

Blockchain Training



An initiative of the Lagos State Ministry of Education

Week 2

Creating a Private Ethereum Blockchain

- Every transaction on the Ethereum Virtual Machine costs money
- You need a safe offline environment for testing your applications
- The options to do so are:
 - 1)Geth
 - 2)Eth
 - 3)Pyethapp

Geth



- Geth is the the command line interface for running a full Ethereum node
- It allows you setup a private Ethereum blockchain
- Written in Go

Geth Use Cases



- Mining Ethereum blocks
- Create and manage accounts
- Create contracts

Things to Define before Starting

- Custom Genesis File
- Custom Data Directory (C:\Geth)
- Custom Network ID (1217. LAG in Numbers)
- Disable Node Discovery

Creating the Genesis Block

```
{  
  "nonce"           : "0x00000000000000055",  
  "mixHash"         : "0x0000000000000000000000000000000000000000000000000000000000000000",  
  "parentHash"      : "0x0000000000000000000000000000000000000000000000000000000000000000",  
  "difficulty"      : "0x20000",  
  "gasLimit"        : "0x800000",  
  "timestamp"       : "0x0",  
  "extraData"       : "",  
  "coinbase"        : "0x0000000000000000000000000000000000000000000000000000000000000000",  
  "alloc"           : {},  
  "config"          : {  
    "chainId": 100,  
    "homesteadBlock": 0,  
    "eip155Block": 0,  
    "eip158Block": 0  
  }  
}
```

Save file in C:\Geth\configs\
genesis.json

What the Parameters Mean

mixhash A 256-bit hash which proves, combined with the `nonce`, that a sufficient amount of computation has been carried out on this block: the Proof-of-Work (PoW). The combination of `nonce` and `mixhash` must satisfy a mathematical condition described in the Yellowpaper, 4.3.4. Block Header Validity, (44). It allows to verify that the Block has really been cryptographically mined, thus, from this aspect, is valid.

nonce A 64-bit hash, which proves, combined with the mix-hash, that a sufficient amount of computation has been carried out on this block: the Proof-of-Work (PoW). The combination of `nonce` and `mixhash` must satisfy a mathematical condition described in the Yellowpaper, 4.3.4. Block Header Validity, (44), and allows to verify that the Block has really been cryptographically mined and thus, from this aspect, is valid. The `nonce` is the cryptographically secure mining proof-of-work that proves beyond reasonable doubt that a particular amount of computation has been expended in the determination of this token value. (Yellowpaper, 11.5. Mining Proof-of-Work).

difficulty A scalar value corresponding to the difficulty level applied during the `nonce` discovering of this block. It defines the mining Target, which can be calculated from the previous block's difficulty level and the timestamp. The higher the difficulty, the statistically more calculations a Miner must perform to discover a valid block. This value is used to control the Block generation time of a Blockchain, keeping the Block generation frequency within a target range. On the test network, we keep this value low to avoid waiting during tests, since the discovery of a valid Block is required to execute a transaction on the Blockchain.

alloc Allows defining a list of pre-filled wallets. That's an Ethereum specific functionality to handle the "Ether pre-sale" period. Since we can mine local Ether quickly, we don't use this option.

What the Parameters Mean

coinbase The 160-bit address to which all rewards (in Ether) collected from the successful mining of this block have been transferred. They are a sum of the mining reward itself and the Contract transaction execution refunds. Often named "beneficiary" in the specifications, sometimes "etherbase" in the online documentation. This can be anything in the Genesis Block since the value is set by the setting of the Miner when a new Block is created.

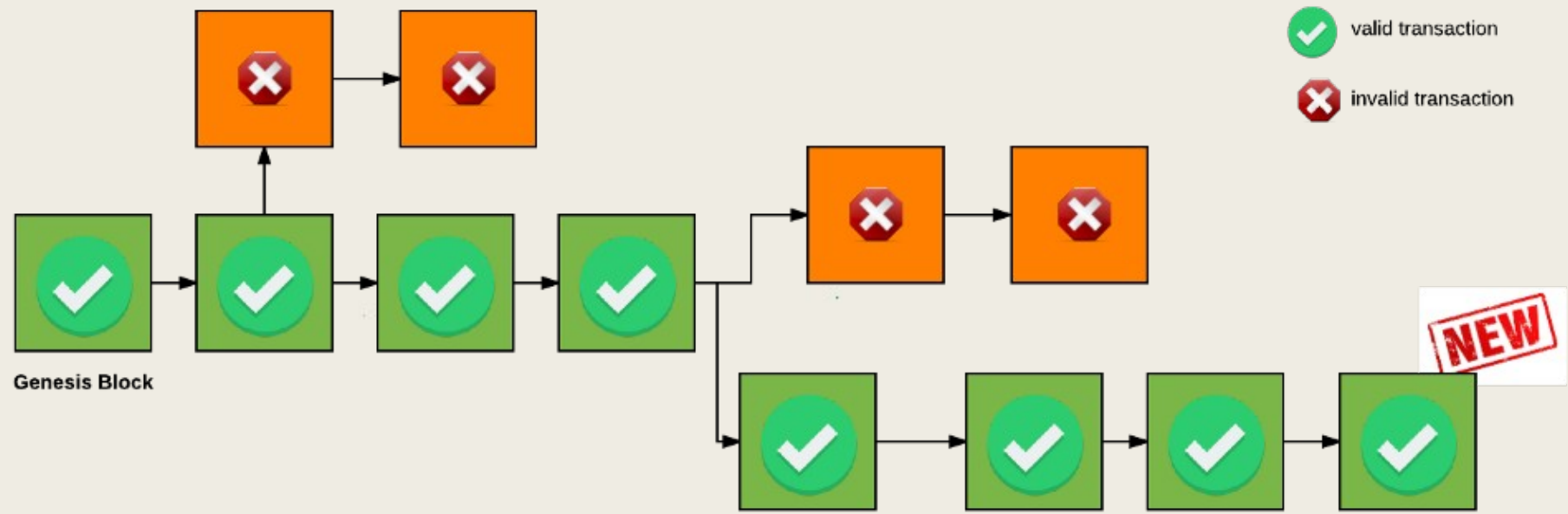
timestamp A scalar value equal to the reasonable output of Unix `time()` function at this block inception. This mechanism enforces a homeostasis in terms of the time between blocks. A smaller period between the last two blocks results in an increase in the difficulty level and thus additional computation required to find the next valid block. If the period is too large, the difficulty, and expected time to the next block, is reduced. The timestamp also allows verifying the order of block within the chain (Yellowpaper, 4.3.4. (43)).

parentHash The Keccak 256-bit hash of the entire parent block header (including its `nonce` and `mixhash`). Pointer to the parent block, thus effectively building the chain of blocks. In the case of the Genesis block, and only in this case, it's `0`.

extraData An optional free, but max. 32-byte long space to conserve smart things for eternity. :)

gasLimit A scalar value equal to the current chain-wide limit of Gas expenditure per block. High in our case to avoid being limited by this threshold during tests. Note: this does not indicate that we should not pay attention to the Gas consumption of our Contracts.

What is the Genesis Block

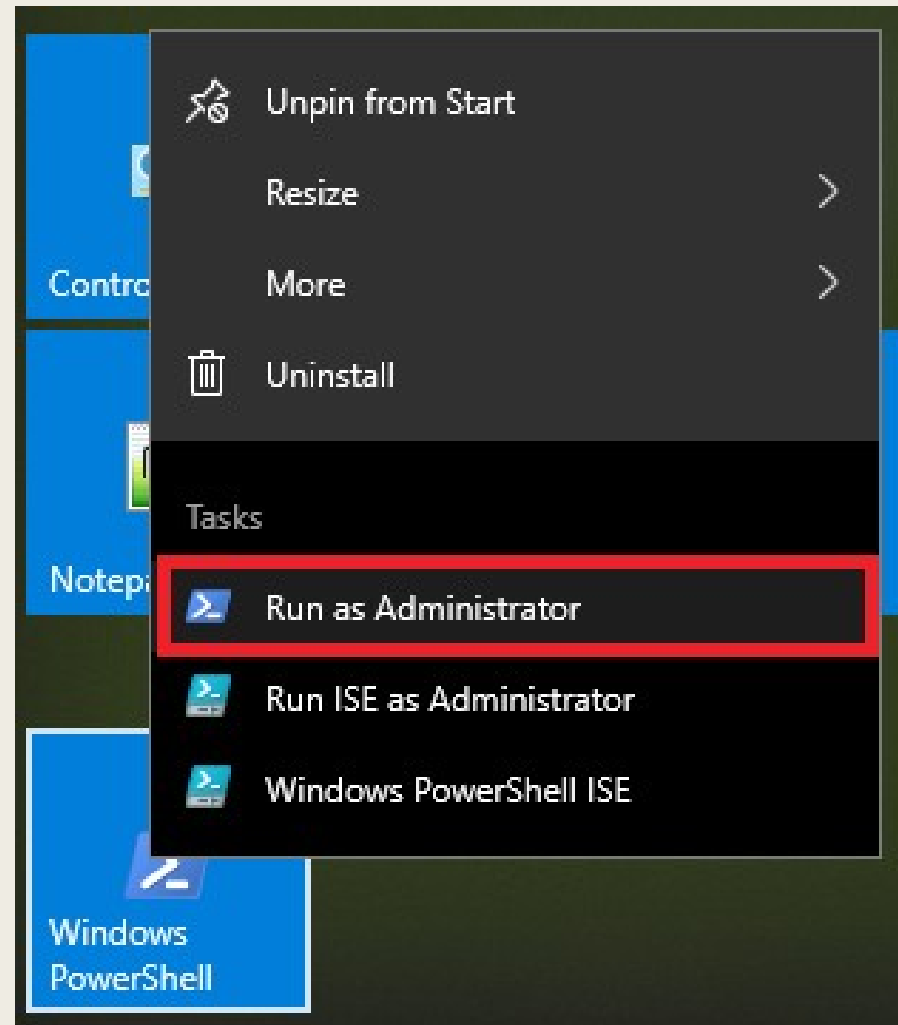


It is the first block of the blockchain that has no predecessors

Initialize Geth to Create the Genesis Block

- genesis.json is in "C:\Geth\configs\genesis.json"
- datadir is "C:\Geth\data-private"
- Using Windows, run Command Prompt or Powershell with Administrator privileges

Initialize Geth to Create the Genesis Block



Initialize Geth to Create the Genesis Block

Execute geth with the parameters—datadir and init:

```
geth --datadir "C:\Geth\data-private" init "C:\Geth\configs\genesis.json"
```

```
PS C:\WINDOWS\system32> geth --datadir "C:\Geth\data-private" init "C:\Geth\configs\genesis.json"
INFO [06-14|23:07:30] Maximum peer count          ETH=25 LES=0 total=25
INFO [06-14|23:07:30] Allocated cache and file handles database=C:\\Geth\\data-private\\geth\\chaindata
cache=16 handles=16
INFO [06-14|23:07:30] Writing custom genesis block
INFO [06-14|23:07:30] Persisted trie from memory database nodes=0 size=0.00B time=0s gcnodes=0 gcsiz=0.00B
gctime=0s livenodes=1 livesize=0.00B
INFO [06-14|23:07:30] Successfully wrote genesis state database=chaindata
hash=a37d01...1029e9
INFO [06-14|23:07:30] Allocated cache and file handles database=C:\\Geth\\data-private\\geth\\lightchain
data cache=16 handles=16
INFO [06-14|23:07:31] Writing custom genesis block
INFO [06-14|23:07:31] Persisted trie from memory database nodes=0 size=0.00B time=0s gcnodes=0 gcsiz=0.00B
gctime=0s livenodes=1 livesize=0.00B
INFO [06-14|23:07:31] Successfully wrote genesis state database=lightchaindata
hash=a37d01...1029e9
PS C:\WINDOWS\system32>
```

Creating an Account at Geth

Run Geth with the following parameters:

```
geth --networkid 1217 --port 60303 --rpc --lightkdf --cache 16 --datadir "C:\Geth\data-private" console
```

```
PS C:\WINDOWS\system32> geth --networkid 1217 --port 60303 --rpc --lightkdf --cache 16 --datadir "C:\Geth\data-private" console
INFO [06-14|23:28:09] Maximum peer count               ETH=25 LE=0 tota=25
INFO [06-14|23:28:09] Starting peer-to-peer node       instance=Geth/v1.8.10-stable-eae63c51/windows-amd64/go1.10.2
INFO [06-14|23:28:09] Allocated cache and file handles database=C:\Geth\data-private\geth\chaindata cache=16 handle=1024
INFO [06-14|23:28:09] Initialised chain configuration  config="{ChainID: 100 Homestead: 0 DAO: <nil> DAOSupport: false EIP150: <nil> EIP155: 0 EIP158: 0 Byzantium: <nil> Constantinople: <nil> Engine: unknown}"
INFO [06-14|23:28:09] Disk storage enabled for ethash caches dir=C:\Geth\data-private\geth\ethash count=3
INFO [06-14|23:28:09] Disk storage enabled for ethash DAGs dir="C:\Users\Iruston Ailende\AppData\Ethash" count=2
INFO [06-14|23:28:09] Initialising Ethereum protocol  versions="[63 62]" network=1217
INFO [06-14|23:28:09] Loaded most recent local header  number=0 hash=a37d01..1029e9 id=131072
INFO [06-14|23:28:09] Loaded most recent local full block number=0 hash=a37d01..1029e9 id=131072
INFO [06-14|23:28:09] Loaded most recent local fast block number=0 hash=a37d01..1029e9 id=131072
INFO [06-14|23:28:09] Loaded local transaction journal transactions=0 dropped=0
INFO [06-14|23:28:09] Regenerated local transaction journal transactions=0 account=0
INFO [06-14|23:28:09] Starting P2P networking
INFO [06-14|23:28:11] UDP listener up                  self=enode://ef297e4bc71d58f75c2b906f3685d5db8832bbcc45ca35d9928d7a21647d4d24e88d70d7c7917613bc6cca860fee97c338904ecf01cac2cd8ede
ebf423929236@[::]:60303
INFO [06-14|23:28:12] RLPx listener up                self=enode://ef297e4bc71d58f75c2b906f3685d5db8832bbcc45ca35d9928d7a21647d4d24e88d70d7c7917613bc6cca860fee97c338904ecf01cac2cd8ede
ebf423929236@[::]:60303
INFO [06-14|23:28:12] IPC endpoint opened              url=\\\\.\\pipe\\geth.ipc
INFO [06-14|23:28:12] HTTP endpoint opened             url=http://127.0.0.1:8545 cors= vhosts=localhost
Welcome to the Geth JavaScript console!

instance: Geth/v1.8.10-stable-eae63c51/windows-amd64/go1.10.2
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
>
```

Create New Account



```
> personal.newAccount()
```

```
Passphrase:
```

```
Repeat passphrase:
```

```
"0xadd47a2fa6f82e5e9282aafdad9e50859304f56"
```

Check Your Account

```
> personal
{
  listAccounts: ["0xadd47a2fa6f82e5e9282aafdad9e50859304f56"],
  listWallets: [{
    accounts: [{...}],
    status: "Locked",
    url: "keystore://C:\\Geth\\data-private\\keystore\\UTC--2018-06-14T22-32-15.562367900Z--add47a2fa6f82e5e9282aafdad9e50859304f56"
  }],
  deriveAccount: ,
  ecRecover: ,
  getListAccounts: ,
  getListWallets: ,
  importRawKey: ,
  lockAccount: ,
  newAccount: ,
  openWallet: ,
  sendTransaction: ,
  sign: ,
  signTransaction: ,
  unlockAccount:
}
>
>
```

Configure enode



- An enode is a way to describe an Ethereum node in the form of a URI, separated from the host by an @ sign
- At geth console, run this command and verify the node information:

```
> admin.nodeInfo.enode  
"enode://ef297e4bc71d58f75c2b906f3685d5db8832bbcc45ca35d9928d  
7a21647d4d24e88d70d7c7917613bc6cca860fee97c338904ecf01cac2cd8  
edeefb423929236@[::]:60303"
```

IP and Port for enode



```
> admin.nodeInfo.enode  
"enode://ef297e4bc71d58f75c2b906f3685d5db8832bbcc45ca35d9928d  
7a21647d4d24e88d70d7c7917613bc6cca860fee97c338904ecf01cac2cd8  
edeebf423929236@[::]:60303"
```

- The port underlined is the port defined when we execute Geth

Exit The Blockchain



```
> exit
INFO [06-15|11:35:10] HTTP endpoint closed      url=http://127.0.0.1:8545
INFO [06-15|11:35:10] IPC endpoint closed        endpoint=\\\\.\\pipe\\geth.ipc
INFO [06-15|11:35:10] Blockchain manager stopped
INFO [06-15|11:35:10] Stopping Ethereum protocol
INFO [06-15|11:35:10] Ethereum protocol stopped
INFO [06-15|11:35:10] Transaction pool stopped
INFO [06-15|11:35:11] Database closed           database=C:\\Geth\\data-private\\geth\\chaindata
PS C:\\WINDOWS\\system32>
```

Your Private Blockchain is Ready



Congratulations

Conclusion



- A private Ethereum blockchain provides a staging area for our ideas
- You can do everything that is possible on the live Ethereum network on your private blockchain

Mining Ether on Your Private Ethereum Blockchain

- Last time you created a private blockchain
- To work with a blockchain, you must have ether
- Mining is the process of creating ether

Recreate Your Blockchain

- Delete the files in your Geth folder and create the folder structure shown below:

 config	16/06/2018 8:34 PM	File folder
 node1	16/06/2018 8:34 PM	File folder
 node2	16/06/2018 8:34 PM	File folder

Modified Genesis File

```
{
  "alloc": {},
  "config": {
    "homesteadBlock": 0,
    "chainID": 72,
    "eip155Block": 0,
    "eip158Block": 0
  },
  "nonce": "0x0000000000000000",
  "difficulty": "0x4000",
  "mixhash":
  "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x000000000000000000000000000000000000000000000000",
  "timestamp": "0x00",
  "parentHash":
  "0x0000000000000000000000000000000000000000000000000000000000000000",
  "extraData":
  "0x0000000000000000000000000000000000000000000000000000000000000000",
  "gasLimit": "0xffffffff"
}
```

Note that the difficulty has been reduced

Create Nodes



- Run the code shown below to create the first node:

```
geth --datadir "C:\Geth\node1" init "C:\Geth\config\genesis.json"
```

- Run the code shown below to create the second node:

```
geth --datadir "C:\Geth\node2" init "C:\Geth\config\genesis.json"
```



Create Nodes



```
PS C:\WINDOWS\system32> geth --datadir "C:\Geth\node1" init "C:\Geth\config\genesis.json"
INFO [06-16|20:58:22] Maximum peer count                ETH=25 LES=0 total=25
INFO [06-16|20:58:22] Allocated cache and file handles  database=C:\\Geth\\node1\\geth\\chaindata cache=16 handles=16
INFO [06-16|20:58:23] Writing custom genesis block
INFO [06-16|20:58:23] Persisted trie from memory database nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [06-16|20:58:23] Successfully wrote genesis state   database=chaindata hash=62d4be...0e3bb5
INFO [06-16|20:58:23] Allocated cache and file handles  database=C:\\Geth\\node1\\geth\\lightchaindata cache=16 handles=16
INFO [06-16|20:58:23] Writing custom genesis block
INFO [06-16|20:58:23] Persisted trie from memory database nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [06-16|20:58:23] Successfully wrote genesis state   database=lightchaindata hash=62d4be...0e3bb5
PS C:\WINDOWS\system32> geth --datadir "C:\Geth\node2" init "C:\Geth\config\genesis.json"
INFO [06-16|21:01:24] Maximum peer count                ETH=25 LES=0 total=25
INFO [06-16|21:01:24] Allocated cache and file handles  database=C:\\Geth\\node2\\geth\\chaindata cache=16 handles=16
INFO [06-16|21:01:24] Writing custom genesis block
INFO [06-16|21:01:24] Persisted trie from memory database nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [06-16|21:01:24] Successfully wrote genesis state   database=chaindata hash=62d4be...0e3bb5
INFO [06-16|21:01:24] Allocated cache and file handles  database=C:\\Geth\\node2\\geth\\lightchaindata cache=16 handles=16
INFO [06-16|21:01:25] Writing custom genesis block
INFO [06-16|21:01:25] Persisted trie from memory database nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [06-16|21:01:25] Successfully wrote genesis state   database=lightchaindata hash=62d4be...0e3bb5
PS C:\WINDOWS\system32>
```

Open Your Consoles



- You need to open a second PowerShell
- In your first PowerShell, run this command:
`geth --datadir "C:\Geth\node1" --networkid 72 --port 30301 --nodiscover console`
- In your second PowerShell, run this command:
`geth --datadir "C:\Geth\node2" --networkid 72 --port 30302 --ipcdisable console`

Create Your Initial Coinbase

```
> personal.listAccounts  
[]  
> personal.newAccount()  
Passphrase:  
Repeat passphrase:  
"0x4af0f6ff915efde1847b253004c7f99c6949515a"  
>
```

Run this also in the second
node's console

Connecting Nodes as Peers

```
> admin.peers
[]
> admin.nodeInfo.enode
"enode://f056a7539194e2f0d78293f5e122b1807d47d607960dee4bdc9c82432d8dd717153934d2b7e1de831f9adee9dcef9f7ffa2237e8c2159fb1437e6f6c0422b869@[::]:30301?discport=0"
>
```

Run the above commands in
your first node's console

Connecting Nodes as Peers

```
> admin.addPeer("enode://f056a7539194e2f0d78293f6e122b1807d47d607960dee4bdc9c82432d8dd717153934d2b7e1de831f9adee9dcef9f7ffa2237e8c2159fb1437e6f6c0422b869@[:]:30301?discport=0")
true
> admin.peers
[[{
  caps: ["eth/63"],
  id: "f056a7539194e2f0d78293f6e122b1807d47d607960dee4bdc9c82432d8dd717153934d2b7e1de831f9adee9dcef9f7ffa2237e8c2159fb1437e6f6c0422b869",
  name: "Geth/v1.8.10-stable-eae63c51/windows-amd64/go1.10.2",
  network: {
    inbound: false,
    localAddress: "127.0.0.1:59697",
    remoteAddress: "127.0.0.1:30301",
    static: true,
    trusted: false
  },
  protocols: {
    eth: {
      difficulty: 16384,
      head: "0xc2d4beb598ef8db3870972a311bcb97a6878c6e940a763400284c48d470e3bb5",
      version: 63
    }
  }
}]
>
```

Run this command in your
second node's console



Check Your Balance and start Mining

```
> eth.getBalance(eth.coinbase)
[INFO [06-16|21:36:07] Etherbase automatically configured      address=0x4aF0F6Ff915eFde1847B253004C7f99c6949515A
0
> miner.start()
[INFO [06-16|21:37:21] Updated mining threads          threads=0
[INFO [06-16|21:37:21] Transaction pool price threshold updated price=18000000000
[INnFu01 1[0
6> -16|21:37:21] Starting mining operation
[INFO [06-16|21:37:21] Commit new mining work      number=1 txs=0 uncles=0 elapsed=8.009ms
```

Run this command in your
first node's console





Unlock Your Account



```
> coinbaseAddress = eth.coinbase
"0x4af0f6ff915efde1847b253004c7f99c6949515a"
> personal.unlockAccount(coinbaseAddress)
Unlock account 0x4af0f6ff915efde1847b253004c7f99c6949515a
Passphrase:
true
>
```

Run this command in your
first node's console



Transfer Wei



- Run the command in your second node's console to get its address:
`coinbaseAddress = eth.coinbase`
- Run the command shown below in your first node's console:

```
> hisAddress = "0x024c91b599fce287e3463102f8025f02787f6b9b"  
"0x024c91b599fce287e3463102f8025f02787f6b9b"  
>  
> eth.sendTransaction({from: eth.coinbase, to:hisAddress, value: 100000000})  
INFO [06-16|21:47:26] Submitted transaction fullhash=0x4d05600d3ba  
865c3ad82ef21fe068066ebd024c35b37bc450bb259ba4bf851bc recipient=0x024c91b599fCe287E34  
63102f8025f02787f6B9b  
"0x4d05600d3ba865c3ad82ef21fe068066ebd024c35b37bc450bb259ba4bf851bc"  
>
```

Transfer Wei



- To transfer Wei, you need to use the `miner.start()` command and stop it after some blocks have been mined

```
> miner.stop()
true
>
> web3.eth.getBalance(eth.coinbase)
5999999999999900000000
> web3.eth.getBalance(hisAddress)
100000000
>
```

Run this command in your first node's console



Confirm Transfer



- Confirm that you have received the funds transferred to your second node

```
> eth.getBalance(eth.coinbase)
INFO [06-16|21:51:53] Etherbase automatically configured   address=0x024c91b599FCe287E3463102f8025f02787f6B9b
1000000000
>
```

Run this command in your
second node's console



Conclusion



- This week we created a private Ethereum node
- We have mined Ether on our local machine
- We have a local setup to deploy our own smart contracts
- Ensure to finish the CryptoZombies lessons by next week