



COMPLETE
NOTES ON
BLOCKCHAIN

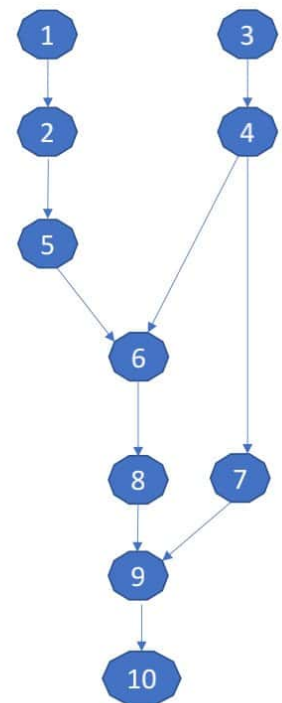
Different tools provide different functionality

	Tools	Remix	Ganache	MyEtherWallet	Geth
Activities					
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



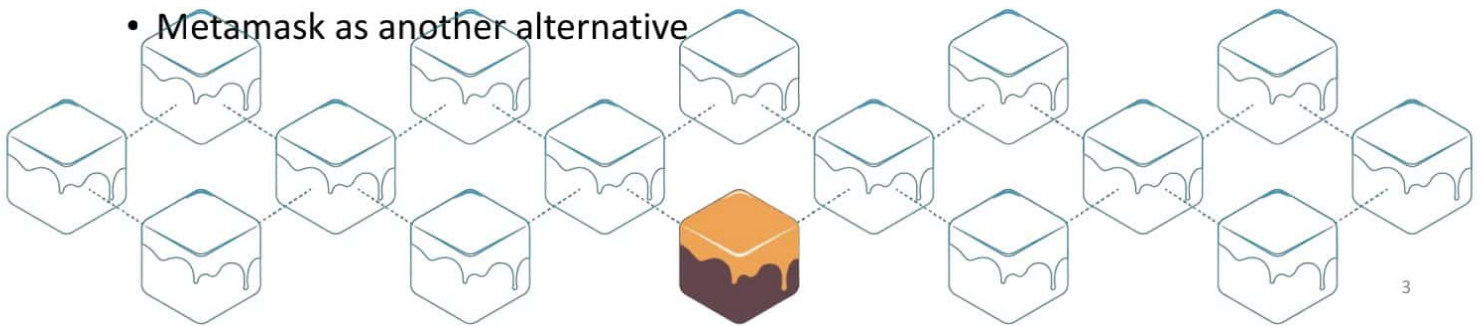
Use which tool for what purpose? (2/2)

- Why use Ganache?

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

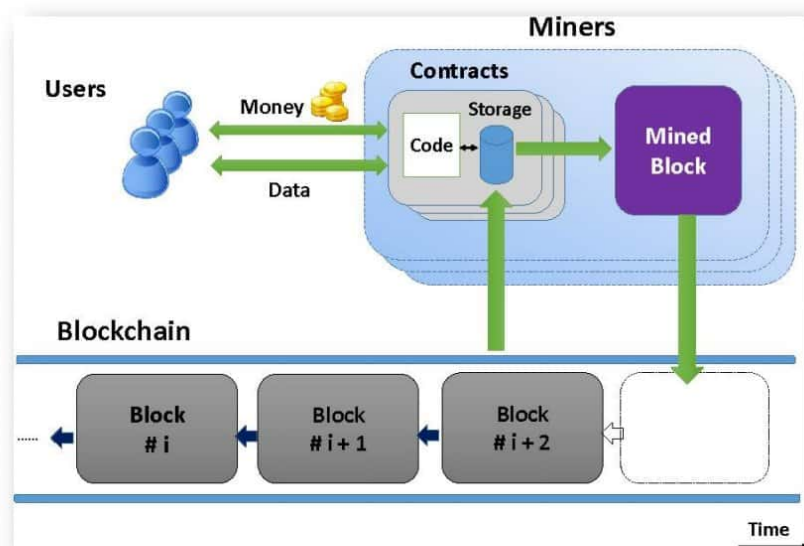
- Why use MyEtherWallet?

- 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

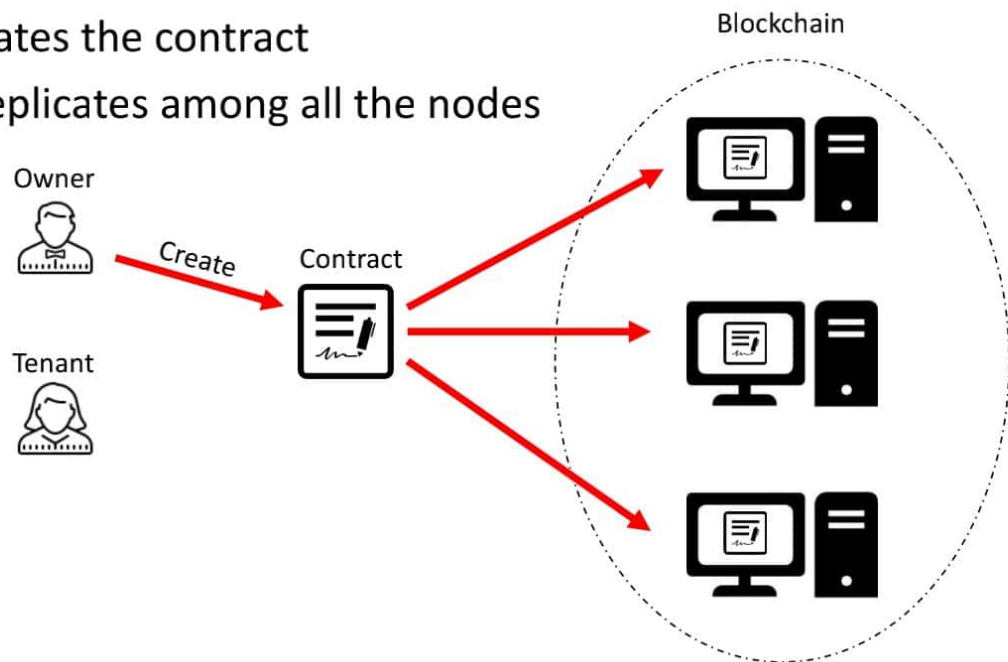


- K. Delmolino, M. Arnett, A. E. Kosba, A. Miller, and E. Shi, "Step by Step Towards Creating a Safe Smart Contract: Lessons and Insights from a Cryptocurrency Lab," *IACR Cryptology ePrint Archive*, vol. 2015, p. 460, 2015.

4

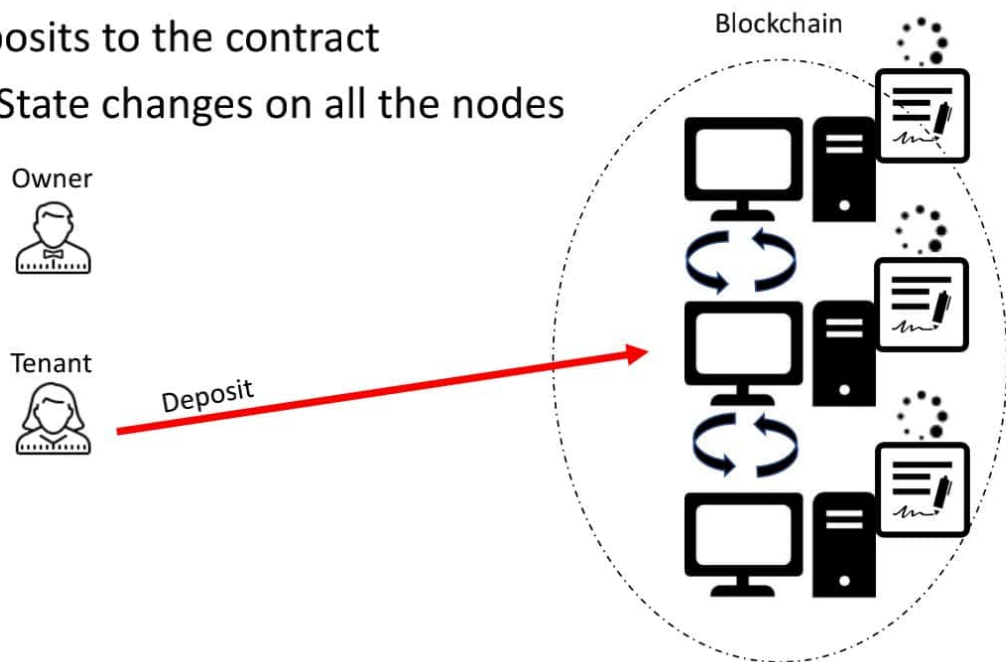
Smart Contracts Example (1/3)

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



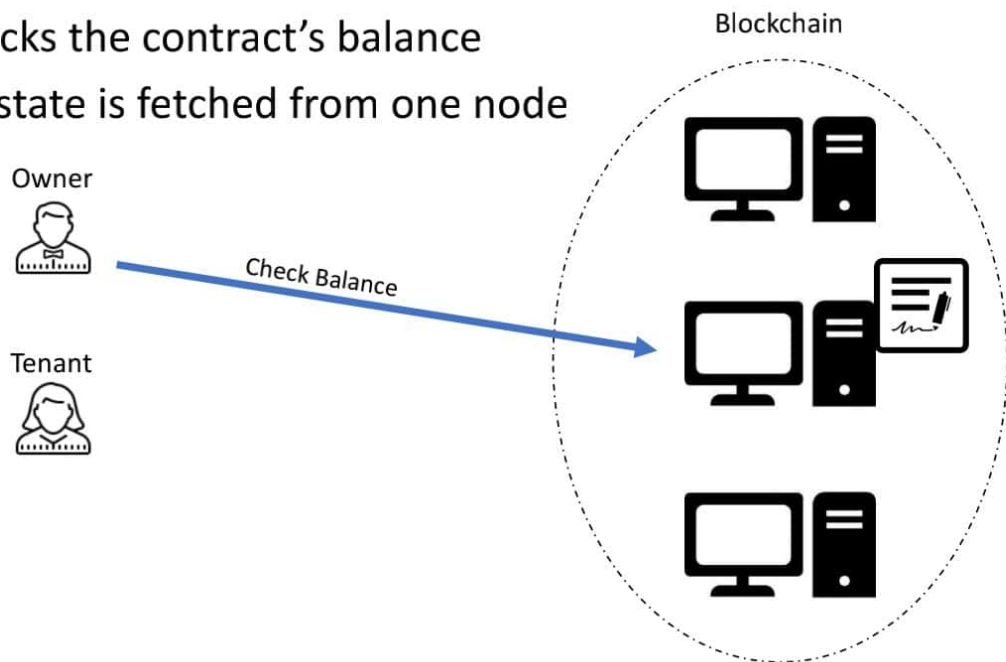
Smart Contracts Example (2/3)

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



Smart Contracts Example (3/3)

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

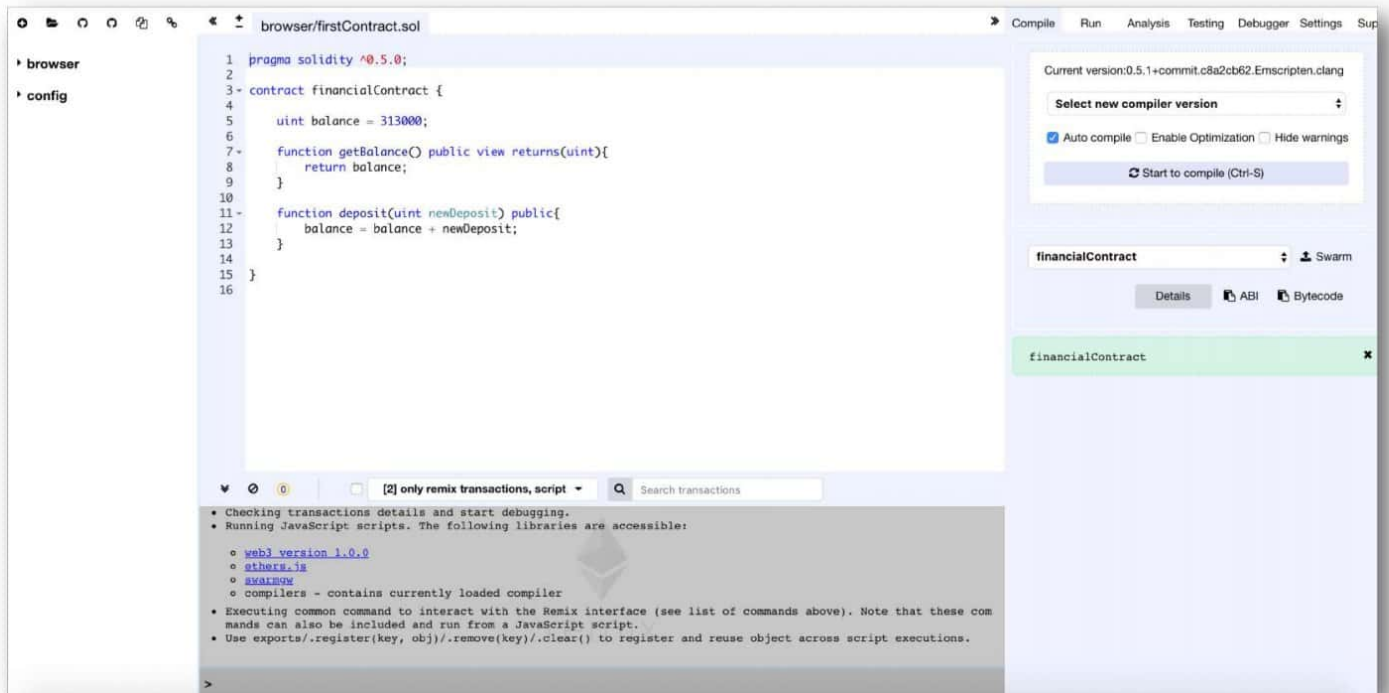


Smart Contracts

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



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 }
```

Variable

Getter function

Setter function

The diagram illustrates a Solidity smart contract named 'financialContract'. It contains three main components: a state variable 'balance' of type 'uint' initialized to 313000, a 'getBalance' function that returns the current balance, and a 'deposit' function that increases the balance by a specified amount. Annotations with colored boxes and arrows identify these components: a yellow box for the variable, a blue box for the getter function, and a red box for the setter function.

Compile the Contract

- Compile tab: Start to compile button



Set Deployment Parameters (1/2)

- Run tab: Environment = JavaScript VM

The screenshot displays the Remix IDE interface. On the left, the 'browser/firstContract.sol' file is open, showing Solidity code for a 'financialContract'. The code includes a pragma statement for Solidity version ^0.5.0, a contract definition, a constant balance of 313000, a 'getBalance' function, and a 'deposit' function. On the right, the 'Run' tab is active, showing deployment parameters. The 'Environment' is set to 'JavaScript VM', which is highlighted with a red rectangle. Other parameters include 'Account' (0xca3...a733c (100 ether)), 'Gas limit' (3000000), and 'Value' (0 wei). Below these, the contract name 'financialContract' is entered in the deployment field. A 'Deploy' button is visible, along with an 'At Address' option to load a contract from a specific address.

```
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 }
```

Environment: JavaScript VM (VM (-) ⓘ)

Account: 0xca3...a733c (100 ether) ⓘ

Gas limit: 3000000

Value: 0 wei ⓘ

financialContract ⓘ

Deploy

or

At Address: Load contract from Address

Set Deployment Parameters (2/2)

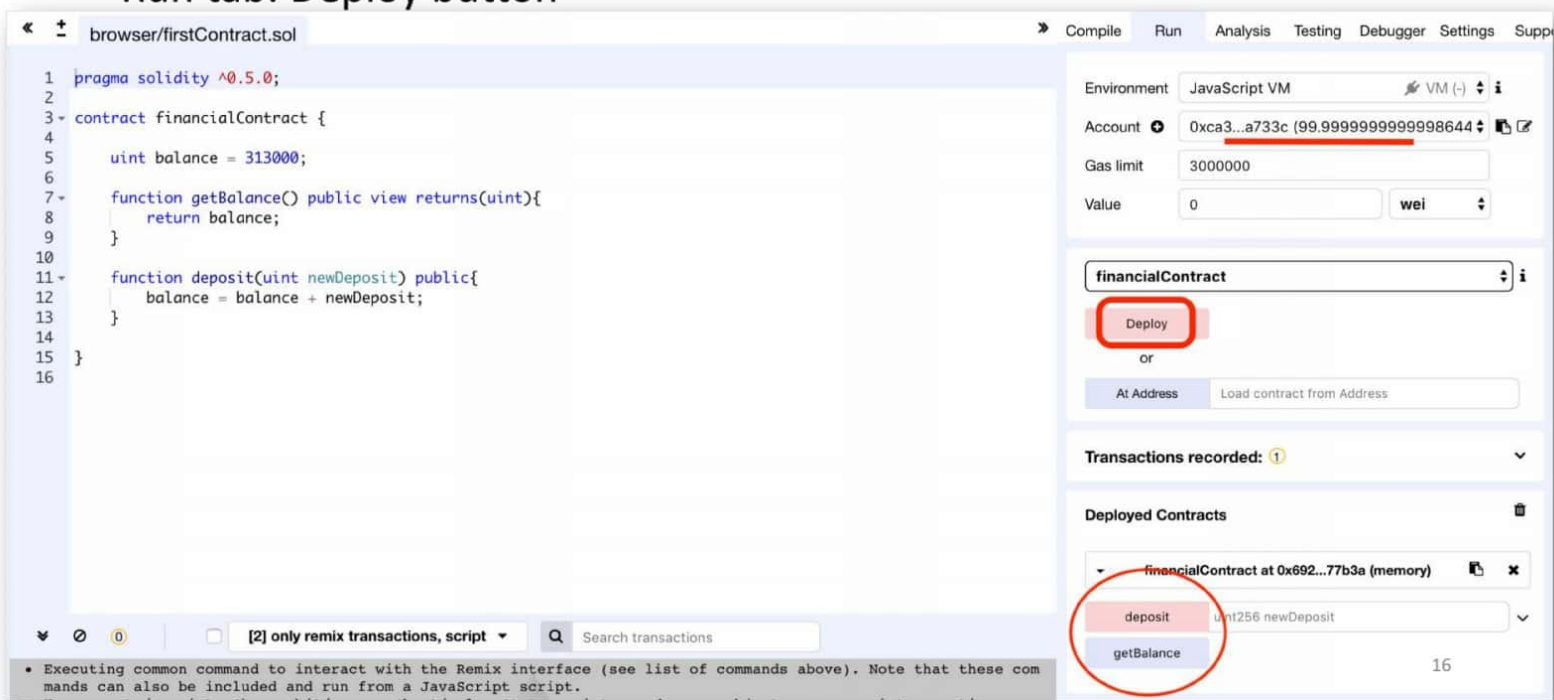
- JavaScript VM: All the transactions will be executed in a sandbox blockchain in the browser. Nothing will be persisted and a page reload will restart a new blockchain from scratch, the old one will not be saved.
- Injected Provider: Remix will connect to an injected web3 provider. Mist and Metamask 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 ($\text{wei} = 10^{-18}$ 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.

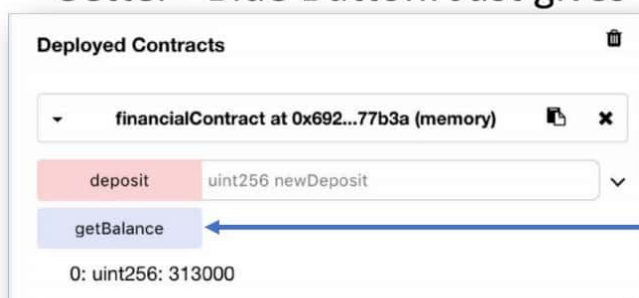
Deploy the Contract on the Private Blockchain of Remix

- Run tab: Deploy button



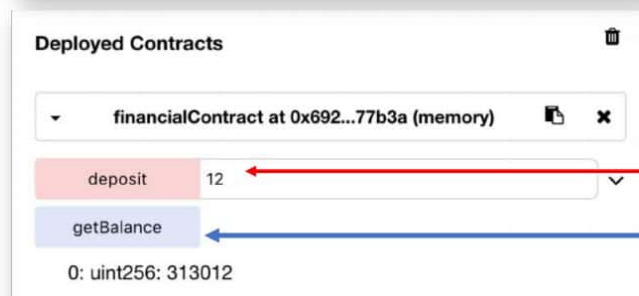
Interact with the Contract

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



Press getBalance to see the initial amount

1



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

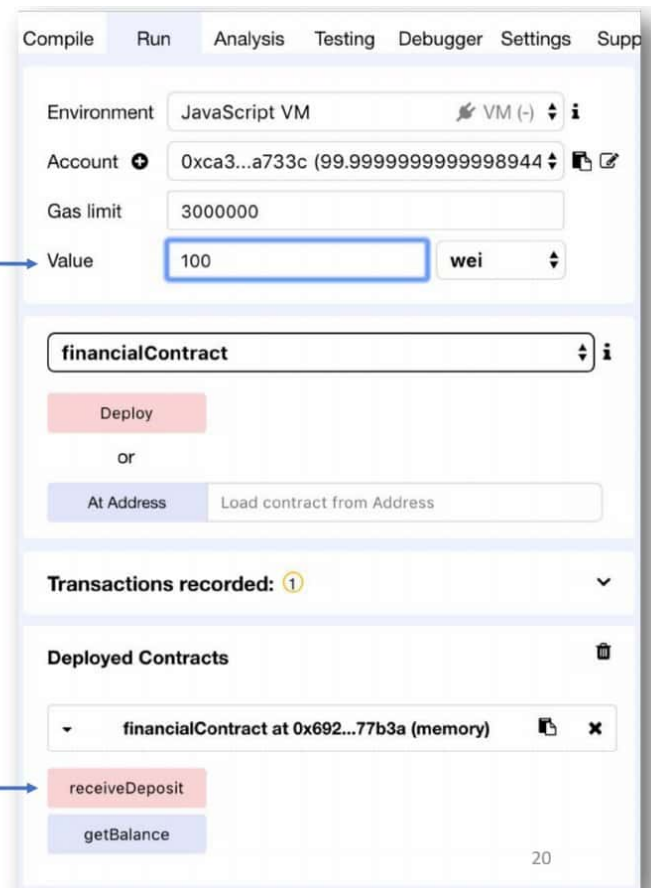
Receive ether (2/2)

1

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

2

Click the receiveDeposit button to
transfer the money to the
contract



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

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 }
```

22

Now deploying a smart contract on an external blockchain

	Tools Activities	Remix	Ganache	MyEtherWallet	Geth
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

Ganache

ACCOUNTS

BLOCKS

TRANSACTIONS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK
0

GAS PRICE
20000000000

GAS LIMIT
6721975

NETWORK ID
5777

RPC SERVER
HTTP://127.0.0.1:7545

MINING STATUS
AUTOMINING

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

MyEtherWallet

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

The screenshot shows the MyEtherWallet website interface. The main heading is "Create New Wallet". Below it, there is a section titled "Enter a password" with a text input field containing the placeholder "Do NOT forget to save this!". To the right of the input field is an eye icon. Below the input field is a blue button labeled "Create New Wallet".

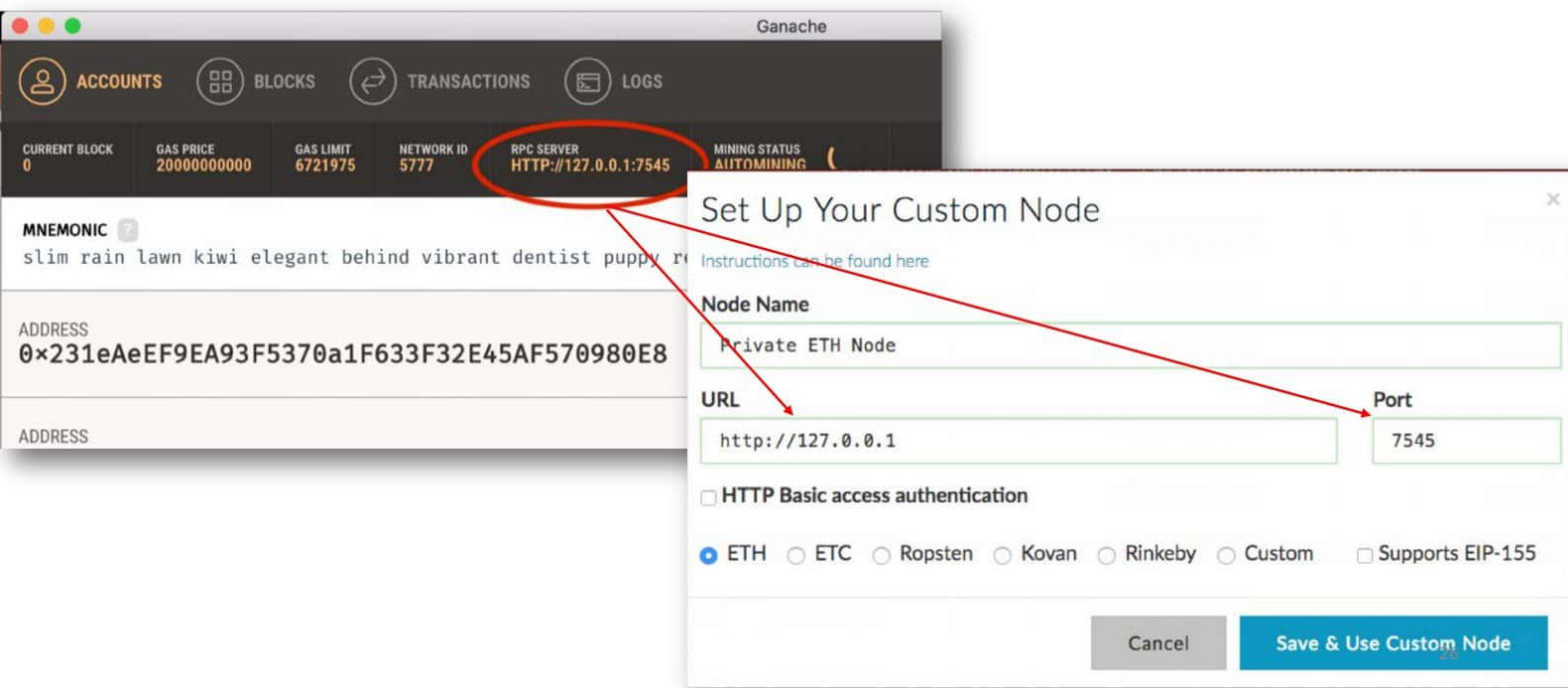
At the top right of the page, there is a dropdown menu for selecting a network. The current selection is "Network ETH (myetherapi.com)". The dropdown menu is open, showing a list of networks. A red arrow points to the "Add Custom Network / Node" option at the bottom of the list.

The list of networks includes:

- ETH (myetherapi.com)
- ETH (etherscan.io)
- ETH (infura.io)
- ETH (livethat)
- ETC (Ethereum Commonwealth)
- ETC (etccoin.io)
- Ropsten (myetherapi.com)
- Ropsten (infura.io)
- Kovan (etherscan.io)
- Kovan (infura.io)
- Rinkeby (etherscan.io)
- Rinkeby (infura.io)
- EXP (expnet.tech)
- USQ (ubiqscan.io)
- POA (core.poa.network)
- TOMO (core.tomonetwork.io)
- ELLA (ellamint.org)
- ETC (etccoin.io)
- Add Custom Network / Node

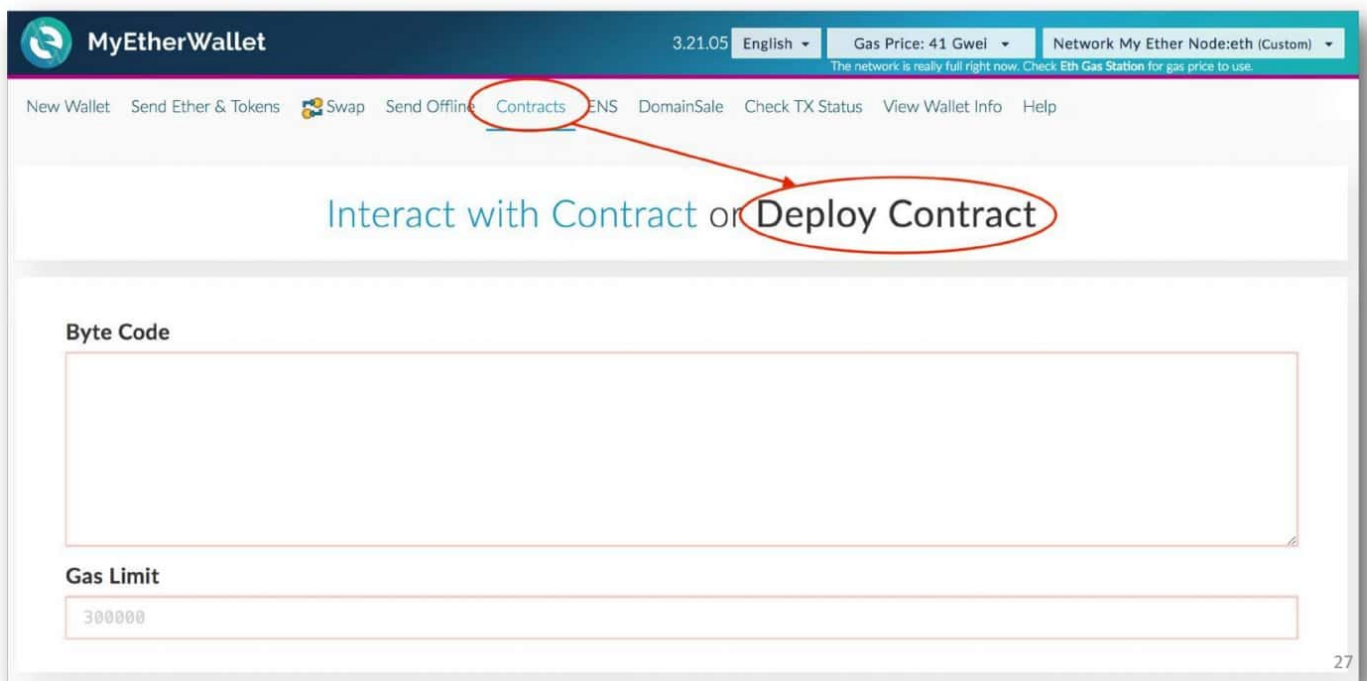
At the bottom of the page, there is a disclaimer: "MyEtherWallet.com does not hold your keys for you. We cannot access accounts, recover keys, reset passwords, nor reverse transactions. Protect your keys & always check that you are on correct URL. You are responsible for your security."

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



MyEtherWallet

- Contracts tab: Deploy Contract



The screenshot shows the MyEtherWallet interface. At the top, there is a header bar with the MyEtherWallet logo, version 3.21.05, language set to English, gas price of 41 Gwei, and network set to My Ether Node:eth (Custom). Below the header, a navigation bar contains links: New Wallet, Send Ether & Tokens, Swap, Send Offline, **Contracts**, ENS, DomainSale, Check TX Status, View Wallet Info, and Help. The **Contracts** link is circled in red, and an arrow points from it to the **Deploy Contract** option in the main content area. The main content area has a heading "Interact with Contract or **Deploy Contract**". Below this, there are two input fields: "Byte Code" and "Gas Limit". The "Gas Limit" field is pre-filled with the value "300000".

MyEtherWallet 3.21.05 English Gas Price: 41 Gwei Network My Ether Node:eth (Custom)

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

Interact with Contract or **Deploy Contract**

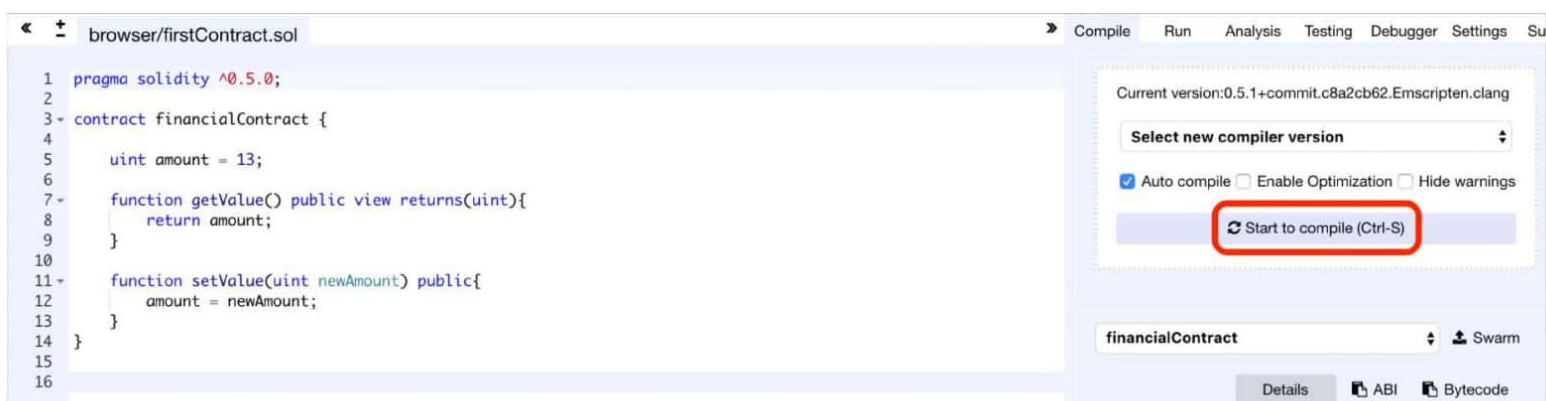
Byte Code

Gas Limit

300000

Remix

- Type your contract and compile it



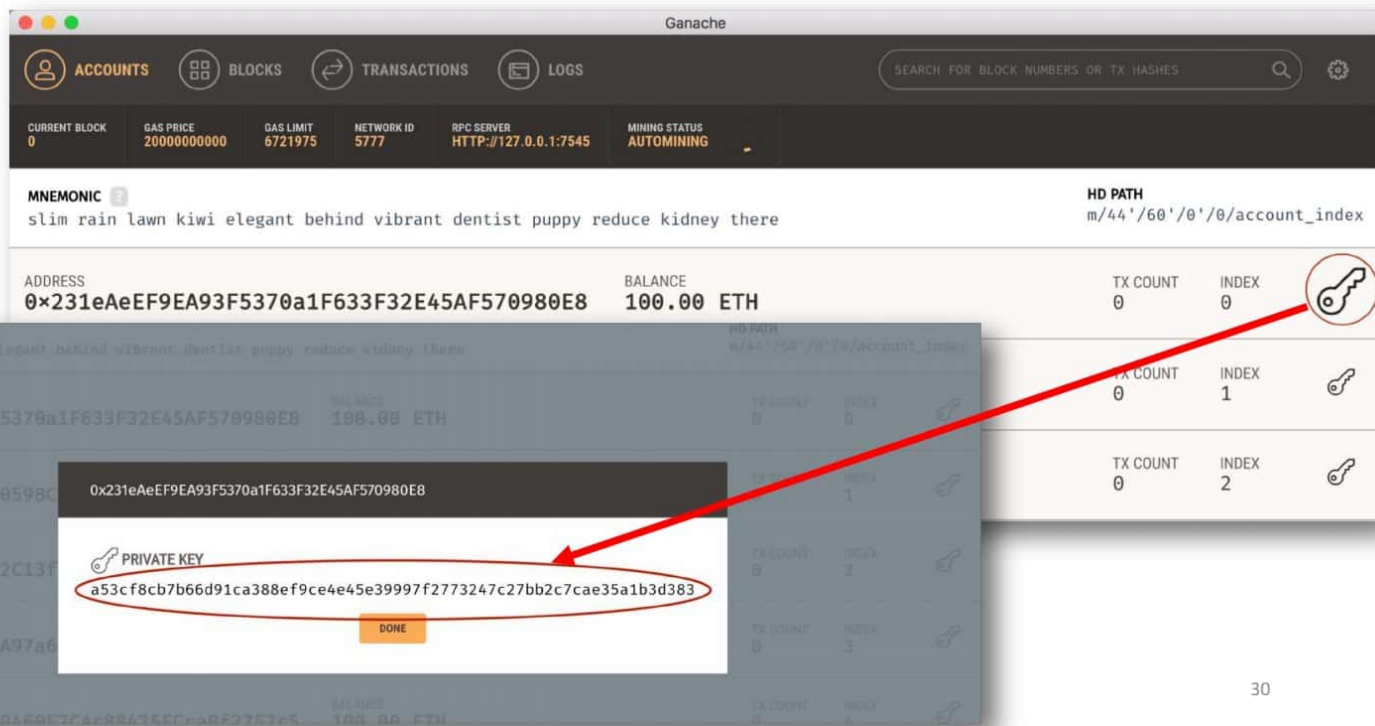
Remix

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



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

[illegible]

MyEtherWallet

Click on *Sign Transaction* button to deploy your contract

[illegible]

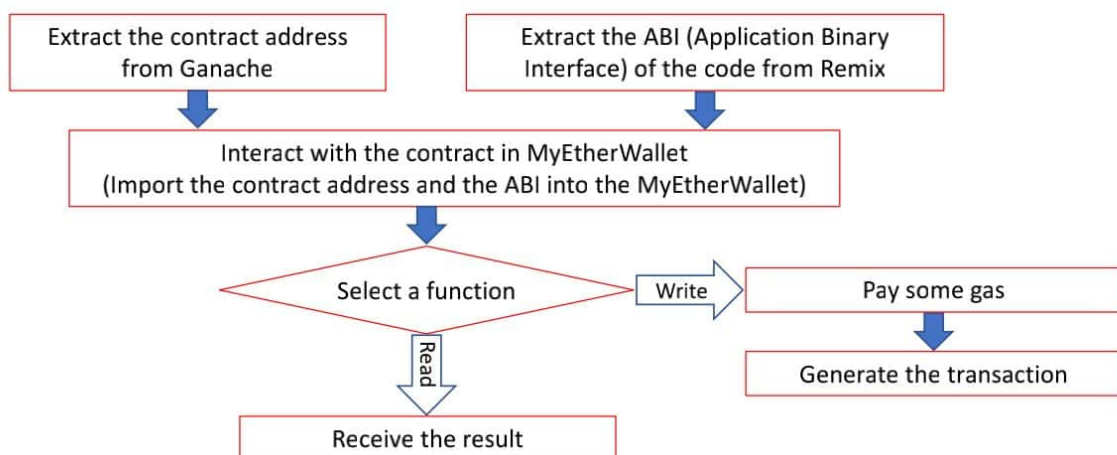
Ganache

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, a status bar displays various network metrics. The main area shows a list of accounts with their addresses, balances, and transaction counts. The first account's balance (99.99 ETH) and transaction count (1) are circled in red.

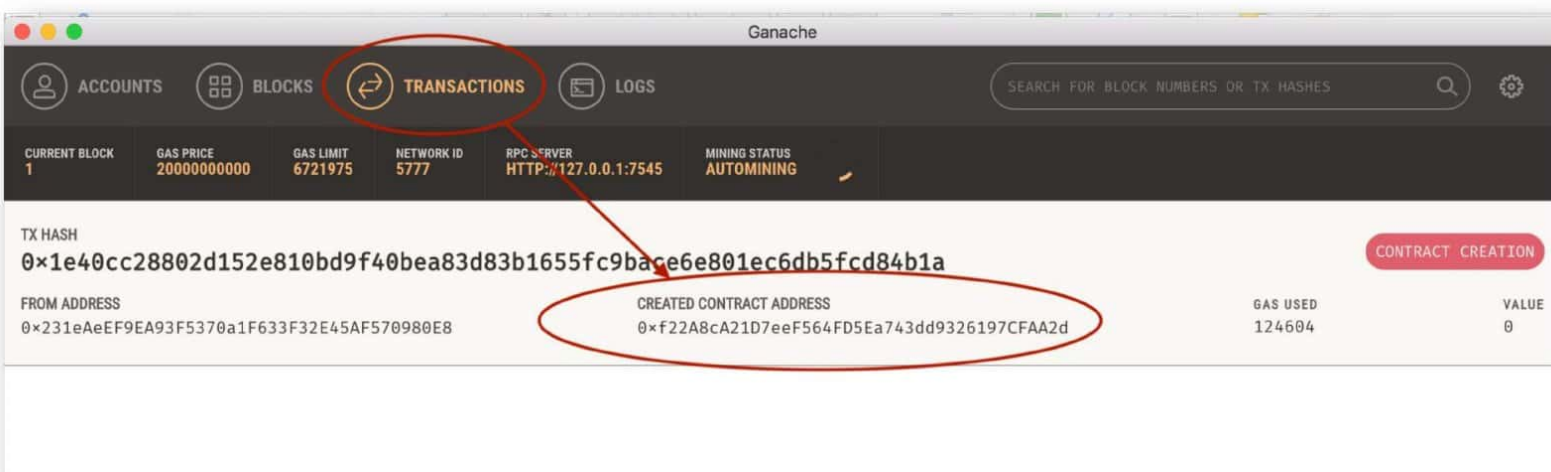
ACCOUNTS	BLOCKS	TRANSACTIONS	LOGS
ACCOUNTS			
CURRENT BLOCK: 1			
GAS PRICE: 20000000000			
GAS LIMIT: 6721975			
NETWORK ID: 5777			
RPC SERVER: HTTP://127.0.0.1:7545			
MINING STATUS: AUTOMINING			
MNEMONIC: slim rain lawn kiwi elegant behind vibrant dentist puppy reduce kidney there			
HD PATH: m/44'/60'/0'/0/account_index			
ADDRESS: 0x231eAeEF9EA93F5370a1F633F32E45AF570980E8	BALANCE: 99.99 ETH	TX COUNT: 1	INDEX: 0
ADDRESS: 0x970fc818790E900598C57E48b89B6D3D8896D416	BALANCE: 100.00 ETH	TX COUNT: 0	INDEX: 1
ADDRESS: 0xb59BD5568d0be42C13fB521f845243F1CDaF2eF1	BALANCE: 100.00 ETH	TX COUNT: 0	INDEX: 2
ADDRESS: 0x280AFA533B9fa1A97a6D2E4640412FD86FC5dd36	BALANCE: 100.00 ETH	TX COUNT: 0	INDEX: 3
ADDRESS: 0xD6D39E82AB17c30460F2CAc88425ECcaBf2757c5	BALANCE: 100.00 ETH	TX COUNT: 0	INDEX: 4

Interacting with the smart contract



Ganache

Transactions tab: Copy the created contract address



Remix

Copy the ABI

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



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

ABI / JSON Interface
{
 "outputs": [],
 "payable": false,
 "stateMutability": "nonpayable",
 "type": "function"
}

Select Existing Contract
Select a contract...

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)



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 image shows the MyEtherWallet interface. On the left, the 'Read / Write Contract' section displays a contract address: 0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d. A dropdown menu shows 'setValue'. Below it, the 'newValue' field is set to '6'. A red circle highlights the 'newValue' field, and an arrow points from it to the 'WRITE' button at the bottom. On the right, a 'Warning!' modal is open, indicating that a function will be executed on the contract. It shows the 'Amount to Send' as 0 and the 'Gas Limit' as 41633. A red circle highlights the 'Generate Transaction' button in the modal. Below this, the 'Raw Transaction' and 'Signed Transaction' are displayed. The 'Raw Transaction' is a JSON object: {"nonce": "0x01", "gasPrice": "0x098bca5a00", "gasLimit": "0xa2a1", "to": "0xf22A8cA21D7eeF564FD5Ea743d...". The 'Signed Transaction' is a long hexadecimal string: 0xf8680185098bca5a0082a2a194f22a8ca21d7eeef564fd5ea743dd9326197cf... At the bottom of the modal, there are two buttons: 'No, get me out of here!' and 'Yes, I am sure! Make transaction.'.

Read / Write Contract

0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d

setValue

newValue uint256

6

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.*

0

Gas Limit

41633

Generate Transaction

Raw Transaction

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

Signed Transaction

```
0xf8680185098bca5a0082a2a194f22a8ca21d7eeef564fd5ea743dd9326197cf...
```

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

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...]

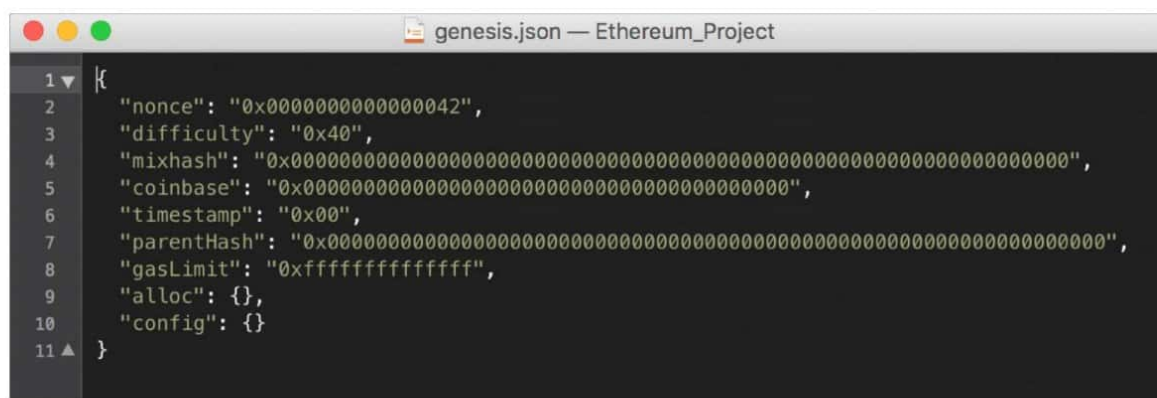
VERSION:
  1.8.9-stable

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 a value of "0x0000000000000042". The "difficulty" field has a value of "0x40". The "mixhash" field has a value of "0x00". The "coinbase" field has a value of "0x00". The "timestamp" field has a value of "0x00". The "parentHash" field has a value of "0x00". The "gasLimit" field has a value of "0xffffffffffffffff". The "alloc" field has a value of "{}". The "config" field has a value of "{}".

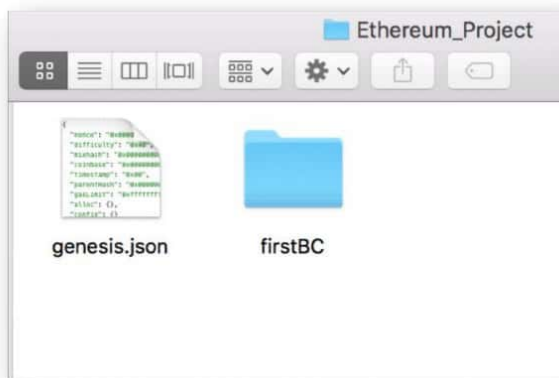
```
1 {  
2   "nonce": "0x0000000000000042",  
3   "difficulty": "0x40",  
4   "mixhash": "0x0000000000000000000000000000000000000000000000000000000000000000",  
5   "coinbase": "0x0000000000000000000000000000000000000000000000000000000000000000",  
6   "timestamp": "0x00",  
7   "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",  
8   "gasLimit": "0xffffffffffffffff",  
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

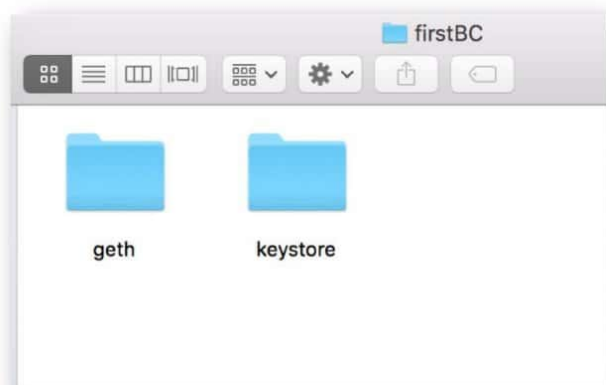
```
ds-install:Documents mohammht$ cd Ethereum_Project/  
ds-install:Ethereum_Project mohammht$ geth --datadir firstBC init genesis.json
```



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=10000000000
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