# Exercises: Create Simple Voting System

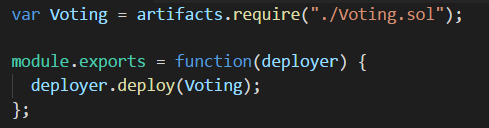
We will build a simple decentralized voting application. You will build a voting application where you will initialize a set of candidates who will be contesting in the election and then vote for the candidates. The votes will be stored on the blockchain. You will go through the process of writing the voting contract, deploy to the blockchain and interact with it.

## Create the Smart Contract

1. Create a new folder named **“Voting System”**. Open your command prompt and go in the folder.
2. There, initialize truffle by the command:

|  |
| --- |
| truffle init |

1. Create a new contract called **Voting.sol** and put it in **contracts** folder where will be our smart contract logic
2. In the **migrations** folder create file **2\_deploy\_contracts.js** and add the following code which will deploy the smart contract



For the **Voting** contract:

1. Create a **public** field, which for each candidate stores his votes.
2. Create a **private** array storing every candidate.
3. Create **addCandidate(bytes32)** function, which adds the candidate to an array.
4. Create **validCandidate(bytes32)** function, which checks whether the given candidate is contained in the array.
5. Then create function **voteForCandidate(bytes32)**, which votes for given candidate.
6. Also, you will need to create function **totalVotesFor(bytes32)** which will return the count of votes for a candidate.

The contract source code is available here: <https://raw.githubusercontent.com/sMustafov/Voting/master/Voting.sol> .

## Setup the Development Environment

1. Install and start testrpc

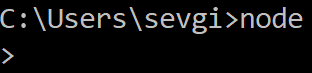
|  |
| --- |
| npm install –g ethereumjs-testrpc |



1. Install **node.js**
2. Install **web3.js** library

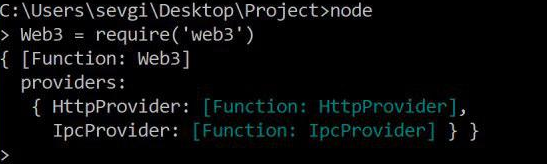
|  |
| --- |
| npm install –g web3 |

1. Run **node**



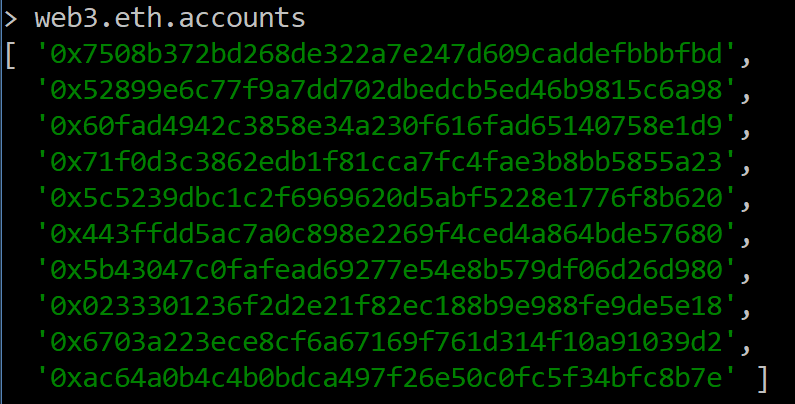
## Compiling the contract

1. Require the **web3** library:

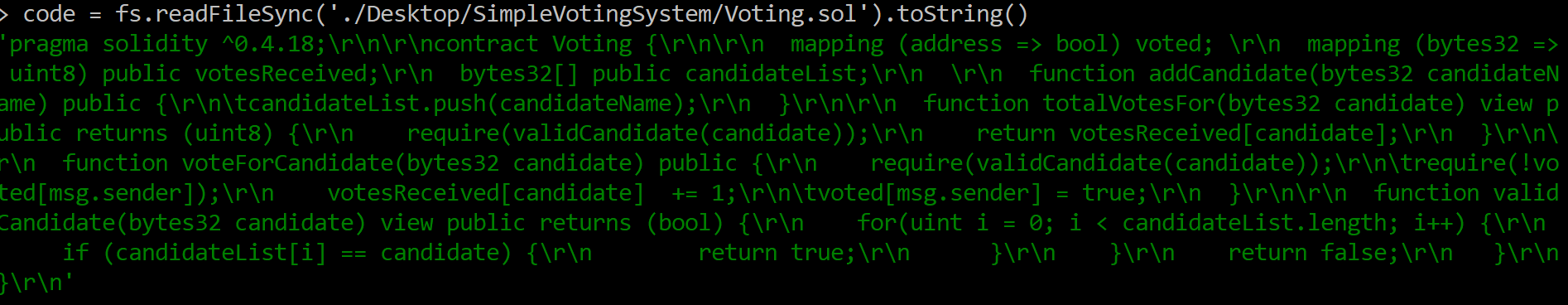




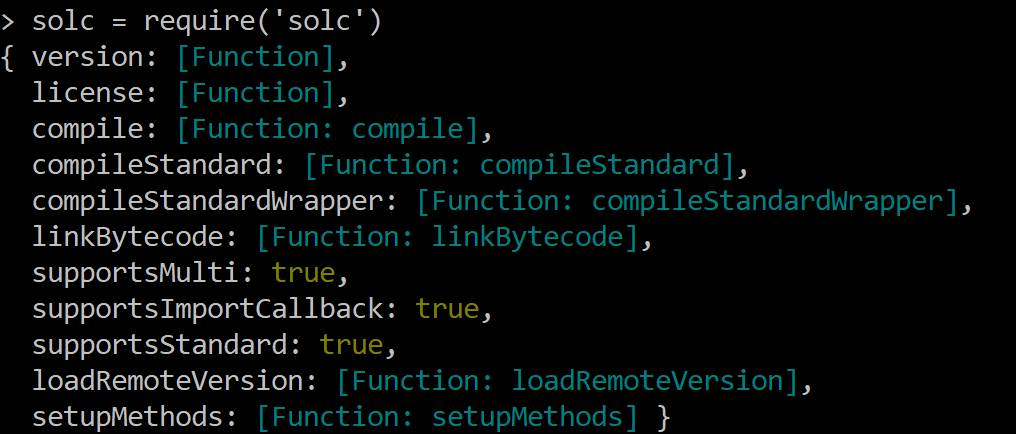
1. You can get all the accounts you have in your private network:



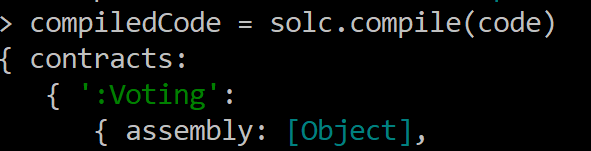
1. You can read the smart contract from the node console:



1. Install **solc** (solidity compiler) and require it:



1. Compile the code and you will get the bytecode, metadata, interface and so on:



## Deploy the contract

1. We should get the **abiDefinition** in JSON format by calling the command:



1. Create **VotingContract**:



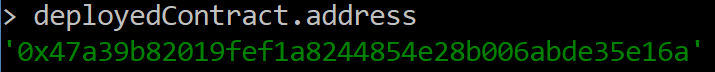
1. We should get the **bytecode**:



1. Now we will **deploy** the contract:



1. Get the **address** of the deployed contract:

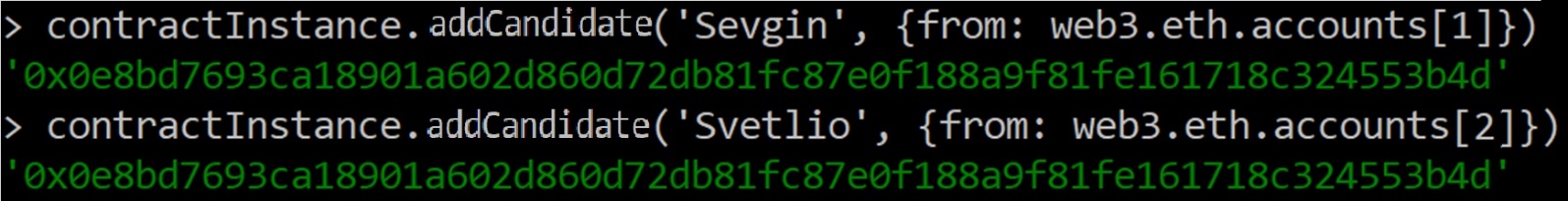


1. Get the contract **instance:**

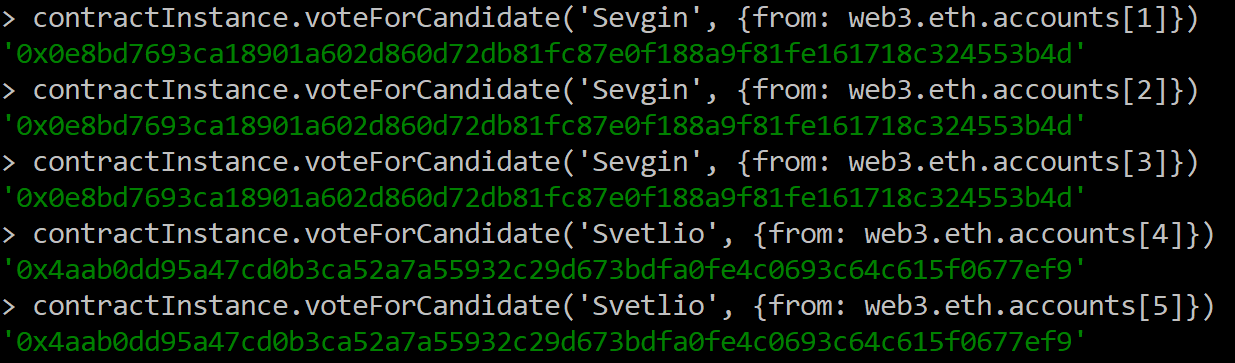


## Interact with the contract

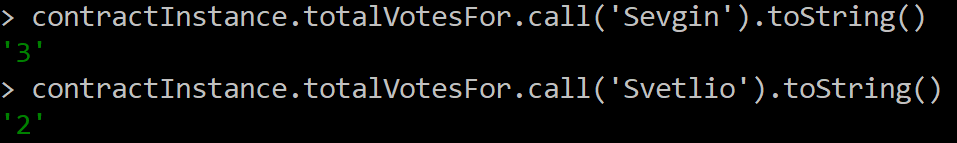
1. Add Candidates:



1. Vote for Candidates:



1. Check the votes for the candidates:



You can check whole code from here:

# What to Submit?

Submit as exercise outcome the **zip file with 3 images and the source code.** The first image should be the one with the deployed contract address, the second - adding of the candidates and the third one - the votes of the candidates you added.