Installing ethereum on Rpi

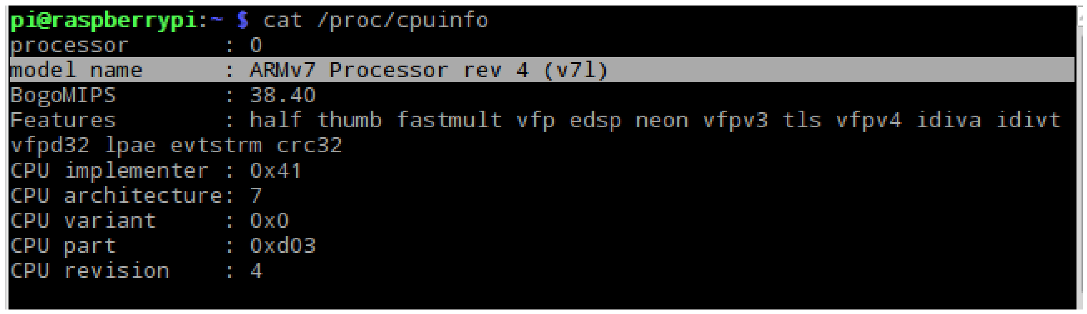
What is Geth?

Geth is a multipurpose command line tool that runs a full Ethereum node implemented in Go. It offers three interfaces: the command line subcommands and options, a Json-rpc server and an interactive console. http://www.talkcrypto.org/blog/2018/01/23/what-is-geth/

--On raspberry pi--

Check type of CPU of your Rpi

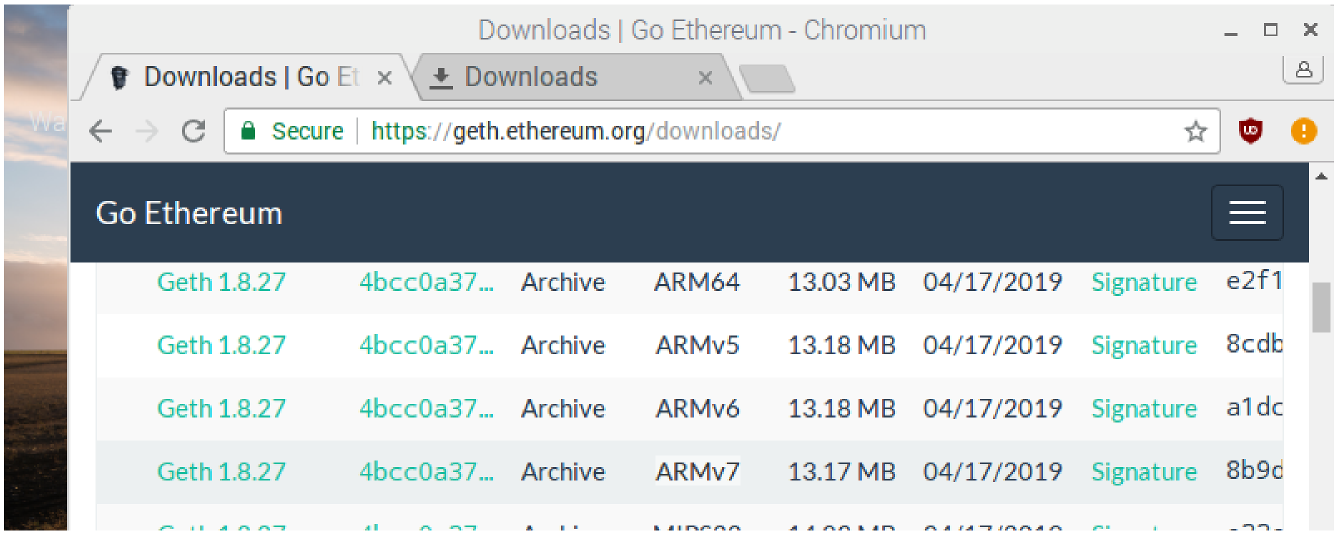
cat /proc/cpuinfo



```
pi@raspberrypi:~ $ cat /proc/cpuinfo
processor       : 0
model name      : ARMv7 Processor rev 4 (v7l)
BogoMIPS        : 38.40
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt
vfpd32 lpae evtstrm crc32
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xd03
CPU revision    : 4
```

Now to downloads page of geth/downloads at https://geth.ethereum.org/downloads/

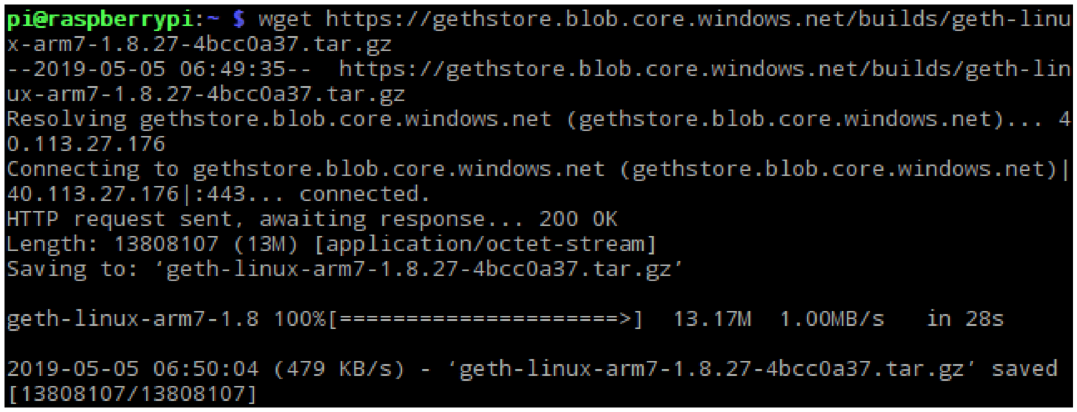
Search for the appropriate CPU version and copy the link address by right clicking on geth


Downloads | Go Ethereum - Chromium
Downloads | Go Et
Downloads
Secure
https://geth.ethereum.org/downloads/
Go Ethereum

Geth 1.8.27	4bcc0a37...	Archive	ARM64	13.03 MB	04/17/2019	Signature	e2f1
Geth 1.8.27	4bcc0a37...	Archive	ARMv5	13.18 MB	04/17/2019	Signature	8cdb
Geth 1.8.27	4bcc0a37...	Archive	ARMv6	13.18 MB	04/17/2019	Signature	a1dc
Geth 1.8.27	4bcc0a37...	Archive	ARMv7	13.17 MB	04/17/2019	Signature	8b9d

Run the following code:

wget <copied link>



```
pi@raspberrypi:~ $ wget https://gethstore.blob.core.windows.net/builds/geth-linu
x-arm7-1.8.27-4bcc0a37.tar.gz
--2019-05-05 06:49:35--  https://gethstore.blob.core.windows.net/builds/geth-lin
ux-arm7-1.8.27-4bcc0a37.tar.gz
Resolving gethstore.blob.core.windows.net (gethstore.blob.core.windows.net)... 4
0.113.27.176
Connecting to gethstore.blob.core.windows.net (gethstore.blob.core.windows.net)|
40.113.27.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13808107 (13M) [application/octet-stream]
Saving to: ‘geth-linux-arm7-1.8.27-4bcc0a37.tar.gz’

geth-linux-arm7-1.8 100%[====================>]  13.17M  1.00MB/s   in 28s

2019-05-05 06:50:04 (479 KB/s) - ‘geth-linux-arm7-1.8.27-4bcc0a37.tar.gz’ saved
[13808107/13808107]
```

1. Unzip and install

   tar zxvf <geth version saved>

2. Copy the Geth application to *usr*/local/bin

   cd <geth version>

   sudo cp geth /usr/local/bin

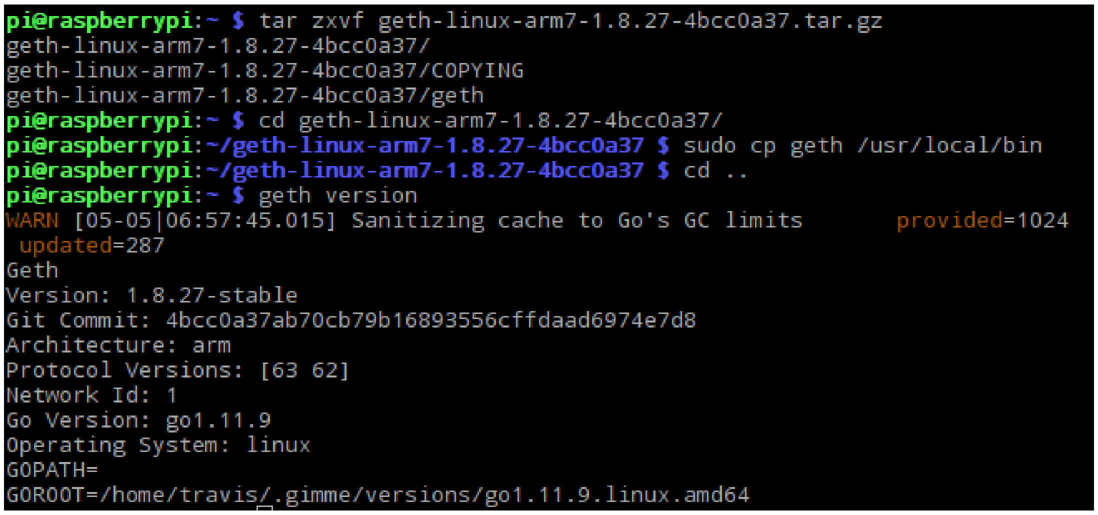
3. Now check if geth has been installed successfully, by checking the version of the geth application installed

   geth version

4. Now, ethereum is installed with success.

5. Run Geth

   geth



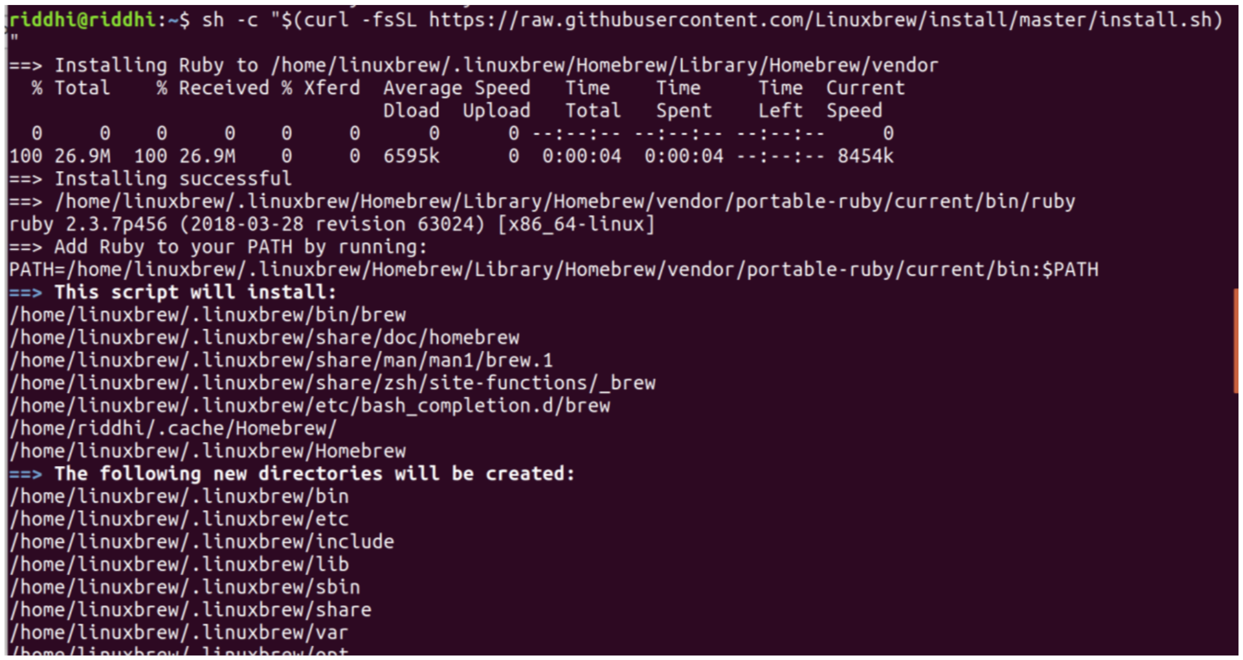
```
pi@raspberrypi:~ $ tar zxvf geth-linux-arm7-1.8.27-4bcc0a37.tar.gz
geth-linux-arm7-1.8.27-4bcc0a37/
geth-linux-arm7-1.8.27-4bcc0a37/COPYING
geth-linux-arm7-1.8.27-4bcc0a37/geth
pi@raspberrypi:~ $ cd geth-linux-arm7-1.8.27-4bcc0a37/
pi@raspberrypi:~/geth-linux-arm7-1.8.27-4bcc0a37 $ sudo cp geth /usr/local/bin
pi@raspberrypi:~/geth-linux-arm7-1.8.27-4bcc0a37 $ cd ..
pi@raspberrypi:~ $ geth version
WARN [05-05|06:57:45.015] Sanitizing cache to Go's GC limits       provided=1024
 updated=287
Geth
Version: 1.8.27-stable
Git Commit: 4bcc0a37ab70cb79b16893556cffdaad6974e7d8
Architecture: arm
Protocol Versions: [63 62]
Network Id: 1
Go Version: go1.11.9
Operating System: linux
GOPATH=
GOROOT=/home/travis/.gimme/versions/go1.11.9.linux.amd64
```

At this stage ethereum is successfully installed and ready to synchronise with live chain(blockchain-datachain).

--On computer --

Install homebrew on linux system(You might will have to install curl by running ***sudo apt install curl***)

sh -c "$(curl -fsSL https://raw.githubusercontent.com/Linuxbrew/install/master/install.sh)"

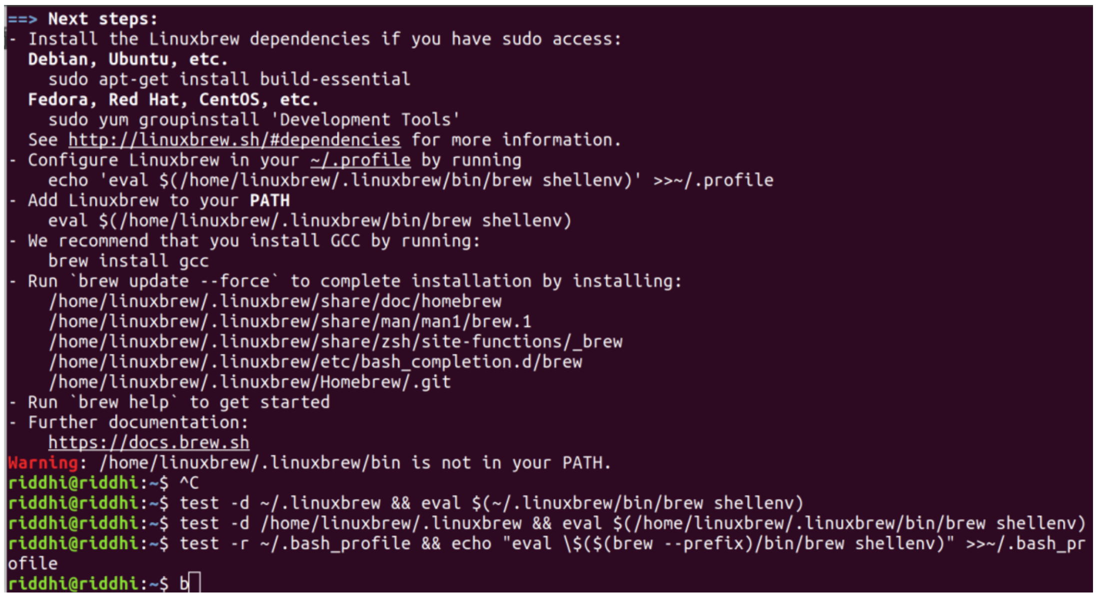


```
riddhi@riddhi:~$ sh -c "$(curl -fsSL https://raw.githubusercontent.com/Linuxbrew/install/master/install.sh)
"
==> Installing Ruby to /home/linuxbrew/.linuxbrew/Homebrew/Library/Homebrew/vendor
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 26.9M  100 26.9M    0     0  6595k      0  0:00:04  0:00:04 --:--:-- 8454k
==> Installing successful
==> /home/linuxbrew/.linuxbrew/Homebrew/Library/Homebrew/vendor/portable-ruby/current/bin/ruby
ruby 2.3.7p456 (2018-03-28 revision 63024) [x86_64-linux]
==> Add Ruby to your PATH by running:
PATH=/home/linuxbrew/.linuxbrew/Homebrew/Library/Homebrew/vendor/portable-ruby/current/bin:$PATH
==> This script will install:
/home/linuxbrew/.linuxbrew/bin/brew
/home/linuxbrew/.linuxbrew/share/doc/homebrew
/home/linuxbrew/.linuxbrew/share/man/man1/brew.1
/home/linuxbrew/.linuxbrew/share/zsh/site-functions/_brew
/home/linuxbrew/.linuxbrew/etc/bash_completion.d/brew
/home/riddhi/.cache/Homebrew/
/home/linuxbrew/.linuxbrew/Homebrew
==> The following new directories will be created:
/home/linuxbrew/.linuxbrew/bin
/home/linuxbrew/.linuxbrew/etc
/home/linuxbrew/.linuxbrew/include
/home/linuxbrew/.linuxbrew/lib
/home/linuxbrew/.linuxbrew/sbin
/home/linuxbrew/.linuxbrew/share
/home/linuxbrew/.linuxbrew/var
```

Setting homebrew in PATH:

Run following commands

1. test -d ~/.linuxbrew && eval $(~/.linuxbrew/bin/brew shellenv)
2. ***test -d /home/linuxbrew/.linuxbrew && eval $(/home/linuxbrew/.linuxbrew/bin/brew shellenv)***
3. ***test -r ~/.bash_profile && echo "eval \$($(brew --prefix)/bin/brew shellenv)" >>~/.bash_profile***
4. ***echo "eval \$($(brew --prefix)/bin/brew shellenv)" >>~/.profile***



```
==> Next steps:
- Install the Linuxbrew dependencies if you have sudo access:
  Debian, Ubuntu, etc.
    sudo apt-get install build-essential
  Fedora, Red Hat, CentOS, etc.
    sudo yum groupinstall 'Development Tools'
  See http://linuxbrew.sh/#dependencies for more information.
- Configure Linuxbrew in your ~/.profile by running
    echo 'eval $(/home/linuxbrew/.linuxbrew/bin/brew shellenv)' >>~/.profile
- Add Linuxbrew to your PATH
    eval $(/home/linuxbrew/.linuxbrew/bin/brew shellenv)
- We recommend that you install GCC by running:
    brew install gcc
- Run `brew update --force` to complete installation by installing:
    /home/linuxbrew/.linuxbrew/share/doc/homebrew
    /home/linuxbrew/.linuxbrew/share/man/man1/brew.1
    /home/linuxbrew/.linuxbrew/share/zsh/site-functions/_brew
    /home/linuxbrew/.linuxbrew/etc/bash_completion.d/brew
    /home/linuxbrew/.linuxbrew/Homebrew/.git
- Run `brew help` to get started
- Further documentation:
    https://docs.brew.sh
Warning: /home/linuxbrew/.linuxbrew/bin is not in your PATH.
riddhi@riddhi:~$ ^C
riddhi@riddhi:~$ test -d ~/.linuxbrew && eval $(~/.linuxbrew/bin/brew shellenv)
riddhi@riddhi:~$ test -d /home/linuxbrew/.linuxbrew && eval $(/home/linuxbrew/.linuxbrew/bin/brew shellenv)
riddhi@riddhi:~$ test -r ~/.bash_profile && echo "eval \$($(brew --prefix)/bin/brew shellenv)" >>~/.bash_pr
ofile
riddhi@riddhi:~$ b
```

Updating brew:

1. First lets update brew first

   brew update

2. Installing geth

   brew tap ethereum/ethereum
   brew install ethereum

3. Check geth Version to see it got installed properly

   geth version

4. Run geth

   geth

As soon you hit the above command, processing will start which will take longer than 24 hours to mine. Press ctrl+c to stop the processing.

------------------------------Setting up blockchain (private chain) in computer---------------------------------
-

Few points to consider:

- Each node will use a distinct data directory to store the database and the wallet.
- Each node must initialize a blockchain based on the same genesis file.

- Each node must join the same network id different from the one reserved by Ethereum (0 to 3 are already reserved).
- The port numbers must be different if different nodes are installed on the same computer.

Step 1: Creating data directory folder for private blockchain

```
mkdir -p ~/FirstBlock/miner1
mkdir -p ~/FirstBlock/miner2
```

Step 2: Create Genesis File

Enter the Project folder and make genesis file for minning the chain

```
cd FirstBlock
nano genesis.json
        {
 "nonce": "0x0000000000000042",
 "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",
 "difficulty": "20",
 "alloc": {},
 "coinbase": "0x0000000000000000000000000000000000000000",
 "timestamp": "0x00",
 "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
 "extraData": "0x436861696e536b696c6c732047656e6573697320426c6f636b",
 "gasLimit": "0xffffffff",
 "config": {
   "chainId": 44,
   "homesteadBlock": 0,
   "eip155Block": 0,
   "eip158Block": 0
 }
}
```

Enter the above block into the .json file.

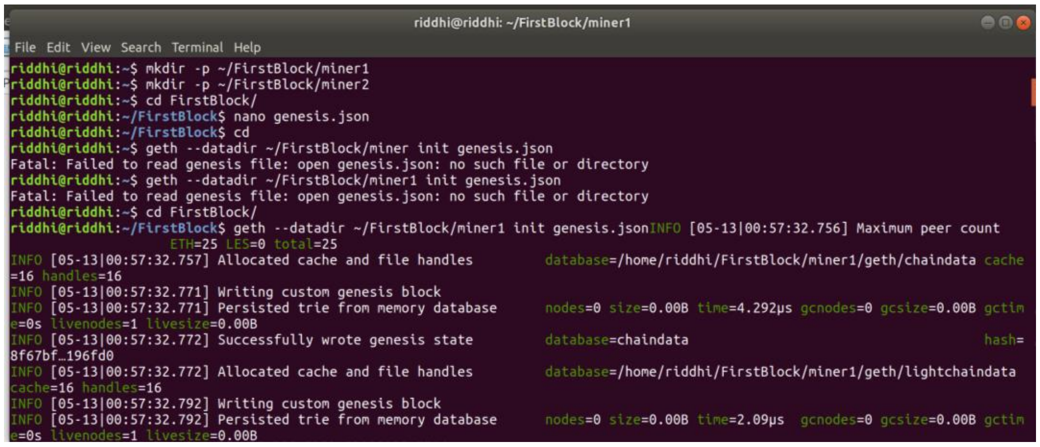
difficulty: if the value is low, the transactions will be quickly processed within our private blockchain.
gasLimit: define the limit of Gas expenditure per block. The gasLimit is set to the maximum to avoid being limited to our tests.

Step 3: Initialize the private chain

It’s time to initialize the private blockchain with the genesis block.
This operation will create the initial database stored under the data directory dedicated to each miner.

```
cd FirstBlock
geth --datadir ~/FirstBlock/miner1 init genesis.json
geth --datadir ~/FirstBlock/miner2 init genesis.json
```

geth contains database of private blockchain


riddhi@riddhi: ~/FirstBlock/miner1
File Edit View Search Terminal Help

```
riddhi@riddhi:~$ mkdir -p ~/FirstBlock/miner1
riddhi@riddhi:~$ mkdir -p ~/FirstBlock/miner2
riddhi@riddhi:~$ cd FirstBlock/
riddhi@riddhi:~/FirstBlock$ nano genesis.json
riddhi@riddhi:~/FirstBlock$ cd
riddhi@riddhi:~$ geth --datadir ~/FirstBlock/miner init genesis.json
Fatal: Failed to read genesis file: open genesis.json: no such file or directory
riddhi@riddhi:~$ geth --datadir ~/FirstBlock/miner1 init genesis.json
Fatal: Failed to read genesis file: open genesis.json: no such file or directory
riddhi@riddhi:~$ cd FirstBlock/
riddhi@riddhi:~/FirstBlock$ geth --datadir ~/FirstBlock/miner1 init genesis.jsonINFO [05-13|00:57:32.756] Maximum peer count
                    ETH=25 LES=0 total=25
INFO [05-13|00:57:32.757] Allocated cache and file handles        database=/home/riddhi/FirstBlock/miner1/geth/chaindata cache
=16 handles=16
INFO [05-13|00:57:32.771] Writing custom genesis block
INFO [05-13|00:57:32.771] Persisted trie from memory database      nodes=0 size=0.00B time=4.292µs gcnodes=0 gcsize=0.00B gctim
e=0s livenodes=1 livesize=0.00B
INFO [05-13|00:57:32.772] Successfully wrote genesis state        database=chaindata                                     hash=
8f67bf…196fd0
INFO [05-13|00:57:32.772] Allocated cache and file handles        database=/home/riddhi/FirstBlock/miner1/geth/lightchaindata
cache=16 handles=16
INFO [05-13|00:57:32.792] Writing custom genesis block
INFO [05-13|00:57:32.792] Persisted trie from memory database      nodes=0 size=0.00B time=2.09µs  gcnodes=0 gcsize=0.00B gctim
e=0s livenodes=1 livesize=0.00B
```

Step 4: Create account for miners

geth --datadir ~/FirstBlock/miner1 account new

You will be prompt to enter new password. Make sure you save the password properly.

Run the same code again for creating testing account

geth --datadir ~/FirstBlock/miner1 account new

Repeat the same steps for miner2
geth --datadir ~/FirstBlock/miner2 account new

Step 5: Preparing the miners

Make password.sec in each of the miners folder to save the password as config file.

cd FirstBlock/miner1
nano password.sec

<write the password you created for account of miner>

Repeat same steps for miner2

cd FirstBlock/miner2
nano password.sec

Creating runnable script startminer.sh file

cd FirstBlock/miner1
nano startminer.sh

Write below command in terminal

geth --identity "miner1r" --networkid 42 --datadir "~/FirstBlock/miner1" --nodiscover --mine --rpc --rpcport "8042" --port "30303" --unlock 0 --password ~/FirstBlock/miner1/password.sec --ipcpath "~/Library/Ethereum/geth.ipc"

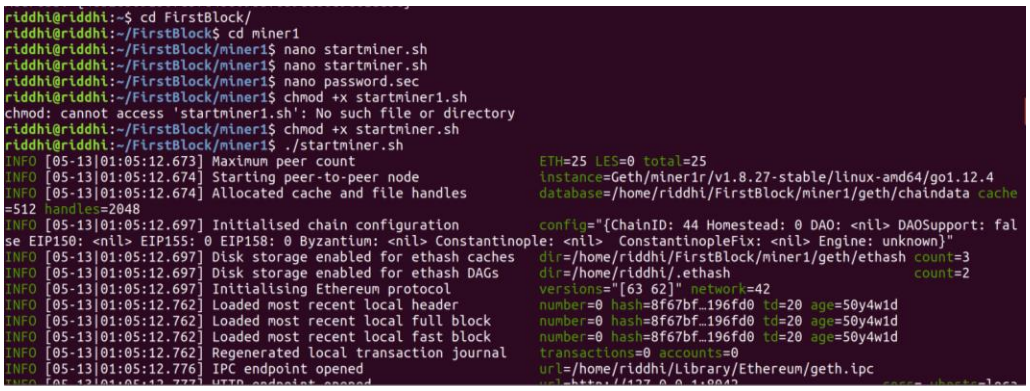
The meaning of the main parameters is the following:

1. **identity**: name of our node
2. **networkid**: this network identifier is an arbitrary value that will be used to pair all nodes of the same network. This value must be different from 0 to 3 (already used by the live chains)
3. **datadir**: folder where our private blockchain stores its data
4. **rpc and rpcport**: enabling HTTP-RPC server and giving its listening port number
5. **port**: network listening port number, on which nodes connect to one another to spread new transactions and blocks
6. **nodiscover**: disable the discovery mechanism (we will pair our nodes later)
7. **mine**: mine ethers and transactions
8. **unlock**: id of the default account
9. **password**: path to the file containing the password of the default account
10. **ipcpath**: path where to store the filename for IPC socket/pipe

Running startminer.sh

Before running we will have to change the execute priviledges of startminer.sh

cd ~/FirstBlock/miner1
chmod +x startminer1.sh



```
riddhi@riddhi:~$ cd FirstBlock/
riddhi@riddhi:~/FirstBlock$ cd miner1
riddhi@riddhi:~/FirstBlock/miner1$ nano startminer.sh
riddhi@riddhi:~/FirstBlock/miner1$ nano startminer.sh
riddhi@riddhi:~/FirstBlock/miner1$ nano password.sec
riddhi@riddhi:~/FirstBlock/miner1$ chmod +x startminer1.sh
chmod: cannot access 'startminer1.sh': No such file or directory
riddhi@riddhi:~/FirstBlock/miner1$ chmod +x startminer.sh
riddhi@riddhi:~/FirstBlock/miner1$ ./startminer.sh
INFO [05-13|01:05:12.673] Maximum peer count                       ETH=25 LES=0 total=25
INFO [05-13|01:05:12.674] Starting peer-to-peer node               instance=Geth/miner1r/v1.8.27-stable/linux-amd64/go1.12.4
INFO [05-13|01:05:12.674] Allocated cache and file handles         database=/home/riddhi/FirstBlock/miner1/geth/chaindata cache
=512 handles=2048
INFO [05-13|01:05:12.697] Initialised chain configuration          config="{ChainID: 44 Homestead: 0 DAO: <nil> DAOSupport: fal
se EIP150: <nil> EIP155: 0 EIP158: 0 Byzantium: <nil> Constantinople: <nil>  ConstantinopleFix: <nil> Engine: unknown}"
INFO [05-13|01:05:12.697] Disk storage enabled for ethash caches   dir=/home/riddhi/FirstBlock/miner1/geth/ethash count=3
INFO [05-13|01:05:12.697] Disk storage enabled for ethash DAGs     dir=/home/riddhi/.ethash                       count=2
INFO [05-13|01:05:12.697] Initialising Ethereum protocol           versions="[63 62]" network=42
INFO [05-13|01:05:12.762] Loaded most recent local header          number=0 hash=8f67bf…196fd0 td=20 age=50y4w1d
INFO [05-13|01:05:12.762] Loaded most recent local full block      number=0 hash=8f67bf…196fd0 td=20 age=50y4w1d
INFO [05-13|01:05:12.762] Loaded most recent local fast block      number=0 hash=8f67bf…196fd0 td=20 age=50y4w1d
INFO [05-13|01:05:12.762] Regenerated local transaction journal    transactions=0 accounts=0
INFO [05-13|01:05:12.776] IPC endpoint opened                      url=/home/riddhi/Library/Ethereum/geth.ipc
```

To run:
./startminer1.sh

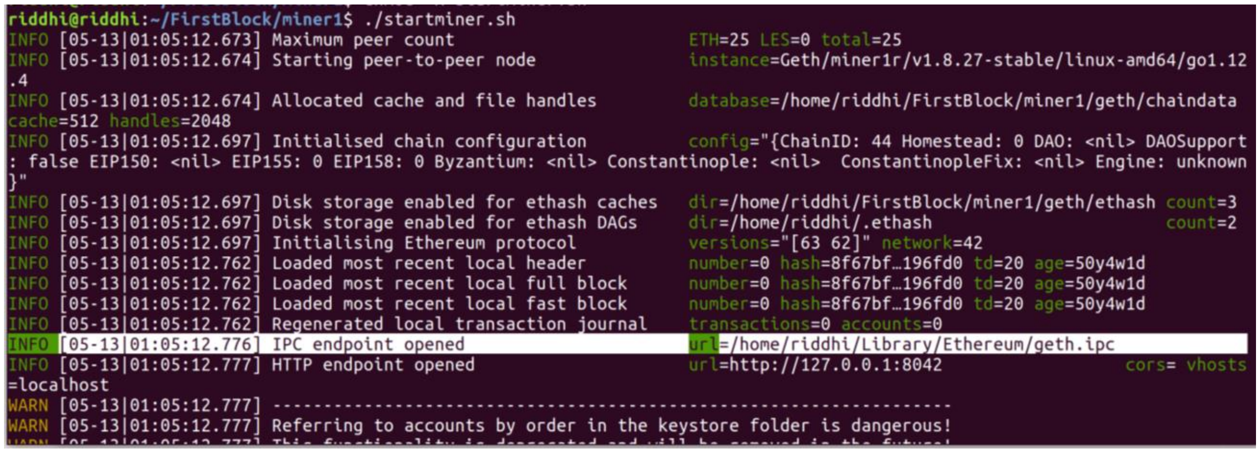
On running this code:

1. You will notice that the server and the mining process start. You default account will receive ethers mined by the node.
2. You can manage your miner using the Geth Javascript console
3. For example, you can start and stop mining from the console or send transactions.
4. This console needs to be attached to a running instance of Geth.
5. Open a new terminal session and type "**geth attach**".

If you want to start or stop the mining process, proceed as below:

geth attach "home/<directory for geth.ipc>"

*****Note: You can notice geth.ipc directory from first terminal under command***
IPC endpoint opened   url=/.../geth.ipc



```
riddhi@riddhi:~/FirstBlock/miner1$ ./startminer.sh
INFO [05-13|01:05:12.673] Maximum peer count                       ETH=25 LES=0 total=25
INFO [05-13|01:05:12.674] Starting peer-to-peer node               instance=Geth/miner1r/v1.8.27-stable/linux-amd64/go1.12
.4
INFO [05-13|01:05:12.674] Allocated cache and file handles         database=/home/riddhi/FirstBlock/miner1/geth/chaindata
cache=512 handles=2048
INFO [05-13|01:05:12.697] Initialised chain configuration          config="{ChainID: 44 Homestead: 0 DAO: <nil> DAOSupport
: false EIP150: <nil> EIP155: 0 EIP158: 0 Byzantium: <nil> Constantinople: <nil>  ConstantinopleFix: <nil> Engine: unknown
}"
INFO [05-13|01:05:12.697] Disk storage enabled for ethash caches   dir=/home/riddhi/FirstBlock/miner1/geth/ethash count=3
INFO [05-13|01:05:12.697] Disk storage enabled for ethash DAGs     dir=/home/riddhi/.ethash                       count=2
INFO [05-13|01:05:12.697] Initialising Ethereum protocol           versions="[63 62]" network=42
INFO [05-13|01:05:12.762] Loaded most recent local header          number=0 hash=8f67bf…196fd0 td=20 age=50y4w1d
INFO [05-13|01:05:12.762] Loaded most recent local full block      number=0 hash=8f67bf…196fd0 td=20 age=50y4w1d
INFO [05-13|01:05:12.762] Loaded most recent local fast block      number=0 hash=8f67bf…196fd0 td=20 age=50y4w1d
INFO [05-13|01:05:12.762] Regenerated local transaction journal    transactions=0 accounts=0
INFO [05-13|01:05:12.776] IPC endpoint opened                      url=/home/riddhi/Library/Ethereum/geth.ipc
INFO [05-13|01:05:12.777] HTTP endpoint opened                     url=http://127.0.0.1:8042                cors= vhosts
=localhost
WARN [05-13|01:05:12.777] -----------------------------------------------------------------
WARN [05-13|01:05:12.777] Referring to accounts by order in the keystore folder is dangerous!
```

Repeat the above step for miner2.

Preparing raspberry pi for blockchain

Create datadir folder in raspberry pi

mkdir -p ~/FirstBlock/node

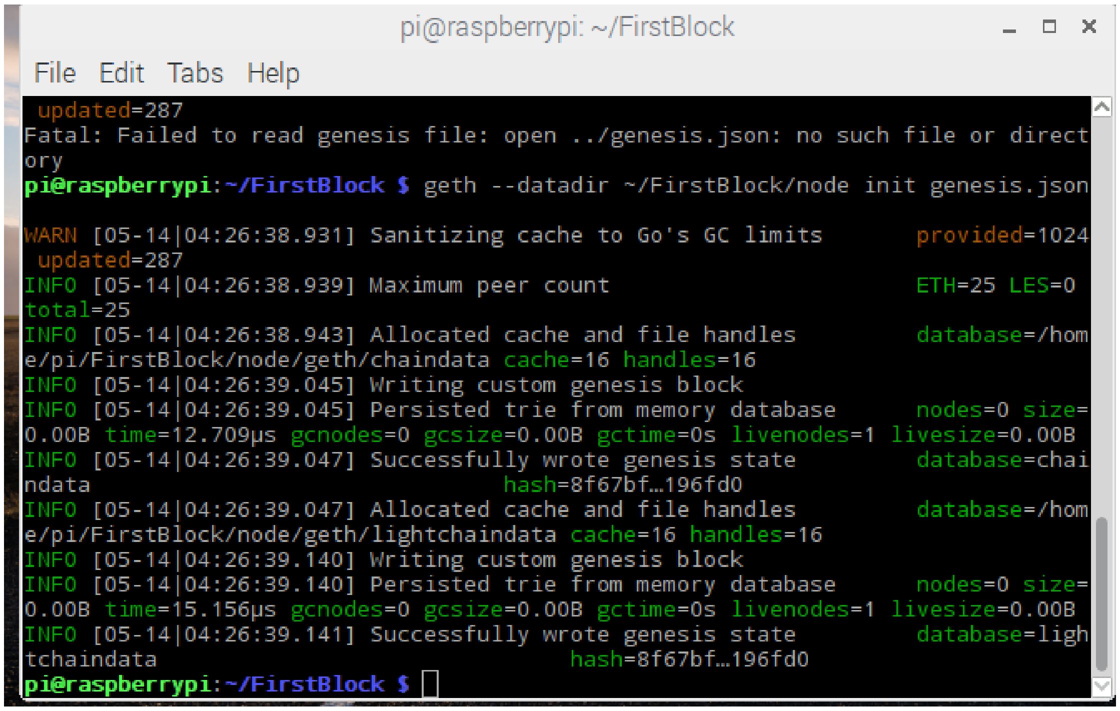
Transfer genesis file from laptop to raspberry pi

-----------------------on computer---
sftp pi@192.186.137.176
Note: write IP address of the raspberry pi, And write password of the laptop.

sftp> cd Project
sftp> put genesis.json
sftp>exit

Initialize the node

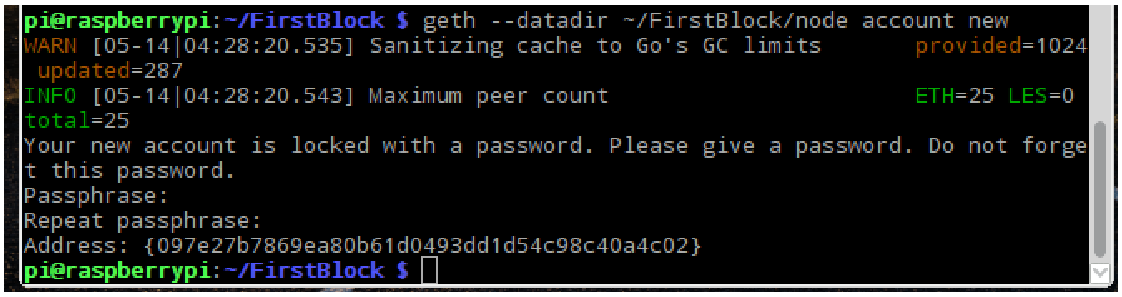
cd ~/FirstBlock
geth –datadir ~/FirstBlock/node init genesis.json



```
 updated=287
Fatal: Failed to read genesis file: open ../genesis.json: no such file or direct
ory
pi@raspberrypi:~/FirstBlock $ geth --datadir ~/FirstBlock/node init genesis.json

WARN [05-14|04:26:38.931] Sanitizing cache to Go's GC limits       provided=1024
 updated=287
INFO [05-14|04:26:38.939] Maximum peer count                       ETH=25 LES=0
total=25
INFO [05-14|04:26:38.943] Allocated cache and file handles         database=/hom
e/pi/FirstBlock/node/geth/chaindata cache=16 handles=16
INFO [05-14|04:26:39.045] Writing custom genesis block
INFO [05-14|04:26:39.045] Persisted trie from memory database      nodes=0 size=
0.00B time=12.709µs gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-14|04:26:39.047] Successfully wrote genesis state         database=chai
ndata                              hash=8f67bf…196fd0
INFO [05-14|04:26:39.047] Allocated cache and file handles         database=/hom
e/pi/FirstBlock/node/geth/lightchaindata cache=16 handles=16
INFO [05-14|04:26:39.140] Writing custom genesis block
INFO [05-14|04:26:39.140] Persisted trie from memory database      nodes=0 size=
0.00B time=15.156µs gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-14|04:26:39.141] Successfully wrote genesis state         database=ligh
tchaindata                              hash=8f67bf…196fd0
pi@raspberrypi:~/FirstBlock $ 
```

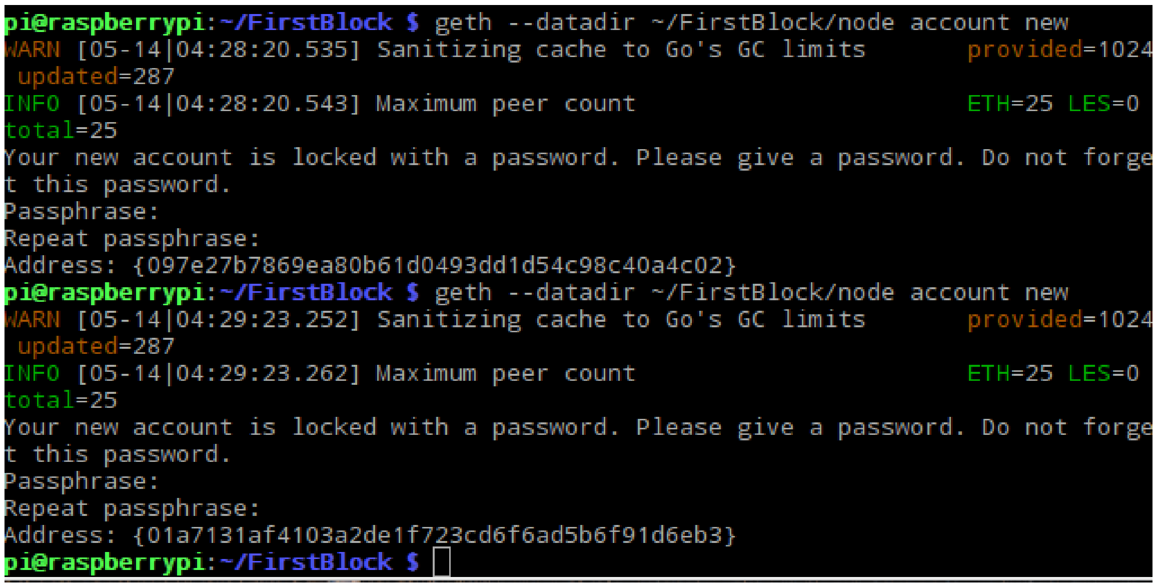
Cerate Account
geth --datadir ~/FirstBlock/node account new
Note: You will be prompt to enter the password. Save the entered password safely for future use.



```
pi@raspberrypi:~/FirstBlock $ geth --datadir ~/FirstBlock/node account new
WARN [05-14|04:28:20.535] Sanitizing cache to Go's GC limits       provided=1024
 updated=287
INFO [05-14|04:28:20.543] Maximum peer count                       ETH=25 LES=0
total=25
Your new account is locked with a password. Please give a password. Do not forge
t this password.
Passphrase:
Repeat passphrase:
Address: {097e27b7869ea80b61d0493dd1d54c98c40a4c02}
pi@raspberrypi:~/FirstBlock $ 
```

Again, repeat above step to create test account.

geth --datadir ~/FirstBlock/node account new



```
pi@raspberrypi:~/FirstBlock $ geth --datadir ~/FirstBlock/node account new
WARN [05-14|04:28:20.535] Sanitizing cache to Go's GC limits       provided=1024
 updated=287
INFO [05-14|04:28:20.543] Maximum peer count                       ETH=25 LES=0
total=25
Your new account is locked with a password. Please give a password. Do not forge
t this password.
Passphrase:
Repeat passphrase:
Address: {097e27b7869ea80b61d0493dd1d54c98c40a4c02}
pi@raspberrypi:~/FirstBlock $ geth --datadir ~/FirstBlock/node account new
WARN [05-14|04:29:23.252] Sanitizing cache to Go's GC limits       provided=1024
 updated=287
INFO [05-14|04:29:23.262] Maximum peer count                       ETH=25 LES=0
total=25
Your new account is locked with a password. Please give a password. Do not forge
t this password.
Passphrase:
Repeat passphrase:
Address: {01a7131af4103a2de1f723cd6f6ad5b6f91d6eb3}
pi@raspberrypi:~/FirstBlock $
```

To check all account is create type command:

geth --datadir ~/FirstBlock/node account list

Prepare the node
Go to directory ~/FirstBlock/node

Create "password.sec" file and write both the password you selected while creating the account.

Create running shell script in this directory "startnode.sh"

Geth --identity "node1" --fast --networkid 42 --datadir /home/pi/FirstBlock/node --nodiscover --rpc –rpcport "8042" --port "30303" --unlock 0 --password "/home/pi/Firstnode/password.sec" --ipcpath /home/pi/.ethereum/geth.ipc

Start Node

Make the "startnode.sh" script runnable

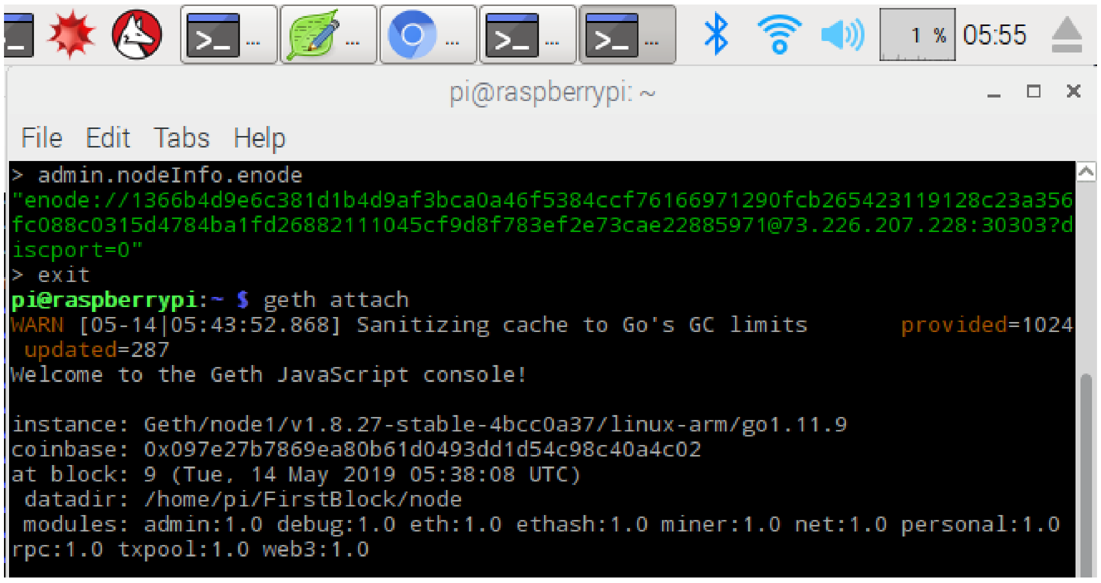
cd ~/FirstNode/node
chmod +x startnode.sh

./startnode.sh

Note: On successfully running this code it opens javascript console.

Now open new terminal. And attach that to the chain created using

geth attach



```
> admin.nodeInfo.enode
"enode://1366b4d9e6c381d1b4d9af3bca0a46f5384ccf76166971290fcb265423119128c23a356
fc088c0315d4784ba1fd26882111045cf9d8f783ef2e73cae22885971@73.226.207.228:30303?d
iscport=0"
> exit
pi@raspberrypi:~ $ geth attach
WARN [05-14|05:43:52.868] Sanitizing cache to Go's GC limits       provided=1024
 updated=287
Welcome to the Geth JavaScript console!

instance: Geth/node1/v1.8.27-stable-4bcc0a37/linux-arm/go1.11.9
coinbase: 0x097e27b7869ea80b61d0493dd1d54c98c40a4c02
at block: 9 (Tue, 14 May 2019 05:38:08 UTC)
 datadir: /home/pi/FirstBlock/node
 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
```

From this console get the node information using
admin.nodeInfo.enode
---------------------***On Computer***---------------------------

Go to ***directory ~/FirstBlock/miner1***
Create ***static-nodes.json on computer***

Enter node information of miner1 and node in this file
[
"enode://b8863bf7c8bb13c3afc459d5bf6e664ed4200f50b86aebf5c70d205d32dd77cf2a888b8adf4a8e55ab13e8ab5ad7ec93b7027e73ca70f87af5b425197712d272@192.168.1.39:30303",
"enode://b8863bf7c8bb13c3afc459d5bf6e664ed4200f50b86aebf5c70d205d32dd77cf2a888b8adf4a8e55ab13e8ab5ad7ec93b7027e73ca70f87af5b425197712d272@192.168.1.39:30303"
]

Now using sftp send the static-nodes.json

cd ~/FirstBlock/miner1
sftp *pi@192.168.1.31*

sftp>cd ~/FirstBlock/node
sftp>put static-nodes.json
sftp>exit
-----------------------------------On Computer--
From computer terminal attach to private blockchain using geth command
computer> geth attach "home/<directory for geth.ipc>"
…
eth.account[1]
Note down the account information for future use to send the ether

```
> eth.accounts[1]
"0x6af547b83493fd59bf5a2e67546b65191392f45a"
```

----------------------------------On Raspberry pi---
Sending ether from pi to miner1
Pi>geth attach
Eth.sendTransaction({from: eth.coinbase, to: “<enter the account information>”, value: web3.toWei(10, “ether”)})

Repeat same step to send ether from miner1(computer) to raspberry pi