PRACTICAL 1

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

Background Information:

Docker is an open-source containerization platform used for developing, deploying, and managing applications in lightweight virtualized environments called containers. It is mainly used as a software development platform for developing distributed applications that work efficiently in different environments. By making the software system agnostic, developers don't have to worry about compatibility issues. Packaging apps into isolated environments (containers) also makes it easier to develop, deploy, maintain, and use applications.

Installation steps:

1. Download Docker: Go to the Docker website and download the appropriate Docker installer for your operating system. Or download docker from here, https://www.docker.com/products/docker-desktop/
2. Install Docker: Follow the on-screen instructions to install Docker
3. Verify installation: Open a terminal window and run the following command:

Command 1: #docker version

```
C:\Users\Pratik> docker version
Client:
 Cloud integration: v1.0.35+desktop.5
 Version:           24.0.6
 API version:       1.43
 Go version:        go1.20.7
 Git commit:        ed223bc
 Built:             Mon Sep  4 12:32:48 2023
 OS/Arch:           windows/amd64
 Context:           default

Server: Docker Desktop 4.25.0 (126437)
 Engine:
  Version:          24.0.6
  API version:      1.43 (minimum version 1.12)
  Go version:       go1.20.7
  Git commit:       1a79695
  Built:            Mon Sep  4 12:32:16 2023
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.22
  GitCommit:        8165feabfdfe38c65b599c4993d227328c231fca
 runc:
  Version:          1.1.8
  GitCommit:        v1.1.8-0-g82f18fe
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0

C:\Users\Pratik>
```

4. Start the Docker daemon: The Docker daemon is the service that runs Docker containers.
5. Test Docker: To test Docker, run the following command:

Command 2: # docker run hello-world

```
C:\Users\Pratik> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:88ec0acaa3ec199d3b7eaf73588f4518c25f9d34f58ce9a0df68429c5af48e8d
Status: Downloaded newer image for hello-world:latest

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

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

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

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

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

6. Pull a UBUNTU image from docker hub and run the CentOS container. Make sure the Docker daemon is running.
7. Pull a UBUNTU image from docker hub and run

Command 3:

#docker pull ubuntu

#docker run -it ubuntu

8. Now Install and understand Hyperledger Fabric and Ethereum.

Command 4:

apt-get update && apt-get install -y \

soware-properes-common \

curl \

wget \

unzip \

git \

apt-uls \

build-essenal \

gcc \

g++ \

make

curl -sL hps://deb.nodesource.com/setup_20.x | bash -

apt-get install -y nodejs

apt-get install -y openjdk-11-jdk

add-apt-repository ppa:ethereum/Ethereum

apt-get update -y

apt-get install -y solc

wget hps://go.dev/dl/go1.21.3.linux-amd64.tar.gz

tar -C /usr/local -xzf go1.21.3.linux-amd64.tar.gz

rm go1.21.3.linux-amd64.tar.gz

apt-get install -y \

docker.io \

libtool \

libltdl-dev \

libssl-dev \

autoconf \

automake \

bison \

libboost-all-dev \

libgflags-dev \

libprotobuf-dev \

libleveldb-dev \

libsnappy-dev \

libsodium-dev \

liblz4-tool

service --status-all

PRACTICAL 2

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

Background Informaon:

Hyperledger Fabric:

Hyperledger Fabric is a permissioned blockchain plaorm that offers a modular architecture, allowing components such as consensus and membership services to be plug-and-play. It is designed for use in enterprise contexts and supports smart contracts wrien in various programming languages.

SDK for Java:

The Hyperledger Fabric SDK for Java enables Java applicaons to interact with Hyperledger Fabric networks. It provides a set of APIs for creang, deploying, and interacng with smart contracts, as well as managing idenes and transacons.

Command 1:

Docker –version

Command 2:

docker –version

Command 3:

docker-compose –version

Command 4:

export FABRIC_HOME=/path/to/fabric-sdk-java

Command 5:

export JAVA_HOME=/path/to/jdk11.0.17

Command 6:

git clone hps://github.com/hyperledger/fabric-samples.git

Command 7:

mkdir pract2

cd pract2

ls

Command 8:

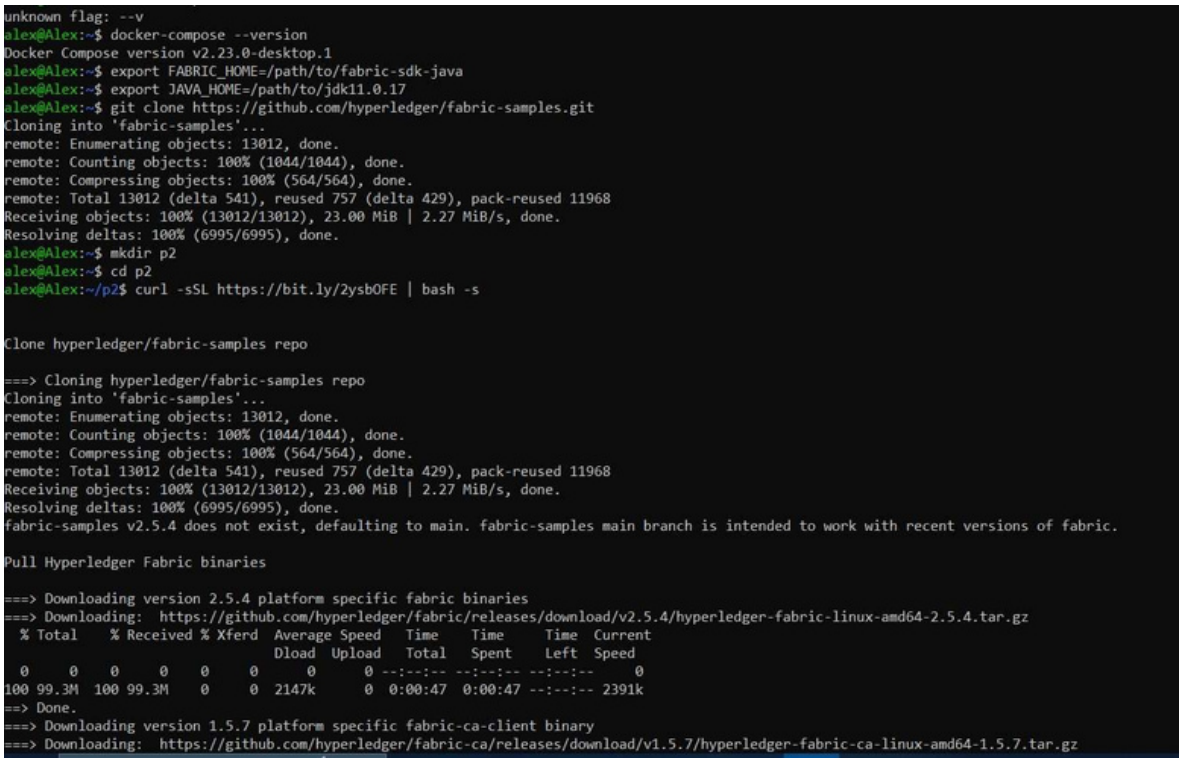
curl -sSL hps://bit.ly/2ysbOFE | bash -s

Command 9:

cd fabric-samples/test-network

Command 10:

./network.sh up

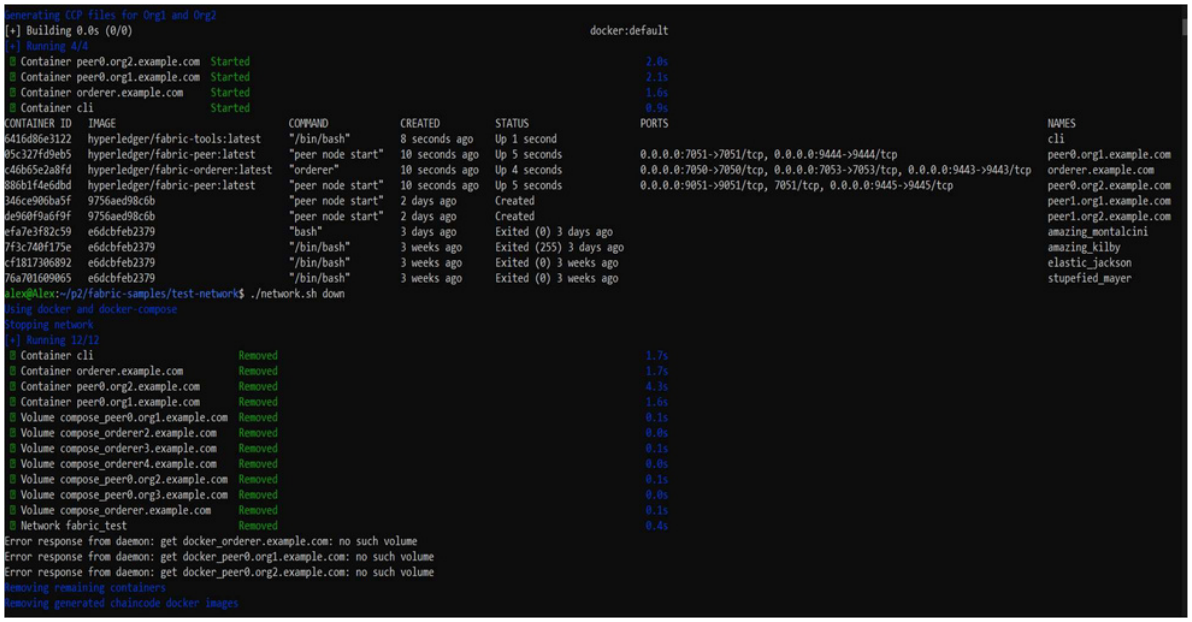


```
Pull Hyperledger Fabric docker images

FABRIC_IMAGES: peer orderer ccenv tools baseos
===> Pulling fabric Images
====>  docker.io/hyperledger/fabric-peer:2.5.4
2.5.4: Pulling from hyperledger/fabric-peer
01085d60b3a6: Pull complete
5920bb1ab585: Pull complete
0c13247db338: Pull complete
f1ebf2febfff: Pull complete
7ba246813ff2: Pull complete
20e090879749: Pull complete
Digest: sha256:c2e735a3cb8250c47ed3589294b3f0c078004ec001b949710f334736651e106d
Status: Downloaded newer image for hyperledger/fabric-peer:2.5.4
docker.io/hyperledger/fabric-peer:2.5.4

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview docker.io/hyperledger/fabric-peer:2.5.4
====>  docker.io/hyperledger/fabric-orderer:2.5.4
2.5.4: Pulling from hyperledger/fabric-orderer
01085d60b3a6: Already exists
0ec39628d119: Pull complete
c9ef912f2449: Pull complete
5f267c8c968e: Pull complete
54a738441be6: Pull complete
d6c2bf9dde2f: Pull complete
Digest: sha256:1a7144705b435062f4be2886de98d0408165cf51326ec721c3f58b40bf8bf139
Status: Downloaded newer image for hyperledger/fabric-orderer:2.5.4
docker.io/hyperledger/fabric-orderer:2.5.4
```

```
hyperledger/fabric-baseos    2.5.4     3854817b987d   4 months ago   122MB
hyperledger/fabric-baseos    latest    3854817b987d   4 months ago   122MB
hyperledger/fabric-tools     1.4       7552e1968c0b   4 years ago    1.49GB
hyperledger/fabric-tools     1.4.4     7552e1968c0b   4 years ago    1.49GB
hyperledger/fabric-ccenv     1.4       ca4780293e4c   4 years ago    1.37GB
hyperledger/fabric-ccenv     1.4.4     ca4780293e4c   4 years ago    1.37GB
hyperledger/fabric-orderer   1.4       dbc9f65443aa   4 years ago    120MB
hyperledger/fabric-orderer   1.4.4     dbc9f65443aa   4 years ago    120MB
hyperledger/fabric-peer      1.4       9756aed98c6b   4 years ago    128MB
hyperledger/fabric-peer      1.4.4     9756aed98c6b   4 years ago    128MB
alex@Alex:~/p2$
alex@Alex:~/p2$ cd fabric-samples/test-network
alex@Alex:~/p2/fabric-samples/test-network$ ./network.sh up
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto from 'cryptogen'
LOCAL_VERSION=v2.5.4
DOCKER_IMAGE_VERSION=v2.5.4
/home/alex/p2/fabric-samples/test-network/../bin/cryptogen
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
```

Command 11:

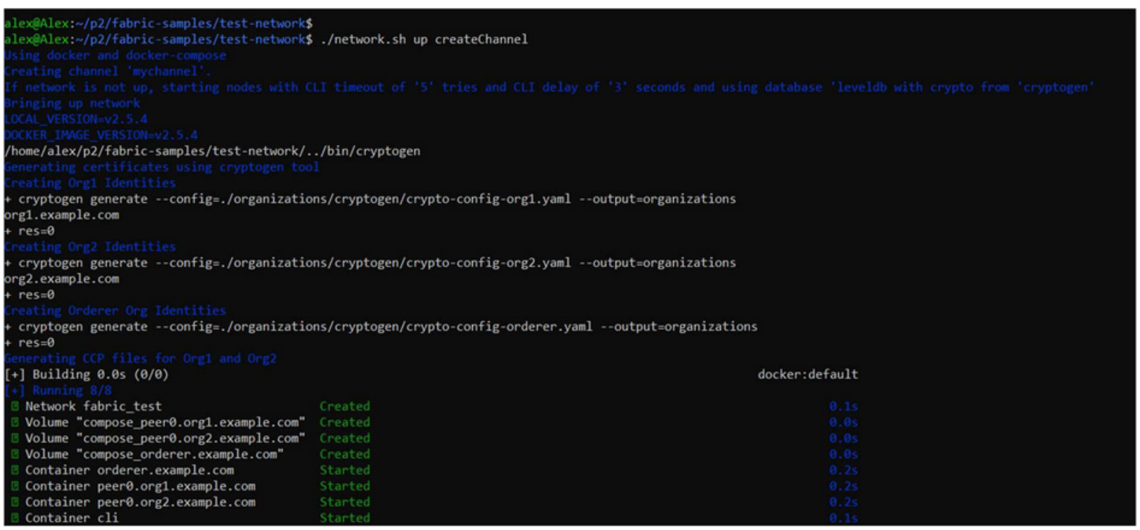
./network.sh down



```
Generating CCP files for Org1 and Org2
[+] Building 0.0s (0/0)                                                    docker:default
[+] Running 4/4
 ✔ Container peer0.org2.example.com  Started                                         2.0s
 ✔ Container peer0.org1.example.com  Started                                         2.1s
 ✔ Container orderer.example.com     Started                                         1.6s
 ✔ Container cli                     Started                                         0.9s
CONTAINER ID   IMAGE                               COMMAND              CREATED          STATUS                     PORTS                                                                     NAMES
6416d86e3122   hyperledger/fabric-tools:latest     "/bin/bash"          8 seconds ago    Up 1 second                                                                                          cli
05c327fd9eb5   hyperledger/fabric-peer:latest      "peer node start"    10 seconds ago   Up 5 seconds               0.0.0.0:7051->7051/tcp, 0.0.0.0:9444->9444/tcp                            peer0.org1.example.com
c46b65e2a8fd   hyperledger/fabric-orderer:latest   "orderer"            10 seconds ago   Up 4 seconds               0.0.0.0:7050->7050/tcp, 0.0.0.0:7053->7053/tcp, 0.0.0.0:9443->9443/tcp   orderer.example.com
886b1f4e6dbd   hyperledger/fabric-peer:latest      "peer node start"    10 seconds ago   Up 5 seconds               0.0.0.0:9051->9051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp                  peer0.org2.example.com
346ce906ba5f   9756aed98c6b                        "peer node start"    2 days ago       Created                                                                                              peer1.org1.example.com
de960f9a6f9f   9756aed98c6b                        "peer node start"    2 days ago       Created                                                                                              peer1.org2.example.com
efa7e3f82c59   e6dcbfeb2379                        "bash"               3 days ago       Exited (0) 3 days ago                                                                                amazing_montalcini
7f3c740f175e   e6dcbfeb2379                        "/bin/bash"          3 weeks ago      Exited (255) 3 days ago                                                                              amazing_kilby
cf1817306892   e6dcbfeb2379                        "/bin/bash"          3 weeks ago      Exited (0) 3 weeks ago                                                                               elastic_jackson
76a701609065   e6dcbfeb2379                        "/bin/bash"          3 weeks ago      Exited (0) 3 weeks ago                                                                               stupefied_mayer
alex@Alex:~/p2/fabric-samples/test-network$ ./network.sh down
Using docker and docker-compose
Stopping network
[+] Running 12/12
 ✔ Container cli                          Removed                                    1.7s
 ✔ Container orderer.example.com          Removed                                    1.7s
 ✔ Container peer0.org2.example.com       Removed                                    4.3s
 ✔ Container peer0.org1.example.com       Removed                                    1.6s
 ✔ Volume compose_peer0.org1.example.com  Removed                                    0.1s
 ✔ Volume compose_orderer2.example.com    Removed                                    0.0s
 ✔ Volume compose_orderer3.example.com    Removed                                    0.1s
 ✔ Volume compose_orderer4.example.com    Removed                                    0.0s
 ✔ Volume compose_peer0.org2.example.com  Removed                                    0.1s
 ✔ Volume compose_peer0.org3.example.com  Removed                                    0.0s
 ✔ Volume compose_orderer.example.com     Removed                                    0.1s
 ✔ Network fabric_test                    Removed                                    0.4s
Error response from daemon: get docker_orderer.example.com: no such volume
Error response from daemon: get docker_peer0.org1.example.com: no such volume
Error response from daemon: get docker_peer0.org2.example.com: no such volume
Removing remaining containers
Removing generated chaincode docker images
```

Command 12:

./network.sh up createChannel



```
alex@Alex:~/p2/fabric-samples/test-network$
alex@Alex:~/p2/fabric-samples/test-network$ ./network.sh up createChannel
Using docker and docker-compose
Creating channel 'mychannel'.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb with crypto from 'cryptogen'
Bringing up network
LOCAL_VERSION=v2.5.4
DOCKER_IMAGE_VERSION=v2.5.4
/home/alex/p2/fabric-samples/test-network/../bin/cryptogen
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Creating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2
[+] Building 0.0s (0/0)                                                    docker:default
[+] Running 8/8
 ✔ Network fabric_test                       Created                         0.1s
 ✔ Volume "compose_peer0.org1.example.com"   Created                         0.0s
 ✔ Volume "compose_peer0.org2.example.com"   Created                         0.0s
 ✔ Volume "compose_orderer.example.com"      Created                         0.0s
 ✔ Container orderer.example.com             Started                         0.2s
 ✔ Container peer0.org1.example.com          Started                         0.2s
 ✔ Container peer0.org2.example.com          Started                         0.2s
 ✔ Container cli                             Started                         0.1s
```

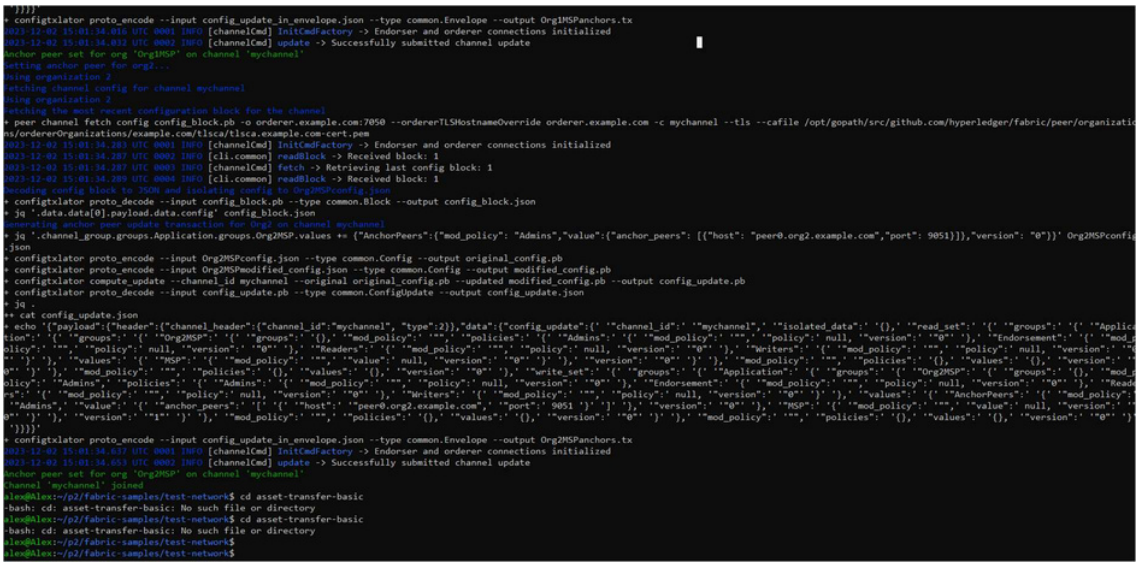
Command 13:

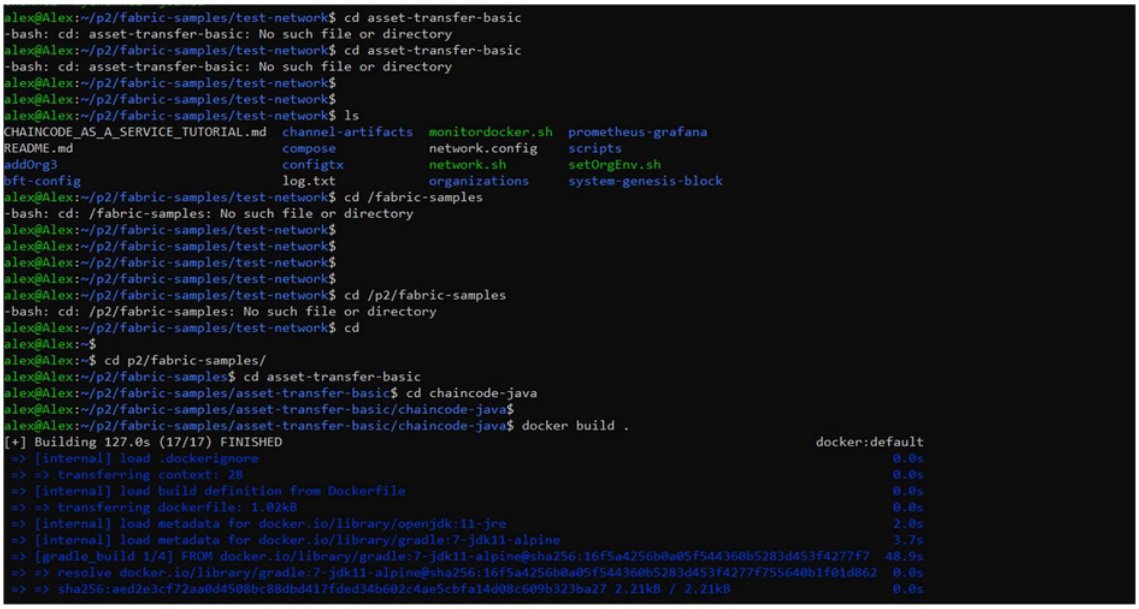
cd asset-transfer-basic

Command 14:

cd chaincode-java

Command 15:





docker build .

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.

Background Information:

Creating a blockchain app involves understanding decentralized ledgers, smart contracts, and user wallets. Use Web3 libraries, test on blockchain testnets, prioritize security, and deploy on the live network after testing. Transactions require user initiation, consensus verification, and may involve fees.

Command 1:

git clone https://github.com/IBM/blockchain-application-using-fabric-java-sdk/

```
root@DESKTOP-CU5U5QE:~#  git clone https://github.com/IBM/blockchain-application-using-fabric-java-sdk
Cloning into 'blockchain-application-using-fabric-java-sdk'...
remote: Enumerating objects: 810, done.
remote: Counting objects: 100% (99/99), done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 810 (delta 86), reused 80 (delta 80), pack-reused 711
Receiving objects: 100% (810/810), 728.73 KiB | 2.38 MiB/s, done.
Resolving deltas: 100% (285/285), done.
```

Command 2:

cd / blockchain-application-using-fabric-java-sdk/

```
root@DESKTOP-CU5U5QE:~# cd blockchain-application-using-fabric-java-sdk
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk# ll
total 68
drwxr-xr-x 7 root root  4096 Nov 30 13:36 ./
drwx------ 8 root root  4096 Nov 30 13:36 ../
drwxr-xr-x 8 root root  4096 Nov 30 13:36 .git/
-rw-r--r-- 1 root root   569 Nov 30 13:36 CONTRIBUTING.md
-rw-r--r-- 1 root root  1549 Nov 30 13:36 DEBUGGING.md
-rw-r--r-- 1 root root 11357 Nov 30 13:36 LICENSE
-rw-r--r-- 1 root root  3131 Nov 30 13:36 MAINTAINERS.md
-rw-r--r-- 1 root root 15582 Nov 30 13:36 README.md
drwxr-xr-x 2 root root  4096 Nov 30 13:36 images/
drwxr-xr-x 3 root root  4096 Nov 30 13:36 java/
drwxr-xr-x 2 root root  4096 Nov 30 13:36 network/
drwxr-xr-x 5 root root  4096 Nov 30 13:36 network_resources/
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk# cd network
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/network# chmod +x build.sh stop.sh teardown.sh
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/network# ./build.sh generate
```

Command 3:

cd network

chmod +x build.sh stop.sh teardown.sh

```
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk# cd network
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/network# chmod +x build.sh stop.sh teardown.sh
```

./build.sh generate – after generating if error show

If below error is shown then delete the container as shown in below ss.

```
 ✓ Network network_custom         Created                 0.0s
 ⠿ Container orderer.example.com  Creating                0.0s
 ⠿ Container ca_peerOrg1          Creating                0.0s
 ⠿ Container ca_peerOrg2          Creating                0.0s
Error response from daemon: Conflict. The container name "/orderer.example.com" is already in use by container "db3d79657ebed2c896ba8
4abdc91b02dd8366ec7b24ad545458693968b863d06". You have to remove (or rename) that container to be able to reuse that name.
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/network#  ./network.sh up
-bash: ./network.sh: No such file or directory
```

Command 4:

./build.sh generate

```
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/network# ./build.sh generate

Stopping the previous network (if any)
[+] Running 3/3
 ✓ Container ca_peerOrg1   Removed
 ✓ Container ca_peerOrg2   Removed
 ✓ Network network_custom  Removed

Setting up the Hyperledger Fabric 1.1 network
[+] Building 0.0s (0/0)
[+] Running 8/8
 ✓ Network network_custom            Created
 ✓ Container orderer.example.com     Started
 ✓ Container ca_peerOrg1             Started
 ✓ Container ca_peerOrg2             Started
 ✓ Container peer0.org2.example.com  Started
 ✓ Container peer0.org1.example.com  Started
 ✓ Container peer1.org2.example.com  Started
 ✓ Container peer1.org1.example.com  Started

Network setup completed!!
```

Command 5:

docker-compose –f docker-compose.yaml up

```
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/network# docker-compose -f docker-compose.yaml up

Usage:  docker compose [OPTIONS] COMMAND

Define and run multi-container applications with Docker.

Options:
      --ansi string                Control when to print ANSI control characters ("never"|"always"|"auto") (default "auto")
      --compatibility              Run compose in backward compatibility mode
      --dry-run                    Execute command in dry run mode
      --env-file stringArray       Specify an alternate environment file.
  -f, --file stringArray           Compose configuration files
      --parallel int               Control max parallelism, -1 for unlimited (default -1)
      --profile stringArray        Specify a profile to enable
      --progress string            Set type of progress output (auto, tty, plain, quiet) (default "auto")
      --project-directory string   Specify an alternate working directory
                                   (default: the path of the, first specified, Compose file)
  -p, --project-name string        Project name

Commands:
```

Command 6:

cd ../java/

```
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/network# cd ../java/
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/java#  java -cp blockchain-clie
```

Command 7:

apt install maven

```
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/java# apt install maven
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libaopalliance-java libapache-pom-java libatinject-jsr330-api-java libcdi-api-java libcommons-cli-java libcommons-io-java
```

Command 8:

mvn install

```
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/java# mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] --------------< blockchain-java-sdk:blockchain-java-sdk >---------------
[INFO] Building blockchain-java-sdk 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
```

Command 9:

cd target/

mv blockchain-java-sdk-0.0.1-SNAPSHOT-jar-with-dependencies.jar blockchain-client.jar

cp blockchain-client.jar ../../network_resources

cd ../../network_resources

java -cp blockchain-client.jar org.example.network.CreateChannel

```
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/java# cd target/
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/java/target#  mv blockchain-java-sdk-0.0.1-SNAPSHOT-jar-with-depe
ndencies.jar blockchain-client.jar
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/java/target# cp blockchain-client.jar ../../network_resources
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/java/target#  cd ../../network_resources
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/network_resources#  java -cp blockchain-client.jar org.example.ne
twork.CreateChannel
log4j:WARN No appenders could be found for logger (org.hyperledger.fabric.sdk.helper.Config).
```

Command 10:

java -cp blockchain-client.jar org.example.user.RegisterEnrollUser

```
root@DESKTOP-CU5U5QE:~/blockchain-application-using-fabric-java-sdk/network_resources# java -cp blockchain-client.jar org.example.use
r.RegisterEnrollUser
Nov 30, 2023 2:24:00 PM org.example.util.Util deleteDirectory
INFO: Deleting - users
log4j:WARN No appenders could be found for logger (org.hyperledger.fabric.sdk.helper.Config).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Nov 30, 2023 2:24:01 PM org.example.client.CAClient enrollAdminUser
INFO: CA -http://localhost:7054 Enrolled Admin.
Nov 30, 2023 2:24:01 PM org.example.client.CAClient registerUser
INFO: CA -http://localhost:7054 Registered User - user1701334441172
Nov 30, 2023 2:24:01 PM org.example.client.CAClient enrollUser
INFO: CA -http://localhost:7054 Enrolled User - user1701334441172
```

PRACTICAL 4

Aim:

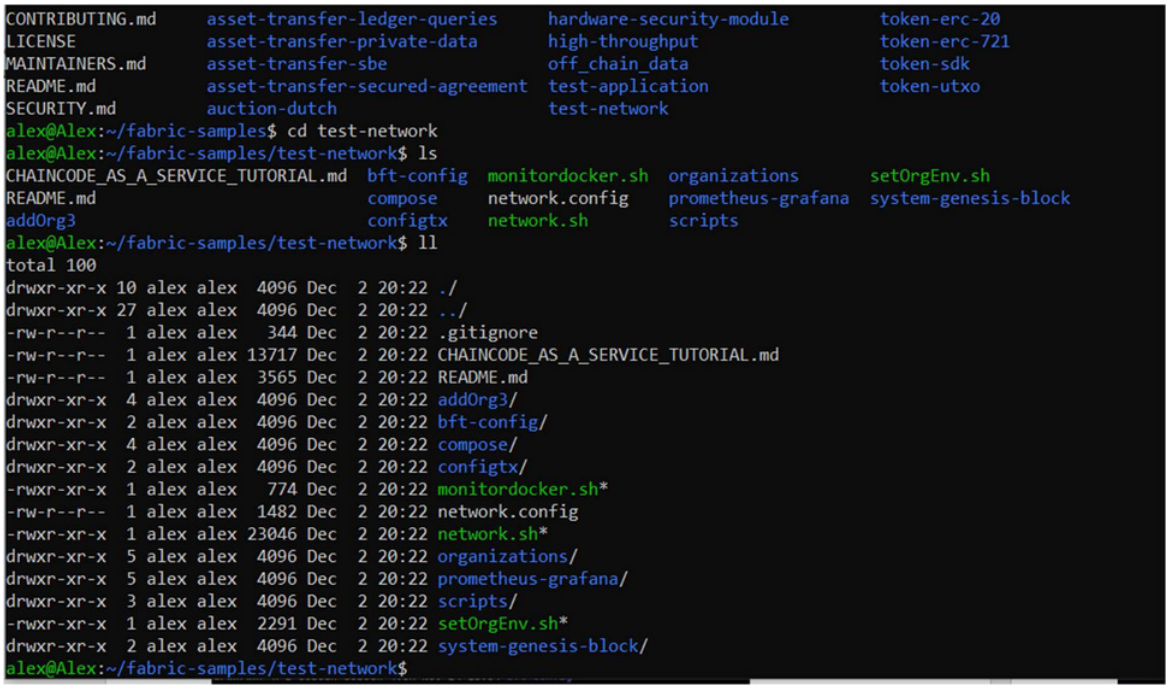
Deploy an asset-transfer app using block chain. Learn app development within a Hyperledger Fabric network.

Background Information:

Hyperledger Fabric is a permissioned blockchain platform for enterprise. It features a modular architecture, smart contracts (chaincode), and supports private channels. Development involves network setup, chaincode coding, user registration, and application development. Security considerations include access control and encryption. Deploy the application after testing on a server or cloud platform.

Command 1:

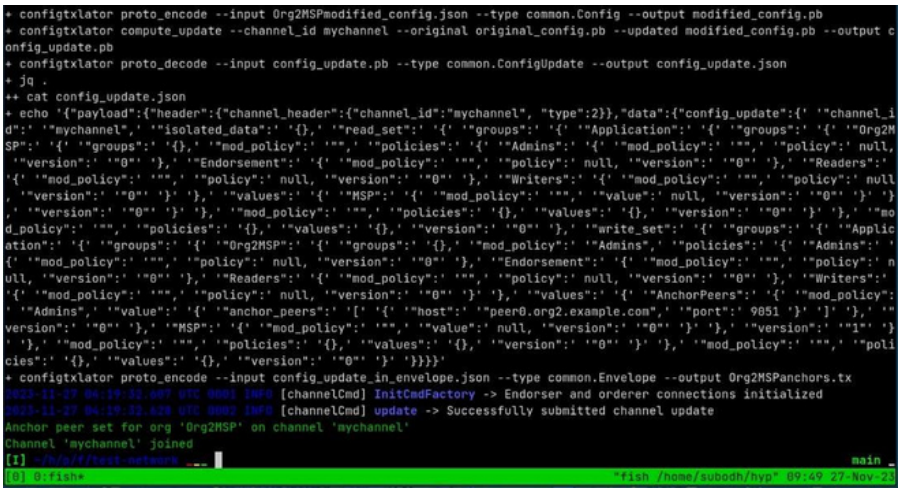
$ cd fabric-samples/test-network



Command 2:

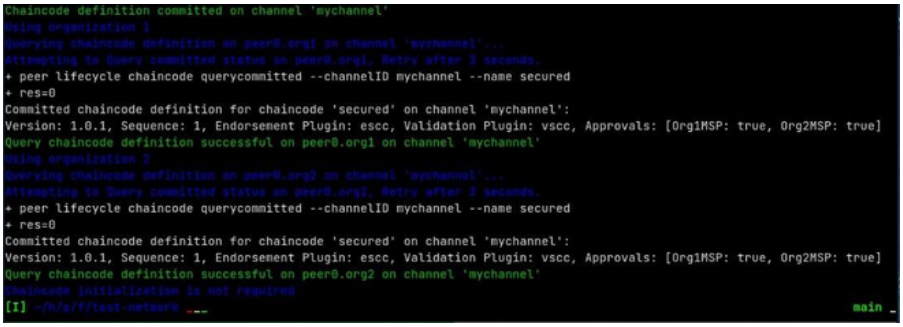
$./network.sh down

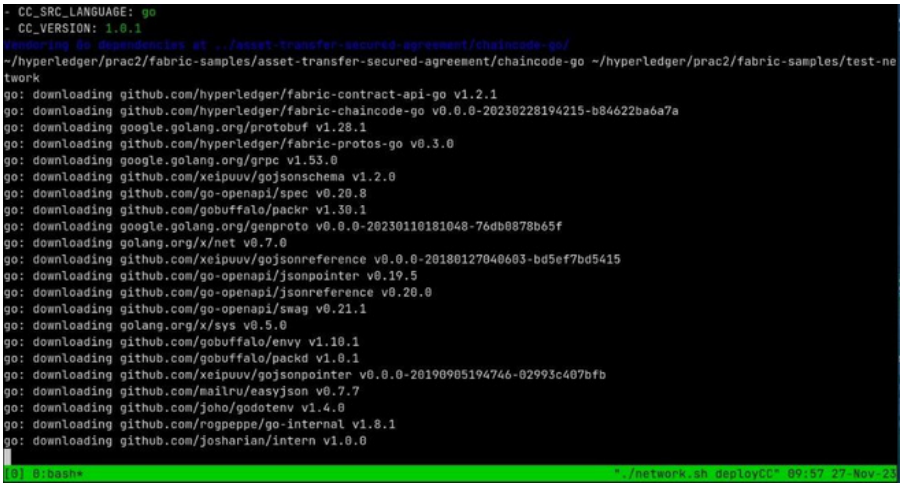
$ ./network.sh up createChannel -c mychannel



Command 3:

$ ./network.sh deployCC -ccn secured -ccp ../asset-transfer-secured-agreement/chaincode-go/ -ccl go -ccep "OR('Org1MSP.peer','Org2MSP.peer')"





Command 4:

Set the environment variables to operate as Org1 (posix compl shell)

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

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

$ export CORE_PEER_TLS_ENABLED=true

$ export CORE_PEER_LOCALMSPID="Org1MSP"

$ export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1
.example.com/users/Admin@org1.example.com/msp

$ export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/o rg1.example.com/peers/peer0.org1.example.com/tls/ca.crt

$ export CORE_PEER_ADDRESS=localhost:7051

Command 5:

Set the environment variables to operate as Org2

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

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

$ export CORE_PEER_TLS_ENABLED=true

$ export CORE_PEER_LOCALMSPID="Org2MSP"

$ export
CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2
.example.com/users/Admin@org2.example.com/msp

$ export
CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/o rg2.example.com/peers/peer0.org2.example.com/tls/ca.crt

$ export CORE_PEER_ADDRESS=localhost:9051

Command 6:

Create an asset

Operate from the Org1 terminal (top Terminal):

$ export ASSET_PROPERTIES=$(echo -n
"{\"object_type\":\"asset_properties\",\"color\":\"blue\",\"size\":35,\"salt\":\"a94a8fe5ccb19ba61c4c0873d391e987982fbbd3\"}" | base64 | tr -d \\n)

```
subodh@subodh-HP-Laptop-15-bs0xx:~/hyperledger/prac2/fabric-samples/test-network$ peer chaincode invoke -o localhost:7050 -
-ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/order
ers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n secured -c '{"function":"CreateAsset","A
rgs":["A new asset for Org1MSP"]}' --transient "{\"asset_properties\":\"$ASSET_PROPERTIES\"}"
2023-11-27 10:36:23.984 IST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:
200 payload:"d9923f21b770adbc79cbcc47a3aeecc81dc7f030bd129155301ce3932be7fbcc"
```

Command 7:

$ export
ASSET_ID=d9923f21b770adbc79cbcc47a3aeecc81dc7f030bd129155301ce393
2 be7fbcc

Success message:

{"object_type":"asset_properties","color":"blue","size":35,"salt":"a94a8fe5ccb
19ba61c4c0873d391e987982fbbd3"}

```
subodh@subodh-HP-Laptop-15-bs0xx:~/hyperledger/prac2/fabric-samples/test-network$ peer chaincode query -o localhost:7050 -
ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/ordere
rs/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n secured -c "{\"function\":\"GetAssetPriva
teProperties\",\"Args\":[\"$ASSET_ID\"]}"
{"object_type":"asset_properties","color":"blue","size":35,"salt":"a94a8fe5ccb19ba61c4c0873d391e987982fbbd3"}
```

Command 8:

$ peer chaincode query -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n secured -c "{\"function\":\"ReadAsset\",\"Args\":[\"$ASSET_ID\"]}"

Success message:

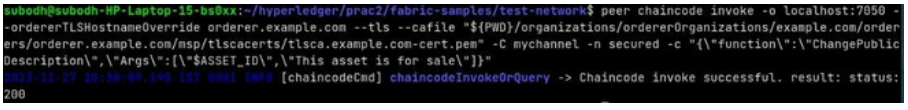
{"objectType":"asset","assetID":"d9923f21b770adbc79cbcc47a3aeecc81dc7f030bd129155301ce3932be7fbcc","ownerOrg":"Org1MSP","publicDescription":" A new asset for Org1MSP"}

Command 9:

$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n secured -c "{\"function\":\"ChangePublicDescription\",\"Args\":[\"$ASSET_ID\",\"This asset is for sale\"]}"

Success message:

2023-11-27 10:38:09.198 IST 0001 INFO [chaincodeCmd]
chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200



Command 10:

$ peer chaincode query -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n secured -c "{\"function\":\"ReadAsset\",\"Args\":[\"$ASSET_ID\"]}"

Success message:

{"objectType":"asset","assetID":"d9923f21b770adbc79cbcc47a3aeecc81dc7f030bd129155301ce3932be7fbcc","ownerOrg":"Org1MSP","publicDescription":"This asset is for sale"}

Command 11:

Operate from the Org2 terminal (bottom Terminal):

$ export ASSET_ID=d9923f21b770adbc79cbcc47a3aeecc81dc7f030bd129155301ce3932 be7fbcc

$ peer chaincode query -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n secured -c "{\"function\":\"ReadAsset\",\"Args\":[\"$ASSET_ID\"]}"

Success message:

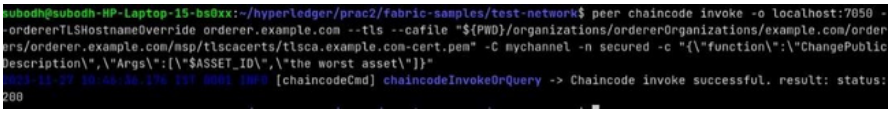
{"objectType":"asset","assetID":"d9923f21b770adbc79cbcc47a3aeecc81dc7f030bd129155301ce3932be7fbcc","ownerOrg":"Org1MSP","publicDescription":"This asset is for sale"}

Command 12:

$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n secured -c "{\"function\":\"ChangePublicDescription\",\"Args\":[\"$ASSET_ID\",\"the worst asset\"]}"

Success message:

2023-11-27 10:46:36.176 IST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200



Command 13:

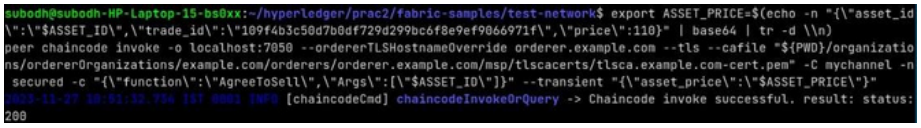
Agree to sell the asset

$ export ASSET_PRICE=$(echo -n "{\"asset_id\":\"$ASSET_ID\",\"trade_id\":\"109f4b3c50d7b0df729d299bc6f8e 9ef9066971f\",\"price\":110}" | base64 | tr -d \\n)

$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.e xample.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n secured -c "{\"function\":\"AgreeToSell\",\"Args\":[\"$ASSET_ID\"]}" --transient "{\"asset_price\":\"$ASSET_PRICE\"}"

Success message:

2023-11-27 10:51:32.734 IST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200



Command 14:

$ peer chaincode query -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.e xample.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n secured -c "{\"function\":\"GetAssetSalesPrice\",\"Args\":[\"$ASSET_ID\"

Success message:

{"asset_id":"d9923f21b770adbc79cbcc47a3aeecc81dc7f030bd129155301ce39
32be7fbcc","trade_id":"109f4b3c50d7b0df729d299bc6f8e9ef9066971f","price ":110}

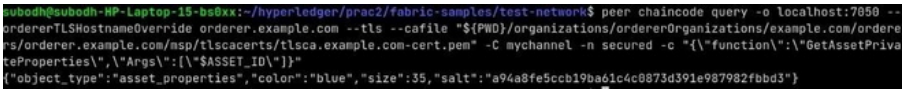
Command 15:

Update the asset description as Org2

$ peer chaincode query -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.e xample.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n secured -c "{\"function\":\"GetAssetPrivateProperties\",\"Args\":[\"$ASSET_ID\"]}"

Success message:

{"object_type":"asset_properties","color":"blue","size":35,"salt":"a94a8fe5ccb 19ba61c4c0873d391e987982fbbd3"}



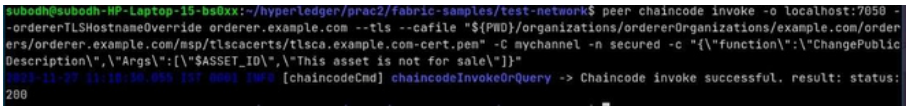
Command 16:

$ peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/order er.e xample.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n secured -c "{\"function\":\"ChangePublicDescription\",\"Args\":[\"$ASSET_ID\",\"This asset is not for sale\"]}"

Success message:

2023-11-27 11:18:30.055 IST 0001 INFO [chaincodeCmd]

chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200

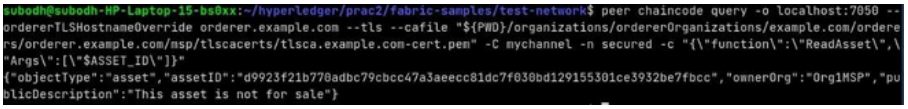


Command 17:

$ peer chaincode query -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile "${PWD}/organizations/ordererOrganizations/example.com/orderers/order er.e xample.com/msp/tlscacerts/tlsca.example.com-cert.pem" -C mychannel -n secured -c "{\"function\":\"ReadAsset\",\"Args\":[\"$ASSET_ID\"]}"

Success message:

{"objectType":"asset","assetID":"d9923f21b770adbc79cbcc47a3aeecc81dc7f0
30bd129155301ce3932be7fbcc","ownerOrg":"Org1MSP","publicDescription":
" This asset is not for sale"}



Command 18:

Clean up (stopping chaincode):

$ ./network.sh down

PRACTICAL 5

Aim: Set up MetaMask, set up Ganache, create two accounts with 100 ETH each, and transfer ETH between the accounts.

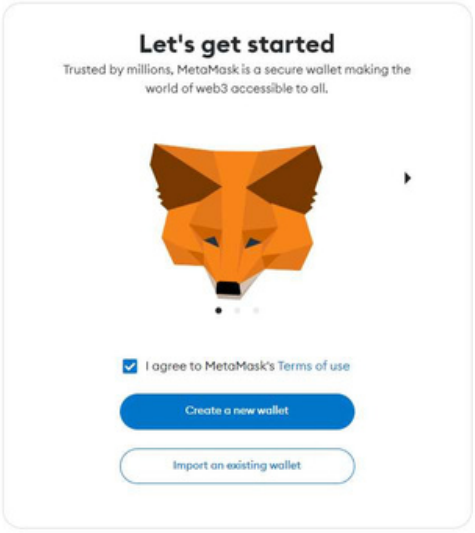
Background Information:

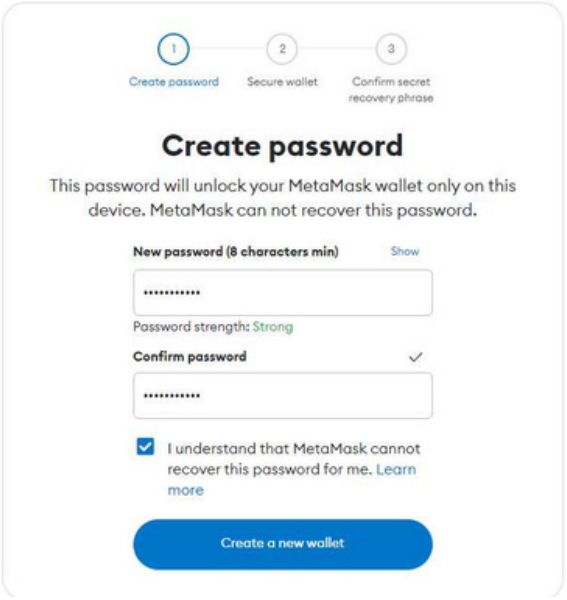
MetaMask is a cryptocurrency wallet and browser extension for Ethereum. It lets users store and manage Ethereum-based assets, interact with decentralized applications (DApps), and provides a secure way to access the Ethereum blockchain.

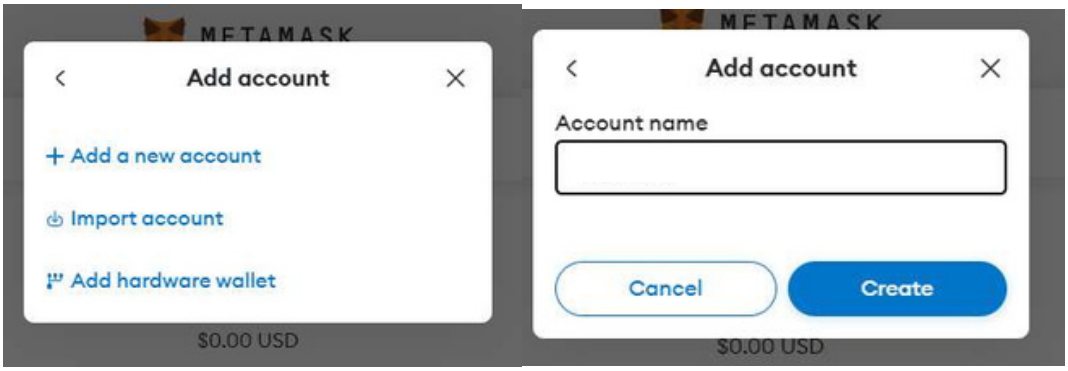
Install MetaMask Extension:

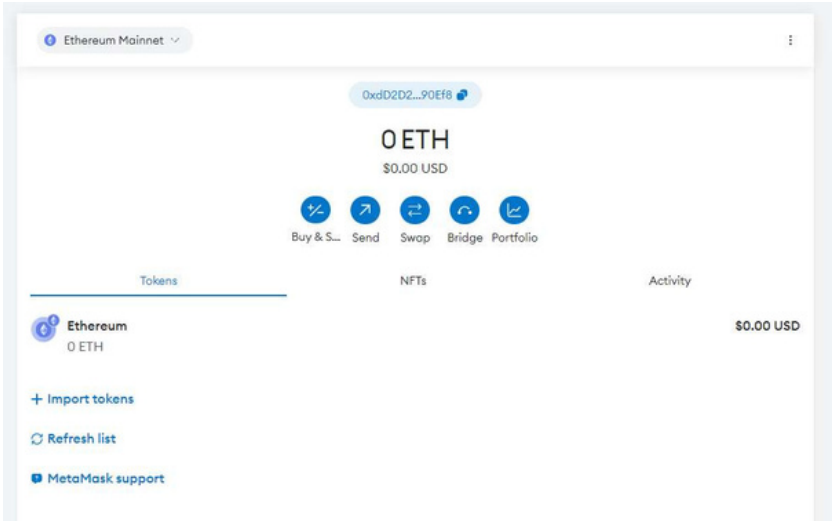
1. Go to the MetaMask website or the appropriate extension store for your browser (e.g., Chrome Web Store, Firefox Add-ons).

2. Add MetaMask to your browser and follow the on-screen instructions to create an account and set up a password. Save the seed phrase generated during the setup process in a secure location.

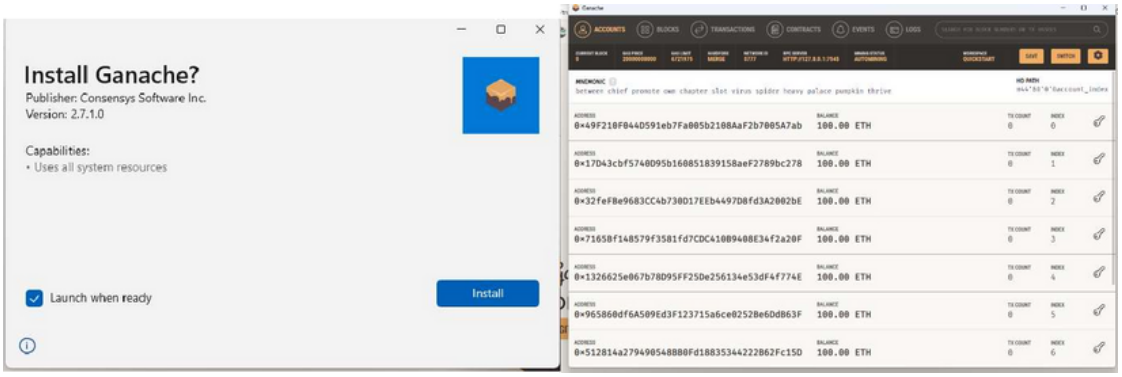








Setting up and Ganache:



Accessing Ganache Accounts:

After Ganache starts, it will display a list of accounts along with their private keys, addresses, and initial balances.

Import Accounts into MetaMask:

1. Open MetaMask in your browser and click on "Import Account."

2. Copy the private key of the first account from Ganache and import it into MetaMask by pasting the key.

3. Repeat the process for the second account.

Transferring ETH Between Accounts:

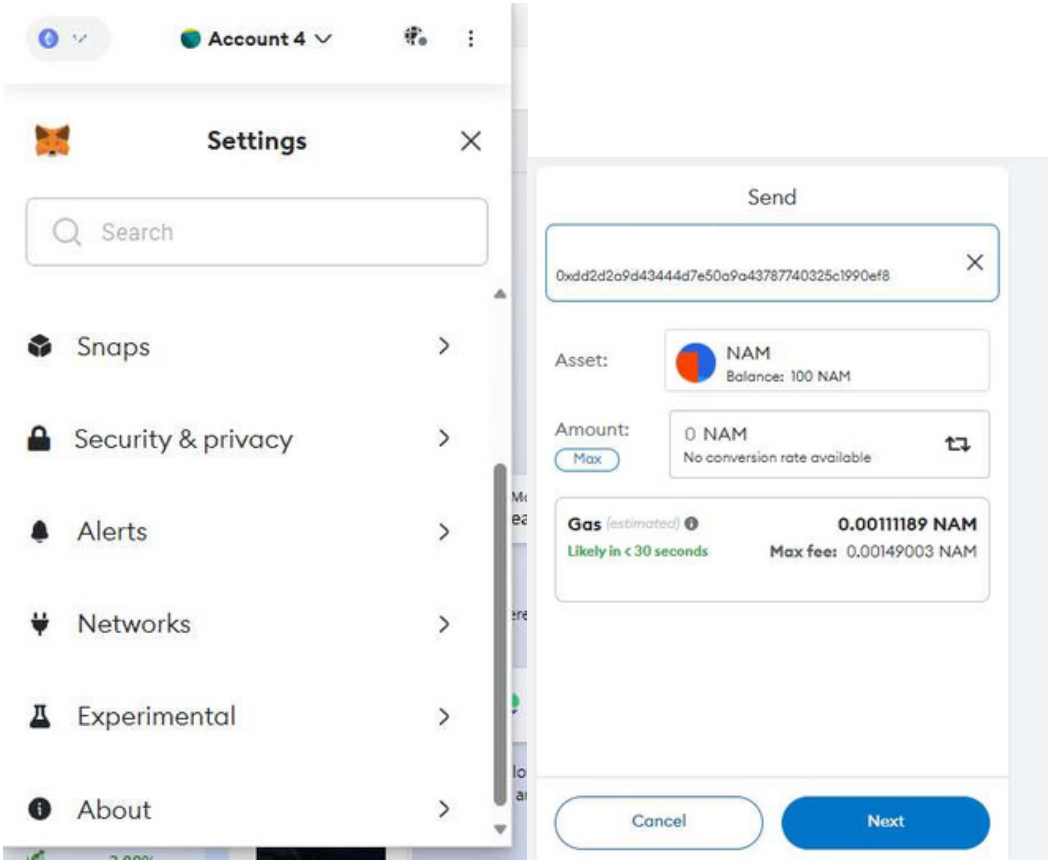
Access Ganache Accounts:

Note:

The addresses of the two accounts you imported into MetaMask from Ganache.

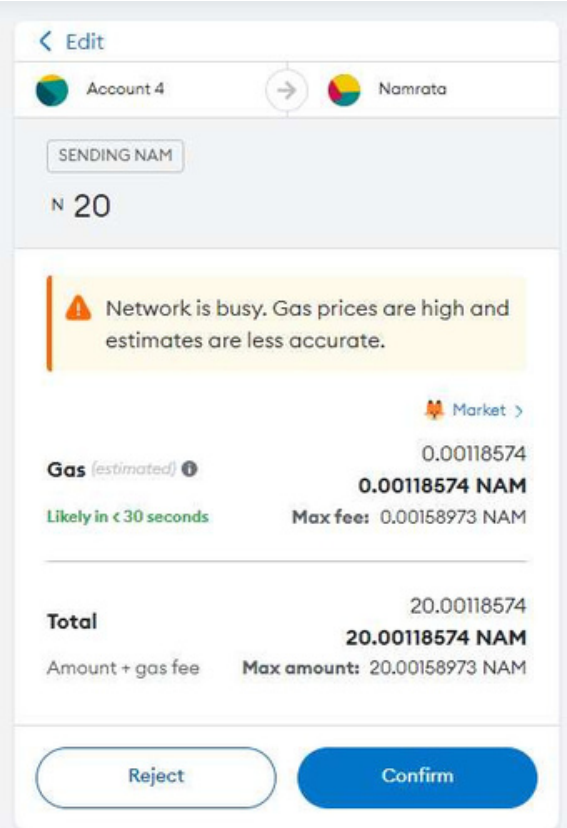
Transfer ETH from Account 1 to Account 2:

1. In MetaMask, select Account 1



Click on "Send" or "Transfer" and enter the address of Account 2 in the recipient field.

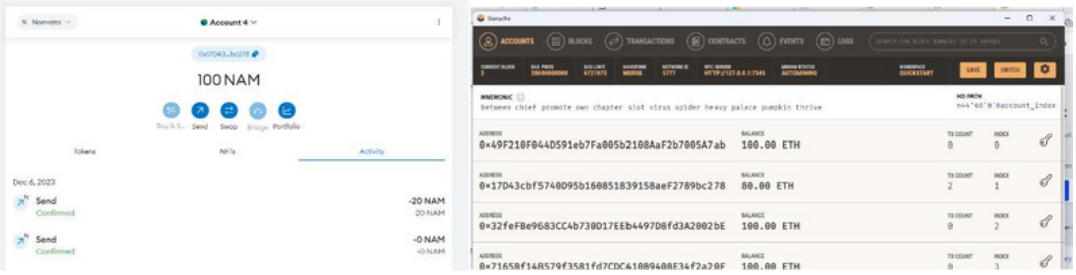
2.



3. Enter the amount of ETH you want to send and confirm the transaction.

4.Confirm the transaction in MetaMask by paying the gas fee.

5. Wait for the transaction to be confirmed on the local Ganache blockchain.



PRACTICAL 6

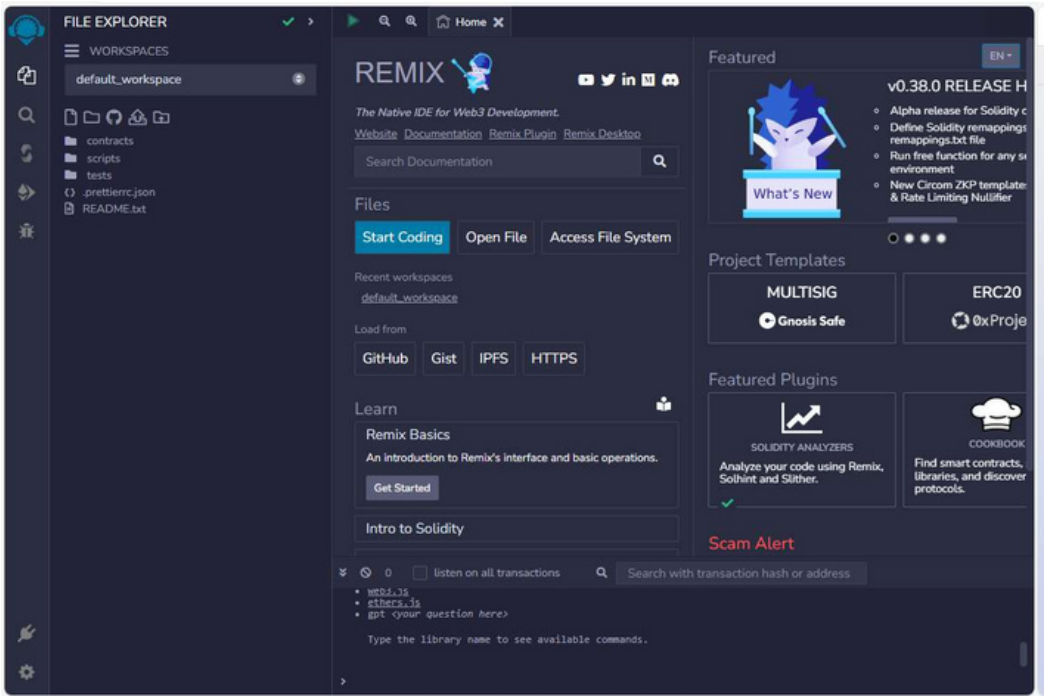
Aim: setup and deploy Voting Smart contract in solidity programming language using Remix Ide.

Background Information:

- Solidity : Programming language for Ethereum smart contracts.
Remix IDE : Web tool for coding and deploying smart contracts.
Smart Contracts : Self-executing agreements on the blockchain.
- SPDX-License-Identifier : Identifies the license for Solidity code.
Mappings and Events : Key-value storage and logging in Solidity.
Require Statement : Validates conditions; reverts if not met.
Front-end Integration : Web interfaces for user interaction.
Truffle and Hardhat : Frameworks for Ethereum development.

Access Remix IDE (https://remix.ethereum.org/) :

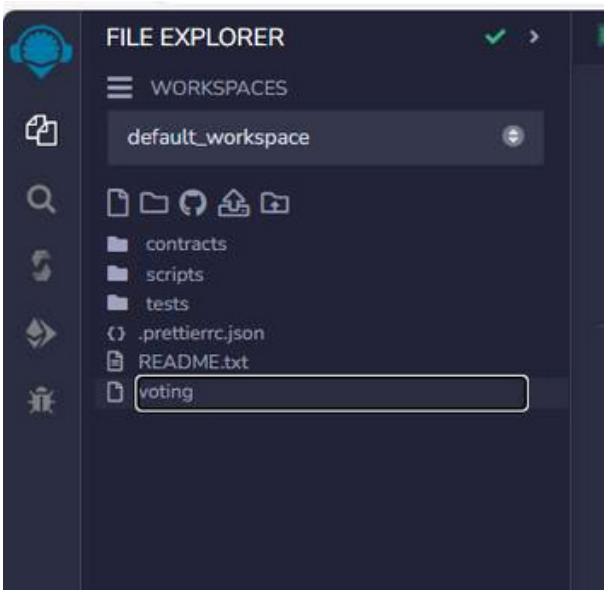
Open your web browser and go to Remix IDE. This is an online Solidity IDE provided by Ethereum.



Create a New File:

Click on the "file" icon on the left sidebar to create a new file in contract folder.

Name the file Voting.sol or any other relevant name for your voting contract.

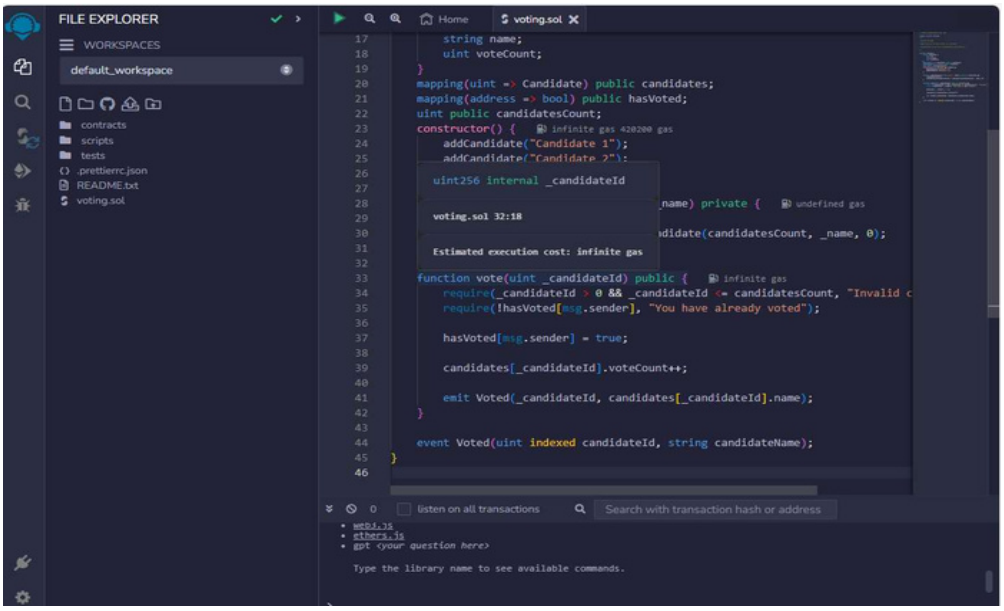


Write Your Smart Contract:

Code:

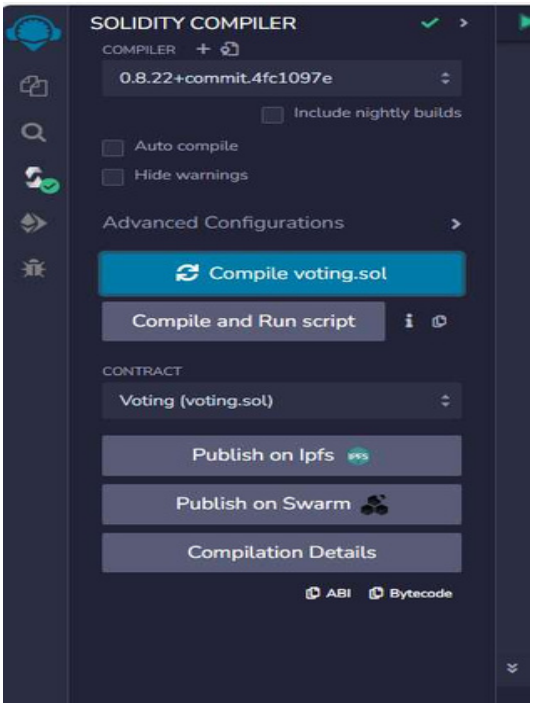
```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;
/**

* @title Storage

* @dev Store & retrieve value in a variable

* @custom:dev-run-script scripts/deploy_with_ethers.ts

*/

contract Voting {
struct Candidate {
uint id;
string name;
uint voteCount;
}
mapping(uint => Candidate) public candidates;
mapping(address => bool) public hasVoted;
uint public candidatesCount;
constructor() {
```

```
 addCandidate("Candidate 1");
addCandidate("Candidate 2");
}

function addCandidate(string memory _name) private {
candidatesCount++;
candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
}

function vote(uint _candidateId) public {
        require(_candidateId > 0 && _candidateId <= candidatesCount, "Invalid candidate ID");
require(!hasVoted[msg.sender], "You have already voted");

hasVoted[msg.sender] = true;

candidates[_candidateId].voteCount++;

emit Voted(_candidateId, candidates[_candidateId].name);
}

event Voted(uint indexed candidateId, string candidateName);
}
```



Compiler Settings:

On the Remix interface, go to the "Solidity Compiler" tab located on the left sidebar. Choose the appropriate compiler version based on your smart contract code. Click "Compile" to compile your contract.



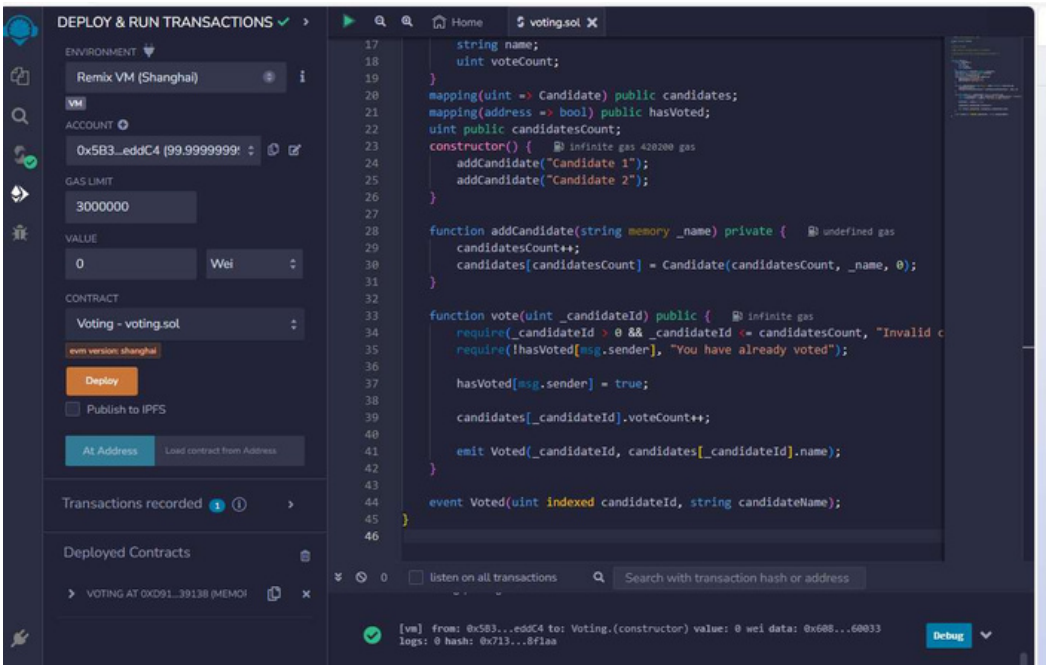
Deploying and Interacting with the Smart Contract:

Deploy the Contract:

Switch to the "Deploy & Run Transactions" tab in Remix.

Ensure the correct environment is selected (e.g., JavaScript VM, Injected Web3, etc.).

Click on "Deploy" to deploy your contract. You'll get a deployed contract address.



PRACTICAL 7

Aim: Build a decentralized Lottery application (DApp), Using ERC-20 Tokens that combines Ethereum's Web3 and Solidity smart contracts deploy on Test network.

Background Information:

To create a decentralized lottery DApp:

1. Smart Contract (Solidity): Write a Solidity smart contract handling ticket purchases, random number generation, and prize distribution.

2. Deployment: Deploy the smart contract on an Ethereum test network.

3. Frontend (Web3.js): Develop a frontend interface using Web3.js to interact with the smart contract. Enable users to buy lottery tickets and trigger the random winner selection.

Steps to follow:

Step 1: Go to Remix IDE (https://remix.ethereum.org/).

• It's an online Solidity IDE provided by Ethereum, where you can write, compile, and deploy Solidity smart contracts.

Step 2: Creating a New File

• Click on the "+" icon on the left-hand side or the "File" menu at the top to create a new file.

• Name the file with a .sol extension, for example, Lottery.sol.

• Write your Solidity code into this file.

Code:

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity ^0.8.11;
/**
* @title Storage
* @dev Store & retrieve value in a variable
* @custom:dev-run-script ./scripts/deploy_with_ethers.ts
*/
contract Lottery {
address public owner;
```

```
 address payable[] public players;
uint public lotteryId;
mapping (uint => address payable) public lotteryHistory;

constructor() {

owner = msg.sender;
lotteryId = 1;
}
      function getWinnerByLottery(uint lottery) public view returns (address payable) {
return lotteryHistory[lottery];
}

function getBalance() public view returns (uint) {
return address(this).balance;
}

function getPlayers() public view returns (address payable[] memory) {
return players;
}

function enter() public payable {
require(msg.value > 1 ether);
// address of player entering lottery
players.push(payable(msg.sender));
}

function getRandomNumber() public view returns (uint) {
return uint(keccak256(abi.encodePacked(owner, block.timestamp)));
}

function pickWinner() public onlyowner {
uint index = getRandomNumber() % players.length;
players[index].transfer(address(this).balance);

lotteryHistory[lotteryId] = players[index];
lotteryId++;


players = new address payable[](0);
}
modifier onlyowner() {
require(msg.sender == owner);
```

```
    _;
    }
}
```

Step 3: Compiling and Run the Contract

• On the left-hand side panel, go to the "Solidity Compiler" tab (it looks like a stack of papers).

• Ensure that the appropriate Solidity compiler version is selected.

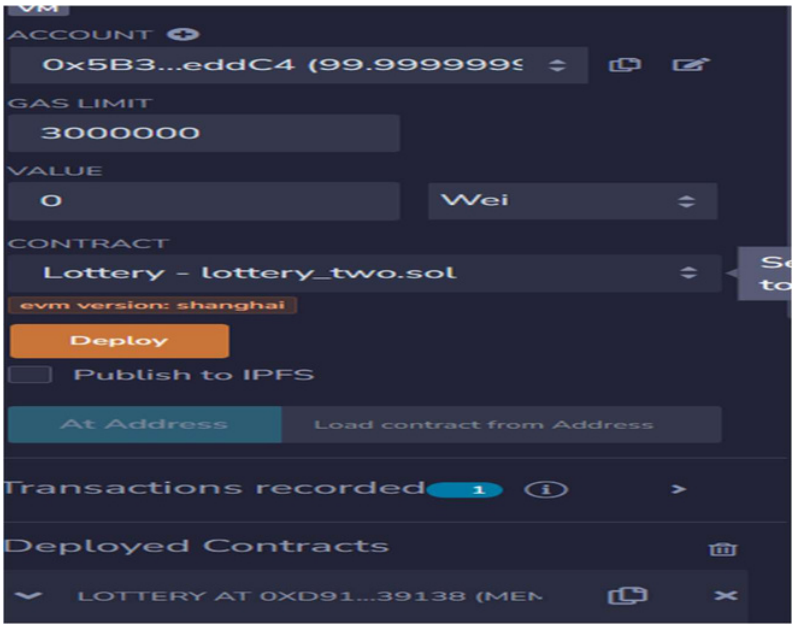
• Click on "Compile Lottery.sol" (or the name of your file) to compile the contract.

• Verify that there are no errors in the compilation output panel.

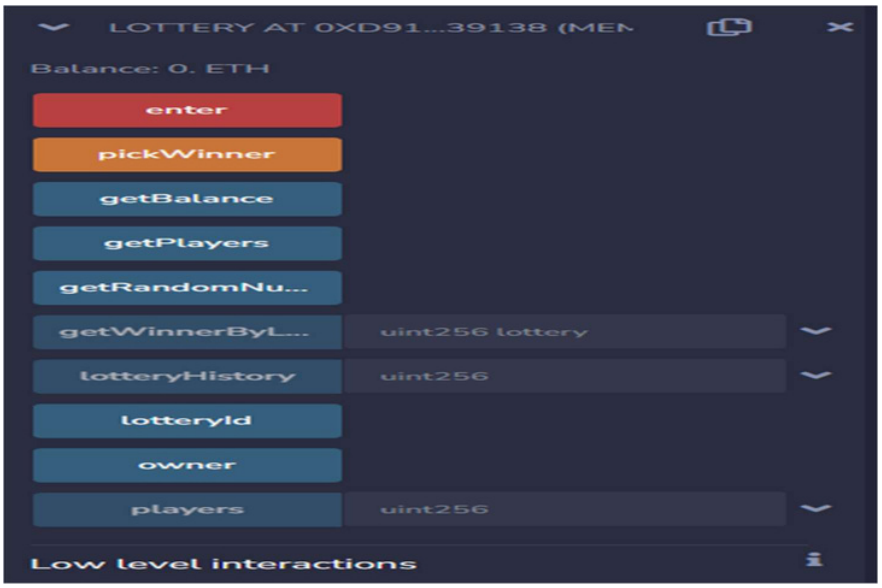
• Click on deploy for first account to set it as manager.

• Then select another account set value to 2 Ether.

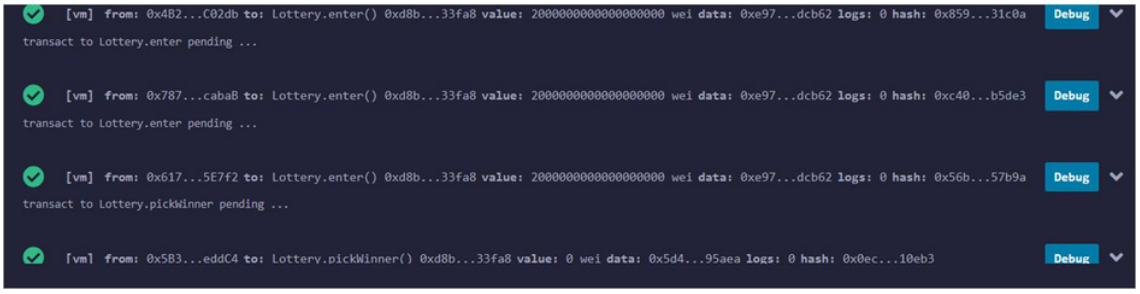
• Click on Transact button.



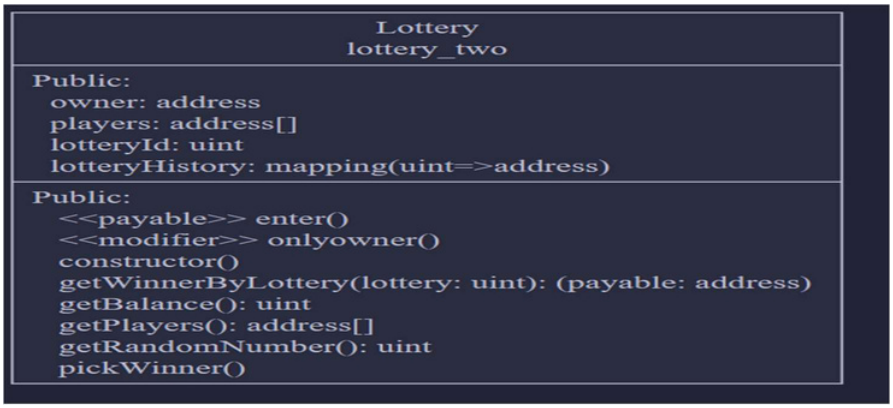
• Repeat more than equal to 3 times to set the number of participants to create Transaction pool for lottery. Remix will display transaction details and logs when you perform actions like deploying a contract or calling its functions. You can also check the transaction hash, gas usage, and status of transactions executed on the Ethereum blockchain.



• You can see the balance of the Transaction pool as manager • Also, can see the transactions executed successfully. • To Execute lottery, click on pickwinner button as manager the winner will be picker from all the participants.



UML diagram to show the architecture, design, and implementation of our Lottery DApp.



PRACTICAL 8

Aim: Practical on Supply Chain Dapp.

Background Information:

A Supply Chain DApp leverages blockchain technology and smart contracts to enhance the efficiency, transparency, and security of supply chain processes. Key components include product traceability, decentralized supplier management, real-time inventory tracking, quality assurance with immutable records, and transparent logistics operations. Smart contracts automate various aspects, such as agreements, reordering, and quality checks, ensuring a reliable and tamper-resistant supply chain.

Steps to follow:

Step 1:

Go to Remix IDE (https://remix.ethereum.org/).

• It's an online Solidity IDE provided by Ethereum, where you can write, compile, and deploy Solidity smart contracts.

Step 2: Creating a New File

• Click on the "+" icon on the left-hand side or the "File" menu at the top to create a new file.

• Name the file with a .sol extension, for example, Supplychain.sol.

• Write your Solidity code into this file.

Code:

```
// SPDX-License-Identifier: GPL-3

pragma solidity ^0.8.0;


/**

* @title Storage

* @dev Store & retrieve value in a variable

* @custom:dev-run-script ./scripts/deploy_with_ethers.ts
```

```
 */

contract SupplyChain {
uint public productCount = 0;

struct Product {
uint id;
string name;
uint quantity;
address owner;
address payable[] history;
}

mapping(uint => Product) public products;

event ProductCreated(uint id, string name, uint quantity, address owner);
event ProductTransferred(uint id, address from, address to);

function createProduct(string memory _name, uint _quantity) public {
productCount++;
address payable[] memory initialHistory;
        products[productCount] = Product(productCount, _name, _quantity, msg.sender,
initialHistory);
emit ProductCreated(productCount, _name, _quantity, msg.sender);
}

function transferProduct(uint _productId, address _newOwner) public {
require(_productId > 0 && _productId <= productCount, "Invalid ID");
```

```
 Product storage product = products[_productId];
        require(msg.sender == product.owner, "Only the owner can transfer the product");


product.owner = _newOwner;
product.history.push(payable(_newOwner));


emit ProductTransferred(_productId, msg.sender, _newOwner);
}


    function getProductHistory(uint _productId) public view returns (address payable[]
memory) {
require(_productId > 0 && _productId <= productCount, "Invalid product ID");
return products[_productId].history;
}
}
```

Step 3: Compiler Settings

On the Remix interface, go to the "Solidity Compiler" tab located on the left sidebar.

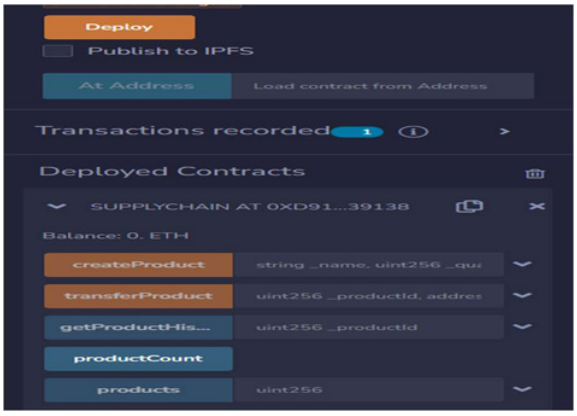
Choose the appropriate compiler version based on your smart contract code.

Click "Compile" to compile your contract.

Step 4: Deploy the Contract

Switch to the "Deploy & Run Transactions" tab in Remix.

Click on "Deploy" to deploy your contract. You'll get a deployed contract address.



Step 5:

Click on createProduct and add your product name in double quotes for example “cake” and then add quantity of the product.

Click on transact button you will get succeed transaction address and then call product.