# Exercises: Play with a Simple Blockchain with PoW Mining

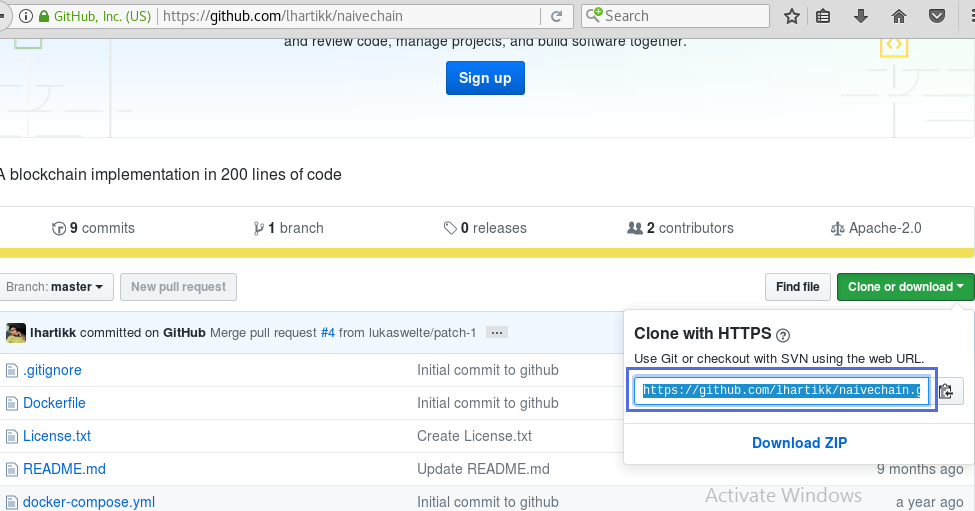
The goal of this exercise is to understand the principles of **proof-of-work mining**. The exercise is continuation of the previous one, where two nodes of Naivechain app were configured and started successfully.

These exercises are based on this project: <https://github.com/lhartikk/naivechain>. Thanks to the original authors.

For this exercise you will need **Linux** or Linux-like command-line environment. We use “Debian” based distribution, but each other will work fine in most cases. Also, you will need "**node.js**", "**npm**" and "**cURL**" installed.

## Clone the “Naivechain” Project

Clone the GitHub project <https://github.com/lhartikk/naivechain>. Prepare the project for run like it is explained in previous exercise.

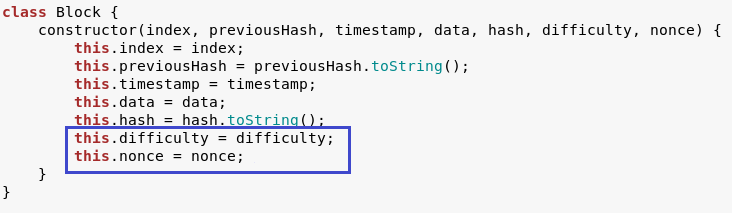


## Implement Proof-of-Work Mining

1. **Open** the file “**main.js**” in your preferred text editor or IDE. Modify the code and **implement Proof Of Work**:
2. First we must create variable for **difficulty**. This is the requirement **number of zeroes** in the beginning of the current block **hash**.



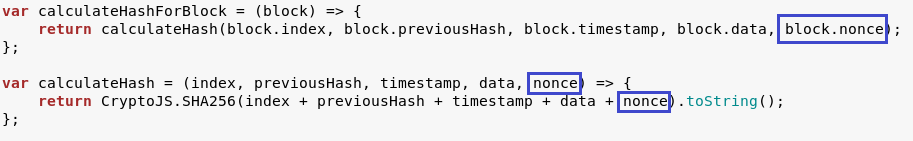
1. Next, we must add the **nonce** property and the **difficulty** property in the **Block** constructor. Nonce is the number which we will **increase** in attempt to find the appropriate hash and **mine the block**. Difficulty is important because without its value nobody is able to proof is founded hash valid or not.



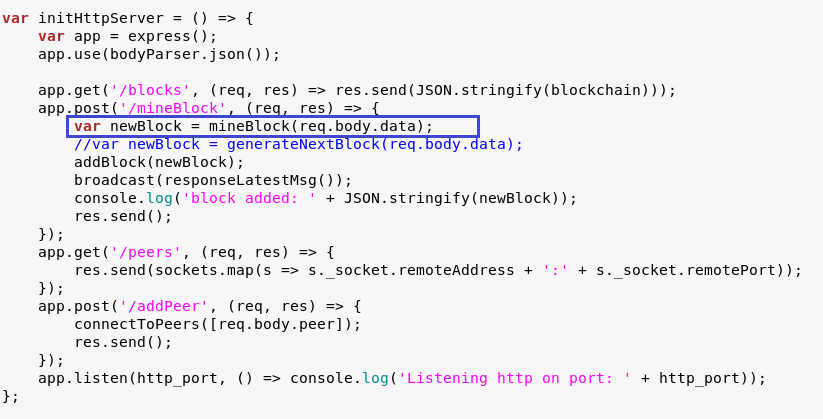
1. For now on we will add **nonce** and **difficulty** in every block our program create. In this case we modified **function** which **creating hardcoded genesis block** like add **nonce** and **difficulty** withzerovalues.



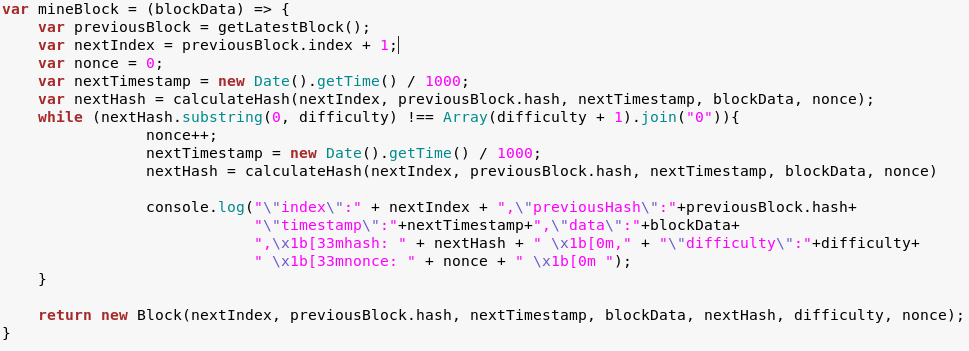
1. Then we must modify functions **calculateHashForBlock** and **calculateHash**.



1. Without Proof of Work mechanism when executing command “**mineBlock**” we just created new block. But now we need new **function “mineBlock”** responsible for mining with Proof Of Work.



1. This is how function **mineBlock** looks like. It receives the **block info** and in any **cycle** iteration **check** if generated **hash** satisfied the **difficulty** requirements. If the answer is “no” we increment **nonce**, if “yes” then return new successfully created **block**.

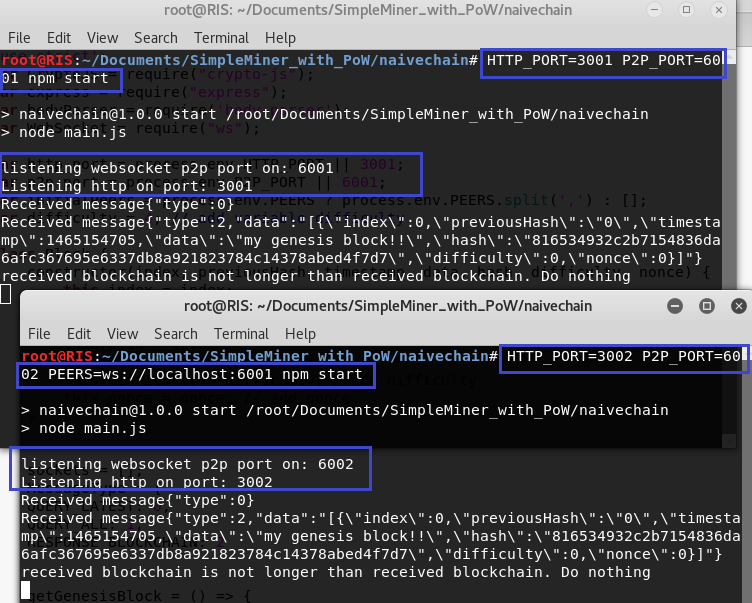


1. You can **download** the **main.js code** from <https://github.com/SMandazhiev/Simple-Miner-with-Proof-of-Work-in-JS/blob/master/main.js>

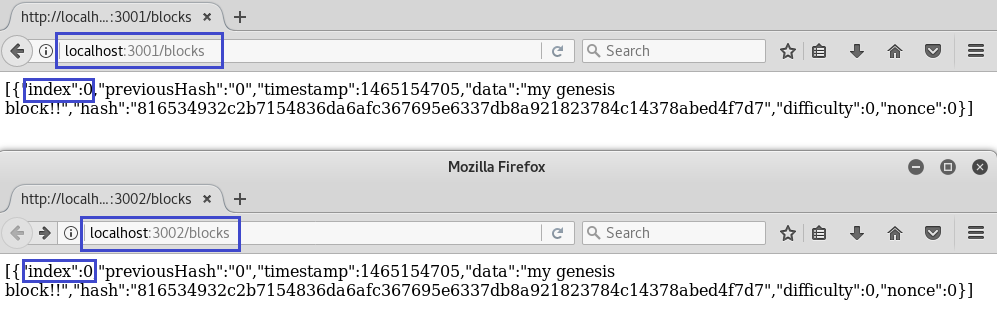
## Run Two “Naivechain” Nodes and Run PoW Mining

1. Run two naivechain nodes at ports:

* REST: 3001, P2P:6001
* REST: 3002, P2P:6002



1. Open the two nodes blockchains is browser. On browser write “**localhost:3001/blocks**" and “**localhost:3001/blocks**". Here is only the hardcoded **genesis** block with **index 0**.

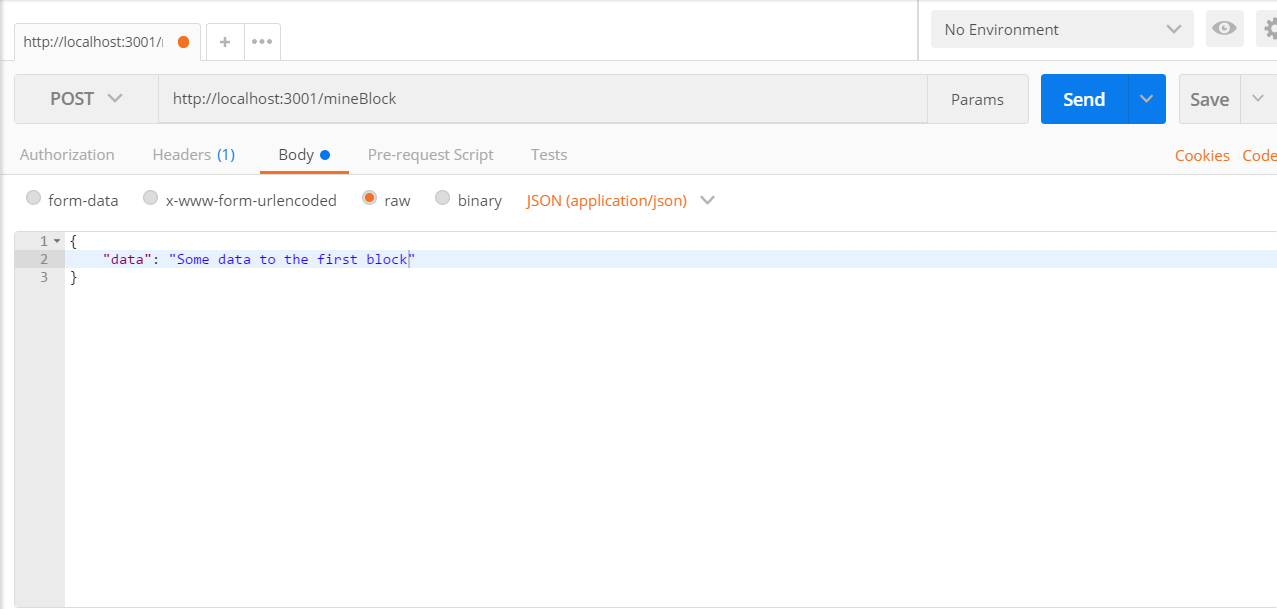


1. Now is time to mine. Open new console and type: "**curl -H "Content-type:application/json" --data '{"data" : "MyInfo 1"}' http://localhost:3001/mineBlock**". With this we command **first node** to **mine new block**.

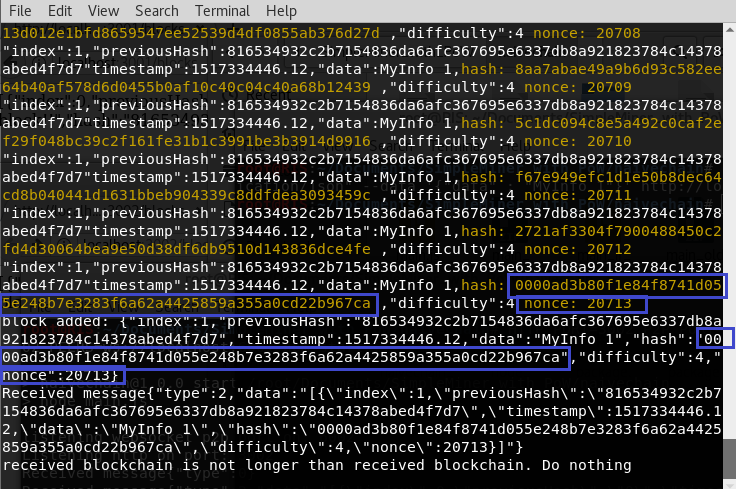
**Linux:**

|  |
| --- |
| curl -H "Content-type:application/json" --data '{"data" : "Some data to the first block"}' http://localhost:3001/mineBlock |

**Windows:**



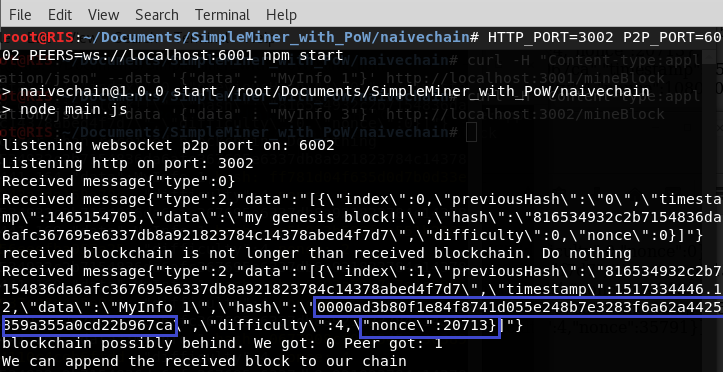
1. Look at the **first block console**. The miner start mining and **increment nonce** searching for the **hash satisfaing the difficulty** requirements.



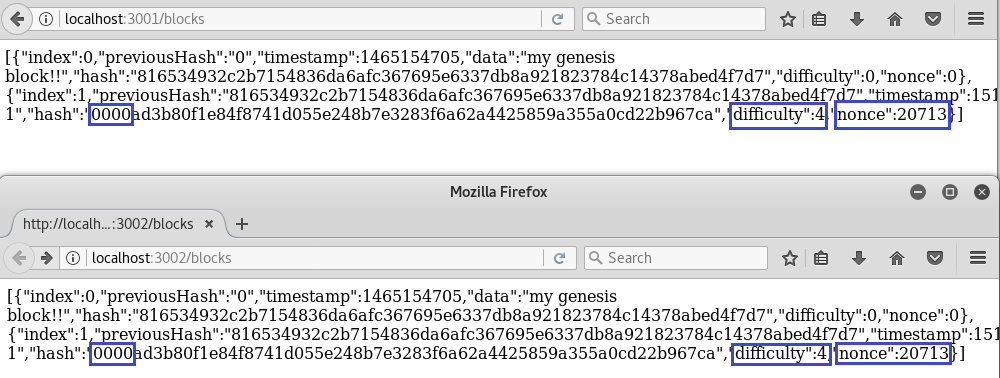
Note the following results:

* Block mining **takes some time** (a few seconds). This is due to the proof-of-work algorithm.
* The last block hash has **several leading zeroes** depends on **difficulty**. This is the proof-of-work result.

1. The second node also accepted new block in blockchain.



1. Look at the nodes in the browser. Here is the new block with **several leading zeroes, difficulty** and **nonce**. With this info the founded hash can be checked by another miners.



## Follow the Described Steps

Your task is follow the tutorial described above and perform the experiments.

1. Create nodes.
2. Mine some blocks.
3. Try different difficulty.
4. Play with code.

# What to Submit?

Create a **zip file** (e.g. your-name-simple\_miner\_with\_proof\_of\_work-exercise.zip) holding the screenshots with your experiments. Make screenshots of terminals and data in browser.

Submit your **zip** file as **homework** at the course Web site.