

A hands-on tutorial: Working with Smart Contracts in Ethereum

Mohammad H. Tabatabaei

Roman Vitenberg

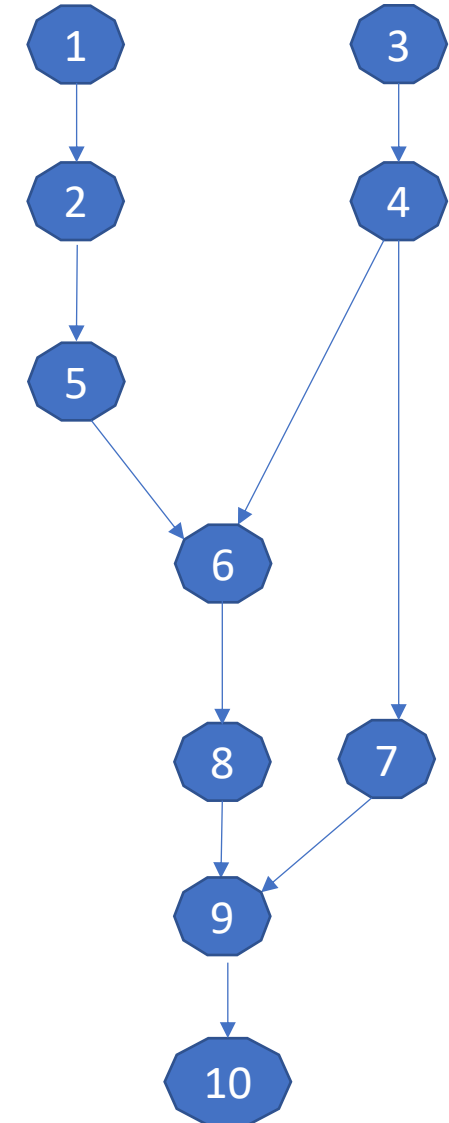
Kaiwen Zhang

Mohammad Sadoghi

Hans-Arno Jacobsen

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	-	+	-	+



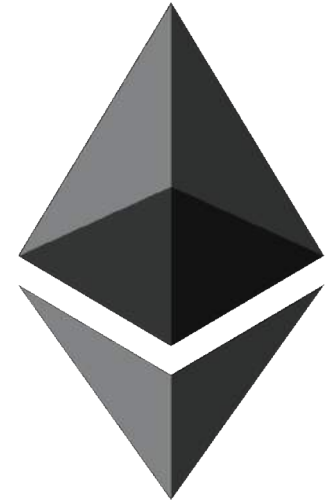
<https://remix.ethereum.org/>

<http://truffleframework.com/ganache/>

<https://github.com/kvhnuke/etherwallet/releases/tag/v3.21.06>

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



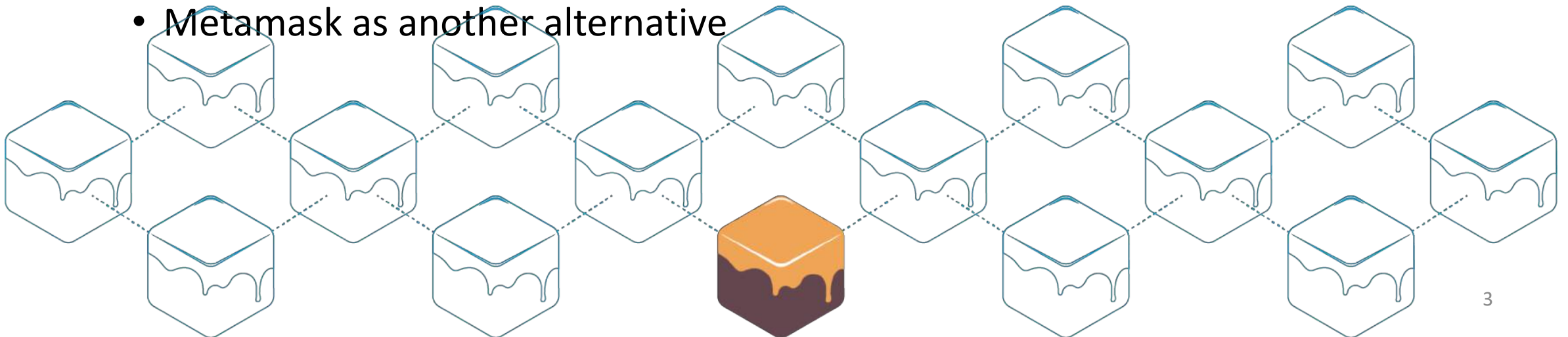
Use which tool for what purpose? (2/2)

- Why use Ganache?

- To inspect and monitor the execution
- To visualize certain elements in a better way

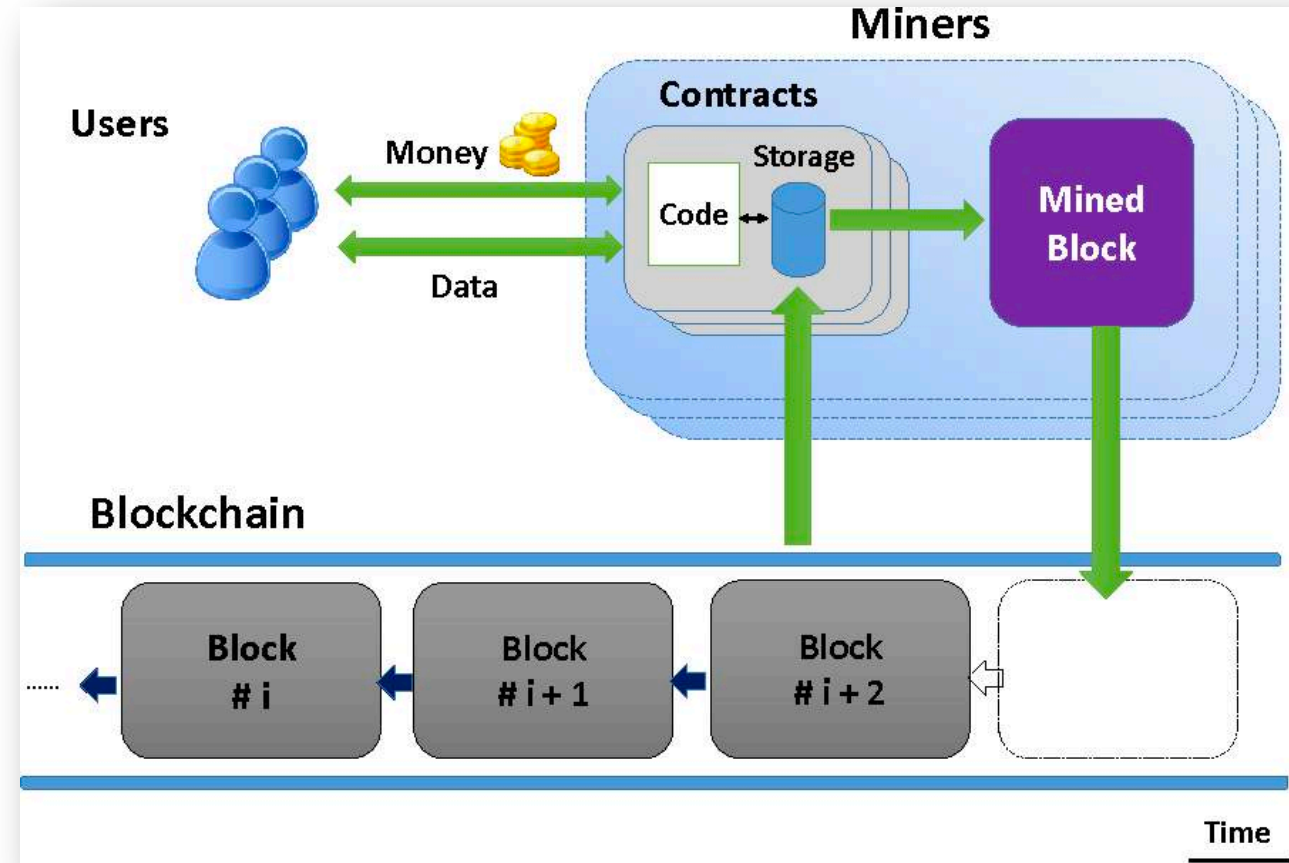
- Why use MyEtherWallet?

- To create a personal wallet (real user account), transfer funds between user accounts, and interact with contracts
- Metamask as another alternative



Smart Contracts

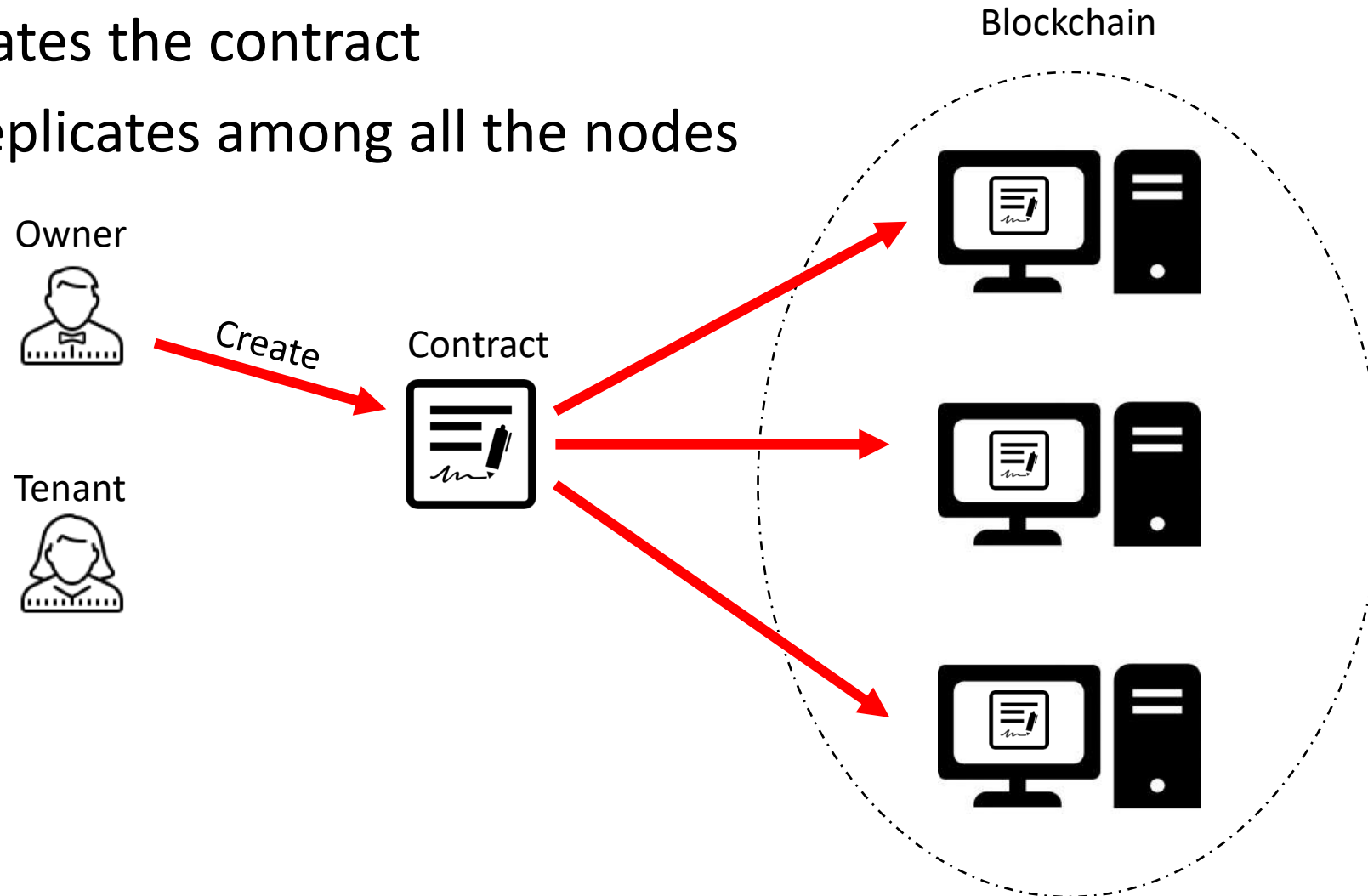
- In the form of code
- Stored on a blockchain
- Executes under given conditions



- 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

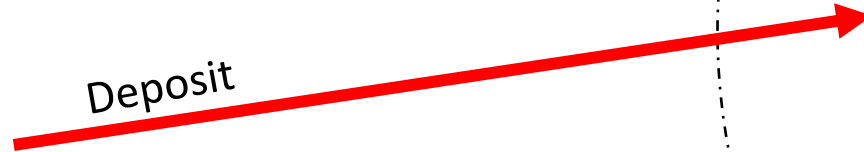
Owner



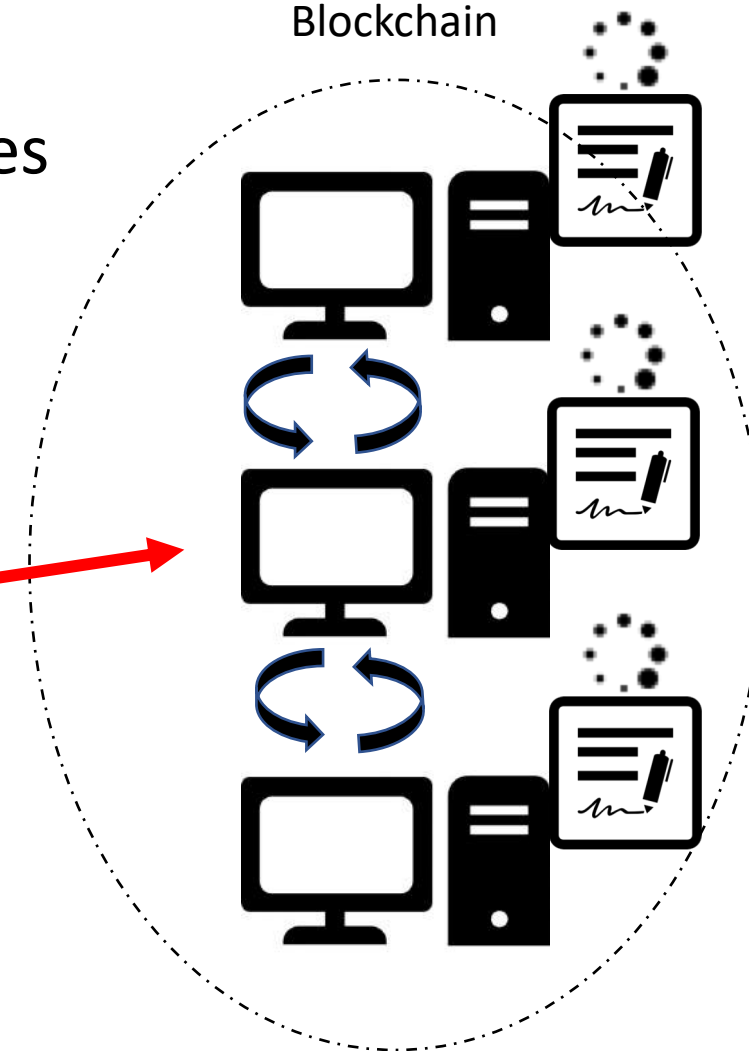
Tenant



Deposit

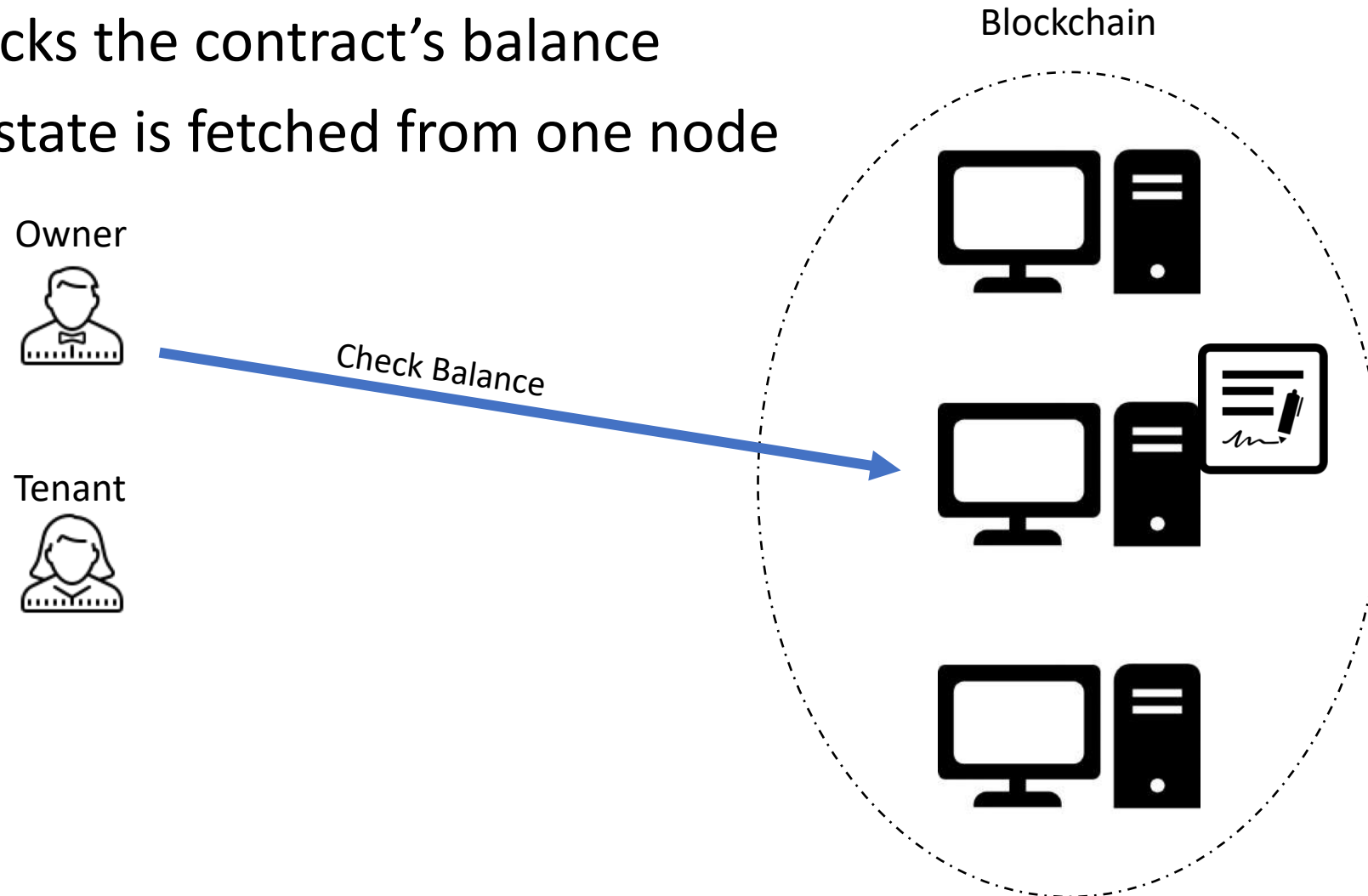


Blockchain



Smart Contracts Example (3/3)

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

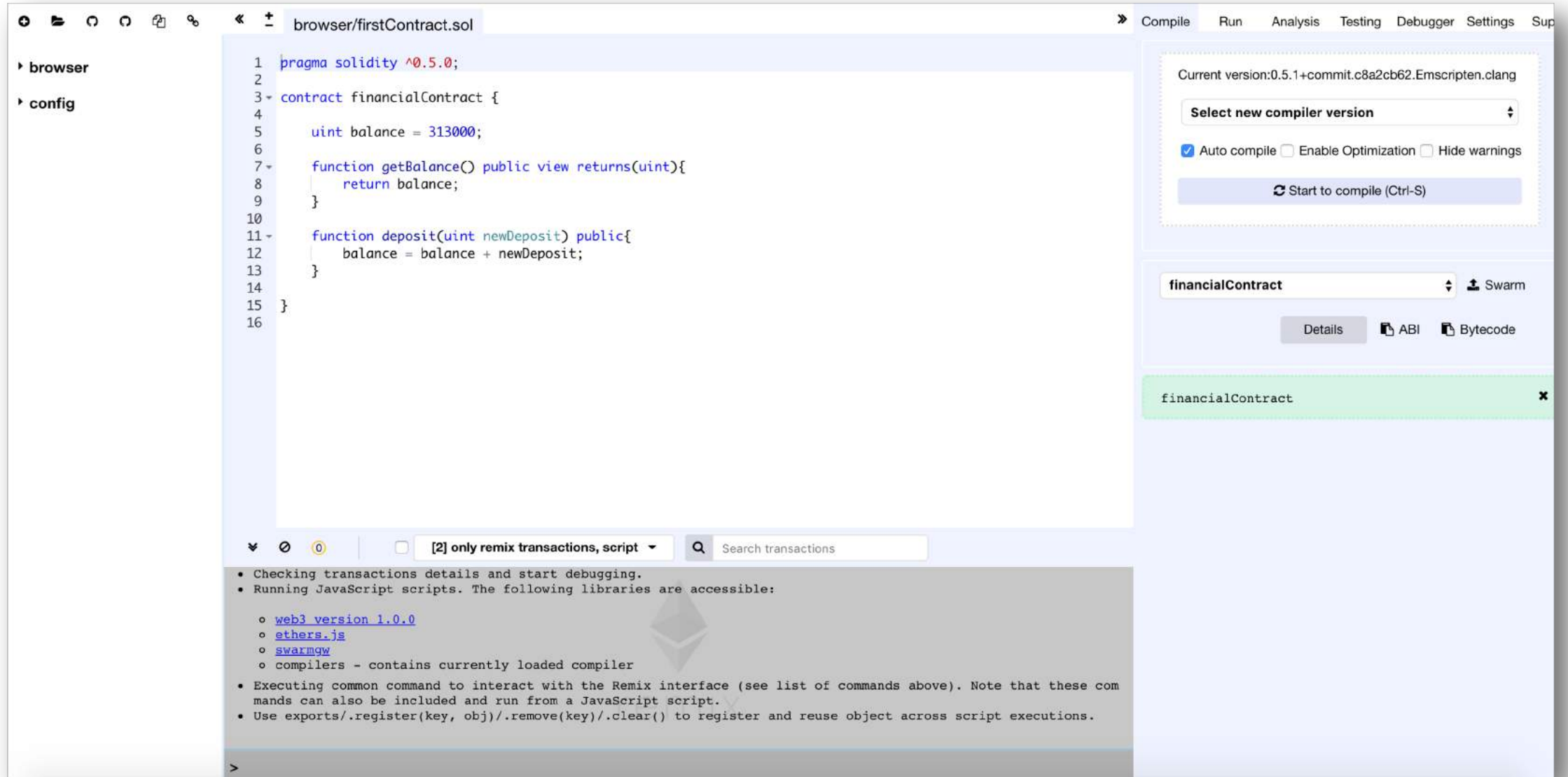


Smart Contracts

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 displays the Remix IDE interface. The main editor shows a Solidity contract named `financialContract` with the following 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
```

The right sidebar contains the compiler settings and contract details. The current version is `0.5.1+commit.c8a2cb62.Emscripten.clang`. The `Auto compile` checkbox is checked. The contract name is `financialContract`, and the `ABI` and `Bytecode` buttons are visible.

The bottom panel shows the console output, which includes the following information:

- Checking transactions details and start debugging.
- Running JavaScript scripts. The following libraries are accessible:
 - [web3 version 1.0.0](#)
 - [ethers.js](#)
 - [swarmgw](#)
 - `compilers` - contains currently loaded compiler
- Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run from a JavaScript script.
- Use `exports.register(key, obj).remove(key).clear()` to register and reuse object across script executions.

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;
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 }
```

The diagram illustrates the components of the Solidity code. A yellow oval highlights the variable declaration `uint balance = 313000;` on line 5, with a yellow box labeled "Variable" pointing to it. A blue rounded rectangle highlights the `getBalance()` function on lines 7-9, with a blue box labeled "Getter function" pointing to it. A red rounded rectangle highlights the `deposit()` function on lines 11-13, with a red box labeled "Setter function" pointing to it.

Compile the Contract

- Compile tab: Start to compile button



The screenshot shows a Solidity IDE interface. On the left, a code editor displays the following Solidity 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
11     function deposit(uint newDeposit) public{
12         balance = balance + newDeposit;
13     }
14
15 }
```

On the right, the 'Compile' tab is active. It shows the current compiler version as '0.5.1+commit.c8a2cb62.Emscripten.clang'. Below this, there is a dropdown menu labeled 'Select new compiler version'. There are three checkboxes: 'Auto compile' (checked), 'Enable Optimization' (unchecked), and 'Hide warnings' (unchecked). A red box highlights the 'Start to compile (Ctrl-S)' button. At the bottom, there is a dropdown menu showing 'financialContract' and a 'Swarm' icon.

Set Deployment Parameters (1/2)

- Run tab: Environment = JavaScript VM

The screenshot shows a Solidity IDE interface. On the left, a code editor displays the following Solidity code for a contract named `financialContract`:

```
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
11 function deposit(uint newDeposit) public{
12     balance = balance + newDeposit;
13 }
14 }
15 }
16 }
```

On the right, the 'Run' tab is active, showing deployment parameters. The 'Environment' dropdown is highlighted with a red box and is set to 'JavaScript VM'. Other parameters include:

- Account: 0xca3...a733c (100 ether)
- Gas limit: 3000000
- Value: 0 wei

Below these settings, the contract name 'financialContract' is selected in a dropdown. There are 'Deploy' and 'At Address' buttons, with the latter having a text input field for '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 displays the Remix IDE interface. On the left, a code editor shows a Solidity contract named `financialContract` with 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
11 function deposit(uint newDeposit) public{
12     balance = balance + newDeposit;
13 }
14 }
15 }
16 }
```

On the right, the 'Run' tab is active, showing deployment settings:

- Environment: JavaScript VM
- Account: 0xca3...a733c (99.9999999999998644)
- Gas limit: 3000000
- Value: 0 wei

The `financialContract` dropdown menu is expanded, and the **Deploy** button is highlighted with a red circle. Below it, the 'At Address' field is visible with the text 'Load contract from Address'.

Below the deployment options, the 'Transactions recorded' section shows 1 transaction. The 'Deployed Contracts' section shows the `financialContract` at address `0x692...77b3a` (memory). The `deposit` function is highlighted with a red circle, and its parameters are `uint256 newDeposit`. The `getBalance` function is also visible below it.

At the bottom of the interface, there is a search bar for transactions and a status bar indicating '[2] only remix transactions, script'.

• Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run from a JavaScript script.

Interact with the Contract

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

Deployed Contracts

financialContract at 0x692...77b3a (memory)

deposit uint256 newDeposit

getBalance

0: uint256: 313000

Press getBalance to see the initial amount

1

Deployed Contracts

financialContract at 0x692...77b3a (memory)

deposit 12

getBalance

0: uint256: 313012

Input a value and press deposit button to create and confirm the transaction

2

Press getBalance again to see the result

3

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

Payable keyword
allows receiving
ether

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4
5     function receiveDeposit() payable public{
6
7     }
8
9     function getBalance() public view returns(uint){
10         return address(this).balance;
11     }
12 }
```

Hidden Code:
Address(this).balance += msg.value;

We can get the
balance of 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

Compile Run Analysis Testing Debugger Settings Support

Environment JavaScript VM VM (-) i

Account + 0xca3...a733c (99.99999999999998944) i

Gas limit 3000000

Value 100 wei

financialContract i

Deploy

or

At Address Load contract from Address

Transactions recorded: 1

Deployed Contracts

financialContract at 0x692...77b3a (memory) i x

receiveDeposit

getBalance

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
15     function getBalance() public view returns(uint){
16         return address(this).balance;
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

```
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 }
```

Only the contract's creator is permitted to withdraw

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

Now deploying a smart contract on an external blockchain

	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	-	+	-	+

Run Ganache

The screenshot shows the Ganache desktop application interface. At the top, there are navigation tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, and LOGS. A search bar is located on the right side of the top bar. Below the navigation, a status bar displays various network parameters: CURRENT BLOCK (0), GAS PRICE (20000000000), GAS LIMIT (6721975), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:7545), and MINING STATUS (AUTOMINING). The main content area is divided into two sections: MNEMONIC and HD PATH. The MNEMONIC section displays the phrase "slim rain lawn kiwi elegant behind vibrant dentist puppy reduce kidney there". The HD PATH section displays "m/44'/60'/0'/0/account_index". Below these sections, a table lists three accounts with their respective addresses, balances, transaction counts, and indices.

ADDRESS	BALANCE	TX COUNT	INDEX	
0x231eAeEF9EA93F5370a1F633F32E45AF570980E8	100.00 ETH	0	0	
0x970fc818790E900598C57E48b89B6D3D8896D416	100.00 ETH	0	1	
0xb59BD5568d0be42C13fB521f845243F1CDaF2eF1	100.00 ETH	0	2	

MyEtherWallet

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

The screenshot shows the MyEtherWallet interface. At the top, the logo and name 'MyEtherWallet' are on the left, and the version '3.21.05', language 'English', gas price '41 Gwei', and current network 'Network ETH (myetherapi.com)' are on the right. A navigation bar contains links for 'New Wallet', 'Send Ether & Tokens', 'Swap', 'Send Offline', 'Contracts', 'ENS', 'DomainSale', 'Check TX Status', 'View Wallet Info', and 'Help'. The main content area is titled 'Create New Wallet' and features a password input field with the text 'Enter a password' and a warning 'Do NOT forget to save this!'. Below the field is a blue 'Create New Wallet' button. A note 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.' At the bottom of the main area are links for 'How to Create a Wallet' and 'Getting Started'. On the right side, a dropdown menu is open, showing a list of networks. The 'Add Custom Network / Node' option at the bottom of the list is circled in red, with a red arrow pointing to it from the top right of the menu.

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

The image shows a screenshot of the Ganache application interface and a dialog box from MyEtherWallet. In the Ganache interface, the 'RPC SERVER' field is circled in red and contains the text 'HTTP://127.0.0.1:7545'. Two red arrows point from this field to the 'URL' and 'Port' fields in the 'Set Up Your Custom Node' dialog. The dialog has a title bar with a close button (X) and a subtitle 'Instructions can be found here'. It contains the following fields and options:

- Node Name:** Private ETH Node
- URL:** http://127.0.0.1
- Port:** 7545
- HTTP Basic access authentication
- ETH ETC Ropsten Kovan Rinkeby Custom Supports EIP-155

At the bottom of the dialog are two buttons: 'Cancel' and 'Save & Use Custom Node'.

MyEtherWallet

- Contracts tab: Deploy Contract

The screenshot displays the MyEtherWallet interface. At the top, the logo and name 'MyEtherWallet' are on the left, and the version '3.21.05', language 'English', gas price '41 Gwei', and network 'My Ether Node:eth (Custom)' are on the right. A warning message states: 'The network is really full right now. Check Eth Gas Station for gas price to use.' Below the header is a navigation menu with options: 'New Wallet', 'Send Ether & Tokens', 'Swap', 'Send Offline', 'Contracts', 'ENS', 'DomainSale', 'Check TX Status', 'View Wallet Info', and 'Help'. The 'Contracts' tab is circled in red, with an arrow pointing to the 'Deploy Contract' option in the main content area, which is also circled in red. The main content area contains the text 'Interact with Contract or Deploy Contract'. Below this, there is a 'Byte Code' section with a large empty text input field, and a 'Gas Limit' section with a text input field containing the value '300000'.

MyEtherWallet 3.21.05 English Gas Price: 41 Gwei Network My Ether Node:eth (Custom)
The network is really full right now. Check Eth Gas Station for gas price to use.

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

Interact with Contract or **Deploy Contract**

Byte Code

Gas Limit

300000

Remix

- Type your contract and compile it



The screenshot displays the Remix IDE interface. On the left, a code editor shows a Solidity contract named `financialContract` with the following code:

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4     uint amount = 13;
5
6     function getValue() public view returns(uint){
7         return amount;
8     }
9
10
11 function setValue(uint newAmount) public{
12     amount = newAmount;
13 }
14 }
15
16
```

On the right, the **Compile** tab is active. It shows the current compiler version as `0.5.1+commit.c8a2cb62.Emscripten.clang`. Below this, there is a dropdown menu labeled **Select new compiler version**. There are three checkboxes: **Auto compile**, **Enable Optimization**, and **Hide warnings**. A red circle highlights the **Start to compile (Ctrl-S)** button. At the bottom of the right panel, the contract name `financialContract` is displayed, along with a **Swarm** icon and buttons for **Details**, **ABI**, and **Bytecode**.

Remix

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

The image shows the Remix IDE interface with the 'financialContract' details panel open. The panel is divided into sections: NAME, METADATA, and BYTECODE. The 'Details' button in the right-hand panel is circled in red, and an arrow points from it to the 'BYTECODE' button in the details panel, which is also circled in red. The 'BYTECODE' section displays the contract's metadata and bytecode.

```
1 pragma solidity
2
3 contract financ
4
5     uint amount
6
7     function ge
8         return
9     }
10
11     function se
12         amount
13     }
14 }
15
16
```

financialContract

NAME

financialContract

METADATA

- ▶ compiler:
- ▶ language: Solidity
- ▶ output:
- ▶ settings:
- ▶ sources:
- ▶ version: 1

BYTECODE

```
{
  "linkReferences": {},
  "object": "6080604052600d60005534801561001557600080fd5b5060e6806100246000396000f3fe608",
  "opcodes": "PUSH1 0x80 PUSH1 0x40 MSTORE PUSH1 0xD PUSH1 0x0 SSTORE CALLVALUE DUP1 ISZ",
  "sourceMap": "25:227:0:-;;;77:2;63:16;;25:227;8:9:-1;5:2;;;30:1;27;20:12;5:2;25:227:0;"
}
```

financialContract

financialContract

file Run Analysis Testing Debugger Settings

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

Details ABI Bytecode

financialContract

Ganache

Access your private key for signing your contract in MyEtherWallet.

The image shows the Ganache application interface. At the top, there are navigation tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, and LOGS. Below these, a status bar displays network information: CURRENT BLOCK 0, GAS PRICE 20000000000, GAS LIMIT 6721975, NETWORK ID 5777, RPC SERVER HTTP://127.0.0.1:7545, and MINING STATUS AUTOMINING. The main area shows account details for the first account: MNEMONIC (slim rain lawn kiwi elegant behind vibrant dentist puppy reduce kidney there), HD PATH (m/44'/60'/0'/0/account_index), ADDRESS (0x231eAeEF9EA93F5370a1F633F32E45AF570980E8), BALANCE (100.00 ETH), TX COUNT (0), and INDEX (0). A red circle highlights a key icon next to the account details. A modal window is open in the foreground, showing the same account details and a PRIVATE KEY field containing the hexadecimal string a53cf8cb7b66d91ca388ef9ce4e45e39997f2773247c27bb2c7cae35a1b3d383, which is circled in red. A red arrow points from the key icon in the background to the private key in the modal. A DONE button is visible at the bottom of the modal.

ADDRESS	BALANCE	TX COUNT	INDEX	KEY
0x231eAeEF9EA93F5370a1F633F32E45AF570980E8	100.00 ETH	0	0	Key icon
0x970fc818790E900598C...		0	1	Key icon
0xb59BD5568d0be42C13f...		0	2	Key icon
0x280AFA533B9fa1A97a6...		0	3	Key icon
0xD6D30F824B17c30460F2CAc88425FCa8f2757c5...	100.00 ETH	0	4	Key icon

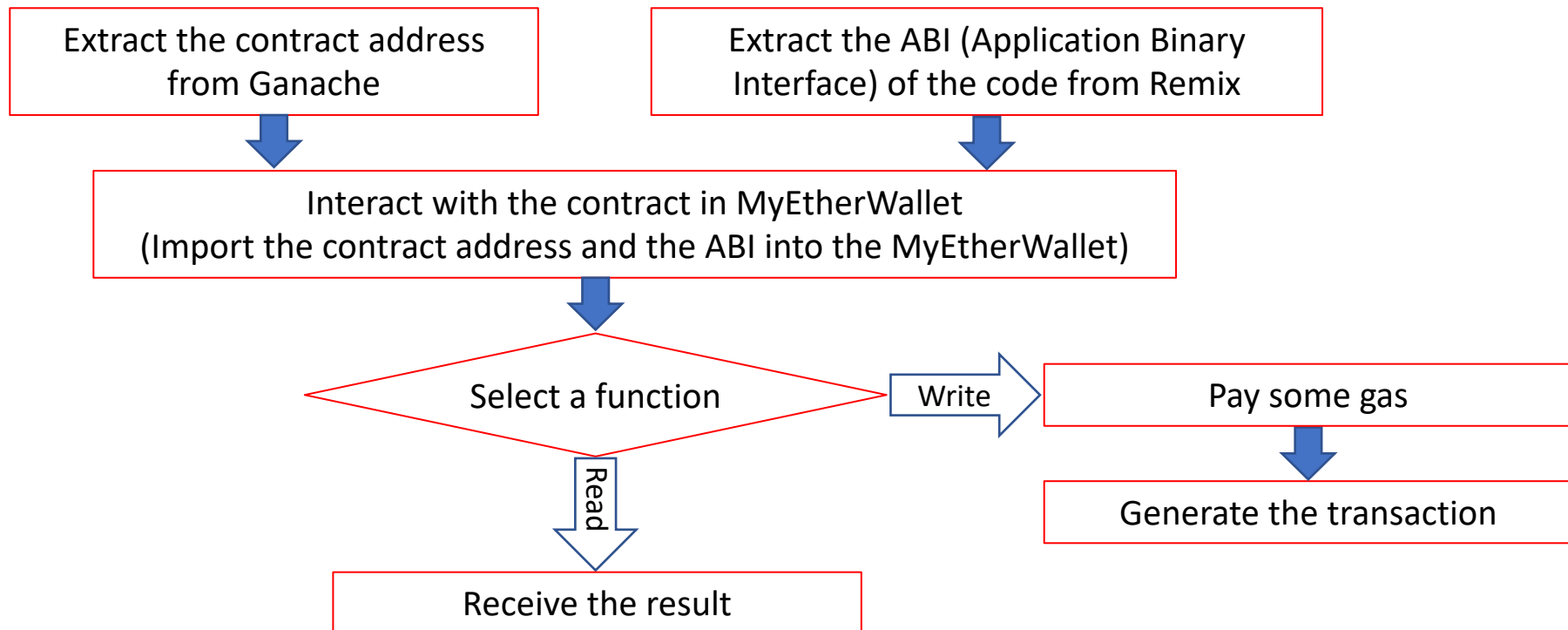
Ganache

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 desktop application interface. At the top, there are navigation tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, and LOGS. A search bar is located on the right side of the top bar. Below the navigation tabs, there is a status bar with the following information: CURRENT BLOCK 1 (circled in red), GAS PRICE 20000000000, GAS LIMIT 6721975, NETWORK ID 5777, RPC SERVER HTTP://127.0.0.1:7545, and MINING STATUS AUTOMINING. Below the status bar, there is a section for the MNEMONIC (slim rain lawn kiwi elegant behind vibrant dentist puppy reduce kidney there) and the HD PATH (m/44'/60'/0'/0/account_index). The main part of the interface is a table of accounts. The first account is circled in red and has a balance of 99.99 ETH and a TX COUNT of 1. The other four accounts have a balance of 100.00 ETH and a TX COUNT of 0.

ADDRESS	BALANCE	TX COUNT	INDEX	KEY
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 interface. The top navigation bar includes 'ACCOUNTS', 'BLOCKS', 'TRANSACTIONS', and 'LOGS'. The 'TRANSACTIONS' tab is highlighted with a red circle. Below the navigation bar, there is a status bar with the following information:

CURRENT BLOCK	GAS PRICE	GAS LIMIT	NETWORK ID	RPC SERVER	MINING STATUS
1	20000000000	6721975	5777	HTTP://127.0.0.1:7545	AUTOMINING

Below the status bar, the transaction details are displayed:

TX HASH				CONTRACT CREATION
0x1e40cc28802d152e810bd9f40bea83d83b1655fc9bace6e801ec6db5fcd84b1a				
FROM ADDRESS	CREATED CONTRACT ADDRESS	GAS USED	VALUE	
0x231eAeEF9EA93F5370a1F633F32E45AF570980E8	0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d	124604	0	

Remix

Copy the ABI

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



The screenshot displays the Remix IDE interface. On the left, a code editor shows a Solidity contract named `financialContract` with the following code:

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4     uint amount = 13;
5
6     function getValue() public view returns(uint){
7         return amount;
8     }
9
10    function setValue(uint newAmount) public{
11        amount = newAmount;
12    }
13 }
14
15
16
```

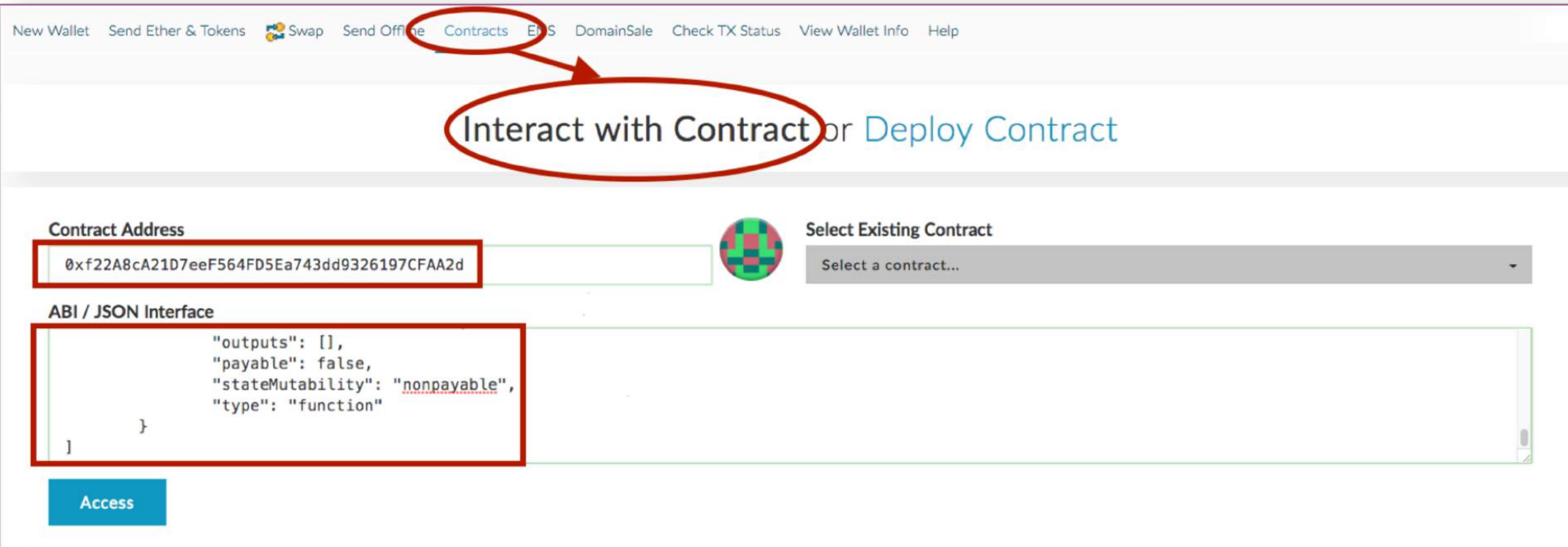
On the right, the compiler settings panel shows the current version as `0.5.1+commit.c8a2cb62.Emscripten.clang`. Below this, there is a dropdown menu for selecting a new compiler version, a checked `Auto compile` checkbox, and unchecked checkboxes for `Enable Optimization` and `Hide warnings`. A `Start to compile (Ctrl-S)` button is also present.

Below the compiler settings, the contract name `financialContract` is displayed in a dropdown menu. To the right of the dropdown is a `Swarm` icon. Below the dropdown, there are three buttons: `Details`, `ABI`, and `Bytecode`. The `ABI` button is circled in red. Below these buttons, a green box displays the contract name `financialContract` with a close icon.

MyEtherWallet

Contracts tab:

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



The screenshot shows the MyEtherWallet interface with the 'Contracts' tab selected. The 'Interact with Contract' link is circled in red, with an arrow pointing to the 'Contract Address' input field, which also contains a red box around the address. The 'ABI / JSON Interface' field contains a red box around the JSON code. The 'Access' button is highlighted in blue.

New Wallet Send Ether & Tokens Swap Send Offline **Contracts** Ethers 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

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

Read / Write Contract
0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d

Select a function ▾

- getValue
- setValue

MyEtherWallet

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

Read / Write Contract

0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d

getValue ▾

↳ uint256

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.

Read / Write Contract

0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d

setValue ▾

newValue uint256

WRITE

Warning!

You are about to execute a function on contract.
It will be deployed on the following network: ETH (Custom).

Amount to Send *In most cases you should leave this as 0.*

Gas Limit

Generate Transaction

Raw Transaction

```
{"nonce": "0x01", "gasPrice": "0x098bca5a00", "gasLimit": "0xa2a1", "to": "0xf22A8cA21D7eeF564FD5Ea743d"}
```

Signed Transaction

```
0xf8680185098bca5a0082a2a194f22a8ca21d7eeF564fd5ea743dd9326197cfaa2d80845b34b96626a04285bd52ad31
```

No, get me out of here! Yes, I am sure! Make transaction.

197CFAA2d

WRITE

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...]

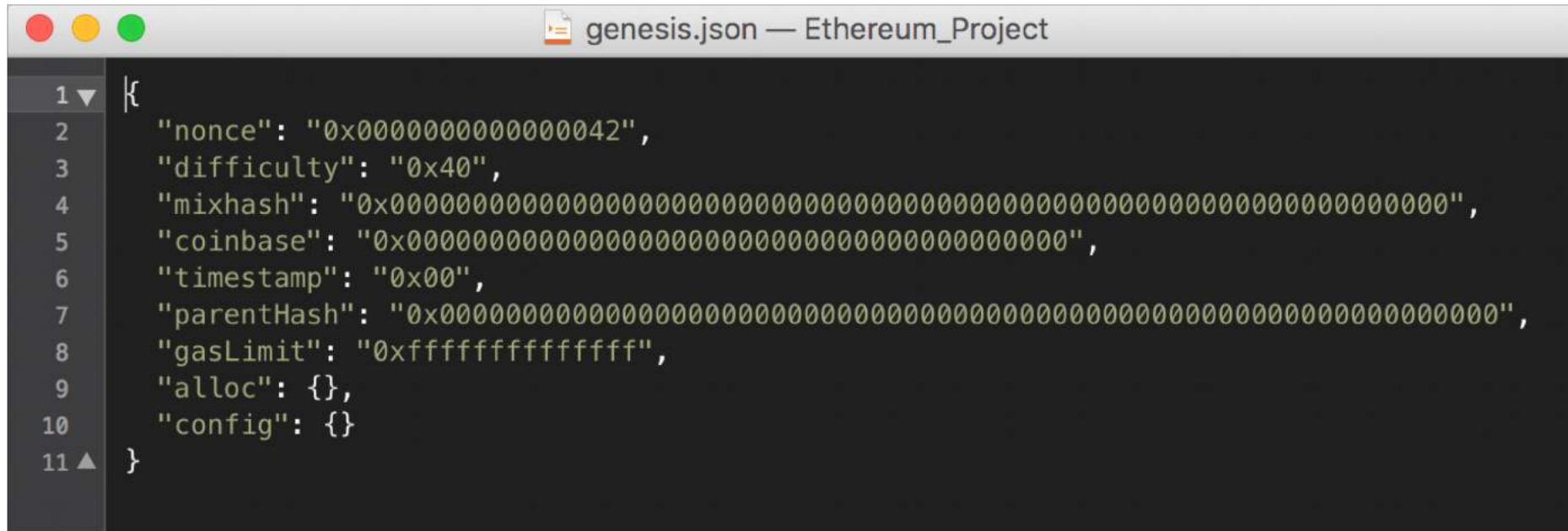
VERSION:
  1.8.9-stable

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

A screenshot of a code editor window titled "genesis.json — Ethereum_Project". The editor shows a JSON object with the following fields: "nonce", "difficulty", "mixhash", "coinbase", "timestamp", "parentHash", "gasLimit", "alloc", and "config". The "nonce" field has the value "0x00000000000000042", "difficulty" is "0x40", "mixhash" is a long string of 64 zeros, "coinbase" is a long string of 64 zeros, "timestamp" is "0x00", "parentHash" is a long string of 64 zeros, "gasLimit" is "0xffffffff", "alloc" is an empty object {}, and "config" is an empty object {}.

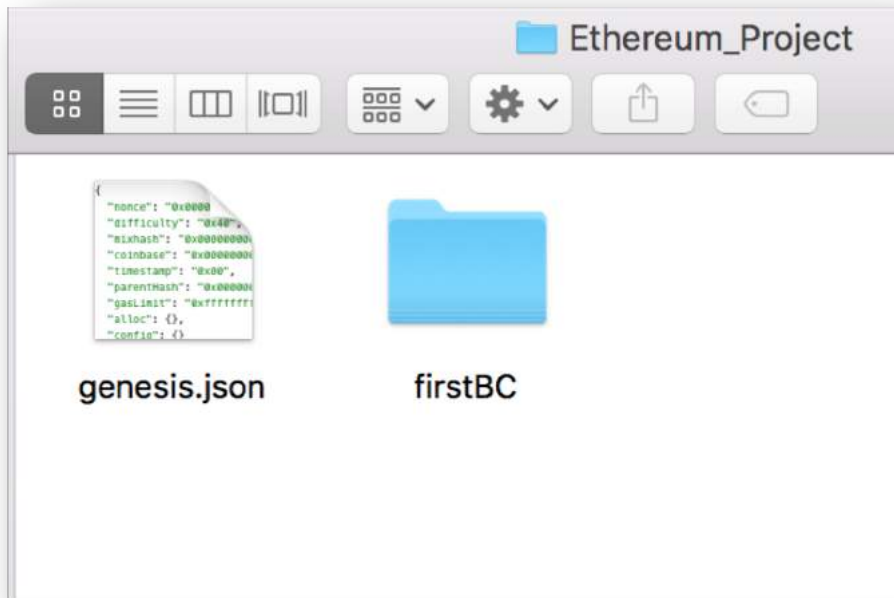
```
1 {  
2   "nonce": "0x00000000000000042",  
3   "difficulty": "0x40",  
4   "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",  
5   "coinbase": "0x0000000000000000000000000000000000000000000000000000000000000000",  
6   "timestamp": "0x00",  
7   "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",  
8   "gasLimit": "0xffffffff",  
9   "alloc": {},  
10  "config": {}  
11 }
```

- 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

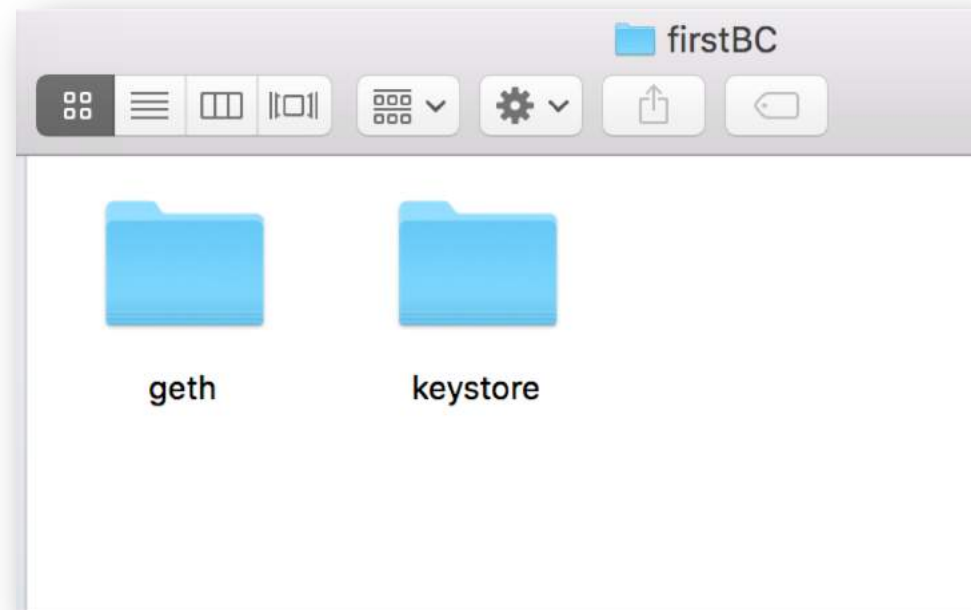
```
[ds-install:Documents mohammht$ cd Ethereum_Project/  
ds-install:Ethereum_Project mohammht$ geth --datadir firstBC init genesis.json
```



Folder name of your
blockchain

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://4561ccdd7fdf3f0bdbc903b7bef7d472e136fe2b63012151a1dd3c27e52f49bda2ef66631e67022b7ca7b9fba06bb0eda8b47210b198f3eeff7e67414d695ed6@[::]:30303",
  id: "4561ccdd7fdf3f0bdbc903b7bef7d472e136fe2b63012151a1dd3c27e52f49bda2ef66631e67022b7ca7b9fba06bb0eda8b47210b198f3eeff7e67414d695ed6",
  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
    }
  }
}
```


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=18000000000
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!

Any Questions?



Mohammad H. Tabatabaei
mohammht@ifi.uio.no