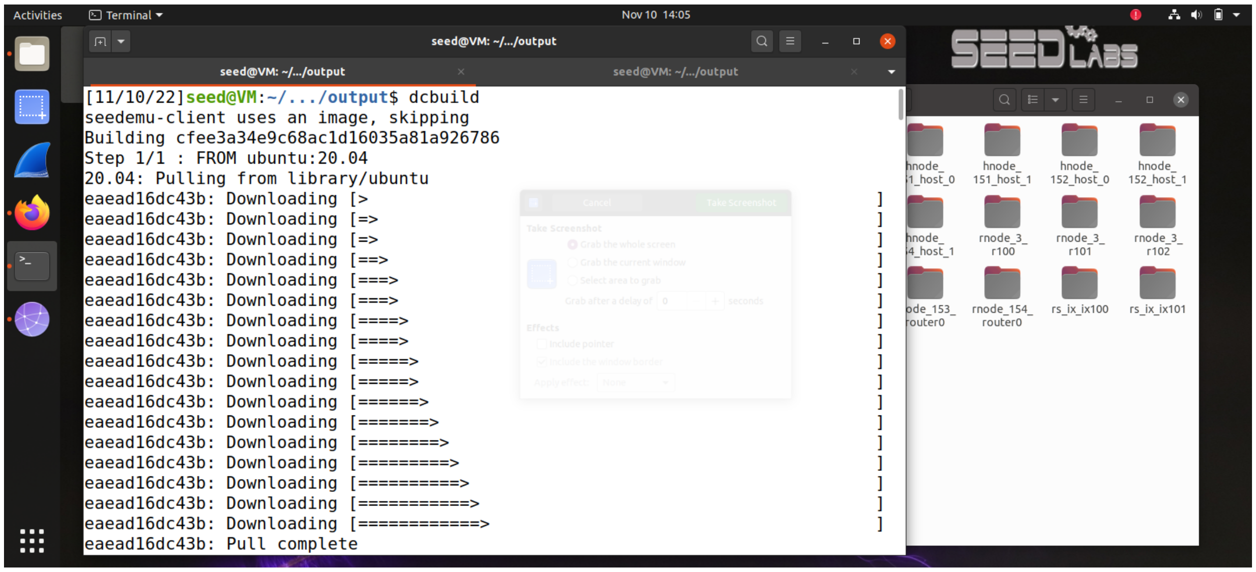
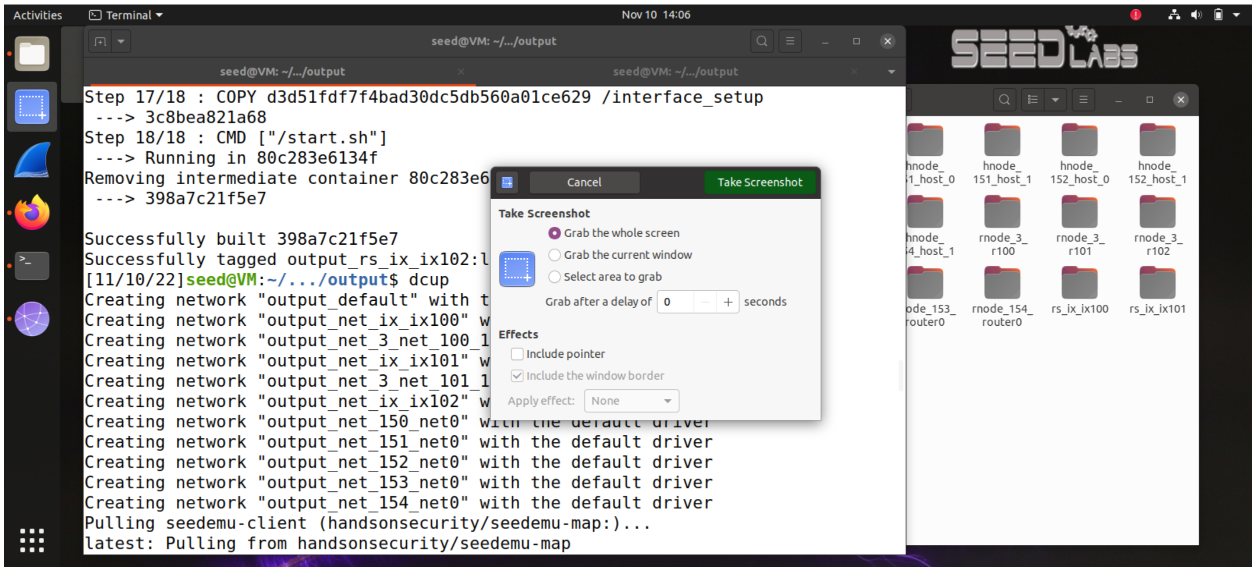
Computer Security: Lab 09- Blockchain Lab: HENIL.V

Lab Setup:

Dcbuild and Dcup:

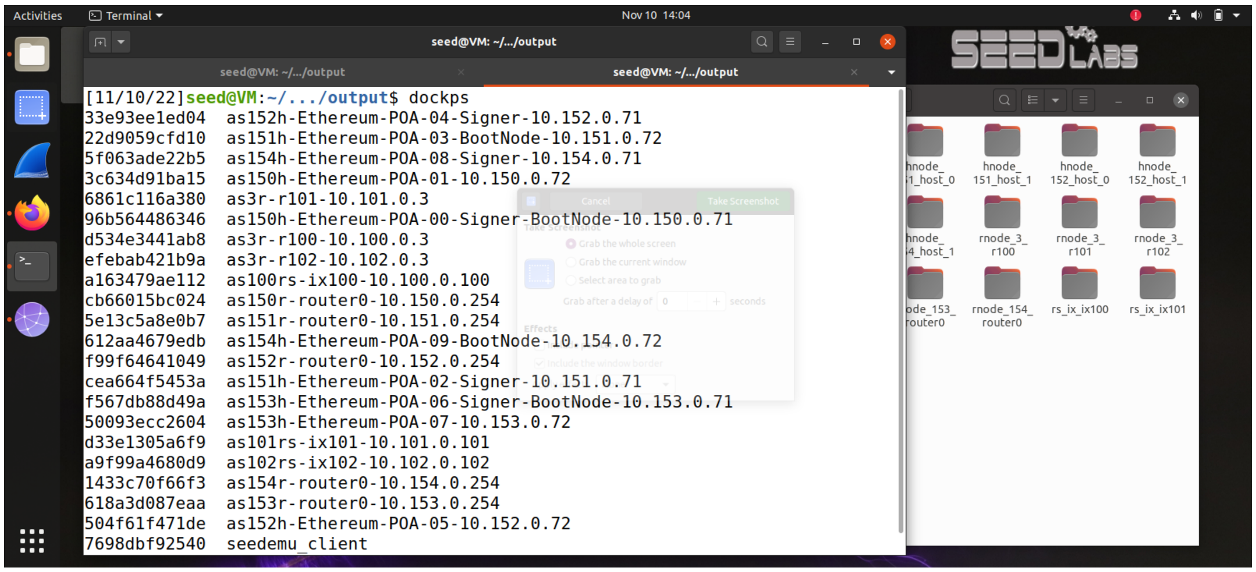


```
[11/10/22]seed@VM:~/.../output$ dcbuild
seedemu-client uses an image, skipping
Building cfee3a34e9c68ac1d16035a81a926786
Step 1/1 : FROM ubuntu:20.04
20.04: Pulling from library/ubuntu
eaead16dc43b: Downloading [>                                                  ]
eaead16dc43b: Downloading [=>                                                 ]
eaead16dc43b: Downloading [=>                                                 ]
eaead16dc43b: Downloading [==>                                                ]
eaead16dc43b: Downloading [===>                                               ]
eaead16dc43b: Downloading [===>                                               ]
eaead16dc43b: Downloading [====>                                              ]
eaead16dc43b: Downloading [====>                                              ]
eaead16dc43b: Downloading [=====>                                             ]
eaead16dc43b: Downloading [=====>                                             ]
eaead16dc43b: Downloading [======>                                            ]
eaead16dc43b: Downloading [=======>                                           ]
eaead16dc43b: Downloading [========>                                          ]
eaead16dc43b: Downloading [=========>                                         ]
eaead16dc43b: Downloading [==========>                                        ]
eaead16dc43b: Downloading [===========>                                       ]
eaead16dc43b: Downloading [============>                                      ]
eaead16dc43b: Pull complete
```



```
Step 17/18 : COPY d3d51fdf7f4bad30dc5db560a01ce629 /interface_setup
 ---> 3c8bea821a68
Step 18/18 : CMD ["/start.sh"]
 ---> Running in 80c283e6134f
Removing intermediate container 80c283e6
 ---> 398a7c21f5e7

Successfully built 398a7c21f5e7
Successfully tagged output_rs_ix_ix102:l
[11/10/22]seed@VM:~/.../output$ dcup
Creating network "output_default" with t
Creating network "output_net_ix_ix100" w
Creating network "output_net_3_net_100_1
Creating network "output_net_ix_ix101" w
Creating network "output_net_3_net_101_1
Creating network "output_net_ix_ix102" w
Creating network "output_net_150_net0" with the default driver
Creating network "output_net_151_net0" with the default driver
Creating network "output_net_152_net0" with the default driver
Creating network "output_net_153_net0" with the default driver
Creating network "output_net_154_net0" with the default driver
Pulling seedemu-client (handsonsecurity/seedemu-map:)...
latest: Pulling from handsonsecurity/seedemu-map
```

Lab Docker's :



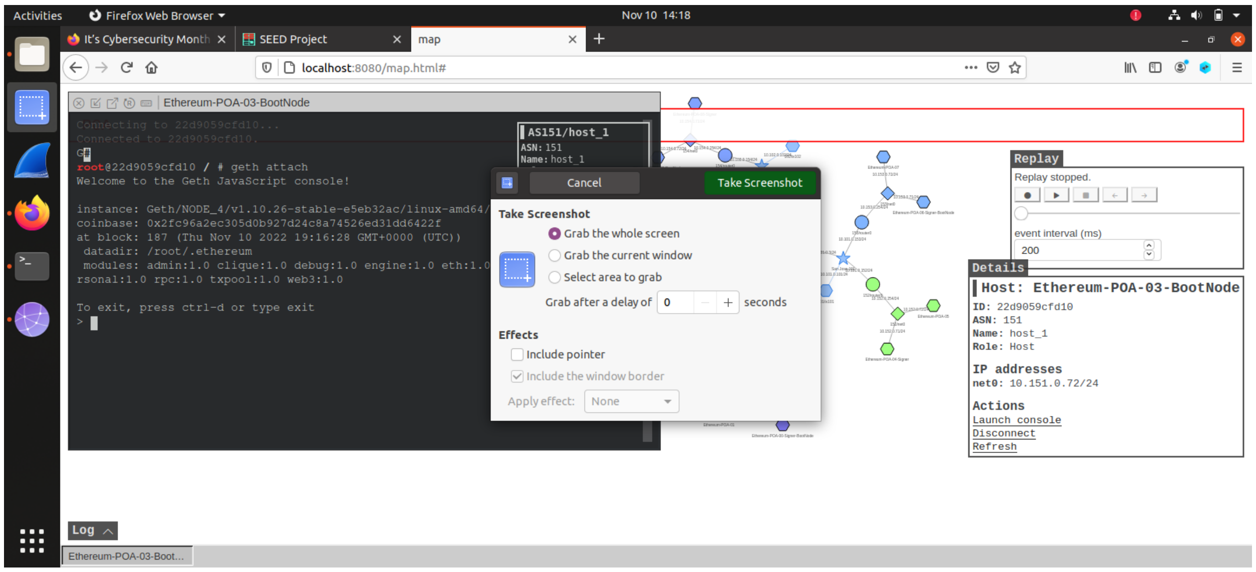
```
[11/10/22]seed@VM:~/.../output$ dockps
33e93ee1ed04  as152h-Ethereum-POA-04-Signer-10.152.0.71
22d9059cfd10  as151h-Ethereum-POA-03-BootNode-10.151.0.72
5f063ade22b5  as154h-Ethereum-POA-08-Signer-10.154.0.71
3c634d91ba15  as150h-Ethereum-POA-01-10.150.0.72
6861c116a380  as3r-r101-10.101.0.3
96b564486346  as150h-Ethereum-POA-00-Signer-BootNode-10.150.0.71
d534e3441ab8  as3r-r100-10.100.0.3
efebab421b9a  as3r-r102-10.102.0.3
a163479ae112  as100rs-ix100-10.100.0.100
cb66015bc024  as150r-router0-10.150.0.254
5e13c5a8e0b7  as151r-router0-10.151.0.254
612aa4679edb  as154h-Ethereum-POA-09-BootNode-10.154.0.72
f99f64641049  as152r-router0-10.152.0.254
cea664f5453a  as151h-Ethereum-POA-02-Signer-10.151.0.71
f567db88d49a  as153h-Ethereum-POA-06-Signer-BootNode-10.153.0.71
50093ecc2604  as153h-Ethereum-POA-07-10.153.0.72
d33e1305a6f9  as101rs-ix101-10.101.0.101
a9f99a4680d9  as102rs-ix102-10.102.0.102
1433c70f66f3  as154r-router0-10.154.0.254
618a3d087eaa  as153r-router0-10.153.0.254
504f61f471de  as152h-Ethereum-POA-05-10.152.0.72
7698dbf92540  seedemu_client
```

The docker that we get up using these commands goes on to to create a virtual environment with multiple nodes or so and each of these nodes have various functionalities such as Booth-Node, Ehtereum Node etc, we can use the grep command to search for the particular kind of docker that we intend to use to communicate.

There are 2 ways to achieve this, we can achieve it using terminal and going on to find the node we want to communicate using docksh in terminal.

There is another way to achieve this node that is via visiting the localhost:8080 http link and then selecting the node individually and then getting the terminal graphically.

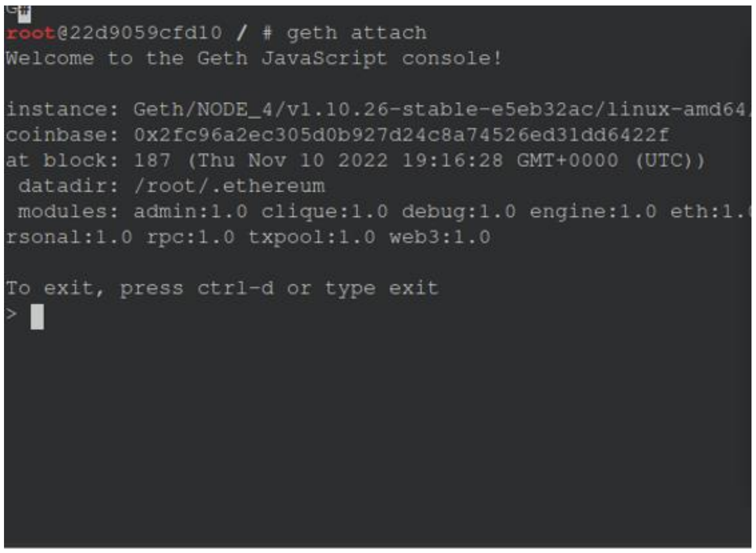
The localhost 8080 link simulates an entire network of blockchain and its various nodes have various functionalities and together it acts as one glued internet for us to perform our blockchain tasks.



```
Connecting to 22d9059cfd10...
Connected to 22d9059cfd10.
root@22d9059cfd10 / # geth attach
Welcome to the Geth JavaScript console!

instance: Geth/NODE_4/v1.10.26-stable-e5eb32ac/linux-amd64/
coinbase: 0x2fc96a2ec305d0b927d24c8a74526ed31dd6422f
at block: 187 (Thu Nov 10 2022 19:16:28 GMT+0000 (UTC))
 datadir: /root/.ethereum
 modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0
rsonal:1.0 rpc:1.0 txpool:1.0 web3:1.0

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

```
Host: Ethereum-POA-03-BootNode
ID: 22d9059cfd10
ASN: 151
Name: host_1
Role: Host

IP addresses
net0: 10.151.0.72/24
```

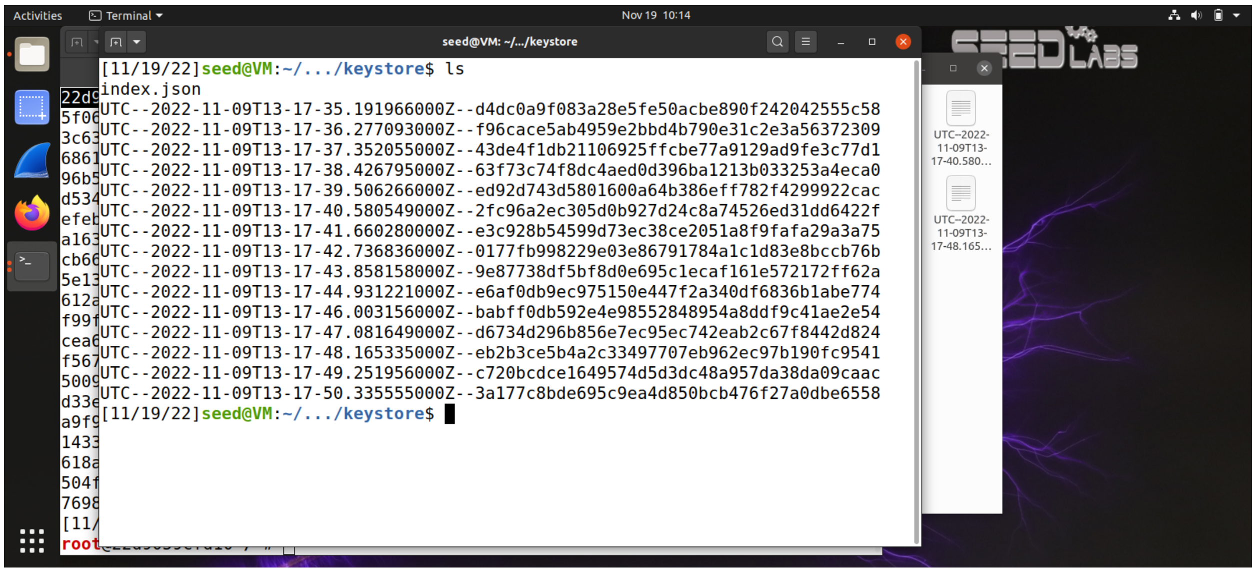
Here as we can see we can get the JavaScript terminal using geth attach:



```
root@22d9059cfd10 / # geth attach
Welcome to the Geth JavaScript console!

instance: Geth/NODE_4/v1.10.26-stable-e5eb32ac/linux-amd64
coinbase: 0x2fc96a2ec305d0b927d24c8a74526ed31dd6422f
at block: 187 (Thu Nov 10 2022 19:16:28 GMT+0000 (UTC))
 datadir: /root/.ethereum
 modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.
rsonal:1.0 rpc:1.0 txpool:1.0 web3:1.0

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

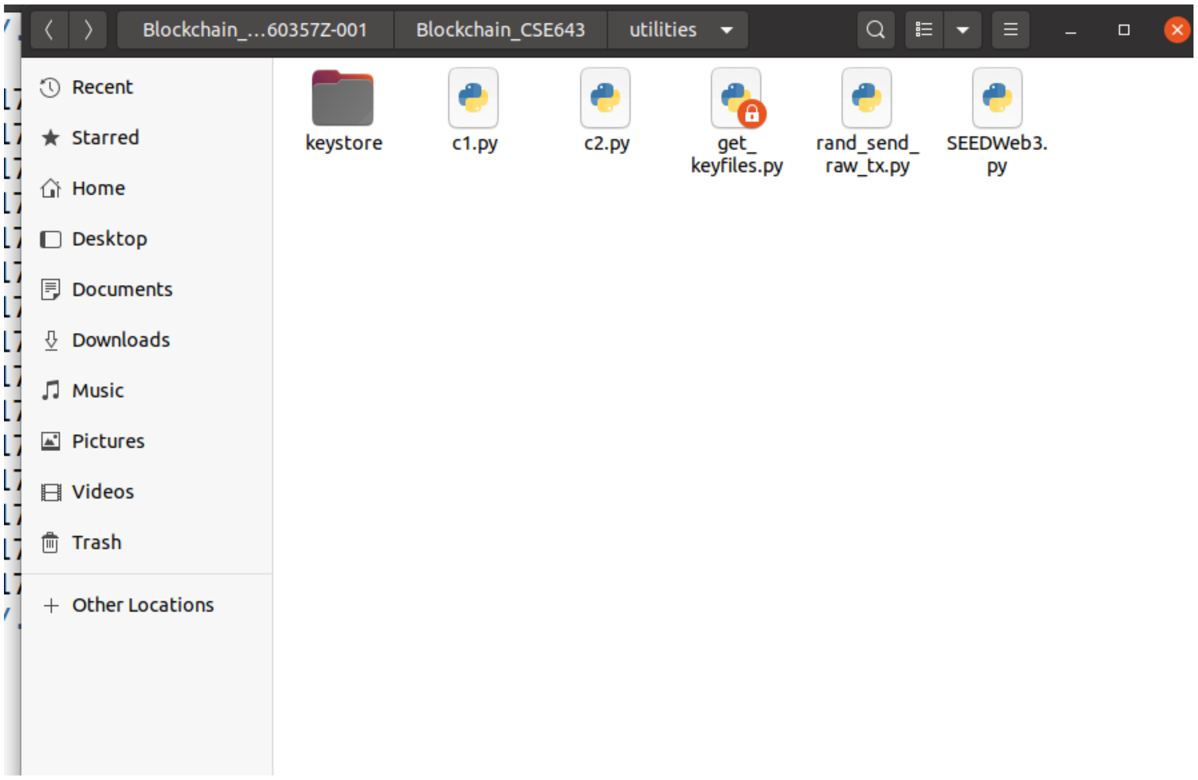
Keyfiles:

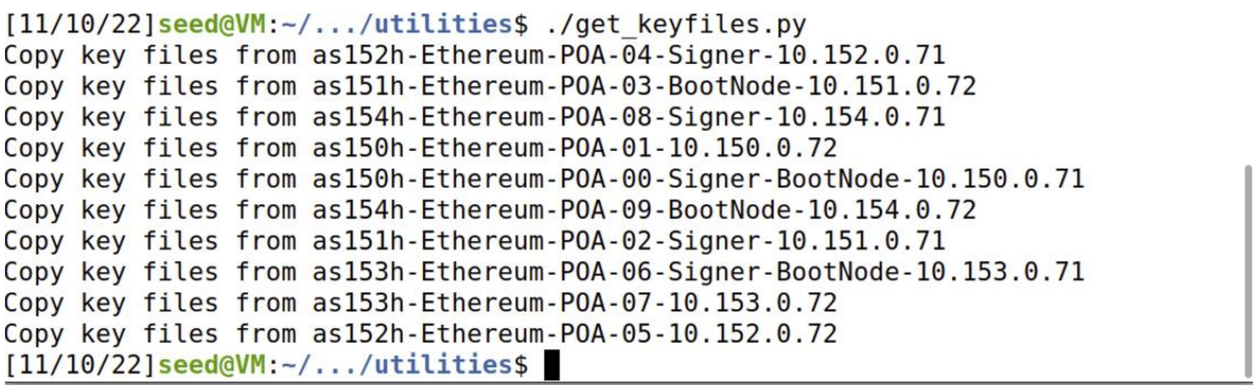
Go to the Utilities folder and make the key files executable and also in simple-server/server/ folder , make the key files executable and then we run get_keyfiles.py: copy the key files from Ethereum nodes and then go back to simple-server/, run run.sh to start the server.



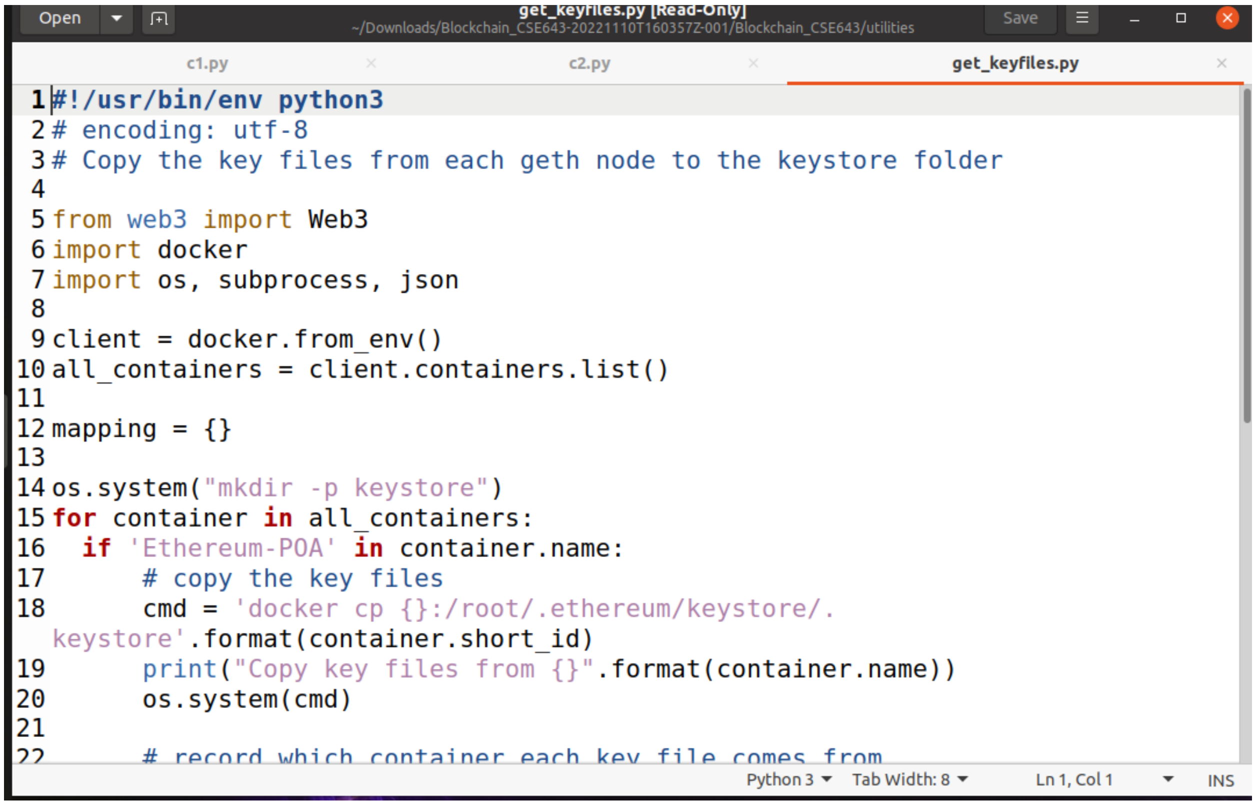
```
[11/19/22]seed@VM:~/.../keystore$ ls
index.json
UTC--2022-11-09T13-17-35.191966000Z--d4dc0a9f083a28e5fe50acbe890f242042555c58
UTC--2022-11-09T13-17-36.277093000Z--f96cace5ab4959e2bbd4b790e31c2e3a56372309
UTC--2022-11-09T13-17-37.352055000Z--43de4f1db21106925ffcbe77a9129ad9fe3c77d1
UTC--2022-11-09T13-17-38.426795000Z--63f73c74f8dc4aed0d396ba1213b033253a4eca0
UTC--2022-11-09T13-17-39.506266000Z--ed92d743d5801600a64b386eff782f4299922cac
UTC--2022-11-09T13-17-40.580549000Z--2fc96a2ec305d0b927d24c8a74526ed31dd6422f
UTC--2022-11-09T13-17-41.660280000Z--e3c928b54599d73ec38ce2051a8f9fafa29a3a75
UTC--2022-11-09T13-17-42.736836000Z--0177fb998229e03e86791784a1c1d83e8bccb76b
UTC--2022-11-09T13-17-43.858158000Z--9e87738df5bf8d0e695c1ecaf161e572172ff62a
UTC--2022-11-09T13-17-44.931221000Z--e6af0db9ec975150e447f2a340df6836b1abe774
UTC--2022-11-09T13-17-46.003156000Z--babff0db592e4e98552848954a8ddf9c41ae2e54
UTC--2022-11-09T13-17-47.081649000Z--d6734d296b856e7ec95ec742eab2c67f8442d824
UTC--2022-11-09T13-17-48.165335000Z--eb2b3ce5b4a2c33497707eb962ec97b190fc9541
UTC--2022-11-09T13-17-49.251956000Z--c720bcdce1649574d5d3dc48a957da38da09caac
UTC--2022-11-09T13-17-50.335555000Z--3a177c8bde695c9ea4d850bcb476f27a0dbe6558
[11/19/22]seed@VM:~/.../keystore$ 
```

Make getkeyfiles executable and generate keys in utilities:





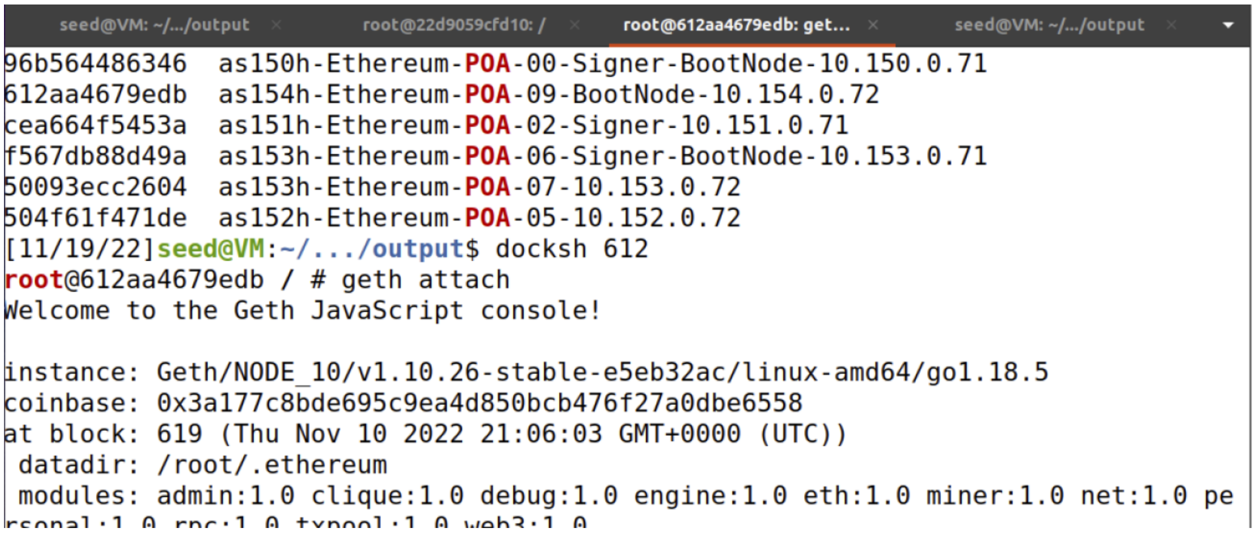
```
[11/10/22]seed@VM:~/.../utilities$ ./get_keyfiles.py
Copy key files from as152h-Ethereum-POA-04-Signer-10.152.0.71
Copy key files from as151h-Ethereum-POA-03-BootNode-10.151.0.72
Copy key files from as154h-Ethereum-POA-08-Signer-10.154.0.71
Copy key files from as150h-Ethereum-POA-01-10.150.0.72
Copy key files from as150h-Ethereum-POA-00-Signer-BootNode-10.150.0.71
Copy key files from as154h-Ethereum-POA-09-BootNode-10.154.0.72
Copy key files from as151h-Ethereum-POA-02-Signer-10.151.0.71
Copy key files from as153h-Ethereum-POA-06-Signer-BootNode-10.153.0.71
Copy key files from as153h-Ethereum-POA-07-10.153.0.72
Copy key files from as152h-Ethereum-POA-05-10.152.0.72
[11/10/22]seed@VM:~/.../utilities$
```



Open
get_keyfiles.py [Read-Only]
~/Downloads/Blockchain_CSE643-20221110T160357Z-001/Blockchain_CSE643/utilities
Save
c1.py
c2.py
get_keyfiles.py
Python 3
Tab Width: 8
Ln 1, Col 1
INS

```python
#!/usr/bin/env python3
# encoding: utf-8
# Copy the key files from each geth node to the keystore folder

from web3 import Web3
import docker
import os, subprocess, json

client = docker.from_env()
all_containers = client.containers.list()

mapping = {}

os.system("mkdir -p keystore")
for container in all_containers:
    if 'Ethereum-POA' in container.name:
        # copy the key files
        cmd = 'docker cp {}:/root/.ethereum/keystore/.
keystore'.format(container.short_id)
        print("Copy key files from {}".format(container.name))
        os.system(cmd)

        # record which container each key file comes from
```

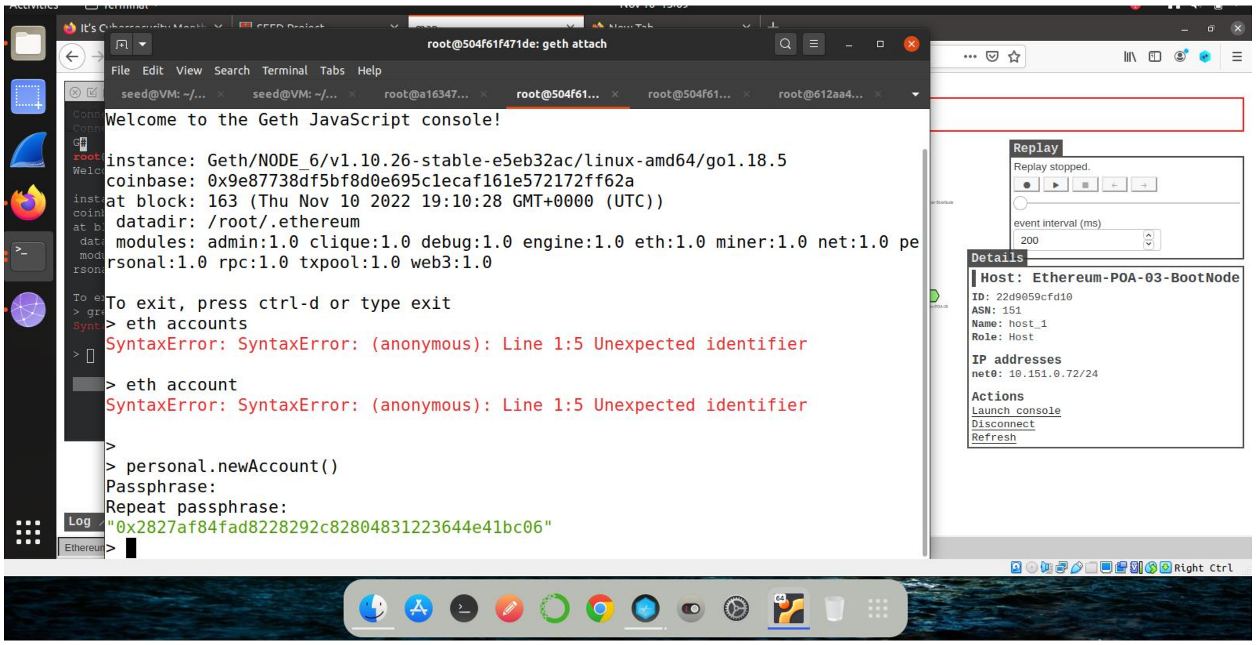
Create a new account(), for that we get a Node using
Dockps: grep- POA
Then enter in it and get use geth attach and get java-script terminal, this is the terminal where we then first create a new account.



```
96b564486346   as150h-Ethereum-POA-00-Signer-BootNode-10.150.0.71
612aa4679edb   as154h-Ethereum-POA-09-BootNode-10.154.0.72
cea664f5453a   as151h-Ethereum-POA-02-Signer-10.151.0.71
f567db88d49a   as153h-Ethereum-POA-06-Signer-BootNode-10.153.0.71
50093ecc2604   as153h-Ethereum-POA-07-10.153.0.72
504f61f471de   as152h-Ethereum-POA-05-10.152.0.72
[11/19/22]seed@VM:~/.../output$ docksh 612
root@612aa4679edb / # geth attach
Welcome to the Geth JavaScript console!

instance: Geth/NODE_10/v1.10.26-stable-e5eb32ac/linux-amd64/go1.18.5
coinbase: 0x3a177c8bde695c9ea4d850bcb476f27a0dbe6558
at block: 619 (Thu Nov 10 2022 21:06:03 GMT+0000 (UTC))
 datadir: /root/.ethereum
 modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 pe
rsonal:1.0 rpc:1.0 txpool:1.0 web3:1.0
```

Create Account Using Geth:

Keys for this account should be encrypted when we enter the passphrase, and we should see it encrypted.



```
Welcome to the Geth JavaScript console!

instance: Geth/NODE_6/v1.10.26-stable-e5eb32ac/linux-amd64/go1.18.5
coinbase: 0x9e87738df5bf8d0e695c1ecaf161e572172ff62a
at block: 163 (Thu Nov 10 2022 19:10:28 GMT+0000 (UTC))
 datadir: /root/.ethereum
 modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 pe
rsonal:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> eth accounts
SyntaxError: SyntaxError: (anonymous): Line 1:5 Unexpected identifier

> eth account
SyntaxError: SyntaxError: (anonymous): Line 1:5 Unexpected identifier

>
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0x2827af84fad8228292c82804831223644e41bc06"
>
```

Replay stopped.
event interval (ms) 200

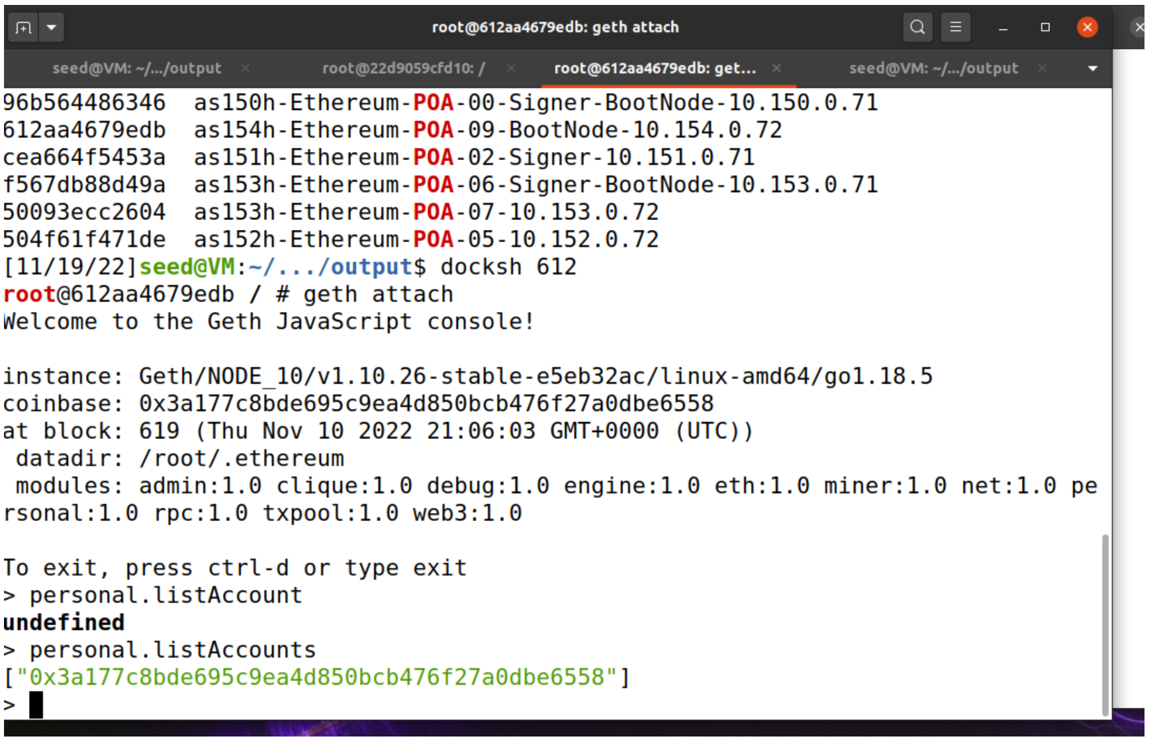
Host: Ethereum-POA-03-BootNode
ID: 22d9059cfd10
ASN: 151
Name: host_1
Role: Host

IP addresses
net0: 10.151.0.72/24

Actions
Launch console
Disconnect
Refresh

Get Account and bal ()

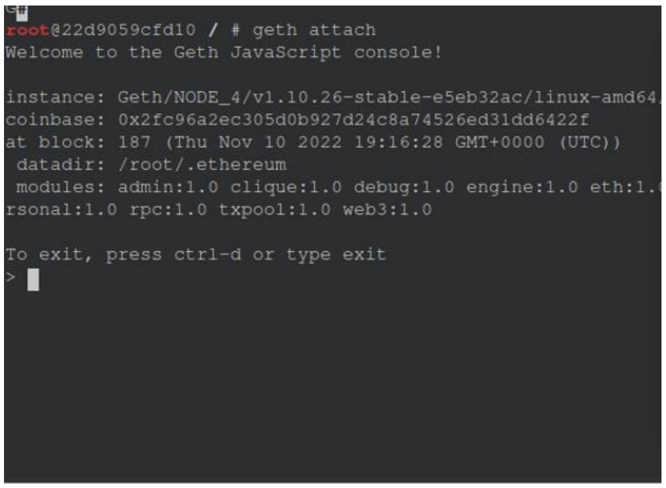
On Running get.accounts we can see the following:



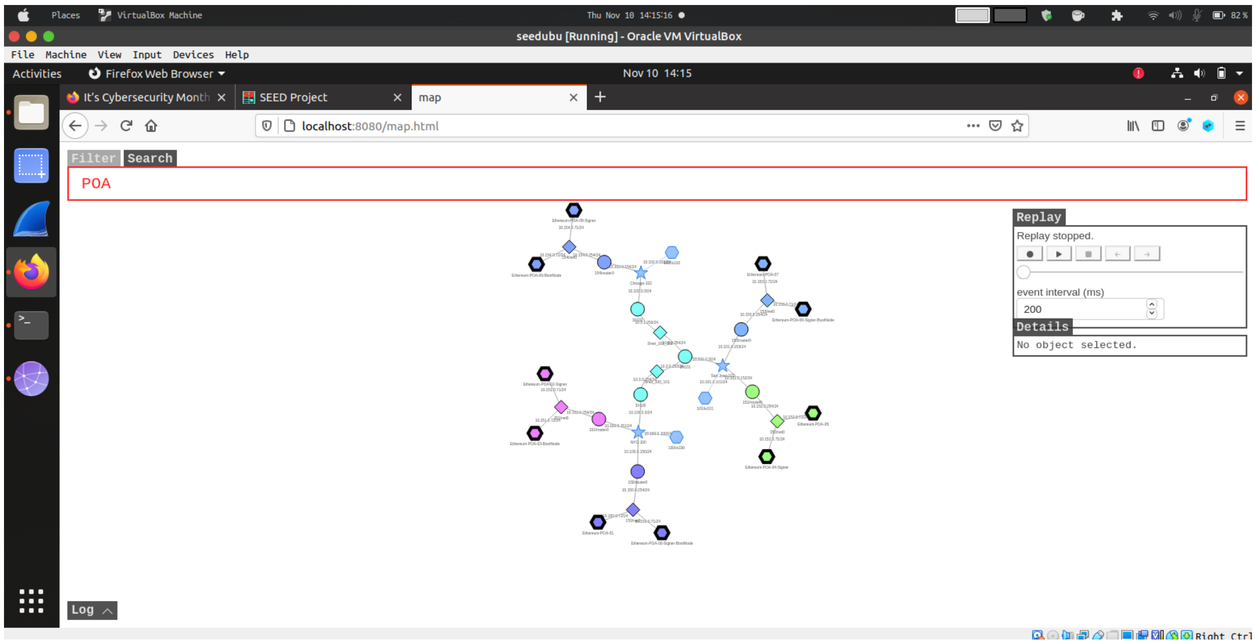
```
96b564486346  as150h-Ethereum-POA-00-Signer-BootNode-10.150.0.71
612aa4679edb  as154h-Ethereum-POA-09-BootNode-10.154.0.72
cea664f5453a  as151h-Ethereum-POA-02-Signer-10.151.0.71
f567db88d49a  as153h-Ethereum-POA-06-Signer-BootNode-10.153.0.71
50093ecc2604  as153h-Ethereum-POA-07-10.153.0.72
504f61f471de  as152h-Ethereum-POA-05-10.152.0.72
[11/19/22]seed@VM:~/.../output$ docksh 612
root@612aa4679edb / # geth attach
Welcome to the Geth JavaScript console!

instance: Geth/NODE_10/v1.10.26-stable-e5eb32ac/linux-amd64/go1.18.5
coinbase: 0x3a177c8bde695c9ea4d850bcb476f27a0dbe6558
at block: 619 (Thu Nov 10 2022 21:06:03 GMT+0000 (UTC))
 datadir: /root/.ethereum
 modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 pe
rsonal:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> personal.listAccount
undefined
> personal.listAccounts
["0x3a177c8bde695c9ea4d850bcb476f27a0dbe6558"]
> 
```

Similarly, we can also go visit localhost https:8080 and from there get the boothnode POA we need and then get the terminal from there and then use java script console from there as seen below:

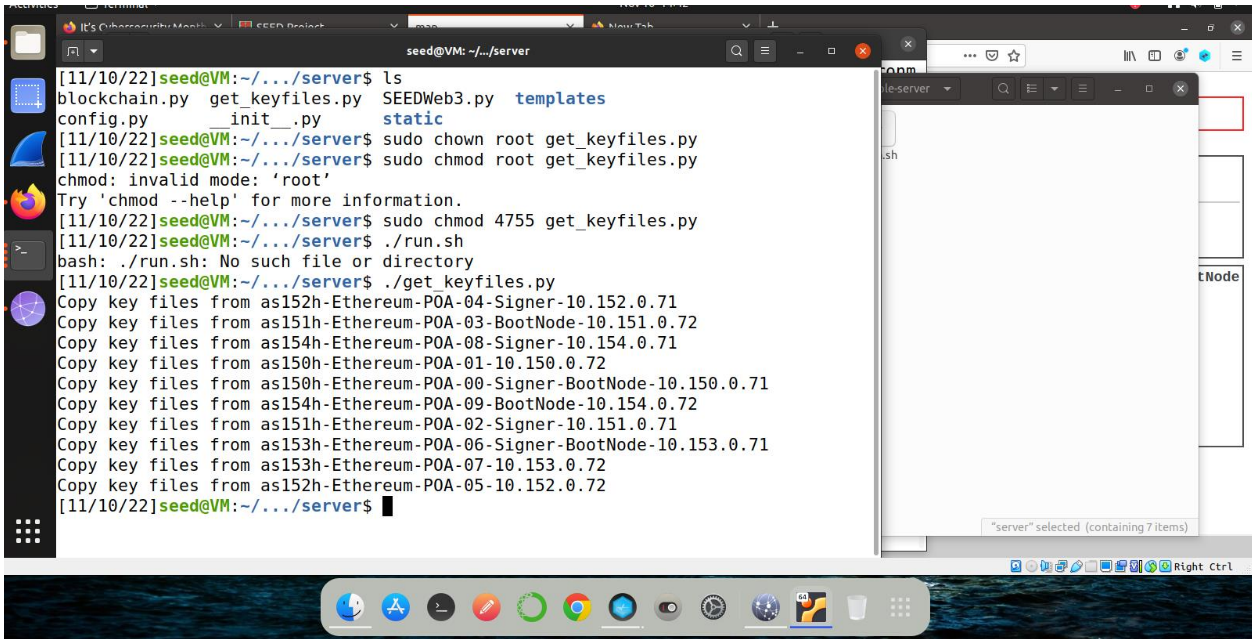
Geth Node via Localhost:8080 :



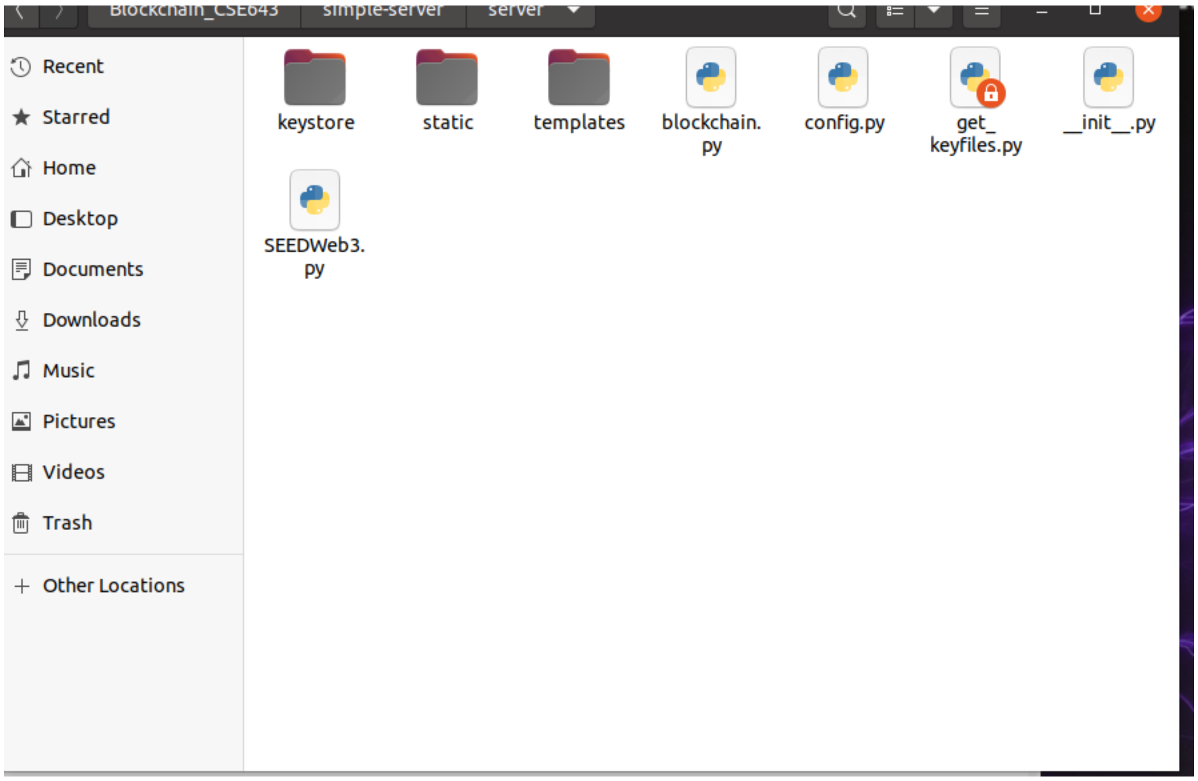
```
root@22d9059cfd10 / # geth attach
Welcome to the Geth JavaScript console!

instance: Geth/NODE_4/v1.10.26-stable-e5eb32ac/linux-amd64
coinbase: 0x2fc96a2ec305d0b927d24c8a74526ed31dd6422f
at block: 187 (Thu Nov 10 2022 19:16:28 GMT+0000 (UTC))
 datadir: /root/.ethereum
 modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.
rsonal:1.0 rpc:1.0 txpool:1.0 web3:1.0

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



Similarly now in the server we make the keys.sh executable and generate the keys.

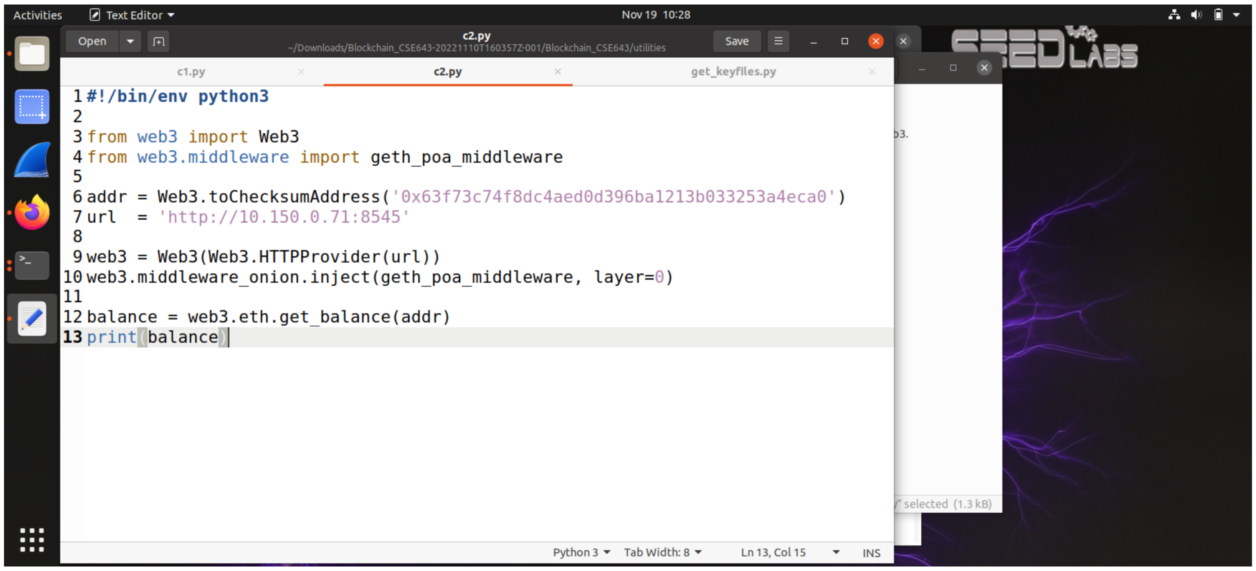


```
[11/10/22]seed@VM:~/.../server$ ls
blockchain.py  get_keyfiles.py  SEEDWeb3.py  templates
config.py      __init__.py      static
[11/10/22]seed@VM:~/.../server$ sudo chown root get_keyfiles.py
[11/10/22]seed@VM:~/.../server$ sudo chmod root get_keyfiles.py
chmod: invalid mode: ‘root’
Try 'chmod --help' for more information.
[11/10/22]seed@VM:~/.../server$ sudo chmod 4755 get_keyfiles.py
[11/10/22]seed@VM:~/.../server$ ./run.sh
bash: ./run.sh: No such file or directory
[11/10/22]seed@VM:~/.../server$ ./get_keyfiles.py
Copy key files from as152h-Ethereum-POA-04-Signer-10.152.0.71
Copy key files from as151h-Ethereum-POA-03-BootNode-10.151.0.72
Copy key files from as154h-Ethereum-POA-08-Signer-10.154.0.71
Copy key files from as150h-Ethereum-POA-01-10.150.0.72
Copy key files from as150h-Ethereum-POA-00-Signer-BootNode-10.150.0.71
Copy key files from as154h-Ethereum-POA-09-BootNode-10.154.0.72
Copy key files from as151h-Ethereum-POA-02-Signer-10.151.0.71
Copy key files from as153h-Ethereum-POA-06-Signer-BootNode-10.153.0.71
Copy key files from as153h-Ethereum-POA-07-10.153.0.72
Copy key files from as152h-Ethereum-POA-05-10.152.0.72
[11/10/22]seed@VM:~/.../server$
```



Get Balance:

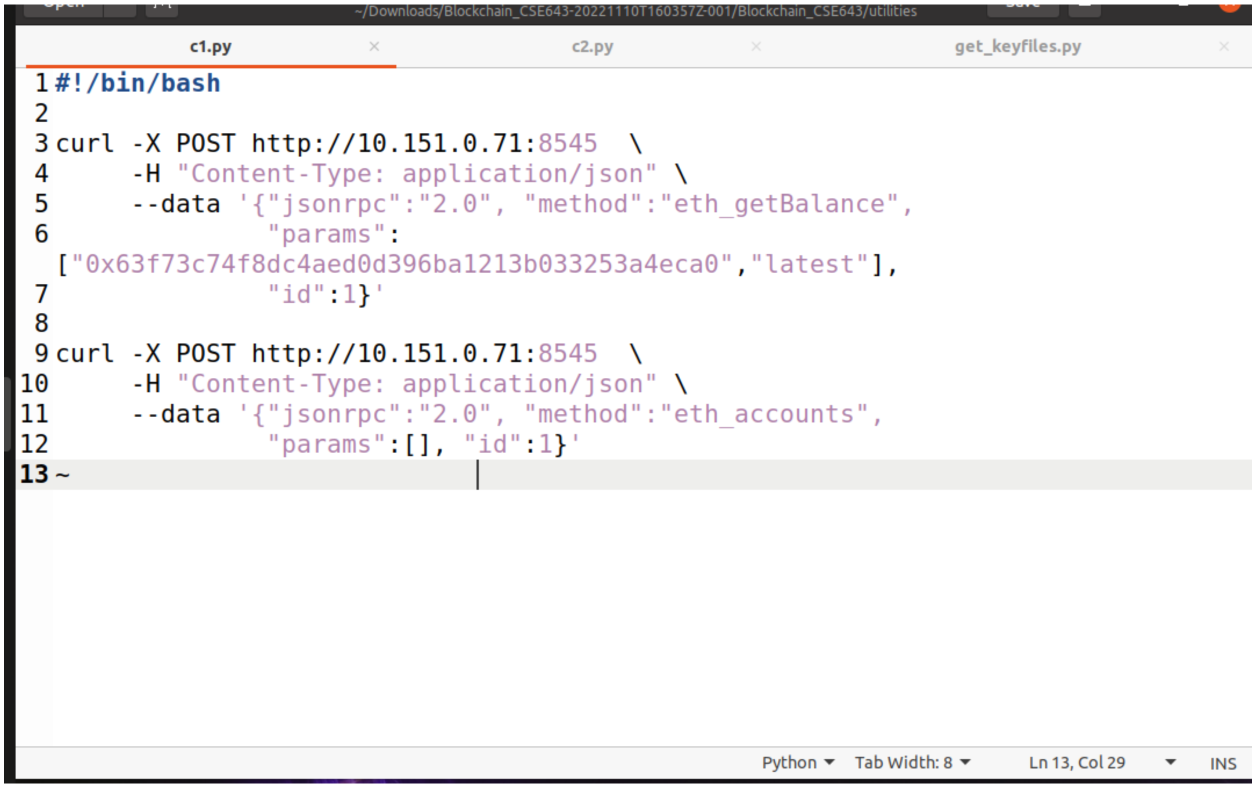
In checksumaddress all the letters in capslock are used to implement checksum.
Blockchain cares about checksum bit whereas Ethereum does not consider caps alphabets for checksum.



```
#!/bin/env python3

from web3 import Web3
from web3.middleware import geth_poa_middleware

addr = Web3.toChecksumAddress('0x63f73c74f8dc4aed0d396ba1213b033253a4eca0')
url  = 'http://10.150.0.71:8545'

web3 = Web3(Web3.HTTPProvider(url))
web3.middleware_onion.inject(geth_poa_middleware, layer=0)

balance = web3.eth.get_balance(addr)
print(balance)
```

First thing we have to do is pip3 install web3. The difference between the two in the following is the way they are passed. To create and send a transaction with web3, you first build a dictionary that contains the basic attributes of the transaction. You then invoke the API method send transaction. As the key of the sender is controlled by the node, the node will then automatically sign the transaction.

The return value is the hash of the transaction that has been generated. Finally, you can check the balance of the involved accounts to see that this worked.



~/Downloads/Blockchain_CSE643-20221110T160357Z-001/Blockchain_CSE643/utilities

c1.py | c2.py | get_keyfiles.py

```
#!/bin/bash

curl -X POST http://10.151.0.71:8545  \
     -H "Content-Type: application/json" \
     --data '{"jsonrpc":"2.0", "method":"eth_getBalance",
             "params":
["0x63f73c74f8dc4aed0d396ba1213b033253a4eca0","latest"],
             "id":1}'

curl -X POST http://10.151.0.71:8545  \
     -H "Content-Type: application/json" \
     --data '{"jsonrpc":"2.0", "method":"eth_accounts",
             "params":[], "id":1}'
~
```