Blockchain Programming 2021 hands-on Ethereum*

Matija Piškorec

September 25, 2021

Contents

In these hands-on we will run our own private Ethereum network (UZHEthereum), mine some UZHETH, execute transactions and deploy ERC-20 tokens on it. These hands-on will mostly follow Luca Ambrosini's hands-on for the Deep Dive into Blockchain (DDiB) Summer School:

https://gitlab.uzh.ch/luca.ambrosini/go-ethereum/-/wikis/home

These hands-on will demonstrate how to run the command line commands for `geth` in Linux. However, the process should be very similar to MacOS and Windows as well.

1 Installing geth

In this section we will show how to install `geth` - a Go implementation of the official Ethereum protocol, which will allow us to run our own UZHEthereum node. Instructions for how to install `geth` are available here:

https://geth.ethereum.org/docs/install-and-build/installing-geth

For Ubuntu Linux you just include a custom PPA (Personal Package Archive) repository and install with `apt-get`:

```
sudo add-apt-repository -y ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install ethereum
```

In Arch Linux you can find it in official Pacman repository:

*Email `piskorec@ifi.uzh.ch`

```
pacman -S geth
```

For MacOS you can install it through Homebrew:

```
brew tap ethereum/ethereum
brew install ethereum
```

For Windows you can use precompiled binaries available here:

https://geth.ethereum.org/downloads/

Finally, check that your installation is correct by running:

```
geth --help
```

You can also see full list of `geth` command-line options, as well as examples of basic functionalities on the official Go Ethereum documentation:

https://geth.ethereum.org/docs/interface/command-line-options

2 Creating an account

Define a folder where you will store you account and create a new account:

```
geth --datadir ~/.uzheth account new
```

You will have to provide a password for it, the output should look something like this:

```
Your new key was generated

Public address of the key:   0x98Ea8a7503939b70C9dF58CF5b592B8bC8658B8C
Path of the secret key file: /home/matija/.uzheth/keystore/UTC--2021-09-24T09-40-59[...]

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

The `~/.uzheth` folder now stores the private key associated with your account - keep it safe as whoever has the access to this key and your password can fully control your account! You should also store the public address of the key mentioned in the output, you can do this in a simple text file.

You can check your newly created account with:

```
geth account list --datadir ~/.uzheth
```

More information on managing your accounts with `geth` can be found here:

https://geth.ethereum.org/docs/interface/managing-your-accounts

3 Initialize the UZHEthereum network with a genesis block

We will bootstrap our own UZHEthereum network by initializing our `geth` client with a completely new genesis block. In this way we won't share any common history with the Ethereum mainnet, but will still have full functionality of the Ethereum network itself. Download the official UZH Genesis block:

```
curl -O https://gitlab.uzh.ch/luca.ambrosini/go-ethereum/-/wikis/uploads/\
f36cd66fb248d69cdcaf6b5b27685d5b/uzheth.json
```

Initialize the node with the given genesis block:

```
geth --datadir ~/.uzheth init uzheth.json
```

We will now start to sync the node, but will run the process in a terminal multiplexer `screen` (which should be available by default in Linux and MacOS) so that we can detach from the terminal while process is still running:

```
screen geth --datadir ~/.uzheth --http --http.port 8545 --http.corsdomain "*" --http.vhosts "*"\
--http.api miner,eth,admin,net,web3 --networkid 702 --nodiscover --syncmode "full"\
--http.addr 0.0.0.0
```

Few useful commands while running screen:

```
- Detach a terminal screen with `Ctrl-a d`
- List all running screens with `screen -list`
- Reattach to the running screen with `screen -r`
```

When you stop the running process (for example with `Ctrl-c`) its screen will automatically terminate as well.

We are now running the UZHEthereum node but our node does not know about any other nodes in the network, so we have to add manually at least a couple of them. We will add nodes which we started on the UZH network just for his purpose. Open a new terminal and run the following commands:

```
geth --datadir ~/.uzheth --exec "admin.addPeer('enode://9e6935ba567720b330aaca94af80bf032ef8d\
0cb1b7cf8bd73c88320756639e08056f4f08c6a3837cde82a9b575ed26a7391d70fadfaec368e60841bb5654118\
@130.60.244.246:30303')"  attach http://localhost:8545
```

NOTE: These commands need to be run every time that you restart your `geth` client!

After a while your running `geth` client should report `Imported new chain segment` or `Imported new block receipts` messages, which means that it started to download the blockchain blocks from the UZHEthereum network. As this is not Ethereum mainnet but our own private network the synchronization should be quick as there are not that many blocks (full Ethereum blockchain has around 980 GB).

A web blockchain explorer for our UZHEthereum network is available here:

http://130.60.24.79:1234/?network=UZHETH

4 Setup a Metamask wallet

We already have a wallet which we created at the start of the process, but we have to interface with it over the command line using `geth`. Instead, we will be creating a Metamask wallet which can be used through a web browser extension. Metamask is available for all major web browsers (Firefox, Chrome) on desktop and mobile, you can install it from this website:

https://metamask.io/download.html

After you add Metamask to your browser make sure you set it up - you will have to backup the recovery seed and set a password. Once you do this you can connect it to various Ethereum networks - Ethereum mainnet, Ropsten Test Network. We want to connect it to our own UZHEthereum network. In order to do this you have to go to `Settings->Networks->Add network` and fill in the following data:

```
- Network Name: `UZHETH`
- New RPC URL: `http://130.60.244.246:8545`
- Chain Id: `702`
- Currency Symbol: `UZHETH`
```

You just created your Metamask wallet and connected it to our UZHEthereum network! You can copy the address at the top of the menu and use it to receive some UZHETH coins!

5 Setup a miner

Before you run a miner you should make sure that you UZHEthereum node is fully synced and that you have your Metamask address ready. We will be communicating with our running `geth` node through an interactive JavaScript console.

First, attach to your `geth` node console:

```
geth --datadir ~/.uzheth attach http://localhost:8545
```

Before you continue make sure that your node is actually connected to other peers in the network, otherwise your transactions will not be relayed to the UZHEthereum network. Run the following command in the interactive console to see whether you have any output:

```
admin.peers
```

If the output is empty you have to run the earlier commands to add seed nodes manually. This time you can do it in the interactive console to which you are just attached (see the exact parameters in the earlier command, and repeat for all seed nodes):

```
admin.addPeer('...')
```

Also, you can check whether your client is still syncing and the current block number of the blockchain:

```
eth.syncing
eth.blockNumber
```

Now we can setup the miner. Set the address to where the mined UZHETH coins will be stored, use your Metamask address for this:

```
miner.setEtherbase('<Metamask address>')
```

Now you can start mining with the following command:

```
miner.start()
```

Wait for some time and then run the following command to check your current hash rate:

```
eth.hashrate
```

If the value is larger than zero this means you are successfully mining UZHETH coins and should see the balance in your Metamask wallet increase!

To stop the miner run:

```
miner.stop()
```

6 Execute transactions

Once you mine some UZHETH coins you can try to execute transactions. If you can obtain Metamask addresses from your colleges feel free to send them some coins, if not you can use the address in the wallet that you created earlier. You can check all the accounts that you have locally in the interactive JavaScript console:

```
eth.accounts
```

You can check balance of any account from JavaScript interactive console with `eth.getBalance` function. Remember that `eth.coinbase` is currently set to your Metamask address, so in order to be more explicit we will also store it in a separate variable by following JavaScript syntax:

```
var metamaskAddress = eth.coinbase
web3.fromWei(eth.getBalance(metamaskAddress), "ether")
web3.fromWei(eth.getBalance(eth.accounts[0]), "ether")
```

You can also insert any other address as a string directly into `eth.getBalance` function. Function `fromWei`[1] converts from Wei's to other units.

Now try to send some UZHETH from your Metamask wallet to your primary account and check the balance afterwards!

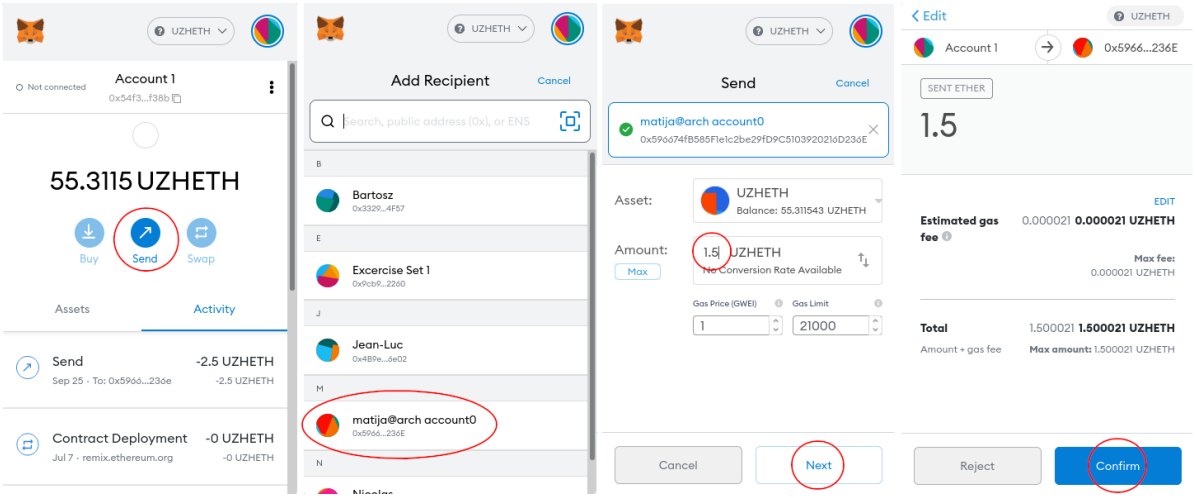

Figure 1: Send transaction in Metamask

7 Create and deploy your own token

In this section we will create and deploy our own ERC-20 token using a Remix - a web based IDE for writing and deploying smart contracts on Ethereum network using Solidity programming language:

https://remix.ethereum.org/

Click on the "New file" button and create a new file called `ERC20Basic.sol`. Now copy and paste the following snippet into the new file:

https://gitlab.uzh.ch/-/snippets/38

Feel free to change the `name`, `symbol` and `decimals` variables to your liking. I have named my token `UZHMatijaToken` and gave it a symbol `UZHMAT`.

Now go to the compile menu and select the proper version of the compiler, in our case this will be `0.5.17+commit.d19bba13`. You can now compile the token contract by clicking the `Compile` button. This will compile our token contract into the Ethereum Virtual Machine (EVM) byte code which can then be deployed on the Ethereum network.

[1] https://web3js.readthedocs.io/en/v1.2.11/web3-utils.html#fromwei

To deploy a contract, switch to the `Deploy` menu and choose `Injected Web3` as the environment. Make sure your `ERC20Basic.sol` is selected as the contract, then set the amount of tokens you want to create - I have chosen 88. You can now click on `Deploy` button to deploy your contract.

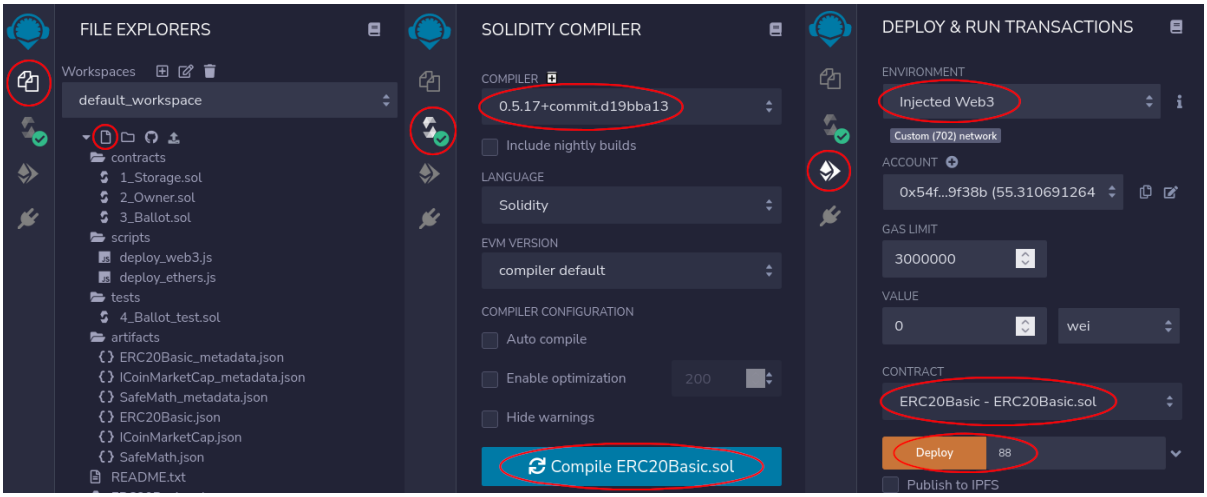

Figure 2: Deploying token contract in Remix IDE.

You will have to confirm the deployment with your Metamask. Once the deployment is finished a contract address will appear in the Remix IDE, along with all the functions which are exposed through the contract. In order to import the token to our Metamask wallet we have to copy the address which is associated with the contract.

Finally, you can add the token to your Metamask. On the main screen go to `Add Token` and paste the token address in the `Token Contract Address` box, then follow the interface to add the token.

Congratulations, you just created your own ERC-20 token and imported it into your Metamask wallet. You can send it to any UZHEthereum address in the same way that you sent Ether in the previous section - try it now!

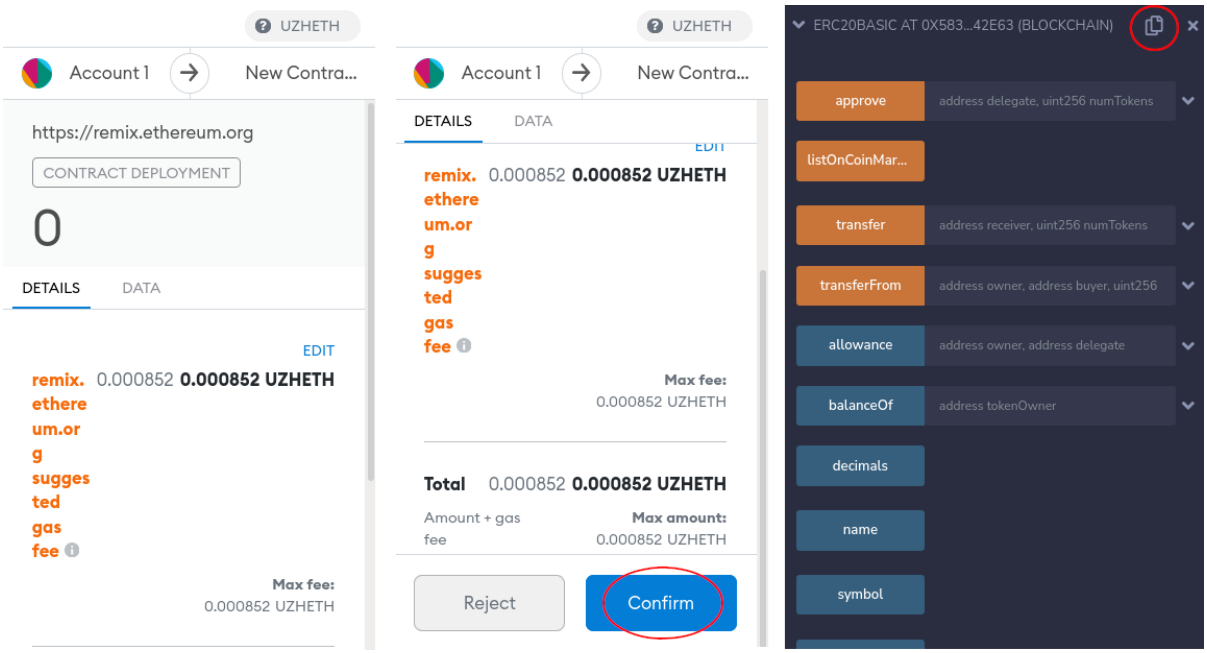

Figure 3: Confirm deployment of token contract on Metamask

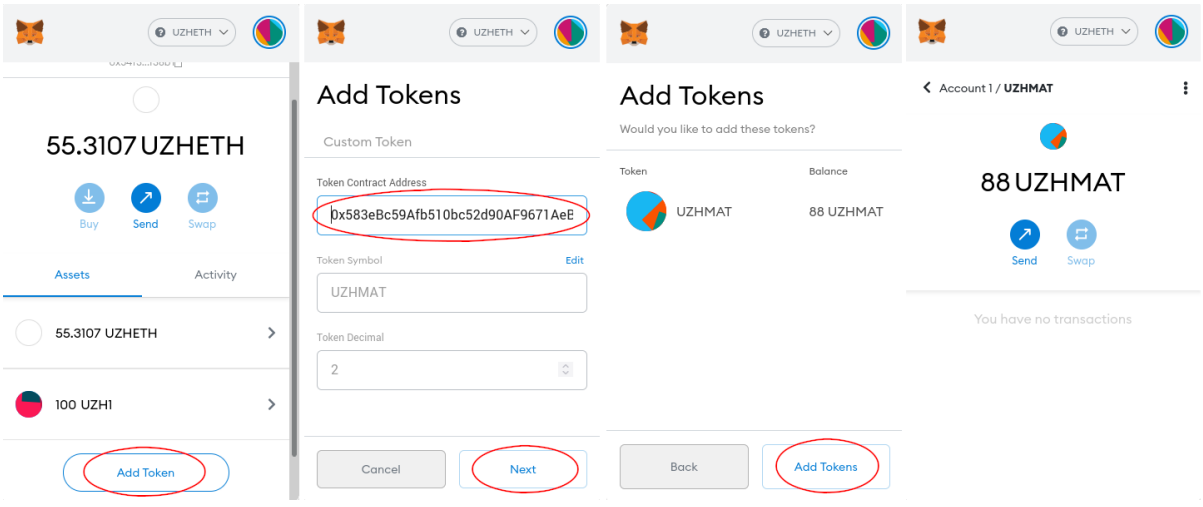

Figure 4: Add the token to Metamask.