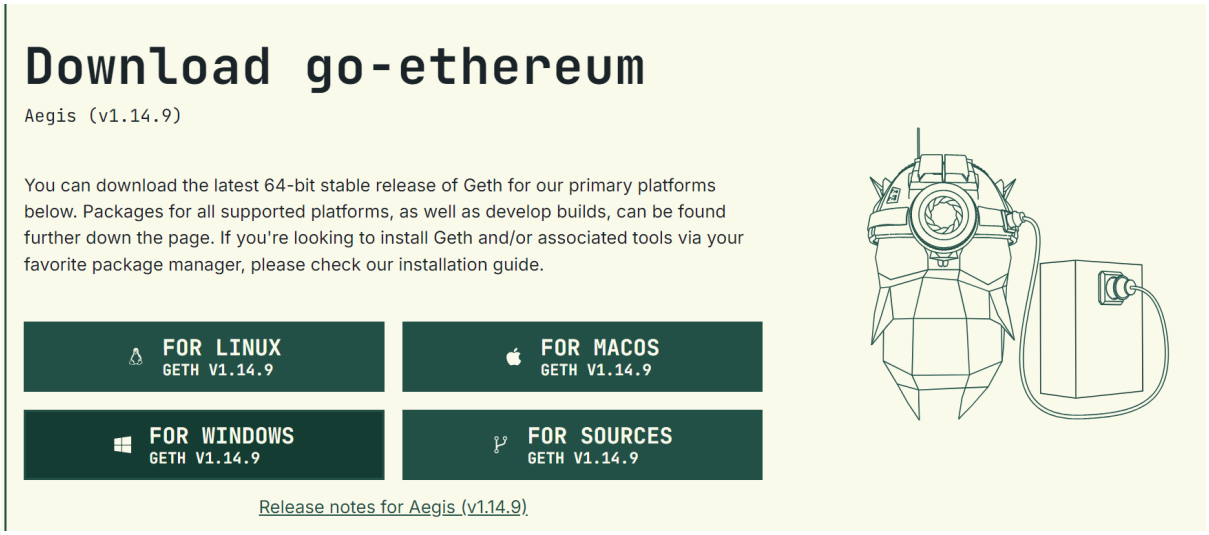
Experiment 5

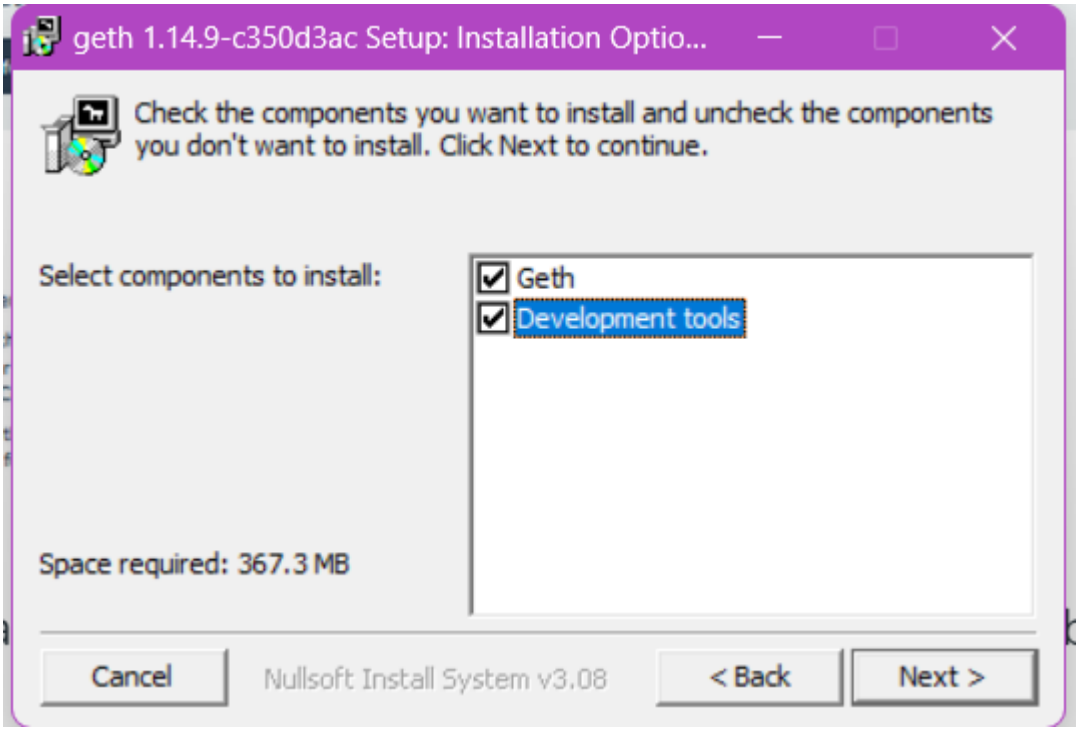
Aim – Implementation of the blockchain platform Ethereum using Geth

Code and Output

Step 1: Install Geth on Your System



While installing Geth make sure to select both checkboxes as shown below.
Select checkboxes

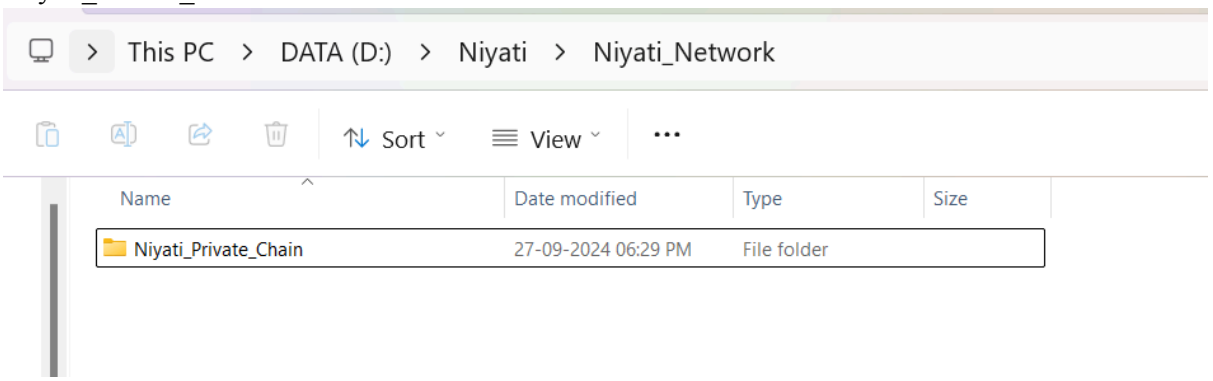


After installing Geth on your system open PowerShell or command prompt and type geth and press enter, the following output will be displayed.

```
PS C:\Users\Niyati Savant> cd D:\Niyati
PS D:\Niyati> .\geth.exe
INFO [09-27|18:27:17.710] Starting Geth on Ethereum mainnet...
INFO [09-27|18:27:17.710] Bumping default cache on mainnet         provided=1024 updated=4096
INFO [09-27|18:27:17.713] Maximum peer count                       ETH=50 total=50
WARN [09-27|18:27:17.713] Sanitizing cache to Go's GC limits       provided=4096 updated=2692
INFO [09-27|18:27:17.719] Set global gas cap                       cap=50,000,000
INFO [09-27|18:27:17.719] Initializing the KZG library             backend=gokzg
INFO [09-27|18:27:17.750] Allocated trie memory caches             clean=403.00MiB dirty=673.00MiB
INFO [09-27|18:27:17.751] Defaulting to pebble as the backing database
INFO [09-27|18:27:17.751] Allocated cache and file handles         database="C:\Users\Niyati Savant\AppData\Local\Ethere
um\geth\chaindata" cache=1.31GiB handles=8192
INFO [09-27|18:27:17.791] Opened ancient database                  database="C:\Users\Niyati Savant\AppData\Local\Ethere
um\geth\chaindata\ancient\chain" readonly=false
INFO [09-27|18:27:17.791] State schema set to default              scheme=path
ERROR[09-27|18:27:17.791] Head block is not reachable
INFO [09-27|18:27:17.791] Initialising Ethereum protocol           network=1 dbversion=<nil>
WARN [09-27|18:27:17.794] Sanitizing invalid node buffer size      provided=673.00MiB updated=256.00MiB
INFO [09-27|18:27:17.820] Opened ancient database                  database="C:\Users\Niyati Savant\AppData\Local\Ethere
um\geth\chaindata\ancient\state" readonly=false
INFO [09-27|18:27:17.820] Writing default main-net genesis block
INFO [09-27|18:27:18.081]
INFO [09-27|18:27:18.081] ---------------------------------------------------------------------------------------------------
----------------------------------------------------------
INFO [09-27|18:27:18.081] Chain ID:  1 (mainnet)
```

Step 2: Create a Folder For Private Ethereum

Create a separate folder for this project. In this case, the folder is ‘Niyati_Network’.
Create a new folder inside the folder ‘Niyati_Network’ for the private Ethereum network as it keeps your Ethereum private network files separate from the public files. In this example folder is ‘Niyati_Private_Chain’.



Step 3: Create a Genesis Block

The blockchain is a distributed digital register in which all transactions are recorded in sequential order in the form of blocks. There are a limitless number of blocks, but there is always one separate block that gave rise to the whole chain i.e. the genesis block.

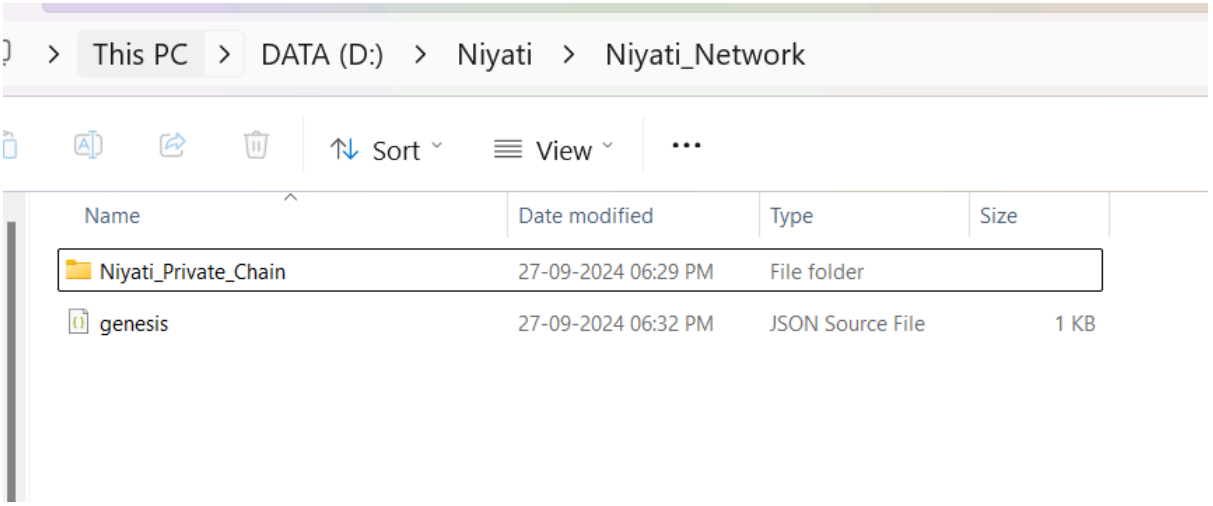
To create a private blockchain, a genesis blockis needed. To do this, create a genesis file, which is a JSON file with the following commands-

```
{

    “config”:{

      “chainId”:987,
```

```
    “homesteadBlock”:0,

    “eip150Block”:0,

    “eip155Block”:0,

    “eip158Block”:0

  },

  “difficulty”:”0x400″,

  “gasLimit”:”0x8000000″,

  “alloc”:{}

}
```


- config: It defines the blockchain configuration and determines how the network will work.
- chainId: This is the chain number used by several blockchains. The Ethereum main chain number is “1”. Any random number can be used, provided that it does not match with another blockchain number.
- homesteadBlock: It is the first official stable version of the Ethereum protocol and its attribute value is “0”.
- One can connect other protocols such as Byzantium, eip155B, and eip158. To do this, under the homesteadBlock add the protocol name with the Block prefix (for example, eip158Block) and set the parameter “0” to them.
- difficulty: It determines the difficulty of generating blocks. Set it low to keep the complexity low and to avoid waiting during tests.
- gasLimit: Gas is the “fuel” that is used to pay transaction fees on the Ethereum network. The more gas a user is willing to spend, the higher will be the priority of his transaction in the queue. It is recommended to set this value to a high enough level to avoid limitations during tests.
- alloc: It is used to create a cryptocurrency wallet for our private blockchain and fill it with fake ether. In this case, this option will not be used to show how to initiate mining on a private blockchain.

This file can be created by using any text editor and save the file with JSON extension in the folder Niyati_Network.



Step 4: Execute genesis file

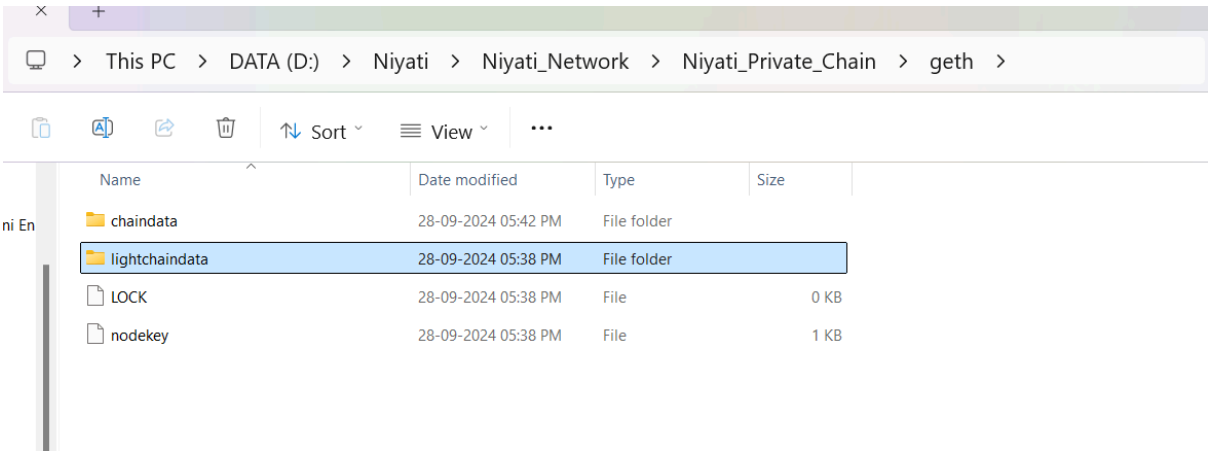
Open cmd or PowerShell in admin mode enter the following command-

geth –identity "yourIdentity" init \path_to_folder\CustomGenesis.json –datadir \path_to_data_directory\MyPrivateChain

```
PS D:\Niyati> .\geth.exe init "D:\Niyati\Niyati_Network\genesis.json"
INFO [09-27|21:20:35.462] Maximum peer count                       ETH=50 total=50
INFO [09-27|21:20:35.468] Set global gas cap                       cap=50,000,000
INFO [09-27|21:20:35.469] Initializing the KZG library             backend=gokzg
INFO [09-27|21:20:35.498] Using pebble as the backing database
INFO [09-27|21:20:35.498] Allocated cache and file handles         database="C:\Users\Niyati Savant\AppData\Local\Ethere
um\geth\chaindata" cache=16.00MiB handles=16
INFO [09-27|21:20:35.525] Opened ancient database                  database="C:\Users\Niyati Savant\AppData\Local\Ethere
um\geth\chaindata\ancient\chain" readonly=false
INFO [09-27|21:20:35.525] State scheme set to already existing     scheme=path
INFO [09-27|21:20:35.538] Opened ancient database                  database="C:\Users\Niyati Savant\AppData\Local\Ethere
um\geth\chaindata\ancient\state" readonly=false
```

Parameters-
path_to_folder- Location of Genesis file.
path_to_data_directory- Location of the folder in which the data of our private chain will be stored.
The above command instructs Geth to use the CustomGenesis.json file. After executing the above command Geth is connected to the Genesis file and it seems like this:

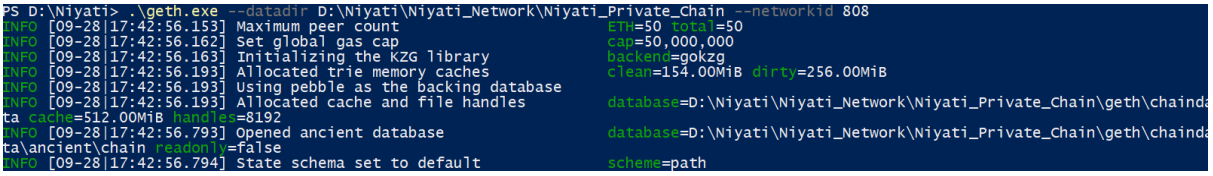
```
PS D:\Niyati> .\geth.exe --datadir .\Niyati_Network\Niyati_Private_Chain init .\Niyati_Network\genesis.json
INFO [09-28|17:38:10.904] Maximum peer count                       ETH=50 total=50
INFO [09-28|17:38:10.917] Set global gas cap                       cap=50,000,000
INFO [09-28|17:38:10.917] Initializing the KZG library             backend=gokzg
INFO [09-28|17:38:10.973] Defaulting to pebble as the backing database
INFO [09-28|17:38:10.973] Allocated cache and file handles         database=D:\Niyati\Niyati_Network\Niyati_Private_Chain\geth\chaindat
a cache=16.00MiB handles=16
INFO [09-28|17:38:11.782] Opened ancient database                  database=D:\Niyati\Niyati_Network\Niyati_Private_Chain\geth\chaindat
a\ancient\chain readonly=false
INFO [09-28|17:38:11.782] State schema set to default              scheme=path
ERROR[09-28|17:38:11.783] Head block is not reachable
INFO [09-28|17:38:12.287] Opened ancient database                  database=D:\Niyati\Niyati_Network\Niyati_Private_Chain\geth\chaindat
a\ancient\state readonly=false
INFO [09-28|17:38:12.287] Writing custom genesis block
INFO [09-28|17:38:12.652] Successfully wrote genesis state         database=chaindata hash=b052b0..1553c1
INFO [09-28|17:38:12.656] Defaulting to pebble as the backing database
INFO [09-28|17:38:12.656] Allocated cache and file handles         database=D:\Niyati\Niyati_Network\Niyati_Private_Chain\geth\lightcha
indata cache=16.00MiB handles=16
INFO [09-28|17:38:13.522] Opened ancient database                  database=D:\Niyati\Niyati_Network\Niyati_Private_Chain\geth\lightcha
indata\ancient\chain readonly=false
INFO [09-28|17:38:13.529] State schema set to default              scheme=path
ERROR[09-28|17:38:13.529] Head block is not reachable
INFO [09-28|17:38:14.048] Opened ancient database                  database=D:\Niyati\Niyati_Network\Niyati_Private_Chain\geth\lightcha
indata\ancient\state readonly=false
INFO [09-28|17:38:14.055] Writing custom genesis block
INFO [09-28|17:38:14.431] Successfully wrote genesis state         database=lightchaindata hash=b052b0..1553c1
PS D:\Niyati>
```



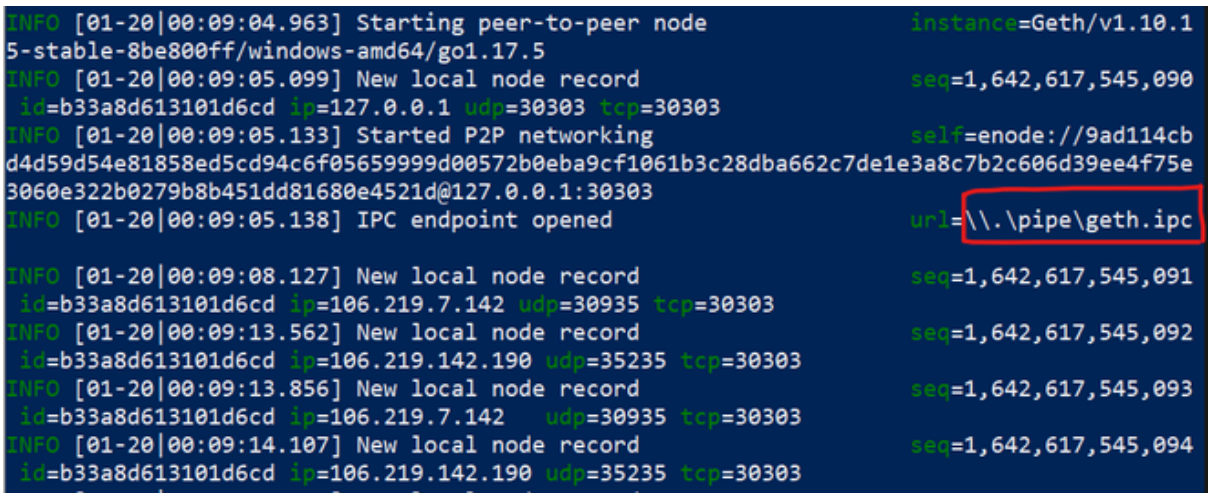
Step 5: Initialize the private network

Launch the private network in which various nodes can add new blocks for this we have to run the command-

geth –datadir \path_to_your_data_directory\MyPrivateChain –networkid 8080



The command also has the identifier 8080. It should be replaced with an arbitrary number that is not equal to the identifier of the networks already created, for example, the identifier of the main network Ethereum (“networkid = 1”). After successfully executing the command we can see like this-



Note:
The highlighted text is the address of geth.ipc file finds it in your console and copy it for use in the next step.
Every time there is a need to access the private network chain, one will need to run commands in the console that initiate a connection to the Genesis file and the private network.

Now a personal blockchain and a private Ethereum network is ready.

Step 6: Create an Externally owned account(EOA)
Externally Owned Account(EOA) has the following features-
Controlled by an External party or person.
Accessed through private Keys.
Contains Ether Balance.
Can send transactions as well as 'trigger' contract accounts.
Steps to create EOA are:
To manage the blockchain network, one need EOA. To create it, run Geth in two windows. In the second window console enter the following command-
geth attach \path_to_your_data_directory\YOUR_FOLDER\geth.ipc
or
geth attach \\.\pipe\geth.ipc

This will connect the second window to the terminal of the first window. The terminal will display the following-

```
PS C:\WINDOWS\system32> geth attach \\.\pipe\geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.15-stable-8be800ff/windows-amd64/go1.17.5
at block: 0 (Thu Jan 01 1970 05:30:00 GMT+0530 (IST))
 datadir: E:\MyNetwork\MyPrivateChain
 modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

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

Create an account by using the command-
personal.newAccount()

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x125c7bce5af112d0e271092be64c87ce5c31696c"
>
```

After executing this command enter Passphrase and you will get your account number and save this number for future use.

save account number

To check the balance status of the account execute the following command-

ether balance

```
> eth.getBalance("0x125c7bce5af112d0e271092be64c87ce5c31696c")
0
```

It can be seen from the above screenshot that it shows zero balance. This is because when starting a private network in the genesis file, we did not specify anything in the alloc attribute.

Step 7: Mining our private chain of Ethereum

If we mine in the main chain of Ethereum it will require expensive equipment with powerful graphics processors. Usually, ASICs are used for this but in our chain high performance is not required and we can start mining by using the following command-

miner.start()

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

If the balance status is checked after a couple of seconds the account is replenished with fake ether. After that, one can stop mining by using the following command-

miner.stop()

```
> eth.getBalance("0x125c7bce5af112d0e271092be64c87ce5c31696c")
2000000000000000000000
> miner.stop()
null
>
```