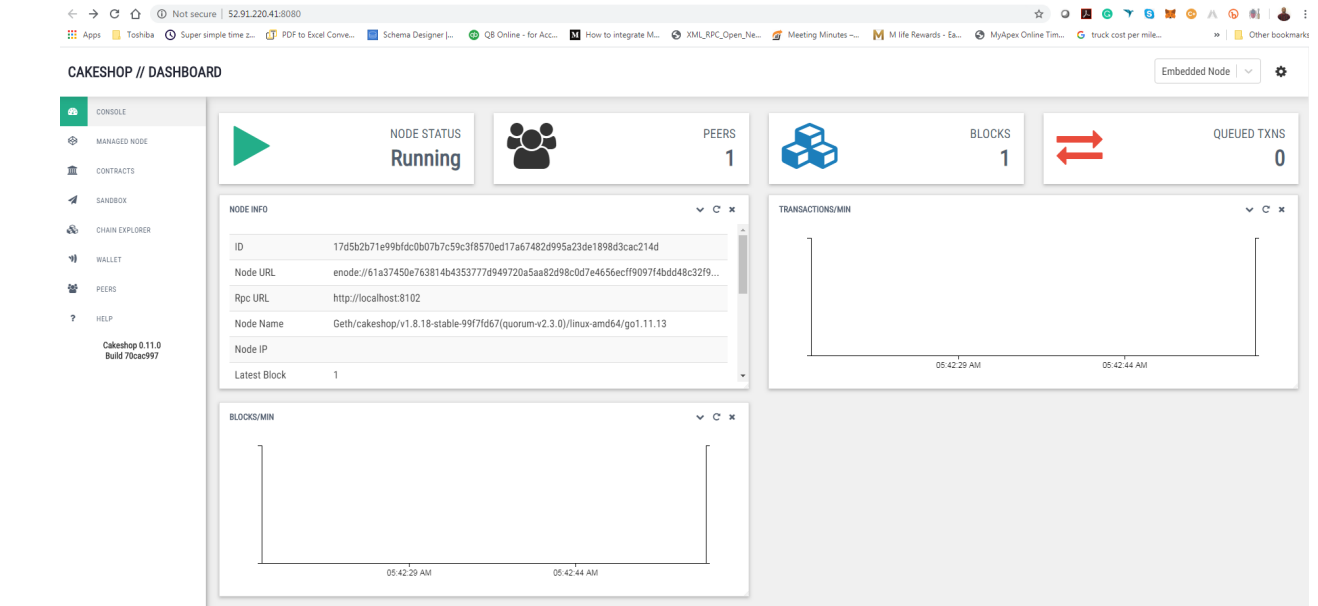


Quorum

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”

The screenshot shows the Cakeshop Sandbox interface. On the left is a Solidity code editor with a contract named 'Ballot.sol'. The code defines a 'Ballot' contract with a 'Voter' struct, a 'Proposal' struct, and functions for creating proposals and voting. On the right is the 'Sandbox' panel. It has a 'Choose Contract' dropdown set to 'Ballot.sol'. Below it are fields for 'From Deployed Contracts', 'Or Deploy From Editor', and 'Or Enter Address'. To the right of these fields is an 'Accounts' table with three entries: '0x2e219248f44546d966...' with a balance of 1000000000.00, '0xcd5b17da5ad176905c...' with 100.00, and '0x50bb02281de5f00cc1f...' with 100.00. Below the accounts table are buttons for 'Contract State', 'Paper Tape', and 'Transact'.

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

Untitled 1

```

1 pragma solidity ^0.5.4;
2
3 /// @title Voting with delegation.
4 contract Ballot {
5     // This declares a new complex type which will
6     // be used for variables later.
7     // It will represent a single voter.
8     struct Voter {
9         uint weight; // weight is accumulated by delegation

```

- 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

Ballot.sol ✖

```

1 pragma solidity ^0.5.4;
2
3 /// @title Voting with delegation.
4 contract Ballot {
5     // This declares a new complex type which will
6     // be used for variables later.
7     // It will represent a single voter.
8     struct Voter {
9         uint weight; // weight is accumulated by delegation
10        bool voted; // if true, that person already voted
11        address delegate; // person delegated to
12        uint vote; // index of the voted proposal
13    }

```

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

📁 🏠 </> ⚙️ ?

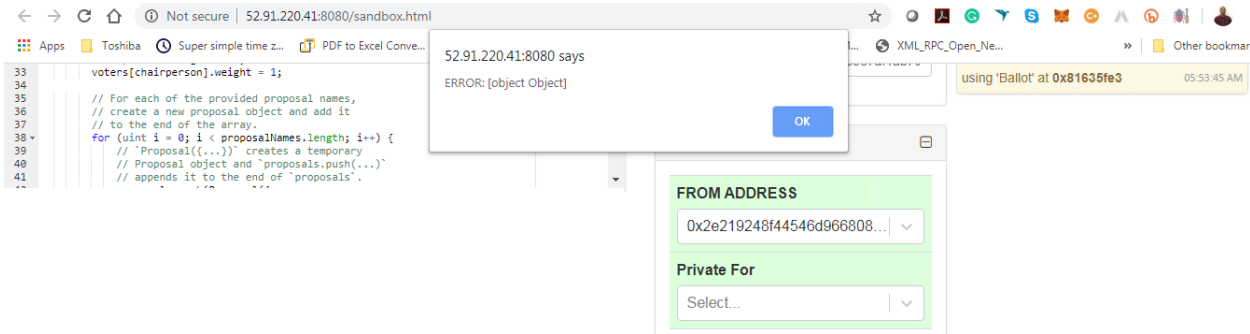
Choose Contract ☰

From Deployed Contracts:

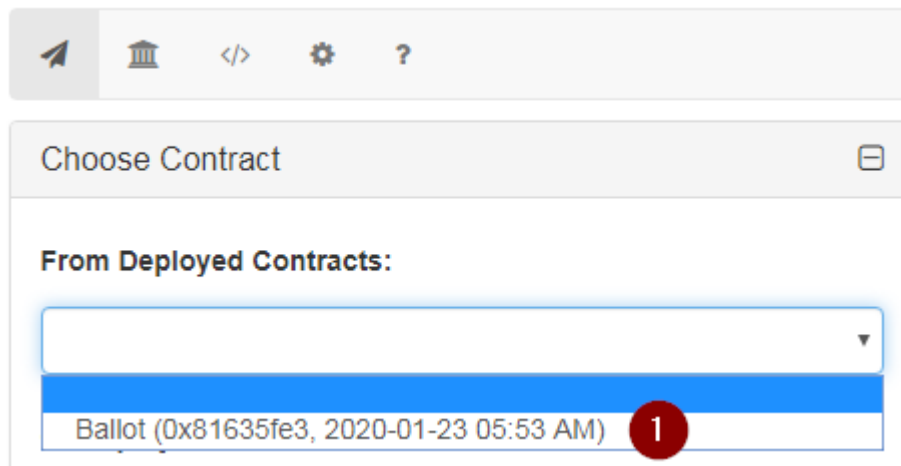
Or Deploy From Editor:

Ballot

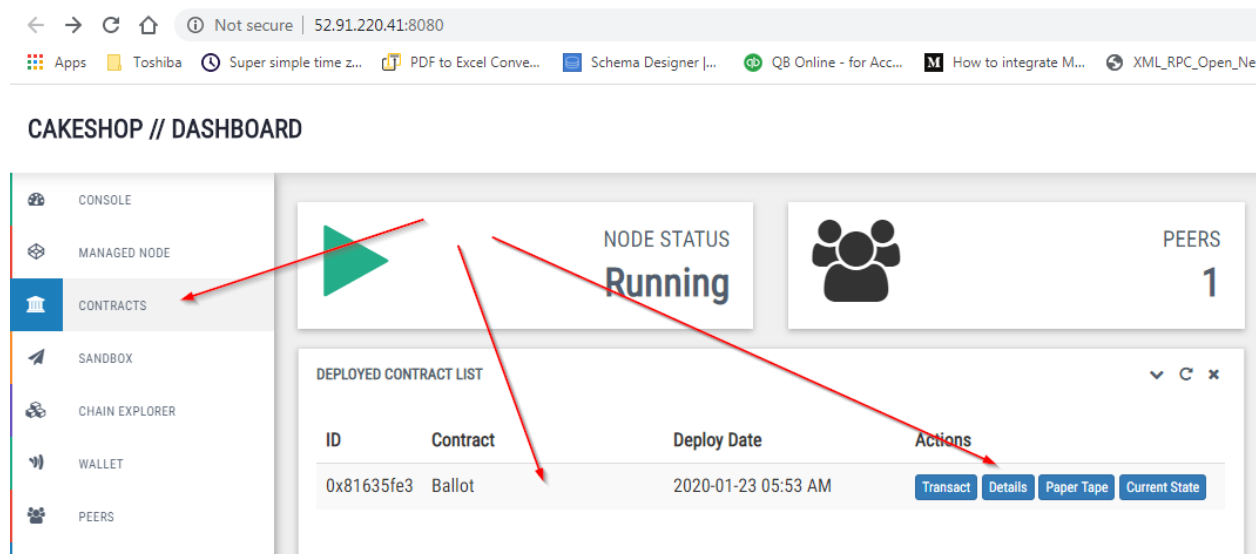
- a.
- b. Click Deploy



- 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:


```

    /// @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];
    }

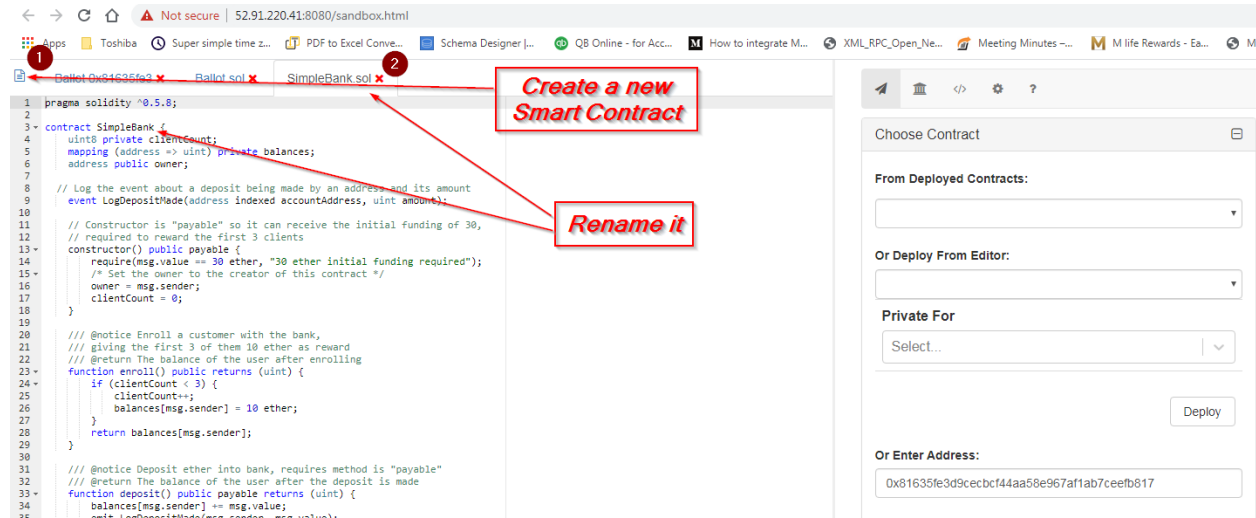
    /// @notice Withdraw ether from bank
    /// @return The balance remaining for the user
    function withdraw(uint withdrawAmount) public returns (uint
    remainingBal) {
        // Check enough balance available, otherwise just return
        balance
        if (withdrawAmount <= balances[msg.sender])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

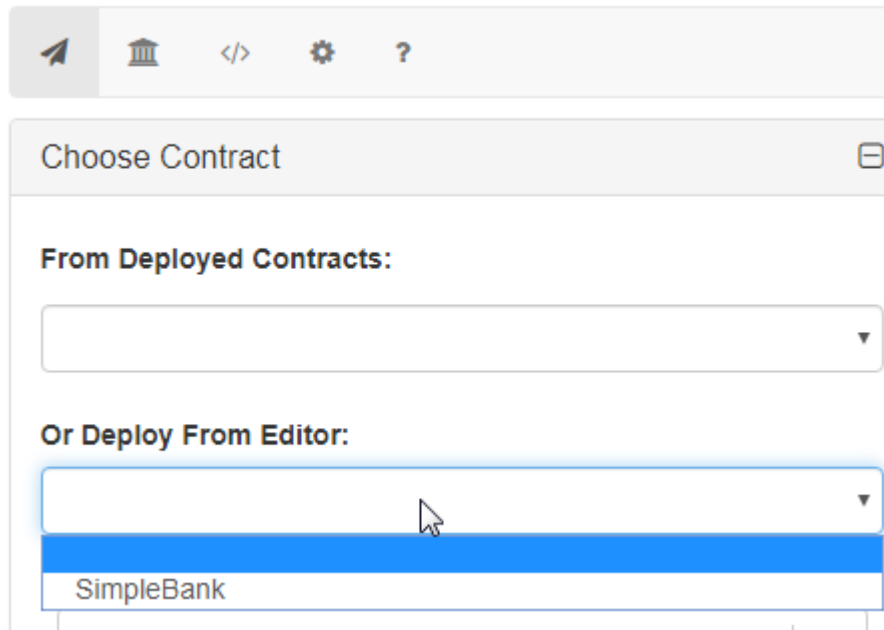
```

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



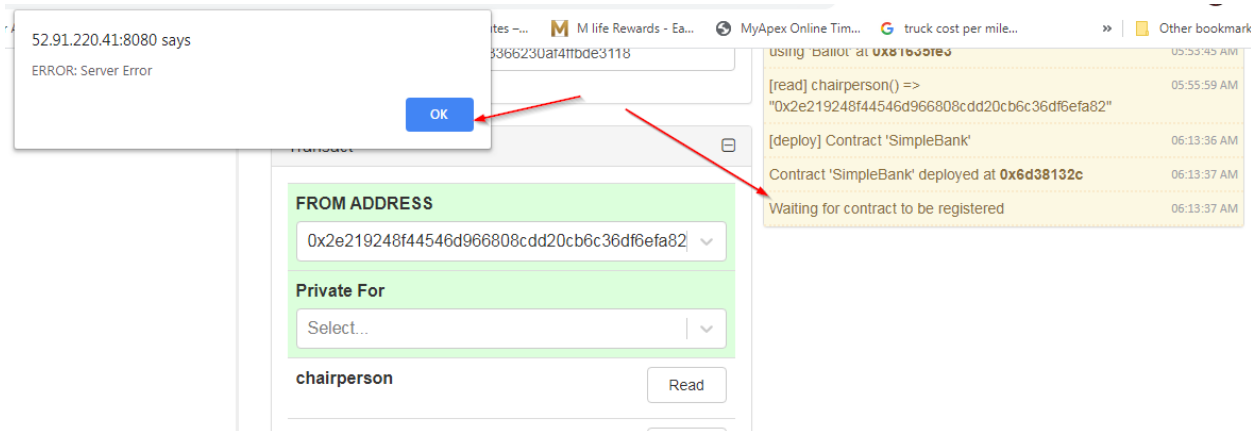
a.

12. Deploy SimpleBank from Editor



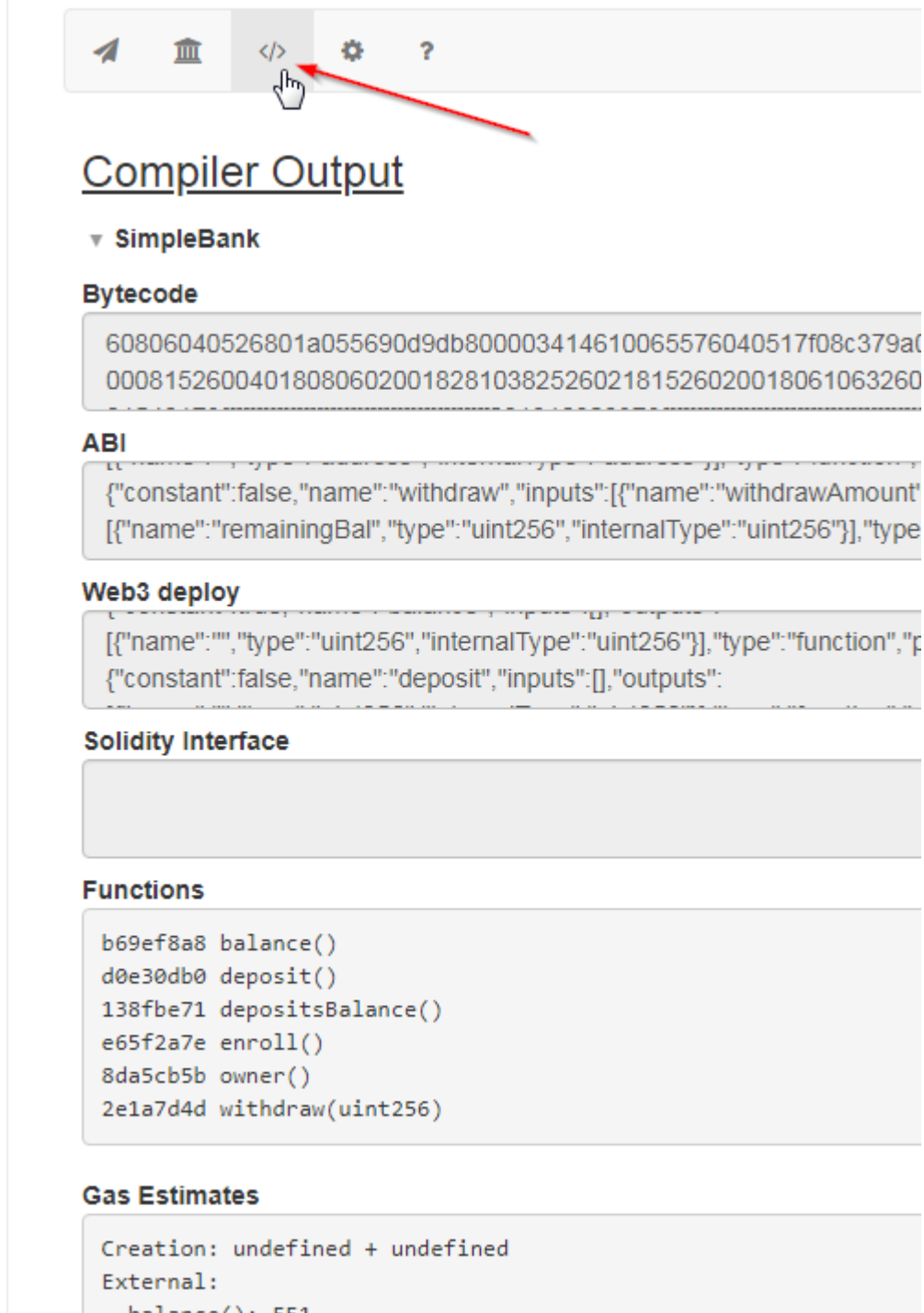
a.

- 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:



The screenshot shows the CakeShop interface with the compiler details for a Smart Contract named **SimpleBank**. The interface includes a top navigation bar with icons for a home page, a bank, a code editor, settings, and help. A red arrow points to the code editor icon. Below the navigation bar, the title **Compiler Output** is displayed. The main content area shows the following information:

- Bytecode**: A long hexadecimal string representing the compiled bytecode.
- ABI**: A JSON array of function definitions, including `withdraw` and `deposit`.
- Web3 deploy**: A JSON array of deployment information, including the `deploy` function.
- Solidity Interface**: A section for the Solidity interface, currently empty.
- Functions**: A list of functions and their corresponding hashes, including `balance()`, `deposit()`, `depositsBalance()`, `enroll()`, `owner()`, and `withdraw(uint256)`.
- Gas Estimates**: A section for gas estimates, showing `Creation: undefined + undefined` and `External: balance() 551`.

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.