# Exercises: Playing with Mist

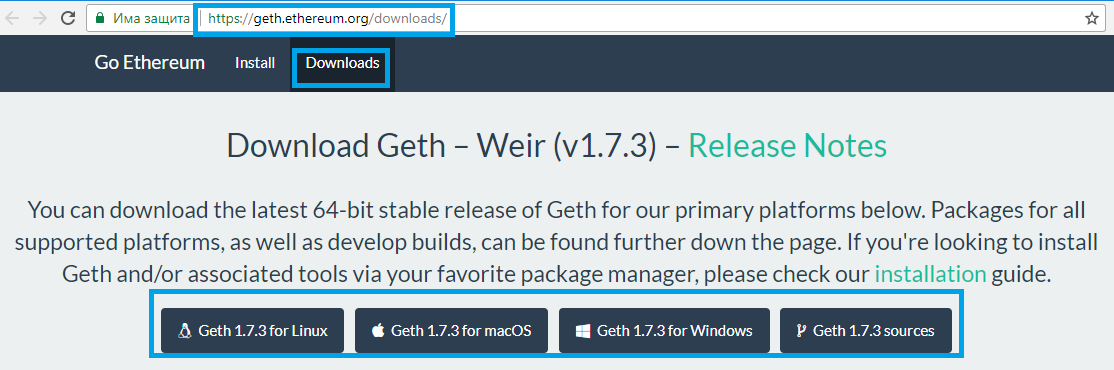
In this exercise we shall install a **Mist** and **Geth** and will play with them. We shall **create account**, **mine ethers** to it, **send ethers** from one account to another, **create a smart contract** and play with it.

* [**Geth**](https://geth.ethereum.org) is the command line interface for running a **full Ethereum node** implemented in Go. It is one of the most popular Ethereum clients along with [**Parity**](https://www.parity.io). By installing and running Geth, you can take part in the Ethereum frontier live network and interact with it, e.g.
  + Mine ethers
  + Transfer funds between addresses
  + Create contracts and send transactions
  + Explore the block history
  + and much, much more …
* [**Mist**](https://github.com/ethereum/mist) is a **GUI Ethereum client** and **wallet** software which provides a nice-looking GUI for Geth. It is an Electron-based JavaScript desktop application (a desktop hybrid application with a Web interface).

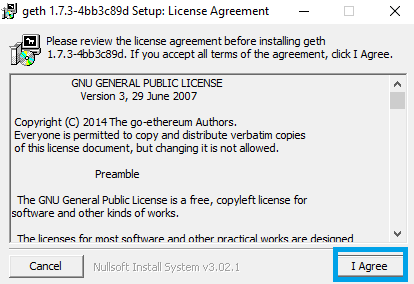
## Download and Install Geth

First, let’s **install a Geth** on your local machine.

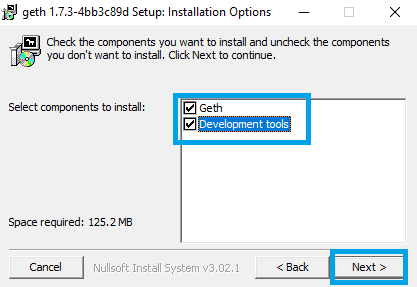
1. Go to <https://geth.ethereum.org/downloads/> and choose one of the following files.



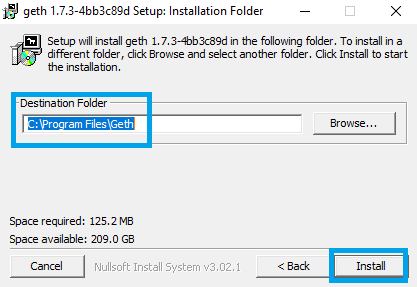
1. Then start the downloaded file and **Accept** the agreement.



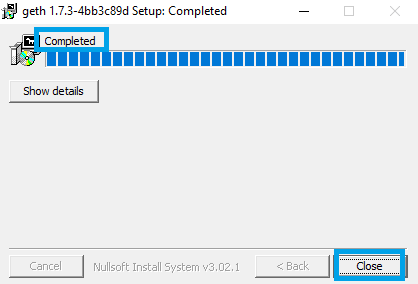
1. Choose the **Geth** and **Development tools** components and click [**Next**].



1. Select the location where you want to install Geth and click [**Next**].



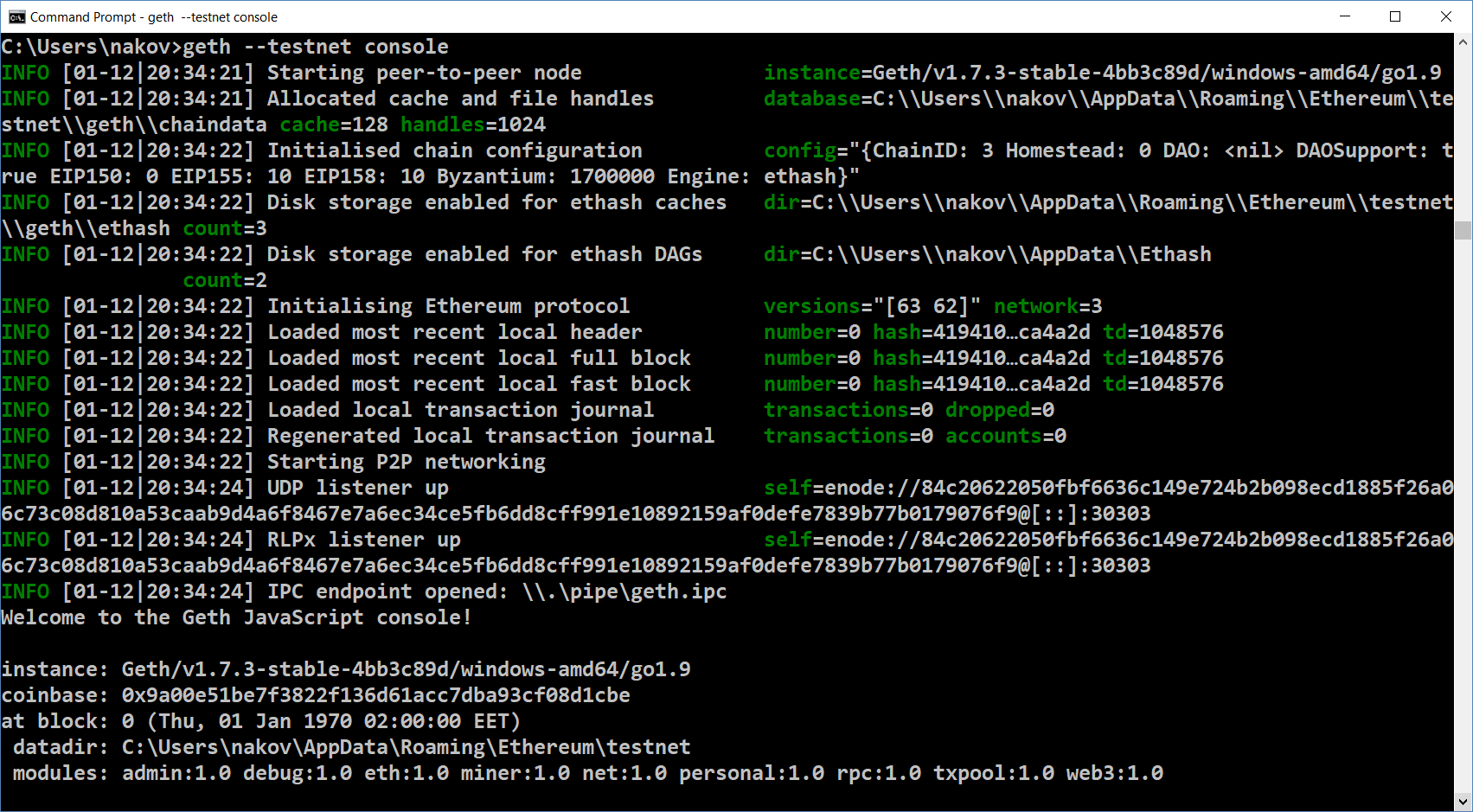
1. After it is completed, click [**Close**].



## Play with Geth

**Run Geth** and select the **Ropsten testnet** and open the JavaScript **console**:

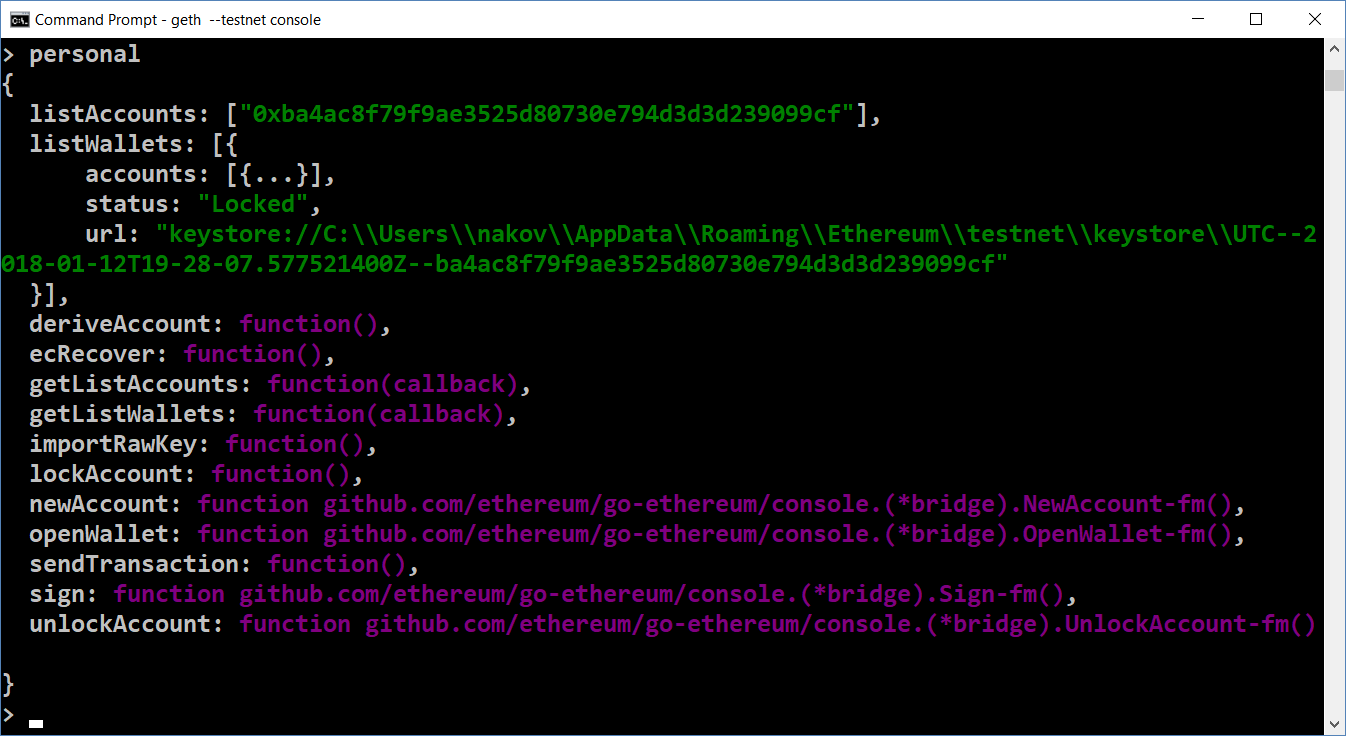
|  |
| --- |
| geth --testnet console |



At the console, you can interactively execute commands, just as in the JavaScript REPL in Node.js.

Display the available **wallets** and **accounts**. Type the following command at the console:

|  |
| --- |
| personal |



Create a **new account** in your wallet (if you don’t have already):

|  |
| --- |
| personal.newAccount() |

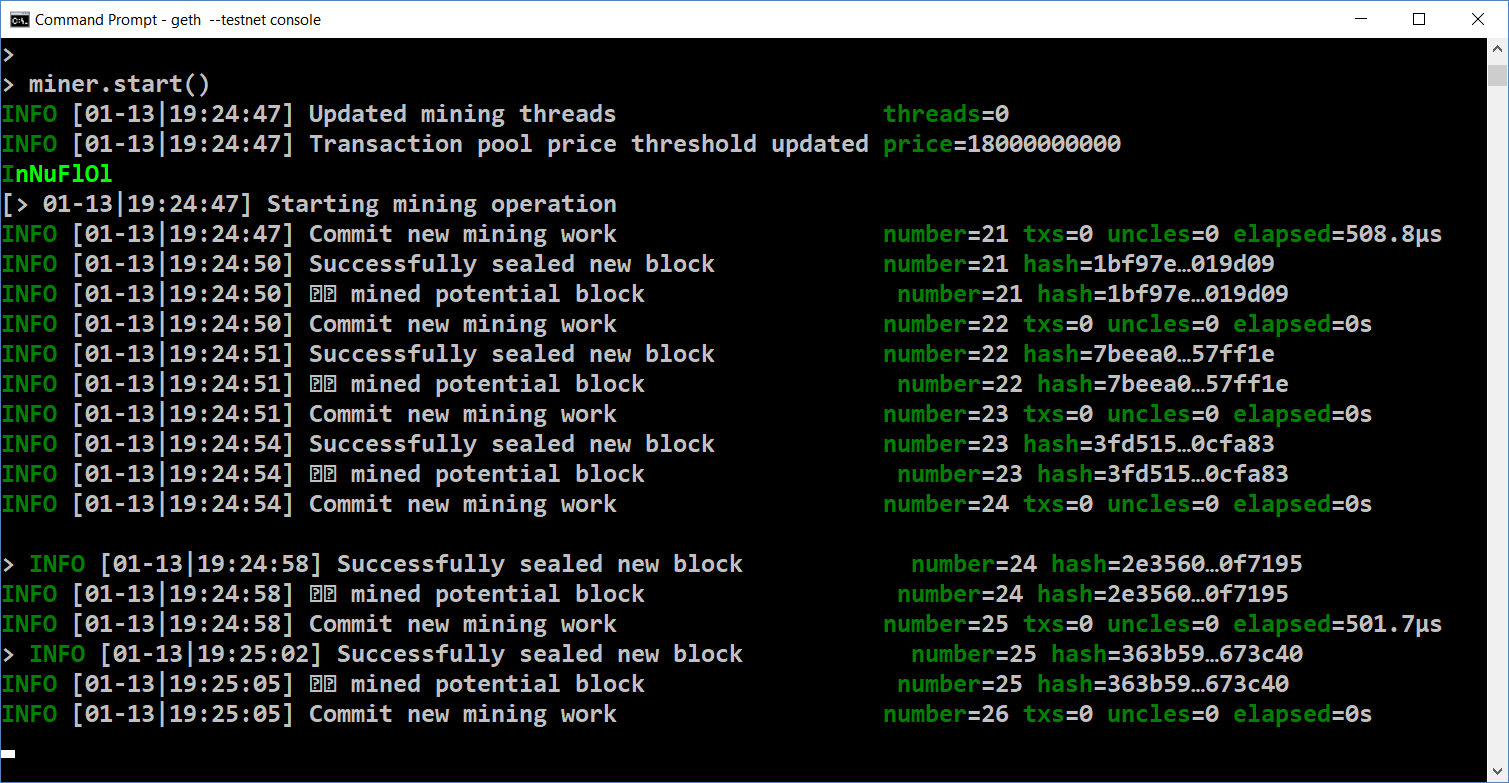
List all your accounts **again** after you have created a new account:

|  |
| --- |
| personal |

Start **CPU mining** inside your local Ethereum node:

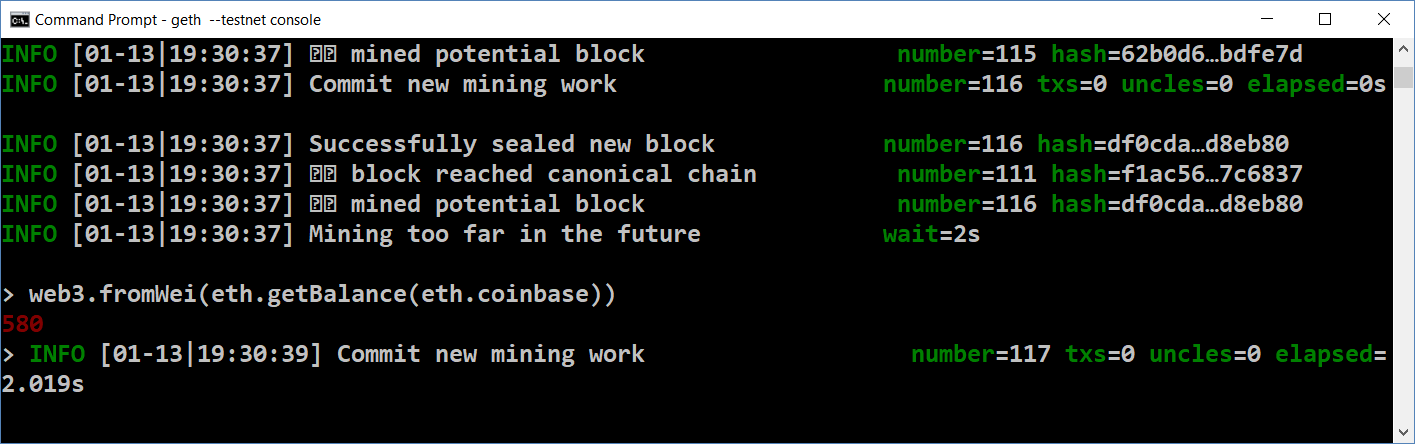
|  |
| --- |
| miner.start() |

Note that the **mining will happen in your local node only**, because it is not connected to any existing networks. The miner will **mine 5 ETH per ~ 12 seconds** (stored in your first account in your local wallet). If you want your node to process transactions, the miner should be running continuously. The miner’s activity looks like this:



Check the **account balance** to see how much ETH you have mined:

|  |
| --- |
| web3.fromWei(eth.getBalance(eth.coinbase)) |



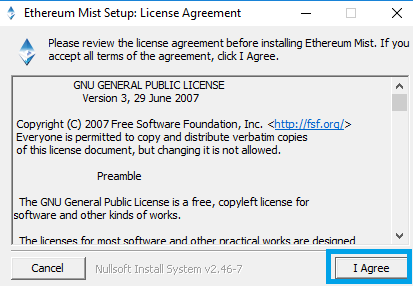
## Download and Install Mist

Now install **Mist** – the official Ethereum **GUI client**, that attaches to your Geth and interacts with it.

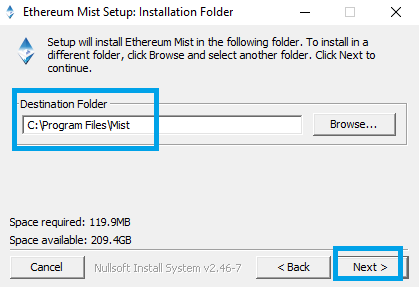
1. Go to <https://github.com/ethereum/mist/releases/> and download one of the following files:



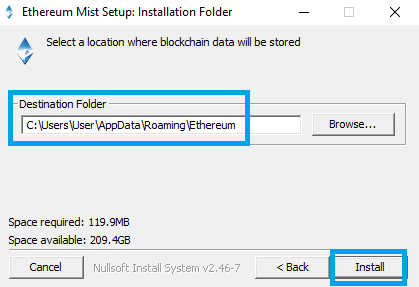
1. Run the downloaded file and **Accept** the license agreement.



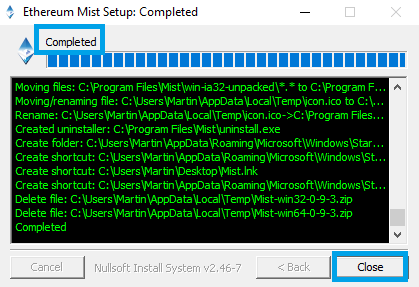
1. Select the location where you want to install **Mist** and press [**Next**].



1. Select the location where the blockchain data will be stored and click [**Install**].



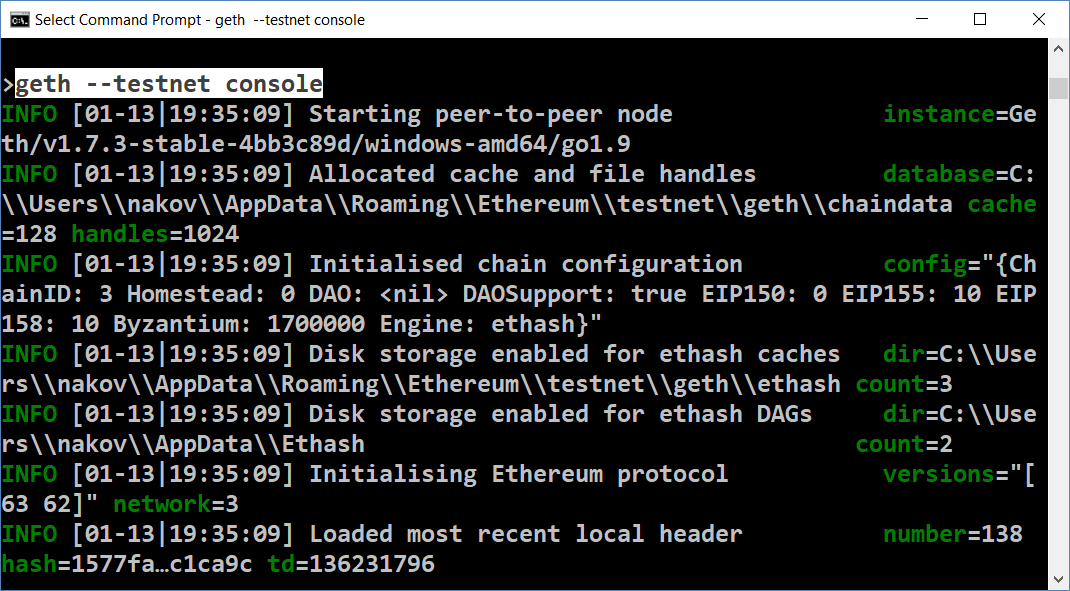
1. After the installation is completed, click [**Close**].



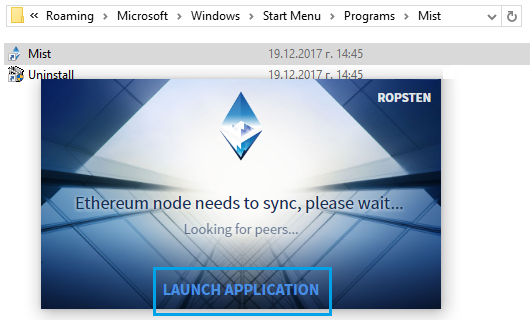
## Start Geth and Mist and Create New Account

1. Open command prompt and **start Geth** on the **testnet** with interactive **console**:

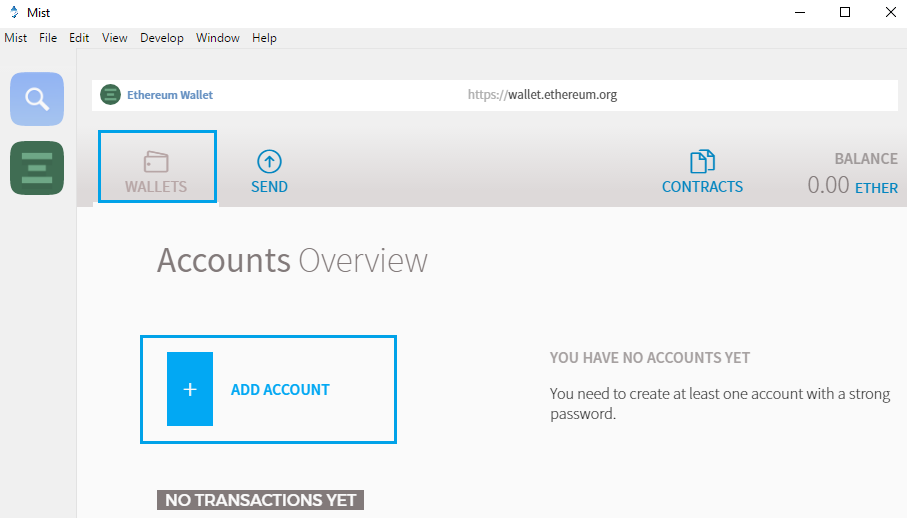
|  |
| --- |
| geth --testnet console |



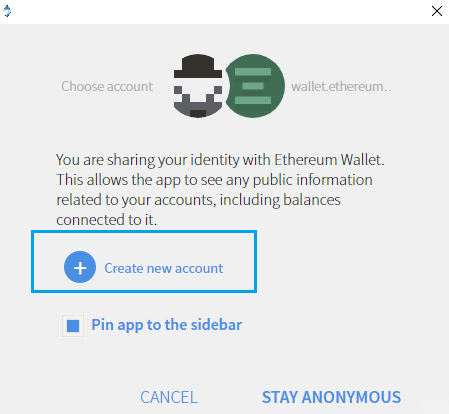
1. Start **Mist**. It will automatically connect to the local **Geth** node. Press [**Launch Application**].



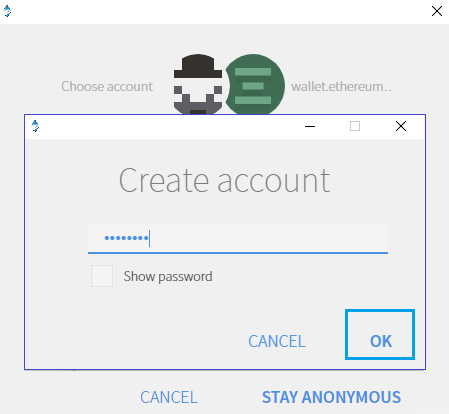
1. You will see the following window. Go to the [**Wallets**] tab. Your accounts will appear. Press [**Add Account**] to create a new account. You will need at least two accounts in your wallet in order to send transaction between them.



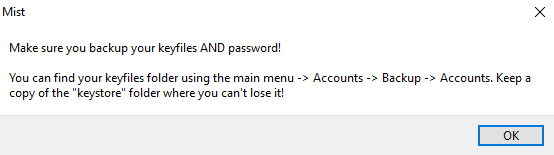
1. Press [**Create new account**].



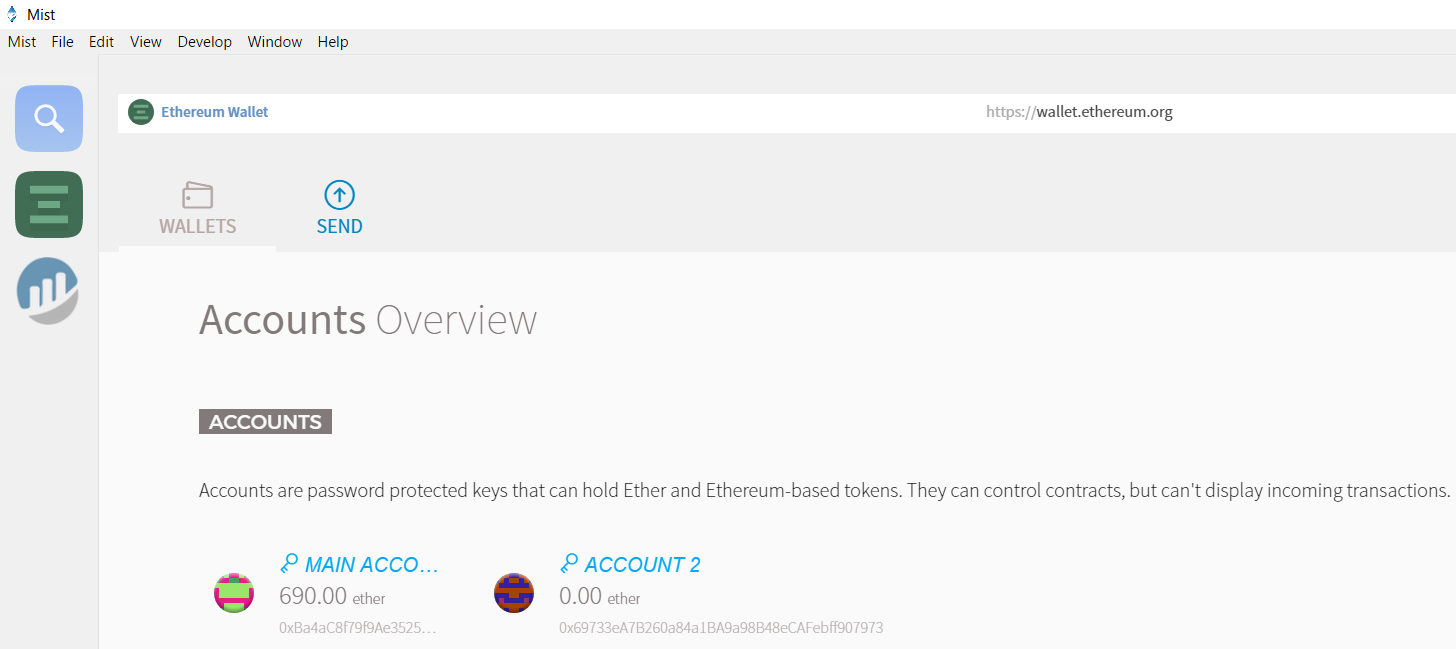
1. Enter a password you want and press [**OK**].



1. You should see the following window after successful account creation.



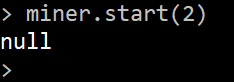
Your wallet should now hold at least **two accounts**:



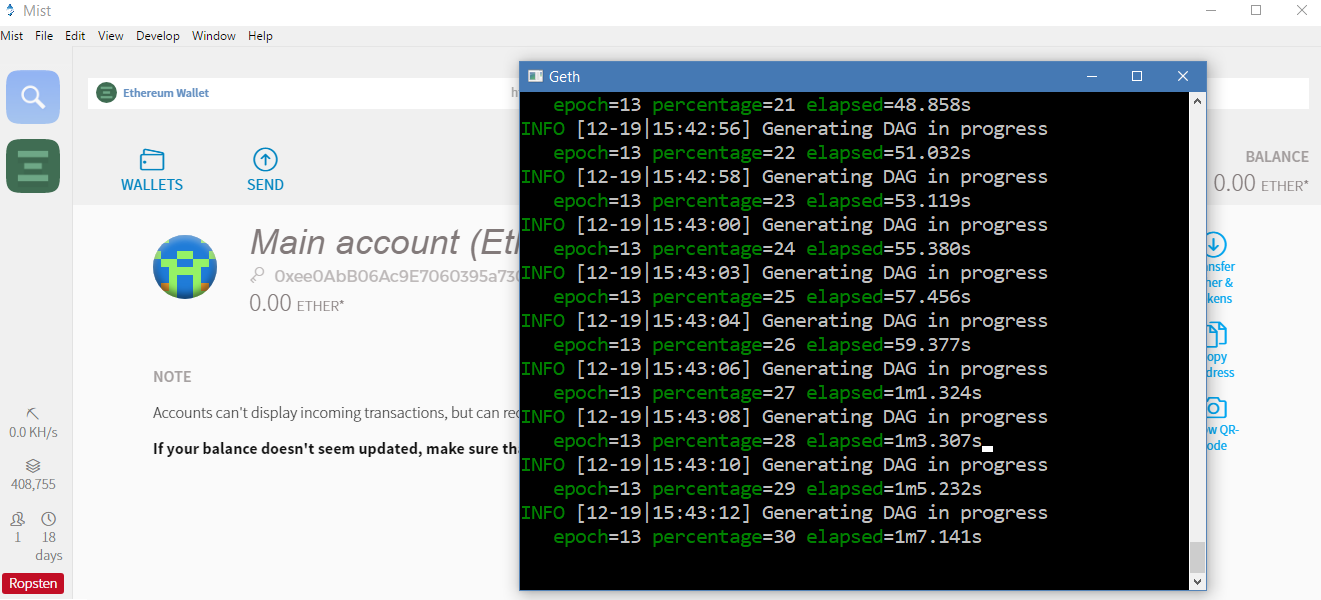
## Mine and Send Between Accounts

When your Mist browser is started and connected to your local Ethereum node, your mining is visualized in Mist.

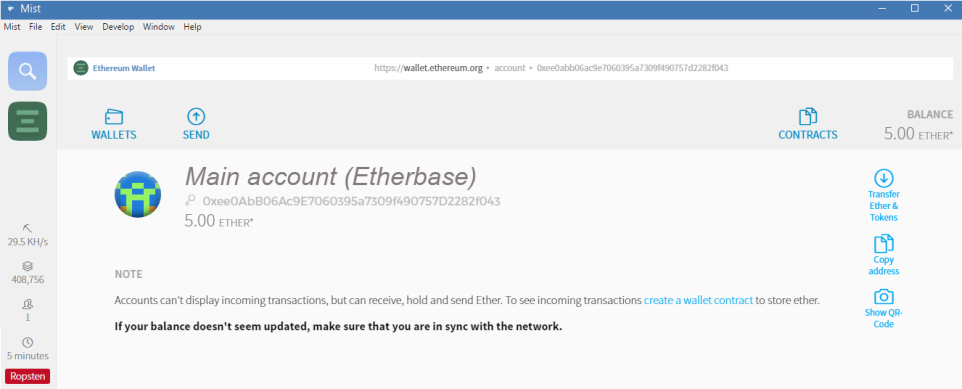
1. **Start the miner** in your Geth console with 2 threads (it is an optional parameter).



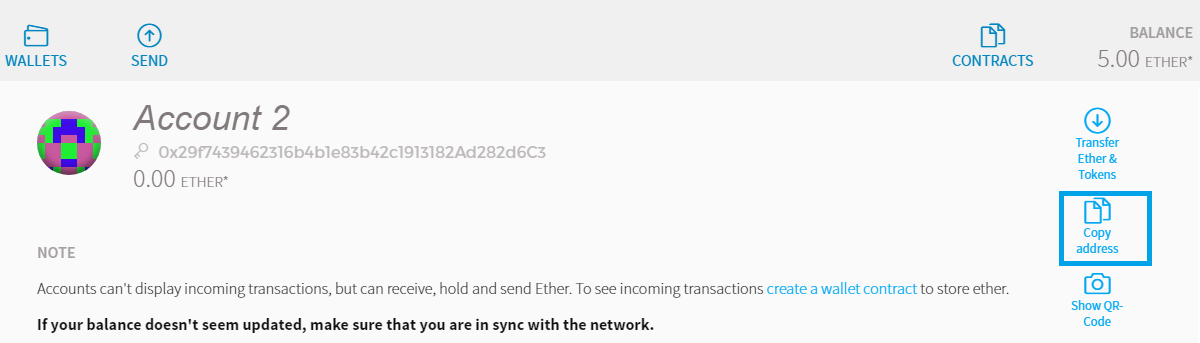
1. You can look at the Geth console that the miner is started mining.



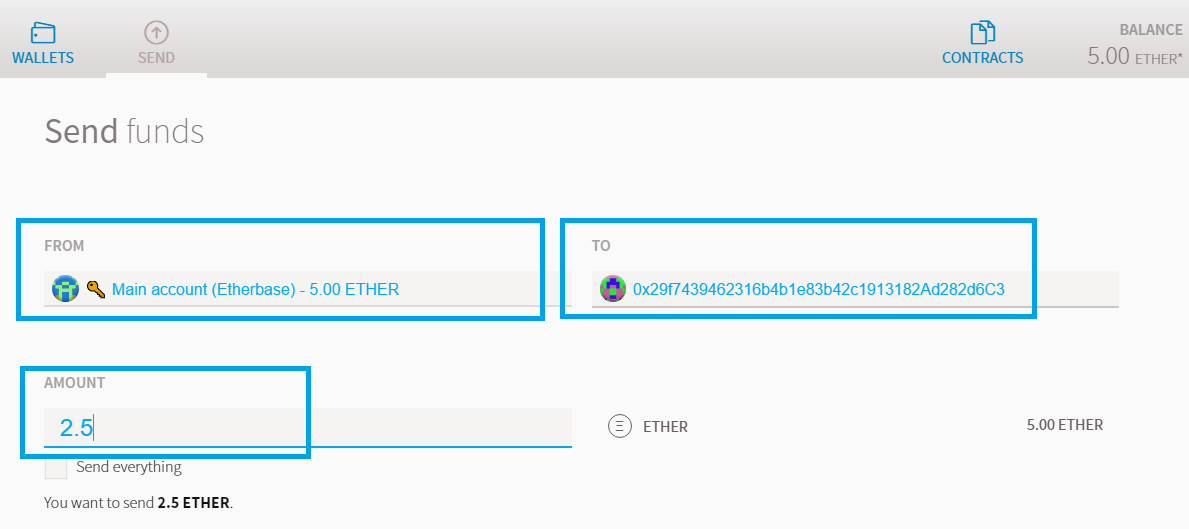
1. After a while you will see additional **5.00 ethers** at your main account.



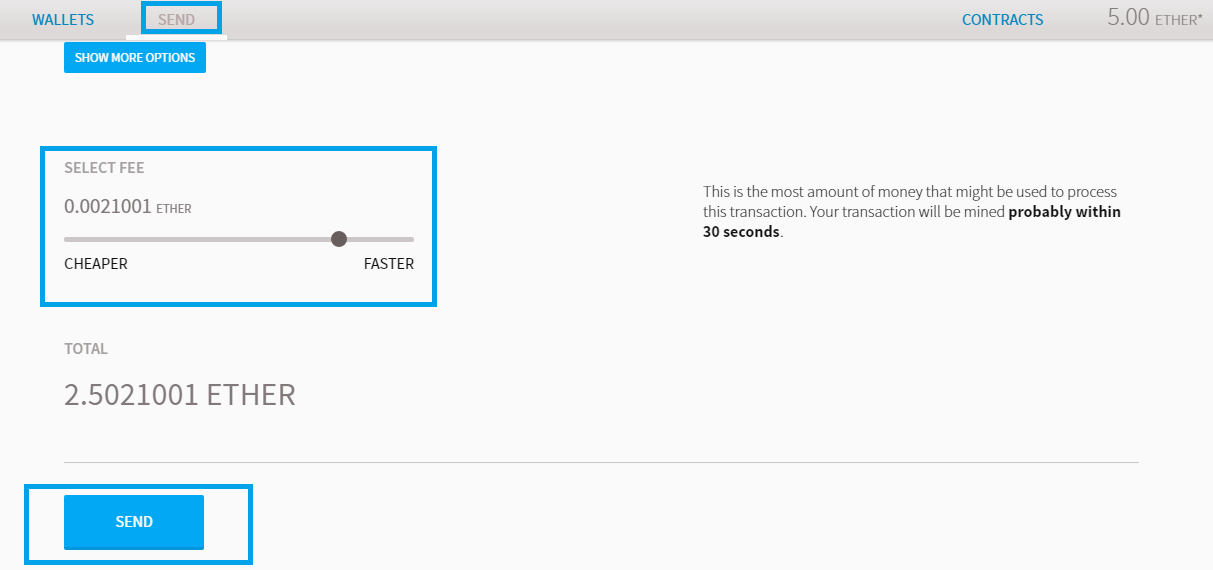
1. Copy the **address** of **Account 2** that you created previously:



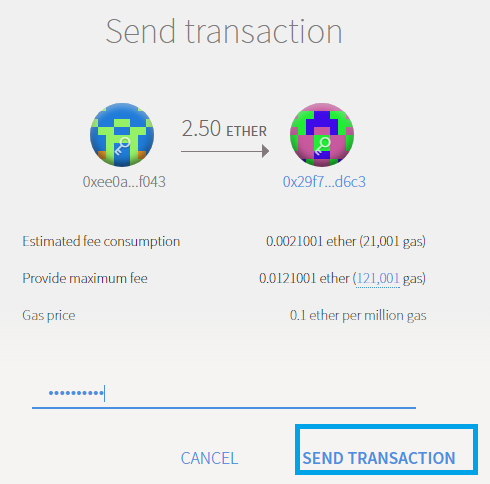
1. Go to [**Send**]tab, paste the **Account 2** address and enter amount to send, e.g. **2.5 ethers**.



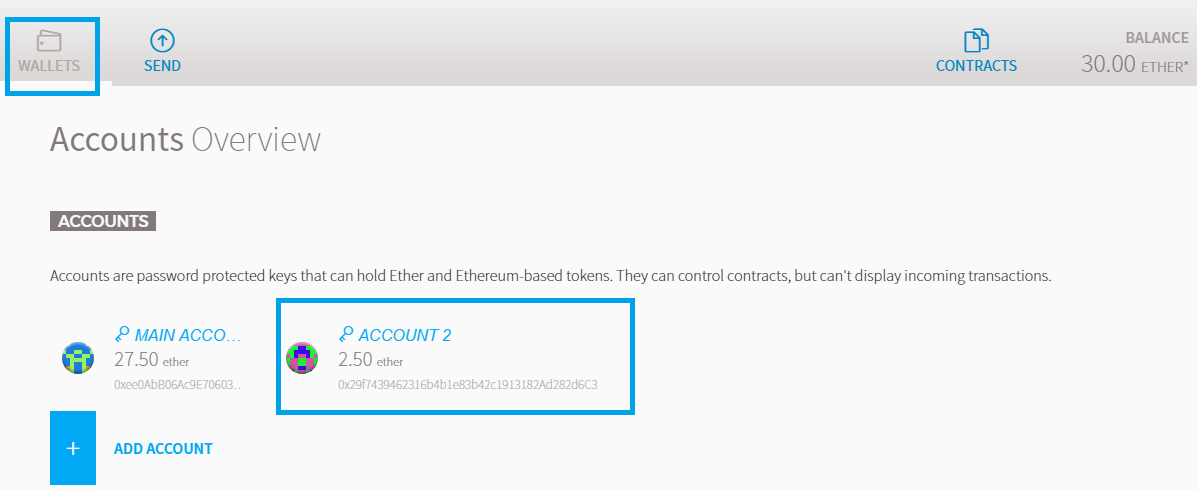
1. You can adjust the miner’s fee for faster (and more expensive) transaction or slower (and cheaper) transaction. Finally click [**Send**].



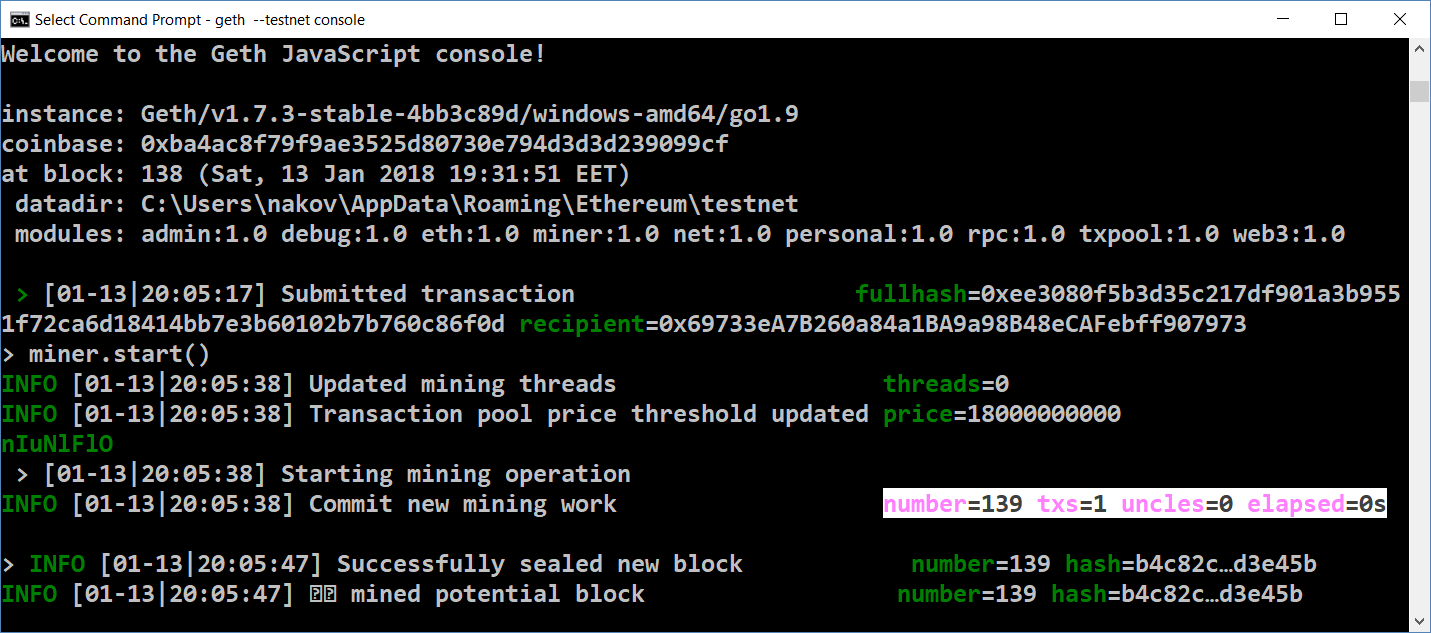
1. Enter your **sender account password** and click [**Send Transaction**].



1. After the successful transaction you should see the following window.

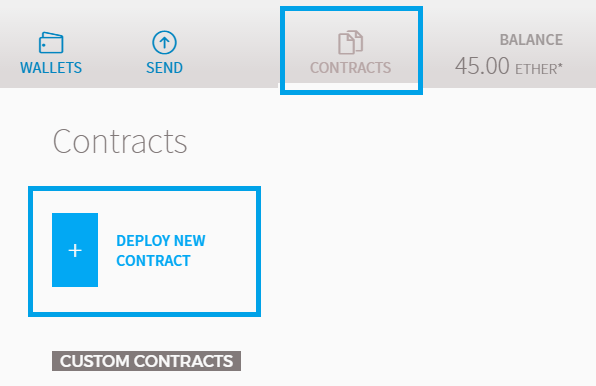


See the Geth console that runs the miner. It should hold a block with 1 transaction (your transaction you just sent):

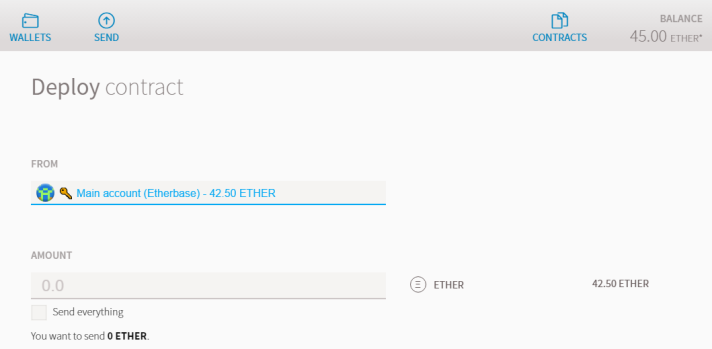


## Create a Smart Contract

1. Go to **“Contracts”** tab and click **“Deploy new contract”**.



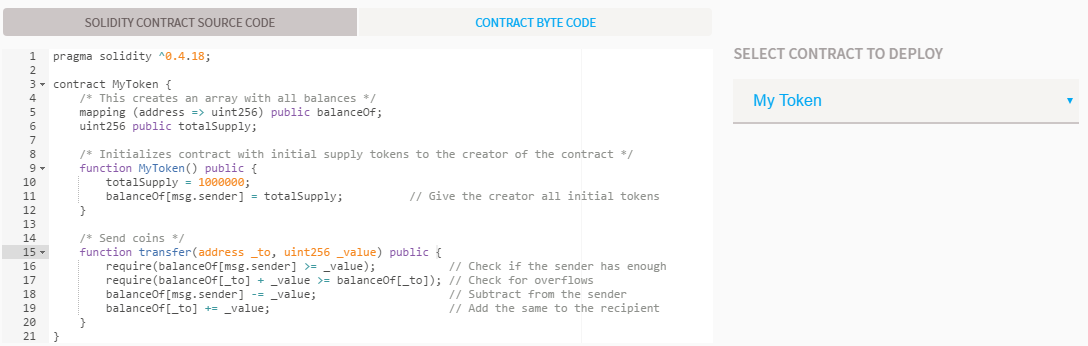
1. Choose the account from which you want to deploy the contract.



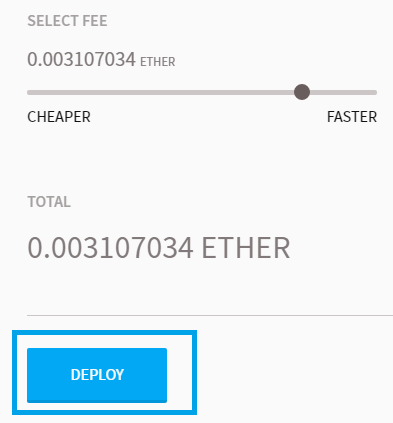
1. Write this code to **“Solidity Contract Source Code”**:

|  |
| --- |
| pragma solidity ^0.4.18;  contract MyToken {  /\* This creates an array with all balances \*/  mapping (address => uint256) public balanceOf;  uint256 public totalSupply;  /\* Initializes contract with initial supply tokens to the creator of the contract \*/  function MyToken() public {  totalSupply = 100000;  balanceOf[msg.sender] = totalSupply; // Give the creator all initial tokens  }  /\* Send coins \*/  function transfer(address \_to, uint256 \_value) public {  require(balanceOf[msg.sender] >= \_value); // Check if the sender has enough  require(balanceOf[\_to] + \_value >= balanceOf[\_to]); // Check for overflows  balanceOf[msg.sender] -= \_value; // Subtract from the sender  balanceOf[\_to] += \_value; // Add the same to the recipient  }  } |

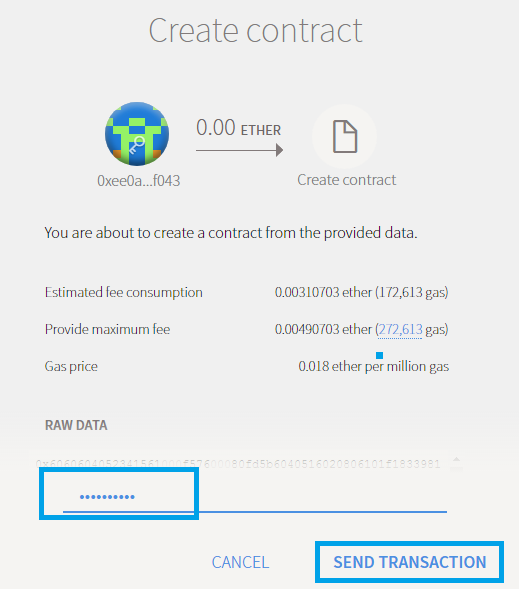
It is very basic **token contract** which have **totalSupply** = **1000000** and function **“Transfer”** to transfer tokens between accounts.



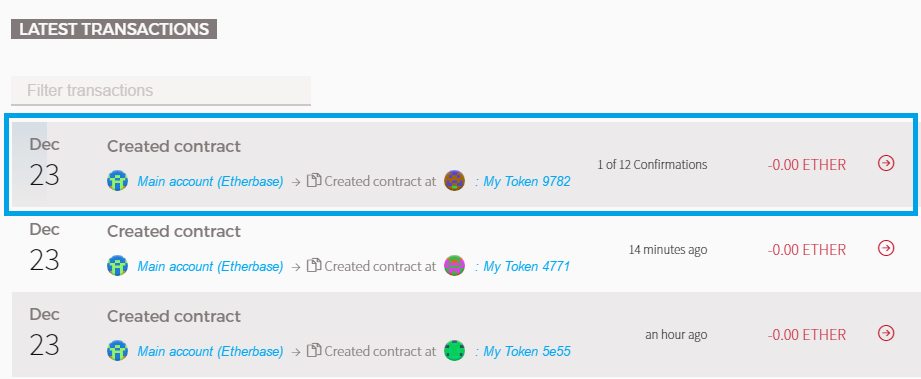
1. You can create the token faster or cheaper. Click **“Deploy”**.



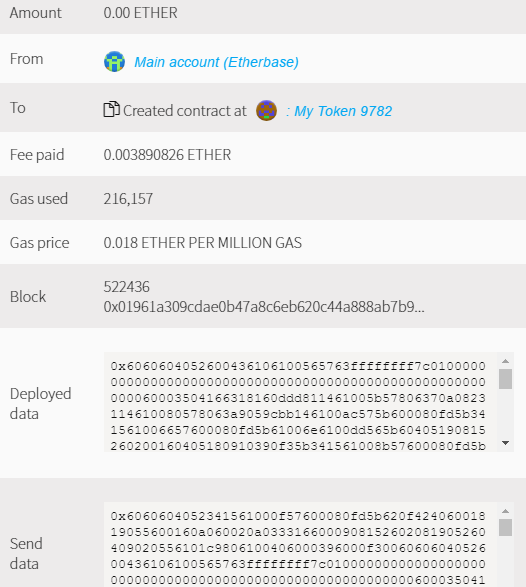
1. Then write your password and **“Send Transaction”**.



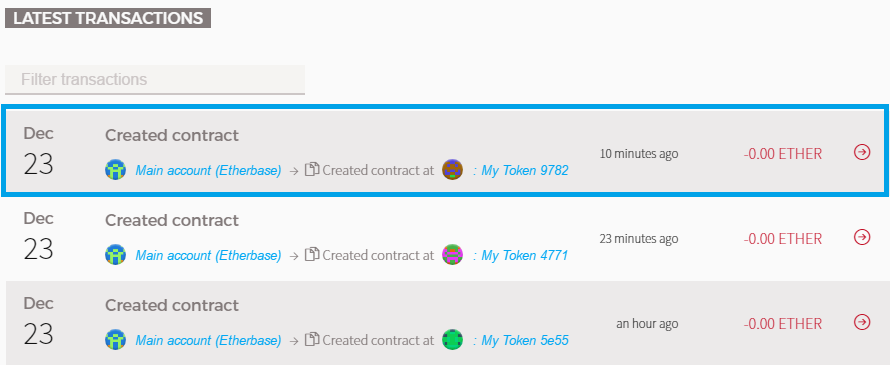
1. You can see your transaction at **“Wallets”** tab.



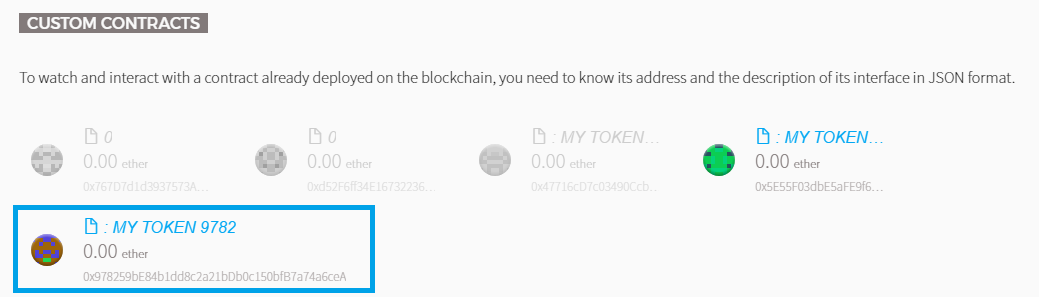
1. This is the information about the transaction.



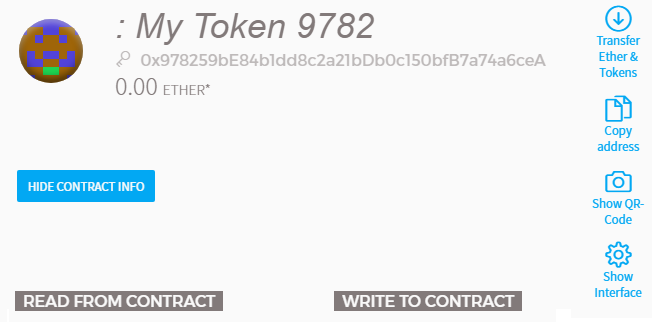
1. You should wait until the transaction is mined.



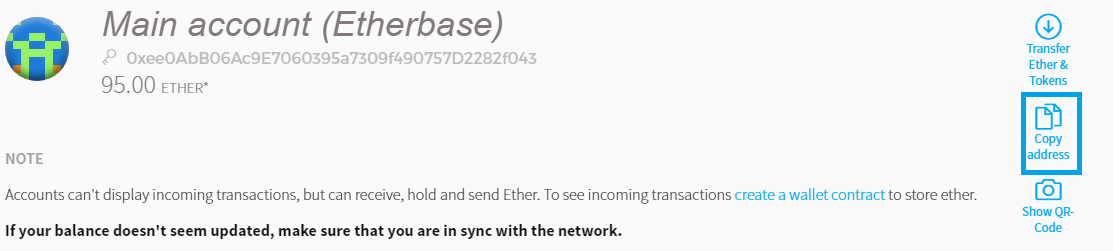
1. Then go to the **“Contracts”** tab and choose the token you created.



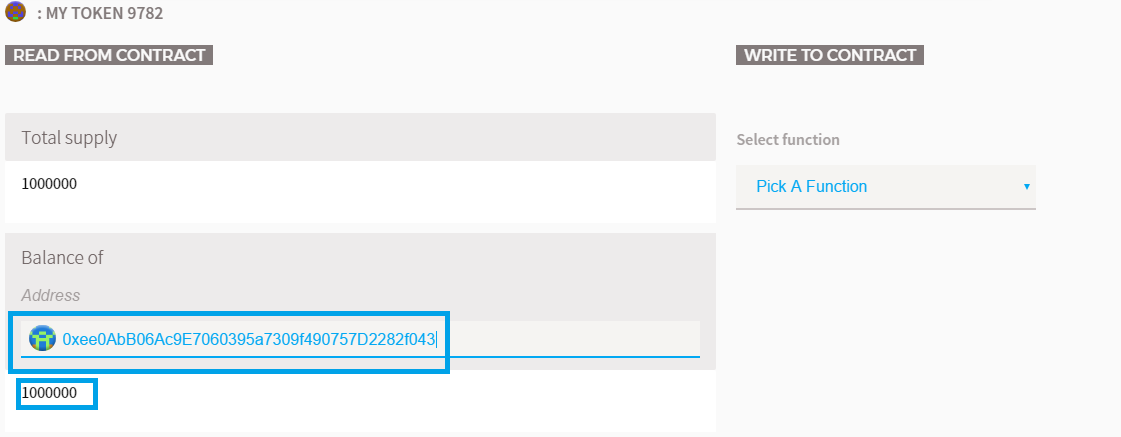
1. It should look like this.



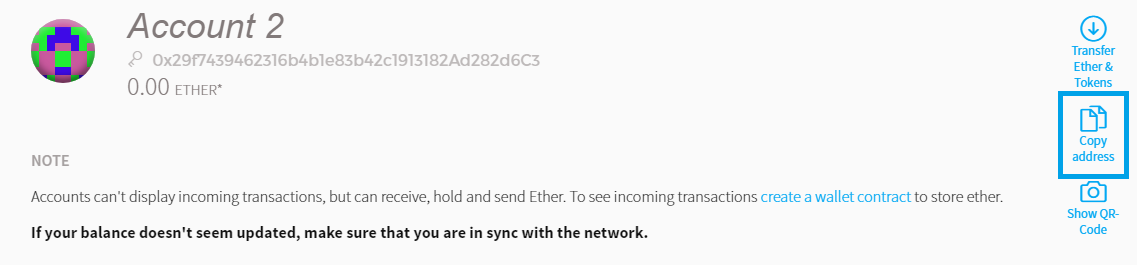
1. Go and copy your Main accounts address (account which created the contract).



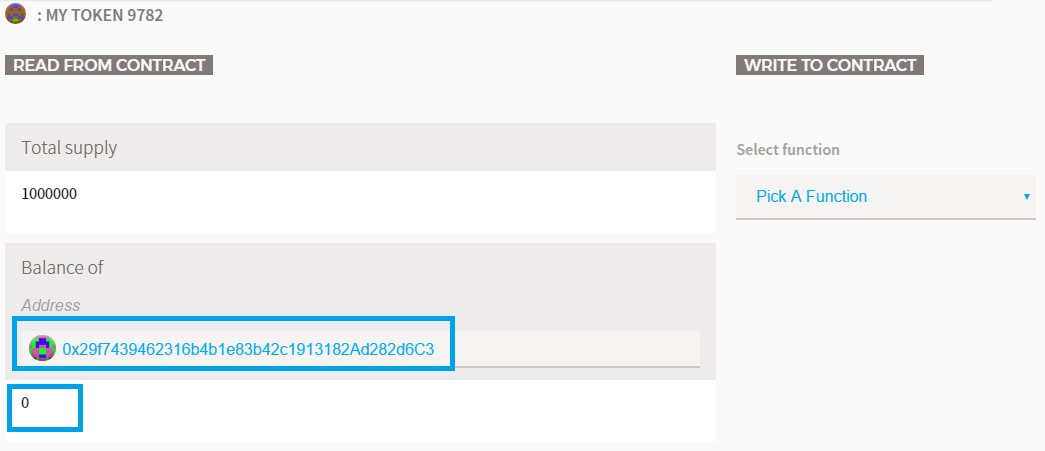
1. Paste the address to **“balanceOf**” place and it should have 100000. All the tokens go to the creator address balance.



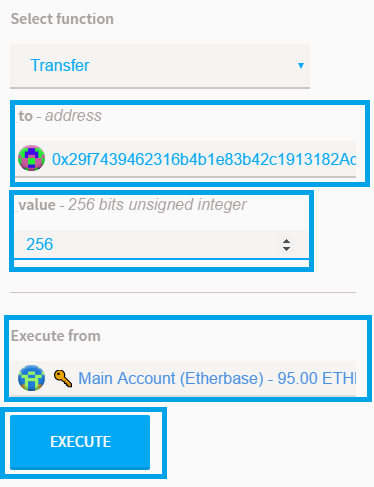
1. Go and copy the second account.



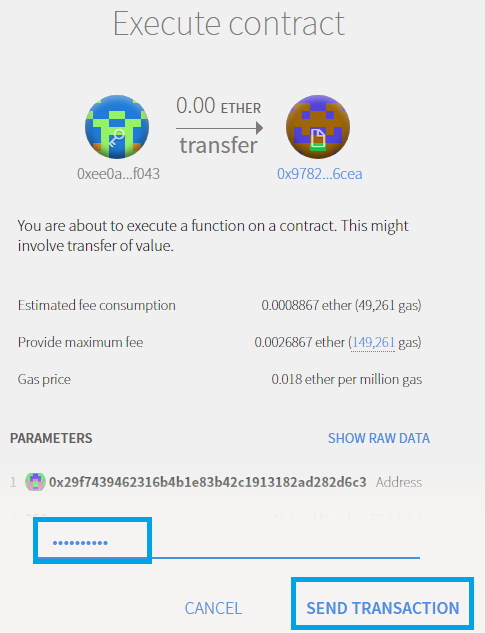
1. It should have 0 tokens balance.



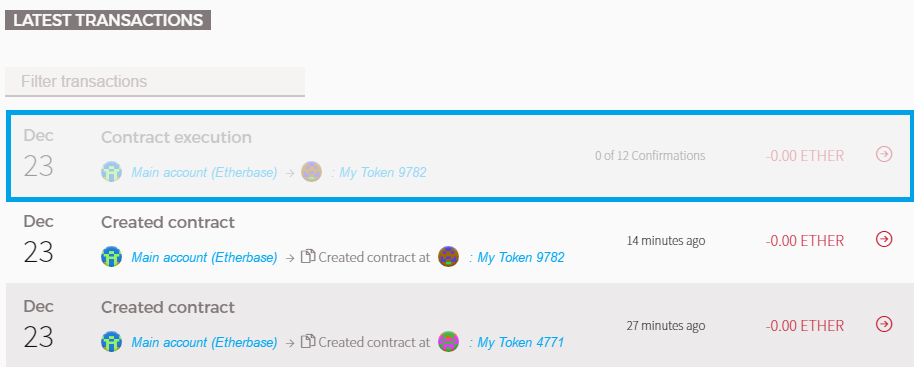
1. Select the only function of the token – **“Transfer”** and paste the second address to the **“to”** place and the amount to send to **“value”** place (e.g. 256 tokens). Choose the account to execute (Main account which have tokens) and **“Execute”**.



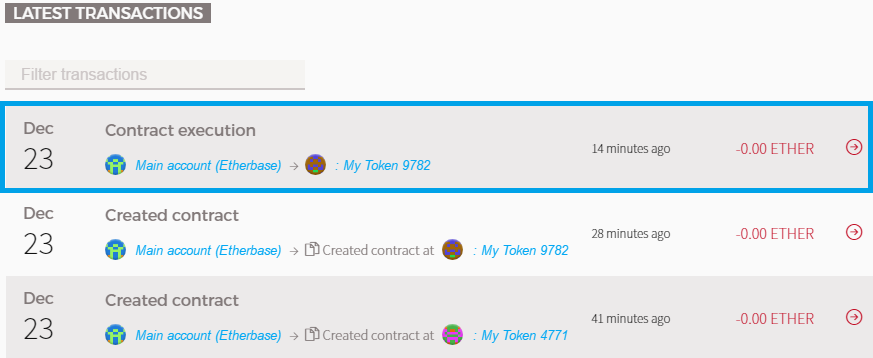
1. Write your password and **“Send Transaction”**.



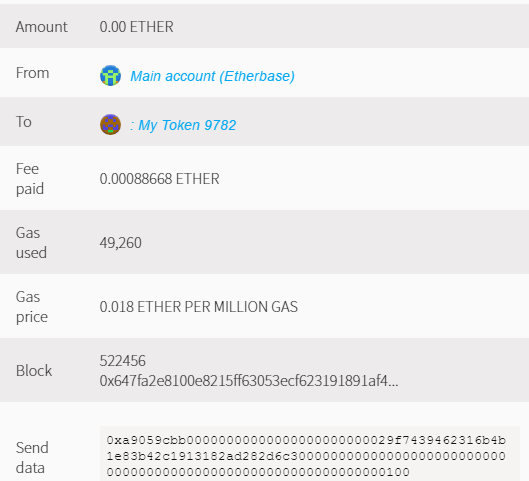
1. You can see your transaction.



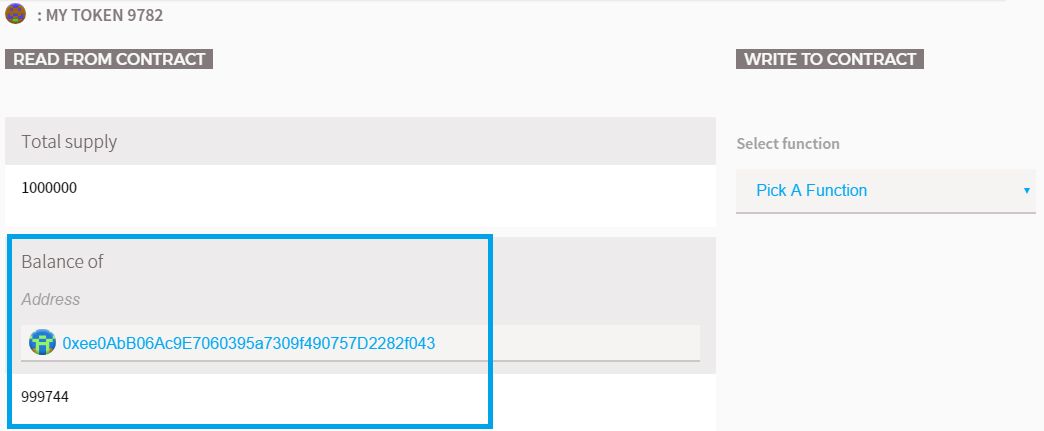
1. After it is ready you will see the following window:



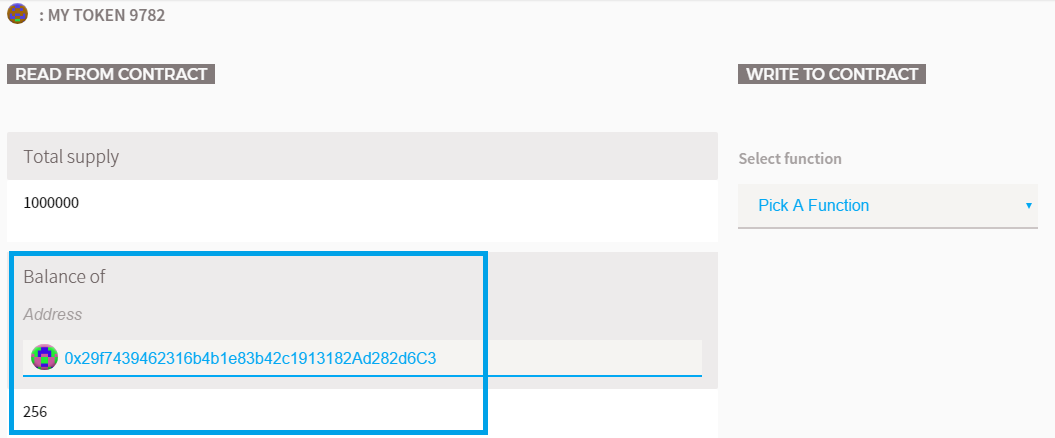
1. This is the information for the transaction:



1. Go again to the token and paste the main account address. You should see the following. The balance of the main account should be 999744 (after 256 sent).



1. And account 2 should have 256 tokens.



# What to Submit?

Submit as exercise outcome your “**Main account” and “account 2” addresses, screenshots of the ether balances of the addresses and token balances**. Examples:

