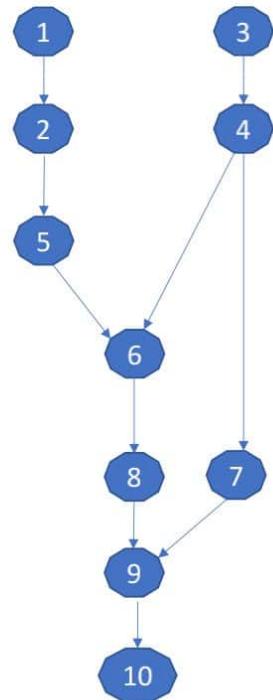


# **COMPLETE HANDS ON NOTE ON BLOCKCHAIN**

# Different tools provide different functionality

	Activities	Tools	Remix	Ganache	MyEtherWallet	Geth
1	Configure the Blockchain		-	-	-	+
2	Deploy the Blockchain	Not Persistent		+	-	+
3	Develop the contract		+	-	-	+
4	Compile the contract		+	-	-	+
5	Create user account		+	+	+	+
6	Deploy the contract		+	-	+	+
7	Create the UI for interacting		+	-	+	+
8	Run the client		+	-	+	+
9	Interact with the contract & have fun		+	-	+	+
10	Monitor the execution		-	+	-	+

<https://remix.ethereum.org/>  
<http://truffleframework.com/ganache/>  
<https://github.com/kvhnuke/etherwallet/releases/tag/v3.21.06>



1

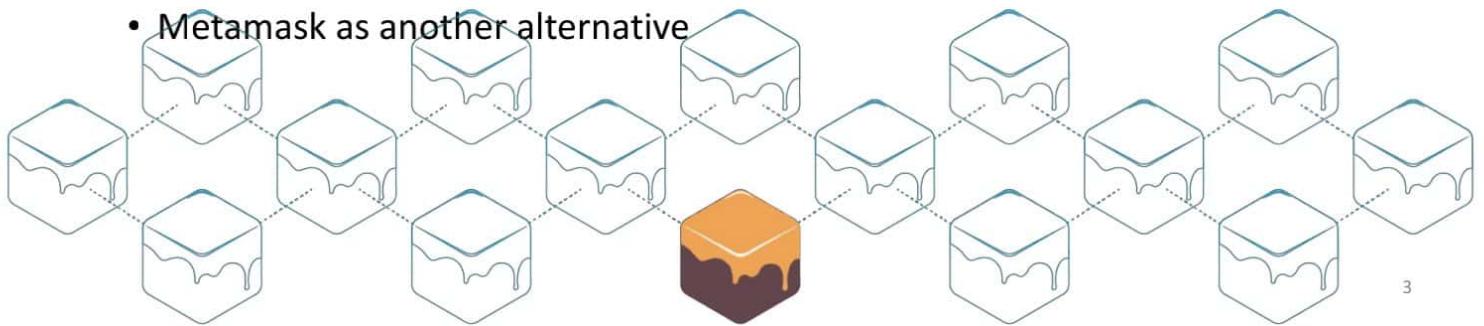
## Use which tool for what purpose? (1/2)

- Use Geth for everything?
  - Powerful but command-line only
- What should I use?
  - As a starting point for developing contracts – mostly Remix
- What cannot Remix do?
  - Configure the blockchain
  - Create real (non-test) user accounts and transfer funds between user accounts
  - Monitor the execution
  - Other advanced operations



## Use which tool for what purpose? (2/2)

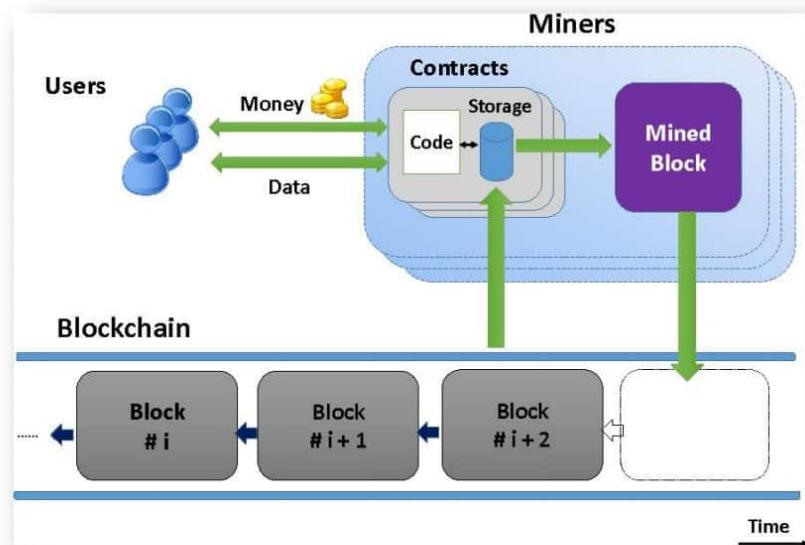
- Why use Ganache?
  - To inspect and monitor the execution
  - To visualize certain elements in a better way
- Why use MyEtherWallet?
  - To create a personal wallet (real user account), transfer funds between user accounts, and interact with contracts
  - Metamask as another alternative



3

# Smart Contracts

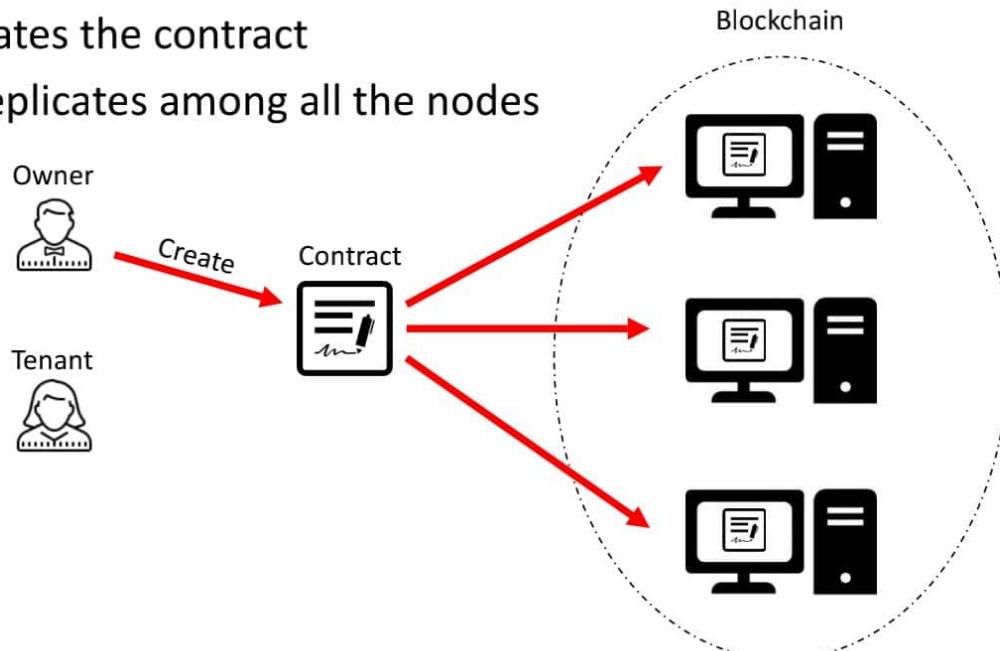
- In the form of code
- Stored on a blockchain
- Executes under given conditions



- K. Delmolino, M. Arnett, A. E. Kosba, A. Miller, and E. Shi, "Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab," *IACR Cryptology ePrint Archive*, vol. 2015, p. 460, 2015.

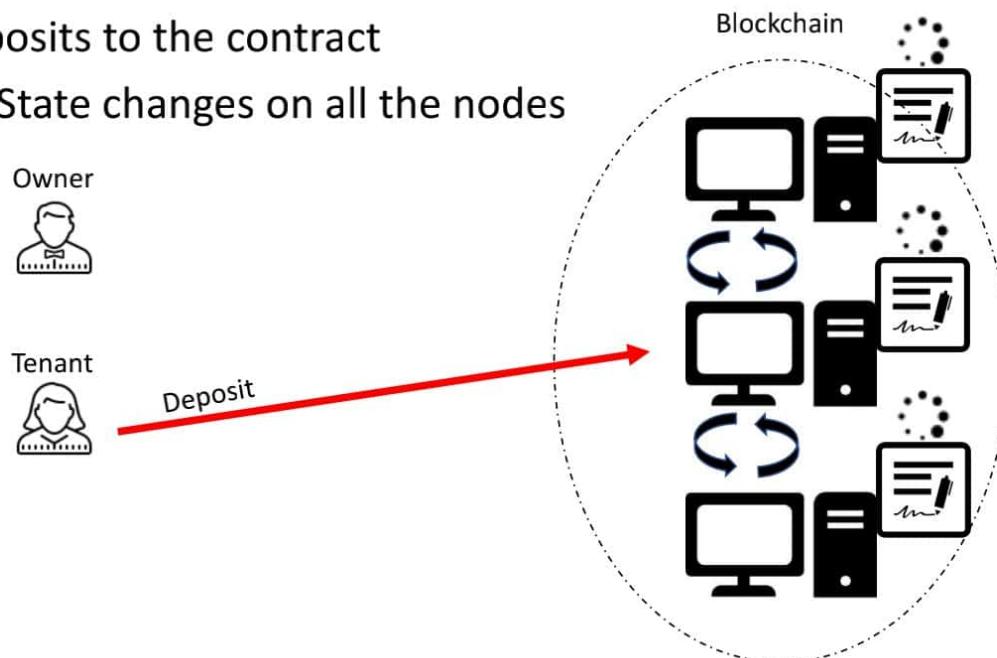
## Smart Contracts Example (1/3)

- Owner creates the contract
- Contract replicates among all the nodes



## Smart Contracts Example (2/3)

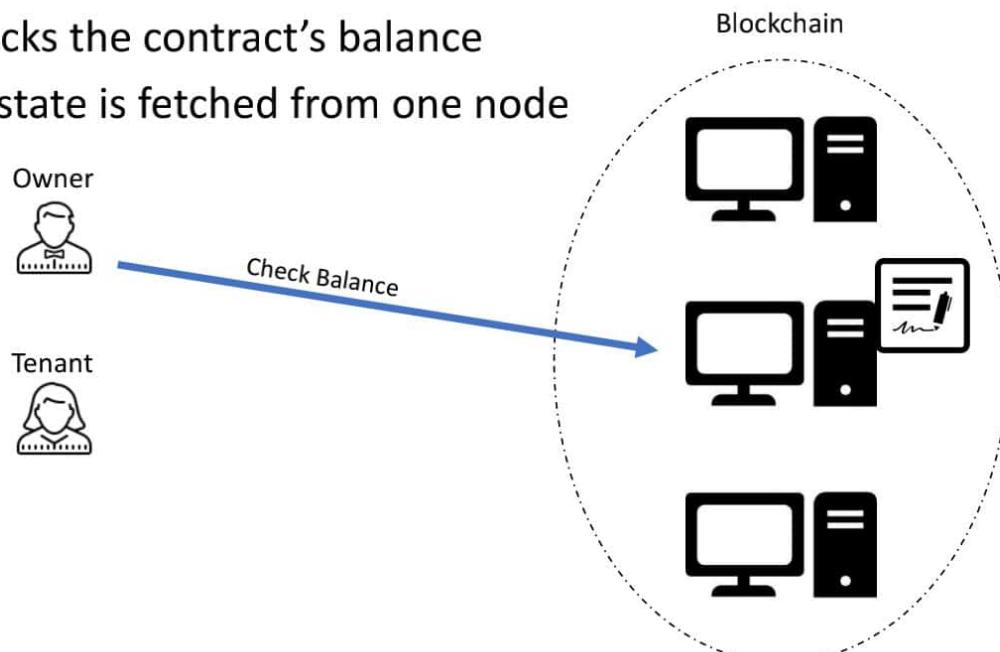
- Tenant deposits to the contract
- Contract's State changes on all the nodes



6

## Smart Contracts Example (3/3)

- Owner checks the contract's balance
- Contract's state is fetched from one node



7

# Smart Contracts

1. Developing a simple contract
2. Compiling the contract
3. Deploying the contract
4. Interacting with the contract
5. Adding more functions to our code to make it more practical

# Open Remix : remix.ethereum.org

- An open source tool for writing, compiling and testing Solidity contracts

The screenshot shows the Remix IDE interface. On the left, there's a sidebar with 'browser' and 'config' sections. The main area displays a Solidity code editor with the file 'browser/firstContract.sol' containing the following code:

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4     uint balance = 313000;
5
6     function getBalance() public view returns(uint){
7         return balance;
8     }
9
10    function deposit(uint newDeposit) public{
11        balance = balance + newDeposit;
12    }
13
14 }
15
16 }
```

To the right of the code editor is a toolbar with 'Compile', 'Run', 'Analysis', 'Testing', 'Debugger', 'Settings', and 'Support'. A dropdown menu shows 'Current version: 0.5.1+commit.c8a2cb62.Emscripten clang'. Below it are checkboxes for 'Auto compile', 'Enable Optimization', and 'Hide warnings', with a button labeled 'Start to compile (Ctrl-S)'.

The bottom right corner shows a preview of the contract with tabs for 'Details', 'ABI', and 'Bytecode'.

At the bottom of the interface, there's a help section with a list of common commands and a search bar for transactions.

# Solidity

- Object-oriented
- Contract-oriented
- High-level language
- Influenced by C++, Python, and JavaScript
- Target Ethereum Virtual Machine (EVM)



## Serpent as an Alternative?

- Low-level language
- Complex compiler

# Start Coding

- Setter and Getter: Set and get the information.

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4
5     uint balance = 313000;           ← Variable
6
7     function getBalance() public view returns(uint){
8         return balance;
9     }                                ← Getter function
10
11    function deposit(uint newDeposit) public{
12        balance = balance + newDeposit;
13    }                                ← Setter function
14
15 }
```

# Compile the Contract

- Compile tab: Start to compile button

```
pragma solidity ^0.5.0;
contract financialContract {
    uint balance = 313000;
    function getBalance() public view returns(uint){
        return balance;
    }
    function deposit(uint newDeposit) public{
        balance = balance + newDeposit;
    }
}
```

Current version:0.5.1+commit.c8a2cb62.Emscripten clang  
Select new compiler version  
 Auto compile  Enable Optimization  Hide warnings  
**Start to compile (Ctrl-S)**

financialContract Swarm

# Set Deployment Parameters (1/2)

- Run tab: Environment = JavaScript VM

The screenshot shows the Truffle UI interface. On the left, there is a code editor window titled "browser/firstContract.sol" containing the following Solidity code:

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4
5     uint balance = 313000;
6
7     function getBalance() public view returns(uint){
8         return balance;
9     }
10
11     function deposit(uint newDeposit) public{
12         balance = balance + newDeposit;
13     }
14
15 }
16
```

On the right, the "Run" tab is selected, showing deployment parameters:

- Environment: JavaScript VM (highlighted with a red box)
- Account: 0xca3...a733c (100 ether)
- Gas limit: 3000000
- Value: 0 wei

Below these fields, there is a "financialContract" input field, a "Deploy" button, and options for "At Address" or "Load contract from Address".

## Set Deployment Parameters (2/2)

- JavaScript VM: All the transactions will be executed in a sandbox blockchain in the browser. Nothing will be persisted and a page reload will restart a new blockchain from scratch, the old one will not be saved.
- Injected Provider: Remix will connect to an injected web3 provider. Mist and Metamask are example of providers that inject web3, thus they can be used with this option.
- Web3 Provider: Remix will connect to a remote node. You will need to provide the URL address to the selected provider: geth, parity or any Ethereum client.
- Gas Limit: The maximum amount of gas that can be set for all the instructions of a contract.
- Value: Input some ether with the next created transaction (wei =  $10^{-18}$  of ether).

# Types of Blockchain Deployment

- Private: e.g., Ganache sets a personal Ethereum blockchain for running tests, executing commands, and inspecting the state while controlling how the chain operates.
- Public Test (Testnet): Like Ropsten, Kovan and Rinkeby which are existing public blockchains used for testing and which do not use real funds. Use faucet for receiving initial virtual funds.
- Public Real (Mainnet): Like Bitcoin and Ethereum which are used for real and which available for everybody to join.

# Deploy the Contract on the Private Blockchain of Remix

- Run tab: Deploy button

The screenshot shows the Remix IDE interface with the following details:

- Code Area:** The code editor displays a Solidity contract named `financialContract.sol`. The code defines a contract with two functions: `getBalance()` and `deposit(uint newDeposit)`.
- Run Tab:** The top navigation bar has a "Run" tab selected. The configuration panel on the right shows:
  - Environment: JavaScript VM
  - Account: 0xca3...a733c (99.999999999998644) (highlighted with a red box)
  - Gas limit: 3000000
  - Value: 0 wei
- Deployment Panel:** A central panel contains a text input field "financialContract" and a "Deploy" button (highlighted with a red box). Below it are options "At Address" and "Load contract from Address".
- Transactions Recorded:** A section titled "Transactions recorded: 1" shows a single transaction for the deployed contract.
- Deployed Contracts:** A section titled "Deployed Contracts" lists the deployed contract: "financialContract at 0x692...77b3a (memory)". Below it, two function calls are listed: "deposit" and "getBalance", both highlighted with a red circle.
- Bottom Bar:** The bottom bar includes a note about common commands and a search bar.

# Interact with the Contract

- Setter = Red Button: Creates transaction
- Getter= Blue Button: Just gives information

The image consists of two screenshots of a blockchain interface, likely from a Ethereum wallet or browser extension. Both screenshots show a deployed contract named 'financialContract' at address 0x692...77b3a (memory).

**Screenshot 1:** This screenshot shows the initial state of the contract. The 'deposit' field is empty, and the 'getBalance' field shows a value of 0: uint256: 313000. A blue callout box with the number 1 contains the text: "Press getBalance to see the initial amount".

deposit	uint256 newDeposit
getBalance	0: uint256: 313000

**Screenshot 2:** This screenshot shows the state after a deposit has been made. The 'deposit' field now contains the value 12. A red callout box with the number 2 contains the text: "Input a value and press deposit button to create and confirm the transaction". Another blue callout box with the number 3 contains the text: "Press getBalance again to see the result".

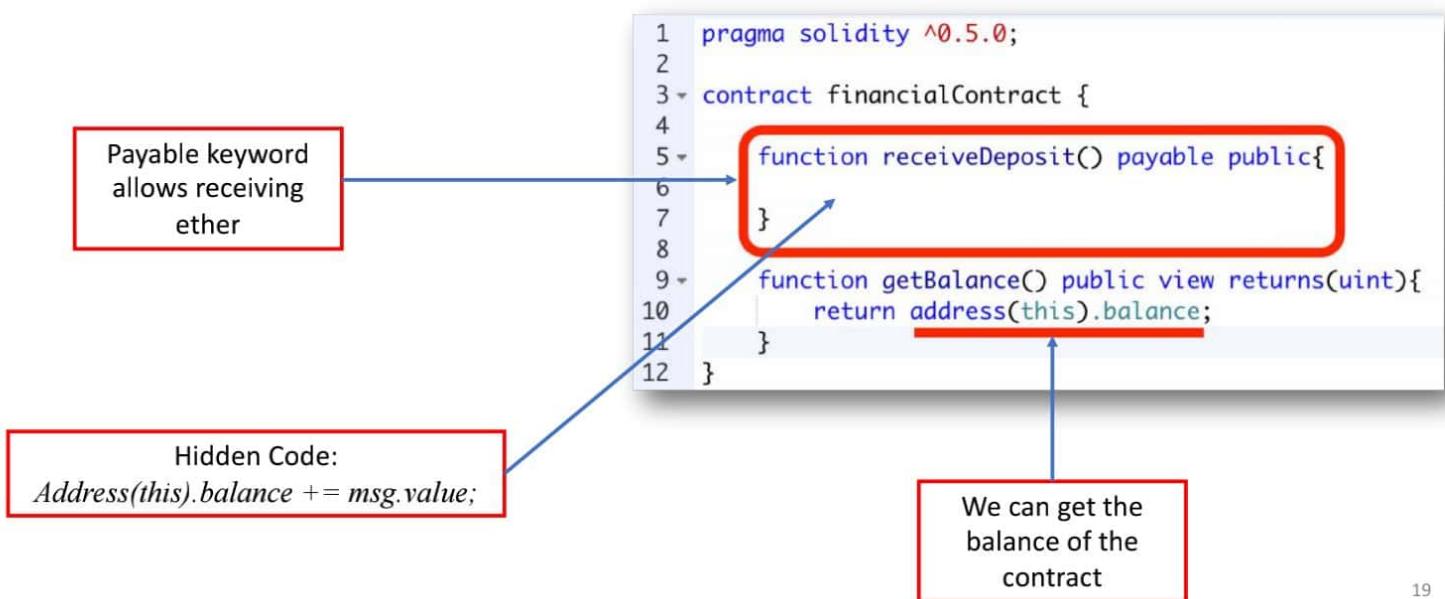
deposit	12
getBalance	0: uint256: 313012

## Additional features

- Transferring funds from an account to the contract
- Saving the address of the contract creator
- Limiting the users' access to functions
- Withdrawing funds from the contract to an account

## Receive ether (1/2)

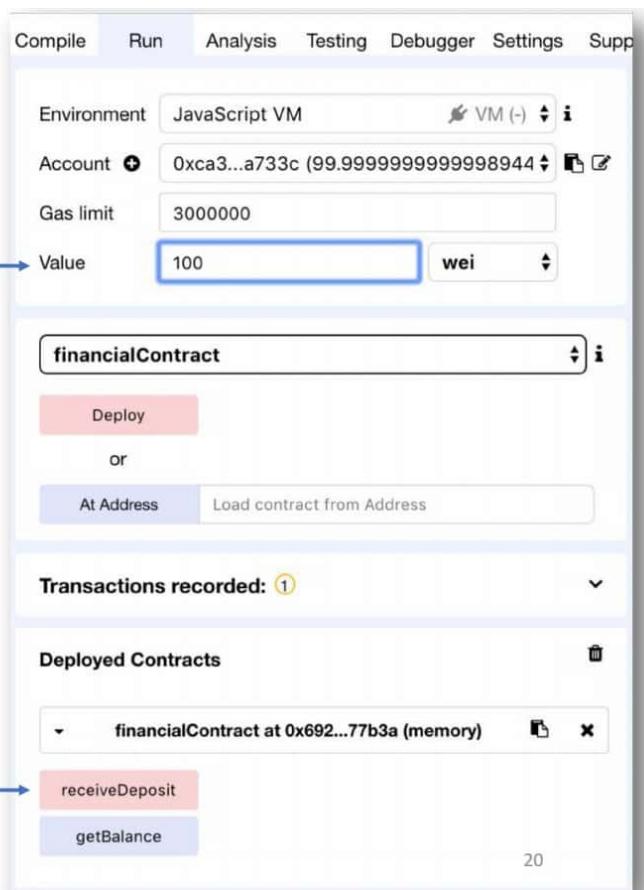
- Transfer money to the contract



## Receive ether (2/2)

1

Input the value as wei  
( $10^{-18}$  of ether)



2

Click the receiveDeposit button to transfer the money to the contract

# Constructor

- Will be called at the creation of the instance of the contract

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4
5     address owner;
6
7     constructor() public{
8         owner = msg.sender;
9     }
10
11    function receiveDeposit() payable public{
12    }
13
14    function getBalance() public view returns(uint){
15        return address(this).balance;
16    }
17
18 }
```

We want to save  
the address of the  
contract creator

# Withdraw funds

- Modifier: Conditions you want to test in other functions
- First the modifier will execute, then the invoked function

Only the contract's creator is permitted to withdraw

Transfer some money from the contract's balance to the owner

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4
5     address owner;
6
7 constructor() public{
8     owner = msg.sender;
9 }
10
11 modifier ifOwner(){
12     if(owner != msg.sender){
13         revert();
14     }else{
15         -
16     }
17 }
18
19
20 function receiveDeposit() payable public{
21 }
22
23
24 function getBalance() public view returns(uint){
25     return address(this).balance;
26 }
27
28 function withdraw(uint funds) public ifOwner{
29     msg.sender.transfer(funds);
30 }
31 }
```

## Now deploying a smart contract on an external blockchain

	Activities \ Tools	Remix	Ganache	MyEtherWallet	Geth
Activities					
1	Configuring the Blockchain	-	-	-	+
2	Deploying the Blockchain	Not Persistent	+	-	+
3	Developing the contract	+	-	-	+
4	Compiling the contract	+	-	-	+
5	Creating user account	+	+	+	+
6	Deploying the contract	+	-	+	+
7	Creating the UI for interacting	+	-	+	+
8	Run the client	+	-	+	+
9	Interact with the contract & have fun	+	-	+	+
10	Monitoring the execution	-	+	-	+

23

# Run Ganache

The screenshot shows the Ganache interface with the following details:

- Accounts:** Three accounts are listed, each with a balance of **100.00 ETH**.
  - Address:** 0x231eAeEF9EA93F5370a1F633F32E45AF570980E8 (Index 0)
  - Address:** 0x970fc818790E900598C57E48b89B6D3D8896D416 (Index 1)
  - Address:** 0xb59BD5568d0be42C13fB521f845243F1CDaF2eF1 (Index 2)
- Mnemonic:** slim rain lawn kiwi elegant behind vibrant dentist puppy reduce kidney there
- HD Path:** m/44'/60'/0'/0/account\_index
- RPC Server:** HTTP://127.0.0.1:7545
- Miner Status:** AUTOMINING
- Logs:** A search bar for logs.

24

# MyEtherWallet

- add your custom network that you want to test your contracts on

The screenshot shows the MyEtherWallet website interface. On the left, there's a form titled "Create New Wallet" with a password input field and a "Create New Wallet" button. A note below the form states: "This password encrypts your private key. This does not act as a seed to generate your keys. You will need this password + your private key to unlock your wallet." On the right, a "Network" dropdown menu is open, listing various Ethereum networks. A red arrow points from the text "add your custom network that you want to test your contracts on" to the "Add Custom Network / Node" option at the bottom of the list.

New Wallet Send Ether & Tokens Swap Send Offline Contracts ENS DomainSale Check TX Status View Wallet Info Help

3.21.05 English Gas Price: 41 Gwei The network is really full right now.

Network ETH (myetherapi.com)

- ETH (myetherapi.com)
- Ropsten (myetherapi.com)
- ETH (infura.io)
- Kovan (infura.io)
- ETC (Ethereum Commonwealth)
- ETC (ropsten.io)
- Ropsten (infura.io)
- Kovan (infura.io)
- Rinkeby (infura.io)
- EXP (expansetech.io)
- UBQ (ubiqscout.io)
- POA (core.poa.network)
- TOMO (core.tomo.network)
- ELLA (ellaism.org)
- ETSC (etsc.network)

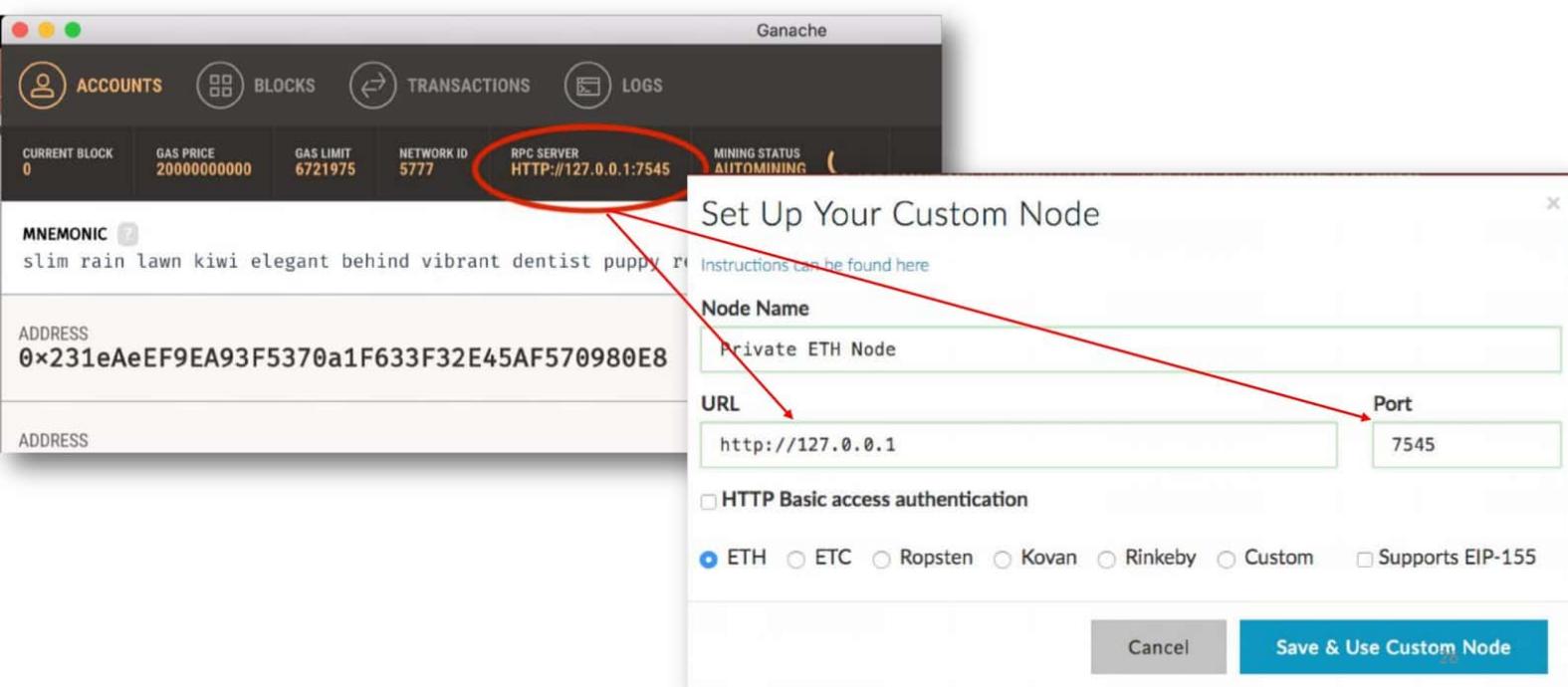
Add Custom Network / Node

This password encrypts your private key. This does not act as a seed to generate your keys. You will need this password + your private key to unlock your wallet.

How to Create a Wallet • Getting Started

MyEtherWallet.com does not hold your keys for you. We cannot access accounts, recover keys, reset passwords, nor reverse transactions. Protect your keys & always check that you are on correct URL. You are responsible for your security.

Import your RPC server address and the port number from Ganache to MyEtherWallet



# MyEtherWallet

- Contracts tab: Deploy Contract

The screenshot shows the MyEtherWallet web application interface. At the top, there is a dark blue header bar with the "MyEtherWallet" logo, version "3.21.05", language "English", gas price "Gas Price: 41 Gwei", and network "Network My Ether Node:eth (Custom)". A message at the top right says, "The network is really full right now. Check Eth Gas Station for gas price to use." Below the header, there is a navigation bar with links: "New Wallet", "Send Ether & Tokens", "Swap", "Send Offline", "Contracts" (which is underlined and circled in red), "ENS", "DomainSale", "Check TX Status", "View Wallet Info", and "Help". A large red arrow points from the "Contracts" link down to the "Deploy Contract" button. Below the navigation bar, there are two main input fields: "Byte Code" (empty) and "Gas Limit" (set to 300000). The bottom right corner of the screenshot has the number "27".

# Remix

- Type your contract and compile it

```
pragma solidity ^0.5.0;
contract financialContract {
    uint amount = 13;
    function getValue() public view returns(uint){
        return amount;
    }
    function setValue(uint newAmount) public{
        amount = newAmount;
    }
}
```

# Remix

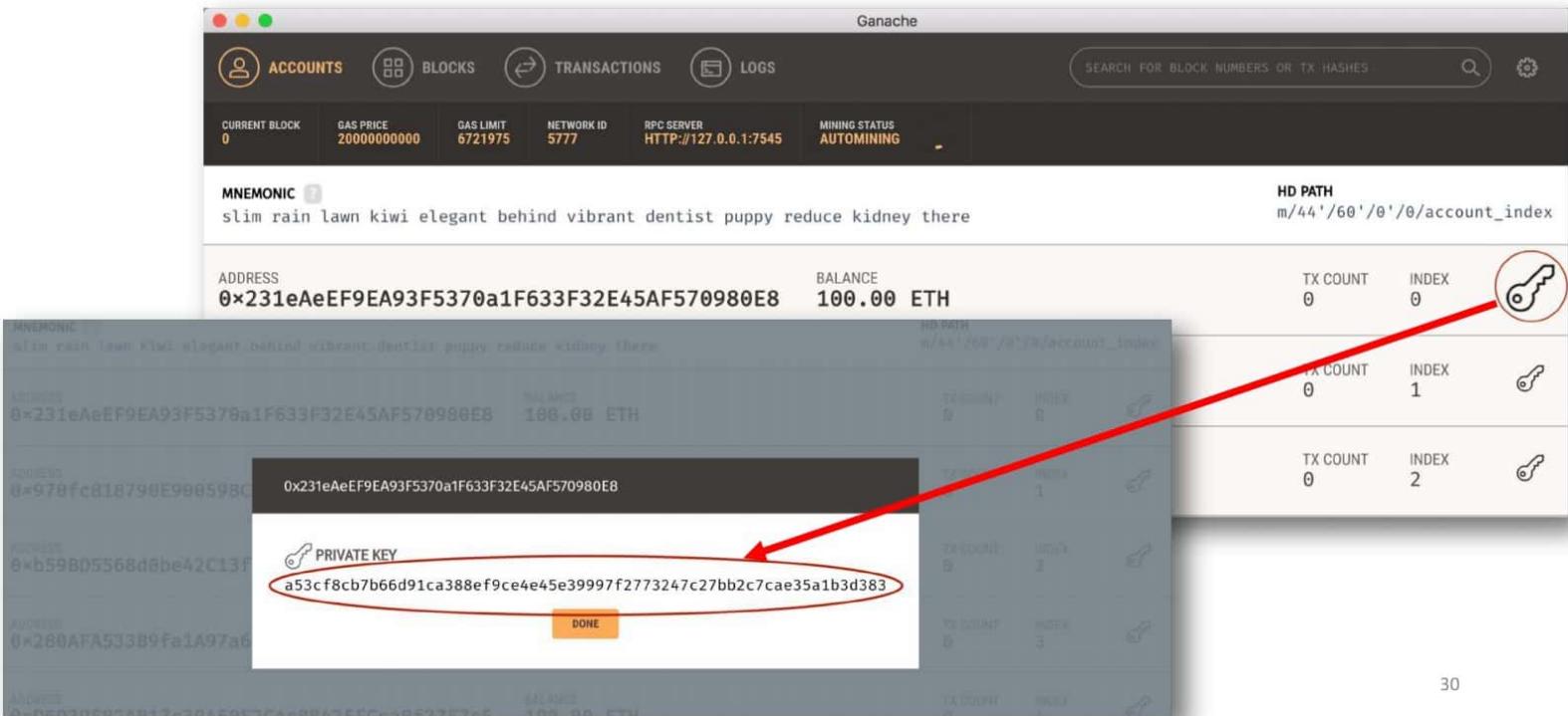
Click on Details Button: access ByteCode to import it to MyEtherWallet



29

# Ganache

Access your private key for signing your contract in MyEtherWallet.



# MyEtherWallet

1. Paste the contract's ByteCode from Remix
  2. Gas Limit will automatically be calculated
  3. Paste your private key from Ganache
  4. Click Unlock
  5. Now you have access to your wallet

# MyEtherWallet

Click on *Sign Transaction* button to deploy your contract

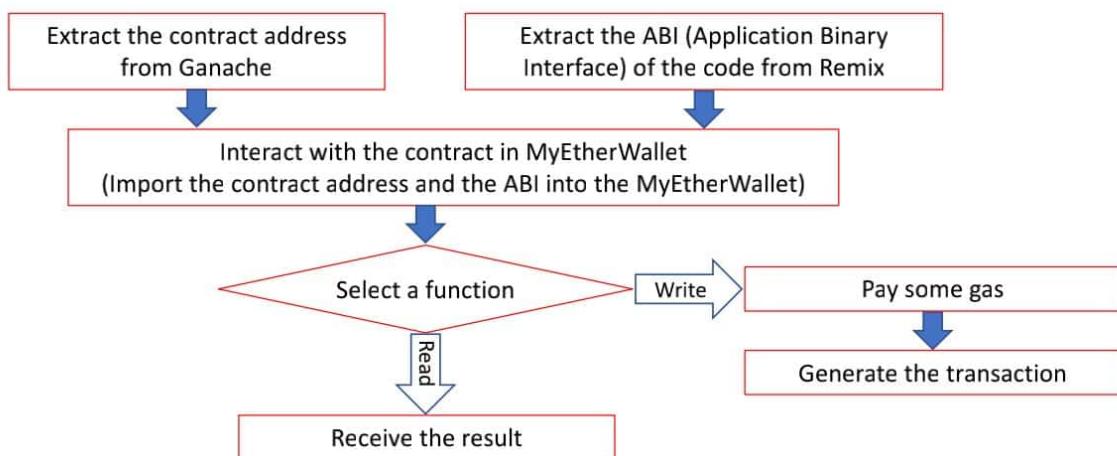
# Ganache

You can see now you have one transaction for your address and your balance has been changed because of the amount of gas you paid for creating the contract.

The screenshot shows the Ganache application interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, and LOGS. Below the tabs, there are several status indicators: CURRENT BLOCK (1), GAS PRICE (20000000000), GAS LIMIT (6721975), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:7545), and MINING STATUS (AUTOMINING). A search bar and a gear icon are also present. The main area displays a table of accounts:

MNEMONIC	ADDRESS	BALANCE	TX COUNT	INDEX	Key
slim rain lawn kiwi elegant behind vibrant dentist puppy reduce kidney there	0x231eAeEF9EA93F5370a1F633F32E45AF570980E8	99.99 ETH	1	0	🔑
	0x970fc818790E900598C57E48b89B6D3D8896D416	100.00 ETH	0	1	🔑
	0xb59BD5568d0be42C13fB521f845243F1CDaF2eF1	100.00 ETH	0	2	🔑
	0x280AFA533B9fa1A97a6D2E4640412FD86FC5dd36	100.00 ETH	0	3	🔑
	0xD6D39E82AB17c30460F2CAC88425ECcaBf2757c5	100.00 ETH	0	4	🔑

# Interacting with the smart contract



# Ganache

Transactions tab: Copy the created contract address

The screenshot shows the Ganache application window. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS (which is highlighted with a red oval), and LOGS. Below the tabs, there are several status indicators: CURRENT BLOCK (1), GAS PRICE (20000000000), GAS LIMIT (6721975), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:7545), and MINING STATUS (AUTOMINING). A search bar at the top right says "SEARCH FOR BLOCK NUMBERS OR TX HASHES". In the main area, a transaction is listed with the following details: TX HASH (0x1e40cc28802d152e810bd9f40bea83d83b1655fc9bace6e801ec6db5fcd84b1a), FROM ADDRESS (0x231eAeEF9EA93F5370a1F633F32E45AF570980E8), CREATED CONTRACT ADDRESS (0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d), GAS USED (124604), and VALUE (0). A red arrow points from the text "Copy the created contract address" to the "CREATED CONTRACT ADDRESS" field.

35

# Remix

Copy the ABI

(ABI is the interface that tells MyEtherWallet how to interact with the contract)



```
pragma solidity ^0.5.0;
contract financialContract {
    uint amount = 13;
    function getValue() public view returns(uint){
        return amount;
    }
    function setValue(uint newAmount) public{
        amount = newAmount;
    }
}
```

36

# MyEtherWallet

Contracts tab:

Interact with Contract = Paste the contract address from Ganache and the ABI from Remix

New Wallet Send Ether & Tokens Swap Send Offline **Contracts** ENS DomainSale Check TX Status View Wallet Info Help

Interact with Contract or Deploy Contract

Contract Address: 0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d

Select Existing Contract: Select a contract...

ABI / JSON Interface:

```
"outputs": [],
"payable": false,
"stateMutability": "nonpayable",
"type": "function"}]
```

Access

# MyEtherWallet

You now can interact with the contract by selecting a function and invoking it

The screenshot shows the MyEtherWallet interface with the 'Contracts' tab selected. The main title is 'Interact with Contract or Deploy Contract'. On the left, there's a 'Contract Address' input field containing the address 0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d. To its right is a 'Select Existing Contract' button with the placeholder 'Select a contract...'. Below these are sections for 'ABI / JSON Interface' and 'Access'. The 'Access' section has a blue button labeled 'Access'. At the bottom, there's a 'Read / Write Contract' section with a red oval highlighting the 'Select a function' dropdown menu, which contains 'getValue' and 'setValue' options.

38

# MyEtherWallet

If you select the getValue function you will receive the value without paying any gas  
(There is no operation cost for getting information)

The screenshot shows the MyEtherWallet interface. In the top left, it says "Read / Write Contract" and displays the contract address "0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d". Below this, there is a dropdown menu with "getValue" selected. To the right of the dropdown, there is a small icon of a person with a tail and the text "uint256". At the bottom of the interface, there is a button labeled "13".

# MyEtherWallet

If you choose a function that updates the state of the contract, you will need to pay gas for it in a transaction.

The screenshot shows the MyEtherWallet interface for interacting with a smart contract. On the left, under 'Read / Write Contract', a contract address is listed: 0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d. Below this, a dropdown menu 'newValue uint256' is set to '6'. A red circle highlights the input field for 'newValue'. An arrow points from this field to a large blue 'WRITE' button at the bottom of the section. To the right, a 'Warning!' dialog box is displayed. It contains fields for 'Amount to Send' (set to 0), 'Gas Limit' (set to 41633), and a 'Generate Transaction' button. A red circle highlights the 'Generate Transaction' button. Below the dialog, a 'Raw Transaction' section shows JSON data, and a 'Signed Transaction' section shows a long hex string. At the bottom of the dialog are two buttons: 'No, get me out of here!' and 'Yes, I am sure! Make transaction.' A red circle highlights the 'Yes, I am sure! Make transaction.' button. The entire process is intended to generate and sign a transaction to update the contract's state.

## Create Custom Ethereum Blockchain

- Instead of using Ganache with its default properties for private blockchain you can run your own blockchain
- Install Geth: One of the implementations of Ethereum written in Go
- Create the genesis block
- Create storage of the blockchain
- Deploy blockchain nodes
- Connect MyEtherWallet to your blockchain to interact with it

# Geth help

```
mohammht ~ -bash — 97x40
ds-install:~ mohammht$ geth help
NAME:
  geth - the go-ethereum command line interface

  Copyright 2013-2018 The go-ethereum Authors

USAGE:
  geth [options] command [command options] [arguments...]

VERSION:
  1.8.9-stable

COMMANDS:
  account      Manage accounts
  attach        Start an interactive JavaScript environment (connect to node)
  bug          opens a window to report a bug on the geth repo
  console      Start an interactive JavaScript environment
  copydb       Create a local chain from a target chaindata folder
  dump         Dump a specific block from storage
  dumpconfig   Show configuration values
  export        Export blockchain into file
  export-preimages Export the preimage database into an RLP stream
  import        Import a blockchain file
  import-preimages Import the preimage database from an RLP stream
  init          Bootstrap and initialize a new genesis block
  js            Execute the specified JavaScript files
  license       Display license information
  makecache    Generate ethash verification cache (for testing)
  makedag      Generate ethash mining DAG (for testing)
  monitor      Monitor and visualize node metrics
  removedb    Remove blockchain and state databases
  version      Print version numbers
  wallet       Manage Ethereum presale wallets
  help, h      Shows a list of commands or help for one command

ETHEREUM OPTIONS:
  --config value           TOML configuration file
  --datadir "/Users/mohammht/Library/Ethereum" Data directory for the databases and keystore
  --keystore               Directory for the keystore (default = inside the
  datadir)
```

## Genesis block

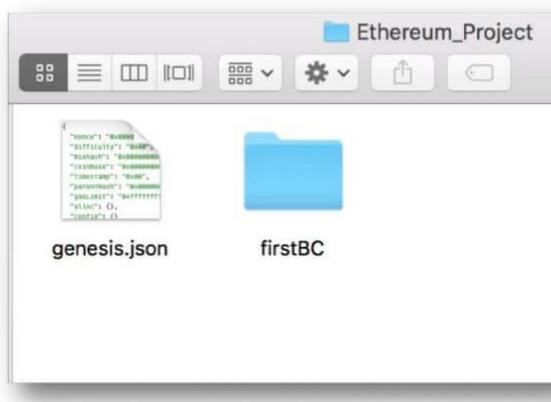
- The first block in the chain and a json file that stores the configuration of the chain

- Create and store the file as genesis.json

## Create the storage of the blockchain

- Go to the directory of the genesis.json file
- Specify directory of your blockchain
- Create the storage from the genesis block

```
[ds-install:Documents mohammht$ cd Ethereum_Project/  
ds-install:Ethereum_Project mohammht$ geth --datadir firstBC init genesis.json
```

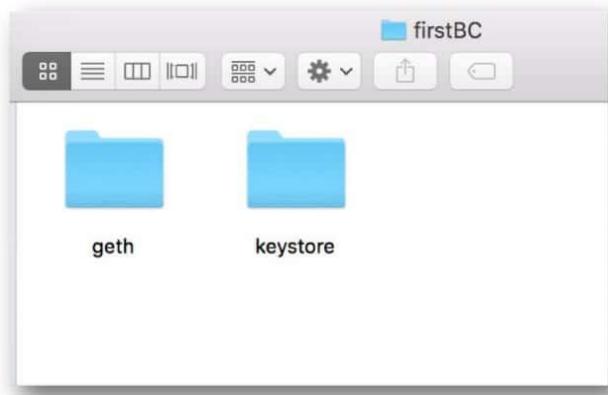


Folder name of your blockchain

44

## Inside the Blockchain Folder

- geth folder: Store your database
- keystore: Store your Ethereum accounts



## Start the Ethereum peer node

- Start the blockchain

```
geth --datadir fistBC --networkid 100 console
```

- Networkid provides privacy for your network.
- Other peers joining your network must use the same networkid.

# Blockchain started

- Type  
`admin.nodeInfo`  
to get the  
information  
about your  
current node

```
|> admin.nodeInfo
{
  enode: "enode://4561ccdd7fdf3f0bdbc903b7bef7d472e136fe2b63012151a1dd3c27e52f49bda2ef66631e67022b7ca7b9fba06bb0eda8b47210b198f3eff7e67414d695ed6@[::]:30303",
  id: "4561ccdd7fdf3f0bdbc903b7bef7d472e136fe2b63012151a1dd3c27e52f49bda2ef66631e67022b7ca7b9fba06bb0eda8b47210b198f3eff7e67414d695ed6",
  ip: "::",
  listenAddr: "[::]:30303",
  name: "Geth/v1.8.9-stable/darwin-amd64/go1.10.2",
  ports: {
    discovery: 30303,
    listener: 30303
  },
  protocols: {
    eth: {
      config: {
        byzantiumBlock: 4370000,
        chainId: 1,
        daoForkBlock: 1920000,
        daoForkSupport: true,
        eip150Block: 2463000,
        eip150Hash: "0x2086799aeebeae135c246c65021c82b4e15a2c451340993aacfd2751886514f0",
        eip155Block: 2675000,
        eip158Block: 2675000,
        ethash: {},
        homesteadBlock: 1150000
      },
      difficulty: 17179869184,
      genesis: "0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3",
      head: "0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3",
      network: 100
    }
  }
}
|>
```

47

## Create an account

- Type *personal.newAccount* to create as many accounts as you need

```
> personal.newAccount('Type your password here')
"0xa78eb41a10f096d4d8c4c9ca5196427aaa3fdb33"
> █
```

- See the created account(s)

```
> eth.accounts
["0xa78eb41a10f096d4d8c4c9ca5196427aaa3fdb33", "0x354d952e40fc35a47562d479c86e41f6623e5f8c"]
>
```

# Mining

- Type *miner.start()* to start mining

```
|> miner.start()
INFO [05-30|12:07:54] Updated mining threads          threads=0
INFO [05-30|12:07:54] Transaction pool price threshold updated price=180000000000
null
> INFO [05-30|12:07:54] Starting mining operation
INFO [05-30|12:07:54] Commit new mining work          number=1 txs=0 uncles=0 elapsed=22
8.827µs
INFO [05-30|12:07:57] Generating DAG in progress    epoch=1 percentage=0 elapsed=2.013
s
INFO [05-30|12:07:59] Generating DAG in progress    epoch=1 percentage=1 elapsed=4.151
s
INFO [05-30|12:08:03] Generating DAG in progress    epoch=1 percentage=2 elapsed=7.322
s
INFO [05-30|12:08:06] Generating DAG in progress    epoch=1 percentage=3 elapsed=10.70
5s
INFO [05-30|12:08:09] Generating DAG in progress    epoch=1 percentage=4 elapsed=14.04
3s
INFO [05-30|12:08:13] Generating DAG in progress    epoch=1 percentage=5 elapsed=17.56
5s
INFO [05-30|12:08:16] Generating DAG in progress    epoch=1 percentage=6 elapsed=20.99
9s
INFO [05-30|12:08:20] Generating DAG in progress    epoch=1 percentage=7 elapsed=24.40
9s
```

- Type *miner.stop()* to stop mining

Thank you

ADARSH CHETAN