Course 1 Project

Create and Transact on Ethereum Private Blockchain

Introduction

Course 1 covered the basic blockchain concepts, structure, operational details, algorithms and techniques. We explored some of the blockchain concepts such as hashing and public-key cryptography in the quizzes associated with the module1, 2 and 3. In this course project, we will explore the following operations on a blockchain: create nodes on a private Ethereum blockchain, create accounts, unlock accounts, mine, transact, transfer Ethers, and check balances.

Approach

We have provided a virtual image (VM) that has a preinstalled Ethereum client (Go Language Ethereum – geth) and graphical user interface (GUI) to invoke the commands to accomplish project tasks. Please understand that this is not a simulation. You are actually working on local Ethereum nodes on your laptop/ computing device. We have used a virtual image so that the project provides same instructions irrespective of the computing systems that our diverse learners are working on. It does not require you to use command line interface (CLI) commands, since the commands are encoded as buttons. This lab provides quick and easy accessibility to the commands. If you are curious about the list of commands we used for this lab, it is provided at the end of this lab description.

Learning outcomes

Demonstrate that you can follow the steps to transact on a blockchain, with the concepts learned in the course such as account, account balance, node, peers, peer-to-peer transaction, genesis block, ethers, mining and interacting with a private blockchain.

Project Implementation details: What to do?

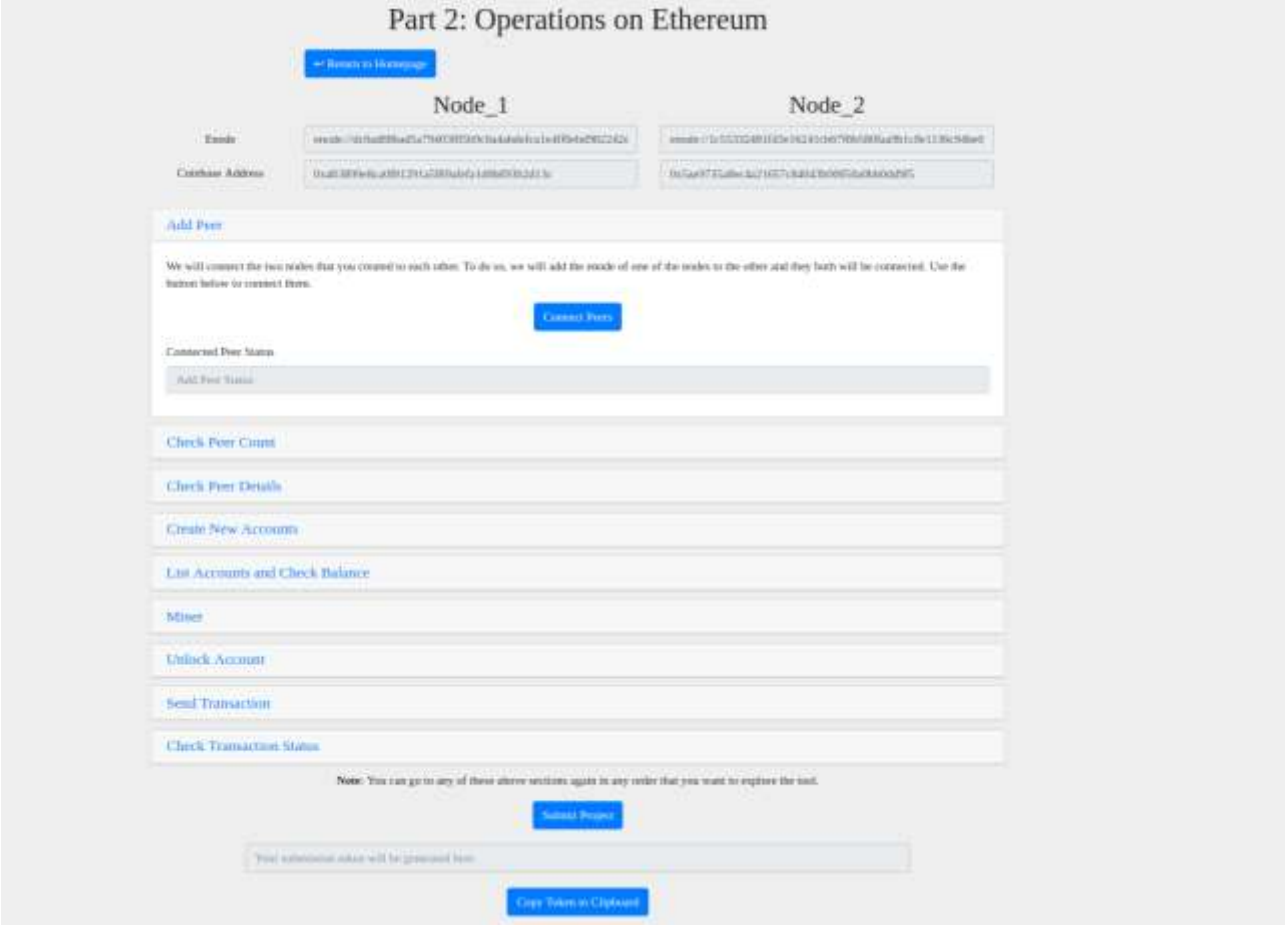
In this project, you will deploy an Ethereum test blockchain with two nodes and transact between them. We will do it in two steps, (1) setting up the environment and (2) working with the blockchain provided in Part 1 and Part 2. **Read all instructions below before you begin the project.**

1. **Setting Up the Environment**
   Please refer to environment setup file provided in Step 1 of the Instructions tab. Follow the instructions to setup and get started. DO NOT miss this step. This document also has troubleshooting tips in case you run into problems.

2. **Initializing and Starting the Ethereum Nodes**
   If you successfully complete the previous step, you will be seeing a web interface with brief instructions in each step. You can complete the exercise by reading through the instructions provided on the web interface. Complete all the steps indicated.

3. **What are we doing in Part 2?**
   We are creating two nodes, connecting them as peers, check peer details, create some more accounts at the nodes, check the balances to be 0 to start with; starting a miner will add (miner fees) to the balance of the coinbase account of the miner node; then we transact between the account with the balance (first node) and to any account in the second node or first node; You also learn that you need

to unlock the accounts before you transact, and can examine the status of the transaction (pending, completed etc.). These steps are shown in the screenshot below.



4. **Passwords for the nodes**
   For this exercise, use simple passwords (say, b for node1, c for node2) so that you will remember them when you need to unlock the accounts. If you forget the passwords, you have to redo the steps to create the accounts all over again. Of course, in a production blockchain, you will have to use strong passwords that others will not be able to guess.

5. **Read the instructions and explanations** at each step of the interface. You need to click on the buttons in each of the steps to execute the operation. Learn as you go. Associate the steps you are executing with lessons in the video. Execution of each operation takes some time depending on the resources (such as RAM memory) on your computer. Be patient. **Pay attention to the notifications request you to wait until the operation is completed.** We have allocated arbitrarily long/generous duration for each operation to complete. This is to accommodate any computation power variances of the learners.
   In case your interaction stalls due to some reason please do not hesitate to go back and restart. There are many options for restart, partial or full restart from the beginning. Grading is based on the steps you complete and not how many times you restarted. You can refer to troubleshooting tips in case you run into any issues.

   Observe the enode values, address[0] or coinbase address of accounts, peer details in JSON format, creation of accounts in a single node and miner operations. Genesis block is the very first block of a blockchain. You will observe DAG or directed acyclic graph generation when a miner is started. This process is needed for the memory based proof of work consensus of Ethereum.

6. **Starting and Stopping the Miner**

   In this project, there is only one miner that is started on the first node. Do not stop the miner until you complete the transactions. Recall from the lesson a miner is needed to confirm your transactions. The transactions will never be confirmed if you do not have a miner running.

7. **Grading and Submission**

   We are using auto-grading for grading your project. You are assessed for the steps you completed. The grade does NOT depend on the number of attempts. Also, it does NOT depend on the time you took to complete the project. When you complete the project you will get a unique hash value based on the steps you completed. Please submit that to Coursera learning system as directed to get your grade for the project. Save the hash generated into a text file (.txt file) and submit this file. You will get an error if you simply submit the hash. See this link if you need help: https://learner.coursera.help/hc/en-us/articles/209818753

8. **Exploring beyond**

   This project is always available for you to explore and learn. Do not hesitate to explore further and try different transactions. We have provided only one miner. You can always restart the VM and try out various operations including CLI commands given at the end of the document. The same VM will be used in the Course 3 for development of a Decentralized Application or Dapp.

Summary

In this exercise, you got a feel for setting up your own Ethereum client and working on some basic operations. We strongly encourage using the Online Discussions to discuss the project with other students and to resolve any problems you may face.

In the next two courses, you will get to design and develop applications for the Ethereum blockchain.

Extra Material

In this project, we have already installed geth in the "Ethereum Ubuntu.ova" virtual appliance provided to you. To make things easier, we have provided a web interface to help you out in interacting with the geth instance running in the backend. If you are curious here are commands behind those buttons you clicked. We will learn more details about geth commands and the supporting APIs in Course 3.

1. Create new Accounts: geth account new --datadir ~/Node_1 and geth account new --datadir ~/Node_2

2. Initialize Genesis File: geth init customGenesis.json --datadir ~/Node_1 and geth init

customGenesis.json --datadir ~/Node_2

3. Start Geth:

NODE_1:

geth --datadir ~/Node_1 --maxpeers 95 --networkid 13 --nodiscover --rpc --rpccorsdomain "*" --port 30301

--rpcport 8544 --rpcapi="txpool,db,eth,net,web3,personal,admin,miner"

NODE_2:

geth --datadir ~/Node_2 --maxpeers 95 --networkid 13 --nodiscover --rpc --rpccorsdomain "*" --port 30302

--rpcport 8545 --rpcapi="txpool,db,eth,net,web3,personal,admin,miner"

4. Connect Peers: admin.addPeer(<eNode of the other node>);

5. Peer Count: net.peerCount

6. Peer Details: admin.peers

7. Create New Accounts: personal.newAccount(<password>)

8. List accounts: eth.accounts

9. Check Balance: eth.getBalance(<account>)

10. Start Miner: miner.start(4)

11. Stop Miner: miner.stop()

12. Unlock Accounts: personal.unlockAccount(<address>,<password>,<duration in sec>)

13. Send Transaction: eth.sendTransaction({from:<address>, to:<address> , value: <value>})

14. Check Transaction Status: txpool.status