# Setup private Ethereum Network

## Installing geth on Mac

brew tap ethereum/ethereum

brew install ethereum

### Or build from source

xcode-select --install

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

brew install go

cd go-ethereum

make geth

## Installing geth on Ubuntu

sudo apt-get install software-properties-common

sudo add-apt-repository -y ppa:ethereum/ethereum

sudo apt-get update

sudo apt-get install ethereum

### Browser source code of Geth (Go-Ethereum client)

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

cd go-ethereum/

### Verify install

geth version

haseeb$ geth version

Geth

Version: 1.8.10-stable

Architecture: amd64

Protocol Versions: [63 62]

Network Id: 1

Go Version: go1.10.2

Operating System: darwin

GOPATH=

GOROOT=/usr/local/Cellar/go/1.10.2/libexec

## Create workspace for your test network

mkdir my-ethereum-network

mkdir node1 node2

## Create wallets

The wallets (or node accounts) hold a key-pair (private-public key pair) required interacting with blockchain network.

There are 2 key types of nodes

1. Boot nodes
2. Peers/nodes
   1. Minding nodes or Voters or Sealers
   2. Transaction nodes (non-miners / non-voters etc)

We need atleast 2 accounts/wallets for 2 nodes that we want to create a Blockchain network with.

haseeb$ geth --datadir node1/ account new

INFO [06-07|07:35:51] Maximum peer count ETH=25 LES=0 total=25

Your new account is locked with a password. Please give a password. Do not forget this password.

Passphrase:

Repeat passphrase:

Address: {4db50f170ad86be76293efc6e825d450f85d00b6}

### Save the generated account address, i.e. 4db50f170ad86be76293efc6e825d450f85d00b6

haseeb$ geth --datadir node2/ account new

INFO [06-07|07:37:23] Maximum peer count ETH=25 LES=0 total=25

Your new account is locked with a password. Please give a password. Do not forget this password.

Passphrase:

Repeat passphrase:

Address: {f84363c5b0f84aa178ca2f4d075baeb41d155d87}

### Save the generated account address, i.e. f84363c5b0f84aa178ca2f4d075baeb41d155d87

### Validate that keystores are created

haseeb$ ls node1/keystore/

haseeb$ ls node2/keystore/

### Explore the generated keystore files for your understanding.

### Copy account addresses for your ease

haseeb$ echo '4db50f170ad86be76293efc6e825d450f85d00b6' >> accounts.txt

haseeb$ echo 'f84363c5b0f84aa178ca2f4d075baeb41d155d87' >> accounts.txt

### Save passwords for your ease as well

haseeb$ echo 'your\_node\_pwd' > node1/password.txt

haseeb$ echo 'your\_node\_pwd' > node2/password.txt

### Create Genesis file

Genesis file/block is used to initialize a chain. This is also used to join an existing network.

For that, you will need accounts for purposes of approving them as sealers and pre-funding them.

haseeb$ cat accounts.txt

4db50f170ad86be76293efc6e825d450f85d00b6

f84363c5b0f84aa178ca2f4d075baeb41d155d87

haseeb$ puppeth

+-----------------------------------------------------------+

| Welcome to puppeth, your Ethereum private network manager |

| |

| This tool lets you create a new Ethereum network down to |

| the genesis block, bootnodes, miners and ethstats servers |

| without the hassle that it would normally entail. |

| |

| Puppeth uses SSH to dial in to remote servers, and builds |

| its network components out of Docker containers using the |

| docker-compose toolset. |

+-----------------------------------------------------------+

Please specify a network name to administer (no spaces or hyphens, please)

> mynetwork

Sweet, you can set this via --network=mynetwork next time!

INFO [06-07|07:47:41] Administering Ethereum network name=mynetwork

WARN [06-07|07:47:41] No previous configurations found path=/Users/haseeb/.puppeth/mynetwork

What would you like to do? (default = stats)

1. Show network stats

2. Configure new genesis

3. Track new remote server

4. Deploy network components

> 2

Which consensus engine to use? (default = clique)

1. Ethash - proof-of-work

2. Clique - proof-of-authority

> 2

How many seconds should blocks take? (default = 15)

> 10

Which accounts are allowed to seal? (mandatory at least one)

> 0x4db50f170ad86be76293efc6e825d450f85d00b6

> 0xf84363c5b0f84aa178ca2f4d075baeb41d155d87

> 0x

Which accounts should be pre-funded? (advisable at least one)

> 0x4db50f170ad86be76293efc6e825d450f85d00b6

> 0xf84363c5b0f84aa178ca2f4d075baeb41d155d87

> 0x

Specify your chain/network ID if you want an explicit one (default = random)

> 1500

INFO [06-07|07:49:01] Configured new genesis block

What would you like to do? (default = stats)

1. Show network stats

2. Manage existing genesis

3. Track new remote server

4. Deploy network components

> 2

1. Modify existing fork rules

2. Export genesis configuration

3. Remove genesis configuration

> 2

Which file to save the genesis into? (default = mynetwork.json)

>

INFO [06-07|07:49:22] Exported existing genesis block

What would you like to do? (default = stats)

1. Show network stats

2. Manage existing genesis

3. Track new remote server

4. Deploy network components

> ^C

### Explore your genesis block and try to understand it

haseeb$ less mynetwork.json | more

{

"config": {

"chainId": 1500,

"homesteadBlock": 1,

"eip150Block": 2,

"eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",

"eip155Block": 3,

"eip158Block": 3,

"byzantiumBlock": 4,

"clique": {

"period": 10,

"epoch": 30000

}

},

"nonce": "0x0",

….

….

"4db50f170ad86be76293efc6e825d450f85d00b6": {

"balance": "0x200000000000000000000000000000000000000000000000000000000000000"

},

"f84363c5b0f84aa178ca2f4d075baeb41d155d87": {

"balance": "0x200000000000000000000000000000000000000000000000000000000000000"

}

},

"number": "0x0",

"gasUsed": "0x0",

"parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000"

}

### Initialize your nodes

Initialize your nodes with generated genesis block. Remember anytime you change your genesis block you must re-initialize your nodes. Generally this is not the case.

haseeb$ geth --datadir node1/ init mynetwork.json

haseeb$ geth --datadir node2/ init mynetwork.json

These should result in something like this

INFO [06-07|07:56:54] Maximum peer count ETH=25 LES=0 total=25

INFO [06-07|07:56:54] Allocated cache and file handles database=/Users/haseeb/my-ethereum-network/node2/geth/chaindata cache=16 handles=16

INFO [06-07|07:56:54] Writing custom genesis block

INFO [06-07|07:56:54] Persisted trie from memory database nodes=356 size=65.53kB time=1.47555ms gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B

INFO [06-07|07:56:54] Successfully wrote genesis state database=chaindata hash=8efe81…c4c13f

INFO [06-07|07:56:54] Allocated cache and file handles database=/Users/haseeb/my-ethereum-network/node2/geth/lightchaindata cache=16 handles=16

INFO [06-07|07:56:54] Writing custom genesis block

INFO [06-07|07:56:54] Persisted trie from memory database nodes=356 size=65.53kB time=975.244µs gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B

INFO [06-07|07:56:54] Successfully wrote genesis state database=lightchaindata hash=8efe81…c4c13f

### Create a boot node

Boot node helps node discovery for network. You can create your network without a boot node but then adding a node to network might require manual addition throught management api. Please see reference, the desired API will be admin.addPeer(…..).

haseeb$ bootnode -genkey boot.key

haseeb$ cat boot.key

0110f5bd321ba8f08f16af1bfe12cbc333a5f53afad2c13d0f71e063fecec106

haseeb$ bootnode -nodekey boot.key -verbosity 9 -addr :30310

INFO [06-07|08:04:30] UDP listener up self=enode://f7693502b46789545f6b48d6ca46f8cec79f075a0a467e9e18e0bc729c7eacc7f6e9d3fd27797f4aabc1497069478dbf589222d8bb5194b0d913feec1e4af377@[::]:30310

Note enode for latter reference. Enode uniquely refers to a node, in this instance our created boot node. Also note we started boot node on port 30310, we can start it on any other port as well (avoid commonly used ports though)

### Get Node 1 live

haseeb$ geth --datadir node1/ --syncmode 'full' --port 30311 --rpc --rpcaddr 'localhost' --rpcport 8501 --rpcapi 'personal,db,eth,net,web3,txpool,miner' --bootnodes 'enode://f7693502b46789545f6b48d6ca46f8cec79f075a0a467e9e18e0bc729c7eacc7f6e9d3fd27797f4aabc1497069478dbf589222d8bb5194b0d913feec1e4af377@127.0.0.1:30310' --networkid 1500 --gasprice '1' -unlock '0x4db50f170ad86be76293efc6e825d450f85d00b6' --password node1/password.txt --mine console

Please note

1. Replace your boot node’s enode in this command
2. Please make a note of port you are running this miner
3. Please make a note of rpc port
4. In unlock use the account generated for this node (if any confusion, look in respective node’s keystore file
5. Please make sure to use the same networkid you choose for your genesis block

### Get Node 2 live

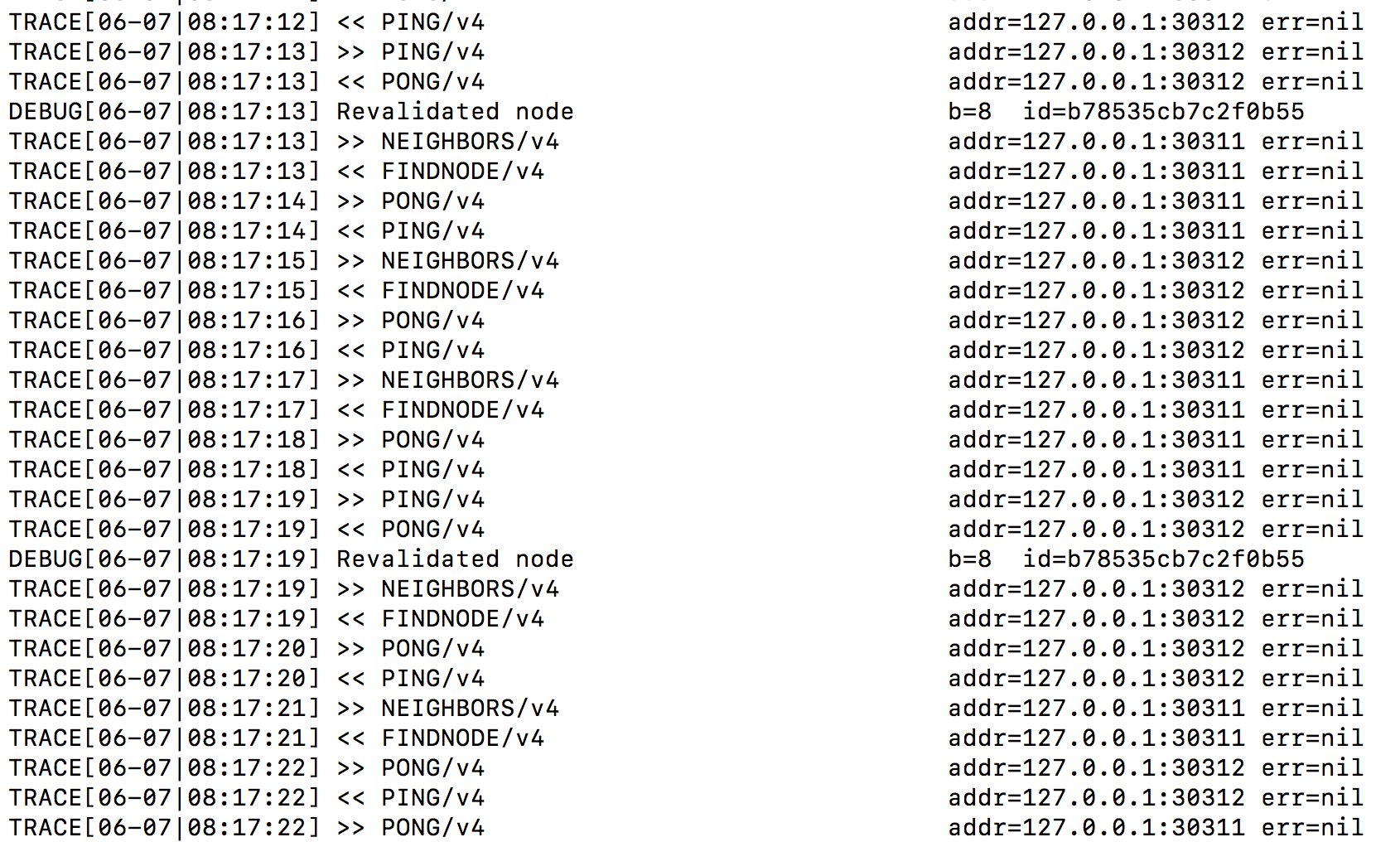
haseeb$ geth --datadir node1/ --syncmode 'full' --port 30312 --rpc --rpcaddr 'localhost' --rpcport 8501 --rpcapi 'personal,db,eth,net,web3,txpool,miner' --bootnodes 'enode://f7693502b46789545f6b48d6ca46f8cec79f075a0a467e9e18e0bc729c7eacc7f6e9d3fd27797f4aabc1497069478dbf589222d8bb5194b0d913feec1e4af377@127.0.0.1:30310' --networkid 1500 --gasprice '1' -unlock '0xf84363c5b0f84aa178ca2f4d075baeb41d155d87' --password node2/password.txt --mine console

Please note

1. Make sure there is no port conflict (if running on same machine) in my case in incremented both node port and rpc port by 1.
2. In unlock use the account generated for node 2 (if any confusion, look in respective node 2’s keystore file
3. Please make sure to use the same networkid you choose for your genesis block

### Validate that Boot nodes is getting Pings from miners

Check your boot node and see PING/PONG from both nodes. You can identify them by ports (if you used same machine or by IP if both nodes are on different machines)



### Validate that both miners/validators are mining/sealing



### Recap

Both accounts for both miners were approved as signers in genesis block. That’s why they could start mining upon start. No voting was needed for them to be approved.

Try out some management API.

1. **Interaction with console:** As we started nodes with console, you can interact with nodes directly.
2. **Interaction through IPC:** Another way is to attach to node’s IPC to get access to console

$ geth attach node1/geth.ipc

* Try this out – this has to be done locally from same machine

1. Interaction through RPC:

$ geth attach 'http://IP:RPC\_PORT'

Please note that we allowed RPC for nodes, please go back to command and see –-rpc and in --rpcapi we allowed following APIs for a node 'personal,db,eth,net,web3,txpool,miner

e.g. try out following commands through any of the above

> admin.peers

e.g. Check out signers/sealers/miners in network (remember we used clique as consensus algorithm)

> clique.getSigners()

["0x4db50f170ad86be76293efc6e825d450f85d00b6", "0xf84363c5b0f84aa178ca2f4d075baeb41d155d87"]

Also note that we instructed nodes upon starting to mine using --mine. You can use API’s like miner:start() or miner.stop() etc.

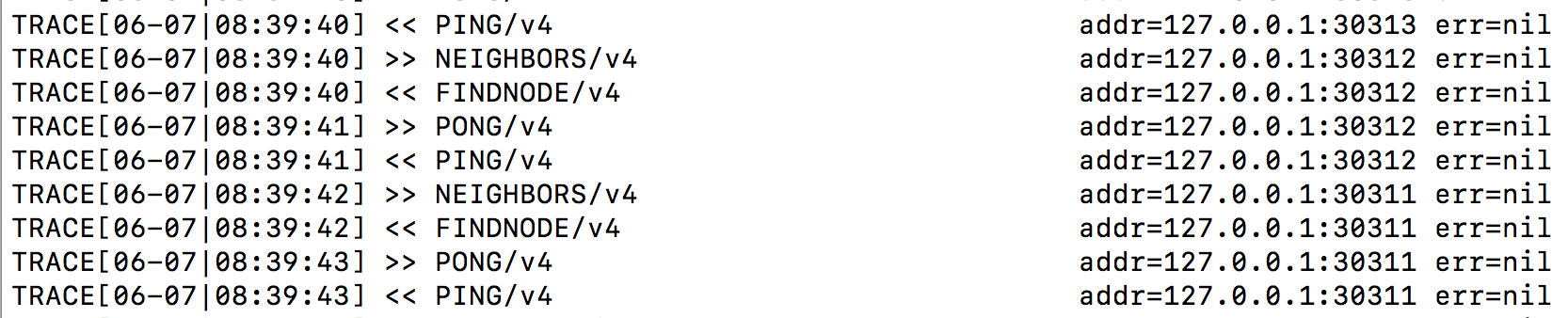
### How to run a voting to add a miner

Start node3 by following similar steps

1. Creating a datadir for node
2. Creating an account for node
3. Initializing the node with same gensis block
4. Making node live with same boot node, networkid. (again be careful of ports you use if you are using the same machine)

For ease, start the node with --mine and console. i.e. start a console and instruct it to start mining right away.

First check that boot node is getting PINGS from new node. (note, all ports listed here – from ports you can identify if using same IP for all nodes)



Check all 3 nodes how does a peer count look like using

> admin.peers

Now verify signers in network (all 3 nodes, including node 3 should have the already pre-approved 2 signers from node1 and node2) i.e. node3 even though instructed to mine can’t mine and is not approved.

> clique.getSigners()

["0x4db50f170ad86be76293efc6e825d450f85d00b6", "0xf84363c5b0f84aa178ca2f4d075baeb41d155d87"]

On node3 you should see something like

INFO [06-07|08:43:26] Commit new mining work number=171 txs=0 uncles=0 elapsed=216.97µs

WARN [06-07|08:43:26] Block sealing failed err=unauthorized

INFO [06-07|08:43:36] Imported new chain segment blocks=1 txs=0 mgas=0.000 elapsed=459.692µs mgasps=0.000 number=171 hash=dba37f…783825 cache=0.00B

INFO [06-07|08:43:36] Commit new mining work number=172 txs=0 uncles=0 elapsed=121.55µs

WARN [06-07|08:43:36] Block sealing failed err=unauthorized

INFO [06-07|08:43:46] Imported new chain segment blocks=1 txs=0 mgas=0.000 elapsed=655.961µs mgasps=0.000 number=172 hash=a68d12…4b26bb cache=0.00B

INFO [06-07|08:43:46] Commit new mining work number=173 txs=0 uncles=0 elapsed=235.461µs

WARN [06-07|08:43:46] Block sealing failed err=unauthorized

Now propose/vote node3 to be a signer/sealer from node1 and node2 (note that to approve a signer, majority needs to vote). Propose the account for node3.

> clique.propose("0x3d8e07c1f21a74d18c87849abbc2ee6da50fb0ed", true)

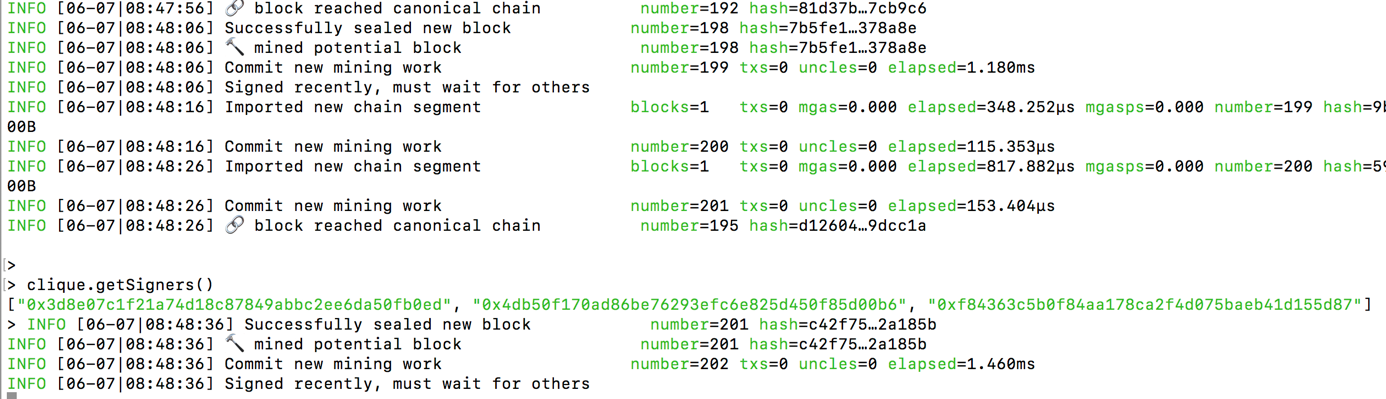
null

Now you should be able to see node3 signing/committing blocks. All nodes should show node3’s account as a valid signers as well.

> clique.getSigners()

["0x3d8e07c1f21a74d18c87849abbc2ee6da50fb0ed", "0x4db50f170ad86be76293efc6e825d450f85d00b6", "0xf84363c5b0f84aa178ca2f4d075baeb41d155d87"]

Node 3 should show something like this



### Next steps:

1. Try out [API’s](https://github.com/ethereum/go-ethereum/wiki/Management-APIs) yourself
2. Try out [Metamask](https://github.com/cubedro/eth-netstats) for wallet creation. (here you might have an issue with RPC ports – make sure those are open).
3. Install [ethstats](https://github.com/cubedro/eth-netstats) to have a view like [this](https://ethstats.net/) for your network.
4. Install [block explorer](https://github.com/carsenk/explorer) to have a visual way to explore your blocks.
5. Explore [remix](https://remix.ethereum.org/) (online solidity compiler), try out some smart contracts.
6. Deploy [smart contracts through remix](https://codeburst.io/build-your-first-ethereum-smart-contract-with-solidity-tutorial-94171d6b1c4b) or try transactions between accounts.