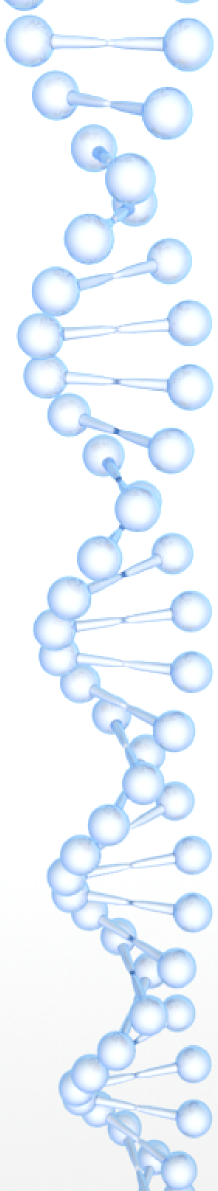
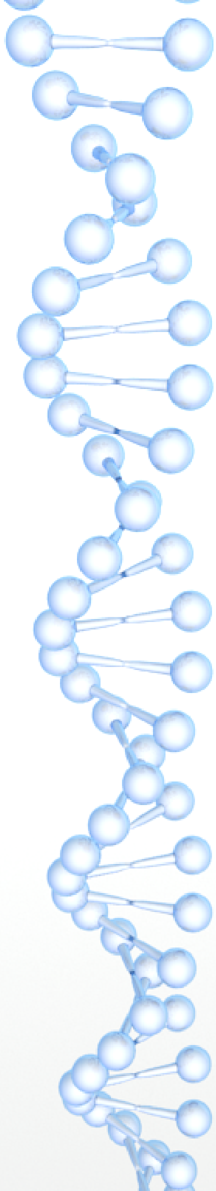


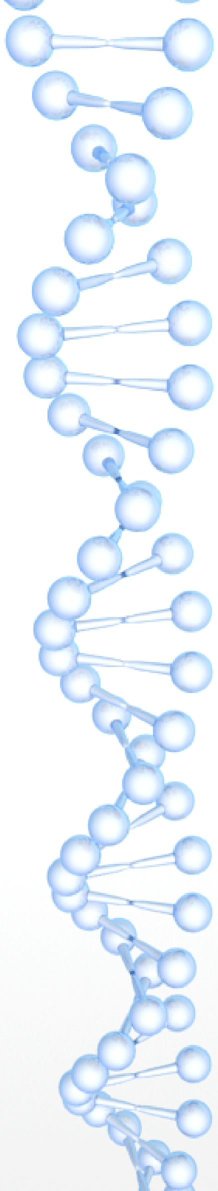
Running a Private Blockchain

Austin Hester
University of Missouri – St. Louis
01/28/18



Software Requirements

- Linux OS
- go-1.8.x+ `// add-apt-repository ppa:gophers/archive`
- go-ethereum-1.7.3-stable `// add-apt-repository ppa:ethereum/ethereum`
- Nodejs-1.8.x+
  - npm
    - truffle
    - testrpc
- Solidity compiler
- [opt] Mist browser
- [opt] VSCode w/ Solidity ext.



go-ethereum (geth)

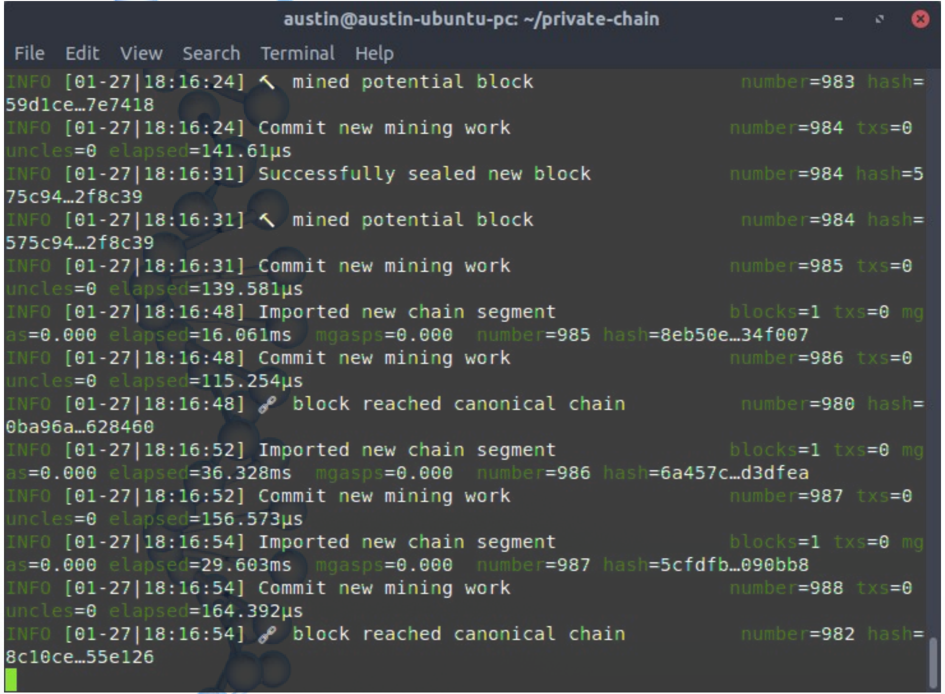
- What is it?
  - Geth is a command-line interface for running nodes on the Ethereum network.
- What is it used for?
  - Geth is widely used for development of **Decentralized applications** (Dapps).
  - Also used for deploying development / testing blockchains.
  - It’s a decent CPU miner.
- How can we use it?
  - Running a private blockchain with custom genesis block and testing Smart Contracts.

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

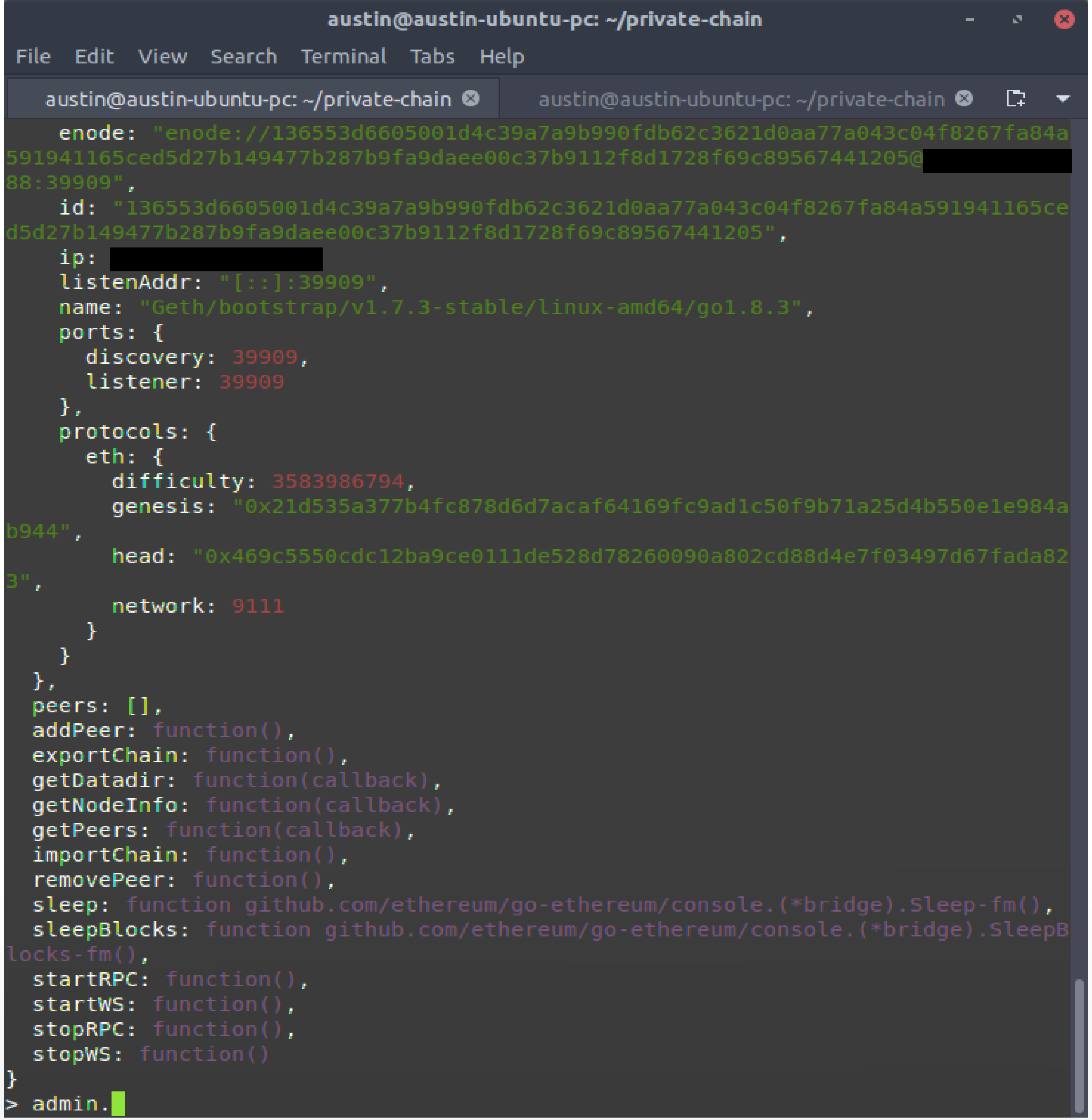
go-ethereum

- What does it look like?

Mining


austin@austin-ubuntu-pc: ~/private-chain
File Edit View Search Terminal Help

```
INFO [01-27|18:16:24] ⛏ mined potential block                number=983 hash=
59d1ce…7e7418
INFO [01-27|18:16:24] Commit new mining work                number=984 txs=0
uncles=0 elapsed=141.61µs
INFO [01-27|18:16:31] Successfully sealed new block         number=984 hash=5
75c94…2f8c39
INFO [01-27|18:16:31] ⛏ mined potential block                number=984 hash=
575c94…2f8c39
INFO [01-27|18:16:31] Commit new mining work                number=985 txs=0
uncles=0 elapsed=139.581µs
INFO [01-27|18:16:48] Imported new chain segment            blocks=1 txs=0 mg
as=0.000 elapsed=16.061ms  mgasps=0.000  number=985 hash=8eb50e…34f007
INFO [01-27|18:16:48] Commit new mining work                number=986 txs=0
uncles=0 elapsed=115.254µs
INFO [01-27|18:16:48] 🔗 block reached canonical chain        number=980 hash=
0ba96a…628460
INFO [01-27|18:16:52] Imported new chain segment            blocks=1 txs=0 mg
as=0.000 elapsed=36.328ms  mgasps=0.000  number=986 hash=6a457c…d3dfea
INFO [01-27|18:16:52] Commit new mining work                number=987 txs=0
uncles=0 elapsed=156.573µs
INFO [01-27|18:16:54] Imported new chain segment            blocks=1 txs=0 mg
as=0.000 elapsed=29.603ms  mgasps=0.000  number=987 hash=5cfdfb…090bb8
INFO [01-27|18:16:54] Commit new mining work                number=988 txs=0
uncles=0 elapsed=164.392µs
INFO [01-27|18:16:54] 🔗 block reached canonical chain        number=982 hash=
8c10ce…55e126
```

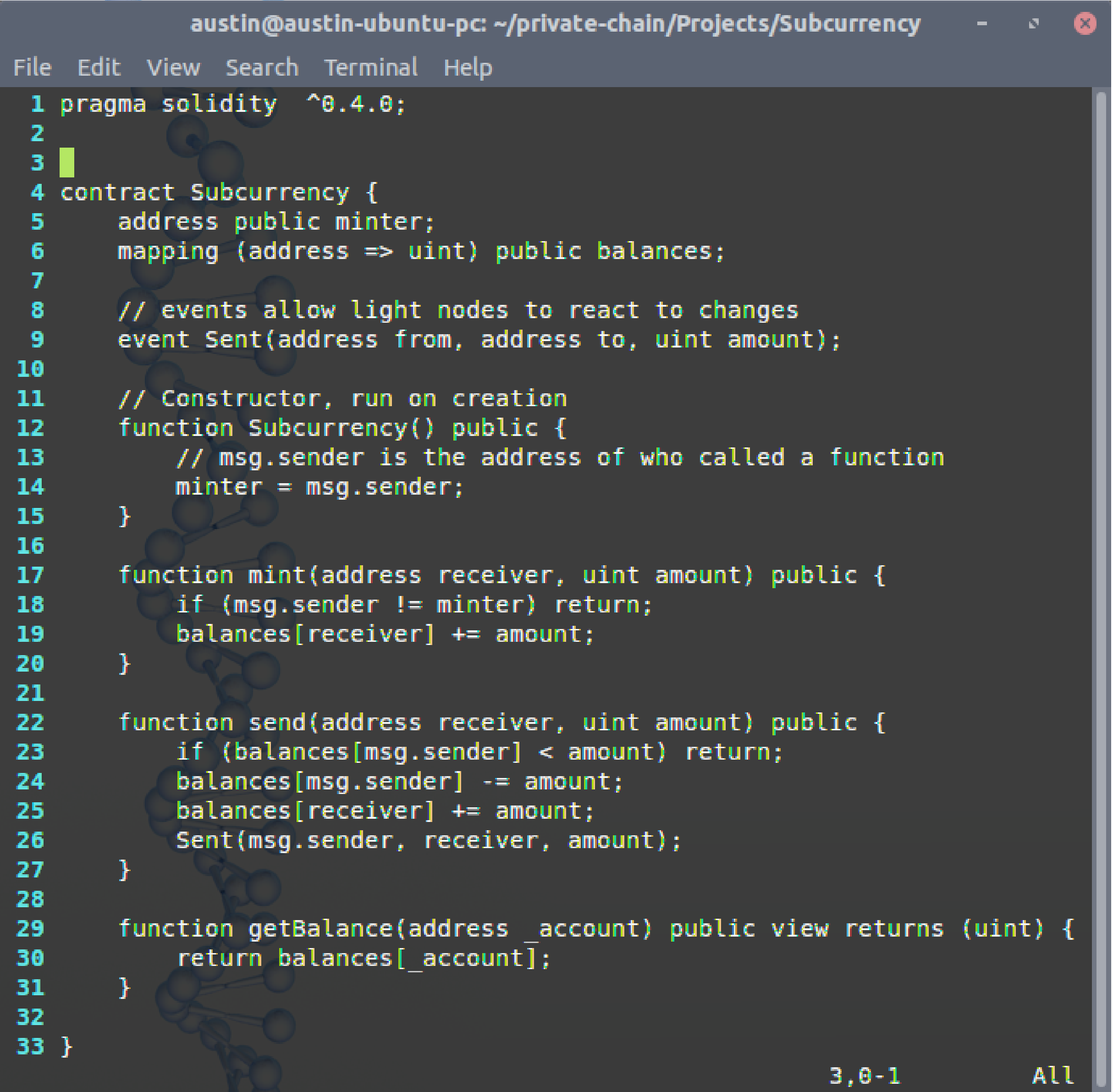

austin@austin-ubuntu-pc: ~/private-chain
File Edit View Search Terminal Tabs Help
austin@austin-ubuntu-pc: ~/private-chain
austin@austin-ubuntu-pc: ~/private-chain

```
    enode: "enode://136553d6605001d4c39a7a9b990fdb62c3621d0aa77a043c04f8267fa84a
591941165ced5d27b149477b287b9fa9daee00c37b9112f8d1728f69c89567441205@[redacted]
88:39909",
    id: "136553d6605001d4c39a7a9b990fdb62c3621d0aa77a043c04f8267fa84a591941165ce
d5d27b149477b287b9fa9daee00c37b9112f8d1728f69c89567441205",
    ip: [redacted]
    listenAddr: "[::]:39909",
    name: "Geth/bootstrap/v1.7.3-stable/linux-amd64/go1.8.3",
    ports: {
      discovery: 39909,
      listener: 39909
    },
    protocols: {
      eth: {
        difficulty: 3583986794,
        genesis: "0x21d535a377b4fc878d6d7acaf64169fc9ad1c50f9b71a25d4b550e1e984a
b944",
        head: "0x469c5550cdc12ba9ce0111de528d78260090a802cd88d4e7f03497d67fada82
3",
        network: 9111
      }
    }
  },
  peers: [],
  addPeer: function(),
  exportChain: function(),
  getDatadir: function(callback),
  getNodeInfo: function(callback),
  getPeers: function(callback),
  importChain: function(),
  removePeer: function(),
  sleep: function github.com/ethereum/go-ethereum/console.(*bridge).Sleep-fm(),
  sleepBlocks: function github.com/ethereum/go-ethereum/console.(*bridge).SleepB
locks-fm(),
  startRPC: function(),
  startWS: function(),
  stopRPC: function(),
  stopWS: function()
}
> admin.
```

Administration

/go-ethereum/Management-API

Solidity

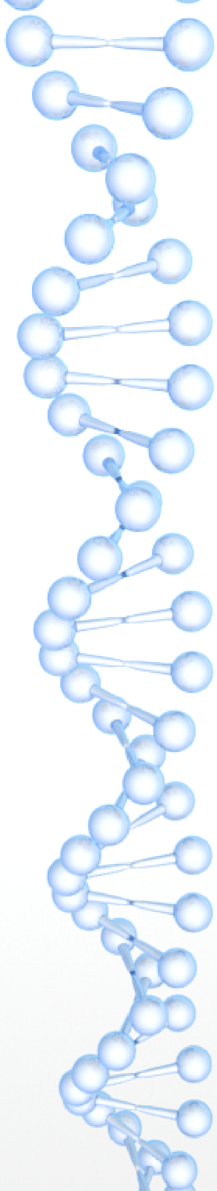

austin@austin-ubuntu-pc: ~/private-chain/Projects/Subcurrency
File
Edit
View
Search
Terminal
Help
3,0-1
All

```
pragma solidity  ^0.4.0;


contract Subcurrency {
    address public minter;
    mapping (address => uint) public balances;

    // events allow light nodes to react to changes
    event Sent(address from, address to, uint amount);

    // Constructor, run on creation
    function Subcurrency() public {
        // msg.sender is the address of who called a function
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        if (msg.sender != minter) return;
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        if (balances[msg.sender] < amount) return;
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        Sent(msg.sender, receiver, amount);
    }

    function getBalance(address _account) public view returns (uint) {
        return balances[_account];
    }

}
```

> “Solidity is a contract-oriented, high-level language for implementing smart contracts. It was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine.”

- Contracts exist in **contract accounts**, while user accounts are called **external accounts**.
- Contract accounts are controlled by **code**, while external accounts are controlled by public-private **key pairs**.
- Ex: Simple contract which maps a `balance` to an `address`.
    - Event `Sent` for watching contract.
    - Minter can `mint` tokens to a `receiver`
    - Users can `send` tokens to others.
    - You can also `getBalance` of an address.

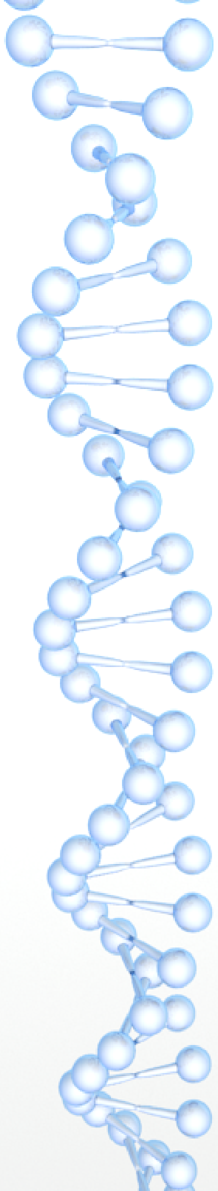
https://solidity.readthedocs.io/en/develop/



Truffle

- What is it?
  - Truffle is a development environment for Ethereum contracts.
- What is it used for?
  - Smart contract compilation / testing
  - Contract deployment
  - Interactive console for using the contract
  - Install packages via Ethereum package manager (EthPM).
- Using `testrpc` we can deploy contracts on a temporary blockchain.

https://github.com/trufflesuite/truffle



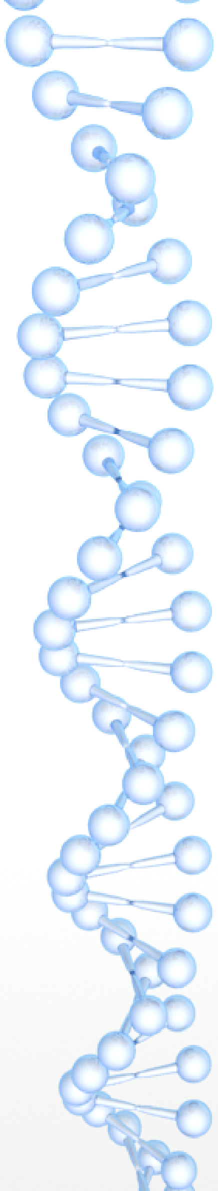
Truffle

- How can we use it?

```
$ mkdir SmartToken && cd SmartToken
$ truffle init                  // initialize
$ truffle compile
$ truffle migrate --reset       // deploy contract on test network
$ truffle console               // start the console

truffle> SmartToken.address
truffle> JSON.stringify(SmartToken.abi)
```

- The above two lines retrieve the information needed to interact with your contract on the Mist browser.

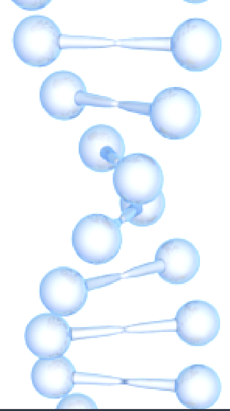
https://github.com/trufflesuite/truffle



Mist Browser

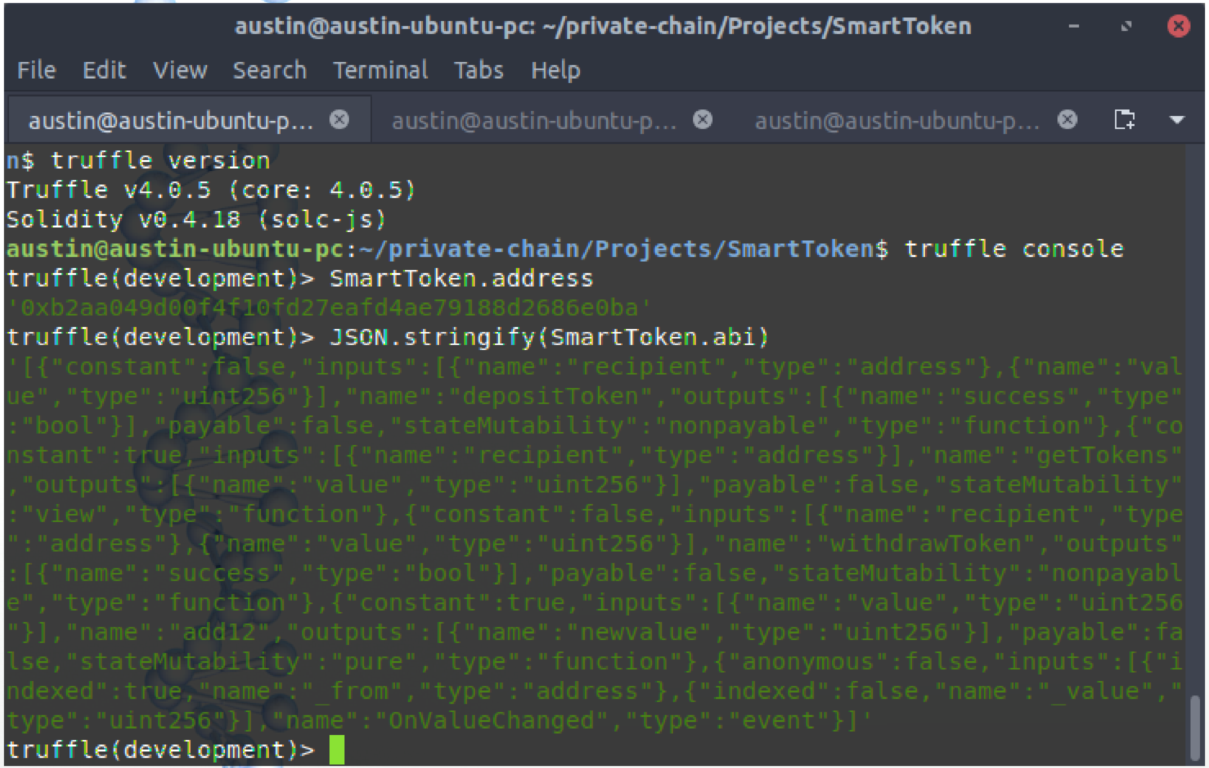
- Mist browser is a GUI for monitoring and manipulating your private blockchain.
- You can link to a node by opening Mist using its rpc port.
- Mist browser doubles as an Ethereum wallet.
- You can:
  - Send / receive transactions.
  - Watch smart contract activity.

https://github.com/ethereum/mist

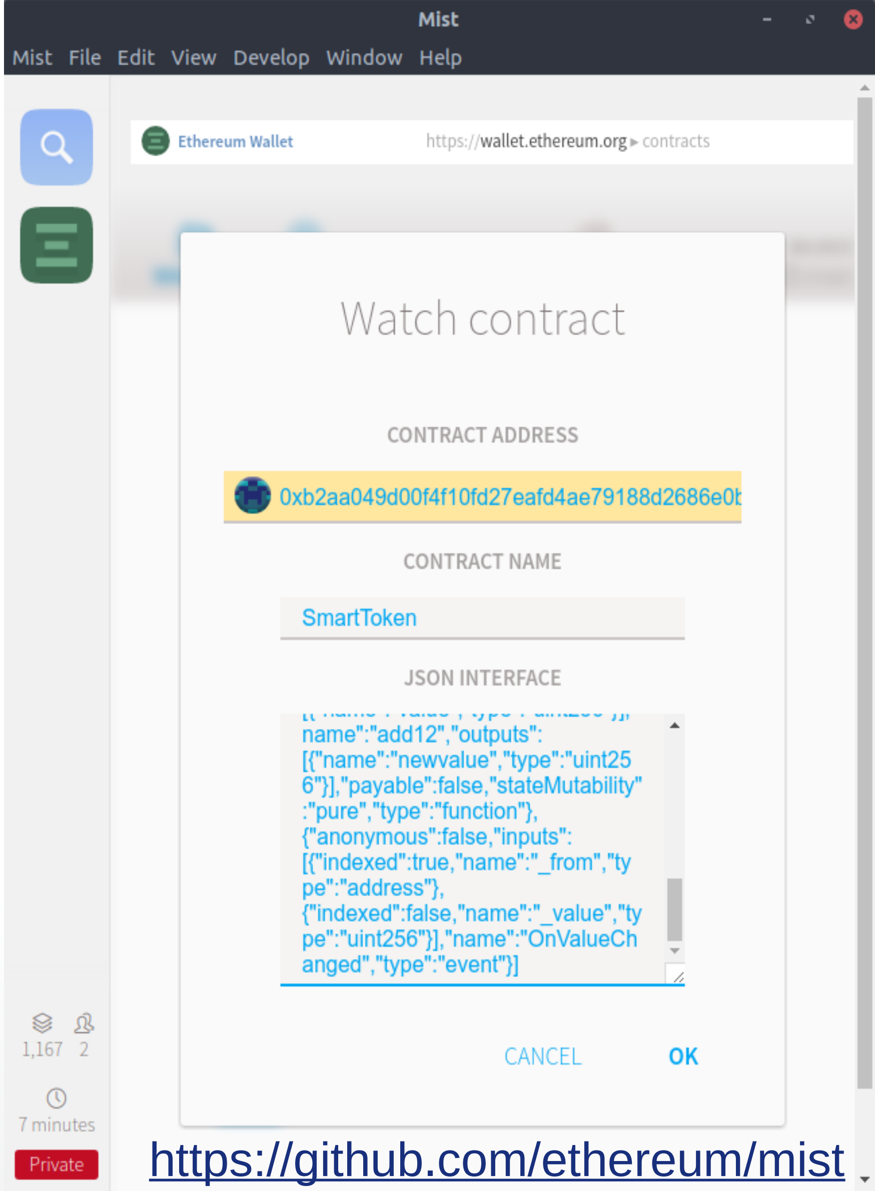


Mist Browser

You can watch a contract for events by giving it the `address` and `JSON interface` of the contract.


austin@austin-ubuntu-pc: ~/private-chain/Projects/SmartToken
File Edit View Search Terminal Tabs Help
austin@austin-ubuntu-p...
austin@austin-ubuntu-p...
austin@austin-ubuntu-p...

```
n$ truffle version
Truffle v4.0.5 (core: 4.0.5)
Solidity v0.4.18 (solc-js)
austin@austin-ubuntu-pc:~/private-chain/Projects/SmartToken$ truffle console
truffle(development)> SmartToken.address
'0xb2aa049d00f4f10fd27eafd4ae79188d2686e0ba'
truffle(development)> JSON.stringify(SmartToken.abi)
'[{"constant":false,"inputs":[{"name":"recipient","type":"address"},{"name":"val
ue","type":"uint256"}],"name":"depositToken","outputs":[{"name":"success","type"
:"bool"}],"payable":false,"stateMutability":"nonpayable","type":"function"},{"co
nstant":true,"inputs":[{"name":"recipient","type":"address"}],"name":"getTokens"
,"outputs":[{"name":"value","type":"uint256"}],"payable":false,"stateMutability"
:"view","type":"function"},{"constant":false,"inputs":[{"name":"recipient","type
":"address"},{"name":"value","type":"uint256"}],"name":"withdrawToken","outputs"
:[{"name":"success","type":"bool"}],"payable":false,"stateMutability":"nonpayabl
e","type":"function"},{"constant":true,"inputs":[{"name":"value","type":"uint256
"}],"name":"add12","outputs":[{"name":"newvalue","type":"uint256"}],"payable":fa
lse,"stateMutability":"pure","type":"function"},{"anonymous":false,"inputs":[{"i
ndexed":true,"name":"_from","type":"address"},{"indexed":false,"name":"_value","
type":"uint256"}],"name":"OnValueChanged","type":"event"}]'
truffle(development)>
```


Mist
Mist File Edit View Develop Window Help
Ethereum Wallet
https://wallet.ethereum.org ▸ contracts
1,167
2
7 minutes
Private

Watch contract

CONTRACT ADDRESS

0xb2aa049d00f4f10fd27eafd4ae79188d2686e0b

CONTRACT NAME

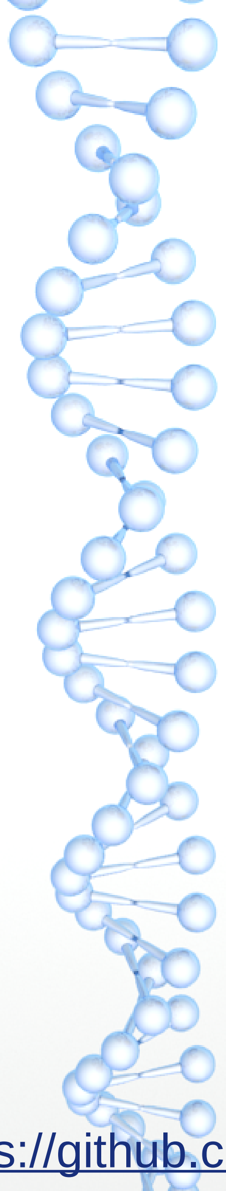
SmartToken

JSON INTERFACE

```
name":"add12","outputs":
[{"name":"newvalue","type":"uint25
6"}],"payable":false,"stateMutability"
:"pure","type":"function"},
{"anonymous":false,"inputs":
[{"indexed":true,"name":"_from","ty
pe":"address"},
{"indexed":false,"name":"_value","ty
pe":"uint256"}],"name":"OnValueCh
anged","type":"event"}]
```

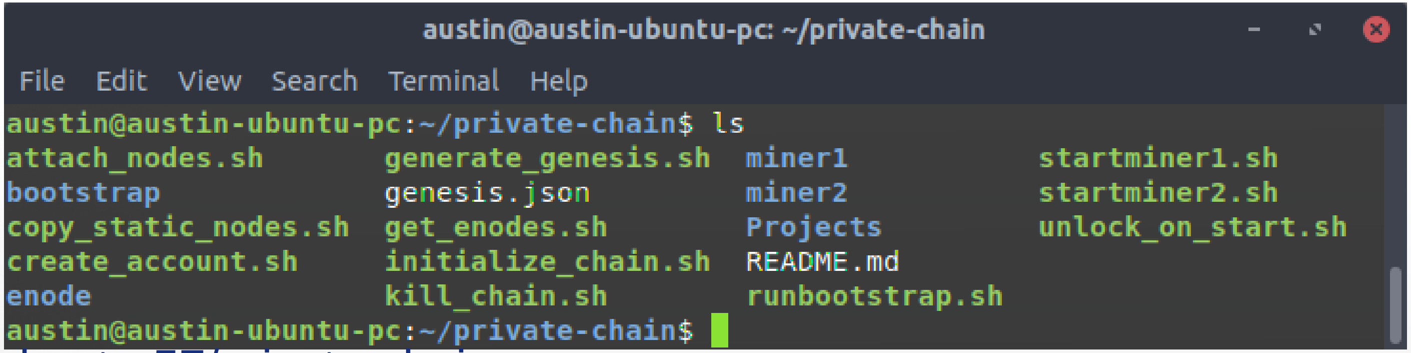
CANCEL OK

https://github.com/ethereum/mist



My Scripts for geth

- **`generate_genesis.sh`** *// generates* `genesis.json` *for initialization (recommend changes)*
- **`initialize_chain.sh`** *// initialize blockchain with generated genesis block*
- **`create_account.sh`** *// create 2 accounts for each node*
- **`unlock_on_start.sh`** *// create* `passwd.sec` *in datadirs*
- **`get_enodes.sh`** & **`copy_static_nodes.sh`** *// retrieve* `enodes`*, required for syncing nodes*
- **`runbootstrap.sh`** & **`startminer[1,2].sh`** *// for starting each*
- **`attach_nodes.sh`** *// attach to node and open console*
- **`kill_chain.sh`** *// kill all instances of* `geth`

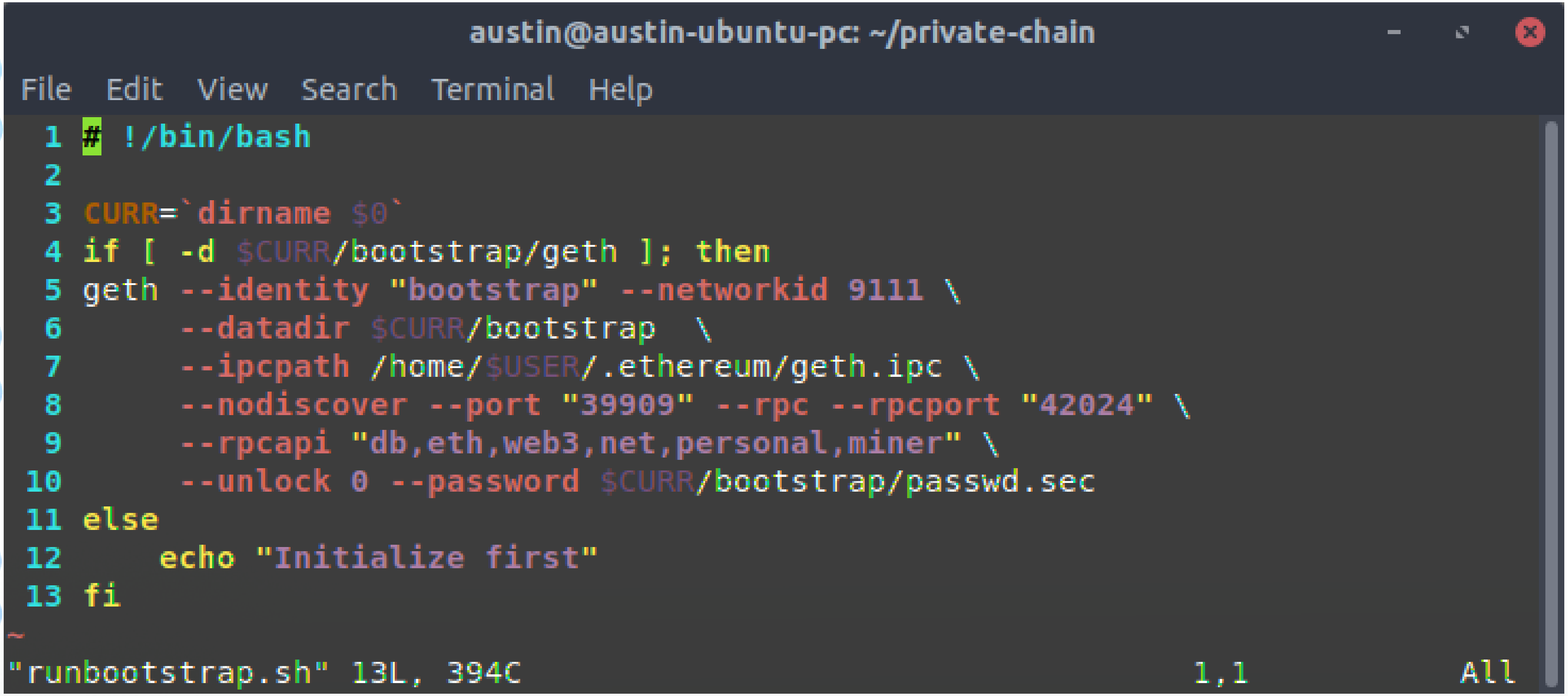


austin@austin-ubuntu-pc: ~/private-chain
File Edit View Search Terminal Help

```
austin@austin-ubuntu-pc:~/private-chain$ ls
attach_nodes.sh        generate_genesis.sh  miner1            startminer1.sh
bootstrap              genesis.json         miner2            startminer2.sh
copy_static_nodes.sh   get_enodes.sh        Projects          unlock_on_start.sh
create_account.sh      initialize_chain.sh  README.md
enode                  kill_chain.sh        runbootstrap.sh
austin@austin-ubuntu-pc:~/private-chain$ 
```

https://github.com/ahester57/private-chain

runbootstrap.sh

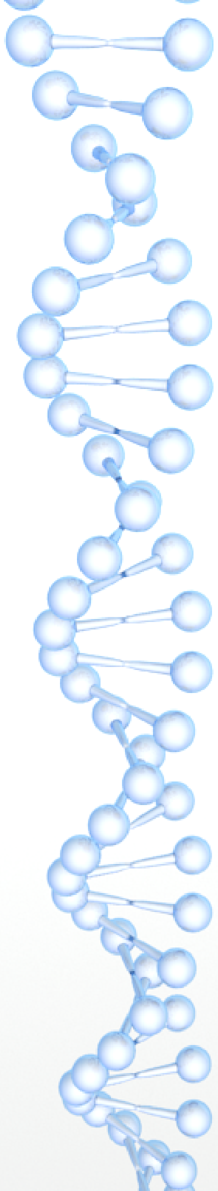


```
# !/bin/bash

CURR=`dirname $0`
if [ -d $CURR/bootstrap/geth ]; then
geth --identity "bootstrap" --networkid 9111 \
     --datadir $CURR/bootstrap  \
     --ipcpath /home/$USER/.ethereum/geth.ipc \
     --nodiscover --port "39909" --rpc --rpcport "42024" \
     --rpcapi "db,eth,web3,net,personal,miner" \
     --unlock 0 --password $CURR/bootstrap/passwd.sec
else
    echo "Initialize first"
fi
~
"runbootstrap.sh" 13L, 394C                                   1,1           All
```

This script starts the `bootstrap` node. Avoids typing all this every time.

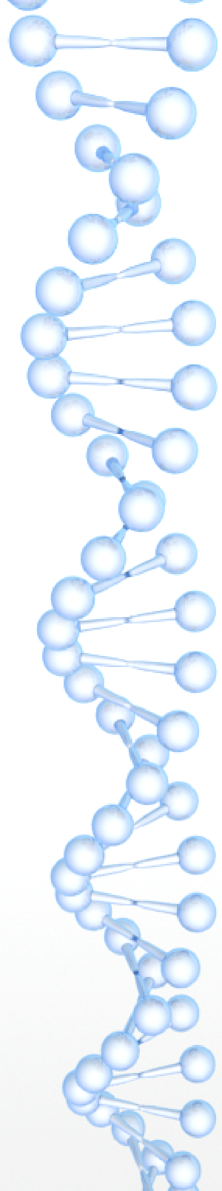
Raspberry Pi as Node

- I was successfully able to use a Raspberry Pi 3 as a light node, which syncs with the miners running on my PC.
- On the Pi:
  - Clone the repo: https://github.com/ahester57/private-chain.git
  - `$ sftp <to PC>; > get genesis.json`
  - `$ geth --datadir ./pinode0 init genesis.json`
  - Create accounts and run to get its `“enode://...”` address
  - Save `“enode://...”` to `<file>`
  - `$ sftp <to PC>; > put <file>`
  - Restart the Pi node, it will now wait until synced with our blockchain.



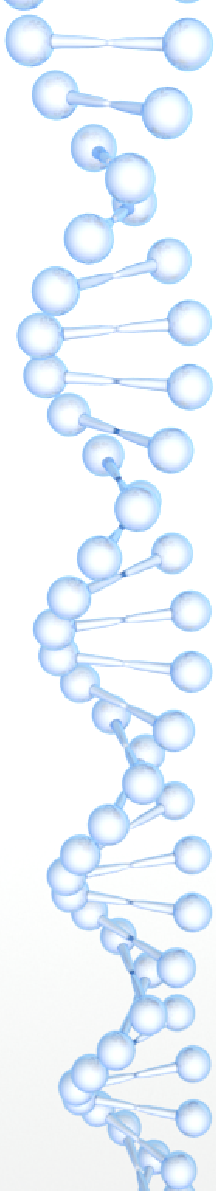
Raspberry Pi as Node

- On PC:
  - `./attach_nodes.sh` *// choose ‘b’*
  - `> admin.addPeer(“enode://..@<pi_ip>:<pi_port>”);`
    - Should return `true`
- Wait a short while, and check it is connected using
  - `> admin.peers` while attached to `bootstrap` node.
  - The `geth` instance running on the Pi should start importing **chain segments** once connected to our blockchain.
  - We can add its `enode` address to `static-nodes.json` to do this automatically on restart.



Next Steps for Blockchain

- Currently, I have a few simple contracts deployed on the bootstrap node. I have a few auctioning and buy/sell contracts as well as contracts which store byte arrays.
- Next, I’d like to create a web page which requires a form of authentication. Once authenticated, this web page will interact with the blockchain by adding a new peer, defined by the user of the page.
- Along with the web page, I want to put together a Smart Contract which unlocks a secure arbitrary data transmission channel between 2 or more nodes.



Resources

- https://github.com/ethereum/wiki/wiki
- https://github.com/ethereum/go-ethereum
- https://github.com/ahester57/private-chain
- https://ethereum.org/token
- https://solidity.readthedocs.io/en/develop/
- http://truffleframework.com/docs/getting_started/project
- https://github.com/ethereum/mist/wiki