Group C

Assignment no:1

Title of the Assignment: Installation of MetaMask and study spending Ether per transaction

Objective of the Assignment: Students should be able to learn new technology such as metamask.Its application and implementations

Prerequisite:

1. Basic knowledge of cryptocurrency
2. Basic knowledge of distributed computing concept
3. Working of blockchain

Introduction to Blockchain

- Blockchain can be described as a data structure that holds transactional records and while ensuring security, transparency, and decentralization. You can also think of it as a chain or records stored in the forms of blocks which are controlled by no single authority.
- A blockchain is a distributed ledger that is completely open to any and everyone onthe network. Once an information is stored on a blockchain, it is extremely difficult to change or alter it.
- Each transaction on a blockchain is secured with a digital signature that proves its authenticity. Due to the use of encryption and digital signatures, the data stored on the blockchain is tamper-proof and cannot be changed.
- Blockchain technology allows all the network participants to reach an agreement, commonly known as consensus. All the data stored on a blockchain is recorded digitally and has a common history which is available for all the network participants. This way, the chances of any fraudulent activity or duplication of transactions is eliminated without the need of a third-party.

Blockchain Features

The following features make the revolutionary technology of blockchain stand out:

- **Decentralized**

  Blockchains are decentralized in nature meaning that no single person or group holds the authority of the overall network. While everybody in the network has the copy of the distributed ledger with them, no one can modify it on his or her own. This unique feature of blockchain allows transparency and security while giving power to the users.

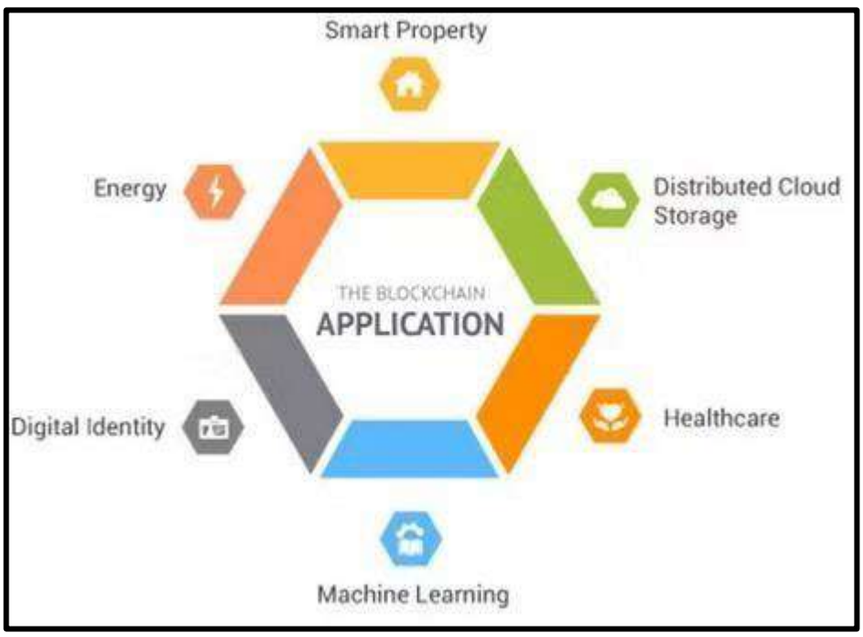
- **Immutable**

  The immutability property of a blockchain refers to the fact that any data once written on the blockchain cannot be changed. To understand immutability, consider sending email as an example. Once you send an email to a bunch of people, you cannot take it back. In order to find a way around, you'll have to ask all the recipients to delete your

email which is pretty tedious. This is how immutability works.

- **Tamper-Proof**
  With the property of immutability embedded in blockchains, it becomes easier to detect tampering of any data. Blockchains are considered tamper-proof as any change in even one single block can be detected and addressed smoothly. There are two key ways of detecting tampering namely, hashes and blocks.

Popular Applications of Blockchain Technology



Benefits of Blockchain Technology:

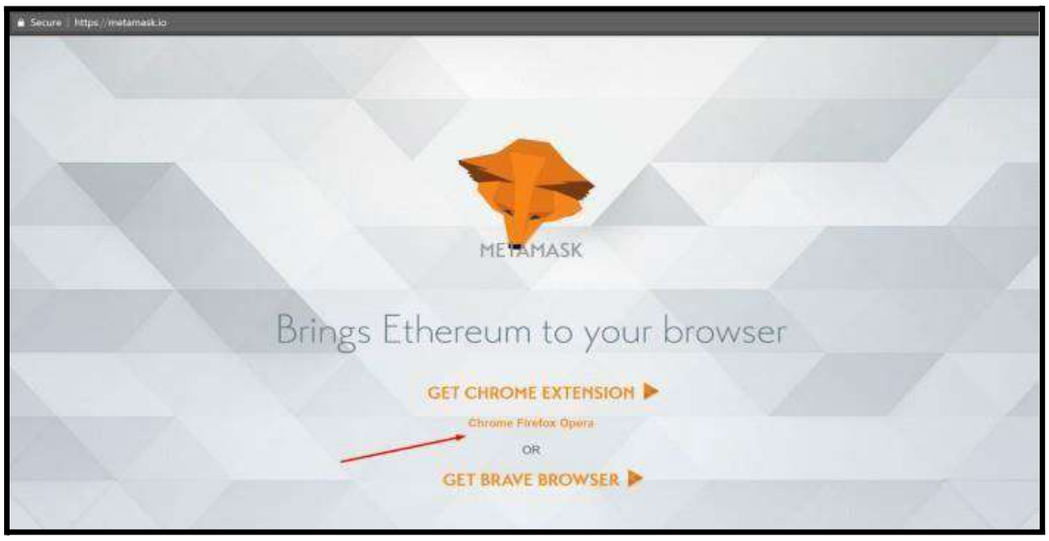
- **Time-saving:** No central Authority verification needed for settlements making the process faster and cheaper.
- **Cost-saving:** A Blockchain network reduces expenses in several ways. No need for third-party verification. Participants can share assets directly. Intermediaries are reduced. Transaction efforts are minimized as every participant has a copy of shared ledger.
- **Tighter security:** No one can temper with Blockchain Data as it is shared among millions of participants. The system is safe against cybercrimes and Fraud.
- In finance market trading, Fibonacci retracement levels are widely used in technical analysis.

How to use MetaMask: A step by step guide

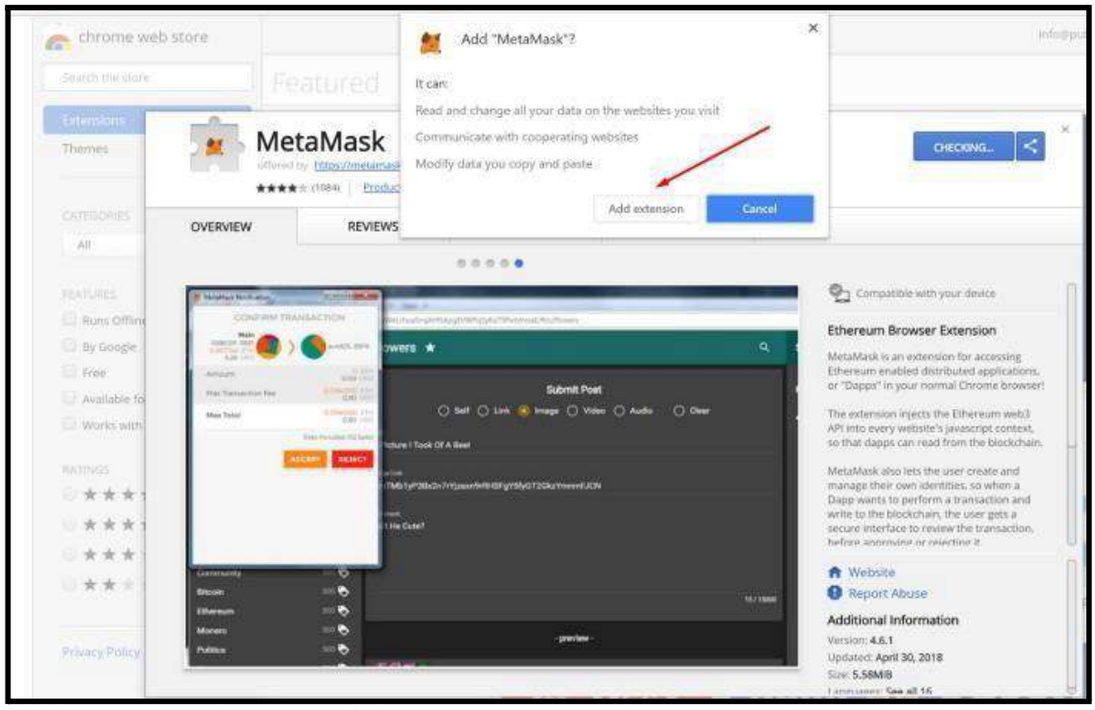
MetaMask is one of the most popular browser extensions that serves as a way of storing your Ethereum and other ERC-20 Tokens. The extension is free and secure, allowing web applications to read and interact with Ethereum's blockchain.

Step 1. Install MetaMask on your browser.

To create a new wallet, you have to install the extension first. Depending on your browser, there are different marketplaces to find it. Most browsers have MetaMask on their stores, so it's not that hard to see it, but either way, here they are Chrome, Firefox, and Opera.



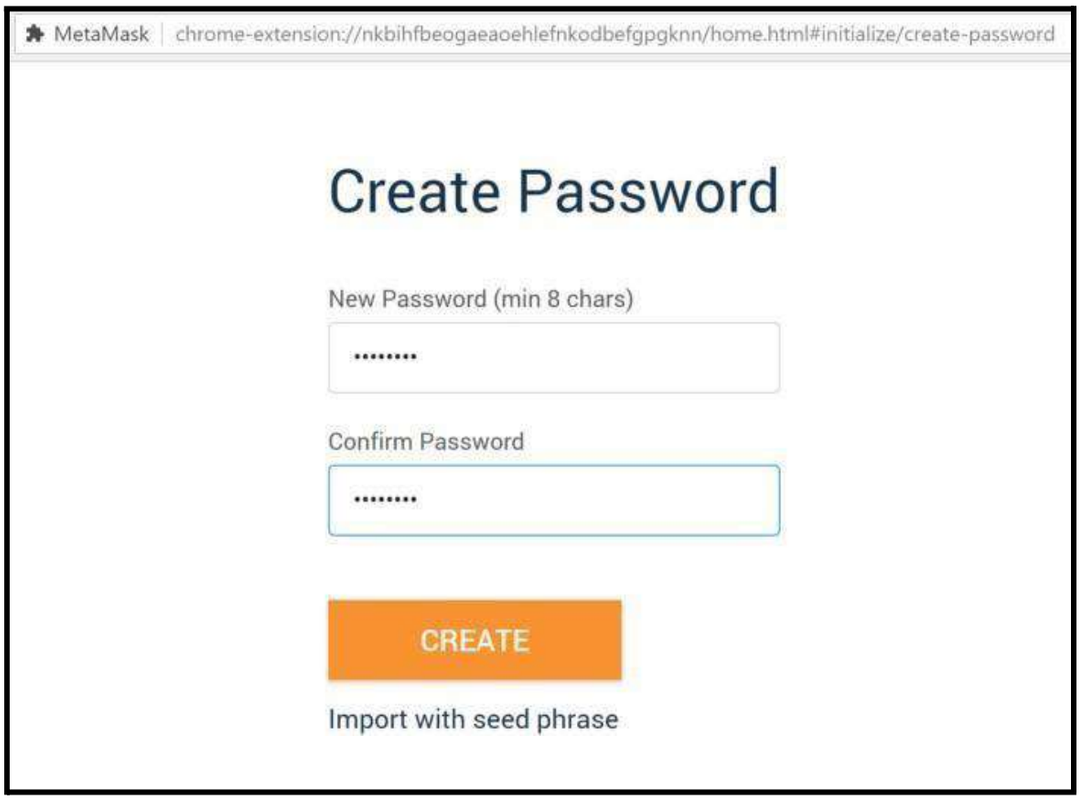
- Click on **Install MetaMask** as a Google Chrome extension.
- Click **Add to Chrome**.
- Click **Add Extension**.



nd it's as easy as that to install the extension on your browser, continue reading the next step to figure out how to create an account.

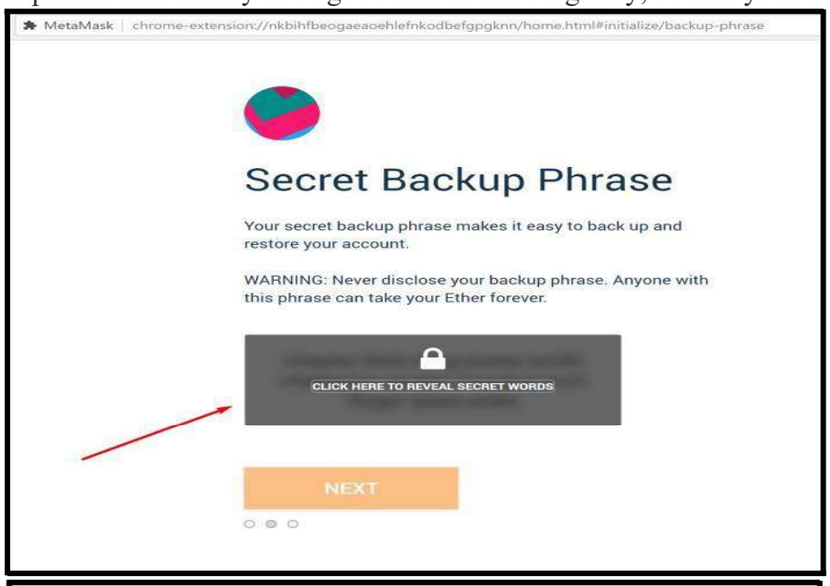
Step 2. Create an account.

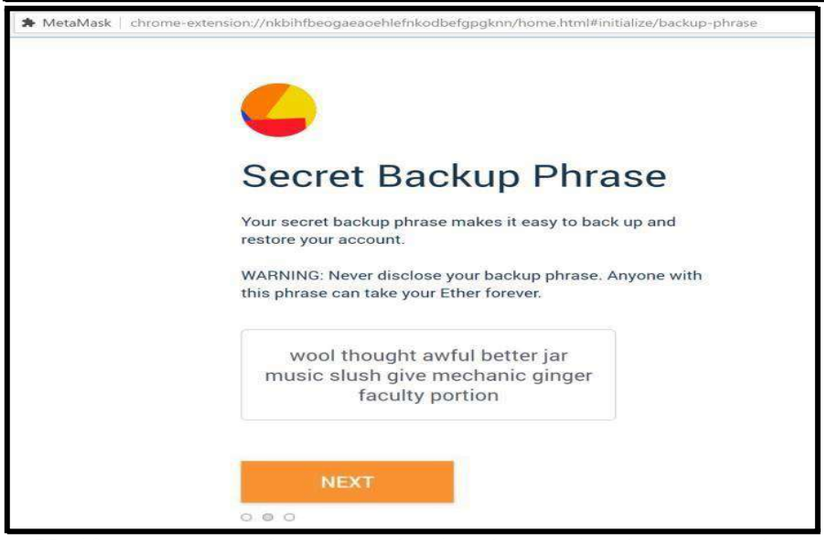
- Click on the extension icon in the upper right corner to open MetaMask.
- To install the latest version and be up to date, **click Try it now**.
- **Click Continue**.
- You will be prompted to create a new password. **Click Create**.



- Proceed by **clicking Next** and accept the Terms of Use.

Click Reveal Secret Words. There you will see a 12 words seed phrase. This is really important and usually not a good idea to store digitally, so take your time and write it down





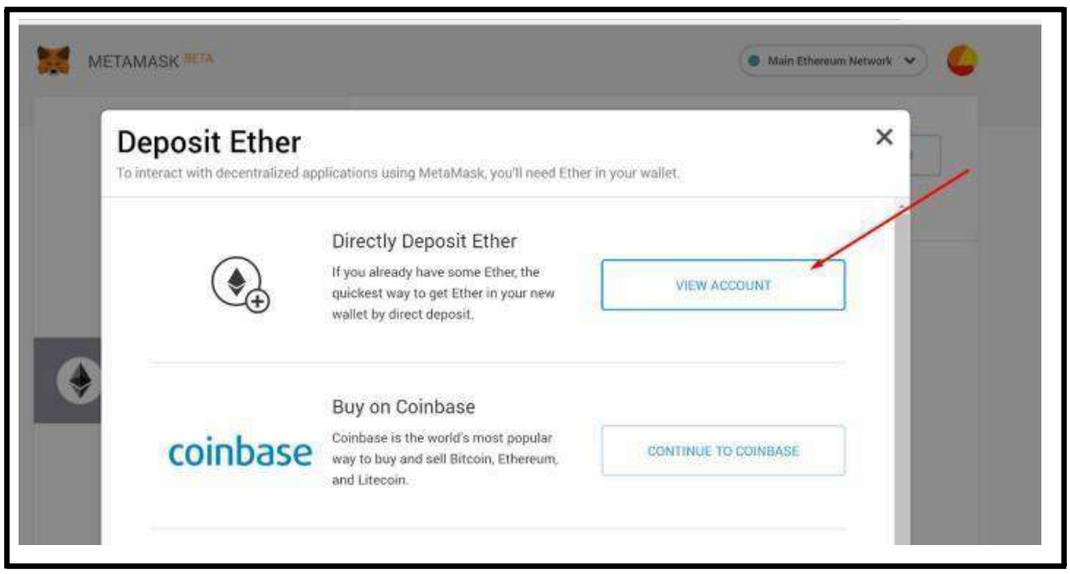
- Verify your secret phrase by selecting the previously generated phrase in order. **Click Confirm**.

And that's it; now you have created your MetaMask account successfully. A new Ethereum wallet

address has just been created for you. It's waiting for you to deposit funds, and if you want to learn how to do that, look at the next step below.

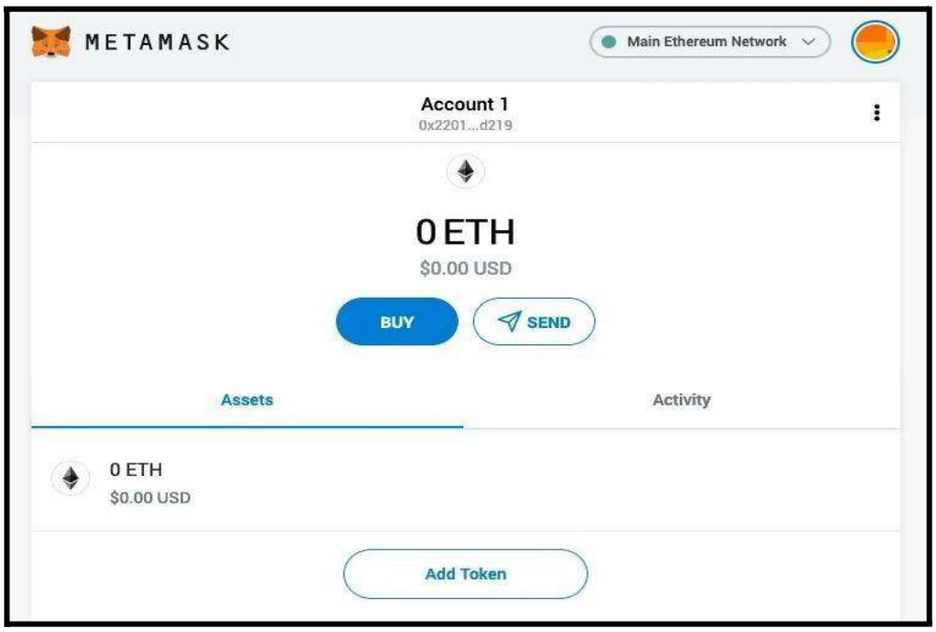
Step 3. Depositing funds.

- Click on **View Account**.



You can now see your public address and share it with other people. There are some methods to buy coins offered by MetaMask, but you can do it differently as well; you just need your address.

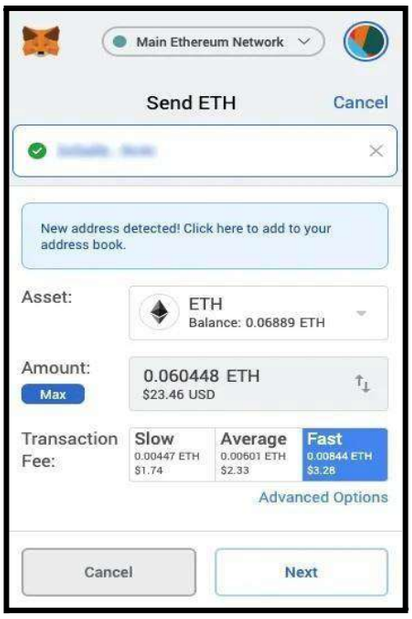
If you ever get logged out, you'll be able to log back in again by clicking the MetaMask icon, which will have been added to your web browser (usually found next to the URL bar).



You can now access your list of assets in the 'Assets' tab and view your transaction history in the 'Activity' tab.

- Sending crypto is as simple as clicking the 'Send' button, entering the recipient address and amount to send, and selecting a transaction fee. You can also manually adjust the transaction fee using the 'Advanced Options' button, using information from ETH Gas Station or similar platforms to choose a more acceptable gas price.

- After clicking ‘Next’, you will then be able to either confirm or reject the transaction on the subsequent page.



- To use MetaMask to interact with a dapp or smart contract, you’ll usually need to find a ‘Connect to Wallet’ button or similar element on the platform you are trying to use. After clicking this, you should then see a prompt asking whether you want to let the dapp connect to your wallet.

What advantages does MetaMask have?

- **Popular** - It is commonly used, so users only need one plugin to access a wide range of dapps.
- **Simple** - Instead of managing private keys, users just need to remember a list of words, and transactions are signed on their behalf.
- **Saves space** - Users don’t have to download the Ethereum blockchain, as MetaMask sends requests to nodes outside of the user’s computer.
- **Integrated** - Dapps are designed to work with MetaMask, so it becomes much easier to send Ether in and out.

Conclusion- In this way we have explored Concept Blockchain and metamat wallet for transaction of digital currency

Assignment Question

1. **What Are the Different Types of Blockchain Technology?**
2. **What Are the Key Features/Properties of Blockchain?**
3. **What Type of Records You Can Keep in A Blockchain?**
4. **What is the difference between Ethereum and Bitcoin?**
5. **What are Merkle Trees? Explain their concept.**
6. **What is Double Spending in transaction operation**
7. **Give real-life use cases of blockchain.**

Assignment no:2

Title of the Assignment: Create your own wallet using Metamask for crypto transactions

Objective of the Assignment: Students should be able to learn about cryptocurrencies and learn how transaction done by using different digital currency

Prerequisite:

1. Basic knowledge of cryptocurrency
2. Basic knowledge of distributed computing concept
3. Working of blockchain

Contents for Theory:

1. **Cryptocurrency**
2. **Transaction Wallets**
3. **Ether transaction**

Introduction to Cryptocurrency

- Cryptocurrency is a digital payment system that doesn't rely on banks to verify transactions. It's a peer-to-peer system that can enable anyone anywhere to send and receive payments. Instead of being physical money carried around and exchanged in the real world, cryptocurrency payments exist purely as digital entries to an online database describing specific transactions. When you transfer cryptocurrency funds, the transactions are recorded in a public ledger. Cryptocurrency is stored in digital wallets.
- Cryptocurrency received its name because it uses encryption to verify transactions. This means advanced coding is involved in storing and transmitting cryptocurrency data between wallets and to public ledgers. The aim of encryption is to provide security and safety.
- The first cryptocurrency was Bitcoin, which was founded in 2009 and remains the best known today. Much of the interest in cryptocurrencies is to trade for profit, with speculatorsat times driving prices skyward.

How does cryptocurrency work?

- Cryptocurrencies run on a distributed public ledger called blockchain, a record of all transactions updated and held by currency holders.
- Units of cryptocurrency are created through a process called mining, which involves using computer power to solve complicated mathematical problems that generate coins. Users can also buy the currencies from brokers, then store and spend them using cryptographic wallets.
- If you own cryptocurrency, you don't own anything tangible. What you own is a key that allows you to move a record or a unit of measure from one person to another without a trusted third party.
- Although Bitcoin has been around since 2009, cryptocurrencies and applications of blockchain technology are still emerging in financial terms, and more uses are expected in the future. Transactions including bonds, stocks, and other financial assets could eventually be traded using the technology.

Cryptocurrency examples

There are thousands of cryptocurrencies. Some of the best known include:

- **Bitcoin:**

  Founded in 2009, Bitcoin was the first cryptocurrency and is still the most commonly traded. The currency was developed by Satoshi Nakamoto – widely believed to be a pseudonym for an individual or group of people whose precise identity remains unknown.

- **Ethereum:**

  Developed in 2015, Ethereum is a blockchain platform with its own cryptocurrency, called Ether (ETH) or Ethereum. It is the most popular cryptocurrency after Bitcoin.

- **Litecoin:**

  This currency is most similar to bitcoin but has moved more quickly to develop new innovations, including faster payments and processes to allow more transactions.

- **Ripple:**
  Ripple is a distributed ledger system that was founded in 2012. Ripple can be used to track different kinds of transactions, not just cryptocurrency. The company behind it has worked with various banks and financial institutions.
- Non-Bitcoin cryptocurrencies are collectively known as “altcoins” to distinguish them from the original.

How to store cryptocurrency

- Once you have purchased cryptocurrency, you need to store it safely to protect it from hacks or theft. Usually, cryptocurrency is stored in crypto wallets, which are physical devices or online software used to store the private keys to your cryptocurrencies securely. Some exchanges provide wallet services, making it easy for you to store directly through the platform. However, not all exchanges or brokers automatically provide wallet services for you.
- There are different wallet providers to choose from. The terms “hot wallet” and “cold wallet” are used:
- **Hot wallet storage:** "hot wallets" refer to crypto storage that uses online software to protect the private keys to your assets.
- **Cold wallet storage:** Unlike hot wallets, cold wallets (also known as hardware wallets) rely on offline electronic devices to securely store your private keys.

Conclusion- In this way we have explored Concept Cryptocurrency and learn how transactions are done using digital currency

Assignment Question

1. **What is Bitcoin?**
2. **What Are the biggest Four common cryptocurrency scams**
3. **Explain How safe are money e-transfers?**
4. **What is cryptojacking and how does it work?**

Assignment no:3

Title of the Assignment: Write a smart contract on a test network, for Bank account of a customer for following operations:

- Deposit money
- Withdraw Money
- Show balance

Objective of the Assignment: Students should be able to learn new technology such as metamask.Its application and implementations

Prerequisite:

1. Basic knowledge of cryptocurrency
2. Basic knowledge of distributed computing concept
3. Working of blockchain.

Contents for Theory:

The contract will allow deposits from any account, and can be trusted to allow withdrawals only by accounts that have sufficient funds to cover the requested withdrawal.

This post assumes that you are comfortable with the ether-handling concepts introduced in our post, Writing a Contract That Handles Ether.

That post demonstrated how to restrict ether withdrawals to an "owner's" account. It did this by persistently storing the owner account's address, and then comparing it to the msg.sender value for any withdrawal attempt. Here's a slightly simplified version of that smart contract, which allows anybody to deposit money, but only allows the owner to make withdrawals:

```
pragma solidity ^0.4.19;

contract TipJar {

    address owner;    // current owner of the contract

    function TipJar() public {
        owner = msg.sender;
    }

    function withdraw() public {
        require(owner == msg.sender);
        msg.sender.transfer(address(this).balance);
    }

    function deposit(uint256 amount) public payable {
        require(msg.value == amount);
    }

    function getBalance() public view returns (uint256) {
        return address(this).balance;
    }
}
```

I am going to generalize this contract to keep track of ether deposits based on the account address of the depositor, and then only allow that same account to make withdrawals of that ether. To do this, we need a way keep track of account balances for each depositing account–a mapping from accounts to balances. Fortunately, Solidity provides a ready-made mapping data type that can map account addresses to integers,

which will make this bookkeeping job quite simple. (This mapping structure is much more general key/value mapping than just addresses to integers, but that's all we need here.)
Here's the code to accept deposits and track account balances:

```
pragma solidity ^0.4.19;

contract Bank {

    mapping(address => uint256) public balanceOf;  // balances, indexed by addresses

    function deposit(uint256 amount) public payable {
        require(msg.value == amount);

        balanceOf[msg.sender] += amount;     // adjust the account's balance
    }
}
```

Here are the new concepts in the code above:

- mapping(address => uint256) public balanceOf; declares a persistent public variable, balanceOf, that is a mapping from account addresses to 256-bit unsigned integers. Those integers will represent the current balance of ether stored by the contract on behalf of the corresponding address.
- Mappings can be indexed just like arrays/lists/dictionaries/tables in most modern programming languages.
- The value of a missing mapping value is 0. Therefore, we can trust that the beginning balance for all account addresses will effectively be zero prior to the first deposit.

It's important to note that balanceOf keeps track of the ether balances assigned to each account, but it does not actually move any ether anywhere. The bank contract's ether balance is the sum of all the balances of all accounts—only balanceOf tracks how much of that is assigned to each account.
Note also that this contract doesn't need a constructor. There is no persistent state to initialize other thanthe balanceOf mapping, which already provides default values of 0.
Given the balanceOf mapping from account addresses to ether amounts, the remaining code for a fully-functional bank contract is pretty small. I'll simply add a withdrawal function:

bank.sol

```
pragma solidity ^0.4.19;

contract Bank {

    mapping(address => uint256) public balanceOf;  // balances, indexed by addresses

    function deposit(uint256 amount) public payable {
        require(msg.value == amount);
        balanceOf[msg.sender] += amount;     // adjust the account's balance
    }

    function withdraw(uint256 amount) public {
        require(amount <= balanceOf[msg.sender]);
        balanceOf[msg.sender] -= amount;
        msg.sender.transfer(amount);
    }
}
```

The code above demonstrates the following:

- The require(amount <= balances[msg.sender]) checks to make sure the sender has sufficient funds to cover the requested withdrawal. If not, then the transaction aborts without making any state changes or ether transfers.
- The balanceOf mapping must be updated to reflect the lowered residual amount after the withdrawal.
- The funds must be sent to the sender requesting the withdrawal.

In the withdraw() function above, it is very important to adjust balanceOf[msg.sender] **before** transferring ether to avoid an exploitable vulnerability. The reason is specific to smart contracts and the fact that a transfer to a smart contract executes code in that smart contract. (The essentials of Ethereum transactions are discussed in How Ethereum Transactions Work.)
Now, suppose that the code in withdraw() did not adjust balanceOf[msg.sender] before making the transfer *and* suppose that msg.sender was a malicious smart contract. Upon receiving the transfer—handled by msg.sender's fallback function—that malicious contract could initiate *another* withdrawal from the banking contract. When the banking contract handles this second withdrawal request, it would have already transferred ether for the original withdrawal, but it would not have an updated balance, so it would allow this second withdrawal!
This vulnerability is called a "reentrancy" bug because it happens when a smart contract invokes code in a different smart contract that then calls back into the original, thereby reentering the exploitable contract. For this reason, it's essential to always make sure a contract's internal state is fully updated before it potentially invokes code in another smart contract. (And, it's essential to remember that every transfer to a smart contract executes that contract's code.)

To avoid this sort of reentrancy bug, follow the "Checks-Effects-Interactions pattern" as described in the Solidity documentation. The `withdraw()` function above is an example of implementing this pattern

Assignment no:4

Title of the Assignment: Write a survey report on types of Blockchains and its real time use cases.

Objective of the Assignment: Students should be able to learn new technology such as metamask.Its application and implementations

Prerequisite:

1. Basic knowledge of cryptocurrency
2. Basic knowledge of distributed computing concept
3. Working of blockchain

Contents for Theory:

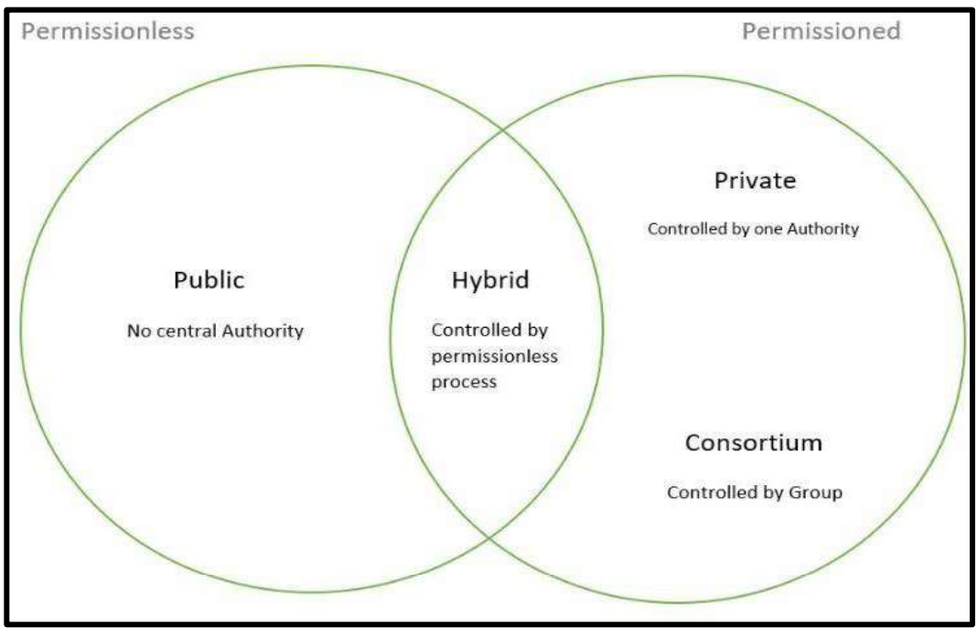
There are 4 types of blockchain:

Public Blockchain.

Private Blockchain.

Hybrid Blockchain.

Consortium Blockchain



---

1. Public Blockchain

These blockchains are completely open to following the idea of decentralization. They don't have any restrictions, anyone having a computer and internet can participate in the network.

As the name is public this blockchain is open to the public, which means it is not owned by anyone.
Anyone having internet and a computer with good hardware can participate in this public blockchain.
All the computer in the network hold the copy of other nodes or block present in the network
In this public blockchain, we can also perform verification of transactions or records
Advantages:

Trustable: There are algorithms to detect no fraud. Participants need not worry about the other nodes in the network
Secure: This blockchain is large in size as it is open to the public. In a large size, there is greater distribution of records
Anonymous Nature: It is a secure platform to make your transaction properly at the same time, you are not required to reveal your name and identity in order to participate.
Decentralized: There is no single platform that maintains the network, instead every user has a copy of the ledger.
Disadvantages:

Processing: The rate of the transaction process is very slow, due to its large size. Verification of each node is a very time-consuming process.
Energy Consumption: Proof of work is high energy-consuming. It requires good computer hardware to participate in the network
Acceptance: No central authority is there so governments are facing the issue to implement the technology faster.
Use Cases: Public Blockchain is secured with proof of work or proof of stake they can be used to displace traditional financial systems. The more advanced side of this blockchain is the smart contract that enabled this blockchain to support decentralization. Examples of public blockchain are Bitcoin, Ethereum.

2. Private Blockchain
These blockchains are not as decentralized as the public blockchain only selected nodes can participate in the process, making it more secure than the others.

These are not as open as a public blockchain.
They are open to some authorized users only.
These blockchains are operated in a closed network.
In this few people are allowed to participate in a network within a company/organization.
Advantages:

Speed: The rate of the transaction is high, due to its small size. Verification of each node is less time-consuming.
Scalability: We can modify the scalability. The size of the network can be decided manually.
Privacy: It has increased the level of privacy for confidentiality reasons as the businesses required.
Balanced: It is more balanced as only some user has the access to the transaction which improves the performance of the network.
Disadvantages:

Security- The number of nodes in this type is limited so chances of manipulation are there. These blockchains are more vulnerable.
Centralized- Trust building is one of the main disadvantages due to its central nature. Organizations can use this for malpractices.
Count- Since there are few nodes if nodes go offline the entire system of blockchain can be endangered.
Use Cases: With proper security and maintenance, this blockchain is a great asset to secure information without exposing it to the public eye. Therefore companies use them for internal auditing, voting, and asset management. An example of private blockchains is Hyperledger, Corda.

3. Hybrid Blockchain
It is the mixed content of the private and public blockchain, where some part is controlled by some organization and other makes are made visible as a public blockchain.

It is a combination of both public and private blockchain.
Permission-based and permissionless systems are used.
User access information via smart contracts
Even a primary entity owns a hybrid blockchain it cannot alter the transaction
Advantages:

Ecosystem: Most advantageous thing about this blockchain is its hybrid nature. It cannot be hacked as 51% of users don't have access to the network
Cost: Transactions are cheap as only a few nodes verify the transaction. All the nodes don't carry the verification hence less computational cost.
Architecture: It is highly customizable and still maintains integrity, security, and transparency.
Operations: It can choose the participants in the blockchain and decide which transaction can be made public.
Disadvantages: Efficiency: Not everyone is in the position to implement a hybrid Blockchain. The organization also faces

some difficulty in terms of efficiency in maintenance.
Transparency: There is a possibility that someone can hide information from the user. If someone wants to get access through a hybrid blockchain it depends on the organization whether they will give or not.
Ecosystem: Due to its closed ecosystem this blockchain lacks the incentives for network participation.
Use Case: It provides a greater solution to the health care industry, government, real estate, and financial companies. It provides a remedy where data is to be accessed publicly but needs to be shielded privately.
Examples of Hybrid Blockchain are Ripple network and XRP token.

4. Consortium Blockchain
It is a creative approach that solves the needs of the organization. This blockchain validates the transaction and also initiates or receives transactions.

Also known as Federated Blockchain.
This is an innovative method to solve the organization's needs.
Some part is public and some part is private.
In this type, more than one organization manages the blockchain.
Advantages:

Speed: A limited number of users make verification fast. The high speed makes this more usable for organizations.
Authority: Multiple organizations can take part and make it decentralized at every level. Decentralized authority, makes it more secure.
Privacy: The information of the checked blocks is unknown to the public view. but any member belonging to the blockchain can access it.
Flexible: There is much divergence in the flexibility of the blockchain. Since it is not a very large decision can be taken faster.
Disadvantages:

Approval: All the members approve the protocol making it less flexible. Since one or more organizations are involved there can be differences in the vision of interest.
Transparency: It can be hacked if the organization becomes corrupt. Organizations may hide information from the users.
Vulnerability: If few nodes are getting compromised there is a greater chance of vulnerability in this blockchain
Use Cases: It has high potential in businesses, banks, and other payment processors. Food tracking of the organizations frequently collaborates with their sectors making it a federated solution ideal for their use.
Examples of consortium Blockchain are Tendermint and Multichain.

Conclusion-In this way we have explored types of blockchain and its applications in real time

Assignment no:5

Title of the Assignment: Write a program to create a Business Network using Hyperledger.

Objective of the Assignment: Students should be able to learn hyperledger .Its application and implementations

Prerequisite:

1. Basic knowledge of cryptocurrency
2. Basic knowledge of distributed computing concept
3. Working of blockchain

Contents for Theory:

Hyperledger Composer is an extensive, open development toolset and framework to make developing blockchain applications easier. The primary goal is to accelerate time to value, and make it easier to integrate your blockchain applications with the existing business systems.

- You can use Composer to rapidly develop use cases and deploy a blockchain solution in days.
- Composer allows you to model your business network and integrate existing systems and data with your blockchain applications.
- Hyperledger Composer supports the existing Hyperledger Fabric blockchain infrastructure and runtime.
- Hyperleder Composer generate business network archive (bna) file which you can deploy on existing Hyperledger Fabric network

You can use Hyperledger Composer to model business network, containing your existing assets and the transactions related to them

Key Concepts of Hyperledger Composer

1. Blockchain State Storage: It stores all transaction that happens in your hyperledger composer application. It stores transaction in Hyperledger fabric network.

2. Connection Profiles: Connection Profiles to configuration JSON file which help composer to connect to Hyperledger Fabric. You can find Connection Profile JSON file in user's home directory.

3. Assets: Assets are tangible or intangible goods, services, or property, and are stored in registries. Assets can represent almost anything in a business network, for example, a house for sale, the sale listing, the land registry certificate for that house. Assets must have a unique identifier, but other than that, they can contain whatever properties you define.

4. Participants: Participants are members of a business network. They may own assets and submit transactions. Participant must have an identifier and can have any other properties.

5. Identities and ID cards: Participants can be associated with an identity. ID cards are a combination of an identity, a connection profile, and metadata. ID cards simplify the process of connecting to a business network.

6. Transactions: Transactions are the mechanism by which participants interact with assets. Transaction processing logic you can define in JavaScript and you can also emit event for transaction.

7. Queries: Queries are used to return data about the blockchain world-state. Queries are defined within a business network, and can include variable parameters for simple customisation. By using queries, data can be easily extracted from your blockchain network. Queries are sent by using the Hyperledger Composer API.

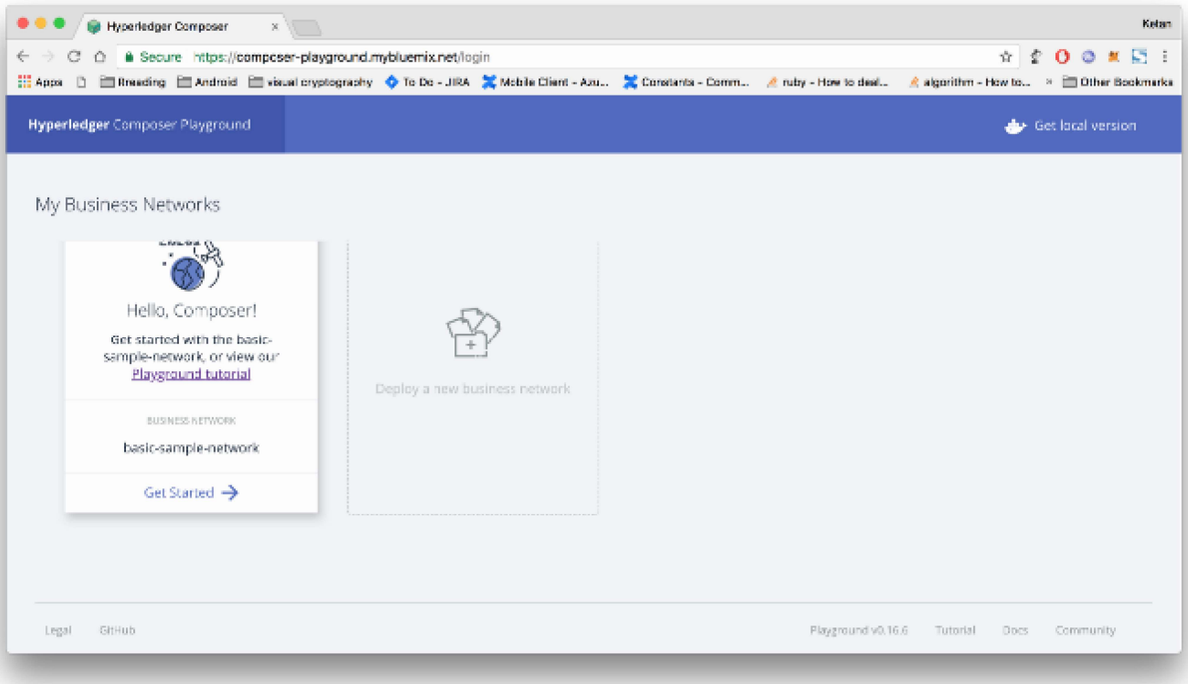
8. Events: Events are defined in the model file. Once events have been defined, they can be emitted by transaction processor functions to indicate to external systems that something of importance has happened to the ledger.

9. Access Control: Hyperledger is enterprise blockchain and access control is core feature of any business blockchain. Using Access Control rules you can define who can do what in Business networks. The access control language is rich enough to capture sophisticated conditions.

10. Historian registry: The historian is a specialised registry which records successful transactions, including the participants and identities that submitted them. The historian stores transactions as HistorianRecord assets, which are defined in the Hyperledger Composer system namespace.

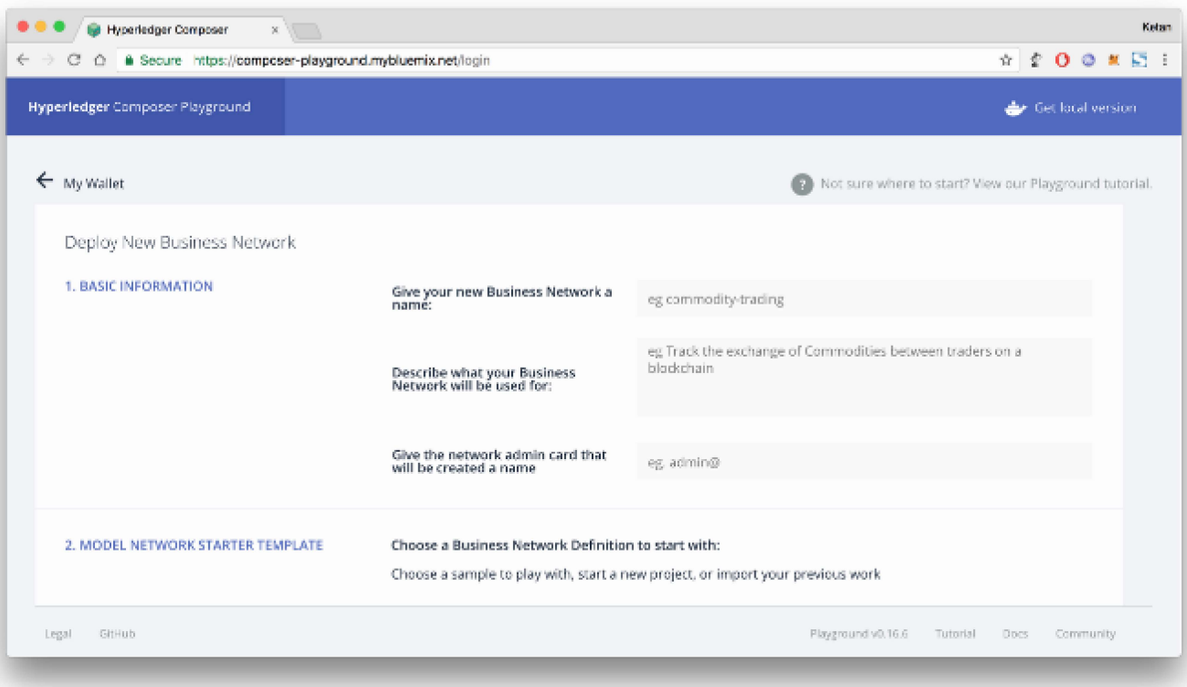
Let's create first Hyperledger Composer Application

Step 1: Start Hyperledger Composer Online version of Local. Click on Deploy a new business network

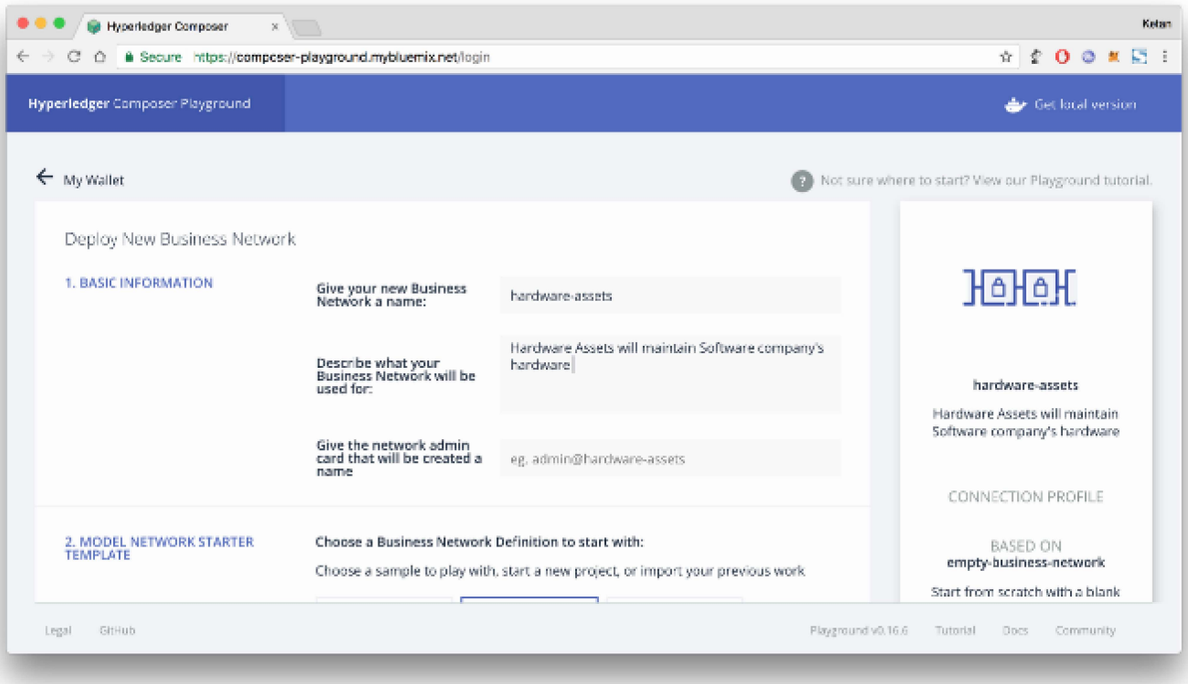


Hyperledger Composer Playground Online version

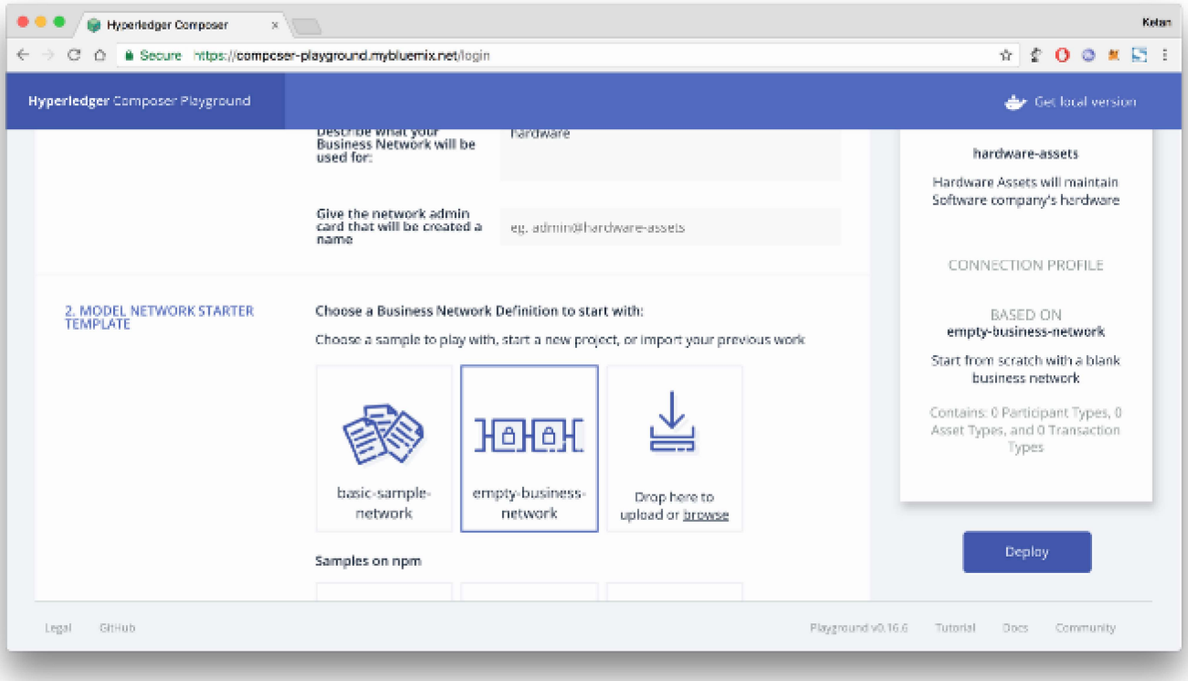
Step 2: Select empty business network



Step 3: Fill basic information, select empty business network and click “deploy” button from right pannel

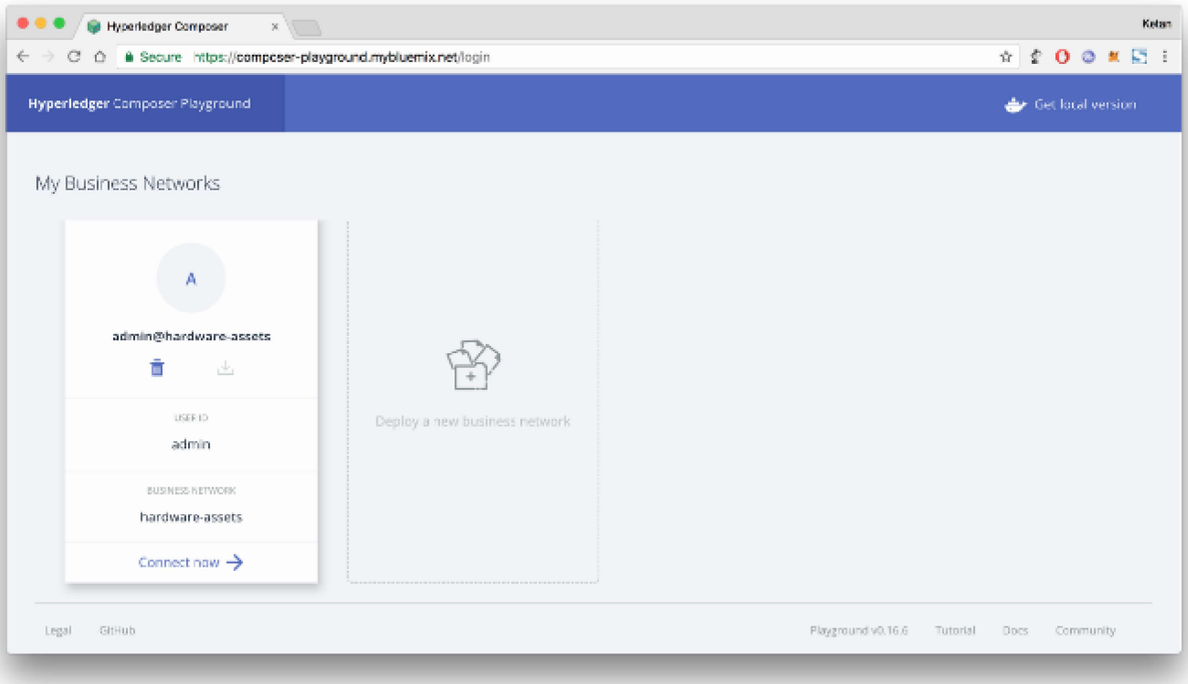


Fill basic information

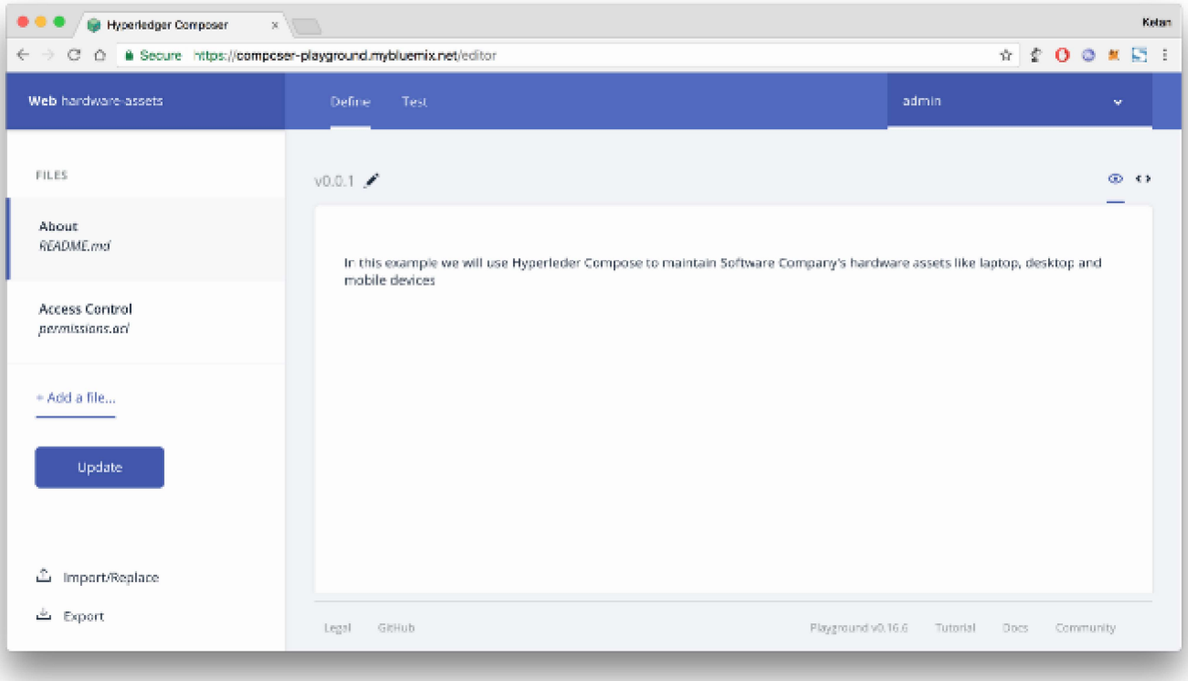


select empty business network

Step 4: Connect to “hardware-assets” business network that we have just deploied

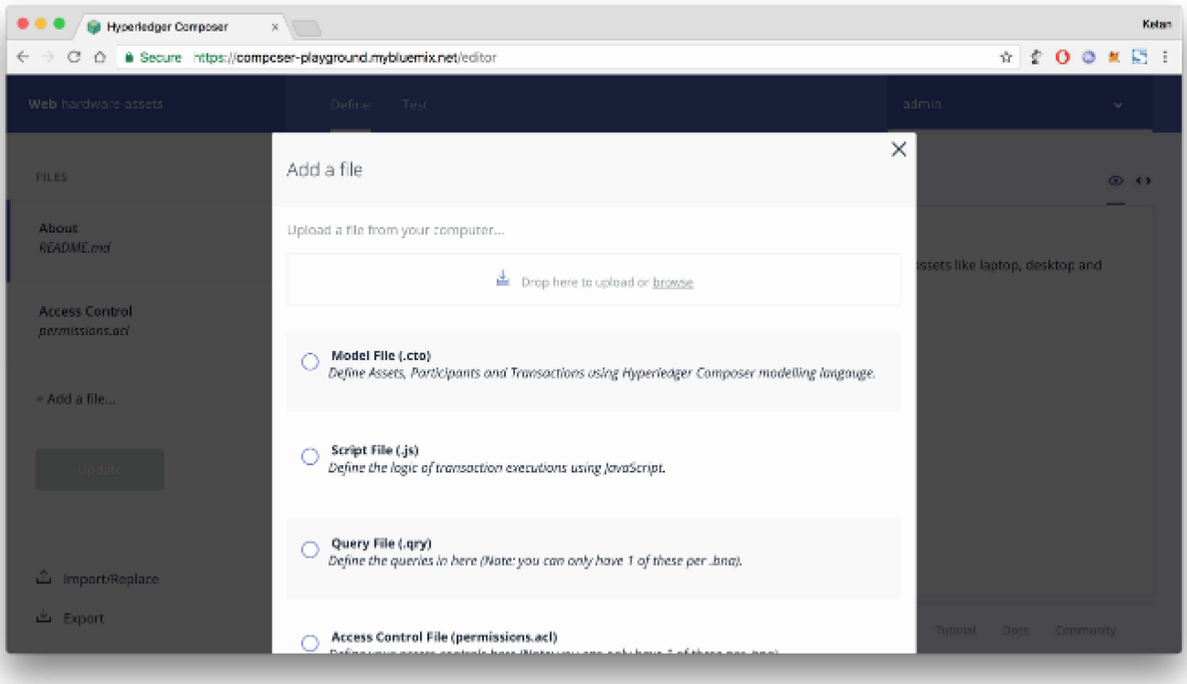


click on “connect now” button



Inside hardware-assets business network

Step 5: Click on “+Add a file...” from left panel and select “model file (.cto)”



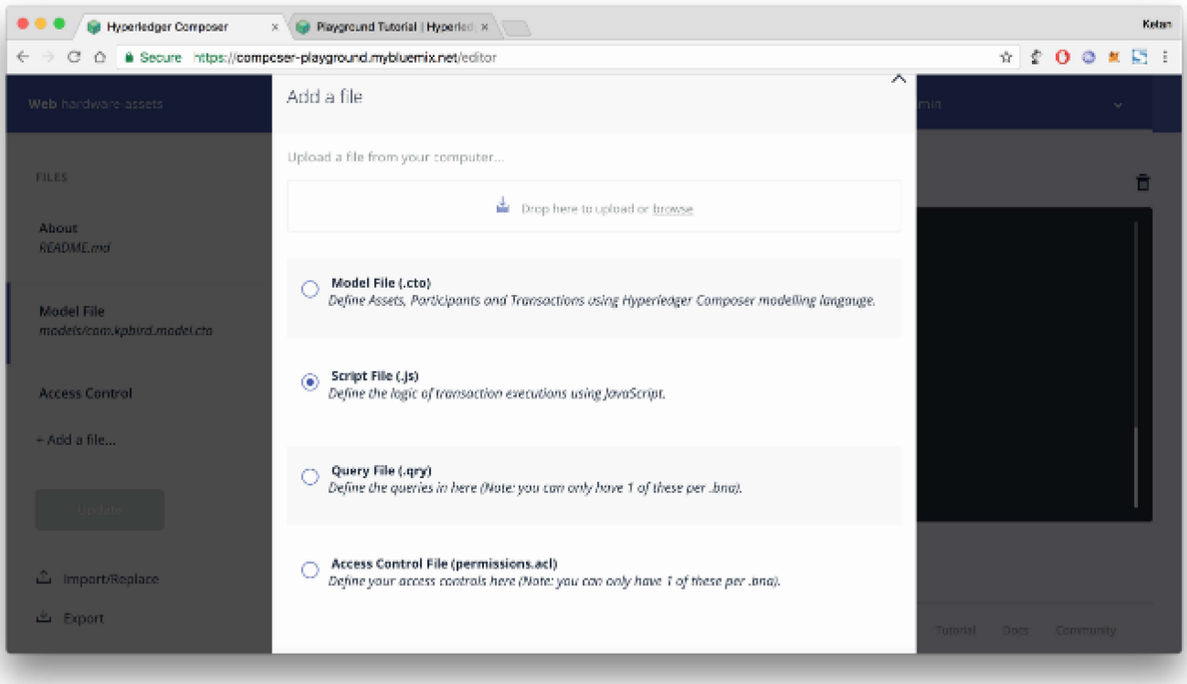
Write following code in model file. Model file contain asset in our case it's hardware, participant in our case participants are employee of organisation and transaction as Allocate hardware to employee. Each model has extra properties. Make sure your have proper and unique namespace. In this example I am using "com.kpbird" as namespace. You can access all models using this namespace i.e. com.kpbird.Hardware, com.kpbird.Employee

```
/**
 * Hardware model
 */namespace com.kpbirdasset Hardware identified by hardwareId {
 o String hardwareId
 o String name
 o String type
 o String description
 o Double quantity
 → Employee owner
}
participant Employee identified by employeeId {
 o String employeeId
 o String firstName
 o String lastName
}
transaction Allocate {
 → Hardware hardware
 → Employee newOwner
}
```

Hyperledger modeling language

reference: https://hyperledger.github.io/composer/reference/cto_language.html

Step 6: Click on "+Add a file..." from left panel and select "script file (*.js)"



Write following code in Script File. In Script we can define transaction processing logic. In our case we want to allocate hardware to the employee so, we will update owner of hardware. Make sure about annotation above functions @params and @transaction

```
/**
 * Track the trade of a commodity from one trader to another
 * @param {com.kpbird.Allocate} trade – the trade to be processed
 * @transaction
 */
function allocateHardware(allocate) {
 allocate.hardware.owner = allocate.newOwner;
 return getAssetRegistry('com.kpbird.Hardware')
 .then(function (assetRegistry) {
 return assetRegistry.update(allocate.hardware);
 });
}
```

Hyperledger Composer Script file reference: https://hyperledger.github.io/composer/reference/js_scripts.html

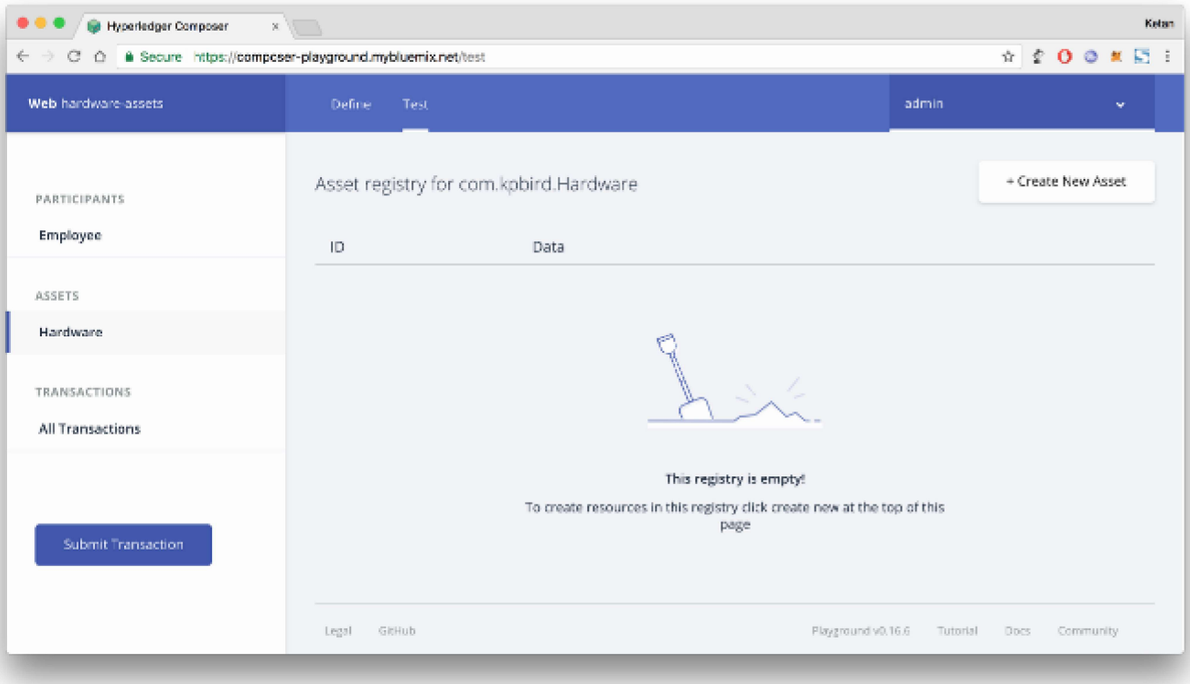
Step 7: permissions.acl file sample is already available, Add following code in permissions.acl file.

```
/**
 * New access control file
 */
 rule AllAccess {
 description: "AllAccess – grant everything to everybody."
 participant: "ANY"
 operation: ALL
 resource: "com.kpbird.**"
 action: ALLOW
 }rule SystemACL{
 description: "System ACL to permit all access"
 participant: "org.hyperledger.composer.system.Participant"
 operation: ALL
 resource: "org.hyperledger.composer.system.**"
 action: ALLOW
 }
```

Hyperledger Composer Access Control Language

reference: https://hyperledger.github.io/composer/reference/acl_language.html

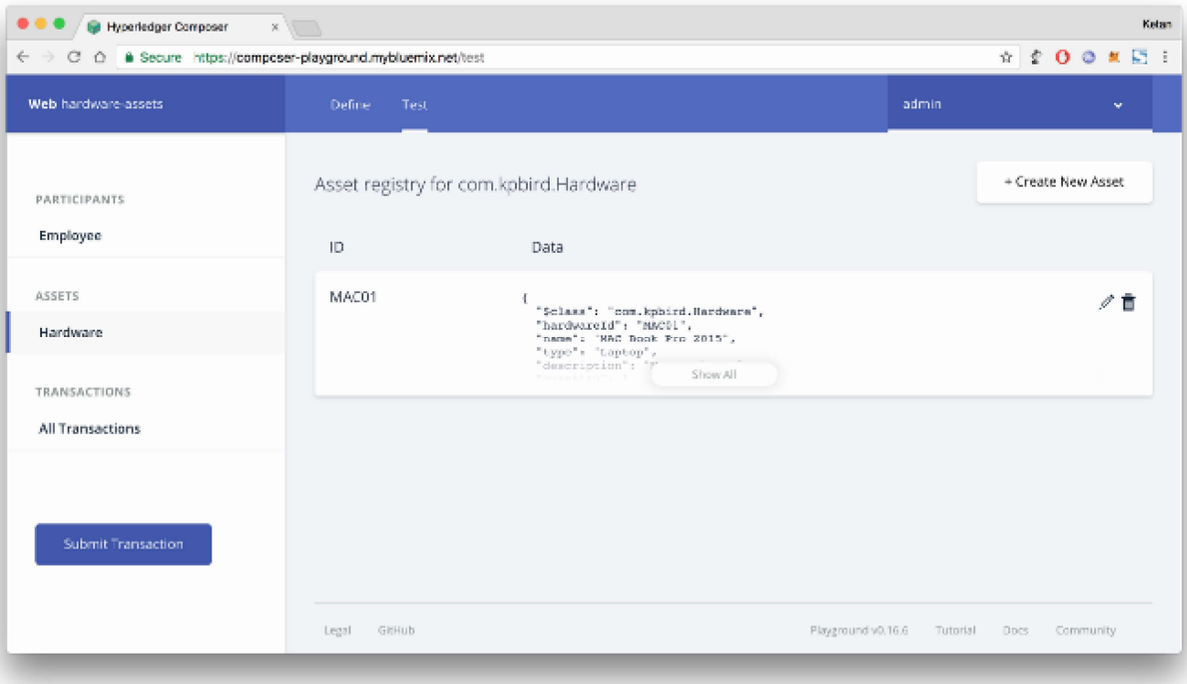
Step 8: Now, It's time to test our hardware assets business network. Hyperledger composer gives "Test" facility from composer panel it self. Click on "Test" tab from top panel



Test feature of Hyperledger Composer

Step 9: Create Assets. Click on "Hardware" from left panel and click "+ Create New Assets" from right top corner and add following code. We will create Employee#01 in next step. Click on "Create New" button

```
{
 "$class": "com.kpbird.Hardware",
 "hardwareId": "MAC01",
 "name": "MAC Book Pro 2015",
 "type": "Laptop",
 "description": "Mac Book Pro",
 "quantity": 1,
 "owner": "resource:com.kpbird.Employee#01"
}
```



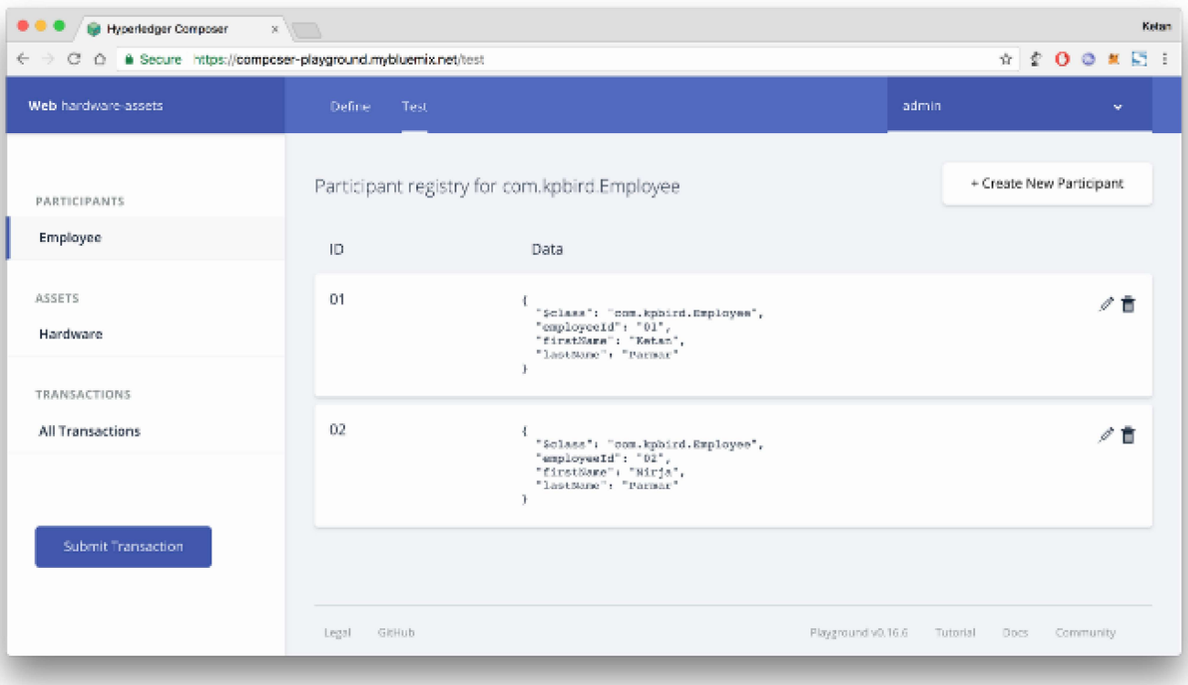
After adding Hardware assets

Steps 10: Let's create participants. Click "Employee" and click "+ Create New Participants" and add following code. We will add two employees

```
{
 "$class": "com.kpbird.Employee",
"employeeId": "01",
"firstName": "Ketan",
"lastName": "Parmar"
}
```

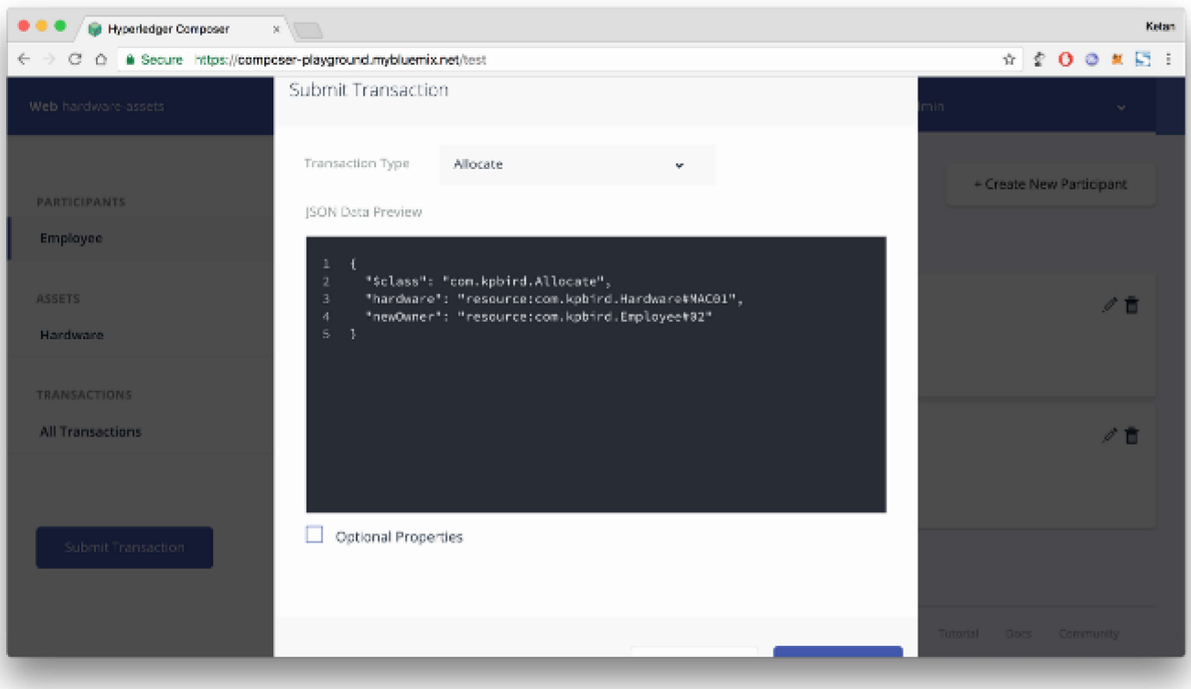
Click on "Create New" on dialog

```
{
"$class": "com.kpbird.Employee",
"employeeId": "02",
"firstName": "Nirja",
"lastName": "Parmar"
}
```



We have two employees

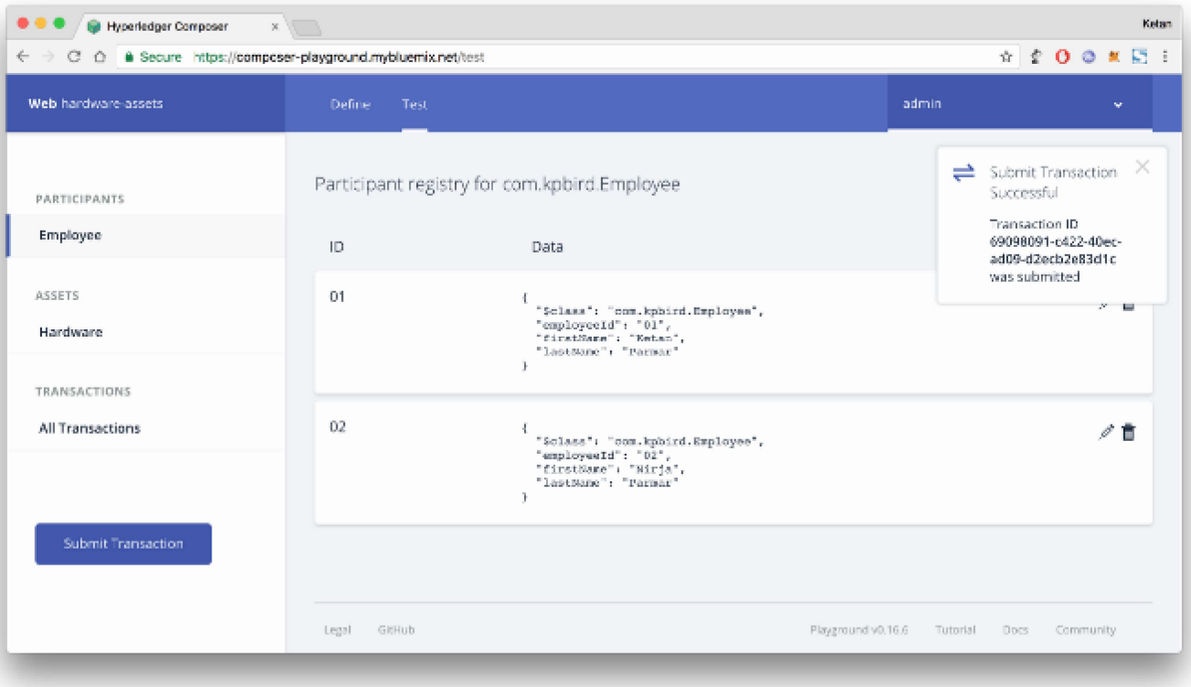
Step 11: It’s time to do transaction, We will allocate Macbook Pro from Ketan (Employee#01) to Nirja (Employee#02). Click on “Submit Transaction” button from left panel. In Transaction dialog, We can see all transaction functions on top “Transaction Type” dropdown.



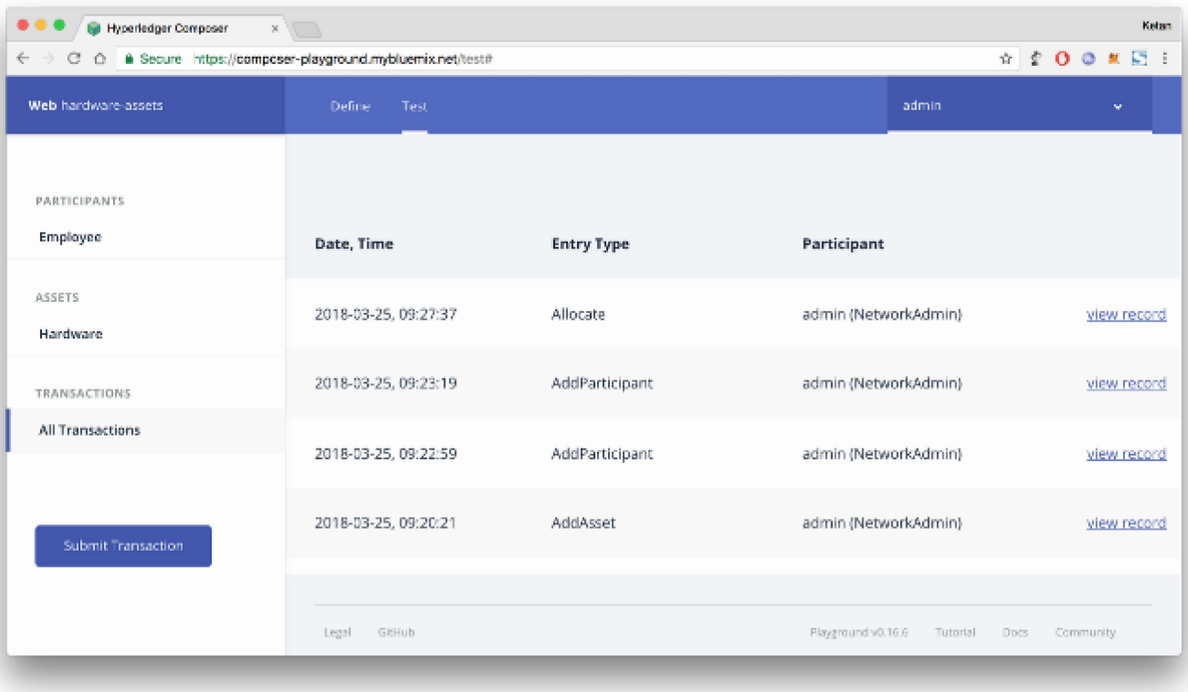
Submit Transaction Dialog

```
{
 “$class”: “com.kpbird.Allocate”,
 “hardware”: “resource:com.kpbird.Hardware#MAC01”,
 “newOwner”: “resource:com.kpbird.Employee#02”
}
```

Now, We are allocating Mac01 to Employee 02. Click Submit button after update above JSON in Transaction Dialog. As soon as you hit submit button. Transaction processed and Transaction Id will generate.

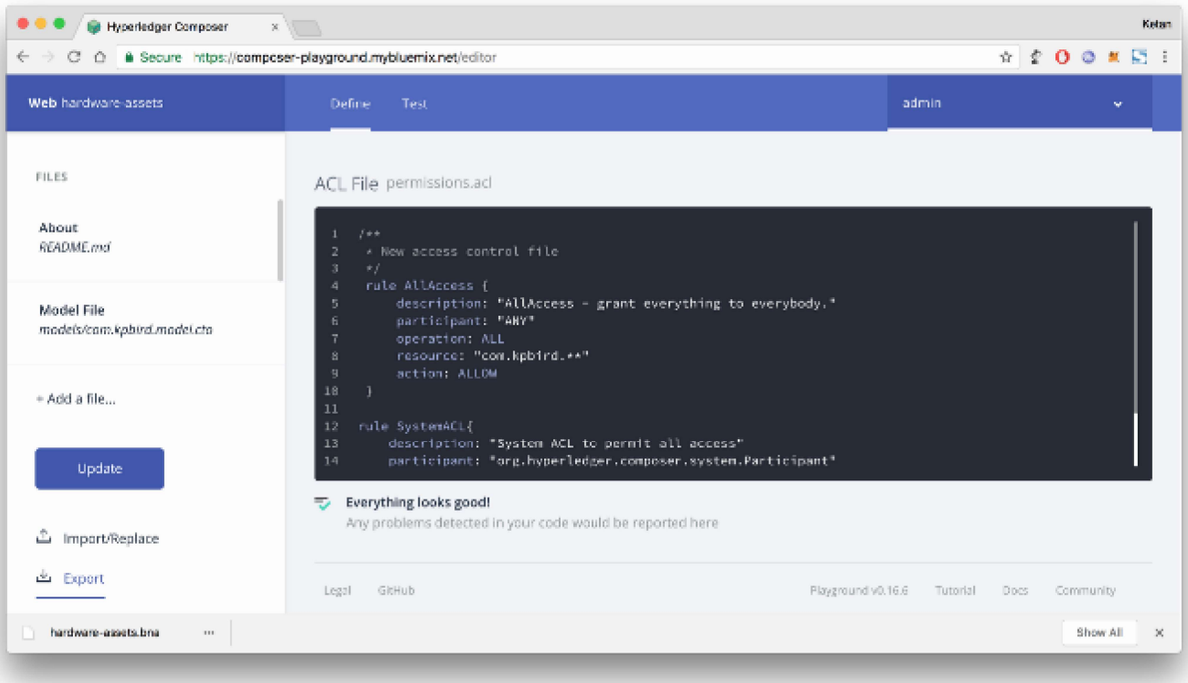


Step 12: Click on “All Transactions” from left panel to verify all transactions. In following screenshots you can see add assets, ass participants and allocation all operation are consider as transactions. “view records” will give us more information about transaction.



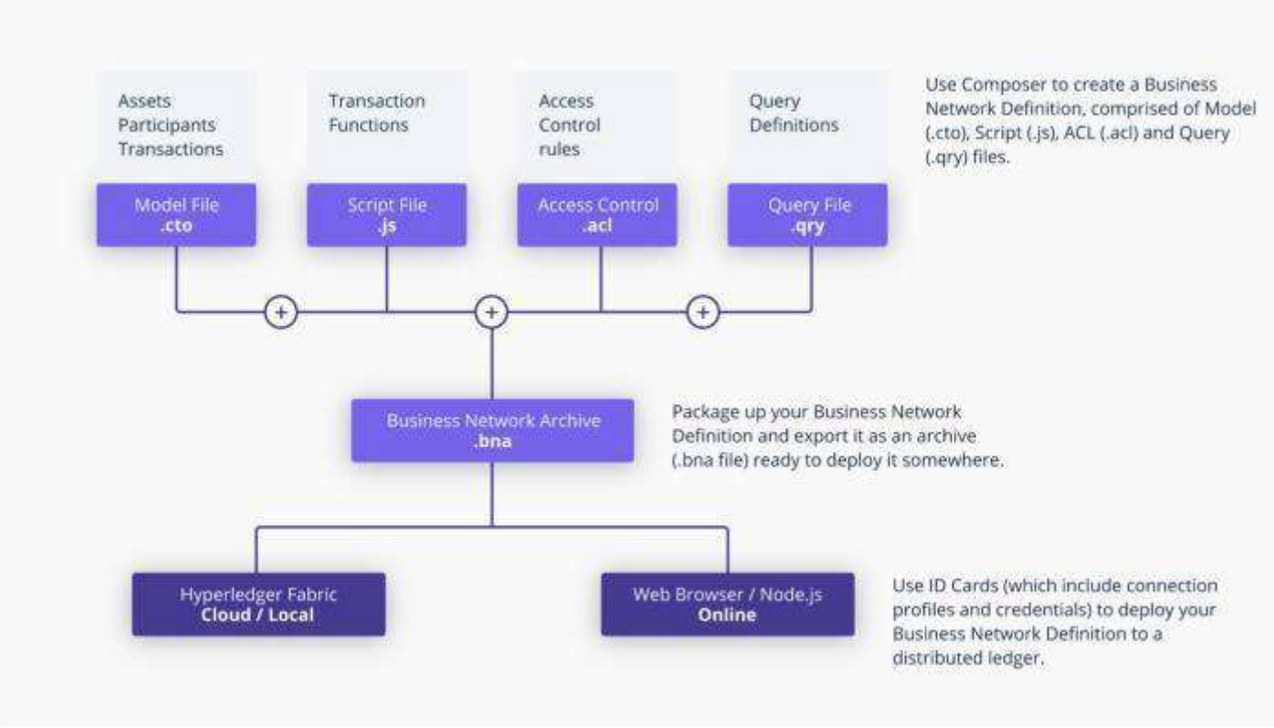
All Transactions

Step 13: Now, It’s time to deploy “hardware-assets” business network to Hyperledger Fabric. Click on “Define” tab from top panel and click “Export” button from left panel. Export will create hardware-assets.bna file.



Download hardware-assets.bna file

.bna is Business Network Archive file which contains model, script, network access and query file



source: https://hyperledger.github.io/composer/introduction/introduction

Step 14: Start Docker and run following commands from ~/fabric-tools directory

Install business network to Hyperledger Fabric, If business network is already installed you can use “update”

instead of “install”

```
$composer runtime install -c PeerAdmin@hlfv1 -n hardware-assets
```

```
Ketan-Parmar:fabric-tools ketan$ composer runtime install -c PeerAdmin@hlfv1 -n hardware-assets
✔ Installing runtime for business network hardware-assets. This may take a minute...

Command succeeded
```

Following command will deploy and start hardware-assets.bna file. Change hardware-assets.bna file before

you execute following command. networkadmin.card file will generate in ~/fabric-tools directory from previous

command.

```
$composer network start –card PeerAdmin@hlfv1 –networkAdmin admin –networkAdminEnrollSecret adminpw –archiveFile /Users/ketan/Downloads/hardware-assets.bna –file networkadmin.card
```

```
Ketan-Parmar:fabric-tools ketan$ composer network start --card PeerAdmin@hlfv1 --networkAdmin admin --networkAdminEnrollSecr
et adminpw --archiveFile /Users/ketan/Downloads/hardware-assets.bna --file networkadmin.card
Starting business network from archive: /Users/ketan/Downloads/hardware-assets.bna
Business network definition:
        Identifier: hardware-assets@0.0.1
        Description: Hardware Assets will maintain Software company's hardware

Processing these Network Admins:
        userName: admin

✔ Starting business network definition. This may take a minute...
Successfully created business network card:
        Filename: networkadmin.card

Command succeeded
```

To connect business network you need connection card. so we can import networkadmin.card using following

command

```
$composer card import -f networkadmin.card
```

To make sure networkadmin.card successfully install you can list cards using following command

```
$composer card list
```

```
Ketan-Parmar:fabric-tools ketan$ composer card list
The following Business Network Cards are available:

Connection Profile: hlfv1
```

Card Name	UserId	Business Network
admin@hardware-assets	admin	hardware-assets
PeerAdmin@trade-network	PeerAdmin	trade-network
admin@trade-network	admin	trade-network
PeerAdmin@hlfv1	PeerAdmin	

```
Issue composer card list --name <Card Name> to get details a specific card

Command succeeded
```

Following command will make sure that our hardware-assets business network is successfully running in Hyperledger Fabric.

$composer network ping – card admin@hardware-assets

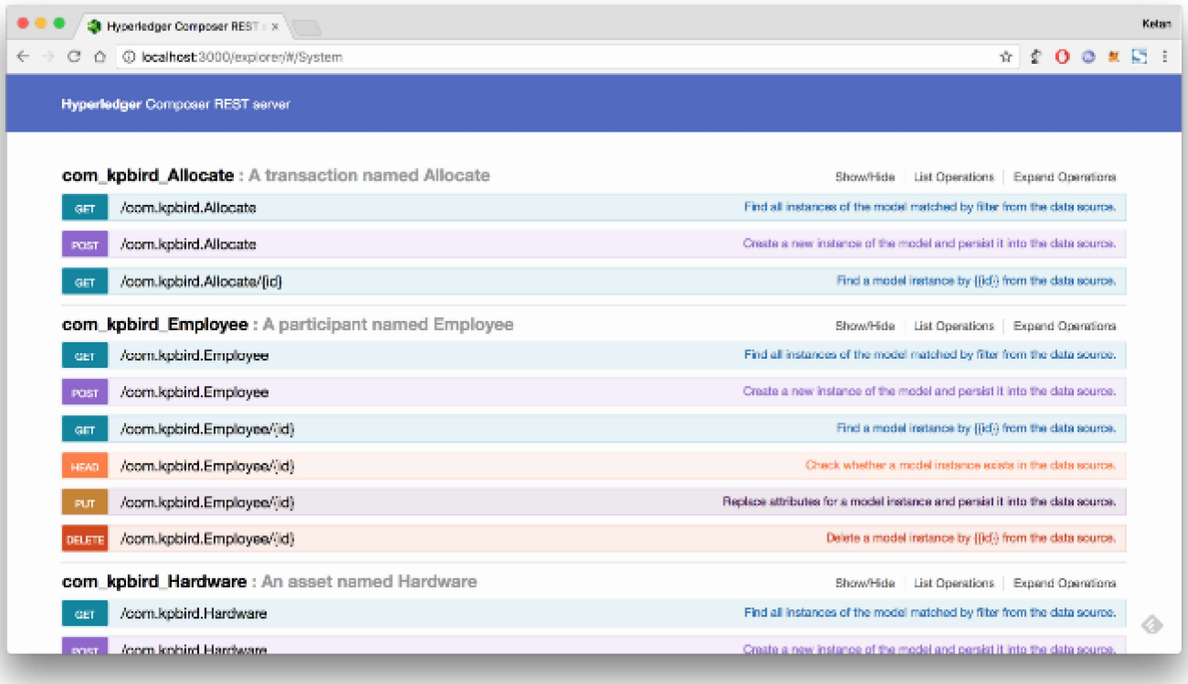
```
Ketan-Parmar:fabric-tools ketan$ composer network ping --card admin@hardware-assets
The connection to the network was successfully tested: hardware-assets
        version: 0.16.0
        participant: org.hyperledger.composer.system.NetworkAdmin#admin

Command succeeded
```

Now It's time to interact with REST API. To develop Web or Mobile Application we require REST API. you car run following command to generate REST API for hardware-assets business network.

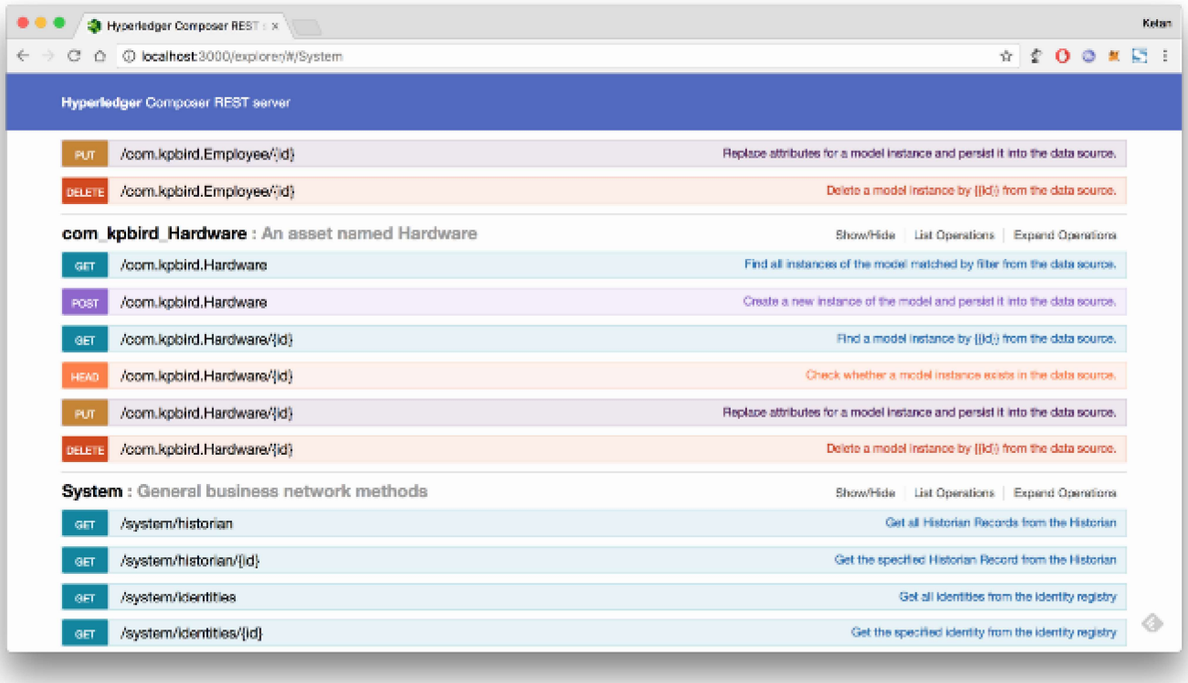
$composer-rest-server

```
Ketan-Parmar:fabric-tools ketan$ composer-rest-server
? Enter the name of the business network card to use: admin@hardware-assets
? Specify if you want namespaces in the generated REST API: always use namespaces
? Specify if you want to enable authentication for the REST API using Passport: No
? Specify if you want to enable event publication over WebSockets: Yes
? Specify if you want to enable TLS security for the REST API: No

To restart the REST server using the same options, issue the following command:
   composer-rest-server -c admin@hardware-assets -n always -w true

Discovering types from business network definition ...
Discovered types from business network definition
Generating schemas for all types in business network definition ...
Generated schemas for all types in business network definition
Adding schemas for all types to Loopback ...
Added schemas for all types to Loopback
Web server listening at: http://localhost:3000
Browse your REST API at http://localhost:3000/explorer
```

rest server will ask few basic information before generate rest api



REST API for our hardware assets



REST API methods for all operations

Conclusion: In this way we have learnt about hyperledger and its use case in business world.
