Private Networks

Reasonably simple to set up from command line, but tools are also available

See operating a private network (https://github.com/ethereum/go-ethereum#operating-a-private-network)

Defining the private genesis state

First, you'll need to create the genesis state of your networks, which all nodes need to be aware of and agree upon. This consists of a small JSON file (e.g. call it genesis.json):

{
“config”: {
“chainId”: <arbitrary positive integer>,
“homesteadBlock”: 0,
“eip150Block”: 0,
“eip155Block”: 0,
“eip158Block”: 0,
“byzantiumBlock”: 0,
“constantinopleBlock”: 0,
“petersburgBlock”: 0,
“istanbulBlock”: 0,
“berlinBlock”: 0,
“londonBlock”: 0
},
“alloc”: {},
“coinbase”: “0x0000000000000000000000000000000000000000”,
“difficulty”: “0x20000”,
“extraData”: “”,
“gasLimit”: “0x2fefd8”,
“nonce”: “0x0000000000000042”,
“mixhash”:
“0x0000000000000000000000000000000000000000000000000000000000000000”,
“parentHash”:
“0x0000000000000000000000000000000000000000000000000000000000000000”,
“timestamp”: “0x00”
}

The above fields should be fine for most purposes, although we'd recommend changing the nonce to some random value so you prevent unknown remote nodes from being able to connect to you. If you'd like to pre-fund some accounts for easier testing, create the accounts and populate the alloc field with their addresses.

“alloc”: {
“0x0000000000000000000000000000000000000001”: {
“balance”: “111111111”
},
“0x0000000000000000000000000000000000000002”: {
“balance”: “222222222”
}
}

With the genesis state defined in the above JSON file, you’ll need to initialize every geth node with it prior to starting it up to ensure all blockchain parameters are correctly set:

```
$ geth init path/to/genesis.json
```

Creating the rendezvous point

With all nodes that you want to run initialized to the desired genesis state, you’ll need to start a bootstrap node that others can use to find each other in your network and/or over the internet. The clean way is to configure and run a dedicated bootnode:

```
$ bootnode --genkey=boot.key
$ bootnode --nodekey=boot.key
```

With the bootnode online, it will display an enode URL that other nodes can use to connect to it and exchange peer information. Make sure to replace the displayed IP address information (most probably [::]) with your externally accessible IP to get the actual enode URL.

Note: You could also use a full-fledged geth node as a bootnode, but it’s the less recommended way.

Starting up your member nodes

With the bootnode operational and externally reachable (you can try telnet <ip> <port> to ensure it’s indeed reachable), start every subsequent geth node pointed to the bootnode for peer discovery via the --bootnodes flag. It will probably also be desirable to keep the data directory of your private network separated, so do also specify a custom --datadir flag.

```
$ geth --datadir=path/to/custom/data/folder --bootnodes=<bootnode-enode-url-fro
```

Note: Since your network will be completely cut off from the main and test networks, you’ll also need to configure a miner to process transactions and create new blocks for you.

Running a private miner

Mining on the public Ethereum network is a complex task as it's only feasible using GPUs, requiring an OpenCL or CUDA enabled ethminer instance. For information on such a setup, please consult the EtherMining subreddit and the ethminer repository.

In a private network setting, however a single CPU miner instance is more than enough for practical purposes as it can produce a stable stream of blocks at the correct intervals without needing heavy resources (consider running on a single thread, no need for multiple ones either).

To start a geth instance for mining, run it with all your usual flags, extended by:

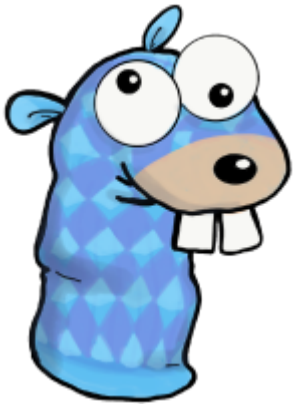
```
$ geth <usual-flags> --mine --miner.threads=1 --miner.etherbase=0x0000000000000
```

Which will start mining blocks and transactions on a single CPU thread, crediting all proceedings to the account specified by --miner.etherbase.

You can further tune the mining by changing the default gas limit blocks converge to (–miner.targetgaslimit) and the price transactions are accepted at (–miner.gasprice).

Geth Command Line Options (https://geth.ethereum.org/docs/interface/command-line-options)

Puppeth



Included with geth build
CLI wizard to help in creaion of genesis file

Mining only when there are pending transactions

(Run in geth console)

```
var mining_threads = 1

function checkWork() {
    if (eth.getBlock("pending").transactions.length > 0) {
        if (eth.mining) return;
        console.log("== Pending transactions! Mining...");
        miner.start(mining_threads);
    } else {
        miner.stop();
        console.log("== No transactions! Mining stopped.");
    }
}

eth.filter("latest", function(err, block) { checkWork(); });
eth.filter("pending", function(err, block) { checkWork(); });

checkWork
```

Also see https://www.npmjs.com/package/eth-mine-when-need (https://www.npmjs.com/package/eth-mine-when-need)