

Bhartiya Vidya Bhavan’s

Sardar Patel Institute of Technology, Mumbai-400058

Department of Computer Science and Engineering

**OEIT1:Blockchain Technology and Applications**

**Lab-2A: Ethereum Blockchain Part-I**

**Set Up an Ethereum Private Blockchain**

**Name : Maaz Shaikh**

**UID : 2021700059**

**Division : TE CSE(DS) OE-C**

**Objective:** **Setup an Ethereum Blockchain**

**Outcomes:** After successful completion of lab students should be able to

1. Implement an Ethereum private blockchain
2. Create the genesis block
3. Start the Ethereum blockchain
4. Create an account on the blockchain
5. Transact Ethers on blockchain
6. Query the blockchain using Geth- Geth console, Geth attach, Geth JSON RPC
7. Use Eth and Web3 interface over RPC

**System Requirements:**

PC (C2D, 4GB RAM, 100GB HDD space and NIC),Ubuntu Linux 14.04/20.04

Internet connectivity,Python Cryptography and Pycrypto,REST API,Go Lanaguage

Go Ethereum

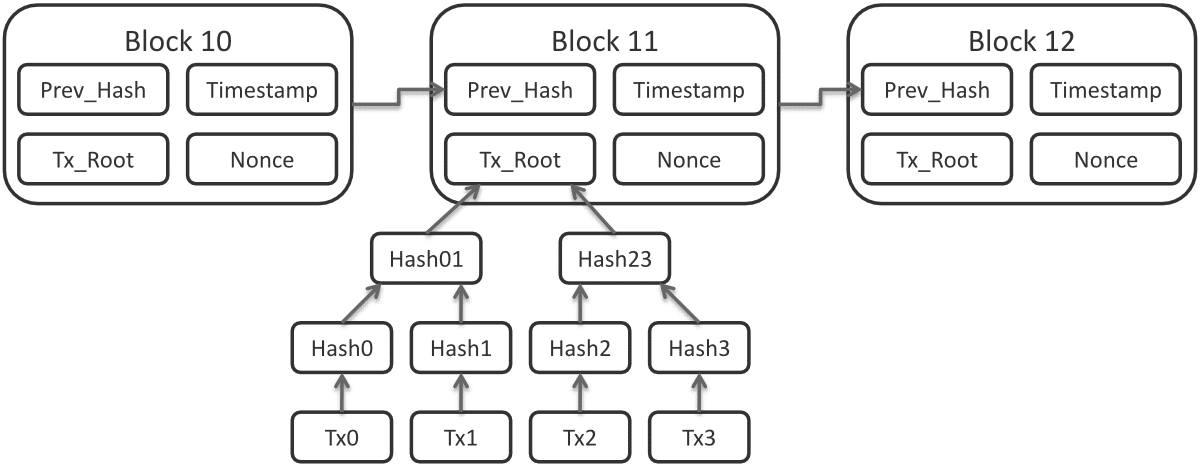


Figure-1: Blockchain Implementation

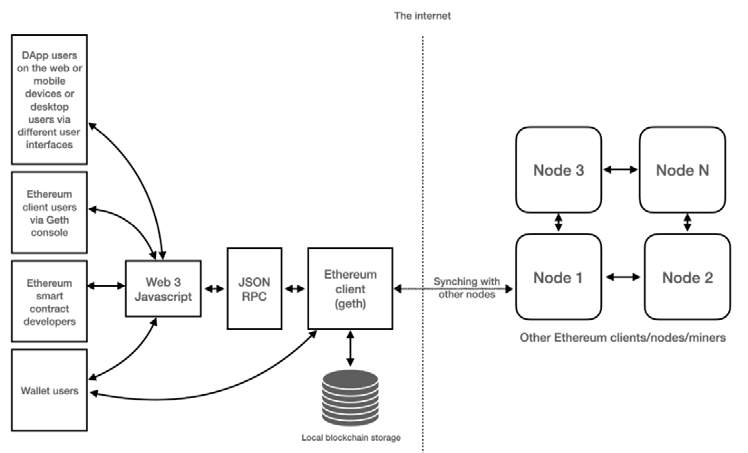


Figure-2: Ethereum high-level ecosystem

**About Ethereum Blockchain:** Ethereum is an open-source, public,blockchain-based distributed computing platform. It features smart contract (scripting) functionality, which facilitates online contractual agreements. The Ethereum elements include:

• Blocks and blockchain

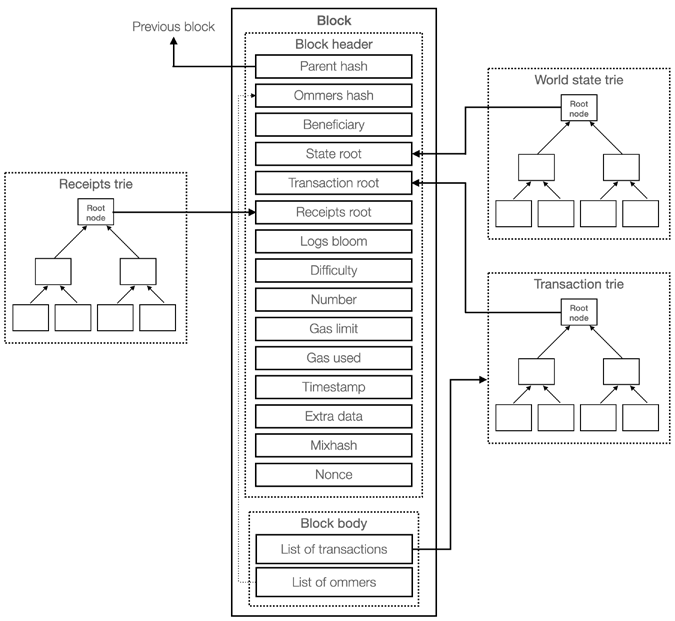
• Wallets and client software

• Nodes and miners

• APIs and tools

• Supporting protocols

• Programming languages

****

**Figure-3: A detailed diagram of the block structure with a block header and relationship with tries**

**Blocks and blockchain**

Blocks are the main building structure of a blockchain. Ethereum blocks consist of various elements, which are described as follows:

• The block header

• The transactions list

• The list of headers of ommers or uncles

The transaction list is simply a list of all transactions included in the block. Also, the list of headers of uncles is also included in the block.

**Block header:** Block headers are the most critical and detailed components of an Ethereum block. The header contains various elements, which are described in detail here:

**• Parent hash:** This is the Keccak 256-bit hash of the parent (previous) block's header.

**• Ommers hash:** This is the Keccak 256-bit hash of the list of ommers (or uncles) blocks included in the block.

**• The beneficiary:** The beneficiary field contains the 160-bit address of the recipient that will receive the mining reward once the block is successfully mined.

**• State root:** The state root field contains the Keccak 256-bit hash of the root node of the state trie. It is calculated once all transactions have been processed and finalized.

**• Transactions root:** The transaction root is the Keccak 256-bit hash of the root node of the transaction trie. The transaction trie represents the list of transactions included in the block.

**• Receipts root:** The receipts root is the Keccak 256-bit hash of the root node of the transaction receipt trie. This trie is composed of receipts of all transactions included in the block. Transaction receipts are generated after each transaction is processed and contain useful post-transaction information. More details on transaction receipts are provided in the next section.

**• Logs bloom:** The logs bloom is a bloom filter that is composed of the logger address and log topics from the log entry of each transaction receipt of the included transaction list in the block. Logging is explained in detail in the next section.

**• Difficulty:** The difficulty level of the current block.

• **Number:** The total number of all previous blocks; the genesis block is block zero.

• **Gas limit:** This field contains the value that represents the limit set on the gas consumption per block.

**• Gas used:** This field contains the total gas consumed by the transactions included in theblock.

**• Timestamp:** The timestamp is the epoch Unix time of the time of block initialization.

**Extra data:** The extra data field can be used to store arbitrary data related to the block.Only up to 32 bytes are allowed in this field.

**• Mixhash:** The mixhash field contains a 256-bit hash that, once combined with the nonce, is used to prove that adequate computational effort (Proof of Work, or PoW) has been spent in order to create this block.

**• Nonce:** Nonce is a 64-bit hash (a number) that is used to prove, in combination with the mixhash field, that adequate computational effort (PoW) has been spent in order to create this block.

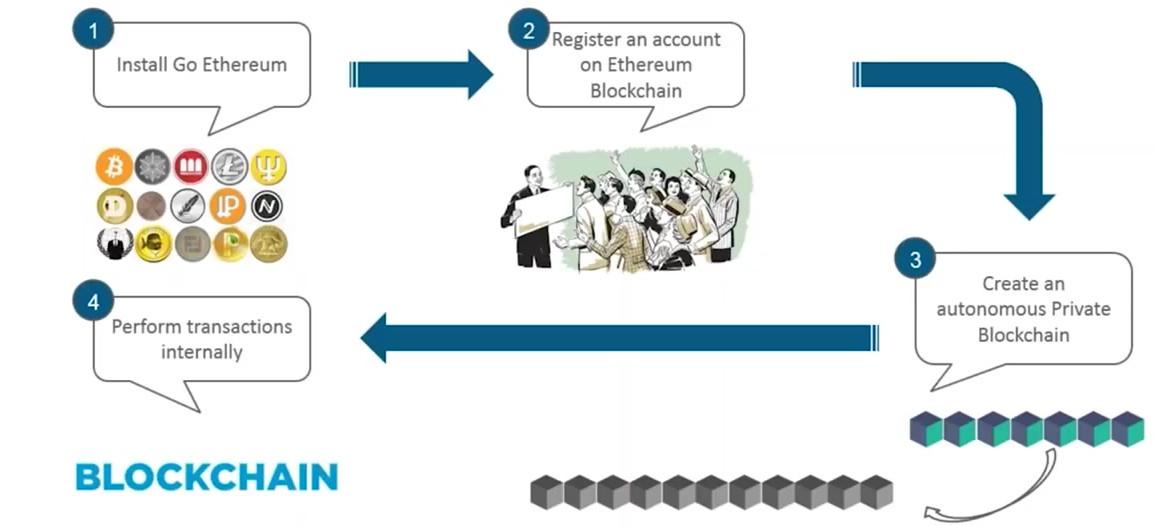
**Problem Statement:**

[1] Create auto nomous private Blockchain with rules on spending money.

[2] Make a cryptocurrency with a fixed market supply and tokens to represent real world asset values.

[3] Mine for new Ether by validating transactions.

Ethereum Blockchain Flow Diagram:



**Figure-4: Ethereum Blockchain Flow**

Cloning the Geth code

Creating a genesis block

Making rules for Blockchain

Validating and Mining Ether

**Figure-5: Ethereum Blockchain setup**

**Procedure:**

**Option-1:** Use VirtualBox with Guest OS Ubuntu 22.04

**Option-2:** Use Docker

***# Use Ubuntu 22.04 docker image***

***#Install docker and docker-compose***

$sudo apt update

$sudo apt install docker.io docker-compose

***#Check the image***

$sudo docker pull ubuntu:22.04

$sudo docker images

$sudo docker run -itd --name ubuntu-ethereum ubuntu:22.04 /bin/bash

**#Check the container**

$sudo docker ps

**#Note down the container-id**

$sudo docker attach <container-id>

Follow the steps below to setup private ethereum blockchain

[1] Install Ethereum Blockchain

Clone it from git and compile it.

$cd ~

$mkdir BTA

$cd BTA

$mkdir lab2a

$cd lab2a

Download or clone and compile ethereum code

<https://github.com/ethereum/go-ethereum>

$ git clone <https://github.com/ethereum/go-ethereum>

$cd go-ethereum

$git tag

$git checkout tags/v1.9.9 -b btaEthereumv1.9.9

#checking the branch

$git branch

#Install golang

Go Language Installation

<https://go.dev/dl/go1.18.linux-amd64.tar.gz>

Download the golang from the official website of the x86-64 tarball image source.

$wget <https://go.dev/dl/go1.18.linux-amd64.tar.gz>

$ls go1.18.linux-amd64.tar.gz

$tar -xzvf go1.18.linux-amd64.tar.gz

$sudo mv go /usr/local

# Append these lines to your .bashrc file and save it.

$cd ~

$nano .bashrc

export GOROOT=/usr/local/go

export GOPATH=$HOME/adaya/BTA/lab2a

export PATH=$GOPATH/bin:$GOROOT/bin:$PATH

$source .bashrc

#Check the go version

$go version

$cd BTA/lab2a/go-ethereum

$make all

Creating Blockchain- Genesis Block

$mkdir genesis

$ cd genesis

#create the genesis block and add these lines

nano genesisblock.json

{ "config":{},

"nonce": "0x0000000000000033",

"difficulty":"0x4000",

"alloc":{},

"mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",

"coinbase": "0x3333333333333333333333333333333333333333",

"timestamp":"0x00000000000000000000000000000000000000000000",

"parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",

"extraData":"0x",

"gasLimit":"0xffffffff"

}

**Starting the Blockchain**

#Initializing the Blockchain

$/home/adaya/BTA/lab2a/go-ethereum/build/bin/geth --datadir ~/ethereum/net3 init genenesis/genesisblock.json

#Starting the geth console

$ /home/adaya/BTA/lab2a/go-ethereum/build/bin/geth --datadir ~/ethereum/net3 --networkid 3 console

>

**# Starting Blockchain**

>personal.newAccount()

>eth.accounts

>eth.blockNumber

>miner.start()

>miner.stop()

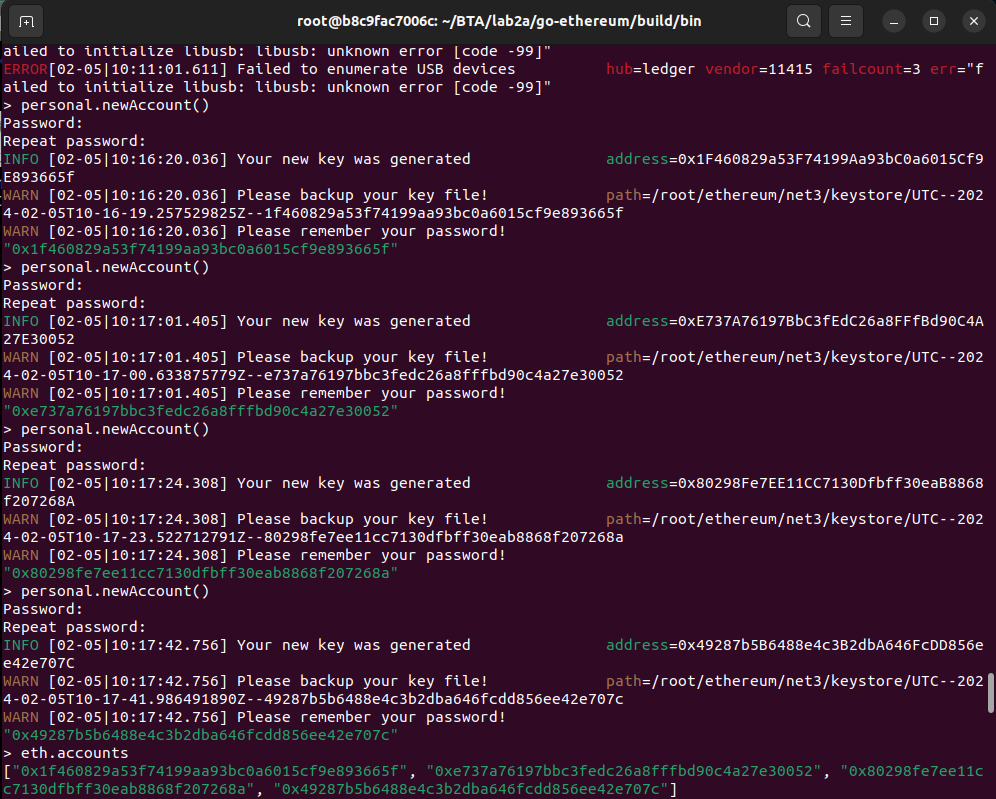
>eth.accounts

>eth.getBalance(“EAO Address”)

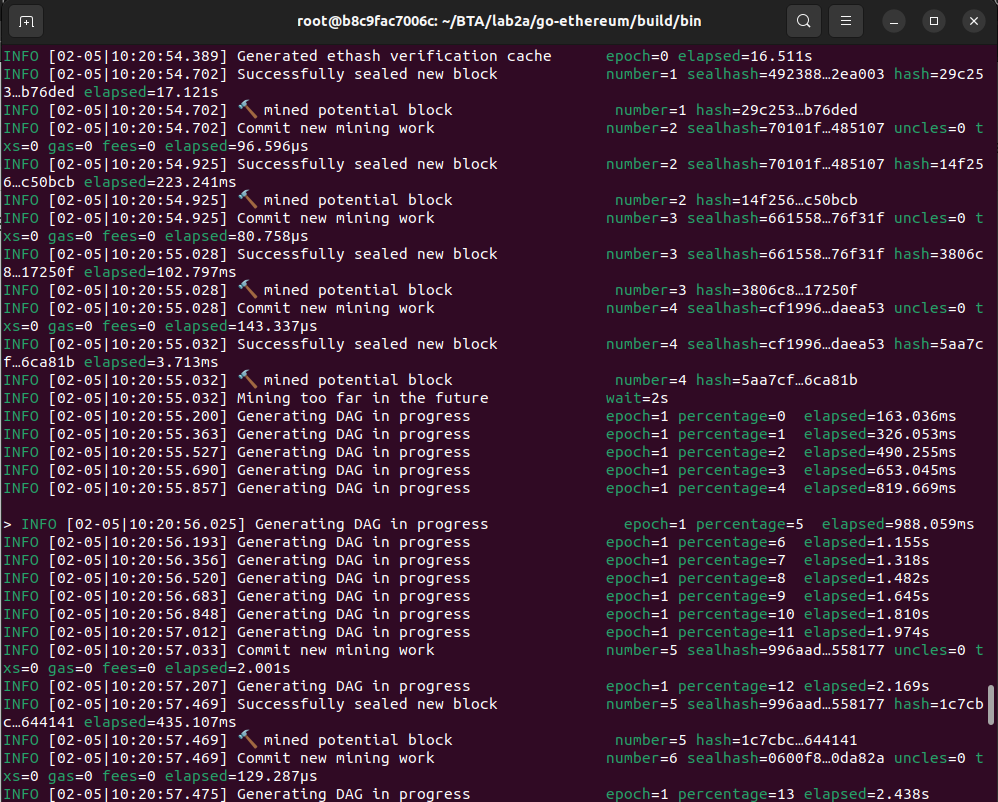
>exit

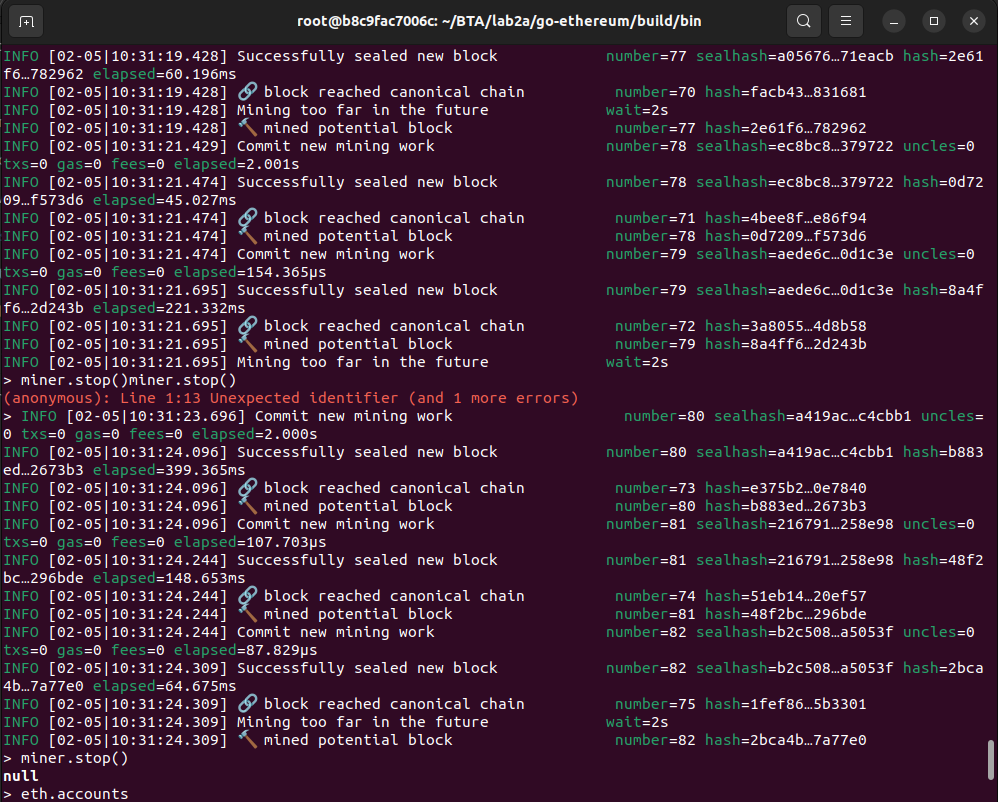
**Tasks:**

[1] Create 4 accounts and list the accounts.

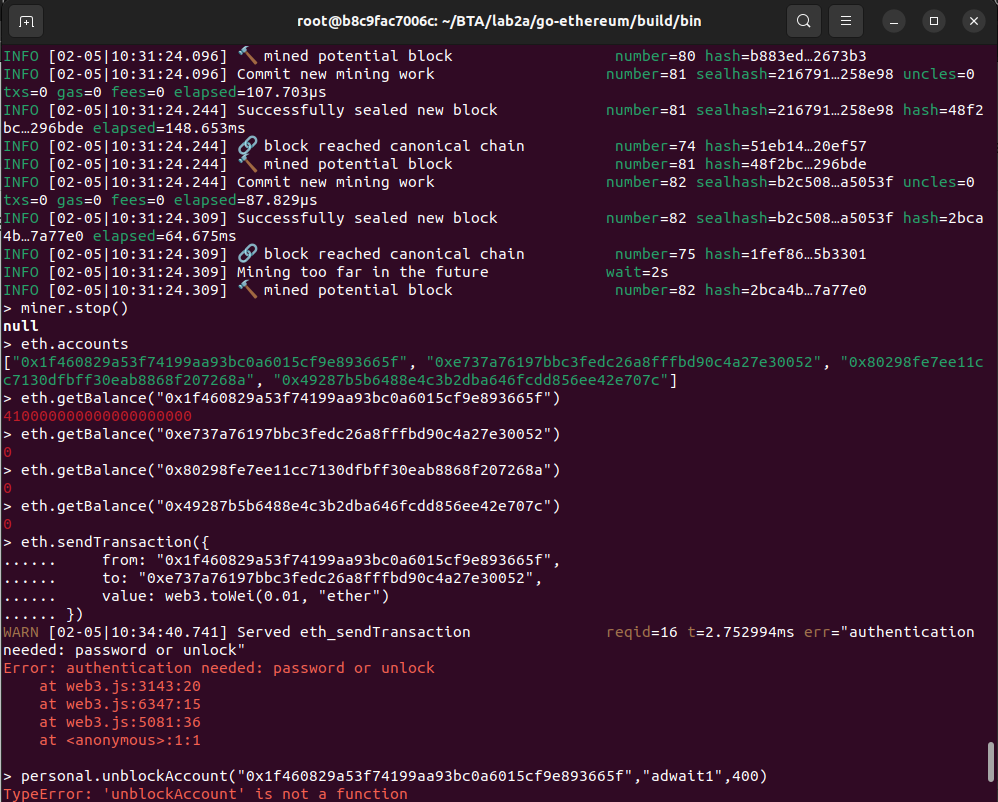


[2] Mine at least 50 blocks and check and verify it





[3] Check the balances of each EAO accounts



[4] Send Ethers and verify

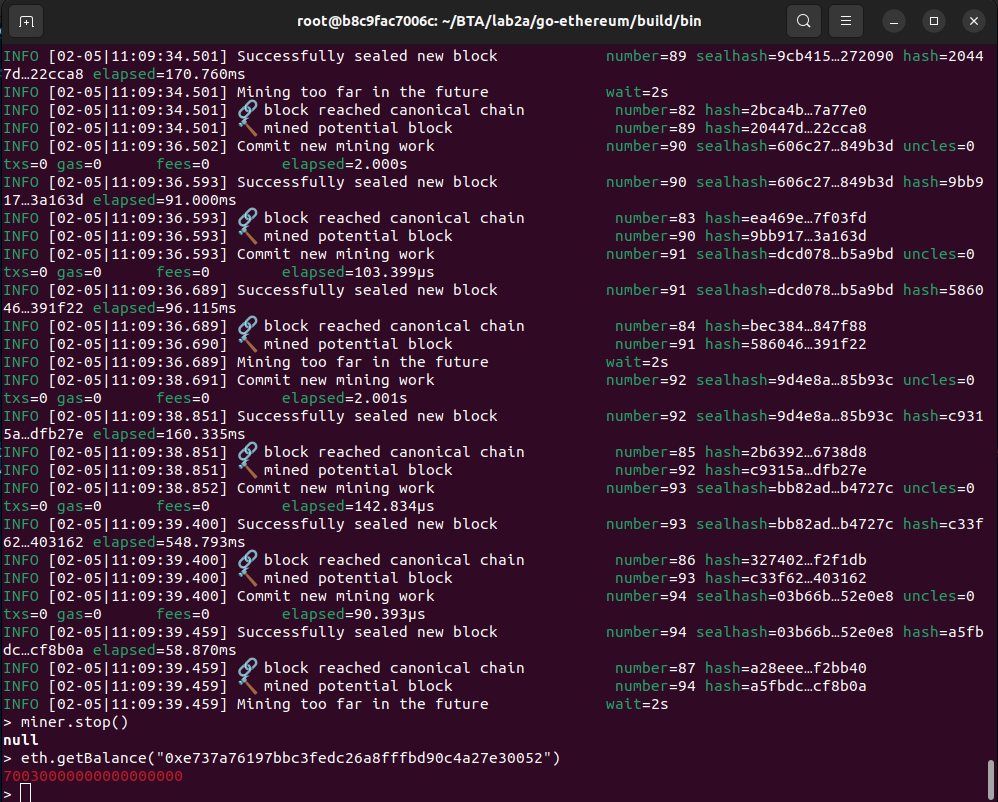
eth.sendTransaction({

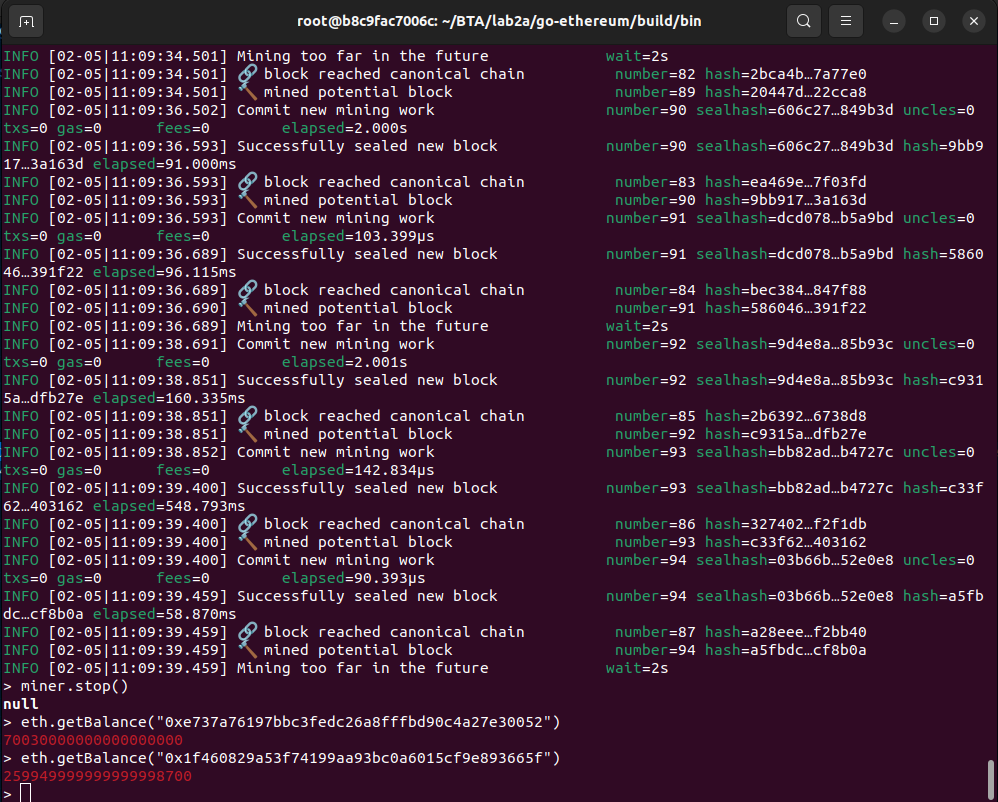
from: "0x1f460829a53f74199aa93bc0a6015cf9e893665f",

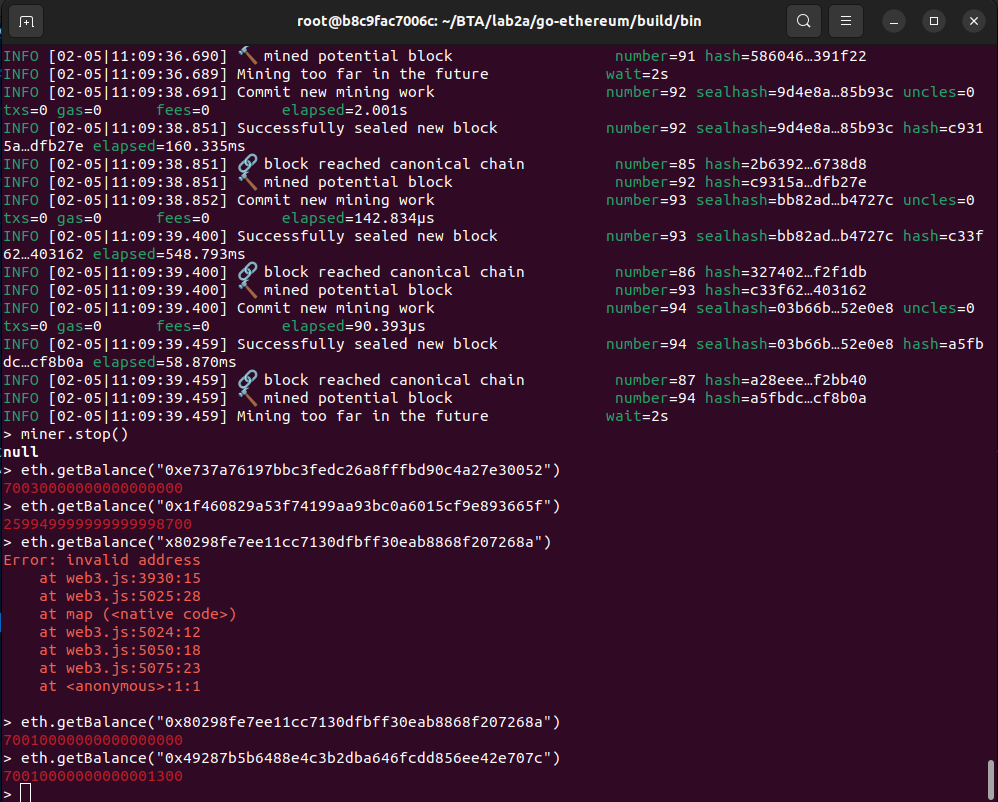
to: "0x49287b5b6488e4c3b2dba646fcdd856ee42e707c",

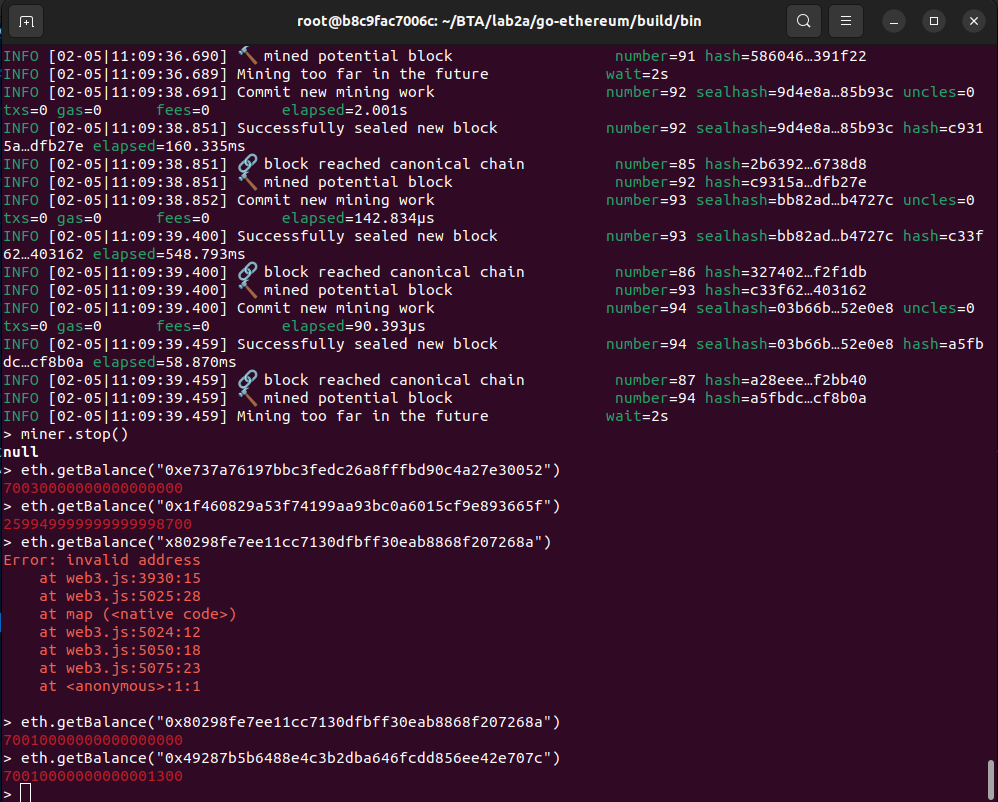
value: web3.toWei(0.01, "ether")

})

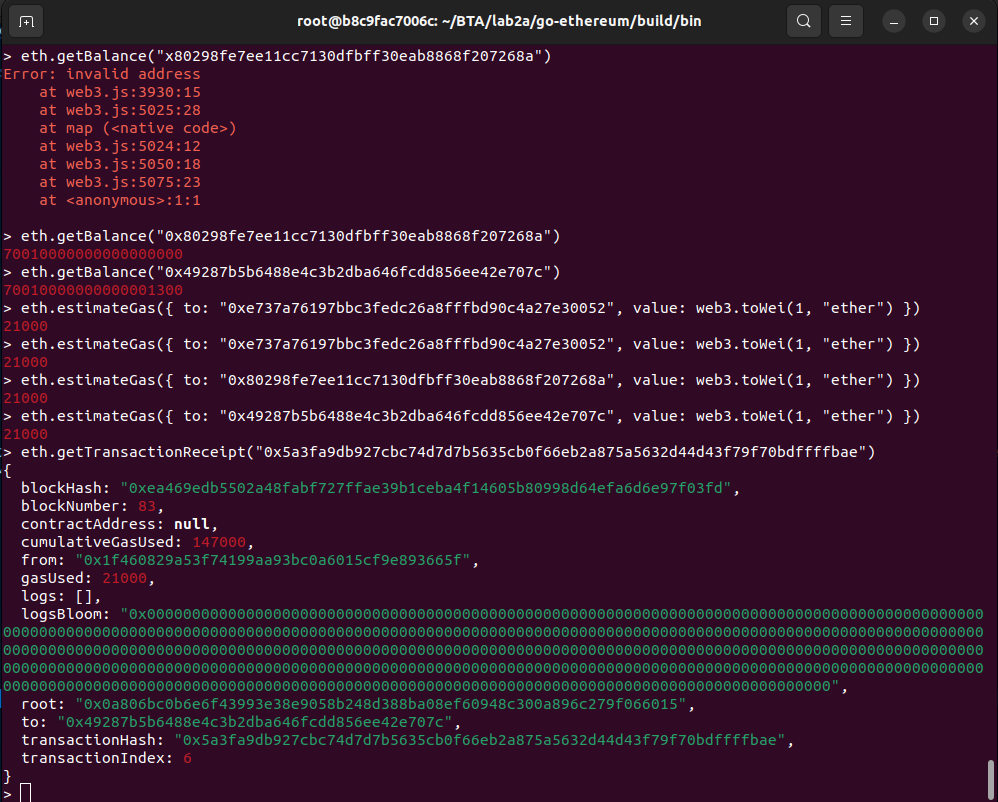


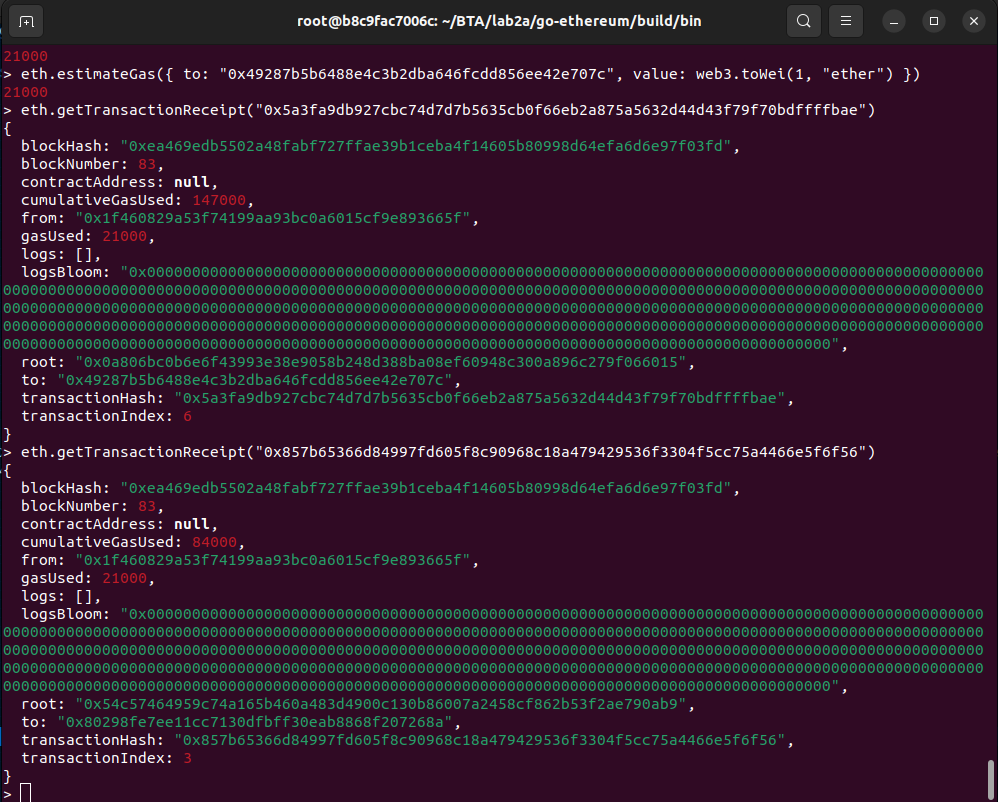






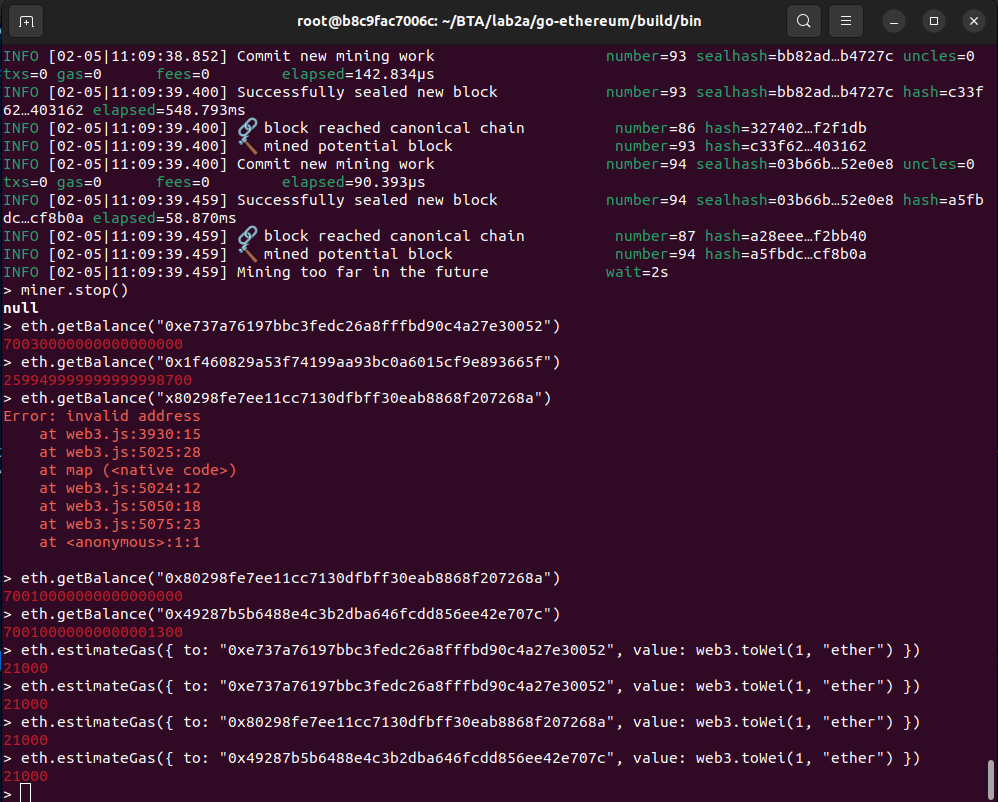
[5] Get Transaction Receipt





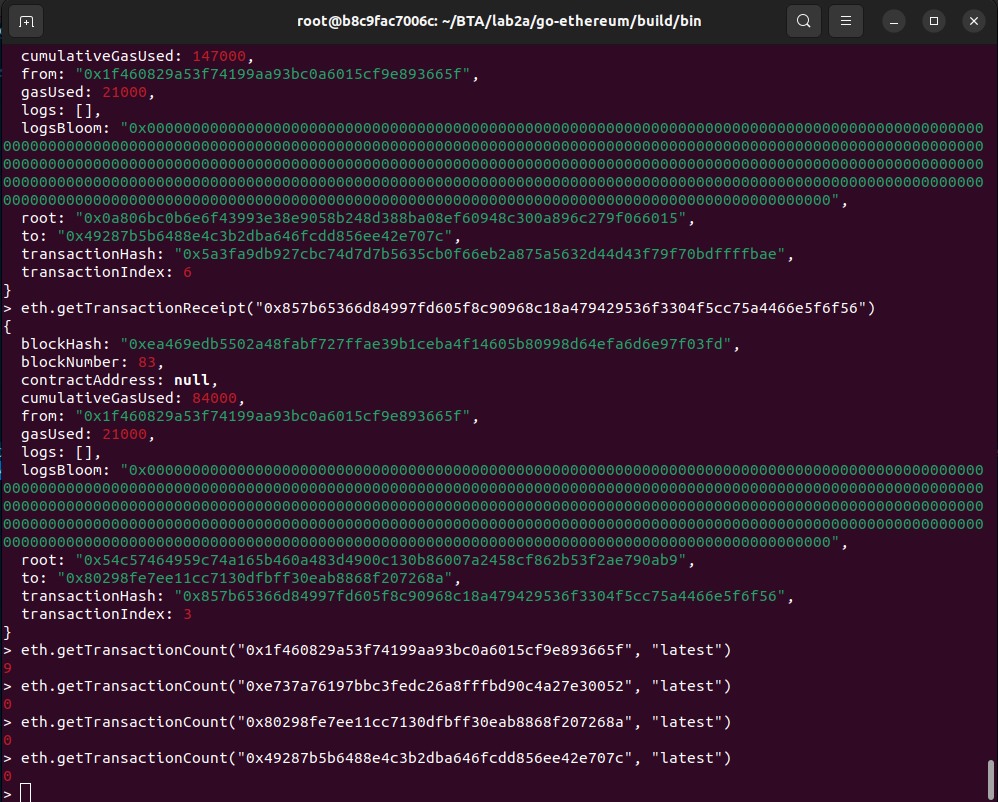
[6] Estimate Gas:

eth.estimateGas({ to: "recipient\_address", value: web3.toWei(1, "ether") })



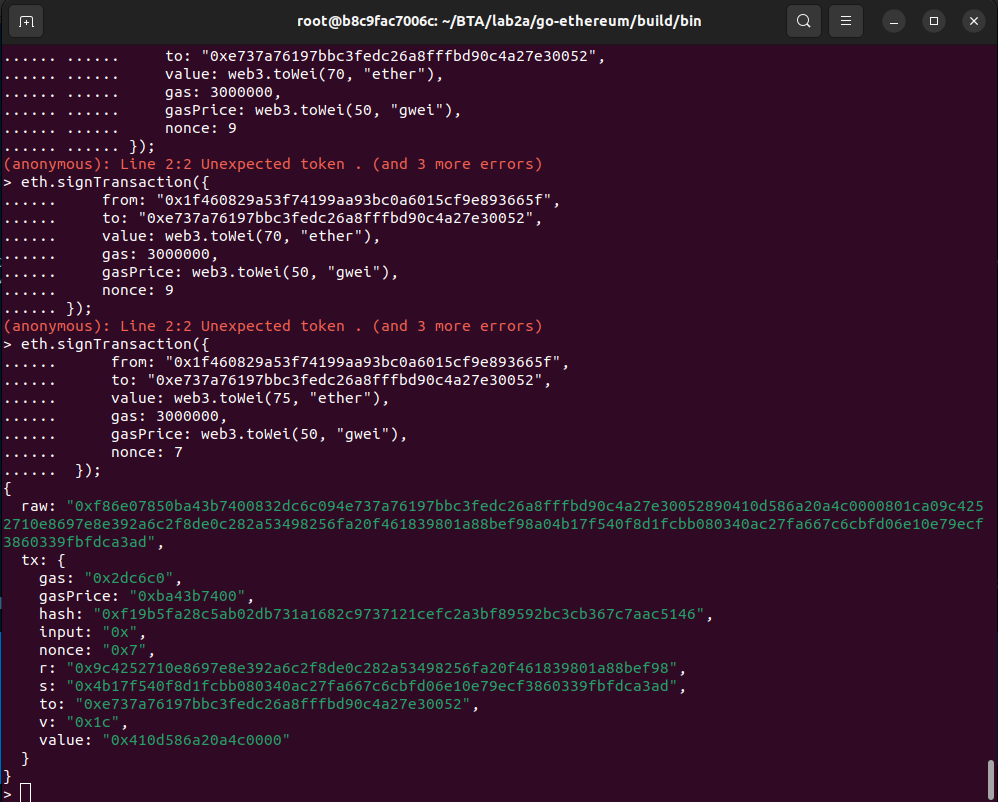
[7] Get Transaction Count from Block

eth.getTransactionCount("account\_address", "block\_number")



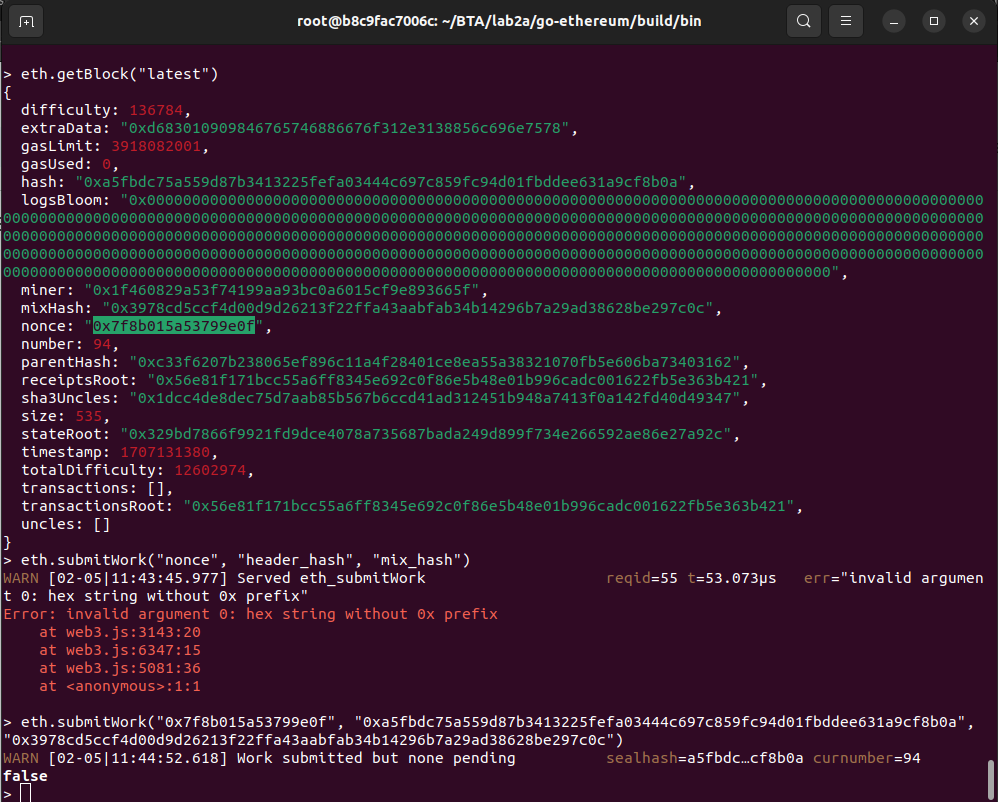
[8] Sign transaction:

eth.signTransaction({ from: "your\_account\_address", to: "recipient\_address", value: web3.toWei(1, "ether") })



[9] Submit Proof-of-Work:

eth.submitWork("nonce", "header\_hash", "mix\_hash")



[10] Check if Account is Mining

eth.getMining("account\_address")



[11] Write web3.js script connect to ethereum private blockchain

const Web3 = require('web3');

// Specify the URL of your Ethereum node

const ethereumUrl = '<http://localhost:8545>'; // Update with your Ethereum node URL

// Create a new instance of Web3, pointing to your Ethereum node

const web3 = new Web3(new Web3.providers.HttpProvider(ethereumUrl));

// Check the connection to the Ethereum node

web3.eth.net.isListening()

.then(() => console.log('Connected to Ethereum node'))

.catch(error => console.error('Error connecting to Ethereum node:', error));

// Example: Get the latest block number

web3.eth.getBlockNumber()

.then(blockNumber => console.log('Latest block number:', blockNumber))

.catch(error => console.error('Error getting latest block number:', error));

// Example: Get the coinbase address

web3.eth.getCoinbase()

.then(coinbase => console.log('Coinbase address:', coinbase))

.catch(error => console.error('Error getting coinbase address:', error));

// Example: Unlock an account (if needed)

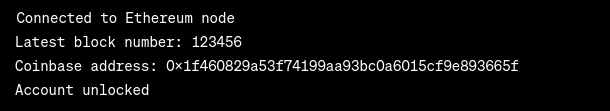
const accountAddress = '0x0x1f460829a53f74199aa93bc0a6015cf9e893665f'; // Update with your account address

const accountPassword = 'YourAccountPassword'; // Update with your account password

web3.eth.personal.unlockAccount(accountAddress, accountPassword)

.then(() => console.log('Account unlocked'))

.catch(error => console.error('Error unlocking account:', error));



**Conclusion:** From this experiment, I learned valuable insights into Ethereum node interaction using the Web3 library in a JavaScript environment. Starting with establishing a connection to the Ethereum node, I gained hands-on experience in retrieving crucial blockchain information and interacting with accounts by unlocking them. Additionally, the experiment honed my ability to adapt and modify code, demonstrated by the replacement of the account address. This practical experience solidified my understanding of fundamental concepts essential for developing decentralized applications, such as debugging, error handling, and seamless code adaptation.

**References:**

[1] Go Language Installation

<https://go.dev/dl/go1.18.linux-amd64.tar.gz>

[2] Download or clone and compile ethereum code

https://github.com/ethereum/go-ethereum

[3] Official Ethereum website

[Go Ethereum](https://geth.ethereum.org/)

[4] Mastering Blockchain Technology by Imran Bashir 4th Edition Chapter 11,12 and 13,Packt Publications