

PRACTICAL: 1**Date:** / /**TITLE:** To Study different crypto-currencies.**What is Cryptocurrency?**

- A cryptocurrency is a coded string of data representing a currency unit. Peer-to-peer networks called blockchains monitor and organize cryptocurrency transactions, such as buying, selling, and transferring, and also serve as secure ledgers of transactions. By utilizing encryption technology, cryptocurrencies can serve as both a currency and an accounting system.
- A cryptocurrency is a digital or virtual currency that is meant to be a medium of exchange. It is quite similar to real-world currency, except it does not have any physical embodiment, and it uses cryptography to work.
- Because cryptocurrencies operate independently and in a decentralized manner, without a bank or a central authority, new units can be added only after certain conditions are met. For example, with Bitcoin, only after a block has been added to the blockchain will the miner be rewarded with bitcoins, and this is the only way new bitcoins can be generated. The limit for bitcoins is 21 million; after this, no more bitcoins will be produced.
- **Different types of Cryptocurrencies are listed below:**
 - Bitcoin (BTC)
 - Ethereum (ETH)
 - Tether (USDT)
 - Binance Coin (BNB)
 - Cardano (ADA)
 - Solana (SOL)
 - Dogecoin (DOGE)
 - Litecoin (LTC)

Bitcoin



- Bitcoin is the clear leader in the crypto sector. It is also the very first cryptocurrency. Bitcoin launched in 2009; created by a person (or possibly a group) that goes by the pseudonym Satoshi Nakamoto. As of June 2022, there are slightly more than 19 million Bitcoin tokens in circulation, against a capped limit of 21 million. Almost a thousand new bitcoins are mined each day, bringing Bitcoin ever closer to its maximum finite number.
- Bitcoin was designed to be independent of any government or central bank. Instead, it relies on blockchain technology, a decentralized public ledger that contains a digital record of every Bitcoin transaction. Bitcoin established the basic system of cryptography and consensus — i.e., peer-to-peer (P2P) verification — that is the foundation of most forms of crypto today.
- As a reminder, a P2P network structure in blockchain technology is generally decentralized and designed to operate in the best interest of all parties involved, as opposed to benefitting a centralized entity primarily. A peer-to-peer blockchain network connects different computers (or nodes) together, so they can function in unison. Ideally, P2P platforms are censorship resistant, open, public networks, which allow important data and other functionalities to be shared.
- Bitcoin miners use powerful computers to verify blocks of transactions and generate more bitcoins. Bitcoin mining uses a complex, time-consuming process called proof of work (PoW). The transactions are logged permanently on the blockchain — which helps to validate and secure each bitcoin and the network as a whole. Recently, the vast amount of energy required to create Bitcoin has raised concerns about environmental pollution.

Ethereum



- Like Bitcoin, Ethereum is a blockchain network. But Ethereum was designed as a programmable blockchain — meaning it wasn't created to support a currency, but rather to enable the network's users to create, publish, monetize, and deploy decentralized applications (dApps). Ether (ETH), the native Ethereum currency, was developed as a form of payment on the Ethereum platform. It might be helpful to think of ETH as a kind of fuel that powers the Ethereum blockchain. Ethereum has helped to launch many initial coin offerings because many ICOs are built on the Ethereum blockchain. Ethereum has also been the blockchain behind the boom in non-fungible tokens (NFTs).
- As the two most widely known blockchains and cryptocurrencies, many people often directly compare Ethereum and Bitcoin against each other. In reality, Bitcoin and Ethereum are designed to achieve different goals, and in many ways can be regarded as complementary forces. Bitcoin is a peer-to-peer digital cash network, which facilitates transactions without the need for a central authority. This novel network architecture has paved the way for the complex blockchain ecosystem that we have today. Ethereum, often referred to as the world computer, iterates on Bitcoin's technology while introducing smart contracts. Smart contracts allow for building dApps that span a broad range of crowdfunding platforms, financial instruments, digital games and collectibles, and decentralized marketplaces.
- As of June 2022, Ether was the number two virtual currency, behind Bitcoin. Also like BTC, ETH is generated using a PoW system. But unlike Bitcoin, there is no limit to the number of ETH that can be created.

Tether



Tether (USDT)

- Tether was the first cryptocurrency marketed as a stable coin — a breed of crypto known as fiat-collateralized stable coins. The value of the tether is pegged to a fiat currency — in this case, the U.S. dollar. Tether is the world's largest stable coin; in 2022, the majority of cryptocurrencies traded using tether.
- Like other stable coins, tether is designed to offer stability, transparency, and lower transaction fees to users. Tether was not meant to be a speculative investment like some cryptocurrencies; originally, investors who wanted to avoid the extreme volatility of the crypto market used USDT. Tether is pegged to the U.S. dollar (which is why the ticker is USDT), and it allegedly maintains a 1:1 value with the dollar, although this claim has come under some scrutiny.
- Many believe that Tether is the lifeblood of the crypto ecosystem. They're concerned that if Tether implodes, then the entire system would crash.
- In May 2022, that's exactly what happened: Tether lost its peg to the dollar briefly, and all cryptocurrencies plummeted. In part, this was a result of another stable coin, terraUSD (USD) falling below 30 cents. The wave of panic in the broader crypto market was palpable. Because of this crash, many crypto investors tried to redeem their tethers, others tried to exit the asset class altogether, and many lost their investments.

Binance coin

- Binance is one of the world's biggest cryptocurrency exchanges. The Binance Coin (BNB) was created as a utility token for use as a medium of exchange on Binance. It was initially built on the Ethereum blockchain, but now lives on Binance's own blockchain platform. Originally, BNB allowed traders to get discounts on trading fees on Binance, but now it also can be used for payments, to book travel, for entertainment, online services, and financial services.



- As one of the top five cryptocurrencies by market cap in 2022, BNB has developed a wide range of use cases and real-world applications. But, as with other digital assets, this crypto platform has also faced regulatory hurdles here and abroad.
- BNB was created with a maximum of 200 million tokens, about half of which were made available to investors during its ICO. Every quarter, to drive demand, Binance buys back and then “burns” — permanently destroys, or removes from circulation — some of the coins it holds. A project burns its tokens to reduce the overall supply. The motivation is often to increase the value of the remaining tokens, as assets tend to rise in price whenever the circulating supply falls, and they become more scarce.

Cardano



- Cardano is a Proof-of-Stake blockchain platform with smart contract functionality. In particular, Cardano is noted for its focus on academic research, high transactions-per-second (TPS) throughput, and an energy-efficient consensus mechanism called Ouroboros. ADA, the native coin of the Cardano network, is used to facilitate transactions and execute smart contracts.
- Cardano bills itself as a third-generation blockchain platform, to cast itself as a next-level player. Cardano relies on proof of stake (PoS), which means that the complicated PoW calculations and high

electricity usage required for mining coins like Bitcoin aren't necessary. This potentially makes Cardano's network more efficient and sustainable than some other crypto networks.

- Cardano's cryptocurrency is called ADA, after Ada Lovelace, a 19th-century mathematician.
- Cardano's main applications are in identity management and traceability. The first application can be used to streamline the collection of data from multiple sources. The latter can be used to audit a product's manufacturing path, and potentially prevent fraud and counterfeit goods.
- Cardano is being built in five phases toward achieving its goal of developing the network into a decentralized application (dApp) platform with a multi-asset ledger and verifiable smart contracts. Each phase, or era, in the Cardano roadmap is anchored by its research-based framework and peer-reviewed insights, which have helped establish its scholarly reputation

Solana



- Solana is a blockchain platform that generates the cryptocurrency, Sol. Solana has made strides in decentralized finance (DeFi) and specifically with its smart contract technology — programs that run on the platform according to pre-set conditions. Smart contracts are similar to paper contracts, but without the middlemen. Solana was also behind the Degenerate Ape Academy, an NFT launched in August 2021. One of the essential innovations Solana brings to the table is its proof-of-history (PoH) consensus. This mechanism allows for greater scalability of the protocol, which in turn boosts usability.
- SOL is designed to facilitate the creation of dApps. It aims to improve scalability by introducing a proof-of-history (PoH) consensus combined with the underlying proof-of-stake (PoS) consensus of the blockchain.
- Because of the innovative hybrid consensus model, Solana enjoys interest from small-time traders and institutional traders alike. A significant focus for the Solana Foundation is to make decentralized finance accessible on a larger scale.

- Solana is known to have an incredibly short processing time. SOL's hybrid protocol allows for significantly decreased validation times for both transaction and smart contract execution. With lightning-fast processing times, Solana has attracted a lot of institutional interest as well.
- The Solana protocol is intended to serve both small-time users and enterprise customers alike. One of Solana's main promises to customers is that they will not be surprised by increased fees and taxes. The protocol is designed in such a way as to have low transaction costs while still guaranteeing scalability and fast processing.

Dogecoin



- Dogecoin is widely known as the first joke cryptocurrency; it was launched in 2013 as a way to poke fun at Bitcoin. Nonetheless, the currency captured people's attention and a fair amount of investment. In April 2019, a tweet from Elon Musk indicated he had a positive view of Dogecoin, which further raised Dogecoin's profile as a legitimate cryptocurrency.
- Dogecoin is an altcoin similar to Bitcoin and Ethereum in that it runs on a blockchain network using a PoW system. But the number of coins that can be mined are unlimited (versus the 21-million-coin cap on Bitcoin).
- Dogecoin has been used primarily as a tipping system on Reddit and Twitter to reward the creation or sharing of quality content. You can get tipped Dogecoin by participating in a community that uses the digital currency, or you can get your Dogecoin from a Dogecoin faucet. A Dogecoin faucet is a website that will give you a small amount of Dogecoin for free as an introduction to the currency, so that you can begin interacting in Dogecoin communities.
- Dogecoin is also associated with some headline moments in crypto — for example, investors paid the equivalent of about \$30,000 in Dogecoin to help send the Jamaican bobsled team to the Winter Olympics in 2014.

- Despite its place as one of the biggest coins by market cap, DOGE trades at one of the lowest prices: \$0.072 cents, as of June 25, 2022.

Litecoin



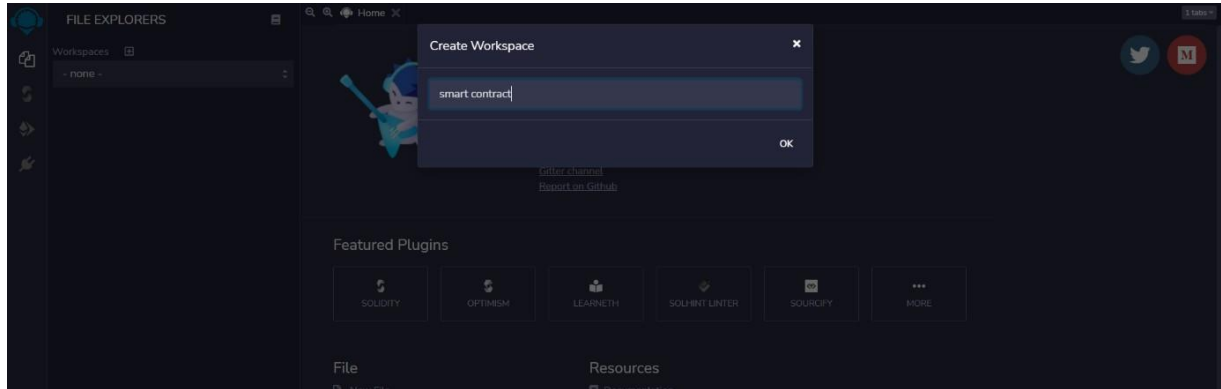
- Litecoin (LTC) is a cryptocurrency created in 2011 as one of the first altcoins (alternatives to bitcoin). Though it's built on bitcoin's original source code and shares certain features with BTC, LTC was designed to improve upon BTC, especially in terms of transaction speed.
- Though Litecoin was initially a popular entry into the crypto category, it has gained and lost value over time, displaying a similar volatility to many cryptocurrencies (or even certain stocks and bonds).
- Like many forms of crypto, Litecoin is a decentralized, peer-to-peer cryptocurrency; it was created from a fork in the Bitcoin blockchain, the transparent, digital public ledger used by most cryptocurrencies. Litecoin was designed to enable almost instant, near-zero cost payments that can be exchanged between people or institutions worldwide.
- As with Bitcoin, Litecoin uses a PoW consensus system to verify transactions on the blockchain, but owing to certain modifications it's considered a lighter, faster version of Bitcoin. The main difference between Litecoin and Bitcoin is that Litecoin uses a mining algorithm called scrypt, to enable faster transaction times.
- Litecoin generates a new block to be mined every 2.5 minutes, which is about four times faster than Bitcoin's 10 minutes. The Litecoin supply is also four times as great. While Bitcoin has a cap of 21 million coins, the Litecoin supply overall has a cap of 84 million.

Rubrics for Evaluation:

Understanding of Problem (3 Marks)	Capability of writing program (8 Marks) / Understanding of Basic content in case of the Study Practical	Documentation (3 Marks)	Knowledge and awareness related to the topic (3 Marks)	Enthusiasm/ audience awareness (3 Marks)	Total (20)

PRACTICAL: 2**Date:** / /**TITLE:** Write a crowd-sale smart contract code in Solidity programming language.

- Firstly, we create workspace on remix IDE.



- Then, create Bank.sol file which contain following code:

Bank.sol

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity >= 0.5.0 < 0.9.0;
//pragma solidity ^0.5.17;
interface token {
    function transfer(address receiver, uint amount) external;
}

contract Crowdsale{

    address payable public beneficiary;
    uint public fundingGoal;
    uint public amountRaised;
    uint public deadline;
    uint public price;
```

```
token public tokenReward;

mapping (address => uint256) public balanceOf;
bool fundingGoalReached = false;
bool crowdsaleClosed = false;

event GoalReached(address receipant, uint totalAmountRaised);
event FundTransfer(address backer, uint amount, bool isContribution);

constructor(
    address payable ifSuccessfulSendTo,
    uint fundingGoalInEthers,
    uint durationInMinutes,
    uint etherCostForEachToken,
    address addressOfTokenUsedAsReward
) public {

    beneficiary = ifSuccessfulSendTo;
    fundingGoal = fundingGoalInEthers * 1 ether;
    deadline = block.timestamp + durationInMinutes * 1 minutes;
    price = etherCostForEachToken * 1 ether;
    tokenReward = token(addressOfTokenUsedAsReward);

}

function() external payable{
//fallback() external payable {
    require(!crowdsaleClosed);
    uint amount = msg.value;
    balanceOf[msg.sender] += amount;
    amountRaised += amount;
    tokenReward.transfer(msg.sender, amount/price);

    emit FundTransfer(msg.sender, amount, true);
```

```
}

modifier afterDeadline(){
    if(block.timestamp >= deadline) _;
}

function checkGoalReached() public afterDeadline {
    if(amountRaised >= fundingGoal){
        fundingGoalReached = true;
        emit GoalReached(beneficiary, amountRaised);
    }
    crowdsaleClosed = true;
}

function safeWithdrawal() public afterDeadline {
    if(!fundingGoalReached){
        uint amount = balanceOf[msg.sender];
        balanceOf[msg.sender] = 0;

        if(amount > 0){
            if(msg.sender.send(amount)){
                emit FundTransfer(msg.sender, amount, false);
            }else{
                balanceOf[msg.sender] = amount;
            }
        }
    }

    if(fundingGoalReached && beneficiary == msg.sender){
        if(beneficiary.send(amountRaised)){
            emit FundTransfer(beneficiary, amountRaised, false);
        }else{
            fundingGoalReached = false;
        }
    }
}
```

```

    }
}
}
}

```

Output:

```

//SPDX-License-Identifier: GPL-3.0
pragma solidity >= 0.5.0 < 0.9.0;
//pragma solidity ^0.5.17;
interface token {
    function transfer (address receiver, uint amount) external;
}

contract Crowdsale{

    address payable public beneficiary;
    uint public fundingGoal;
    uint public amountRaised;
    uint public deadline;
    uint public price;
    token public tokenReward;
    mapping (address => uint256) public balanceOf;
    bool fundingGoalReached = false;
    bool crowdsaleClosed = false;

    event GoalReached(address receipant, uint totalAmountRaised);
    event FundTransfer(address backer, uint amount, bool isContribution);

    constructor(
        address payable ifSuccessfulSendTo,
        uint fundingGoalInEthers,
        uint durationInMinutes,
        uint etherCostForEachToken,
        address addressOfTokenUsedAsReward
    ) public {

```

TRANSACTIONS

ACCOUNT

0x5B3...edc4 (99.999999%)

GAS LIMIT

3000000

VALUE

0 Wei

CONTRACT

Crowdsale - Bank.sol

DEPLOY

IFSUCCESSFULSENDTO: 0x5B38Da6a701c56854E

FUNDINGGOALINETHERS: 2

DURATIONINMINUTES: 1

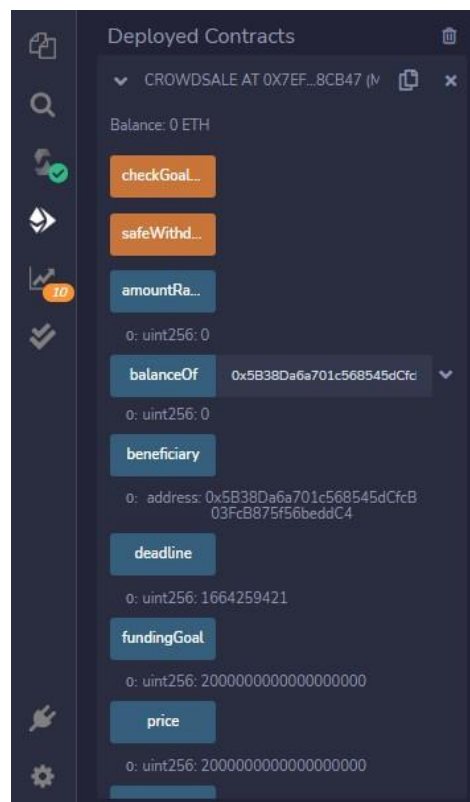
ETHERCOSTFOREACHTOKEN: 2

ADDRESSOFTOKENUSEDASREWARD: 0xAb8483F

transact

☐ Publish to IPFS

OR



Rubrics for Evaluation:

Understanding of Problem (3 Marks)	Capability of writing program (8 Marks) / Understanding of Basic content in case of the Study Practical	Documentation (3 Marks)	Knowledge and awareness related to the topic (3 Marks)	Enthusiasm/ audience awareness (3 Marks)	Total (20)

PRACTICAL: 3**Date:** / /**TITLE:** Ethereum – With respect to Ethereum, carry out following :

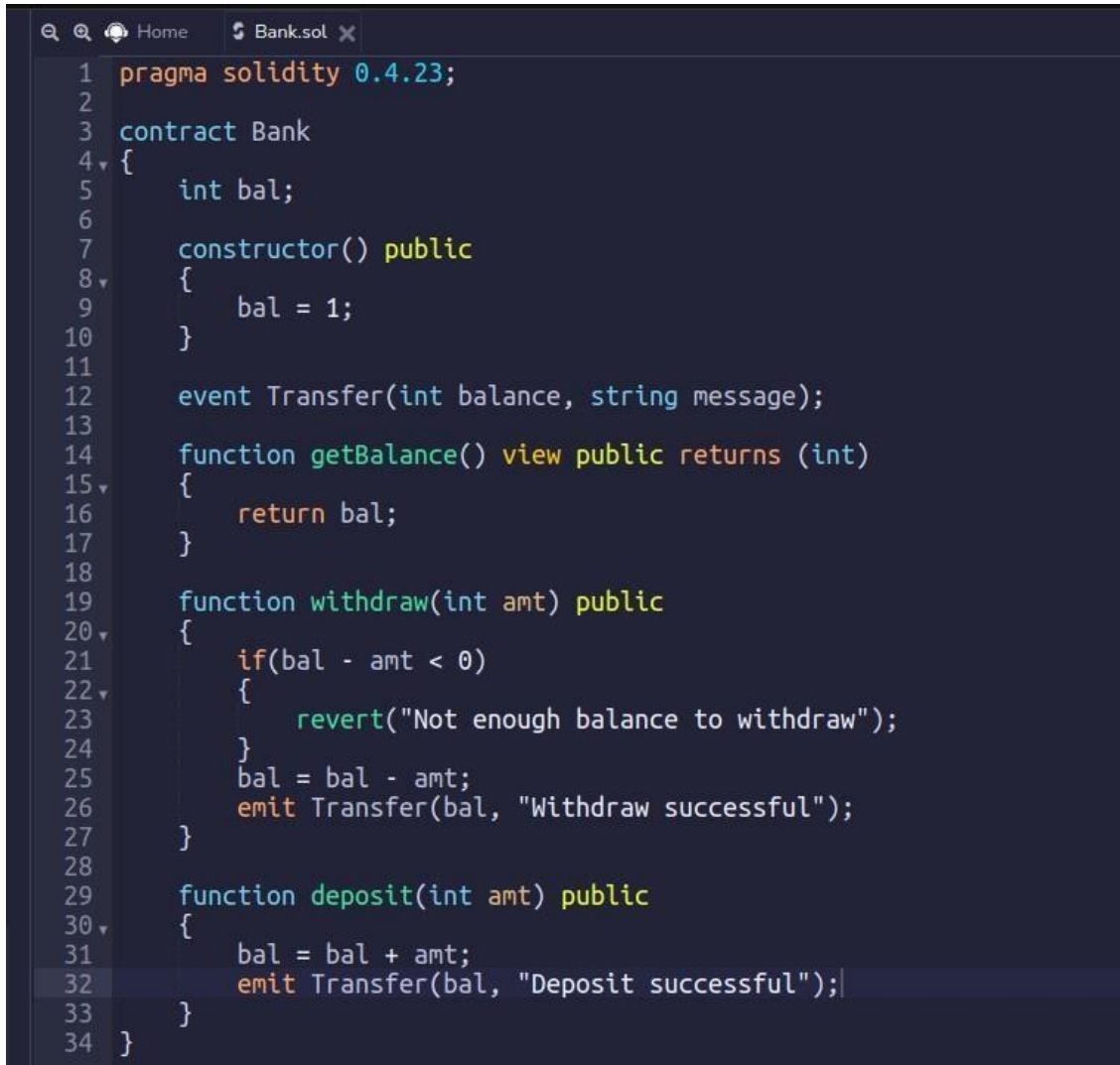
- Install Geth Client & Configure Ethereum Nodes
- Manage Accounts and Account states.
- Enable Mining and checking balance in Ether
- Setting up Metamask and Testing Fund Transfer – work with Ethereum Ecosystem
- Add Parameters to Cryptocurrency
- Check Balance Before Transfer
- Adding Transfer Event for Logging

Ganache installation:

Ganache				
ACCOUNTS	BLOCKS	TRANSACTIONS	CONTRACTS	EVENTS
LOGS	SEARCH FOR BLOCK NUMBERS OR TX HASHES			
CURRENT BLOCK 13	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK MUIRGLACIER	NETWORK ID 5777
RPC SERVER HTTP://127.0.0.1:7545	MINING STATUS AUTOMINING	WORKSPACE QUICKSTART	SAVE	SWITCH
MNEMONIC ? sing obtain assault bridge sad sight month put reopen window invite robot		HD PATH m/44'/60'/0'/0/account_index		
ADDRESS 0x8d058e72cd913E541B212C1495c64B4742344bb2	BALANCE 79.98 ETH	TX COUNT 13	INDEX 0	
ADDRESS 0xFddAbEBa613067f140036d2f8571Ee04c53EcFb5	BALANCE 100.00 ETH	TX COUNT 0	INDEX 1	
ADDRESS 0x9274384865896b6dC47583495F183eFB74C0085F	BALANCE 100.00 ETH	TX COUNT 0	INDEX 2	
ADDRESS 0x73Ced982c15B784F01663a4B235bF6fF343E2Bc4	BALANCE 100.00 ETH	TX COUNT 0	INDEX 3	
ADDRESS 0x933480639716018c4FFDc3e2dFd7419065FBb100	BALANCE 100.00 ETH	TX COUNT 0	INDEX 4	
ADDRESS 0x6D9005F9dee695685DEd5FADcE0744b1C9091CB7	BALANCE 100.00 ETH	TX COUNT 0	INDEX 5	
ADDRESS 0x7a3729D51A38E80E820693D7A40F63665fbd1ce1	BALANCE 100.00 ETH	TX COUNT 0	INDEX 6	
ADDRESS	BALANCE	TX COUNT	INDEX	

- Ganache is a personal blockchain for rapid Ethereum and Corda distributed application development. You can use Ganache across the entire development cycle; enabling you to develop, deploy, and test your dApps in a safe and deterministic environment.

Smart Contract

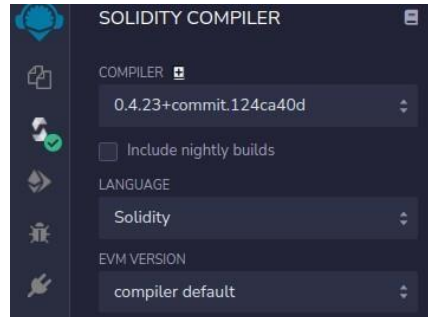


```
1 pragma solidity 0.4.23;
2
3 contract Bank
4 {
5     int bal;
6
7     constructor() public
8     {
9         bal = 1;
10    }
11
12    event Transfer(int balance, string message);
13
14    function getBalance() view public returns (int)
15    {
16        return bal;
17    }
18
19    function withdraw(int amt) public
20    {
21        if(bal - amt < 0)
22        {
23            revert("Not enough balance to withdraw");
24        }
25        bal = bal - amt;
26        emit Transfer(bal, "Withdraw successful");
27    }
28
29    function deposit(int amt) public
30    {
31        bal = bal + amt;
32        emit Transfer(bal, "Deposit successful");
33    }
34 }
```

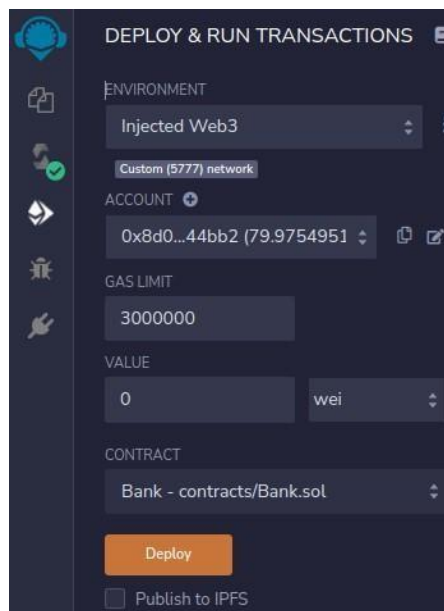
- As seen in the code above, the smart contract consists of three functions:
 1. getBalance: returns the balance in the account
 2. deposit: deposits the given amount into the account
 3. withdraw: withdraws the given amount from the account
- When withdrawing, if the balance is less than the amount, an error is thrown.

- Also, there is a Transfer event which takes balance and a message and it is emitted when successful deposit and withdraw take place.

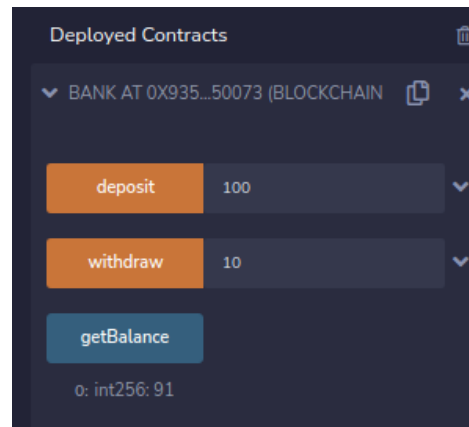
Compile:



Deploy:



Run:



Deployed Contracts

▼ BANK AT 0X935...50073 (BLOCKCHAIN) [icon] x

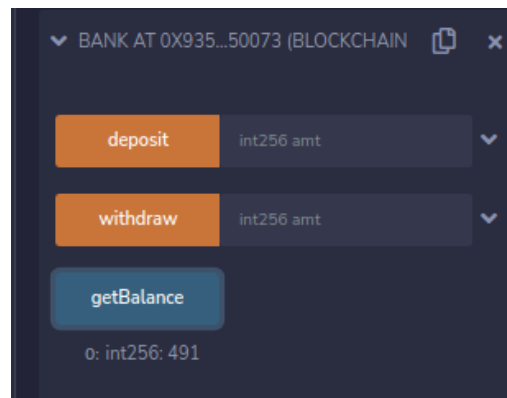
deposit 100 ▼

withdraw 10 ▼

getBalance

0: int256: 91

Get balance:



▼ BANK AT 0X935...50073 (BLOCKCHAIN) [icon] x

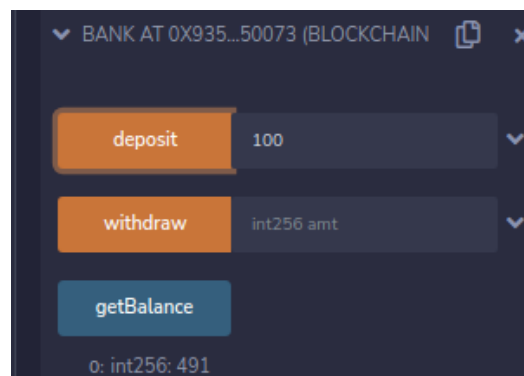
deposit int256 amt ▼

withdraw int256 amt ▼

getBalance

0: int256: 491

Deposit 100:



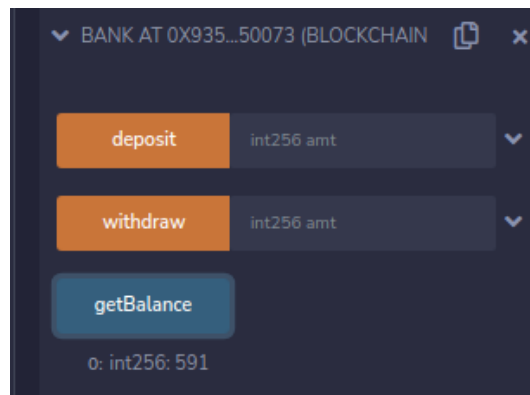
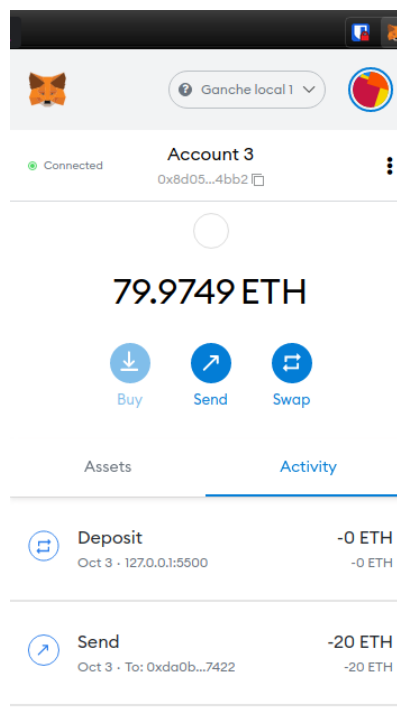
▼ BANK AT 0X935...50073 (BLOCKCHAIN) [icon] x

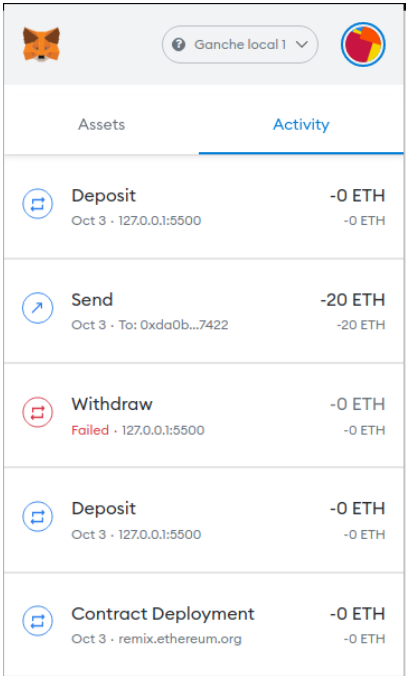
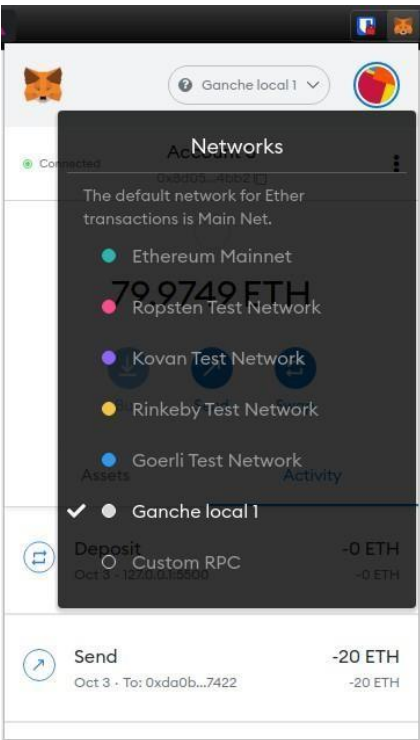
deposit 100 ▼

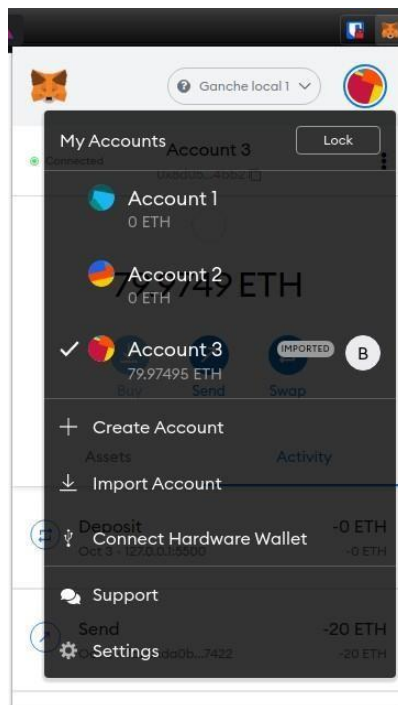
withdraw int256 amt ▼

getBalance

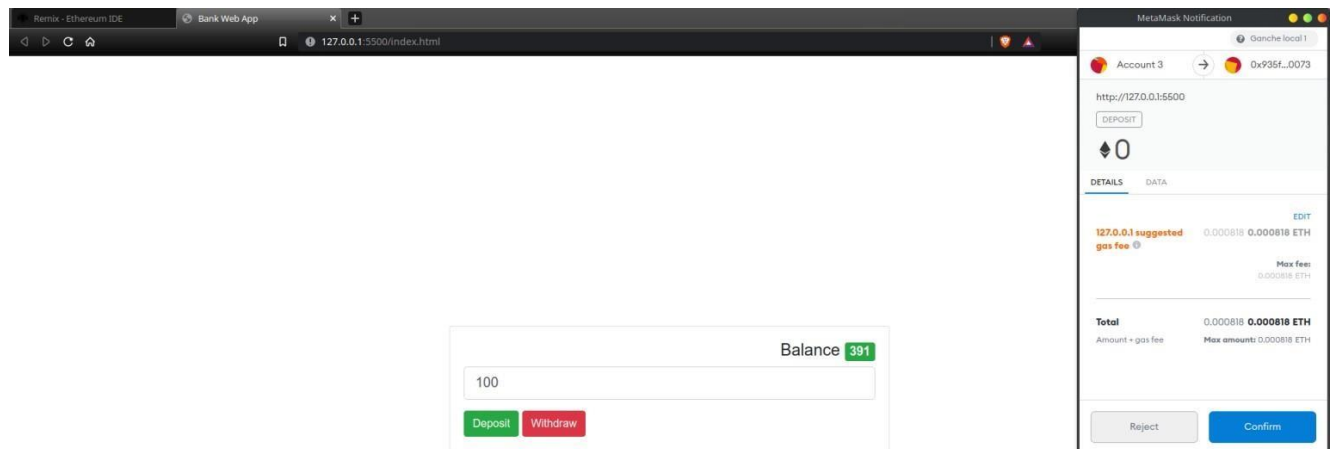
0: int256: 491

Get balance:**Metamask:**





Frontend for the smart contract:



Balance 491

Amount

Deposit
Withdraw

Rubrics for Evaluation:

Understanding of Problem (3 Marks)	Capability of writing program (8 Marks) / Understanding of Basic content in case of the Study Practical	Documentation (3 Marks)	Knowledge and awareness related to the topic (3 Marks)	Enthusiasm/ audience awareness (3 Marks)	Total (20)

PRACTICAL: 4**Date:** / /**TITLE:** Hyperledger - With respect to Hyperledger, carry out following :

- Installing Hyperledger Fabric(latest version)
- Build Network with Network configuration
- Hyperledger Fabric Demo

Installing Hyperledger Fabric(latest version)

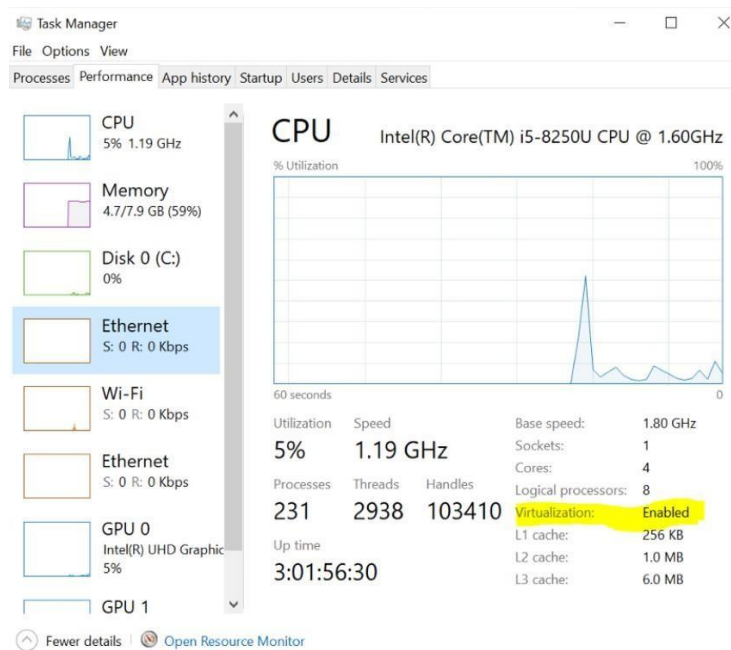
- Let's begin the windows installation.
- I am using Windows 10 Pro for the Fabric v1.4 installation.

Step 1: cURL

- Please check if cURL is already installed in your PC.
curl --help
- If you don't get any error it means cURL is installed in your PC and you can go to the next step. For others please follow the below steps.
 - To install cURL, download the package according to your Windows 32/64 bit. Extract the package and run the curl.exe present in the bin folder.
 - Add the curl in the environment variable.
 - Open the cmd and check the curl --help .
- If you don't get any error it means you curl is installed successfully.

Step 2: Docker and Docker Compose

- Before installing the docker, check if virtualization is Enabled in your PC or not. To check it, open Task Manager >> Performance Tab >> CPU



- From the BIOS settings, virtualization can be turned to Enabled. Instructions to enter BIOS settings vary from the pc manufacturer to manufacturer.
- Once the virtualization is Enabled we can move to download the docker.
- Please be sure which Windows you're using before installing Docker. There are 2 versions of Docker for Windows.
- Docker Toolbox — Windows 8, Windows 10 Home
- Docker Desktop — Windows 10 Pro, Enterprise — 64 bit
- You first need to have an account in DockerHub to download the docker desktop. Please signup if you don't have one. Download the docker.
- **Note: While installing keep the settings default don't change anything.**

Test the Installation

- Open the cmd window
- Run `docker --version` and `docker-compose --version`

`docker --version`

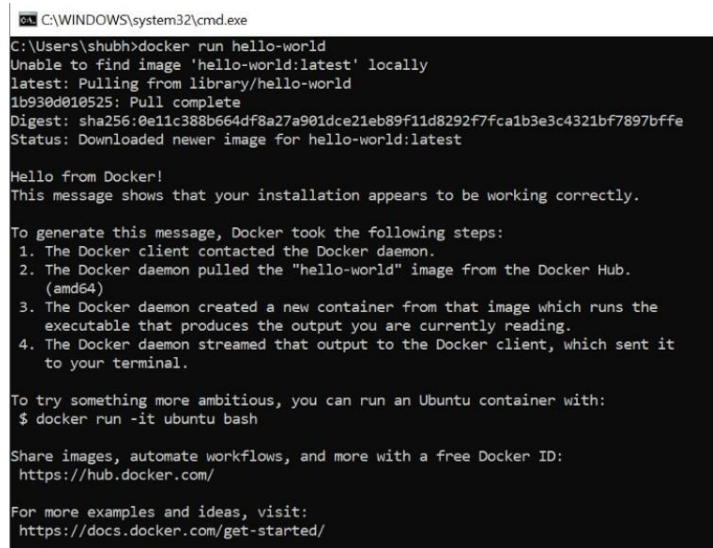
Docker version 18.09.2, build 6247962

docker-compose --version

docker-compose version 1.23.2, build 1110ad01

- Pull the [hello-world image](#) from Docker Hub and run a container:

docker run hello-world



```
C:\WINDOWS\system32\cmd.exe
C:\Users\shubh>docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:0e11c388b664df8a27a901dce21eb89f11d8292f7fca1b3e3c4321bf7897bffe
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/
```

- If you get this message then the Docker installed successfully in your machine. For more information on Docker check [the official documentation](#).

Step 3: Golang

- Download the Golang package from the official site. Once it is installed open the command prompt and run
 - go
 - version
 - Output
 - go version go1.11.5 windows/amd64

Step 4: Nodejs and npm

- Download the node v8.x from this [link](#) and install it. Check if it is installed correctly.

```
node -v
```

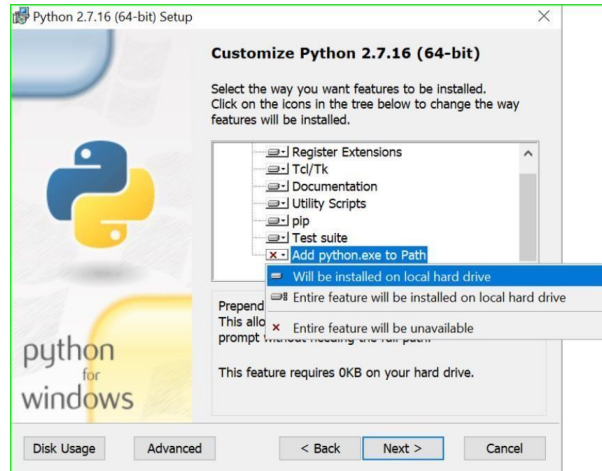
```
v8.16.0
```

```
npm -v
```

```
6.4.1
```

Step 5: Python 2.7

- Download the python 2.7 from its official site.
- While installing add python to the system Path variable. This allows you to type 'python' into a command prompt without needing the full path.
- Change Add python.exe to Path to Will be installed on the local hard drive



- Check the python installed correctly or not.

```
Python--version
```

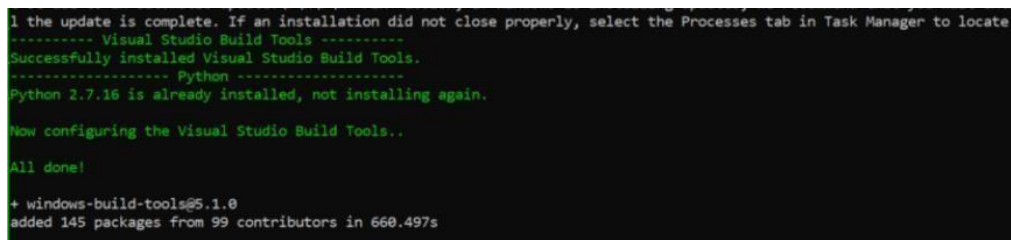
```
Python 2.7.16
```

- The Hyperledger Fabric prerequisites are installed. Now, it is time to install the extra windows dependencies.

Windows Extras

Step 6: Install windows-build-tools and grpc

- Install the windows-build-tools globally using npm .
- Open the command prompt and run the below command.
`npm install --global windows-build-tools`
- It will take some time around 15 minutes or more. Once it will complete you will get the below message.



```
! the update is complete. If an installation did not close properly, select the Processes tab in Task Manager to locate
----- Visual Studio Build Tools -----
Successfully installed Visual Studio Build Tools.
----- Python -----
Python 2.7.16 is already installed, not installing again.

Now configuring the Visual Studio Build Tools..

All done!

+ windows-build-tools@5.1.0
added 145 packages from 99 contributors in 660.497s
```

- Once this is done, you should also install the NPM GRPC module with the following command:
`npm install --global grpc`

Step 7: Install git to run the bash commands

- To run the bash commands we have to install git .
- Git is a set of command line utility programs that are designed to execute on a Unix style command-line environment.
- — [atlassian](#)
- Git Bash is an application for Microsoft Windows environments which provides an emulation layer for a Git commandline experience. — [atlassian](#)
- Download the git.
- Hang on, for a while we are done with prerequisites and now we are on the final step to install the Hyperledger Fabric.

Step 8: Install Samples, Binaries and Docker Images

- Go to the directory where you want to download the fabric samples.
- Once you're in the directory open git bash . Right-click and select Git Bash Here .
- Run the below command to install Samples, Binaries and Docker Images

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

- It will download the latest production release.
- If you want a specific release for ex. v1.4.1, follow the below command.

```
curl -sSL http://bit.ly/2ysbOFE | bash -s -- <fabric_version> <fabric-ca_version> <thirdparty_version>
```

```
curl -sSL http://bit.ly/2ysbOFE | bash -s -- 1.4.1 1.4.1 0.4.15
```

- It'll take some time, once it is finished you can see a new directory "fabric-samples".
- "fabric-samples" come with sample examples to start with Hyperledger Fabric. There are many good examples to play within the fabric samples.

Step 9: Test the fabric network

- As we have successfully completed the setting up the Fabric environment, it's time to test it. We are going to use the first-network sample from the fabric-samples.
- Open the fabric-samples and go to first-network.

```
cd fabric-samples/first-network
```

- To test it, run the byfn.sh . It is a test script, it first setup the network with 2 organizations org1 and org2 with 2 peers each and an orderer.

```
./byfn.sh up
```

```

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
Pulling orderer.example.com (hyperledger/fabric-orderer:latest)...
latest: Pulling from hyperledger/fabric-orderer
Pulling peer0.org1.example.com (hyperledger/fabric-peer:latest)...
latest: Pulling from hyperledger/fabric-peer
Creating peer1.org2.example.com ... done
Creating peer0.org2.example.com ... done
Creating peer1.org1.example.com ... done
Creating orderer.example.com ... done
Creating peer0.org1.example.com ... done
Creating cli ... done

START

Build your first network (BYFN) end-to-end test

Channel name : mychannel
Creating channel...

```

- On successful execution, you'll see the below message.

```

Querying chaincode on peer1.org2...
===== Querying on peer1.org2 on channel 'mychannel'... =====
+ peer chaincode query -C mychannel -n mycc -c '{"Args":["query","a"]}'
Attempting to Query peer1.org2 ...3 secs
+ res=0
+ set +x
90
===== Query successful on peer1.org2 on channel 'mychannel' =====
===== All GOOD, BYFN execution completed =====

END

```

- If you reach till this point it means you have successfully setup the fabric network. Now, we have completed the testing of first-network, clean the network.

./byfn down

- Here, we have completed the Hyperledger Fabric installation in the Windows machine.
- Yes, we successfully installed the Fabric on Windows.

Build Network with Network configuration

- Let's Switch to BYFN directory

```
cd fabric-samples/first-network
```

➤ This is the file or directory list that we will study:

- 1) docker-compose-cli.yaml (Part 1)
- 2) base/docker-compose-base.yaml
- 3) base/peer-base.yaml
- 4) channel-artifacts/
- 5) crypto-config.yaml
- 6) configtx.yaml
- 7) byfn.sh
- 8) scripts/script.sh
- 9) script/utlis.sh

- In order to develop Blockchain application, we need to have a Blockchain network first, right? This is a filerelated to setting up the network.
- This is a Docker compose file, which defines your (virtual) Fabric network, such as what nodes are in thenetwork, their internal use domain names, etc.
- Below is docker-compose-cli.yaml

networks:

byfn:

services:

orderer.example.com:

extends:

file: base/docker-compose-base.yaml

service: orderer.example.com

container_name: orderer.example.com

networks:

- byfn

peer0.org1.example.com:

container_name: peer0.org1.example.com

extends:

file: base/docker-compose-base.yaml

service: peer0.org1.example.com

networks:

- byfn

peer1.org1.example.com:

container_name: peer1.org1.example.com

extends:

file: base/docker-compose-base.yaml

service: peer1.org1.example.com

networks:

- byfn

peer0.org2.example.com:

container_name: peer0.org2.example.com

extends:

file: base/docker-compose-base.yaml

service: peer0.org2.example.com

networks:

- byfn

peer1.org2.example.com:

container_name: peer1.org2.example.com

extends:

file: base/docker-compose-base.yaml

service: peer1.org2.example.com

networks:

- byfn

cli:

container_name: cli

```
image: hyperledger/fabric-  
tools:$IMAGE_TAG  
  
tty: true  
  
stdin_open: true  
  
environment:  
- GOPATH=/opt/gopath  
  
-  
CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock  
- FABRIC_LOGGING_SPEC=INFO  
- CORE_PEER_ID=cli  
  
-  
CORE_PEER_ADDRESS=peer0.org1.example.com:7051  
- CORE_PEER_LOCALMSPID=Org1MSP  
  
- CORE_PEER_TLS_ENABLED=true  
  
working_dir:  
/opt/gopath/src/github.com/hyperledger/fabric/peer  
  
command: /bin/bash  
  
volumes:  
  
- /var/run:/host/var/run/  
  
-  
./chaincode:/opt/gopath/src/github.com/chaincode  
- ./crypto-  
config:/opt/gopath/src/github.com/hyperledger
```



```

er/fabric/peer/crypto/

-

./scripts:/opt/gopath/src/github.com/hyperled
ger/fabric/peer/scripts/

- ./channel-
artifacts:/opt/gopath/src/github.com/hyperled
ger/fabric/peer/channel-
artifacts
depends_on:

- orderer.example.com

- peer0.org1.example.com

- peer1.org1.example.com

- peer0.org2.example.com

- peer1.org2.example.com

networks:

- byfn

```

Rubrics for Evaluation:

Understanding of Problem (3 Marks)	Capability of writing program (8 Marks) / Understanding of Basic content in case of the Study Practical	Documentation (3 Marks)	Knowledge and awareness related to the topic (3 Marks)	Enthusiasm/ audience awareness (3 Marks)	Total (20)

PRACTICAL: 5**Date:** / /**TITLE:** Corda - With respect to Corda, carry out following :

- Corda – Setting up the Environment
- Building the Corda Network

Corda – Setting up the Environment

- Before starting Corda5 CorDapp development, perform the below steps to set up your development environment:

Step 1 – Setup Java 11 and Gradle version 6

- We want to be compliant with the current industry standard and hence Corda 5 is built and tested on Java 11. Corda 5 is not backwards compatible with any older Java versions. So make sure you download Open JDK or Azul. We have upgraded to Gradle version 6, considering the various security bugs fixes made in this version.

Step 2 – Setup Docker

- Download and install docker. We use docker to deploy individual nodes in individual docker containers.

Step 3 – Setup Corda CLI tool

- We have downloaded docker and Java 11. This means we are ready to start up individual nodes on individual docker containers. But how do we generate the correct docker configuration to start a node? How do we deploy a node onto a docker container? How do we start the complete network? How do we install CPB's onto individual nodes?. All of this and much more is taken care of by the Corda CLI tool. You can either perform a manual installation in which case download the tar or zip folder and add the path to the bin/ directory to your PATH.

- Alternatively, use the below command to automate these manual steps. This script downloads the Corda CLI and adds it to the path:

```
curl "https://download.corda.net/corda-cli/1.0.0-DevPreview/get-corda-cli.sh" |  
bash
```

- Make sure the setup is correct by using the below command in your terminal:

```
corda-cli -version
```

Corda CLI

```
Version: 1.0.0-DevPreview-1.0
```

Step 4 – Setup CorDapp builder tool

- To perform a manual installation, you download the [tar](#) or [zip](#) folder and add the path to the bin/ directory to your PATH.
- Make sure the setup is correct by using the below command in your terminal:

```
cordapp-builder --version
```

```
cordapp-builder
```

```
version: 5.0.0-DevPreview-1.0
```

Step 5 – Setup Corda Node CLI

- This tool is used to interact with your node, and you can use the endpoint command to hit a particular flow. You can even use either Swagger UI or the Postman to hit the exposed APIs or to call a flow. To perform a manual installation, you download the tar or zip folder and add the path to the bin/ directory to your PATH.

Building the Corda Network

Creating Base Node:

- The network builder uses a set of nodes as the base for all other operations. A node is anything that satisfies the following layout:
 - node.conf
 - corda.jar
 - cordapps/
- An easy way to build a valid set of nodes is by running `deployNodes`. In this document, we will be using the output of running `deployNodes` for the Java samples repository:
 - `git clone https://github.com/corda/samples-java`
 - `cd samples-java/Basic/cordapp-example`
 - `./gradlew clean workflows-java:deployNodes`

Building a network via the command line

Starting the nodes

- `cd workflows-java/build/nodes`
 - `java -jar <path/to/corda-tools-network-builder.jar> -d .`
- If you run `docker ps` to see the running containers, the following output should be displayed:

Container ID	Image	Command	Created	Status	Ports
406868b4ba69	node-partyc:corda-network	"run-corda"	17 seconds ago	Up 16 seconds	0.0.0.0:32902->10003/tcp, 0.0.0.0:32895->10005/tcp, 0.0.0.0:32898->10020/tcp, 0.0.0.0:32900->12222/tcp partyc0
4546a2fa8de7	node-partyb:corda-network	"run-corda"	17 seconds ago	Up 17 seconds	0.0.0.0:32896->10003/tcp, 0.0.0.0:32899->10005/tcp, 0.0.0.0:32901->10020/tcp, 0.0.0.0:32903->12222/tcp partyb0
c8c44c515bdb	node-partya:corda-network	"run-corda"	17 seconds ago	Up 17 seconds	0.0.0.0:32894->10003/tcp, 0.0.0.0:32897->10005/tcp, 0.0.0.0:32892->10020/tcp, 0.0.0.0:32893->12222/tcp party0
cf7ab689f493	node-notary:corda-network	"run-corda"	30 seconds ago	Up 31 seconds	0.0.0.0:32888->10003/tcp, 0.0.0.0:32889->10005/tcp, 0.0.0.0:32890->10020/tcp, 0.0.0.0:32891->12222/tcp

					notary0
--	--	--	--	--	---------

- Depending on you machine performance, even after all containers are reported as running, the underlyingCorda nodes may be still starting and SSHing to a node may be not available immediately.

Quickstart Remote Azure

- `cd kotlin-source/build/nodes`
- `java -jar <path/to/corda-tools-network-builder.jar> -b AZURE -d.`

Interacting with the nodes

- You can interact with the nodes by SSHing into them on the port that is mapped to 12222. For example, to SSH into theparty0 node, you would run:

```
ssh
user1@localhost -p
32893Password
authentication
Password:
```

Welcome to the Corda interactive shell.

Useful commands include 'help' to see what is available, and 'bye' to shut down the node.

```
>>> run
networkMapSnaps
hot[

{ "addresses" : [ "party0:10020" ], "legalIdentitiesAndCerts" : [ "O=PartyA, L=London, C=GB" ], "platformVersion" : 6, "serial" :
1532701330613 },

{ "addresses" : [ "notary0:10020" ], "legalIdentitiesAndCerts" : [ "O=Notary, L=London, C=GB" ], "platformVersion" : 6, "serial" :
1532701305115 },

{ "addresses" : [ "partyc0:10020" ], "legalIdentitiesAndCerts" : [ "O=PartyC, L=Paris, C=FR" ], "platformVersion" : 6, "serial" :
1532701331608 },

{ "addresses" : [ "partyb0:10020" ], "legalIdentitiesAndCerts" : [ "O=PartyB, L=New York, C=US" ], "platformVersion" : 6, "serial"
: 1532701330118 }

]
```

```
>>>
```

- You can also run a flow from cordapp-example: `flow start com.example.flow.ExampleFlow$Initiator iouValue: 20,otherParty: "PartyB"`
- To verify it, connect into the partyb0 node and run `run vaultQuery contractStateType: "com.example.state.IOUState"`.
- The partyb0 vault should contain IOUState.

Adding additional nodes

- It is possible to add additional nodes to the network by reusing the nodes you built earlier. For example, to add a node by reusing the existing PartyA node, you would run:
- `java -jar <path/to/corda-tools-network-builder.jar> --add "PartyA=O=PartyZ,L=London,C=GB"`
- To confirm the node has been started correctly, run the following in the previously connected SSH session:

```
Tue Jul 17 15:47:14 GMT 2018>>>
```

```
run networkMapSnapshot[
```

```
{ "addresses" : [ "partyA0:10020" ], "legalIdentitiesAndCerts" : [ "O=PartyA, L=London, C=GB" ], "platformVersion" : 6, "serial" : 1532701330613 },
```

```
{ "addresses" : [ "notary0:10020" ], "legalIdentitiesAndCerts" : [ "O=Notary, L=London, C=GB" ], "platformVersion" : 6, "serial" : 1532701305115 },
```

```
{ "addresses" : [ "partyc0:10020" ], "legalIdentitiesAndCerts" : [ "O=PartyC, L=Paris, C=FR" ], "platformVersion" : 6, "serial" : 1532701331608 },
```

```
{ "addresses" : [ "partyb0:10020" ], "legalIdentitiesAndCerts" : [ "O=PartyB, L=New York, C=US" ], "platformVersion" : 6, "serial" : 1532701330118 },
```

```
{ "addresses" : [ "partyA1:10020" ], "legalIdentitiesAndCerts" : [ "O=PartyZ, L=London, C=GB" ], "platformVersion" : 6, "serial" : 1532701630861 }
```

```
]
```

Shutting down the nodes

- Run `docker kill $(docker ps -q)` to kill all running Docker processes.

Rubrics for Evaluation:

Understanding of Problem (3 Marks)	Capability of writing program (8 Marks) / Understanding of Basic content in case of the Study Practical	Documentation (3 Marks)	Knowledge and awareness related to the topic (3 Marks)	Enthusiasm/ audience awareness (3 Marks)	Total (20)