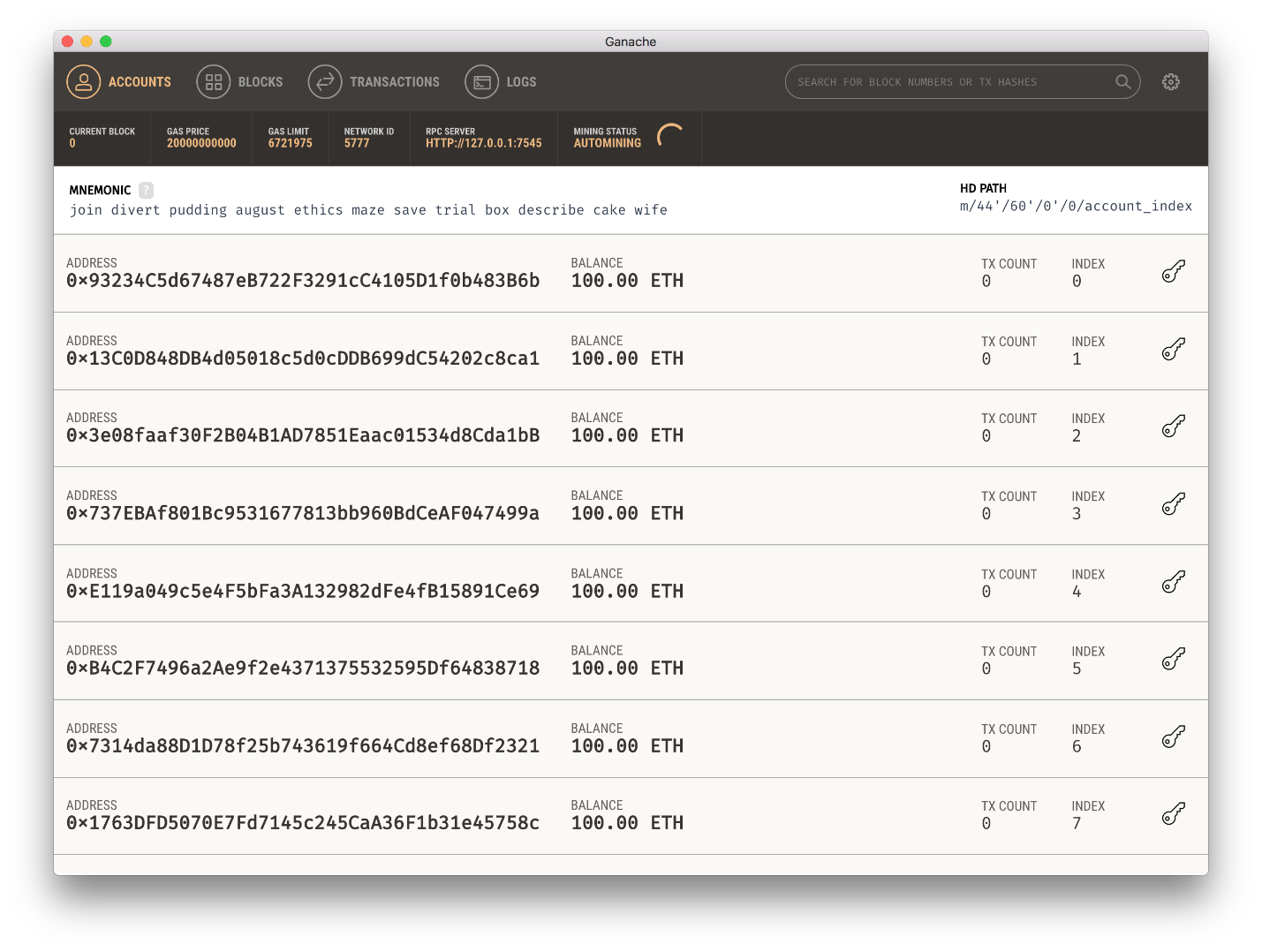
## Procedure / Steps

### 1: Start Ganache

When Ganache is started, you will see 10 Ethereum addresses automatically generated for you. These are the wallets that we can test against later.

By default, Ganache starts at <http://127.0.0.1:7575>. Take note of this address.



### 2: Initialize Truffle

Create a new folder for our demo project and initialize Truffle Framework inside. Open your terminal and type:

cd ~/Ethereum

mkdir demo

cd demo

truffle init

Once it is done You will be able to see a couple of files being created by Truffle in the demo folder.

### 3: Create Demo Contract

Open up the demo project in your favorite editor, create a new file called Demo.sol inside the contracts folder. Demo.sol codes:

pragma solidity ^0.4.23;

contract Demo {

uint public balance;

// Initialize global variables

constructor() public

{

balance = 0;

}

// The payable keyword allows this function to accept Ether

function contribute() public payable

{

// msg.value is the value of Ether sent in a transaction

balance += msg.value;

}

}

Here, we have a very simple smart contract where it has a contribute function that takes in Ether and update its balance.

### 4: Add a New Migration

In order to deploy our Demo contract to our test Ethereum network on Ganache, we need a migration script.

Inside the migrations folder, create a new file called 2\_demo\_migration.js. The 2\_ prefix is important so that Truffle knows it's step 2 of all the migration steps for this project.

Enter the following code:

var Demo = artifacts.require("./Demo.sol");

module.exports = function(deployer) {

// Demo is the contract's name

deployer.deploy(Demo);

};

### 5: Update the Configurations

Open up **truffle-config.js** , enter the following code:

module.exports = {

networks: {

development: {

// from: "", // Defaults to first address from Ganache

host: "127.0.0.1",

port: 7545,

network\_id: "\*"

}

}

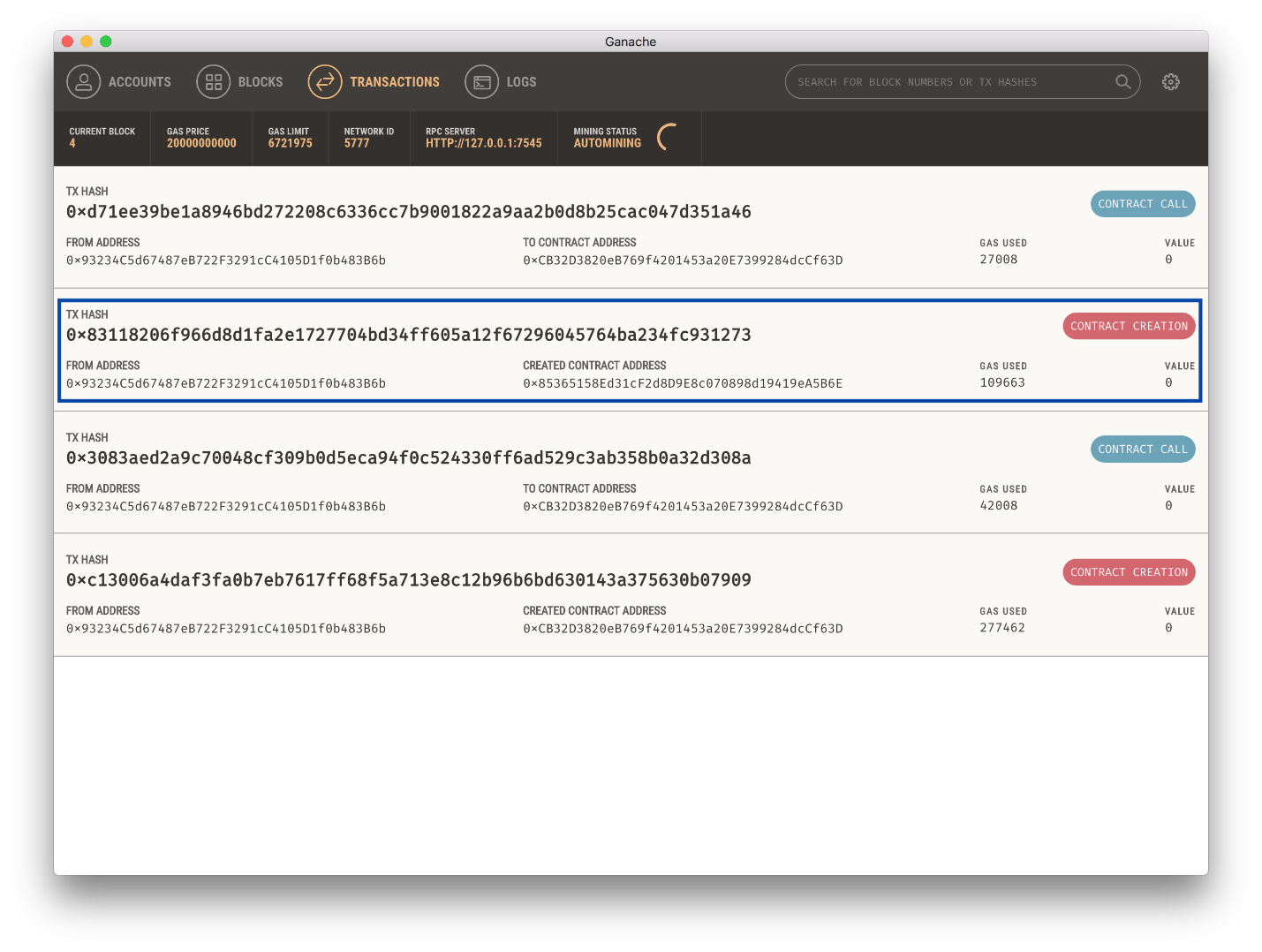
};

### 6: Deploy!

Now we are ready to deploy to Ganache for testing. Open up your terminal and type:

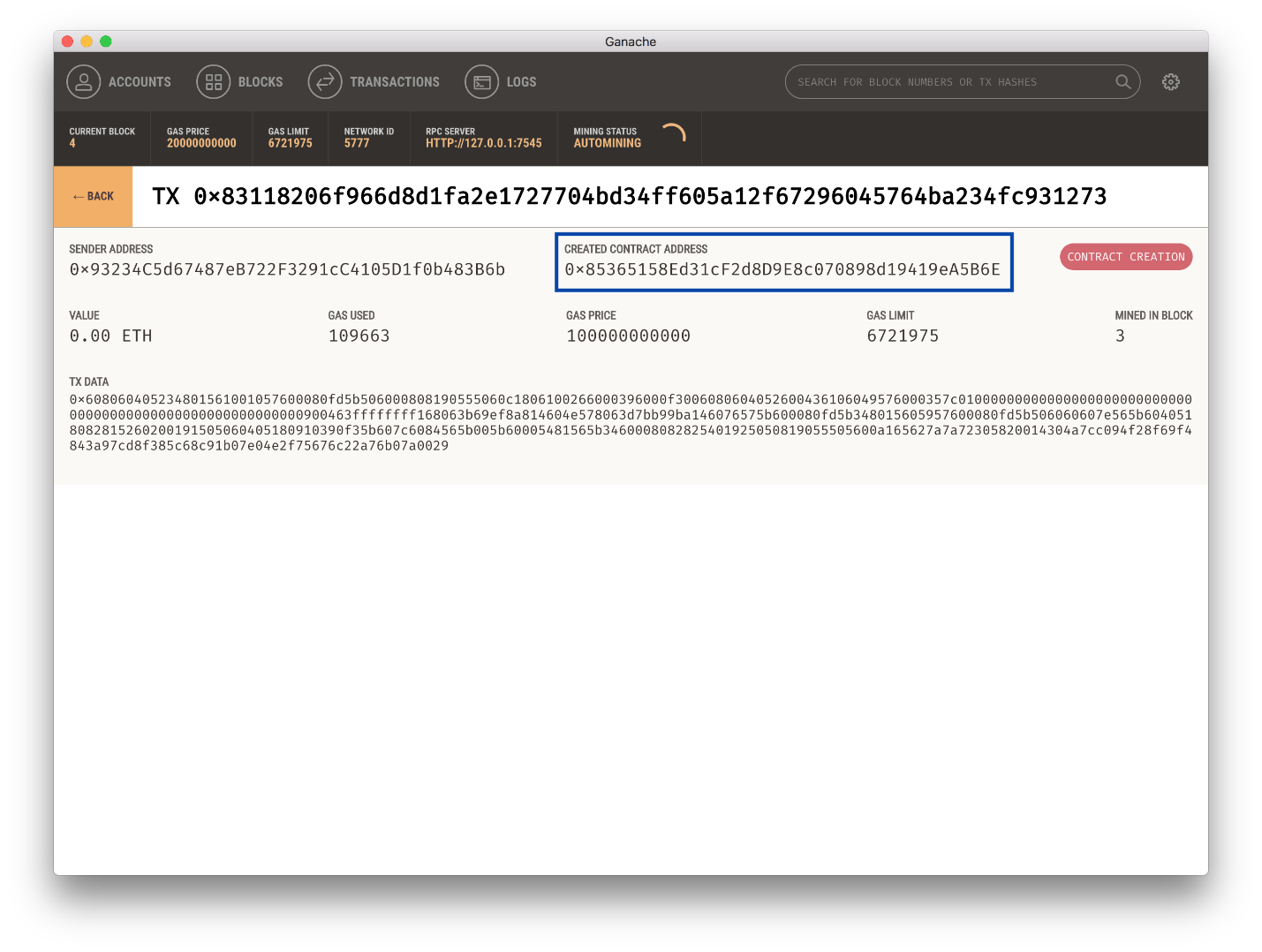
truffle deploy

Once it is done, let us check our Ganache. You should see a few transactions being recorded:



2 contracts are generated by our deployment due to us having 2 migrations, with 1 being the default Truffle Framework's migration.

For our Demo contract, we need to check the second transaction from the top (circled in blue). Click on it and copy the contract address (circled below) for our next step.



## Interact with Contract

### 7a: Check the Balance

Now let us check our balance, which supposed to be 0:

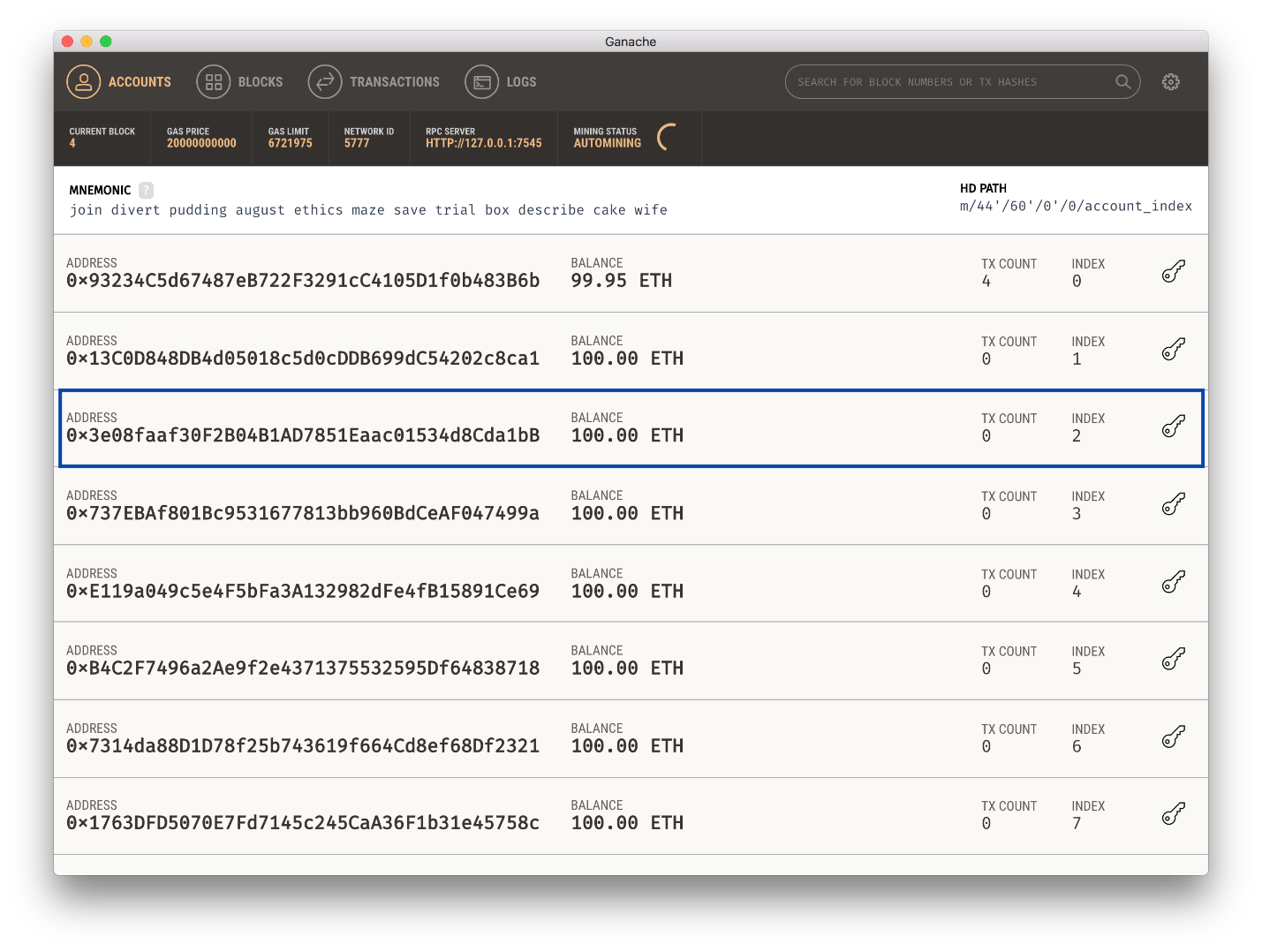
dm.balance().then( function(x) { return x.toString(); });

Output :

'0'

### 7b: Add Some Ether to our wallet from Ganache User

First, you will need to copy one of the wallet address in Ganache. Let us take the third wallet for this:



Now to transfer the money from our wallet to our Demo smart contract, we will need to run this command in the console:

dm.contribute({from:"0x3e08faaf30F2B04B1AD7851Eaac01534d8Cda1bB", value: 5000000000000000000});

Output :

truffle(development)> dm.contribute({from:"0x3e08faaf30F2B04B1AD7851Eaac01534d8Cda1bB ", value: 50000000000000000000});

{ tx:

'0x575c6c8bf243817a3f7fcb98af3680ab69554e514cba7a03c1f5fcd51d73f700',

receipt:

{ transactionHash:

'0x575c6c8bf243817a3f7fcb98af3680ab69554e514cba7a03c1f5fcd51d73f700',

transactionIndex: 0,

blockHash:

'0xad223c273f7f294fe42ccbaaae1b54b8c047675119d86aa9ccbc7951e9f11e66',

blockNumber: 7,

from: '0x48f8a1a5c8e0df9e31ddbd58a40ef3268b88fad1',

to: '0x627a9b60c9a126d138d10f17f9d77d97709b6759',

gasUsed: 41635,

cumulativeGasUsed: 41635,

contractAddress: null,

logs: [],

status: true,

logsBloom:

'0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000',

v: '0x1c',

r:

'0x4a5ccb2aa2e82c48ad6e1883e47481eb25c01ec215b80b9d9aa7f00fa5da67bc',

s:

'0xe5a6470076b330c461cd7843493d0e5d76262f36f5745e911bc11816074a8ce',

rawLogs: [] },

logs: [] }

Here we are sending 50000000000000000000 Wei (or 50 ether) from our third wallet to the smart contract. You should see some data dump on the screen if it is successful.

Note: 1 ether = 10¹⁸ wei.

### 7c: Evaluate the Balance

Next let us check our balance, do you see 5000000000000000000?

dm.balance().then( function(x) { return x.toString(); });

'50000000000000000000'

### 7d: Reset

You need to re-deploy every change you make in your contract. The simplest way to do so is by using truffle deploy --reset. This will also create a new instance of your smart contract.