

## PRACTICAL NO. 4

### Ethereum

**Aim:** To develop and deploy Dapp in Ethereum.

#### Ethereum: <sup>[1]</sup>

- Ethereum is a decentralized, open-source blockchain with smart contract functionality.
- Ether (ETH or  $\Sigma$ ) is the native cryptocurrency of the platform.
- Amongst cryptocurrencies, Ether is second only to Bitcoin in market capitalization.

#### Ethereum Virtual Machine (EVM): <sup>[1]</sup>

- The Ethereum Virtual Machine (EVM) is the runtime environment for transaction execution in Ethereum.

#### Ether: <sup>[1]</sup>

- Ether (ETH): Ether (ETH) is the cryptocurrency generated by the Ethereum protocol as a reward to miners in a proof-of-work system for adding blocks to the blockchain.
- It is the only currency accepted in the payment of transaction fees, which also go to miners.

#### Ether: <sup>[1]</sup>

- The block reward together with the transaction fees provide the incentive to miners to keep the blockchain growing (i.e. to keep processing new transactions).
- Therefore, ETH is fundamental to the operation of the network. Each Ethereum account has an ETH balance and may send ETH to any other account. The smallest subunit of ETH is known as a Wei and is equal to  $10^{-18}$  ETH.

#### Accounts: <sup>[1]</sup>

- There are two types of accounts on Ethereum: user accounts (also known as externally-owned accounts) and contracts.
- Both types have an ETH balance, may send ETH to any account, may call any public function of a contract or create a new contract, and are identified on the blockchain and in the state by their address.

#### Gas: What is Gas in Ethereum? :

- Gas is a unit of account within the EVM used in the calculation of a transaction fee, which is the amount of ETH a transaction's sender must pay to the miner who includes the transaction in the blockchain. <sup>[1]</sup>
- The transaction fee is calculated in Gas, and paid for in Ether.
- The gas is the "fuel" of the Ethereum network, which is used to:
  - Conduct transactions
  - Execute smart contracts
  - Launch Dapps.
- Gas indicates the fee for a particular action or transaction.
- **Gas Limit:** is the maximum amount of Gas that a user is willing to pay for performing this action or confirming a transaction.
- **Gas Price:** is the amount that the user is willing to spend on each unit of Gas.

**MetaMask:** [2]

- MetaMask is a software cryptocurrency wallet used to interact with the Ethereum blockchain.
- It allows users to access their Ethereum wallet through a browser extension or mobile app, which can then be used to interact with decentralized applications(Dapps).
- MetaMask is developed by ConsenSys Software Inc., a blockchain software company focusing on Ethereum-based tools and infrastructure.
- MetaMask allows users to store and manage account keys, broadcast transactions, send and receive Ethereum-based cryptocurrencies and tokens, and securely connect to decentralized applications through a compatible web browser or the mobile app's built-in browser.
- The application includes an integrated service for exchanging Ethereum tokens by aggregating several decentralized exchanges (DEXs) to find the best exchange rate. This feature, branded as MetaMask Swaps, charges a service fee of 0.875% of the transaction amount.
- As of April 2021, MetaMask's browser extension had approximately 10 million monthly active users, according to The Financial Times.

**Setup a MetaMask Ethereum Wallet:**

- MetaMask is just an Ethereum Browser and Ether wallet.
- It interacts with Ethereum Dapps and Smart Contracts without running a full Ethereum node.
- MetaMask add-on can be installed on Chrome, Firefox, Opera, and the new Brave browser.
- URL: <https://metamask.io/>

**MetaMask Installation:**

- Install MetaMask
- Add MetaMask extension to the Browser
- MetaMask will show up 12 words recovery key (Seed).
- These 12 words are the only way to restore MetaMask accounts.
- Use URL: <https://metamask.io/> or type metamask in browser. Click on download.

metamask

About 1,74,00,000 results (0.44 seconds)

<https://metamask.io> :  
MetaMask - A crypto wallet & gateway to blockchain apps  
Available as a browser extension and as a mobile app, MetaMask equips you with a key vault, secure login, token wallet, and token exchange—everything you ...

**Download**  
Install MetaMask for Android. A screenshot of the MetaMask ...  
More results from metamask.io »

[https://chrome.google.com/detail/metamask/nkbih...»](https://chrome.google.com/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn)

**MetaMask**  
MetaMask is a softw

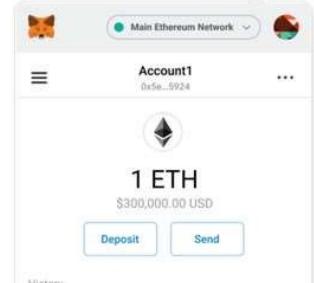
- Select option firefox (According to browser you have open, it shows you appropriate browser extension) and click on installed metamask for firefox.

METAMASK

Features Support About

Firefox iOS Android

Install MetaMask for your browser



- Click on add to firefox button.

MetaMask – Get this Extension

Firefox Add-ons Blog

Firefox Browser ADD-ONS Extensions Themes More...

MetaMask by danfinlay, kumavis

Add to Firefox

536,493 Users

5 ★ 4 ★ 3 ★ 2 ★ 1 ★

- Click on logo of metamask.



- Click on Get Started button.



- Welcome to MetaMask
- Connecting you to Ethereum and the Decentralized Web.
- We're happy to see you.
- Get Started
- Click on Create a Wallet button to create new wallet.
  - If you have already created wallet we can import here using Import Wallet option. Here you need to pass 12 words recovery key (Seed) which you receive.



METAMASK

## New to MetaMask?



- Click on I Agree button to accept the terms.



## Help us improve MetaMask

MetaMask would like to gather usage data to better understand how you interact with the extension. This data will be used to continually improve the usability and user experience of our product and the Ethereum ecosystem.

MetaMask will...

- ✓ Always allow you to opt-out via Settings
- ✓ Send anonymized click & pageview events

- ✗ **Never** collect keys, addresses, transactions, balances, hashes, or other information

...  
...

- Enter new password and confirm password, select check box to accept Terms of Use and then click on create button.



## Create Password

New password (min 8 chars)

.....

Confirm password

.....

I have read and agree to the [Term](#):

- Click on next.



## Secure your wallet

Before getting started, watch this short video to learn about your recovery phrase and how to keep your wallet safe.



W  
Yc  
"n  
ar  
Ho  
ph  
•  
•  
•  
•  
Sh  
ph  
Ni  
re  
Mi  
if:

- Click on Click Here to reveal secret words. This includes 12 secrete words in sequence which can be used to recover the account.



< Back

Tips:

## Secret Backup Phrase

Your secret backup phrase makes it easy to back up and restore your account.

Store this phrase manager like 1Pc

**WARNING:** Never disclose your backup phrase. Anyone with this phrase can take your Ether forever.

Write this phrase paper and store location. If you want security, write it on three pieces of paper - 3 different locations

Memorize this phrase



Download this Secret Backup Phrase and keep it in a secure location or storage media

- Copy these secrete 12 words in same sequence in other files or take screenshot.
- Then click on Next button.



[◀ Back](#)

Tips:

## Secret Backup Phrase

Your secret backup phrase makes it easy to back up and restore your account.

**WARNING:** Never disclose your backup phrase. Anyone with this phrase can take your Ether forever.

surround cricket nuclear rally fuel

Store this phrase manager like IPa

Write this phrase paper and store i location. If you w security, write it c pieces of paper c - 3 different loca

Memorize this ph

Download this Se Phrase and keep an external encr

- Select word of secret phrase in same order which we have seen in earlier step. Then Click on Confirm.



[◀ Back](#)

### Confirm your Secret Backup Phrase

Please select each phrase in order to make sure it is correct.

--	--	--	--

catalog	cricket	donkey	eternal
fuel	immense	nuclear	rally
robot	surround	truly	wine

### Confirm your Secret Backup

Please select each phrase in order to make sure it is correct.

surround	cricket	nuclear	rally
fuel	eternal	immense	wine
donkey	catalog	robot	truly

catalog	cricket	donkey	eternal
fuel	immense	nuclear	rally

- After successfully following all steps for new account creation Congratulations message appears. Finally click on All Done.



## Congratulations

You passed the test - keep your Secret Recovery Phrase safe, it's your responsibility!

### Tips on storing it safely

- Save a backup in multiple places.
- Never share the phrase with anyone.
- Be careful of phishing! MetaMask will never spontaneously ask for your Secret Recovery Phrase.
- If you need to back up your Secret Recovery Phrase again, you can find it in the MetaMask app.
- If you ever have questions or see something fishy, contact our support [here](#).

- A pop up message will appear. Close it.

**What's new**

**Secret Recovery Phrase**

Your "Seed Phrase" is now called your "Secret Recovery Phrase."

6/9/2021

[Read more](#)

**Swapping on mobile is here!**

MetaMask Mobile users can now swap tokens inside the app.

- Finally, it shows you the MetaMask wallet interface.

Account 1

0 ETH

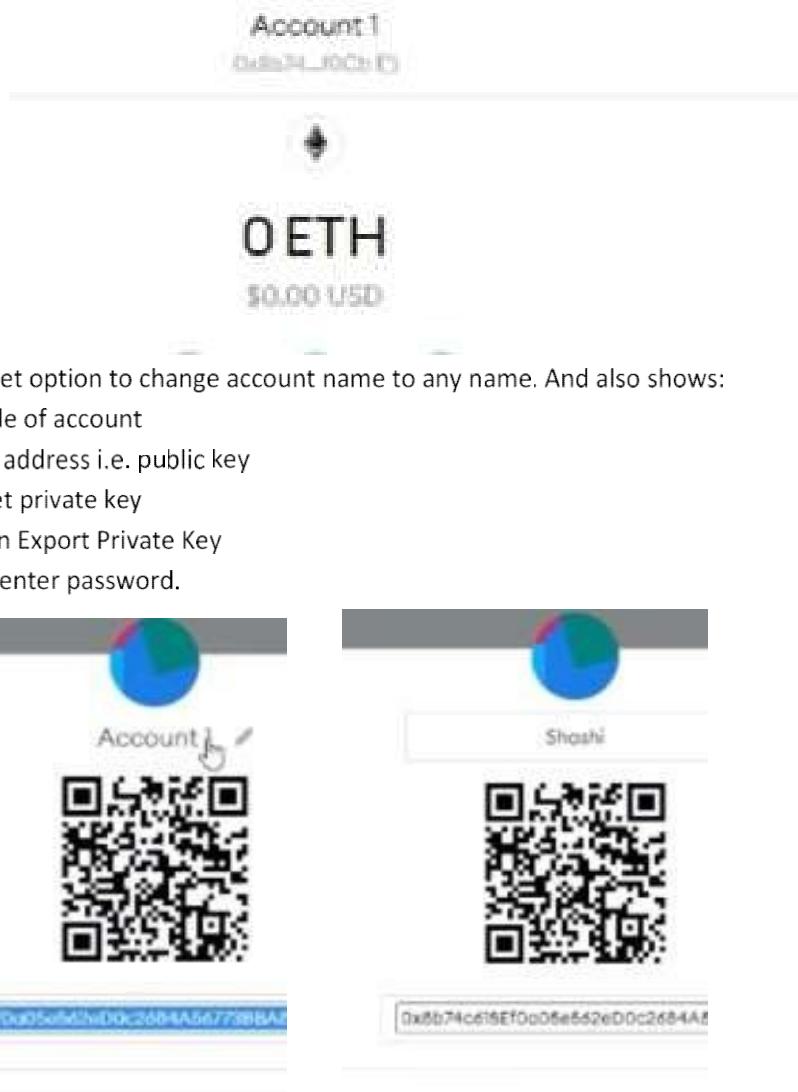
\$0.00 USD

Buy Send Swap

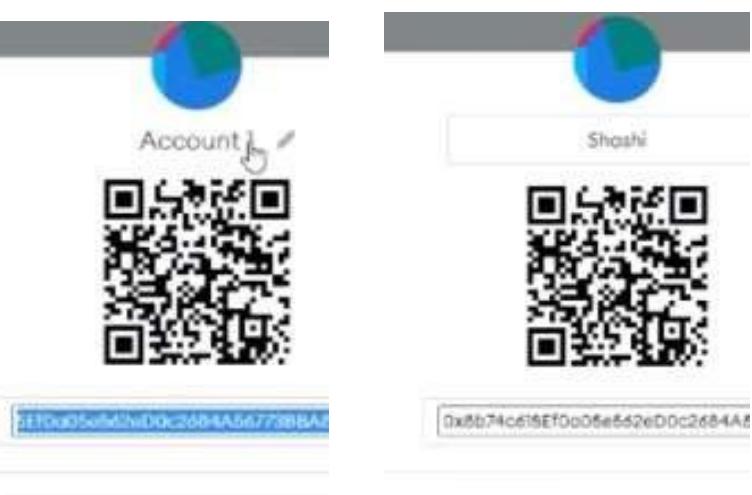
Assets Activity

0 ETH

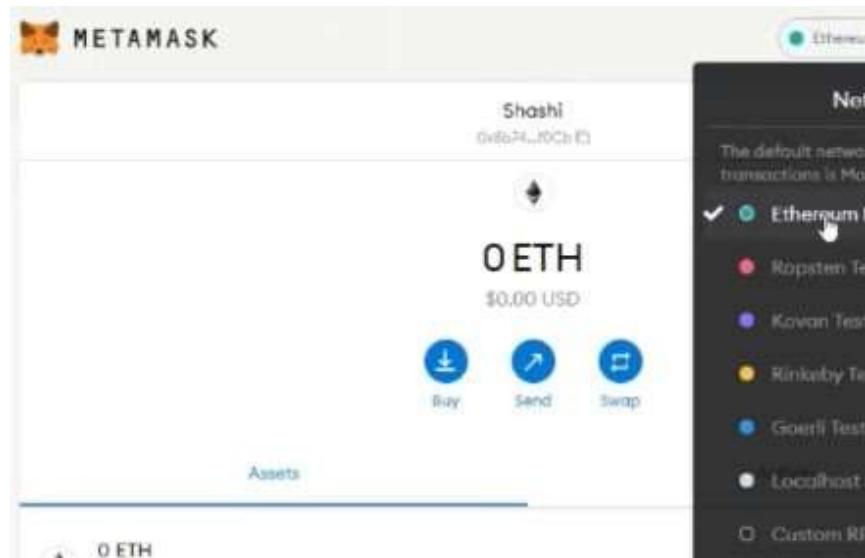
- Click on three vertical dot i.e. Account option. Then select Account Details.



- Here you can get option to change account name to any name. And also shows:
  - QR code of account
  - Wallet address i.e. public key
- You can also get private key
- For that click on Export Private Key
- Here it asks to enter password.



- You can select any networks from the listed networks.



Perform the following task:

- Connect to any test networks and request for ether using buy option.
- Create another account for e.g. Account 2
- Transfer some ether to another account i.e. Account 2. Here gas fee will be applicable which is transaction fee of transaction.
- Check the transaction details.

### **What are Smart Contracts?**

- Smart contracts are lines of code that are stored on blockchain and automatically execute when predetermined terms and conditions are met.
- It's a computer protocol. It is called smart because of its ability to verify and execute a contract without any help from third parties. The contract exists in the decentralized Blockchain network and contains all the terms of a particular agreement.
- The blockchain is then updated when the smart contract is completed.

### **Benefits of Smart Contracts:**

- Security: Blockchain transactions records are encrypted, and that makes them very hard to hack. Because each individual record is connected to previous and subsequent records on a distributed ledger.
- Savings: Smart contracts remove the need for intermediaries because participants can trust the visible data and the technology to properly execute the transaction.
- Trust: Smart contracts automatically execute transactions following predetermined rules, and the encrypted records of those transactions are shared across participants.
- Speed and accuracy: Smart contracts are digital and automated, so you won't have to spend time processing paperwork and correcting the errors.

### **How smart contracts works?**

1. Two users create smart contract.
2. All contract terms are written as a code.
3. Smart contract is stored in blockchain.
4. Smart contract executes itself when events are triggered.

### **Task to be performed:**

1. Develop the smart contract.
2. Compiling the smart contract.
3. Deploying the smart contract.
4. Interacting with smart contract.
5. Adding more functions to our code to make it more practical.

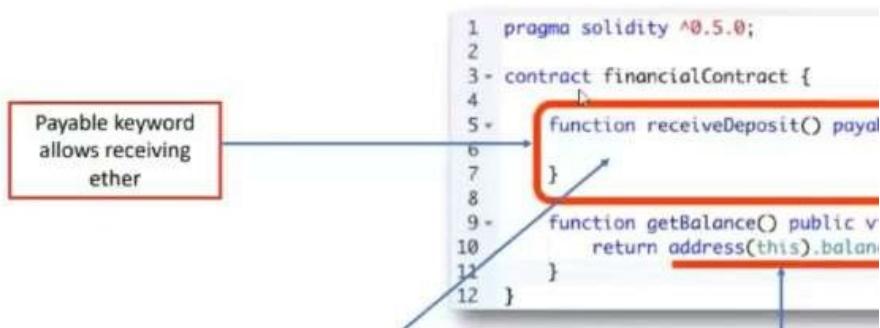
### **Remix IDE: Deployment Parameters:**

1. **JavaScriptVM:** All the transactions will be executed in a sandbox blockchain the browser.
2. **Injected Provider:** Remix will connect to an injectedweb3 provider. Mist and Metamask are example of providers that inject web3, thus they can be used with this option.
3. **Web3Provider:** Remix will connect to a remote node. You will need to provide the URL address to the selected provider: geth, parity or any Ethereum client.

**Example of Contract - Financial Contract in JavaScriptVM environment:**

```
pragma solidity ^0.5.0;
contract FC
{
    uint balance=10500;
    function getBalance() public view returns (uint)
    {
        return balance;
    }
    function deposit(uint newDeposit) public
    {
        balance = balance + newDeposit;
    }
}
```

**Transfer Funds (money) from account to the Contract:**



**Transfer Funds (money) from account to the Contract:**

## Receive ether (2/2)

1 Input the value as wei  
( $10^{-18}$  of ether)

Compile Run Analysis Test

Environment: JavaScript VM

Account: 0xca3...a733c (95)

Gas limit: 3000000

Value: 100

financialContract

Deploy

Transactions recorded: 1

Deployed Contracts

financialContract at 0x692

2 Click the receiveDeposit button to

**Save the address of the Contract Creator:**

Here we can write constructor which will be called at the creation of the instance of the contract.

```

1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4
5     address owner;
6
7     constructor() public{
8         owner = msg.sender;
9     }
10
11     function receiveDeposit() payable public{
12
13 }

```

Withdraw the funds (transferring fund from contract to account back):

Here we use modifier to apply and check condition before executing function.

## Withdraw funds

- Modifier: Conditions you want to test in other functions
- First the modifier will execute, then the invoked function

Transfer some money from the

```

1 pragma solidity ^0.5.0;
2
3 contract financialContr
4
5     address owner;
6
7     constructor() publi
8         owner = msg.ser
9     }
10
11     modifier ifOwner(){
12         if(owner != msg.s
13             revert();
14     }else{
15         -
16     }
17 }
18
19     function getBalance() pub
20         return address(this).ba
21     }
22
23     function withdraw(uint fo
24         msg.sender.transfer(f
25     }
26
27
28
29

```

**Use of special variables:**

Use special variables in solidity contract to get information about blockchain and transaction details.

Name	Returns
blockhash(uint blockNumber) returns (bytes32)	Hash of the given block - or most recent, excluding current block
block.coinbase (address payable)	Current block miner's address
block.difficulty (uint)	Current block difficulty
block.gaslimit (uint)	Current block gas limit
block.number (uint)	Current block number
block.timestamp (uint)	Current block timestamp as unix epoch
gasleft() returns (uint256)	Remaining gas available
msg.data (bytes calldata)	Complete call data
msg.value (uint)	Number of wei sent with the transaction
now (uint)	Current block timestamp
tx.gasprice (uint)	Gas price of the transaction

**Use of Query API:** Use Query API to get information about blockchain and transaction details. Use following URL <https://www.blockchain.com/api/q>

The screenshot shows the Blockchain.com API interface. At the top, there are navigation links: Wallet, Exchange, and Explorer (which is underlined, indicating it's the active page). Below this, there are several sections:

- Charts**: Shows a small chart icon.
- DeFi**: Shows a small DeFi icon.
- NFTs**: Shows a small NFT icon.
- Academy**: Shows a small Academy icon.
- Developers**: Shows a small Developers icon.
- Assets**: A section for different cryptocurrencies:
  - Bitcoin**: Shows a small Bitcoin icon.
  - Ethereum**: Shows a small Ethereum icon.
  - Bitcoin Cash**: Shows a small Bitcoin Cash icon.

**Query API**: Plaintext query api to retrieve data from blockchain.com

Some API calls are available with **CORS headers** if you add a &cors=true parameter to the GET request. Please limit your queries to a maximum of 1 every 10 seconds. All bitcoin values are in Satoshi i.e. divide by 100000000 to get the value in USD.

**Real-time**

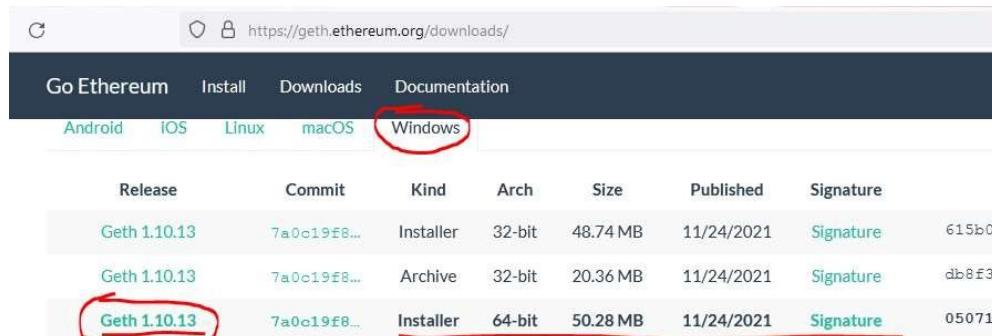
getdifficulty- Current difficulty target as a decimal number  
 getblockcount- Current block height in the longest chain  
 latesthash- Hash of the latest block  
 bcperrblock- Current block reward in BTC  
 totalbc- Total Bitcoins in circulation (delayed by up to 1 hour)  
 probability- Probability of finding a valid block each hash attempt  
 hashestowin- Average number of hash attempts needed to solve a block  
 nextretarget- Block height of the next difficulty re-target  
 avtxsize- Average transaction size for the past 1000 blocks. Change the number of blocks by passing an integer as the parameter

**Setting up an Ethereum Node:**<sup>[3]</sup>

- Ethereum node is any device that is running the Ethereum protocol (blockchain).
- When we connect to the Ethereum protocol we are on the Ethereum blockchain network.
- By running an Ethereum node we can connect to other nodes in the network, have direct access to the blockchain, and even do things like mine blocks, send transactions, and deploy smart contracts.

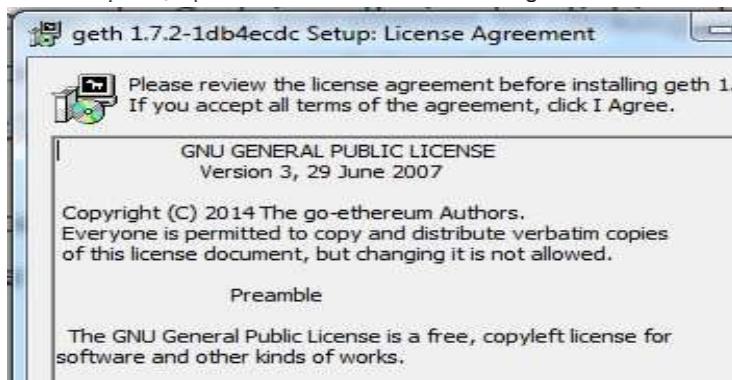
**Creating private Ethereum blockchain using Geth (Go-Ethereum):**<sup>[3]</sup>**Steps to install Geth:**

- Visit the Go Ethereum website and install Geth
- Visit here <http://geth.Ethereum.org/downloads/>
- Download the latest release of Geth for Windows, make sure you download the 64-bit version.



Release	Commit	Kind	Arch	Size	Published	Signature
Geth 1.10.13	7a0c19f8...	Installer	32-bit	48.74 MB	11/24/2021	<a href="#">Signature</a> 615b02
Geth 1.10.13	7a0c19f8...	Archive	32-bit	20.36 MB	11/24/2021	<a href="#">Signature</a> db8f31
<b>Geth 1.10.13</b>	<b>7a0c19f8...</b>	<b>Installer</b>	<b>64-bit</b>	<b>50.28 MB</b>	<b>11/24/2021</b>	<b><a href="#">Signature</a> 050711</b>

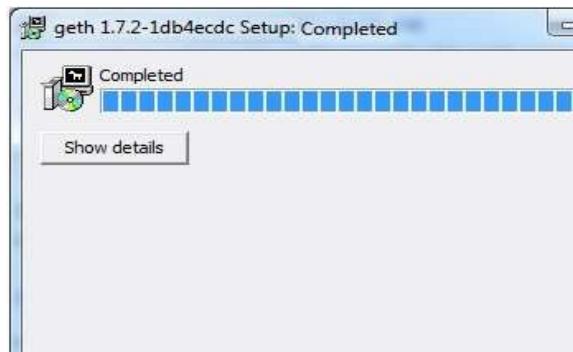
- Once your download is complete, open the installer and click "I Agree"



- You'll be prompted to select a destination folder for your download. By default, Geth will install under C:\Program Files\Geth



- Close installation once complete



Creating private Ethereum blockchain using Geth (Go-Ethereum):<sup>[3]</sup>

## Establishing Our Own Private Ethereum Network:

- Create a new folder on your desktop called “private-chain”.
  - Open command prompt in this folder and create a data directory folder for our chaindata by typing “mkdir chaindata”.
  - Next, we need to create and save our genesis.json block in our Private Chain folder, as the genesis block will be used to initialize our private network and store data in the data directory folder “chaindata”.

**Open up notepad, copy & paste the code below into a new file called “genesis.json” and save this file in our Private-Chain folder.**



### Start the Ethereum peer node (Start the Blockchain)

Run the following command:

```
geth --datadir chaindata init genesis.json
```

```
Pete@Reggie-PC MINGW64 ~/Desktop/Private Chain
$ geth --datadir= ./chaindata/ init ./genesis.json
WARN [11-05|10:05:38] No etherbase set and no accounts found as defau
INFO [11-05|10:05:38] Allocated cache and file handles      data
rs\\Reggie\\Desktop\\Private Chain\\chaindata\\geth\\chaindata" cac
=16
INFO [11-05|10:05:38] Writing custom genesis block
INFO [11-05|10:05:38] Successfully wrote genesis state      data
a
1a
INFO [11-05|10:05:38] Allocated cache and file handles      data
rs\\Reggie\\Desktop\\Private Chain\\chaindata\\geth\\lightchaindata
ndles=16
INFO [11-05|10:05:38] Writing custom genesis block
INFO [11-05|10:05:38] Successfully wrote genesis state      data
indata
...f0181a
```

Now we can start Geth and connect to our own private chain.

```
geth --datadir= ./chaindata/
```

```
Pete@Reggie-PC MINGW64 ~/Desktop/Private Chain
$ geth --datadir= ./chaindata/
WARN [11-05|10:09:20] No etherbase set and no accounts found as defau
INFO [11-05|10:09:20] Starting peer-to-peer node      instan
6.7-stable-ab5646c5/windows-amd64/go1.8.3
INFO [11-05|10:09:20] Allocated cache and file handles      data
rs\\Reggie\\Desktop\\Private Chain\\chaindata\\geth\\chaindata" cache
s=1024
WARN [11-05|10:09:20] Upgrading chain database to use sequential keys
INFO [11-05|10:09:20] Initialised chain configuration      config
15 Homestead: 0 DAO: <nil> DAOSupport: false EIP150: <nil> EIP155: 0
Metropolis: <nil> Engine: unknown"
INFO [11-05|10:09:20] Disk storage enabled for ethash caches      dir="C
eggie\\Desktop\\Private Chain\\chaindata\\geth\\ethash" count=3
INFO [11-05|10:09:20] Disk storage enabled for ethash DAGs      dir=C:
oggie\\AppData\\Ethash
INFO [11-05|10:09:20] Upgrading db log bloom bins
INFO [11-05|10:09:20] Database conversion successful
INFO [11-05|10:09:20] Bloom-bin upgrade completed      elapse
INFO [11-05|10:09:20] Initialising Ethereum protocol      versio
" network=1
INFO [11-05|10:09:20] Loaded most recent local header      number
1a7..f0181a td=131072
INFO [11-05|10:09:20] Loaded most recent local full block      number
1a7..f0181a td=131072
INFO [11-05|10:09:20] Loaded most recent local fast block      number
1a7..f0181a td=131072
INFO [11-05|10:09:20] Starting P2P networking
INFO [11-05|10:09:22] UDP listener up      self=e
```

- Minimize the terminal and Open new terminal.
- Type the command: Here is IPC is used to interact with Geth  
geth attach ipc:\\.\pipe\geth.ipc
- Command to create new account (here you need to account password) personal.newAccount()
- To know the all accounts eth.accounts
- To know the main account eth.coinbase
- To get balance of account eth.getBalance(eth.accounts[0])
- Start mining miner.start()
- Stop mining miner.stop()
- Again check the balance of same account eth.getBalance(eth.accounts[0])
- Create another account personal.newAccount()

**Transaction:**

- Unlock the account 0 to send transaction - personal.unlockAccount(eth.accounts[0])
- Command to send transaction – eth.sendTransaction({from: eth.coinbase, to: eth.accounts[1], value: web3.toWei(10, "ether")})  
start mining and stop mining:
- Start mining miner.start() Stop mining miner.stop()
- Check the account balance after successfully mining the block, some ethers will be added to accounts 0 in Wei unit eth.getBalance(eth.accounts[0])
- Also check the balance of account 1 in Wei unit eth.getBalance(eth.accounts[1])
- Balance of second account in ether web3.fromWei(eth.getBalance(eth.accounts[1]), "ether")
- To get details of latest block eth.getBlock("latest")
- To get details of specific block eth.getBlock(35)

**The Complete Blockchain Developer Toolkit :<sup>[3]</sup>****Introduction to Ganache:**

- Ganache as your personal blockchain for Ethereum development.
- It will allow you to deploy smart contracts, develop applications, and run tests.
- From the above URL link downloaded the archived package, extract the installer and run through the setup steps.

Ganache Personal Blockchain Interface consist of:

- Accounts Page - this shows you all of the accounts that are automatically generated, along with their balances.
- Blocks Page - this shows you each block that has been mined on the personal blockchain network, along with the gas cost and transactions.
- Transactions Page - this list all the transactions that have taken place on the personal blockchain.
- Logs Page - this shows you all the server logs that you might need when debugging your application.
- Link to download Ganache <https://www.trufflesuite.com/ganache>
- Open the Ganache and then select the workspace QUICKSTART.



**Ganache Personal Blockchain Interface :**

The screenshot shows the Ganache interface with the following account details:

ADDRESS	BALANCE
0xEF99Da03209DEffBFe497939Fbb833eeD40FF300	100.00 ETH
0x3Cb28b929A43Bb6f4a9CCB2B86304c791B16f277	100.00 ETH
0xCaB30f46aA13083FF63084aB025d2d243E4DeaF1	100.00 ETH
0xB6C9c2a5cEE01fAc8E5b6c76C451dAC795F8581a	100.00 ETH

#### Developing smart contracts using Truffle:<sup>[4]</sup>

- Install **Node JS** and **Truffle Suite** to develop and migrate the smart contracts into the Private Blockchain network.
- Make use of Truffle tools like compile, migrate and test for compilation, migration and testing the smart contracts through Blockchain.
- Note that you have a private blockchain running, you need to configure your environment for developing smart contracts.
- The first dependency you'll need is Node Package Manager, or NPM, which comes with Node.js.
- You can see if you have node already installed by going to your terminal and typing: **node -v**
- Otherwise, Download from: URL: <https://nodejs.org/en/>

#### Truffle Framework:<sup>[4]</sup>

- Truffle Framework provides a suite of tools for developing Ethereum smart contacts with the Solidity programming language.
  - Smart Contract Management
  - Automated Testing
  - Deployment of smart contract
  - Migration of smart contract onto private blockchain network
  - Network Management
  - Script Runner
  - Client Side Development
- You can install Truffle with NPM in your command line like this:

```
npm install -g truffle
```

```
C:\Users\Ravi>npm install -g truffle
npm WARN deprecated graphql-tools@6.2.6: This package has been depri
w it only exports makeExecutableSchema.\nAnd it will no longer receive
nWe recommend you to migrate to scoped packages such as @graphql-tc
```

#### Create a new truffle project:<sup>[4]</sup>

- Go to cmd prompt :
- ```
mkdir blockchain-toolkit
```

- ```
cd blockchain-toolkit
truffle init
```
- Command to install touch for windows: **npm install -g touch-for-windows**
  - Command to create file: **touch package.json**
  - copy-and-pasting the below code into **package.json** file (It's configuration file required to run truffle project):
 

```
{
  "name": "blockchain-toolkit",
  "version": "1.0.0",
  "description": "The Complete Blockchain Developer Toolkit for 2019 & Beyond",
  "main": "truffle-config.js",
  "directories": {
    "test": "test"
  },
  "scripts": {
    "dev": "lite-server",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "gregory@dappuniversity.com",
  "license": "ISC",
  "devDependencies": {
    "bootstrap": "4.1.3",
    "chai": "^4.1.2",
    "chai-as-promised": "^7.1.1",
    "chai-bignumber": "^2.0.2",
    "dotenv": "^4.0.0",
    "ganache-cli": "^6.1.8",
    "lite-server": "^2.3.0",
    "nodemon": "^1.17.3",
    "solidity-coverage": "^0.4.15",
    "truffle": "5.0.0-beta.0",
    "truffle-contract": "3.0.6",
    "truffle-hdwallet-provider": "^1.0.0-web3one.0"
  }
}
```
  - copy-and-pasting the below code into **package.json** file
  - Save **package.json** file
  - Run the following command:  
**npm install**
  - **Start developing a smart contract using solidity:**
  - Run the touch command to create new contract:  
    **touch ./contracts/MyContract.sol**
  - Copy the below contract code and save in MyContract.sol

```

pragma solidity ^0.5.0;
contract MyContract {
    string value;
    constructor() public {
        value = "myValue";
    }
    function get() public view returns(string memory) {
        return value;
    }
    function set(string memory _value) public {
        value = _value;
    }
}

```

- Run the following command:

**truffle compile**

```

C:\Windows\system32\cmd.exe
C:\Users\Ravi\blockchain-toolkit>truffle compile
Compiling your contracts...
=====
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\MyContract.sol
> Compiling .\contracts\MyContract.sol
> Artifacts written to C:\Users\Ravi\blockchain-toolkit\build\contracts
> Compiled successfully using:
      - solc@0.5.12 via solc-bin@0.5.12

```

- Update the project configuration file to specify the personal blockchain network we want to connect to (Ganache);

- Find the file **truffle-config.js** and paste the following code:

```

module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",
      port: 7545,
      network_id: "*" // Match any network id
    }
  },
  solc: {
    optimizer: {
      enabled: true,
      runs: 200
    }
  }
}

```

- Create a migration script inside the migrations directory to deploy the smart contract to the personal blockchain network.
- Run the following command:

**touch migrations/2\_deploy\_contracts.js**

- Copy the below script into **2\_deploy\_contracts.js**

```
var MyContract = artifacts.require("./MyContract.sol");
```

```
module.exports = function(deployer)
{
  deployer.deploy(MyContract);
};
```

- Run the newly created migration script to deploy the smart contract to the personal blockchain network:  
truffle migrate

```
C:\Windows\system32\cmd.exe
Starting migrations...
=====
> Network name:      'development'
> Network id:        5777
> Block gas limit:   6721975 <0x6691b7>

1_initial_migration.js
=====

  Deploying 'Migrations'
  -----
  > transaction hash: 0xa5ab6b4923fe9f0198
455e9c6c33f2
    > Blocks: 0           Seconds: 0
    > contract address: 0xCe5f6B242E37D8Cdd6
    > block number:      1
    > block timestamp:   1639333688
    > account:           0xEF99Da03209DEffBF
    > balance:            99.9969302
    > gas used:          153490 <0x25792>
    > gas price:          20 gwei
    > value sent:         0 ETH
    > total cost:         0.0030698 ETH

    > Saving migration to chain.
    > Saving artifacts
  -----
  > Total cost:          0.0030698 ETH

2_deploy_contracts.js
=====

  Deploying 'MyContract'
  -----
  > transaction hash: 0xd4673843d06c44bebe
90fb9ca5ca0
    > Blocks: 0           Seconds: 0
    > contract address: 0xF1842D385E6698a99F
    > block number:      3
    > block timestamp:   1639333690
    > account:           0xEF99Da03209DEffBF
```

- Run the following commands: truffle console

```
MyContract.deployed().then((instance) => { app = instance })
```

```
C:\Windows\system32\cmd.exe - truffle console
C:\Users\Ravi\blockchain-toolkit>truffle console
truffle(development)> MyContract.deployed().then((instance) => { app = instance })
```

- Now we can read the storage value

```
app.get()
```

Output as 'myValue'

```
> >
undefined
truffle(development)> app.get()
```

- Now we can set a new value like this:

```
app.set('New Value')
```

- We can read that the value was updated like this:

- ```
app.get()
Output as 'New Value'
• You can exit the Truffle console by typing this command:
.exit
```

### References:

1. <https://en.wikipedia.org/wiki/Ethereum>
2. <https://en.wikipedia.org/wiki/MetaMask>
3. <https://codeburst.io/build-your-first-ethereum-smart-contract-with-solidity-tutorial-94171d6b1c4b>
4. <https://www.dappuniversity.com/articles/blockchain-developer-toolkit>
5. [https://www.tutorialspoint.com/solidity/solidity\\_quick\\_guide.htm](https://www.tutorialspoint.com/solidity/solidity_quick_guide.htm)

### Exercise:

1. Install the metamask in browser. Setup the metamask digital cryptocurrency wallet. Create multiple accounts in metamask and connect with one of the etherum blockchain test network. Perform the task buy ethers and send ethers from one account to another. Take the screenshots of created accounts, account assets and account transactions which showing the details of transaction.  
(Use following url to get free ether for Sepolia Test Network: <https://faucets.chain.link/sepolia>)
2. Start **Ganache** (your personal private blockchain network). Connect Ganache with **MetaMask** and import the account from Ganache to MetaMask. Transfer funds from imported account to other account of MetaMask. Take the screenshots of created accounts, account assets and account transactions which showing the details of transaction from MetaMask and Ganache interface.

### GoEthereum(Geth)

3. Create Ethereum node using **Geth (GoEthereum)** and create genesis block and create your personal private Ethereum blockchain. And use IPC to interact with Geth node to perform following task: create account, transfer funds using send transaction, mine the block, show the account balance before and after the mining the block, show the specific block details and access chain details.

### RemixIDE – Injected Provider - Public Test Network (Goerli, Sepolia) or Ganache

4. Write a solidity smart contract for performing following task using **remixIDE** and deployed it on **public test network – Goerli / Sapolia** using **Injected provider** environment.
  - a. To transfer funds (ethers) from user account to contract account.
  - b. To withdraw funds (ethers) from contract account to user account.
  - c. To apply restriction that only owner of the contract can withdraw funds (ethers) from contract account to his/her user account.
5. Write a smart contract to calculate the compound interest and deploy it on **Ganache** using **injected provider**.

### Truffle - Ganache

6. Build and test decentralized application (**Dapp**) for **Election Voting System** on the local Ethereum Blockchain Network **Ganache** using **truffle suite**.
7. Build and test decentralized application, (**Dapp**) for **Banking System** on the local Ethereum Blockchain Network **Ganache** using **truffle suite**.

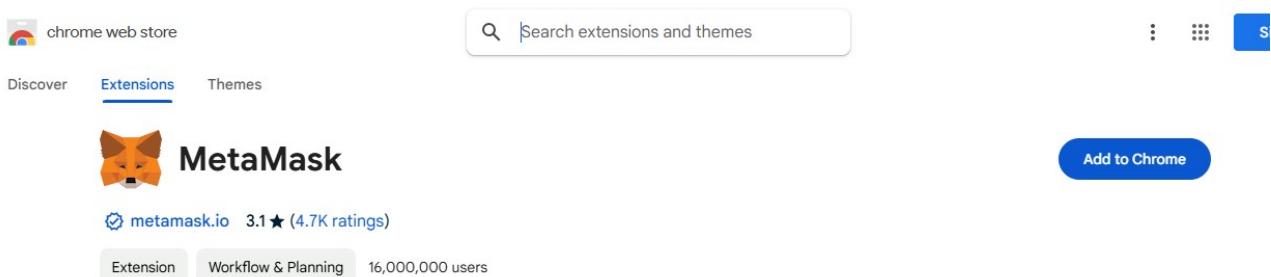
1. Install the metamask in browser. Setup the metamask digital cryptocurrency wallet. Create multiple accounts in metamask and connect with one of the etherum blockchain test network. Perform the task buy ethers and send ethers from one account to another. Take the screenshots of created accounts, account assets and account transactions which showing the details of transaction.  
 (Use following url to get free ether for Sepolia Test Network: <https://faucets.chain.link/sepolia>)

Installation of MetaMask:

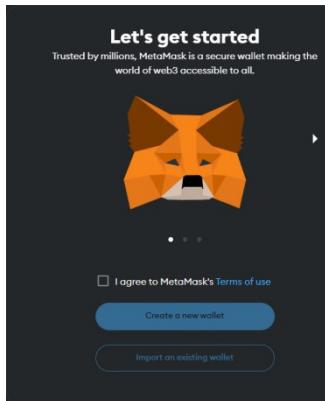
Open URL <https://metamask.io/> in chrome browser



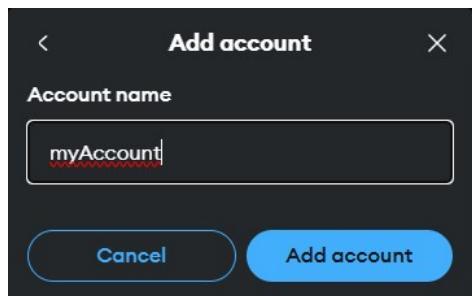
Add Extension to Chrome



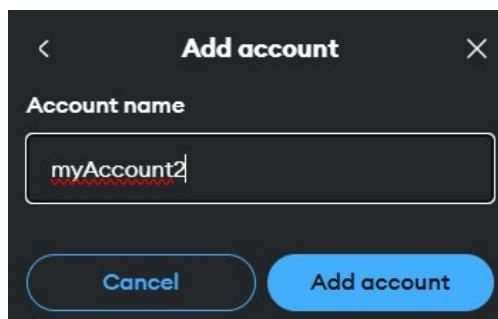
Create Wallet



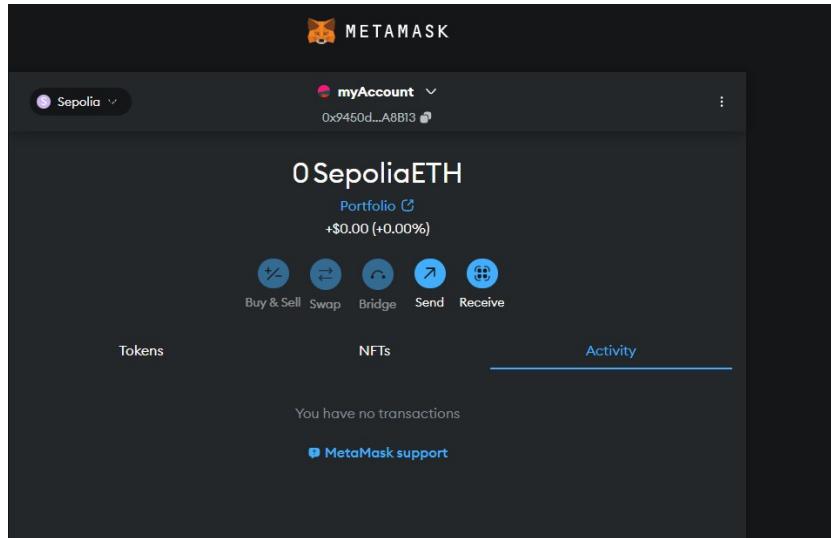
Create Account: myAccount



Create Account: myAccount2



Interface:



Open URL <https://faucets.chain.link/sepolia> in chrome browser for sepolia

**Chainlink**

- Products
- Economics
- Utilities
- Faucets
- PegSwap
- Support

### Claim Sepolia Faucet

Get tokens on the Ethereum Sepolia network. No cost, no strings attached. Experiment with smart contracts today.

Search Faucets  All

|                                                                                                   |                                                                                                                     |                                                                                                                      |                                                                                                                   |
|---------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
|  Sepolia         |  Ethereum Sepolia<br>Drips 0.1 ETH |  BNB Chain Testnet<br>Drips 25 LINK |  Avalanche<br>Drips 0.1 AVAX   |
|  Fuji            |  Fantom Testnet<br>Drips 25 LINK   |  Base Sepolia<br>Drips 0.1 ETH      |  Base Sepolia<br>Drips 25 LINK |
|  Polygon Testnet | Drips 0.1 ETH                                                                                                       | Drips 25 LINK                                                                                                        |  Polygon<br>Drips 0.5 P        |

1 Faucet selected Clear all Continue

## Claim Ethereum Sepolia

### Get tokens

Confirm your addresses and get the tokens.



Ethereum Sepolia  
Drips 0.1 ETH



450dE0Ffb62CB3Ab9B26c9FB904A07b2f5A8B13

 Remove

Get tokens

### Finished

Check to see if the tokens have arrived.

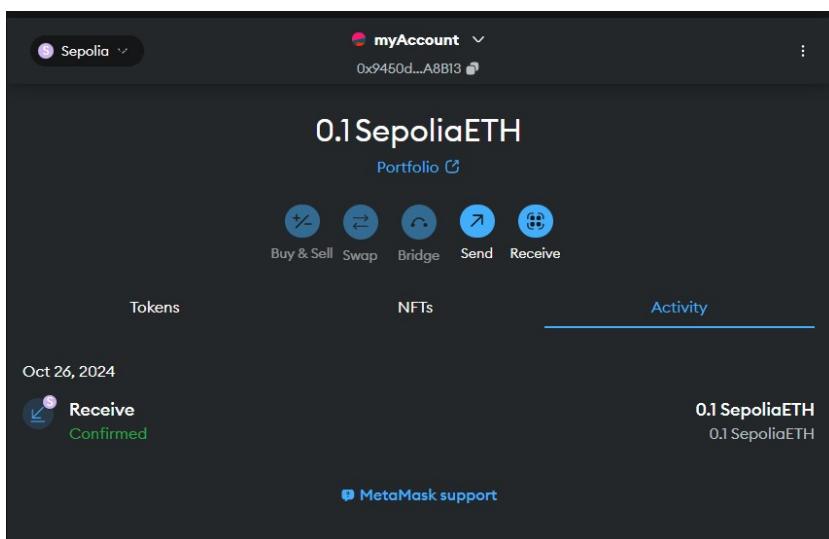


Ethereum Sepolia  
Drips 0.1 ETH



Transaction Hash [0xbd40...d0c0](#)

 Success



Transfer Sepolia to myAccount2

This is a transfer confirmation dialog box titled "Send".

**From:** myAccount  
0x9450d...A8B13

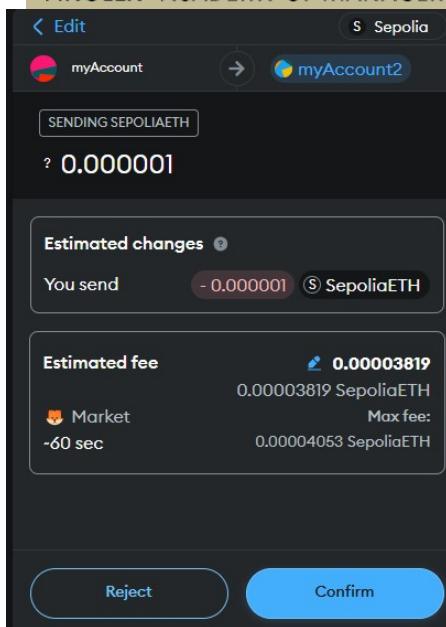
**To:** myAccount2  
0xc9fF0...8Fc02

**Amount:** 0.000001

**Balance:** 0.1 **Max**

**Cancel** **Continue**

The "To" field contains a placeholder "Sepoli..." which has been partially replaced by "myAccount2".



## Activity: myAccount

0.1 SepoliaETH

Portfolio ↗

Buy & Sell Swap Bridge Send Receive

Tokens NFTs Activity

Oct 26, 2024

**Send Confirmed** -0.000001 SepoliaETH -0.000001 SepoliaETH

**Receive Confirmed** 0.1 SepoliaETH 0.1 SepoliaETH

## Activity: myAccount2

<0.000001 SepoliaETH

Portfolio ↗

Buy & Sell Swap Bridge Send Receive

Tokens NFTs Activity

Oct 26, 2024

**Receive Confirmed** 0.000001 SepoliaETH 0.000001 SepoliaETH

MetaMask support

2. Start Ganache (your personal private blockchain network). Connect Ganache with **MetaMask** and import the account from Ganache to MetaMask. Transfer funds from imported account to other account of MetaMask. Take the screenshots of created accounts, account assets and account transactions which showing the details of transaction from MetaMask and Ganache interface.

### 1. Save the Network

Network name  
myGanache

Suggested name: Geth Testnet

According to our records, the network name may not correctly match this chain ID.

New RPC URL  
HTTP://127.0.0.1:7545

Chain ID  
1337

This Chain ID is currently used by the Ganache network.

Currency symbol  
GNHETH

Suggested ticker symbol: ETH

This token symbol doesn't match the network name or chain ID entered. Many popular tokens use similar symbols, which scammers can use to trick you into sending them a more valuable token in return. Verify everything before you continue.

Block explorer URL (Optional)

### 2. Copy the keys

|                                                       |                       |               |            |  |
|-------------------------------------------------------|-----------------------|---------------|------------|--|
| ADDRESS<br>0xd150E46b725682cf9Dc9Cf7De347ef7E4289E48A | BALANCE<br>100.00 ETH | TX COUNT<br>0 | INDEX<br>0 |  |
| ADDRESS<br>0x5080006f6c41f39Be30279f13C0cE0aF2E4979a1 | BALANCE<br>100.00 ETH | TX COUNT<br>0 | INDEX<br>1 |  |
| ADDRESS<br>0x8aE2853a23913a50D817C3DCbeD1bc414EcF1F48 | BALANCE<br>100.00 ETH | TX COUNT<br>0 | INDEX<br>2 |  |
| ADDRESS<br>0x2909b9FE823d5B3C98d0A6CC79e8c0B1BB7a245d | BALANCE<br>100.00 ETH | TX COUNT<br>0 | INDEX<br>3 |  |
| ADDRESS<br>0xE237936b3B9Dd0644F9dB8b95955c51a2eb2f29  | BALANCE<br>100.00 ETH | TX COUNT<br>0 | INDEX<br>4 |  |

### 3. Claim The Ether

M myGanache ▾

Account 7 ▾  
0xd150E...9E48A ⚙️

⋮

# 100 GNHETH

Portfolio ⚙️  
+\$0.00 (+0.00%)

Buy & Sell Swap Bridge Send Receive

Tokens NFTs **Activity**

You have no transactions

MetaMask support

#### 4. Account 7 and Account 3

| Account   | Address         | Balance    | Status   |
|-----------|-----------------|------------|----------|
| Account 3 | 0x00ec5...f19e2 | 0 GNHETH   | Imported |
| Account 7 | 0xd150E...9E48A | 100 GNHETH | Imported |

#### 5. Transfer GNEtH

< Send

From

Account 7  
0xd150E...9E48A

GNHETH ▾ 50 GNHETH

Balance: 100 Max

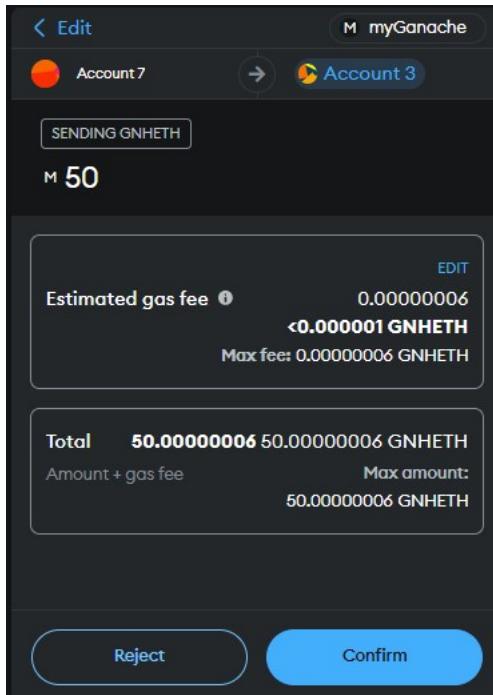
To

Account 3  
0x00ec5...f19e2

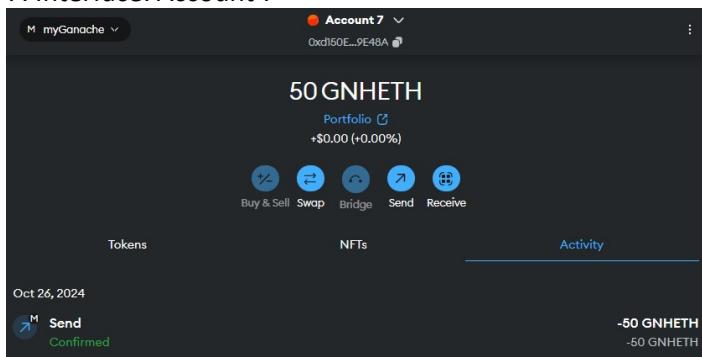
GNHETH ▾ 50 GNHETH

**Cancel** **Continue**

## 6. Confirm the Transaction



## 7. Interface: Account 7



## 8. Ganache:

| ADDRESS                                    | BALANCE   | TX COUNT | INDEX |  |
|--------------------------------------------|-----------|----------|-------|--|
| 0xd150E46b725682cf9Dc9Cf7De347ef7E4289E48A | 50.00 ETH | 1        | 0     |  |

## 9. Transaction:

| BACK           | TX 0x763143ca23bb9e5356370c11ee553bbb67f6210a225b77c66d03ed36f18d6b10 | CONTRACT CALL                                                     |
|----------------|-----------------------------------------------------------------------|-------------------------------------------------------------------|
| SENDER ADDRESS | 0xd150E46b725682cf9Dc9Cf7De347ef7E4289E48A                            | TO CONTRACT ADDRESS<br>0x00ec53Fe2cFc7d4B87363B5659ab8017DEaF19e2 |
| VALUE          | 50.00 ETH                                                             | GAS USED<br>21000                                                 |
|                |                                                                       | GAS PRICE<br>30000000                                             |
| TX DATA        |                                                                       | GAS LIMIT<br>21000                                                |
|                |                                                                       | MINED IN BLOCK<br>1                                               |
| <b>EVENTS</b>  |                                                                       |                                                                   |

### GOEthereum(Geth)

3. Create Ethereum node using Geth (**GoEthereum**) and create genesis block and create your personal private Ethereum blockchain. And use IPC to interact with Geth node to perform following task: create account, transfer funds using send transaction, mine the block, show the account balance before and after the mining the block, show the specific block details and access chain details.

#### 1. Start Geth:

```
Administrator: Command Prompt - geth --datadir=~/chaindata/
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:/Users/dhruv/OneDrive/Desktop/private-chain
C:/Users/dhruv/OneDrive/Desktop/private-chain>geth --datadir=~/chaindata/
INFO [10-26|10:30:57.033] Starting Geth on Ethereum mainnet...
INFO [10-26|10:30:57.043] Bumping default cache on mainnet... provided=1024 updated=4096
INFO [10-26|10:30:57.054] Maximum peer count ETI=50 LES=0 total=50
WARN [10-26|10:30:57.073] Sanitizing cache to Go's GC limits provided=4096 updated=2683
INFO [10-26|10:30:57.079] Set global gas cap cap=50,000,000
INFO [10-26|10:30:57.089] Allocated trie memory caches clean=402.00MiB dirty=670.00MiB
INFO [10-26|10:30:57.103] Allocated cache and file handles database=C:/Users/dhruv/OneDrive/Desktop/private-chain/chaindata\geth\chaindata cache=1.31GiB handles =8192
INFO [10-26|10:30:57.159] Opened ancient database database=C:/Users/dhruv/OneDrive/Desktop/private-chain/chaindata\geth\chaindata\ancient readonly=false
INFO [10-26|10:30:57.170] Writing default main-net genesis block nodes=12356 size=1.78MiB time=45.5779ms gcnodes=0 gcsizes=0.00B gctime=0s livemode=1 livesize=0.00B
INFO [10-26|10:30:57.361] Persisted trie from memory database config="(ChainID: 1 Homestead: 1150000 DAO: 1920000 DAOSupport: true EIP150: 2463000 EIP155: 2675000 EIP158: 2675000 Byzantium: 4370000 Constantinople: 7280000 Petersburg: 7280000 Istanbul: 9069000, Muir Glacier: 9200000, Berlin: 12244000, London: 12965000, Engine: ethash)"
INFO [10-26|10:30:57.382] Disk storage enabled for ethash caches dir=C:/Users/dhruv/OneDrive/Desktop/private-chain/chaindata\geth\ethash count=3
INFO [10-26|10:30:57.387] Disk storage enabled for ethash DAGs dir=C:/Users/dhruv/AppData/Local/Ethash count=2
INFO [10-26|10:30:57.392] Initialising Ethereum protocol network=1 divergence=<nil>
INFO [10-26|10:30:57.400] Loaded most recent local header number=0 hash=d4e567...cb8fa3 td=17,179,869,184 age=55y7mo1w
INFO [10-26|10:30:57.406] Loaded most recent local full block number=0 hash=d4e567...cb8fa3 td=17,179,869,184 age=55y7mo1w
INFO [10-26|10:30:57.410] Loaded most recent local fast block number=0 hash=d4e567...cb8fa3 td=17,179,869,184 age=55y7mo1w
WARN [10-26|10:30:57.413] Failed to load snapshot, regenerating err="missing or corrupted snapshot"
INFO [10-26|10:30:57.415] Rebuilding state snapshot root=d7f897..0ff9544 accounts=0 slots=0 storage=0.00B elapsed=5.541ms
INFO [10-26|10:30:57.424] Resuming state snapshot generation transactions=0 accounts=0
INFO [10-26|10:30:57.424] Regenerated local transaction journal
INFO [10-26|10:30:57.444] Gasprice oracle is ignoring threshold set threshold=2
WARN [10-26|10:30:57.454] Error reading unclean shutdown markers error="leveldb: not found"
INFO [10-26|10:30:57.454] Starting peer-to-peer node instance=Geth/v1.10.9-stable-eae3b194/windows-amd64/go1.17
INFO [10-26|10:30:57.498] New local node record seq=1,729,918,857,488 id=ff46e56086a592f6 ip=127.0.0.1 udp=30303 tcp=30303
INFO [10-26|10:30:57.507] Started P2P networking self=enode://38459ddc1c704323bf5a6329b9b4ab6673de7efd490c13f0c38abfe54b7381acae0f5f3a94fef4de26
INFO [10-26|10:30:57.518] IP endpoint opened url=\.\.\pipe\geth.ipc
INFO [10-26|10:30:57.550] Generated state snapshot accounts=8893 slots=0 storage=409.64KiB elapsed=131.889ms
```

#### 2. Open New Terminal and type geth attach ipc:\\.\geth.ipc

```
C:\Windows\System32>geth attach ipc:\\.\pipe\geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.9-stable-eae3b194/windows-amd64/go1.17
at block: 0 (Thu Jan 01 1970 05:30:00 GMT+0530 (IST))
  datadir: C:/Users/dhruv/OneDrive/Desktop/private-chain/chaindata
  modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

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

#### 3. eth.accounts

```
> eth.accounts
[]
```

#### 4. personal.newAccount()

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x57453470dbfcc5028e04fcfd670e2160edffd7441"
>
```

#### 5. miner.start()

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



**RemixIDE — Injected Provider- public Test Network (Goerli, Sepolia) or Ganache**

4. Write a solidity smart contract for performing following task using **remixIDE** and deployed it on public **test network — Goerli / Sepolia** using **Injected provider** environment.

- a. To transfer funds (ethers) from user account to contract account.
- b. To withdraw funds (ethers) from contract account to user account.
- c. To apply restriction that only owner of the contract can withdraw funds (ethers) from contract account to his/her user account.

1. FundContract.sol:

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

```
contract FundContract {
    address public owner;
    constructor() {
        owner = msg.sender;
    }

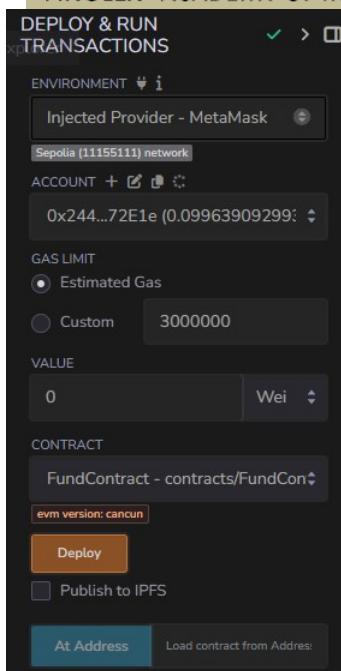
    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can withdraw funds");
        _;
    }

    function deposit() external payable {
        require(msg.value > 0, "Must send some ether to deposit");
    }

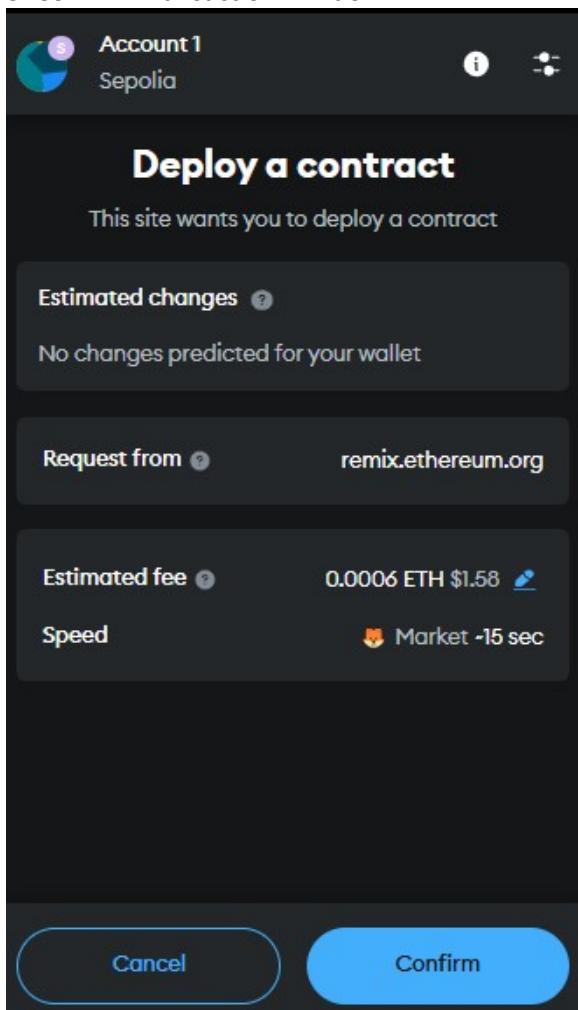
    function withdraw(uint256 amount) external onlyOwner {
        require(amount <= address(this).balance, "Insufficient balance in contract");
        payable(owner).transfer(amount);
    }

    function getContractBalance() external view returns (uint256) {
        return address(this).balance;
    }
}
```

2. Deployment Window:



### 3. Confirm Transaction Window



## 4. MetaMask:

The screenshot shows the MetaMask wallet interface on the Sepolia testnet. The top bar indicates the network as "Sepolia" and the account as "Account 1" with address "0x244aF...72E1e". Below the header, the balance is displayed as "0.099 SepoliaETH". There are three main tabs: "Tokens", "NFTs", and "Activity", with "Activity" currently selected. Under the "Activity" tab, a recent transaction is listed: "Contract deployment Confirmed" on Oct 26, 2024. The transaction details show it was successful, with a gas price of 0 and a gas limit of 6947560. The transaction hash is 0x8a3ee2f2b0962a2b69be2ecb80719c008e467be9e2c4de87650f75f501e1d61e.

## 5. Transaction Details:

This screenshot provides a detailed view of a Sepolia Testnet transaction. The transaction hash is 0x8a3ee2f2b0962a2b69be2ecb80719c008e467be9e2c4de87650f75f501e1d61e. The status is "Success" and it has 17 block confirmations. It was timestamped 3 mins ago (Oct-26-2024 05:55:36 AM UTC). The transaction action was a call to 0x60806040 method 0x244aF935...436A72E1e. The transaction originated from 0x244aF93579e9c381fc23383023F1E5B436A72E1e and was sent to 0x31e485d23e4e42a0297928863149d83a9b4ec7ae. The value transferred was 0 ETH, and the transaction fee was 0.00064791759514174 ETH. The gas price was 0.

5 .Write a smart contract to calculate the compound interest and deploy it on **Ganache** using **injected provider**.

**CompoundIntrest.sol:**

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

contract CompoundInterestCalculator {

    function calculateCompoundInterest(
        uint256 _principal,
        uint256 _rate,
        uint256 _time,
        uint256 _n
    ) public pure returns (uint256) {
        require(_principal > 0, "Principal must be greater than 0");
        require(_rate > 0, "Rate must be greater than 0");
        require(_time > 0, "Time must be greater than 0");
        require(_n > 0, "Number of times interest applied must be greater than 0");

        uint256 base = (1e18 + (_rate * 1e18 / (100 * _n)));
        uint256 exponent = _n * _time;

        uint256 amount = (_principal * power(base, exponent)) / 1e18;
        uint256 compoundInterest = amount - _principal;

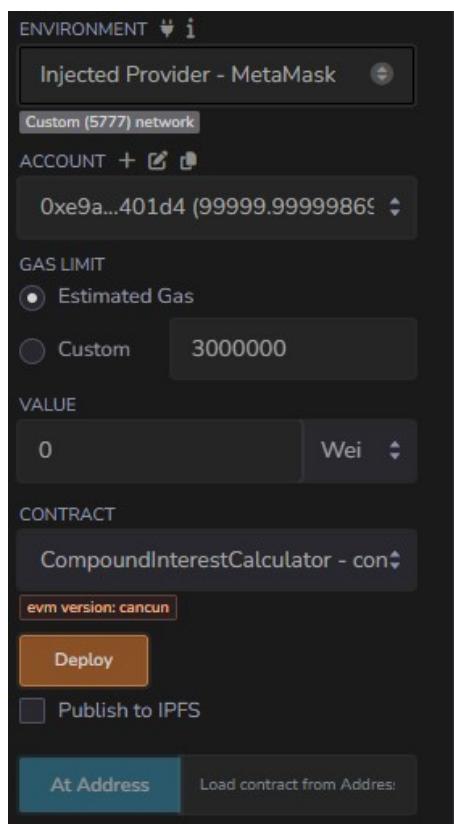
        return compoundInterest;
    }

    function power(uint256 base, uint256 exponent) public pure returns (uint256) {
        uint256 result = 1e18; // Start with 1 (in 18 decimal precision)

        while (exponent > 0) {
            if (exponent % 2 == 1) {
                result = (result * base) / 1e18; // Multiply and maintain precision
            }
            base = (base * base) / 1e18; // Square the base and maintain precision
            exponent /= 2; // Divide exponent by 2
        }

        return result;
    }
}
```

Deploy the contract



Transactions on Ganache

| BLOCK<br>1 | MINED ON<br>2024-10-28 05:41:47 | GAS USED<br>433465 | 1 TRANSACTION   |
|------------|---------------------------------|--------------------|-----------------|
| BLOCK<br>0 | MINED ON<br>2024-10-28 05:28:26 | GAS USED<br>0      | NO TRANSACTIONS |

Transaction

| GAS USED<br>433465                                                                           | GAS LIMIT<br>6721975                                                   | MINED ON<br>2024-10-28 05:41:47 | BLOCK HASH<br>0x8daefaa6c23e114b126c7aa2c1f43ea5b9c35219020e9dbbe1d358f0afdf6f599 |
|----------------------------------------------------------------------------------------------|------------------------------------------------------------------------|---------------------------------|-----------------------------------------------------------------------------------|
| <b>TX HASH</b><br><b>0x6efbfcc2582d2b1af7b3374b9c954f8f734e4164704d9bcd179a66bb0d2825c90</b> |                                                                        |                                 |                                                                                   |
| FROM ADDRESS<br>0xe9aB1E9D008bBF4bDA5677c75c106F3f258401d4                                   | CREATED CONTRACT ADDRESS<br>0x758427DF77dF50Dad3A9CE0580B36ca1D1C1433b | GAS USED<br>433465              | VALUE<br>0                                                                        |

**Truffle - Ganache**

6. Build and test decentralized application (Dapp) for Election Voting System on the local Ethereum Blockchain Network Ganache using truffle suite.

VotingSystem.sol

// SPDX-License-Identifier: MIT

```
pragma solidity >=0.5.0 <0.8.27;

contract VotingSystem {
    struct Candidate {
        uint256 id;
        string name;
        uint256 voteCount;
    }

    mapping(address => bool) public voters;
    mapping(uint256 => Candidate) public candidates;

    uint256 public candidateCount;

    event votedEvent(uint256 indexed _candidateId);

    constructor() public {
        addCandidate("Candidate 1");
        addCandidate("Candidate 2");
        addCandidate("Candidate 3");
    }

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

    function vote(uint256 _candidateId) public {
        require(!voters[msg.sender], "You already voted");
        require(_candidateId > 0 && _candidateId <= candidateCount);
        voters[msg.sender] = true;
        candidates[_candidateId].voteCount++;
        emit votedEvent(_candidateId);
    }

    function getCandidateDetails(uint _candidateId) public view returns (uint, string memory, uint) {
        return (candidates[_candidateId].id, candidates[_candidateId].name, candidates[_candidateId].voteCount);
    }
}
```

2\_deploy\_contracts.js

```
var VotingSystem = artifacts.require("./VotingSystem.sol");
module.exports = function(deployer)
{
    deployer.deploy(VotingSystem);
};
```

### 3. Open Terminal and type truffle migrate

```
C:\Users\dhruv\OneDrive\Desktop\blockchain-toolkit>truffle migrate
Compiling your contracts...
=====
> Compiling ./contracts/VotingSystem.sol
> Artifacts written to C:\Users\dhruv\OneDrive\Desktop\blockchain-toolkit\build\contracts
> Compiled successfully using:
  - solc: 0.5.16+commit.9c3226ce.Emscripten.clang

Starting migrations...
=====
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)

2_deploy_contracts.js
=====
Deploying 'VotingSystem'
-----
> transaction hash: 0x38112f2c9c024b2d8124f7d1726d477436b2415caa5f419188594933ec6327e2
> Blocks: 0          Seconds: 0
> contract address: 0x2Ae052232B70e9E5Bf8b9f97BFcd550b6F5E6776
> block number:     1
> block timestamp:  1729942291
> account:          0x3C0BF12BDf9B792e3E309152847427f5C6E33226
> balance:          99.99998040997
> gas used:         653001 (0x9f6c9)
> gas price:        0.003 gwei
> value sent:       0 ETH
> total cost:       0.000001959003 ETH

> Saving artifacts
-----
> Total cost:       0.000001959003 ETH

Summary
=====
> Total deployments: 1
> Final cost:        0.000001959003 ETH
```

### 7. Build and test decentralized application, (**Dapp**) for Banking System on the local Ethereum Blockchain Network **Ganache** using **truffle suite**.

BankingSystem.sol:

```
pragma solidity >=0.5.16;
contract SimpleBank {
    // State variable to store the balance
    uint256 private balance;
    // Constructor to initialize balance
    constructor () public {
        balance = 0;
    }
    // Function to add (deposit) amount to the balance
    function addAmount(uint256 amount) public {
        balance += amount;
    }
    // Function to withdraw amount from the balance
    function withdrawAmount(uint256 amount) public {
        require(amount <= balance, "Insufficient balance");
        balance -= amount;
    }
    // Function to check the remaining balance
    function checkBalance() public view returns (uint256) {
        return balance;
    }
}
```

```
2_deploy_contracts.js
const SimpleBank = artifacts.require("SimpleBank");

module.exports = function (deployer) {
  deployer.deploy(SimpleBank);
};
```

```
c:\ Administrator: C:\WINDOWS\system32\cmd.exe
C:\Users\dhruv\OneDrive\Desktop\blockchain-toolkit>truffle migrate
Compiling your contracts...
=====
> Compiling ./contracts/BankingSystem.sol
> Artifacts written to C:\Users\dhruv\OneDrive\Desktop\blockchain-toolkit\build\contracts
> Compiled successfully using:
  - solc: 0.5.16+commit.9c3226ce.Emscripten.clang

Starting migrations...
=====
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)

2_deploy_contracts.js
=====

Deploying 'SimpleBank'
-----
> transaction hash: 0x0966874b64a469b2a3b7ffba5b9681e1dc2942d156c9781c20cedd4be2608ae6
> Blocks: 0          Seconds: 0
> contract address: 0x0346bd526acEBB459a88549285E6610B65438e37
> block number:     1
> block timestamp:  1730023929
> account:          0xe8b670f7e703Ab9C1A8b61aA17cbF776f98705ffa
> balance:          99.99999573139
> gas used:         142287 (0x22bcf)
> gas price:        0.003 gwei
> value sent:       0 ETH
> total cost:       0.000000426861 ETH

> Saving artifacts
-----
> Total cost:      0.000000426861 ETH

Summary
=====
> Total deployments: 1
> Final cost:        0.000000426861 ETH

C:\Users\dhruv\OneDrive\Desktop\blockchain-toolkit>
```

**Conclusion:** Developed and deployed Dapp in Ethereum.