Course Outline

1. Cryptocurrency and Block chain
2. Delving into BlockChain
3. Bitcoin and Block chain
4. Bitcoin Mining
5. Ethereum
6. Setting up private Blockchain Environment using Ethereum Platform
7. Hyperledger
8. Setting up Development Program using Hyperledger composer
9. Create or Deploy our private Blockchain on Multi chain
10. Prospect of Blockchain

IntelliPaat

Setting up private Blockchain Environment using Ethereum Platform

Agenda

At the end of this session you will be able to:

- Understand Creating Smart Contracts on Ethereum Remix Browser
- Define MetaMask
- Describe Installing Blockchain
- Perform Go lang Installation
- Explain Creating Blockchain – Genesis Block
- List Genesis.json File Parameters
- Explain Making rules for our Blockchain
- Perform Mining Blockchain

Creating Smart Contracts on Ethereum Remix Browser

Creating testrpc.sh file

- To start with, we will create a file named testrpc.sh
- testrpc.sh file contains the account details. The details like account numbers and there balance can be altered as per your choice

Creating testrpc.sh file

- To create a file in ubuntu execute the following Command:
- Cat > sample.txt (Just replace the “sampletxt” with the name you want to use for your file)
- A file with specified name will be created

Example:

```
Cat > testrpc.sh
```

After the above command the cursor will move to the next line , allowing you to enter contents in the file

The file testrpc.sh is already there in the VM provided. So, you don’t have to create another file.

What is testrpc.sh file

- testrpc is a Node.js based Ethereum client for testing and development
- It uses ethereum js to simulate full client behavior and make developing Ethereum applications much faster
- It also includes all popular RPC functions and features (like events) and can be run deterministically to make development a breeze
- testrpc is written in Javascript and distributed as a Node package via npm

Executing testrpc file

- To execute testrpc file, execute the following command: **testrpc**
- Gets executed and the testrpc will be listening on localhost:8545

```
added 251 packages in 42.846s
blockchain@ubuntu:~$ testrpc
EthereumJS TestRPC v6.0.3 (ganache-core: 2.0.2)

Available Accounts
==================
(0) 0x384a877f902908af5ef427ef4a0ea163cf24ae01
(1) 0xba49ca98490e7c5269453d58b7204873c71ccca5
(2) 0x56333600faa67afd532214a75d3cae67019972d7
(3) 0x1a404cda5ff8d228f32d1502183cf088838c8e49
(4) 0xc3f9aa553d6736e83817f8aa39d0cd7d085b5501
(5) 0x4703c48bb320263f6804398444ce8f903229d244
(6) 0x90c3e607d12bde37c03dbfeef9db75e3af47a5b2
(7) 0x9f4e7345e991a2748aeab83db36458d45466c35a
(8) 0x71863843682a184613adc90dd7fb0885591cda6e
(9) 0xde4f63d02d53b25d7d122ec0c7462d30d4000eda

Private Keys
==================
(0) 464cdf8e6696a0e3b39b8b4d2cde5ea485a985f392fb481c4813b039f56f93fb
(1) 0c2cff6a473a16b498179c88cd9113da4bb3535e4808354c710f3654bd9cce01
```

Private keys of the accounts

Starting Block Explorer

- Click on the drop down icon near in front of the
- To launch the block explorer use the following command: **npm start**
- This command will install all the packages and will launch the block explorer at port 8000

```
File  Edit  View  Search  Terminal  Help
blockchain@ubuntu:~$ cd explorer/
blockchain@ubuntu:~/explorer$ ls
app  bower.json  e2e-tests  karma.conf.js  LICENSE  package.json  README.md
blockchain@ubuntu:~/explorer$ npm start

> EthereumExplorer@0.1.0 prestart /home/blockchain/explorer
> npm install

npm WARN deprecated bower@1.8.2: ...psst! Your project can stop working at any moment because its dependencies can change. Preve
his by migrating to Yarn: https://bower.io/blog/2017/how-to-migrate-away-from-bower/
npm WARN deprecated connect@2.30.2: connect 2.x series is deprecated
npm WARN deprecated graceful-fs@3.0.11: please upgrade to graceful-fs 4 for compatibility with current and future versions of No
s
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated minimatch@0.3.0: Please update to minimatch 3.0.2 or higher to avoid a RegExp DoS issue
npm WARN deprecated node-uuid@1.4.8: Use uuid module instead

> bufferutil@1.2.1 install /home/blockchain/explorer/node_modules/bufferutil
> node-gyp rebuild

make: Entering directory '/home/blockchain/explorer/node_modules/bufferutil/build'
  CXX(target) Release/obj.target/bufferutil/src/bufferutil.o
  SOLINK_MODULE(target) Release/obj.target/bufferutil.node
  COPY Release/bufferutil.node
make: Leaving directory '/home/blockchain/explorer/node_modules/bufferutil/build'

> utf-8-validate@1.2.2 install /home/blockchain/explorer/node_modules/utf-8-validate
> node-gyp rebuild

make: Entering directory '/home/blockchain/explorer/node_modules/utf-8-validate/build'
  CXX(target) Release/obj.target/validation/src/validation.o
```

Block Explorer Window

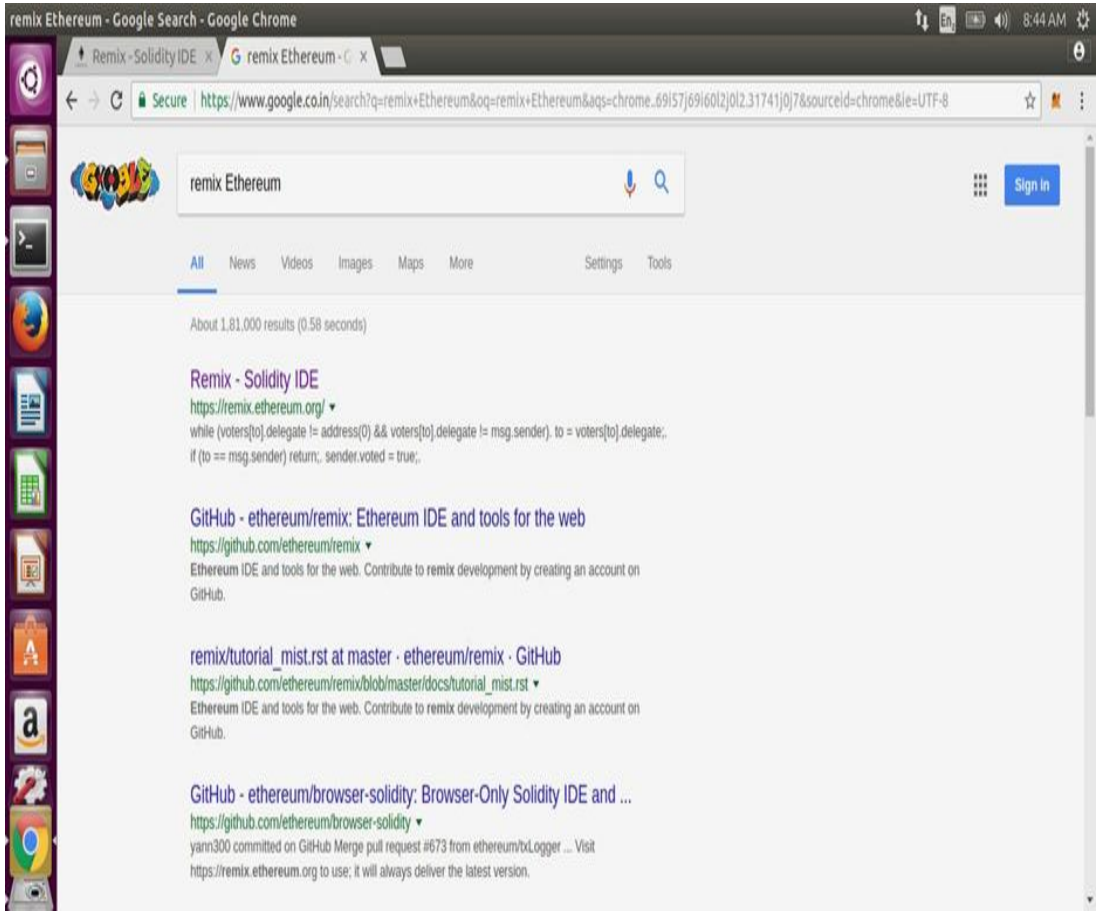
- **To view the block explorer:**
- Open the web browser in your V.M. (chrome recommended)
- Type : localhost:8000 in the browser tab and hit enter
- You will see the screen shown below

Remix IDE in Ethereum

- Remix, was known as Solidity Browser. It is a web browser based IDE that allows you to write Solidity smart contracts, then deploy and run the smart contract from a browser
- Download from https://remix.ethereum.org to us

To open the browser:

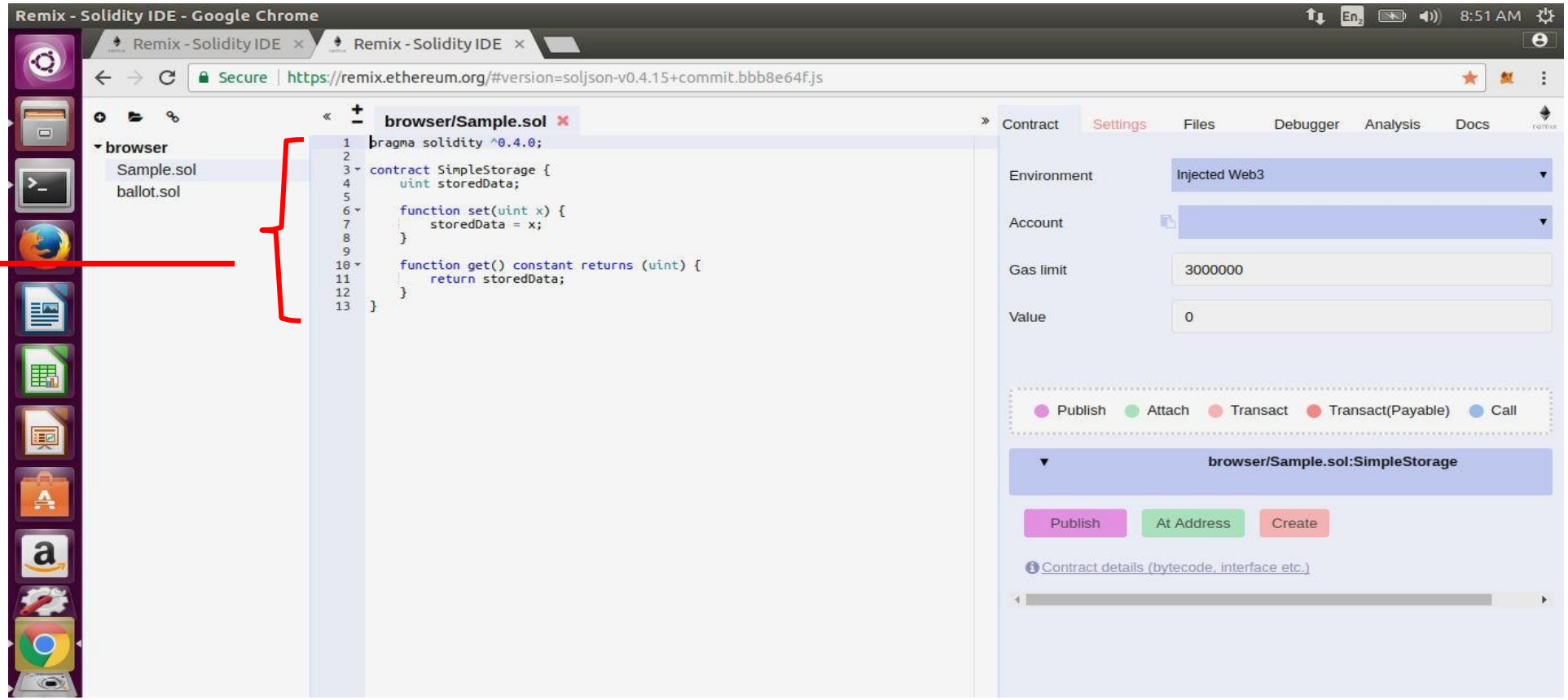
- Open a new tab in your browser and search “Remix Ethereum”
- Click on the first link that pops up



Remix Solidity browser dashboard

- The screen image shown below is the Solidity browser

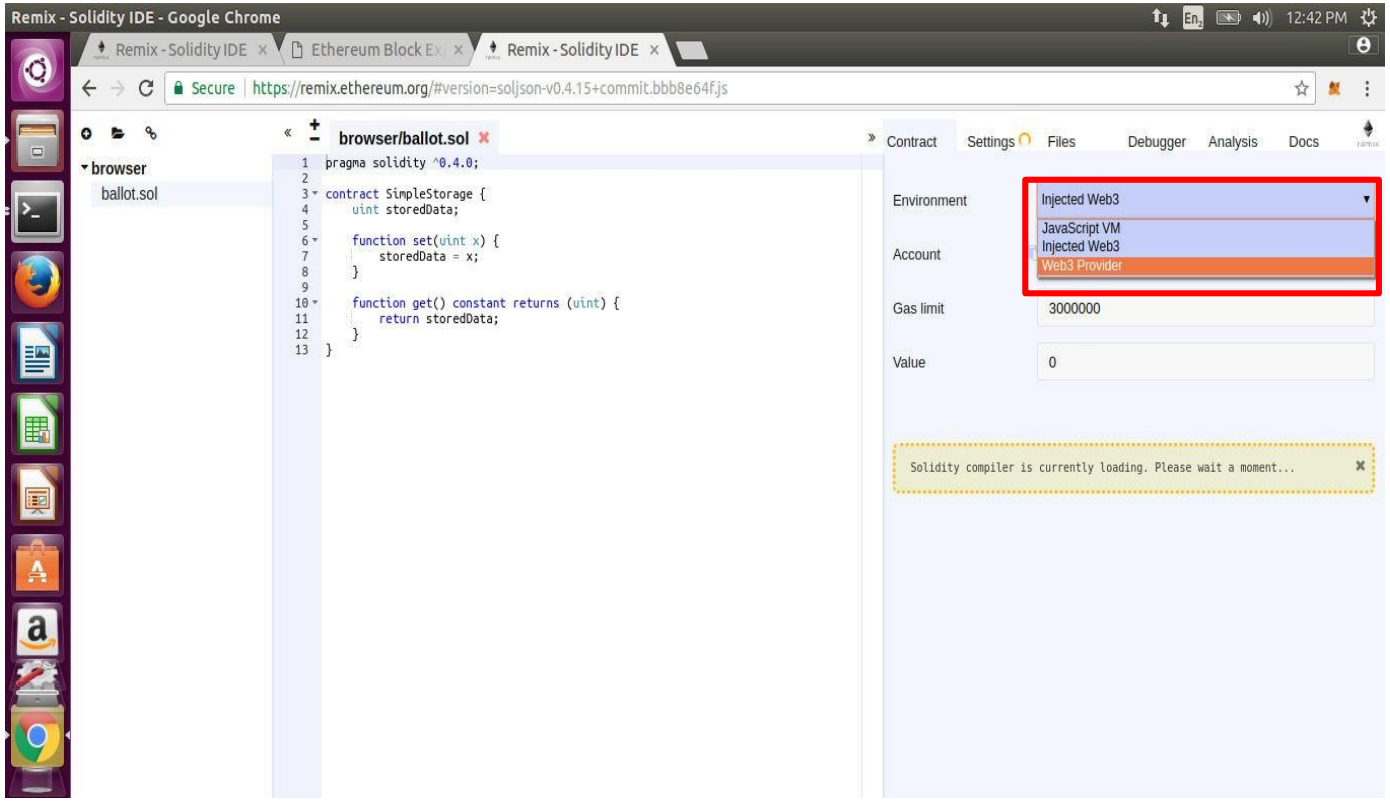
This is a sample of smart contract



Selecting the IDE Environment

Now, the first thing we need to do is select the environment for our solidity browser

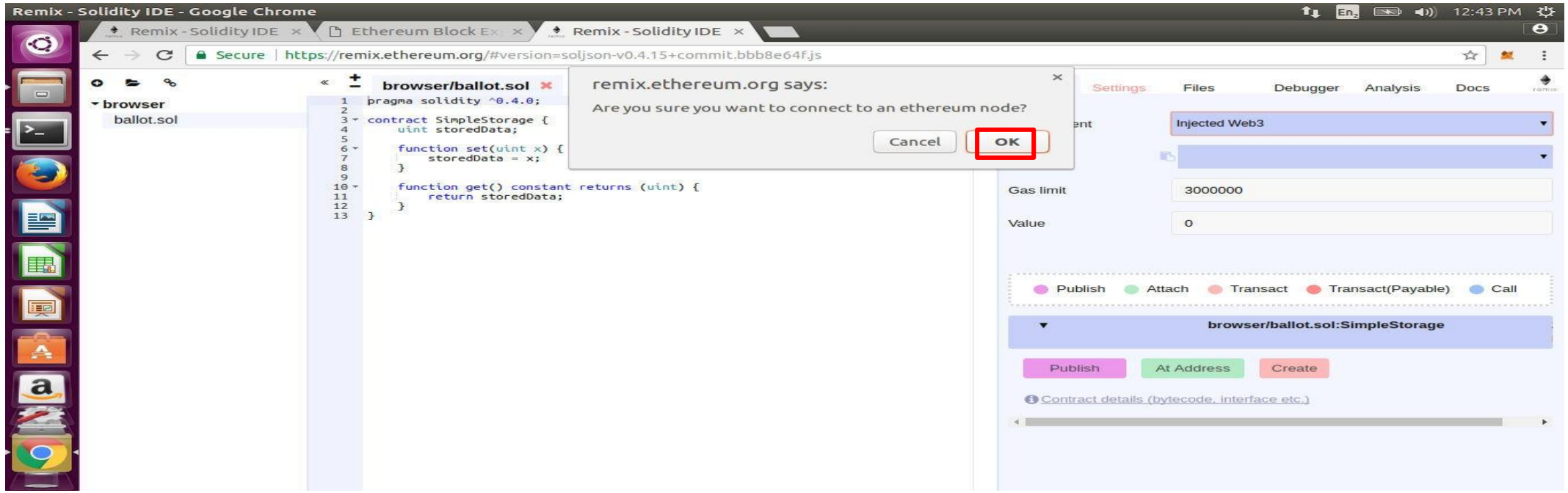
- Click on the drop down icon near in front of the “Environment”
- Three option will be seen
- Select “web3 provider” option from the list



Confirm web3 option

It will ask for your confirmation to connect to ethereum node

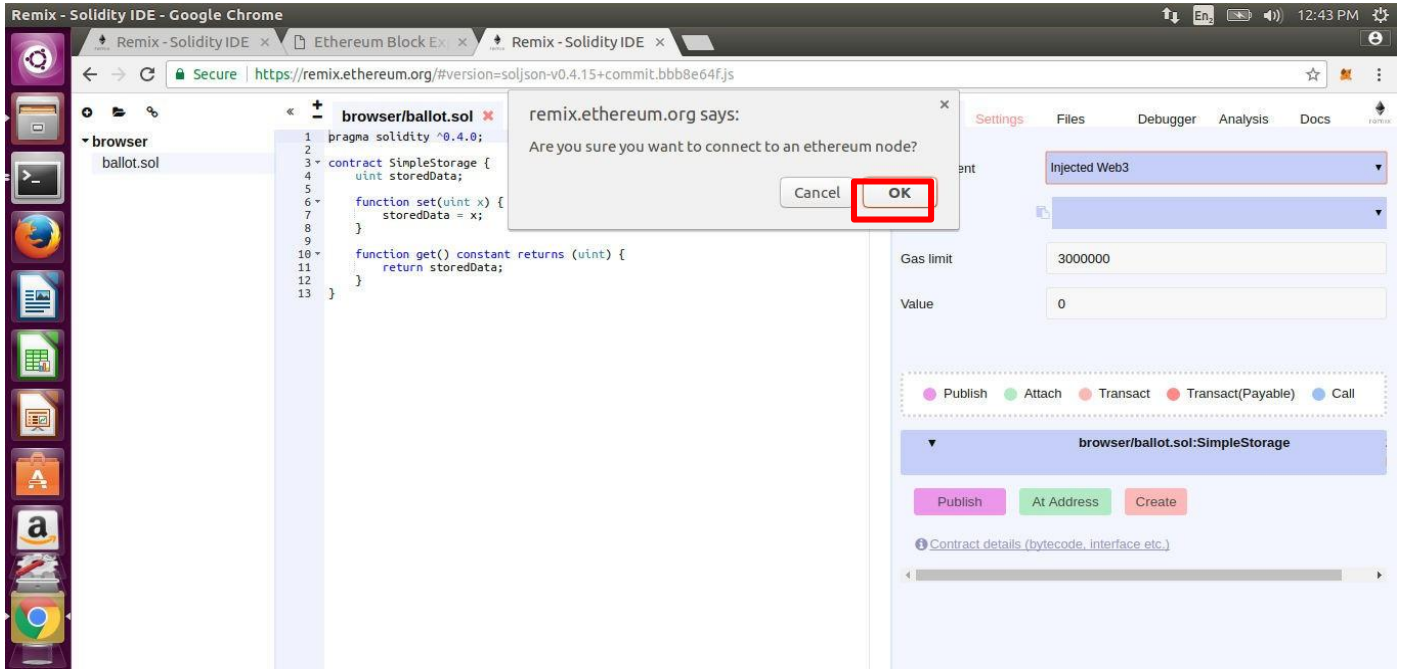
- Click ok on the pop up box and
- Select ok.



Selecting Web3 provider end point

Next, It will ask for the web3 provider endpoint, now this can be any address, like the ethereum blockchain server, or the address of the different machine you wish to connect, or it can be the localhost, which was generated viatestrpc

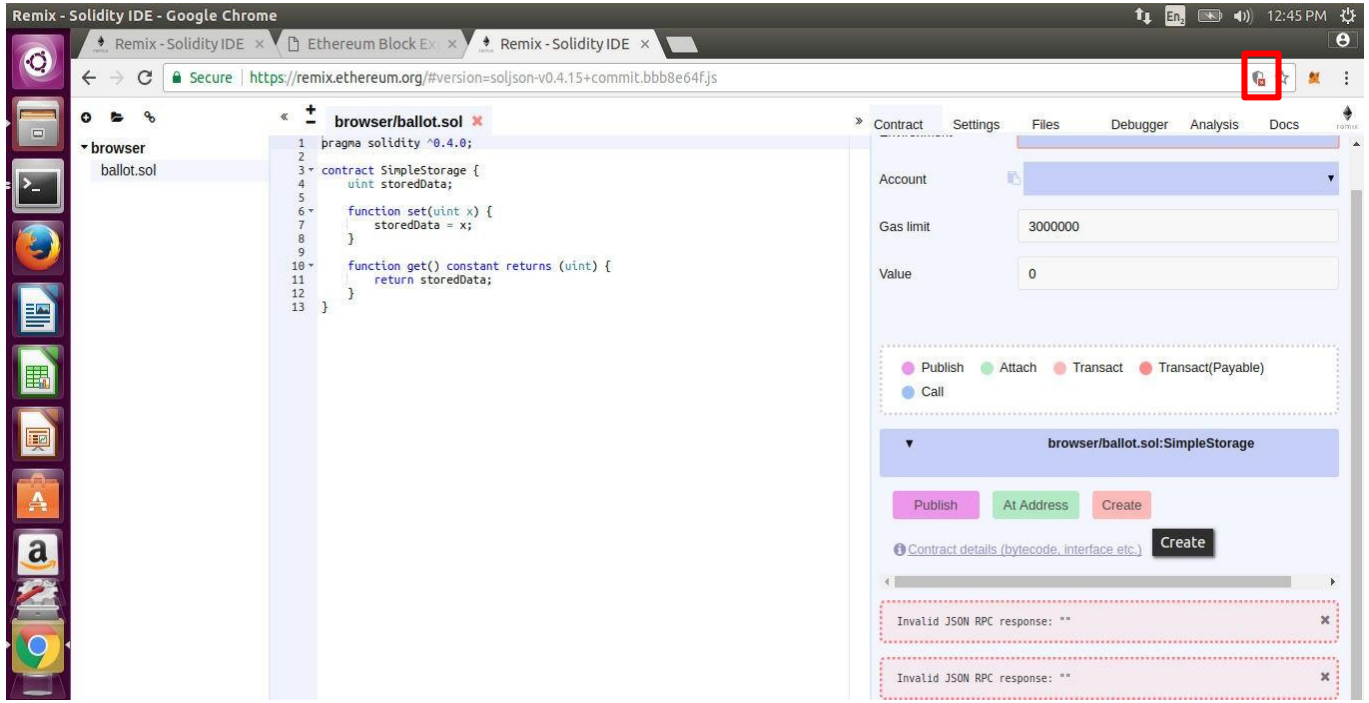
- Since we are using the browser for development and testing purpose, our endpoint will be the local host on the port where our testrpc is listening
- So here, just click ok button on the pop



The browser restricts web3 connection

On clicking the ok button (in previous slide) , the window shown below will pop up, connection to the local host is restricted by the browser

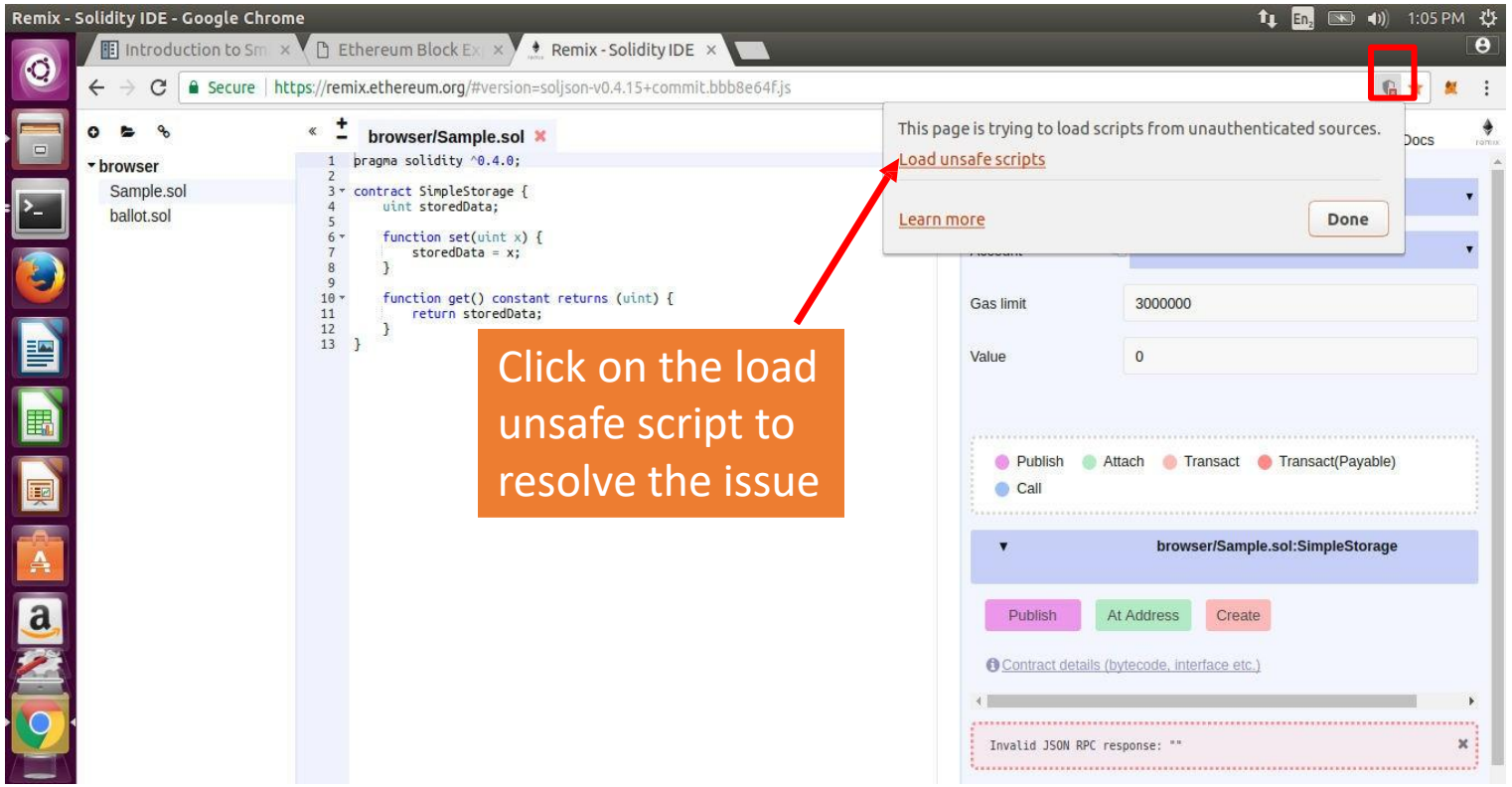
- The icon shown in red indicates the restriction. Let's see how we resolve it



Load Unsafe script

As, we have seen, the browser is restricting the connection, it is because of the unsafe script

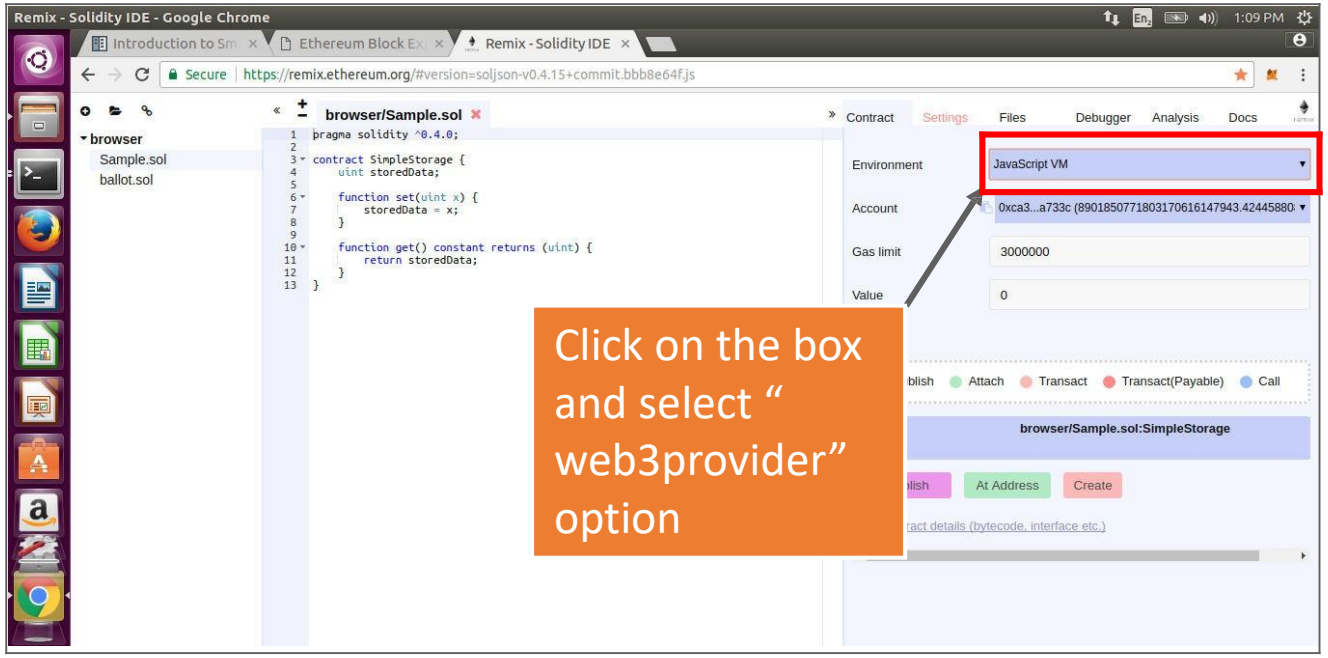
- As indicated, if you'll click the icon, it opens up with the message " The page is trying to load scripts from unauthenticated sources"
- Select the **"load unsafe scripts options"**
- This will now connect you to the local host port 8545



Set the “Environment”

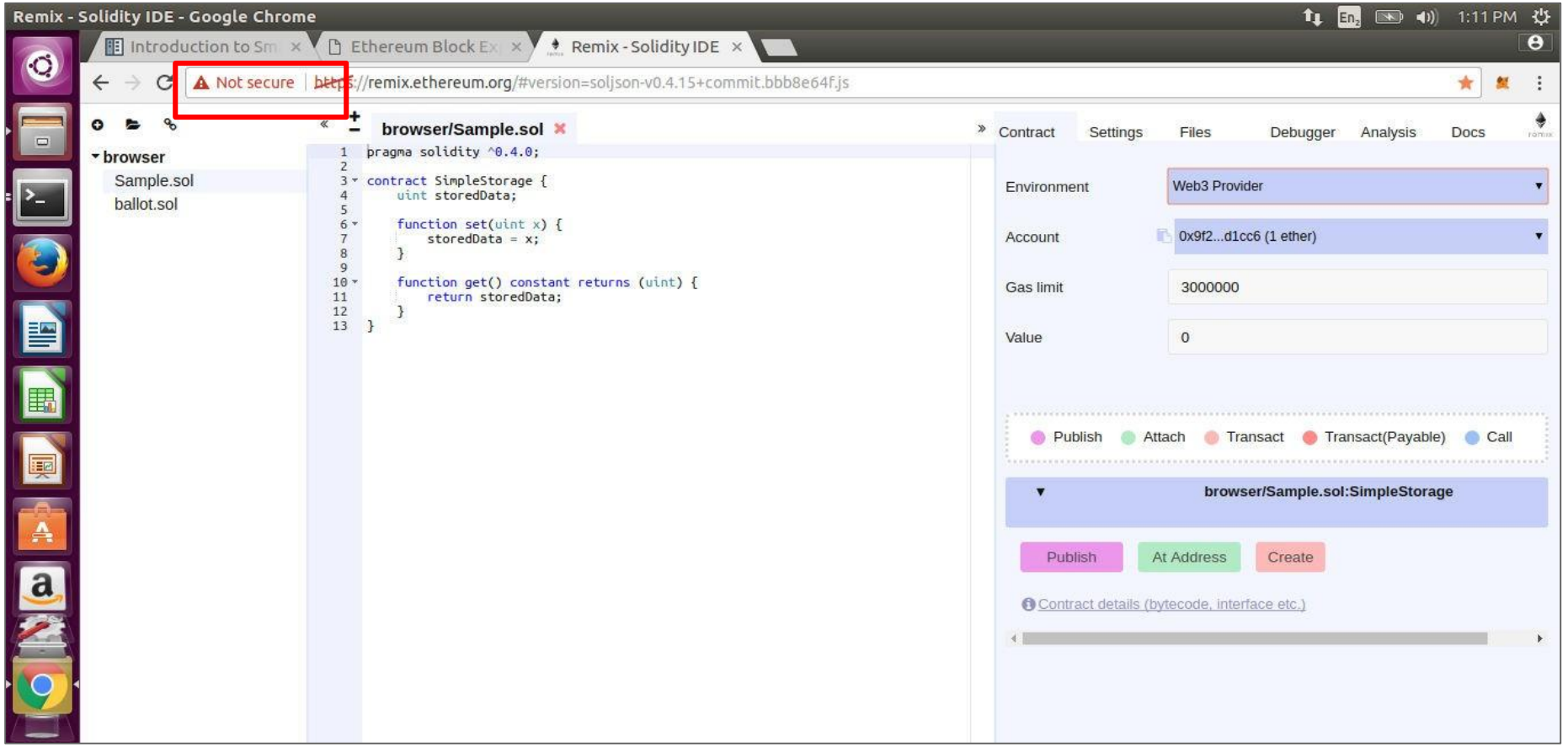
The connection with port is established as you can see the account balances in the window shown below, however we need to setup the “environment” again

- When you select the “load unsafe script” (previously),
- The page reloads and the environment gets changed.
- Now to change it to “web3 provider” , just click on the dropdown box and select the “web3 provider” option as done previously



Connection is now successful

The browser indicates “Not secure” connection, however it connects to the local host port

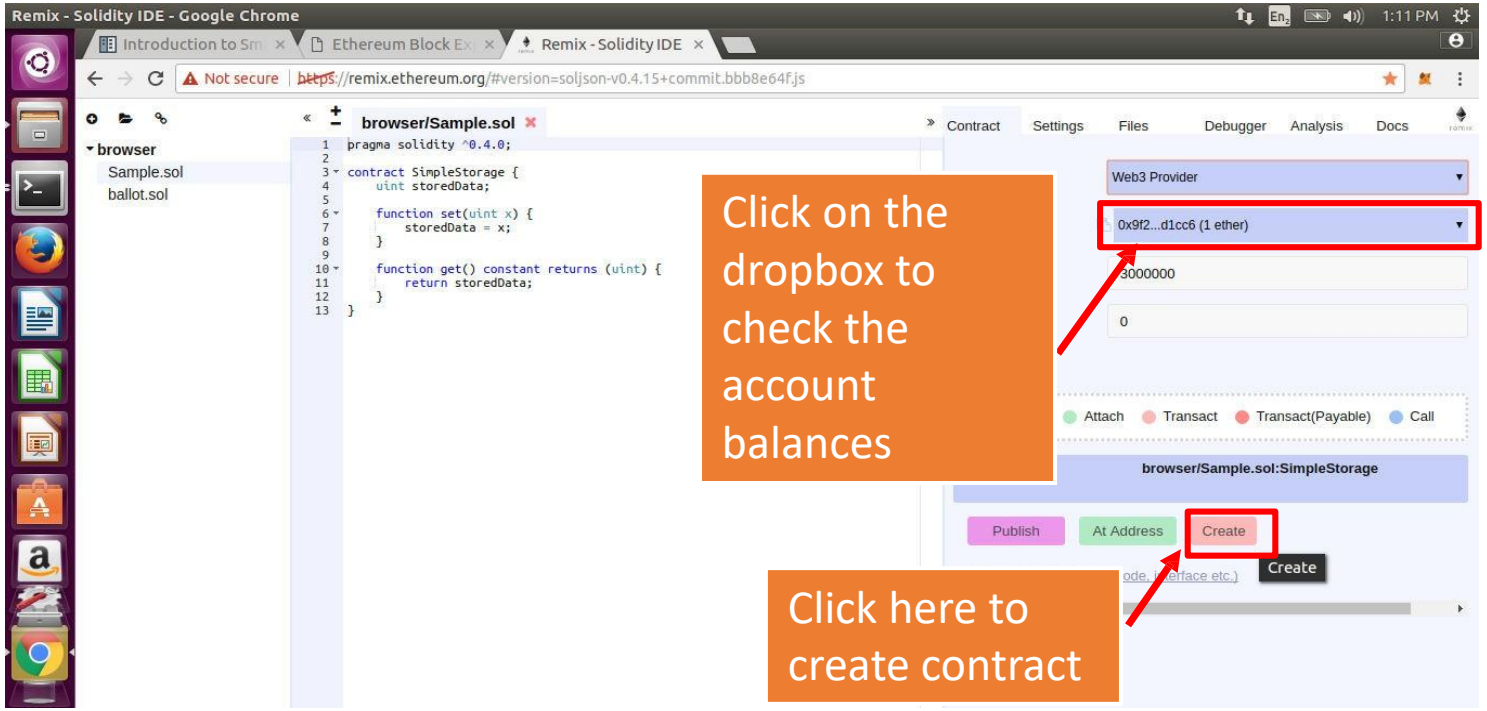


Creating Smart Contract

Now that we are connected to the port, we can create our very first smart contract

To create contract:

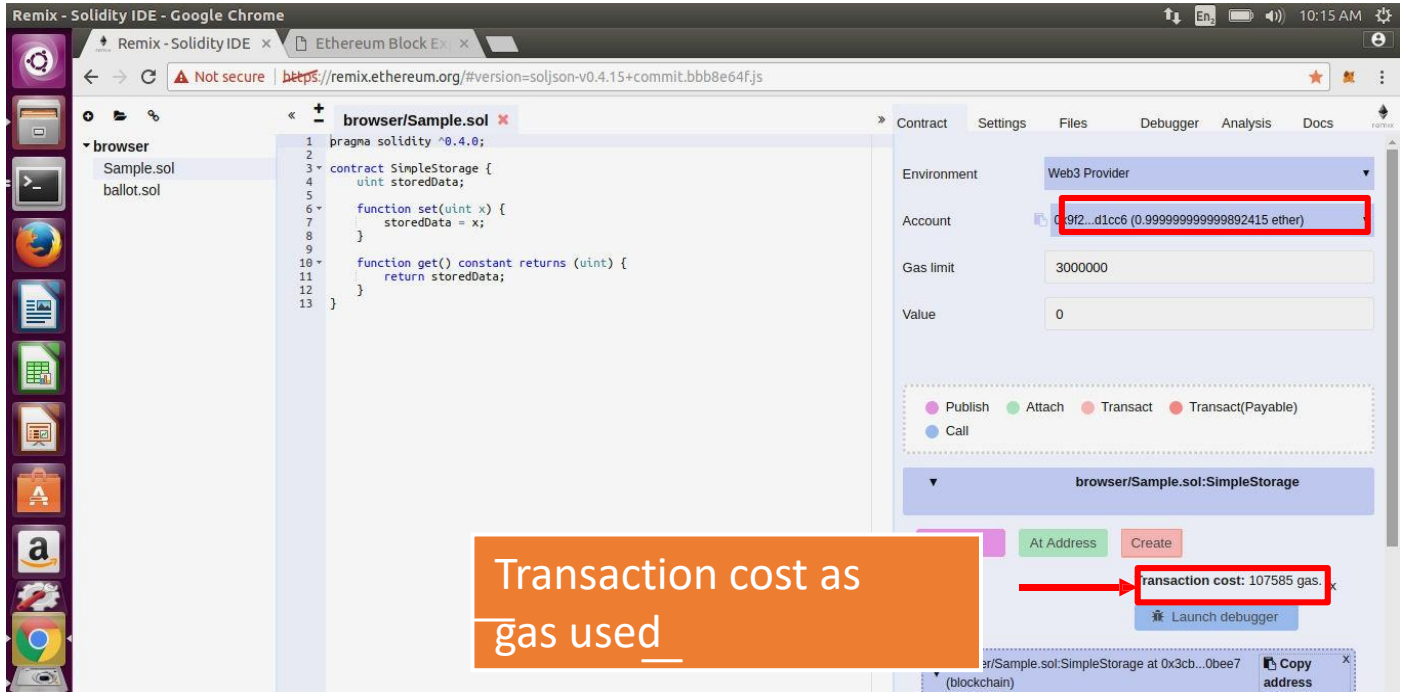
- Select the account that you want to use to push your contract
- After selecting the account click on create button



Contract gets created

The contract gets created and compiled

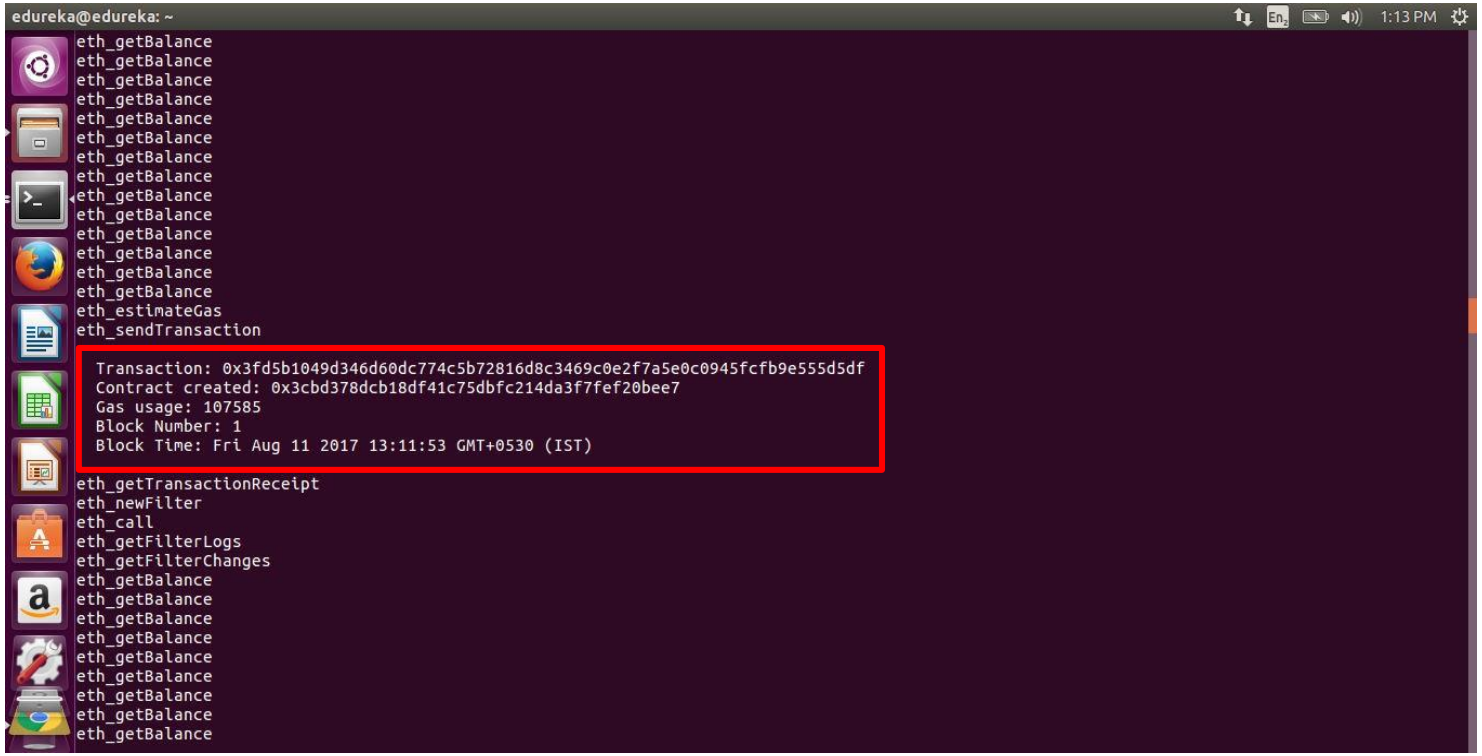
- The contract gets created and the balance of the account used to create the contract goes down by some value
- It is the value of ether which get exhausted as a gas for pushing contract



Check the block that includes your transaction

You can check the block details where your transaction is included

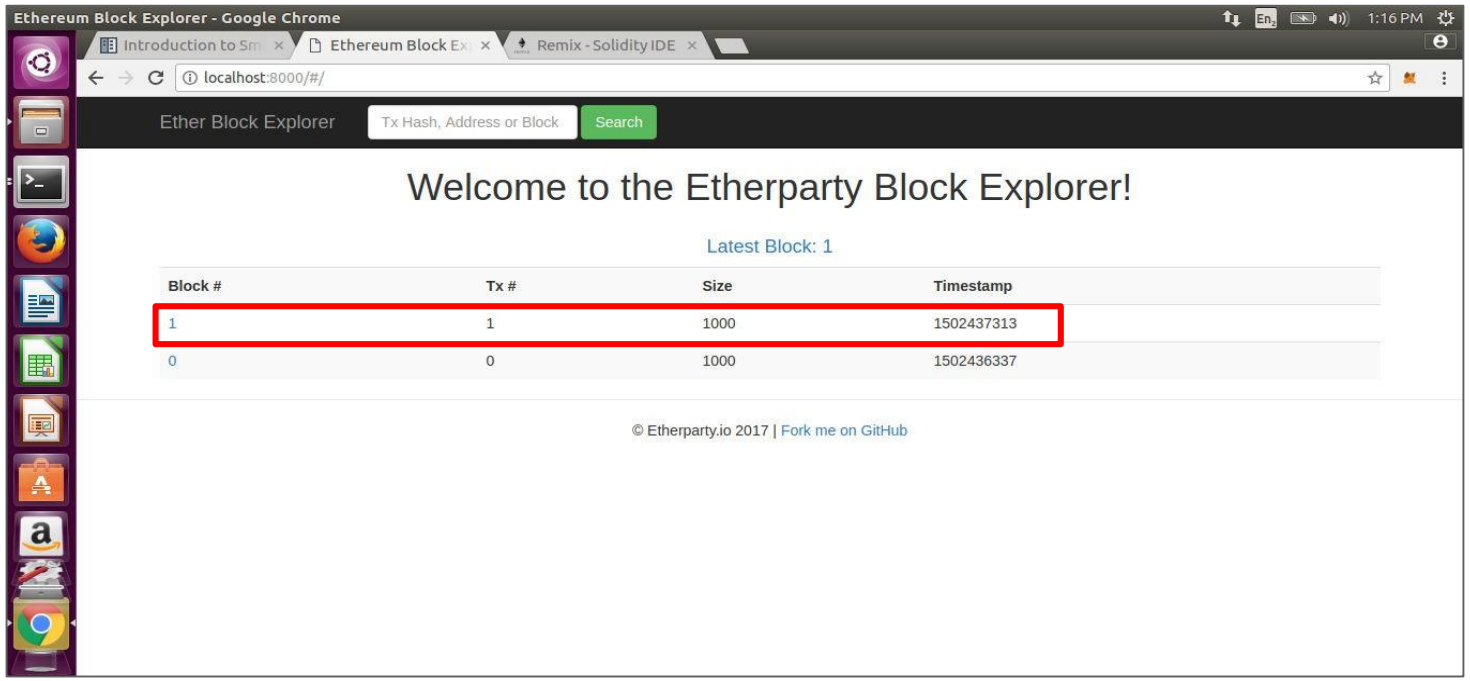
- The block details can be checked in the testrpc which is kept running in the other terminal
- And also listening to the transactions



Transaction reflecting in Block Explorer

The transaction gets included in block explorer, and can be seen after you refresh the page

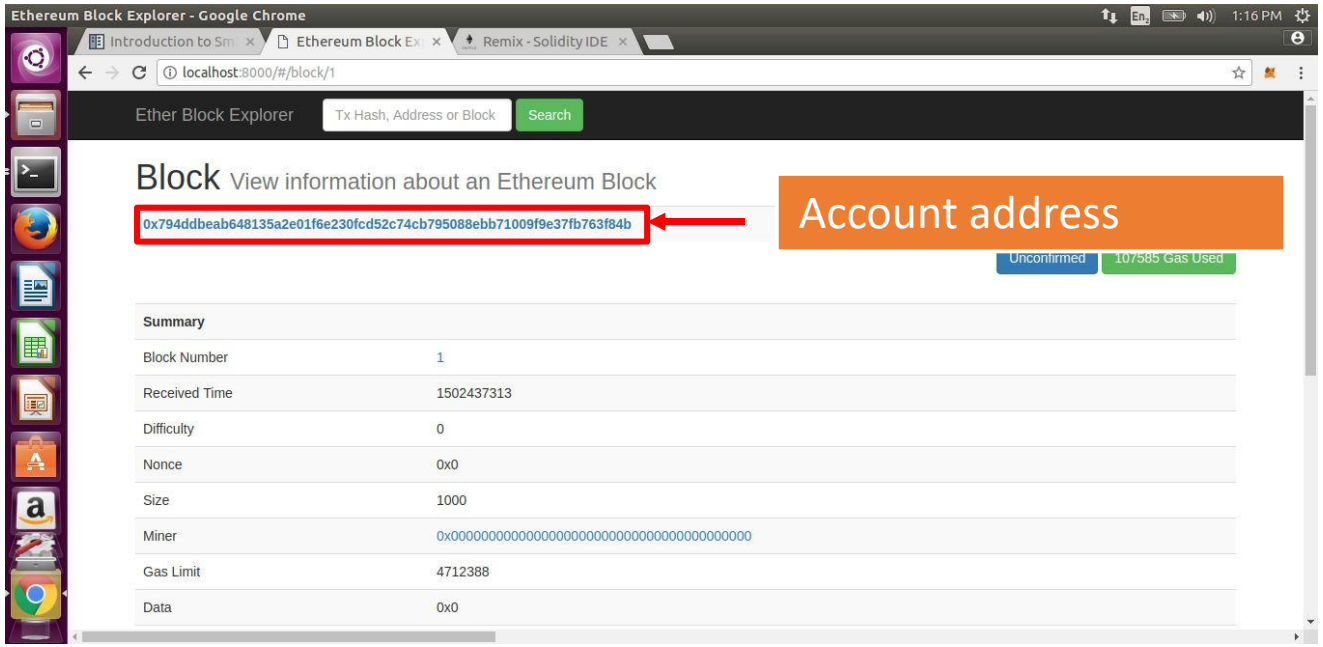
- The transaction gets included in block explorer
- You can get the details of the transaction by clicking on the block number



Block details in Block Explorer

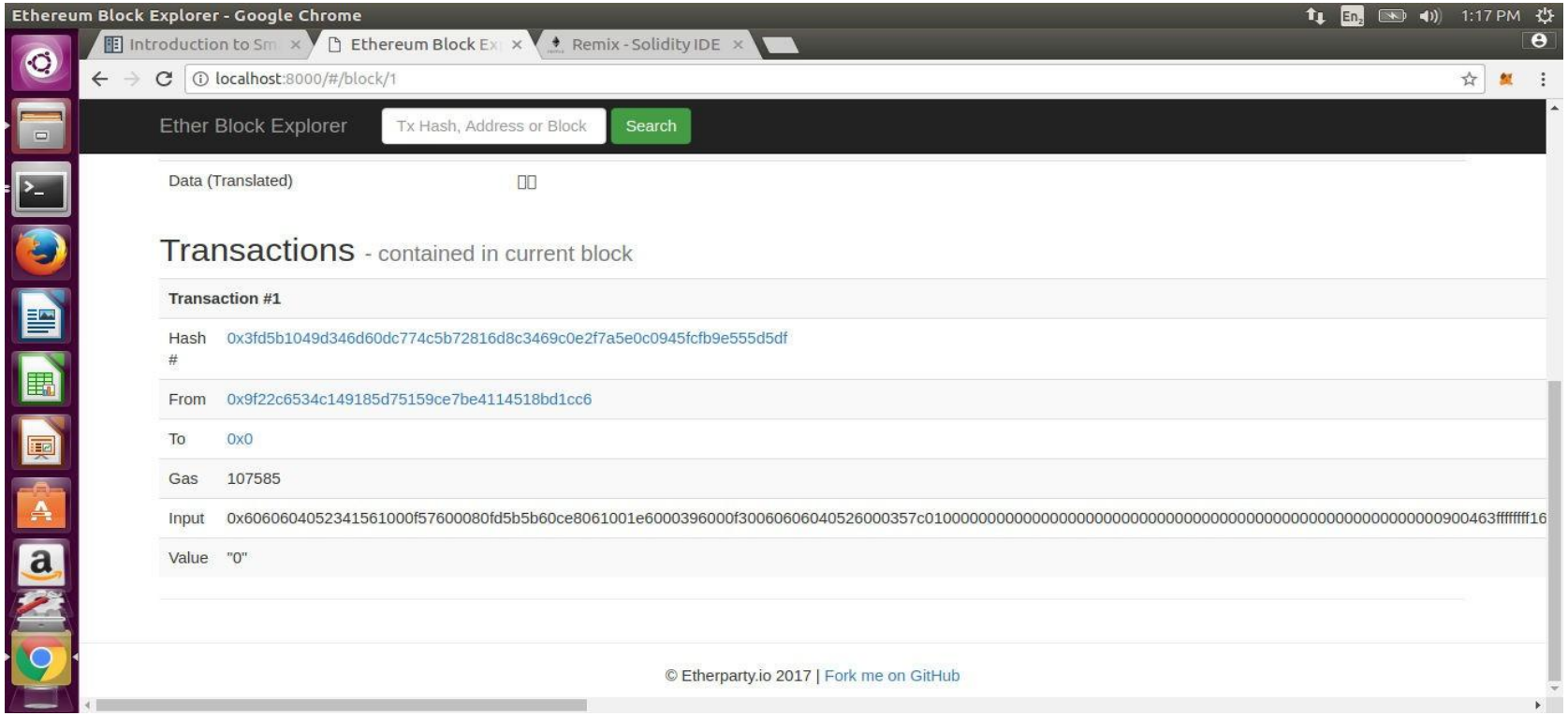
The block contains all the details of the transaction

- The transaction gets included in block explorer
- You can get the details of the transaction by clicking on the block number



Transaction details

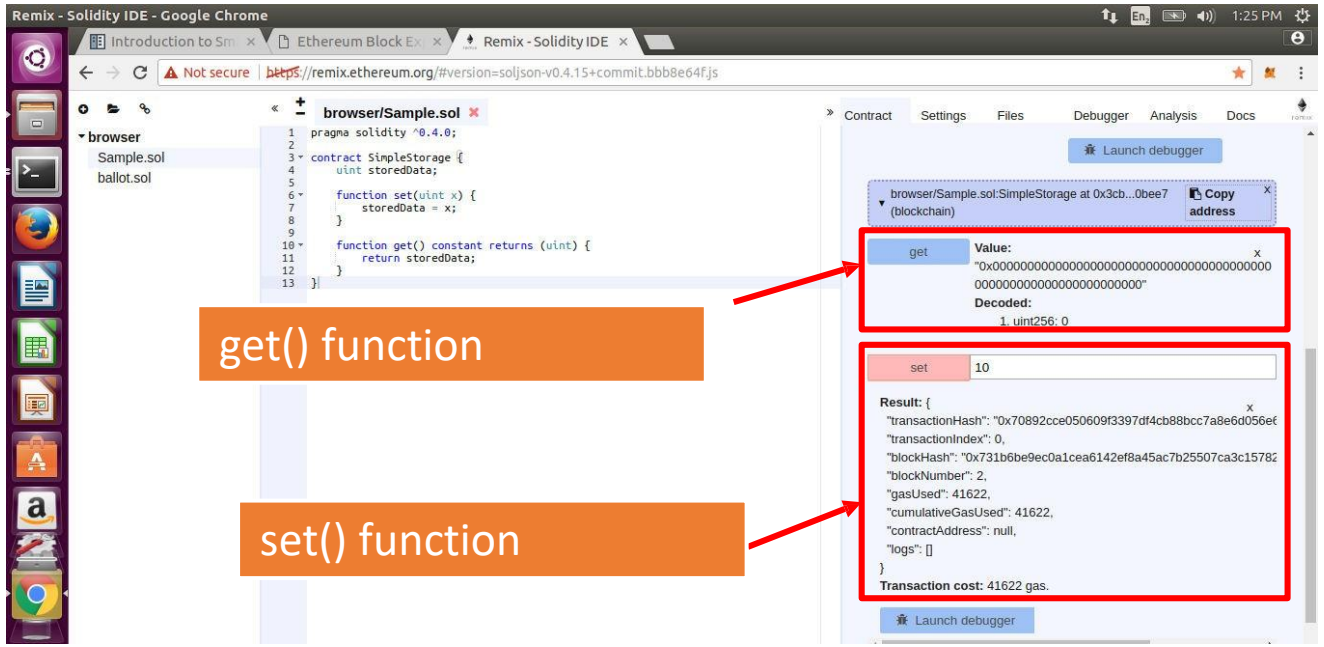
Clicking on the account address displays the transaction details



Check for the Function execution of the Contract

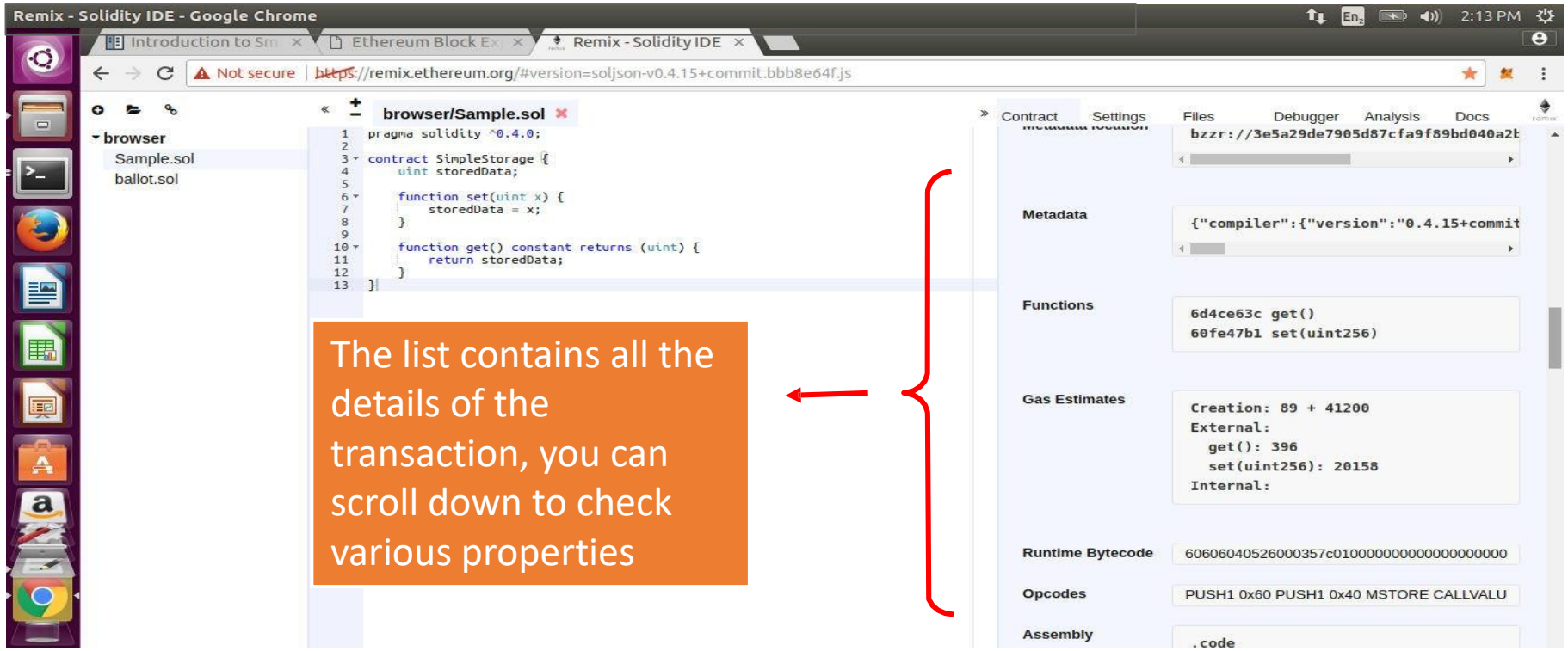
We can also check the function execution of the smart contract

- Here, our contract uses two functions: get() and set()
- We can alter the value of set() function and a newtransaction will get created
- In the indicated set() method the value is set to 10



Checking Contract details

Contract details can be checked by clicking the contract details



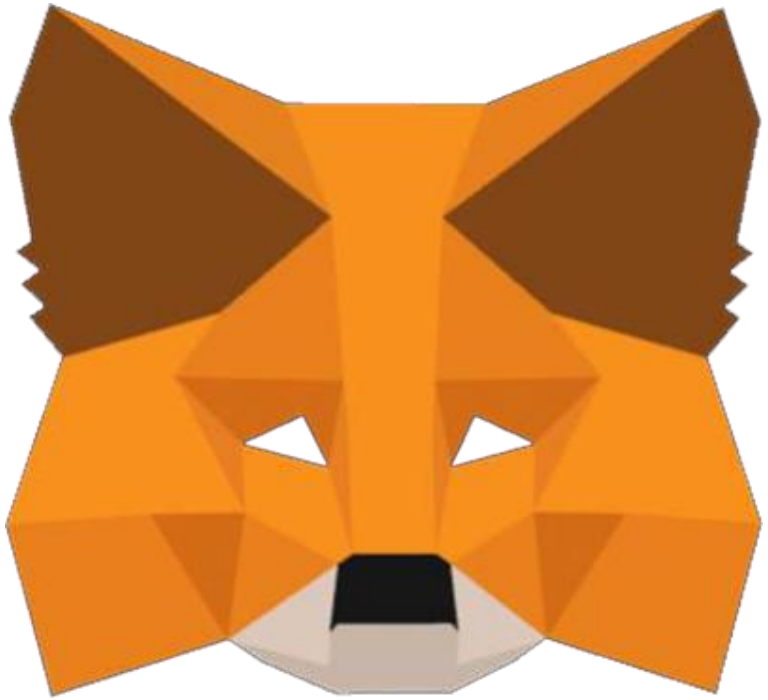
MetaMask

MetaMask

MetaMask brings Ethereum to the browser.
Let’s see how to install MetaMask as a chrome extension.

MetaMask

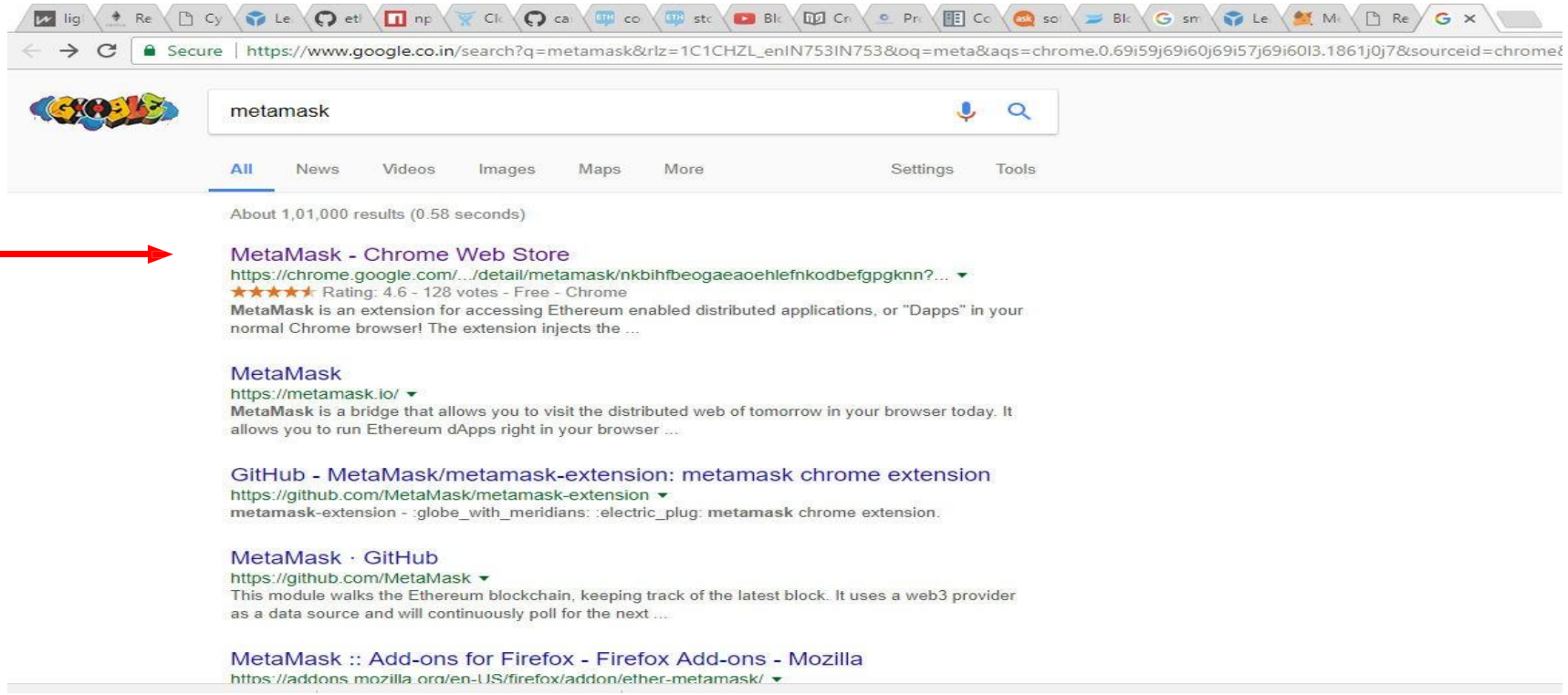
- MetaMask is a bridge that allows you to visit the distributed web of tomorrow in your browser today
- It allows you to run Ethereum dApps right in your browser without running a full Ethereum node
- MetaMask includes a secure identity vault, providing a user interface to manage your identities on different sites and sign blockchain transactions



Open your chrome browser

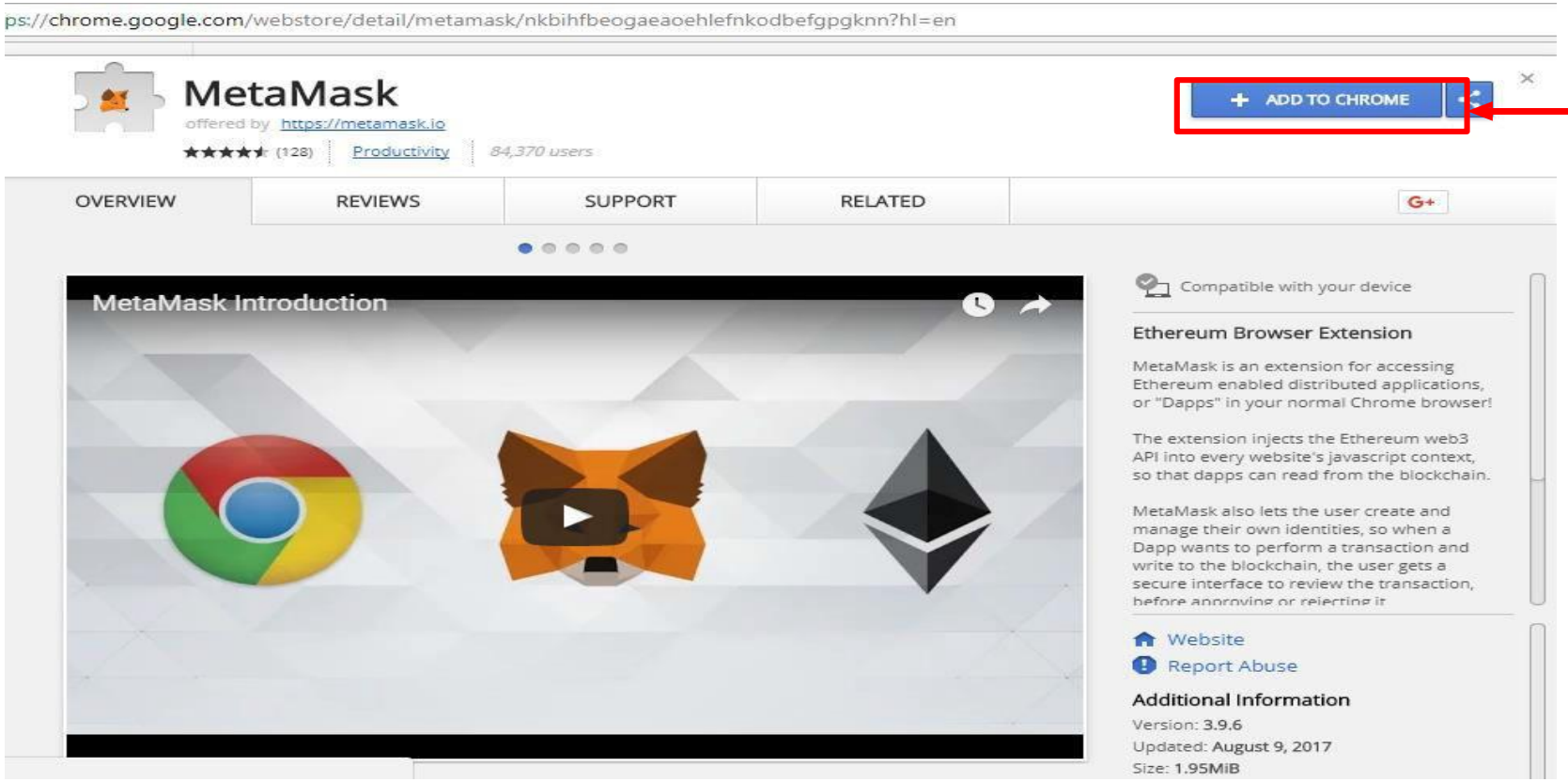
- To install MetaMask as a chrome browser, go to chrome and search “MetaMask” and click on the first link

Click on this link



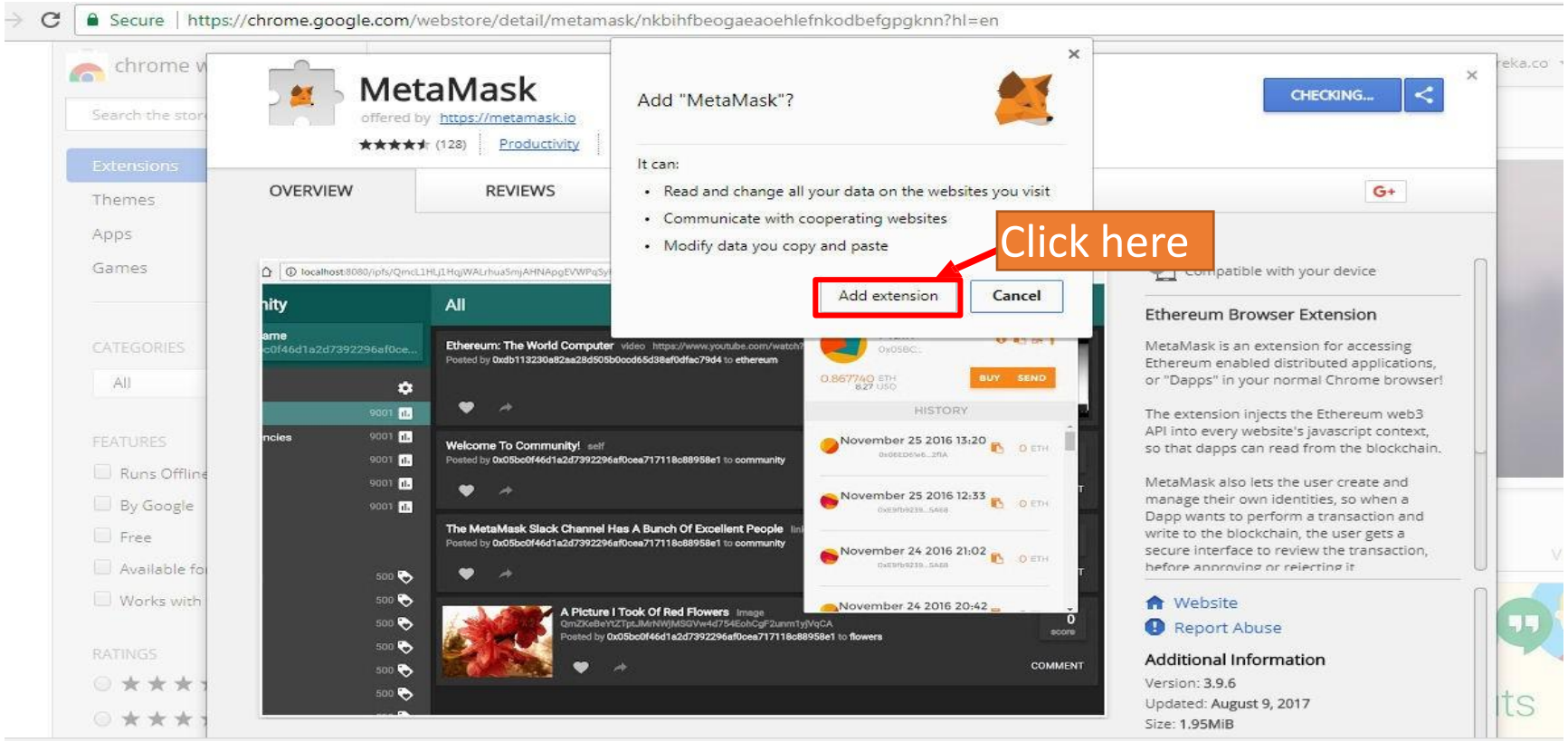
Add to Chrome

- The MetaMask site will open, click on Add to chrome



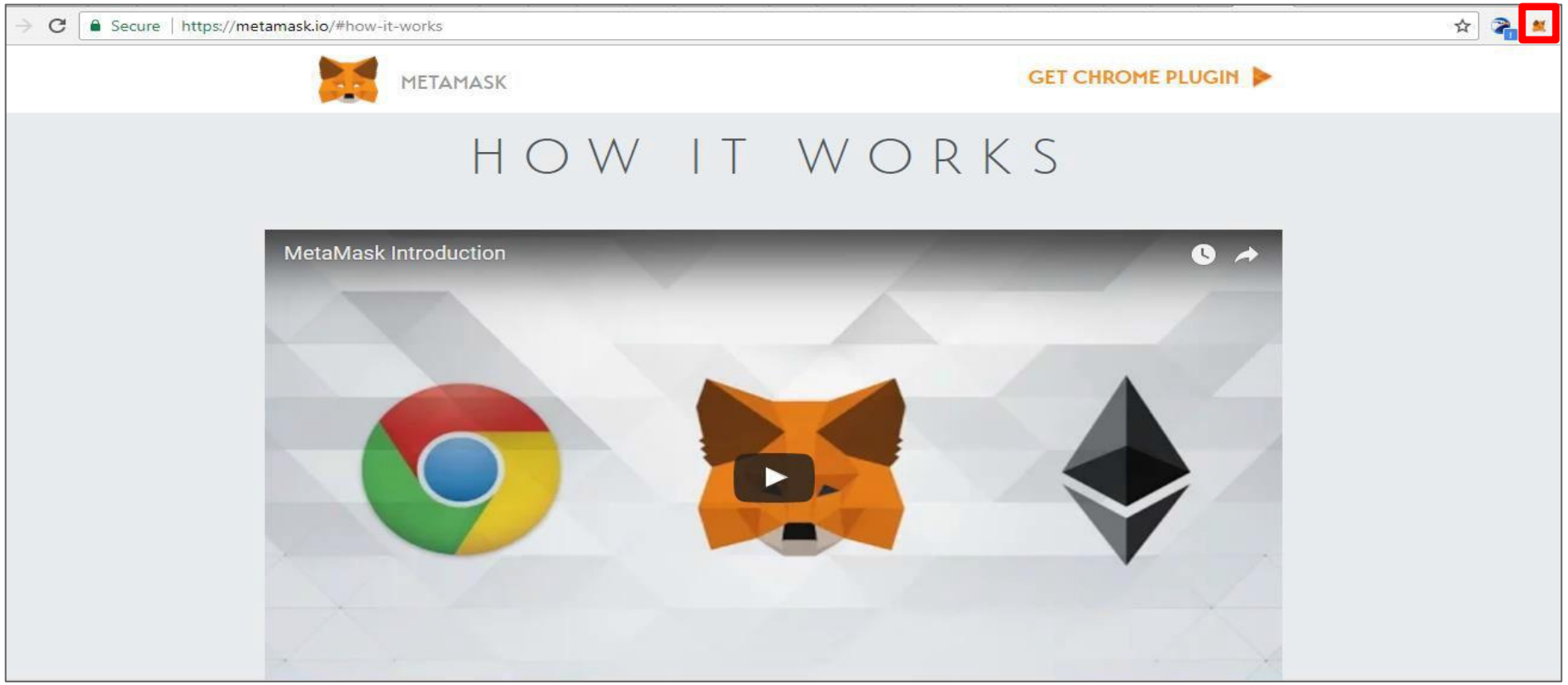
Add Extension

- It will ask you to “Add Extension”



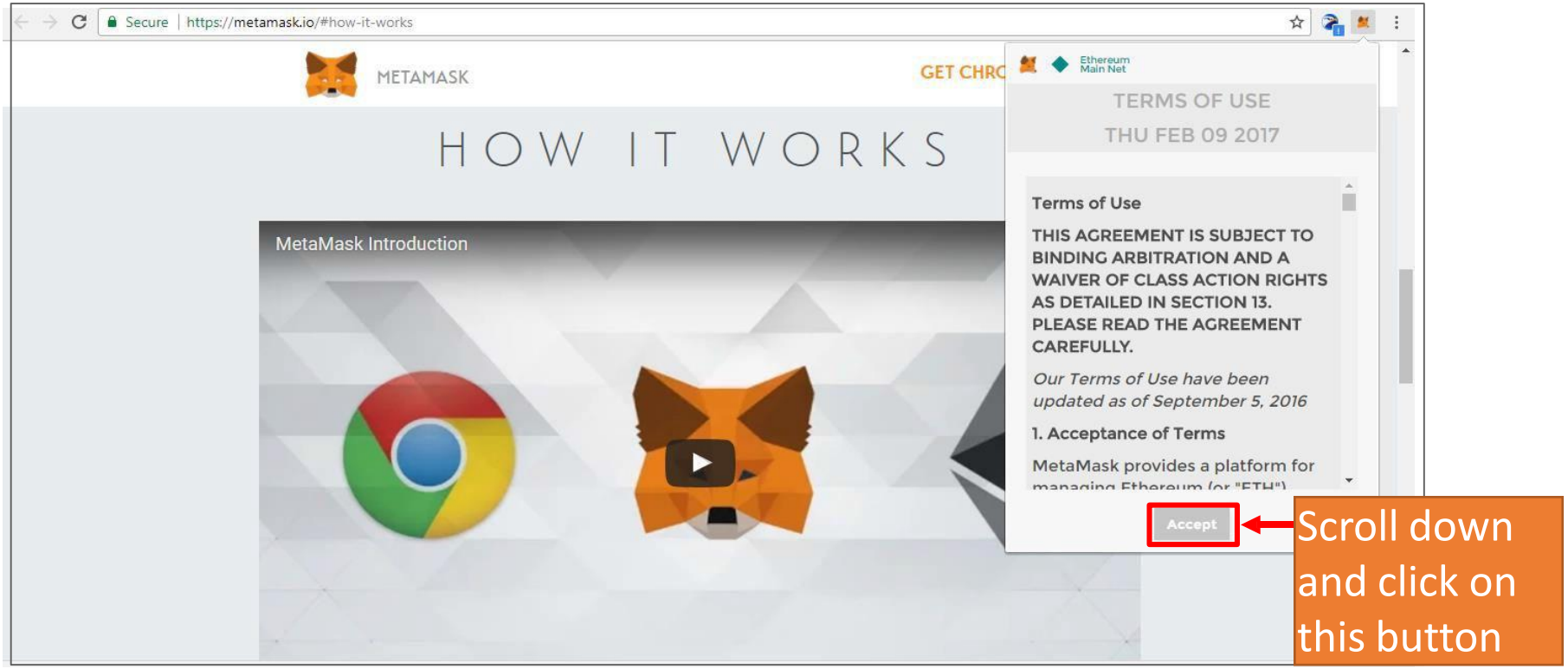
MetaMask extension gets added to the Chrome

- MetaMask gets added as an extension to the chrome. You can see the icon on top right corner



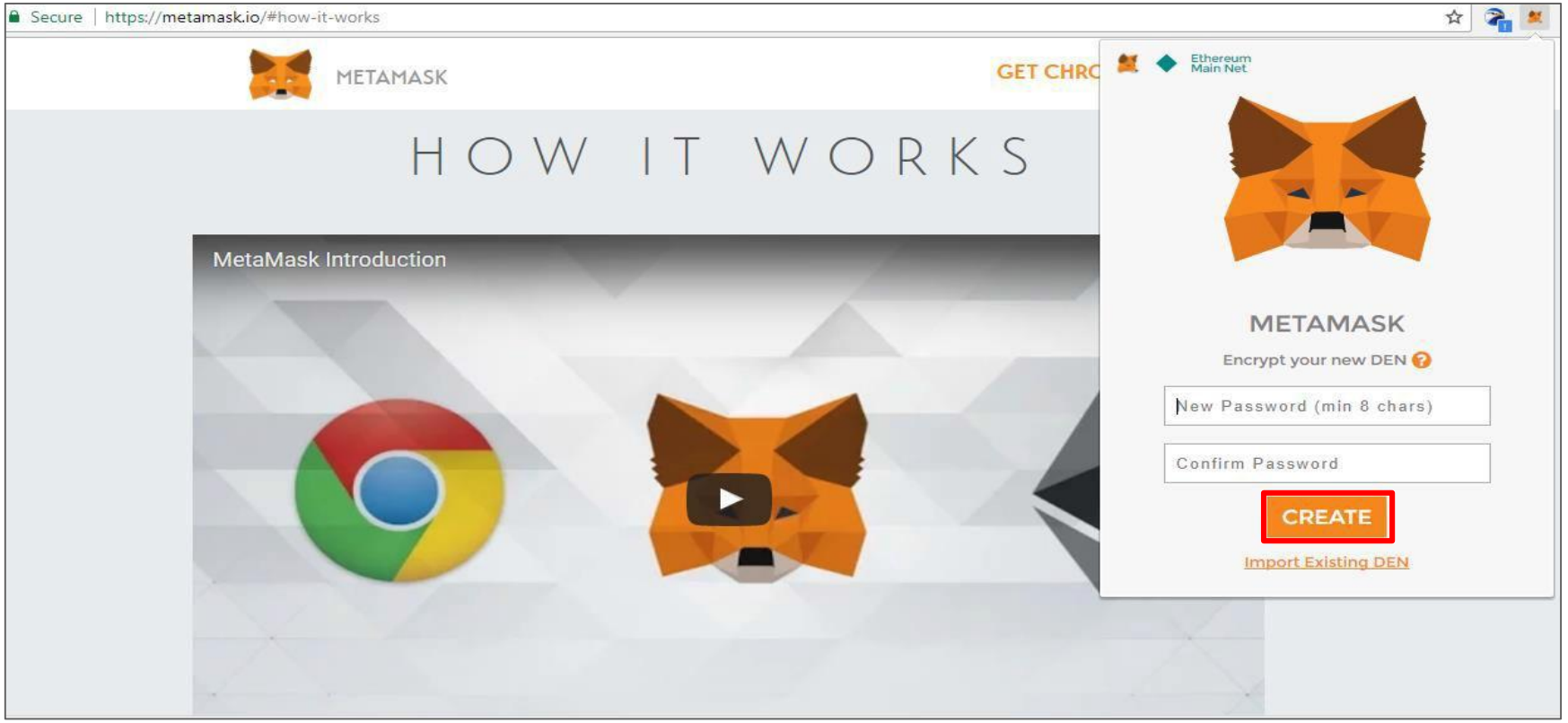
Accept “terms of use”

- Click on the MetaMask icon. It will ask you to Accept “Terms of Use”



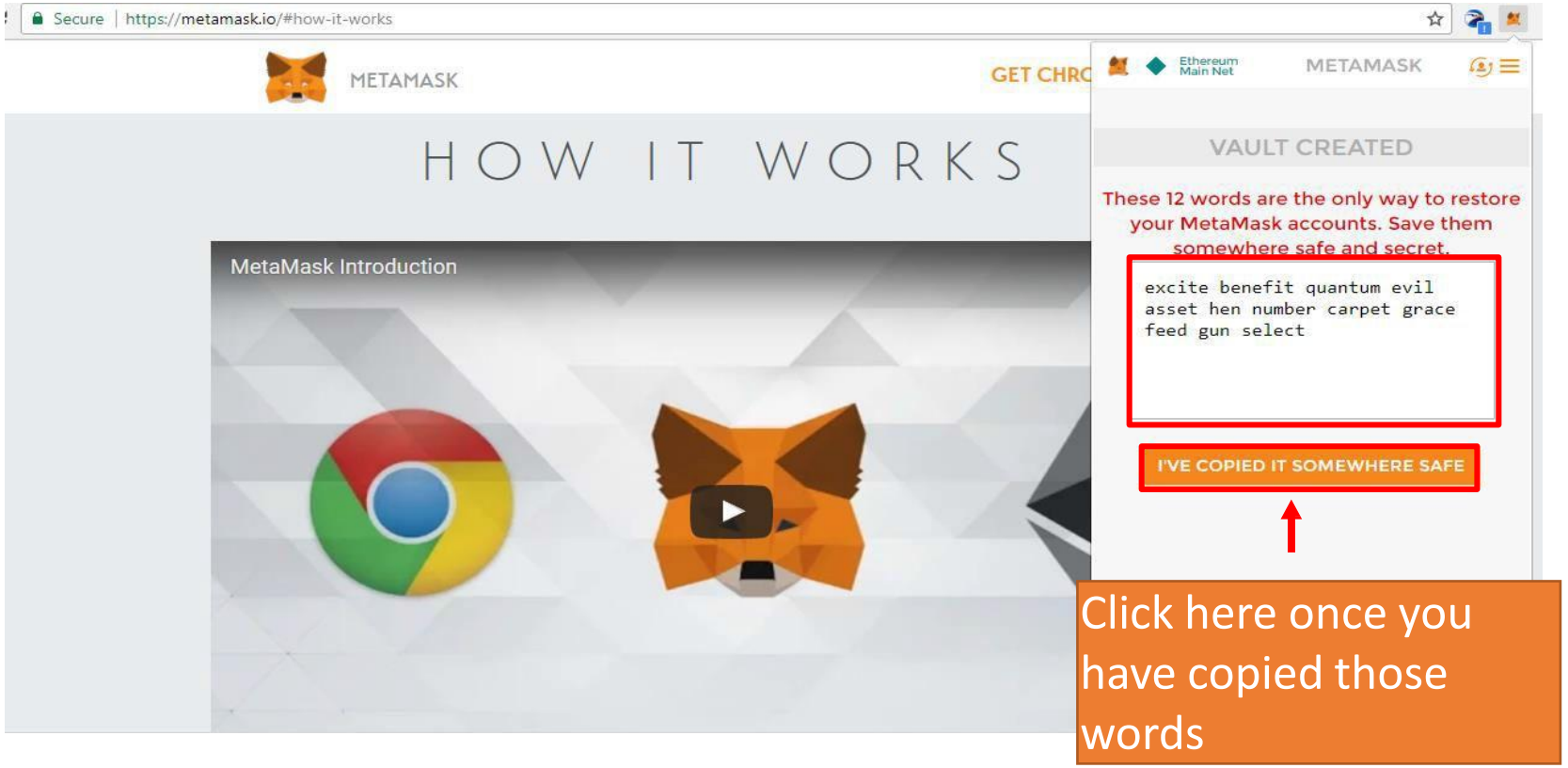
Create Password

- Now, create and confirm the password



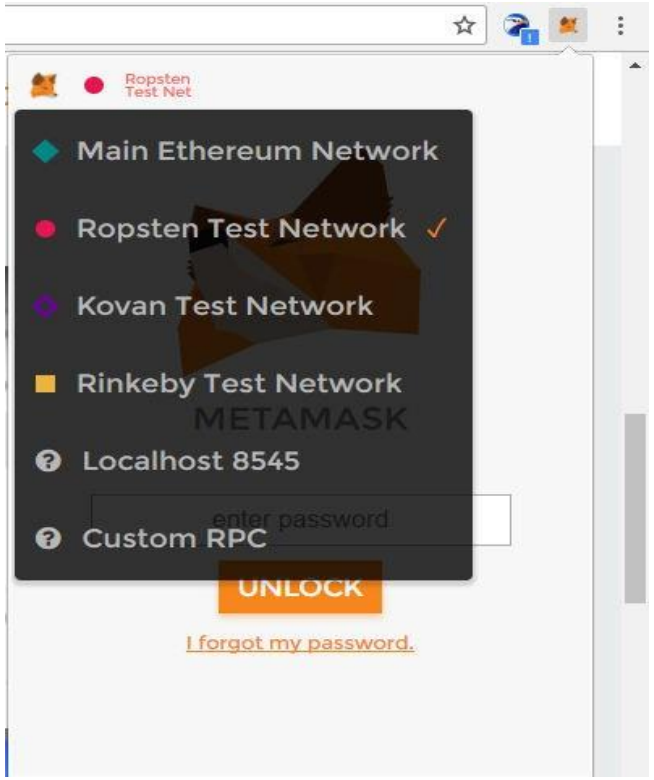
Vault created

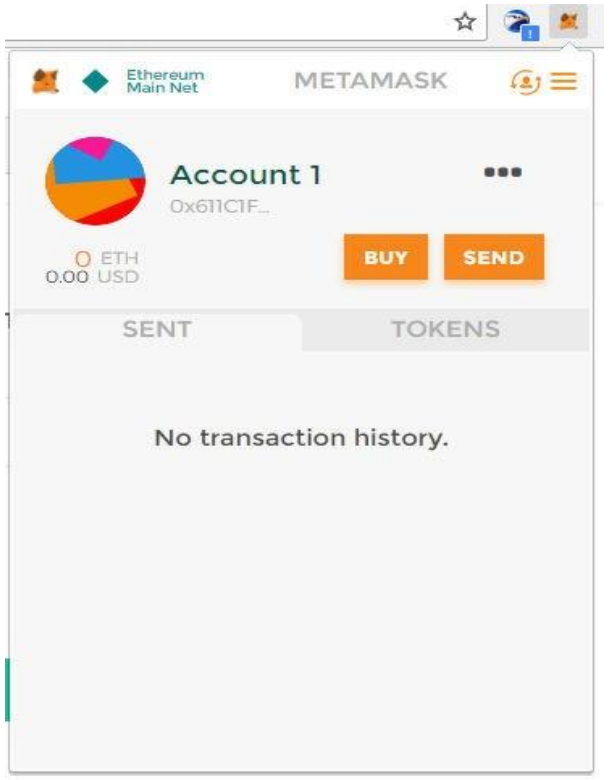
- The Vault is now created. Save the seed phrase (12 words) that are displayed in your vault. Keep it safe and confidential
- These seed words can be used to restore all the accounts you ever created



MetaMask is now ready to use

- With MetaMask you can create multiple accounts and can use them for transactions
- Also, MetaMask allows to switch between the networks





Installing Blockchain

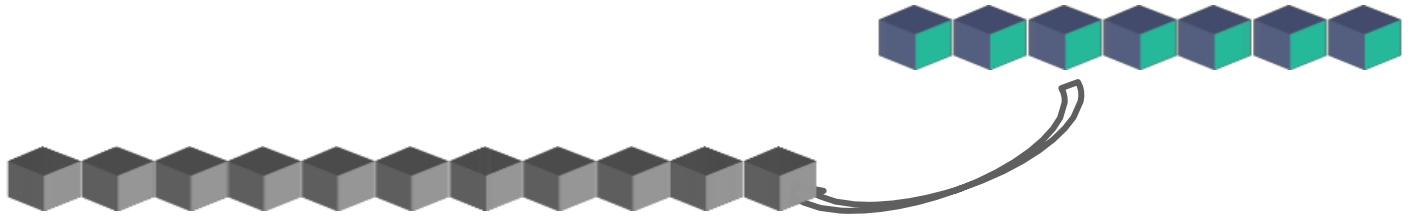
Installing Blockchain

Now, we are going to develop our own private blockchain. Let’s Start

Developing Private Blockchain

1. Install Go Ethereum
2. Register an account on Ethereum Blockchain

3. Create an autonomous Private Blockchain
4. Perform transactions internally



Installing Blockchain

Cloning Geth Code

Creating a Genesis Block

Validating & Mining Ether

Making Rules for our Blockchain

Download Ethereum code and compile it

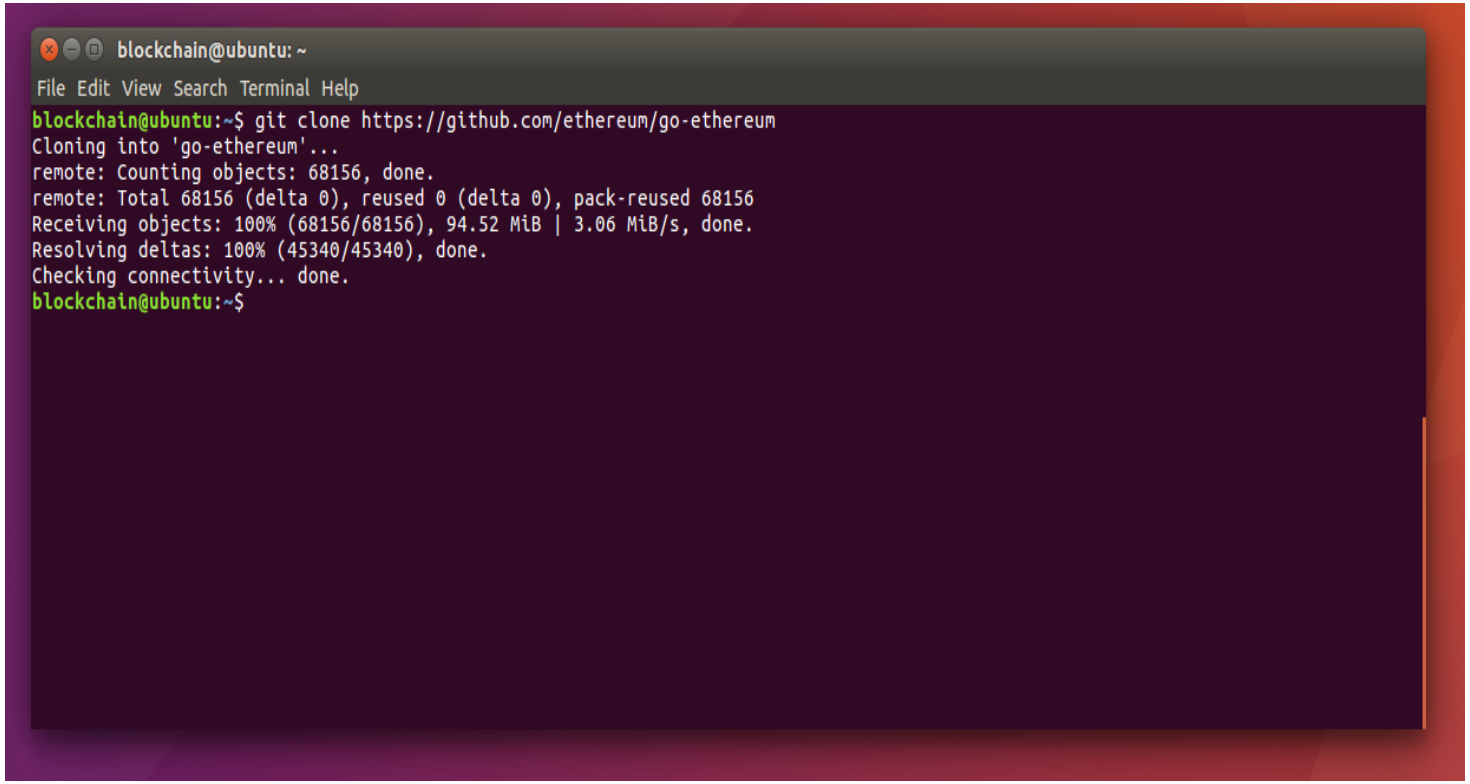
```
sudo apt install git // installing git in case not
installed already
sudo apt -get update
sudo apt -get upgrade

git clone https://github.com/ethereum/go-ethereum cd
go-ethereum
git tag
git checkout tags/v1.6.7 -b EdurekaEthereumV1.6.7

//Checking the branch git
branch

//Install golang
make all
```

Cloning GIT repository

First step is to clone a Go- ethereum repository

To clone a Go-ethereum repository execute the following command:

```
git clone https://github.com/ethereum/go-ethereum
```



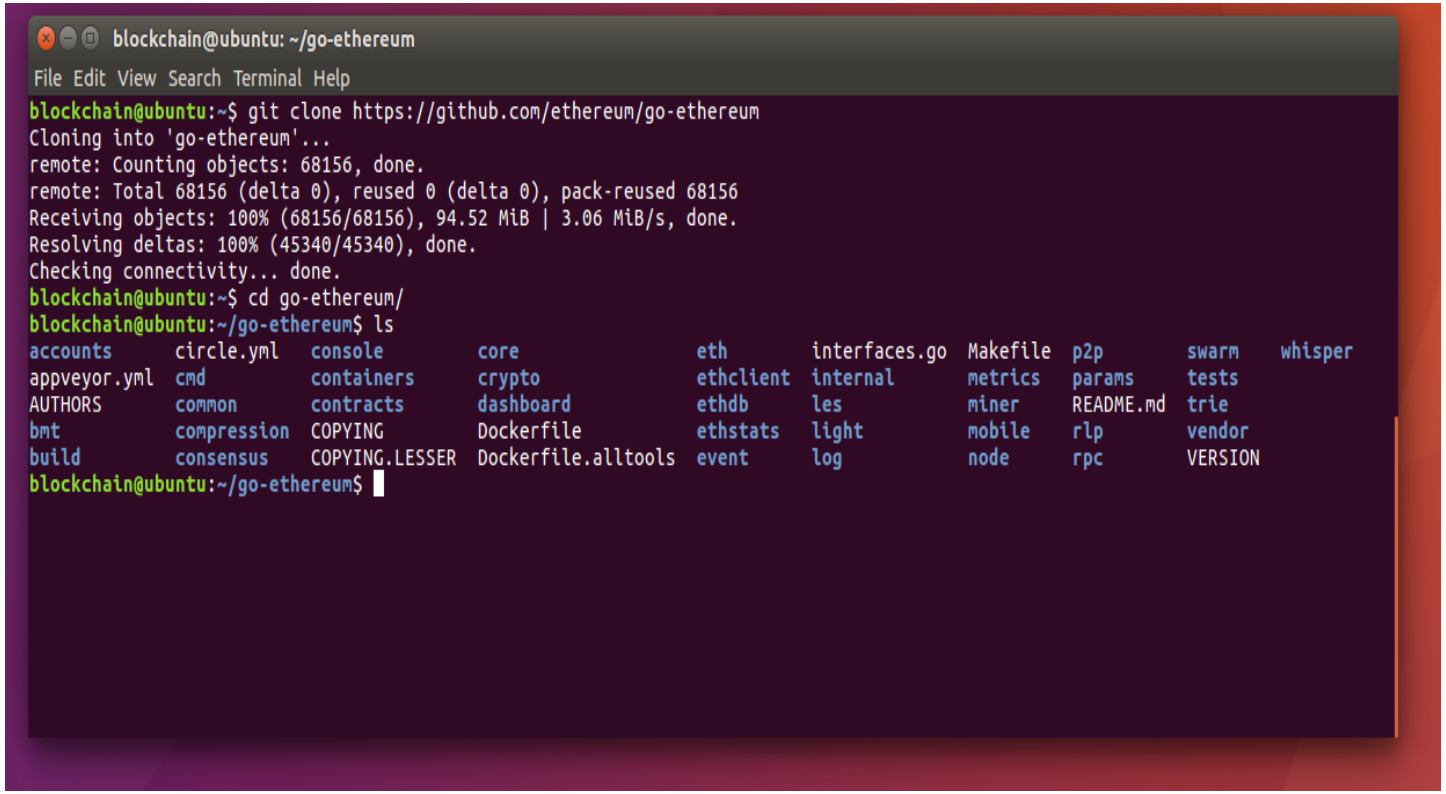
Note: Cloning of Geth will take some time so please be patient

Check the files in Go-Ethereum

Once the go-ethereum is cloned, check for the files present in go-ethereum directory

To check for the files in the go- ethereum, execute the following command:

```
cd go-ethereum ls
```



Go lang Installation

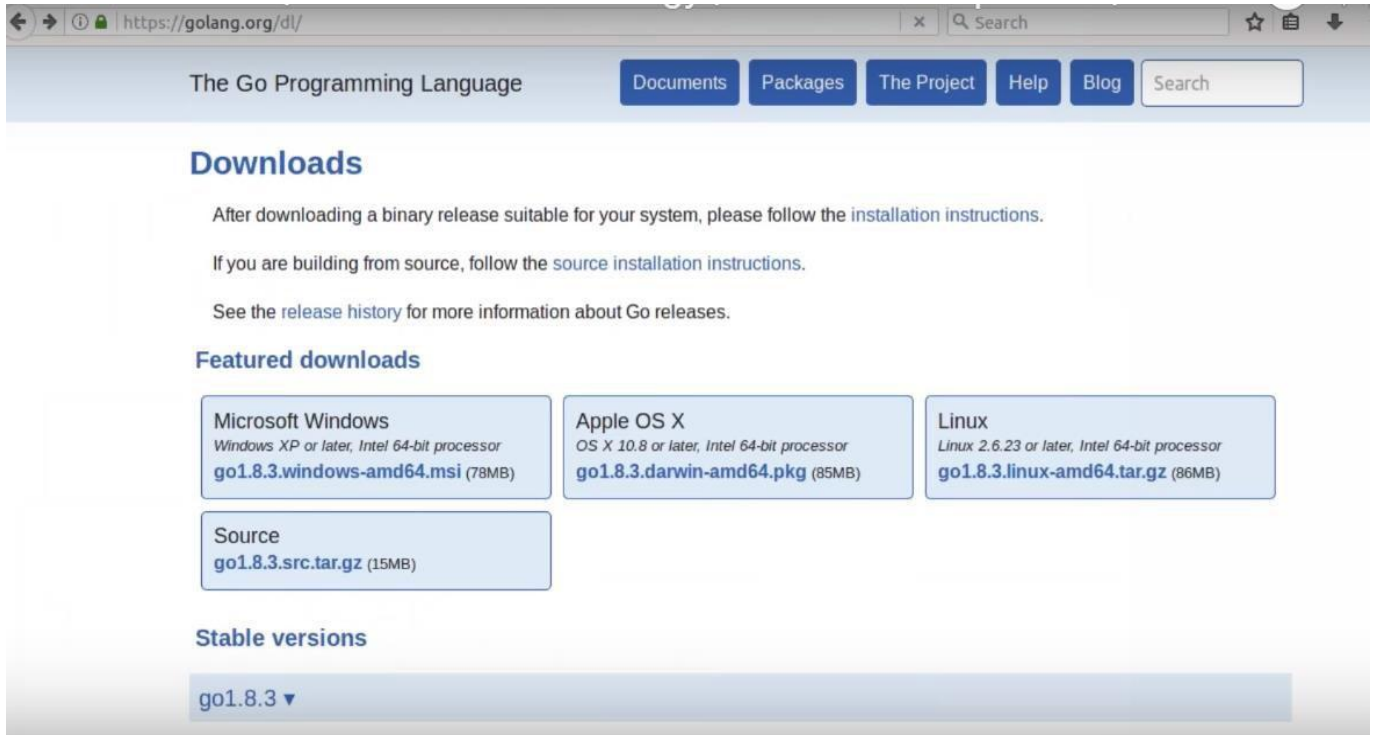
Go Lang Installation

> Now before we begin the blockchain execution, we need to install the Go language as part of the system, as Go Ethereum (geth) is the official golang implementation of the Ethereum protocol

Go Lang Installation

To install Go lang follow the steps:

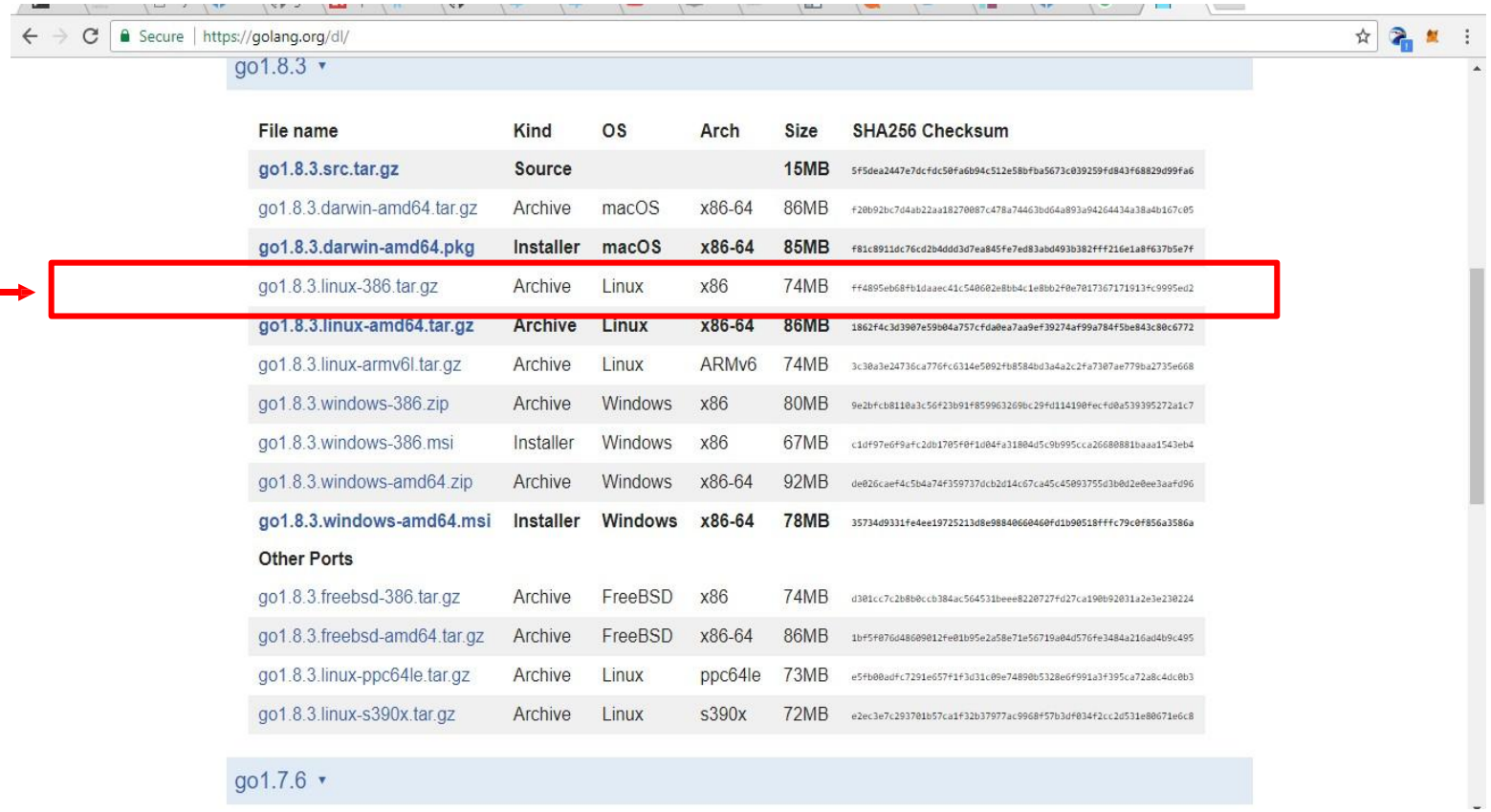
Step 1: Visit the site https://golang.org/dl/
Step 2: Check for the package which meets your system requirements and download it
Step 3: Since I am using 32 bit ubuntu, I have downloaded 32 bit file



Various Packages in Go-lang

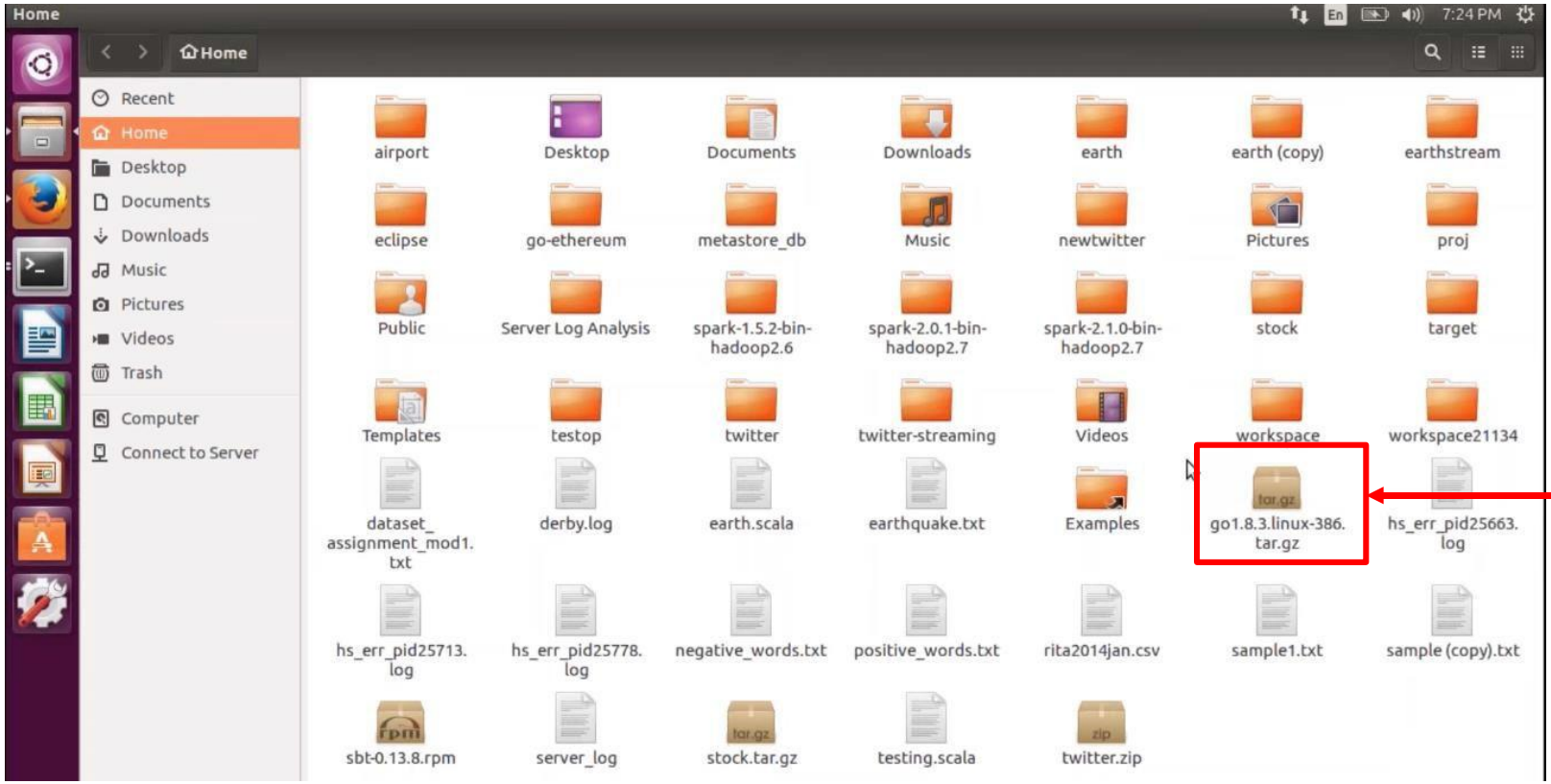
There are various versions of Go-lang available on the site, check for the system requirements and download accordingly

I am using 32 bit version of Go-lang



Check where the file is downloaded

By default the file will be in downloads folder and the name of the file is go1.8.3linux-i386.tar.gz



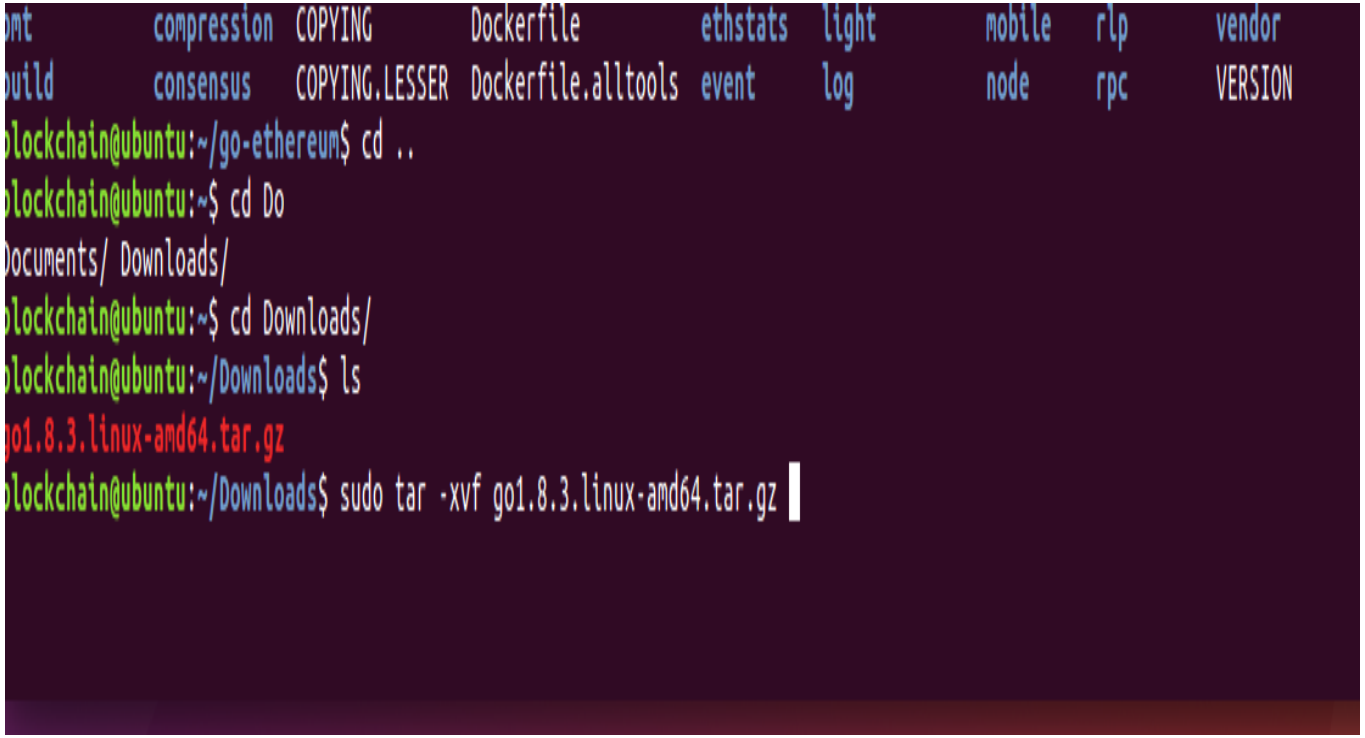
Extract and Install the file

Now, we have to extract the file and install it as part of the system
To extract and installing the file we need to execute the following command:

```
sudo tar –xvf go1.8.3.linux –386.tar.gz
```

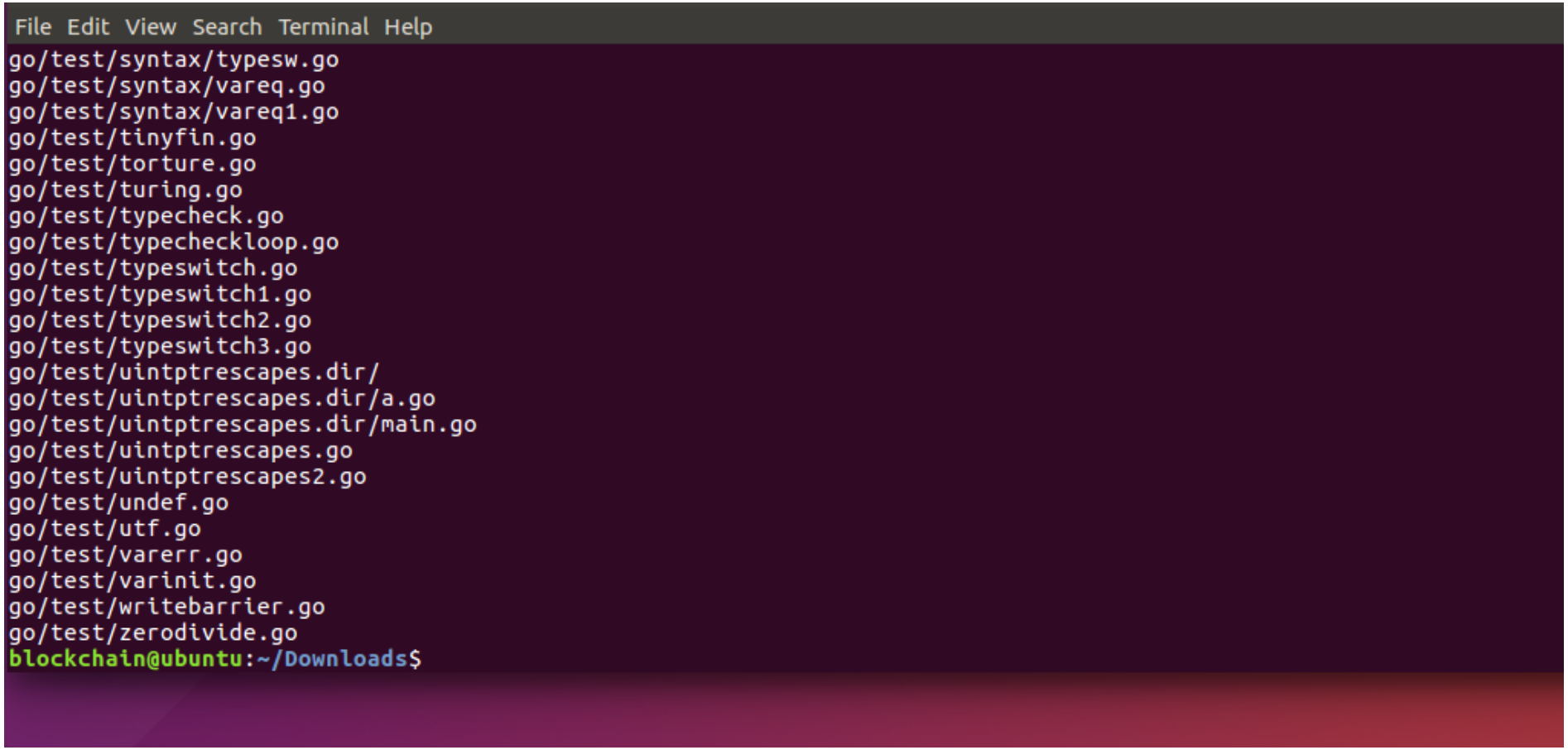
Few points to remember before executing this command:

- Ensure that the file name you are giving is correct
- Also, make sure you are executing this command as a super user (sudo) because this require administrative privileges



Go lang gets extracted and installed

After giving the password the execution starts and Go starts getting extracted and installed

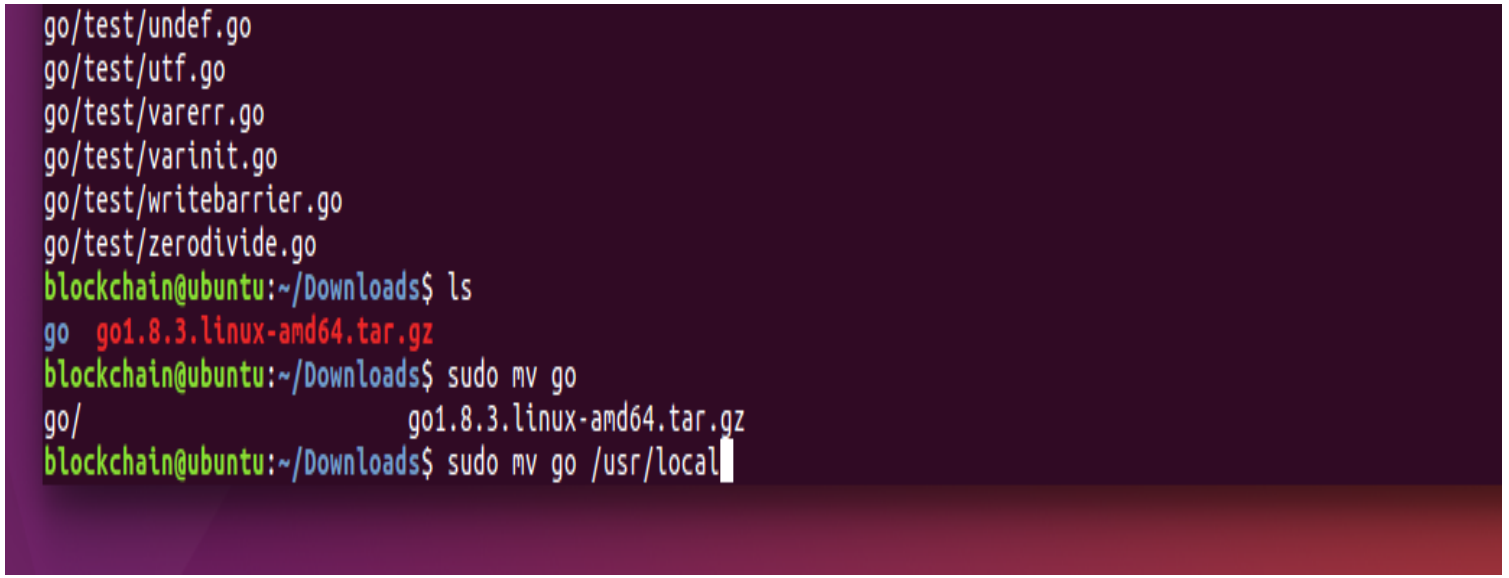


Moving the file to a specific location

The installed file is moved to /usr/local. This is mainly to ensure that I have ease of access to the Go
To move it to the specified folder we need to execute the following command:

```
sudo mv go/usr/local
```

On executing the above command, the file would have moved



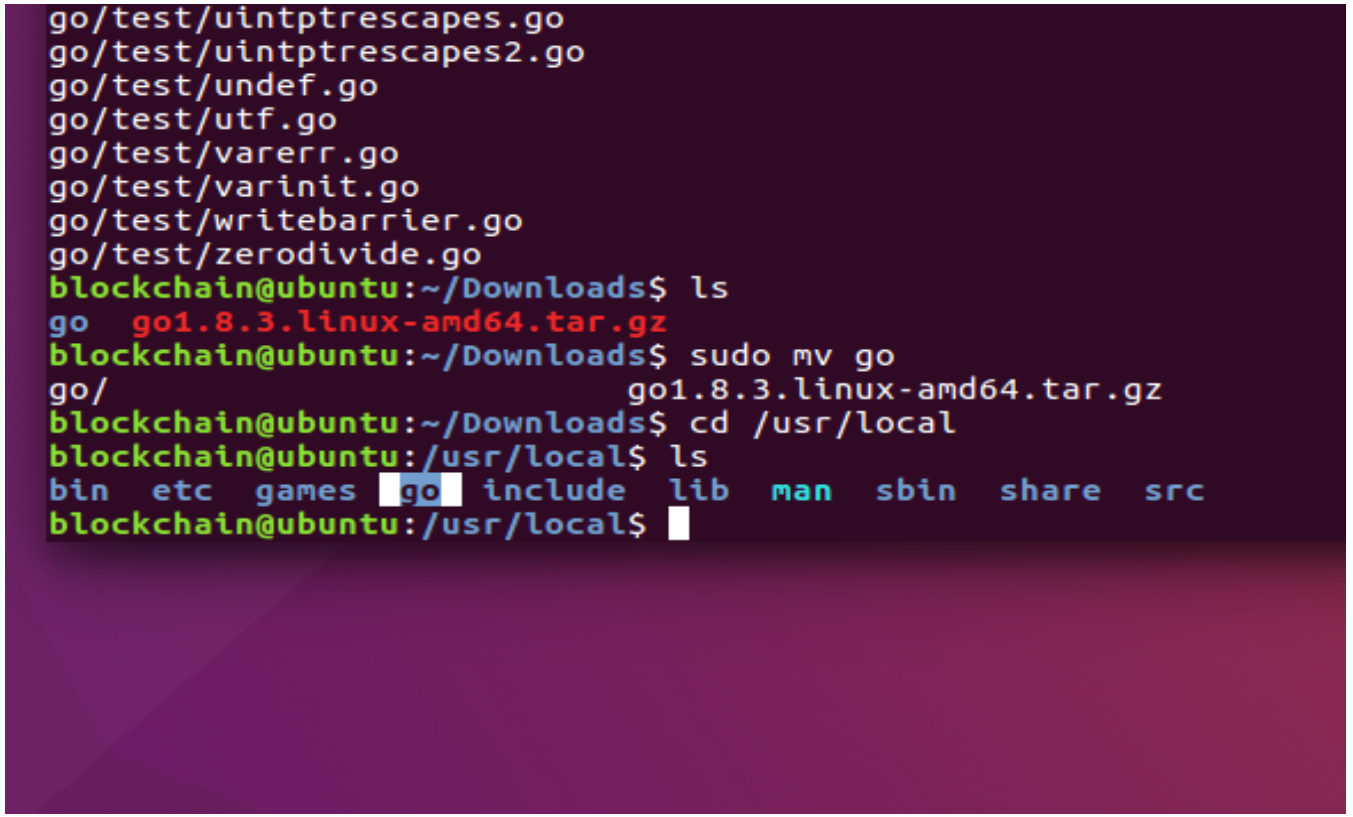
To check whether the file has moved

We should always check whether the file has moved to the specified folder

To check whether the file is moved, you can go to the directory and execute the command:

```
ls
```

On executing the above command, the file would have moved



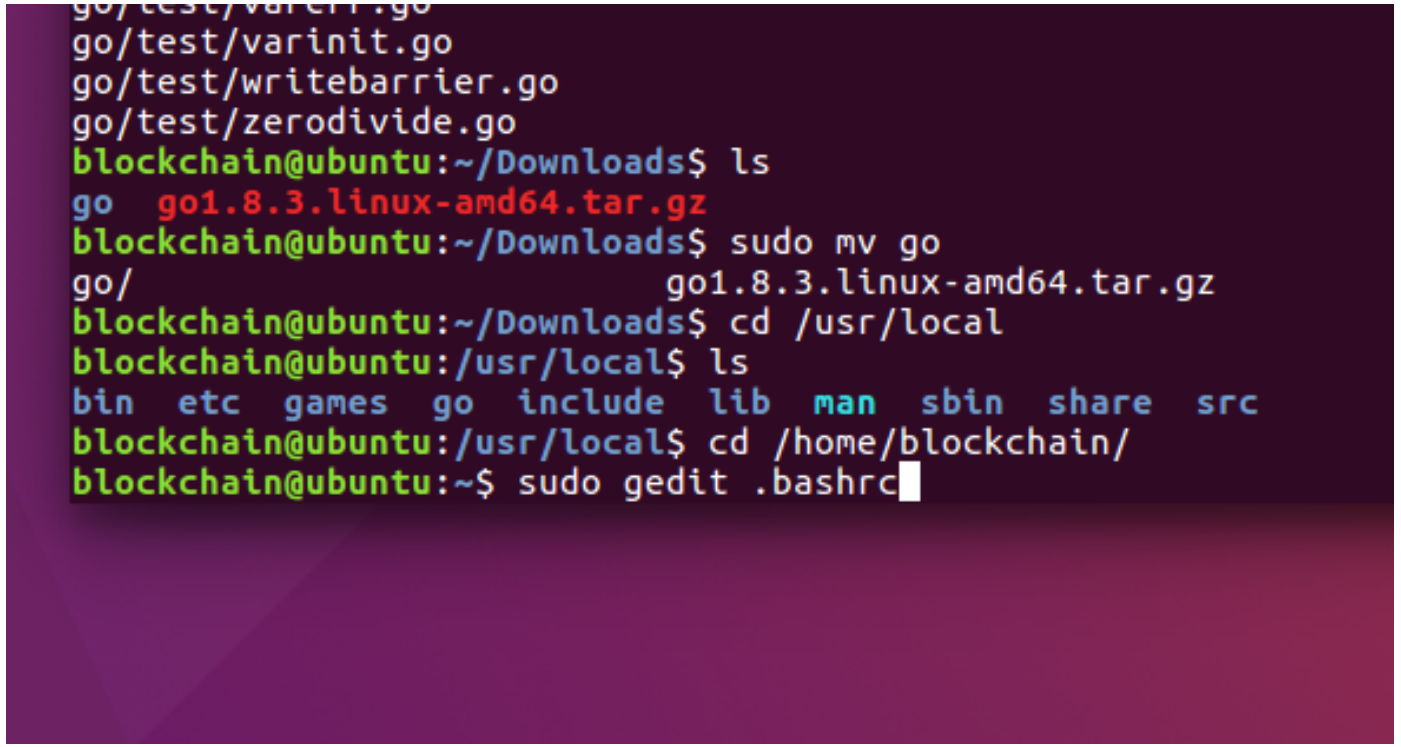
Note: After checking that the file is present, close the terminal

Adding Environment Variables to Variables

Now we need to add the environment variables to the list of variables
For doing so, we need to execute the following command:

```
sudo gedit .bashrc
```

The above command opens the .bashrc file



Adding Environment Variables to Variables

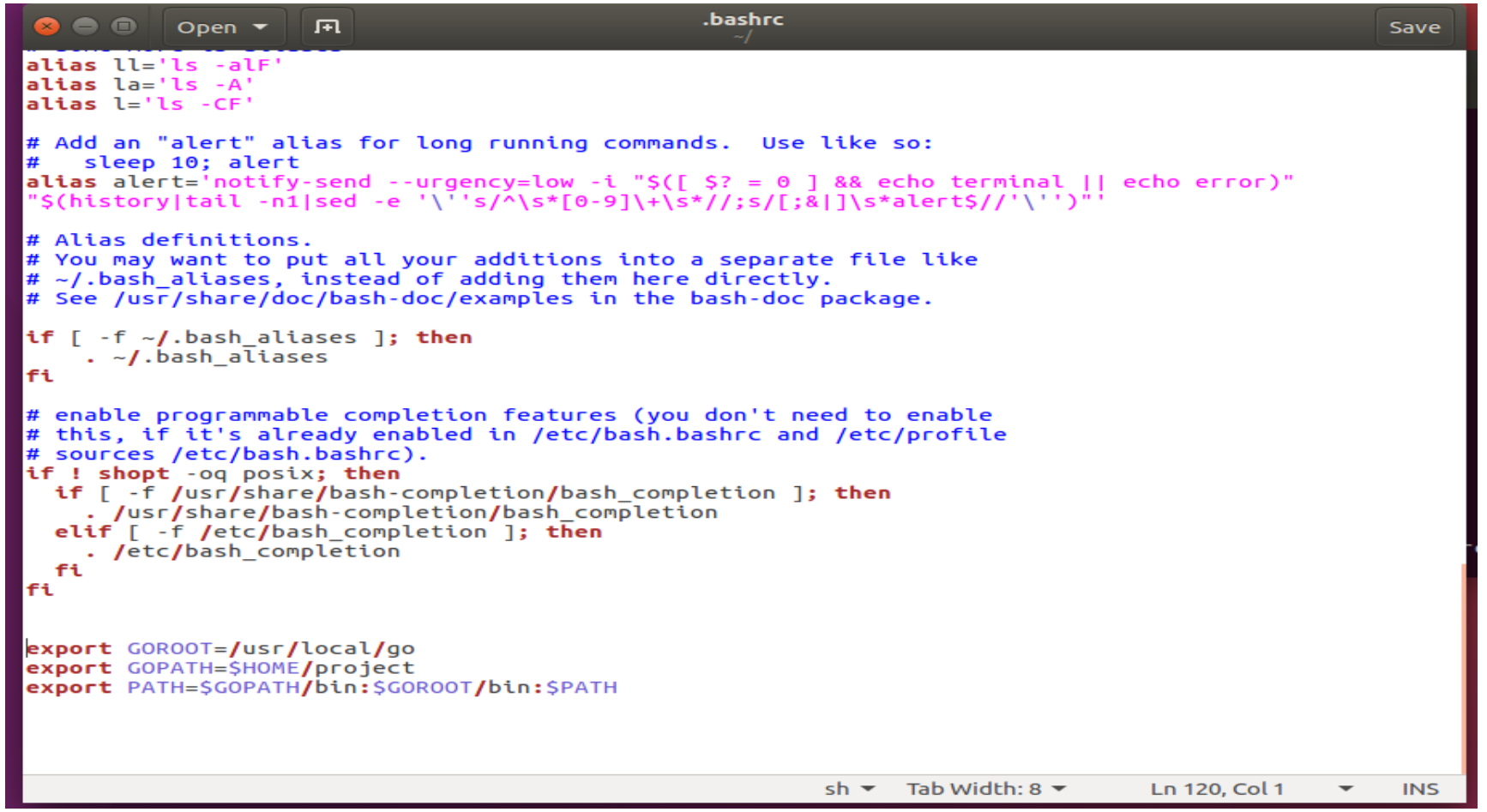
Here three lines you need to add

export GOROOT=/usr/local/go (this is the location where the Go is moved to. Once the Go root is set we need to specify the working folder)

export GOPATH=$HOME/Projects/Proj1 (This can be set to any location)

export PATH=$GOPATH/bin:$GOROOT/bin:$PATH (This sets the go path and go root in our environment variable list)

Files added in .bashrc



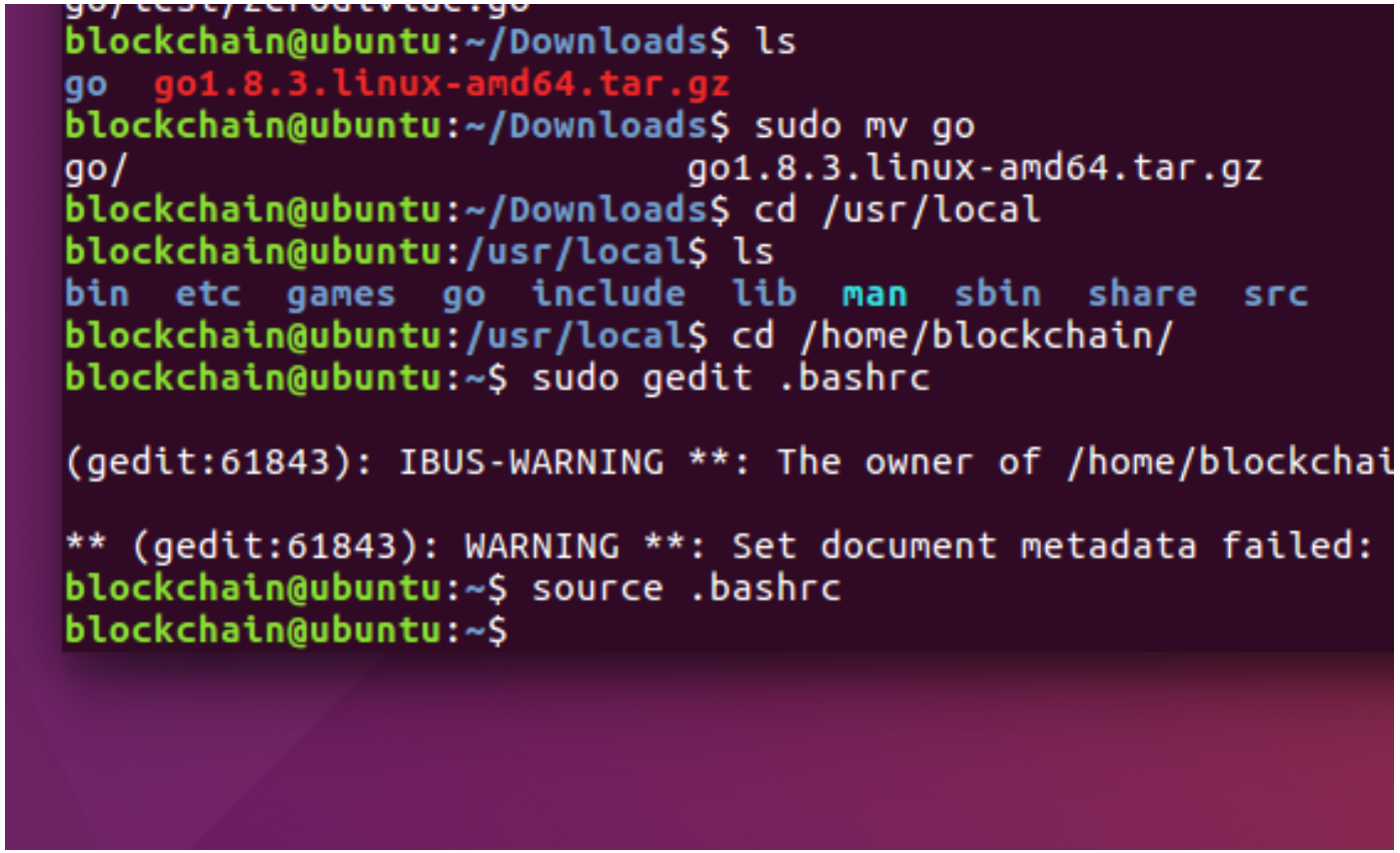
Once you have added the files, save .bashrc and close it

Restart the .bashrc file

To restart the .bashrc file use the following command:

```
source .bashrc
```

The above command will restart the .bashrc file



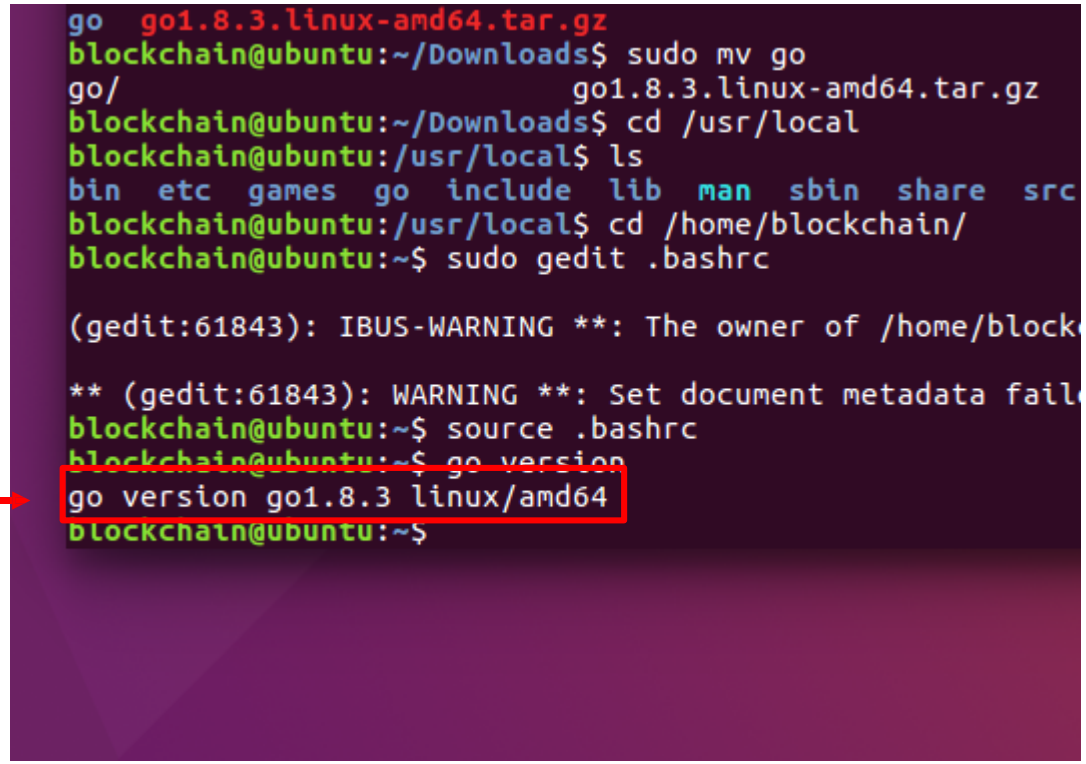
Check whether the Go lang has installed correctly

To ensure that the Go lang has installed correctly, execute the following command:

```
go version
```

The above command will return installed version of the Go lang if you have followed the installation steps correctly

The go version is installed correctly. Now close the terminal window



“Make all” command

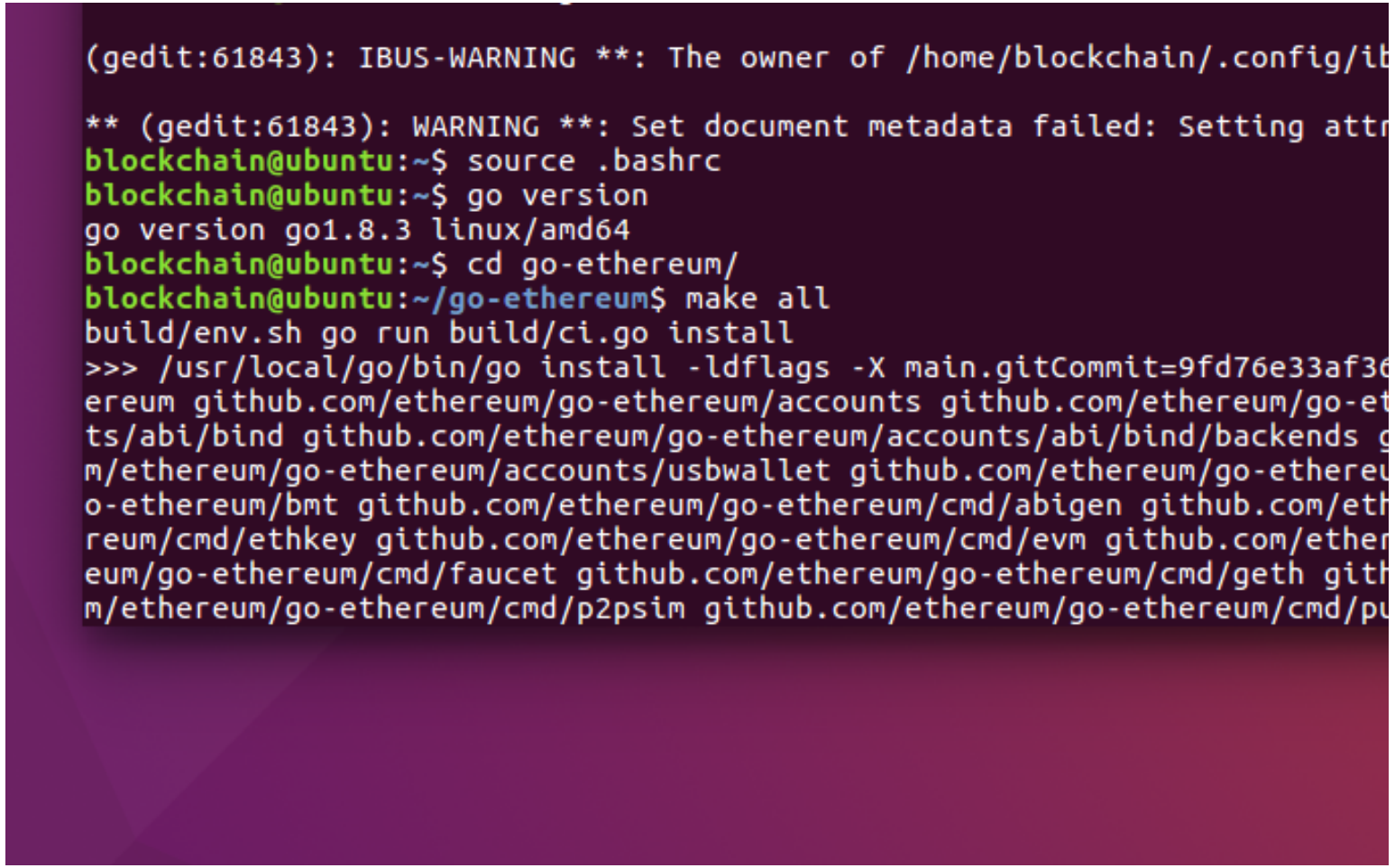
Now, finally we have to execute the make all command. Open a new terminal

Here’s what you have to do :-

Step 1: go to cd go-ethereum

Step 2: Execute make all command

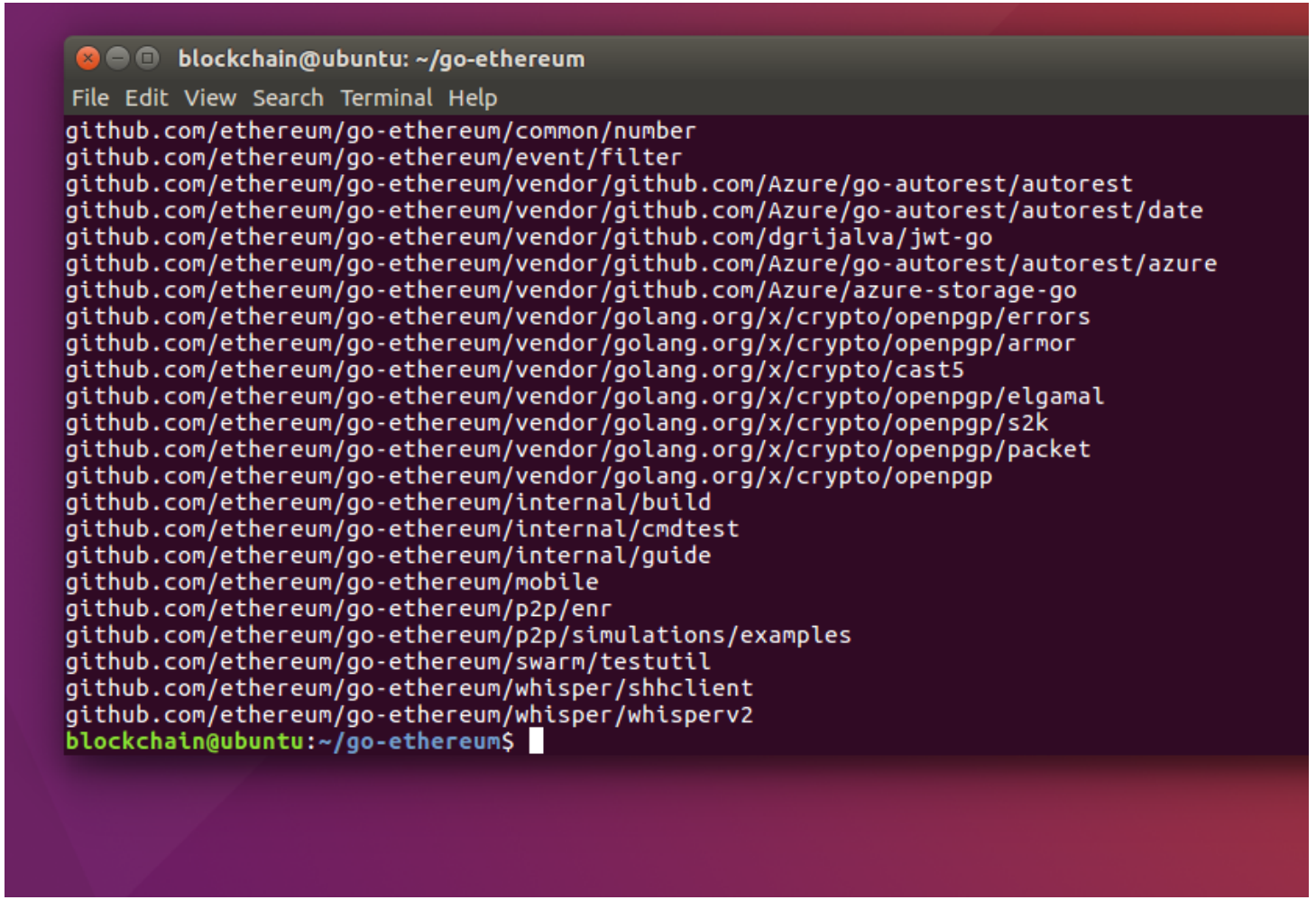
```
make all
```



“Make all” command is executing

It basically test git to build the target file as per the input

This command (make all) is on execution for a long time and so please be patient



Creating Blockchain – Genesis Block

Creating Blockchain – Genesis Block

> With this we have successfully completed the first part of setup. As first step we have successfully cloned the geth

Creating Blockchain – Genesis Block

Cloning Geth Code

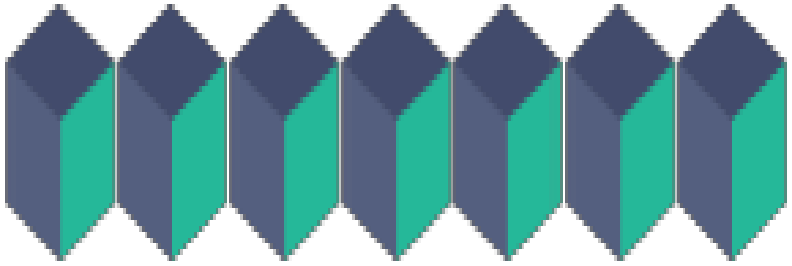
Creating a Genesis Block

Validating & Mining Ether

Making Rules for our Blockchain

```
cd go-ethereum

//Creating Genesis
(first) block mkdir
genesis
cd genesis
gedit genesis06072017.json
```

Genesis block



> Genesis block is the first block in the blockchain
> It contains the rules with respect to the blockchain

Creation of Genesis block is a two step process

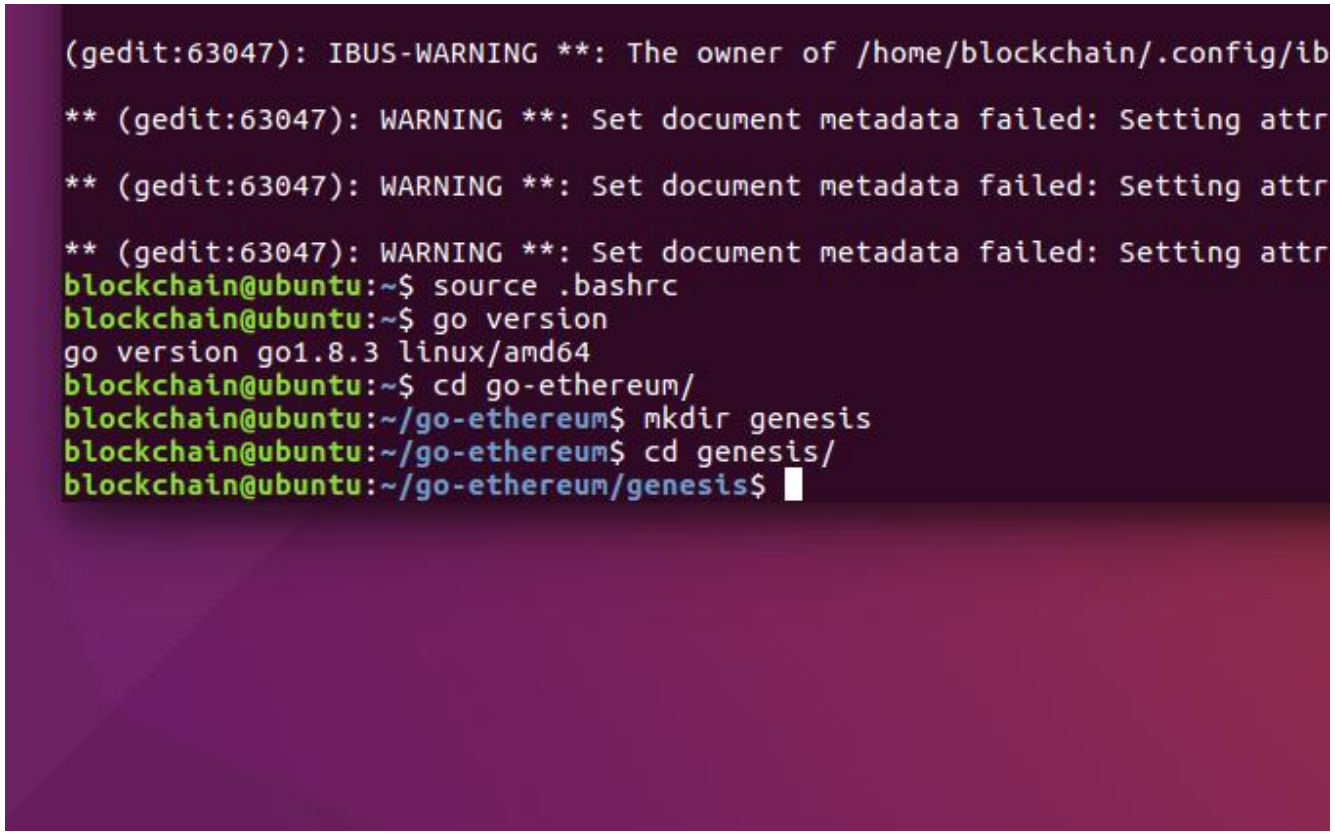
Step 1: Genesis folder is created

Step 2: Then, genesis json file is added to it

Creating Genesis folder

In our go ethereum folder we'll be creating genesis folder
To create genesis folder execute the following command:

```
mkdir genesis
```

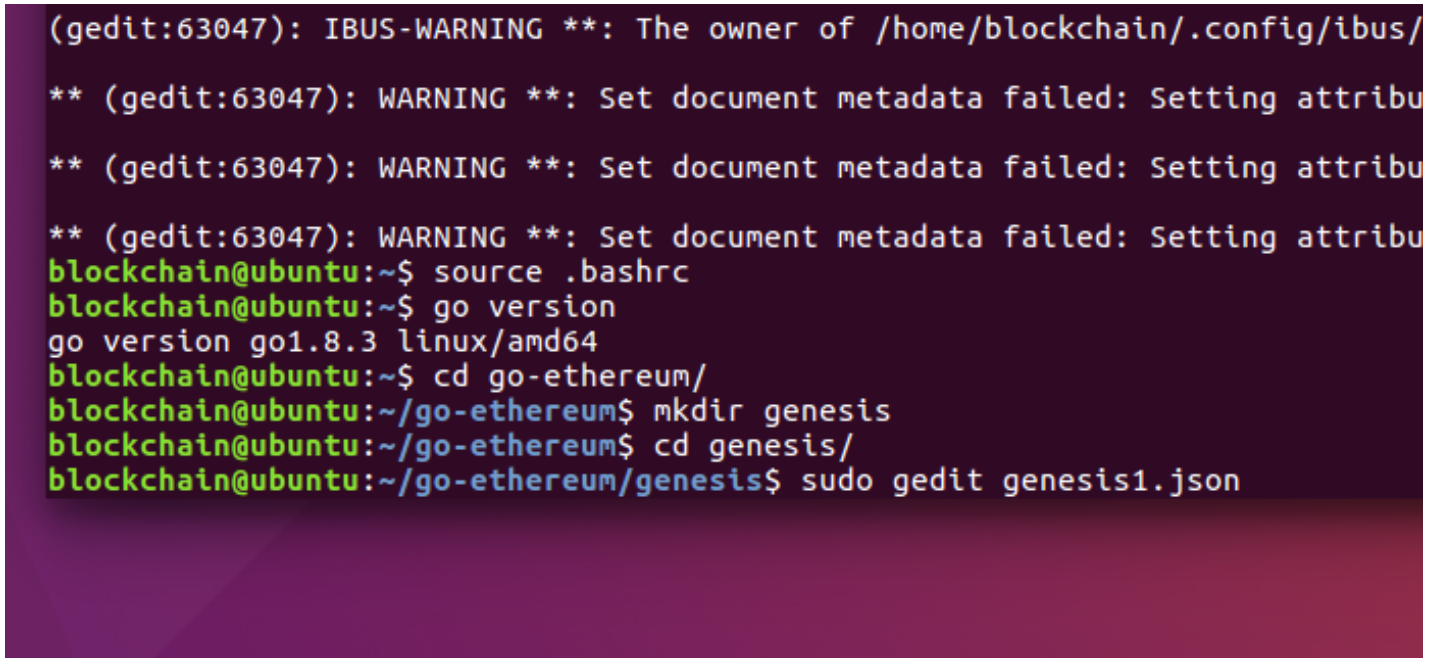


Create a new json file

Now, we go inside the genesis folder and create a new json file with all the rules of the blockchain
To create json file execute the following command:

```
gedit genesis3.json
```

In above command, genesis3 is just ther file name, and it can be anything of your Choice.



Genesis.json File Parameters

Genesis.json File Parameters

Before we write the details in the json file we just created, lets first have a look at the genesis block parameters

Genesis block parameters

nonce: A 64-bit hash, which proves, combined with the mix-hash, that a sufficient amount of computation has been carried out on this block

timestamp: A scalar value equal to the reasonable output of Unix time() function at this block inception

mixhash: A 256-bit hash which proves, combined with the nonce, that a sufficient amount of computation has been carried out on this block

difficulty: A scalar value corresponding to the difficulty level applied during the nonce discovering of block

alloc: Allows defining a list of pre-filled wallets. That's an Ethereum specific functionality to handle the “Ether pre-sale” period.

parentHash: The Keccak 256-bit hash of the entire parent block header (including its nonce and mixhash)

extraData: An optional free, but max. 32-byte long space to conserve smart things for ethernity

gasLimit: A scalar value equal to the current chain-wide limit of Gas expenditure per block

coinbase: The very first transaction included in the block by the miners

Making rules for our Blockchain

Making rules for our Blockchain

> In a genesis block you define a set of rules for a blockchain, which depends on the parameters we just discussed

Making rules for our Blockchain

Cloning Geth Code

Creating a Genesis Block

Validating & Mining Ether

Making Rules for our Blockchain

```
{
    "config": {
    "chainId": 123,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0
    },
     "nonce": "0x3",
     "timestamp": "0x0",
     "parentHash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
     "extraData": "0x0",
     "gasLimit": "0x4c4b40",
     "difficulty": "0x400",
     "mixhash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
     "coinbase": "0x0000000000000000000000000000000000000000",
     "alloc": {
     }
}
```

“Genesis.json” file

The code we just saw in the previous slide is to be added as part of genesis block

So just add the code and save the file

Starting Blockchain

> With that we are done with the third step of our blockchain development process. Now its time to initialize our blockchain

Starting Blockchain

Cloning Geth Code

Creating a Genesis Block

Validating & Mining Ether

Making Rules for our Blockchain

```
//Initializing the Blockchain
/home/blockchain/go-ethereum/build/bin/geth init --datadir ~/ethereum/net3
genesis.json

//Starting the geth console
/home/blockchain/go-ethereum/build/bin/geth --datadir ~/ethereum/net3/ --networkid 3 console
```

Starting Blockchain

- Our next step is to start our blockchain
- For starting our blockchain we need to execute the following command:

```
/home/blockchain/go-ethereum/build/bin/geth --datadir
~/ethereum/net3 init genesis/genesis1.json
```

In the above command /edureka is the user directory and can be replaced by the user that you have specified, here in our case its edureka.

~/ethereum/net3 init genesis/genesis3.json – this is the data directory of the genesis file

Starting Blockchain: Executing the Command

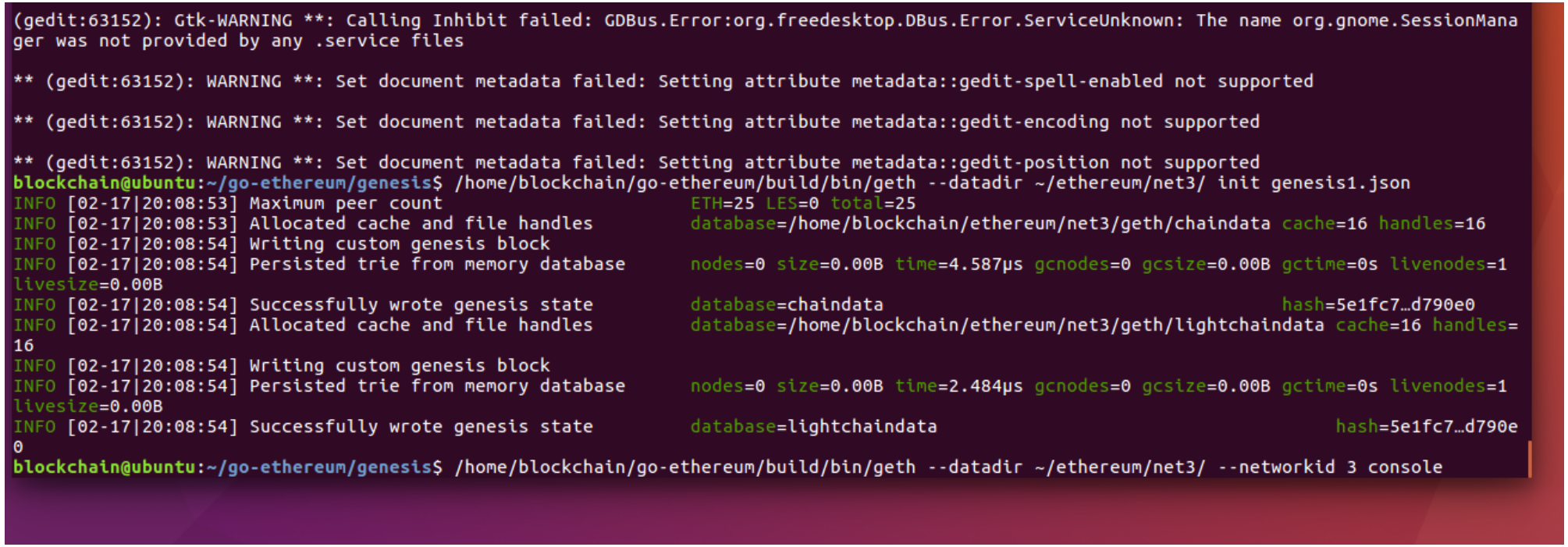
- Executing the command to start our blockchain

```
** (gedit:63152): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-position not supported
blockchain@ubuntu:~/go-ethereum/genesis$ /home/blockchain/go-ethereum/build/bin/geth --datadir ~/ethereum/net3/ init genesis1.json
are FO [02-17|20:08:53] Maximum peer count                       ETH=25 LES=0 total=25
INFO [02-17|20:08:53] Allocated cache and file handles         database=/home/blockchain/ethereum/net3/geth/chaindata cache=16 handles=16
INFO [02-17|20:08:54] Writing custom genesis block
INFO [02-17|20:08:54] Persisted trie from memory database      nodes=0 size=0.00B time=4.587µs gcnodes=0 gcsize=0.00B gctime=0s livenodes=1
livesize=0.00B
INFO [02-17|20:08:54] Successfully wrote genesis state         database=chaindata                                          hash=5e1fc7…d790e0
INFO [02-17|20:08:54] Allocated cache and file handles         database=/home/blockchain/ethereum/net3/geth/lightchaindata cache=16 handles=
16
INFO [02-17|20:08:54] Writing custom genesis block
INFO [02-17|20:08:54] Persisted trie from memory database      nodes=0 size=0.00B time=2.484µs gcnodes=0 gcsize=0.00B gctime=0s livenodes=1
livesize=0.00B
INFO [02-17|20:08:54] Successfully wrote genesis state         database=lightchaindata                                       hash=5e1fc7…d790e
0
blockchain@ubuntu:~/go-ethereum/genesis$
```

Starting the geth console

- Now that we have initialized the blockchain, to perform any operation on blockchain we have to use the geth console. Execute the following command to start the geth console:

```
/home/edureka/go-ethereum/build/bin/geth --datadir
~/ethereum/net3/ --networkid 3 console
```



Starting the geth console(cont'd)

- The geth console has started after executing the command (in the previous slide)

```
blockchain@ubuntu:~/go-ethereum/genesis$ /home/blockchain/go-ethereum/build/bin/geth --datadir ~/ethereum/net3/ --networkid 3 console
INFO [02-17|20:13:50] Maximum peer count                       ETH=25 LES=0 total=25
INFO [02-17|20:13:50] Starting peer-to-peer node               instance=Geth/v1.8.1-unstable-9fd76e33/linux-amd64/go1.8.3
INFO [02-17|20:13:50] Allocated cache and file handles         database=/home/blockchain/ethereum/net3/geth/chaindata cache=768 handles=512
WARN [02-17|20:13:50] Upgrading database to use lookup entries
INFO [02-17|20:13:50] Initialised chain configuration          config="{ChainID: 0 Homestead: 0 DAO: <nil> DAOSupport: false EIP150: <nil> E
IP155: 0 EIP158: 0 Byzantium: <nil> Engine: unknown}"
INFO [02-17|20:13:50] Disk storage enabled for ethash caches   dir=/home/blockchain/ethereum/net3/geth/ethash count=3
INFO [02-17|20:13:50] Disk storage enabled for ethash DAGs     dir=/home/blockchain/.ethash                    count=2
INFO [02-17|20:13:50] Initialising Ethereum protocol           versions="[63 62]" network=3
INFO [02-17|20:13:50] Loaded most recent local header          number=0 hash=5e1fc7…d790e0 td=131072
INFO [02-17|20:13:50] Loaded most recent local full block      number=0 hash=5e1fc7…d790e0 td=131072
INFO [02-17|20:13:50] Loaded most recent local fast block      number=0 hash=5e1fc7…d790e0 td=131072
INFO [02-17|20:13:50] Regenerated local transaction journal    transactions=0 accounts=0
INFO [02-17|20:13:50] Database deduplication successful        deduped=0
INFO [02-17|20:13:50] Starting P2P networking
INFO [02-17|20:13:52] UDP listener up                          self=enode://7a90046e5ebdc734f6a34bdc74f312a1a32af0843f393cff16c9ef7b49fd0332
69f85bc581686c718875369319cd47332f5e2bd48d9400ea5c87493fdcd6d81f@[::]:30303
INFO [02-17|20:13:52] RLPx listener up                         self=enode://7a90046e5ebdc734f6a34bdc74f312a1a32af0843f393cff16c9ef7b49fd0332
69f85bc581686c718875369319cd47332f5e2bd48d9400ea5c87493fdcd6d81f@[::]:30303
INFO [02-17|20:13:52] IPC endpoint opened                      url=/home/blockchain/ethereum/net3/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.1-unstable-9fd76e33/linux-amd64/go1.8.3
 modules: admin:1.0 debug:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

>
```

Mining Blockchain

Mining Blockchain

This brings us to the last step of setting up the blockchain where we'll be executing a set of commands to create accounts, check the balance, mine transactions and check the balance of the miners

Mining Blockchain

Cloning Geth Code

Creating a Genesis Block

Validating & Mining Ether

Making Rules for our Blockchain

- > personal.newAccount()
- > eth.accounts
- > eth.blockNumber()
- > miner.start()
- > miner.stop()
- > eth.blockNumber()
- > eth.getBalance("0x5f069120a76ac078a46b3ce94c6a350b330b68b8")
- > exit

Discussing Commands

personal.newAccount() : it creates a new account as part of your blockchain which has a specific wallet attached to it

eth.accounts: it helps you check the various accounts which are part of your blockchain

eth.blockNumber(): this helps you to identify the number of blocks that are part of your blockchain

miner.start(): this function is used to start the mining process

miner.stop(): It stops the mining process

eth.blockNumber(): executing this command after the mining process tells you at which block number you are at after performing the mining operation

eth.getBalance("0x5f069120a76ac078a46b3ce94c6a350b330b68b8"): this command is used to check the ether balance in the specified account

exit:

Executing the commands: Creating a new account

As discussed, the first command we'll be executing is to create a new account
To create a new account execute the following command:

```
personal.newAccount()
```

The above command creates a new account on your blockchain. To create multiple account you just have to execute the same command

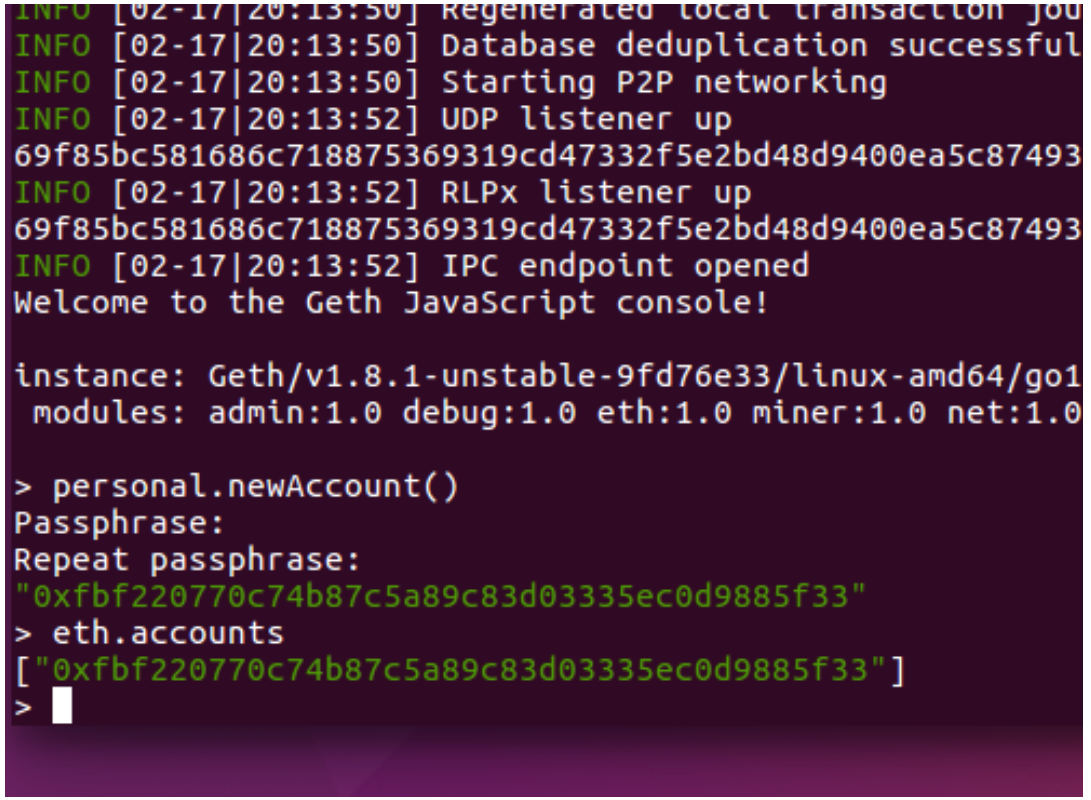
New account is created and an address is assigned

Executing the commands: Checking the accounts

Now, let's see what are the accounts present in your blockchain

To check the accounts, execute the following command:

```
eth.accounts
```



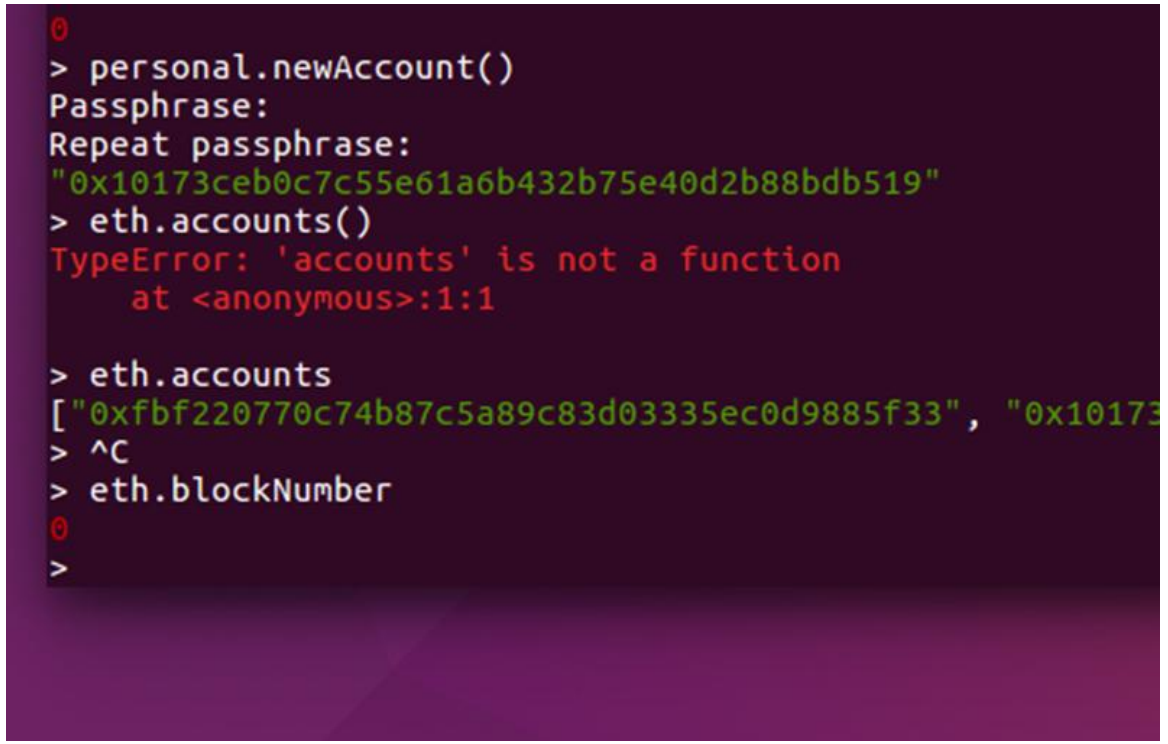
Displays the existing accounts that were created on this blockchain,

Executing the commands: The current block number

Let's find out the latest block number:

To find out the current block number execute the following command:

```
eth.blockNumber()
```



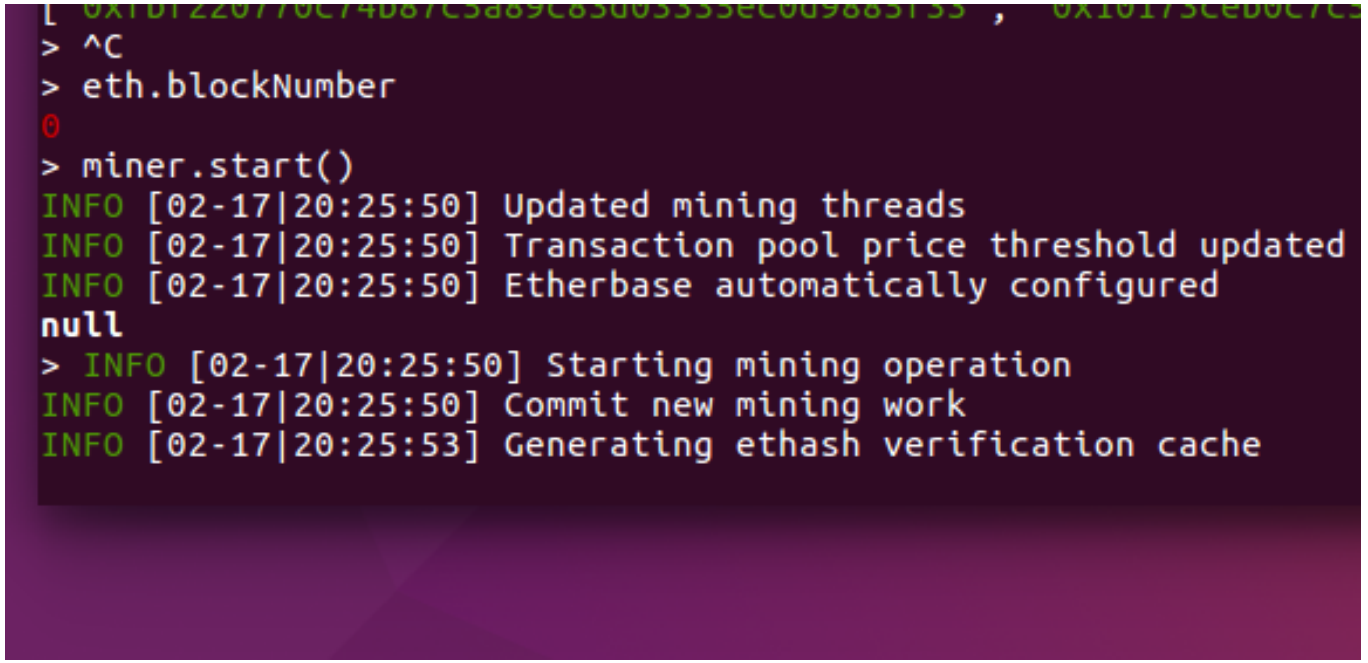
Before executing the above command make sure that you have multiple accounts created

Executing the commands: Starting Mining operation

Now that we have multiple accounts on our blockchain, let's start mining process

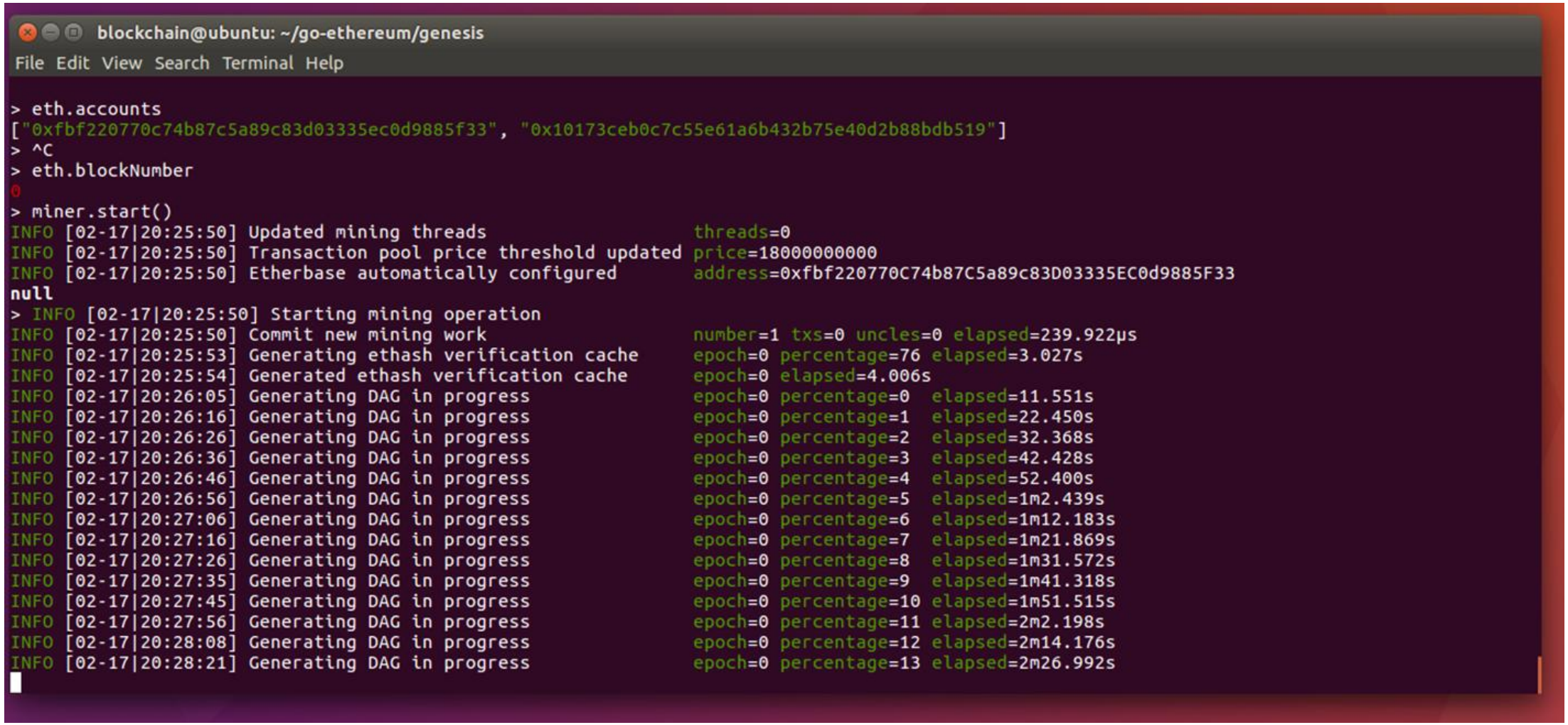
To start the mining process execute the following command:

```
miner.start()
```



Mining process initiates and it starts allocating the resources

Mining starts

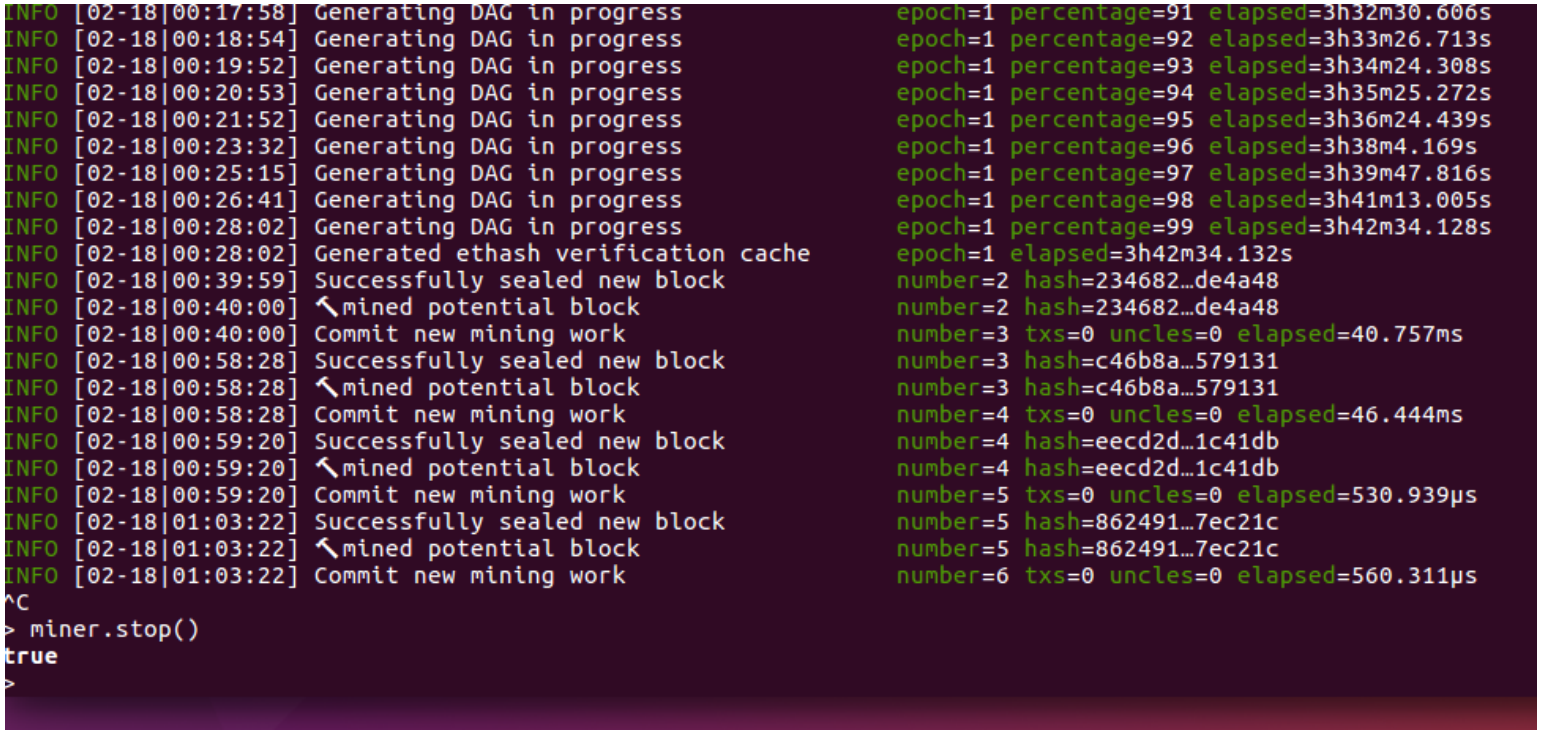


Executing the commands: Terminate the mining

After running the mining process for quite a some time now, let's stop the mining process

To stop the mining process execute the following command:

```
miner.stop()
```

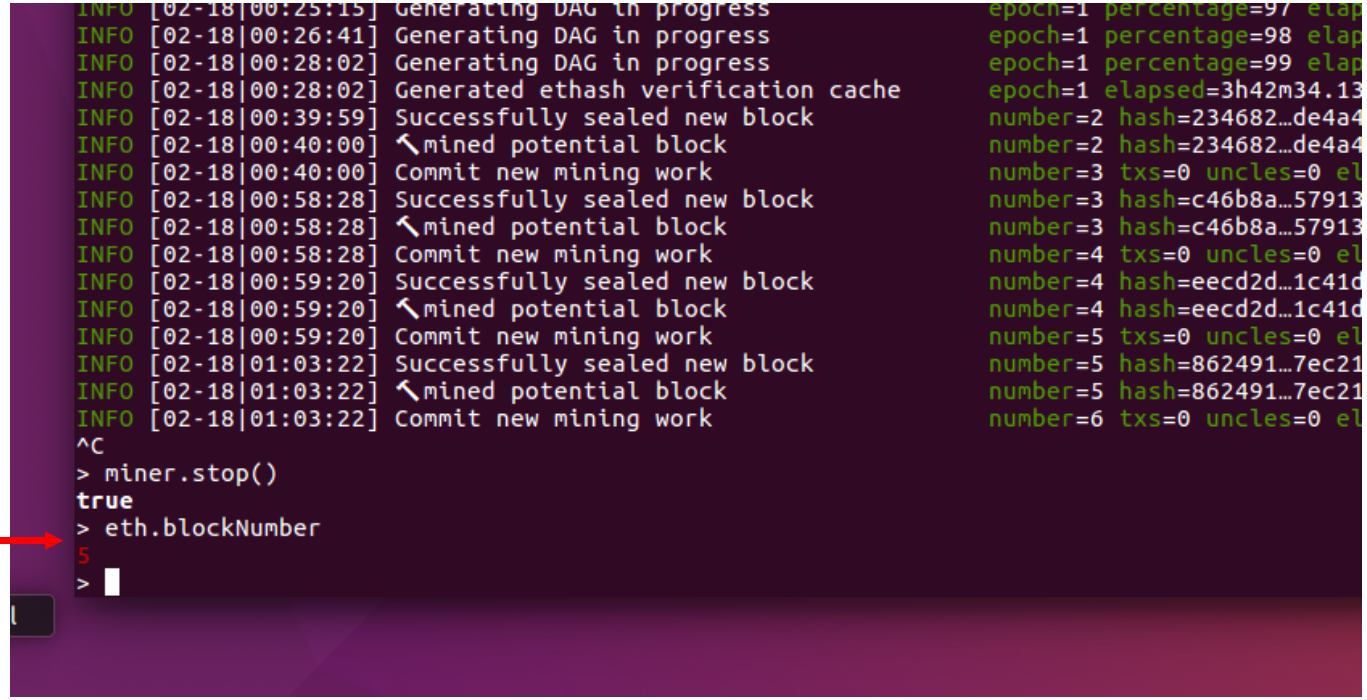


Executing the commands: Current block after Mining

Let's check the current block number after the mining process

To check the current block number execute the following command:

```
eth.blockNumber()
```

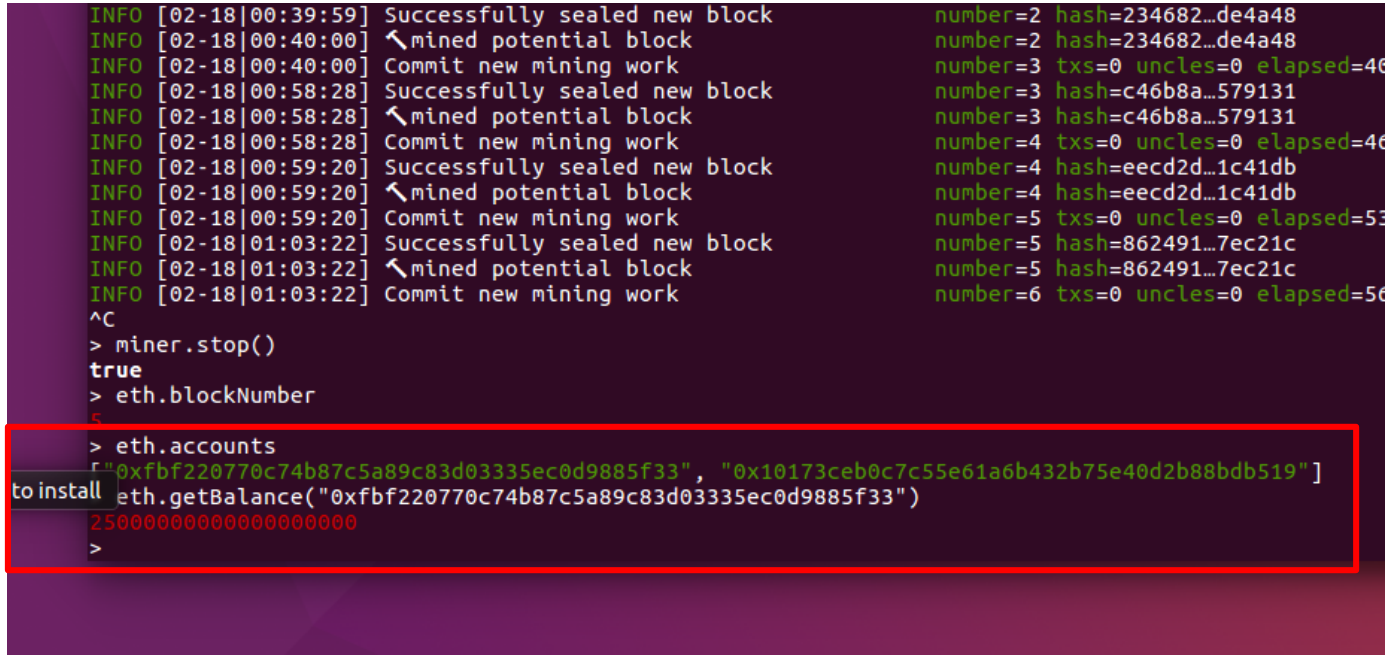


Executing the commands: Checking Balance

Now, lets check the balance of the accounts after the mining

To check the balance execute the following commands:
Step 1: First execute eth.accounts command to view all the accounts on the blockchain
Step 2: execute

```
eth.getBalance("Address of the ethereum")
```



Mining Blockchain

> You have successfully setup and executed your own private blockchain network

Thank You

Email us – support@intellipaat.com

Visit us - https://intellipaat.com