Smart Contracts and Solidity Basics

Prof. (Dr.) Neeraj Kumar

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Prerequisite

- Blockchain
- Ethereum
- Smart contracts

Blockchain

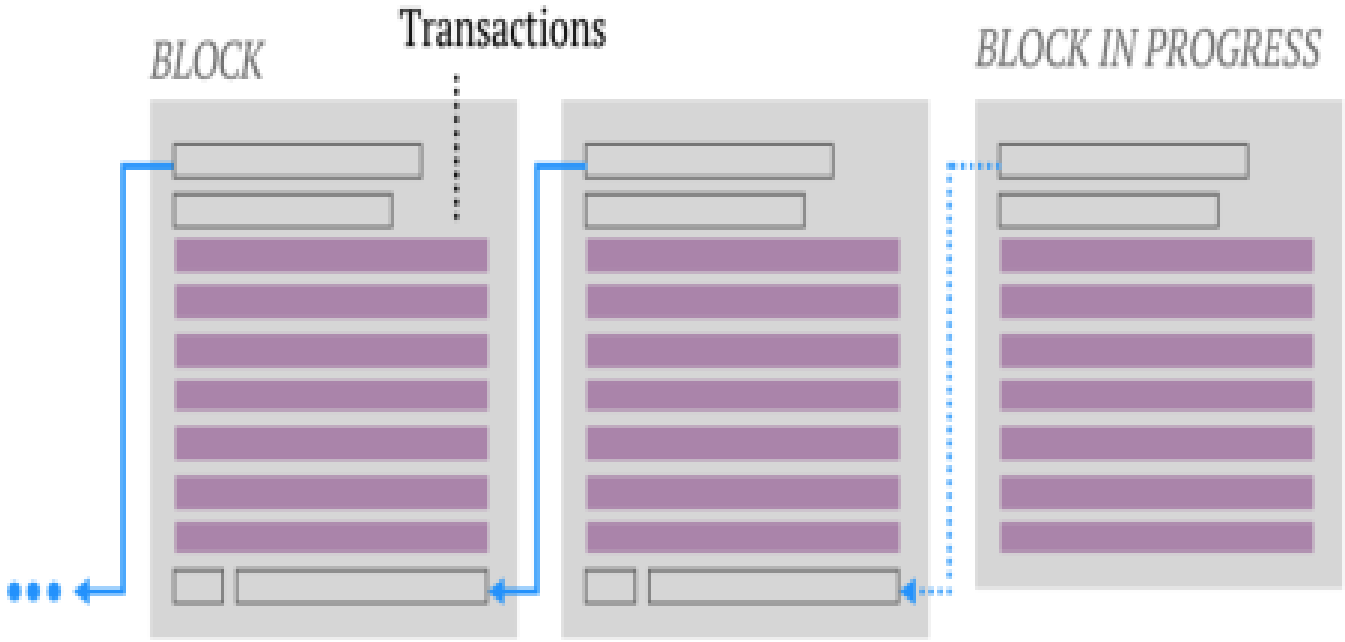
Fully Distributed Database like BTC

<u>Advantages</u>:

- Highly Secure
- Transparent
- Immutable

<u>Disadvantages</u>:

- Scaling
- Performance



Summarizing Blockchain

- It is a decentralized distributed ledger (data structure) where data is being stored inside blocks in form of transactions.
- Remove the dependency on the trusted third party for recording the data in Blocks.
- Since each block is built on top of previous Block immutability has been achieved.
- Very difficult to add a fake block & very easy to detect the fake Block.
- Every participants of the Blockchain contain the almost same copy of the Blockchain.

Blockchain Main Themes

- Longitudinal immutability
- Reliable timestamps
- Cryptographic security
- Fully distributed replication
- Publicly verifiable code governing all transactions
- Value in Smart Contract Assets

What is Ethereum?

- "A planetary scale computer built on blockchain technology"
- Open for all to use; zero infrastructure
- Turing-Complete: Build arbitrarily complex applications
- "Ethereum is a decentralized platform that runs **smart contracts**: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third party interference."
- Ethereum Virtual Machine, runs on **Ether**, Pay per computational step

ethereum

Ethereum Virtual Machine (EVM)

- A Turing Complete Virtual Machine accessible from anywhere in the world has a number of benefits for the automated-economy:
- The Ethereum Blockchain is:
  - A blockchain with a built-in programming language
  - A decentralized, massively replicated database where the current state of all accounts is stored
  - A consensus-based globally executed virtual machine

Source: Melanie Swan; https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial

Accounts and Wallets in Ethereum

Accounts:

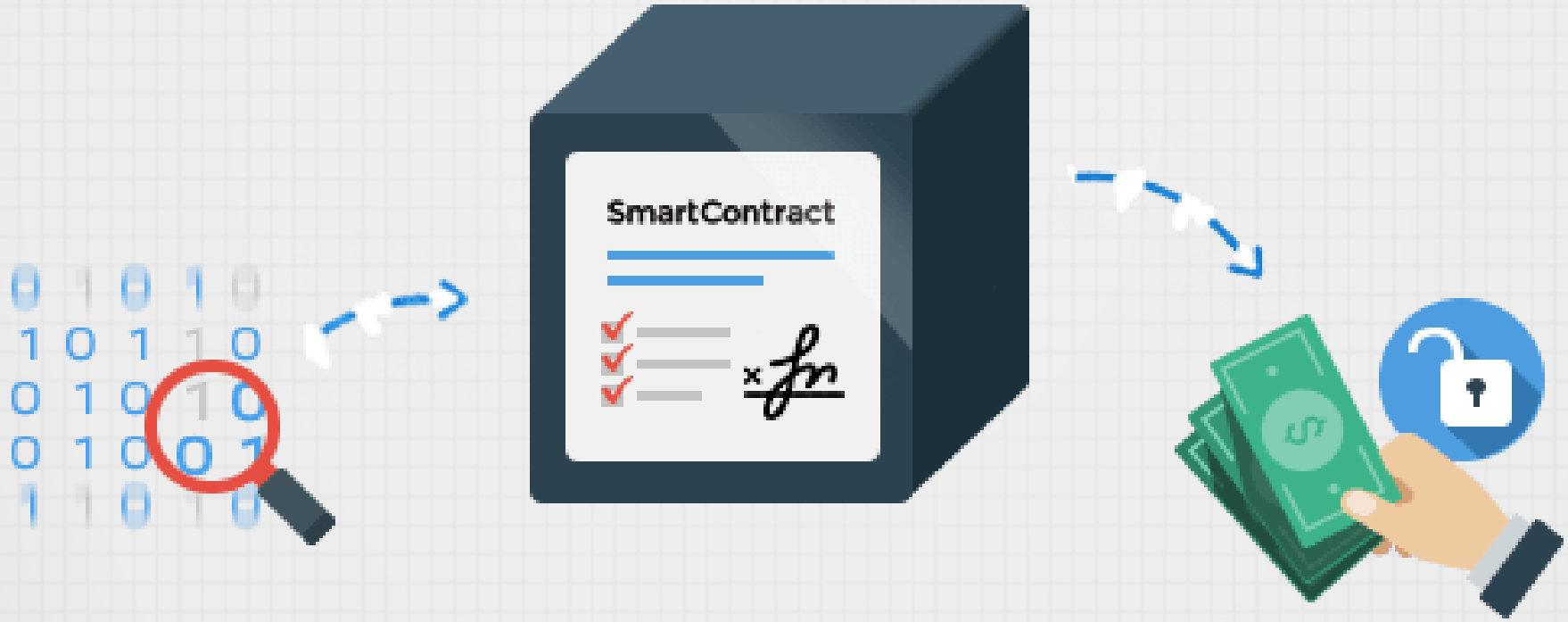
- Two Kinds:
  - External Owned Accounts - (EOA, most common account)
  - Contract Accounts
- Consist of a public/private keypair
- Allow for interaction with the blockchain

Wallets:

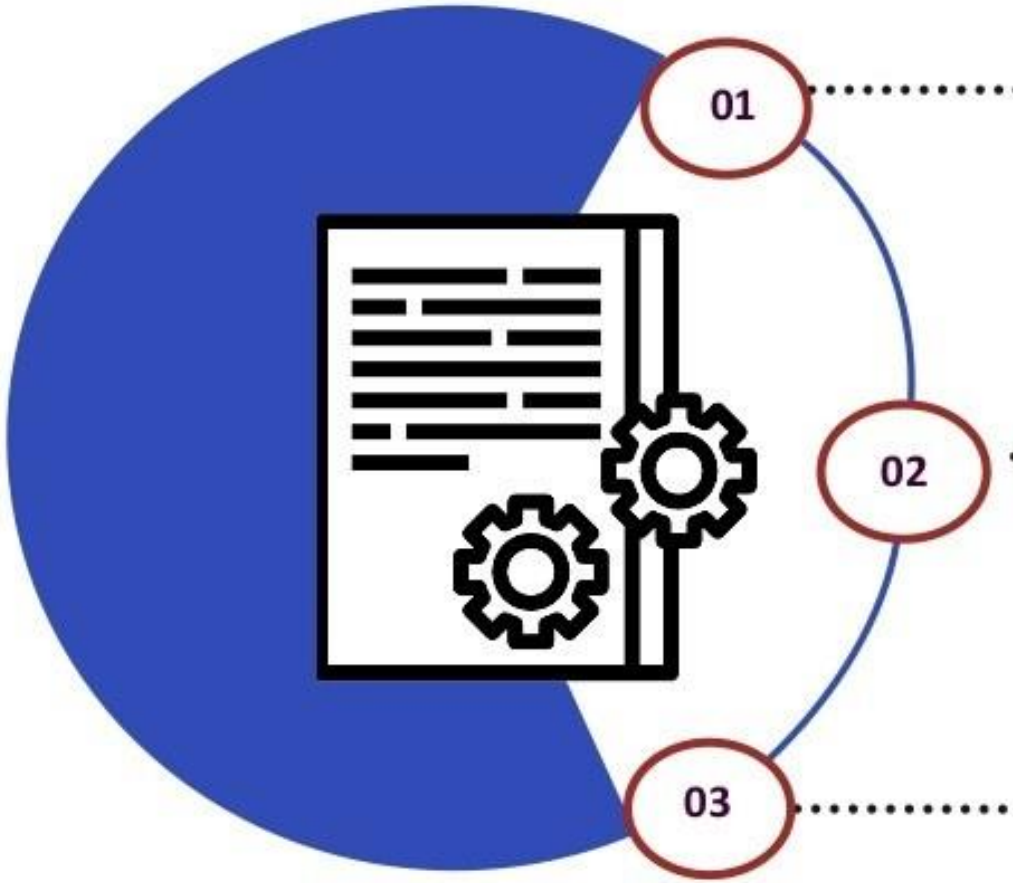
- A set of one or more external accounts
- Used to store/transfer ether

What is a Smart Contract

- Smart contract is a term used to describe computer program code that is capable of facilitating, executing, and enforcing the negotiation or performance of an agreement (i.e. contract) using blockchain technology.
- The entire process is automated can act as a complement, or substitute, for legal contracts, where the terms of the smart contract are recorded in a computer language as a set of instructions.



Key Properties of Smart Contracts



Autonomous
The creator does not have to participate in the process after deploying the contract.

Auto-Sufficient
Contracts are able to collect money, realize transactions, distribute resources, issue and spend funds to allow a larger capacity of storage and computation power.

Decentralized
Smart contracts are not focused on one central server but are distributed by various network points so they can be referred to as being *decentralized*.

Smart Contracts

- Executable code
- Turing Complete
- Function like an external account
  - Hold funds
  - Can interact with other accounts and smart contracts
  - Contain code
- Can be called through transactions

Code Execution

- Every node contains a virtual machine (similar to Java)
  - Called the Ethereum Virtual Machine (EVM)
  - **Compiles** code from high-level language to bytecode
  - Executes smart contract code and broadcasts state
- ***Every full-node on the blockchain processes every transaction and stores the entire state***

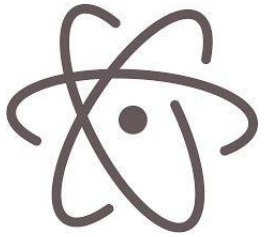
Gas Cost

- Gas Price: current market price of a unit of Gas (in Wei)
  - Check gas price here: https://ethgasstation.info/
  - Is always set before a transaction by user
- Gas Limit: maximum amount of Gas user is willing to spend
- Helps to regulate load on network
- Gas Cost (used when sending transactions) is calculated by gasLimit*gasPrice.
  - All blocks have a Gas Limit (maximum Gas each block can use)

Smart Contract Programming

- Solidity (javascript based), most popular
  - Not yet as functional as other, more mature, programming languages
- Serpent (python based)
- LLL (lisp based)

Smart Contract Programming



Atom Ethereum interface - Plugin for the Atom editor that features syntax highlighting, compilation and a runtime environment (requires backend node).

Atom Solidity Linter - Plugin for the Atom editor that provides Solidity linting.

Vim Solidity - Plugin for the Vim editor providing syntax highlighting.

Vim Syntastic - Plugin for the Vim editor providing compile checking.

B. Smart Contract Programming

- Solidity (javascript based), most popular
  - Not yet as functional as other, more mature, programming languages
- Serpent (python based)
- LLL (lisp based)

B. Smart Contract Programming

Solidity

Solidity is a language similar to JavaScript which allows you to develop contracts and compile to EVM bytecode. It is currently the flagship language of Ethereum and the most popular.

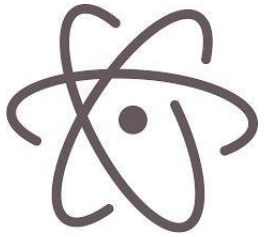
- Solidity Documentation - Solidity is the flagship Ethereum high level language that is used to write contracts.
- Solidity online realtime compiler

Serpent

Serpent is a language similar to Python which can be used to develop contracts and compile to EVM bytecode. It is intended to be maximally clean and simple, combining many of the efficiency benefits of a low-level language with ease-of-use in programming style, and at the same time adding special domain-specific features for contract programming. Serpent is compiled using LLL.

- Serpent on the ethereum wiki
- Serpent EVM compiler

B. Smart Contract Programming



Atom Ethereum interface - Plugin for the Atom editor that features syntax highlighting, compilation and a runtime environment (requires backend node).

Atom Solidity Linter - Plugin for the Atom editor that provides Solidity linting.

Vim Solidity - Plugin for the Vim editor providing syntax highlighting.

Vim Syntastic - Plugin for the Vim editor providing compile checking.

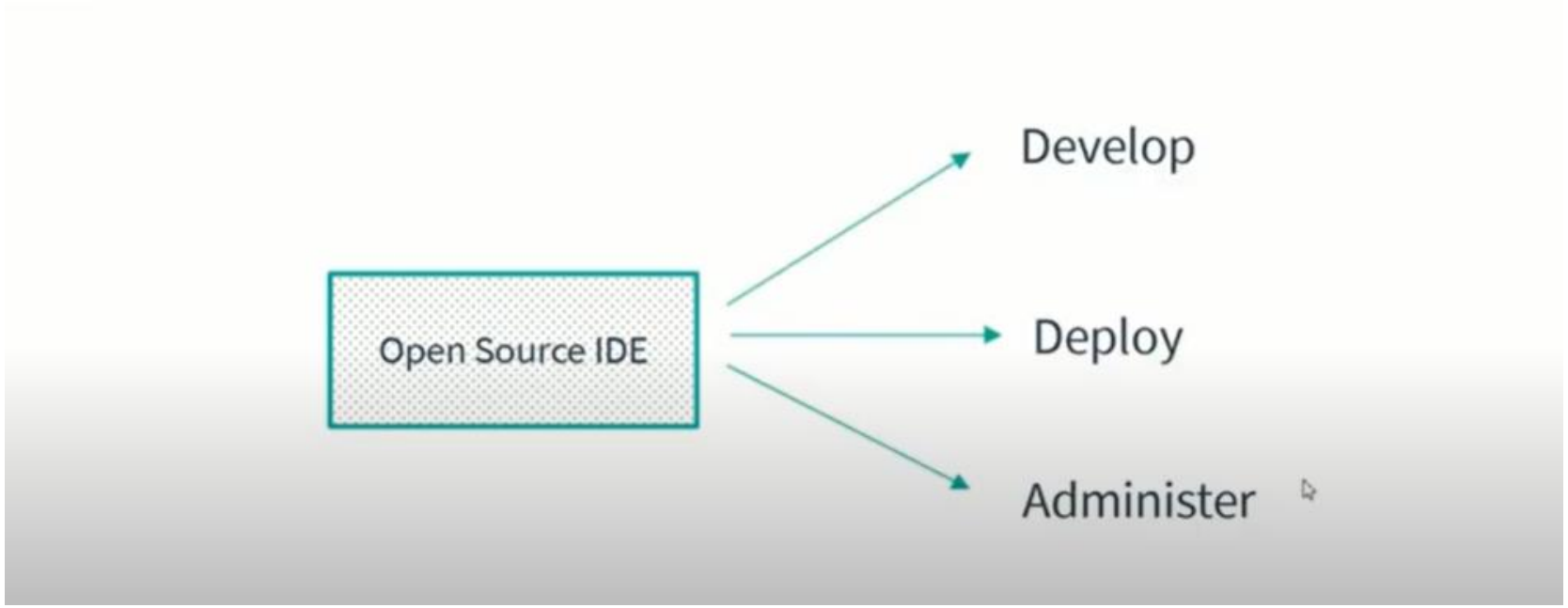
B. Smart Contract Programming: Solidity

```
contract Example {

   uint value;

   function setValue(uint pValue)
      {  value = pValue;
   }

   function getValue() returns (uint)
      {  return value;
   }

}
```

B. Smart Contract Programming: Solidity

```
var logIncrement =
    OtherExample.LogIncrement({sender: userAddress,
uint value});

logIncrement.watch(function(err, result) {
  // do something with result
})
```

Remix IDE

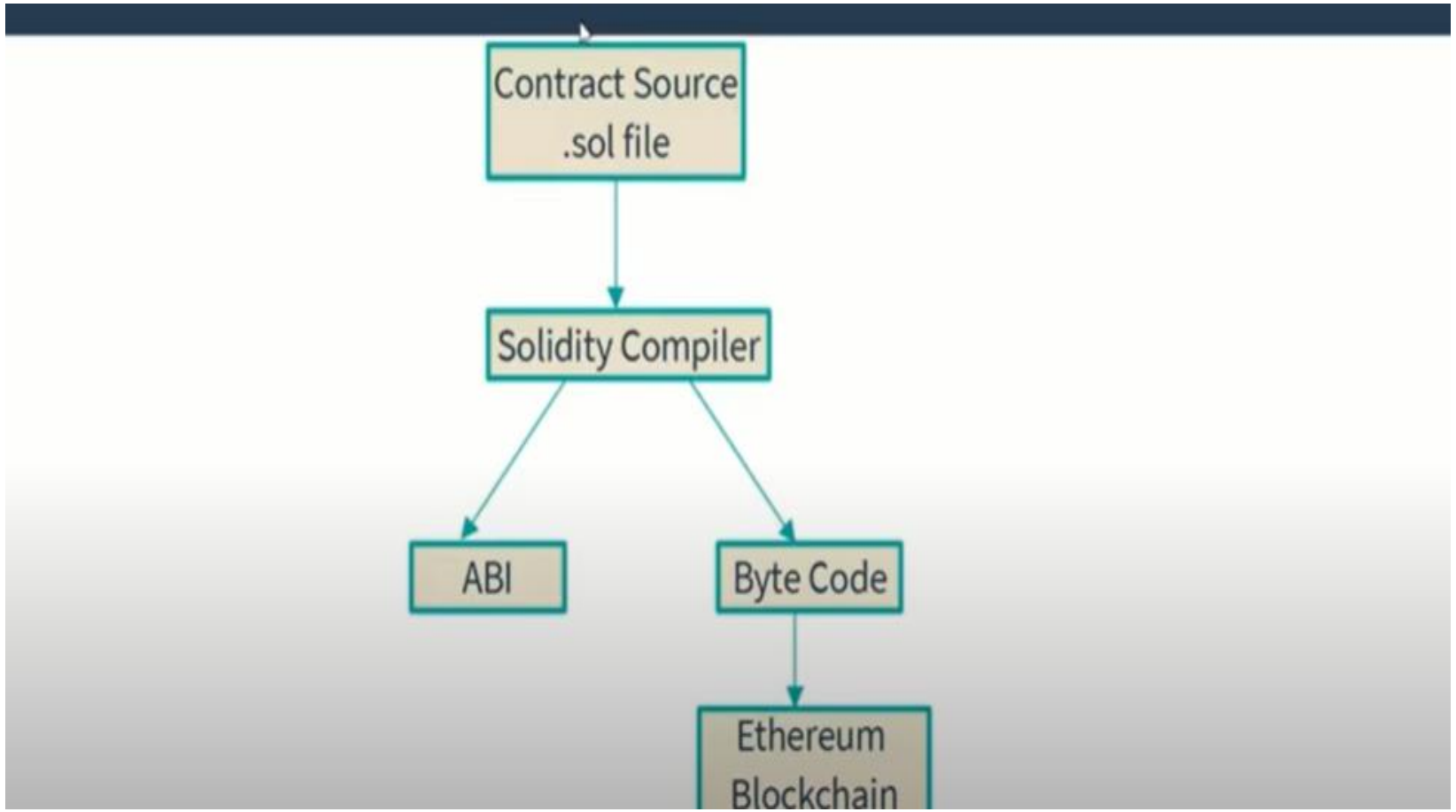


Remix IDE

Some important to know about Remix IDE-

- **Language Support -** Solidity and Vyper
- **Written in -** JavaScript
- **Modules -** Testing , Debugging ,Deploy

Smart contract compilation



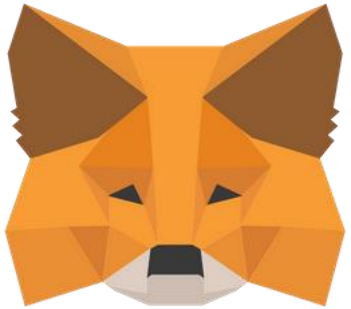
Smart contract compilation

- Contract bytecode is readable in public form
- Contract doesn't have to be public
- Bytecode is immutable
- ABI act as a bridge between application and smart contract
- ABI and bytecode can’t be generated without source code

Mainnet vs Testnet

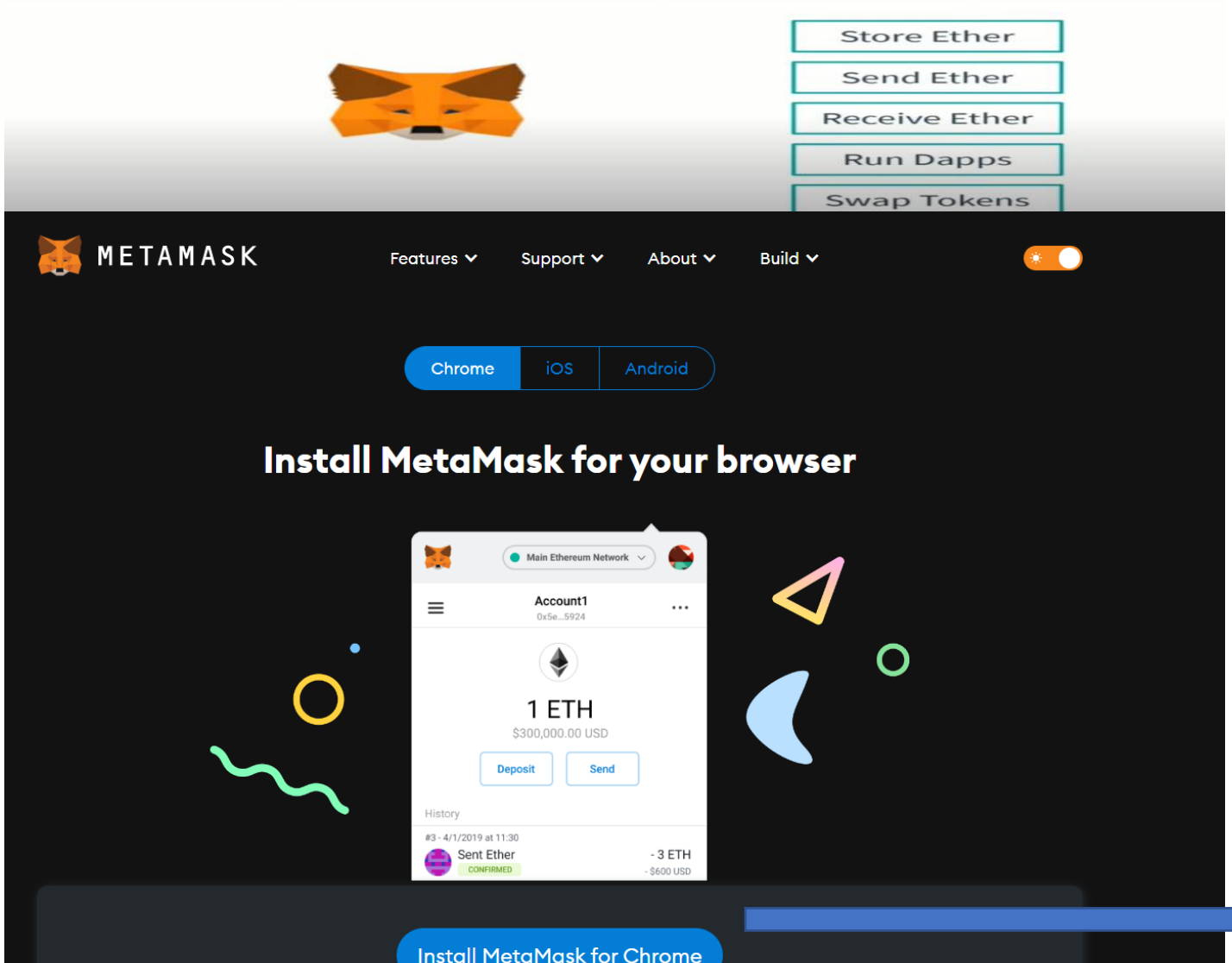
Mainnet	Testnet
Used for actual transaction with value	Used for testing smart contracts and decentralized applications
Mainnet's network ID is 1	Testnets have network IDs of 3, 4, and 42
Example- Ethereum	Example- Rinkeby test networks

Metamask



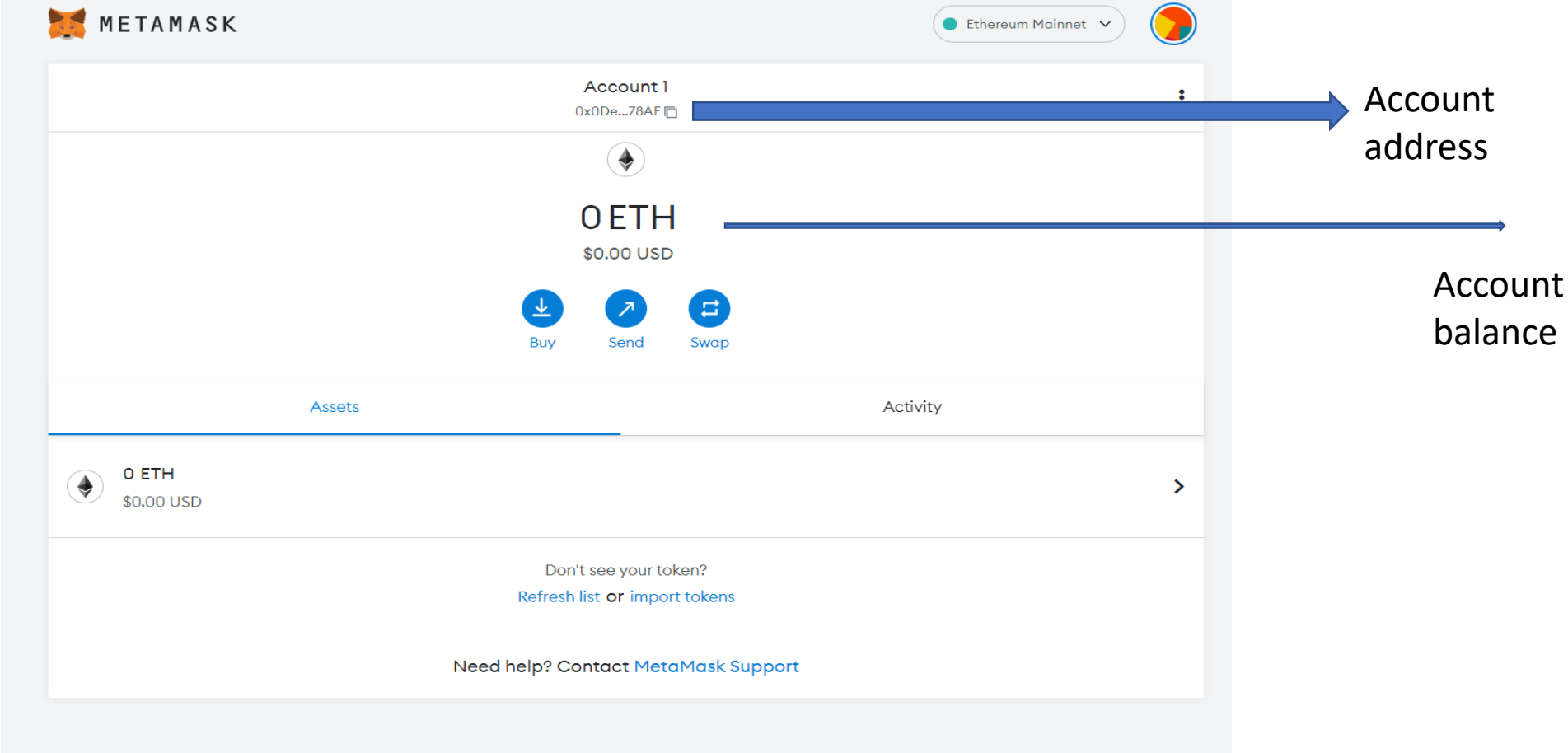
- MetaMask is a lightweight Chrome browser-based extension that helps in interacting with Ethereum networks.
- It is also a wallet that helps in sending and receiving Ether.
- It does not download the entire chaindata locally
- It is a function to inject a javascript library called web3.js into the namespace of each page your browser loads.

Metamask installation



Install from here

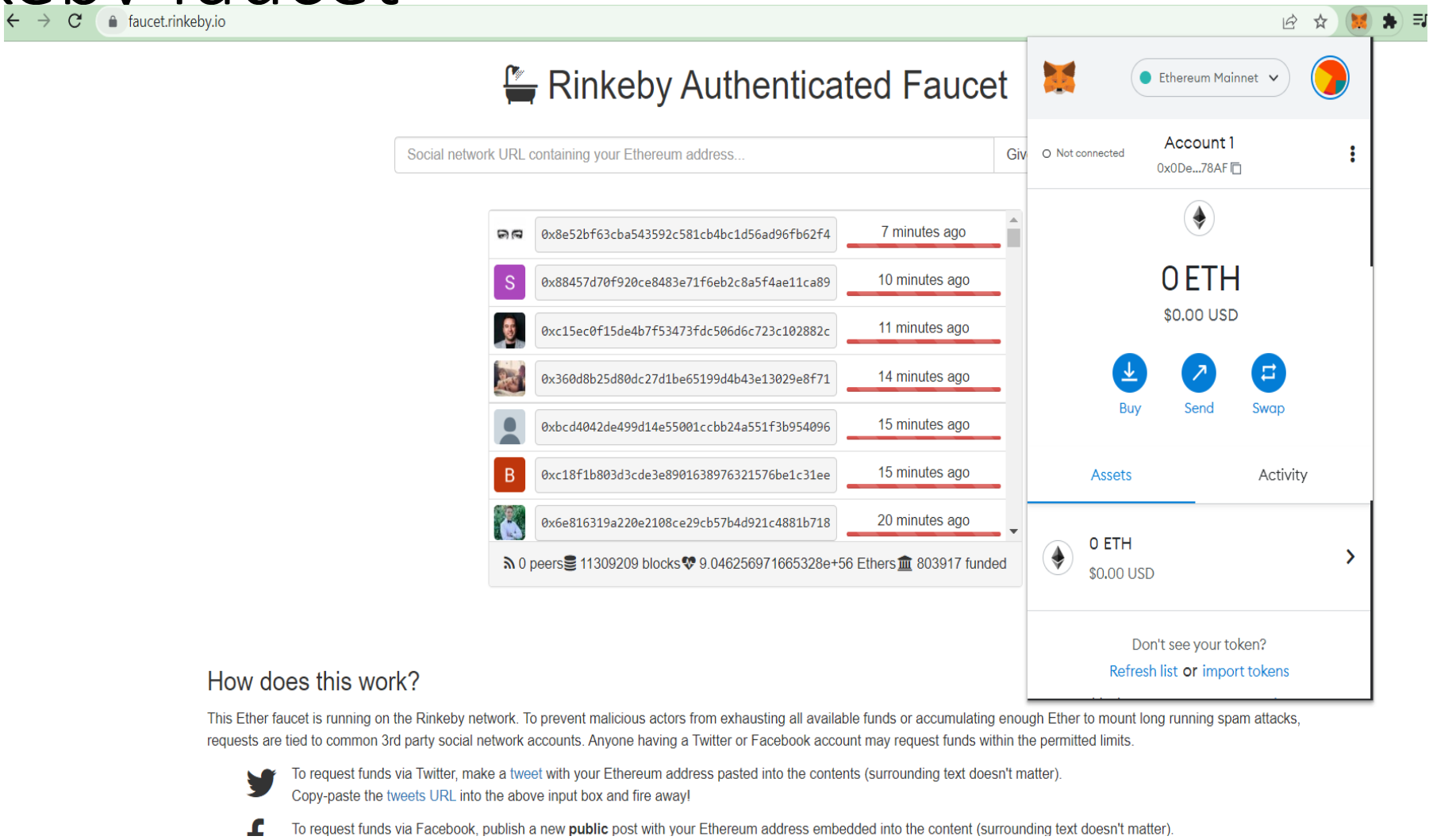
Metamask installation



Testnet (Rinkeby) faucet

- ETH on testnet has no real value; therefore we get testnet ETH from faucet
- Example, Rinkeby faucet, Ropsten faucet

Rinkeby faucet



About solidity

- High level statically typed programming language
- With solidity you can create contracts for uses such as voting, blind auction, crowdfunding, and multi-signature wallets
- Case sensitive
- Latest updates are available on solidity documentation

Variables in solidity

- A variable is basically a placeholder for data which can be manipulated at runtime.
- Synatax:

Syntax:

```
<type> <access modifier> <variable name> ;
```

Example:

```
int public int_var;
```

- A variable can be of three types

1. State variables
2. Local variables
3. Global variables

State variable in solidity

- Permanently stored in contract storage
- Not declared in functions
- Cost gas (expensive)
- Storage not dynamically allowed
- Instance of the contract cannot have other state variable besides these already declared

Local variable in solidity

- Variables whose values are present till function is executing
- Declared inside functions and are kept on stack, not on storage
- Don’t cost gas
- There are some types that reference the storage by default

Local vs. state variable

```
pragma solidity ^0.5.0;
contract SolidityTest {
   uint storedData;      // State variable
   constructor() public {
      storedData = 10;   // Using State variable
   }
}
```

```
pragma solidity ^0.5.0;
contract SolidityTest {
   uint storedData; // State variable
   constructor() public {
      storedData = 10;
   }
   function getResult() public view returns(uint){
      uint a = 1; // local variable
      uint b = 2;
      uint result = a + b;
      return result; //access the local variable
   }
}
```

Solidity variable names

- You should not use any of the Solidity reserved keywords as a variable name. These keywords are mentioned in the next section. For example, break or boolean variable names are not valid.
- Solidity variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, 123test is an invalid variable name but _123test is a valid one
- Solidity variable names are case-sensitive. For example, Name and name are two different variables.

Solidity-Access modifiers

Access Modifiers are the keywords used to specify the declared accessibility of a function or a type. There are four access modifiers available in Solidity.

- **Public** – Public state variables can be accessed internally as well as via messages. For a public state variable, an automatic getter function is generated.
- **Internal** – Internal state variables can be accessed only internally from the current contract or contract deriving from it without using this.
- **Private** – Private state variables can be accessed only internally from the current contract they are defined not in the derived contract from it.
- **External -** The External element can’t be inherited but it can be accessed by external elements. Current contract instance can’t access external element, it can be accessed externally only.

Example

```
pragma solidity ^0.5.0;
contract C {
   uint public data = 30;
   uint internal iData= 10;

   function x() public returns (uint) {
      data = 3; // internal access
      return data;
   }
}
contract Caller {
   C c = new C();
   function f() public view returns (uint) {
      return c.data(); //external access
   }
}
contract D is C {
   function y() public returns (uint) {
      iData = 3; // internal access
      return iData;
   }
   function getResult() public view returns(uint){
      uint a = 1; // local variable
      uint b = 2;
      uint result = a + b;
      return storedData; //access the state variable
   }
}
```

Functions in solidity

- A function is a group of reusable code which can be called anywhere in your program
- Syntax:

```
function function-name(parameter-list) scope returns() {
   //statements
}
```

- Example:

```
pragma solidity ^0.5.0;

contract Test {
   function getResult() public view returns(uint){
      uint a = 1; // local variable
      uint b = 2;
      uint result = a + b;
      return result;
   }
}
```

Solidity- View vs Pure functions

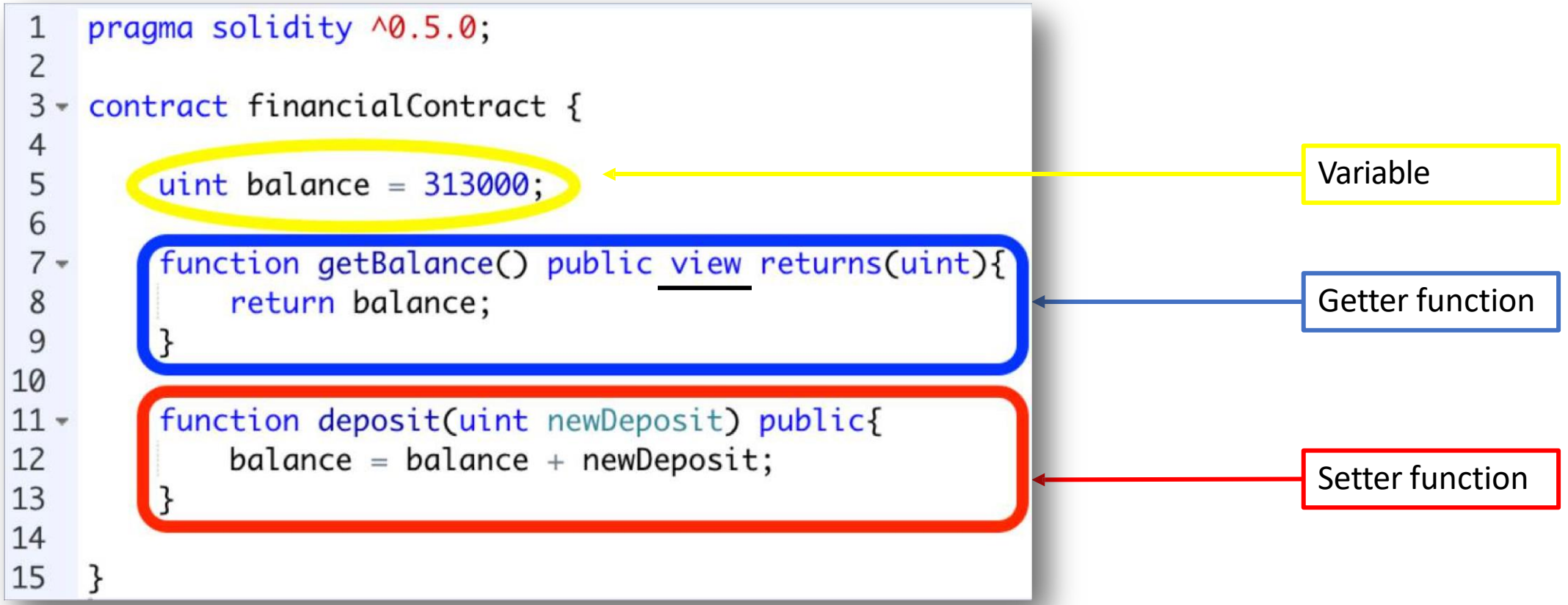
- **View** function can be declared view in which case they promise not to modify the state. they can view the state variable but can't modify it

- **Pure** function declares that no state variable will be changed or read

- Both of these don't cost any gas to call if they're called externally from outside the contract (but they do cost gas if called internally by another function).

Functions-Start Coding

- Setter and Getter: Set and get the information.



Compile the Contract

- Compile tab: Start to compile button

Set Deployment Parameters (1/2)

- Run tab: Environment = JavaScript VM

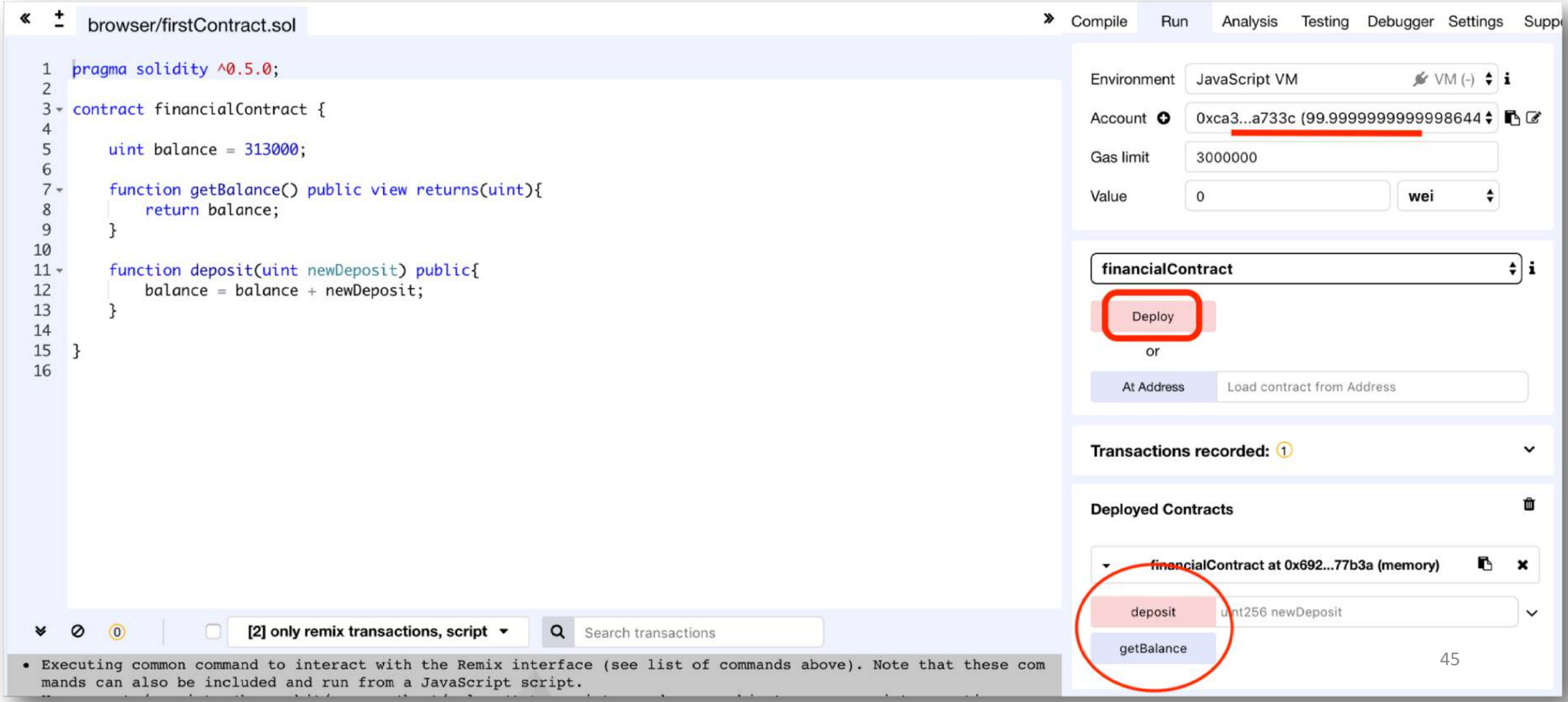
Set Deployment Parameters (2/2)

- 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).

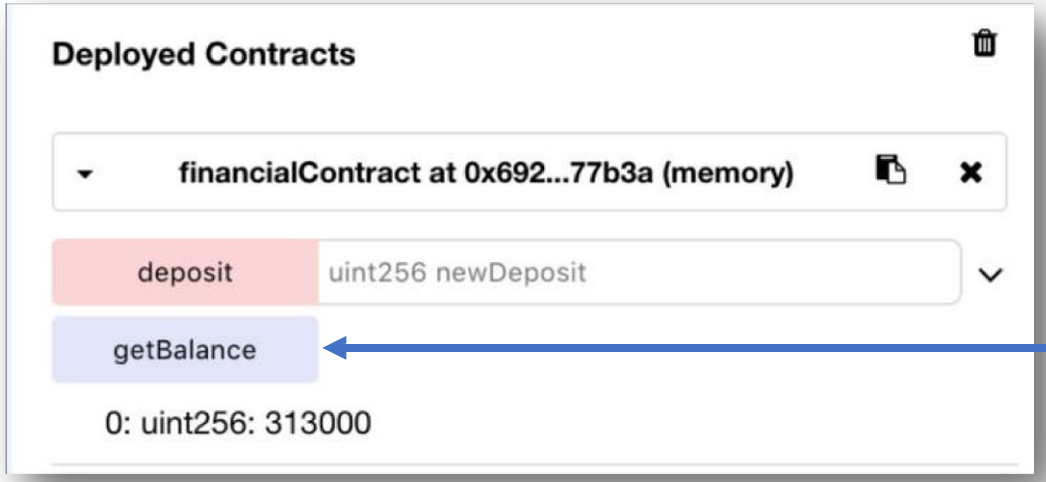
Deploy the Contract on the Private Blockchain of Remix

- Run tab: Deploy button

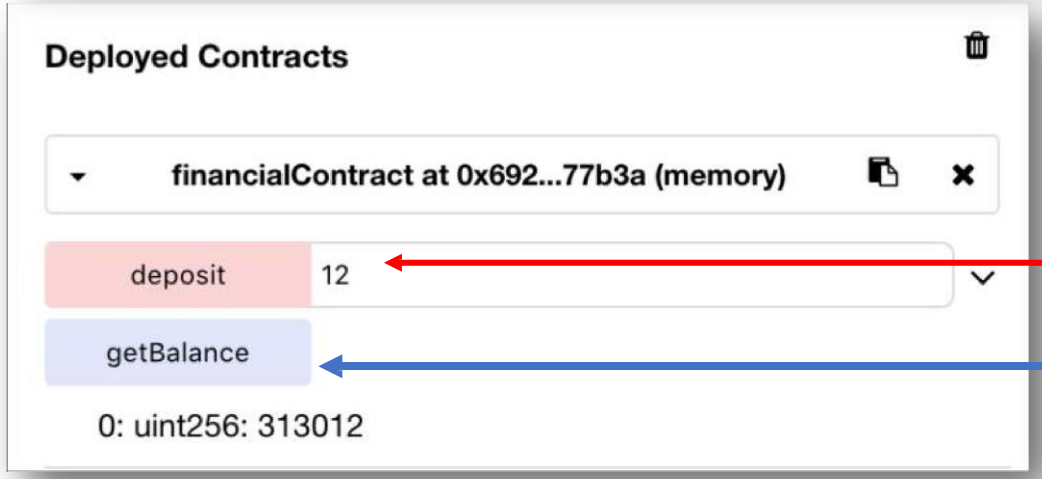


Interact with the Contract

- 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

Constructor function in solidity

- Executed only once
- You can create only one constructor and this is optimal
- A default constructor is created by the compiler if there is no explicit defined constructor
- A constructor can be either public or internal
- After a constructor code executed, the final code is deployed to blockchain.

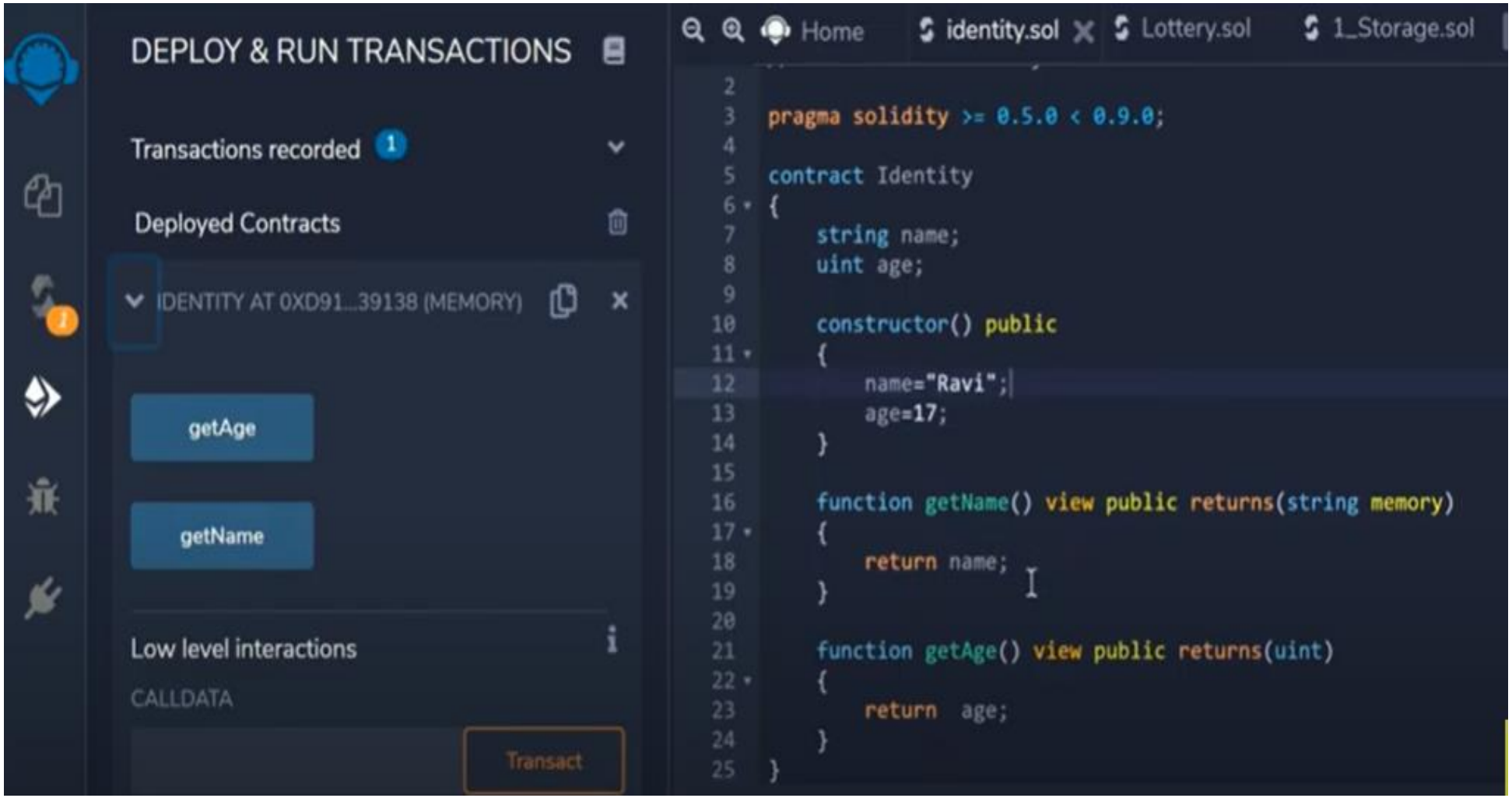
Constructor

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

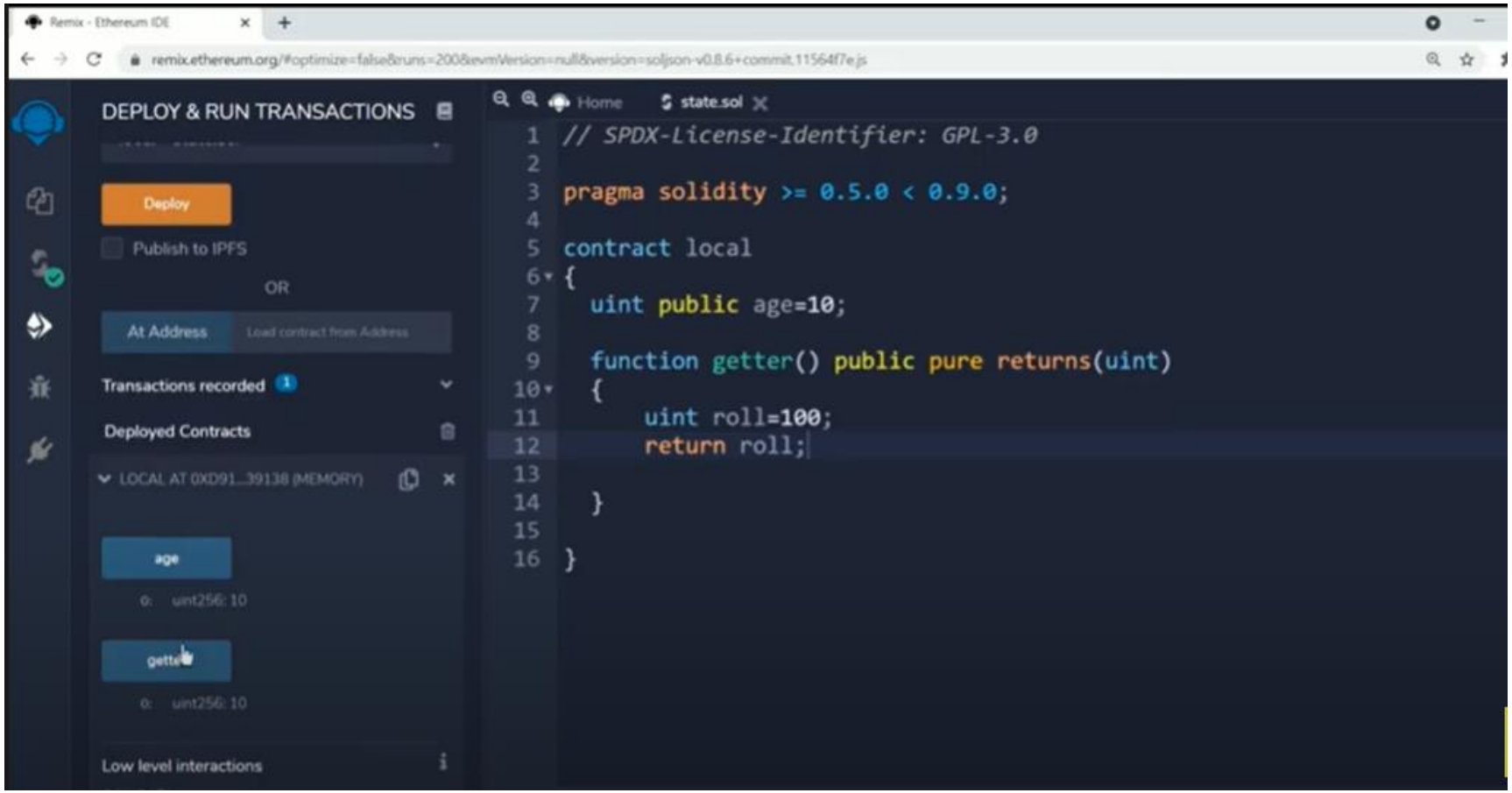
```
1  pragma solidity ^0.5.0;
2
3  contract financialContract {
4
5      address owner;
6
7      constructor() public{
8          owner = msg.sender;
9      }
10
11     function receiveDeposit() payable public{
12
13     }
14
15     function getBalance() public view returns(uint){
16         return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

We want to save the address of the contract creator

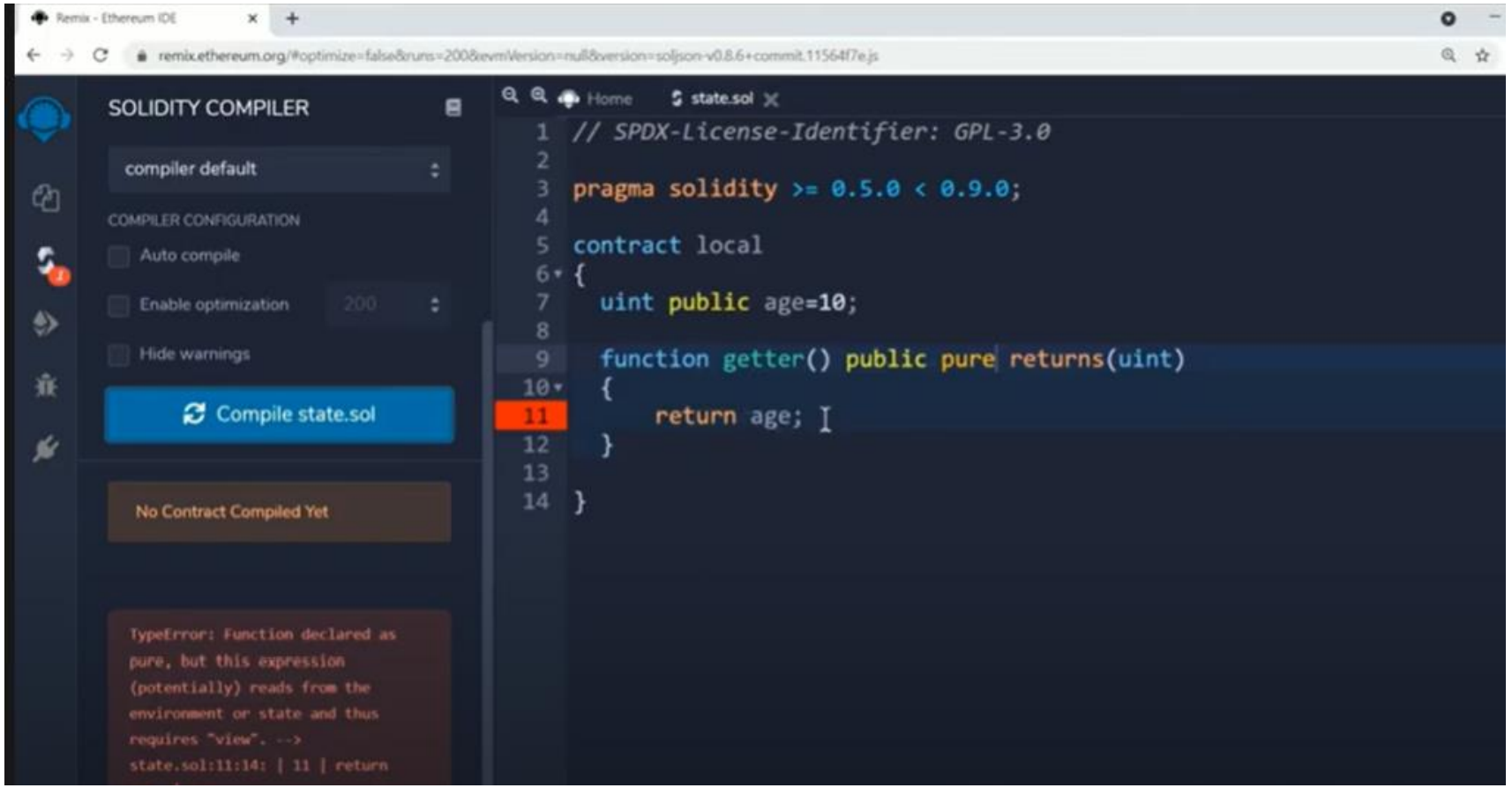
A sample program



Solidity- View vs Pure functions Example

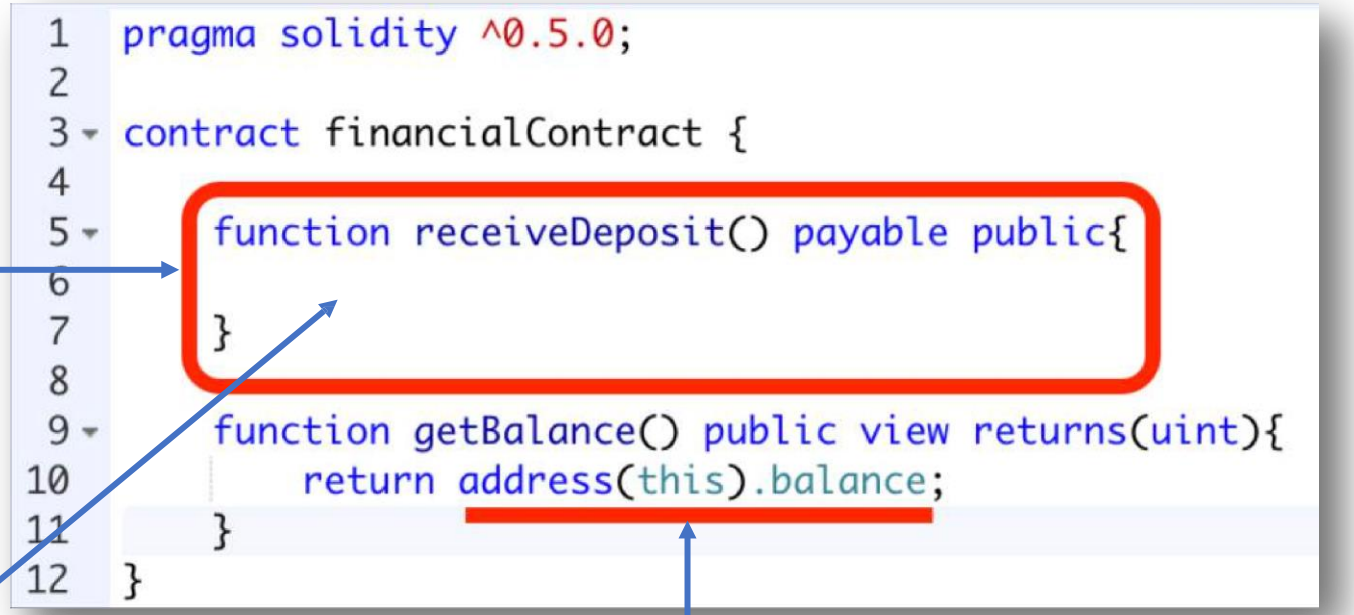


Solidity- View vs Pure functions Example



Receive ether (Payable function)

- Transfer money to the contract



```
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 }
```

Payable keyword allows receiving ether

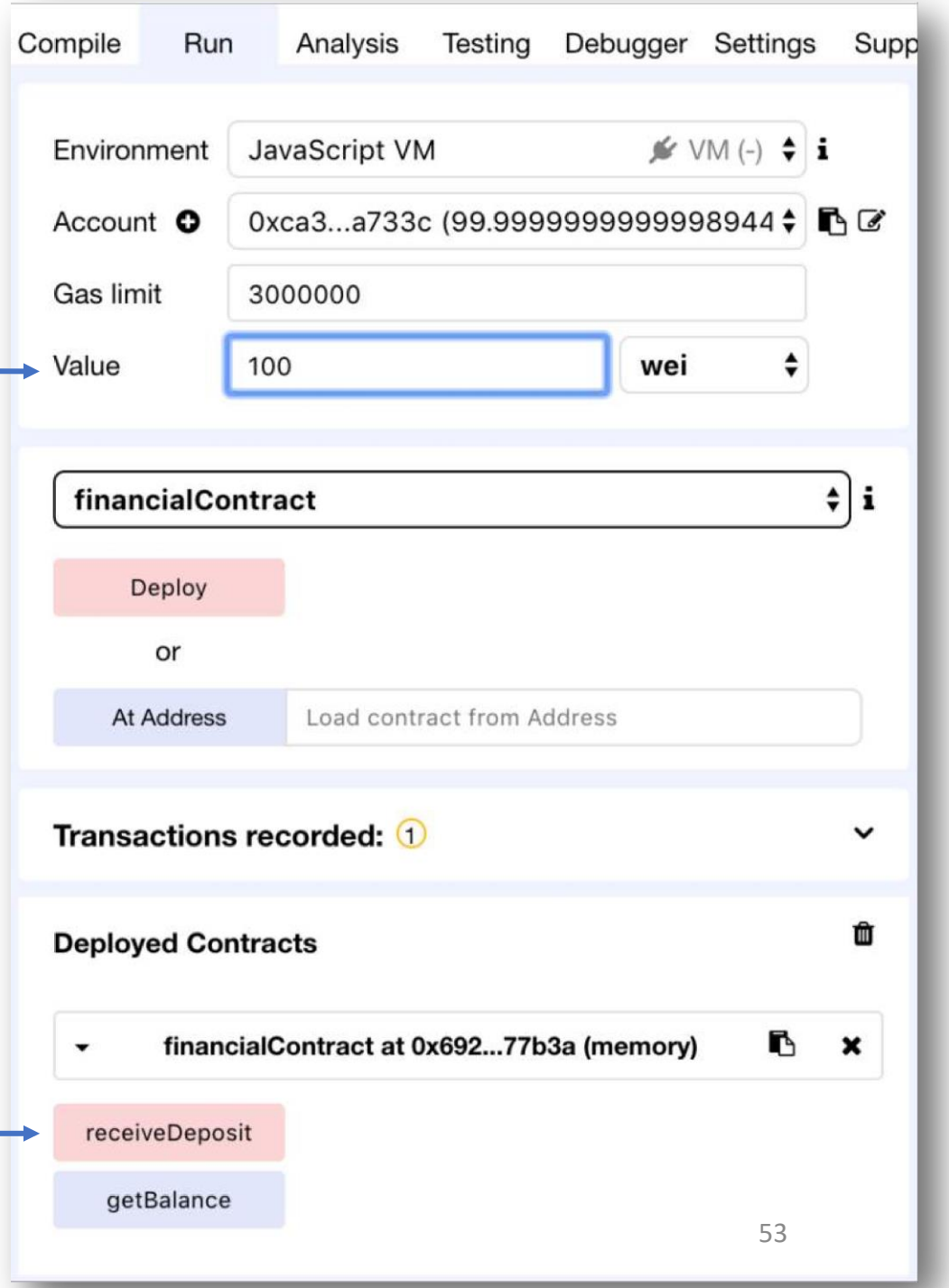
Hidden Code:
Address(this).balance += msg.value;

We can get the balance of the contract

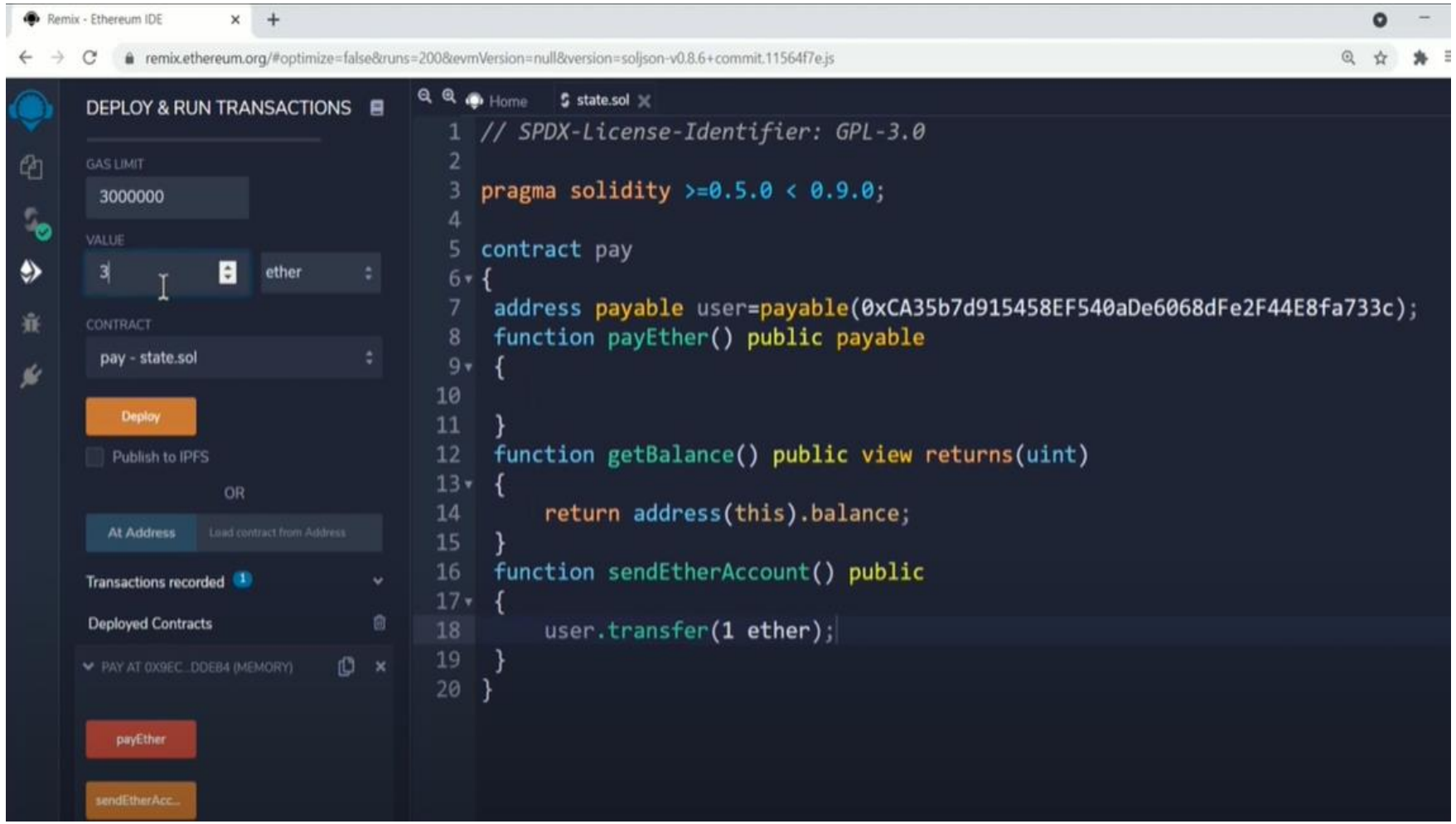
Receive ether

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

2. Click the receiveDeposit button to transfer the money to the contract



Contract balance update



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

Mappings in solidity

- Mappings in solidity acts like a hash table or dictionary in other languages
- Store data in form of a key value pair
- Mostly used to associate the Ethereum address with associated value type
- Syntax:

```
mapping(key => value) <access specifier> <name>;
```

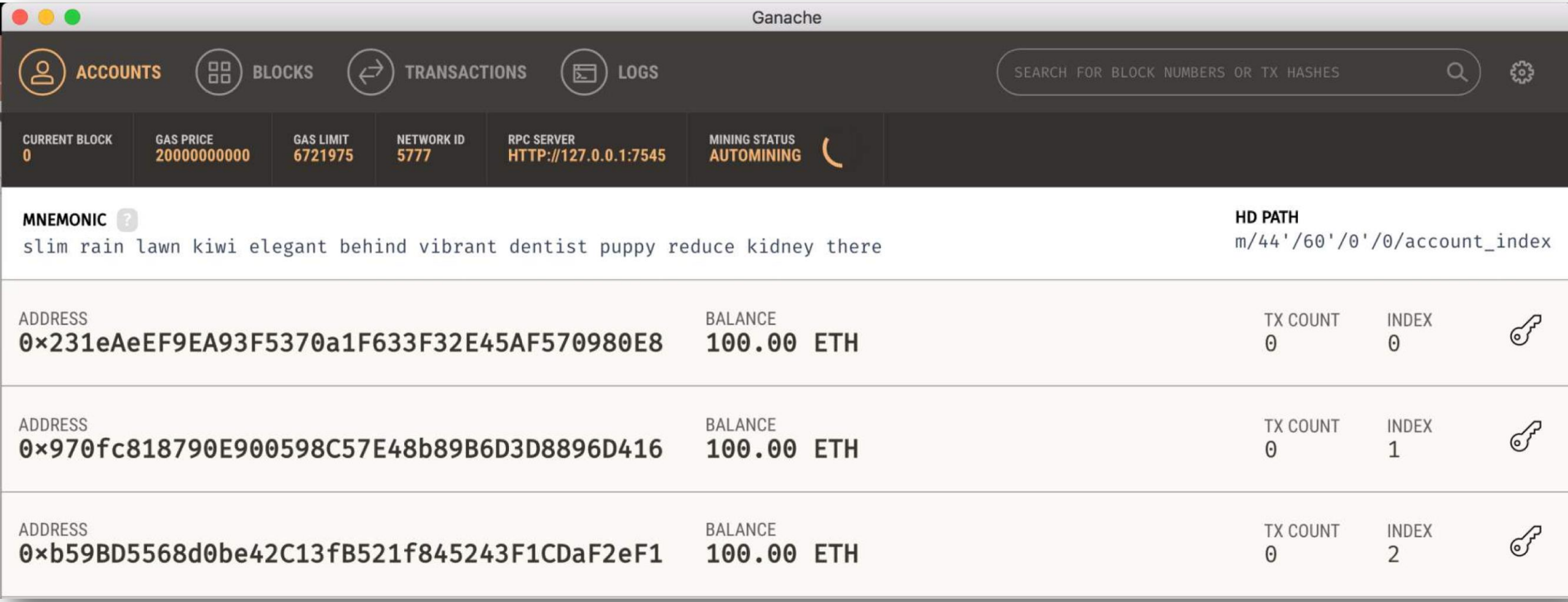
Example-Mappings

```
// Solidity program to
// demonstrate mapping
pragma solidity ^0.4.18;

// Defining contract
contract mapping_example {

    //Defining structure
    struct student
    {
        // Declaring different
        // structure elements
        string name;
        string subject;
        uint8 marks;
    }

    // Creating a mapping
    mapping (
    address => student) result;
    address[] public student_result;
}
```

Now deploying a smart contract on an external blockchain

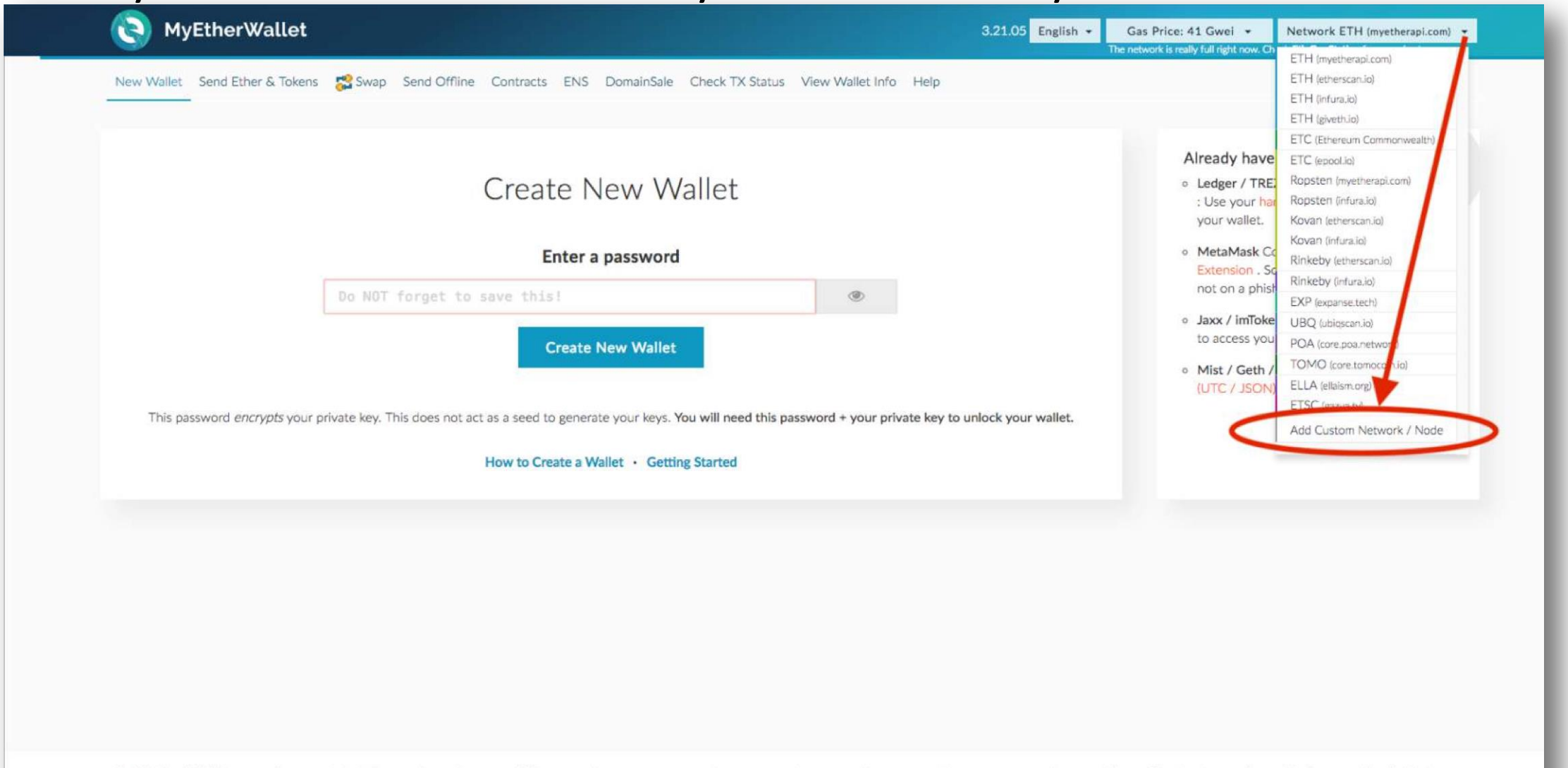
	Tools / Activities	Remix	Ganache	MyEtherWallet	Geth
1	Configuring the Blockchain	-	-	-	+
2	Deploying the Blockchain	Not Persistent	+	-	+
3	Developing the contract	+	-	-	+
4	Compiling the contract	+	-	-	+
5	Creating user account	+	+	+	+
6	Deploying the contract	+	-	+	+
7	Creating the UI for interacting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

Run Ganache

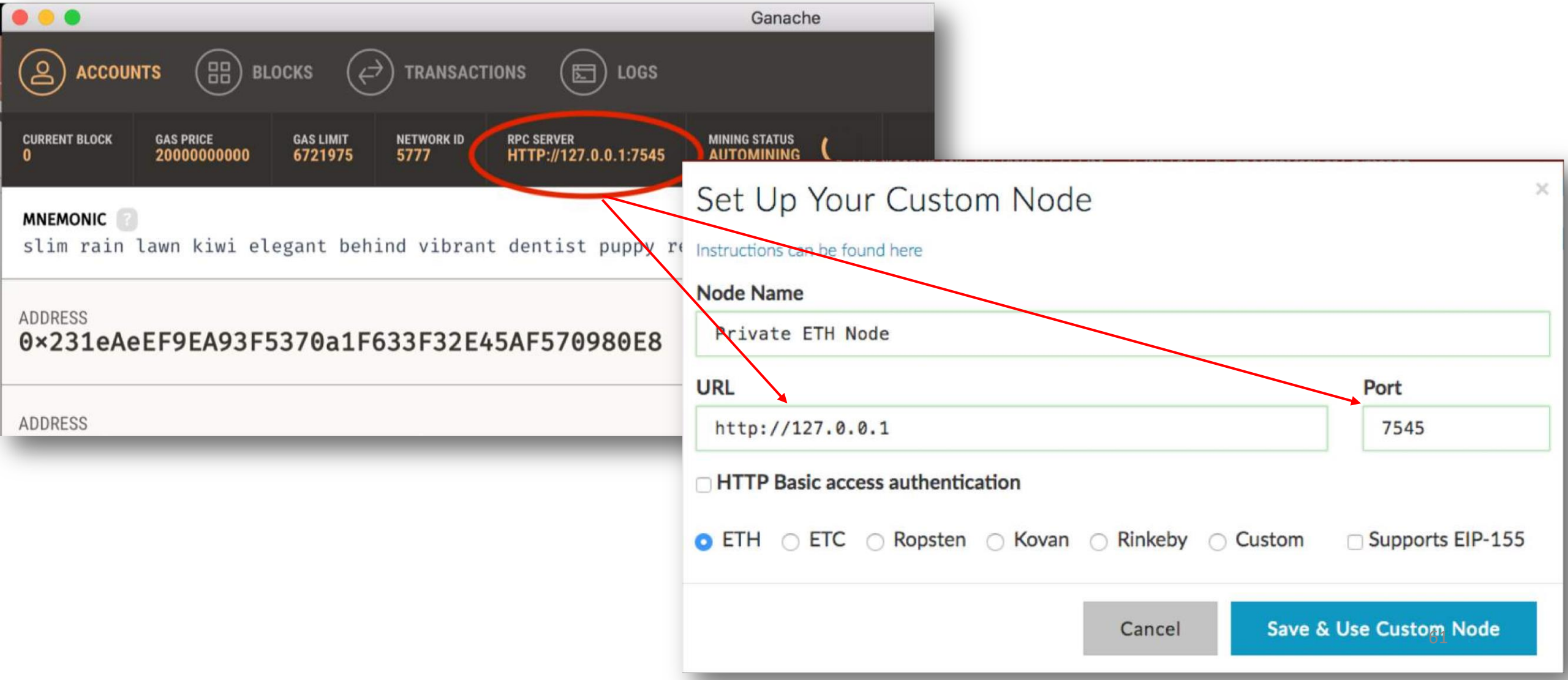

Ganache
ACCOUNTS
BLOCKS
TRANSACTIONS
LOGS
SEARCH FOR BLOCK NUMBERS OR TX HASHES
CURRENT BLOCK
0
GAS PRICE
20000000000
GAS LIMIT
6721975
NETWORK ID
5777
RPC SERVER
HTTP://127.0.0.1:7545
MINING STATUS
AUTOMINING
MNEMONIC
slim rain lawn kiwi elegant behind vibrant dentist puppy reduce kidney there
HD PATH
m/44'/60'/0'/0/account_index
ADDRESS
0×231eAeEF9EA93F5370a1F633F32E45AF570980E8
BALANCE
100.00 ETH
TX COUNT
0
INDEX
0
ADDRESS
0×970fc818790E900598C57E48b89B6D3D8896D416
BALANCE
100.00 ETH
TX COUNT
0
INDEX
1
ADDRESS
0×b59BD5568d0be42C13fB521f845243F1CDaF2eF1
BALANCE
100.00 ETH
TX COUNT
0
INDEX
2

MyEtherWallet

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

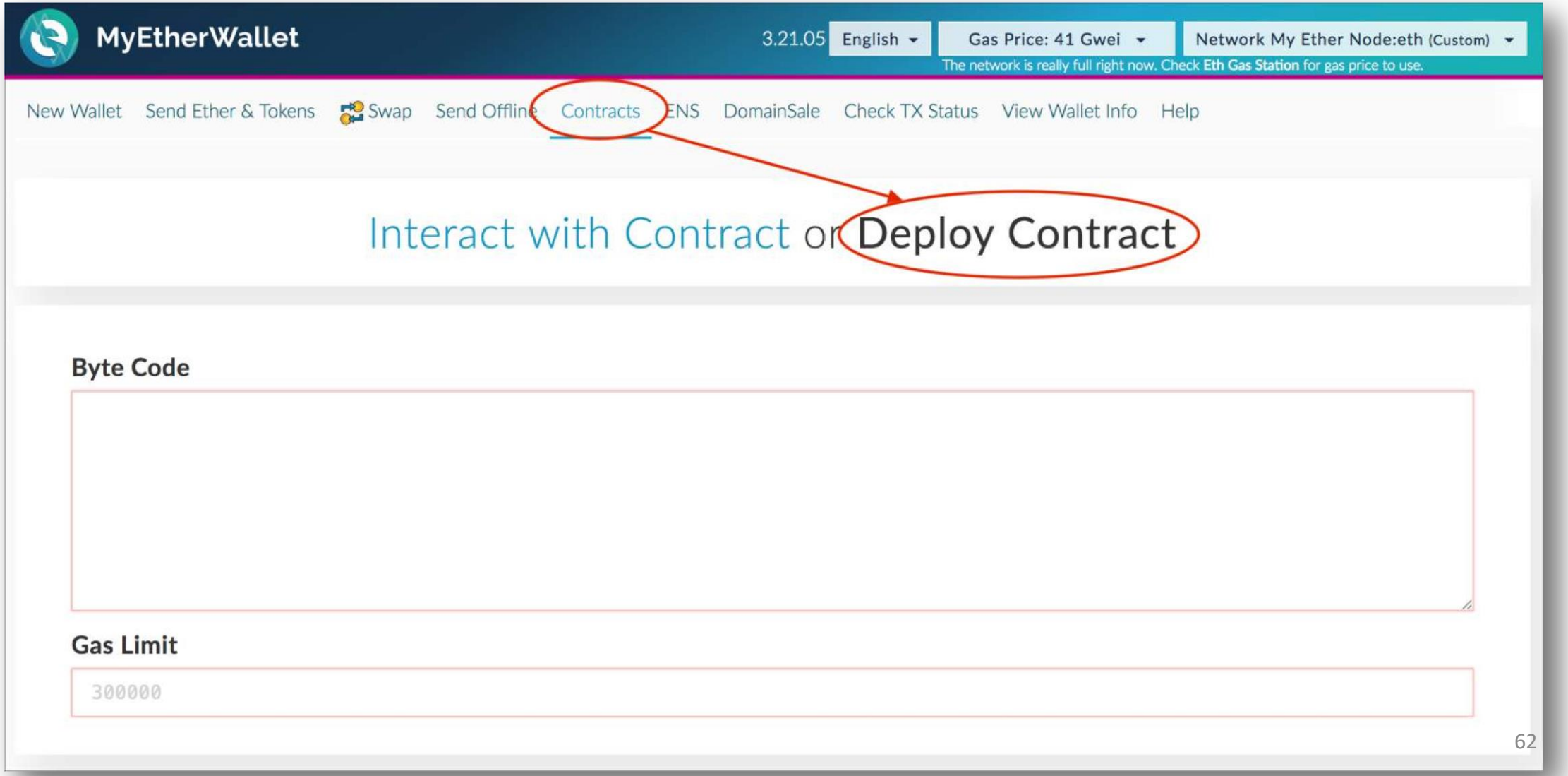


Import your RPC server address and the port number from Ganache to MyEtherWallet



MyEtherWallet

- Contracts tab: Deploy Contract

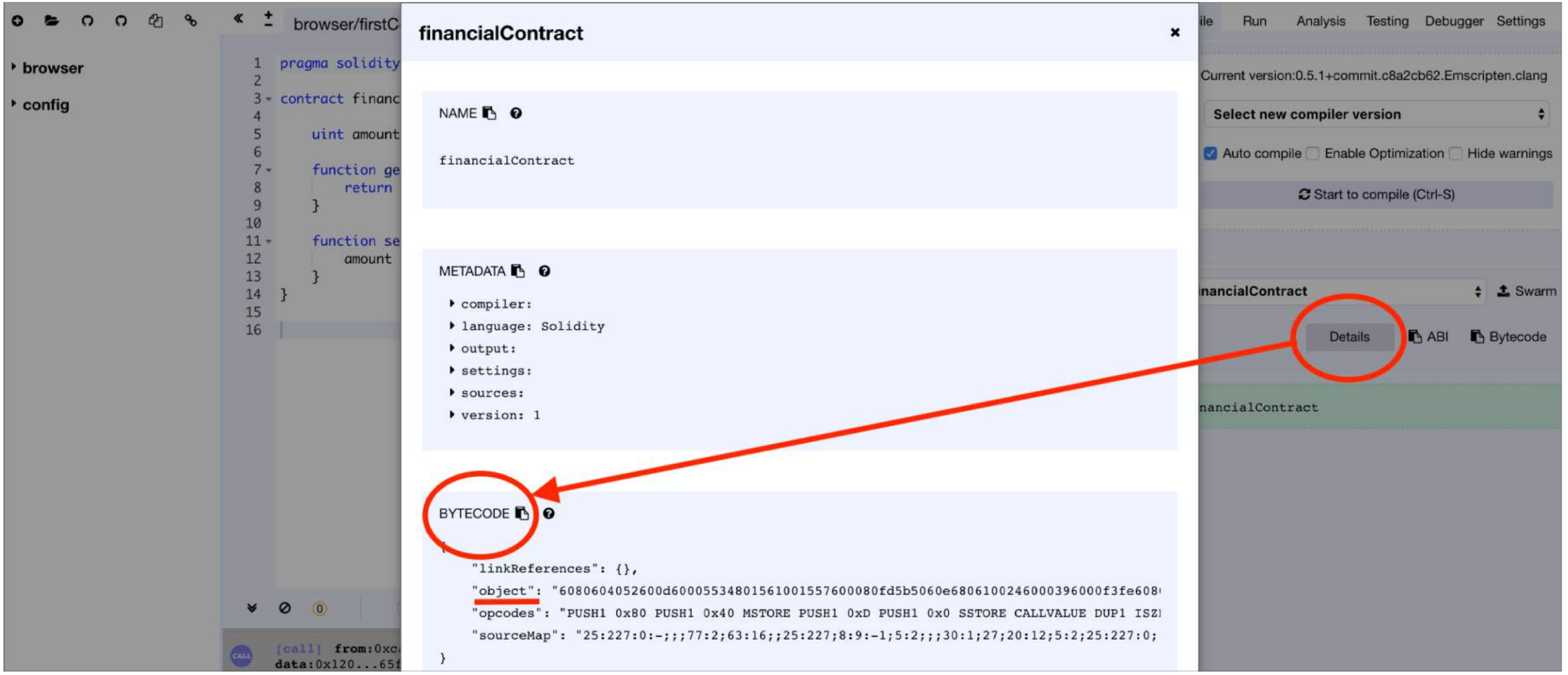


Remix

- Type your contract and compile it

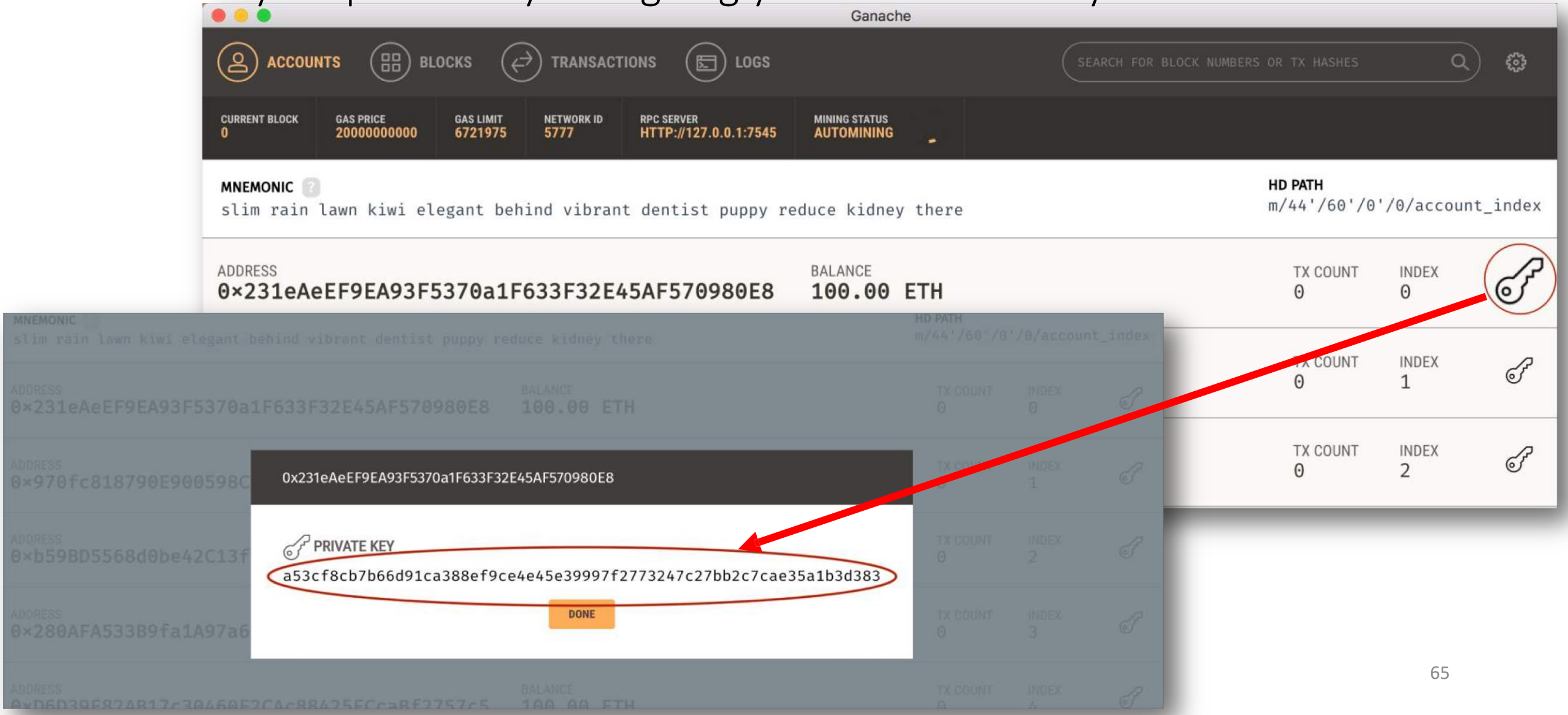
Remix

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



Ganache

Access your private key for signing your contract in MyEtherWallet.



MyEtherWallet

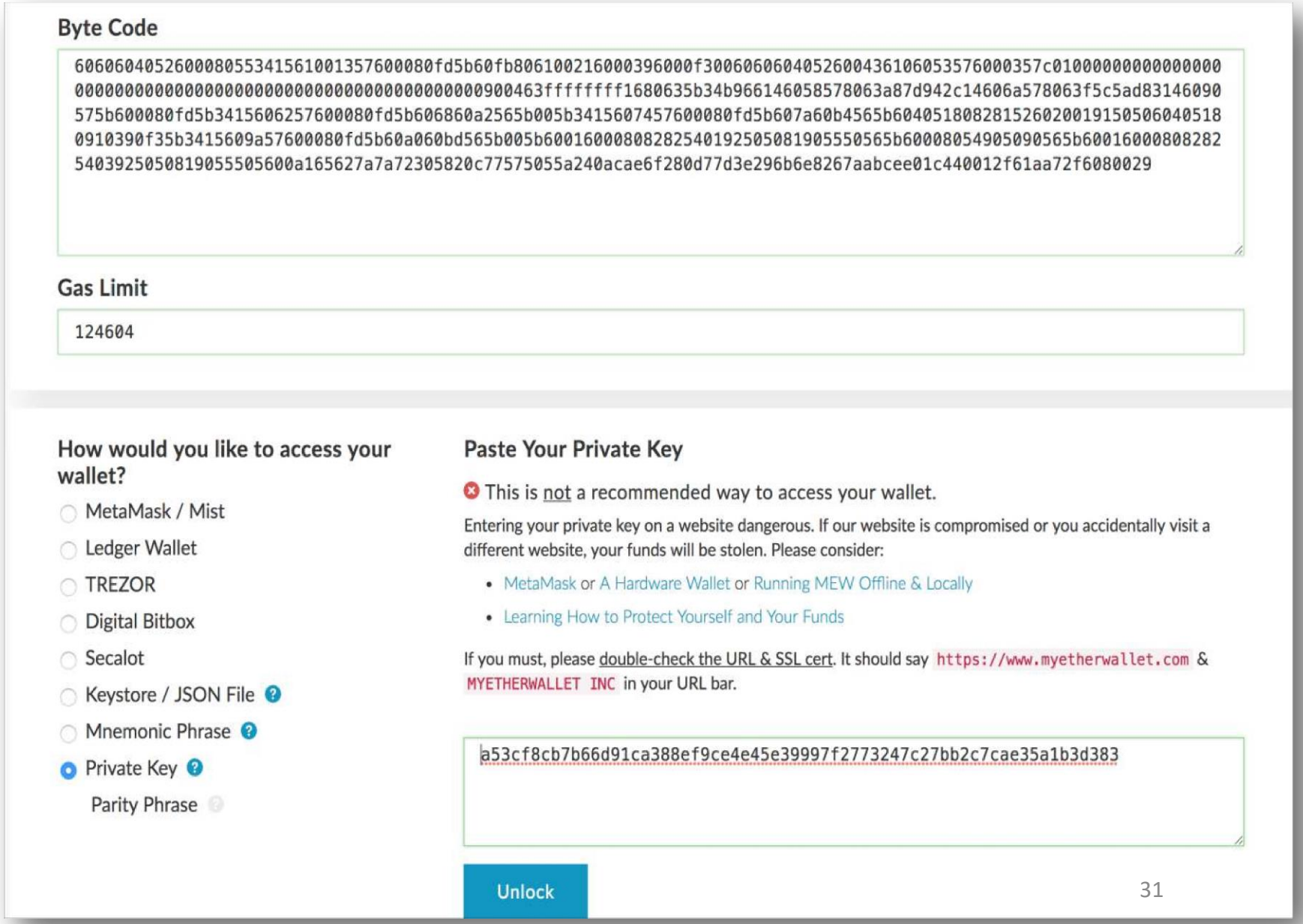
1.Paste the contract's ByteCode from Remix

2.Gas Limit will automatically be calculated

3.Paste your private key from Ganache

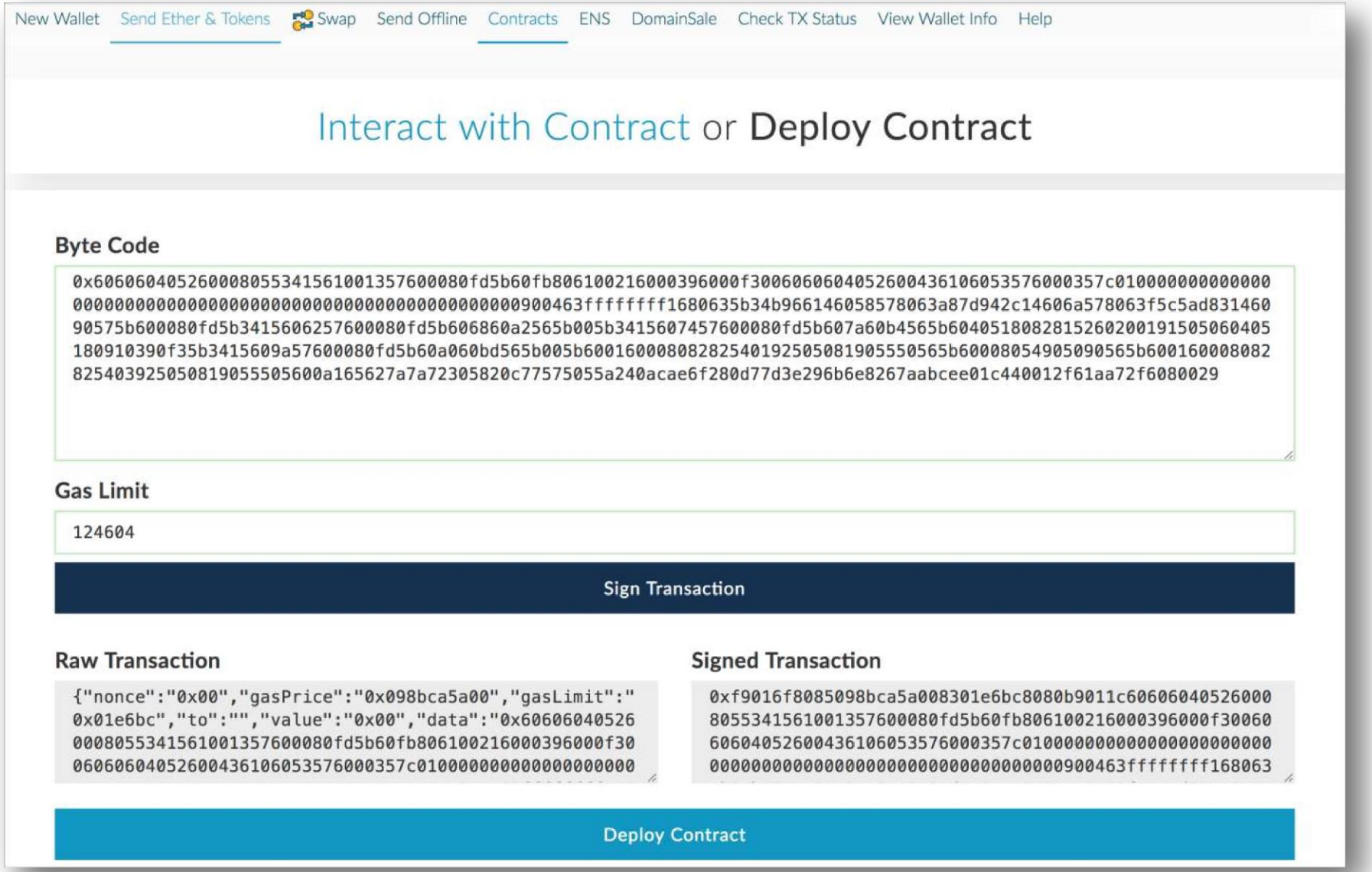
4. Click Unlock

5.Now you have access to your wallet



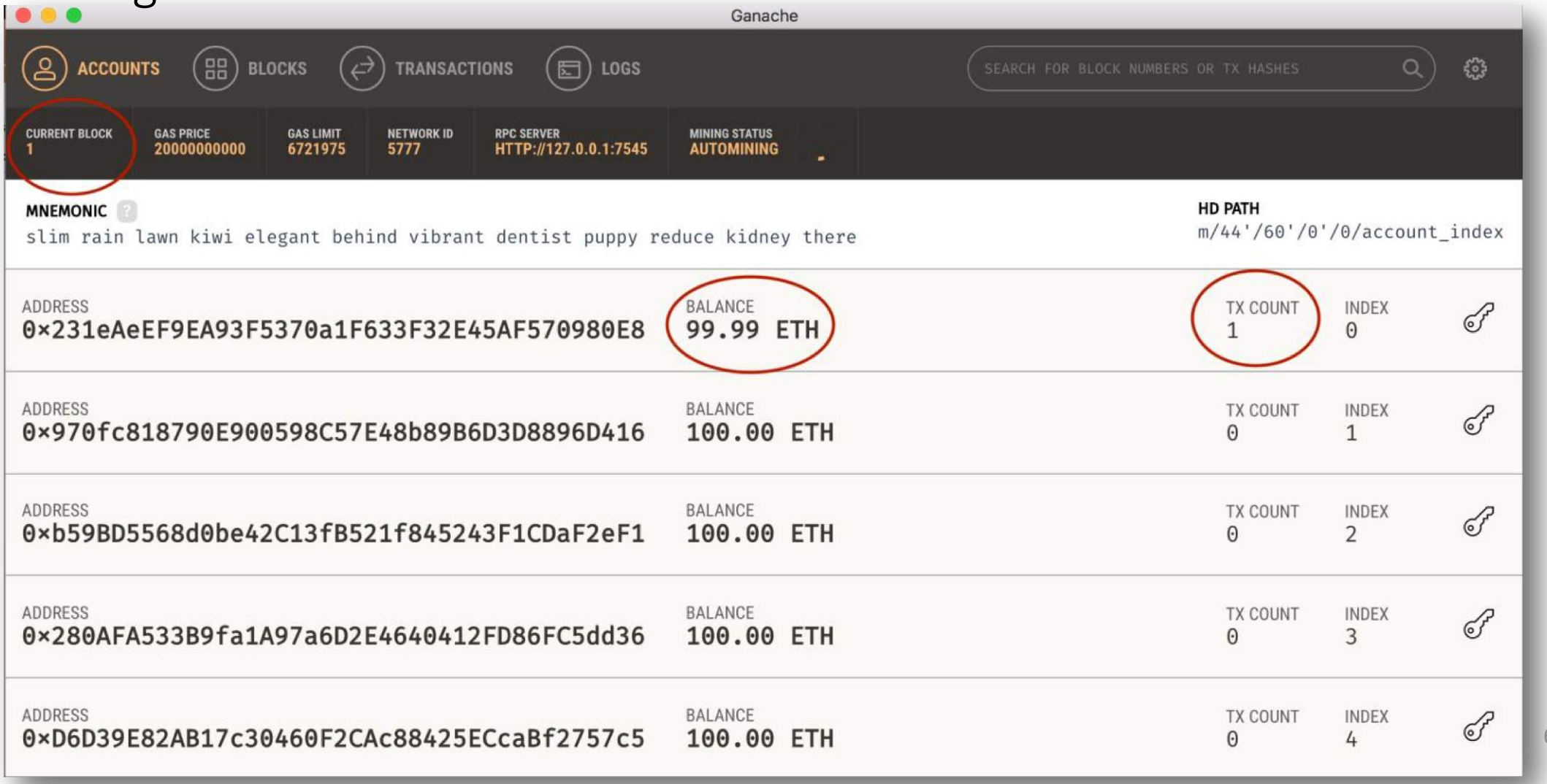
MyEtherWallet

Click on *Sign Transaction* button to deploy your contract

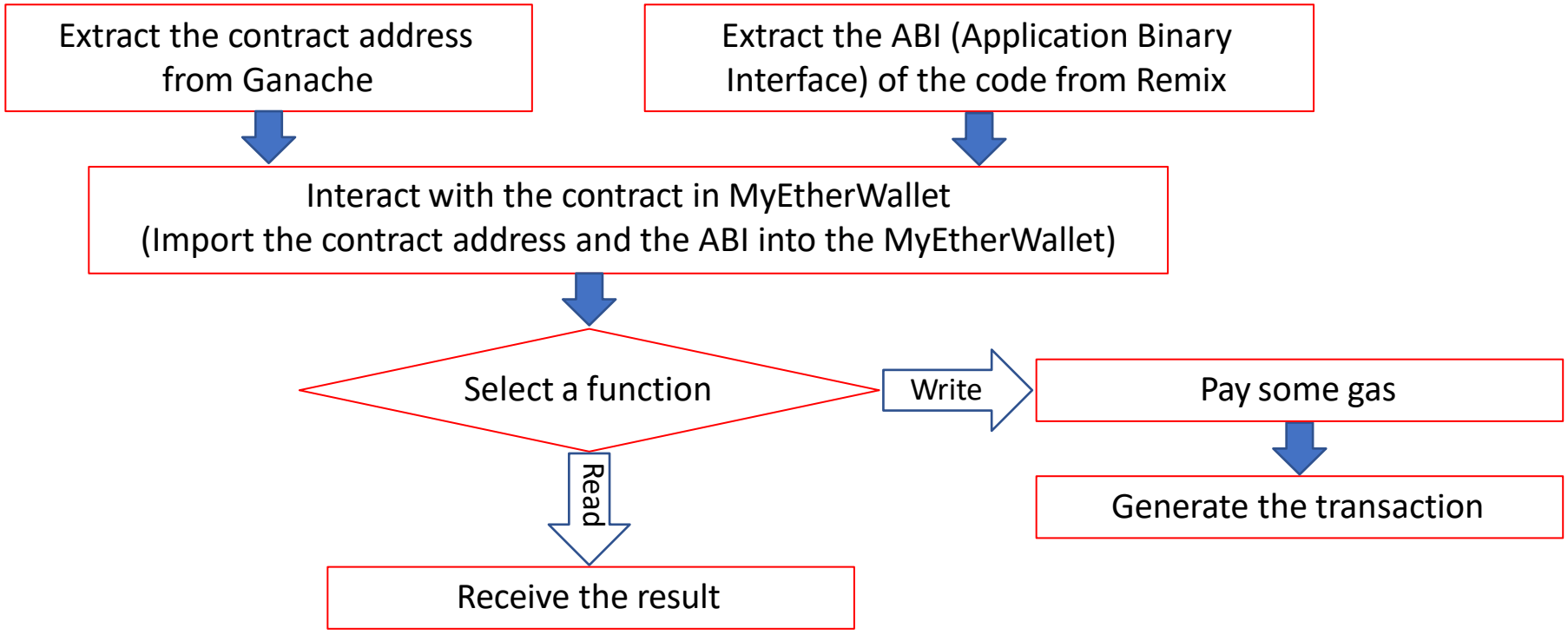


Ganache

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.

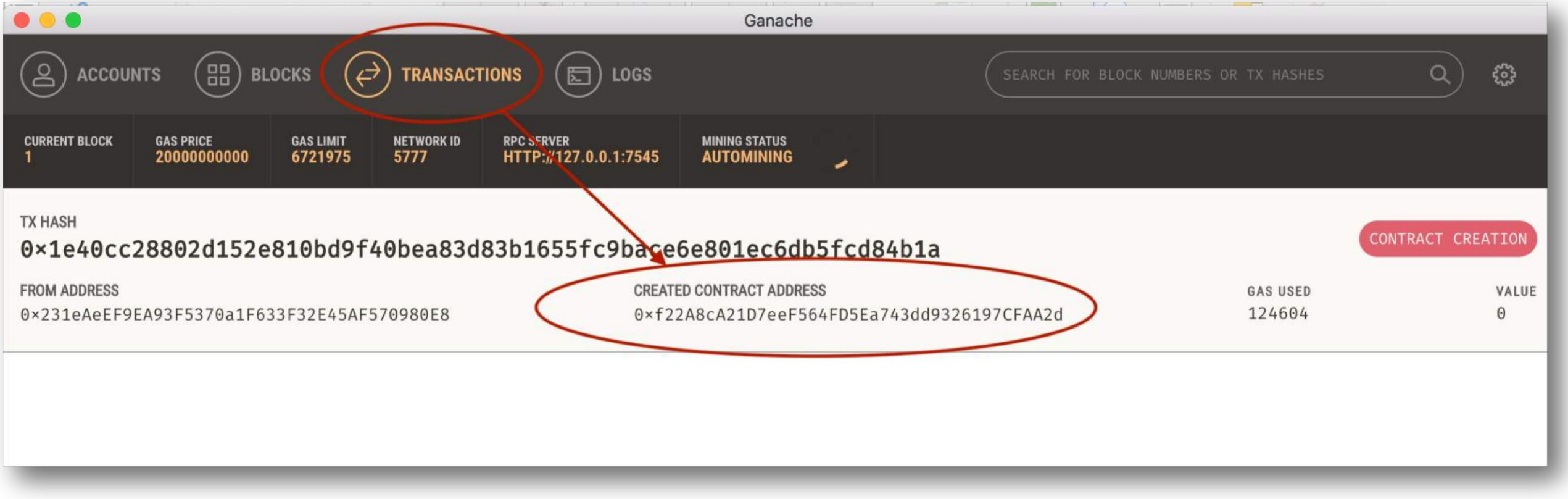


Interacting with the smart contract



Ganache

Transactions tab: Copy the created contract address



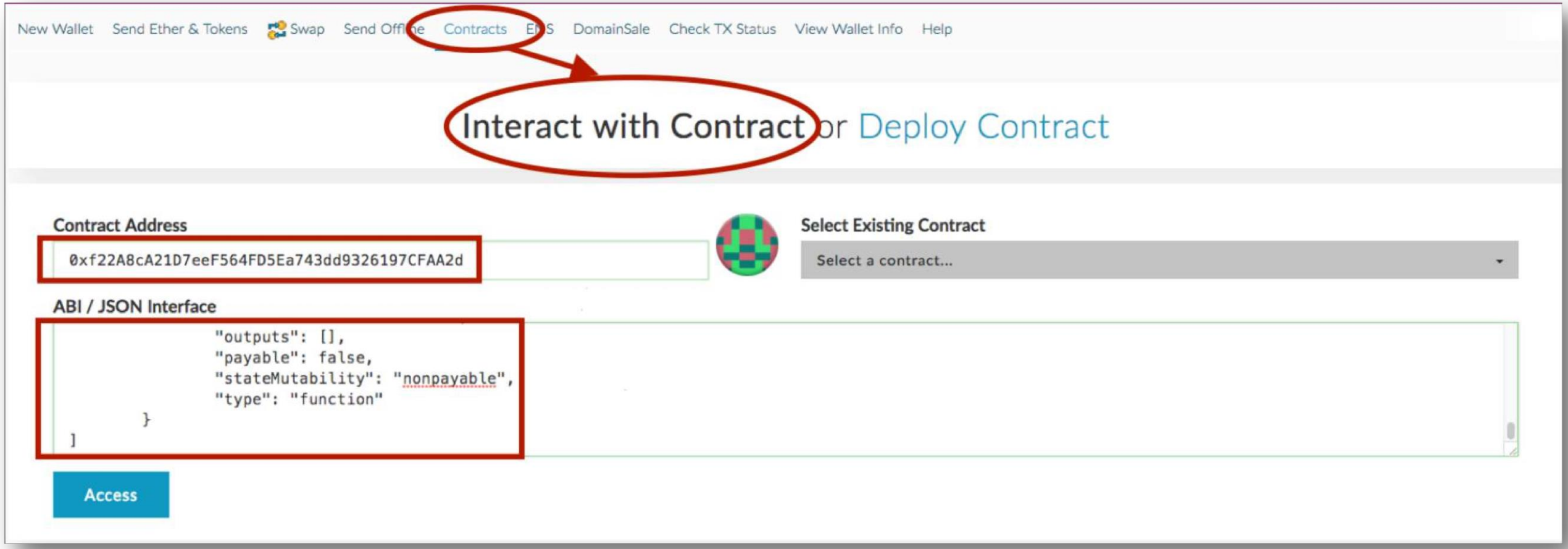
Remix

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

MyEtherWallet

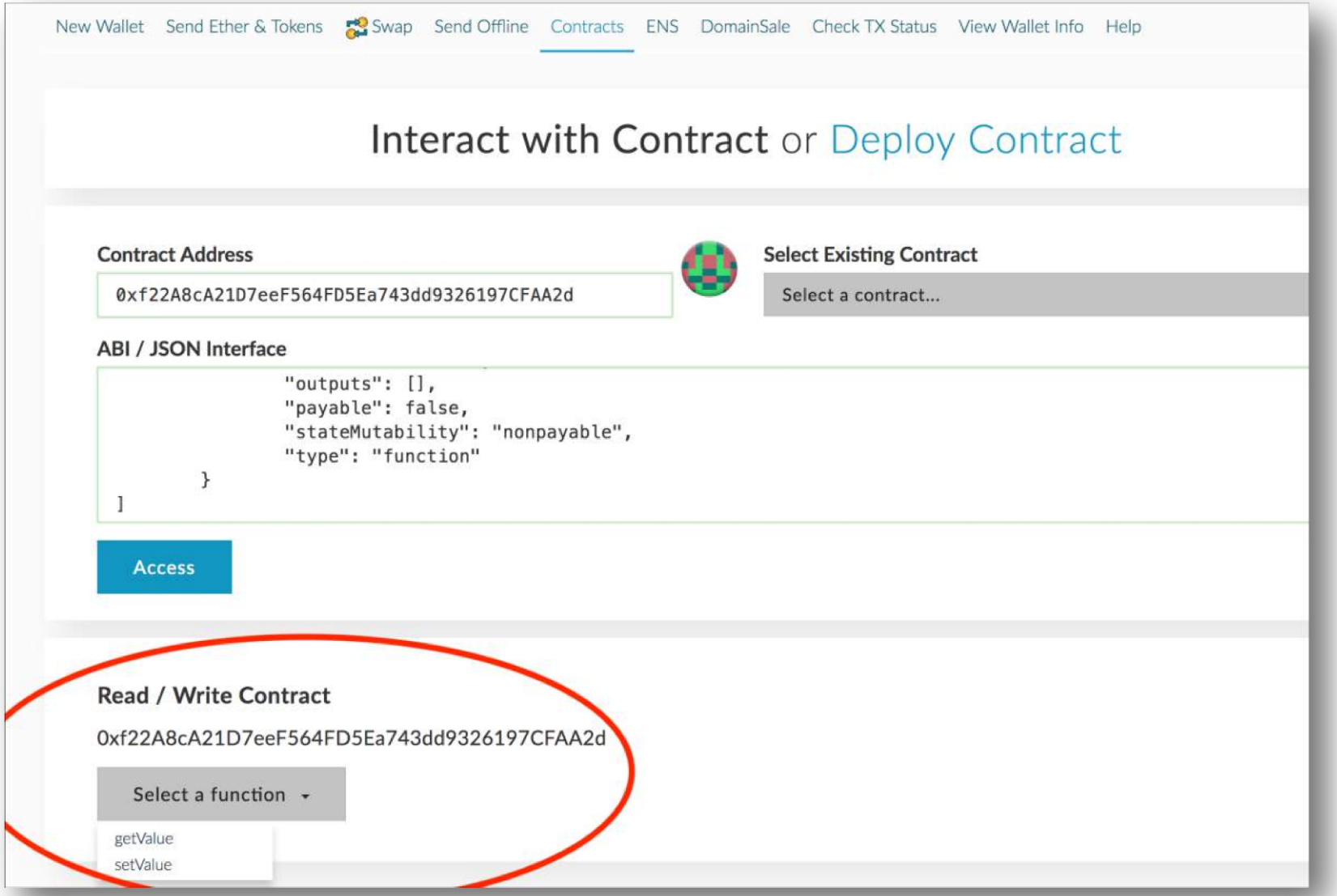
Contracts tab:

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



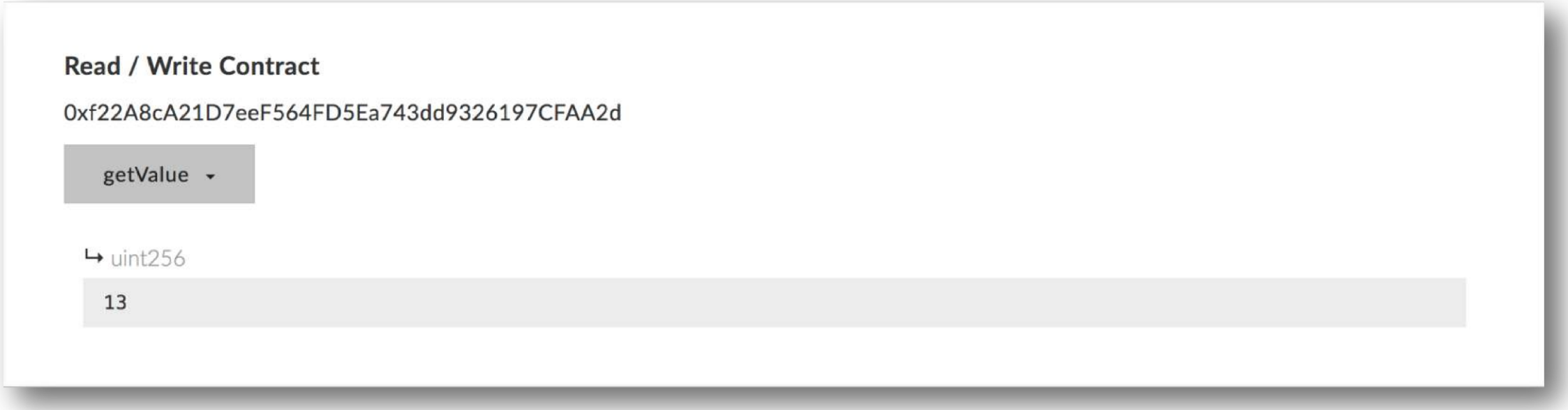
MyEtherWallet

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



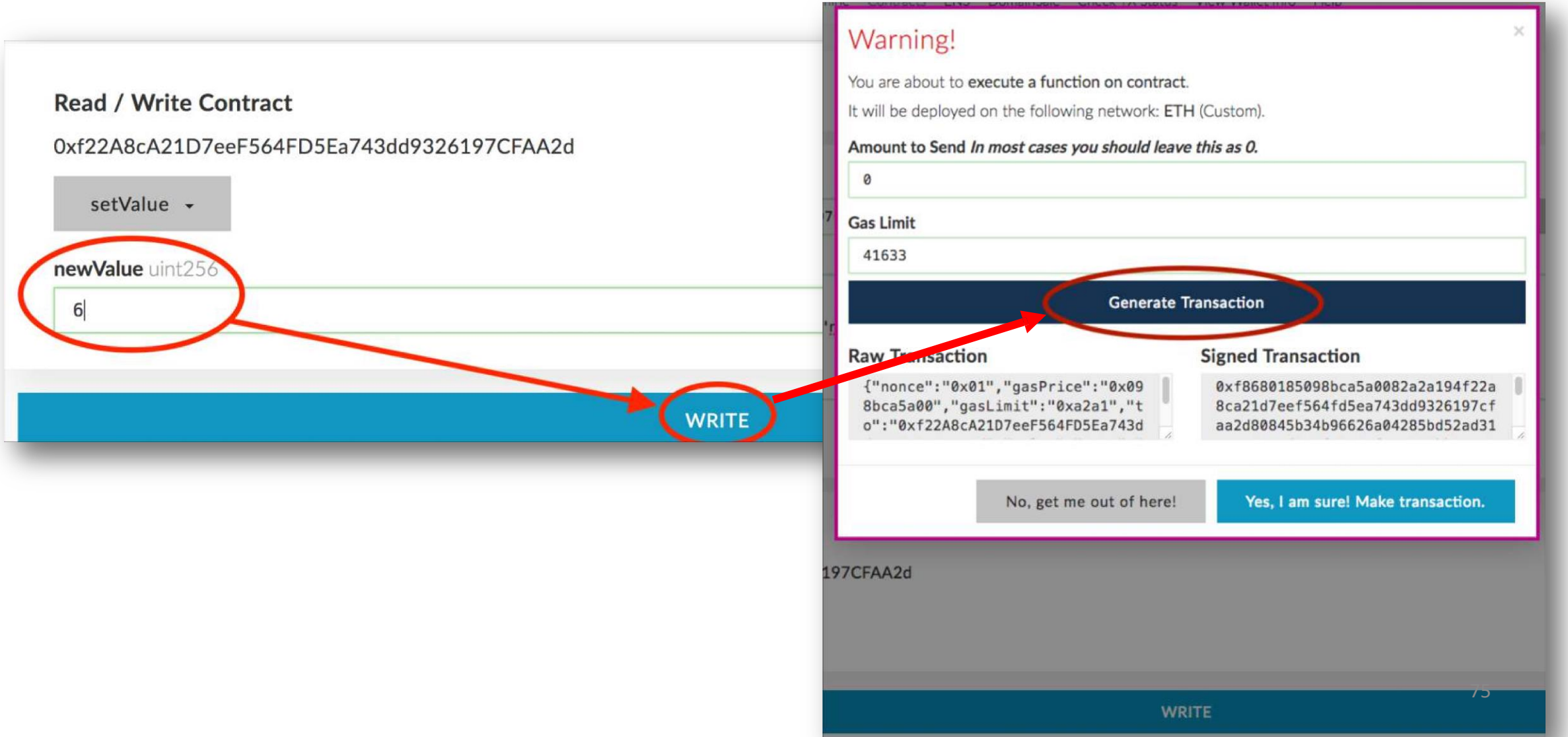
MyEtherWallet

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

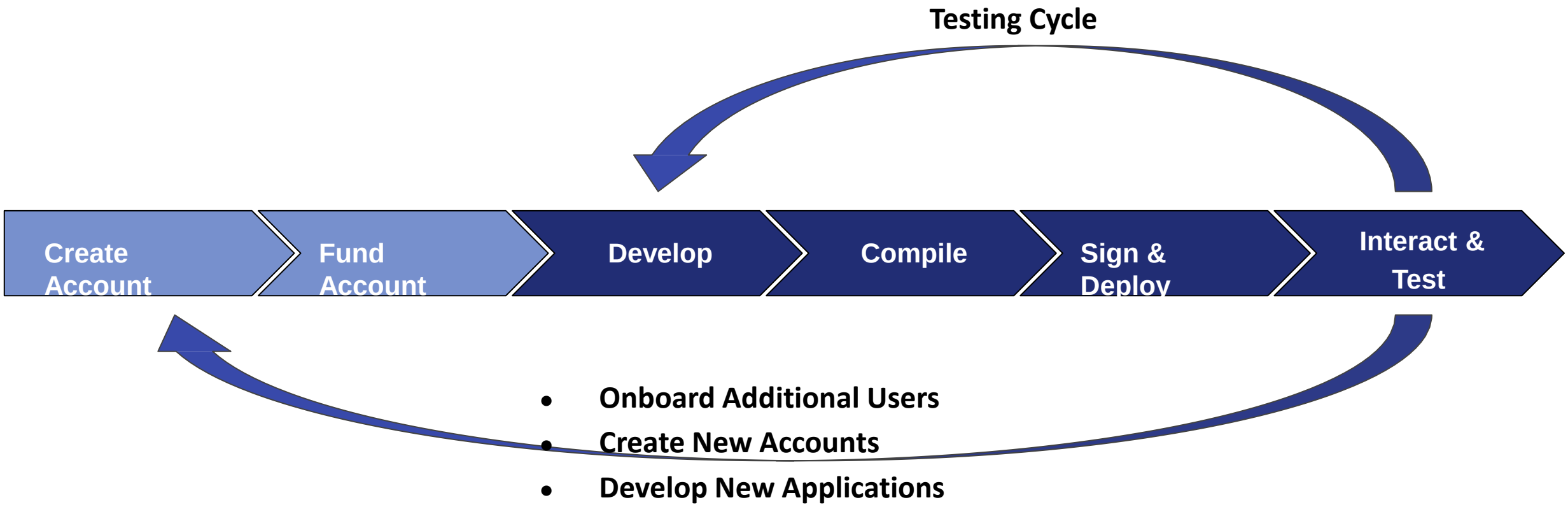


MyEtherWallet

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

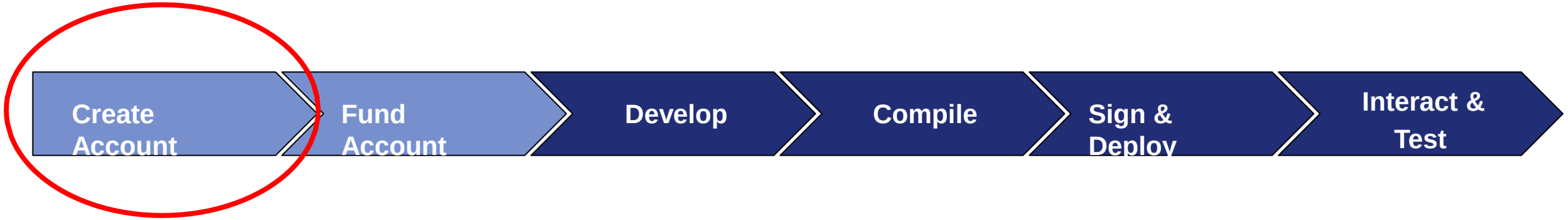


C. Development Workflow



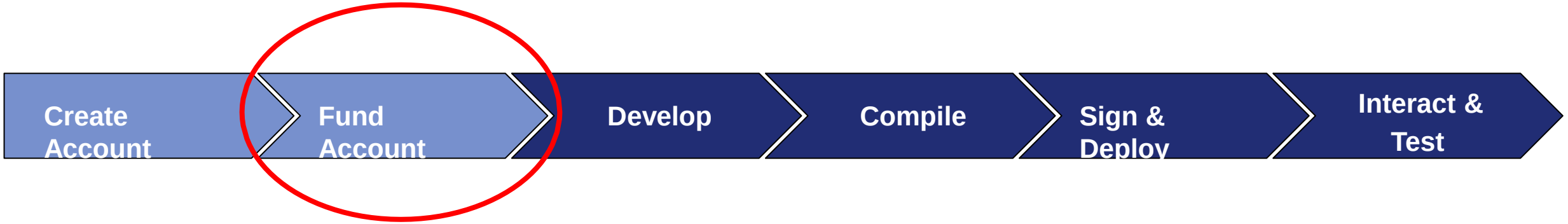
- **Onboard Additional Users**
- **Create New Accounts**
- **Develop New Applications**

C. Development Workflow: Create Account



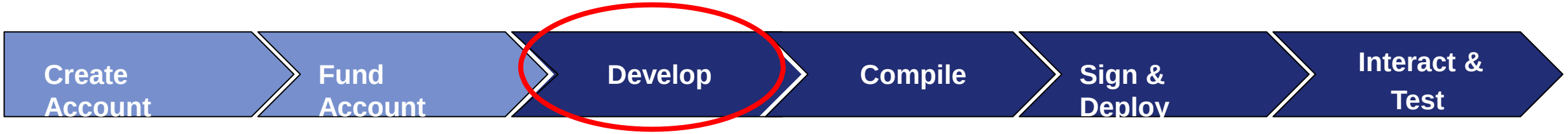
- Programmatically: Go, Python, C++, JavaScript, Haskell
- Tools
  - MyEtherWallet.com
  - MetaMask
  - TestRPC
  - Many other websites

C. Development Workflow: Fund Account



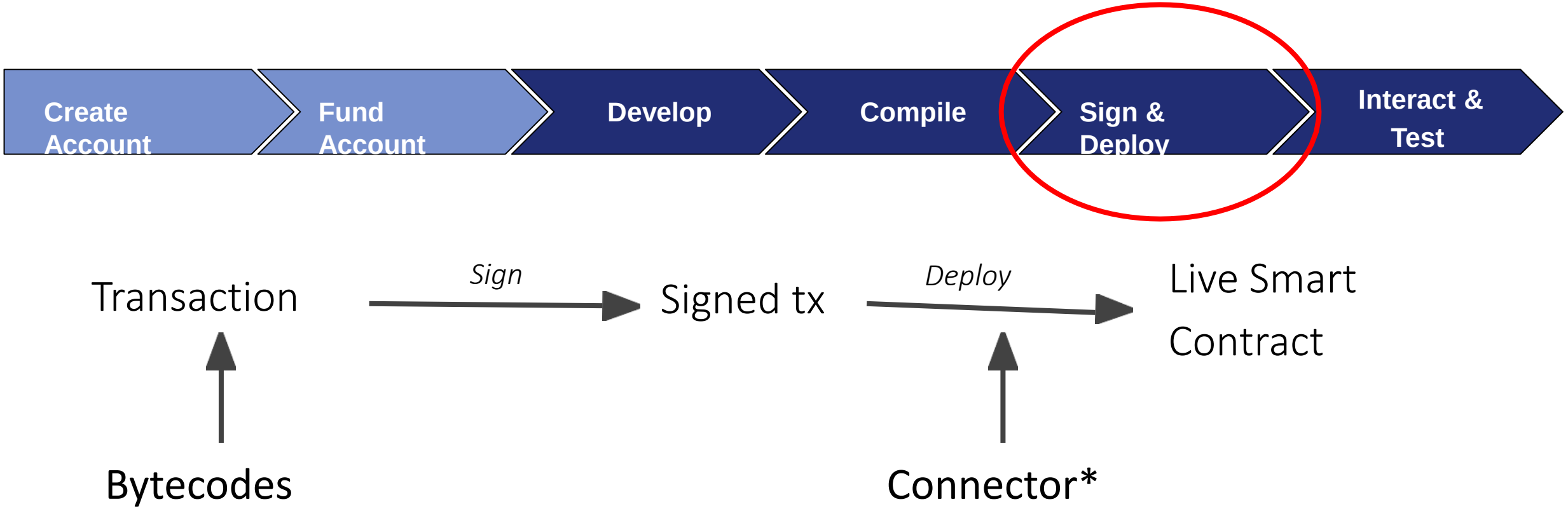
- From friends
- Faucet
- Exchanges (for public blockchain)

C. Development Workflow: Develop



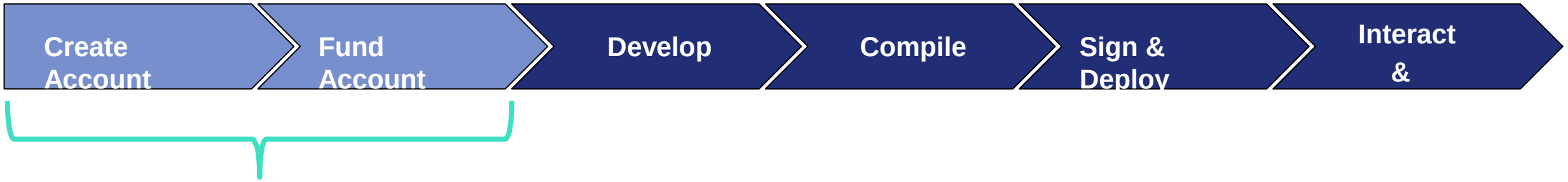
- **Ethereum Application Components:**
  - **Base application**: can be developed in **any** language
  - **Smart contract**: developed in Solidity or one of the other contract compatible languages
  - **Connector library**: facilitates communication between base application and smart contracts (Metamask)

C. Development Workflow: Sign and Deploy



*Library that facilitates communication and connection with Blockchain; Connects your

C. Development Workflow: TestRPC



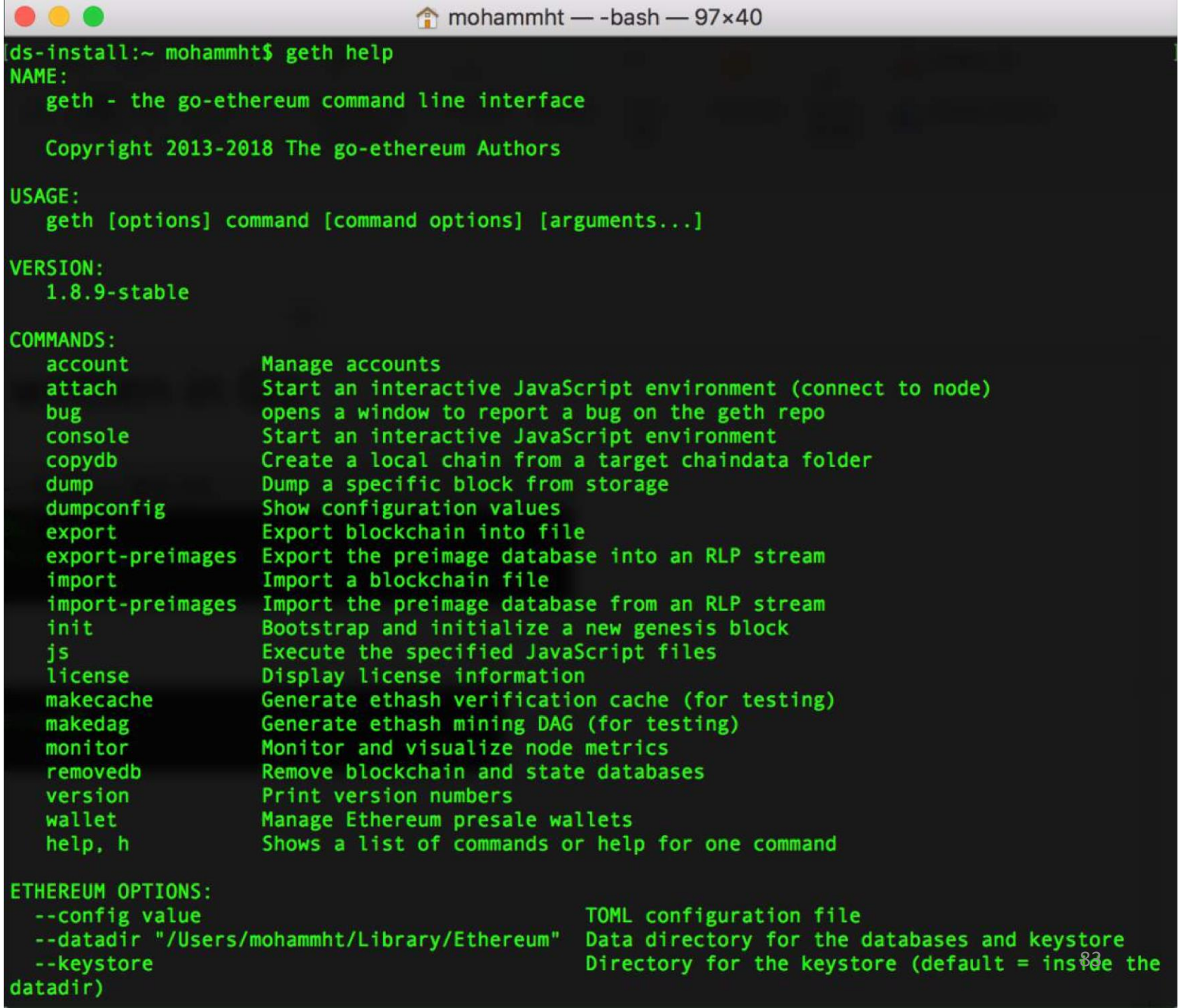
TestRPC/TestChain

- Local development or Test Blockchain
- https://github.com/ethereumjs/testrpc

Create Custom Ethereum Blockchain

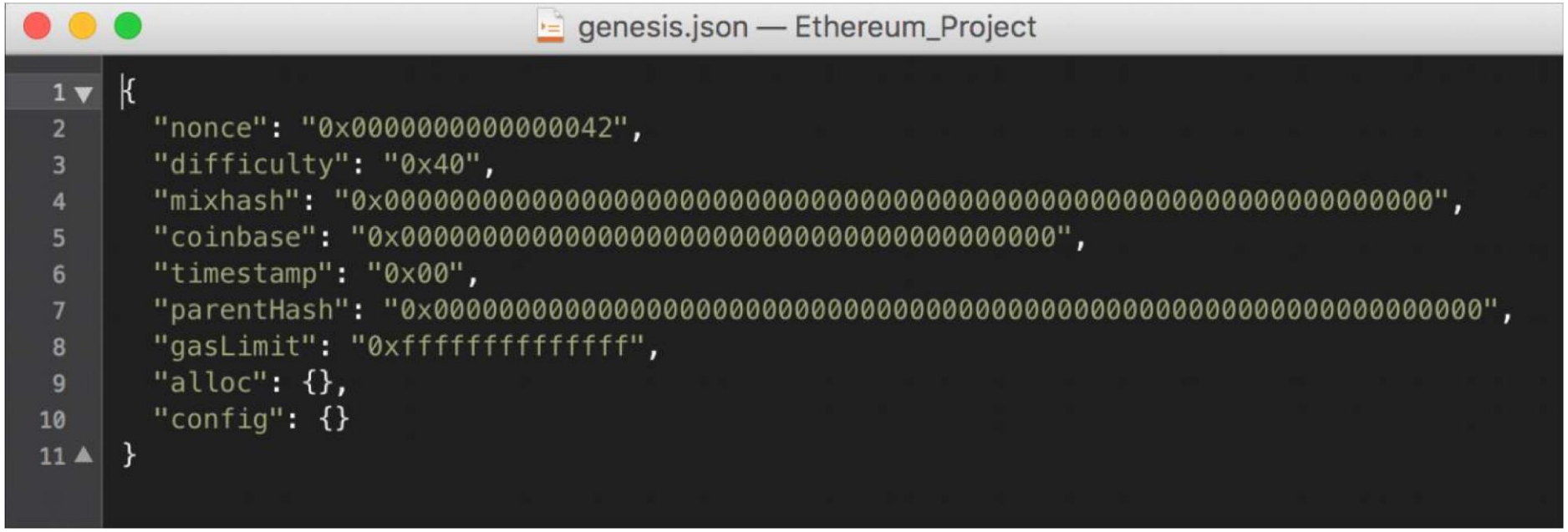
- Instead of using Ganache with its default properties for private blockchain you can run your own blockchain
- Install Geth: One of the implementations of Ethereum written in Go
- Create the genesis block
- Create storage of the blockchain
- Deploy blockchain nodes
- Connect MyEtherWallet to your blockchain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

Geth help


mohammht — -bash — 97×40
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

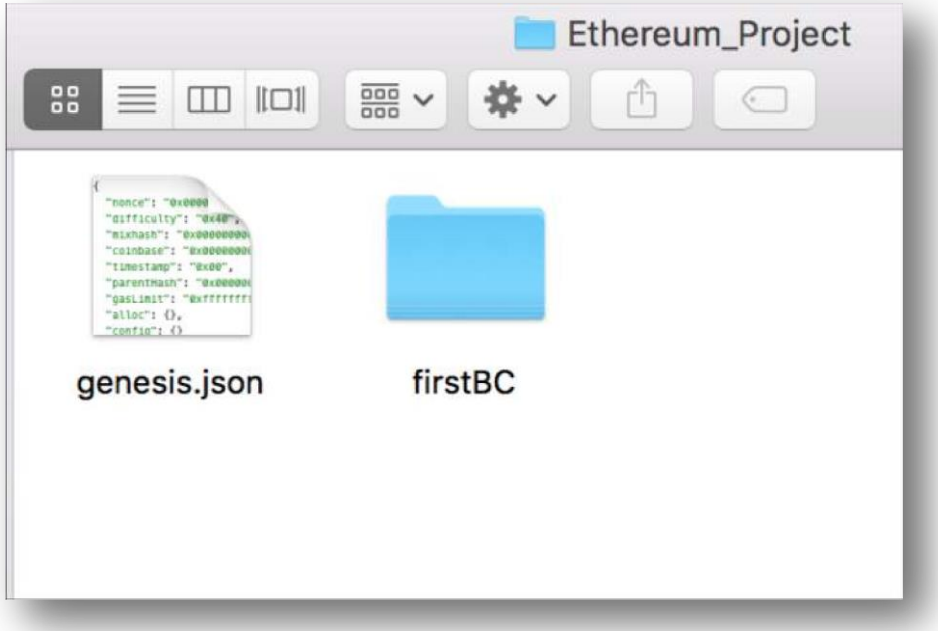


- Create and store the file as genesis.json

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

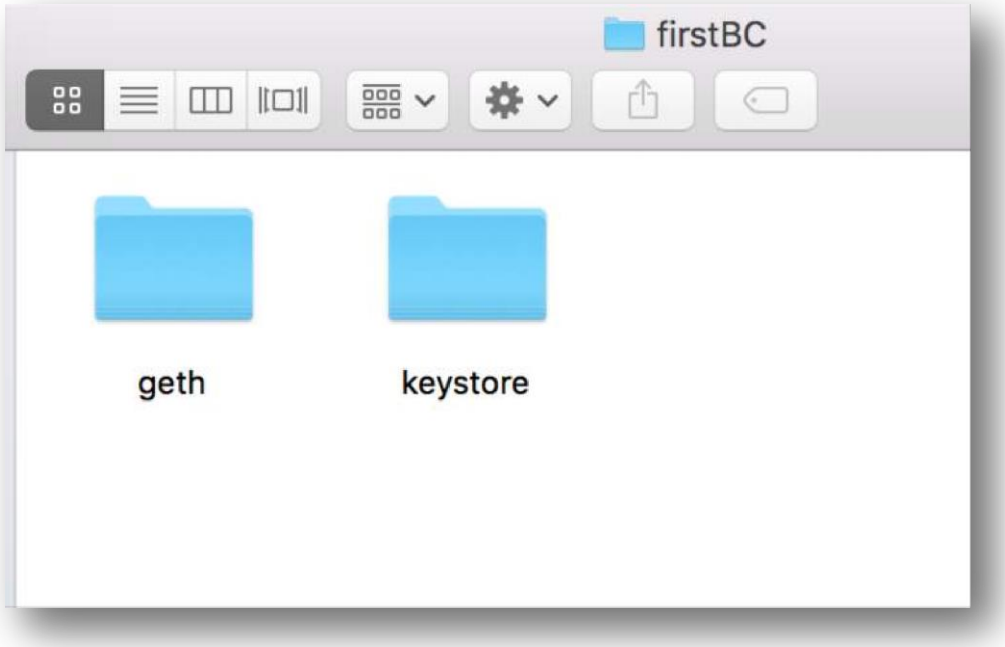
```
[ds-install:Documents mohammht$ cd Ethereum_Project/
ds-install:Ethereum_Project mohammht$ geth --datadir firstBC init genesis.json
```



Folder name of your blockchain

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://4561ccdd7fdf3f0bdbc903b7bef7d472e136fe2b63012151a1dd3c27e52f49bda2ef66631e67022
b7ca7b9fba06bb0eda8b47210b198f3eeff7e67414d695ed6@[::]:30303",
  id: "4561ccdd7fdf3f0bdbc903b7bef7d472e136fe2b63012151a1dd3c27e52f49bda2ef66631e67022b7ca7b9fba0
6bb0eda8b47210b198f3eeff7e67414d695ed6",
  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

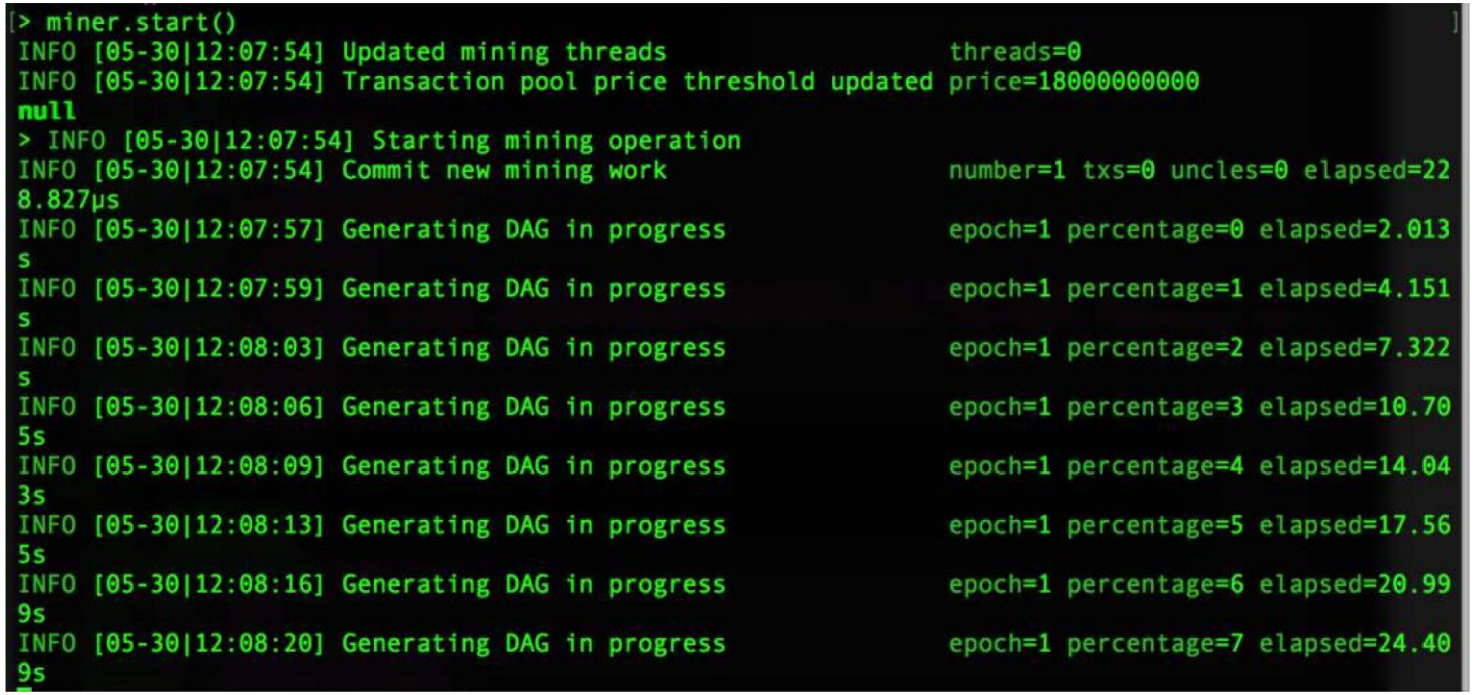
```
> personal.newAccount('Type your password here')
"0xa78eb41a10f096d4d8c4c9ca5196427aaa3fdb33"
>
```

- See the created account(s)

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

Mining

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



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