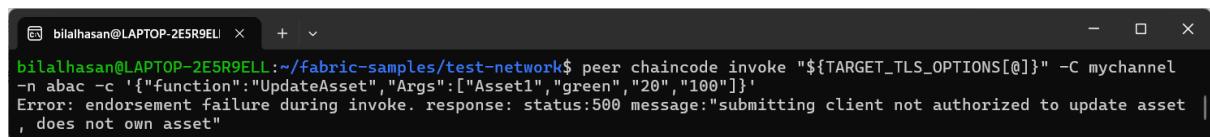
Update the asset:

Now that the asset has been transferred, the new owner can update the asset properties. The smart contract uses the GetID() API to ensure that the update is being submitted by the asset owner. To demonstrate the difference between identity and attribute based access control, lets try to update the asset using the creator1 identity first:

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode invoke  
"${TARGET_TLS_OPTIONS[@]}" -C mychannel -n abac -c  
'{"function":"UpdateAsset","Args":["Asset1","green","20","100"]}'
```

Even though creator1 can create new assets, the smart contract detects that the transaction was not submitted by the identity that owns the asset, user1. The command returns the following error:

```
Error: endorsement failure during invoke. response: status:500 message:"submitting client not authorized to update asset, does not own asset"
```



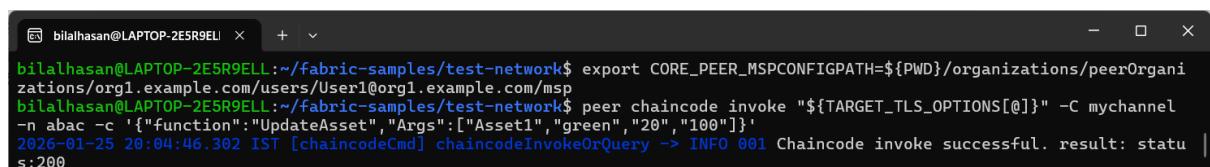
A terminal window titled 'bilalhasan@LAPTOP-2E5R9ELL' showing the command execution. The output shows an error message: 'endorsement failure during invoke. response: status:500 message:"submitting client not authorized to update asset , does not own asset"'.

Run the following command to operate as the asset owner by setting the MSP path to User1:

We can now update the asset. Run the following command to change the asset color from blue to green. All other aspects of the asset will remain unchanged.

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ export  
CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/us  
ers/User1@org1.example.com/msp
```

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode invoke  
"${TARGET_TLS_OPTIONS[@]}" -C mychannel -n abac -c  
'{"function":"UpdateAsset","Args":["Asset1","green","20","100"]}'
```

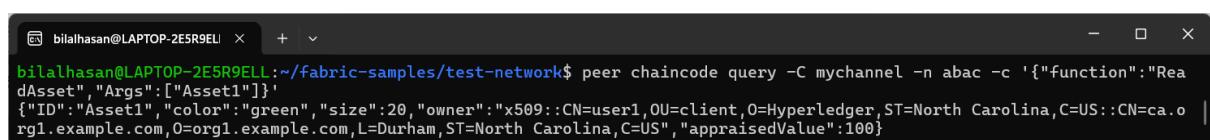


A terminal window titled 'bilalhasan@LAPTOP-2E5R9ELL' showing the command execution. The output shows a success message: 'Chaincode invoke successful. result: status:200'.

Run the query command again to verify that the asset has changed color:

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode query -C mychannel  
-n abac -c '{"function":"ReadAsset","Args":["Asset1"]}'
```

The result will display that Asset1 is now green:



A terminal window titled 'bilalhasan@LAPTOP-2E5R9ELL' showing the command execution. The output shows the asset details: 'Asset1', 'color': 'green', 'size': 20, 'owner': 'x509::CN=user1,OU=client,O=Hyperledger,ST=North Carolina,C=US::CN=ca.o rg1.example.com,L=Durham,ST=North Carolina,C=US', 'appraisedValue': 100}.

Delete the asset:

The owner also has the ability to delete the asset. Run the following command to remove Asset1 from the ledger:

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode invoke
"${TARGET_TLS_OPTIONS[@]}" -C mychannel -n abac -c
'{"function":"DeleteAsset","Args":["Asset1"]}'
```

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode invoke "${TARGET_TLS_OPTIONS[@]}" -C mychannel -n abac -c '{"function":"DeleteAsset","Args":["Asset1"]}'
```

2026-01-25 20:07:06.694 IST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status | s:200

If you query the ledger once more, you will see that Asset1 no longer exists:

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n abac -c '{"function":"ReadAsset","Args":["Asset1"]}'
```

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode query -C mychannel -n abac -c '{"function":"ReadAsset","Args":["Asset1"]}'
```

Error: endorsement failure during query. response: status:500 message:"the asset Asset1 does not exist"

While we are operating as User1, we can demonstrate attribute based access control by trying to create an asset using an identity without the abac.creator=true attribute. Run the following command to try to create Asset1 as User1:

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode invoke
"${TARGET_TLS_OPTIONS[@]}" -C mychannel -n abac -c
'{"function":"CreateAsset","Args":["Asset2","red","20","100"]}'
```

The smart contract will return the following error:

Error: endorsement failure during invoke. response: status:500 message:"submitting client not authorized to create asset, does not have abac.creator role"

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ peer chaincode invoke "${TARGET_TLS_OPTIONS[@]}" -C mychannel -n abac -c '{"function":"CreateAsset","Args":["Asset2","red","20","100"]}'
```

Error: endorsement failure during invoke. response: status:500 message:"submitting client not authorized to create asset | , does not have abac.creator role"

Clean up

When you are finished, you can run the following command to bring down the test network:

```
bilalhasan@LAPTOP-2E5R9ELL:~/fabric-samples/test-network$ ./network.sh down
```

```
Stopping network
Stopping cli      ... done
Stopping orderer.example.com ... done
Stopping peer0.org2.example.com ... done
Stopping peer0.org1.example.com ... done
Stopping ca_org2   ... done
Stopping ca_org1   ... done
Stopping ca_orderer ... done
Removing cli       ... done
Removing orderer.example.com ... done
Removing peer0.org2.example.com ... done
Removing peer0.org1.example.com ... done
Removing ca_org2   ... done
Removing ca_org1   ... done
Removing ca_orderer ... done
Removing network fabric_test
Removing volume docker_orderer.example.com
Removing volume docker_peer0.org1.example.com
Removing volume docker_peer0.org2.example.com
Removing network fabric_test
WARNING: Network fabric_test not found.
Removing volume docker_peer0.org3.example.com
WARNING: Volume docker_peer0.org3.example.com not found.
No containers available for deletion
WARNING: This output is designed for human readability. For machine-readable output, please use --format.
Error response from daemon: invalid reference format: repository name (library/36.2MB) must be lowercase
Error response from daemon: invalid reference format: repository name (library/36.2MB) must be lowercase
```

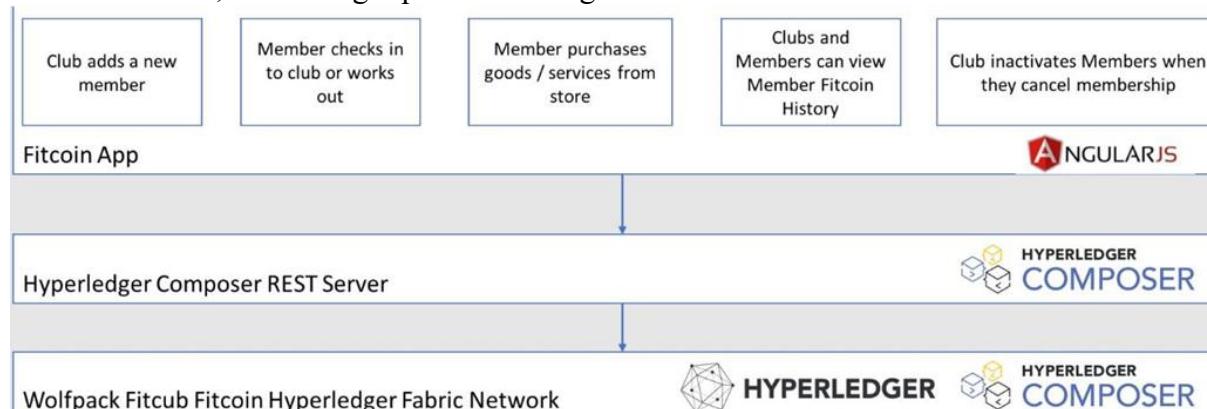
Practical No. 5

Aim - Use block chain to track fitness club rewards.

Objective: To implement a loyalty or rewards program where points are immutable and transparent.

Theory:

- Tokenization of Value: Rewards points are digital assets. Using blockchain ensures that points cannot be double-spent or forged.
- Identity & Membership: In a fitness club scenario, the blockchain verifies the identity of the member (User) and the club (Admin).
- Transaction Flow:
 - Earn: A member completes a workout -> IoT sensor or manual entry triggers a transaction -> Smart contract calculates points -> Ledger is updated.
 - Burn (Redeem): Member requests a reward -> Smart contract checks balance -> Deducts points -> Records exchange.
- Benefit: This eliminates disputes between the club and members regarding point balances, as the ledger provides a single source of truth.



Steps:

1. Install Hyperledger Fabric and Composer.

```
rdnc@ubuntu:~/Desktop$ curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh && chmod u+x prereqs-ubuntu.sh
% Total    % Received % Xferd  Average Speed   Time     Time     Time  Current
          Dload  Upload Total   Total  Spent   Left  Speed
100  4001  100  4001    0      0  9830      0 --:--:-- --:--:-- --:--:--  9830
```

2. Build and Deploy the Fitcoin Blockchain Network.

3. Build and run the Fitcoin Angular Web App.

Note:

If you're running on Ubuntu, you can download the prerequisites using the following commands:

```
curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh
chmod u+x prereqs-ubuntu.sh
```

Next run the script - as this briefly uses sudo during its execution, you will be prompted for your password.

```
./prereqs-ubuntu.sh
```

Install the latest (long term support) version of Node:

```
nvm install --lts
```

Switch to the LTS version of Node:

```
nvm use --lts
```

Check that Node is installed:

```
node --version
```

```
(fade㉿kali)-[~]
$ nvm install v20.11.1
Downloading and installing node v20.11.1 ...
Downloading https://nodejs.org/dist/v20.11.1/node-v20.11.1-linu
x-x64.tar.xz ...
#####
Computing checksum with sha256sum
Checksums matched!
Now using node v20.11.1 (npm v10.2.4)
Creating default alias: default → v20.11.1
```

Practical No. 6

Aim - Build a web app that uses Hyperledger Fabric to track and trace member rewards.

Objective: To learn how to build a user-friendly frontend (Web App) that interacts with the backend blockchain network.

Theory:

- Application Layers:
 1. Frontend: HTML/CSS/JS interface (e.g., Angular, React) where the user clicks "Check Balance".
 2. Middleware (API Layer): Usually a Node.js Express server. It receives the request from the frontend and uses the Hyperledger Fabric SDK to talk to the blockchain.
 3. Blockchain Network: The peers and orderers that execute the transaction.
- Track and Trace: This concept is vital for supply chains but applied here to rewards. It allows an admin to query the history of a specific member's rewards (e.g., "Show me the entire history of Member A's points from Jan to Dec").
- Fabric Node SDK: A library that allows the application to connect to the Gateway, submit transactions, and listen for events (e.g., notifying the app when a reward is successfully issued).

Step:

Creating the smart contract project

Installing hardhart

```
mkdir smart-contract
```

```
cd smart-contract
```

```
npm i -D hardhat
```

Creating a hardhat project

```
npx hardhat init
```

Choose TypeScript

```
? What do you want to do? ...
  Create a JavaScript project
▶ Create a TypeScript project
  Create a TypeScript project (with Viem)
  Create an empty hardhat.config.js
  Quit
```

Rest can be left default

```
✓ What do you want to do? · Create a TypeScript project
✓ Hardhat project root: · /home/hackerman/web3/prac7/smart-contract
✓ Do you want to add a .gitignore? (Y/n) · y
✓ Help us improve Hardhat with anonymous crash reports & basic usage data? (Y/n)
· n
```

Contents after project initialization

```
ls
```

```
→ smart-contract ls
contracts      node_modules  package-lock.json  scripts  tsconfig.json
hardhat.config.ts  package.json  README.md        test
→ smart-contract
```

Writing the smart contract:

Navigate the contracts directory and delete the existing Contract Lock.sol and

Create a new Solidity file LoyaltyPoints.sol

rm contracts/Lock.sol

touch contracts/LoyaltyPoints.sol

ls ./contracts

```
→ smart-contract rm contracts/Lock.sol
→ smart-contract touch contracts/LoyaltyPoints.sol
→ smart-contract ls ./contracts
LoyaltyPoints.sol
```

Open the project in an editor and append the below code in LoyaltyPoints.sol

Code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract LoyaltyPoints {
    // model a member
    struct Member {
        address memberAddress;
        string firstName;
        string lastName;
        string email;
        uint points;
        bool isRegistered;
    }
    // model a partner
    struct Partner {
        address partnerAddress;
        string name;
        bool isRegistered;
    }
    // model points transaction
    enum TransactionType { Earned, Redeemed }
    struct PointsTransaction {
        uint points;
        TransactionType transactionType;
        address memberAddress;
        address partnerAddress;
    }
    //members and partners on the network mapped with their address
    mapping(address => Member) public members;
    mapping(address => Partner) public partners;
    //public transactions and partners information
    Partner[] public partnersInfo;
    PointsTransaction[] public transactionsInfo;
    //register sender as member
    function registerMember (string memory _firstName, string memory
    _lastName, string memory _email) public {
        //check msg.sender in existing members
        require(!members[msg.sender].isRegistered, "Account already registered
as Member");
```

```

//check msg.sender in existing partners
require(!partners[msg.sender].isRegistered, "Account already
registered as Partner");
//add member account
members[msg.sender] = Member(msg.sender, _firstName, _lastName,
_email, 0, true);
}
//register sender as partner
function registerPartner (string memory _name) public {
//check msg.sender in existing members
require(!members[msg.sender].isRegistered, "Account already registered
as Member");
//check msg.sender in existing partners
require(!partners[msg.sender].isRegistered, "Account already
registered as Partner");
//add partner account
partners[msg.sender] = Partner(msg.sender, _name, true);
//add partners info to be shared with members
partnersInfo.push(Partner(msg.sender, _name, true));
}
//update member with points earned
function earnPoints (uint _points, address _partnerAddress ) public {
// only member can call
require(members[msg.sender].isRegistered, "Sender not registered as
Member");
// verify partner address
require(partners[_partnerAddress].isRegistered, "Partner address not
found");
// update member account
members[msg.sender].points = members[msg.sender].points + _points;
// add transaction
transactionsInfo.push(PointsTransaction({
    points: _points,
    transactionType: TransactionType.Earned,
    memberAddress: members[msg.sender].memberAddress,
    partnerAddress: _partnerAddress
}));
}
//update member with points used
function usePoints (uint _points, address _partnerAddress) public {
// only member can call
require(members[msg.sender].isRegistered, "Sender not registered as Member");
// verify partner address
require(partners[_partnerAddress].isRegistered, "Partner address not found");
// verify enough points for member
require(members[msg.sender].points >= _points, "Insufficient points");
// update member account
members[msg.sender].points = members[msg.sender].points - _points;
}

```

```

// add transaction
transactionsInfo.push(PointsTransaction({
    points: _points,
    transactionType: TransactionType.Redeemed,
    memberAddress: members[msg.sender].memberAddress,
    partnerAddress: _partnerAddress
}));
}

//get length of transactionsInfo array
function transactionsInfoLength() public view returns(uint256) {
    return transactionsInfo.length;
}

//get length of partnersInfo array
function partnersInfoLength() public view returns(uint256) {
    return partnersInfo.length;
}
}

```

Compiling the smart contract

Save the file and run the following

npx hardhat compile

```

→ smart-contract npx hardhat compile
Downloading compiler 0.8.24
Generating typings for: 1 artifacts in dir: typechain-types for target: ethers-v6
Successfully generated 6 typings!
Compiled 1 Solidity file successfully (evm target: paris).

```

Deploying the smart contract

We will be deploying the smart contract to the local hardhat network.

First we need to write the script that will deploy the smart contract open scripts/deploy.ts and replace its content with below code

Code:

```

import { ethers } from "hardhat";
async function main() {
    const loyaltyPoints = await
ethers.deployContract("LoyaltyPoints");
    await loyaltyPoints.waitForDeployment();
    console.log('LoyaltyPoints deployed to
${loyaltyPoints.target}`);
}
main().catch((error) => {
    console.error(error);
    process.exitCode = 1;
});

```

To deploy the contract we need to make sure the hardhat local network is running, open a new terminal in the same project and run the following code and keep it open
npx hardhat node

```

→ smart-contract npx hardhat node
Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/

Accounts
=====
WARNING: These accounts, and their private keys, are publicly known.
Any funds sent to them on Mainnet or any other live network WILL BE LOST.

Account #0: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266 (10000 ETH)
Private Key: 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cb65efcae784d7bf4f2ff80

Account #1: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8 (10000 ETH)
Private Key: 0x59c6995e998f97a5a0044966f0945389dc9e86dae88c7a8412f4603b6b78690d

Account #2: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC (10000 ETH)
Private Key: 0x5de4111afa1a4b94908f83103eb1f1706367c2e68ca870fc3fb9a804cdab365a

```

The output gives us the JSON RPC server information and some accounts that we can use

Go back to the previous terminal and deploy the contract

npx hardhat run scripts/deploy.ts --network localhost

```

→ smart-contract npx hardhat run scripts/deploy.ts --network localhost
LoyaltyPoints deployed to 0x5fdb2315678afecb367f032d93f642f64180aa3

```

The contract is successfully deployed. Save the address for later use

Making the web app

Clone the given repository

git clone <https://github.com/IBM/loyalty-points-evm-fabric.git>

```

→ prac7 git clone https://github.com/IBM/loyalty-points-evm-fabric.git
Cloning into 'loyalty-points-evm-fabric'...
remote: Enumerating objects: 1497, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 1497 (delta 0), reused 1 (delta 0), pack-reused 1494
Receiving objects: 100% (1497/1497), 4.66 MiB | 9.02 MiB/s, done.
Resolving deltas: 100% (336/336), done.

```

Navigate into the webapp directory and install the dependencies

cd loyalty-points-evm-fabric/web-app

npm i

```

→ prac7 cd loyalty-points-evm-fabric/web-app
→ web-app git:(master) npm i
npm WARN old lockfile
npm WARN old lockfile The package-lock.json file was c
npm WARN old lockfile so supplemental metadata must be
npm WARN old lockfile
npm WARN old lockfile This is a one-time fix-up, please
npm WARN old lockfile
npm WARN deprecated har-validator@5.1.0: this library
npm WARN deprecated uuid@3.3.2: Please upgrade to ver
m for details.
npm WARN deprecated request@2.88.0: request has been d
added 105 packages, and audited 106 packages in 12s

```

Editing the dapp.js config file

There are a series of edits in the dapp.js file

First lets update the contract address and provider. Contract address will be the address it was deployed for you and provider will the JSON RPC url we got when we started the hardhat node

```

var contractAddress = '0x5fdb2315678afecb367f032d93f642f64180aa3';
var provider = "http://127.0.0.1:8545";

```

Next we need to add contract ABI: for this open the smart contract project and navigate to artifacts/contracts/LoyaltyPoints.sol there will be a json file named LoyaltyPoints.json copy it into your web app project directory

```

cd artifacts/contracts/LoyaltyPoints.sol
cp LoyaltyPoints.json ../../../../../loyalty-points-evm-fabric/web
app
cd ../../../../../loyalty-points-evm-fabric/web-app
ls
→ smart-contract cd artifacts/contracts/LoyaltyPoints.sol
→ LoyaltyPoints.sol cp LoyaltyPoints.json ../../../../../loyalty-points-evm-fabric/web-app
→ LoyaltyPoints.sol cd ../../../../../loyalty-points-evm-fabric/web-app
→ web-app git:(master) ✘ ls
app.js dapp.js LoyaltyPoints.json node_modules package.json package-lock.json public validate.js

```

Changing the code for loyaltyABI variable. Remove loyaltyABI variable and
replace it with the below code

```

const fs = require("fs")
const loyaltyABI =
JSON.parse(fs.readFileSync("LoyaltyPoints.json")).abi

```

New code will look like this

```

var address = contractAddress;

console.log("Got address: " + address)

const fs = require("fs")
const loyaltyABI = JSON.parse(fs.readFileSync("LoyaltyPoints.json")).abi;

var LoyaltyContract = web3.eth.contract(loyaltyABI);
myContract = LoyaltyContract.at(address);
return myContract;

```

Running the app

```

npm start
→ web-app git:(master) ✘ npm start

> loyalty-points@0.0.1 start
> node app.js

app running on port: 8000

```

Open the browser and enter the url <http://localhost:8000>

Register as Member or Partner of the loyalty program

Members

Customers can register as Member to this program. Once registered they can make transactions to earn points and use points, and view all transactions.

[Register »](#)

Partners

Companies can register as Partner on the network. They can view all transactions and dashboard to view total points allocated and redeemed by members.

[Register »](#)

Practical No. 7

Aim - Car auction network: A Hello World example with Hyperledger Fabric Node SDK and IBM Block chain Starter Plan. Use Hyperledger Fabric to invoke chaincode while storing results and data in the starter plan.

Objective: To simulate a marketplace where assets (cars) are auctioned to the highest bidder.

Theory:

- State Machine Logic: An auction has distinct states: Open, Bidding, Closed. The smart contract must enforce that bids are only accepted when the state is Open.
- Invoking vs. Querying:
 1. Invoke: A transaction that changes the ledger (e.g., "Make a Bid"). This must be endorsed by peers and ordered.
 2. Query: A read-only operation (e.g., "Check current highest bid"). This is faster as it reads the World State without generating a new block.
- Data Visibility: In a car auction, you might use Private Data Collections if you want the bidder's identity to be kept secret from other bidders but known to the auctioneer.

Code:

Setup the blockchain network

Open a new terminal and make sure the network is down before starting

```
cd test-network
```

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

```
→ fabric-samples git:(v2.2.0) cd test-network
→ test-network git:(v2.2.0) ./network.sh down
```

Remove any existing/running docker containers

```
docker rm -f $(docker ps -aq)
```

```
docker rmi -f $(docker images | grep fabcar | awk '{print $3}')
```

```
→ test-network git:(v2.2.0) docker rm -f $(docker ps -aq)
docker rmi -f $(docker images | grep fabcar | awk '{print $3}')
```

```
"docker rm" requires at least 1 argument.
See 'docker rm --help'.
```

```
Usage: docker rm [OPTIONS] CONTAINER [CONTAINER...]
```

```
Remove one or more containers
```

```
"docker rmi" requires at least 1 argument.
```

```
See 'docker rmi --help'.
```

```
Usage: docker rmi [OPTIONS] IMAGE [IMAGE...]
```

```
Remove one or more images
```

```
→ test-network git:(v2.2.0) S
```

Starting the network

For this Network we will be using Javascript

```
cd fabcar
```

```
./startFabric.sh javascript
```

```
→ fabric-samples git:(v2.2.0) cd fabcar
→ fabcar git:(v2.2.0) ./startFabric.sh javascript
~/web3/fabric-samples/test-network ~/web3/fabric-samples/fabcar
Stopping network
Removing network fabric_test
WARNING: Network fabric_test not found.
```

You need to be in the javascript directory

Run the following command to install the Fabric dependencies for the applications. It will take about a minute to complete:

```
cd javascript
```

```
npm i
```

```
→ fabcar git:(v2.2.0) cd javascript
→ javascript git:(v2.2.0) npm i
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher.
th.random() in certain circumstances, which is known to be problematic.
ath-random for details.
npm WARN deprecated mkdirp@0.5.1: Legacy versions of mkdirp are no longer
to mkdirp 1.x. (Note that the API surface has changed to use Promises)
npm WARN deprecated sinon@7.5.0: 16.1.1

added 358 packages, and audited 359 packages in 29s
```

Once npm i completes, everything is in place to run the application. For this tutorial, you'll primarily be using the application JavaScript files in the fabcar/javascript directory. Let's take a look at what's inside:

```
ls
```

```
→ javascript git:(v2.2.0) ls
enrollAdmin.js  node_modules  package-lock.json  registerUser.js
invoke.js        package.json   query.js         wallet
→ javascript git:(v2.2.0)
```

Enrolling the admin user

We will subsequently register and enroll a new application user which will be used by our application to interact with the blockchain.

```
node enrollAdmin.js
```

```
→ javascript git:(v2.2.0) node enrollAdmin.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Successfully enrolled admin user "admin" and imported it into the wallet
```

Register and Enroll

Now that we have the administrator's credentials in a wallet, we can enroll a new user user1 which will be used to query and update the ledger:

```
node registerUser.js
```

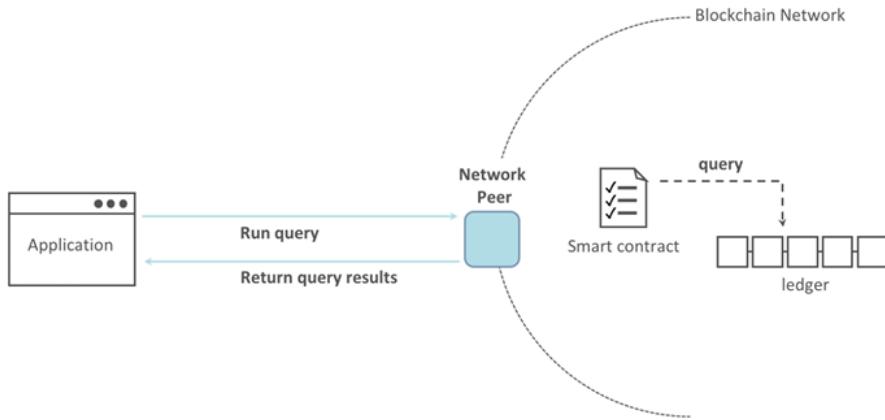
```
→ javascript git:(v2.2.0) node registerUser.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Successfully registered and enrolled admin user "appUser" and imported it into the wallet
→ javascript git:(v2.2.0)
```

Similar to the admin enrollment, this program uses a CSR to enroll user1 and store its credentials alongside those of admin in the wallet. We now have identities for two separate users admin and user1 and these are used by our application.

Querying the ledger

Each peer in a blockchain network hosts a copy of the ledger, and an application program can query the ledger by invoking a smart contract which queries the most recent value of the ledger and returns it to the application.

Here is a simplified representation of how a query works:



First, let's run our query.js program to return a listing of all the cars on the ledger. This program uses our second identity – user1 – to access the ledger:

node query.js

```
→ javascript git:(v2.2.0) node query.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: [{"Key": "CAR0", "Record": {"color": "blue", "docType": "car", "make": "Toyota", "model": "Prius", "owner": "Tomoko"}}, {"Key": "CAR1", "Record": {"color": "red", "docType": "car", "make": "Ford", "model": "Mustang", "owner": "Brad"}}, {"Key": "CAR2", "Record": {"color": "green", "docType": "car", "make": "Hyundai", "model": "Tucson", "owner": "Jin Soo"}}, {"Key": "CAR3", "Record": {"color": "yellow", "docType": "car", "make": "Volkswagen", "model": "Passat", "owner": "Max"}}, {"Key": "CAR4", "Record": {"color": "black", "docType": "car", "make": "Tesla", "model": "S", "owner": "Adriana"}}, {"Key": "CAR5", "Record": {"color": "purple", "docType": "car", "make": "Peugeot", "model": "205", "owner": "Michel"}}, {"Key": "CAR6", "Record": {"color": "white", "docType": "car", "make": "Chery", "model": "S2L", "owner": "Aarav"}}, {"Key": "CAR7", "Record": {"color": "violet", "docType": "car", "make": "Fiat", "model": "Punto", "owner": "Parti"}}, {"Key": "CAR8", "Record": {"color": "indigo", "docType": "car", "make": "Tata", "model": "Nano", "owner": "Valerita"}}, {"Key": "CAR9", "Record": {"color": "brown", "docType": "car", "make": "Holden", "model": "Barina", "owner": "Shotaro"}}]
→ javascript git:(v2.2.0)
```

Changing query.js code

change the evaluateTransaction request to query CAR4. The query program should now look like this:

```
const result = await contract.evaluateTransaction('queryCar', 'CAR4');
```

Save the program and run the query program again:

node query.js

```
→ javascript git:(v2.2.0) X node query.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: {"color": "black", "docType": "car", "make": "Tesla", "model": "S", "owner": "Adriana"}
→ javascript git:(v2.2.0) X
```

Updating the ledger

Our first update to the ledger will create a new car. We have a separate program called invoke.js that we will use to make updates to the ledger. Just as with queries, use an editor to open the program and navigate to the code block where we construct our transaction and submit it to the network:

```
await contract.submitTransaction('createCar', 'CAR12', 'Honda', 'Accord', 'Black', 'Tom');
console.log('Transaction has been submitted');
```

Run the program

node invoke.js

```
→ javascript git:(v2.2.0) X node invoke.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Transaction has been submitted
→ javascript git:(v2.2.0) X
```

Changing the query.js to fetch the new transaction

```
const result = await contract.evaluateTransaction('queryCar', 'CAR12');
console.log('Transaction has been evaluated, result is: ${result.toString()}');
```

Save and run

node query.js

```
→ javascript git:(v2.2.0) ✘ node query.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: {"color":"Black","docType":"car","make":"Honda","model":"Accord","owner":"Tom"}
→ javascript git:(v2.2.0) ✘
```

Changing the ownership of car by making changes in invoke.js To do this, go back to invoke.js and change the smart contract transaction from createCar to changeCarOwner with a corresponding change in input arguments

```
await contract.submitTransaction('changeCarOwner', 'CAR12', 'Dave');
```

```
// await contract.submitTransaction('createCar', 'CAR12', 'Honda', 'Accord', 'Black', 'Tom');
await contract.submitTransaction('changeCarOwner', 'CAR12', 'Dave');
console.log('Transaction has been submitted');
```

Save and run

node invoke.js

```
→ javascript git:(v2.2.0) ✘ node invoke.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Transaction has been submitted
→ javascript git:(v2.2.0) ✘
```

Verifying if the Owner for Car12 is updated

node query.js

```
→ javascript git:(v2.2.0) ✘ node query.js
Wallet path: /home/hackerman/web3/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: {"color":"Black","docType":"car","make":"Honda","model":"Accord","owner":"Dave"}
→ javascript git:(v2.2.0) ✘
```

The ownership of CAR12 has been changed from Tom to Dave.

Practical No. 8

Aim - Develop a voting application using Hyperledger and Ethereum.

Objective: To explore electronic voting (e-voting) where security, anonymity, and immutability are paramount.

Theory:

- The E-Voting Challenge: A voting system must ensure:
 1. Eligibility: Only allowed users can vote.
 2. Uniqueness: Users can only vote once.
 3. Privacy: No one can link a vote to a specific voter.
 4. Integrity: No one can change the vote count.
- Hyperledger (Permissioned) Approach: Good for corporate or internal voting. The MSP handles identity (voter registration), but the vote itself must be anonymized (often using zero-knowledge proofs or separated databases).
- Ethereum (Public) Approach: Good for transparent public voting.
 1. Smart Contract: Stores the candidates and vote counts.
 2. Gas: The cost to execute the vote transaction.
 3. Metamask: Used as the wallet/identity for the voter to sign their ballot.

Code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Voting {
    // Declare variables to store the total votes for each team
    uint256 public votesForTeamA;
    uint256 public votesForTeamB;
    uint256 public votesForTeamC;
    address public owner;
    mapping(address => bool) authorizedVoters;

    constructor() {
        owner = msg.sender;
    }

    modifier onlyAuthorizedVoter() {
        require(
            authorizedVoters[msg.sender] == true,
            "You are not authorized to vote"
        );
    }
}
```

```

        _;
    }

modifier onlyOwner() {
    require(msg.sender == owner, "Only the owner can call this
function");
    _;
}

// Function to allow a voter to cast their vote for a team
function vote(uint256 _team) public onlyAuthorizedVoter {
    // Check the value of _team and increment the corresponding team's
vote count
    if (_team == 1) {
        votesForTeamA += 1;
    } else if (_team == 2) {
        votesForTeamB += 1;
    } else if (_team == 3) {
        votesForTeamC += 1;
    }
}

// Function to add a voter to the list of authorized voters
function addVoter(address _voter) public onlyOwner {
    require(msg.sender == owner, "Only the owner can add a voter");
    authorizedVoters[_voter] = true;
}

// Function to declare results of the vote
function getWinner() public view returns (string memory result) {
    // Check the vote counts and return the winner
    if (votesForTeamA > votesForTeamB && votesForTeamA > votesForTeamC)
    {
        result = "Team A is the winner!";
    } else if (
        votesForTeamB > votesForTeamA && votesForTeamB > votesForTeamC
    ) {
        result = "Team B is the winner!";
    } else if (
        votesForTeamC > votesForTeamA && votesForTeamC > votesForTeamB
    ) {
        result = "Team C is the winner!";
    } else {
        result = "There is a tie!";
    }
}

function getTotalVotes() public view returns (uint256) {

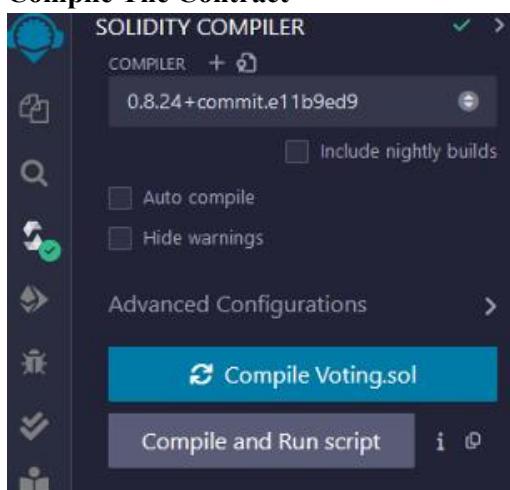
```

```

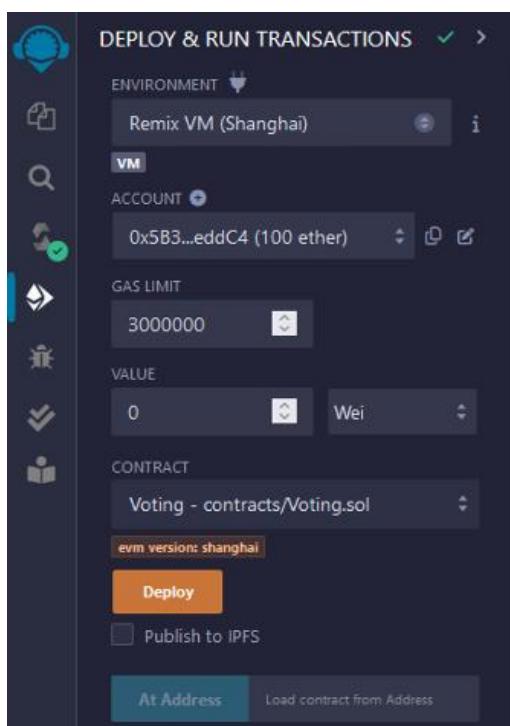
        return votesForTeamA + votesForTeamB + votesForTeamC;
    }
}

```

Compile The Contract

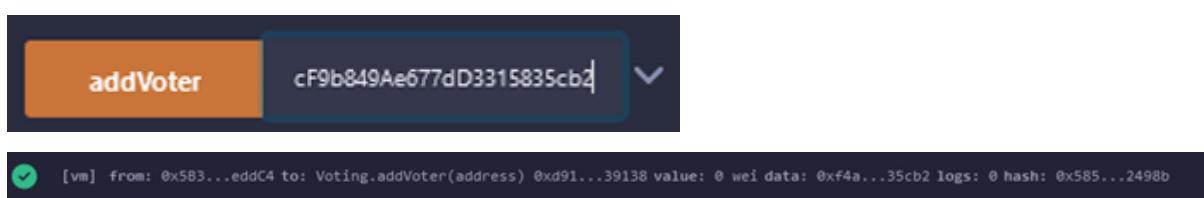


Deploy The Contract

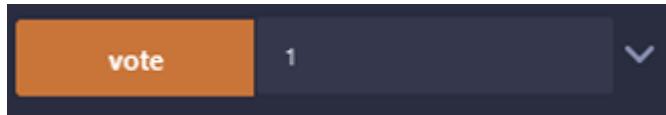


Adding voters

(Note: you can get wallet address from above dropdown; change it to another and copy it)



Voting



```
[✓] [vm] from: 0xAB8...35cb2 to: Voting.vote(uint256) 0xd91...39138 value: 0 wei data: 0x012...00001 logs: 0 hash: 0xd19...53ff6
```

Checking the winner

