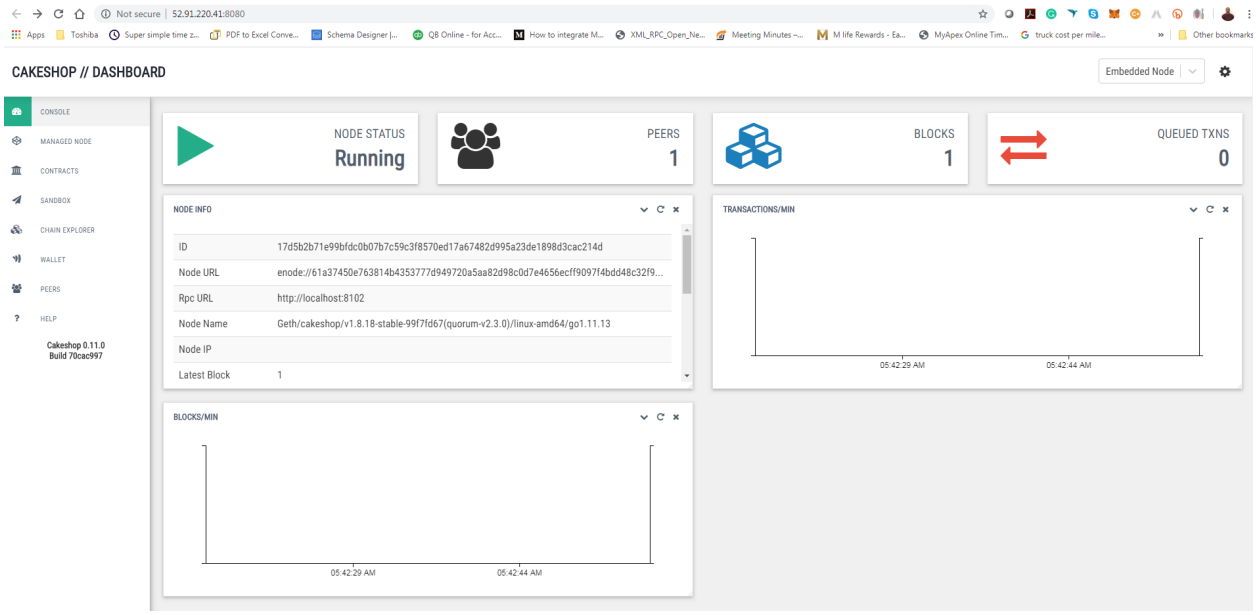
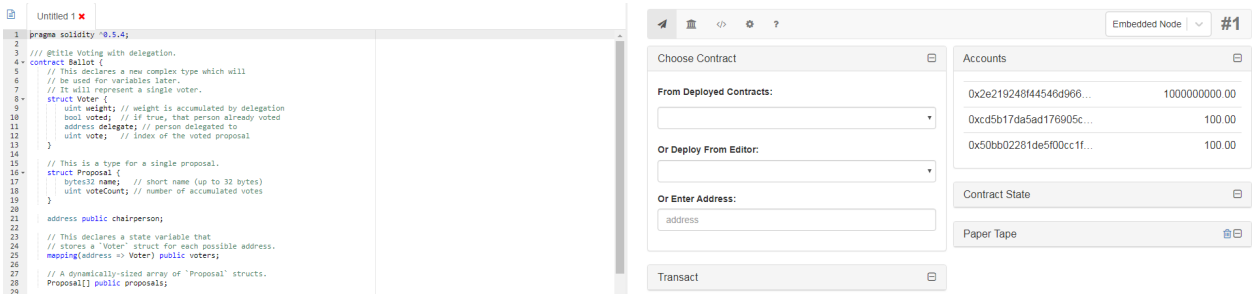
Quorum

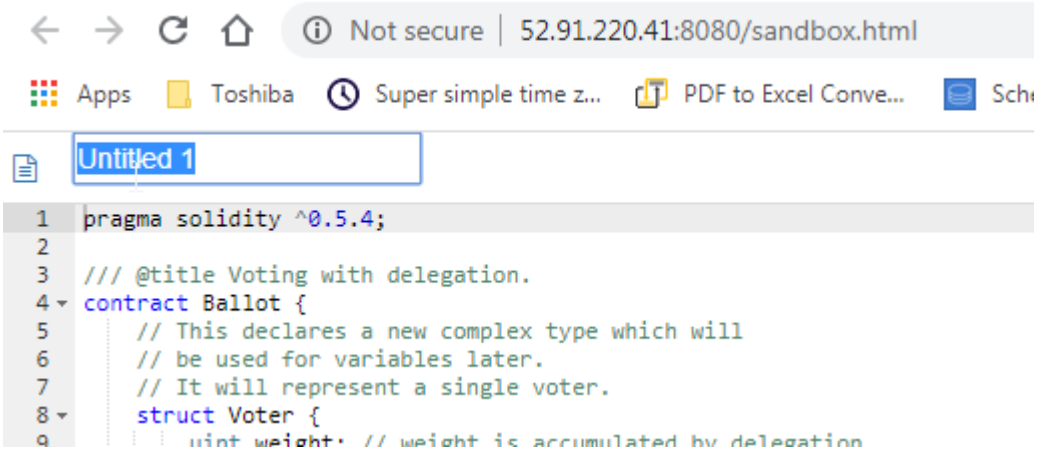
1. Launch Cakeshop : docker run -p 8080:8080 quorumengineering/cakeshop
    a. If this doesn't launch, either stop and remove prior instances using docker or change the name.
    b. If it says there is already an instance running, go to: <IP>:8080



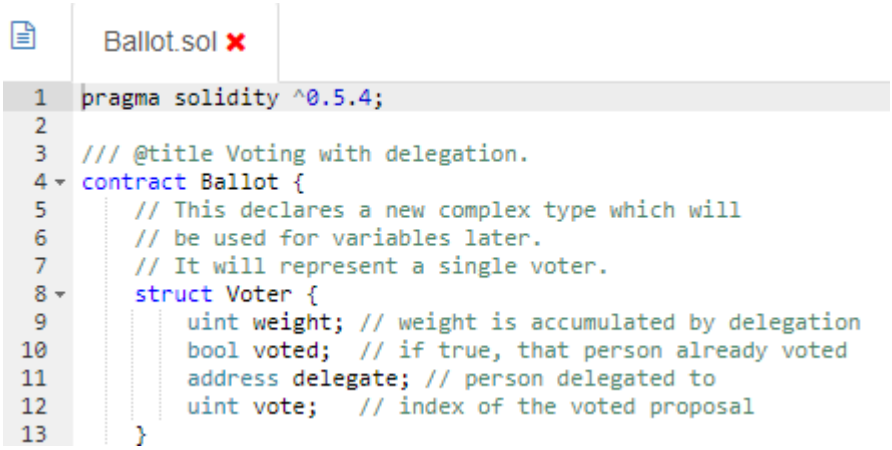
2. Go to "NODE INFO" and get the status and the number of blocks created in this blockchain
3. On the menu on the left, click on the "SANDBOX"



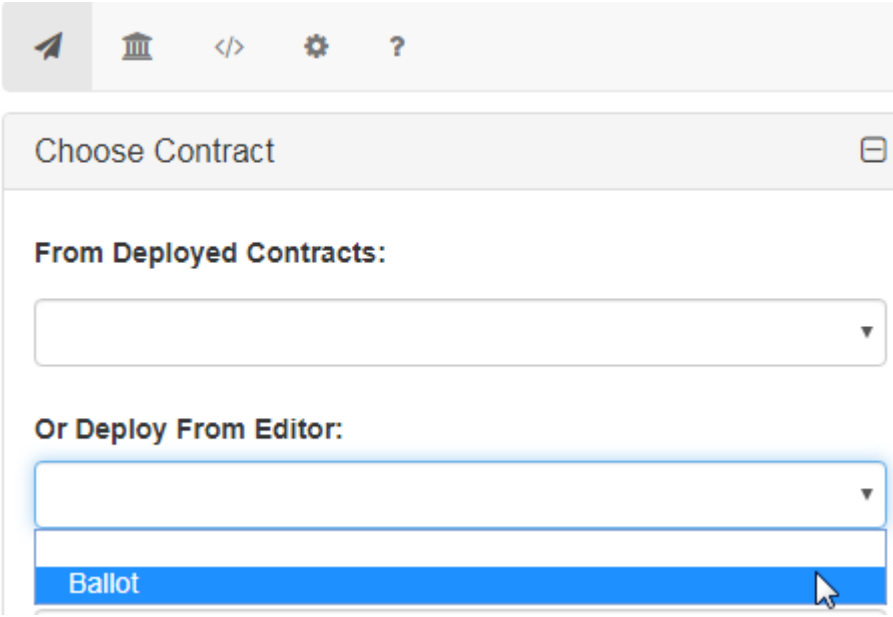
4. Click on "Untitled 1" and change the name to "Ballot.sol"

a. 

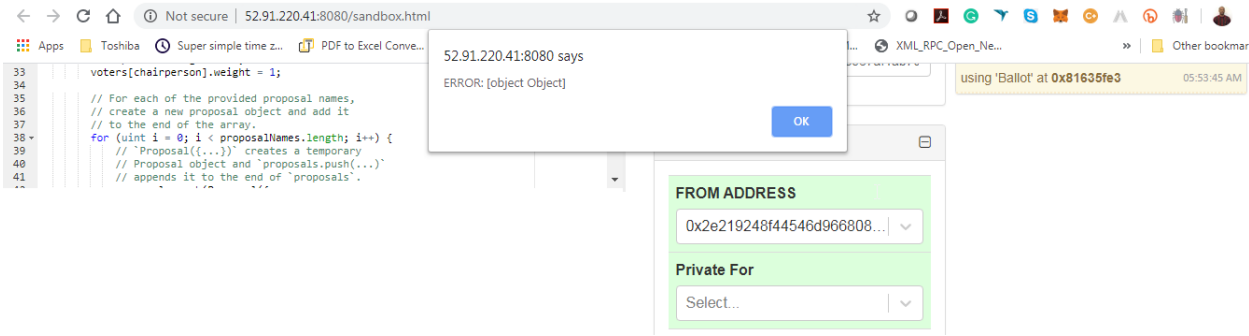
b. It is important that the name of the smart contract, always match the name of the file. In this case, the smart contract is named Ballot so you must create a file named Ballot.sol

c. 

5. On the right of the screen, please “Choose Contract” --- Ballot --- from Editor:

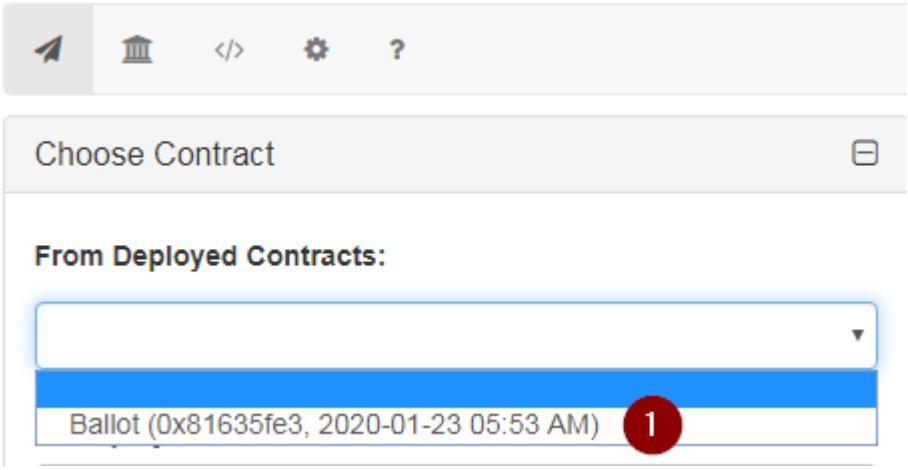
a. 

b. Click Deploy

c. 

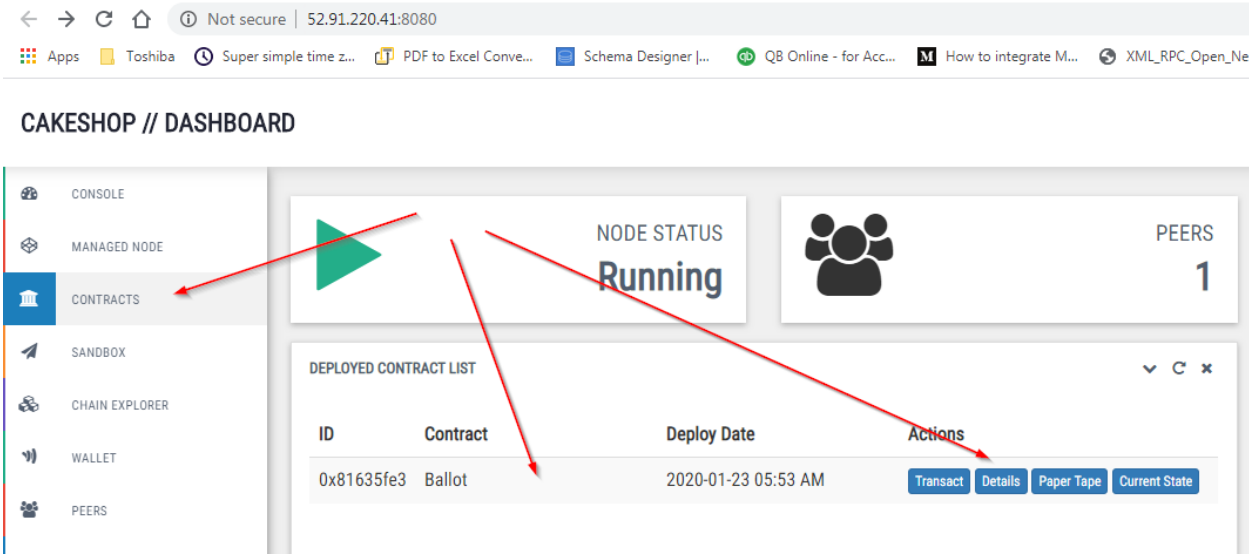
d. ***You may get an error but you can ignore if it deploys***

6. Click on "Choose Contract" From Deployed Contracts: and ensure that your smart contract deployed.

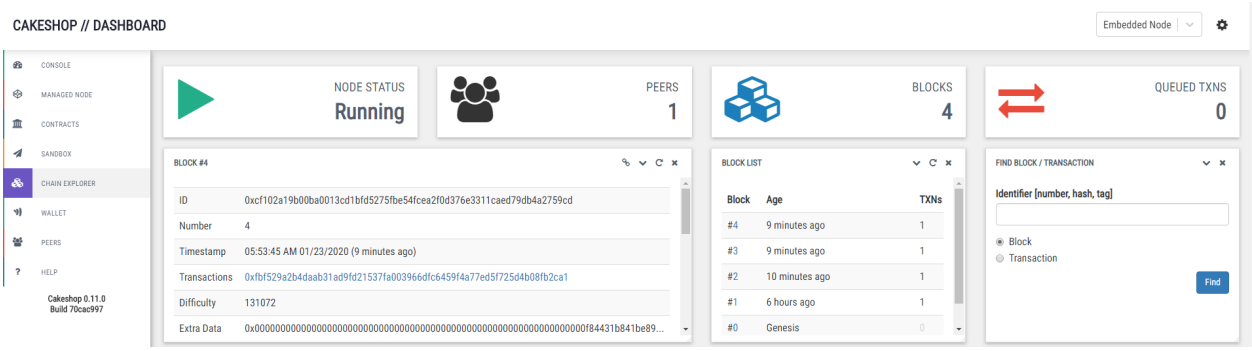
a. 

7. Go back to your CakeShop dashboard and click on CONTRACTS and ensure that your contract is deployed:

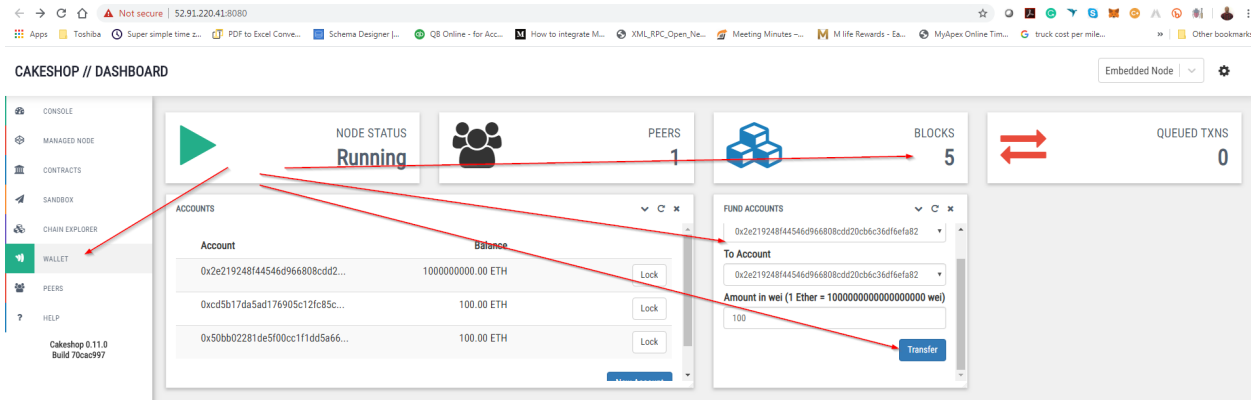
a. http://52.91.220.41:8080/

b. 

8. Click on Chain Explorer and explore the blocklist:



9. Click on Wallets (You could have done this at any time) and transferr 100 from any one wallet to another in “FUND ACCOUNTS”

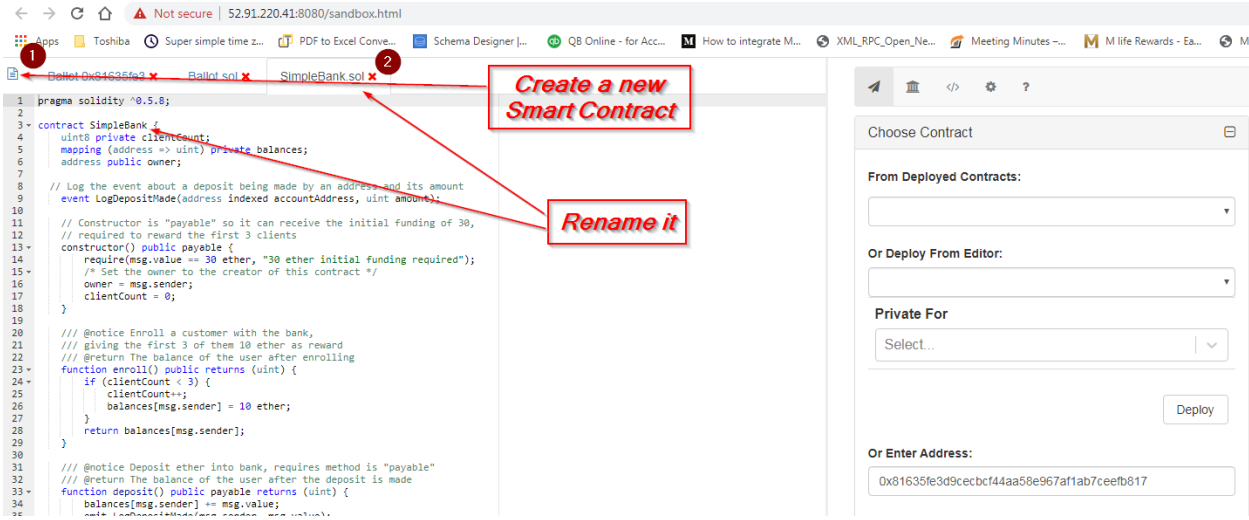
a. 

10. Please repeat the same process with the following code:

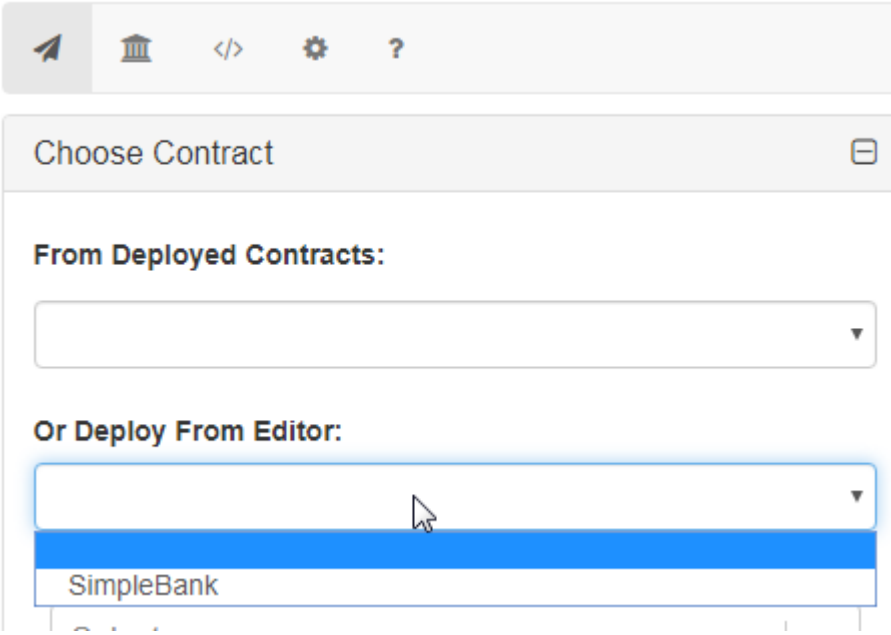
```
pragma solidity ^0.5.8;


contract SimpleBank {
    uint8 private clientCount;
    mapping (address => uint) private balances;
    address public owner;

  // Log the event about a deposit being made by an address and
its amount
    event LogDepositMade(address indexed accountAddress, uint
amount);

    // Constructor is "payable" so it can receive the initial
funding of 30,
    // required to reward the first 3 clients
    constructor() public payable {
        require(msg.value == 30 ether, "30 ether initial funding
required");
        /* Set the owner to the creator of this contract */
        owner = msg.sender;
        clientCount = 0;
    }
```

```
    /// @notice Enroll a customer with the bank,
    /// giving the first 3 of them 10 ether as reward
    /// @return The balance of the user after enrolling
    function enroll() public returns (uint) {
        if (clientCount < 3) {
            clientCount++;
            balances[msg.sender] = 10 ether;
        }
        return balances[msg.sender];
    }

    /// @notice Deposit ether into bank, requires method is
"payable"
    /// @return The balance of the user after the deposit is made
    function deposit() public payable returns (uint) {
        balances[msg.sender] += msg.value;
        emit LogDepositMade(msg.sender, msg.value);
        return balances[msg.sender];
    }

    /// @notice Withdraw ether from bank
    /// @return The balance remaining for the user
    function withdraw(uint withdrawAmount) public returns (uint
remainingBal) {
        // Check enough balance available, otherwise just return
balance
        if (withdrawAmount <= balances[msg.sender]) {
            balances[msg.sender] -= withdrawAmount;
            msg.sender.transfer(withdrawAmount);
        }
        return balances[msg.sender];
    }

    /// @notice Just reads balance of the account requesting, so
"constant"
    /// @return The balance of the user
    function balance() public view returns (uint) {
        return balances[msg.sender];
    }

    /// @return The balance of the Simple Bank contract
    function depositsBalance() public view returns (uint) {
        return address(this).balance;
    }
}
```

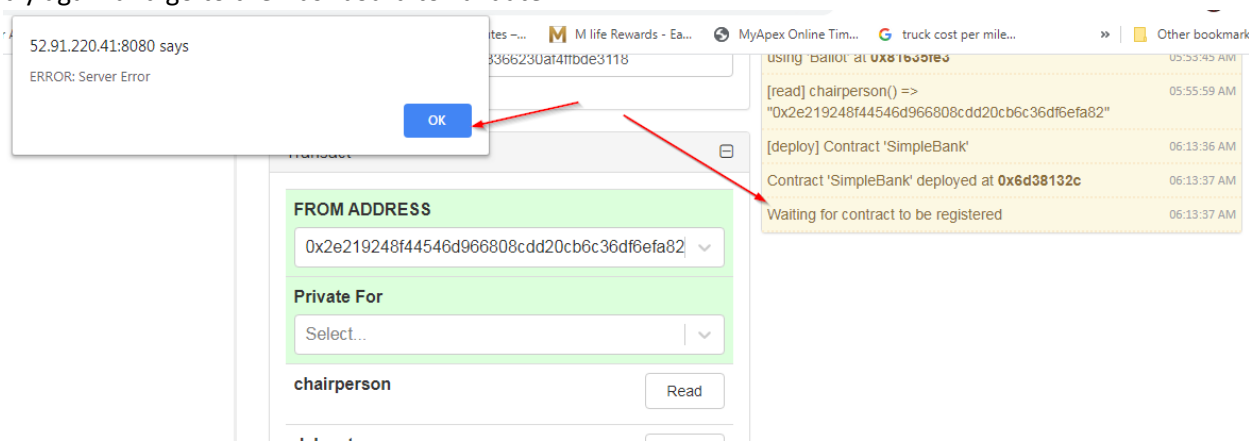
11. Rename the smart contract “SimpleBank.sol” since that is the name of the contract

a. 

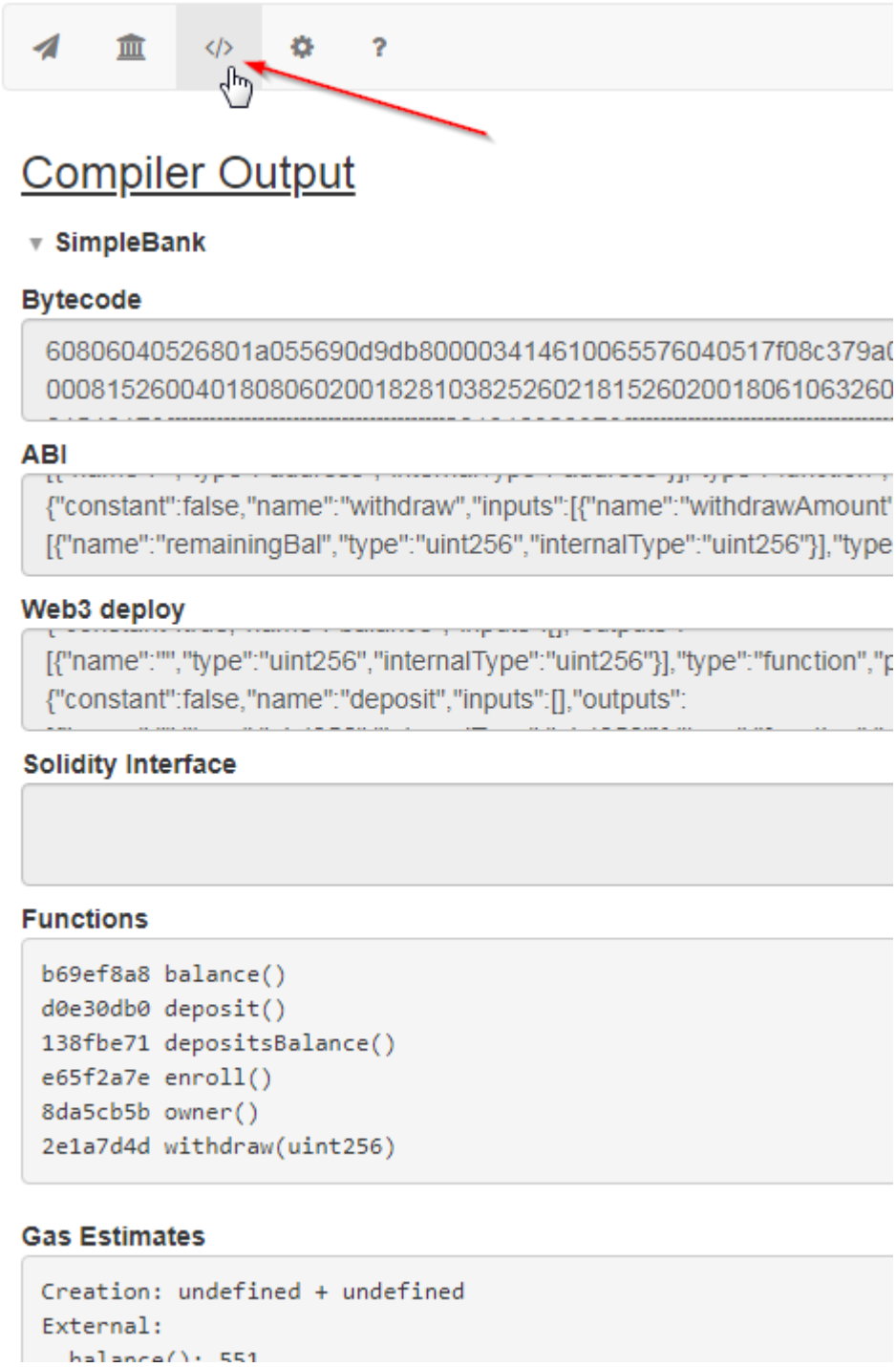
12. Deploy SimpleBank from Editor

a. 

b. Again, if you get an error – it can be ignored if it is waiting to be registered. Otherwise, try again and go to the Dashboard to validate.

c. 

13. Open the Compiler details and take note of the Smart Contract ABI and other information:



You will not be able to interface with this Smart Contract in the CakeShop interface but it has compiled successfully

14. Now, explore the CakeShop Dashboard and the Sandbox! Feel free to search for simple smart contracts on the web and take them in to see if the compile under version 0.5.8 of solidity.
15. Finally, write your own simple smart contract.