目录

[1 Install Openethereum 1](#_Toc7042)

[2 Brief introduction of Openethereum 3](#_Toc23038)

[2.1 commands 3](#_Toc6945)

[2.2 Chain Specification 4](#_Toc3490)

[2.3 How to Set Up a POA Blockchain Network Using Openethereum 6](#_Toc20815)

[3. Hard Fork 13](#_Toc15846)

# 1 Install Openethereum

* 1. Download the source code from Repo: <https://github.com/openethereum/openethereum> and Build Openethereum Client
  2. Need to install prerequisite:

Rust: curl https://sh.rustup.rs -sSf | sh

Perl:

sudo apt-get update

sudo apt-get install perl

perl -v to verify the installation

Yasm: sudo apt-get update

sudo apt-get install yasm

* 1. There are two ways to get an executable openethereum client: 1 build from the source code, 2 use built version provided by openethereum

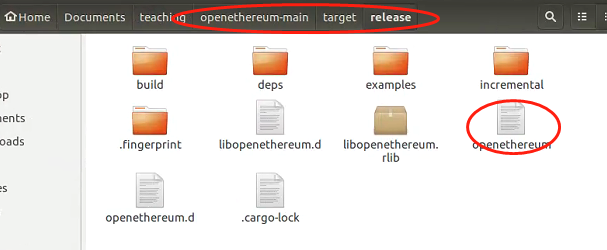
**Build from the source code**

Build an executable openethereum client from the source

cd (the source code folder)

cargo build --release --features final

After it successfully runs, it generates a file call openethereum under target/release folder



**Use built version**

download directly from: https://github.com/openethereum/openethereum/releases

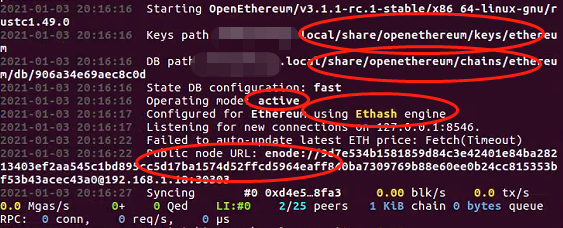
sudo chmod +x openethereum

* 1. Start openethereum and connect to the mainnet:

cd (path to the openethereum folder)

./openethereum

This will start an ethereum client and the client will sync the block from ethereum mainnet



Notes:

1. Keys Path is where the json keystore files located.
2. DB path is where the blockchain data stored

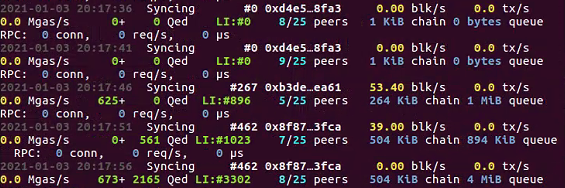
State DB config: METHOD may be one of auto, archive, fast:

archive - keep all state data. No pruning.

fast - maintain journal overlay. Fast but 50MB used.

auto - use the method most recently synced or default to fast if none synced. (default: auto)

1. Configured for Ethereum using Ethash engine. It indicates the Network this client connected to and the consensus. In this case, it connects to Ethereum main net using POW consensus.
2. After it ran, you would see #462 0x8f87...3fca. It indicates that this client has synced to block 462, and the blockhash is 0x8f87...3fca, and it connected with 8 peers



# 2 Brief introduction of Openethereum

## 2.1 commands

**Use hep to find the usage of the command**

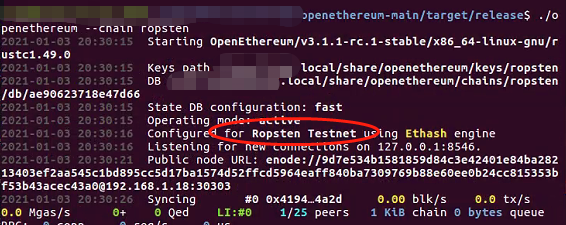
./openethereum --help

If we want to connect to other blockchain network, we could add --chain + the network name or chain spec json file path to join other blockchain networks

For example

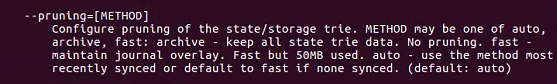
./openethereum --chain ropsten

This command ask the openethereum client to connect to Ropsten test net



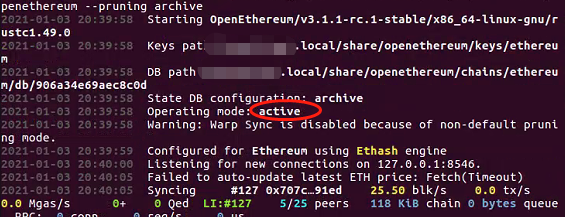
./openethereum --pruning: config pruning of the state/storge trie

--pruning archive



try ./openethereum --pruning archive

Then we can see the DB configuration switched to archive



## 2.2 Chain Specification

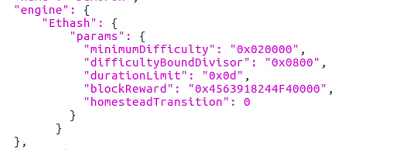
Chain Specification uniquely defines a blockchain network, and every nodes/clients that want to join the network must read the same Chain Spec json file

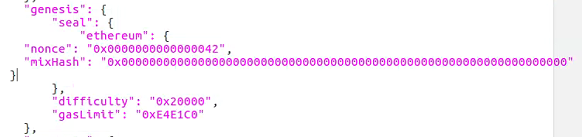
Template is as follow:



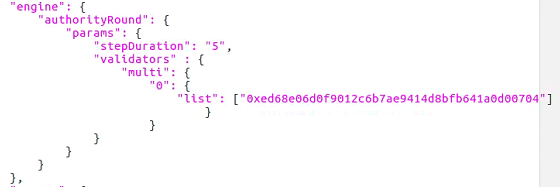
We could defines a consensus by replace the engine part in the template above

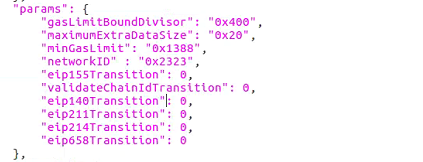
POW Consensus:

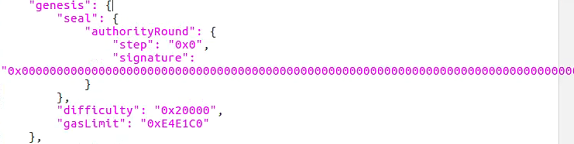




POA:







## 2.3 How to Set Up a POA Blockchain Network Using Openethereum

1. create a the chain spec json file for this network. Can go to <https://github.com/onebit256/poa> this repo, use these configuration files to compose your blockchain network





Notes:

stepDuration: how long to seal a block

Multi: specifies the validator address to seal the blocks

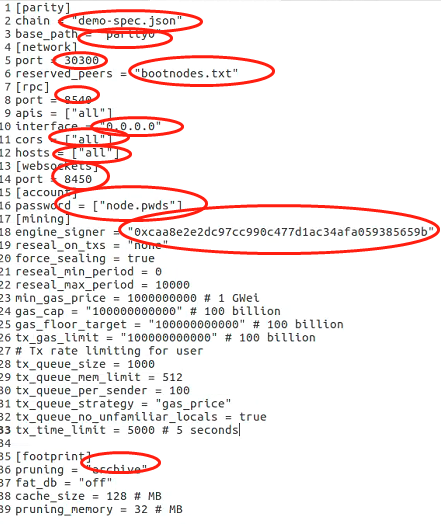
networkID: uniquely defines a network; Ethereum mainnet is 1; Ropsten 3; Kovan 4

Accounts: pre-assign some fund this specified account

1. There are two ways to start a network: create a toml file, and put all the configurations to this toml file or typing command with parameters to start the network

./openethereum --chain demo-spec.json

./openethereum -c node0.toml



\* Explain the parameters one by one

Notes:

- Chain specifies the chain spec json, which uniquely defines a network

- Base path defines where to store the blockchain data

- network 30303 is the port that allows peer connection

- reserved peers defines the bootnodes

- rpc defines the RPC connection

Interface set to be 0.0.0.0 allows the connection from all ip

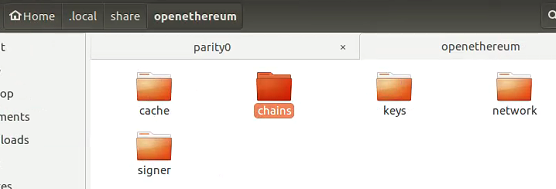
- accounts: you have to put the password for the address created by the openethereum client to the password file

- mining sections defines the mining behavior of this network, please note that you have to have the keystore json of the engine signer in this client so that this node can seal the block. And the engine signer must be set in chain spec json in the engine section

1. If you don’t have an account, you can create one by the following command

create a validator account in openethereum

./openethereum account new ->



Chains: Store blockchain data

Keys: All the private keys created by this client

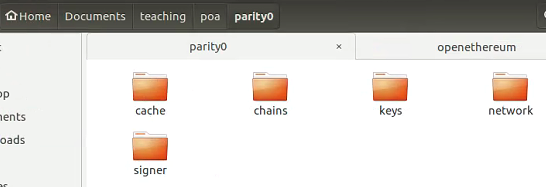
Network: peer data

Cache: all cached transactions and so on

Signer: Metamask injection(make a transaction by a signer, rather than by the accounts with pk stored in this client)

\*go through each folder

./openethereum account new --chain demo-spec.json --key-path parity0/keys

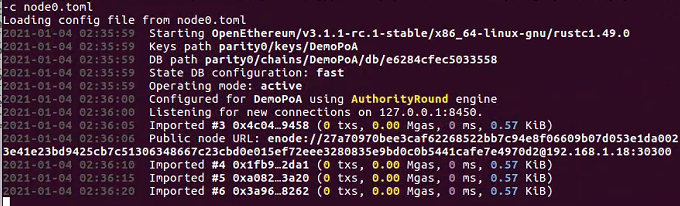


https://openethereum.github.io/Configuring-OpenEthereum

1. Start a real private network and connect to it via Metamask

Cors if don’t set this to be all and interface to 0.0.0.0, then Metamask can’t connect to the blockchain

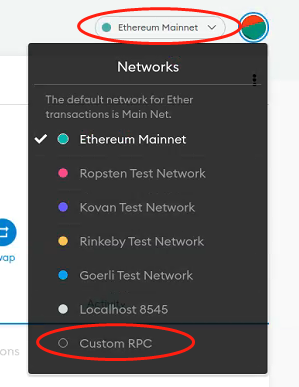
Start the network: ./openethereum - c node0.json



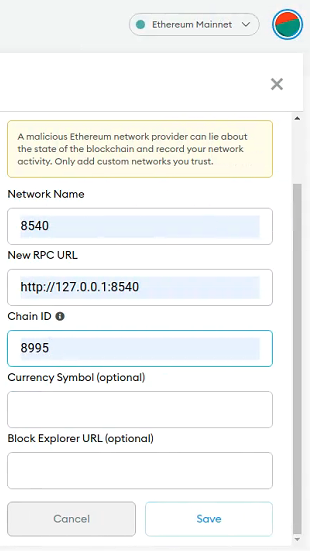
When you see import #x means you managed to start the network

Use Metamask to connect to the Blockchain

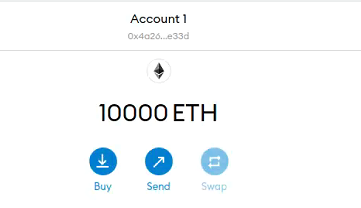
Step 1 configure the network on Metamask, click Custom RPC



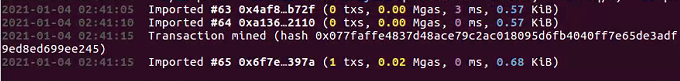
Then set the new network’s parameters on Metamask



Then import the fund pre-assigned account to Metamask, you will see the available funds



Setp 2. you could Make a Transfer to test if this chain works well

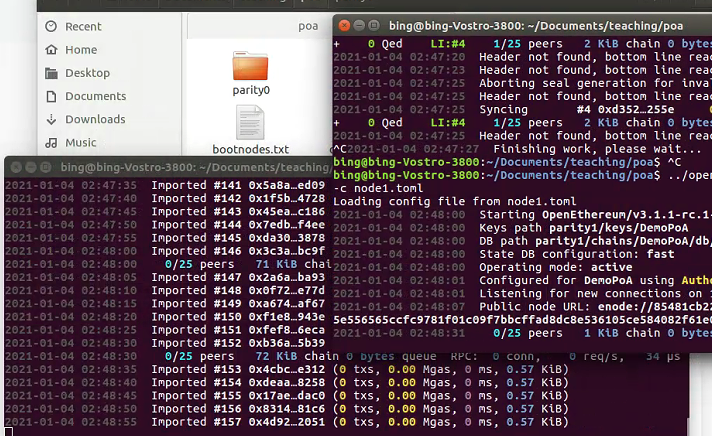


After you made the transfer you would find the transaction hash on the console log as above

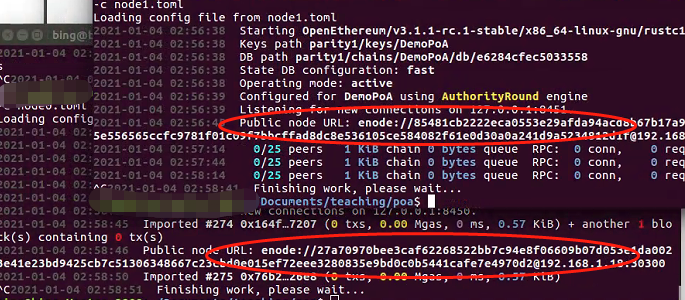
1. Composing a 2 nodes network

Create another toml file with different ports, and leave other parameters unchanged. Then start another client with this config

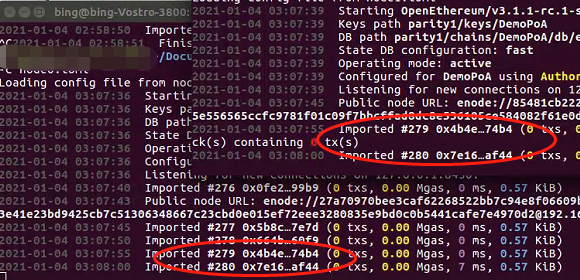
./openethereum - c node1.toml



It is better to set the bootnodes, so that a new node can quickly discover the peer nodes on the network



Add these to bootnodes.txt



After you start a second node, you will see the same block will have the same block hash

# Hard Fork



And change the chain to demo-spec2.json in node1.toml

One copy demo-spec.json to demo-spec2.json

Demo-spec.json



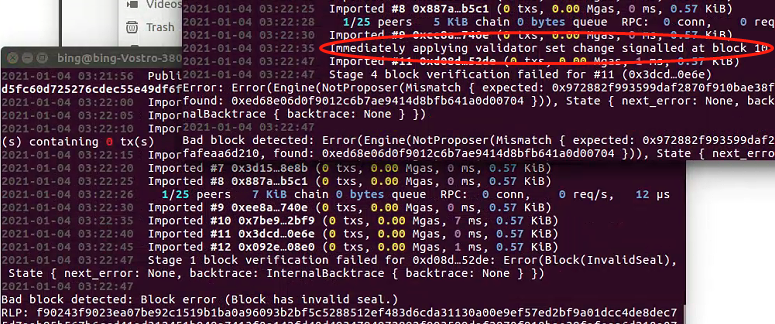
Demo-spec2.json



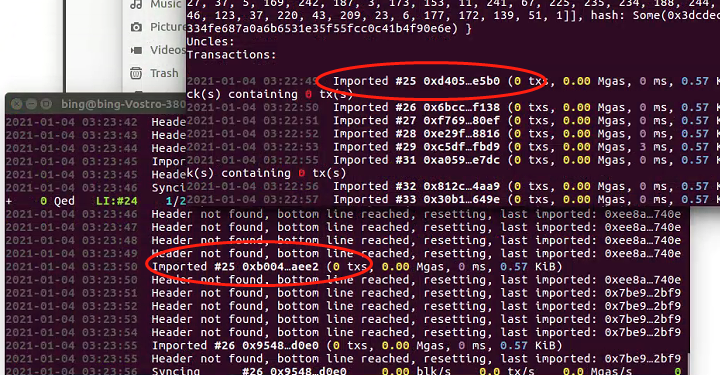
And add a new validator to Demo-spec2.json in the above case, it adds 0x97 as the validator and let this validator start to seal blocks from block 2, but you could set it to other block. And in this case before block 2 these 2 node are under the same block, and after block 2 they fork to 2 blockchain networks

How to check

Fork from block 2



After Fork



The block hash is different