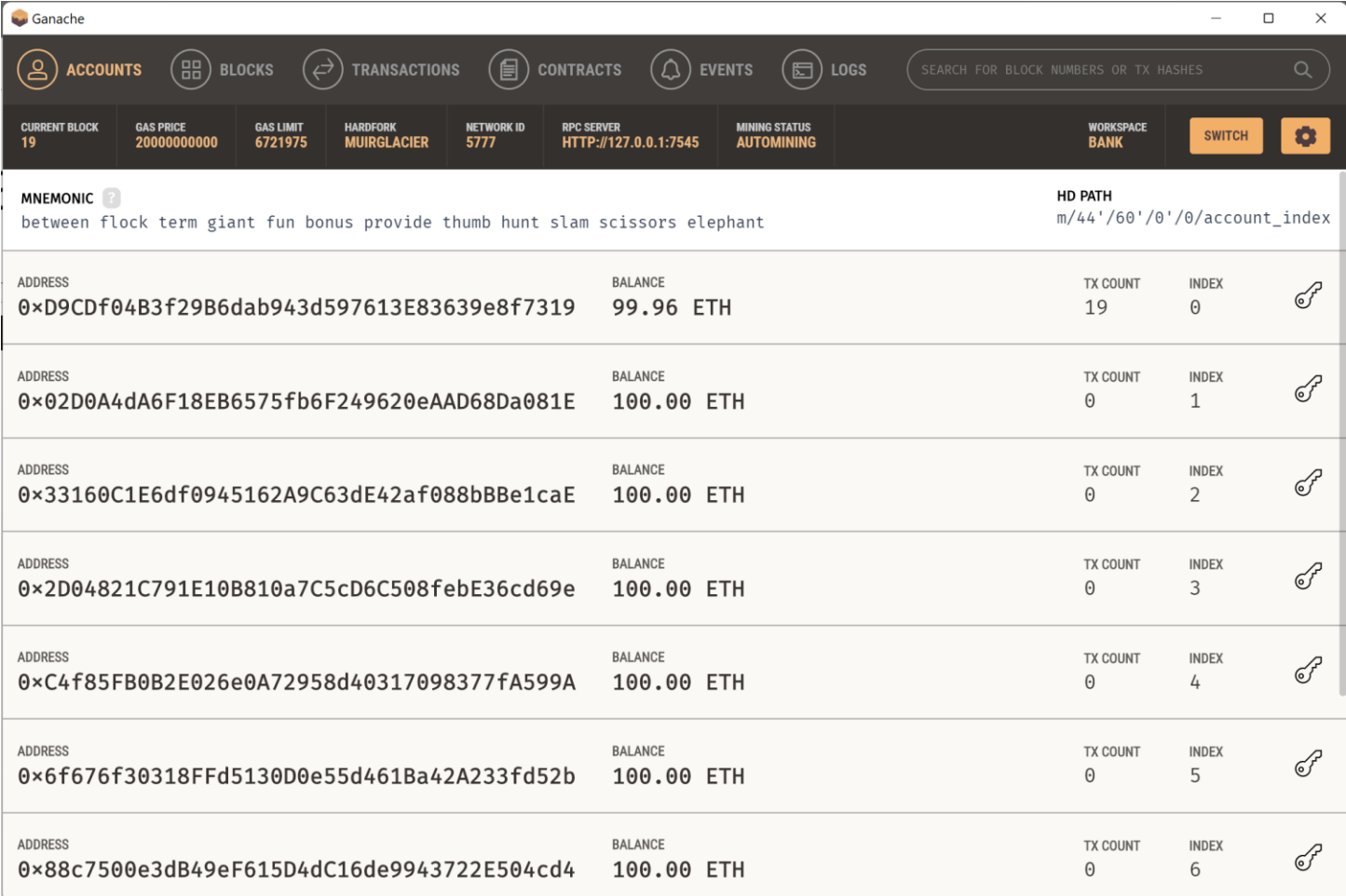


Nakamoto Bank Inc.

Ganache

Provides a simple development environment that allows us to code against.



The screenshot displays the Ganache application window. The top navigation bar includes icons for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS, along with a search bar. Below this, a status bar shows various network parameters: CURRENT BLOCK 19, GAS PRICE 20000000000, GAS LIMIT 6721975, HARDFORK MUIRGLACIER, NETWORK ID 5777, RPC SERVER HTTP://127.0.0.1:7545, MINING STATUS AUTOMINING, and WORKSPACE BANK. The main content area shows the MNEMONIC (between flock term giant fun bonus provide thumb hunt slam scissors elephant) and the HD PATH (m/44'/60'/0'/0/account_index). Below this is a table listing accounts with their addresses, balances, transaction counts, and indices.

ADDRESS	BALANCE	TX COUNT	INDEX
0xD9CDf04B3f29B6dab943d597613E83639e8f7319	99.96 ETH	19	0
0x02D0A4dA6F18EB6575fb6F249620eAAD68Da081E	100.00 ETH	0	1
0x33160C1E6df0945162A9C63dE42af088bBBe1caE	100.00 ETH	0	2
0x2D04821C791E10B810a7C5cD6C508febE36cd69e	100.00 ETH	0	3
0xC4f85FB0B2E026e0A72958d40317098377fA599A	100.00 ETH	0	4
0x6f676f30318FFd5130D0e55d461Ba42A233fd52b	100.00 ETH	0	5
0x88c7500e3dB49eF615D4dC16de9943722E504cd4	100.00 ETH	0	6

Contracts written in Solidity

Every contract starts with a version constraint

We use the ^ (caret) restriction to ensure only major version 0 can be used

```
pragma solidity ^0.5.0;
```

Contract details

A dictionary that contains addresses and their respective amount.

The two methods utilize solidity's require methods that serves as validation for the input.

Lastly a view method, that is able to show us the current balance of a the asking user.

```
pragma solidity ^0.5.0;

contract Bank {
    mapping(address => uint) public balances;

    function deposit(uint amount) public {
        require(amount >= 0);
        balances[msg.sender] += amount;
    }

    function withdraw(uint amount) public {
        uint balance = balances[msg.sender];
        require(balance >= amount);
        balances[msg.sender] -= amount;
    }

    function getAccount() public view returns (uint) {
        return balances[msg.sender];
    }
}
```

Migration

We migrate the contract onto the blockchain to make it available to end users, using the “truffle migrate” command



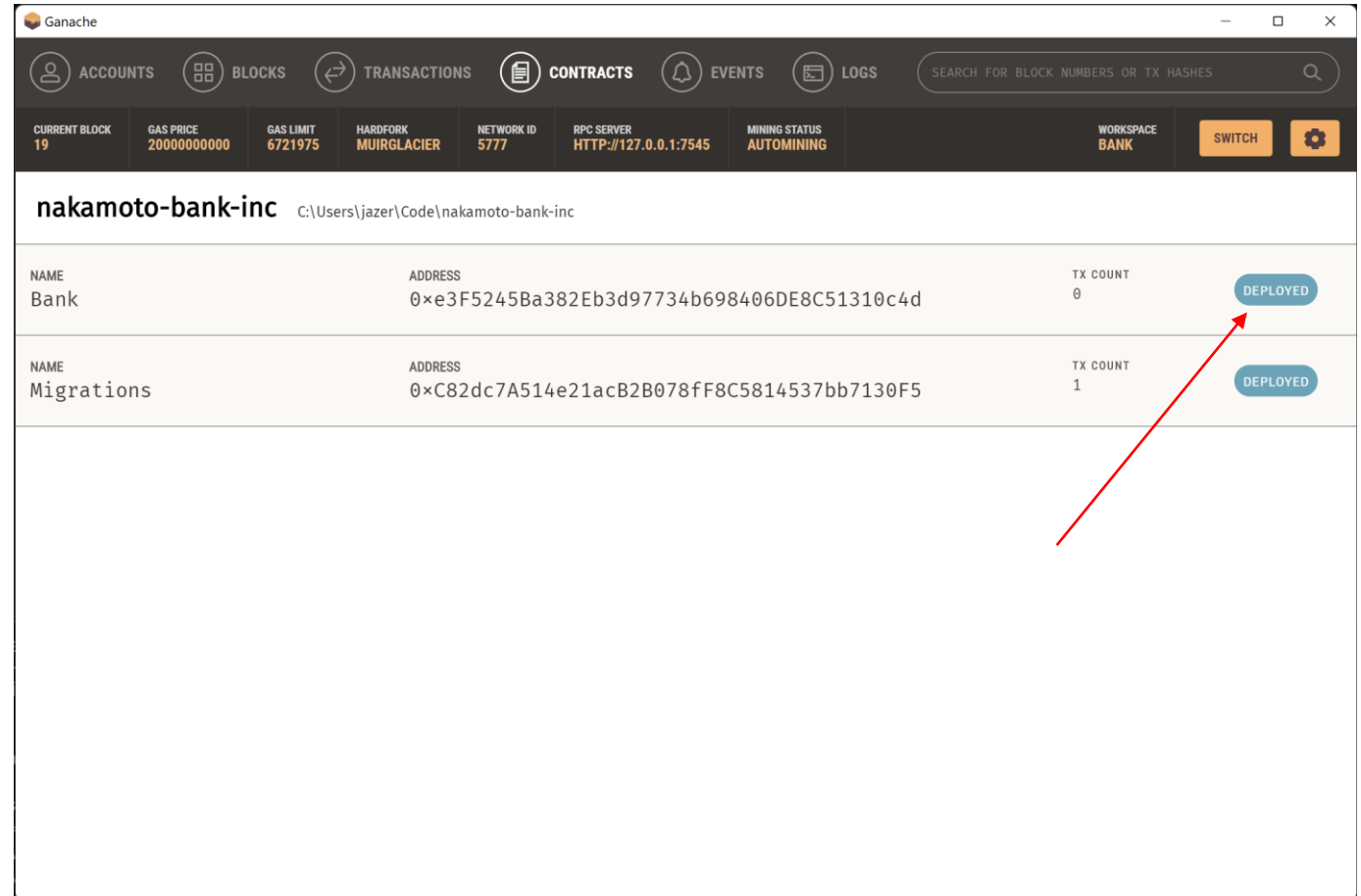
```
var Bank = artifacts.require("Bank");

module.exports = function(deployer) {
  deployer.deploy(Bank);
}
```

Migration

We migrate the contract onto the blockchain to make it available to end users, using the “truffle migrate” command

Verified within the Contracts tab of Ganache



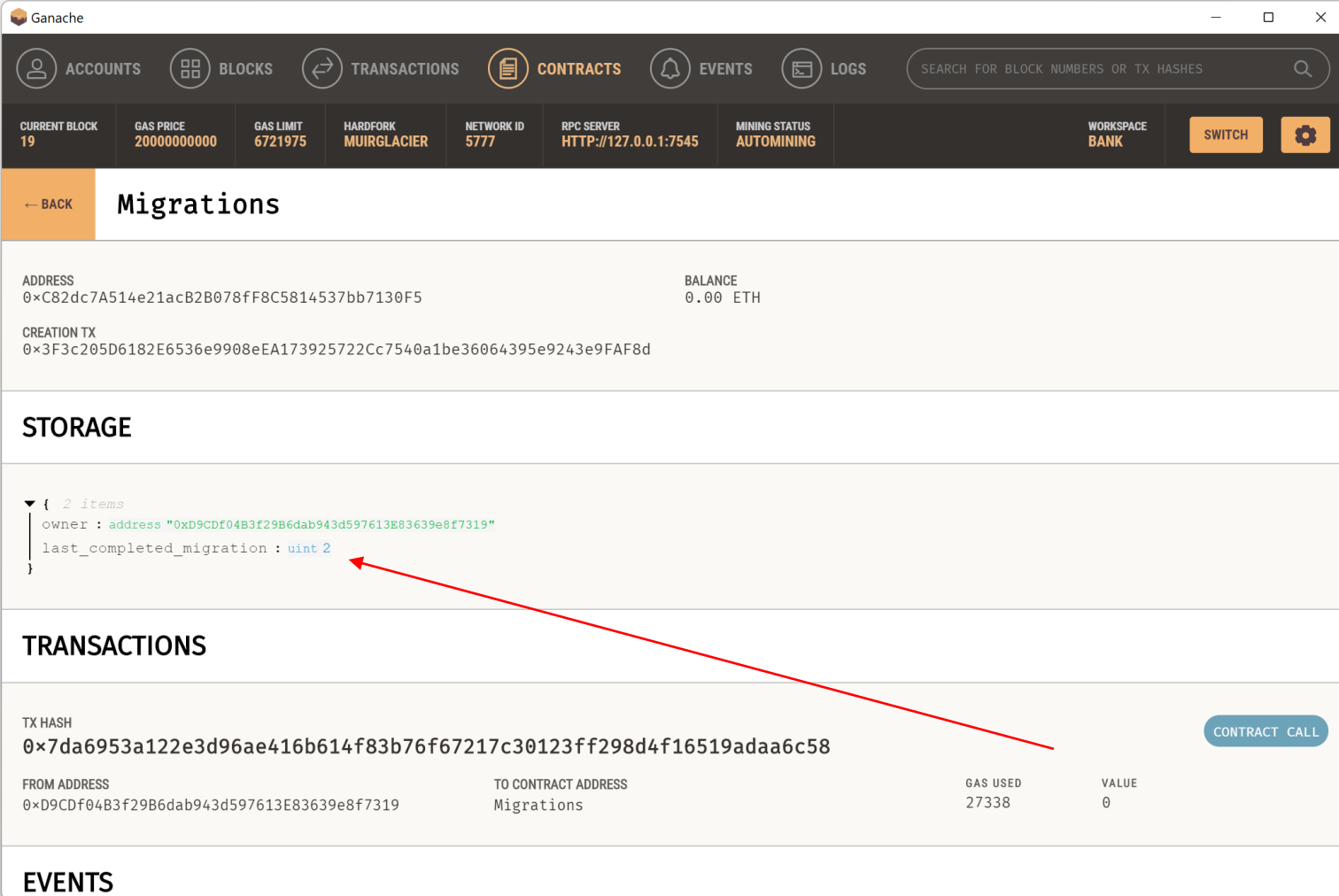
The screenshot shows the Ganache application window with the 'CONTRACTS' tab selected. The workspace is named 'nakamoto-bank-inc' and is located at 'C:\Users\jazer\Code\nakamoto-bank-inc'. The table below lists the deployed contracts:

NAME	ADDRESS	TX COUNT	STATUS
Bank	0xe3F5245Ba382Eb3d97734b698406DE8C51310c4d	0	DEPLOYED
Migrations	0xC82dc7A514e21acB2B078fF8C5814537bb7130F5	1	DEPLOYED

A red arrow points to the 'DEPLOYED' button for the 'Migrations' contract.

Migration

The migration contract keeps track of what needs to be migrated



The screenshot displays the Ganache web interface. At the top, a navigation bar includes icons for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS (highlighted), EVENTS, and LOGS. Below this is a status bar with various network metrics: CURRENT BLOCK 19, GAS PRICE 20000000000, GAS LIMIT 6721975, HARDFORK MUIRGLACIER, NETWORK ID 5777, RPC SERVER HTTP://127.0.0.1:7545, MINING STATUS AUTOMINING, and WORKSPACE BANK. A search bar is also present.

The main content area is titled "Migrations" with a "← BACK" button. It is divided into several sections:

- ADDRESS:** 0xC82dc7A514e21acB2B078fF8C5814537bb7130F5. **BALANCE:** 0.00 ETH.
- CREATION TX:** 0x3F3c205D6182E6536e9908eEA173925722Cc7540a1be36064395e9243e9FAF8d.
- STORAGE:** Displays a JSON object:

```
{ 2 items
  owner : address "0xD9CDf04B3f29B6dab943d597613E83639e8f7319"
  last_completed_migration : uint 2
}
```

. A red arrow points from the "last_completed_migration" value to the "Migrations" entry in the Transactions section.
- TRANSACTIONS:** Shows a transaction with TX HASH 0x7da6953a122e3d96ae416b614f83b76f67217c30123ff298d4f16519adaa6c58. A "CONTRACT CALL" button is visible. Below the hash, it lists FROM ADDRESS (0xD9CDf04B3f29B6dab943d597613E83639e8f7319), TO CONTRACT ADDRESS (Migrations), GAS USED (27338), and VALUE (0).
- EVENTS:** This section is currently empty.

Testing

We chose the JS way using the Mocha framework (bundled in Truffle)



```
describe("depositing and withdrawing money", async() => {  
  before(`deposit ${amountToDeposit} TNC to the first account`, async () => {  
    await bank.deposit(amountToDeposit, { from: accounts[0] });  
  });  
  ...  
})
```


Testing

We chose the JS way using the Mocha framework (bundled in Truffle)

```
describe("depositing and withdrawing money", async() => {  
  
  ...  
  
  it(`can get balance in TNC`, async () => {  
    const balance = (await bank.getAccount({ from: accounts[0] })).toNumber();  
    assert.equal(balance, amountToDeposit, `The account balance should be  
    ${amountToDeposit} TNC.`)  
  })  
  
  it(`can withdraw ${amountToWithdraw} TNC`, async () => {  
    await bank.withdraw(amountToWithdraw, {from: accounts[0]});  
    const balance = (await bank.getAccount({ from: accounts[0] })).toNumber();  
    assert.equal(balance, amountToDeposit - amountToWithdraw, `The account balance should  
    be ${amountToDeposit - amountToWithdraw} TNC.`)  
  })  
  
  ...  
})
```

Testing

We chose the JS way using the Mocha framework (bundled in Truffle)

```
describe("depositing and withdrawing money", async() => {  
  
  ...  
  
  it('cannot withdraw more than current balance', async () => {  
    let fails = false;  
    try  
    {  
      await bank.withdraw(amountToDeposit + 1, {from: accounts[0]})  
    }  
    catch  
    {  
      fails = true;  
    }  
    assert.equal(fails, true);  
  })  
  
  it('cannot deposit negative amount', async () => {  
    let fails = false;  
    try  
    {  
      await bank.deposit(-1, {from: accounts[0]})  
    }  
    catch  
    {  
      fails = true;  
    }  
    assert.equal(fails, true);  
  })  
  
})
```