Truffle suite

What is Truffle Suite?

- Truffle Suite is a development framework for
- Ethereum-based decentralized applications(dApps).
- Truffle Suite provides a complete development environment for building, testing, and deployingEthereum smart contracts.
- It supports popular Ethereum networks like Ganachefor local development and testing, and itcan be
- integrated with frameworks like Drizzle or Web3.jstointeract with smart contracts from the front end.


TRUFFLE

What is Truffle?

- Truffle is a popular development framework for
- Ethereum-based decentralized applications (dApps). Itprovides a suite of tools and utilities thatsimplify the process of building, testing, and deploying smart
- contracts on the Ethereum blockchain.
- Truffle supports multiple Ethereum networks, includingthe local development network (Ganache)and various public and private testnets and mainnets.

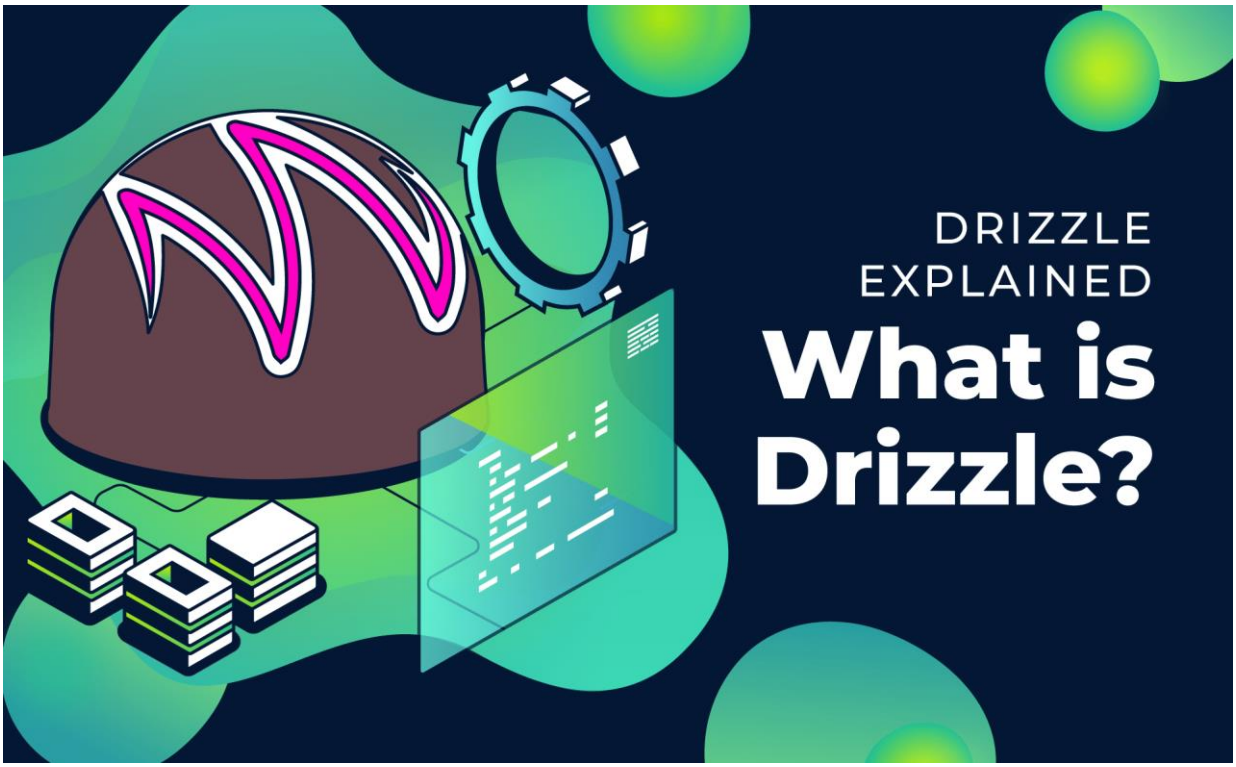
What is Ganache?

- Ganache is a personal blockchain network that serves as a local development environment for Ethereum- based applications.

Ganache generates a set of preloaded accounts with fake Ether (ETH) that developers can use for testing their applications.

- Ganache offers several features tailored for developers,such as detailed transaction logs, stacktraces, and gas usage information.
- Ganache seamlessly integrates with popular
- development tools and frameworks, includingTruffle and Remix IDE.
- It can be easily configured as the target network for deployment and testing, making it a preferredchoice for Ethereum developers.

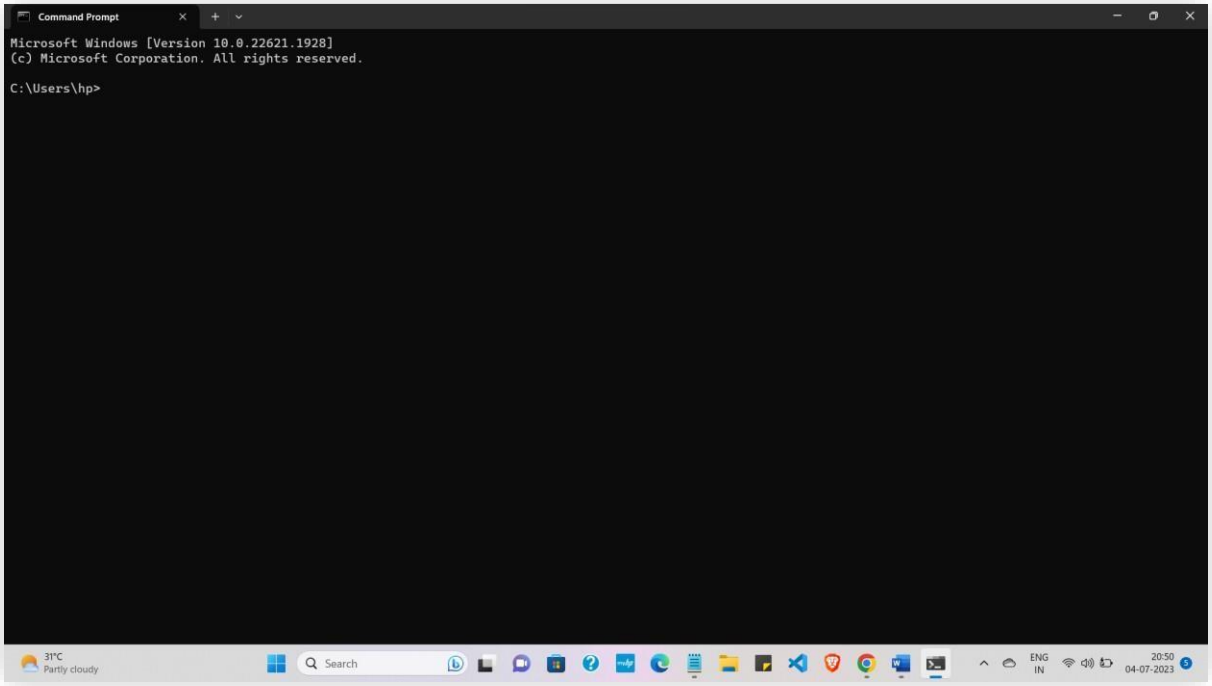
What is Drizzle?



- Drizzle is a library that simplifies the process ofintegrating Ethereum smart contracts with user
- interfaces (UIs) in decentralized applications (dApps).
- Drizzle integrates seamlessly with Web3.js, a JavaScriptlibrary for interacting with Ethereum.
- Drizzle includes an event system that enables
- developers to subscribe to and handle events emitted by smart contracts. This allows the dApp torespond to contract events, such as new transactions or changes incontract state, in real- time.

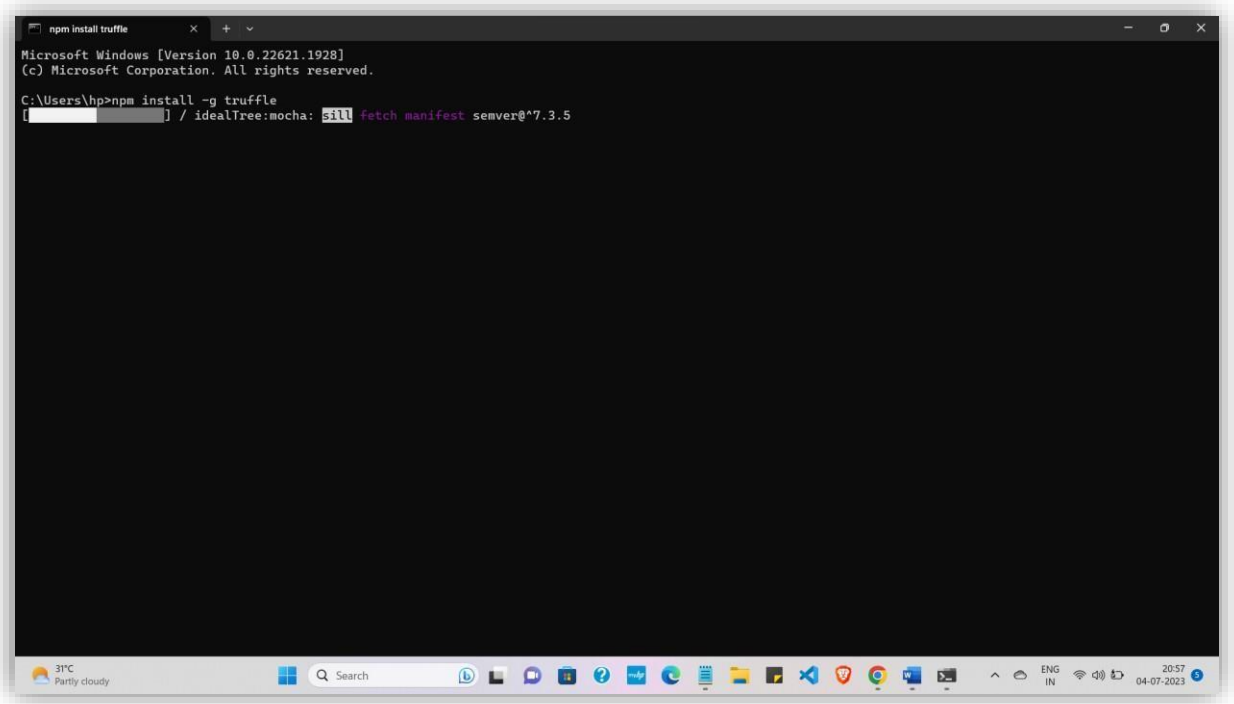
Installation of Truffle

For Installing Truffle in your personal computer, first we need to open Command Prompt.



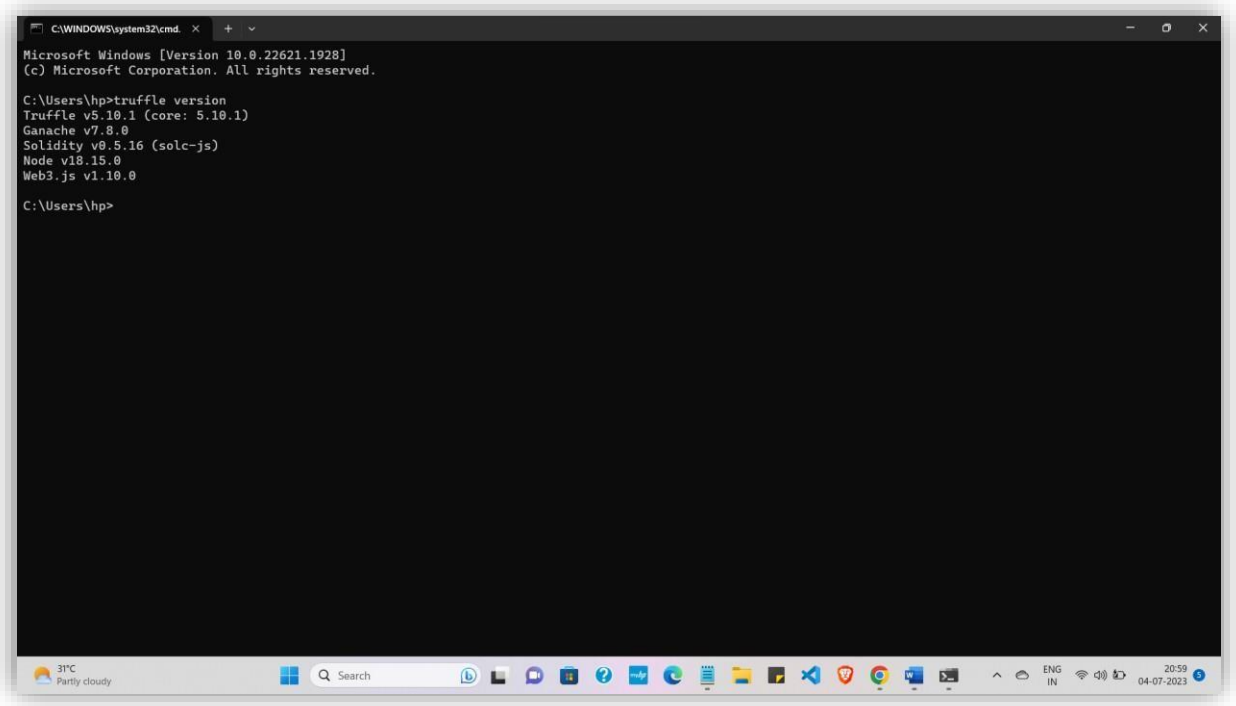
- Next run the below command:
- **npm install -g truffle**
- -g refers that we are installing this truffle package globally.
- Note: Before running the command check whether node is installed or not by running the command **node version.**
- If you are getting the node version then you can run the command otherwise you need to install node.js using the link **https://nodejs.org/en/download/**

After running the above command your installationprocess will start like shown below.



To check whether Truffle package is installed or not, weneed to run the below command :

truffle version



If you are able to see the above screen output then youare successfully installed truffle package.

Getting started with Truffle:

Installing Truffle:

- Before you can use Truffle, you will have to installitusing npm.
- Open a terminal and use the following to install itglobally.

npm install -g truffle

Creating a Project:

- To use most Truffle commands, you need to run themagainst an existing Truffle project.
- So the first step is to create a Truffle project.
- You can create a bare project template, but for those just getting started, you can use Truffle boxes, which are example applications andproject templates.
- We will use MetaCoin box, which creates a token thatcan be transferred between accounts.

Step:1

Create a new directory for your Truffle project:

```
mkdir MetaCoin
cd MetaCoin
```

Step:2

Download "unbox" the MetaCoin box using below command:

```
truffle unbox metacoin
```

Important Note:

If you want to create a bare Truffle project with nosmart contracts included, use the below command:

truffle init

Step:3

Once this operation of downloading the truffle project.metacoin is completed, you will now have a project. structure with the following items:

- MetaCoin/
- contracts/--> Directory for solidity contracts.
- migrations/-->Directory for scriptable deploymentfiles.
- test/-->Directory for test files for testing yourapplication and contracts.
- truffle.js/-->Truffle configuration file.

Exploring the Truffle Project:

Step:1

Open the **contracts/MetaCoin.sol** file in a text editor.

This is a smart contract (written in Solidity) that createsa MetaCoin token.

Note that this also references another Solidity file **contracts/ConvertLib.sol** in the same directory.

Step:2

Open the **migrations/1_deploy_contracts.js** file. This file is the migration (deployment) script.

Step:3

Open the **test/TestMetaCoin.sol** file.

This is a test file written in Solidity which ensures thatyour contract is working as expected.

Step:4

Open the **test/metacoin.js** file.

This is a test file written in JavaScript which performs a similar function to the Solidity test above.

The box does not include one, but Truffle tests can alsobe written in Type Script Programming Language.

Step:5

Open the **truffle-config.js** file.

This is the Truffle configuration file, for setting network information and other project-related settings.

Testing the Truffle Project:

Step:1

In a terminal, run the Solidity test:

truffle test./test/TestMetaCoin.sol

You will see the following output:

<u>TestMetacoin</u>

- o testInitial Balance Using DeployedContract (71ms)
- o testInitialBalanceWithNewMetaCoin

(59ms)2 passing (794ms)

Step:2

In a terminal, run the JavaScript test:

truffle test./test/metacoin.js

You will see the following output:

Contract: MetaCoin

- should put 10000 MetaCoin in the first account
- should call a function that depends on a linked library

(40ms) should send coin correctly

(129ms).3 passing (255ms)

- To run all tests, you can simply run **truffle test.**
- Because development is commented out in **truffle-**
- **config.js, truffle test** will spin up and tear down alocaltest instance **(ganache).**
- These two test were run against the contract with descriptions displayed on what tests are supposedtodo.

Compiling the Truffle Project:

Step:1

Compile the smart contracts using the belowcommand:

truffle compile

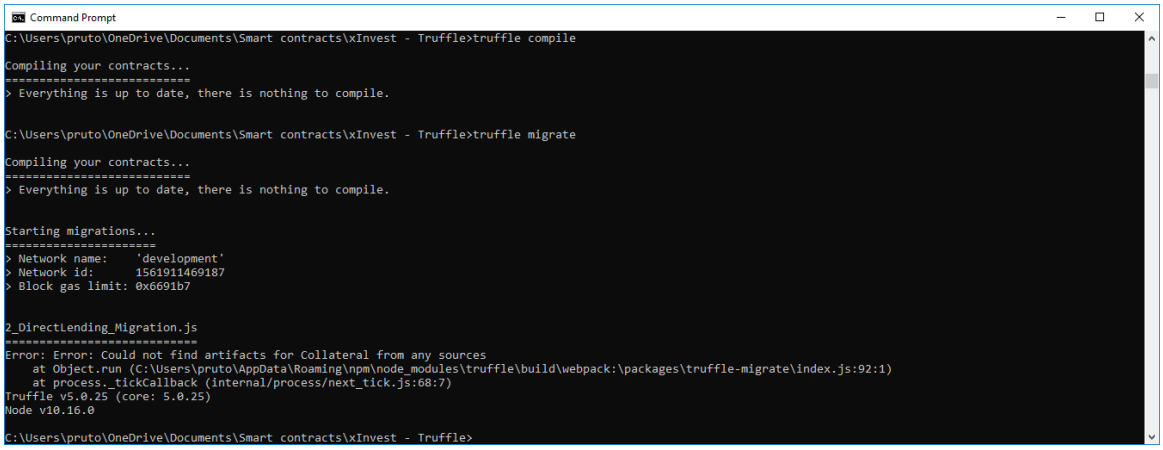
You will see the following output:

Compiling.\contracts\Convert Lib.sol...
Compiling.\contracts\MetaCoin.sol.
..
Compiling.\contracts\Migrations.sol...

Writing artifacts to .\build\contracts

Migrating with Truffle Develop:

- To deploy our smart contracts, we are going to needtoconnect to a Blockchain.



- Truffle has a built-in personal Blockchain that can

be used for testing.

- This Blockchain is local to your system and does notinteract with the main Ethereum network.
- You can create this Blockchain and interact with itusingTruffle Develop.

- The following are the steps to be followed:
- **Step:1**

- Run Truffle Develop using the below command:
- **truffle develop**

- You will see the following information:
- Truffle Develop started at http://127.0.0.1:9545/

- **Accounts:**
- 0xe63792504e730554f691e38506b792e8e3525d32
- 0x8dc5cb22604cf4ad279cb0939b615008dbeaab6c
- 0x057d4cc3259726b60611bded5133bb346a0e09c9
- 0xba1e6b0a8d368fb58209ea428ddebdef6e54d757
- 0x69351eb18b55b641bb0bdee132c1dd4560f3468f
- 0x31066e60be2548950c1aad6e093a19133069886

9

- 0xf526476994059f8da2a12d3e89bcb2d3ade328a8
- 0xebe855bb98afa3c50d7d45d4213358860d5349d2
- 0x1f744743fee8890e4142765523be610c29c502ca
- 0x16706084d0362d0c38cc4e61faf9b51857e579bc

- **Private Keys:**
- (0)
- 4ac6856287d21cebe6d3840488d200611dab30f0a28ff1f31172cd9ba6322e87
- (1)
- 4a402f692b3c6c4f8e3c4572423db83eb5b97205e1 644e0281f3982848e40743
- (2)
- 72814c6fe577b2efbbfd023525b062809eb93491b2 348e233d339c9e46cfde0f
- (3)
- 946c4685401777bf66938ee6d4afdc21d2a2b2463a0 7e9820195d0872c56e7b3
- (4)
- 7e85cd6398aedad7e6288fd772edc367fd9d8cd5b569c416e3f6df8995f68712
- (5)
- 9c54fb0602454551b32c73b550973fb48841d1290a4 81
- 3bb59ca29307075adec

- (6)
- 06db886838165acb4698147558951b874064337ab 06e2b797715c8293e2c59aa
- (7)
- 11f574ed35312913e0175ed57489e6cf9ed2a471d 59 10
- 4c4d85ae50800bf

  2 295(8)
- 2fc3f987b4efb41c2ed8beeb89c6c05fb839783bf1 02 982
- 406beca839412

  c 0bc(9)
- 0befedc2864a6eee27053c6e50b1c5871e929b9a4 6a d1
- 3434790a55fe5b226bc

- Mnemonic: erosion song outer truck puppy mentiondumb lend sustain idea powder bottom

- Important : This mnemonic was created for you by Truffle. It is not secure.
- Ensure you do not use it on production blockchains,orelse you risk losing funds.
- Truffle(development)>

- This shows ten accounts (and their private keysthatcan be used when interacting with the Blockchain.

- **Commands to run for interacting with Blockchain:**
- On the Truffle Develop prompt, Truffle commandscanbe run by omitting the truffle prefix.
- For example, to run **truffle compile** on the prompt,type **compile.**
- The command to deploy the contracts to the
- Blockchain is **truffle migrate**, so at the prompt, type:
- **migrate**
- You will see the following output:

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

1_to-do-list.js
===============
```

```
   Replacing 'TodoList'
   ----------------------------
> transaction hash:
0x93c7e5cfe00122b6436f64fcab21ee512f5d1c7719f
ab ed12dc05630e21faef2


   > Blocks: 0         Seconds: 0
   > contract address:
0xaB2CC2abBa8f184107099245F8281d2BeC930
8Ab
   > block number:      1
   > block timestamp:   1688488368
   > account:
0xe63792504e730554F691e38506b792E8E3525
d32
   > balance:        99.99792423325
   > gas used:        615042 (0x96282)
   > gas price:       3.375 gwei
   > value sent:       0 ETH
   > total cost:       0.00207576675 ETH
   > Saving artifacts
   -------------------------------------------------------
   > Total cost:     0.00207576675 ETH
```

Summary

========

> Total deployments:1

> Final cost: 0.00207576675 ETH

This shows the transaction IDs and addresses of your deployed contracts.

It also includes a cost summary and real-time statusupdates.

Migrating with Ganache:

- While Truffle Develop is an all-in-one personal
- Blockchain and console, you can also use Ganache, adesktop application, to launch your personal
- Blockchain.
- Ganache can be a more easy-to-understand tool forthose new to Ethereum and the Blockchain, asit displays much more information front.
- The only extra step, aside from running Ganache, is
- that it requires editing the Truffle configuration filetopoint to the Ganache instance.

Installing Ganache:

Step:1

Download and install Ganache from the following link:

https://trufflesuite.com/ganache/

Step:2

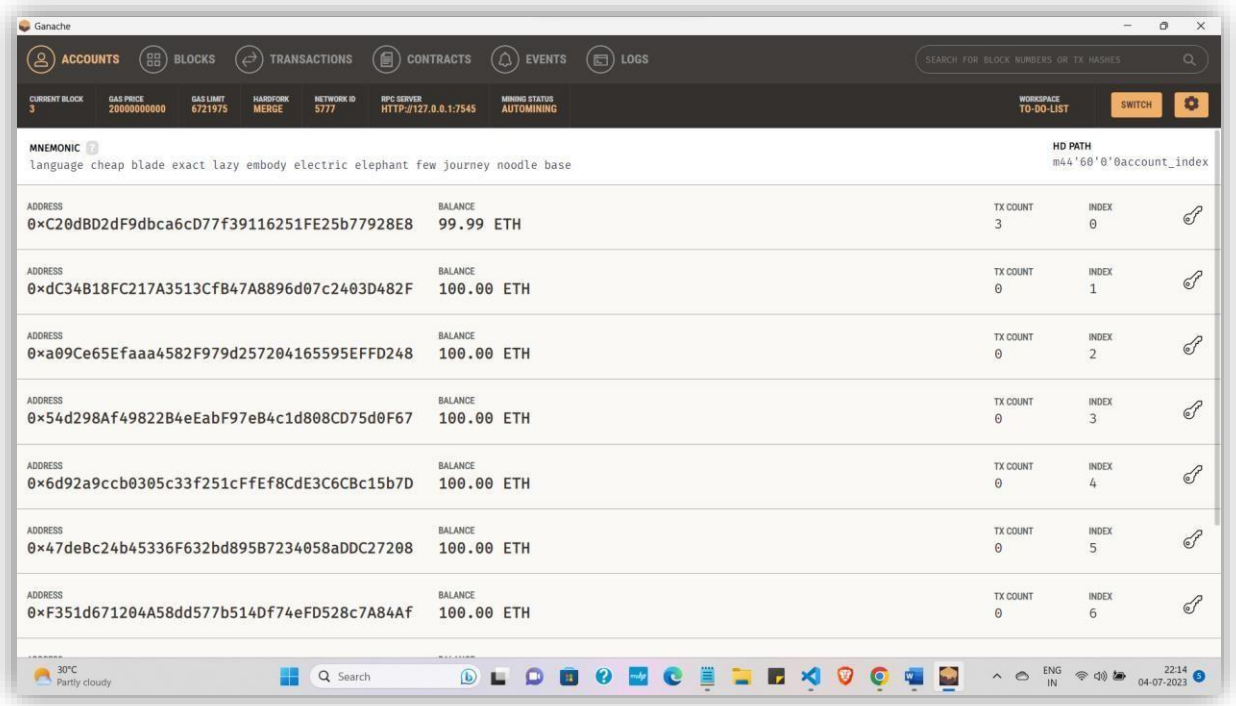
Open truffle-config.js in a text editor. Replace the content with the following:

```
module.exports = { networks: { development: { host: "127.0.0.1", port: 7545, network_id: "*"
}
}
};
```

This will allow a connection using Ganache's default connection parameters.

After replacing the content with the above code saveand close the file.

Now its time to launch our Ganache.



After launching the Ganache desktop application go tothe terminal.

On the terminal, migrate the contract to the Blockchaincreated by Ganache:

truffle migrate

You will see the following output:

Compiling your contracts...

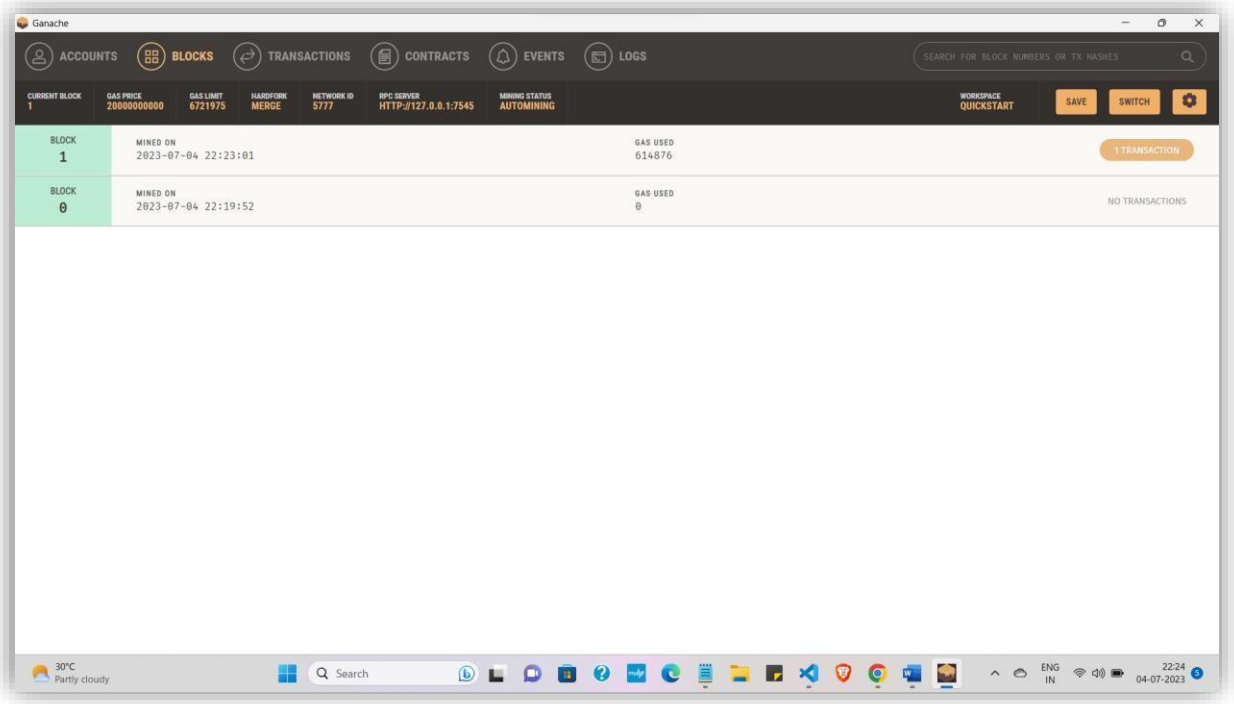
> Compiling ./contracts/Convert

Lib.sol 6194 Compiling

./contracts/MetaCoin.sol

```
> Compiling .\contracts\to-do-list.sol
> Artifacts written to
E:\BLOCKCHAIN\Truffle\build\contr
acts
> Compiled successfully using:
   - solc: 0.8.19+commit.7dd6d404.Emscripten.clang

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

1_to-do-list.js
===============

   Replacing 'TodoList'
   ----------------------

   > transaction hash:
0x27e557d2bb4cd1efe228fe5350a9cd84e55352a
933c3e060e2ca7cda0c7b2491
```

```
   > Blocks: 0        Seconds: 0
   > contract address:
0xa9C7226CcC171e9657BFE48B92f49cb352383
28D
   > block number:      1
   > block timestamp:   1688489581
   > account:
0x01b3d7B295191A1944bb9993426ffBC4970Bd
F39
   > balance:        99.9979247935
   > gas used:        614876 (0x961dc)
   > gas price:       3.375 gwei
   > value sent:       0 ETH
   > total cost:       0.0020752065 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:     0.0020752065

ETH Summary
=======
> Total deployments: 1
> Final cost:       0.0020752065 ETH
```



This shows the transaction IDs and addresses of your deployed contracts.

It also includes a cost summary and real-time statusupdates.

In Ganache desktop application, click the "Transactions" button to see that the transactions havebeen processed.

To interact with the contract, you can use the Truffleconsole.

The Truffle console is similar to Truffle Develop, exceptit connects to an existing Blockchain (in this case, the one is generated by Ganache).

Type the following command for getting the prompt ofdevelopment:

truffle console

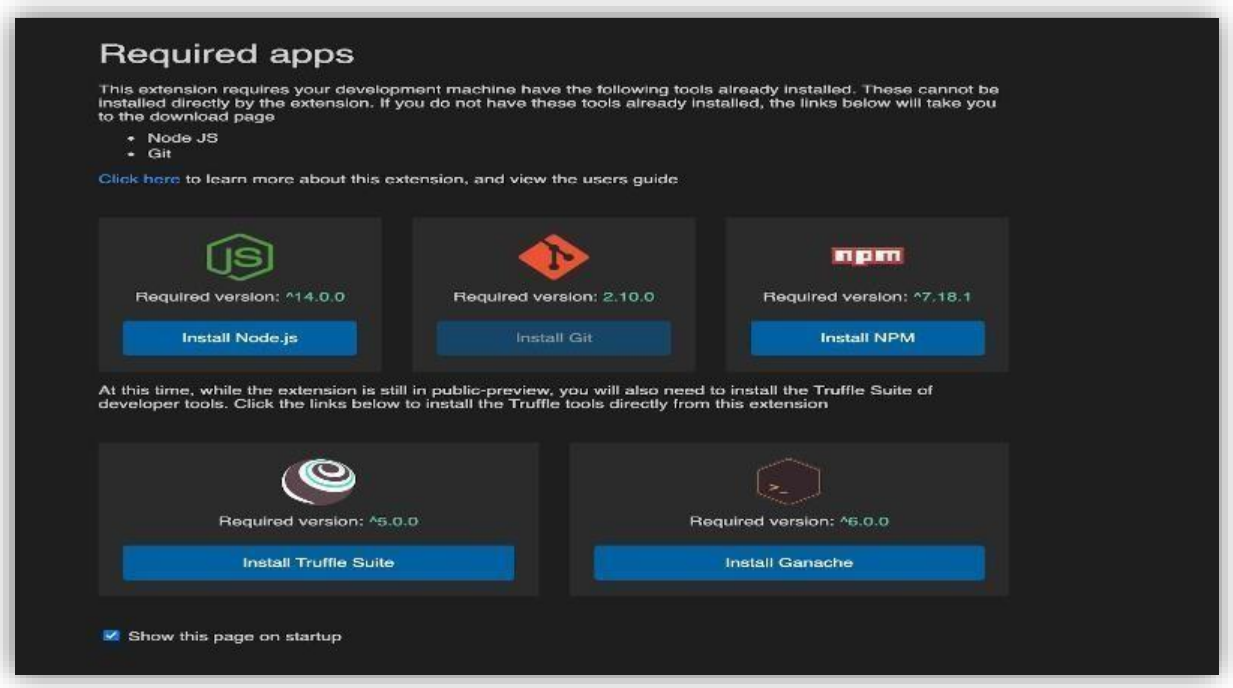
You will see the following prompt:

truffle(development)>

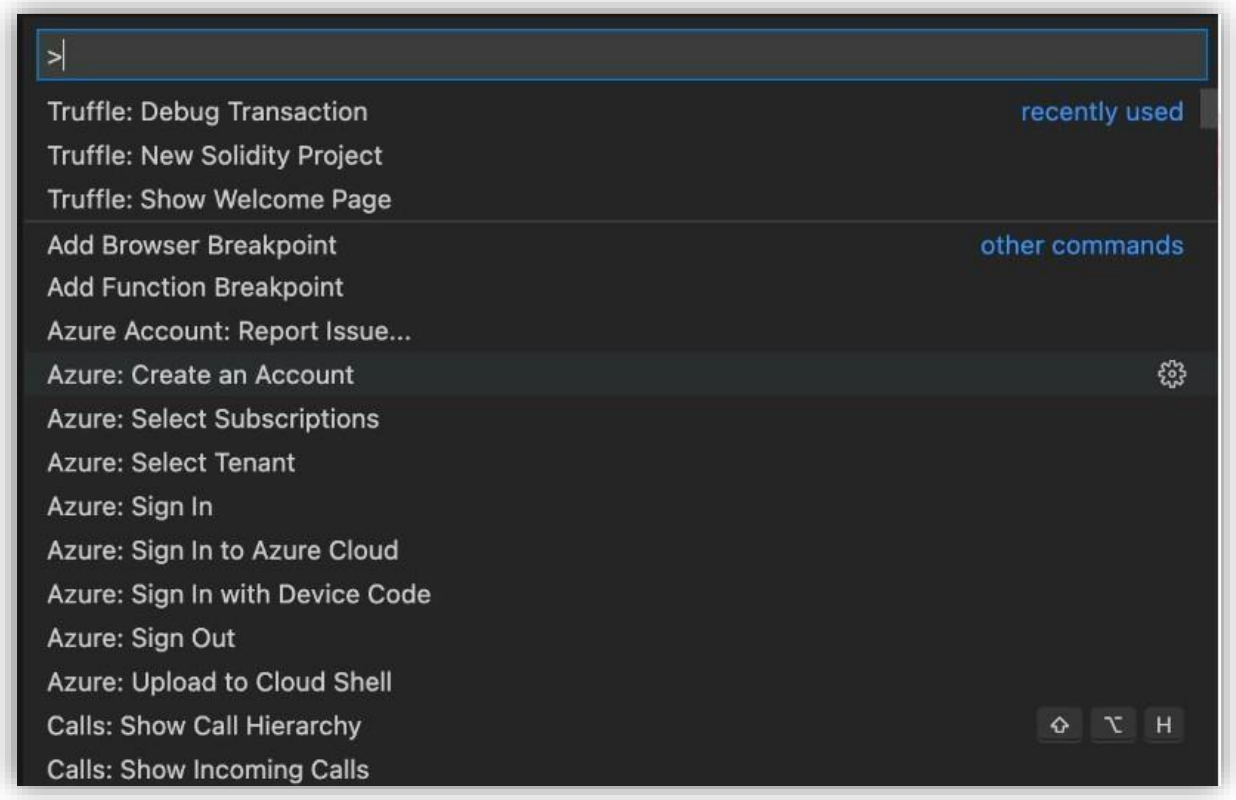
Installing the VS Code extension forTruffle:

- The easiest way to get started is by browsing the VSCode built-in marketplace tab.
- Search for Truffle for VS Code, and click the installbutton.

- Upon installation, you will get a prompt to downloadsome dependencies or upgrade theversions you already have to newer versions.

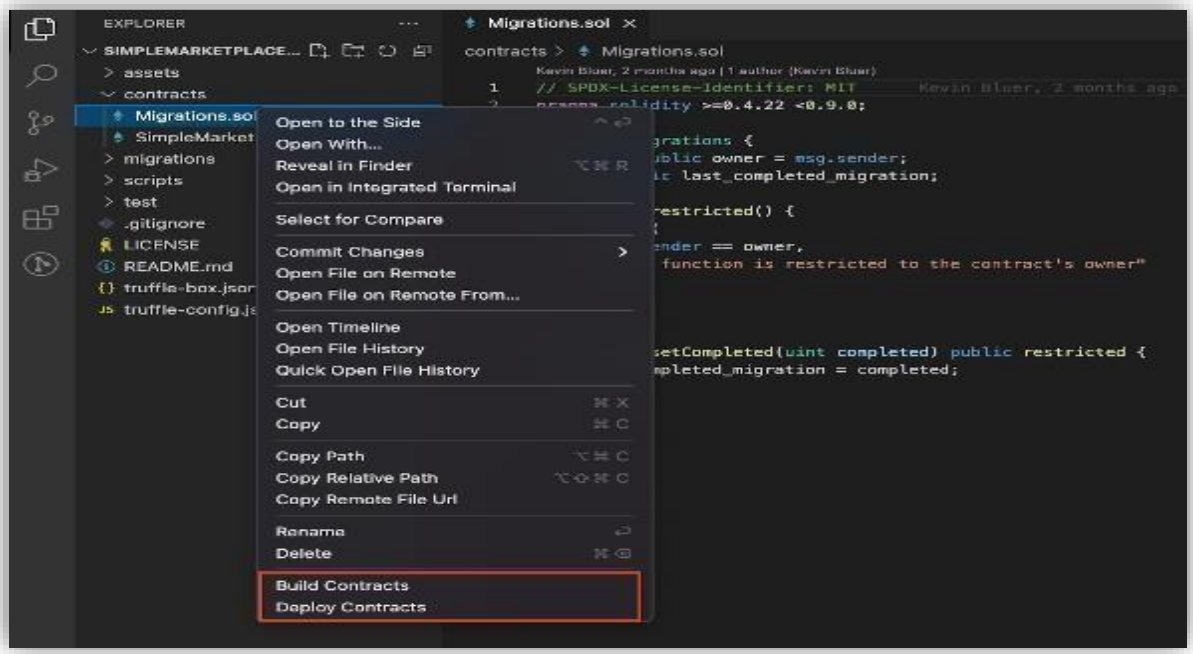


- The Truffle for VS Code extension surfaces in the following areas in VS Code, each with different setsofcommands:
- The VS Code command palette (SHIFT + CMD +P):Here, you can perform tasks such as:
- Creating a new Solidity Project.
- Creating and connecting to a new network.
- Starting and stopping Ganache local Blockchain.
- Debugging transactions using VS Code nativedebugger

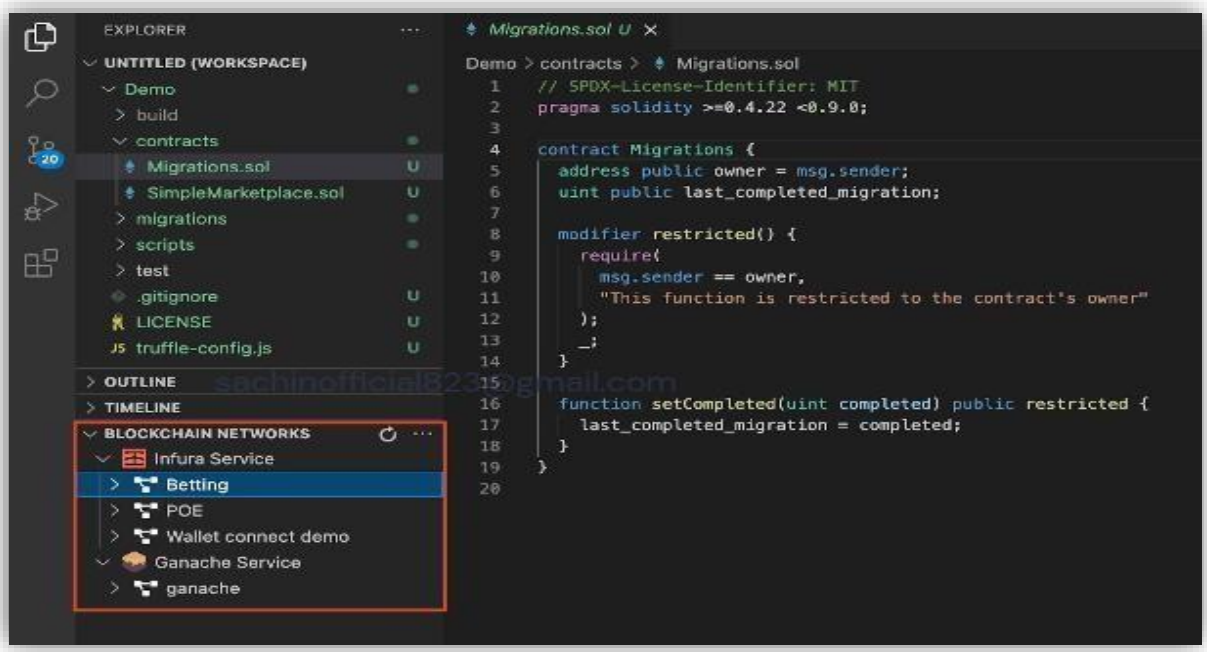


The VS Code context menu (Right-click on a .sol file). Here, you can perform tasks such as:

- Adding a new contract from OpenZeppelin.
- Building your Contracts.
- Deploying your contracts to your network of choice.

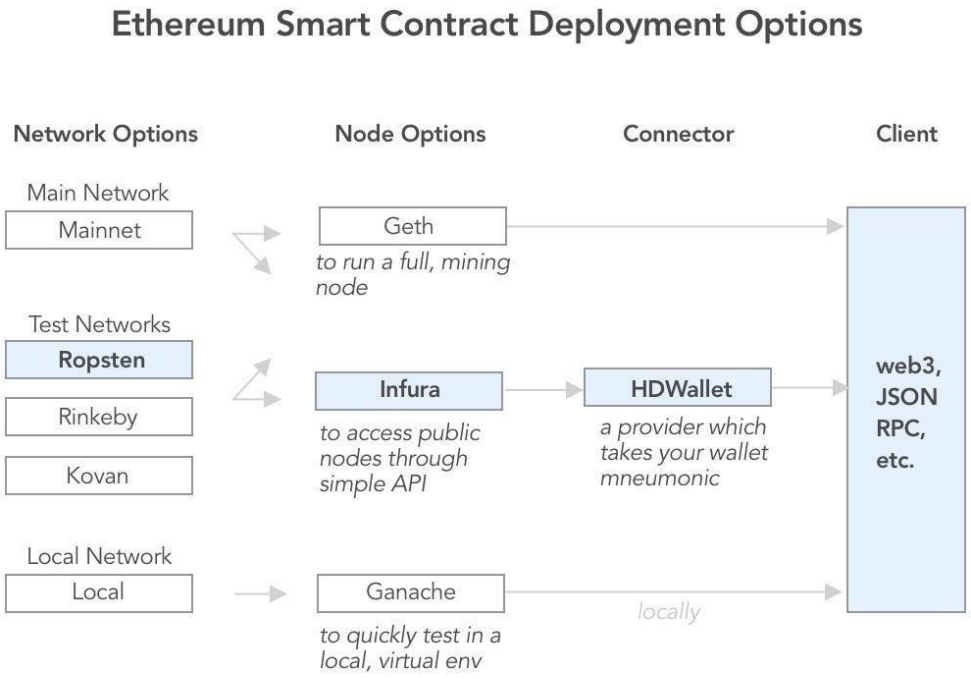


- The VS Code Tree view (Under the Explorer tab).
- Here, you can create and connect to an Infura or Ganache network without leaving the VS Code application.



Interacting with contract:

- Interact with the contract using the console in thefollowing ways:
- As of Truffle v5. the console supports async/await functions, enabling much simpler interactions withthecontrac



Step:1

- Begin by establishing both the deployed MetaCoincontract instance and the accounts created by eitherTruffle's built-in Blockchain or Ganache:
- truffle(development)> let instance = await

MetaCoindeployed

- truffle(development)> let accounts awaitweb3.eth.getAccounts()

Step:2

Check the metacoin balance of the account thatdeployed the contract:

- truffle (development)> let balance = await instance.getBalance(accounts[0])
- truffle (development)> balance.toNumber()
- Step:3
- See how much ether that balance is worth (and notethat the contract defines a metacoin to be worth 2 ether):
- truffle(development)> let ether = await instance.getBalanceInEth(accounts[0]) truffle(development)> ether.toNumber()
- Transfer some metacoin from one account to another:
- truffle(development)> instance.sendCoin(accounts[1],500);
- **Step:4**

- Check the balance of the account that received the metacoin:

- truffle(development)> let received = await instance.getBalance(accounts[1])
  truffle(development)> received.toNumber()

- **Step:5**
- Check the balance of the account that sent the metacoin:

- truffle(development)> let newBalance = await instance.getBalance(accounts[0])
  truffle(development)> newBalance.toNumber()