Ethereum Basics: Setting Up a Development Environment

1 Setup

In this exercise sheet, you are asked to set up an environment for developing smart contracts on the Ethereum blockchain. While we advise you to use all tools we propose, you are free to use any tool you want to.

Recommended Software:

- Node.js - `https://nodejs.org/en/`
  An open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.
- NPM - `https://www.npmjs.com/`
  Node Package Manager (NPM) comes installed with Node.js.
- Truffle - `https://trufflesuite.com/docs/truffle/quickstart/`
  Truffle is a development framework for Ethereum, aiming to make life as an Ethereum developer easier.
- Ganache by Truffle - `https://trufflesuite.com/ganache/`
  A personal blockchain for Ethereum development that you can use to deploy contracts, develop applications, and run tests. It is available as both a desktop application as well as a command-line tool.
- MetaMask - `https://metamask.io/`
  MetaMask is an extension for accessing Ethereum-enabled distributed applications (dApps) in your browser. The extension injects the Ethereum web3 API into every website's JavaScript context, so that dApps can read from the blockchain. MetaMask also lets the user create and manage their own identities, so when a dApp wants to perform a transaction and write to the blockchain, the user gets a secure interface to review the transaction, before approving or rejecting it.
- Visual Studio Code - `https://code.visualstudio.com/`
- Visual Studio Code Extension: *Solidity*

Recommended Services:

- Infura - `https://infura.io/`
  Provides instant access over HTTPS and WebSockets to the Ethereum network. Alternative to running your own full node.

Task **Deploy a contract to the Rinkeby test network**

Rinkeby is an Ethereum test network that allows for blockchain development testing before deployment on Mainnet, the main Ethereum network. There are also other testnets available for Ethereum which have different properties than Rinkeby.

- `https://www.rinkeby.io/#stats`
- `https://rinkeby.etherscan.io/`
- `https://ethereum.org/en/developers/docs/networks/`

2 Setting Up a Rinkeby Account

- Install MetaMask.
- Configure your MetaMask account (Write down your seed phrase! You will need it later).
- Open MetaMask and switch the network to Rinkeby.
- Find some Rinkeby Test Ether using a Faucet like `https://rinkebyfaucet.com/` or ask your fellow students for a small donation. For this exercise, 0.1 Rinkeby Ether will be sufficient.
- Send 0.001 Rinkeby Ether to 0xaCb576A3f9F9ffa1e2276A1C06C0Ade4fE2197f8 (our address on Rinkeby) and inspect your transaction on Rinkeby Etherscan.

3 Setting Up Your Local Development Environment

- Install Visual Studio Code.
- Install the Solidity extension inside Visual Studio Code.
- Install Truffle globally on your computer: `npm install -g truffle`
- Create an empty working directory and initialize a bare Truffle project with no smart contracts in it.
  - Run `truffle init`
  - **Contracts**: Directory for Solidity smart contracts. *Migrations.sol* is a default Truffle contract
  - **Migrations**: Directory for scriptable deployment files. *1_initial_migration.js* is a default Truffle deployment script
  - **test**: Directory for test files for testing your application and contracts
  - **truffle-config.js**: Truffle configuration file
- Add the following Solidity smart contract (*Greeting.sol*) under your contracts directory.

```
//SPDX-License-Identifier: Unlicense
pragma solidity ^0.8.0;

contract Greeter {
    string private greeting;

    constructor (string memory _greeting) {
        greeting = _greeting;
    }

    function greet() public view returns (string memory) {
        return greeting;
    }
}
```

- Compile your smart contracts `truffle compile`

- You should now see your contracts' ABIs under the *build* folder.
- Add the following script (*2_deploy.js*) under your *migrations* folder. This script tells Truffle to deploy the Greetings smart contract.

```
const Greeter = artifacts.require("Greeter");

module.exports = async function (deployer) {
  //deploy Greeter contract
  await deployer.deploy(Greeter, "Hello!");
};
```

4 Deploying to the Truffle Built-in Blockchain

- Run `truffle develop`
- Check the console output. See all the available accounts and their associated private keys.
- Run `migrate` to deploy the smart contracts.
- Check the metrics of the deployment of the Greeter contract. Find the responsible account address and the used gas amount.
- Exit with `ctrl+c`

5 Deploying to the Local Ganache Network

- Install Ganache `https://trufflesuite.com/ganache/` and run the desktop application.
- See all the available accounts using the Ganache GUI.
- Run `truffle migrate`
- Using Ganache, find which address deployed the smart contracts. Find the transaction that includes the creation of the Greeter contract.

6 Deploying to the Rinkeby Test Network

- Open a free **Infura** account and create an Ethereum project. Find your Rinkeby endpoint.
- Run `npm i dotenv`. We will store our environment variables (private key and Infura api key) using this library.
- Create *.env* file under the root directory with following content.

```
PRIVATE_KEY_1 = <YOUR_PRIVATE_KEY>
INFURA_API_KEY = <YOUR_INFURA_API_KEY>
```

You can find your private key by clicking the Account details option on MetaMask.

- Edit the truffle-config.js file to deploy to the remote networks like Rinkeby. By default, Truffle is configured to deploy to the local Ganache network. In the provided configuration, we add Rinkeby as another network that we can interact with.

```
require("dotenv").config();
const HDWalletProvider = require("@truffle/hdwallet-provider");

module.exports = {
  networks: {
    development: {
      host: "127.0.0.1", // Localhost (default: none)
      port: 7545, // Standard Ethereum port (default: none)
      network_id: "*", // Any network (default: none)
    },
    rinkeby: {
      provider: function () {
        return new HDWalletProvider({
          privateKeys: [process.env.PRIVATE_KEY_1],
          providerOrUrl: `https://rinkeby.infura.io/v3/${process.env.
              INFURA_API_KEY}`,
          numberOfAddresses: 1,
        });
      },
      gas: 5000000,
      gasPrice: 5000000000, // 5 gwei
      network_id: 4,
    },
  },

  // Configure your compilers
  compilers: {
    solc: {
      version: "0.8.11", // Fetch exact version from solc-bin (
          default: truffle's version)
    },
  },
};
```

- Run `npm i @truffle/hdwallet-provider` to install the truffle-hdwallet-provider. This library handles the account management process (e.g., configuring the account to sign the transactions to deploy the smart contracts) and network connection. When we deploy to Ganache, Truffle uses the first available account by default for issuing the deployment transactions.
- Run `truffle migrate –network rinkeby`
- Once the deployment is done, check out MetaMask to see how your balance changed.
- Check if the account listed on the console for deploying the smart contracts matches your public key on MetaMask.
- Find the transaction hash of the Greeter contract deployment and search it on `https://rinkeby.etherscan.io/`.
- Navigate to the contract page and find the bytecode. Check if it is the same with the bytecode in the ABI of the Greeter contract (*build/contracts/Greeter.json*).