PROJECT REPORT

EXPLORING BLOCKCHAIN FEATURES USING VIRTUALIZATION CONCEPTS

By

Drishti Jain 15BCE1014

Hemanth Teja 15BCE1144

B.Aiswarya 15BCE1235

Submitted for the project component of the course

Virtualization (CSE)

under the guidance of Prof Kumar.R

ABSTRACT

Blockchain technology is experiencing prosperity as it is a very powerful and revolutionary trend. A wide variety of open source blockchains emerge recent years, which are different in architecture design or protocol parameters. When a developer wants to compare working of different blockchains, he could conduct a simulation with event simulator, or run application on physical machines. Either way will result in problems of unconvincing or costly. Container, a lightweight virtualization technique, is suitable for solving this dilemma. Tens of containers could run simultaneously in a single-core CPU computer, with each container having a running blockchain client. Hence, in this project we attempt to build a private ethereum blockchain with two clients using containers. We also use the RPC method to enable them to communicate and also get rewards for finding blocks by mining.

Introduction

Blockchain

A **blockchain** is a continuously growing list of records, called *blocks*, which are linked and secured using cryptography.Each block typically contains a cryptographic hash of the previous block,a timestamp and transaction data.By design, a blockchain is inherently resistant to modification of the data. It is "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way".For use as a distributed ledger, a blockchain is typically managed by a peer-to-peer network collectively adhering to a protocol for inter-node communication and validating new blocks. Once recorded, the data in any given block cannot be altered retroactively without the alteration of all subsequent blocks, which requires collusion of the network majority.

Blockchains are secure by design and exemplify a distributed computing system with high Byzantine fault tolerance. Decentralized consensus has therefore been achieved with a blockchain.This makes blockchains potentially suitable for the recording of events, medical records and other records management activities, such as identity management, transaction processing, food traceability or voting.

Ethereum

Ethereum is a decentralized platform that runs smart contracts: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference.These apps run on a custom built blockchain, an enormously powerful shared global infrastructure that can move value around and represent the ownership of property.This enables developers to create markets, store registries of debts or promises, move funds in accordance with instructions given long in the past (like a will or a future contracts) and many other things that have not been invented yet, all without a middleman or counterparty risk.

Dockers

Docker is a computer program that performs operating-system-level virtualization also known as containerization.Docker is primarily developed for Linux, where it uses the resource isolation features of the Linux kernel such as cgroups and kernel namespaces, and a union-capable file system such as OverlayFS and others to allow independent "containers" to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines (VMs). The Linux kernel's support for namespaces mostly isolates an application's view of the operating environment, including process trees, network, user IDs and mounted file systems, while the kernel's cgroups provide resource limiting for memory and CPU.

Containers

A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings. Available for both Linux and Windows based apps, containerized software will always run the same, regardless of the environment. Containers isolate software from its surroundings, for example differences between development and staging environments and help reduce conflicts between teams running different software on the same infrastructure. Docker containers running on a single machine share that machine's operating system kernel; they start instantly and use less compute and RAM. Images are constructed from filesystem layers and share common files. This minimizes disk usage and image downloads are much faster. Docker containers isolate applications from one another and from the underlying infrastructure. Docker provides the strongest default isolation to limit app issues to a single container instead of the entire machine. Containers and virtual machines have similar resource isolation and allocation benefits, but function differently because containers virtualize the operating system instead of hardware. Containers are more portable and efficient.

Public vs Private blockchain

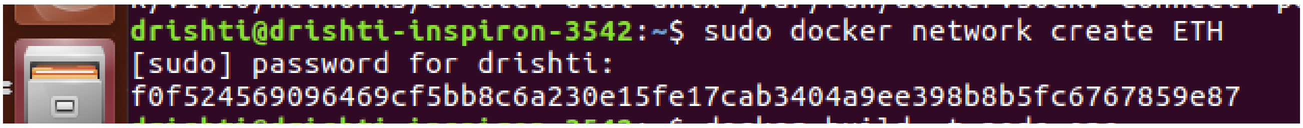
The public blockchain is open to anyone who wants to deploy smart contracts and have their executions performed by public mining nodes. Bitcoin is one of the largest public blockchain networks today. As such, there is limited privacy in the public blockchain. Mining nodes in the public blockchain requires a substantial amount of computational power to maintain the distributed ledger at a large scale. In the Ethereum public blockchain, smart contract codes can be viewed openly.

A private blockchain can be set up within the safety confines of a private network within an organization. Hence, nodes participating in transactions are authenticated and authorized machines within the organizational network.A fully private blockchain is a blockchain where write permissions are kept centralized to one organization. Read permissions may be public or restricted to an arbitrary extent.

IMPLEMENTATION

1. To allow nodes to *talk* to each other, we must create the network between containers.

```
$ sudo docker network create ETH
```



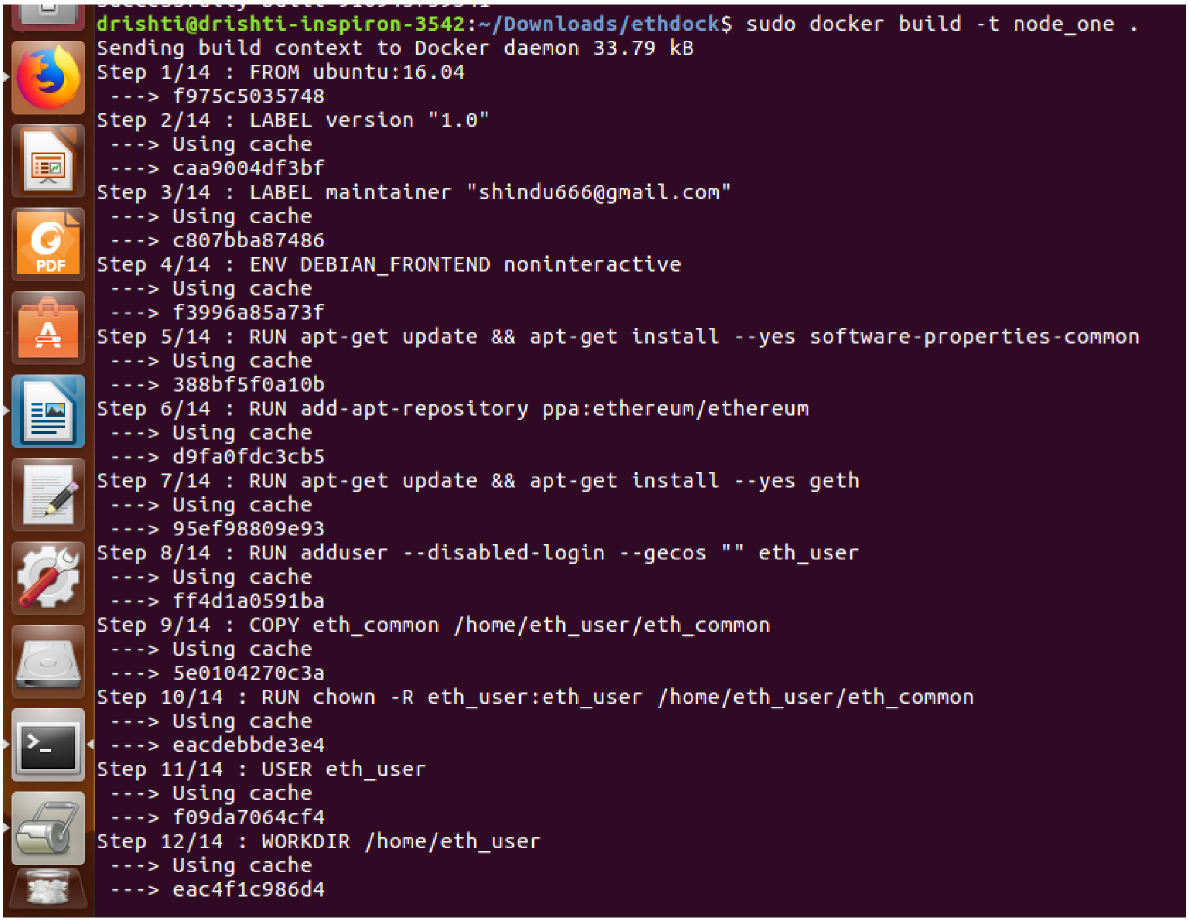
```
drishti@drishti-inspiron-3542:~$ sudo docker network create ETH
[sudo] password for drishti:
f0f524569096469cf5bb8c6a230e15fe17cab3404a9ee398b8b5fc6767859e87
```

2. build two containers

```
$ sudo docker build -t node_one .
```

And similarly

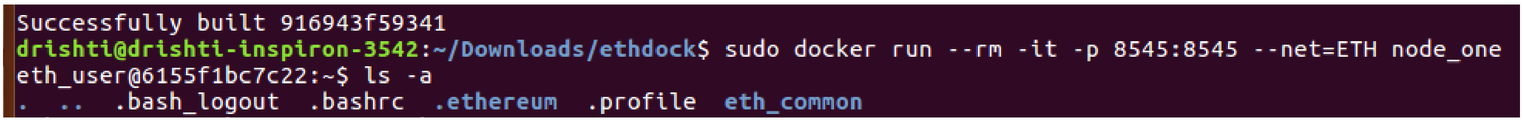
```
$ sudo docker build -t node_two .
```



```
drishti@drishti-inspiron-3542:~/Downloads/ethdock$ sudo docker build -t node_one .
Sending build context to Docker daemon 33.79 kB
Step 1/14 : FROM ubuntu:16.04
 ---> f975c5035748
Step 2/14 : LABEL version "1.0"
 ---> Using cache
 ---> caa9004df3bf
Step 3/14 : LABEL maintainer "shindu666@gmail.com"
 ---> Using cache
 ---> c807bba87486
Step 4/14 : ENV DEBIAN_FRONTEND noninteractive
 ---> Using cache
 ---> f3996a85a73f
Step 5/14 : RUN apt-get update && apt-get install --yes software-properties-common
 ---> Using cache
 ---> 388bf5f0a10b
Step 6/14 : RUN add-apt-repository ppa:ethereum/ethereum
 ---> Using cache
 ---> d9fa0fdc3cb5
Step 7/14 : RUN apt-get update && apt-get install --yes geth
 ---> Using cache
 ---> 95ef98809e93
Step 8/14 : RUN adduser --disabled-login --gecos "" eth_user
 ---> Using cache
 ---> ff4d1a0591ba
Step 9/14 : COPY eth_common /home/eth_user/eth_common
 ---> Using cache
 ---> 5e0104270c3a
Step 10/14 : RUN chown -R eth_user:eth_user /home/eth_user/eth_common
 ---> Using cache
 ---> eacdebbde3e4
Step 11/14 : USER eth_user
 ---> Using cache
 ---> f09da7064cf4
Step 12/14 : WORKDIR /home/eth_user
 ---> Using cache
 ---> eac4f1c986d4
```

3. Run

```
$ sudo docker run --rm -it -p 8545:8545 --net=ETH node_one
$ ls -a
```



```
Successfully built 916943f59341
drishti@drishti-inspiron-3542:~/Downloads/ethdock$ sudo docker run --rm -it -p 8545:8545 --net=ETH node_one
eth_user@6155f1bc7c22:~$ ls -a
.  ..  .bash_logout  .bashrc  .ethereum  .profile  eth_common
```

and similarly for node_two with (-p 8546:8546).

`docker run` command line options:

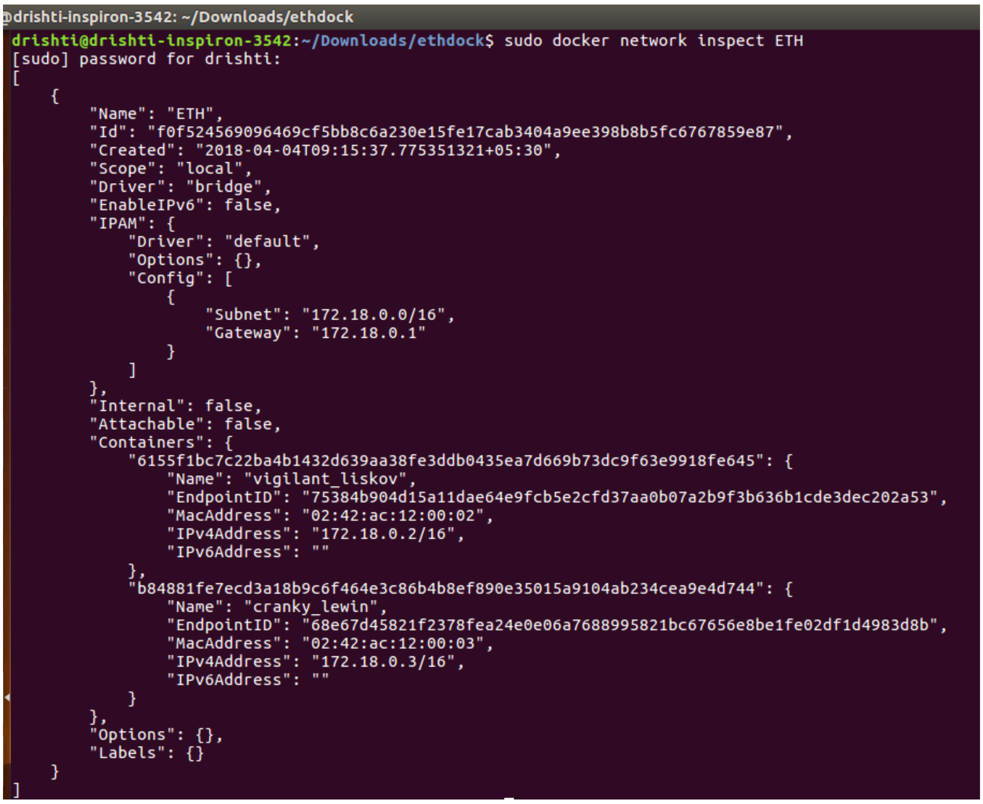
`-p 8545:8545` in the node_one expose the default RPC port of geth.

`-p 8546:8546` in the node_two expose port, which will be used later in geth.

– `net=ETH` is a custom docker network, to allow containers communicate each other.

4. Run the command to get the Ipv4

```
$ sudo docker network inspect ETH
```



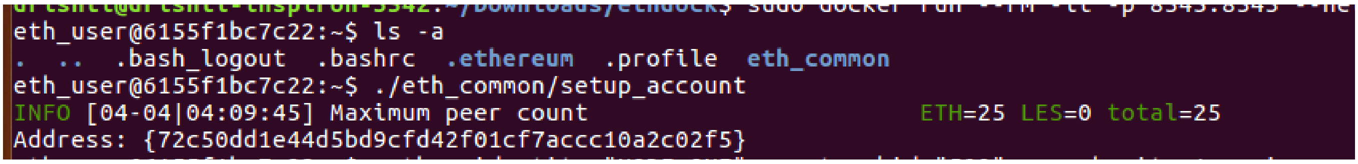
@drishti-inspiron-3542: ~/Downloads/ethdock

```
drishti@drishti-inspiron-3542:~/Downloads/ethdock$ sudo docker network inspect ETH
[sudo] password for drishti:
[
    {
        "Name": "ETH",
        "Id": "f0f524569096469cf5bb8c6a230e15fe17cab3404a9ee398b8b5fc6767859e87",
        "Created": "2018-04-04T09:15:37.775351321+05:30",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Containers": {
            "6155f1bc7c22ba4b1432d639aa38fe3ddb0435ea7d669b73dc9f63e9918fe645": {
                "Name": "vigilant_liskov",
                "EndpointID": "75384b904d15a11dae64e9fcb5e2cfd37aa0b07a2b9f3b636b1cde3dec202a53",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            },
            "b84881fe7ecd3a18b9c6f464e3c86b4b8ef890e35015a9104ab234cea9e4d744": {
                "Name": "cranky_lewin",
                "EndpointID": "68e67d45821f2378fea24e0e06a7688995821bc67656e8be1fe02df1d4983d8b",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

5. Now we have two docker containers connected each other. Need to generate coinbase account, in order to mine ether we must provide an address to receive reward for found blocks.

Run the setup_account from the console of node_one

```
./eth_common/setup_account
```

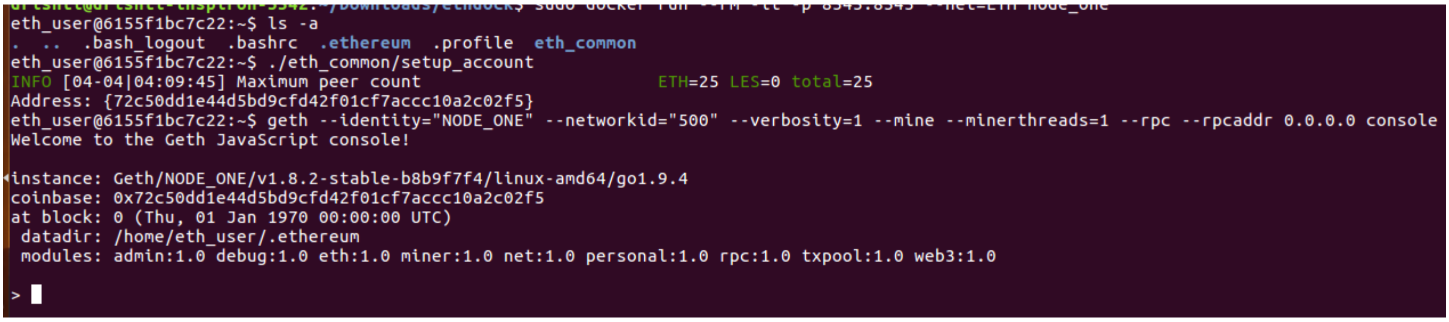


```
eth_user@6155f1bc7c22:~$ ls -a
.  ..  .bash_logout  .bashrc  .ethereum  .profile  eth_common
eth_user@6155f1bc7c22:~$ ./eth_common/setup_account
INFO [04-04|04:09:45] Maximum peer count                       ETH=25 LES=0 total=25
Address: {72c50dd1e44d5bd9cfd42f01cf7accc10a2c02f5}
```

Similarly for node_two in a new terminal

6. Time to start minning. On the console of node_one run the command

```
$ geth --identity="NODE_ONE" --networkid="500" --verbosity=1
--mine --minerthreads=1 --rpc --rpcaddr 0.0.0.0 console
```

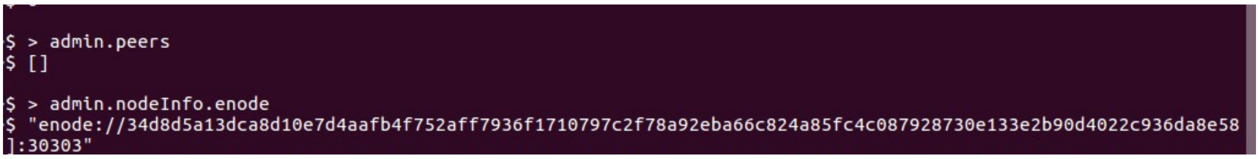


```
eth_user@6155f1bc7c22:~$ ls -a
.  ..  .bash_logout  .bashrc  .ethereum  .profile  eth_common
eth_user@6155f1bc7c22:~$ ./eth_common/setup_account
INFO [04-04|04:09:45] Maximum peer count                       ETH=25 LES=0 total=25
Address: {72c50dd1e44d5bd9cfd42f01cf7accc10a2c02f5}
eth_user@6155f1bc7c22:~$ geth --identity="NODE_ONE" --networkid="500" --verbosity=1 --mine --minerthreads=1 --rpc --rpcaddr 0.0.0.0 console
Welcome to the Geth JavaScript console!

instance: Geth/NODE_ONE/v1.8.2-stable-b8b9f7f4/linux-amd64/go1.9.4
coinbase: 0x72c50dd1e44d5bd9cfd42f01cf7accc10a2c02f5
at block: 0 (Thu, 01 Jan 1970 00:00:00 UTC)
 datadir: /home/eth_user/.ethereum
 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

> 
```

```
command line options to geth:
 — identity must be unique identifier of the node
— networkid must be the same on the both nodes
—verbosity there are many levels 0=silent, 1=error, 2=warn,
3=info, 4=core, 5=debug, 6=detail (default: 3)
— mine enable minning
— rpc enable RPC
— rpcport=8546 change the (default: 8545) RPC port in order to
reach container from the host machine
```

Similarly, run the command for node_two in its console.

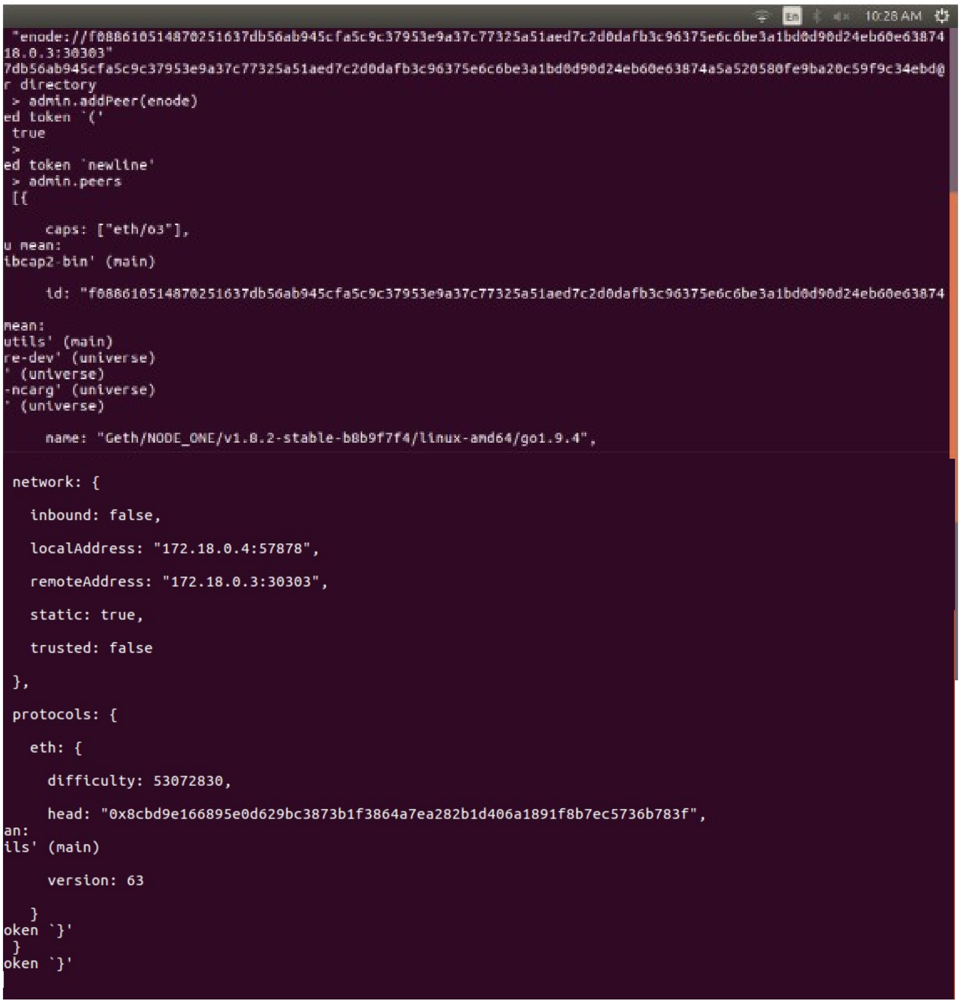
7. check peers, our nodes must detect each other. Use >admin.peers to get info on peers and >admin.nodeInfo.enode to get enode address



```
$ > admin.peers
$ []

$ > admin.nodeInfo.enode
$ "enode://34d8d5a13dca8d10e7d4aafb4f752aff7936f1710797c2f78a92eba66c824a85fc4c087928730e133e2b90d4022c936da8e58
]:30303"
```

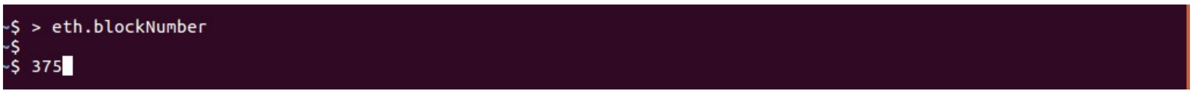
8. Peers are empty, in order to add peers we must provide a full enode url to the `admin.addPeer()` method, to get the enode url run command

`[::]` —means localhost, that’s the enode url of node_one.



```
 "enode://f088610514870251637db56ab945cfa5c9c37953e9a37c77325a51aed7c2d0dafb3c96375e6c6be3a1bd0d90d24eb60e63874
18.0.3:30303"
7db56ab945cfa5c9c37953e9a37c77325a51aed7c2d0dafb3c96375e6c6be3a1bd0d90d24eb60e63874a5a520580fe9ba20c59f9c34ebd@
r directory
 > admin.addPeer(enode)
ed token `('
 true
 >
ed token `newline'
 > admin.peers
 [{

     caps: ["eth/63"],
u mean:
ibcap2-bin' (main)

     id: "f088610514870251637db56ab945cfa5c9c37953e9a37c77325a51aed7c2d0dafb3c96375e6c6be3a1bd0d90d24eb60e63874

mean:
utils' (main)
re-dev' (universe)
' (universe)
-ncarg' (universe)
' (universe)

     name: "Geth/NODE_ONE/v1.8.2-stable-b8b9f7f4/linux-amd64/go1.9.4",

 network: {

   inbound: false,

   localAddress: "172.18.0.4:57878",

   remoteAddress: "172.18.0.3:30303",

   static: true,

   trusted: false

 },

 protocols: {

   eth: {

     difficulty: 53072830,

     head: "0x8cbd9e166895e0d629bc3873b1f3864a7ea282b1d406a1891f8b7ec5736b783f",
an:
ils' (main)

     version: 63

   }
oken `}'
 }
oken `}'
```

9. nodes are seeing each other.Now, check the block number

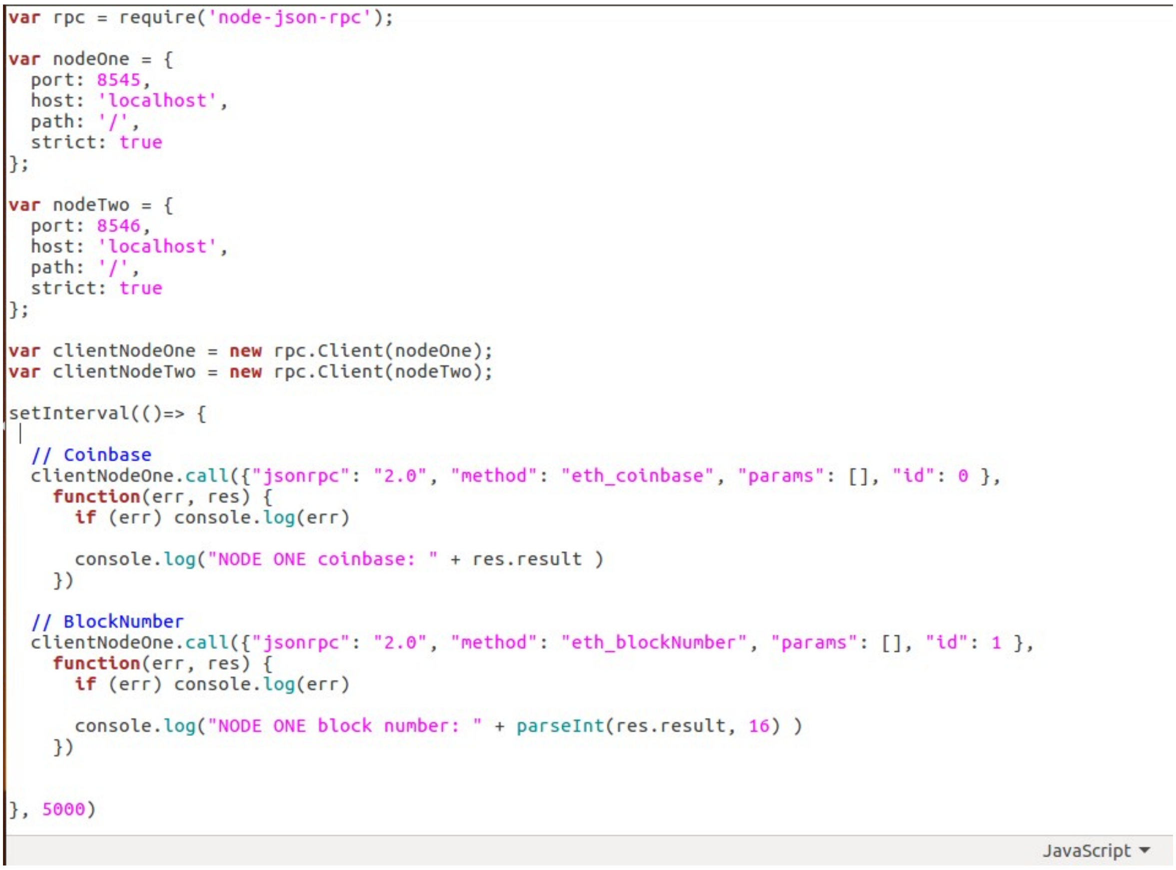


```
~$ > eth.blockNumber
~$
~$ 375
```

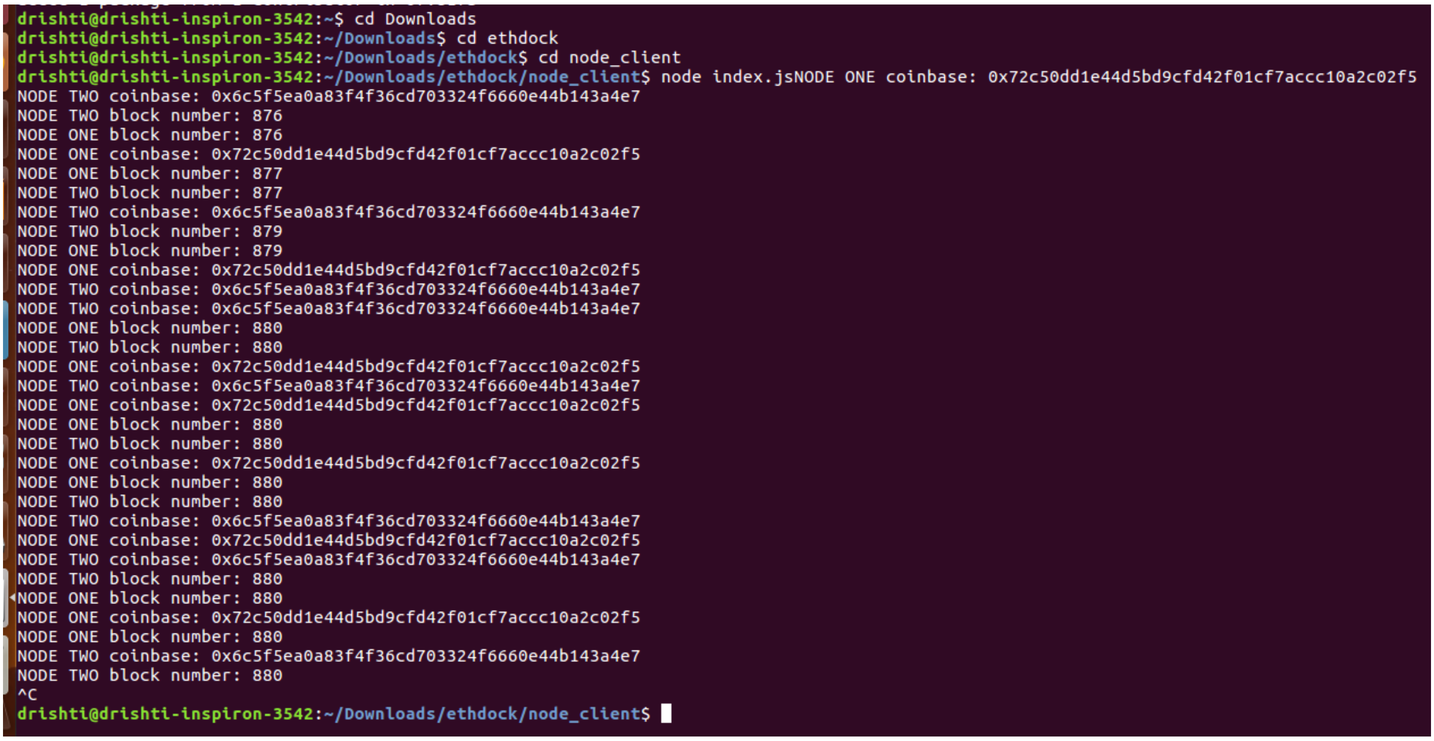
This means that we are successful in mining.

EXPERIMENTAL DETAILS

Now we will try to run the NodeJS client script and check the output



```javascript
var rpc = require('node-json-rpc');

var nodeOne = {
  port: 8545,
  host: 'localhost',
  path: '/',
  strict: true
};

var nodeTwo = {
  port: 8546,
  host: 'localhost',
  path: '/',
  strict: true
};

var clientNodeOne = new rpc.Client(nodeOne);
var clientNodeTwo = new rpc.Client(nodeTwo);

setInterval(()=> {

  // Coinbase
  clientNodeOne.call({"jsonrpc": "2.0", "method": "eth_coinbase", "params": [], "id": 0 },
    function(err, res) {
      if (err) console.log(err)

      console.log("NODE ONE coinbase: " + res.result )
    })

  // BlockNumber
  clientNodeOne.call({"jsonrpc": "2.0", "method": "eth_blockNumber", "params": [], "id": 1 },
    function(err, res) {
      if (err) console.log(err)

      console.log("NODE ONE block number: " + parseInt(res.result, 16) )
    })


}, 5000)
```



```
drishti@drishti-inspiron-3542:~$ cd Downloads
drishti@drishti-inspiron-3542:~/Downloads$ cd ethdock
drishti@drishti-inspiron-3542:~/Downloads/ethdock$ cd node_client
drishti@drishti-inspiron-3542:~/Downloads/ethdock/node_client$ node index.jsNODE ONE coinbase: 0x72c50dd1e44d5bd9cfd42f01cf7accc10a2c02f5
NODE TWO coinbase: 0x6c5f5ea0a83f4f36cd703324f6660e44b143a4e7
NODE TWO block number: 876
NODE ONE block number: 876
NODE ONE coinbase: 0x72c50dd1e44d5bd9cfd42f01cf7accc10a2c02f5
NODE ONE block number: 877
NODE TWO block number: 877
NODE TWO coinbase: 0x6c5f5ea0a83f4f36cd703324f6660e44b143a4e7
NODE TWO block number: 879
NODE ONE block number: 879
NODE ONE coinbase: 0x72c50dd1e44d5bd9cfd42f01cf7accc10a2c02f5
NODE TWO coinbase: 0x6c5f5ea0a83f4f36cd703324f6660e44b143a4e7
NODE TWO coinbase: 0x6c5f5ea0a83f4f36cd703324f6660e44b143a4e7
NODE ONE block number: 880
NODE TWO block number: 880
NODE ONE coinbase: 0x72c50dd1e44d5bd9cfd42f01cf7accc10a2c02f5
NODE TWO coinbase: 0x6c5f5ea0a83f4f36cd703324f6660e44b143a4e7
NODE ONE coinbase: 0x72c50dd1e44d5bd9cfd42f01cf7accc10a2c02f5
NODE ONE block number: 880
NODE TWO block number: 880
NODE ONE coinbase: 0x72c50dd1e44d5bd9cfd42f01cf7accc10a2c02f5
NODE ONE block number: 880
NODE TWO block number: 880
NODE TWO coinbase: 0x6c5f5ea0a83f4f36cd703324f6660e44b143a4e7
NODE ONE coinbase: 0x72c50dd1e44d5bd9cfd42f01cf7accc10a2c02f5
NODE TWO coinbase: 0x6c5f5ea0a83f4f36cd703324f6660e44b143a4e7
NODE TWO block number: 880
NODE ONE block number: 880
NODE ONE coinbase: 0x72c50dd1e44d5bd9cfd42f01cf7accc10a2c02f5
NODE ONE block number: 880
NODE TWO coinbase: 0x6c5f5ea0a83f4f36cd703324f6660e44b143a4e7
NODE TWO block number: 880
^C
drishti@drishti-inspiron-3542:~/Downloads/ethdock/node_client$
```

CONCLUSION

We have successfully created two containers on a docker and implemented a private ethereum blockchain. We have also enabled the nodes to mine and get rewards as the block numbers increase. Finally, the Node-RPC communication is successful as the nodes are able to communicate with our cluster using JSON-RPC protocol.

APPENDIX

Code

- <u>Dockerfile</u>

```
FROM ubuntu:16.04

LABEL version="1.0"
LABEL maintainer="djjain07@gmail.com"

ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update && apt-get install --yes software-properties-common
RUN add-apt-repository ppa:ethereum/ethereum
RUN apt-get update && apt-get install --yes geth

RUN adduser --disabled-login --gecos "" eth_user

COPY eth_common /home/eth_user/eth_common
RUN chown -R eth_user:eth_user /home/eth_user/eth_common
USER eth_user
WORKDIR /home/eth_user

RUN geth init eth_common/genesis.json

ENTRYPOINT bash
```

- <u>NodeJS client</u>

```
var rpc = require('node-json-rpc');

var nodeOne = {
  port: 8545,
  host: 'localhost',
  path: '/',
  strict: true
};
var nodeTwo = {
  port: 8546,
  host: 'localhost',
  path: '/',
  strict: true
};

var clientNodeOne = new rpc.Client(nodeOne);
var clientNodeTwo = new rpc.Client(nodeTwo);

setInterval(()=> {
  // Coinbase
  clientNodeOne.call({"jsonrpc": "2.0", "method": "eth_coinbase",
"params": [], "id": 0 },
    function(err, res) {
      if (err) console.log(err)
      console.log("NODE ONE coinbase: " + res.result )
    })
   // blockNumber
  clientNodeTwo.call({"jsonrpc": "2.0", "method": "eth_blockNumber",
"params": [], "id": 1 },
    function(err, res) {
      if (err) console.log(err)
      console.log("NODE TWO block number: " + parseInt(res.result, 16) )
    })

  // Coinbase
  clientNodeOne.call({"jsonrpc": "2.0", "method": "eth_coinbase",
"params": [], "id": 0 },
    function(err, res) {
      if (err) console.log(err)
      console.log("NODE ONE coinbase: " + res.result )
    })

  // blockNumber
  clientNodeTwo.call({"jsonrpc": "2.0", "method": "eth_blockNumber",
"params": [], "id": 1 },
    function(err, res) {
      if (err) console.log(err)
      console.log("NODE TWO block number: " + parseInt(res.result, 16) )
    })
}, 5000) // 5 seconds timeout
```

- Genesis.json

```
{
  "alloc": {
  },
  "config": {
    "chainId": 15,
    "homesteadBlock": 0,
    "eip155Block": 0,
    "eip158Block": 0
  },
  "nonce": "0x000000000000002a",
  "difficulty": "0x020000",
  "mixhash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "timestamp": "0x00",
  "parentHash":
"0x0000000000000000000000000000000000000000000000000000000000000000",
  "extraData": "0x",
  "gasLimit": "0x2fefd8"
}
```