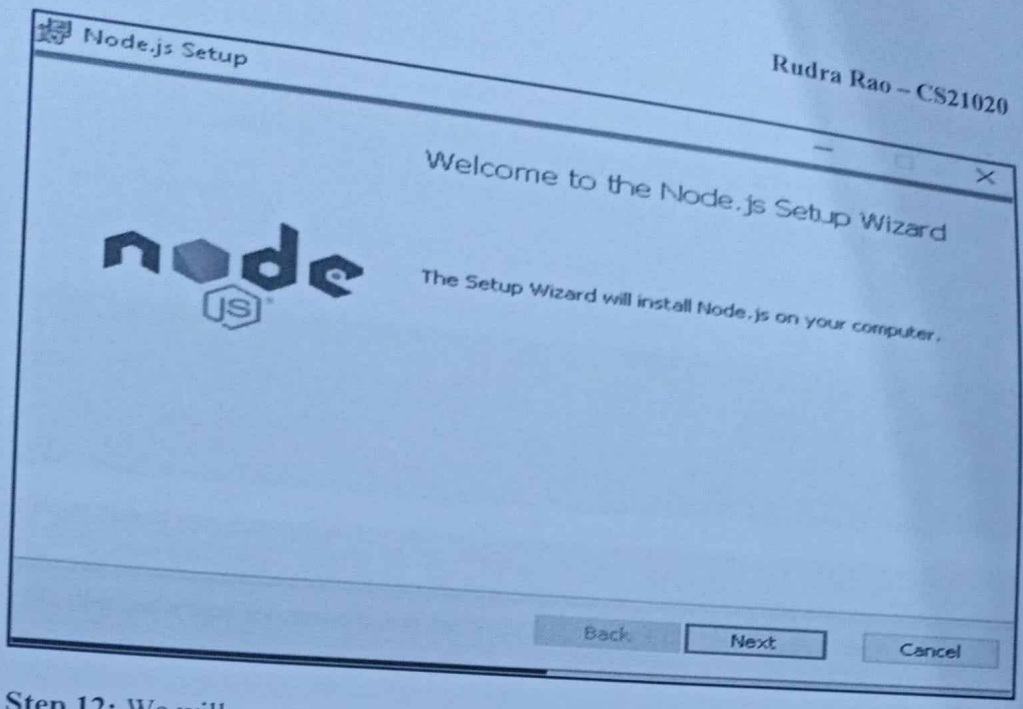
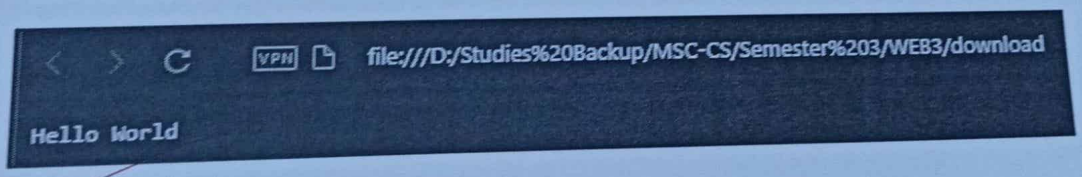


Step 12: We will execute some programs using node.

1) First Program:

```
var http = require("http");
http
 .createServer(function (req, res) {
  res.writeHead(200, { "Content-Type": "txt/html" });
  res.end("Hello World");
 })
 .listen(8081);
```

```
PS D:\Studies Backup\MSC-CS\Semester 3\WEB3\Javascript Code> node firstCode.js
```



2) Second Program:

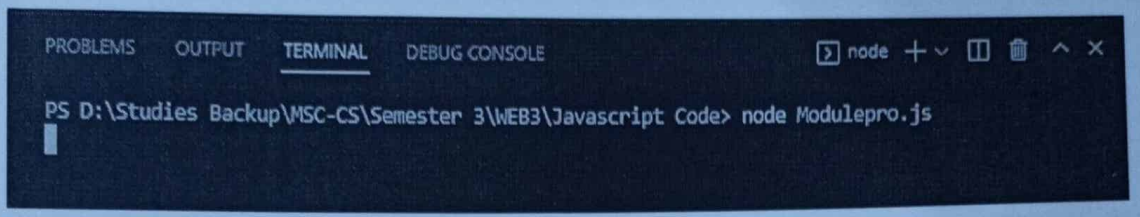
myfirstmodule.js

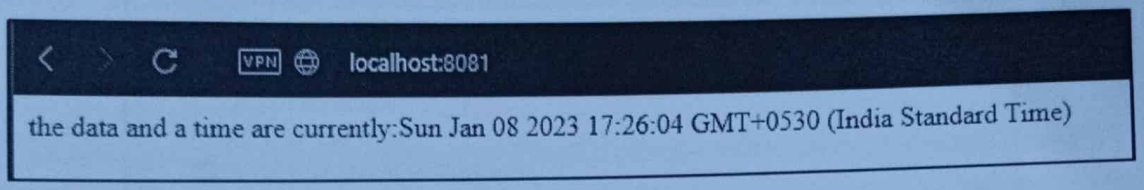
```
exports.myDateTime = function () {
  return Date();
};
```

Modulepro.js

```
var http = require("http");
var dt = require("./myfirstmodule");
```

```
http
  .createServer(function (reg, res) {
    res.writeHead(200, { "Content-Type": "text/html" });
    res.write("the data and a time are currently:" + dt.myDateTime());
    res.end();
  })
  .listen(8081);
```



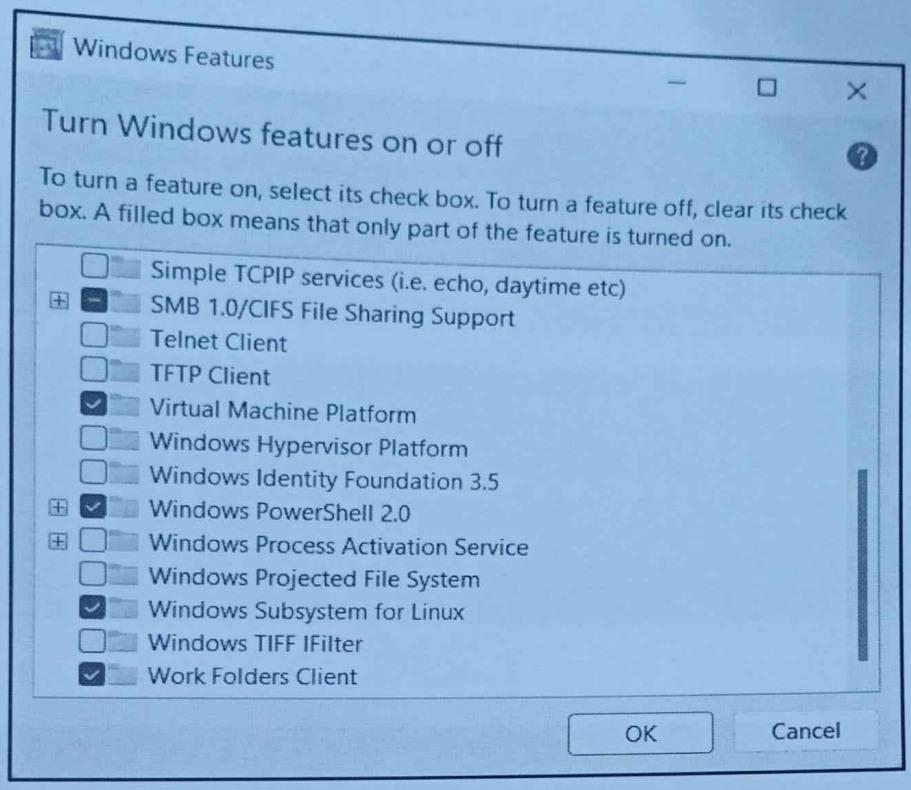


10/10

Practical 2

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

Step 1: Check if WSL is enabled.



Step 2: Check if Docker, Curl, Python, Node and install Golang (version).

```
rudra@DESKTOP-0BATOB7:~$ curl --version
curl 7.81.0 (x86_64-pc-linux-gnu) libcurl/7.81.0 OpenSSL/3.0.2 zlib/1.2.11 brotli/1.0.9 zstd/1.4.8 libidn2/2.3
.2 libpsl/0.21.0 (+libidn2/2.3.2) libssh/0.9.6/openssl/zlib nghttp2/1.43.0 librtmp/2.3 OpenLDAP/2.5.13
Release-Date: 2022-01-05
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ldaps mqtt pop3 pop3s rtmp rtsp scp sf
tp smb smbs smtp smtps telnet tftp
Features: alt-svc AsynchDNS brotli GSS-API HSTS HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM NTLM_W
B PSL SPNEGO SSL TLS-SRP UnixSockets zstd
```

```
rudra@DESKTOP-0BATOB7:~$ docker --version
Docker version 20.10.21, build baeda1f
rudra@DESKTOP-0BATOB7:~$
```

```
rudra@DESKTOP-0BATOB7:~$ curl --version
curl 7.81.0 (x86_64-pc-linux-gnu) libcurl/7.81.0 OpenSSL/3.0.2 zlib/1.2.11 brotli/1.0.9 zstd/1.4.8 libidn2/2.3
.2 libpsl/0.21.0 (+libidn2/2.3.2) libssh/0.9.6/openssl/zlib nghttp2/1.43.0 librtmp/2.3 OpenLDAP/2.5.13
Release-Date: 2022-01-05
Protocols: dict file ftp ftps gopher gophers http https imap imaps ldap ldaps mqtt pop3 pop3s rtmp rtsp scp sf
tp smb smbs smtp smtps telnet tftp
Features: alt-svc AsynchDNS brotli GSS-API HSTS HTTP2 HTTPS-proxy IDN IPv6 Kerberos Largefile libz NTLM NTLM_W
B PSL SPNEGO SSL TLS-SRP UnixSockets zstd
rudra@DESKTOP-0BATOB7:~$ docker --version
Docker version 20.10.21, build baeda1f
rudra@DESKTOP-0BATOB7:~$ docker-compose --version
Docker Compose version v2.13.0
rudra@DESKTOP-0BATOB7:~$
```

```
rudra@DESKTOP-0BATOB7:~$ python3 --version
Python 3.10.6
rudra@DESKTOP-0BATOB7:~$ go --version
Command 'go' not found, but can be installed with:
sudo apt install golang-go  # version 2:1.18~0ubuntu2, or
sudo apt install gccgo-go   # version 2:1.18~0ubuntu2
rudra@DESKTOP-0BATOB7:~$ sudo apt install golang-go
[sudo] password for rudra:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bzip2 cpp cpp-11 fontconfig-config fonts-dejavu-core g++ g++-11
  golang-1.18-src golang-src libasan6 libatomic1 libc-dev-bin libc
  libdeflate0 libdpkg-perl libfile-fcntllock-perl libfontconfig1
```

```
rudra@DESKTOP-0BATOB7:~$ npm -v
8.5.1
rudra@DESKTOP-0BATOB7:~$ node -v
v12.22.9
rudra@DESKTOP-0BATOB7:~$
```

Step 3: Then we will configure git.

```
rudra@DESKTOP-0BATOB7:~$ npm -v
8.5.1
rudra@DESKTOP-0BATOB7:~$ node -v
v12.22.9
rudra@DESKTOP-0BATOB7:~$ git config --global core.autocrlf false
rudra@DESKTOP-0BATOB7:~$ git config --global core.longpaths true
rudra@DESKTOP-0BATOB7:~$
```

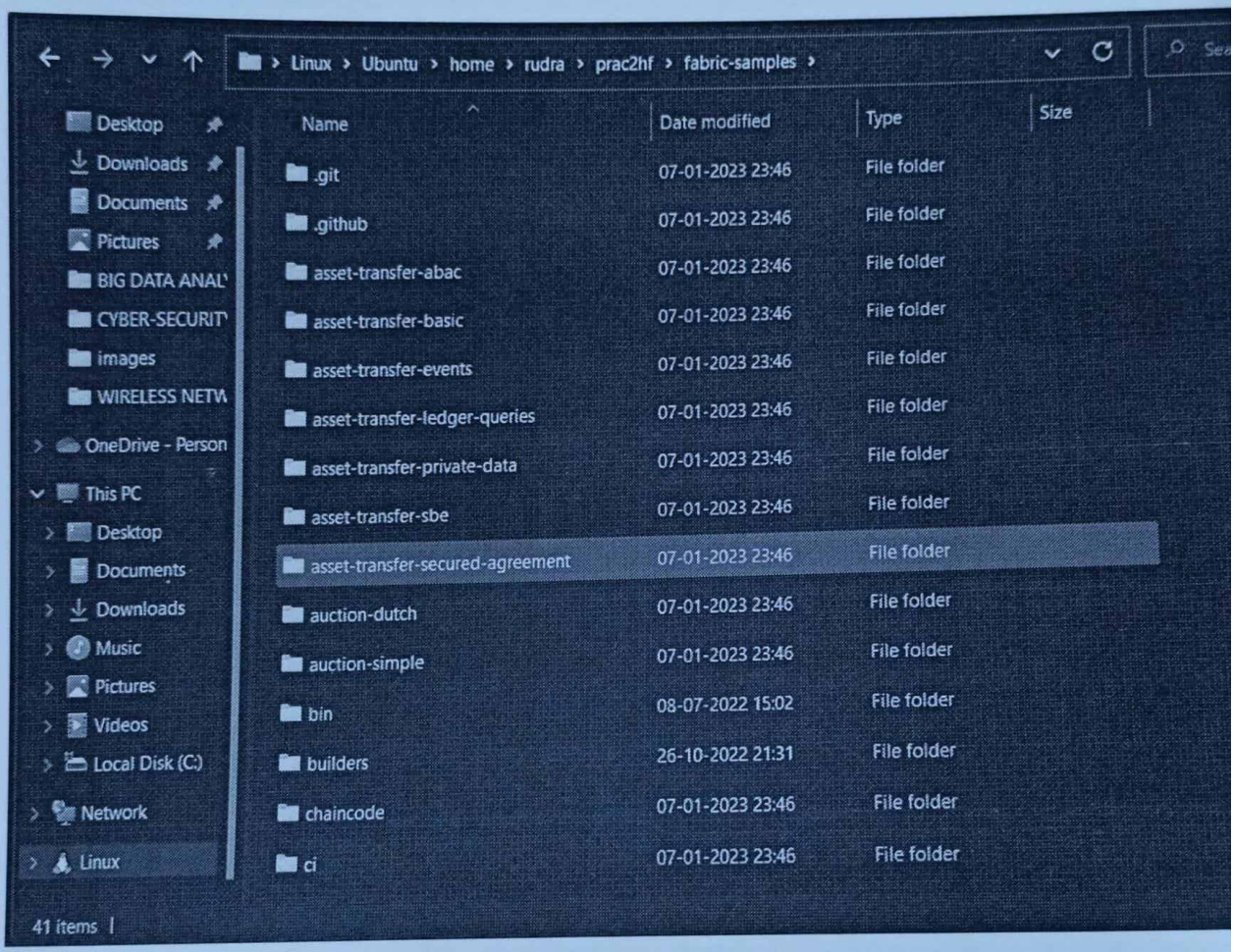
Step 4: Then we will create a folder (in my case prac2hf) and navigate into it. Then download the download the fabric-samples repo.

```
rudra@DESKTOP-0BATOB7:~/prac2hf$ curl -sSL http://bit.ly/2ysbOFE | bash -s

Clone hyperledger/fabric-samples repo

===> Cloning hyperledger/fabric-samples repo
Cloning into 'fabric-samples'...
remote: Enumerating objects: 11717, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 11717 (delta 0), reused 0 (delta 0), pack-reused 11712
Receiving objects: 100% (11717/11717), 22.17 MiB | 5.61 MiB/s, done.
Resolving deltas: 100% (6281/6281), done.
fabric-samples v2.4.7 does not exist, defaulting to main. fabric-samples main branch is intend
recent versions of fabric.

Pull Hyperledger Fabric binaries

===> Downloading version 2.4.7 platform specific fabric binaries
===> Downloading:  https://github.com/hyperledger/fabric/releases/download/v2.4.7/hyperledger-
64-2.4.7.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
 25 85.9M   25 22.1M    0     0  4366k      0  0:00:20  0:00:05  0:00:15 5557k
```

```
2408cc74d12b: Pull complete
979557f40538: Pull complete
b3bcd28c0311: Pull complete
Digest: sha256:f93cd9f32702c3a6b9cb305d75bed5edd884cae0674374fd7c26467bf6a0ed9b
Status: Downloaded newer image for hyperledger/fabric-ca:1.5.5
docker.io/hyperledger/fabric-ca:1.5.5
===> List out hyperledger docker images
hyperledger/fabric-tools     2.4       545af418d284   2 months ago   489MB
hyperledger/fabric-tools     2.4.7     545af418d284   2 months ago   489MB
hyperledger/fabric-tools     latest    545af418d284   2 months ago   489MB
hyperledger/fabric-peer      2.4       d77dd7cfbcd7   2 months ago   64.2MB
hyperledger/fabric-peer      2.4.7     d77dd7cfbcd7   2 months ago   64.2MB
hyperledger/fabric-peer      latest    d77dd7cfbcd7   2 months ago   64.2MB
hyperledger/fabric-orderer   2.4       77c489caa81b   2 months ago   36.7MB
hyperledger/fabric-orderer   2.4.7     77c489caa81b   2 months ago   36.7MB
hyperledger/fabric-orderer   latest    77c489caa81b   2 months ago   36.7MB
hyperledger/fabric-ccenv     2.4       4eb7ee7f4af5   2 months ago   520MB
hyperledger/fabric-ccenv     2.4.7     4eb7ee7f4af5   2 months ago   520MB
hyperledger/fabric-ccenv     latest    4eb7ee7f4af5   2 months ago   520MB
hyperledger/fabric-baseos    2.4       b20d2dde6941   2 months ago   6.82MB
hyperledger/fabric-baseos    2.4.7     b20d2dde6941   2 months ago   6.82MB
hyperledger/fabric-baseos    latest    b20d2dde6941   2 months ago   6.82MB
hyperledger/fabric-ca        1.5       93f19fa873cb   6 months ago   76.5MB
hyperledger/fabric-ca        1.5.5     93f19fa873cb   6 months ago   76.5MB
hyperledger/fabric-ca        latest    93f19fa873cb   6 months ago   76.5MB
rudra@DESKTOP-0BATOB7:~/prac2hf$
```

Step 5: We will check if the Repository has been successfully cloned.

```
rudra@DESKTOP-0BAT0B7:~/prac2hf$ ls
fabric-samples
rudra@DESKTOP-0BAT0B7:~/prac2hf$ cd fabric-samples/
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples$ ls
CHANGELOG.md                    asset-transfer-sbe                  high-throughput
CODEOWNERS                      asset-transfer-secured-agreement    interest_rate_swaps
CODE_OF_CONDUCT.md              auction-dutch                       off_chain_data
CONTRIBUTING.md                 auction-simple                      scripts
LICENSE                         bin                                 test-application
MAINTAINERS.md                  builders                            test-network
README.md                       chaincode                           test-network-k8s
SECURITY.md                     ci                                  test-network-nano-bash
asset-transfer-abac             commercial-paper                    token-erc-1155
asset-transfer-basic            config                              token-erc-20
asset-transfer-events           fabcar                              token-erc-721
asset-transfer-ledger-queries   full-stack-asset-transfer-guide     token-utxo
asset-transfer-private-data     hardware-security-module
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples$
```



Rudra Rao – CS21020

Practical 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.

Step 1: Installing Geth (Go Ethereum)
Geth is a CLI with some resources to connect you to Ethereum network. It will be used to start our private network in local environment.

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

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples$ sudo add-apt-repository -y ppa:ethereum/ethereum
[sudo] password for rudra:
Repository: 'deb https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu/ jammy main'
More info: https://launchpad.net/~ethereum/+archive/ubuntu/ethereum
Adding repository.
Adding deb entry to /etc/apt/sources.list.d/ethereum-ubuntu-ethereum-jammy.list
Adding disabled deb-src entry to /etc/apt/sources.list.d/ethereum-ubuntu-ethereum-jammy.list
Adding key to /etc/apt/trusted.gpg.d/ethereum-ubuntu-ethereum.gpg with fingerprint 2A518C819BE37D2C2031944D1C5
2189C923F6CA9
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy InRelease [17.5 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:6 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main amd64 Packages [2812 B]
Get:7 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main Translation-en [828 B]
Fetched 345 kB in 2s (155 kB/s)
Reading package lists... Done
```

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples$ sudo apt-get update
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:4 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy InRelease
Get:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Fetched 324 kB in 1s (245 kB/s)
Reading package lists... Done
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples$
```

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples$ sudo apt-get install ethereum
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  abigen bootnode clef evm geth puppeth rlpdump
The following NEW packages will be installed:
  abigen bootnode clef ethereum evm geth puppeth rlpdump
0 upgraded, 8 newly installed, 0 to remove and 9 not upgraded.
Need to get 34.1 MB of archives.
After this operation, 108 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main amd64 abigen amd64 1.10.26+build282
12+jammy [3825 kB]
Get:2 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main amd64 bootnode amd64 1.10.26+build2
8212+jammy [3452 kB]
Get:3 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main amd64 clef amd64 1.10.26+build28212
+jammy [8994 kB]
Get:4 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main amd64 evm amd64 1.10.26+build28212+
jammy [3951 kB]
Get:5 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main amd64 geth amd64 1.10.26+build28212
+jammy [9428 kB]
Get:6 https://ppa.launchpadcontent.net/ethereum/ethereum/ubuntu jammy/main amd64 puppeth amd64 1.10.26+build28
212+jammy [3753 kB]
```

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples$ geth -h
NAME:
   geth - the go-ethereum command line interface

USAGE:
   geth [global options] command [command options] [arguments...]

COMMANDS:
   account                  Manage accounts
   attach                   Start an interactive JavaScript environment (connect to node)
   console                  Start an interactive JavaScript environment
   db                       Low level database operations
   dump                     Dump a specific block from storage
   dumpconfig               Show configuration values
   dumpgenesis              Dumps genesis block JSON configuration to stdout
   export                   Export blockchain into file
   export-preimages         Export the preimage database into an RLP stream
   import                   Import a blockchain file
   import-preimages         Import the preimage database from an RLP stream
   init                     Bootstrap and initialize a new genesis block
   js                       (DEPRECATED) Execute the specified JavaScript files
   license                  Display license information
```

Step 2: The Genesis Block.

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples$ cd ..
rudra@DESKTOP-0BATOB7:~/prac2hf$ cd ..
rudra@DESKTOP-0BATOB7:~$ mkdir my-blockchain
rudra@DESKTOP-0BATOB7:~$ cd my-blockchain
rudra@DESKTOP-0BATOB7:~/my-blockchain$ touch genesis.json
rudra@DESKTOP-0BATOB7:~/my-blockchain$
```

```
GNU nano 6.2

{
  "config": {
    "chainId": 1234,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "ethash": {}
  },
  "difficulty": "4",
  "gasLimit": "8000000",
  "alloc": {}
}
```

Step 3: Start Database (The Bootnode)

```
rudra@DESKTOP-0BATOB7:~/my-blockchain$ geth init --datadir node1 genesis.json
INFO [01-08|00:12:21.208] Maximum peer count                       ETH=50 LES=0 total=50
INFO [01-08|00:12:21.210] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such fi
le or directory"
INFO [01-08|00:12:21.214] Set global gas cap                       cap=50,000,000
INFO [01-08|00:12:21.217] Allocated cache and file handles         database=/home/rudra/my-blockchain/node1/ge
th/chaindata cache=16.00MiB handles=16
INFO [01-08|00:12:21.294] Opened ancient database                  database=/home/rudra/my-blockchain/node1/ge
th/chaindata/ancient/chain readonly=false
INFO [01-08|00:12:21.294] Writing custom genesis block
INFO [01-08|00:12:21.295] Persisted trie from memory database      nodes=0 size=0.00B time="7.354µs" gcnodes=0
gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [01-08|00:12:21.297] Successfully wrote genesis state         database=chaindata hash=ac4d9a..2ca582
INFO [01-08|00:12:21.297] Allocated cache and file handles         database=/home/rudra/my-blockchain/node1/ge
th/lightchaindata cache=16.00MiB handles=16
INFO [01-08|00:12:21.372] Opened ancient database                  database=/home/rudra/my-blockchain/node1/ge
th/lightchaindata/ancient/chain readonly=false
INFO [01-08|00:12:21.372] Writing custom genesis block
INFO [01-08|00:12:21.374] Persisted trie from memory database      nodes=0 size=0.00B time="9.117µs" gcnodes=0
gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [01-08|00:12:21.378] Successfully wrote genesis state         database=lightchaindata hash=ac4d9a..2ca582
rudra@DESKTOP-0BATOB7:~/my-blockchain$
```

```
rudra@DESKTOP-0BATOB7:~/my-blockchain$ ls
genesis.json  node1
rudra@DESKTOP-0BATOB7:~/my-blockchain$ cd node1
rudra@DESKTOP-0BATOB7:~/my-blockchain/node1$ ls
geth  keystore
rudra@DESKTOP-0BATOB7:~/my-blockchain/node1$ cd geth
rudra@DESKTOP-0BATOB7:~/my-blockchain/node1/geth$ ls
LOCK  chaindata  lightchaindata  nodekey
rudra@DESKTOP-0BATOB7:~/my-blockchain/node1/geth$ cd chaindata
rudra@DESKTOP-0BATOB7:~/my-blockchain/node1/geth/chaindata$ ls
000001.log  CURRENT  LOCK  LOG  MANIFEST-000000  ancient
rudra@DESKTOP-0BATOB7:~/my-blockchain/node1/geth/chaindata$
```

Step 4: Start Node

```
rudra@DESKTOP-0BATOB7:~/my-blockchain/node1/keystore$ geth --datadir node1 --networkid 1234 --http --allow-ins
ecure-unlock --nodiscover
INFO [01-08|00:16:40.231] Maximum peer count                       ETH=50 LES=0 total=50
INFO [01-08|00:16:40.233] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such fi
le or directory"
INFO [01-08|00:16:40.238] Set global gas cap                       cap=50,000,000
INFO [01-08|00:16:40.240] Allocated trie memory caches             clean=154.00MiB dirty=256.00MiB
INFO [01-08|00:16:40.240] Allocated cache and file handles         database=/home/rudra/my-blockchain/node1/ke
ystore/node1/geth/chaindata cache=512.00MiB handles=524,288
INFO [01-08|00:16:40.313] Opened ancient database                  database=/home/rudra/my-blockchain/node1/ke
ystore/node1/geth/chaindata/ancient/chain readonly=false
INFO [01-08|00:16:40.313] Writing default main-net genesis block
INFO [01-08|00:16:40.938] Persisted trie from memory database      nodes=12356 size=1.78MiB time=119.977526ms
gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [01-08|00:16:40.989]
```

```
rudra@DESKTOP-0BATOB7:~/my-blockchain$ geth attach node1/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.26-stable-e5eb32ac/linux-amd64/go1.18.5
at block: 0 (Thu Jan 01 1970 05:30:00 GMT+0530 (IST))
 datadir: /home/rudra/my-blockchain/node1
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0
web3:1.0

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

To see node information's, execute the following command in this JavaScript terminal opened:

```
To exit, press ctrl-d or type exit
> admin.nodeInfo
{
  enode: "enode://ce1a89ab411b422131f3c8db7dbd98f34a835da8dae1f6b5c3db86dadbe7bb984cb83d2cd5c1f62fe078432c48db
585eb93513e5966aef5668d633ea7ff7c11a@127.0.0.1:30303?discport=0",
  enr: "enr:-Jy4QLDfHK323MtxaVJnTydGtf7yqBkGn3UpiQ8shq6mSOOfHdCoAgoJNVMbtVAfhcgzO30t-1cAD-Fwq9AZwm...hBWGAVWNlg
CHg2V0aMfGhL3NduuAgmlkgnY0gmlwhHBAAAGJc2Vjcdl1NmsxoQLOGomrQHtCIfHzwNt0vZjz5gNdqNrh9rXDSuBe28e7mIRzbeFvm1N9Y3cC
dl8",
  id: "1ed82e522c2c07cb17c649657e1ddfbce3963ece81690feb02f861a9b164d4f0",
  ip: "127.0.0.1",
  listenAddr: "[::]:30303",
  name: "Geth/v1.10.26-stable-e5eb32ac/linux-amd64/go1.18.5",
  ports: {
    discovery: 30303,
    listener: 30303
  },
  protocols: {
    eth: {
      config: {
        byzantiumBlock: 0,
        chainId: 1234,
        constantinopleBlock: 0,
        eip150Block: 0,
        eip150Hash: "0x0000000000000000000000000000000000000000000000000000000000000000",
        eip155Block: 0,
        eip158Block: 0,
```

Step 5: Adding member peers

```
rudra@DESKTOP-8BATOB7:~$ cd my-blockchain/
rudra@DESKTOP-8BATOB7:~/my-blockchain$ geth init --datadir node2 genesis.json
INFO [01-08|00:28:24.582] Maximum peer count                       ETH=50 LES=0 total=50
INFO [01-08|00:28:24.585] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such fi
le or directory"
INFO [01-08|00:28:24.589] Set global gas cap                       cap=50,000,000
INFO [01-08|00:28:24.596] Allocated cache and file handles         database=/home/rudra/my-blockchain/node2/ge
th/chaindata cache=16.00MiB handles=16
INFO [01-08|00:28:24.667] Opened ancient database                  database=/home/rudra/my-blockchain/node2/ge
th/chaindata/ancient/chain readonly=false
INFO [01-08|00:28:24.667] Writing custom genesis block
INFO [01-08|00:28:24.668] Persisted trie from memory database      nodes=0 size=0.00B time="65.244µs" gcnodes=
0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [01-08|00:28:24.670] Successfully wrote genesis state         database=chaindata hash=ac4d9a..2ca582
INFO [01-08|00:28:24.670] Allocated cache and file handles         database=/home/rudra/my-blockchain/node2/ge
th/lightchaindata cache=16.00MiB handles=16
INFO [01-08|00:28:24.727] Opened ancient database                  database=/home/rudra/my-blockchain/node2/ge
th/lightchaindata/ancient/chain readonly=false
INFO [01-08|00:28:24.727] Writing custom genesis block
INFO [01-08|00:28:24.730] Persisted trie from memory database      nodes=0 size=0.00B time="15.119µs" gcnodes=
0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [01-08|00:28:24.732] Successfully wrote genesis state         database=lightchaindata hash=ac4d9a..2ca582
rudra@DESKTOP-8BATOB7:~/my-blockchain$
```

```
rudra@DESKTOP-8BATOB7:~/my-blockchain$ geth --datadir node2 --networkid 1234 --port 30304 --authrpc.port 8552
INFO [01-08|00:30:22.517] Maximum peer count                       ETH=50 LES=0 total=50
INFO [01-08|00:30:22.518] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such fi
le or directory"
INFO [01-08|00:30:22.521] Set global gas cap                       cap=50,000,000
INFO [01-08|00:30:22.523] Allocated trie memory caches             clean=154.00MiB dirty=256.00MiB
INFO [01-08|00:30:22.524] Allocated cache and file handles         database=/home/rudra/my-blockchain/node2/ge
th/chaindata cache=512.00MiB handles=524,288
INFO [01-08|00:30:23.885] Opened ancient database                  database=/home/rudra/my-blockchain/node2/ge
th/chaindata/ancient/chain readonly=false
INFO [01-08|00:30:23.886]
INFO [01-08|00:30:23.886] ------------------------------------------------------------------------------
------------------------------------------------------------
```

```
rudra@DESKTOP-0BATOB7:~$ cd my-blockchain/
rudra@DESKTOP-0BATOB7:~/my-blockchain$ geth attach node2/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.26-stable-e5eb32ac/linux-amd64/go1.18.5
at block: 0 (Thu Jan 01 1970 05:30:00 GMT+0530 (IST))
datadir: /home/rudra/my-blockchain/node2
modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0
web3:1.0

To exit, press ctrl-d or type exit
> admin.nodeInfo
{
  enode: "enode://fe72e308acac0764d4371f3b50961d150da7aded789e490bfa8666863c967c8a681e868b9b696dc163e3965bae4d
22df0385d025186Ba5bedf22d94f882b90bd@45.127.44.4:30304?discport=22776",
  enr: "enr:-Ku4QN_0NncOo78JWpduCB78A3qbVO-aUL32YVHcq7yzMB4TZ4xvTM9exU8cygaCEGvoDPd_7X8tEA7MzutvO_uiJv-GAYyHmm
FRg2V0AMfGhL3NduuAgmlkgnV0gmlwhC1_LA5Jc2VjcDI1NmsxoQP-cuMTrKeHZEQ3HztQth0VBKet7XgOSQv6hm2WPJZ8ieRzbmFsaTH0Y3CC
dmCDdWRwgljUhjHVkcDaCdmA",
  id: "8546cf20cb3c4aebdcaa973711fa767ad2d086a31b00ff8f5b235580c08b1cf7",
  ip: "45.127.44.4",
  listenAddr: "[::]:30304",
  name: "Geth/v1.10.26-stable-e5eb32ac/linux-amd64/go1.18.5",
  ports: {
    discovery: 22776,
    listener: 30304
  },
```

Step 6: Connecting Peers

```
> admin.addPeer("enode://ce1a89ab411b422131f3c0db7dbd98f34a035da8dae1f6b5c34b86dadbc7bb984cb03d2cd5c1f62fe978c
32c45db585eb93513e5966aef5668d633ea7ff7c14a@127.0.0.1:30303?discport=0")
true
> admin.peers
[{
    caps: ["eth/66", "eth/67", "snap/1"],
    enode: "enode://ce1a89ab411b422131f3c0db7dbd98f34a035da8dae1f6b5c34b86dadbc7bb984cb03d2cd5c1f62fe978c32c45
db585eb93513e5966aef5668d633ea7ff7c14a@127.0.0.1:30303?discport=0",
    id: "1ed82e522c2c07cb17c649657e1ddfbce3963ece51690feb02f861a9b16f40f9",
    name: "Geth/v1.10.26-stable-e5eb32ac/linux-amd64/go1.18.5",
    network: {
      inbound: false,
      localAddress: "127.0.0.1:51666",
      remoteAddress: "127.0.0.1:30303",
      static: true,
      trusted: false
    },
    protocols: {
```

Step 7: Mining

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x0c0beaf945c08542b0fdec225a3e0f286fd79283"
>
```

```
> eth.getBalance("0x0c0beaf945c08542b0fdec225a3e0f286fd79283")
0
>
```

```
> eth.getBalance("0x0c0beaf945c08542b0fdec225a3e0f286fd79283")
0
> eth.blockNumber
0
>
```

```
> miner.start()
null
> miner.stop()
null
>
```

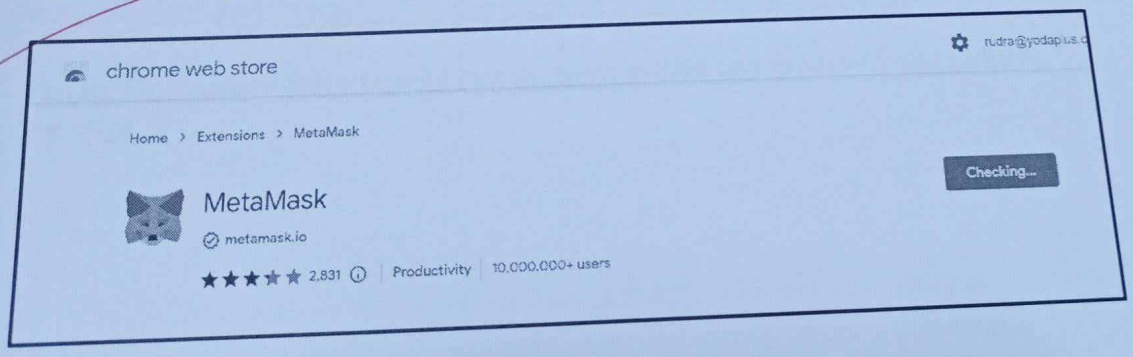
```
> miner.start()
null
> miner.stop()
null
> eth.blockNumber
45
> eth.getBalance("0x0c0beaf945c08542b0fdec225a3e0f286fd79283")
900000000000000000000
>
```

```
> eth.getBalance("0x0c0beaf945c08542b0fdec225a3e0f286fd79283")
900000000000000000000
> eth.blockNumber
45
>
```

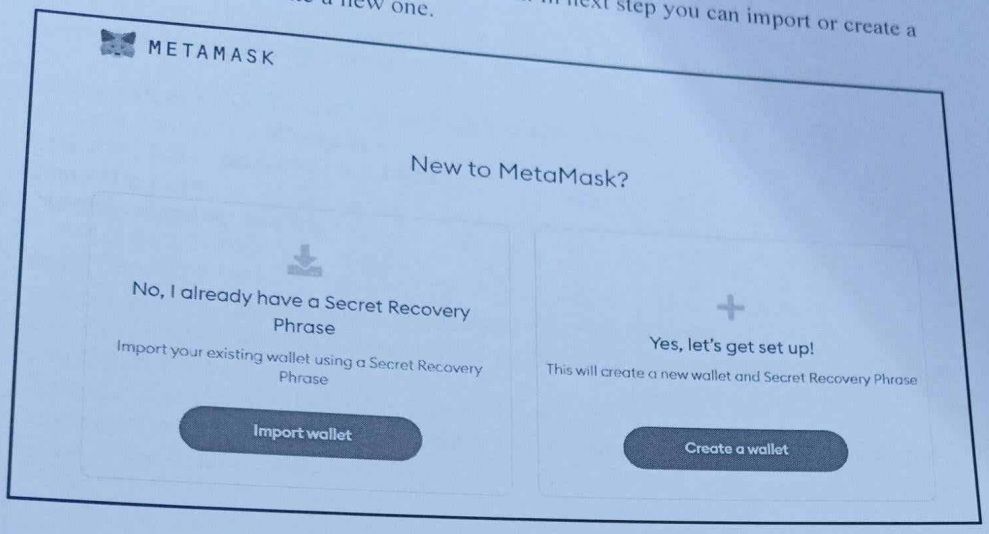
Step 8: Install MetaMask and Create the First Account

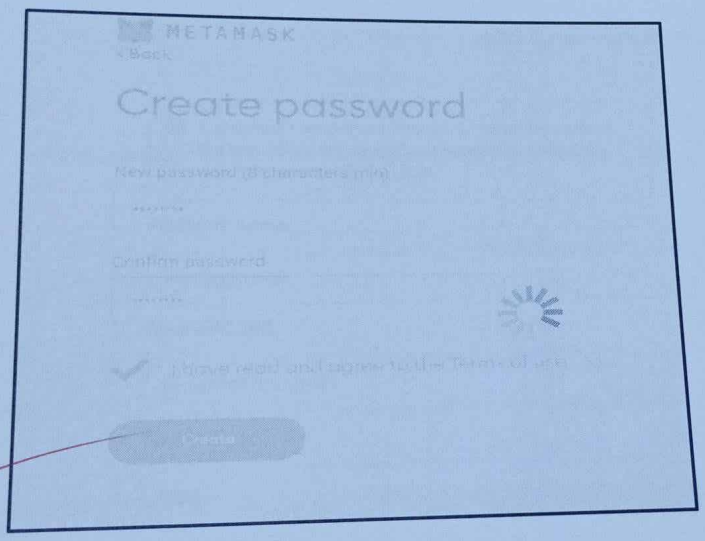
The MetaMask can be installed like an extension to most popular browsers or a mobile app. In this post, we will use MetaMask like an extension in Brave. This process is similar for other browsers like Chrome or Firefox.

Go to download page and click at "Install Metamask For BROWSER_NAME". Follow the installation steps until finish.



Click at "Get Started" to initialize MetaMask. In next step you can import or create a new wallet. Let us create a new one.





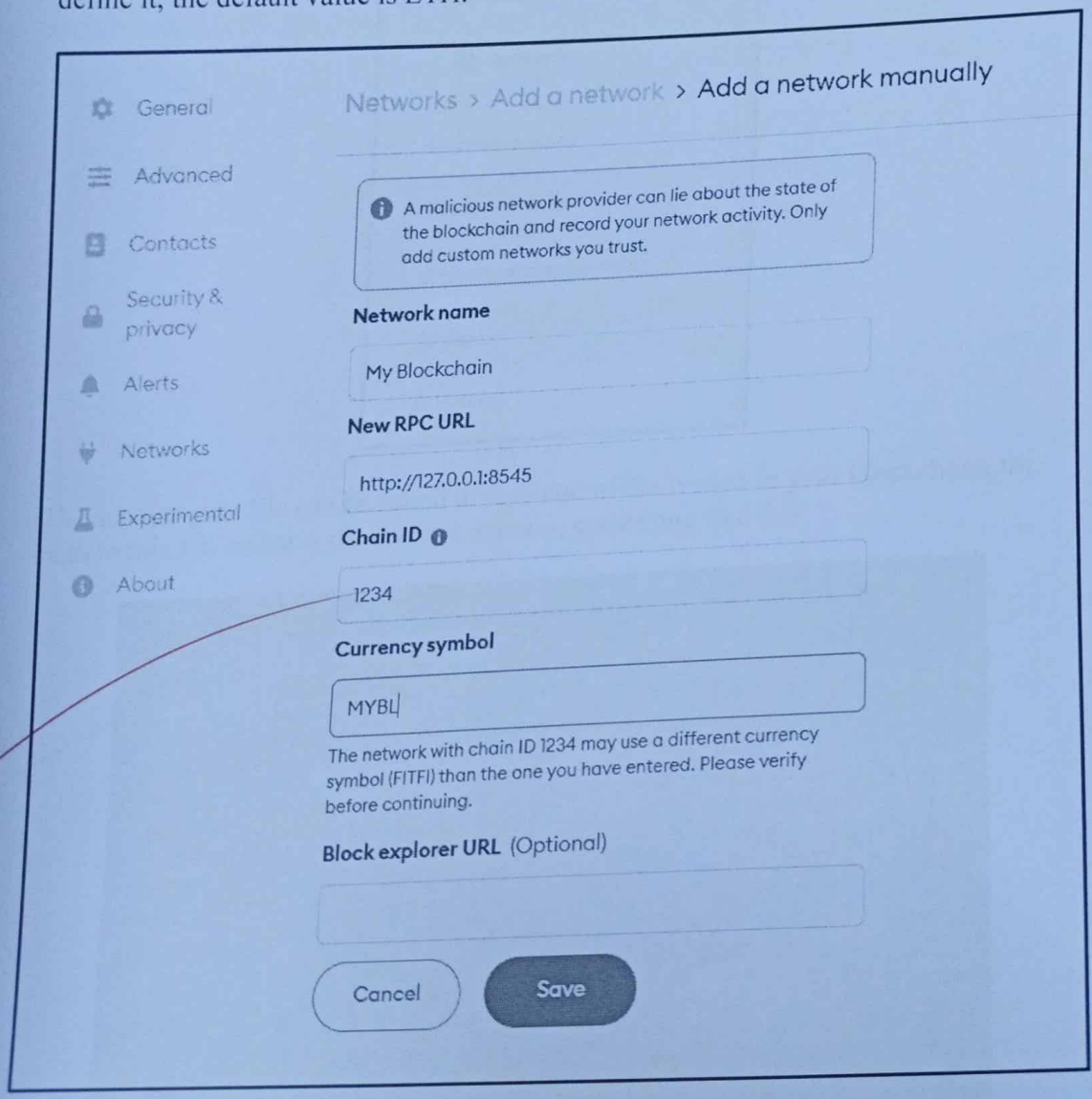
Private key: sudden leave beyond month tennis mother laptop barrel lounge liberty use edge

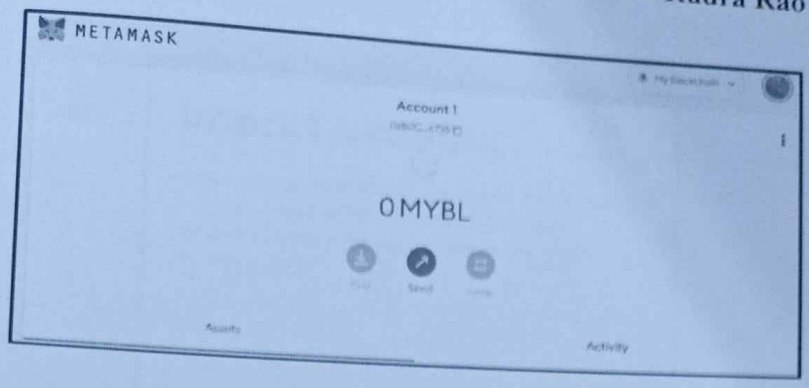
Step 9: Add My Blockchain Network

Let us add our Blockchain Network to MetaMask. Click on Ethereum Mainnet and select Add New Network option.

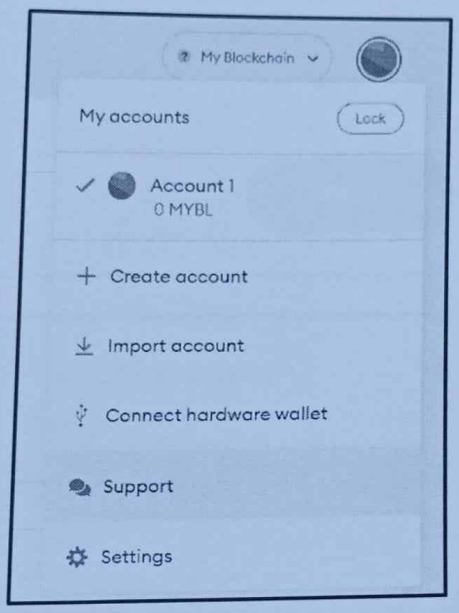
Define values to My Blockchain and click o Save button.
Network Name: My Blockchain
New RPC URL: http://127.0.0.1:8545
Chain ID: 1234
Currency Symbol: MYBL

Here,
Network Name: Any string to reference your blockchain name.
New RPC URL: The HTTP Server URL. In the HTTP Server is running on Node1 in default port 8545.
Chain ID: The chainId defined in genesis block.
Currency Symbol: A symbol to reference the coin in your blockchain. If you do not define it, the default value is ETH.

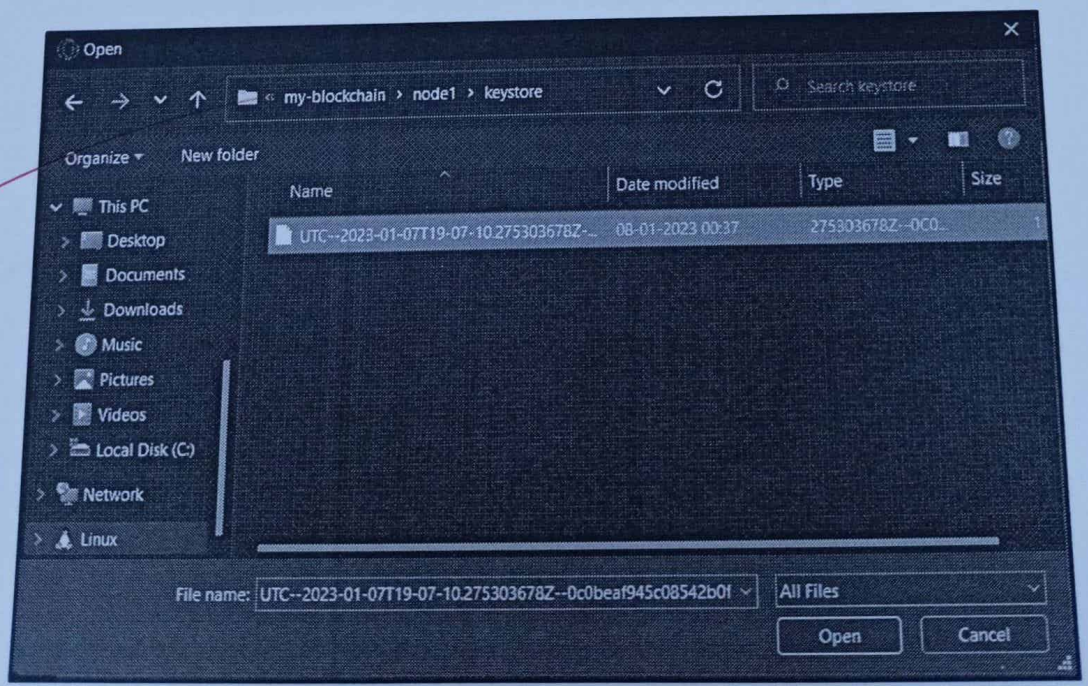


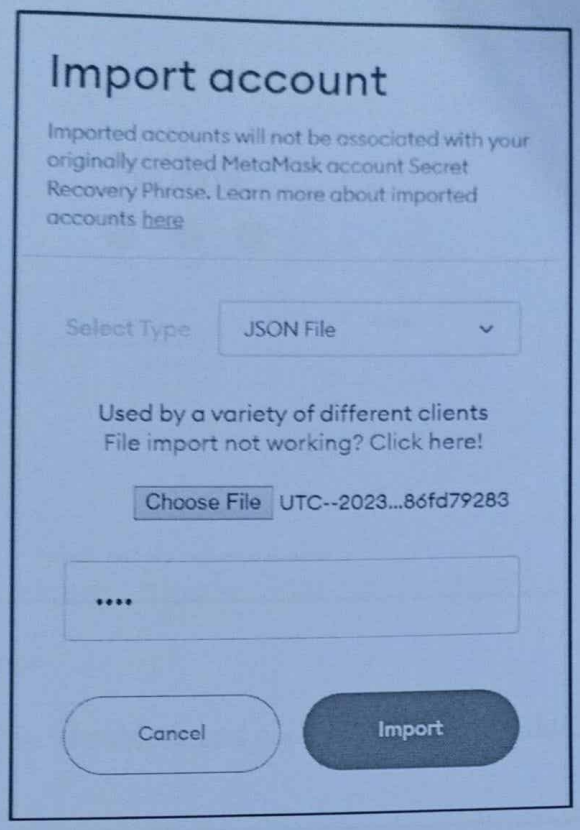


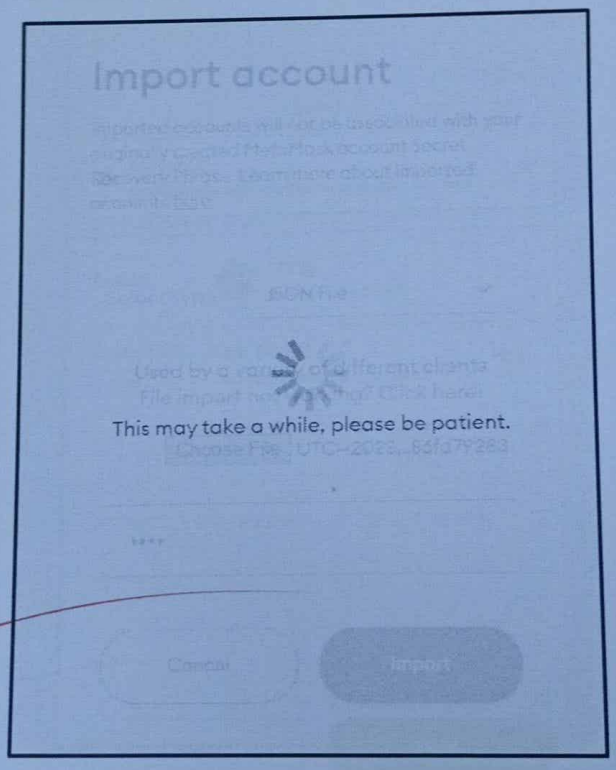
Step 10: Import Account
Select "Import Account" option in MetaMask, choose "Select Type" as JSON File, choose account json file in your blockchain, type the password and click on "Import" button.

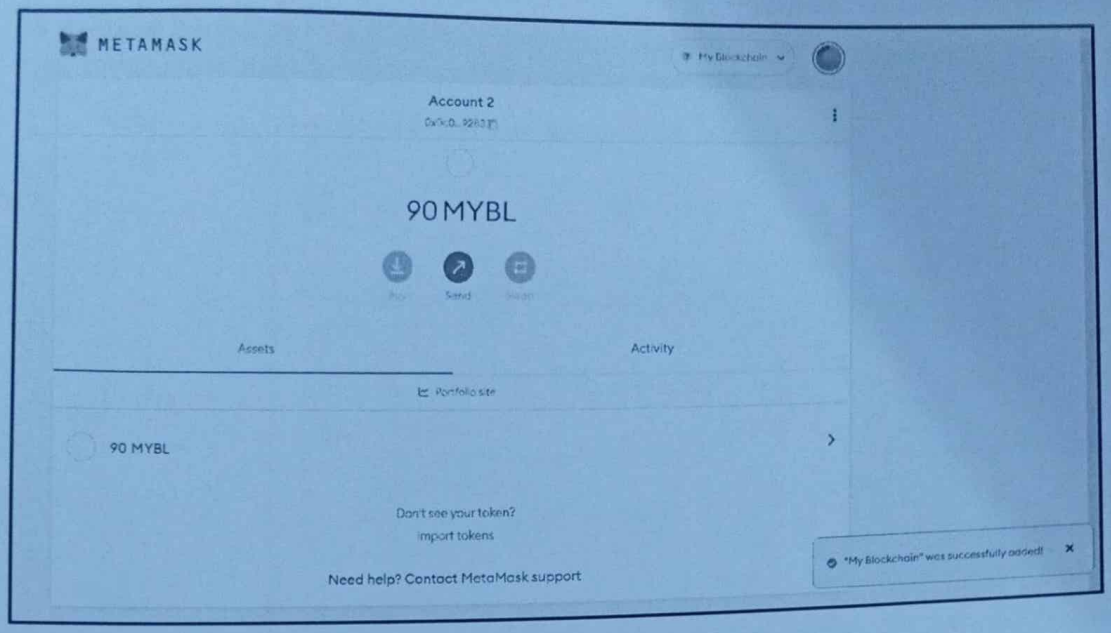


The account json file can be found in database of the Node2 in your blockchain, the path to this file contains the account address, something like this:



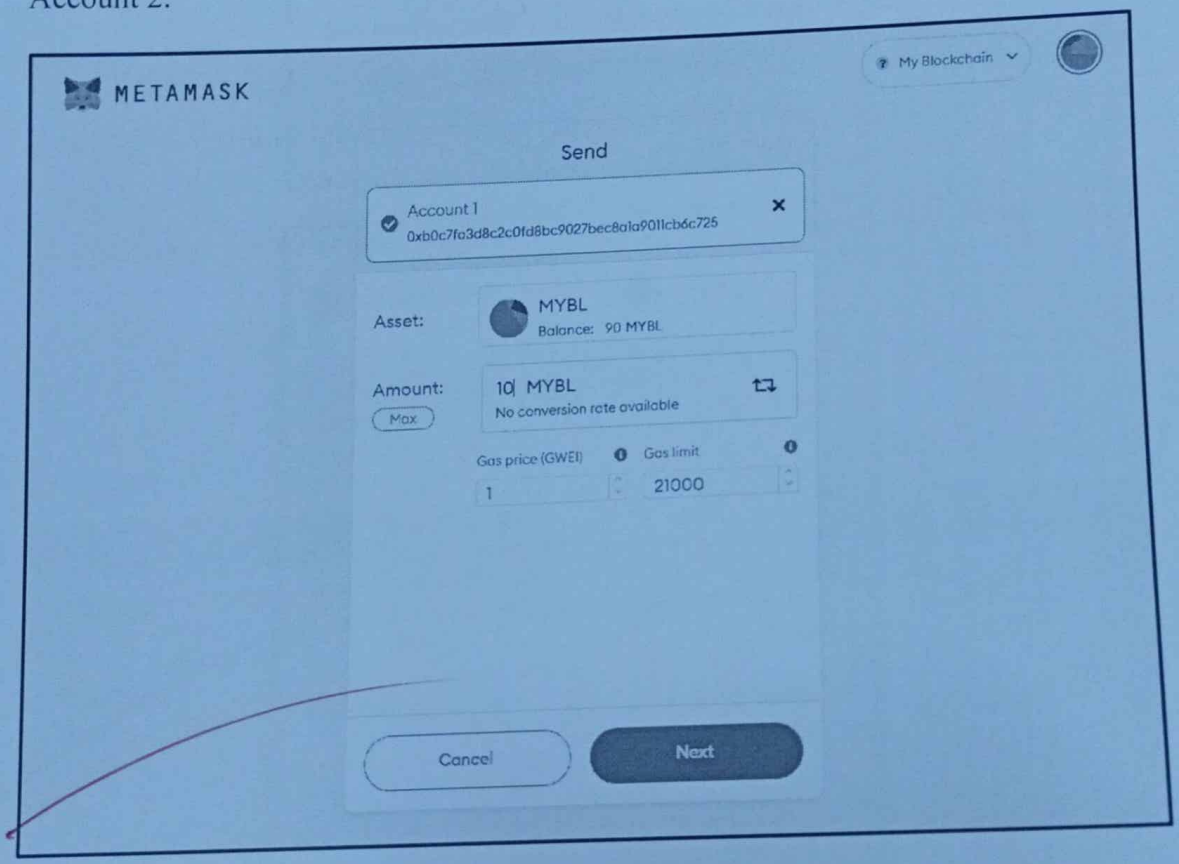

Import account
Imported accounts will not be associated with your originally created MetaMask account Secret Recovery Phrase. Learn more about imported accounts here
Select Type
JSON File
Used by a variety of different clients
File import not working? Click here!
Choose File
UTC--2023...86fd79283
••••
Cancel
Import


Import account
This may take a while, please be patient.

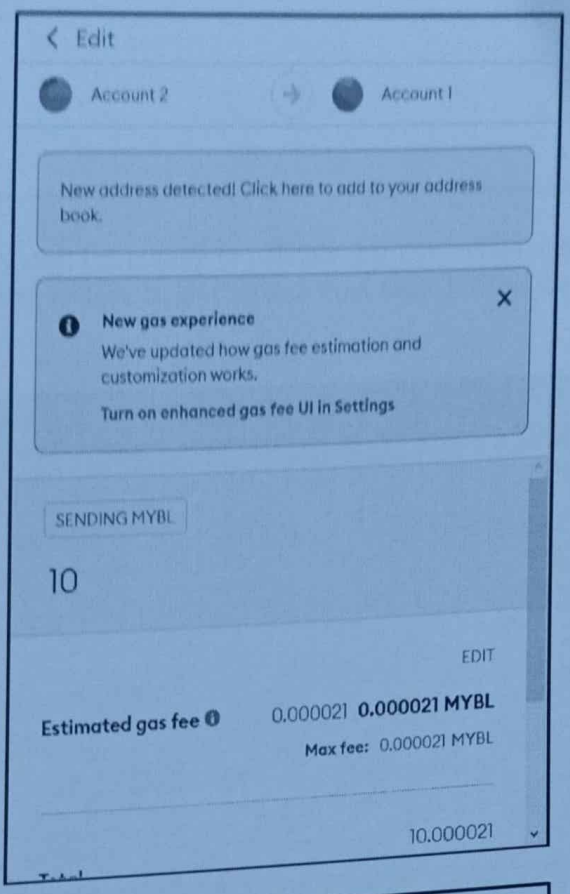


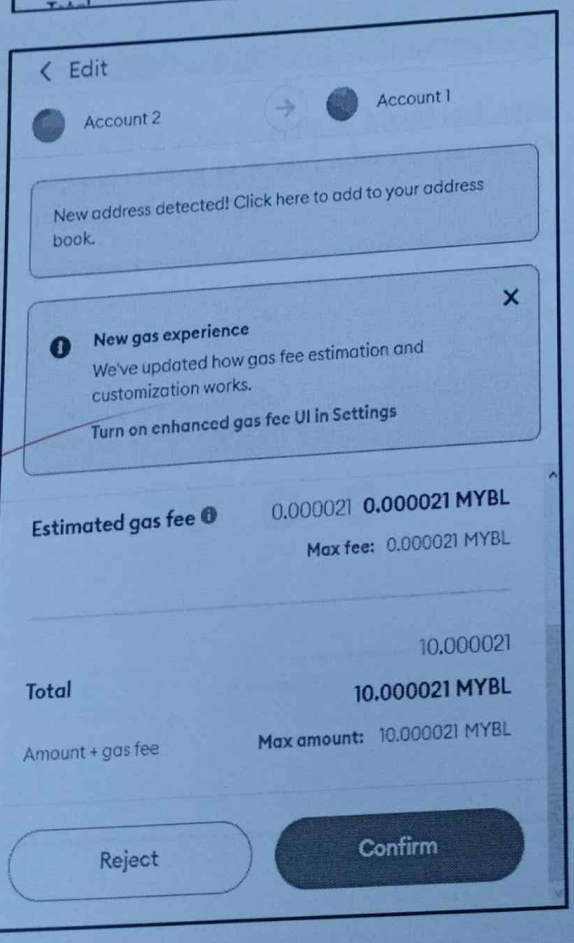
Step 11: Transfer Funds

First, go to Account 1 in MetaMask and copy the Account Public Address and back to Account 2.

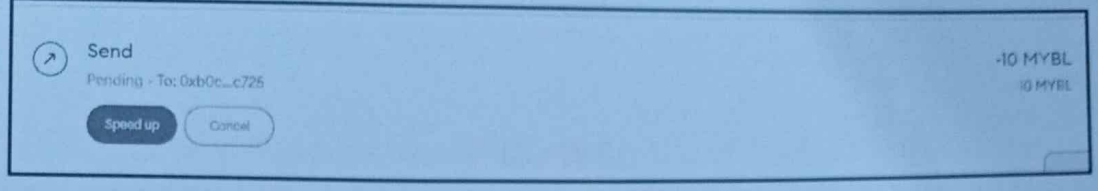


In Account 2 click on "Send" button, paste the public address of the Account 1, and type the value to transfer. Click on "Next" button and then click on "Confirm" button.

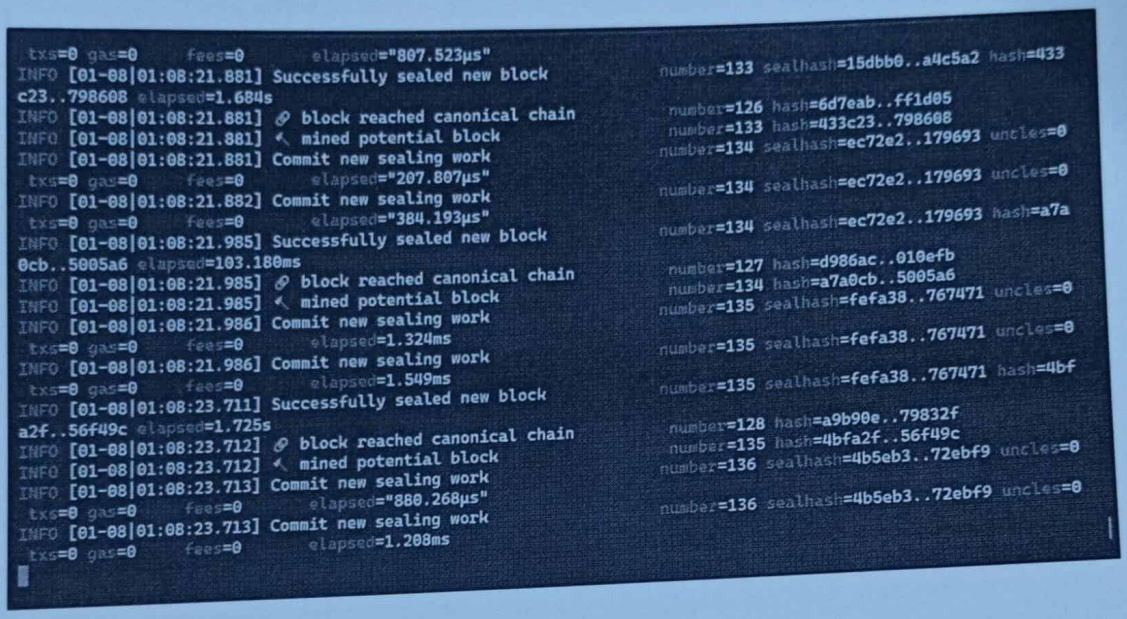




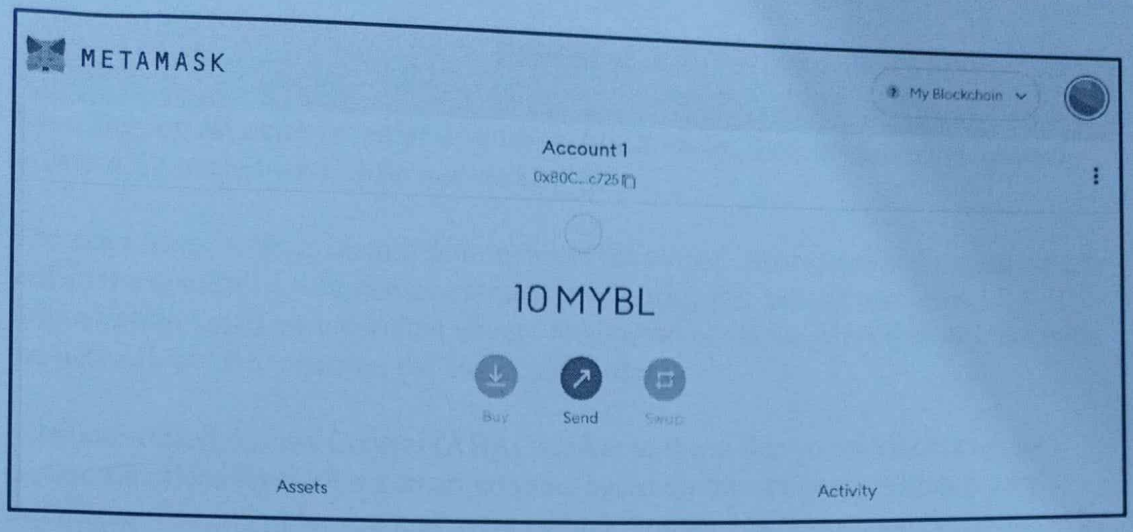
The transaction will be with status "Pending", waiting for a block to be mined for validation.



Now, it is necessary start the mining to validate that transaction. Let us do this in our blockchain using Geth in JavaScript Console from Node1. Type the command: miner.start()



Waiting some seconds and verify your account in MetaMask again.
If you want, you can stop the mining in Node1 after the transaction validation. Type command to stop the mining: miner.stop()


METAMASK
My Blockchain
Account 1
0x80C...c725
10 MYBL
Buy
Send
Swap
Assets
Activity

Practical 4

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

The asset-transfer-abac sample demonstrates the use of Attribute-based access control within the context of a simple asset transfer scenario. The sample also uses authorization based on individual client identities to allow the users that interact with the network to own assets on the blockchain ledger.

Attribute-Based Access Control (ABAC) refers to the ability to restrict access to certain functionality within a smart contract based on the attributes within a user's certificate. For example, you may want certain functions within a smart contract to be used only by application administrators. ABAC allows organizations to provide elevated access to certain users without requiring those users be administrators of the network. For more information, see Attribute-based access control in the Fabric CA users guide.

The asset-transfer-abac smart contract allows you to create assets that can be updated or transferred by the asset owner. However, the ability to create or remove assets from the ledger is restricted to identities with the abac.creator=true attribute. The identity that creates the asset is assigned as the asset owner. Only the owner can transfer the asset to a new owner or update the asset properties. In the course of the tutorial, we will use the Fabric CA client to create identities with the attribute required to create a new asset. We will then transfer the asset to another identity and demonstrate how the GetID() function can be used to enforce asset ownership. We will then use the owner identity to delete the asset. Both Attribute based access control and the GetID() function are provided by the Client Identity Chaincode Library.

Step 1: Start the network and deploy the smart contract

We can use the Fabric test network to deploy and interact with the asset-transfer-abac smart contract. Run the following command to change into the test network directory and bring down any running nodes:

```
rudra@DESKTOP-0BATOB7:~/prac2hf$ cd fabric-samples/test-network
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ ./network.sh down
Using docker and docker-compose
Stopping network
[+] Running 4/0
 ⠿ Volume compose_peer0.org2.example.com  Removed                    0.0s
 ⠿ Volume compose_peer0.org1.example.com  Removed                    0.0s
 ⠿ Volume compose_orderer.example.com     Removed                    0.0s
 ⠿ Volume compose_peer0.org1.example.com  Removed                    0.0s
Error: No such volume: docker_orderer.example.com
Error: No such volume: docker_peer0.org1.example.com
Error: No such volume: docker_peer0.org2.example.com
Removing remaining containers
Removing generated chaincode docker images
"docker kill" requires at least 1 argument.
See 'docker kill --help'.

Usage:  docker kill [OPTIONS] CONTAINER [CONTAINER...]

Kill one or more running containers
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

Run the following command to deploy the test network using Certificate Authorities:

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ ./network.sh up createChannel -ca
Creating channel 'mychannel'
Using certificate authorities
Bringing up network
LOCAL_VERSION=2.4.7
DOCKER_IMAGE_VERSION=2.4.7
CA_LOCAL_VERSION=1.5.5
CA_DOCKER_IMAGE_VERSION=1.5.5
Generating certificates using Fabric CA
Creating network "fabric_test" with the default driver
Creating ca_orderer ... done
Creating ca_org1    ... done
Creating ca_org2    ... done
Creating peer0.org1.example.com ... done
Creating peer0.org2.example.com ... done
Creating cli                    ... done
Creating orderer.example.com    ... done
+ fabric-ca-client enroll -u https://admin:adminpw@localhost:7054 --caname ca-org1 --tls.certfiles /home/rudra
/prac2hf/fabric-samples/test-network/organizations/fabric-ca/org1/ca-cert.pem
2023/01/08 13:51:11 [INFO] Created a default configuration file at /home/rudra/prac2hf/fabric-samples/test-net
work/organizations/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
2023/01/08 13:51:11 [INFO] TLS Enabled
2023/01/08 13:51:11 [INFO] generating key: &{A:ecdsa S:256}
2023/01/08 13:51:11 [INFO] encoded CSR
2023/01/08 13:51:11 [INFO] Stored client certificate at /home/rudra/prac2hf/fabric-samples/test-network/organi
```

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

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ ./network.sh deployCC -ccn abac -ccp ../asset-tra
nsfer-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
Vendoring Go dependencies at ../asset-transfer-abac/chaincode-go/
~/prac2hf/fabric-samples/asset-transfer-abac/chaincode-go ~/prac2hf/fabric-samples/test-network
~/prac2hf/fabric-samples/test-network
Finished vendoring Go dependencies
+ peer lifecycle chaincode package abac.tar.gz --path ../asset-transfer-abac/chaincode-go/ --lang golang --lab
el abac_1.0
+ res=0
++ peer lifecycle chaincode calculatepackageid abac.tar.gz
+ PACKAGE_ID=abac_1.0:83ea506205b13679754823369da6675b5bc6130d8729ce2cc0cee1f16b75ac1c
```

```
+ res=0
2023-01-08 13:51:37.117 EST 0001 INFO [chaincodeCmd] ClientWait -> txid [70abd358e2e1289bfba66865962132e265dec
6b21ac9cb4d8bdc2b5cba71bf81] committed with status (VALID) at localhost:9051
2023-01-08 13:51:37.117 EST 0002 INFO [chaincodeCmd] ClientWait -> txid [70abd358e2e1289bfba66865962132e265dec
6b21ac9cb4d8bdc2b5cba71bf81] committed with status (VALID) at localhost:7051
Chaincode definition committed on channel 'mychannel'
Using organization 1
Querying chaincode definition on peer0.org1 on channel 'mychannel'...
Attempting to Query committed status on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name abac
+ res=0
Committed chaincode definition for chaincode 'abac' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2M
SP: true]
Query chaincode definition successful on peer0.org1 on channel 'mychannel'
Using organization 2
Querying chaincode definition on peer0.org2 on channel 'mychannel'...
Attempting to Query committed status on peer0.org2, Retry after 3 seconds.
+ peer lifecycle chaincode querycommitted --channelID mychannel --name abac
+ res=0
Committed chaincode definition for chaincode 'abac' on channel 'mychannel':
Version: 1.0, Sequence: 1, Endorsement Plugin: escc, Validation Plugin: vscc, Approvals: [Org1MSP: true, Org2M
SP: true]
Query chaincode definition successful on peer0.org2 on channel 'mychannel'
Chaincode initialization is not required
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

Step 2: 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.

```
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$ export PATH=${PWD}/../bin:${PWD}:$PATH
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$ export FABRIC_CFG_PATH=$PWD/../config/
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$
```

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

```
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$ export PATH=${PWD}/../bin:${PWD}:$PATH
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$ export FABRIC_CFG_PATH=$PWD/../config/
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$ 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.

```
rudra@DESKTOP-0BAT0B7:~/prac2hf/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
 "${PWD}/organizations/fabric-ca/org1/tls-cert.pem"
2023/01/08 14:31:14 [INFO] Configuration file location: /home/rudra/prac2hf/fabric-samples/test-network/organi
zations/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
2023/01/08 14:31:14 [INFO] TLS Enabled
2023/01/08 14:31:14 [INFO] TLS Enabled
Password: creator1pw
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$
```

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.

```
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$ fabric-ca-client enroll -u https://creator1:creat
or1pw@localhost:7054 --caname ca-org1 -M "${PWD}/organizations/peerOrganizations/org1.example.com/users/creato
r1@org1.example.com/msp" --tls.certfiles "${PWD}/organizations/fabric-ca/org1/tls-cert.pem"
2023/01/08 14:35:05 [INFO] TLS Enabled
2023/01/08 14:35:05 [INFO] generating key: &{A:ecdsa S:256}
2023/01/08 14:35:05 [INFO] encoded CSR
2023/01/08 14:35:05 [INFO] Stored client certificate at /home/rudra/prac2hf/fabric-samples/test-network/organi
zations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp/signcerts/cert.pem
2023/01/08 14:35:05 [INFO] Stored root CA certificate at /home/rudra/prac2hf/fabric-samples/test-network/organ
izations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp/cacerts/localhost-7054-ca-org1
.pem
2023/01/08 14:35:05 [INFO] Stored Issuer public key at /home/rudra/prac2hf/fabric-samples/test-network/organiz
ations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp/IssuerPublicKey
2023/01/08 14:35:05 [INFO] Stored Issuer revocation public key at /home/rudra/prac2hf/fabric-samples/test-netw
ork/organizations/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp/IssuerRevocationPubli
cKey
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$
```

Now that we have enrolled the identity, run the command below to copy the Node OU configuration file into the creator1 MSP folder.

```
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$ cp "${PWD}/organizations/peerOrganizations/org1.e
xample.com/msp/config.yaml" "${PWD}/organizations/peerOrganizations/org1.example.com/users/creator1@org1.examp
le.com/msp/config.yaml"
```

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.

```
rudra@DESKTOP-0BATOB7:~/prac2hf/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 "${P
WD}/organizations/fabric-ca/org1/tls-cert.pem"
2023/01/08 14:35:52 [INFO] Configuration file location: /home/rudra/prac2hf/fabric-samples/test-network/organi
zations/peerOrganizations/org1.example.com/fabric-ca-client-config.yaml
2023/01/08 14:35:52 [INFO] TLS Enabled
2023/01/08 14:35:52 [INFO] TLS Enabled
Password: creator2pw
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

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

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ fabric-ca-client enroll -u https://creator2:creat
or2pw@localhost:7054 --caname ca-org1 --enrollment.attrs "abac.creator" -M "${PWD}/organizations/peerOrganizat
ions/org1.example.com/users/creator2@org1.example.com/msp" --tls.certfiles "${PWD}/organizations/fabric-ca/org
1/tls-cert.pem"
2023/01/08 14:36:17 [INFO] TLS Enabled
2023/01/08 14:36:17 [INFO] generating key: &{A:ecdsa S:256}
2023/01/08 14:36:17 [INFO] encoded CSR
2023/01/08 14:36:17 [INFO] Stored client certificate at /home/rudra/prac2hf/fabric-samples/test-network/organi
zations/peerOrganizations/org1.example.com/users/creator2@org1.example.com/msp/signcerts/cert.pem
2023/01/08 14:36:17 [INFO] Stored root CA certificate at /home/rudra/prac2hf/fabric-samples/test-network/organ
izations/peerOrganizations/org1.example.com/users/creator2@org1.example.com/msp/cacerts/localhost-7054-ca-org1
.pem
2023/01/08 14:36:17 [INFO] Stored Issuer public key at /home/rudra/prac2hf/fabric-samples/test-network/organiz
ations/peerOrganizations/org1.example.com/users/creator2@org1.example.com/msp/IssuerPublicKey
2023/01/08 14:36:17 [INFO] Stored Issuer revocation public key at /home/rudra/prac2hf/fabric-samples/test-netw
ork/organizations/peerOrganizations/org1.example.com/users/creator2@org1.example.com/msp/IssuerRevocationPubli
cKey
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

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

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ cp "${PWD}/organizations/peerOrganizations/org1.e
xample.com/msp/config.yaml" "${PWD}/organizations/peerOrganizations/org1.example.com/users/creator2@org1.examp
le.com/msp/config.yaml"
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

Step 3: 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:

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ export CORE_PEER_TLS_ENABLED=true
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ export CORE_PEER_LOCALMSPID="Org1MSP"
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ export CORE_PEER_MSPCONFIGPATH=${PWD}/organizatio
ns/peerOrganizations/org1.example.com/users/creator1@org1.example.com/msp
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organiz
ations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ export CORE_PEER_ADDRESS=localhost:7051
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ export TARGET_TLS_OPTIONS=(-o localhost:7050 --or
dererTLSHostnameOverride 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 "${PWD}/organizations/peerOrganizations/org2.example.co
m/peers/peer0.org2.example.com/tls/ca.crt")
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

Run the following command to create Asset1:

```
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$ peer chaincode invoke "${TARGET_TLS_OPTIONS[@]}"
-C mychannel -n abac -c '{"function":"CreateAsset","Args":["Asset1","blue","20","100"]}'
2023-01-08 11:08:35.933 IST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. re
sult: status:200
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$
```

You can use the command below to query the asset on the ledger:
The result will list the creator1 identity as the asset owner. The GetID() API reads the name and issuer from the certificate of the identity that submitted the transaction and assigns that identity as the asset owner.

```
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$ peer chaincode query -C mychannel -n abac -c '{"f
unction":"ReadAsset","Args":["Asset1"]}'
{"ID":"Asset1","color":"blue","size":20,"owner":"x509::CN=creator1,OU=org1+OU=client,O=Hyperledger,ST=North Ca
rolina,C=US::CN=ca.org1.example.com,O=org1.example.com,L=Durham,ST=North Carolina,C=US","appraisedValue":100}
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$
```

Step 4: 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:

```
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$ export RECIPIENT="x509::CN=user1,OU=client,O=Hype
rledger,ST=North Carolina,C=US::CN=ca.org1.example.com,O=org1.example.com,L=Durham,ST=North Carolina,C=US"
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$ peer chaincode invoke "${TARGET_TLS_OPTIONS[@]}"
-C mychannel -n abac -c '{"function":"TransferAsset","Args":["Asset1","'"$RECIPIENT"'"]}'
2023-01-08 11:08:35.933 IST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. re
sult: status:200
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$
```

Query the ledger to verify that the asset has a new owner:
We can see that Asset1 with is now owned by User1.

```
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$ peer chaincode query -C mychannel -n abac -c '{"f
unction":"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}
rudra@DESKTOP-0BAT0B7:~/prac2hf/fabric-samples/test-network$
```

Step 5: 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, let's try to update the asset using the creator1 identity first:

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:

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ peer chaincode invoke "${TARGET_TLS_OPTIONS[@]}"
-C mychannel -n abac -c '{"function":"UpdateAsset","Args":["Asset1","green","20","100"]}'
Error: endorsement failure during invoke. response: status:500 message:"submitting client not authorized to up
date asset, does not own asset"
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

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

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ export CORE_PEER_MSPCONFIGPATH=${PWD}/organizatio
ns/peerOrganizations/org1.example.com/users/User1@org1.example.com/msp
```

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.

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ peer chaincode invoke "${TARGET_TLS_OPTIONS[@]}"
-C mychannel -n abac -c '{"function":"UpdateAsset","Args":["Asset1","green","20","100"]}'
                                [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. re
sult: status:200
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

Run the query command again to verify that the asset has changed color:
The result will display that Asset1 is now green.

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ peer chaincode query -C mychannel -n abac -c '{"f
unction":"ReadAsset","Args":["Asset1"]}'
{"ID":"Asset1","color":"green","size":20,"owner":"x509::CN=user1,OU=client,O=Hyperledger,ST=North Carolina,C=U
S::CN=ca.org1.example.com,O=org1.example.com,L=Durham,ST=North Carolina,C=US","appraisedValue":100}
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

Step 5: Delete the asset

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

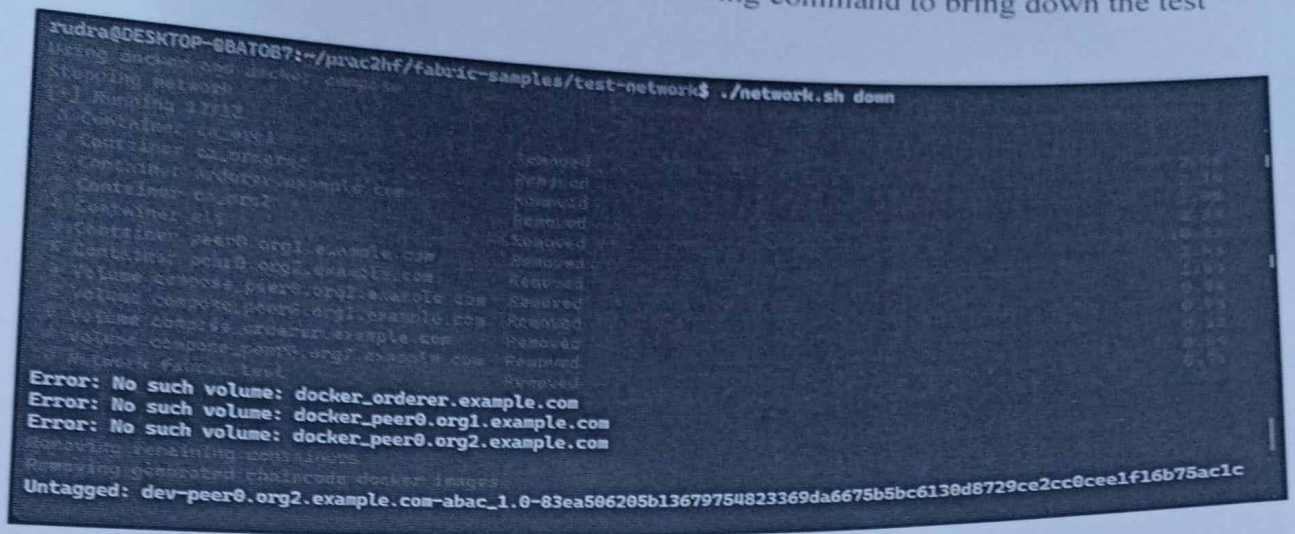
```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ peer chaincode invoke "${TARGET_TLS_OPTIONS[@]}"
-C mychannel -n abac -c '{"function":"DeleteAsset","Args":["Asset1"]}'
                                [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. re
sult: status:200
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

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

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ peer chaincode query -C mychannel -n abac -c '{"f
unction":"ReadAsset","Args":["Asset1"]}'
Error: endorsement failure during query. response: status:500 message:"the asset Asset1 does not exist"
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

Step 6: Cleaning Up

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



Practical 5

Aim: Car auction network

Step 1: Set up the blockchain network

Before starting this practical you must stop this network:

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ ./network.sh down

[+] Running 4/0


Error: No such volume: docker_orderer.example.com
Error: No such volume: docker_peer0.org1.example.com
Error: No such volume: docker_peer0.org2.example.com

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

Usage:  docker kill [OPTIONS] CONTAINER [CONTAINER...]

Kill one or more running containers
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

We will also use the following commands to kill any stale or active containers. Note, this will take down all of your containers whether they're Fabric related or not.

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ docker rm -f $(docker ps -aq)
"docker rm" requires at least 1 argument.
See 'docker rm --help'.

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

Remove one or more containers
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$ docker rmi -f $(docker images | grep fabcar | awk
 '{print $3}')
"docker rmi" requires at least 1 argument.
See 'docker rmi --help'.

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

Remove one or more images
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/test-network$
```

Step 2: Launch the network

Launch your network using the startFabric.sh shell script. This command will spin up a blockchain network comprising peers, orderers, certificate authorities and more. It will also install and instantiate a JavaScript version of the FabCar smart contract which will be used by our application to access the ledger. We'll learn more about these components as we go through the tutorial.

```
rudra@DESKTOP-0BATOB7I:~/prac2hf/fabric-samples$ cd fabcar/
rudra@DESKTOP-0BATOB7I:~/prac2hf/fabric-samples/fabcar$ ./startFabric.sh javascript
~/prac2hf/fabric-samples/test-network ~/prac2hf/fabric-samples/fabcar
[+] Running 4/0

Error: No such volume: docker_orderer.example.com
Error: No such volume: docker_peer0.org1.example.com
Error: No such volume: docker_peer0.org2.example.com

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

Usage:  docker kill [OPTIONS] CONTAINER [CONTAINER...]

Kill one or more running containers
```

```
- CHANNEL_NAME: mychannel
- CC_NAME: fabcar
- CC_SRC_PATH: ../chaincode/fabcar/javascript/
- CC_SRC_LANGUAGE: javascript
- CC_VERSION: 1
- CC_SEQUENCE: 1
- CC_END_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FCN: initLedger
- DELAY: 3
- MAX_RETRY: 5
- VERBOSE: false
+ peer lifecycle chaincode package fabcar.tar.gz --path ../chaincode/fabcar/javascript/ --lang node --label fa
bcar_1
+ res=0
++ peer lifecycle chaincode calculatepackageid fabcar.tar.gz
+ PACKAGE_ID=fabcar_1:6ab85d9d0ab315198d930d1b0eb8cf8c3f59849549226a86553e29eb1fbc0a12
Chaincode is packaged
Installing chaincode on org1...
Using organization 1
+ peer lifecycle chaincode queryinstalled --output json
+ jq -r 'try (.installed_chaincodes[].package_id)'
+ grep '^fabcar_1:6ab85d9d0ab315198d930d1b0eb8cf8c3f59849549226a86553e29eb1fbc0a12$'
+ test 1 -ne 0
+ peer lifecycle chaincode install fabcar.tar.gz
```

```
  Then, install dependencies and run the test using:
    mvn test

  The test will invoke the sample client app which perform the following:
    - Enroll admin and appUser and import them into the wallet (if they don't already exist there)
    - Submit a transaction to create a new car
    - Evaluate a transaction (query) to return details of this car
    - Submit a transaction to change the owner of this car
    - Evaluate a transaction (query) to return the updated details of this car

Go:

  Start by changing into the "go" directory:
    cd go

  Then, install dependencies and run the test using:
    go run fabcar.go

  The test will invoke the sample client app which perform the following:
    - Import user credentials into the wallet (if they don't already exist there)
    - Submit a transaction to create a new car
    - Evaluate a transaction (query) to return details of this car
    - Submit a transaction to change the owner of this car
    - Evaluate a transaction (query) to return the updated details of this car

rudra@DESKTOP-0BATOB7I:~/prac2hf/fabric-samples/fabcar$
```

Step 3: Install the application

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

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar$ cd javascript/
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$ npm install
npm WARN deprecated querystring@0.2.0: The querystring API is considered Legacy. new code should use the URLSe
archParams API instead.
npm WARN deprecated uuid@3.4.0: Please upgrade  to version 7 or higher.  Older versions may use Math.random()
in certain circumstances, which is known to be problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated mkdirp@0.5.1: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1
.x. (Note that the API surface has changed to use Promises in 1.x.)

added 351 packages, and audited 352 packages in 1m

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

4 vulnerabilities (1 moderate, 2 high, 1 critical)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$
```

```
Follow the instructions for the programming language of your choice:

JavaScript:

  Start by changing into the "javascript" directory:
    cd javascript

  Next, install all required packages:
    npm install

  Then run the following applications to enroll the admin user, and register a new user
  called appUser which will be used by the other applications to interact with the deployed
  FabCar contract:
    node enrollAdmin
    node registerUser

  You can run the invoke application as follows. By default, the invoke application will
  create a new car, but you can update the application to submit other transactions:
    node invoke

  You can run the query application as follows. By default, the query application will
  return all cars, but you can update the application to evaluate other transactions:
    node query
```

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

Once npm install 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:

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$ ls
enrollAdmin.js  invoke.js  node_modules  package-lock.json  package.json  query.js  registerUser.js  wallet
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$
```

Step 4: Enrolling the admin user

When we created the network, an admin user — called admin — was created as the registrar for the certificate authority (CA). Our first step is to generate the private key, public key, and X.509 certificate for admin using the enroll.js program. This process uses a Certificate Signing Request (CSR) — the private and public key are first generated locally, and the public key is then sent to the CA which returns an encoded certificate for use by the application. These three credentials are then stored in the wallet, allowing us to act as an administrator for the CA.

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

Let's enroll user admin:

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$ node enrollAdmin.js
Wallet path: /home/rudra/prac2hf/fabric-samples/fabcar/javascript/wallet
Successfully enrolled admin user "admin" and imported it into the wallet
```

This command has stored the CA administrator's credentials in the wallet directory.

Step 5: 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:

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$ node registerUser.js
Wallet path: /home/rudra/prac2hf/fabric-samples/fabcar/javascript/wallet
Successfully registered and enrolled admin user "appUser" and imported it into the wallet
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$
```

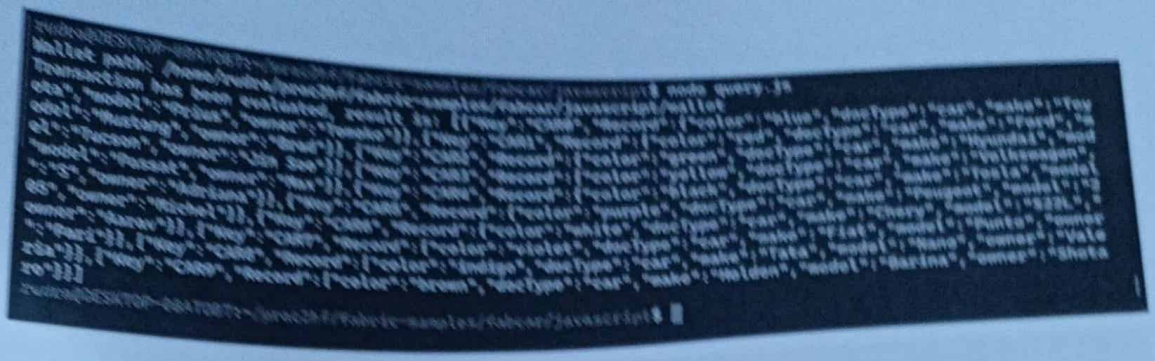
Similar to the admin enrollment, this program uses a CSR to enroll user1 and store its credentials alongside those of admin in the wallet. We now have identities for two separate users — admin and user1 — and these are used by our application.

Step 6: 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.

Applications read data from the ledger using a query. The most common queries involve the current values of data in the ledger – its world state. The world state is represented as a set of key-value pairs, and applications can query data for a single key or multiple keys. Moreover, the ledger world state can be configured to use a database like CouchDB which supports complex queries when key-values are modeled as JSON data. This can be very helpful when looking for all assets that match certain keywords with particular values; all cars with a particular owner, for example.

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:



Step 7: Updating the ledger

Now that we've done a few ledger queries and added a bit of code, we're ready to update the ledger. There are a lot of potential updates we could make, but let's start by creating a new car.

From an application perspective, updating the ledger is simple. An application submits a transaction to the blockchain network, and when it has been validated and committed, the application receives a notification that the transaction has been successful. Under the covers this involves the process of consensus whereby the different components of the blockchain network work together to ensure that every proposed update to the ledger is valid and performed in an agreed and consistent order.

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:

See how the applications calls the smart contract transaction createCar to create a black Honda Accord with an owner named Tom. We use CAR12 as the identifying key here, just to show that we don't need to use sequential keys.

```
    // Evaluate the specified transaction.
    // queryCar transaction - requires 1 argument, ex: ('queryCar', 'CAR4')
    // queryAllCars transaction - requires no arguments, ex: ('queryAllCars')
    const result = await contract.evaluateTransaction('queryCar', 'CAR12');
    console.log(`Transaction has been evaluated, result is: ${result.toString()}`);
```

Save it and run the program:

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$ node invoke.js
Wallet path: /home/rudra/prac2hf/fabric-samples/fabcar/javascript/wallet
Transaction has been submitted
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$
```

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$ node query.js
Wallet path: /home/rudra/prac2hf/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: {"color":"Black","docType":"car","make":"Honda","model":"Accord","owner":"Tom"}
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$
```

Suppose an individual wants to donate the car. Let's say that Tom is feeling generous, and he wants to give his Honda Accord to someone named Dave.
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:

```
// Submit the specified transaction.
// createCar transaction - requires 5 argument, ex: ('createCar', 'CAR12', 'Honda', 'Accord', 'Black', 'Tom')
// changeCarOwner transaction - requires 2 args , ex: ('changeCarOwner', 'CAR12', 'Dave')
await contract.submitTransaction('changeCarOwner', 'CAR10', 'Dave');
console.log('Transaction has been submitted');
```

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$ node invoke.js
Wallet path: /home/rudra/prac2hf/fabric-samples/fabcar/javascript/wallet
Transaction has been submitted
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$
```

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$ node query.js
Wallet path: /home/rudra/prac2hf/fabric-samples/fabcar/javascript/wallet
Transaction has been evaluated, result is: {"color":"Black","docType":"car","make":"Honda","model":"Accord","o
wner":"Dave"}
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$
```

Step 8: Then we will shut down the chain.

```
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar/javascript$ cd ..
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar$ ./network.sh down
-bash: ./network.sh: No such file or directory
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar$ ls
go  java  javascript  networkDown.sh  startFabric.sh  typescript
rudra@DESKTOP-0BATOB7:~/prac2hf/fabric-samples/fabcar$ ./networkDown.sh down
+ pushd ../test-network
~/prac2hf/fabric-samples/test-network ~/prac2hf/fabric-samples/fabcar
+ ./network.sh down

[+] Running 5/7
 ⠿ Container peer0.org2.example.com  Stopping                                                    2.0s
 ⠿ Container peer0.org1.example.com  Stopping                                                    2.0s
```

Practical 6

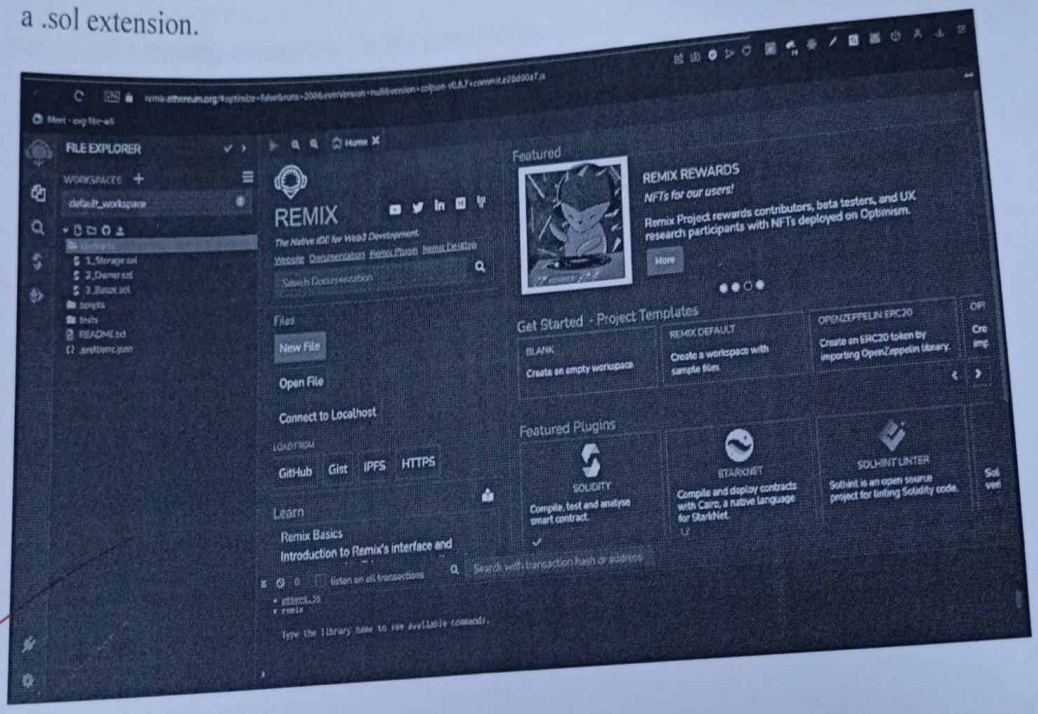
Aim: Develop a voting application using Hyperledger and Ethereum. Build a decentralized app that combines Ethereum's Web3 and Solidity smart contracts with Hyperledger's hosting Fabric and Chaincode EVM(using solidity)

Step 1: Open Remix IDE on the browser.

Using remix.ethereum link: https://remix.ethereum.org/

What is remix.ethereum?
Remix Project is a platform for development tools that use a plugin architecture. It encompasses sub-projects including Remix Plugin Engine, Remix Libraries, and of course Remix IDE. Remix IDE is an open-source web and desktop application. It fosters a fast development cycle and has a rich set of plugins with intuitive GUIs. Remix is used for the entire journey of contract development with Solidity language as well as a playground for learning and teaching Ethereum.

Then click on the "New File" button and name the file. Make sure that the file has a .sol extension.



Step 2: Write the following Solidity Contract.

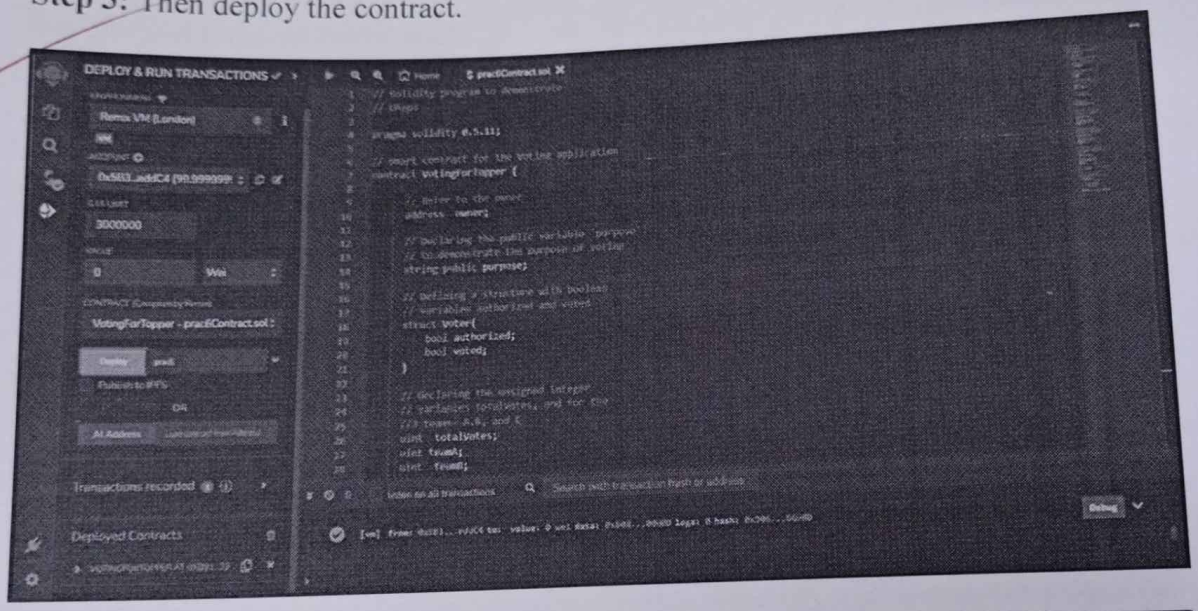
```
// Solidity program to demonstrate
// DApps

pragma solidity 0.5.11;

// Smart Contract for the Voting application
contract VotingForTopper {

    // Refer to the owner
    address owner;
```

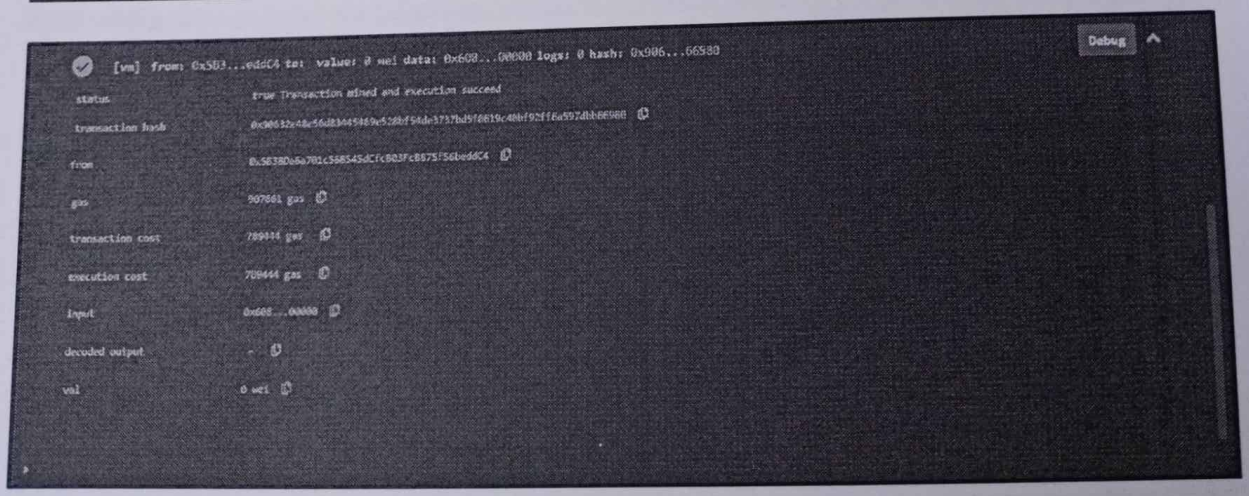
```
    // Declaring the public variable 'purpose'
    // to demonstrate the purpose of voting
    string public purpose;

    // Defining a structure with boolean
    // variables authorized and voted
    struct Voter{
            bool authorized;
            bool voted;
    }

    // Declaring the unsigned integer
    // variables totalVotes, and for the
    //3 teams- A,B, and C
    uint totalVotes;
    uint teamA;
    uint teamB;
    uint teamC;

    // Creating a mapping for the total Votes
    mapping(address=>Voter) info;

    // Defining a constructor indicating
    // the purpose of voting
    constructor(
    string memory _name) public{
            purpose = _name;
            owner = msg.sender;
    }

    // Defining a modifier to
    // verify the ownership
    modifier ownerOn() {
            require(msg.sender==owner);
            _;
    }

    // Defining a function to verify
    // the person is voted or not
    function authorize(
    address _person) ownerOn public {
            info[_person].authorized= true;

    }

    // Defining a function to check and
    // skip the code if the person is already
    // voted else allow to vote and
    // calculate totalvotes for team A
```

```
function temaAF(address _address) public {
        require(
        !info[_address].voted,
        "already voted person");
        require(
        info[_address].authorized,
        "You Have No Right for Vote");
        info[_address].voted = true;
        teamA++;
        totalVotes++;
}

// Defining a function to check
// and skip the code if the person
// is already voted else allow to vote
// and calculate totalvotes for team B
function temaBF(address _address) public {
require(
!info[_address].voted,
"already voted person");
        require(
        info[_address].authorized,
        "You Have No Right for Vote");
        teamB++;
        info[_address].voted = true;
        totalVotes++;
}

// Defining a function to check
// and skip the code if the person
// is already voted else allow to vote
// and calculate totalvotes for team C
function temaCF(address _address) public returns(
string memory){
        require(
        !info[_address].voted,
        "already voted person");
        require(
        info[_address].authorized,
        "You Have No Right for Vote");
        info[_address].voted = true;
        teamC++;
        totalVotes++;
        return("Thanks for Voting");
}

function totalVotesF() public view returns(uint){
        return totalVotes;
}
```

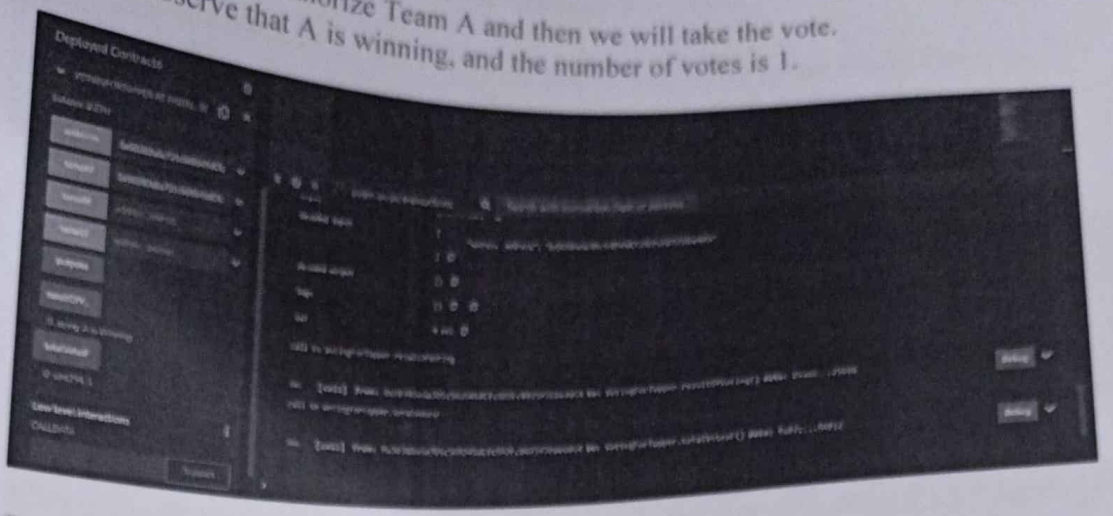
```
// Defining a function to announce
// the result of voting and
// the name of the winning team
function resultOfVoting() public view returns(
string memory){
        if(teamA>teamB){
                if(teamA>teamC){
                        return"A is Winning";
                }
                else if(teamC>teamA){
                        return "C is Winning"; } }
        else if(teamB>teamC) {
                return "B is Winning";
        }
        else if(
        teamA==teamB && teamA==teamC || teamB==teamC ){
                return "No One is Winning";
        }
}
}
```

Step 3: Then deploy the contract.

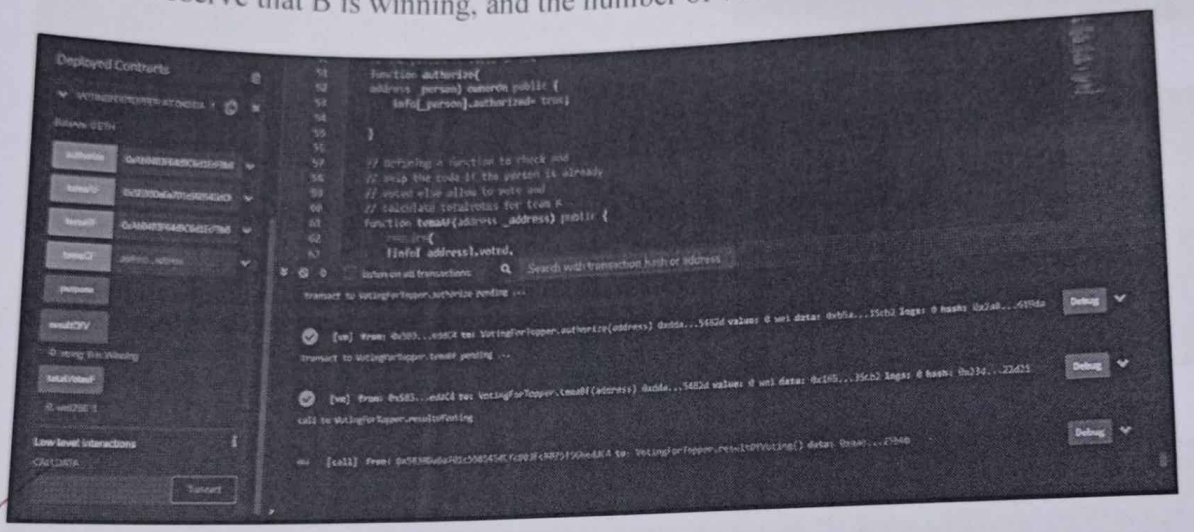




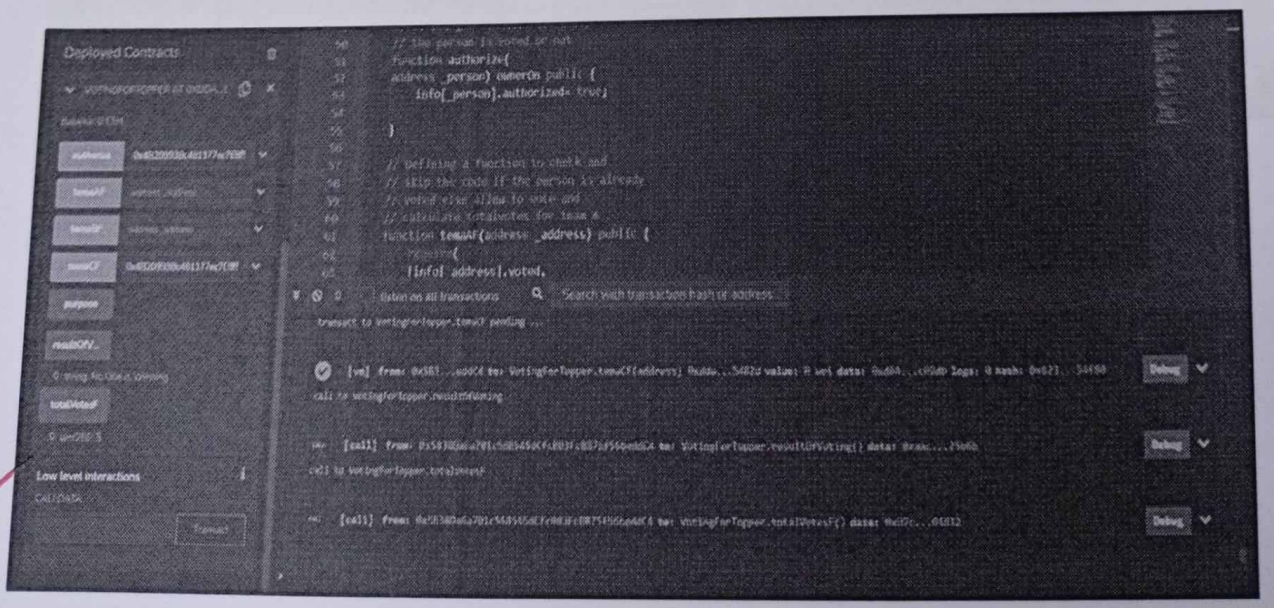
Step 4: Test the vote for Team A.
Here we will first authorize Team A and then we will take the vote.
We can observe that A is winning, and the number of votes is 1.



Step 5: Test the vote for Team B.
Here we will first authorize Team B and then we will take the vote.
We can observe that B is winning, and the number of votes is 2.



Step 6: Test the vote for Team C.
Here we will first authorize Team C and then we will take the vote.
We can observe that C is No One is Winning, and the number of votes is 3.

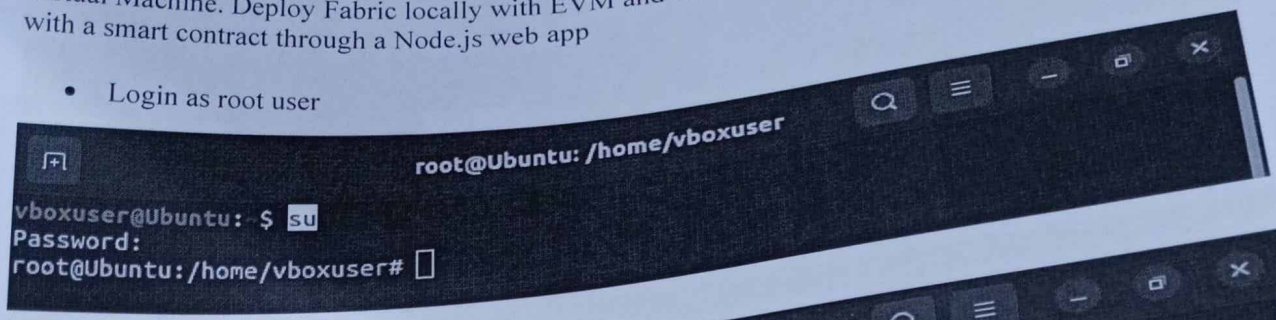


Rudra Rao – CS21020

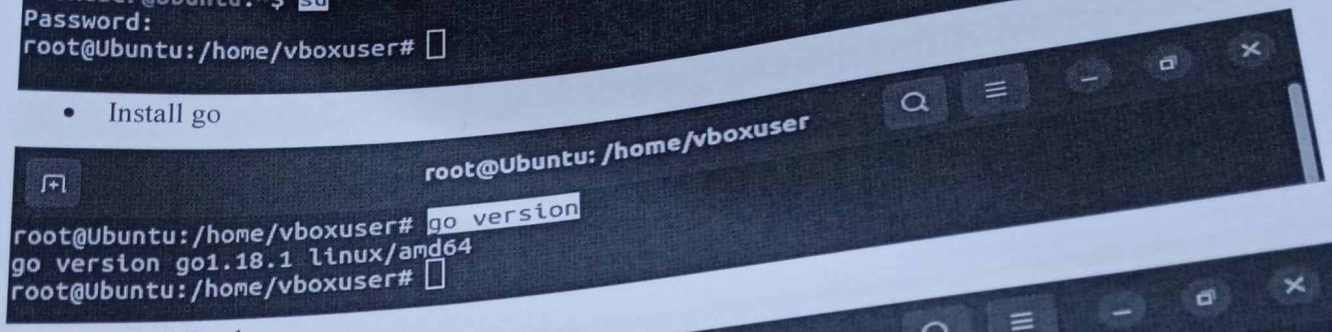
Practical 7

Aim: Create a block chain app for loyalty points with Hyperledger Fabric Ethereum Virtual Machine. Deploy Fabric locally with EVM and create a proxy for interacting with a smart contract through a Node.js web app

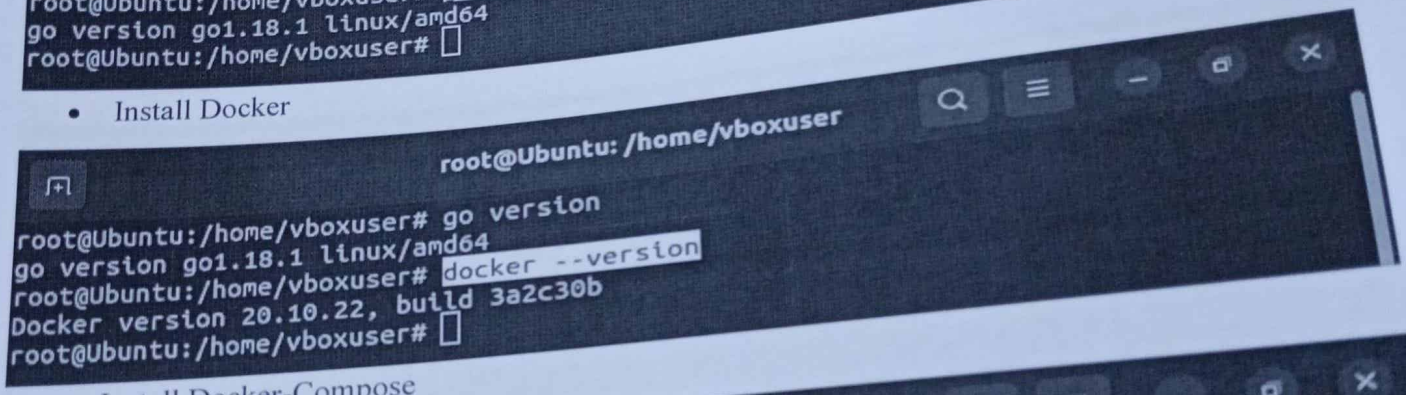
- Login as root user



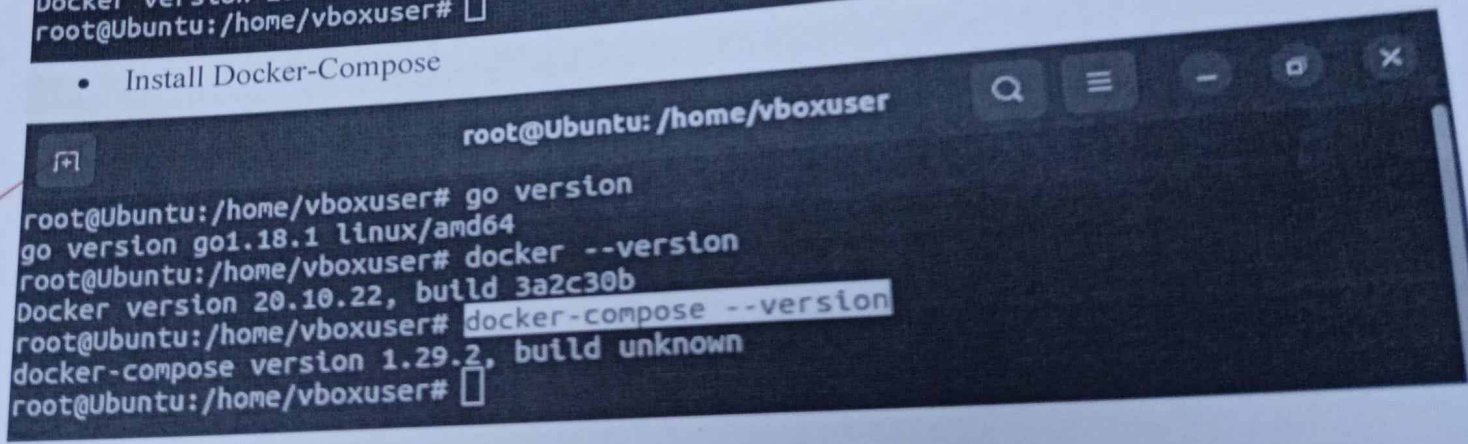
- Install go



- Install Docker



- Install Docker-Compose



- Install node

```
root@Ubuntu: /home/vboxuser
root@Ubuntu:/home/vboxuser# go version
go version go1.18.1 linux/amd64
root@Ubuntu:/home/vboxuser# docker --version
Docker version 20.10.22, build 3a2c30b
root@Ubuntu:/home/vboxuser# docker-compose --version
docker-compose version 1.29.2, build unknown
root@Ubuntu:/home/vboxuser# node -v
v8.17.0
root@Ubuntu:/home/vboxuser#
```

- Install npm

```
root@Ubuntu:/home/vboxuser# docker --version
Docker version 20.10.22, build 3a2c30b
root@Ubuntu:/home/vboxuser# docker-compose --version
docker-compose version 1.29.2, build unknown
root@Ubuntu:/home/vboxuser# node -v
v8.17.0
root@Ubuntu:/home/vboxuser# npm -v
6.13.4
root@Ubuntu:/home/vboxuser#
```

- Remove all your docker containers and images.

```
root@Ubuntu: /home/vboxuser
root@Ubuntu:/home/vboxuser# docker stop $(docker ps -a -q)
"docker stop" requires at least 1 argument.
See 'docker stop --help'.

Usage:  docker stop [OPTIONS] CONTAINER [CONTAINER...]

Stop one or more running containers
root@Ubuntu:/home/vboxuser#
```

```
root@Ubuntu: /home/vboxuser
root@Ubuntu:/home/vboxuser# docker stop $(docker ps -a -q)
"docker stop" requires at least 1 argument.
See 'docker stop --help'.

Usage:  docker stop [OPTIONS] CONTAINER [CONTAINER...]

Stop one or more running containers
root@Ubuntu:/home/vboxuser# docker rm $(docker ps -a -q)
"docker rm" requires at least 1 argument.
See 'docker rm --help'.

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

Remove one or more containers
root@Ubuntu:/home/vboxuser#
```

```
Remove one or more containers
root@Ubuntu:/home/vboxuser# docker rmi $(docker images -q) -f
Untagged: hyperledger/fabric-ca:1.2.0
Untagged: hyperledger/fabric-ca@sha256:5fe6d502e52ec2a8d98ee5653e1ba31952098115
fb57710ddae86f2c3cc82dad
Deleted: sha256:66cc132bd09c7dcf5c1f7075f4b2b7b6304b3919cc73738ac2cd029a92901e4
a
Deleted: sha256:c51a105ee6f098af9c1963d06623815b89eeb08f12c0fd0b33ba57318e2a130
7
Deleted: sha256:1d110e33cb2f593ed9cea02cf609e243db10d42f905c95bef6be29fae4c387d
e
Deleted: sha256:44729effe2fc4c39ca5ed984ac7f866f0dd61d2f30bf7ca63466e56c8bf1df3
c
Deleted: sha256:aea86943cceb7a4e57baf3a38e25ee47b1495edeb34bc7a713e27d987e144bf
0
Deleted: sha256:9565295ea09023547d1ef0b85e0eea259cd2e043ec788629125d1999e002057
```

- Set gopath to your go installation

```
root@Ubuntu: /home/vboxuser
root@Ubuntu:/home/vboxuser# export GOPATH=$HOME/go
root@Ubuntu:/home/vboxuser#
```

```
root@Ubuntu: ~/go
root@Ubuntu:/home/vboxuser# export GOPATH=$HOME/go
root@Ubuntu:/home/vboxuser# pwd
/home/vboxuser
root@Ubuntu:/home/vboxuser# cd GOPATH
bash: cd: GOPATH: No such file or directory
root@Ubuntu:/home/vboxuser# cd $GOPATH
root@Ubuntu:~/go# pwd
/root/go
```

- Clone the fabric-chaincode-evm repo in your go path directory

```
root@Ubuntu: ~/go
root@Ubuntu:~/go# mkdir src
root@Ubuntu:~/go# ls
src
root@Ubuntu:~/go#

root@Ubuntu:~/go# mkdir src
root@Ubuntu:~/go# ls
src
root@Ubuntu:~/go# cd src
root@Ubuntu:~/go/src#
```

```
root@Ubuntu:~/go# cd src
root@Ubuntu:~/go/src# mkdir github.com
root@Ubuntu:~/go/src# ls
github.com
```

```
root@Ubuntu:~/go/src# ls
github.com
root@Ubuntu:~/go/src# cd github.com/
root@Ubuntu:~/go/src/github.com# 
```

```
root@Ubuntu:~/go/src# cd github.com/
root@Ubuntu:~/go/src/github.com# mkdir hyperledger
root@Ubuntu:~/go/src/github.com# ls
hyperledger
root@Ubuntu:~/go/src/github.com# mkdir hyperledger
root@Ubuntu:~/go/src/github.com# ls
hyperledger
root@Ubuntu:~/go/src/github.com# cd hyperledger/
root@Ubuntu:~/go/src/github.com/hyperledger# ls
```

```
root@Ubuntu:~/go/src/github.com# cd hyperledger/
root@Ubuntu:~/go/src/github.com/hyperledger# ls
root@Ubuntu:~/go/src/github.com/hyperledger# cd $GOPATH/src/github.com/hyperled
ger/
root@Ubuntu:~/go/src/github.com/hyperledger# 

root@Ubuntu:~/go/src/github.com/hyperledger# cd $GOPATH/src/github.com/hyperled
ger/
root@Ubuntu:~/go/src/github.com/hyperledger# git clone https://github.com/hyper
ledger/fabric-samples.git
Cloning into 'fabric-samples'...
remote: Enumerating objects: 11694, done.
remote: Counting objects: 100% (131/131), done.
remote: Compressing objects: 100% (94/94), done.
remote: Total 11694 (delta 51), reused 83 (delta 29), pack-reused 11563
Receiving objects: 100% (11694/11694), 22.43 MiB | 4.93 MiB/s, done.
Resolving deltas: 100% (6235/6235), done.
root@Ubuntu:~/go/src/github.com/hyperledger# 
```

- *Checkout release-0.1*

```
root@Ubuntu:~/go/src/github.com/hyperledger# ls
fabric-samples
root@Ubuntu:~/go/src/github.com/hyperledger# cd fabric-samples
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples# 

root@Ubuntu:~/go/src/github.com/hyperledger# cd fabric-samples
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples# git checkout releas
e-1.4
branch 'release-1.4' set up to track 'origin/release-1.4'.
Switched to a new branch 'release-1.4'
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples# 
```

- *Now navigate back to your fabric-samples folder. Here we will use first-network to launch the network.*

```
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-chaincode-evm# cd $GOPATH/sr
c/github.com/hyperledger/fabric-samples/first-network
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/first-network# 
```

- *Update the docker-compose-cli.yaml with the volumes to include the fabric-chaincode-evm.*

```
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/first-network# ls
base                        docker-compose-cli.yaml
byfn.sh                     docker-compose-couch-org3.yaml
ccp-generate.sh             docker-compose-couch.yaml
ccp-template.json           docker-compose-e2e-template.yaml
ccp-template.yaml           docker-compose-etcdraft2.yaml
channel-artifacts           docker-compose-kafka.yaml
configtx.yaml               docker-compose-org3.yaml
connection-org3.json        eyfn.sh
connection-org3.yaml        org3-artifacts
crypto-config.yaml          README.md
docker-compose-ca.yaml      scripts
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/first-network# chmod
 +x docker-compose-cli.yaml
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/first-network#
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/first-network# chmod
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/first-network# gedit
 +x docker-compose-cli.yaml
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/first-network#
docker-compose-cli.yaml
```

Open ▾ | *docker-compose-cli.yaml | ~/go/src/github.com/hyperledger/fabric-samples/first-... | Save

```
76      - CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric/
  peer/crypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/
  tls/server.key
77      - CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/
  fabric/peer/crypto/peerOrganizations/org1.example.com/peers/
  peer0.org1.example.com/tls/ca.crt
78      - CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/
  fabric/peer/crypto/peerOrganizations/org1.example.com/users/
  Admin@org1.example.com/msp
79    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
80    command: /bin/bash
81    volumes:
82        - /var/run/:/host/var/run/
83        - ./../chaincode/:/opt/gopath/src/github.com/chaincode
84        - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/
  crypto/
85        - ./scripts:/opt/gopath/src/github.com/hyperledger/fabric/peer/
  scripts/
86        - ./channel-artifacts:/opt/gopath/src/github.com/hyperledger/fabric/
  peer/channel-artifacts
87        - ./../../fabric-chaincode-evm:/opt/gopath/src/github.com/
  hyperledger/fabric-chaincode-evm
88    depends_on:
89      - orderer.example.com
90      - peer0.org1.example.com
91      - peer1.org1.example.com
92      - peer0.org2.example.com
93      - peer1.org2.example.com
```

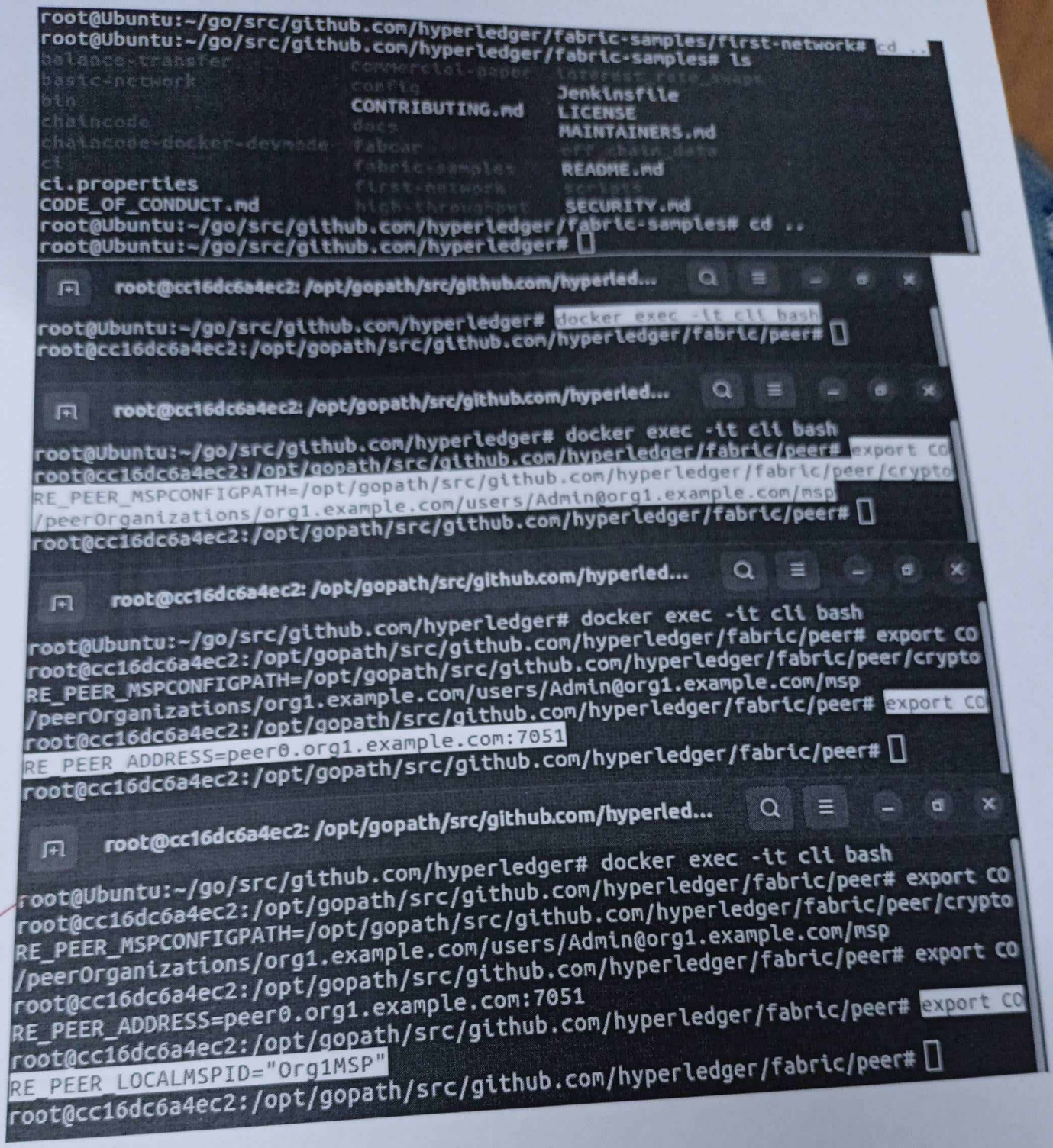
YAML ▾ | Tab Width: 8 ▾ | Ln 87, Col 99 | INS

- Generate certificates and bring up the network

```
root@Ubuntu: ~/go/src/github.com/hyperledger/fabric-sam...
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/first-network# cd ..
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples# cd bin/
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/bin#
```

```
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/bin# cp cryptogen ..
/first-network/
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/bin#
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/bin# cp cryptogen ..
/first-network/
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/bin# cd ..
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples# cd first-network/
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/bin# cp cryptogen ..
/first-network/
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/bin# cd ..
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples# cd first-network/
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/first-network# ./byf
n.sh generate
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/bin# cd ..
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples# cd first-network/
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/first-network# ./byf
n.sh generate
Generating certs and genesis block for channel 'mychannel' with CLI timeout of
'10' seconds and CLI delay of '3' seconds
Continue? [Y/n] Y
proceeding ...
/root/go/src/github.com/hyperledger/fabric-samples/first-network/../bin/cryptog
en

##########################################################
##### Generate certificates using cryptogen tool #########
##########################################################
+ cryptogen generate --config=./crypto-config.yaml
org1.example.com
org2.example.com
+ res=0
+ set +x

root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/first-network# ./byf
n.sh up
Starting for channel 'mychannel' with CLI timeout of '10' seconds and CLI delay
 of '3' seconds
Continue? [Y/n] Y
proceeding ...
LOCAL_VERSION=1.4.4
DOCKER_IMAGE_VERSION=1.4.4
Creating network "net_byfn" with the default driver
Creating volume "net_orderer.example.com" with default driver
Creating volume "net_peer0.org1.example.com" with default driver
Creating volume "net_peer1.org1.example.com" with default driver
Creating volume "net_peer0.org2.example.com" with default driver
Creating volume "net_peer1.org2.example.com" with default driver
Creating orderer.example.com     ... done
Creating peer1.org2.example.com ... done
Creating peer0.org2.example.com ... done
Creating peer1.org1.example.com ... done
Creating peer0.org1.example.com ... done
Creating cli                     ... done
CONTAINER ID   IMAGE                                  COMMAND                  CREATED
```

- Navigate into the cli docker container


root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples/first-network# cd ..
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples# ls
balance-transfer
basic-network
bin
chaincode
chaincode-docker-devmode
ci
ci.properties
CODE_OF_CONDUCT.md
commercial-paper
config
CONTRIBUTING.md
docs
fabcar
fabric-samples
first-network
high-throughput
Jenkinsfile
LICENSE
MAINTAINERS.md
README.md
SECURITY.md
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-samples# cd ..
root@Ubuntu:~/go/src/github.com/hyperledger#
root@cc16dc6a4ec2: /opt/gopath/src/github.com/hyperled...
root@Ubuntu:~/go/src/github.com/hyperledger# docker exec -it cli bash
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer#
root@cc16dc6a4ec2: /opt/gopath/src/github.com/hyperled...
root@Ubuntu:~/go/src/github.com/hyperledger# docker exec -it cli bash
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# export CO
RE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer#
root@cc16dc6a4ec2: /opt/gopath/src/github.com/hyperled...
root@Ubuntu:~/go/src/github.com/hyperledger# docker exec -it cli bash
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# export CO
RE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# export CO
RE_PEER_ADDRESS=peer0.org1.example.com:7051
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer#
root@cc16dc6a4ec2: /opt/gopath/src/github.com/hyperled...
root@Ubuntu:~/go/src/github.com/hyperledger# docker exec -it cli bash
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# export CO
RE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# export CO
RE_PEER_ADDRESS=peer0.org1.example.com:7051
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# export CO
RE_PEER_LOCALMSPID="Org1MSP"
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer#

```
root@cc16dc6a4ec2: /opt/gopath/src/github.com/hyperled...
root@Ubuntu:~/go/src/github.com/hyperledger# docker exec -it cli bash
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# export co
RE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# export co
RE_PEER_ADDRESS=peer0.org1.example.com:7051
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# export co
RE_PEER_LOCALMSPID="Org1MSP"
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# export co
RE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/cr
ypto/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

- Next install the EVM chaincode on all the peers

```
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chai
ncode install -n evmcc -l golang -v 0 -p github.com/hyperledger/fabric-chaincod
e-evm/evmcc
2023-01-02 10:56:22.323 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001
Using default escc
2023-01-02 10:56:22.324 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002
Using default vscc
2023-01-02 10:56:34.255 UTC [chaincodeCmd] install -> INFO 003 Installed remote
ly response:<status:200 payload:"OK" >
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

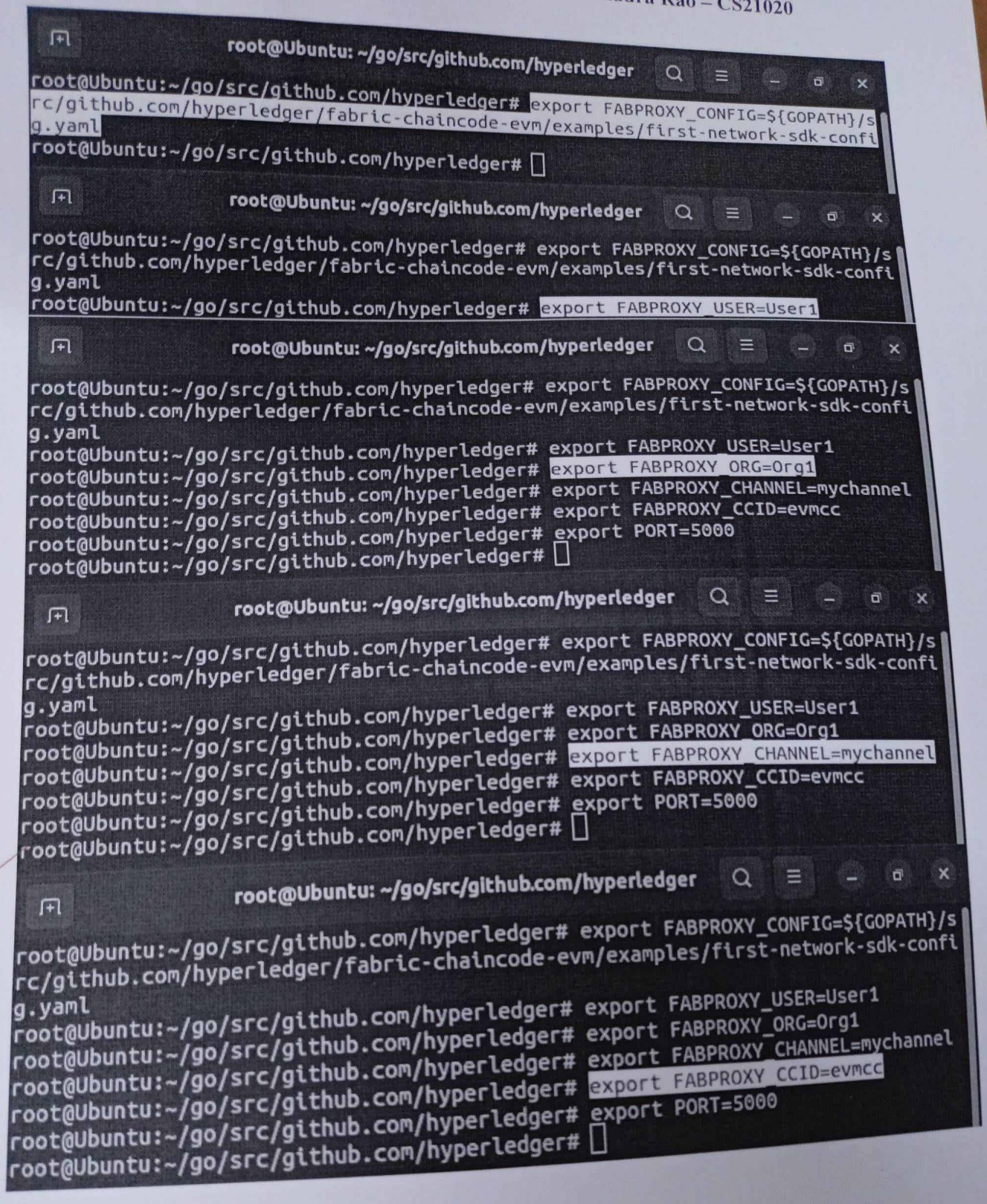
- Instantiate the chaincode.

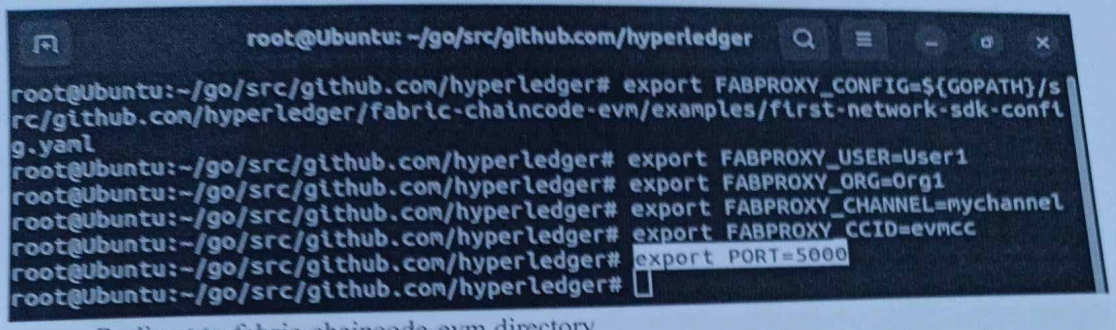
```
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chai
ncode instantiate -n evmcc -v 0 -C mychannel -c '{"Args":[]}' -o orderer.exampl
e.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/cr
ypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacer
ts/tlsca.example.com-cert.pem
2023-01-02 10:57:02.120 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001
Using default escc
2023-01-02 10:57:02.121 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002
Using default vscc
```

- You can exit out of the cli container and return to your terminal.

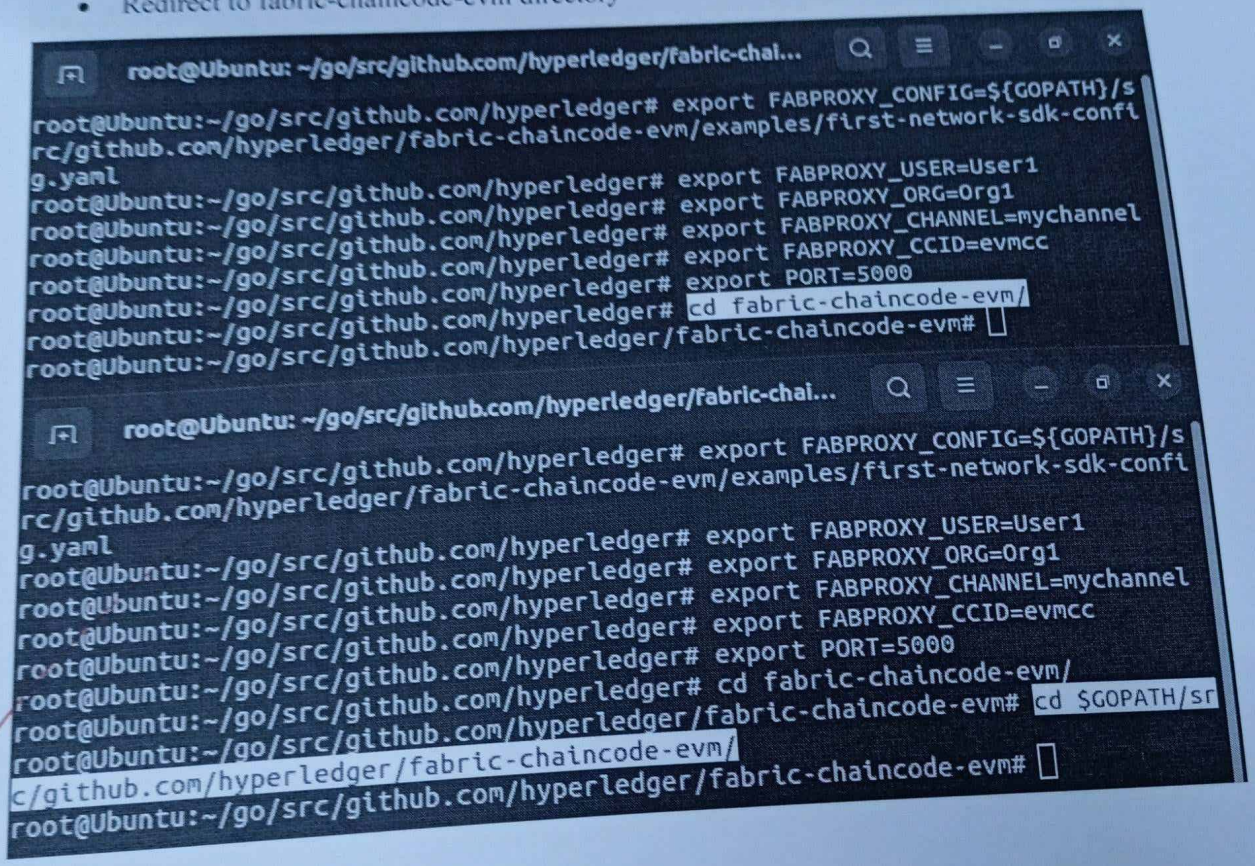
```
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chai
ncode instantiate -n evmcc -v 0 -C mychannel -c '{"Args":[]}' -o orderer.exampl
e.com:7050 --tls --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/cr
ypto/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacer
ts/tlsca.example.com-cert.pem
2023-01-02 10:57:02.120 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001
Using default escc
2023-01-02 10:57:02.121 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002
Using default vscc
root@cc16dc6a4ec2:/opt/gopath/src/github.com/hyperledger/fabric/peer# exit
```

- Execute the following to set certain environment variables required for setting up Fab3


root@Ubuntu: ~/go/src/github.com/hyperledger
root@Ubuntu:~/go/src/github.com/hyperledger# export FABPROXY_CONFIG=${GOPATH}/src/github.com/hyperledger/fabric-chaincode-evm/examples/first-network-sdk-config.yaml
root@Ubuntu:~/go/src/github.com/hyperledger# export FABPROXY_USER=User1
root@Ubuntu:~/go/src/github.com/hyperledger# export FABPROXY_ORG=Org1
root@Ubuntu:~/go/src/github.com/hyperledger# export FABPROXY_CHANNEL=mychannel
root@Ubuntu:~/go/src/github.com/hyperledger# export FABPROXY_CCID=evmcc
root@Ubuntu:~/go/src/github.com/hyperledger# export PORT=5000

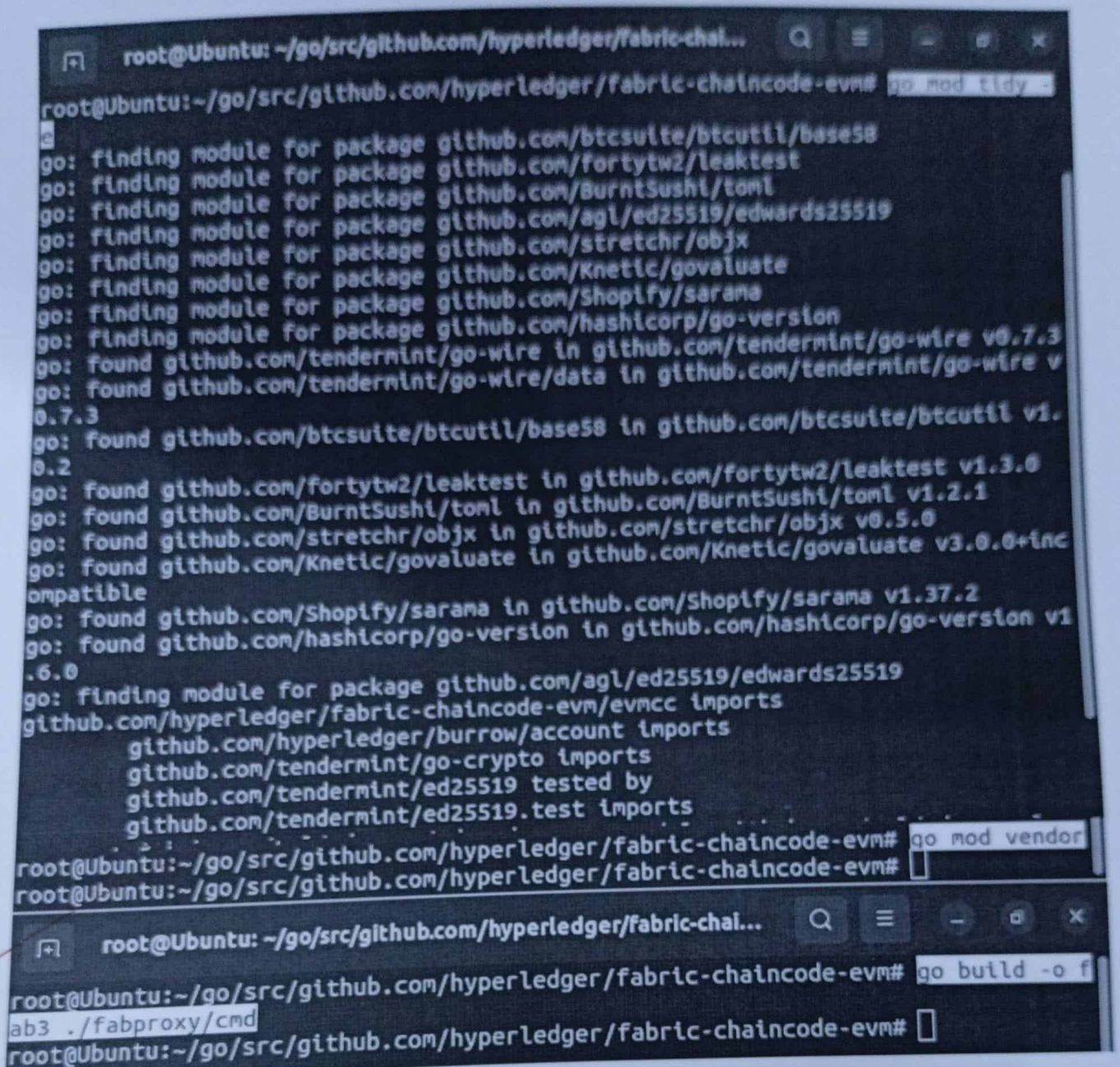


- Redirect to fabric-chaincode-evm directory



- Run the following to build the fab proxy

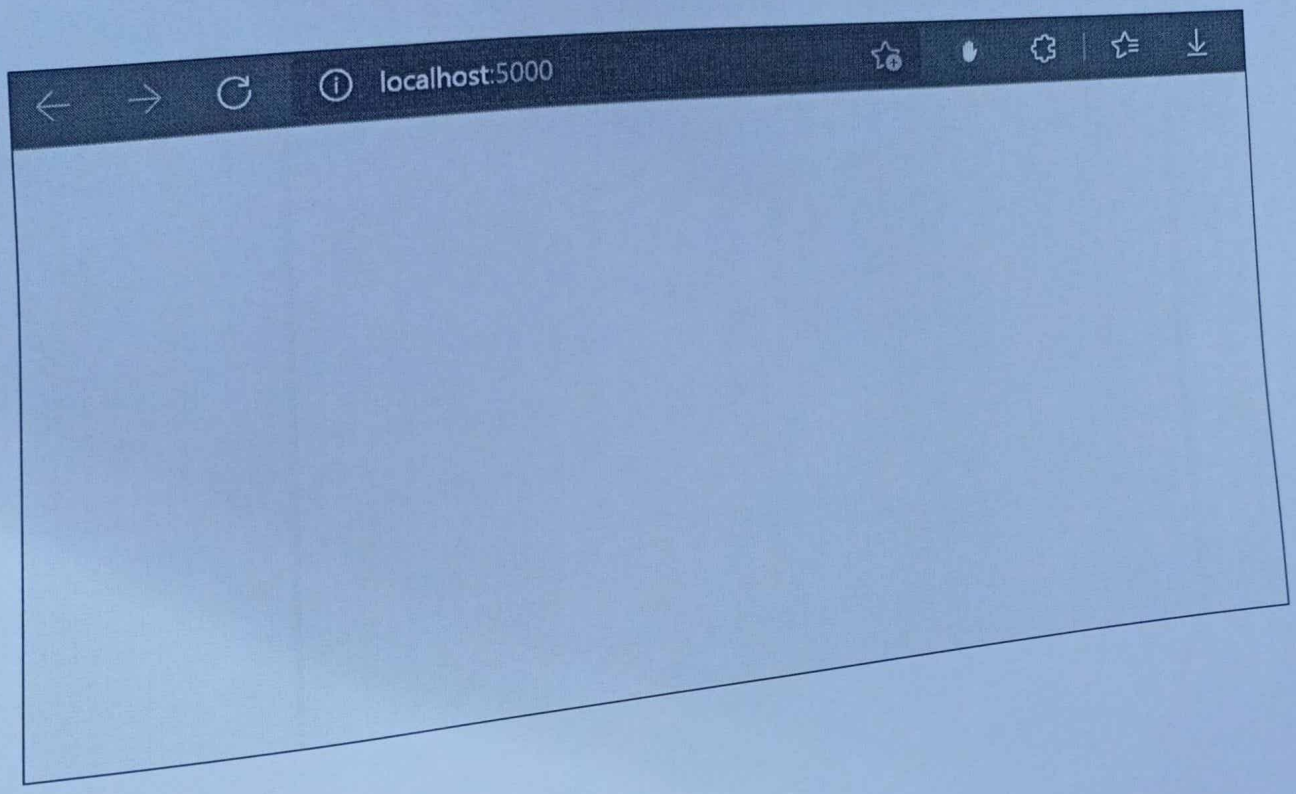
```
root@Ubuntu: ~/go/src/github.com/hyperledger/fabric-chai...
root@Ubuntu:~/go/src/github.com/hyperledger/fabric-chaincode-evm# go mod init
go: creating new go.mod: module github.com/hyperledger/fabric-chaincode-evm
go: downloading github.com/Azure/go-ansiterm v0.0.0-20170929234023-d6e3b3328b78
go: downloading github.com/Microsoft/go-winio v0.4.8
go: downloading github.com/Nvveen/Gotty v0.0.0-20120604004816-cd527374f1e5
go: downloading github.com/beorn7/perks v0.0.0-20180321164747-3a771d992973
go: downloading github.com/btcsuite/btcd v0.0.0-20180706232521-fdfc19097e7a
go: downloading github.com/cactus/go-statsd-client v3.1.1+incompatible
go: downloading github.com/cloudflare/cfssl v0.0.0-20180323000720-5d63dbd981b5
go: downloading github.com/containerd/continuity v0.0.0-20180612233548-246e4905
0efd
go: downloading github.com/davecgh/go-spew v1.1.0
go: downloading github.com/docker/docker v17.12.0-ce-rc1.0.20180827131323-0c5f8
d2b9b23+incompatible
go: downloading github.com/docker/go-connections v0.3.0
go: downloading github.com/docker/go-units v0.3.3
go: downloading github.com/docker/libnetwork v0.8.0-dev.2.0.20180608203834-1927
9f049241
go: downloading github.com/facebookgo/clock v0.0.0-20150410010913-600d898af40a
go: downloading github.com/fsnotify/fsnotify v1.4.7
go: downloading github.com/fsouza/go-dockerclient v1.3.0
go: downloading github.com/go-kit/kit v0.6.0
go: downloading github.com/go-logfmt/logfmt v0.3.0
go: downloading github.com/go-stack/stack v1.7.0
go: downloading github.com/gogo/protobuf v1.0.0
go: downloading github.com/golang/mock v1.1.2-0.20180503014854-22bbf0ddf081
go: downloading github.com/golang/protobuf v1.1.0
go: downloading github.com/google/certificate-transparency-go v1.0.10-0.2018022
2191210-5ab67e519c93
```



- You can then run the proxy

./fab3
This will start Fab3 at http://localhost:5001

Rudra Rao – CS21020



10/10/1