

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



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

```
' ethereumjs-testrpc@v6.0.3
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) 0x56333600faa67af532214a75d3cae67019972d7
(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. Prevent this 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 Node.js
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



The screenshot shows a web browser window with the title bar "Remix - Solidity IDE" and "Ethereum Block Ex". The address bar displays "localhost:8000/#/". The main content area has a dark header with the text "Ether Block Explorer" and search fields for "Tx Hash, Address or Block" and a green "Search" button. Below the header, the text "Welcome to the Etherparty Block Explorer!" is centered. A table titled "Latest Block: 0" is displayed, showing one row with data: Block # 0, Tx # 0, Size 1000, and Timestamp 1502435295.

Block #	Tx #	Size	Timestamp
0	0	1000	1502435295

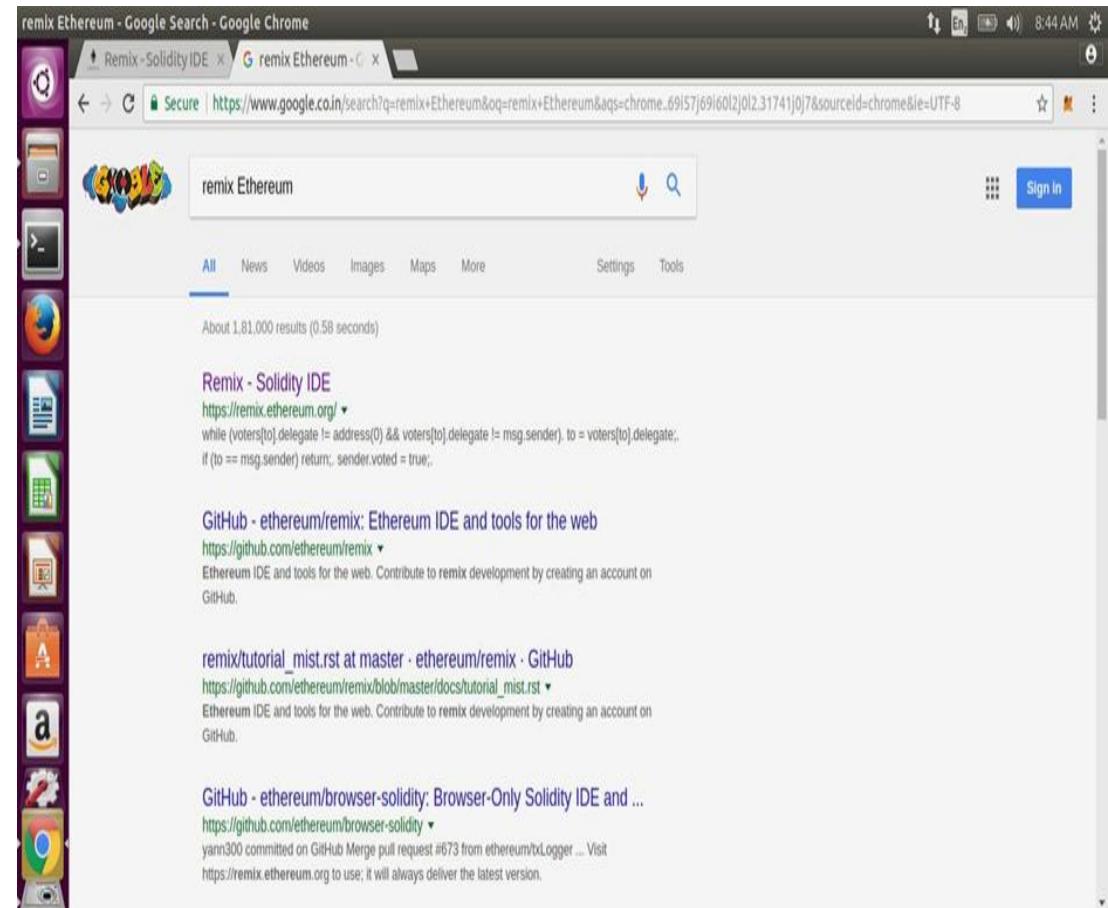
© Etherparty.io 2017 | [Fork me on GitHub](#)

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 screenshot shows the Remix Solidity IDE interface within a Google Chrome browser window. The left sidebar contains icons for various tools and environments, with the Firefox icon highlighted by a red arrow and a callout box containing the text "This is a sample of smart contract". The main workspace displays a Solidity code editor with the file "browser/Sample.sol" open. The code defines a simple storage contract:

```
pragma solidity ^0.4.0;
contract SimpleStorage {
    uint storedData;
    function set(uint x) {
        storedData = x;
    }
    function get() constant returns (uint) {
        return storedData;
    }
}
```

The right side of the interface includes a "Contract" tab, settings for environment, account, gas limit, and value, and buttons for publishing, attaching, transacting, and calling the contract. A preview pane at the bottom shows the contract details.

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

The screenshot shows the Remix Solidity IDE interface in a Google Chrome browser window. The URL is https://remix.ethereum.org/#version=soljson-v0.4.15+commit.bbb8e64f.js. On the left, there's a sidebar with icons for different environments: Ubuntu, Terminal, Ethereum Block Explorer, Remix - Solidity IDE (selected), and Ethereum - Solidity IDE. The main workspace displays a Solidity contract named 'ballot.sol' with the following code:

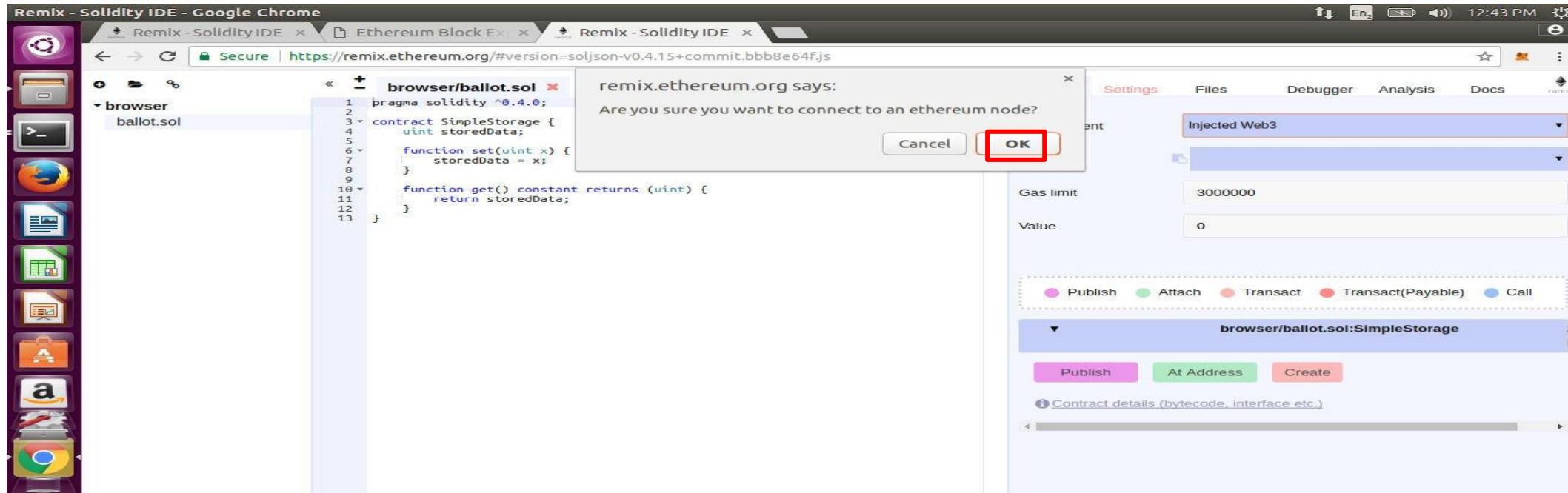
```
pragma solidity ^0.4.0;
contract SimpleStorage {
    uint storedData;
    function set(uint x) {
        storedData = x;
    }
    function get() constant returns (uint) {
        return storedData;
    }
}
```

To the right of the code editor, there are tabs for Contract, Settings, Files, Debugger, Analysis, and Docs. Below the tabs, there are several configuration fields: Environment (set to Injected Web3, highlighted with a red box), Account, Gas limit (3000000), and Value (0). A message at the bottom says "Solidity compiler is currently loading. Please wait a moment...".

Confirm web3 option

It will ask for your confirmation to connect to ethereum node

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

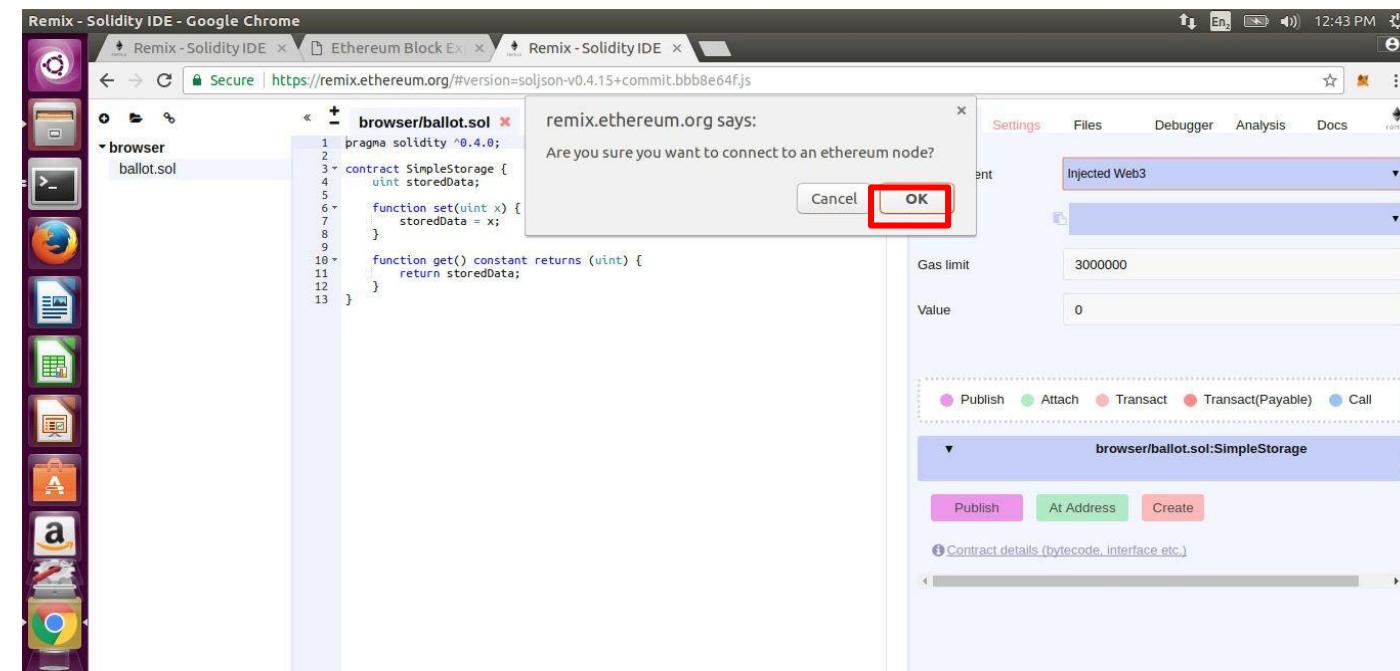


Selecting Web3 provider end point



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 via testrpc

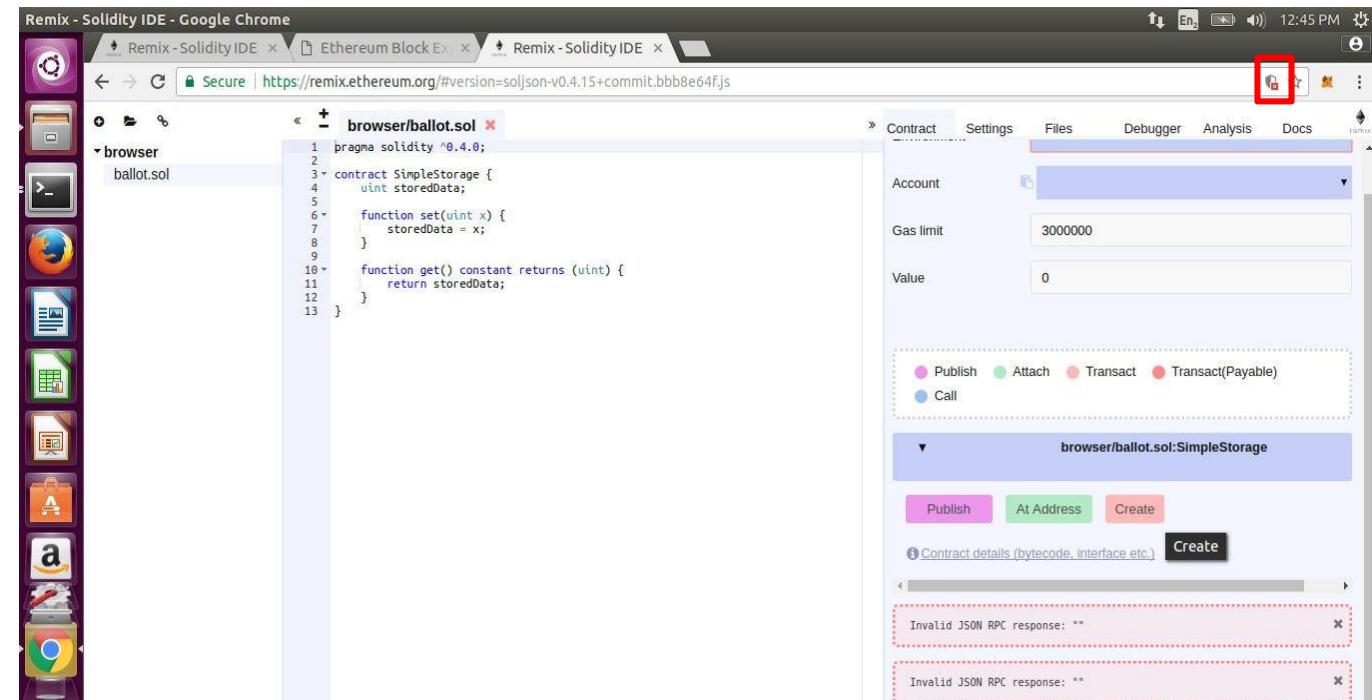
- Since we are using the browser for development and testing purpose, our endpoint will be the local host on the port where our testrpc is listening
- So here, just click ok button on the pop



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

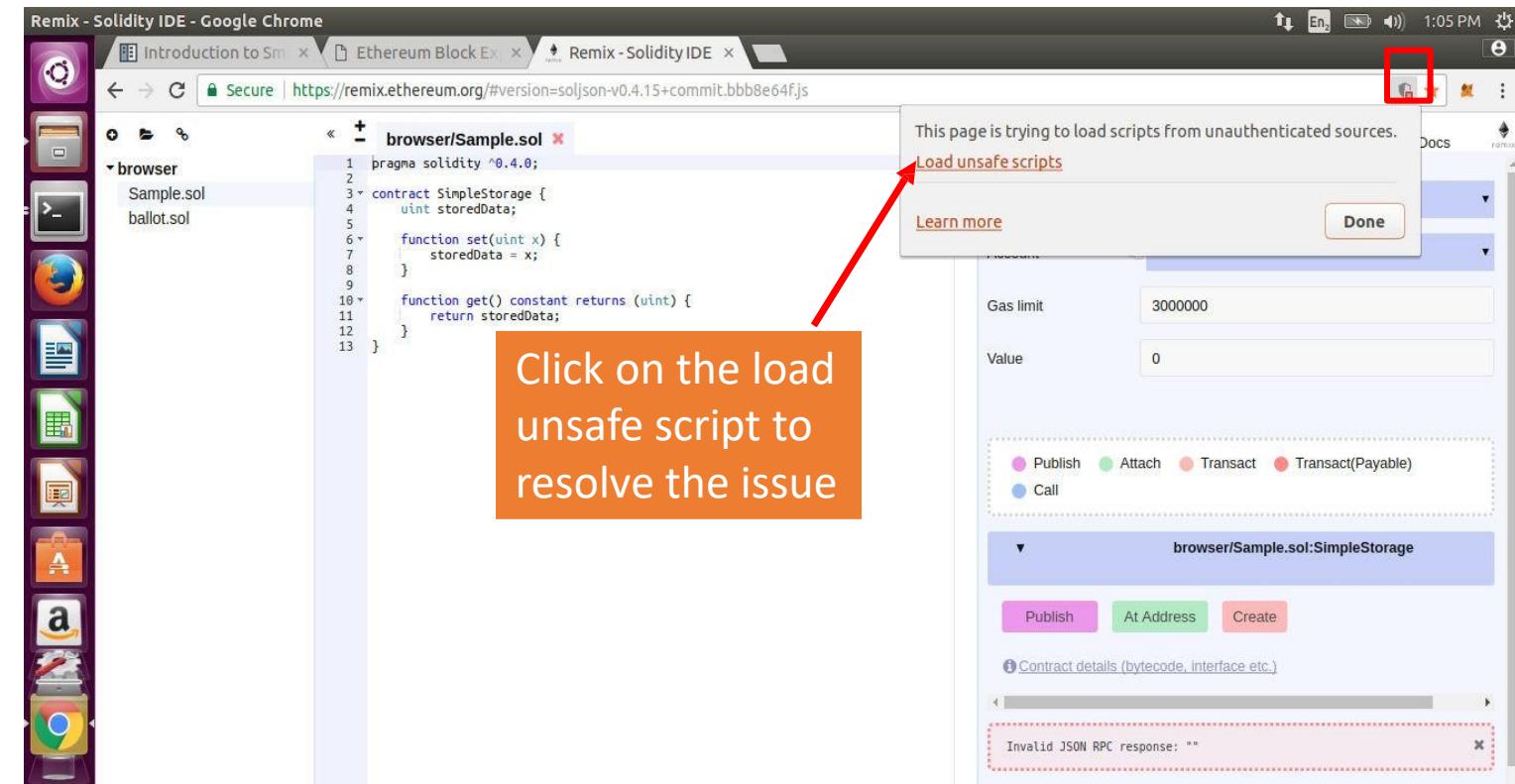
- The icon shown in red indicates the restriction. Let's see how we resolve it



Load Unsafe script

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

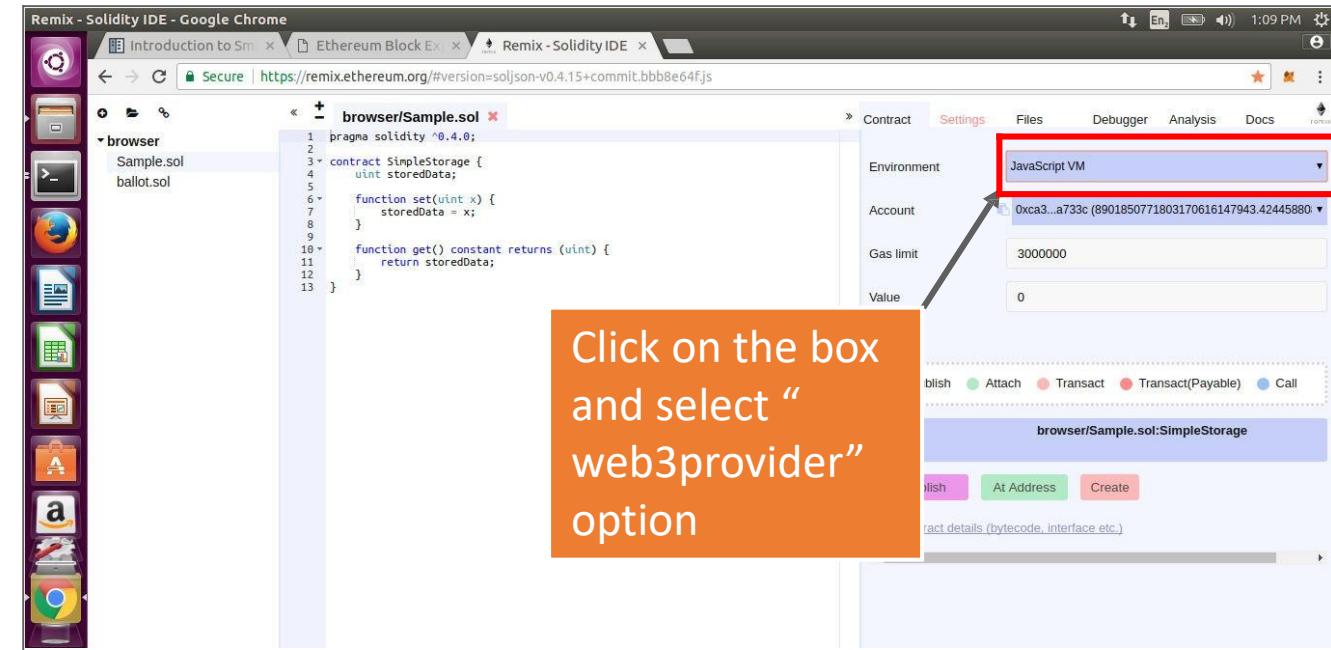
- As indicated, if you'll click the icon, it opens up with the message "The page is trying to load scripts from unauthenticated sources"
- Select the **"load unsafe scripts options"**
- This will now connect you to the local host port 8545



Set the “Environment”

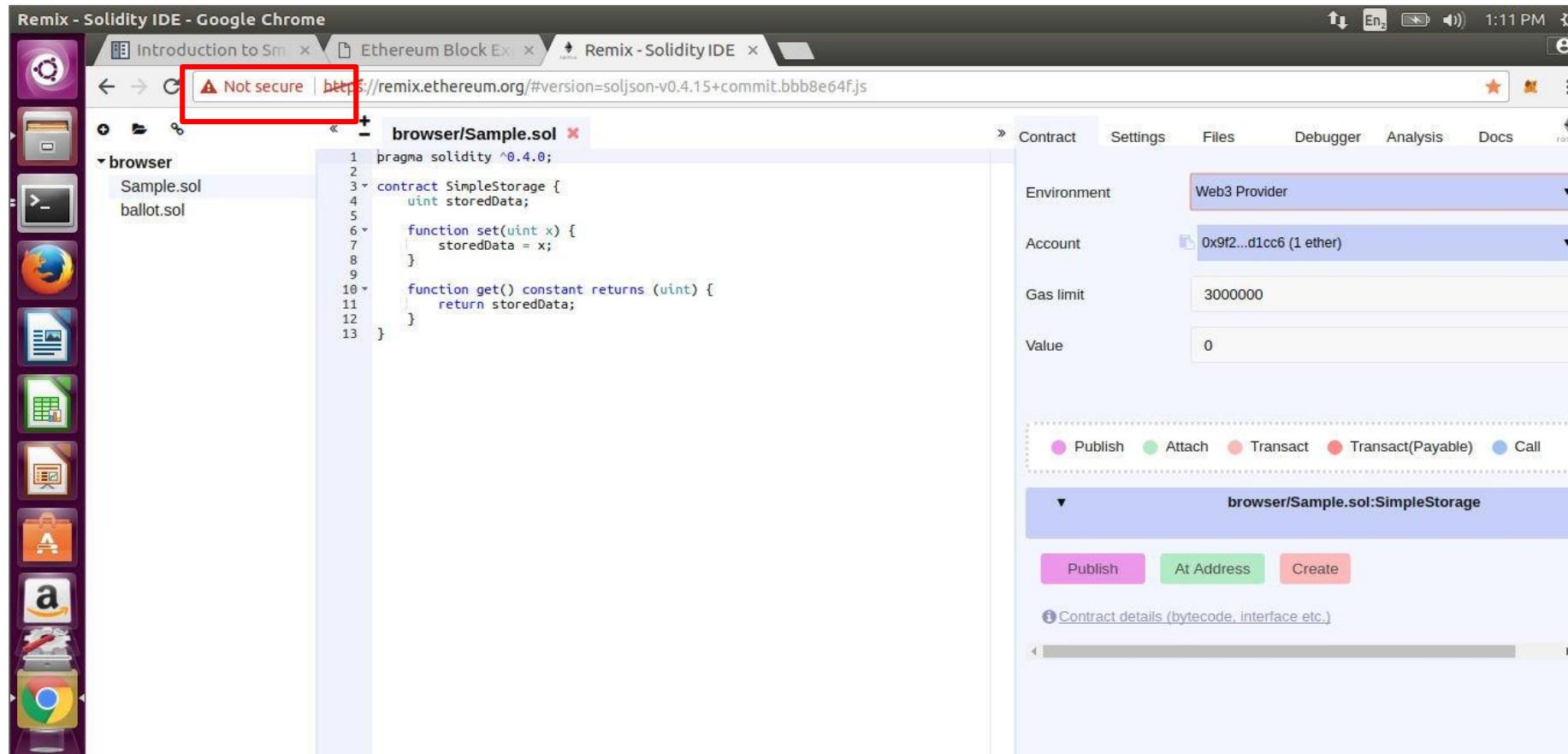
The connection with port is established as you can see the account balances in the window shown below, however we need to setup the “environment” again

- 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

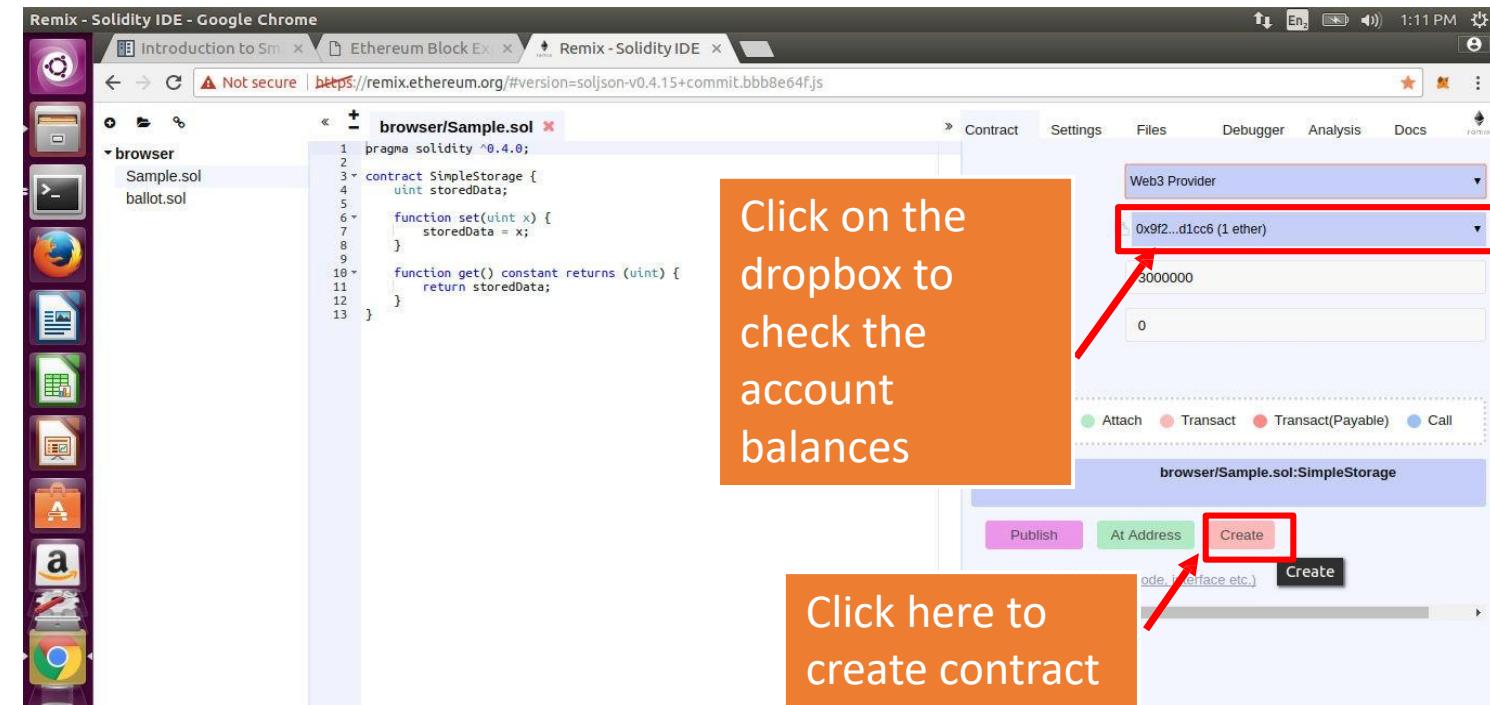


Creating Smart Contract

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

To create contract:

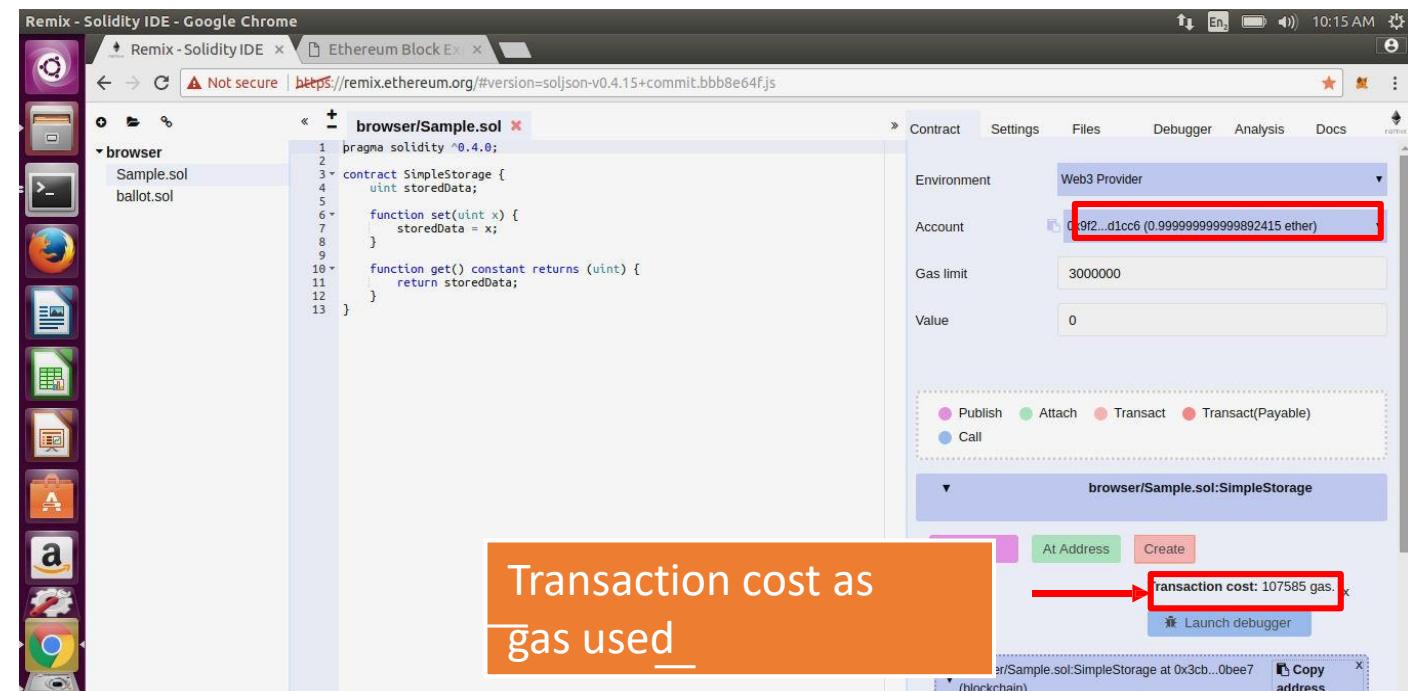
- Select the account that you want to use to push your contract
- After selecting the account click on create button



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



The screenshot shows the Remix Solidity IDE interface. On the left, there's a file tree with 'browser/Sample.sol' selected. The code editor displays the following Solidity code:

```

1 pragma solidity ^0.4.0;
2
3 contract SimpleStorage {
4     uint storedData;
5
6     function set(uint x) {
7         storedData = x;
8     }
9
10    function get() constant returns (uint) {
11        return storedData;
12    }
13 }

```

On the right, the 'Environment' tab is active, showing a Web3 Provider and an account balance of 0.9999999999982415 ether. The 'Gas limit' is set to 3000000 and 'Value' is 0. Below the environment settings, there are buttons for Publish, Attach, Transact, Transact(Payable), Call, At Address, Create, and Launch debugger. A tooltip box highlights the 'Create' button with the text "Transaction cost as gas used". Another red box highlights the 'transaction cost: 107585 gas.' message.

Check the block that includes your transaction

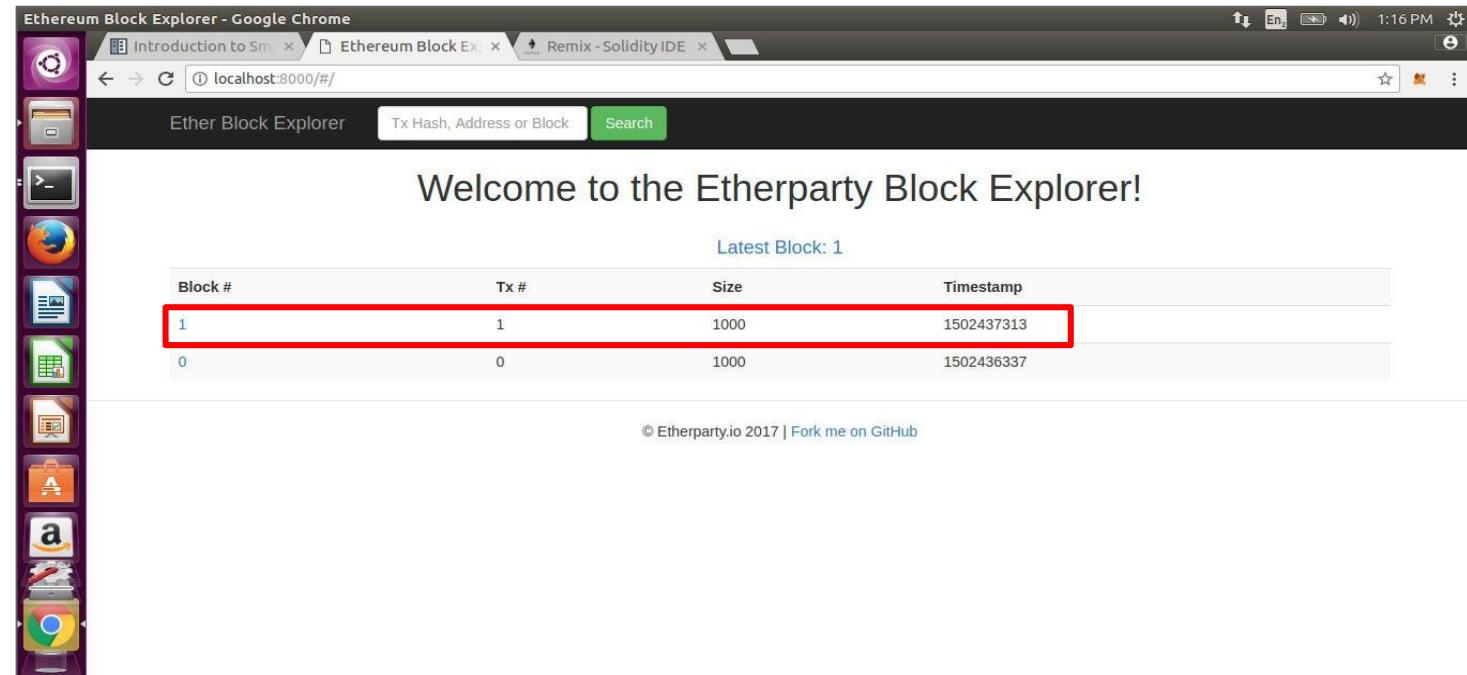
You can check the block details where your transaction is included

- 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

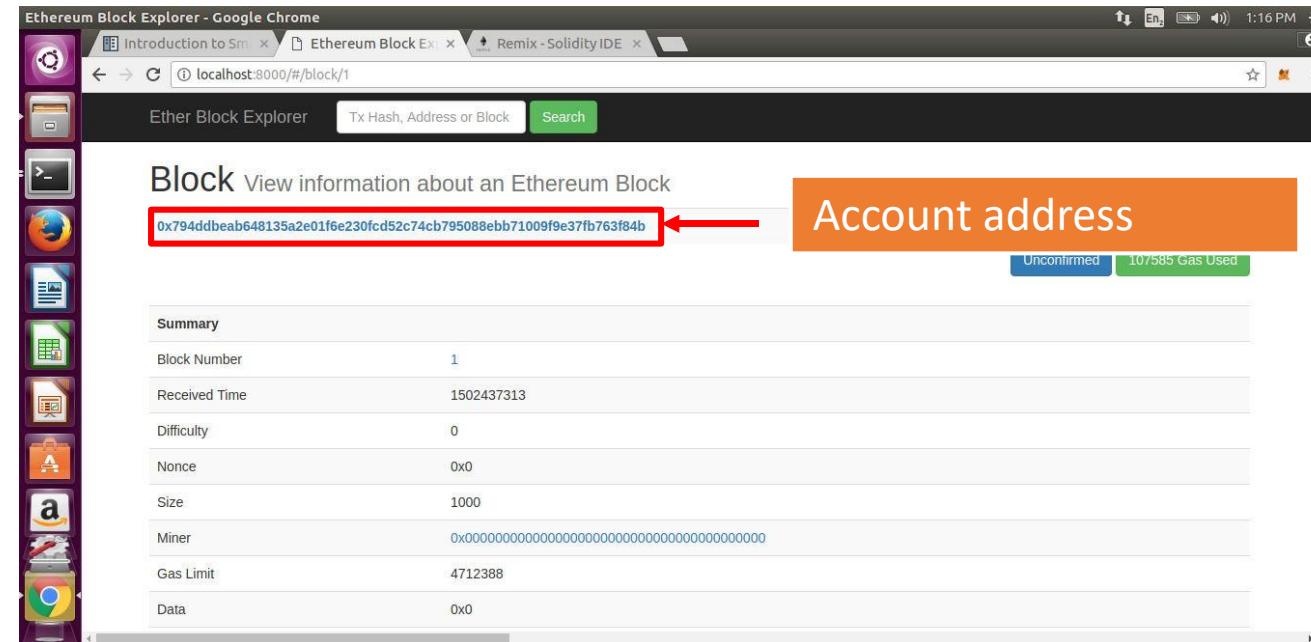


Block #	Tx #	Size	Timestamp
1	1	1000	1502437313
0	0	1000	1502436337

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

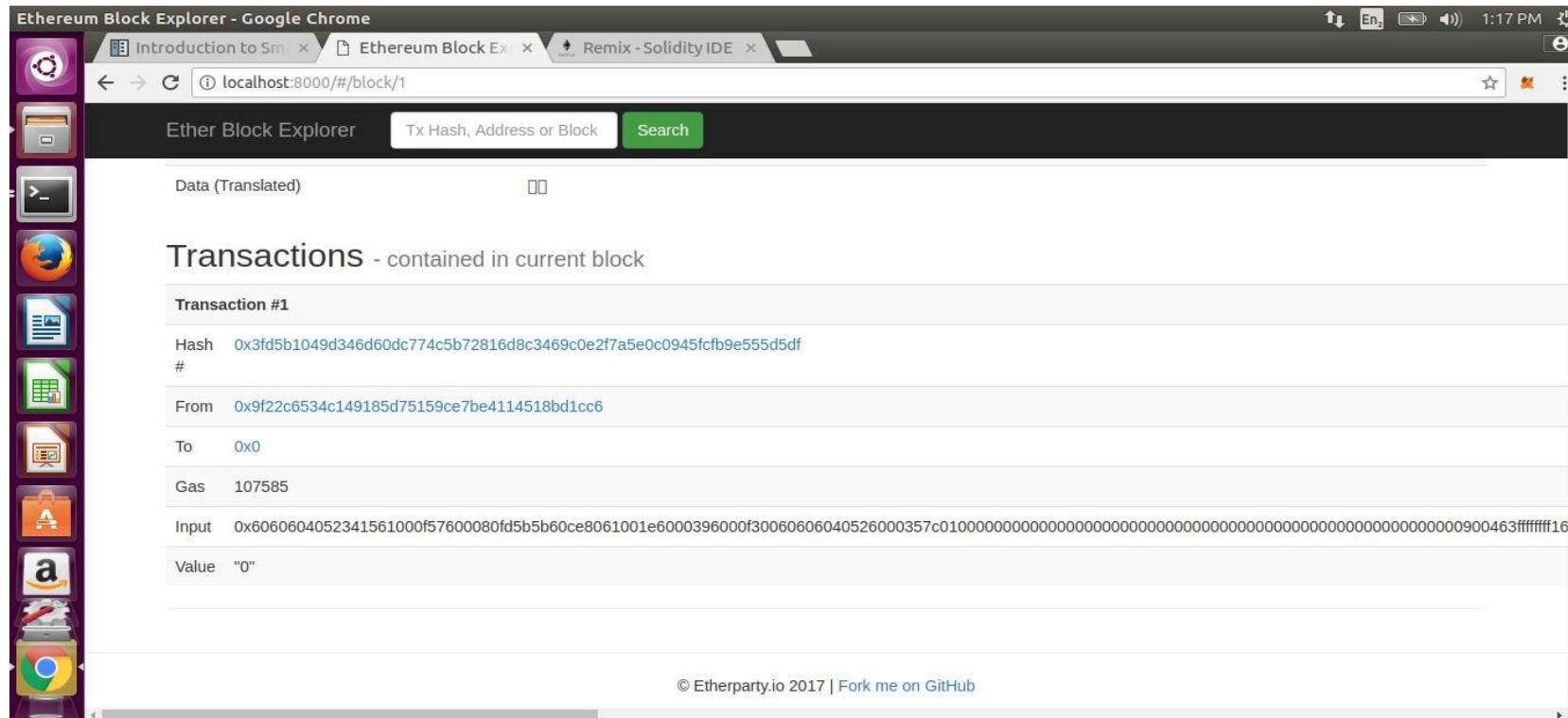


A screenshot of a web browser displaying the Ethereum Block Explorer. The URL in the address bar is `localhost:8000/#/block/1`. The page title is "Block View information about an Ethereum Block". A red arrow points from the text "Account address" to the block hash `0x794ddbeab648135a2e01f6e230fcda52c74cb795088ebb71009f9e37fb763f84b`, which is highlighted with a red box. To the right of the hash, there is an orange box containing the text "Account address". Below the hash, the status "Unconfirmed" is shown next to "107585 Gas Used". On the left, a sidebar shows various application icons. The main content area displays a table with the following data:

Summary	
Block Number	1
Received Time	1502437313
Difficulty	0
Nonce	0x0
Size	1000
Miner	0x000
Gas Limit	4712388
Data	0x0

Transaction details

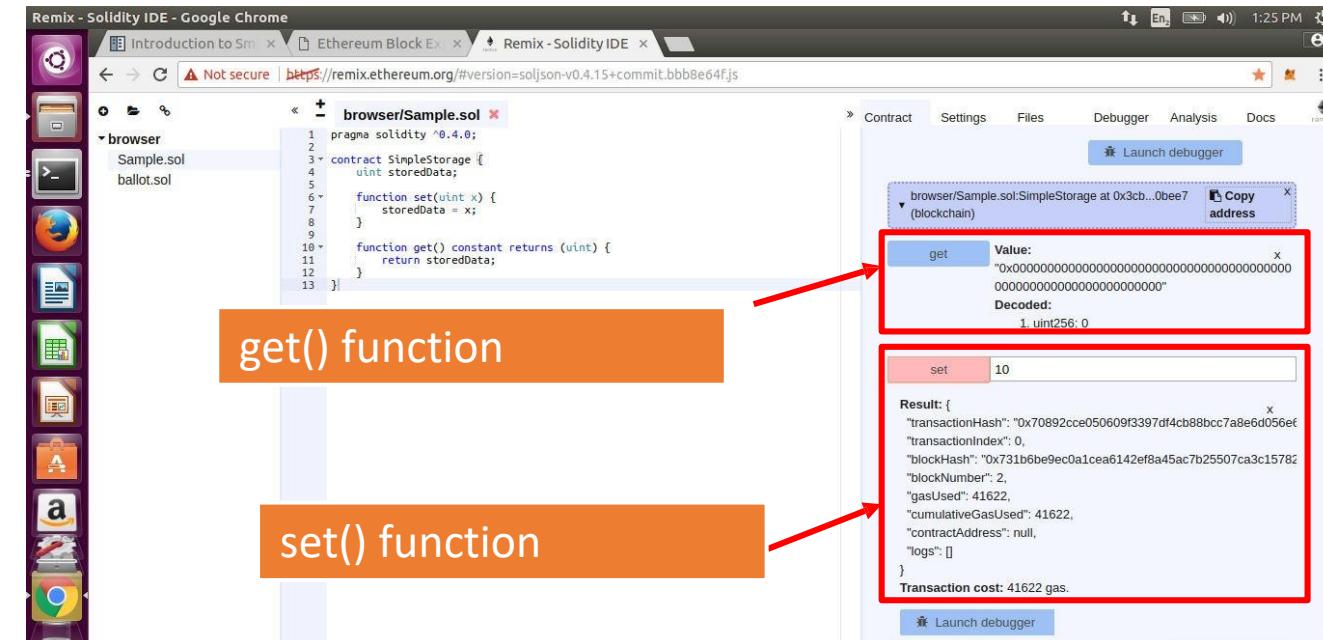
Clicking on the account address displays the transaction details



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 new transaction will get created
- In the indicated set() method the value is set to 10



Remix - Solidity IDE - Google Chrome

Not secure https://remix.ethereum.org/#version=soljson-v0.4.15+commit.bbb8e64f.js

Contract Settings Files Debugger Analysis Docs

Launch debugger

browser/Sample.sol

```
pragma solidity >0.4.0;
contract SimpleStorage {
    uint storedData;
    function set(uint x) {
        storedData = x;
    }
    function get() constant returns (uint) {
        return storedData;
    }
}
```

get Value: x
0x00
Decoded: 1. uint256: 0

set 10

Result: {
"transactionHash": "0x70892cce050609f3397df4cb88bcc7a8e6d056ef",
"transactionIndex": 0,
"blockHash": "0x731b6be9ec0a1ceaf142ef8a45ac7b25507ca3c15782",
"blockNumber": 2,
"gasUsed": 41622,
"cumulativeGasUsed": 41622,
"contractAddress": null,
"logs": []
}
Transaction cost: 41622 gas.

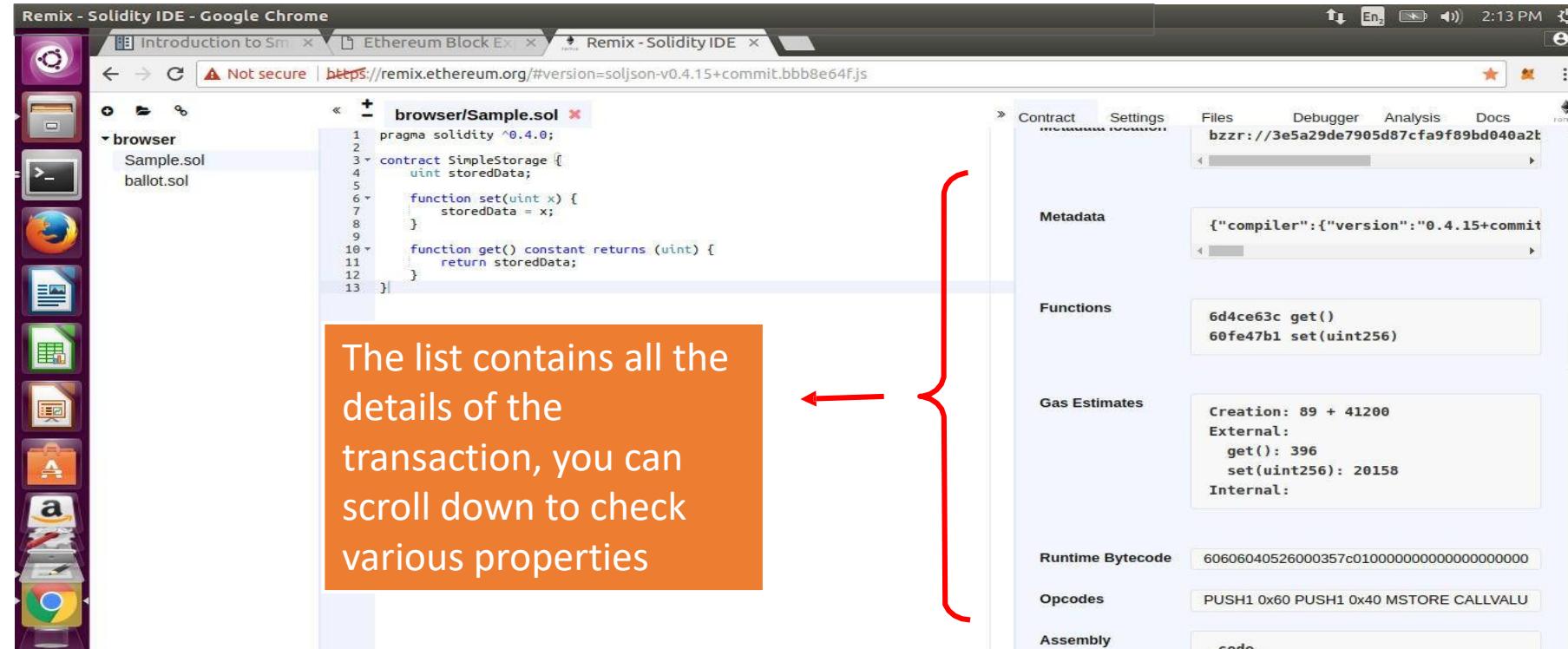
Launch debugger

get() function

set() function

Checking Contract details

Contract details can be checked by clicking the contract details



MetaMask

MetaMask



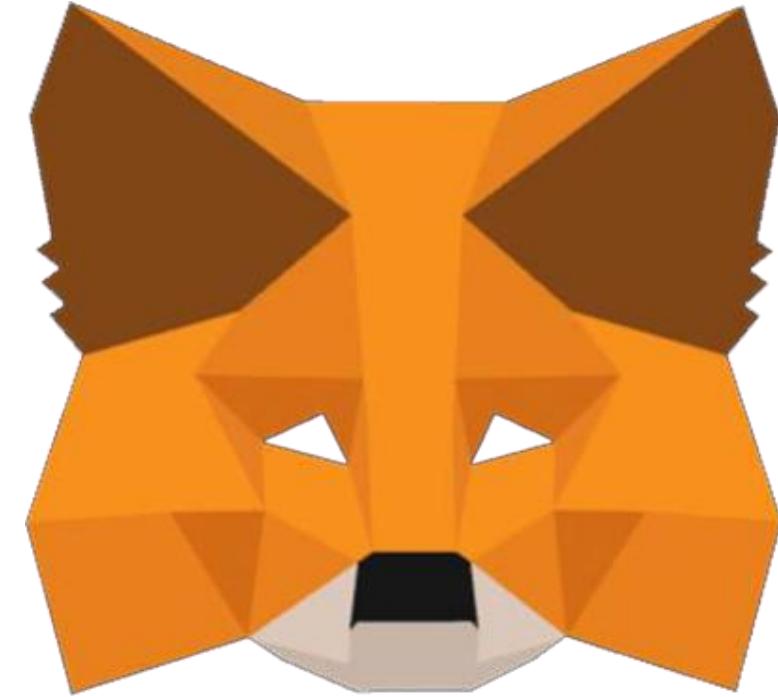
“
MetaMask brings Ethereum to the browser.
Let's see how to install MetaMask as a chrome
extension.
”

MetaMask

“

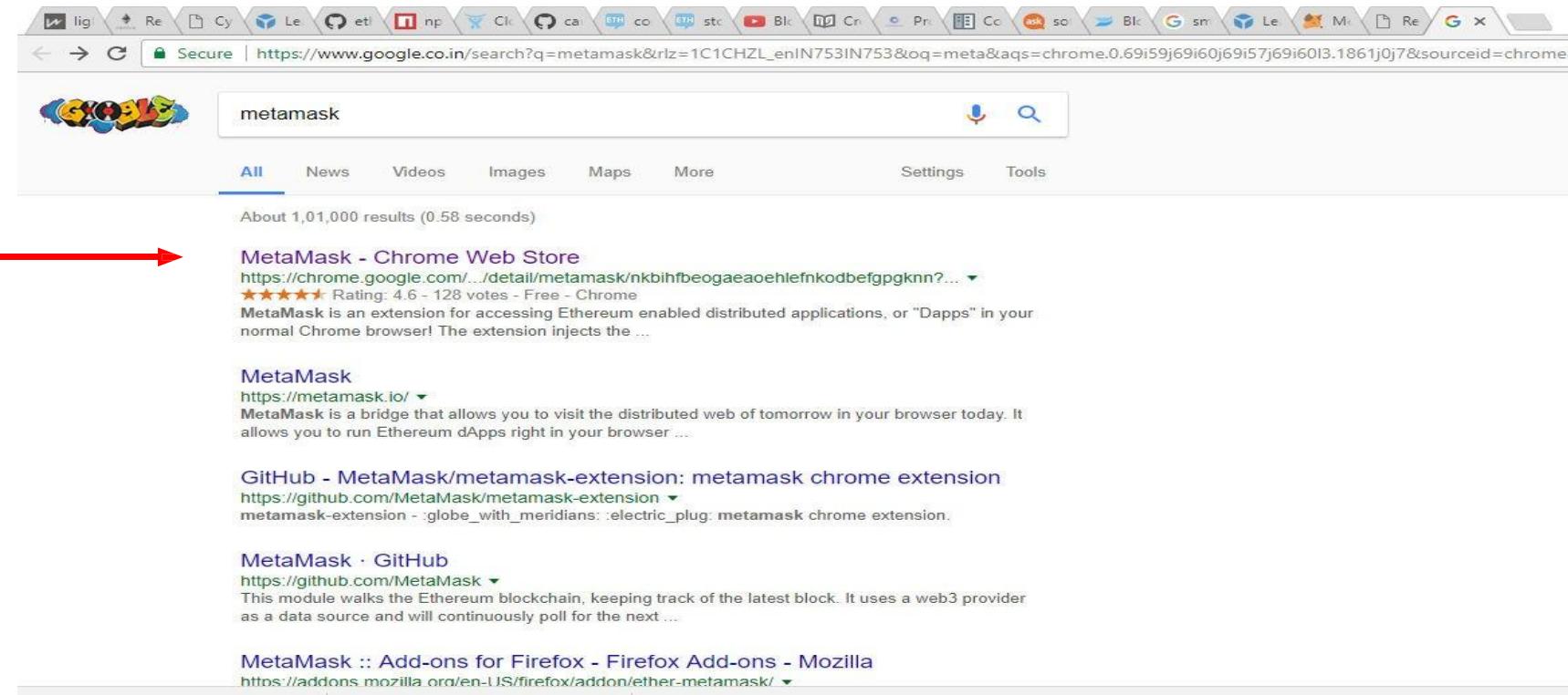
- MetaMask is a bridge that allows you to visit the distributed web of tomorrow in your browser today
- It allows you to run Ethereum dApps right in your browser without running a full Ethereum node
- MetaMask includes a secure identity vault, providing a user interface to manage your identities on different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

”



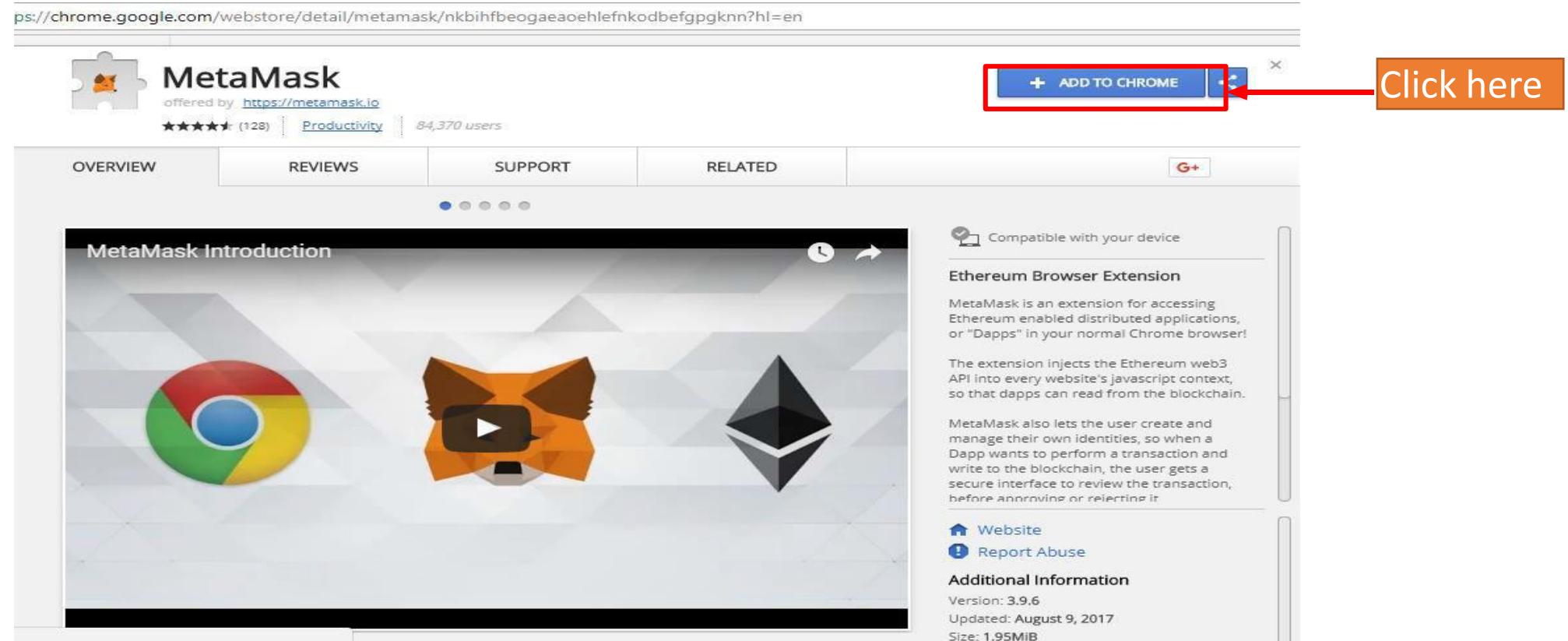
Open your chrome browser

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



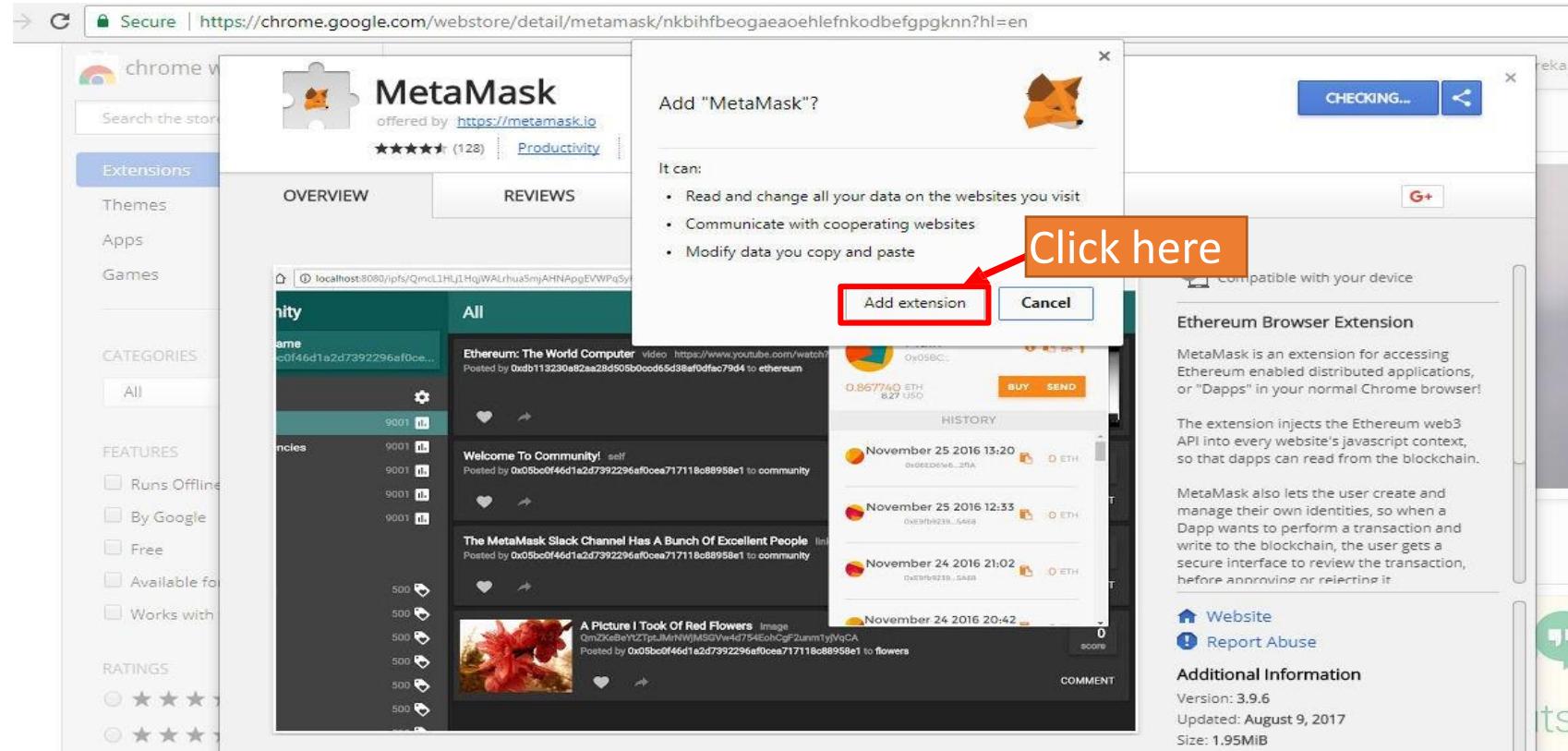
Add to Chrome

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



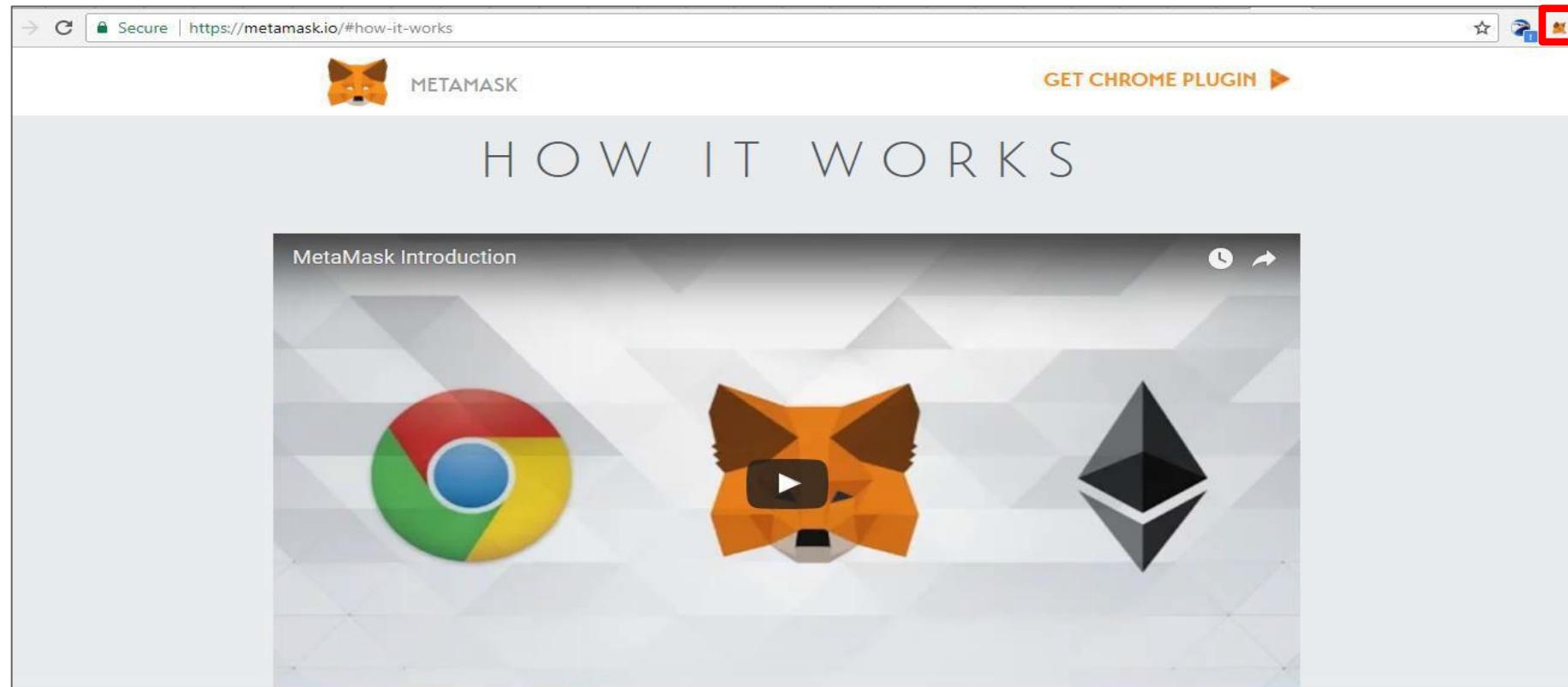
Add Extension

- It will ask you to “Add Extension”



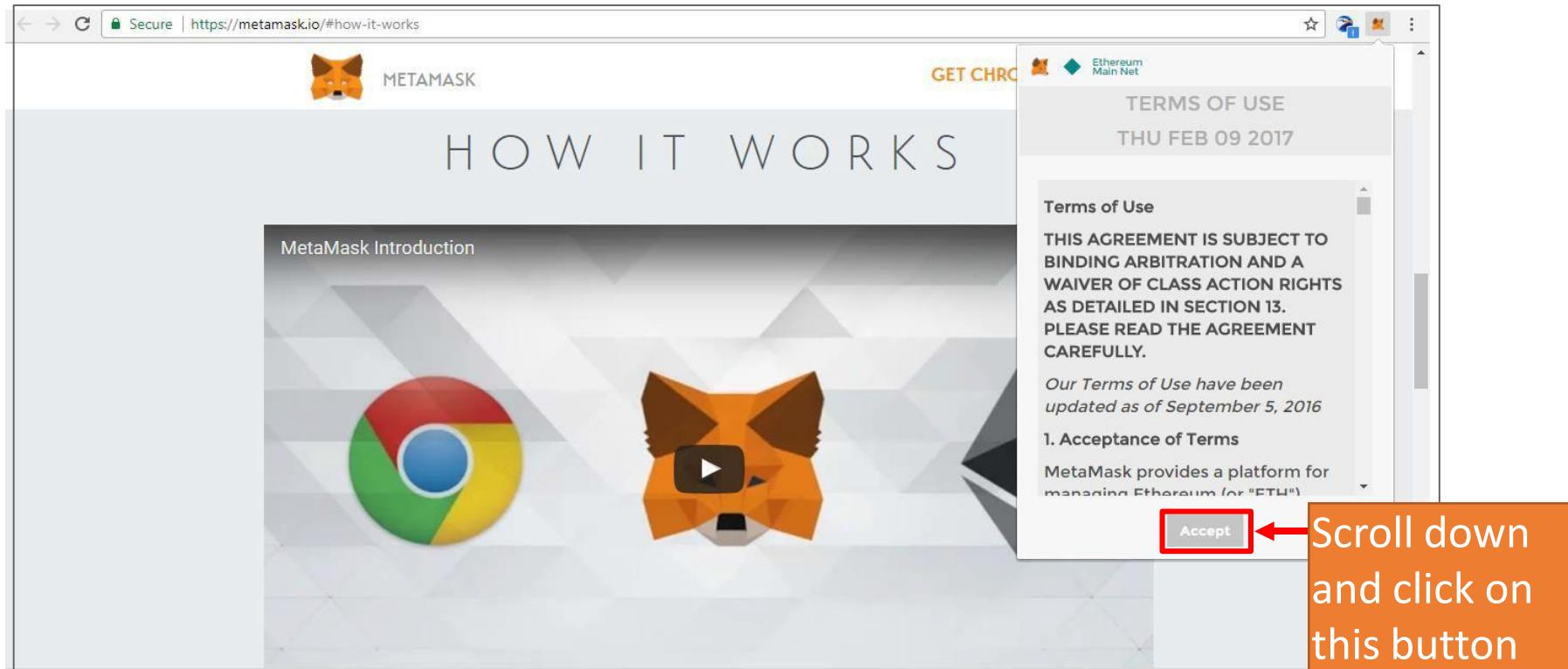
MetaMask extension gets added to the Chrome

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



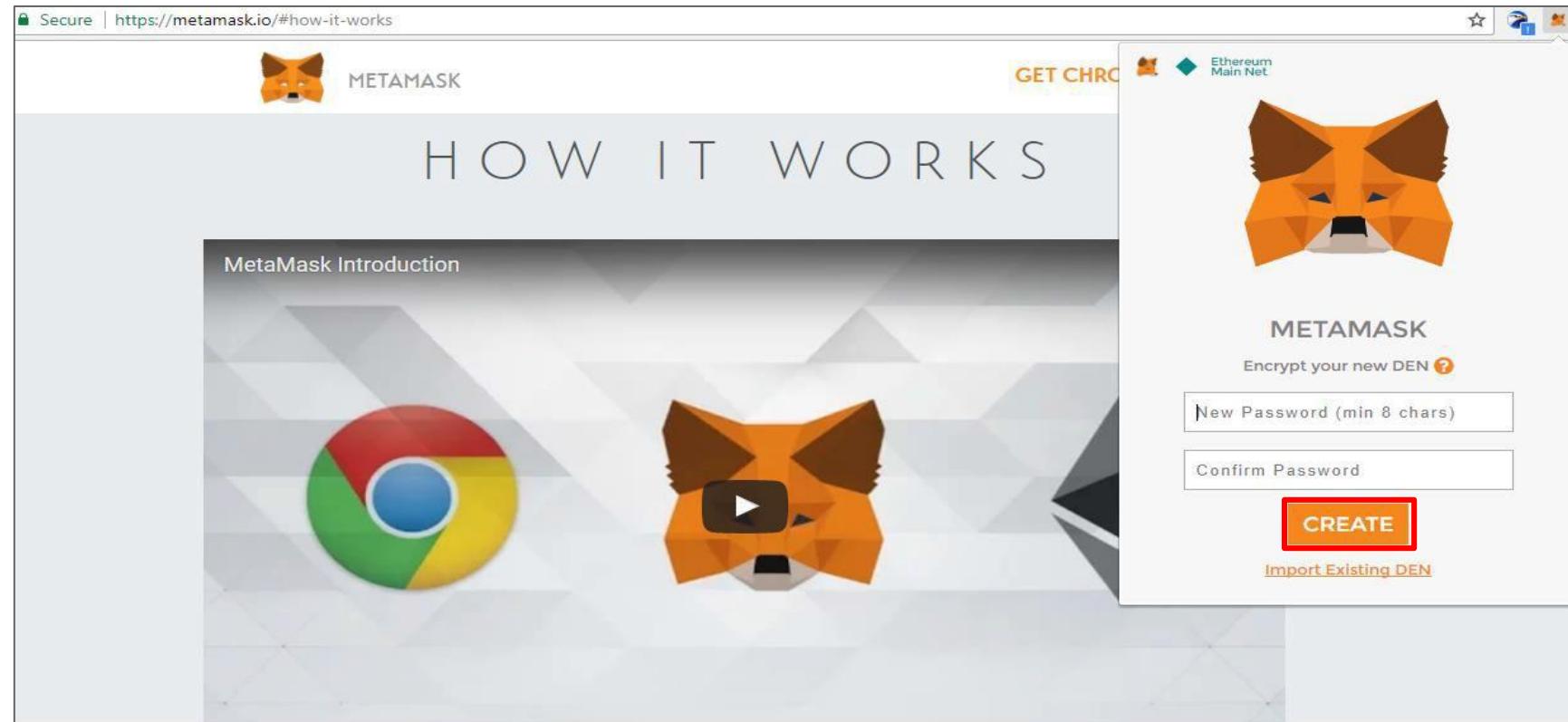
Accept “terms of use”

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



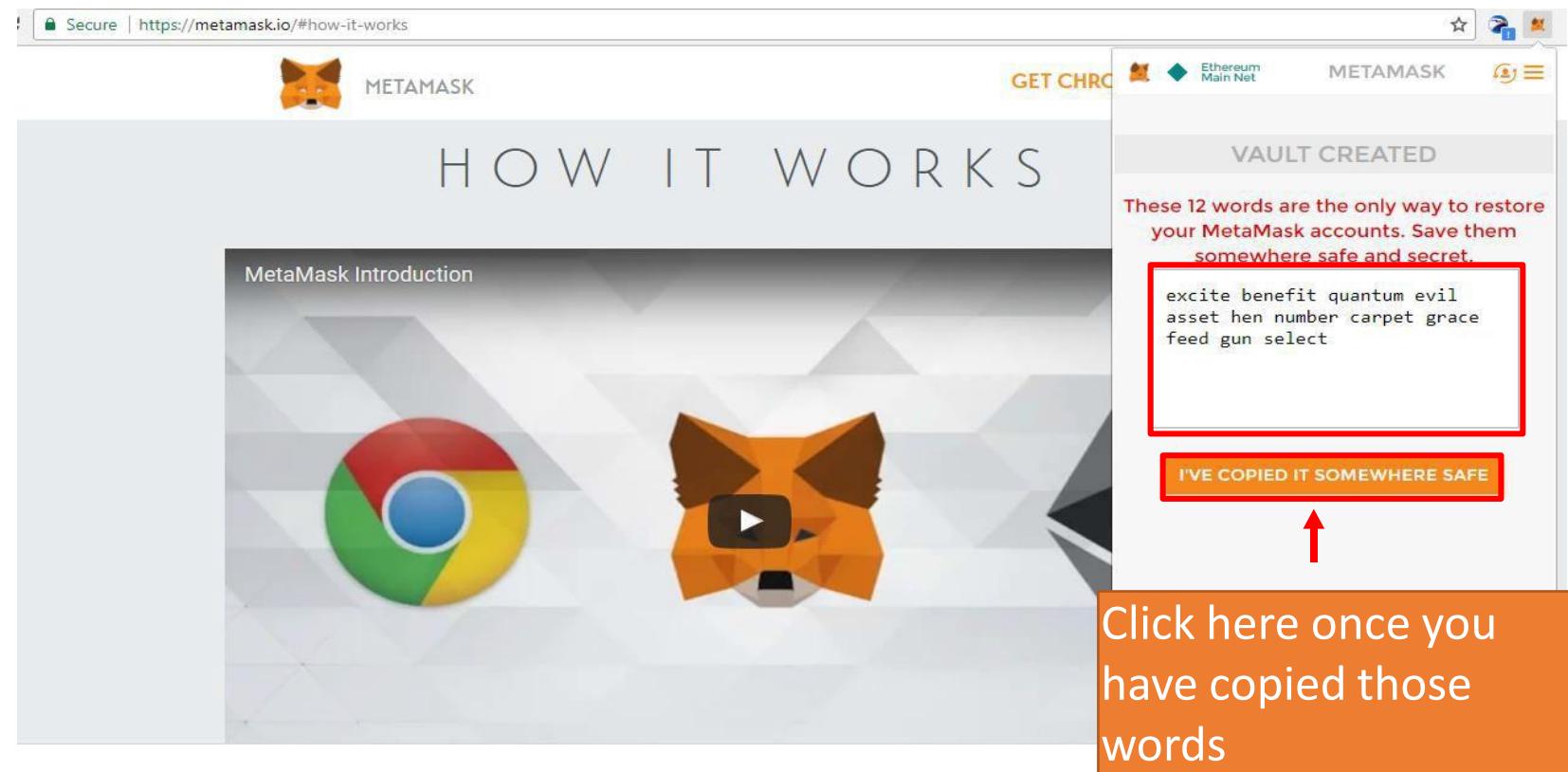
Create Password

- Now, create and confirm the password



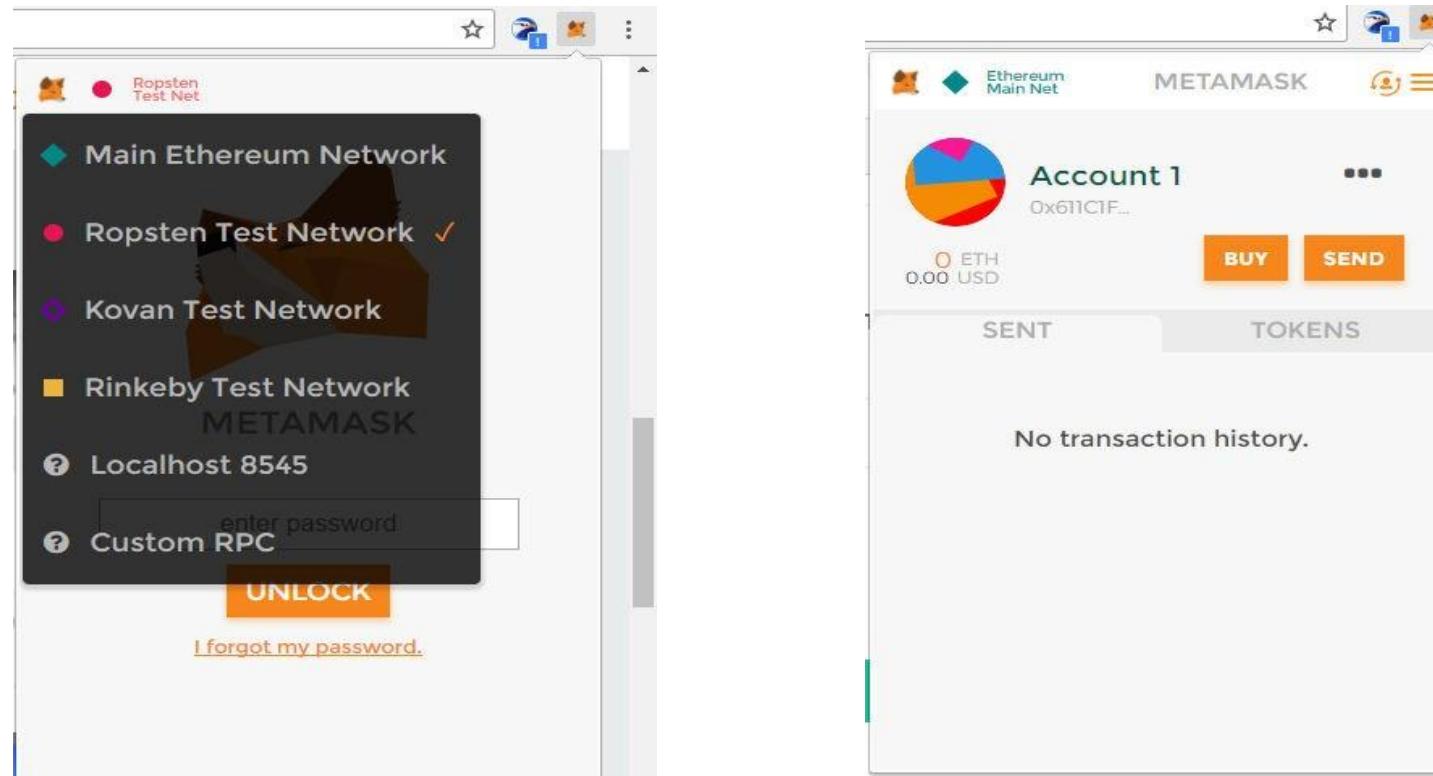
Vault created

- 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

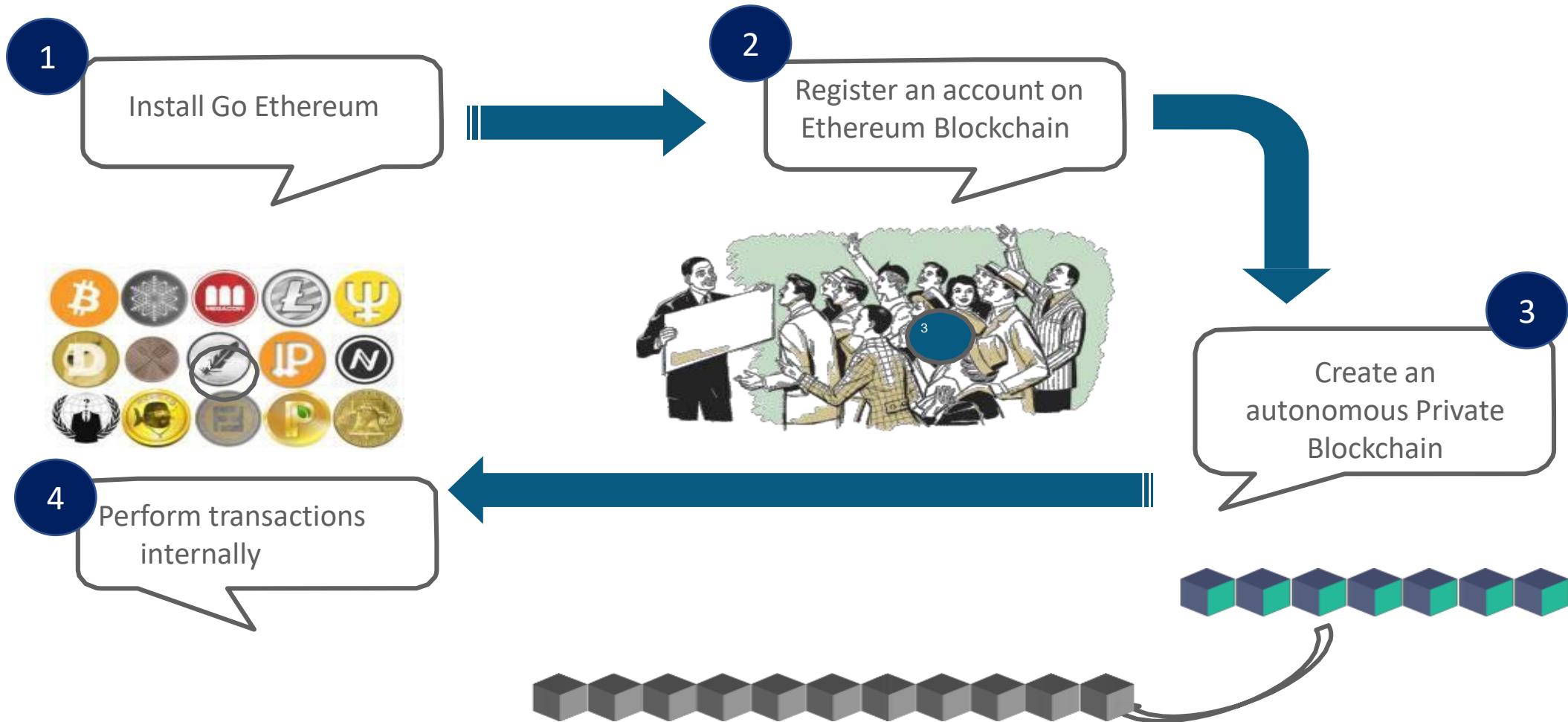


“

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

”

Developing Private Blockchain



Installing Blockchain



Cloning Geth Code

Creating a Genesis Block

Validating & Mining Ether

Making Rules for our
Blockchain

Download Ethereum code and compile it

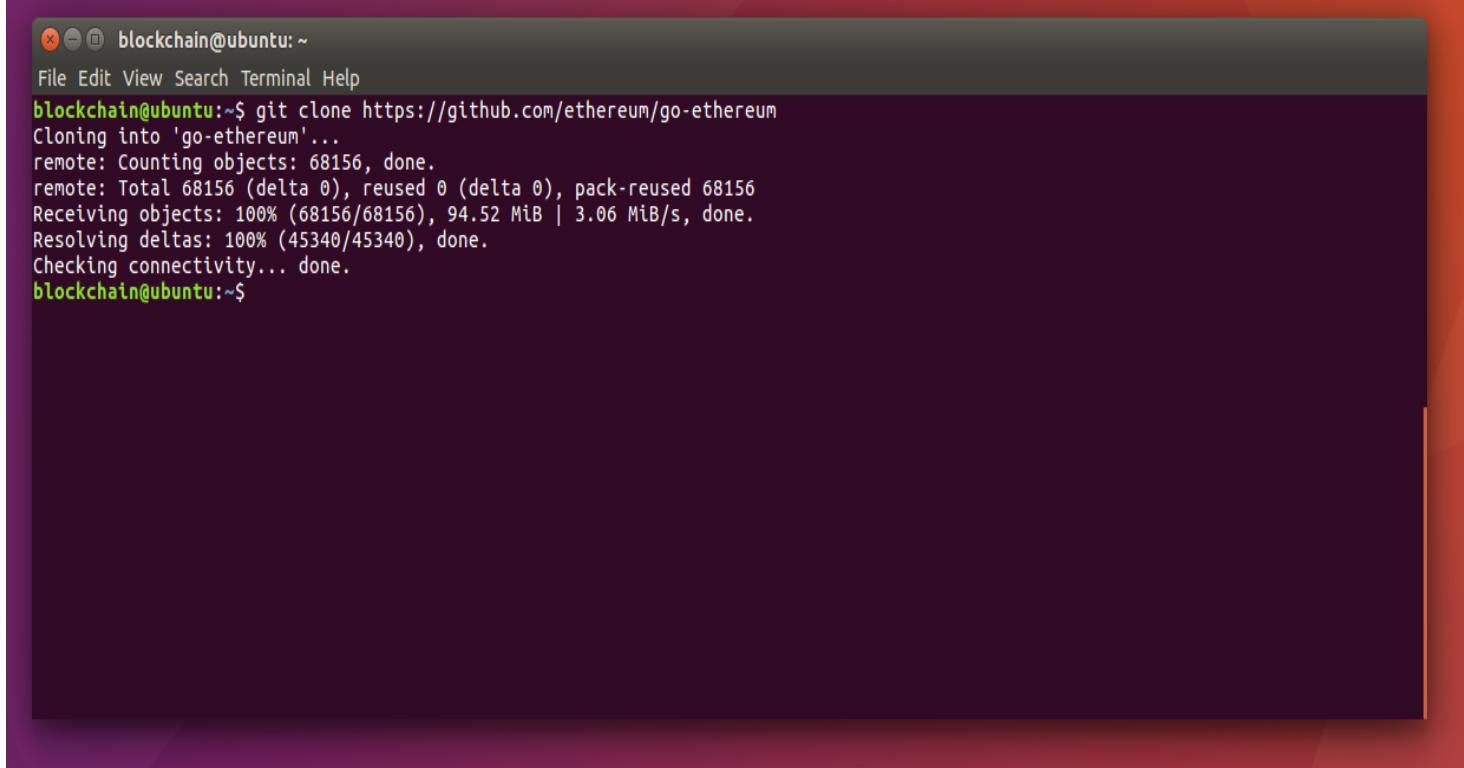
```
sudo apt install git // installing git in case not  
installed already  
sudo apt -get update  
sudo apt -get upgrade  
  
git clone https://github.com/ethereum/go-ethereum cd  
go-ethereum  
git tag  
git checkout tags/v1.6.7 -b EdurekaEthereumv1.6.7  
  
//Checking the branch git  
branch  
  
//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
```

A screenshot of a terminal window titled "blockchain@ubuntu:~". The window shows the output of a "git clone" command. The command is "git clone https://github.com/ethereum/go-ethereum". The output indicates that the cloning process is into a directory named "go-ethereum". It shows the progress of counting objects (68156), receiving objects (100% complete at 94.52 MiB), resolving deltas (100% complete at 45340), and checking connectivity, which is also done.

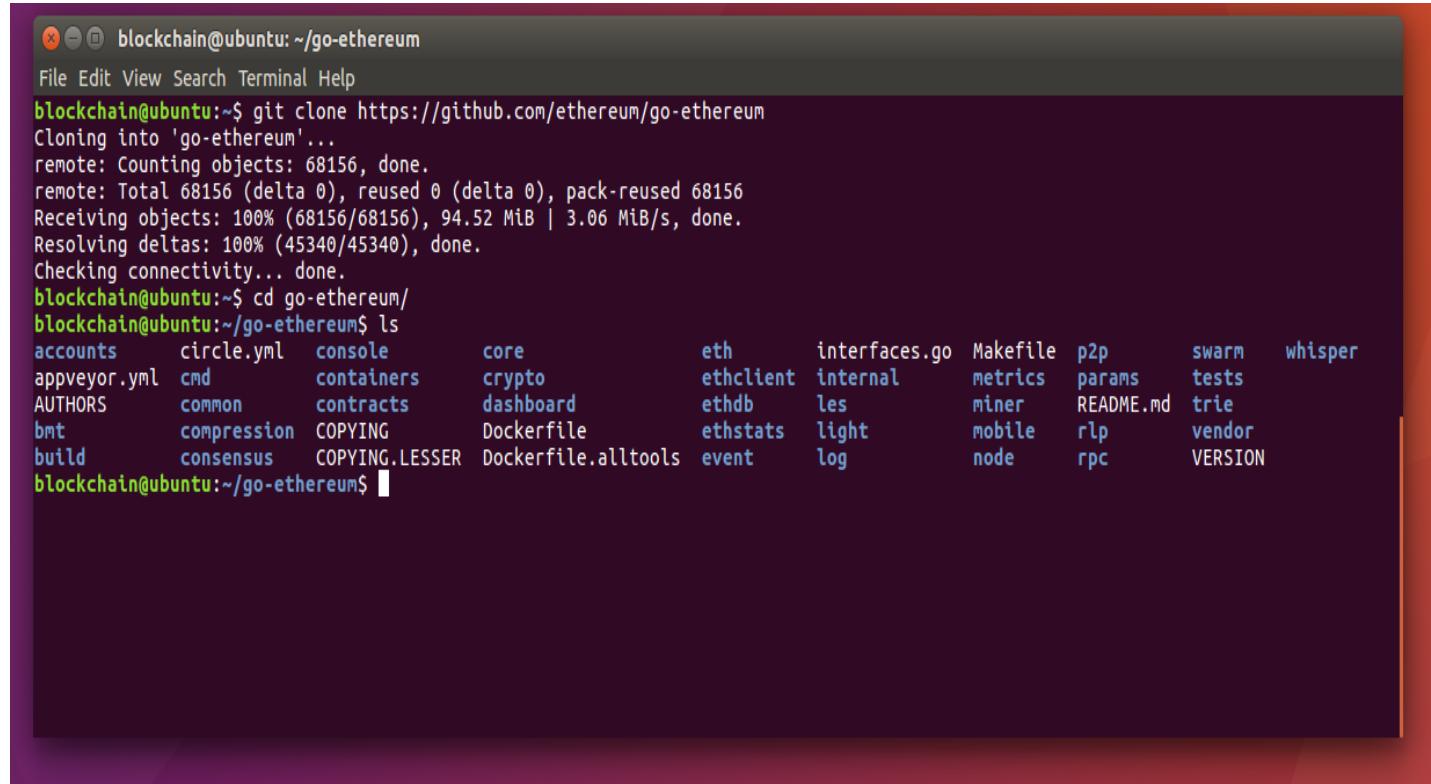
Note: Cloning of Geth will take some time so please be patient

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
```



```
blockchain@ubuntu:~/go-ethereum
File Edit View Search Terminal Help
blockchain@ubuntu:~$ git clone https://github.com/ethereum/go-ethereum
Cloning into 'go-ethereum'...
remote: Counting objects: 68156, done.
remote: Total 68156 (delta 0), reused 0 (delta 0), pack-reused 68156
Receiving objects: 100% (68156/68156), 94.52 MiB | 3.06 MiB/s, done.
Resolving deltas: 100% (45340/45340), done.
Checking connectivity... done.
blockchain@ubuntu:~$ cd go-ethereum/
blockchain@ubuntu:~/go-ethereum$ ls
accounts      circle.yml    console      core          eth        interfaces.go  Makefile   p2p        swarm      whisper
appveyor.yml  cmd          containers   crypto       ethclient  internal     metrics   params    tests
AUTHORS       common        contracts   dashboard   ethdb      les          miner    README.md  trie
bmt          compression  COPYING     Dockerfile  ethstats   light       mobile   rlp        vendor
build         consensus   COPYING.LESSER Dockerfile.alltools event      log          node    rpc        VERSION
blockchain@ubuntu:~/go-ethereum$
```



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

”

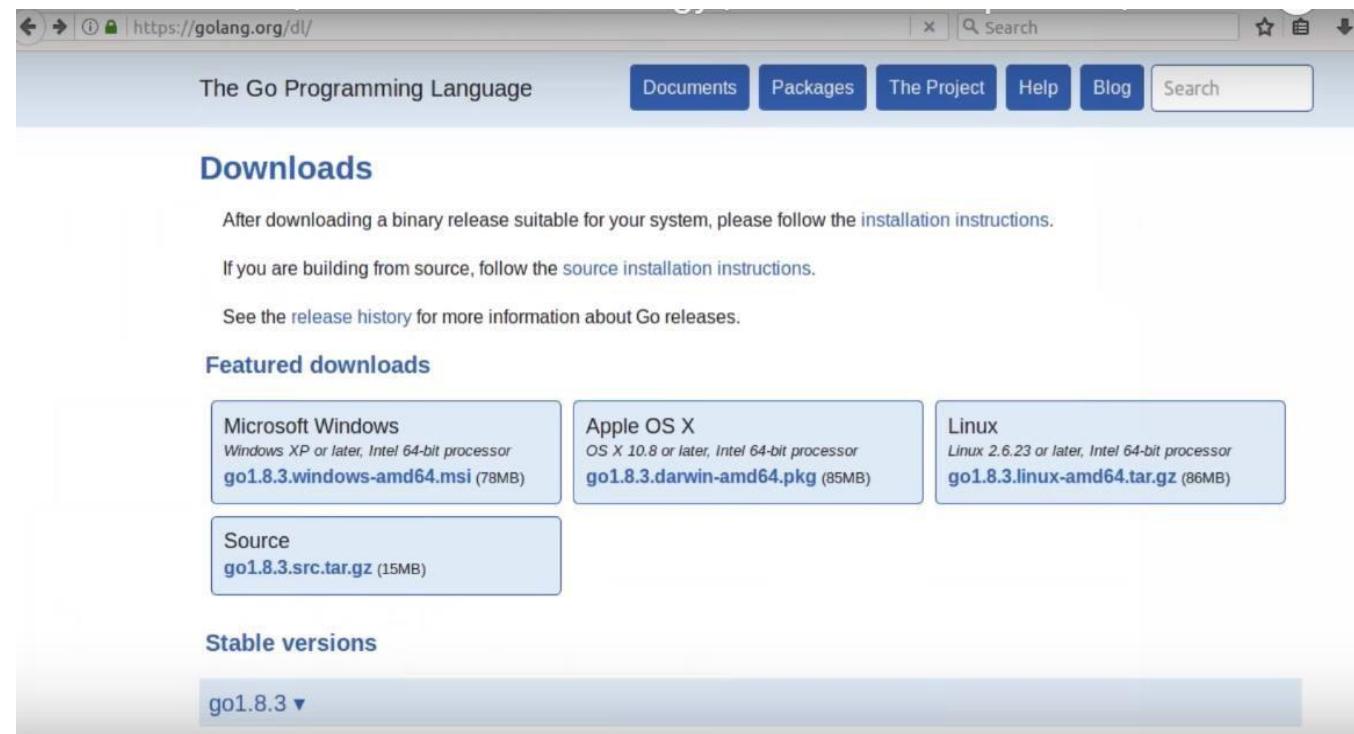
Go Lang Installation

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

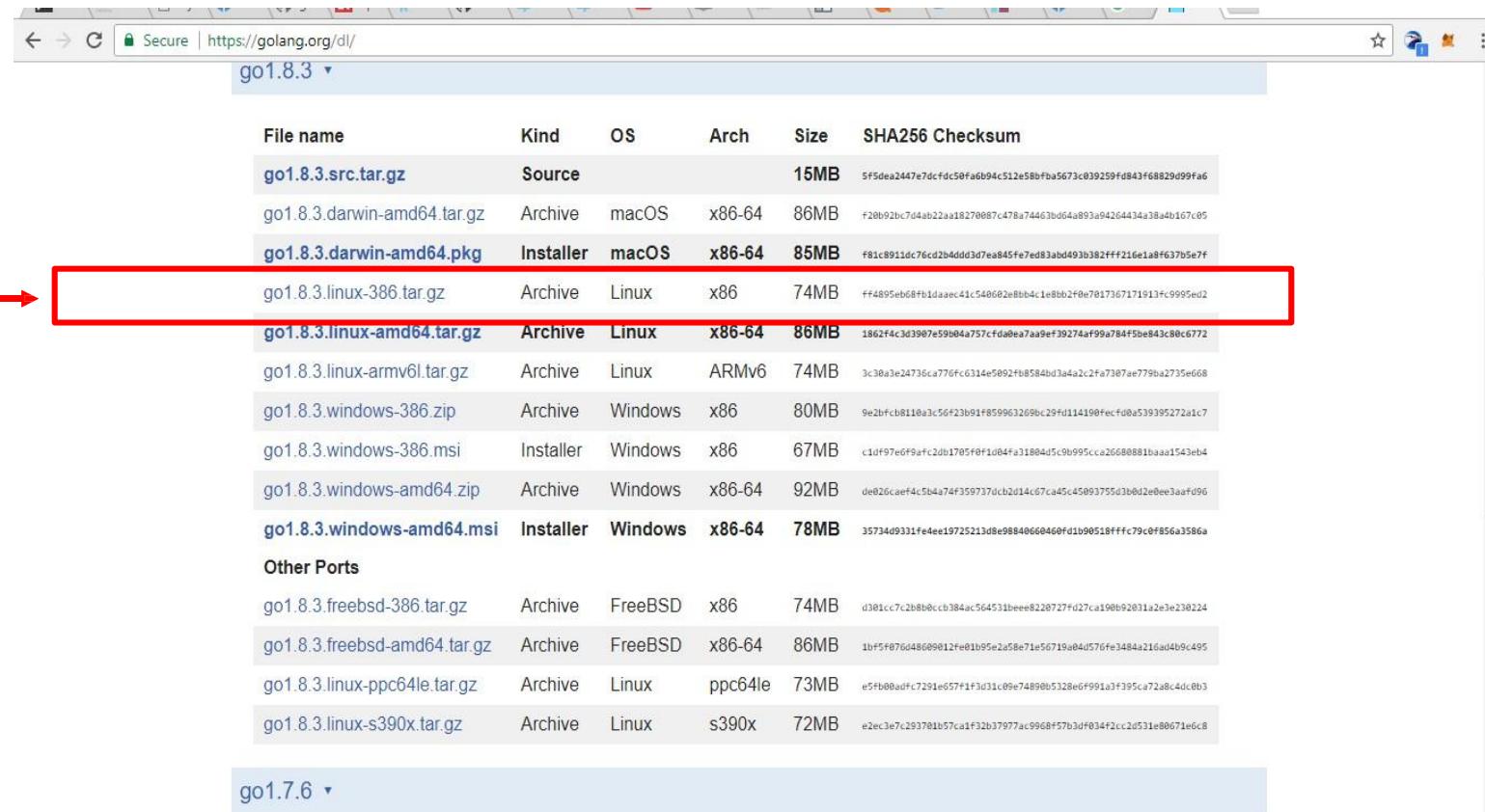


The screenshot shows the official Go Programming Language website at <https://golang.org/dl/>. The page has a header with links for Documents, Packages, The Project, Help, Blog, and Search. Below the header, there's a section titled "Downloads" with instructions for installing a binary release. It includes links for building from source and viewing the release history. A "Featured downloads" section shows packages for Microsoft Windows, Apple OS X, and Linux, each with a link to download the file. At the bottom, there's a "Stable versions" section with a dropdown menu currently set to "go1.8.3".

Various Packages in Go-lang

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

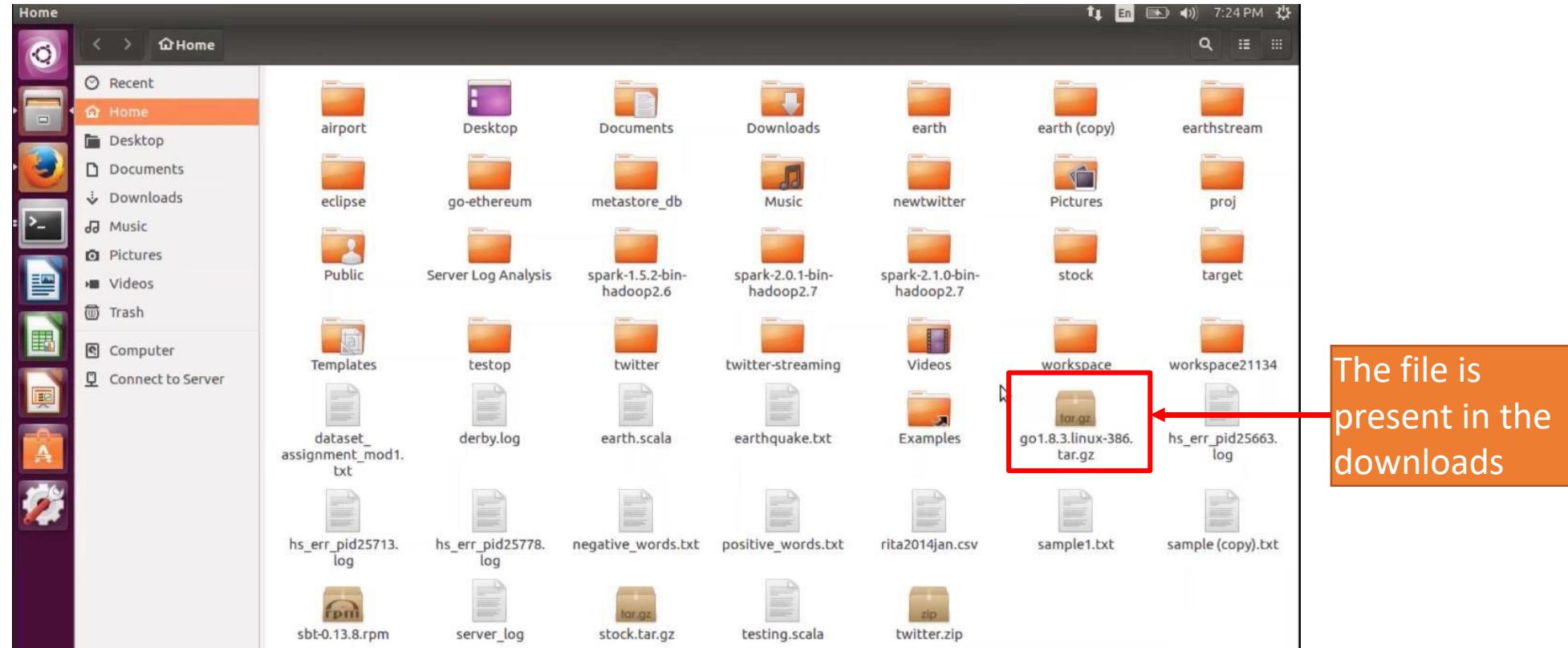
I am using 32 bit version of Go-lang



File name	Kind	OS	Arch	Size	SHA256 Checksum
go1.8.3.src.tar.gz	Source			15MB	5f5dea2447e7dcfdc50fa6b94c512e58bfb5673c039259fd843f68829d99fa6
go1.8.3.darwin-amd64.tar.gz	Archive	macOS	x86-64	86MB	f20t92bc7d4ab22aa18270087c478a74463bd64a893a94264434a38a4b107c05
go1.8.3.darwin-amd64.pkg	Installer	macOS	x86-64	85MB	f81c8911dc76cd2b4ddd3d7ea845fe7ed3abd493b382ffff216e1a8f637b5e7f
go1.8.3.linux-386.tar.gz	Archive	Linux	x86	74MB	ff4895eb68fb1daaec41c548602eabb4c1e8bb2f0e7817367171913fc9995ed2
go1.8.3.linux-amd64.tar.gz	Archive	Linux	x86-64	86MB	1862f4c3d3907e59904a757cfda0ea7aaef39274af99a784f50e843c80c6772
go1.8.3.linux-armv6l.tar.gz	Archive	Linux	ARMv6	74MB	3c30a3e24736c4776fc6314e5092fb8584bd38a2c2fa7307ae779ba2735e668
go1.8.3.windows-386.zip	Archive	Windows	x86	80MB	9e2bfc8110a3c56f23b91f859963269b2c99fd114190fecfd0a539395272a1c7
go1.8.3.windows-386.msi	Installer	Windows	x86	67MB	c1df97e6f9a4fc2db1705f0f1d04fa11804d5c9b995cca2668881baaa1543eb4
go1.8.3.windows-amd64.zip	Archive	Windows	x86-64	92MB	de026caeef4c5b4a74f359737dc2d14c67ca45c45093755d3b0d2e0ee3aaef096
go1.8.3.windows-amd64.msi	Installer	Windows	x86-64	78MB	35734d9331fe4ee1972521d0e98840660460fd1b90518ffffc79c0f856a3586a
Other Ports					
go1.8.3.freebsd-386.tar.gz	Archive	FreeBSD	x86	74MB	d301cc7c2b8b0ccb384ac564531beee8220727fd27ca190b92031a2e3e230224
go1.8.3.freebsd-amd64.tar.gz	Archive	FreeBSD	x86-64	86MB	1bf5f076d48609012fe01b95e2a58e71e56719a04d576fe3484a216ad4b9c495
go1.8.3.linux-ppc64le.tar.gz	Archive	Linux	ppc64le	73MB	e5fb00adfc7291e657f1f3d1c09e74890b5328e6f991a3f395ca72a84dc0b3
go1.8.3.linux-s390x.tar.gz	Archive	Linux	s390x	72MB	e2ec3e7c293701b57ca1f32b37977ac9908f57b3df034f2cc2d531e00671e6c8

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

```
  bmt      compression  COPYING      Dockerfile      ethstats    light      mobile     rlp      vendor  
  build    consensus   COPYING.LESSER Dockerfile.alltools event      log       node      rpc      VERSION  
blockchain@ubuntu:~/go-ethereum$ cd ..  
blockchain@ubuntu:~$ cd Do  
Documents/ Downloads/  
blockchain@ubuntu:~$ cd Downloads/  
blockchain@ubuntu:~/Downloads$ ls  
go1.8.3.linux-amd64.tar.gz  
blockchain@ubuntu:~/Downloads$ sudo tar -xvf go1.8.3.linux-amd64.tar.gz
```

Go lang gets extracted and installed

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

```
File Edit View Search Terminal Help
go/test/syntax/typesw.go
go/test/syntax/vareq.go
go/test/syntax/vareq1.go
go/test/tinyfin.go
go/test/torture.go
go/test/turing.go
go/test/typecheck.go
go/test/typecheckloop.go
go/test/typeswitch.go
go/test/typeswitch1.go
go/test/typeswitch2.go
go/test/typeswitch3.go
go/test/uintptrescapes.dir/
go/test/uintptrescapes.dir/a.go
go/test/uintptrescapes.dir/main.go
go/test/uintptrescapes.go
go/test/uintptrescapes2.go
go/test/undef.go
go/test/utf.go
go/test/varerr.go
go/test/varinit.go
go/test/writebarrier.go
go/test/zerodivide.go
blockchain@ubuntu:~/Downloads$
```

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

```
go/test/undef.go  
go/test/utf.go  
go/test/varerr.go  
go/test/varinit.go  
go/test/writebarrier.go  
go/test/zerodivide.go  
blockchain@ubuntu:~/Downloads$ ls  
go go1.8.3.linux-amd64.tar.gz  
blockchain@ubuntu:~/Downloads$ sudo mv go  
go/ go1.8.3.linux-amd64.tar.gz  
blockchain@ubuntu:~/Downloads$ sudo mv go /usr/local
```

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

```
go/test/uintptrescapes.go
go/test/uintptrescapes2.go
go/test/undef.go
go/test/utf.go
go/test/varerr.go
go/test/varinit.go
go/test/writebarrier.go
go/test/zerodivide.go
blockchain@ubuntu:~/Downloads$ ls
go go1.8.3.linux-amd64.tar.gz
blockchain@ubuntu:~/Downloads$ sudo mv go
go/ go1.8.3.linux-amd64.tar.gz
blockchain@ubuntu:~/Downloads$ cd /usr/local
blockchain@ubuntu:/usr/local$ ls
bin etc games go include lib man sbin share src
blockchain@ubuntu:/usr/local$
```

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

```
go/test/varerr.go
go/test/varinit.go
go/test/writebarrier.go
go/test/zerodivide.go
blockchain@ubuntu:~/Downloads$ ls
go go1.8.3.linux-amd64.tar.gz
blockchain@ubuntu:~/Downloads$ sudo mv go
go/                               go1.8.3.linux-amd64.tar.gz
blockchain@ubuntu:~/Downloads$ cd /usr/local
blockchain@ubuntu:/usr/local$ ls
bin  etc  games  go  include  lib  man  sbin  share  src
blockchain@ubuntu:/usr/local$ cd /home/blockchain/
blockchain@ubuntu:~$ sudo gedit .bashrc
```

Adding Environment Variables to Variables

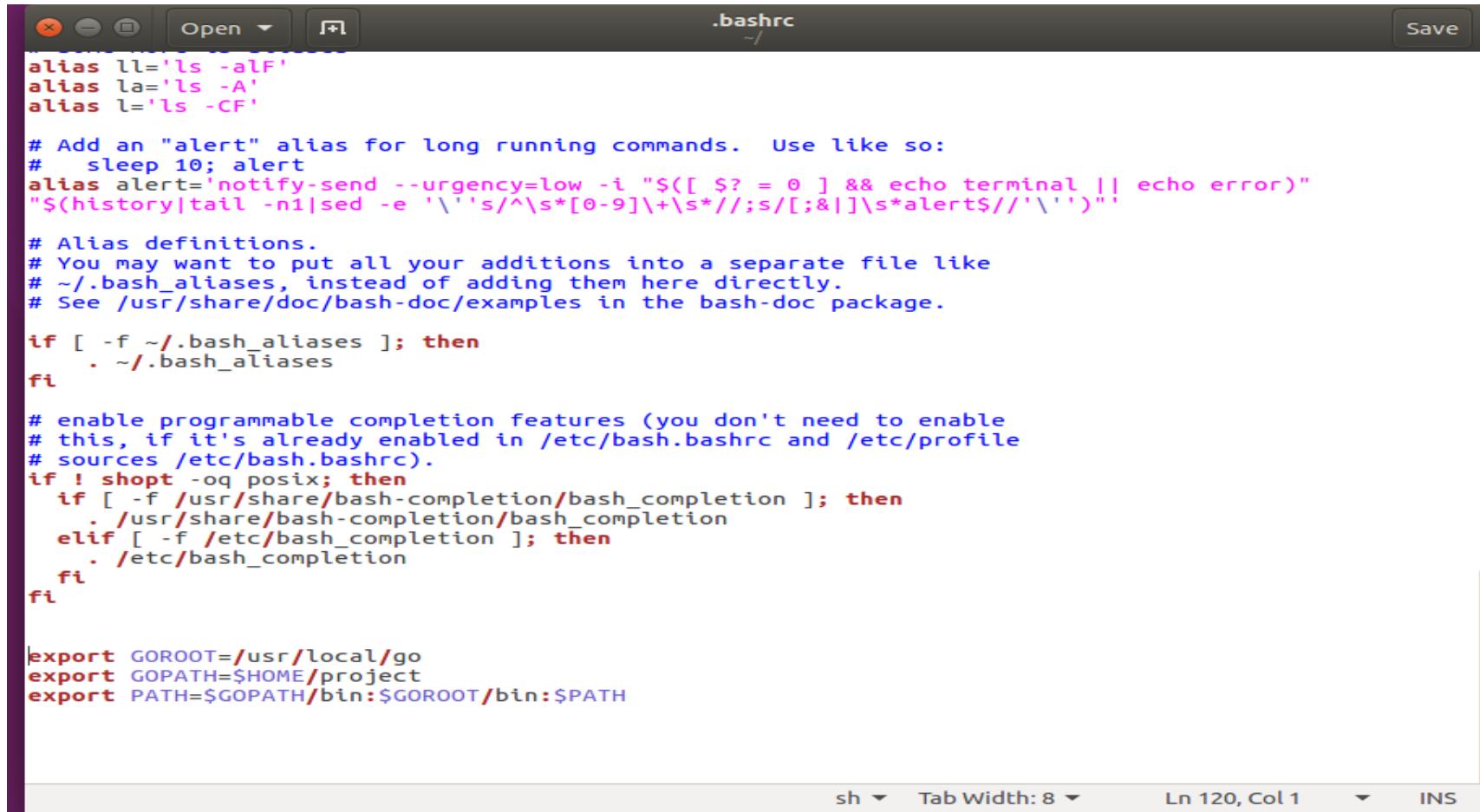
Here three lines you need to add

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



A screenshot of a terminal window titled ".bashrc". The window contains the following code:

```
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

# Add an "alert" alias for long running commands.  Use like so:
#   sleep 10; alert
alias alert='notify-send --urgency=low -i "$( [ $? = 0 ] && echo terminal || echo error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;s/[;&]\s*alert$/'\''")"

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

export GOROOT=/usr/local/go
export GOPATH=$HOME/project
export PATH=$GOPATH/bin:$GOROOT/bin:$PATH
```

The terminal window also shows status information at the bottom: "sh", "Tab Width: 8", "Ln 120, Col 1", and "INS".

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

```
go/test/_test/_test.go
blockchain@ubuntu:~/Downloads$ ls
go go1.8.3.linux-amd64.tar.gz
blockchain@ubuntu:~/Downloads$ sudo mv go
go/ go1.8.3.linux-amd64.tar.gz
blockchain@ubuntu:~/Downloads$ cd /usr/local
blockchain@ubuntu:/usr/local$ ls
bin etc games go include lib man sbin share src
blockchain@ubuntu:/usr/local$ cd /home/blockchain/
blockchain@ubuntu:~$ sudo gedit .bashrc

(gedit:61843): IBUS-WARNING **: The owner of /home/blockchain/.bashrc has changed, but it hasn't been relaunched yet.
** (gedit:61843): WARNING **: Set document metadata failed:
blockchain@ubuntu:~$ source .bashrc
blockchain@ubuntu:~$
```

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

```
go go1.8.3.linux-amd64.tar.gz
blockchain@ubuntu:~/Downloads$ sudo mv go
go/ go1.8.3.linux-amd64.tar.gz
blockchain@ubuntu:~/Downloads$ cd /usr/local
blockchain@ubuntu:/usr/local$ ls
bin etc games go include lib man sbin share src
blockchain@ubuntu:/usr/local$ cd /home/blockchain/
blockchain@ubuntu:~$ sudo gedit .bashrc

(gedit:61843): IBUS-WARNING **: The owner of /home/block
** (gedit:61843): WARNING **: Set document metadata failed
blockchain@ubuntu:~$ source .bashrc
blockchain@ubuntu:~$ go version
go version go1.8.3 linux/amd64
blockchain@ubuntu:~$
```



“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

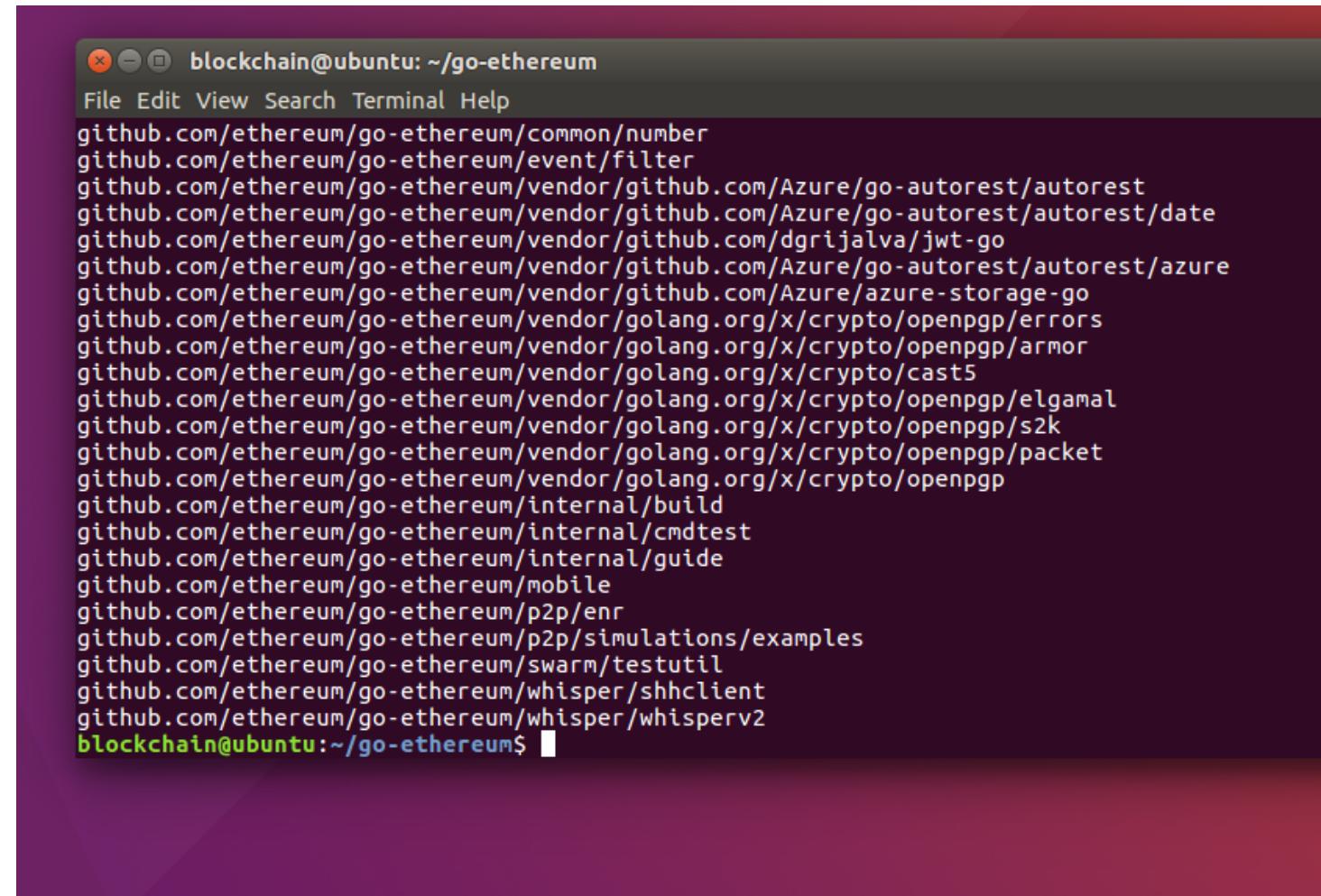
make all

```
(gedit:61843): IBUS-WARNING **: The owner of /home/blockchain/.config/ibus/gedit/61843/ibus-bridge-gtk-1 was denied execute permission on the file /home/blockchain/.config/ibus/gedit/61843/ibus-bridge-gtk-1
** (gedit:61843): WARNING **: Set document metadata failed: Setting attribute 'application/x-go-ethereum' failed
blockchain@ubuntu:~$ source .bashrc
blockchain@ubuntu:~$ go version
go version go1.8.3 linux/amd64
blockchain@ubuntu:~$ cd go-ethereum/
blockchain@ubuntu:~/go-ethereum$ make all
build/env.sh go run build/ci.go install
>>> /usr/local/go/bin/go install -ldflags -X main.gitCommit=9fd76e33af36
ereum github.com/ethereum/go-ethereum/accounts github.com/ethereum/go-eth
ts/abi/bind github.com/ethereum/go-ethereum/accounts/abi/bind/backends g
m/ethereum/go-ethereum/accounts/usbwallet github.com/ethereum/go-ethereu
o-ethereum/bmt github.com/ethereum/go-ethereum/cmd/abigen github.com/eth
reum/cmd/ethkey github.com/ethereum/go-ethereum/cmd/evm github.com/ether
eum/go-ethereum/cmd/faucet github.com/ethereum/go-ethereum/cmd/geth gith
m/ethereum/go-ethereum/cmd/p2psim github.com/ethereum/go-ethereum/cmd/pu
```

“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



A screenshot of a terminal window titled "blockchain@ubuntu: ~/go-ethereum". The window shows a list of GitHub repository URLs being cloned or checked out. The list includes:

- github.com/ethereum/go-ethereum/common/number
- github.com/ethereum/go-ethereum/event/filter
- github.com/ethereum/go-ethereum/vendor/github.com/Azure/go-autorest/autorest
- github.com/ethereum/go-ethereum/vendor/github.com/Azure/go-autorest/autorest/date
- github.com/ethereum/go-ethereum/vendor/github.com/dgrijalva/jwt-go
- github.com/ethereum/go-ethereum/vendor/github.com/Azure/go-autorest/autorest/azure
- github.com/ethereum/go-ethereum/vendor/github.com/Azure/azure-storage-go
- github.com/ethereum/go-ethereum/vendor/golang.org/x/crypto/openpgp/errors
- github.com/ethereum/go-ethereum/vendor/golang.org/x/crypto/openpgp/armor
- github.com/ethereum/go-ethereum/vendor/golang.org/x/crypto/cast5
- github.com/ethereum/go-ethereum/vendor/golang.org/x/crypto/openpgp/elgamal
- github.com/ethereum/go-ethereum/vendor/golang.org/x/crypto/openpgp/s2k
- github.com/ethereum/go-ethereum/vendor/golang.org/x/crypto/openpgp/packet
- github.com/ethereum/go-ethereum/vendor/golang.org/x/crypto/openpgp
- github.com/ethereum/go-ethereum/internal/build
- github.com/ethereum/go-ethereum/internal/cmdtest
- github.com/ethereum/go-ethereum/internal/guide
- github.com/ethereum/go-ethereum/mobile
- github.com/ethereum/go-ethereum/p2p/enr
- github.com/ethereum/go-ethereum/p2p/simulations/examples
- github.com/ethereum/go-ethereum/swarm/testutil
- github.com/ethereum/go-ethereum/whisper/shhclient
- github.com/ethereum/go-ethereum/whisper/whisperv2

The terminal prompt at the bottom is "blockchain@ubuntu:~/go-ethereum\$".

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

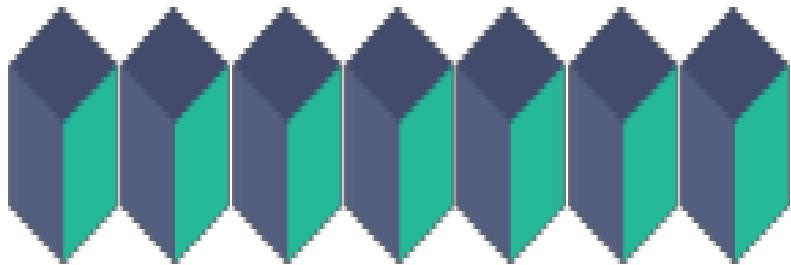
Creating a Genesis Block

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
```

```
(gedit:63047): IBUS-WARNING **: The owner of /home/blockchain/.config/ib
** (gedit:63047): WARNING **: Set document metadata failed: Setting attr
** (gedit:63047): WARNING **: Set document metadata failed: Setting attr
** (gedit:63047): WARNING **: Set document metadata failed: Setting attr
blockchain@ubuntu:~$ source .bashrc
blockchain@ubuntu:~$ go version
go version go1.8.3 linux/amd64
blockchain@ubuntu:~$ cd go-ethereum/
blockchain@ubuntu:~/go-ethereum$ mkdir genesis
blockchain@ubuntu:~/go-ethereum$ cd genesis/
blockchain@ubuntu:~/go-ethereum/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.

```
(gedit:63047): IBUS-WARNING **: The owner of /home/blockchain/.config/ibus/  
** (gedit:63047): WARNING **: Set document metadata failed: Setting attribu  
** (gedit:63047): WARNING **: Set document metadata failed: Setting attribu  
** (gedit:63047): WARNING **: Set document metadata failed: Setting attribu  
blockchain@ubuntu:~$ source .bashrc  
blockchain@ubuntu:~$ go version  
go version go1.8.3 linux/amd64  
blockchain@ubuntu:~$ cd go-ethereum/  
blockchain@ubuntu:~/go-ethereum$ mkdir genesis  
blockchain@ubuntu:~/go-ethereum$ cd genesis/  
blockchain@ubuntu:~/go-ethereum/genesis$ sudo gedit genesis1.json
```

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 eternity

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

--datadir
~/ethereum/net3

--datadir
~/ethereum/net3/

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
[02-17|20:08:53] Maximum peer count
[02-17|20:08:53] Allocated cache and file handles
[02-17|20:08:54] Writing custom genesis block
[02-17|20:08:54] Persisted trie from memory database
[livesize=0.00B]
[02-17|20:08:54] Successfully wrote genesis state
[02-17|20:08:54] Allocated cache and file handles
[16]
[02-17|20:08:54] Writing custom genesis block
[02-17|20:08:54] Persisted trie from memory database
[livesize=0.00B]
[02-17|20:08:54] Successfully wrote genesis state
[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
```

```
(gedit:63152): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided by any .service files

** (gedit:63152): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-spell-enabled not supported
** (gedit:63152): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-encoding not supported
** (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
INFO [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
INFO [02-17|20:08:54] Allocated cache and file handles                      database=chaindata                                         hash=5e1fc7..d790e0
INFO [02-17|20:08:54] Persisted trie from memory database                   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..d790e0
0
blockchain@ubuntu:~/go-ethereum/genesis$ /home/blockchain/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
INFO [02-17|20:13:50] Starting peer-to-peer node
INFO [02-17|20:13:50] Allocated cache and file handles
WARN [02-17|20:13:50] Upgrading database to use lookup entries
INFO [02-17|20:13:50] Initialised chain configuration
IP155: 0 EIP158: 0 Byzantium: <nil> Engine: unknown"
INFO [02-17|20:13:50] Disk storage enabled for ethash caches
INFO [02-17|20:13:50] Disk storage enabled for ethash DAGs
INFO [02-17|20:13:50] Initialising Ethereum protocol
INFO [02-17|20:13:50] Loaded most recent local header
INFO [02-17|20:13:50] Loaded most recent local full block
INFO [02-17|20:13:50] Loaded most recent local fast block
INFO [02-17|20:13:50] Regenerated local transaction journal
INFO [02-17|20:13:50] Database deduplication successful
INFO [02-17|20:13:50] Starting P2P networking
INFO [02-17|20:13:52] UDP listener up
69f85bc581686c718875369319cd47332f5e2bd48d9400ea5c87493fdcd6d81f@[::]:30303
INFO [02-17|20:13:52] RLPx listener up
69f85bc581686c718875369319cd47332f5e2bd48d9400ea5c87493fdcd6d81f@[::]:30303
gs INFO [02-17|20:13:52] IPC endpoint opened
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

```
[root@ip-172-20-1-52 ~]# geth --testnet
```

```
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
```

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0xfbfb220770c74b87c5a89c83d03335ec0d9885f33"
>
```

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

```
INFO [02-17|20:13:50] Regenerated local transaction journal
INFO [02-17|20:13:50] Database deduplication successful
INFO [02-17|20:13:50] Starting P2P networking
INFO [02-17|20:13:52] UDP listener up
69f85bc581686c718875369319cd47332f5e2bd48d9400ea5c87493
INFO [02-17|20:13:52] RLPx listener up
69f85bc581686c718875369319cd47332f5e2bd48d9400ea5c87493
INFO [02-17|20:13:52] IPC endpoint opened
Welcome to the Geth JavaScript console!

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

> personal.newAccount()
Passphrase:
Repeat passphrase:
"0xfbfb220770c74b87c5a89c83d03335ec0d9885f33"
> eth.accounts
[ "0xfbfb220770c74b87c5a89c83d03335ec0d9885f33" ]
>
```

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()
```

```
0
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x10173ceb0c7c55e61a6b432b75e40d2b88bdb519"
> eth.accounts()
TypeError: 'accounts' is not a function
    at <anonymous>:1:1

> eth.accounts
[ "0xfbfb220770c74b87c5a89c83d03335ec0d9885f33", "0x10173
> ^C
> eth.blockNumber
0
>
```

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()
```

```
[ 0x1bd1220770c74d87c5d89c85005555ec0d9885155 , 0x10173ce00c7cc
> ^C
> eth.blockNumber
0
> miner.start()
INFO [02-17|20:25:50] Updated mining threads
INFO [02-17|20:25:50] Transaction pool price threshold updated
INFO [02-17|20:25:50] Etherbase automatically configured
null
> INFO [02-17|20:25:50] Starting mining operation
INFO [02-17|20:25:50] Commit new mining work
INFO [02-17|20:25:53] Generating ethash verification cache
```

Mining process initiates and it starts allocating the resources

Mining starts

```
blockchain@ubuntu: ~/go-ethereum/genesis
File Edit View Search Terminal Help

> eth.accounts
["0xfbfb220770c74b87c5a89c83d03335ec0d9885f33", "0x10173ceb0c7c55e61a6b432b75e40d2b88bdb519"]
> ^C
> eth.blockNumber
0
> miner.start()
INFO [02-17|20:25:50] Updated mining threads
INFO [02-17|20:25:50] Transaction pool price threshold updated
INFO [02-17|20:25:50] Etherbase automatically configured
null
> INFO [02-17|20:25:50] Starting mining operation
INFO [02-17|20:25:50] Commit new mining work
INFO [02-17|20:25:53] Generating ethash verification cache
INFO [02-17|20:25:54] Generated ethash verification cache
INFO [02-17|20:26:05] Generating DAG in progress
INFO [02-17|20:26:16] Generating DAG in progress
INFO [02-17|20:26:26] Generating DAG in progress
INFO [02-17|20:26:36] Generating DAG in progress
INFO [02-17|20:26:46] Generating DAG in progress
INFO [02-17|20:26:56] Generating DAG in progress
INFO [02-17|20:27:06] Generating DAG in progress
INFO [02-17|20:27:16] Generating DAG in progress
INFO [02-17|20:27:26] Generating DAG in progress
INFO [02-17|20:27:35] Generating DAG in progress
INFO [02-17|20:27:45] Generating DAG in progress
INFO [02-17|20:27:56] Generating DAG in progress
INFO [02-17|20:28:08] Generating DAG in progress
INFO [02-17|20:28:21] Generating DAG in progress
INFO [02-17|20:25:50] threads=0
INFO [02-17|20:25:50] price=18000000000
INFO [02-17|20:25:50] address=0xfbfb220770C74b87C5a89c83D03335EC0d9885F33
INFO [02-17|20:25:50] number=1 txs=0 uncles=0 elapsed=239.922µs
INFO [02-17|20:25:50] epoch=0 percentage=76 elapsed=3.027s
INFO [02-17|20:25:50] epoch=0 elapsed=4.006s
INFO [02-17|20:25:50] epoch=0 percentage=0 elapsed=11.551s
INFO [02-17|20:25:50] epoch=0 percentage=1 elapsed=22.450s
INFO [02-17|20:25:50] epoch=0 percentage=2 elapsed=32.368s
INFO [02-17|20:25:50] epoch=0 percentage=3 elapsed=42.428s
INFO [02-17|20:25:50] epoch=0 percentage=4 elapsed=52.400s
INFO [02-17|20:25:50] epoch=0 percentage=5 elapsed=1m2.439s
INFO [02-17|20:25:50] epoch=0 percentage=6 elapsed=1m12.183s
INFO [02-17|20:25:50] epoch=0 percentage=7 elapsed=1m21.869s
INFO [02-17|20:25:50] epoch=0 percentage=8 elapsed=1m31.572s
INFO [02-17|20:25:50] epoch=0 percentage=9 elapsed=1m41.318s
INFO [02-17|20:25:50] epoch=0 percentage=10 elapsed=1m51.515s
INFO [02-17|20:25:50] epoch=0 percentage=11 elapsed=2m2.198s
INFO [02-17|20:25:50] epoch=0 percentage=12 elapsed=2m14.176s
INFO [02-17|20:25:50] epoch=0 percentage=13 elapsed=2m26.992s
```

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()
```

```
INFO [02-18|00:17:58] Generating DAG in progress
INFO [02-18|00:18:54] Generating DAG in progress
INFO [02-18|00:19:52] Generating DAG in progress
INFO [02-18|00:20:53] Generating DAG in progress
INFO [02-18|00:21:52] Generating DAG in progress
INFO [02-18|00:23:32] Generating DAG in progress
INFO [02-18|00:25:15] Generating DAG in progress
INFO [02-18|00:26:41] Generating DAG in progress
INFO [02-18|00:28:02] Generating DAG in progress
INFO [02-18|00:28:02] Generated ethash verification cache
INFO [02-18|00:39:59] Successfully sealed new block
INFO [02-18|00:40:00] ↴mined potential block
INFO [02-18|00:40:00] Commit new mining work
INFO [02-18|00:58:28] Successfully sealed new block
INFO [02-18|00:58:28] ↴mined potential block
INFO [02-18|00:58:28] Commit new mining work
INFO [02-18|00:59:20] Successfully sealed new block
INFO [02-18|00:59:20] ↴mined potential block
INFO [02-18|00:59:20] Commit new mining work
INFO [02-18|01:03:22] Successfully sealed new block
INFO [02-18|01:03:22] ↴mined potential block
INFO [02-18|01:03:22] Commit new mining work
^C
> miner.stop()
true
>
```

Executing the commands: Current block after Mining



Mining

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

To check the current block number execute the following command:

`eth.blockNumber()`

Current block
number after
mining

```
INFO [02-18|00:25:15] Generating DAG in progress
INFO [02-18|00:26:41] Generating DAG in progress
INFO [02-18|00:28:02] Generating DAG in progress
INFO [02-18|00:28:02] Generated ethash verification cache
INFO [02-18|00:39:59] Successfully sealed new block
INFO [02-18|00:40:00] ↵mined potential block
INFO [02-18|00:40:00] Commit new mining work
INFO [02-18|00:58:28] Successfully sealed new block
INFO [02-18|00:58:28] ↵mined potential block
INFO [02-18|00:58:28] Commit new mining work
INFO [02-18|00:59:20] Successfully sealed new block
INFO [02-18|00:59:20] ↵mined potential block
INFO [02-18|00:59:20] Commit new mining work
INFO [02-18|01:03:22] Successfully sealed new block
INFO [02-18|01:03:22] ↵mined potential block
INFO [02-18|01:03:22] Commit new mining work
^C
> miner.stop()
true
> eth.blockNumber
5
> █
```

```
epoch=1 percentage=97 elapsed=3h42m34.13s
epoch=1 percentage=98 elapsed=3h42m34.13s
epoch=1 percentage=99 elapsed=3h42m34.13s
epoch=1 elapsed=3h42m34.13s
number=2 hash=234682...de4a4
number=2 hash=234682...de4a4
number=3 txs=0 uncles=0 elapsed=3h42m34.13s
number=3 hash=c46b8a...57913
number=3 hash=c46b8a...57913
number=4 txs=0 uncles=0 elapsed=3h42m34.13s
number=4 hash=eecd2d...1c41d
number=4 hash=eecd2d...1c41d
number=5 txs=0 uncles=0 elapsed=3h42m34.13s
number=5 hash=862491...7ec21
number=5 hash=862491...7ec21
number=6 txs=0 uncles=0 elapsed=3h42m34.13s
```

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")

```
INFO [02-18|00:39:59] Successfully sealed new block
INFO [02-18|00:40:00] ↵mined potential block
INFO [02-18|00:40:00] Commit new mining work
INFO [02-18|00:58:28] Successfully sealed new block
INFO [02-18|00:58:28] ↵mined potential block
INFO [02-18|00:58:28] Commit new mining work
INFO [02-18|00:59:20] Successfully sealed new block
INFO [02-18|00:59:20] ↵mined potential block
INFO [02-18|00:59:20] Commit new mining work
INFO [02-18|01:03:22] Successfully sealed new block
INFO [02-18|01:03:22] ↵mined potential block
INFO [02-18|01:03:22] Commit new mining work
^C
> miner.stop()
true
> eth.blockNumber
5
> eth.accounts
["0xfbff220770c74b87c5a89c83d03335ec0d9885f33", "0x10173ceb0c7c55e61a6b432b75e40d2b88bdb519"]
to install eth.getBalance("0xfbff220770c74b87c5a89c83d03335ec0d9885f33")
25000000000000000000000000000000
>
```

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>