Getting Started with Ethereum, Solidity, and Smart Contracts

This workshop will be centered around creating and deploying your first Ethereum Smart Contract using a private test network. A Smart Contract is code that is compiled and deployed to a block-chain network to create decentralized applications also known as dapps. The possibilities are endless as to what can be built with smart contracts; a prime example being the ability to create your own cryptocurrency. To follow along with this workshop, there are a few pre-requisite items that you will want to complete beforehand as these steps can prove to be time consuming. Below are the steps that you will need to complete if you want to code along with the workshop:

Installing the Geth Ethereum Client

For this workshop, we will be using the Geth Ethereum client to create our own private Ethereum network. The Geth client will allow us to create an account and deploy smart contracts to our private chain. See below for specific installation instructions:

For Ubuntu Users:

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

For Mac OSX Users (assuming you have homebrew properly installed):

```
brew install ethereum
brew install geth
```

For Windows Users:

All the commands in this tutorial are meant for a unix based system. Please follow the Windows installation here: *https://github.com/ethereum/go-ethereum/wiki/Installation-instructions-for-Windows*.

but know that you will have to translate the rest of this tutorial.

* If you are unable to run the 'geth' command in your terminal, you will have to locate the geth executable and add its location to your PATH environment variable.

Create Private Ethereum Network

First we need to create two accounts that will be used on our Ethereum chain. You will do this with the following geth command. (Do not forget the passphrases you chose, it is needed later.)

```
geth account new                    # Enter the passphrase for the first account
geth account new                    # Enter the passphrase for the second account
```

The above command should print each address, but if you need to see the account addresses again you can use this command.

```
geth account list
```

Now we create a folder in our filesystem to organize all of our files. Name it however you like, we will call it eth-dev and put it in our home directory. Each account will also needs its own folder as well.

```
mkdir eth-dev eth-dev/account1 eth-dev/account2
cd eth-dev
```

Next we need to create a file called genesis.json which will hold the initialization information for our private chain. We will want to insert the addresses we created above so it will pre-fund the account with plenty of ether.

```
# genesis.json
{
    “config”: {
        “chainId”: 1994,
        “homesteadBlock”: 0,
        “eip155Block”: 0,
        “eip158Block”: 0,
        “byzantiumBlock”: 0
    },
    “difficulty”: “400”,
    “gasLimit”: “200000000000”,
    “alloc”: {
        “{ACCOUNT ADDRESS 1}”: {
            “balance”: “100000000000000000000”
        },
         “{ACCOUNT ADDRESS 2}”: {
            “balance”: “10000000000000000000”
        }
    }
}
```

We can now initialize our block-chain using:

```
geth --datadir account1/myDataDir init genesis.json
```

Move the keys for the accounts you created to the new blockchain. Replace {eth_data_dir} with **~/Library/Ethereum in OSX, ~/.ethereum** in Linux

```
cp {eth_data_dir}/keystore/UTC--<rest of wallet 1 file’s name> account1/myDataDir/keystore
cp {eth_data_dir}/keystore/UTC--<rest of wallet 2 file’s name> account2/myDataDir/keystore
```

Start the geth client for your first account. (Reminder 2>> pipes stderr to a log)

```
geth --datadir account1/myDataDir --networkid 1111 --port 11111 --nodiscover console 2>>account1/myEth.log
```

```
> personal.listAccounts                                              # Shows the accounts on the chain
```

Now we are going to start up an instance of our second account in a new console.

```
cd eth-dev
geth --datadir account2/myDataDir init genesis.json
geth --datadir account2/myDataDir --networkid 1111 --port 11112 --nodiscover console 2>>account2/myEth.log
```

```
> personal.listAccounts                                        # Shows the accounts on the chain
```

Now, we're going to link the two nodes together. Go to your first node (first tab) and type:

```
> admin.nodeInfo.enode
```

This should give you an output that includes a hex value for enode. It will look something like "enode://2623e7b7e70...@[::]:11111?discport=0". Copy this value and use it in the next command in the other geth client.

```
> admin.addPeer({enode})
```

You should now be able to run the following in either tab and see the peer info for the other client.

```
> admin.peers
```

And see information about the other node.

Installing Python 3

If you already have Python 3 installed, you can move on to the final step, if not, you can follow the detailed instructions in the provided link below:

https://realpython.com/installing-python/

Needed Requirements

Below are the following Python module requirements you will need to follow along with the code:

- **web3 (4.6.0)**
- **py-solc (3.1.0)**

Come listen and enjoy this workshop on a very important and ever-upcoming piece of technology. We hope that you continue to research and learn about Ethereum, Solidity, and the endless unique technologies that can be built using smart contracts. If you run into issues getting started with this document, please feel free to reach out to one of the mentors at the Hack-The-U event.