Deploying an Ethereum Smart Contract

Final Project – CSCE-492/892

UNL School of Computing

Bala Subramanyam Duggirala
Nhien Nguyen

What is a Smart Contract?

Smart Contracts are computer programs, that are stored on Block Chain technology, which store/execute the terms of agreement between the parties involved in a contract

They are self-executing, transparent, and irreversible, and are executed when a set of pre-defined conditions are met

Since they are stored on the Block Chain, they are also immutable

Why use Smart Contracts ?

Autonomy: Since Smart Contracts do not need brokers/third parties, it eliminates the risk of trusting these third parties, and also saves costs

Safety: Smart Contracts are encrypted, and hence are safe from cyber threats

Speed: Since manual intermediaries are out of scope, it saves a lot of time which would otherwise be spent on various manual business processes

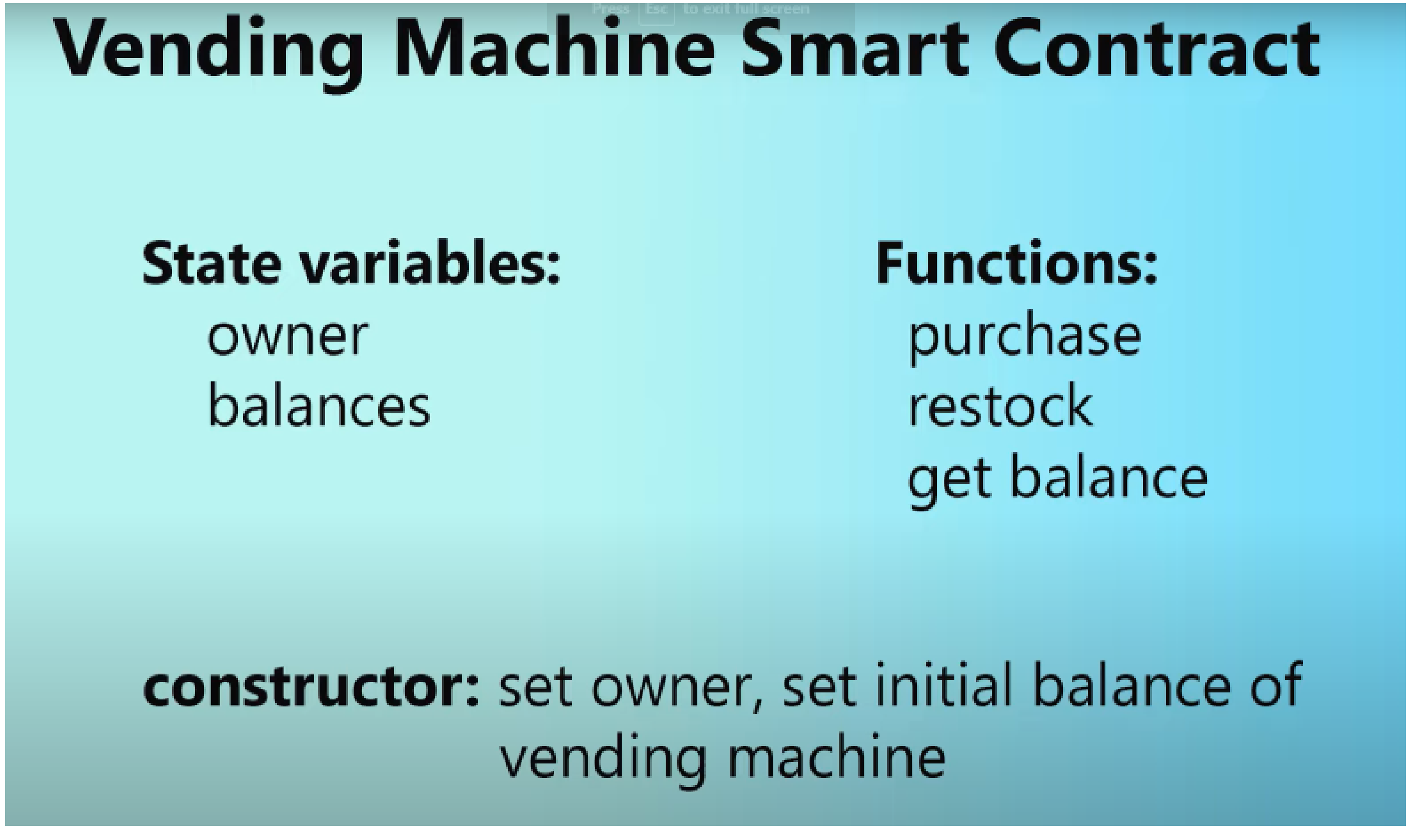
Backup: They are stored on the block chain and are duplicated several times, hence can be easily restored

What is Ethereum and its relation to Smart Contracts?

Ethereum is an open-source Block Chain specifically designed to implement Smart Contract functionality

Ether is its native digital currency

Our Project – Test and Deploy a Smart Contract on an Ethereum Testnet



Vending Machine Smart Contract

State variables:
owner
balances

Functions:
purchase
restock
get balance

constructor: set owner, set initial balance of vending machine

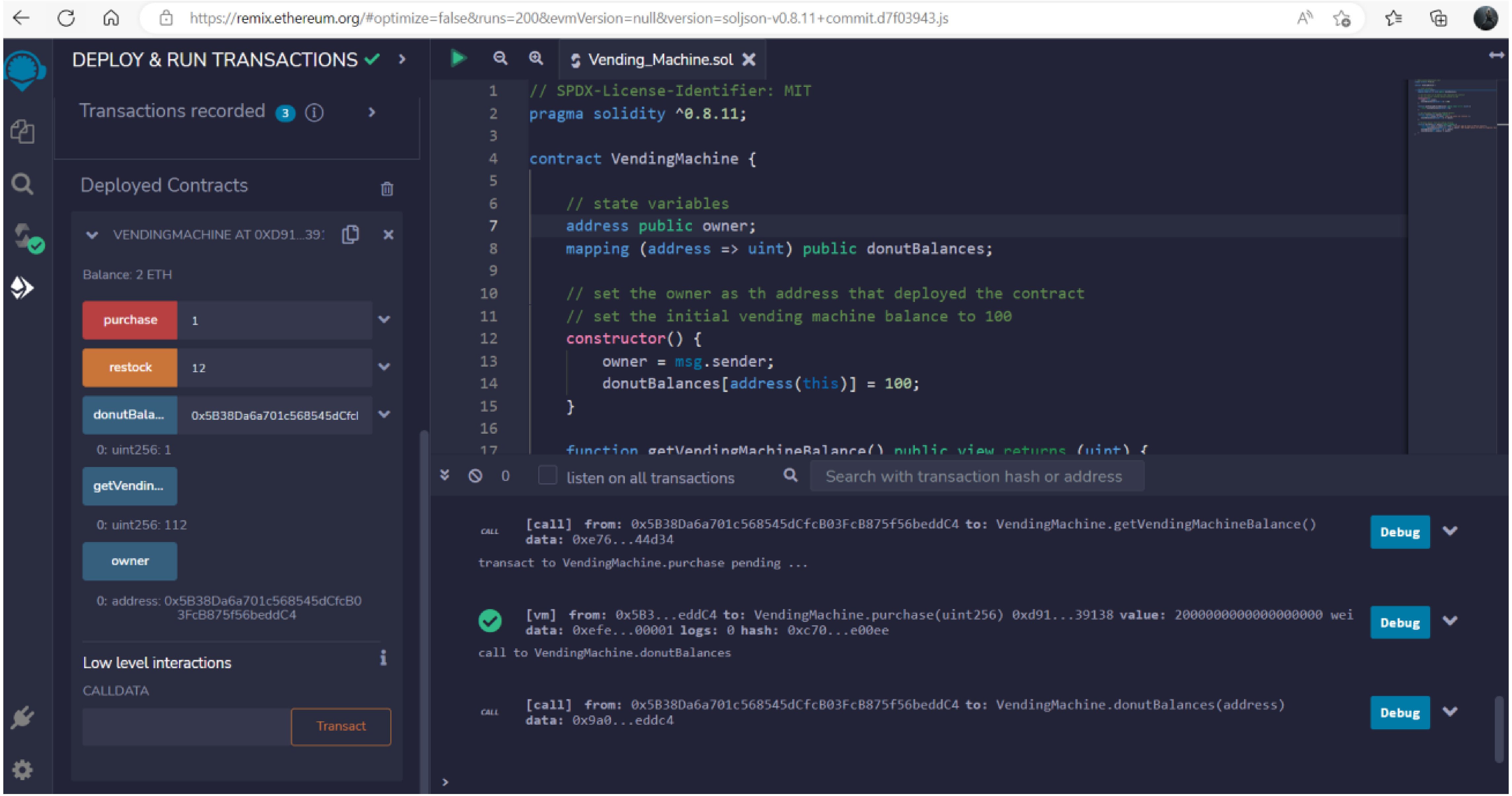
- A simple Vending Machine which accepts Ether and sells donuts

Source: “**Smart Contract Tutorial - Vending Machine Smart Contract in Solidity” by *Block Explorer*, YouTube**

Supporting technologies used

- **Solidity** – programming language
- **Remix IDE** – online editor for initial development and unit testing
- **Truffle Development Suite** – unit test locally, deploy on Ethereum testnets/mainnet
  - **Ganache-cli** – local development block chain that can be accessed using Truffle
  - **Metamask wallet** – provides addresses and required encryption
  - **Infura Development Suite** – Provides us access to Ethereum nodes without us having to create/sync them
  - **Goerli** – testnet to deploy the smart contract

Initial Implementation/Testing: Remix IDE



https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.11+commit.d7f03943.js

Balance: 2 ETH

purchase 1

restock 12

donutBala... 0x5B38Da6a701c568545dCfcl

0: uint256: 1

getVendin...

0: uint256: 112

owner

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

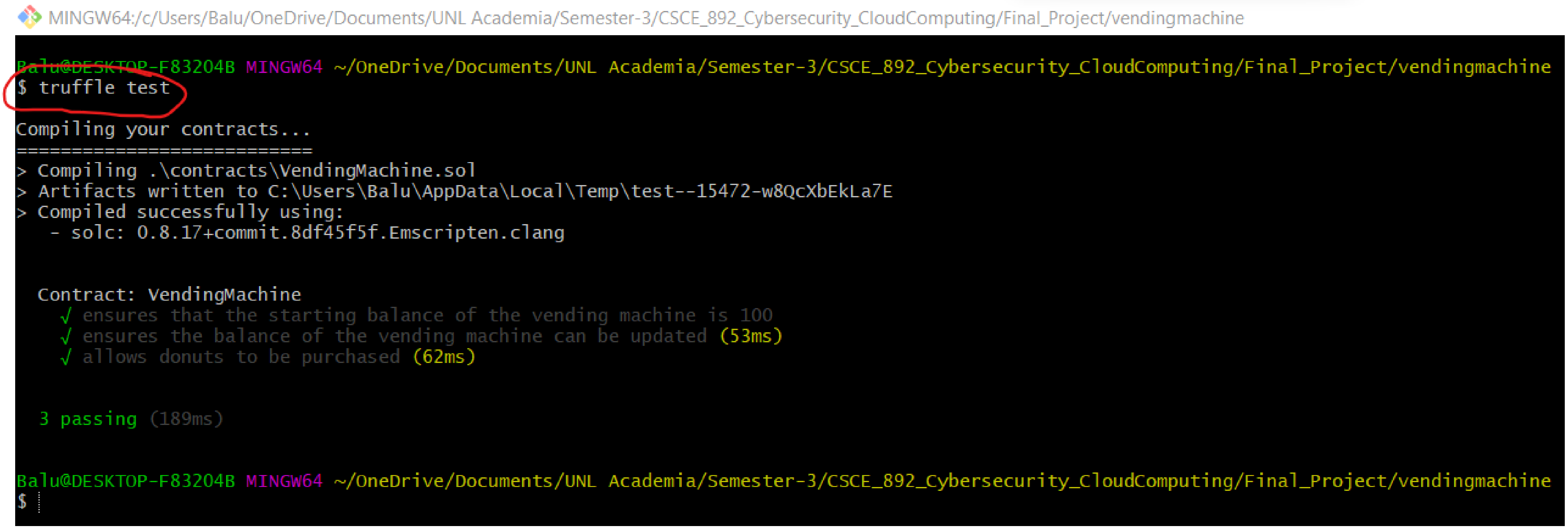
```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.11;

contract VendingMachine {

    // state variables
    address public owner;
    mapping (address => uint) public donutBalances;

    // set the owner as th address that deployed the contract
    // set the initial vending machine balance to 100
    constructor() {
        owner = msg.sender;
        donutBalances[address(this)] = 100;
    }

    function getVendingMachineBalance() public view returns (uint) {
```

0
listen on all transactions
Search with transaction hash or address

```
CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: VendingMachine.getVendingMachineBalance()
      data: 0xe76...44d34
transact to VendingMachine.purchase pending ...

      [vm] from: 0x5B3...eddC4 to: VendingMachine.purchase(uint256) 0xd91...39138 value: 2000000000000000000 wei
      data: 0xefe...00001 logs: 0 hash: 0xc70...e00ee
call to VendingMachine.donutBalances

CALL  [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: VendingMachine.donutBalances(address)
      data: 0x9a0...eddc4
```

Final Implementation: Truffle Suite -- TESTING

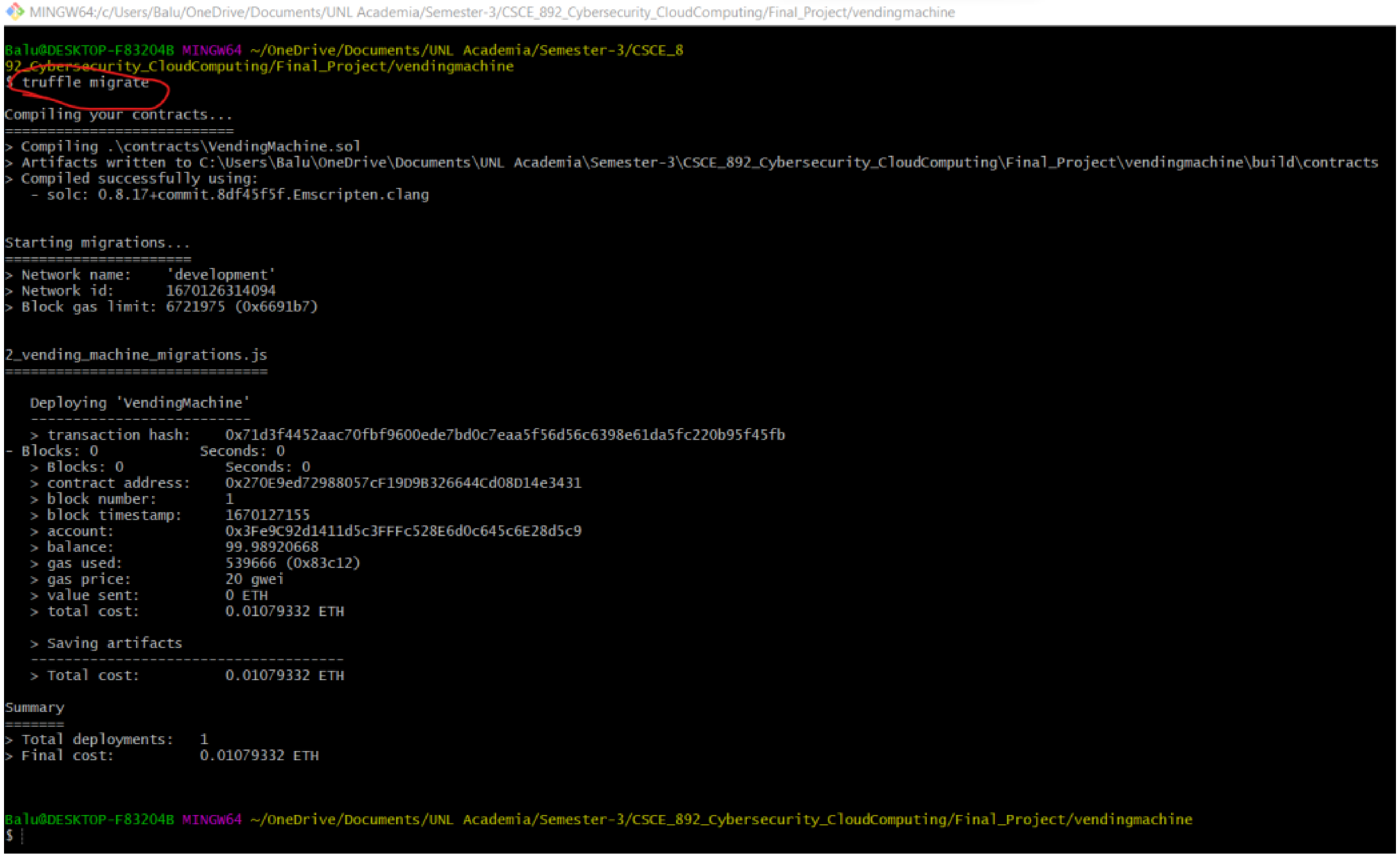
- Truffle allows us to unit test our code using the command ‘*truffle test’*



MINGW64:/c/Users/Balu/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersecurity_CloudComputing/Final_Project/vendingmachine

```
Balu@DESKTOP-F83204B MINGW64 ~/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersecurity_CloudComputing/Final_Project/vendingmachine
$ truffle test

Compiling your contracts...
===========================
> Compiling .\contracts\VendingMachine.sol
> Artifacts written to C:\Users\Balu\AppData\Local\Temp\test--15472-w8QcXbEkLa7E
> Compiled successfully using:
   - solc: 0.8.17+commit.8df45f5f.Emscripten.clang


  Contract: VendingMachine
    √ ensures that the starting balance of the vending machine is 100
    √ ensures the balance of the vending machine can be updated (53ms)
    √ allows donuts to be purchased (62ms)


  3 passing (189ms)


Balu@DESKTOP-F83204B MINGW64 ~/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersecurity_CloudComputing/Final_Project/vendingmachine
$
```

Truffle Suite – Deploying on a Local Block Chain, **Ganache**


MINGW64:/c/Users/Balu/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersecurity_CloudComputing/Final_Project/vendingmachine

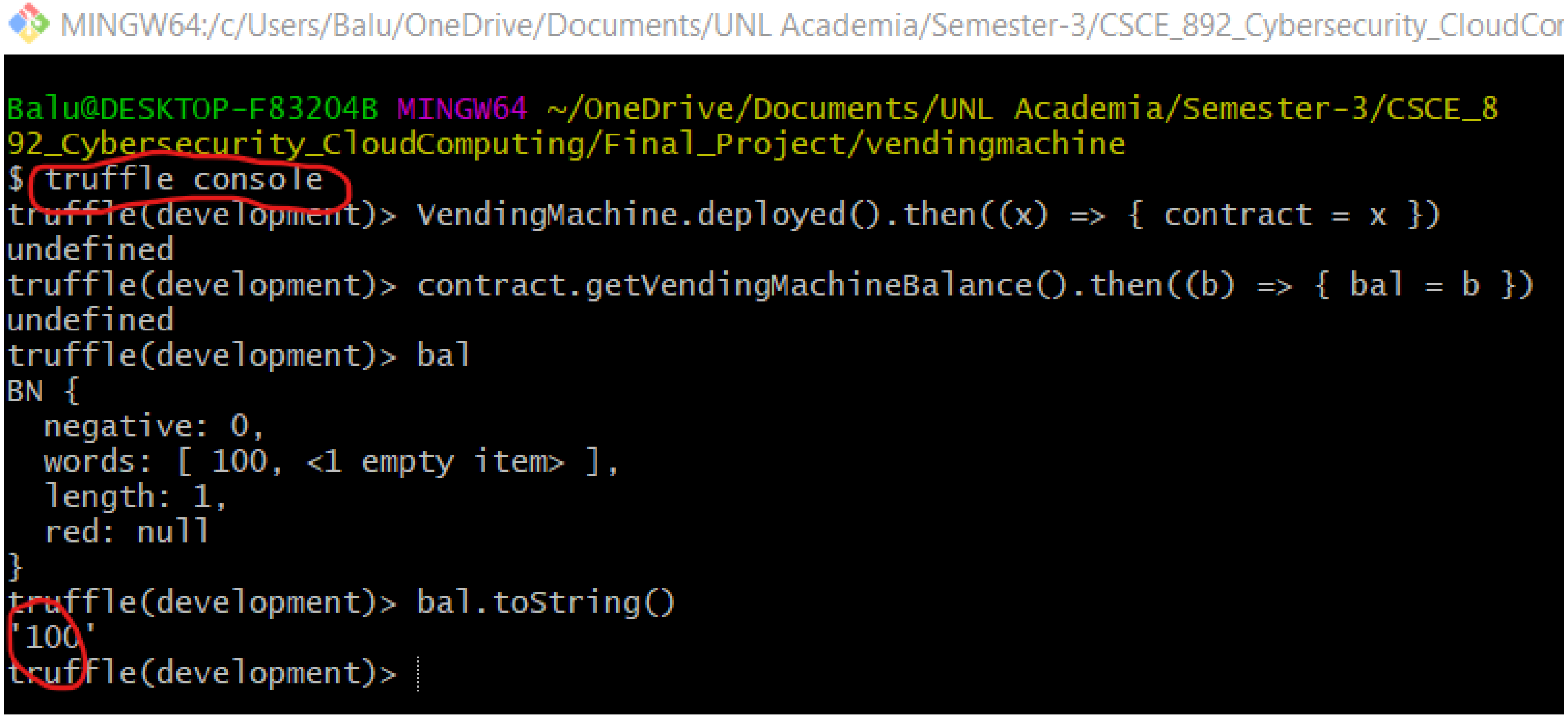
```
Balu@DESKTOP-F83204B MINGW64 ~/OneDrive/Documents/UNL Academia/Semester-3/CSCE_8
92_Cybersecurity_CloudComputing/Final_Project/vendingmachine
$ truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\VendingMachine.sol
> Artifacts written to C:\Users\Balu\OneDrive\Documents\UNL Academia\Semester-3\CSCE_892_Cybersecurity_CloudComputing\Final_Project\vendingmachine\build\contracts
> Compiled successfully using:
   - solc: 0.8.17+commit.8df45f5f.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      1670126314094
> Block gas limit: 6721975 (0x6691b7)


2_vending_machine_migrations.js
===============================

   Deploying 'VendingMachine'
   --------------------------
   > transaction hash:    0x71d3f4452aac70fbf9600ede7bd0c7eaa5f56d56c6398e61da5fc220b95f45fb
- Blocks: 0            Seconds: 0
   > Blocks: 0            Seconds: 0
   > contract address:    0x270E9ed72988057cF19D9B326644Cd08D14e3431
   > block number:        1
   > block timestamp:     1670127155
   > account:             0x3Fe9C92d1411d5c3FFFc528E6d0c645c6E28d5c9
   > balance:             99.98920668
   > gas used:            539666 (0x83c12)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.01079332 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:          0.01079332 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.01079332 ETH



Balu@DESKTOP-F83204B MINGW64 ~/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersecurity_CloudComputing/Final_Project/vendingmachine
$ 
```

“*truffle migrate*” command by default points towards a development block chain like Ganache

We can use the same command, using ‘*--network*’ flags, to access other Ethereum networks including the main net, by setting up the truffle_config.js file accordingly

Truffle Suite – Interacting with the deployed Contract

- “*truffle console*” provides us with an interface to interact with the deployed smart contract

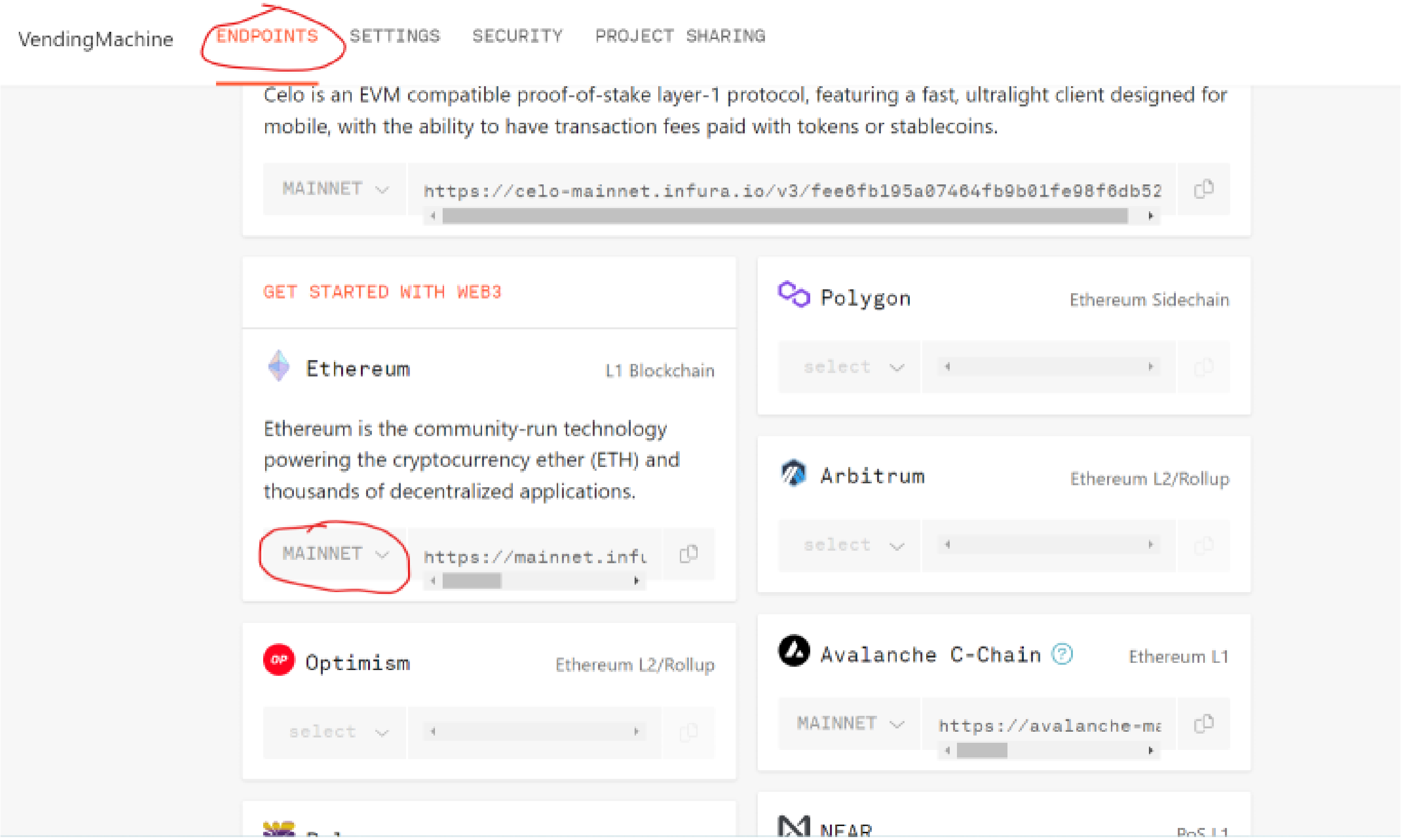


MINGW64:/c/Users/Balu/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersecurity_CloudCor

```
Balu@DESKTOP-F83204B MINGW64 ~/OneDrive/Documents/UNL Academia/Semester-3/CSCE_8
92_Cybersecurity_CloudComputing/Final_Project/vendingmachine
$ truffle console
truffle(development)> VendingMachine.deployed().then((x) => { contract = x })
undefined
truffle(development)> contract.getVendingMachineBalance().then((b) => { bal = b })
undefined
truffle(development)> bal
BN {
  negative: 0,
  words: [ 100, <1 empty item> ],
  length: 1,
  red: null
}
truffle(development)> bal.toString()
'100'
truffle(development)>
```

Truffle Suite – Deployment on Goerli Testnet

- *Truffle* lets us use our wallet to securely make transactions using private keys/mnemonic codes provided by the wallet (like Metamask)
- *Infura* conveniently lets us connect to the block chain without us having to set up a node (This can charge you depending on the network used)
- Use '*truffle migrate*' to deploy to the network of our choice as shown previously



VendingMachine ENDPOINTS SETTINGS SECURITY PROJECT SHARING

Celo is an EVM compatible proof-of-stake layer-1 protocol, featuring a fast, ultralight client designed for mobile, with the ability to have transaction fees paid with tokens or stablecoins.

MAINNET https://celo-mainnet.infura.io/v3/fee6fb195a07464fb9b01fe98f6db52

GET STARTED WITH WEB3

Ethereum L1 Blockchain

Ethereum is the community-run technology powering the cryptocurrency ether (ETH) and thousands of decentralized applications.

MAINNET https://mainnet.infu

Polygon Ethereum Sidechain

select

Arbitrum Ethereum L2/Rollup

select

Optimism Ethereum L2/Rollup

select

Avalanche C-Chain Ethereum L1

MAINNET https://avalanche-ma

NEAR PoS L1

Post Deployment

- We can use ‘*truffle console*’ to interact with the block chain as shown previously for the Ganache testnet, and
- *EtherScan* website to track our transactions on the block chain

References

- Vending Machine domain code taken from https://github.com/jspruance/block-explorer-tutorials/blob/main/smart-contracts/solidity/VendingMachine.sol
- Reference for Truffle code base: Deploy a smart contract to Ethereum using Truffle - A step-by-step guide. – YouTube

THANK YOU!