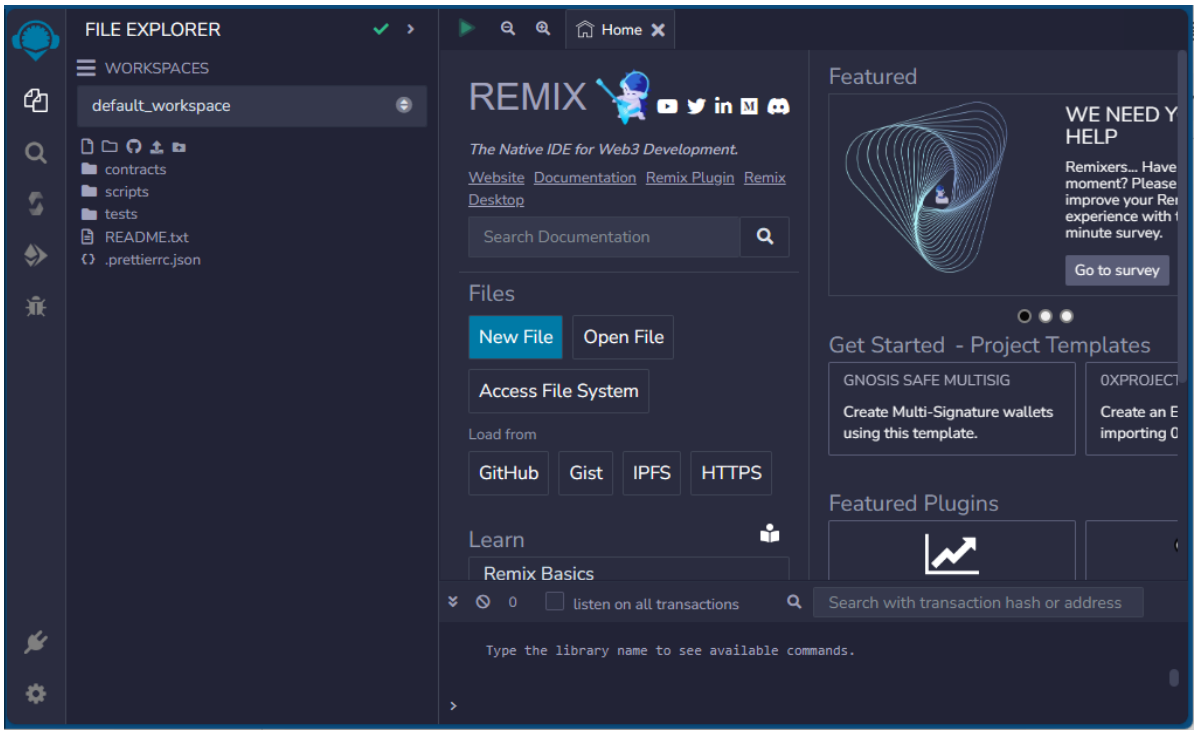
How to Create a CryptoCurrency

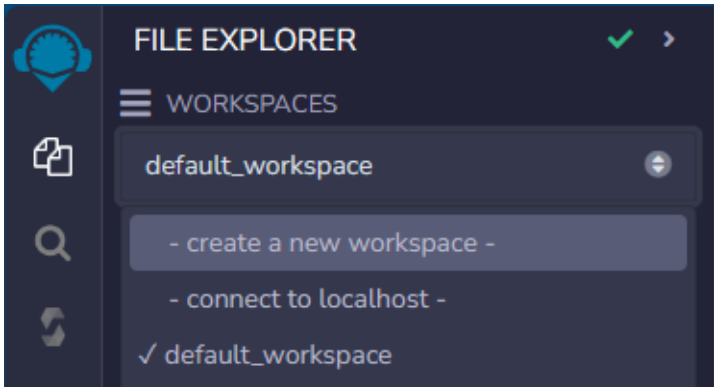
In this guide, you will be creating your first cryptocurrency. It is assumed that you have already installed MetaMask onto your device. If you have not, please refer to the previous guide.

STEP 1: Coding our CryptoCurrency Contract

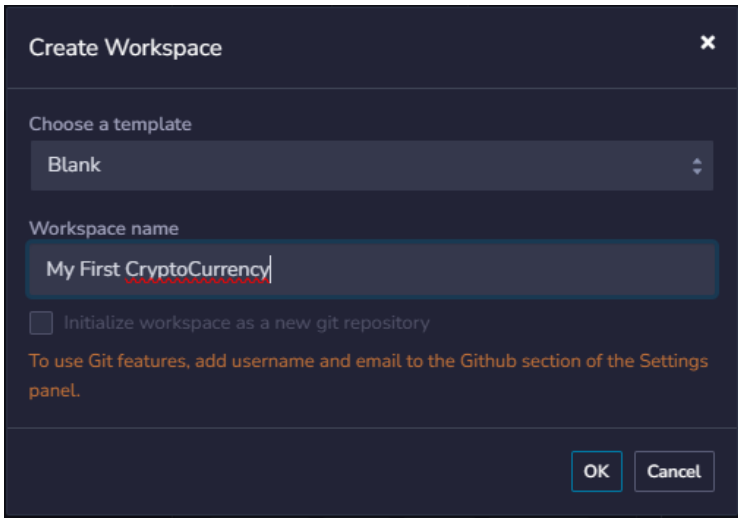
1. To create our first cryptocurrency, we will initially have to access the online IDE (Interactive Development Environment) from the website https://remix.ethereum.org/



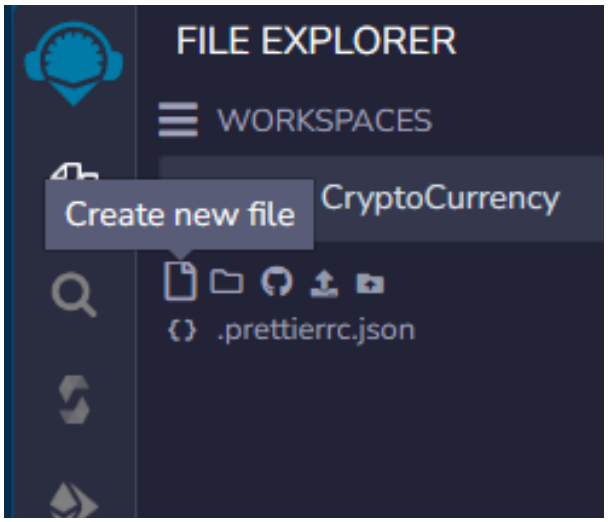
2. We will next create a new workspace. To achieve this, we will click on the **default_workspace** drop down on the upper left corner and select **- create a new workspace -**.



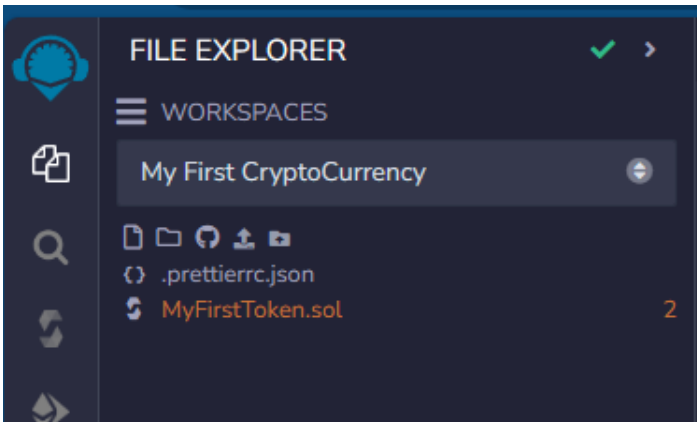
3. Under the **Create a Workspace** dialog, choose **Blank** as your template, and name the workspace as **My First CryptoCurrency.** Then press **OK.**



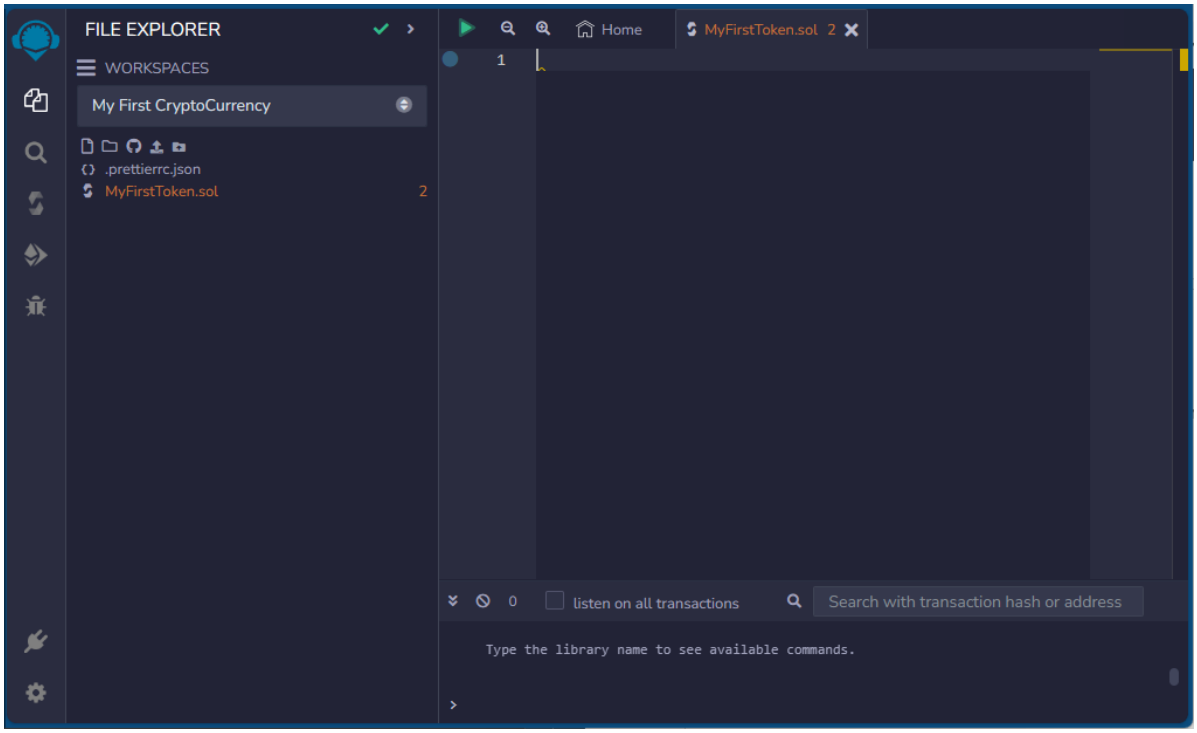
4. Let us create our first smart contract file, which will contain code that interacts with the blockchain. To do this, click on the **Create a new file** icon on the upper left corner of your screen.



5. Name the file as **MyFirstToken.sol**, then press enter to create the file.



6. On the right side, you will see the editor where we will be putting our code.



7. Within the first two lines, put the following code:

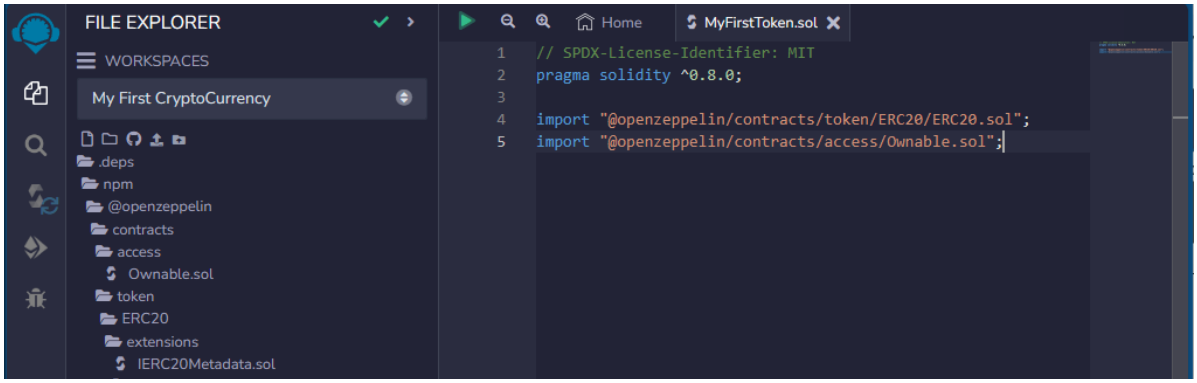
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
```

The first line of code specifies the license, while the second line specifies that we will be using the compiler version 0.8.0 and above

8. We will next import different cryptocurrency libraries to help us in creating our code. Add the two following lines of code:

```
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
```

If you do this correctly, your online IDE should look like the following:



9. We will then create our smart contract. To do this, type the following code below the previous code:

```
contract MyFirstToken is ERC20, Ownable {

}
```

10. Within the curly brackets, add the following lines of code:

```
    constructor() ERC20("OCC Token", "OCC") {
        _mint(msg.sender, 10 * 10 ** 18);
    }

    function mint(uint256 amount) public onlyOwner {
        _mint(msg.sender, amount);
    }

    function burn(uint256 amount) public onlyOwner {
        _burn(msg.sender, amount);
    }
```

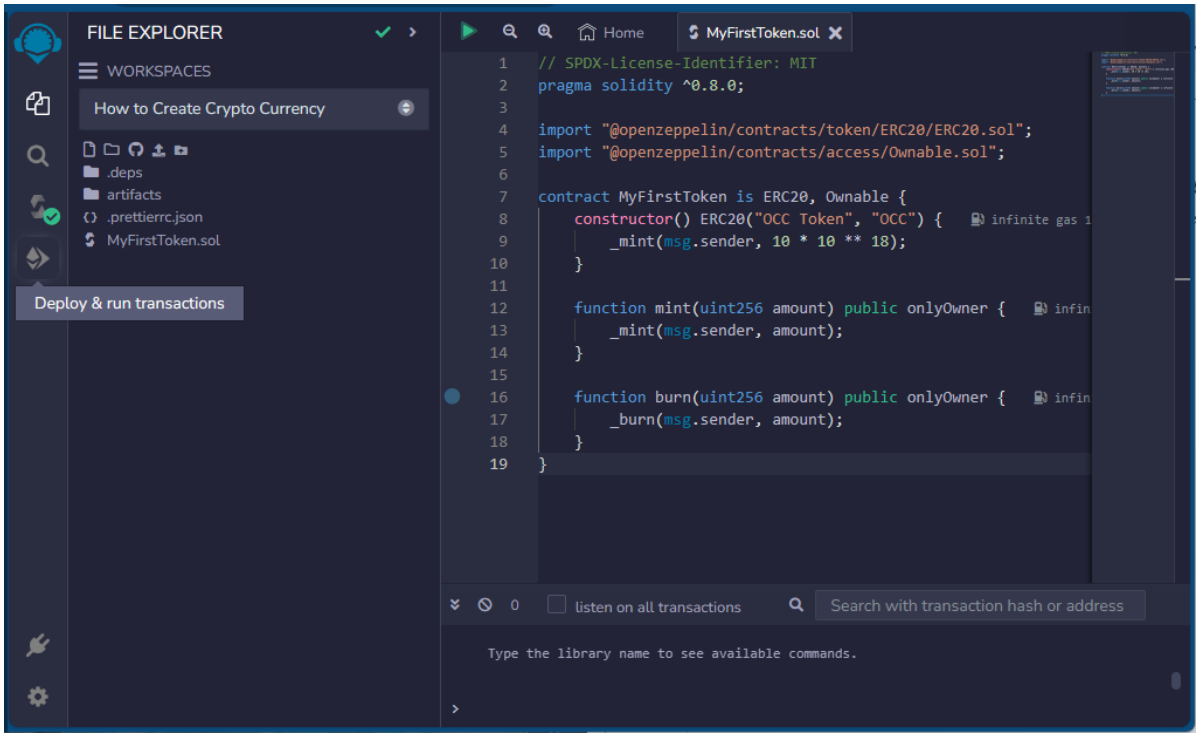
Notice that we have three different function declarations in our code:

1. The first function declaration is what we call a constructor which sets the initial state of our contract.
    a. The two arguments “OCC Token” and “OCC” refer to the token name and the token symbol respectively in the ERC20 constructor
    b. Inside the function, we call on the **_mint** function from the parent ERC20 function and notice that we have two arguments:
        i. **msg.sender** - This is the person or wallet currently interacting with the contract
        ii. **10 * 10 ** 18** - This is how many tokens we initially mint
2. The second function, **mint**, likewise calls the **_mint** function, but this time we specify an amount that will be added to the number of tokens available
3. The last function, **burn**, calls on the **_burn** function within the parent ERC20 contract but does the opposite of minting by decreasing the number of tokens available.

11. Press **CTRL+S** in order to save and compile your code.

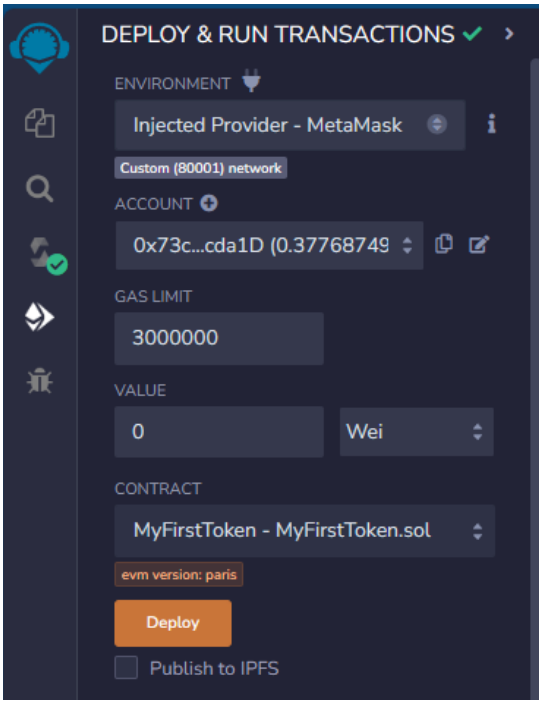
STEP 2: Testing our CryptoCurrency

1. To test our cryptocurrency, we will have to deploy this via MetaMask. Go to the **Deploy & Run Transactions** button on the side

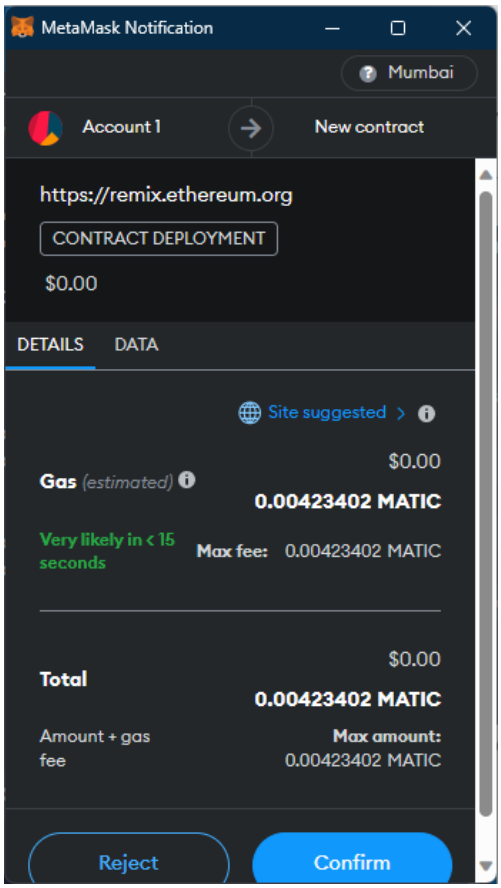


2. Under the **Environment** tab, choose **Injected Provider - MetaMask**

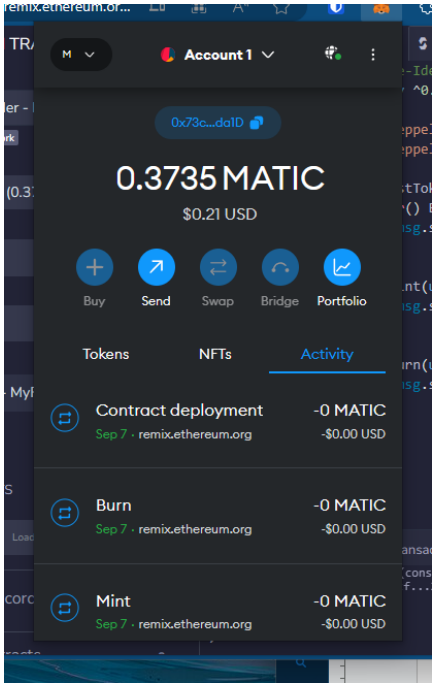
3. Click on the orange **Deploy** button at the bottom of the page



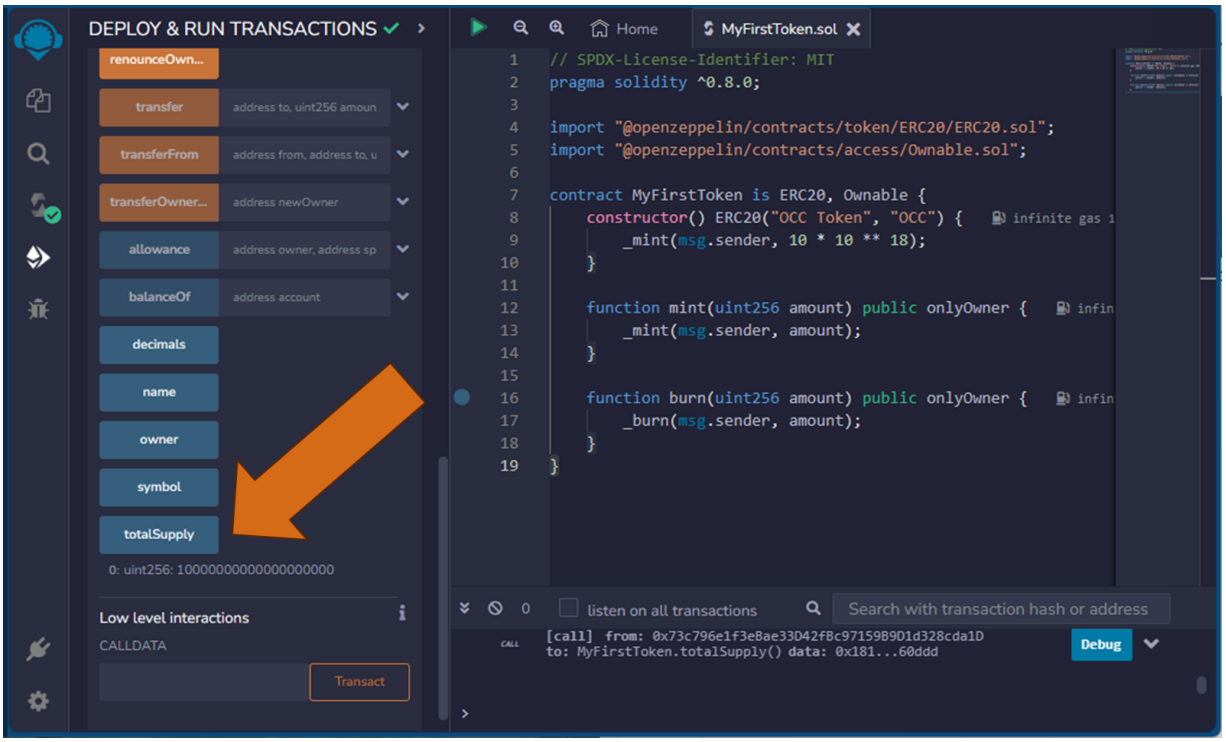
4. Doing so will open **MetaMask,** and just click on the **Confirm** button to make the transaction push through on the blockchain.



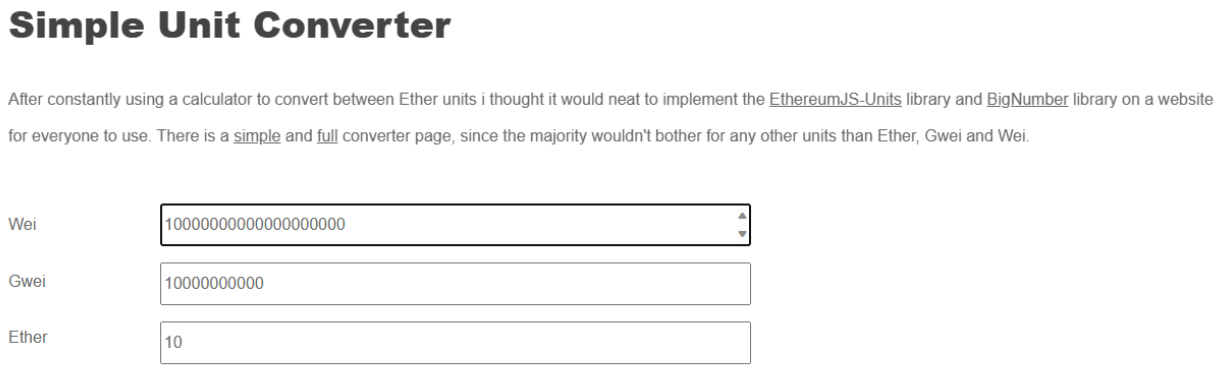
5. Once successful, it should say that the contract was successfully deployed



6. Let us view the total supply of our deployed contract by clicking on the **totalSupply** button at the bottom

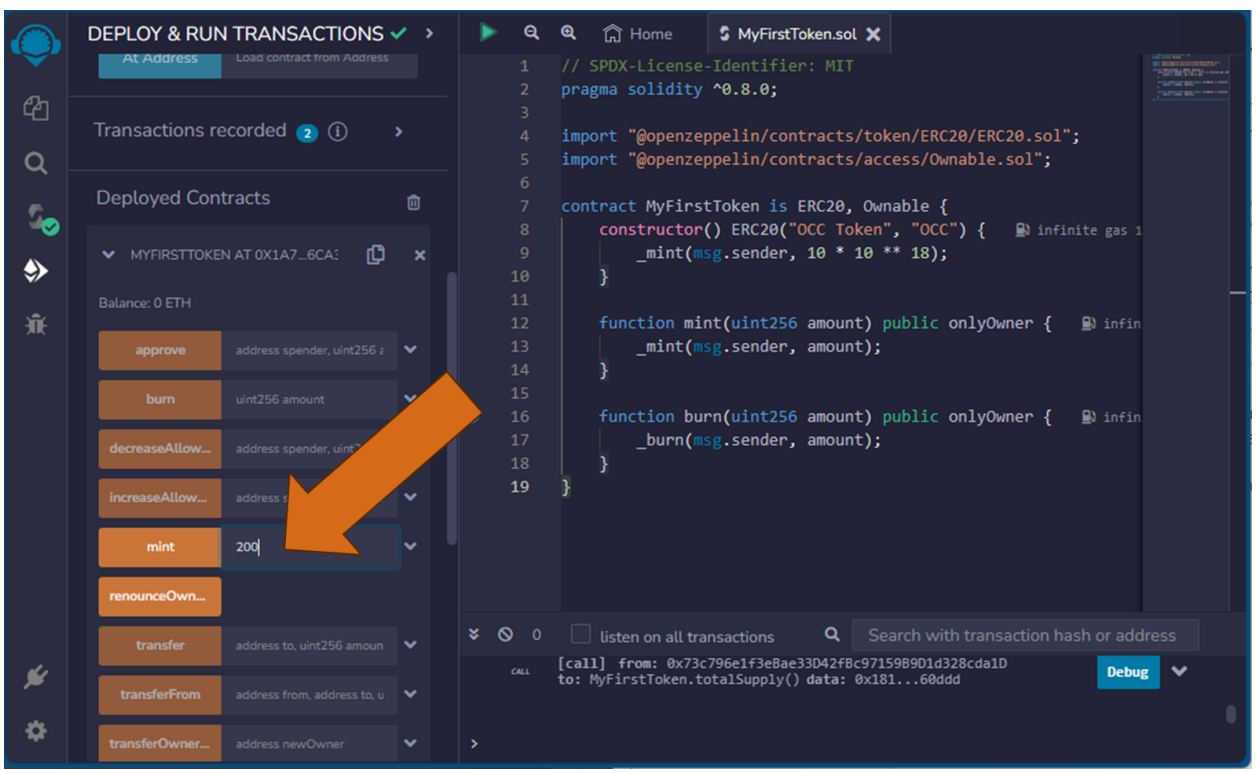


7. To verify how much we initially minted, we can copy the value at the bottom and paste the number that is output into https://eth-converter.com/ under the **Wei** text field.



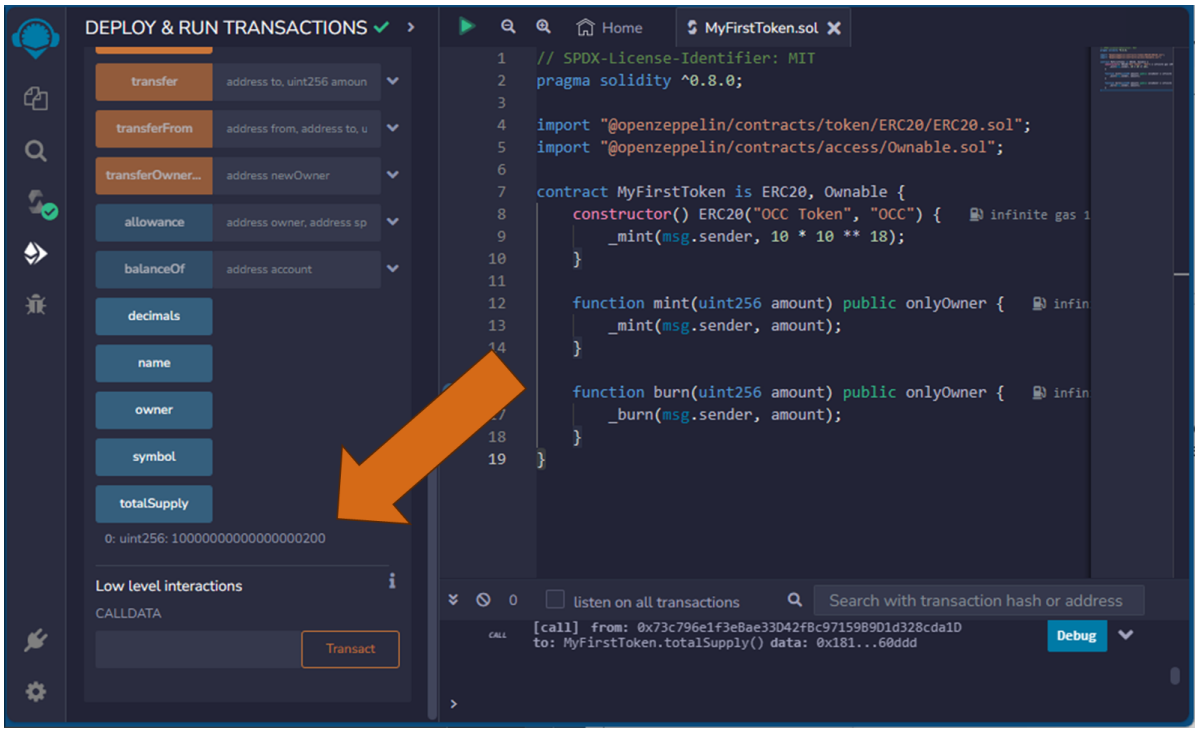
Wei refers to the smallest unit of the token, which you notice has 18 zeros based on our initial 10^18 we declared earlier. We then multiplied this by 10, which is why we have 10 ETH.

8. Let us test the mint functionality of our cryptocurrency. Look for the **mint** button and input an amount (e.g. 200). Click on the **Mint** button afterwards and confirm the transaction in MetaMask.

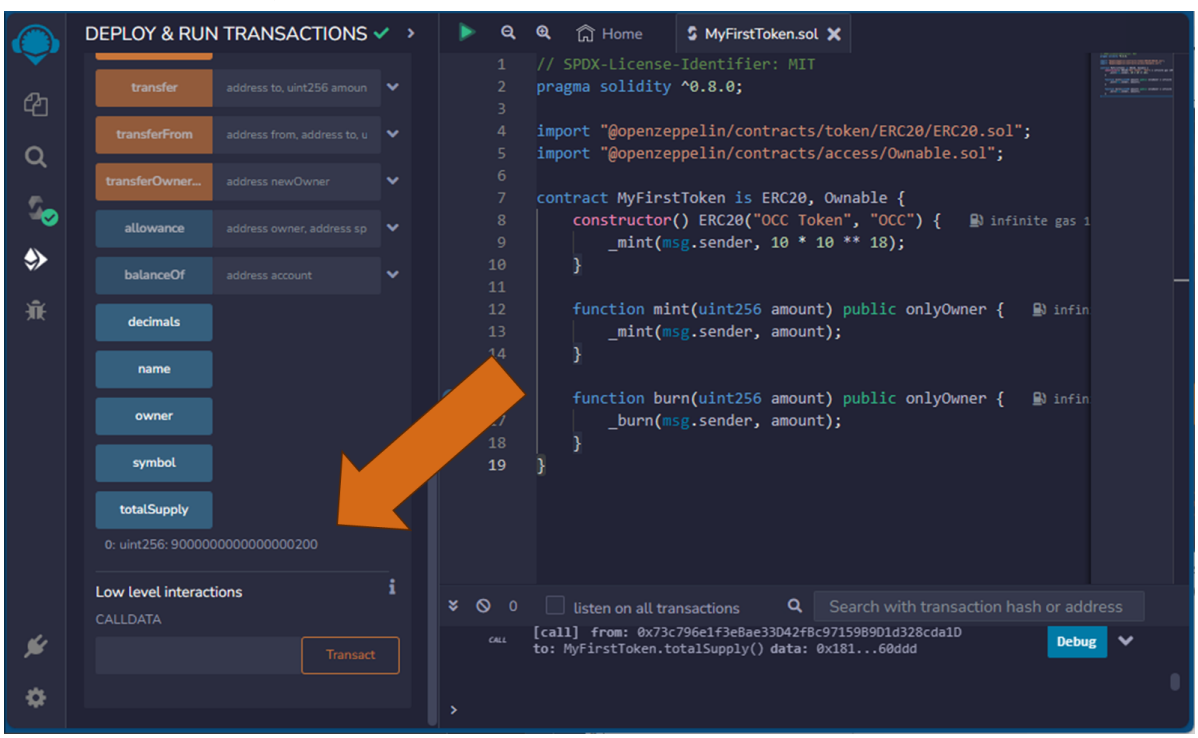


Note that the value we input refers to Wei which will be minted into the cryptocurrency.

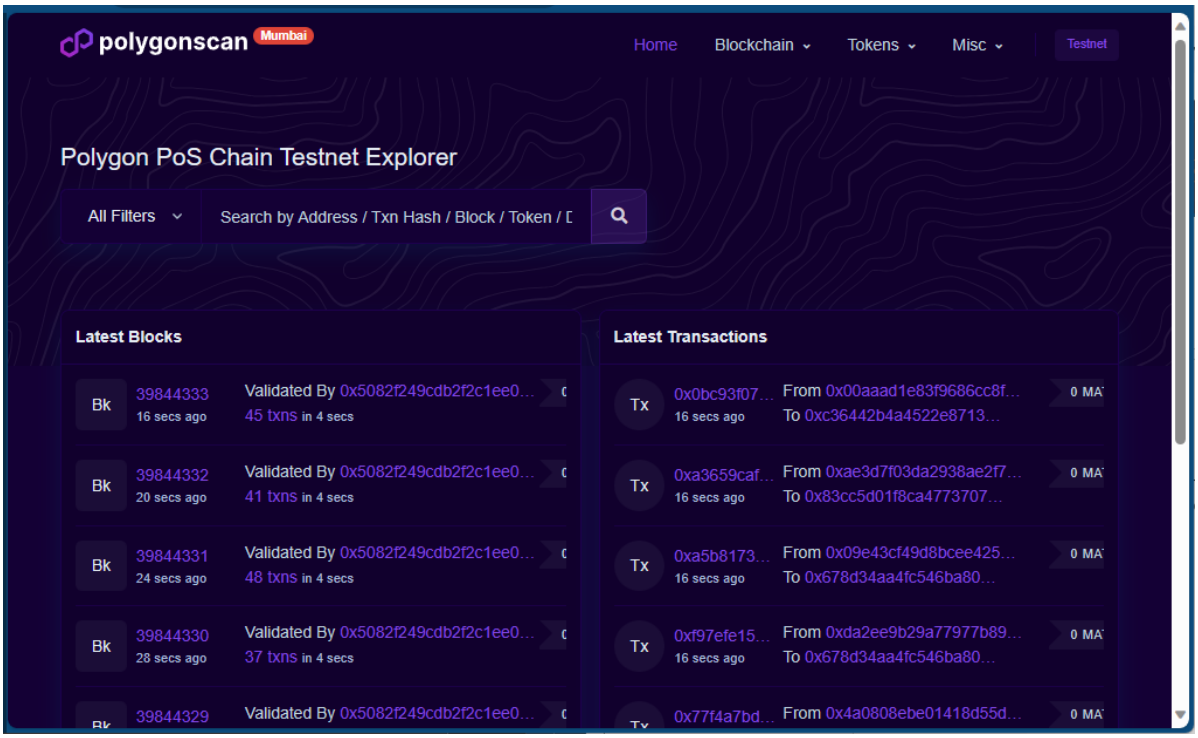
9. Click on the **totalSupply** button again, and notice that the value has now been updated with the value we minted.



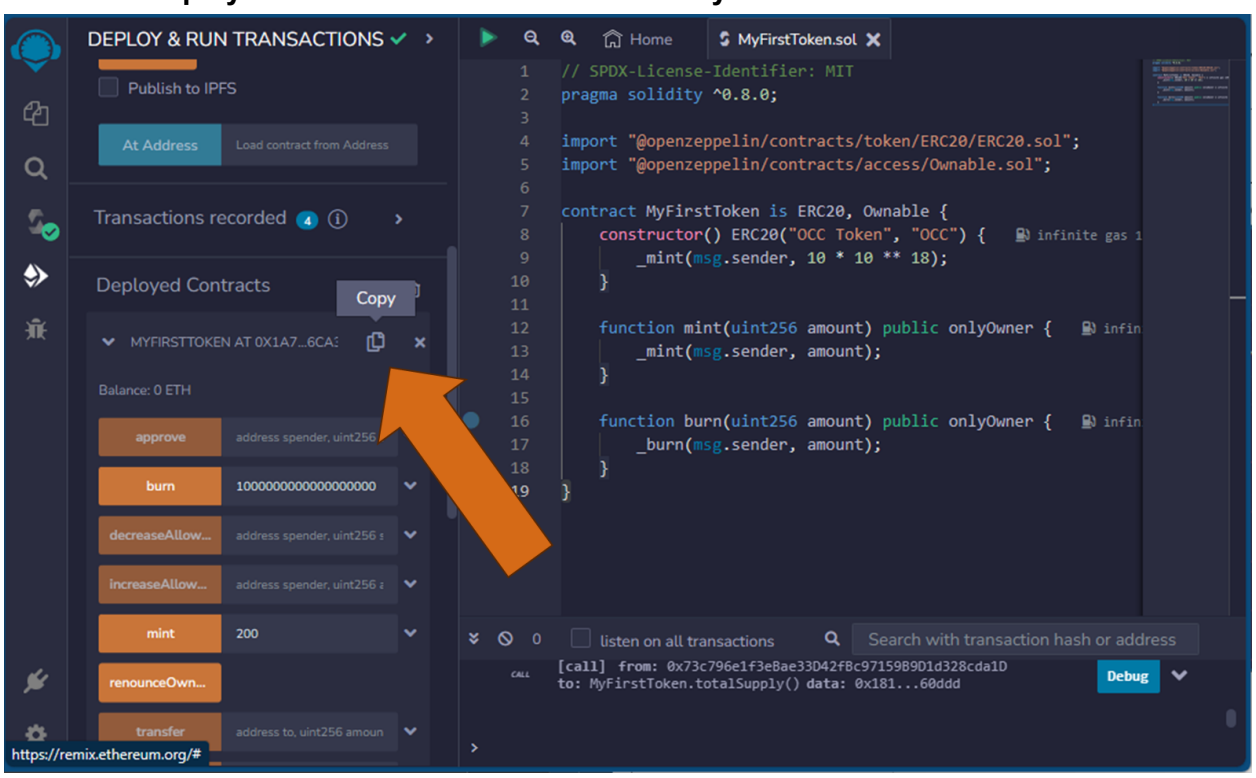
10. Likewise, we can choose a burn by choosing the **burn** button and applying a value (e.g. 1000000000000000000 or 1 ETH). Afterwards, click on the **burn** button and confirm the transaction.
11. Notice that now, we have 9 ETH and 200 Wei as our balance



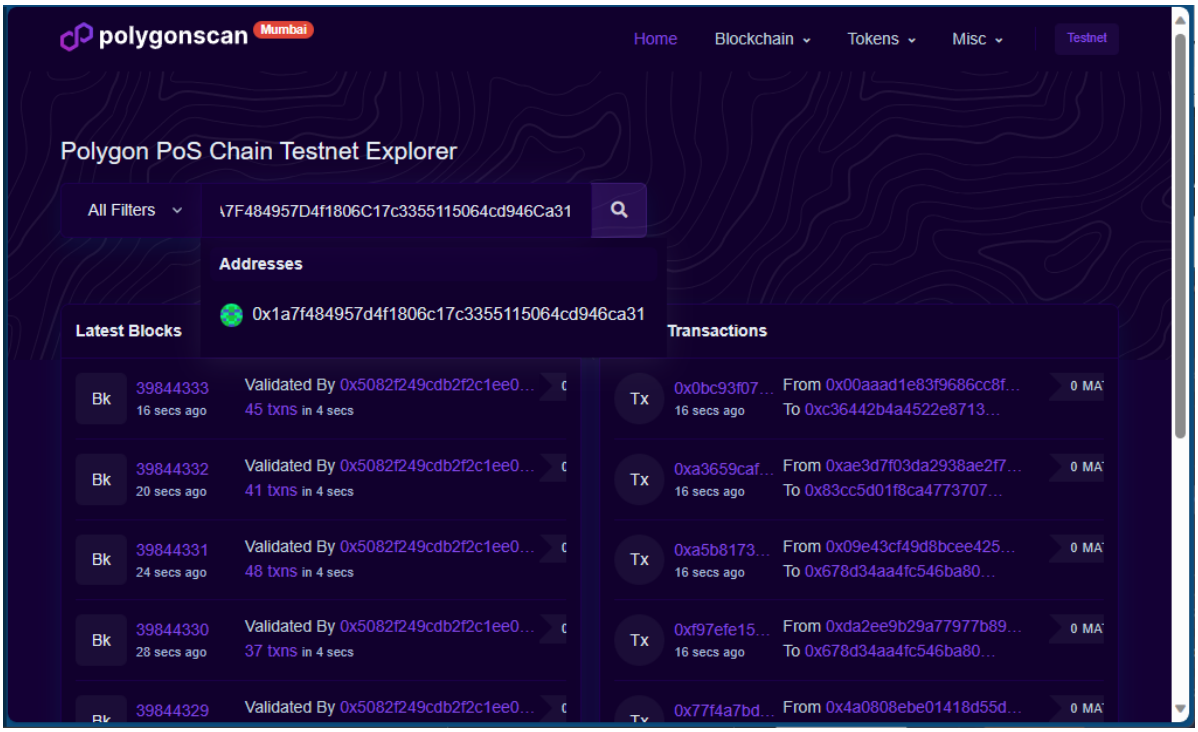
12. We can further see our transaction through the site https://mumbai.polygonscan.com/, assuming that you used the Mumbai token.



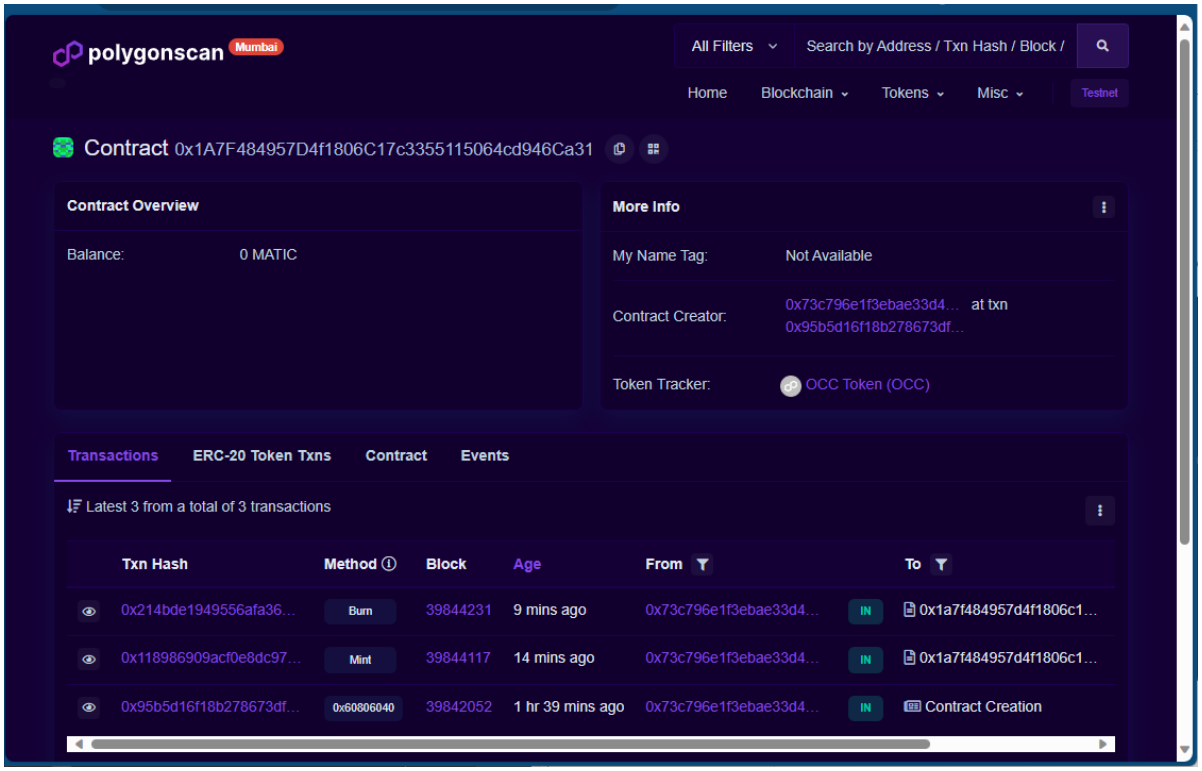
13. To find our smart contract and its transactions, you'll have to copy the transaction address of your smart contract, which can be done by clicking on the copy button under the **Deployed Contracts** area with the name **MyFirstToken**.



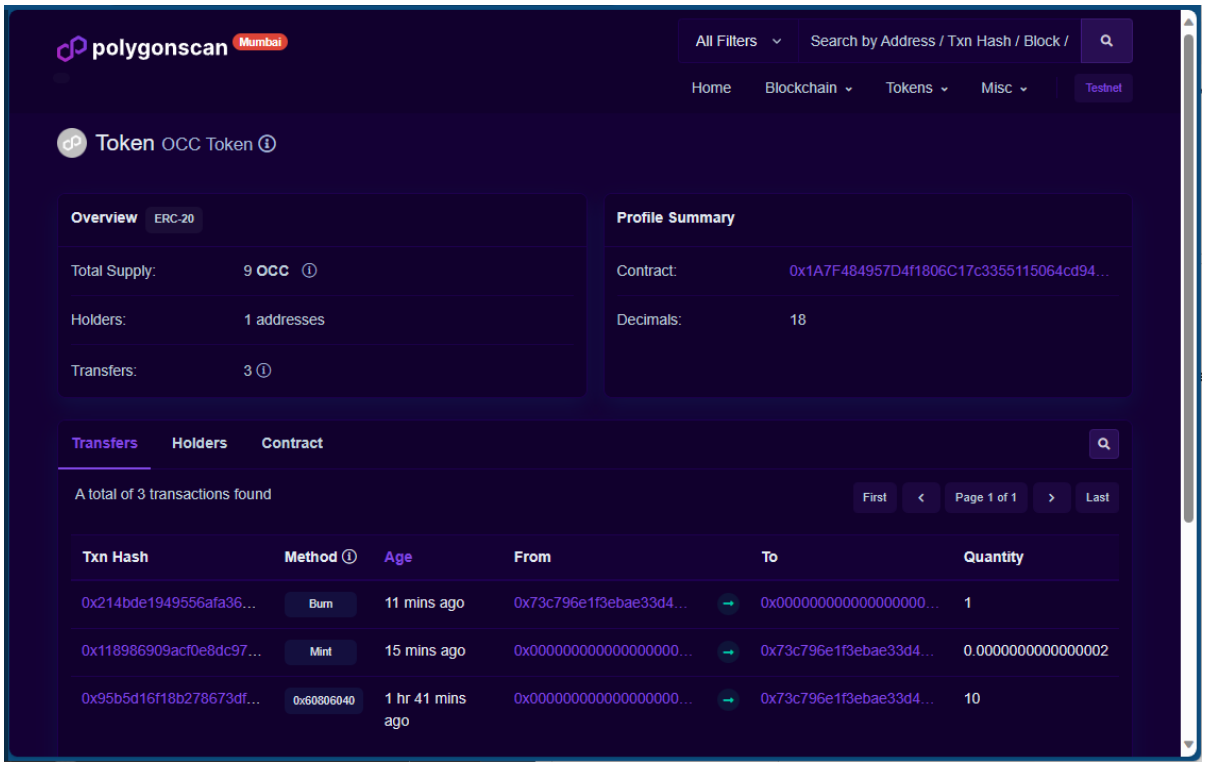
14. Paste this copied contract address under the search field of the polygon scan website as seen below:



15. You will then be able to see your transactions



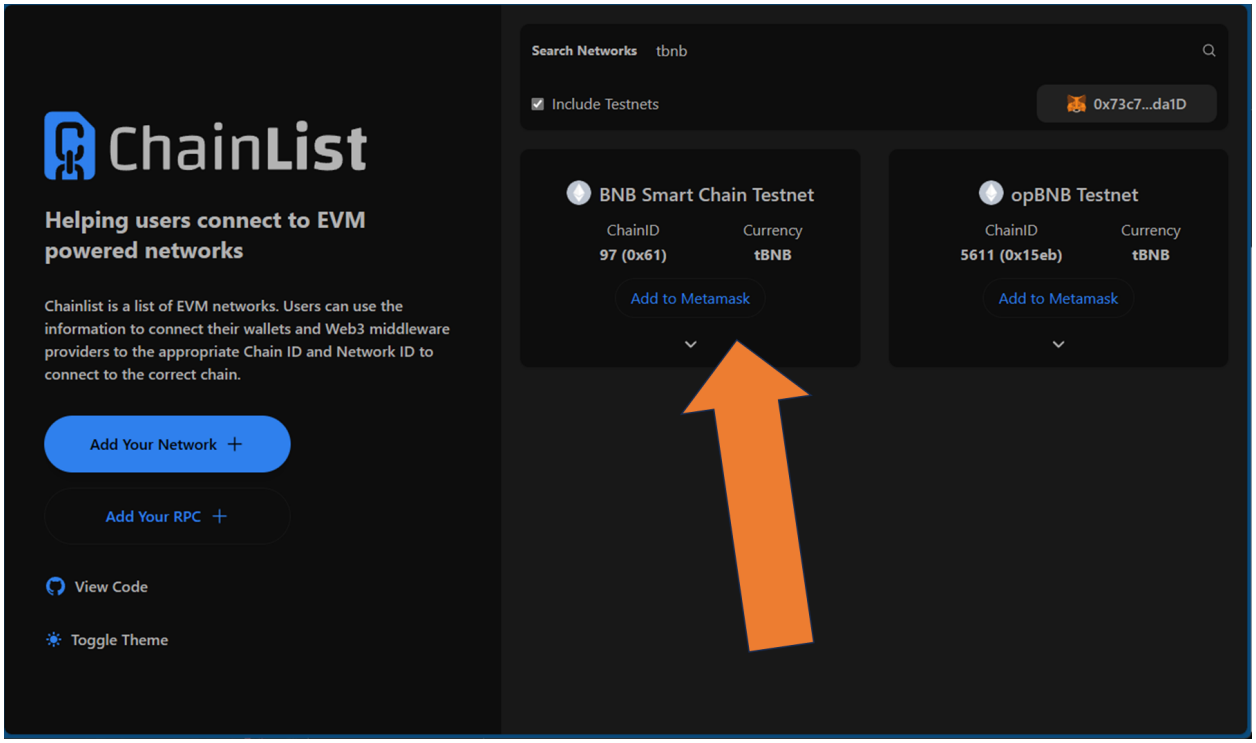
16. You can also click on the **Token Tracker** button to see more information about your token



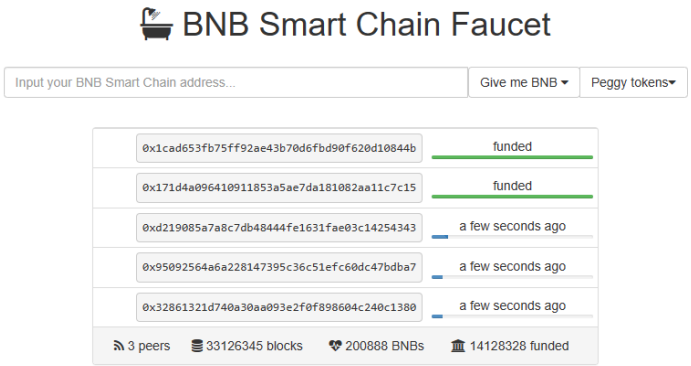
17. Congratulations, you have now created, deployed, and validated your first cryptocurrency!

STEP 3: Adding Liquidity to your Cryptocurrency

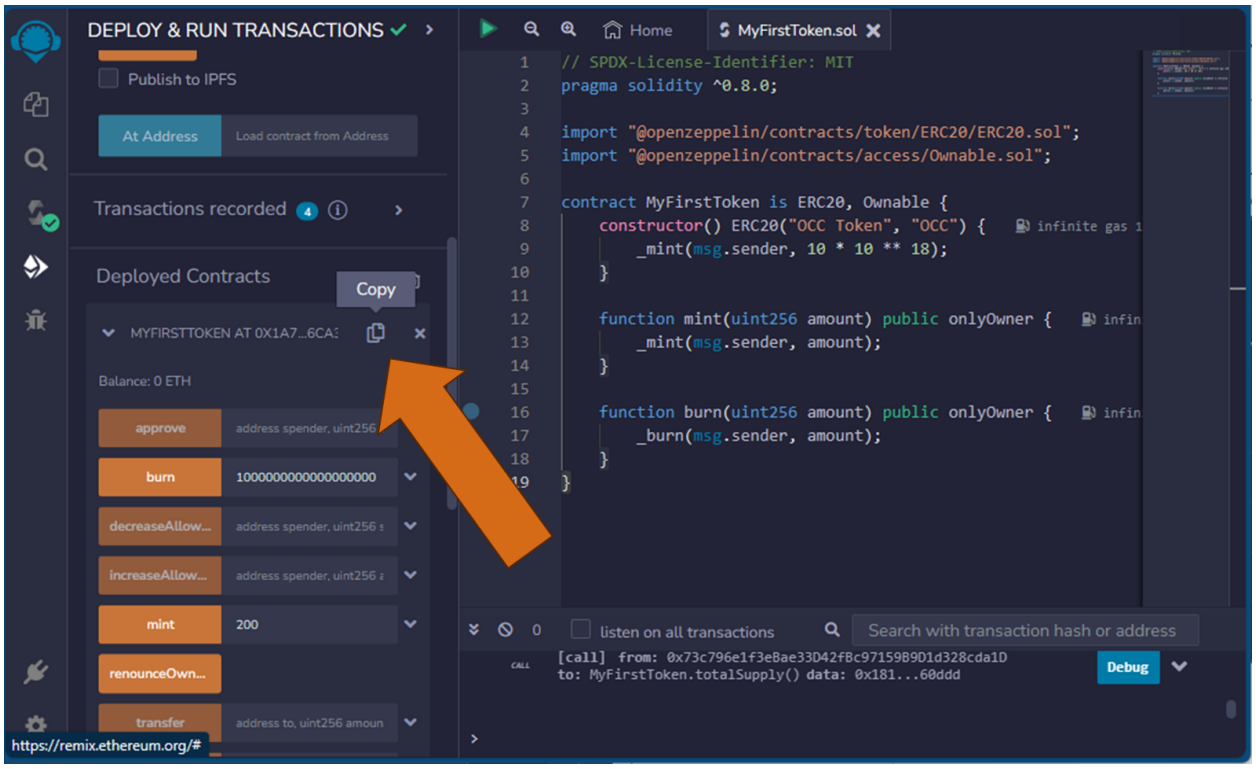
1. To add liquidity to your cryptocurrency, you will first have to add the test bnb test network onto your MetaMask. You can do this by going to https://chainlist.org/ and searching for **tbnb** with the **include testnets** checkbox selected.



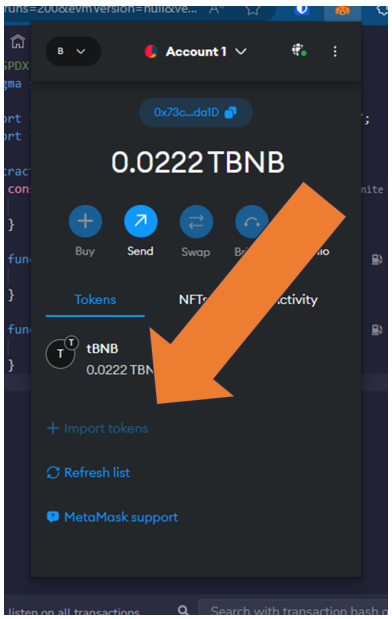
2. Click on **Add to MetaMask**, and then go through the different prompts to successfully add this test network.
3. To add funds to our wallet, go to https://testnet.bnbchain.org/faucet-smart and paste your wallet address to receive **tBNB** funds

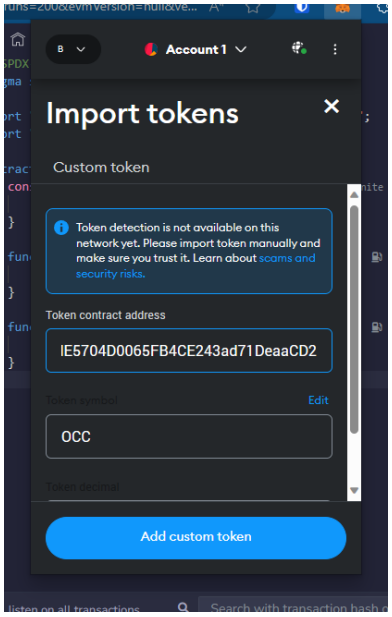


4. Next, we will import the token in Metamask. Within the remix IDE, copy-paste the smart contract address.

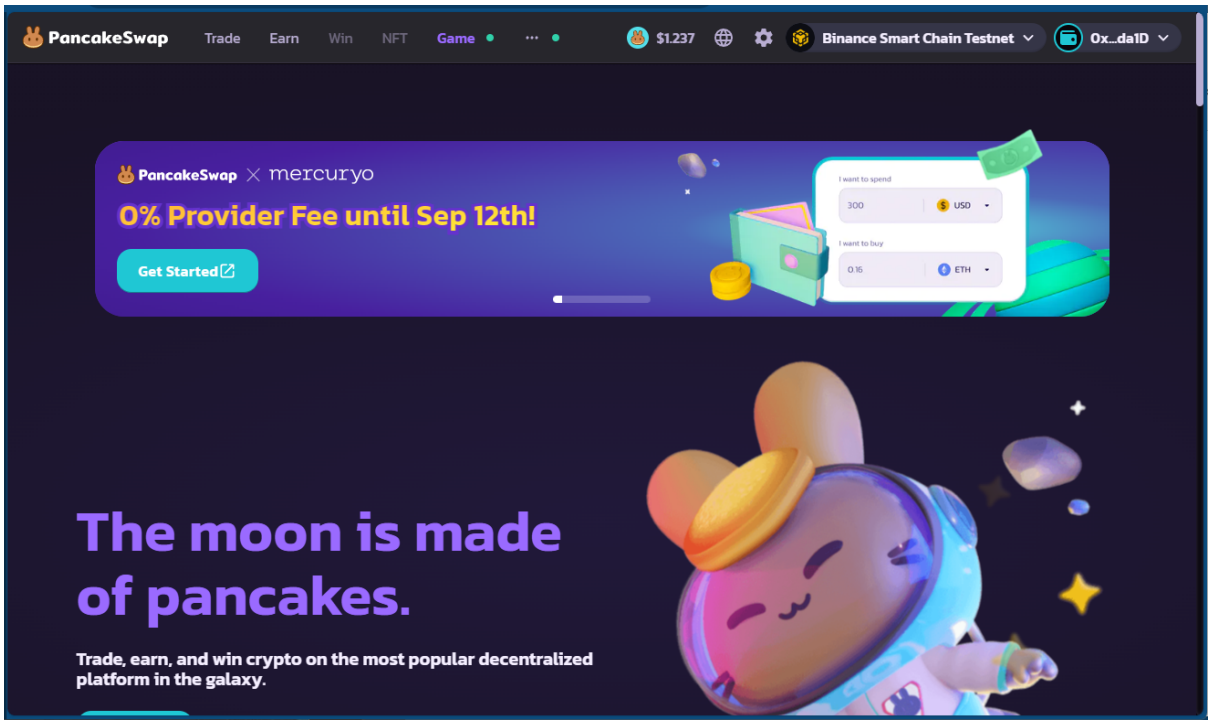


5. Open MetaMask and go to the **Tokens** tab. Click on **Import Token**, and paste the token address there. The field OCC should be automatically filled. Then, click on **Add Custom Token** and then **Import Token**.

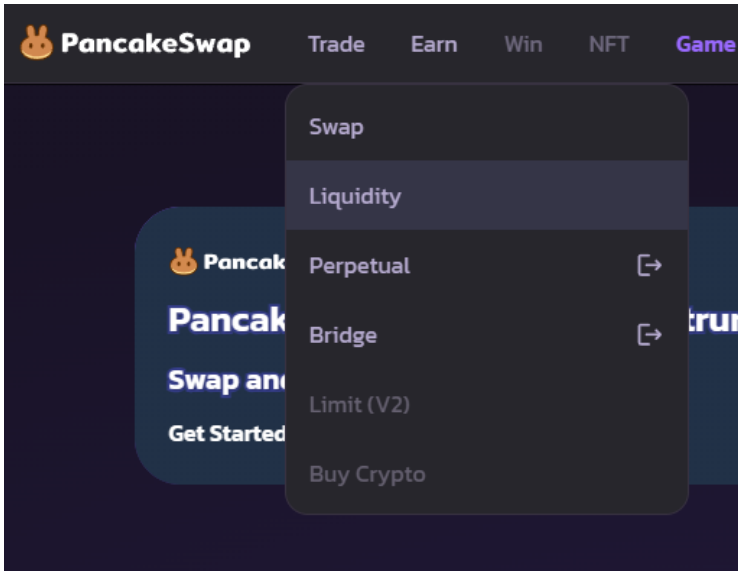




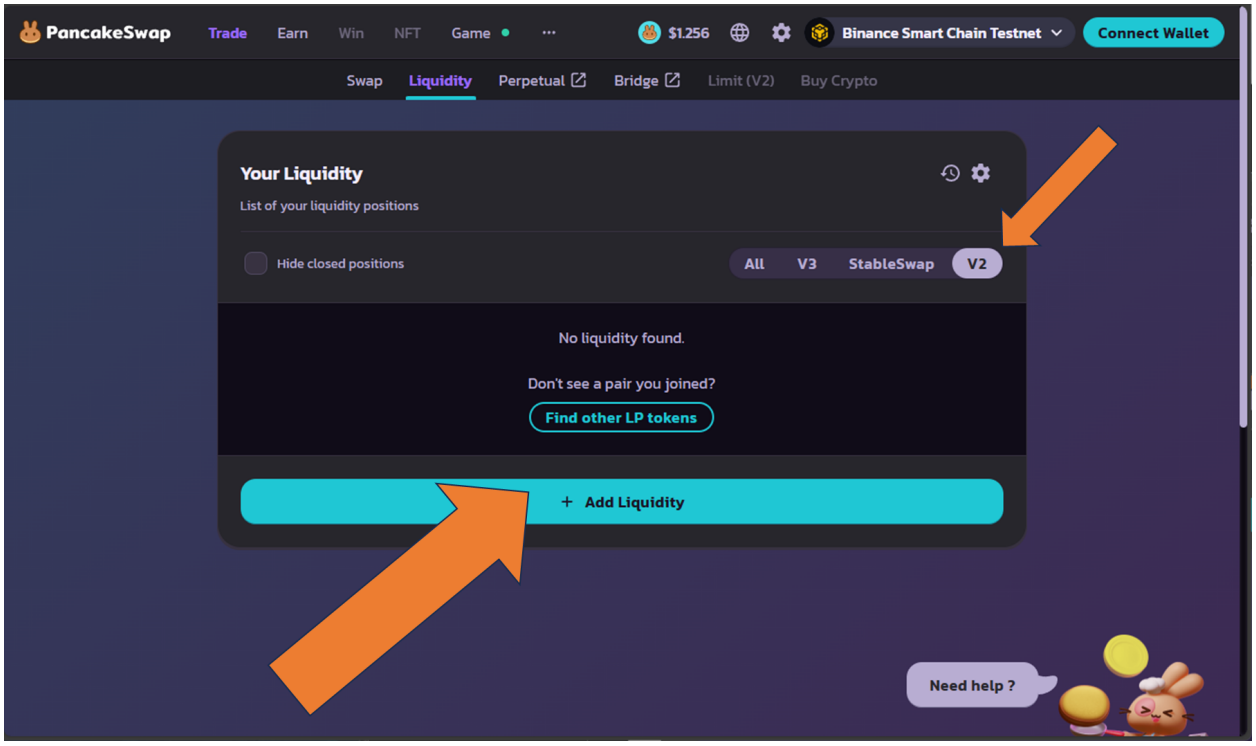
6. Go to link https://pancakeswap.finance/?chain=bscTestnet to be able to add liquidity to our token



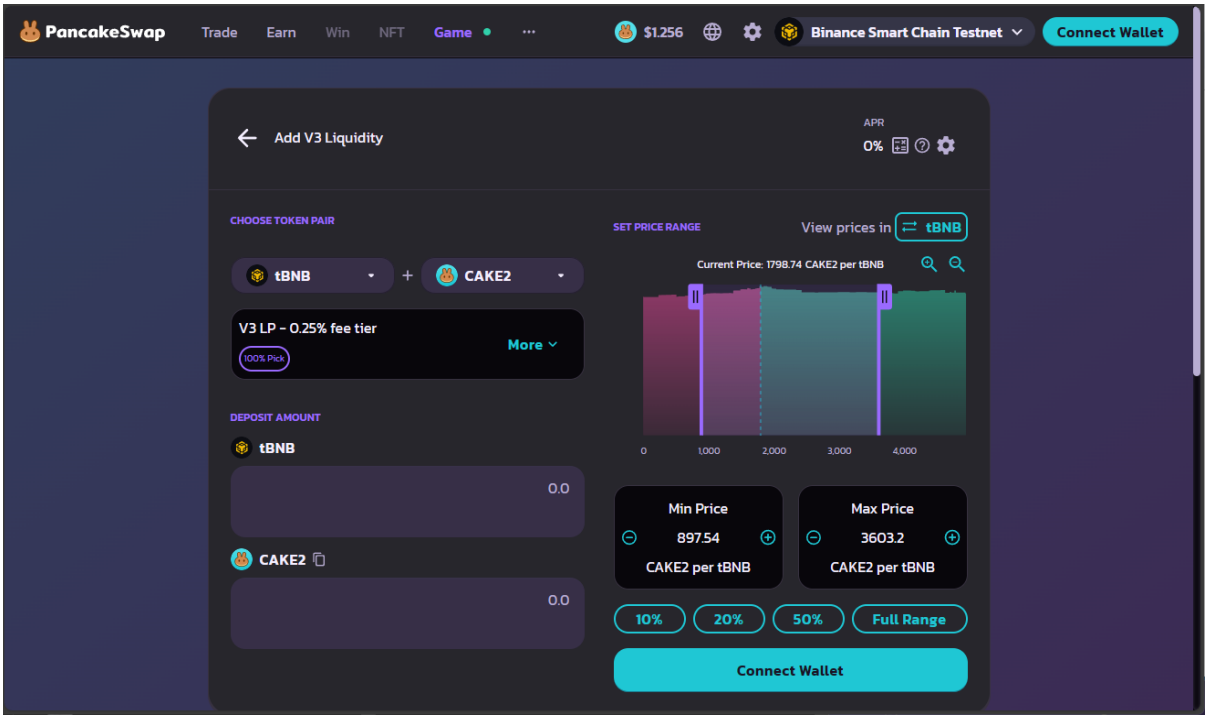
7. Within the website, go to **Trade > Liquidity**

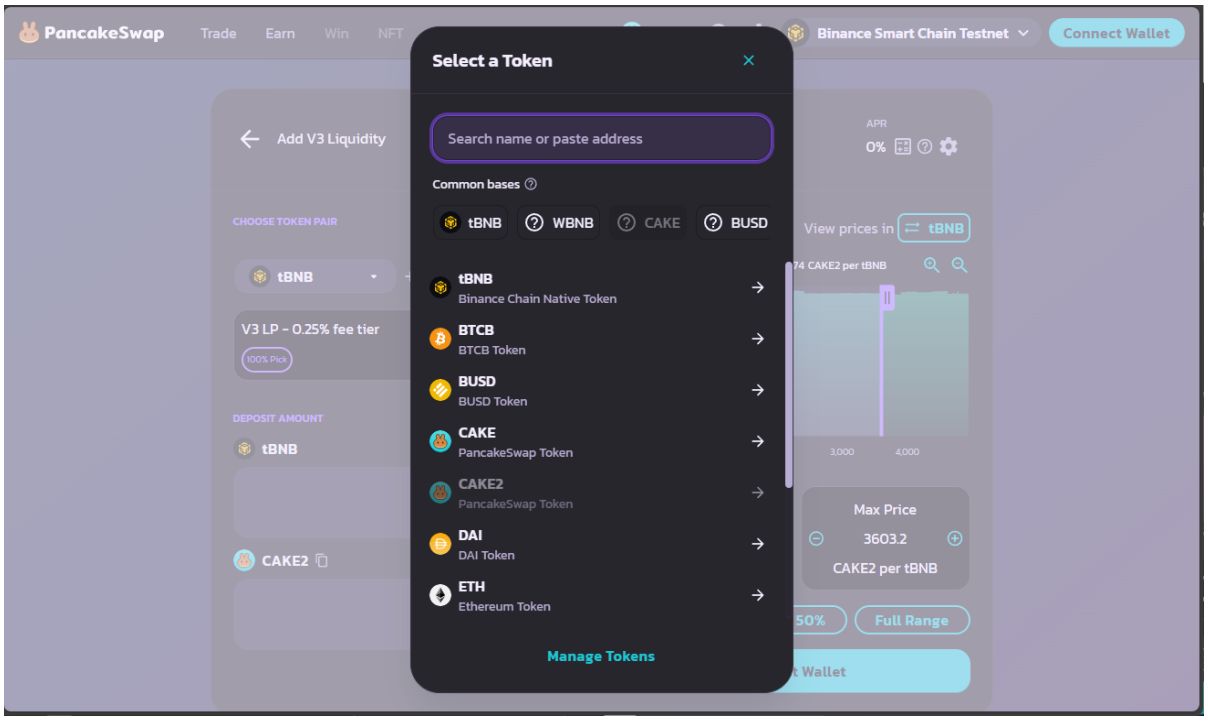


8. This will allow us to add liquidity onto our cryptocurrency. To do this, select **V2** and click on **Add Liquidity**

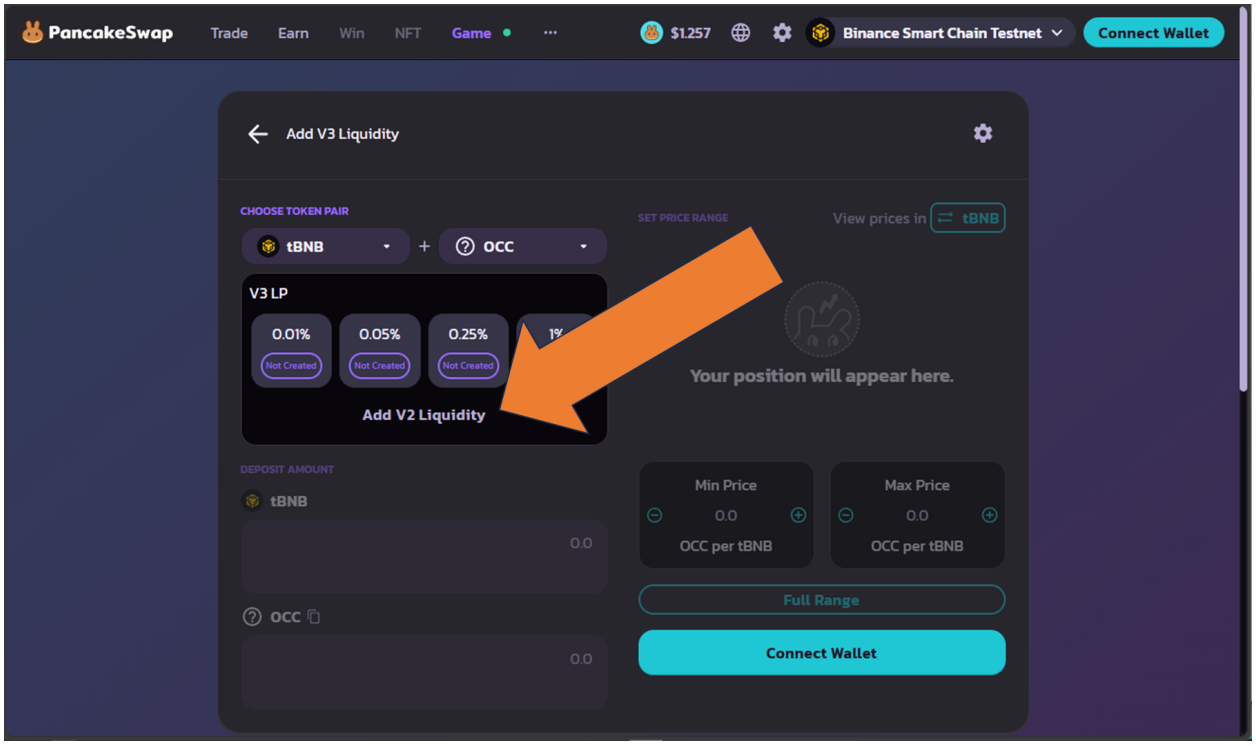


9. Under the **Choose Token Pair** part, choose **tBNB** for the first dropdown and for the second one, paste the address of your smart contract. You can refer to step 13 in the first part on how to find and copy the address of your smart contract.

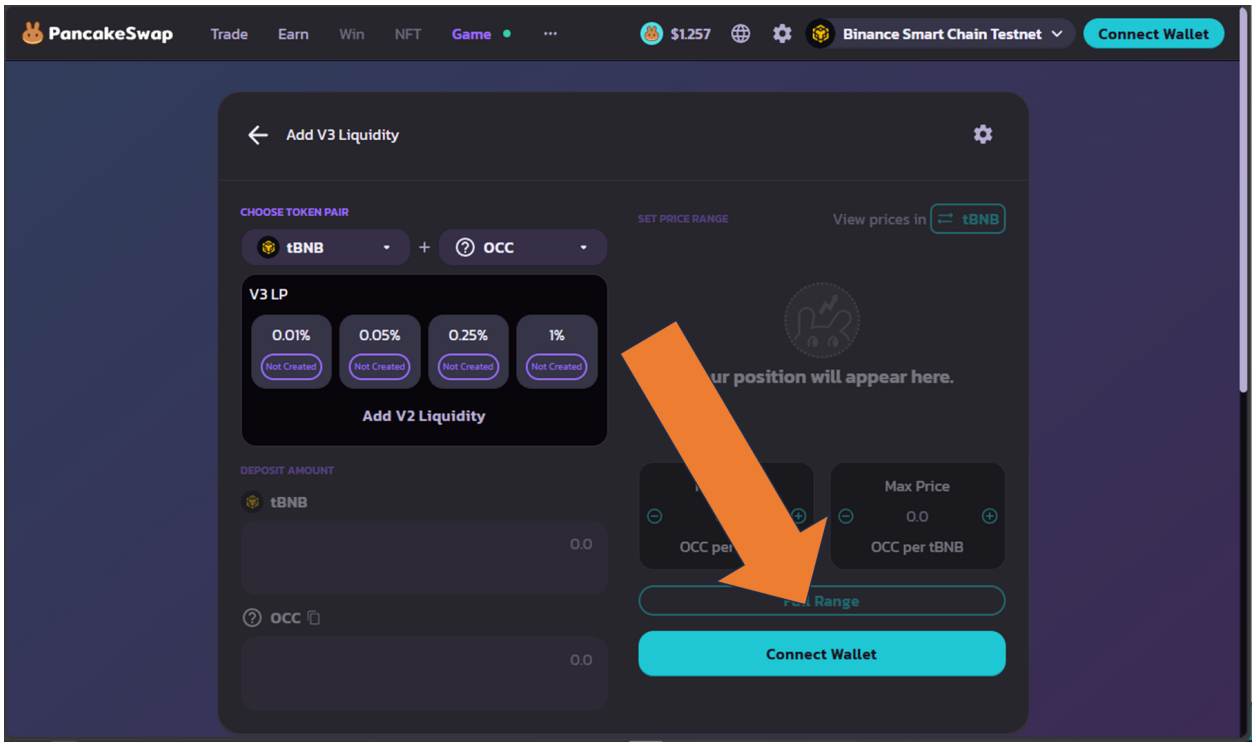




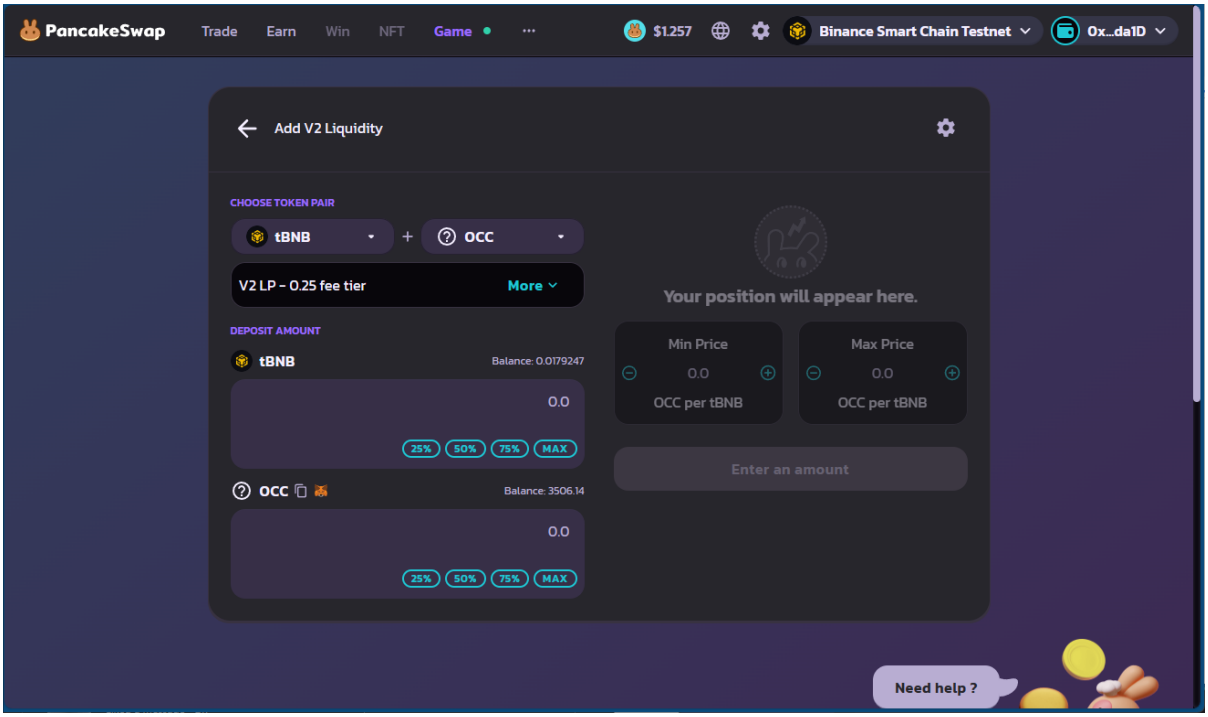
10. Click on **Add V2 Liquidity**



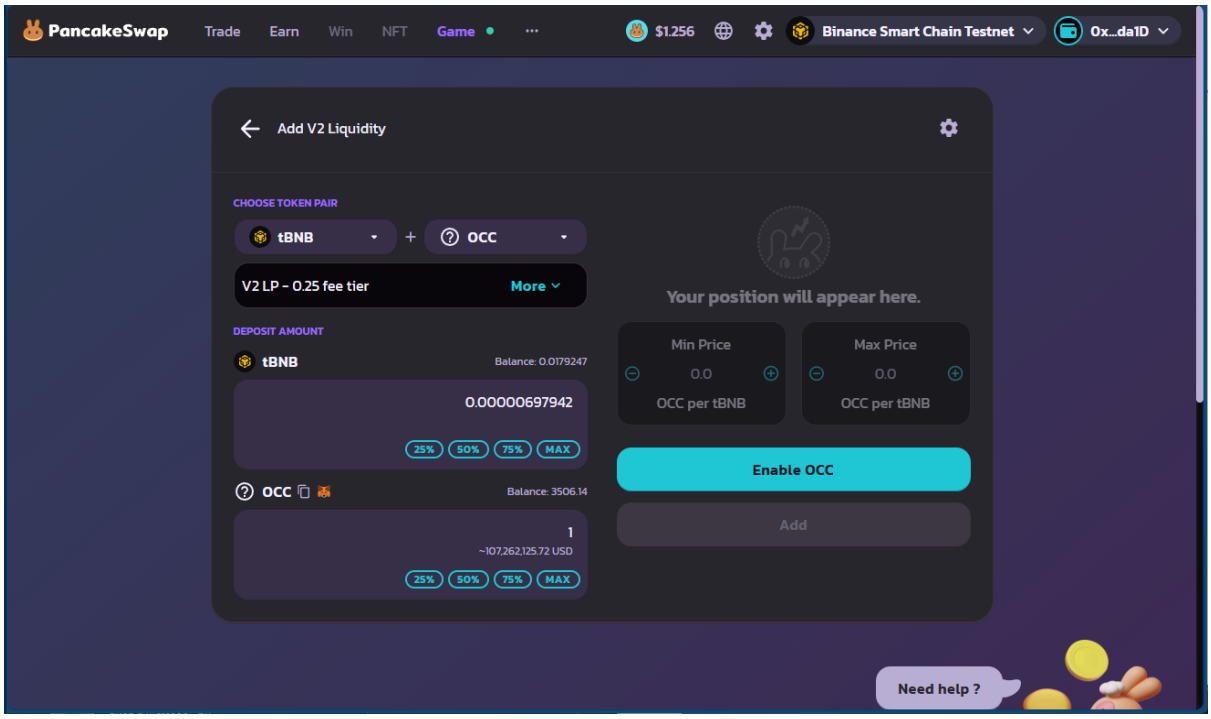
11. Click on the **Connect Wallet** button to connect the site to your MetaMask. Just go through the process in approving your wallet



12. You should now be able to view the balance of your OCC Coin in the site

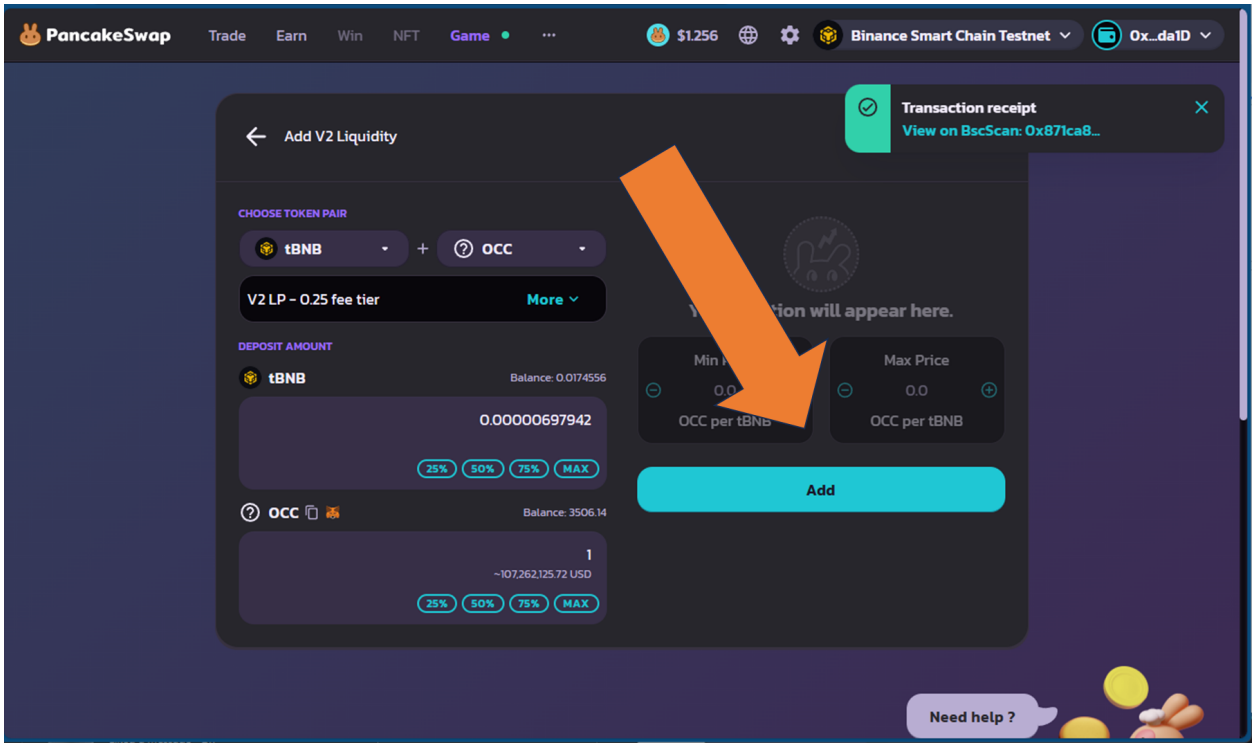


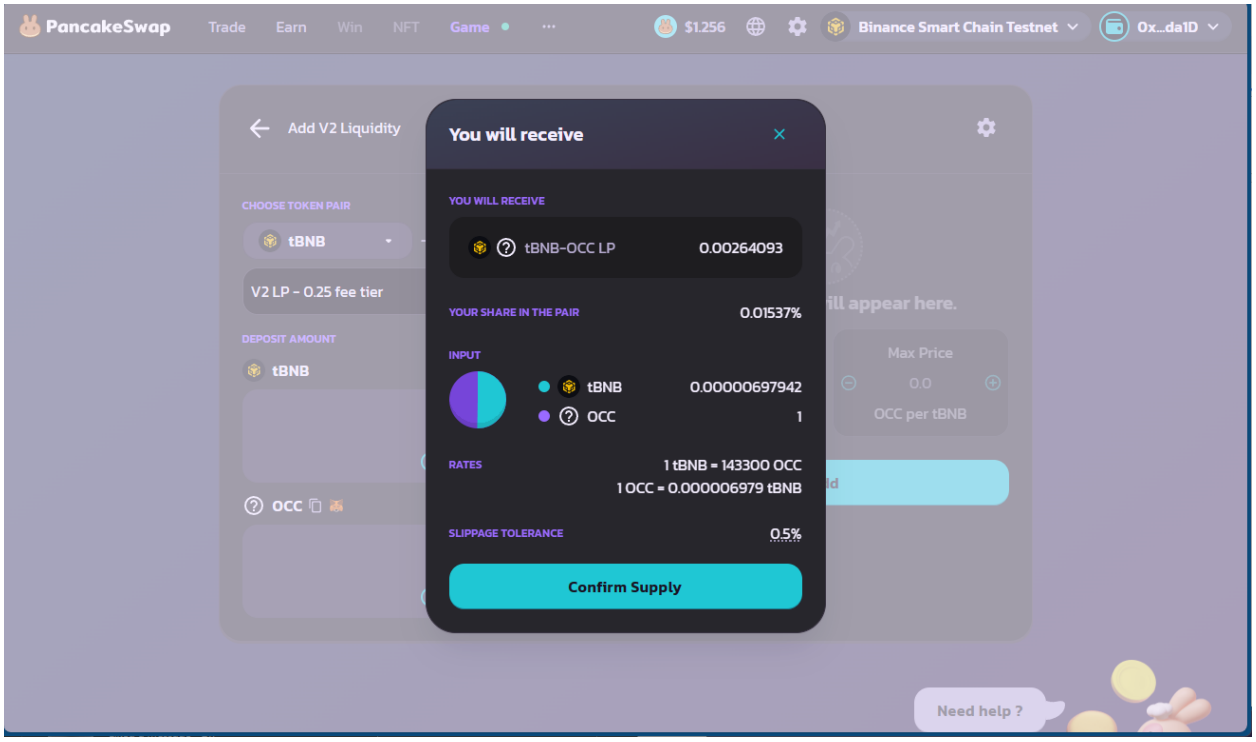
13. Input how much **tBNB** will be backed by your coin
14. Next, input how much **OCC** will be in circulation



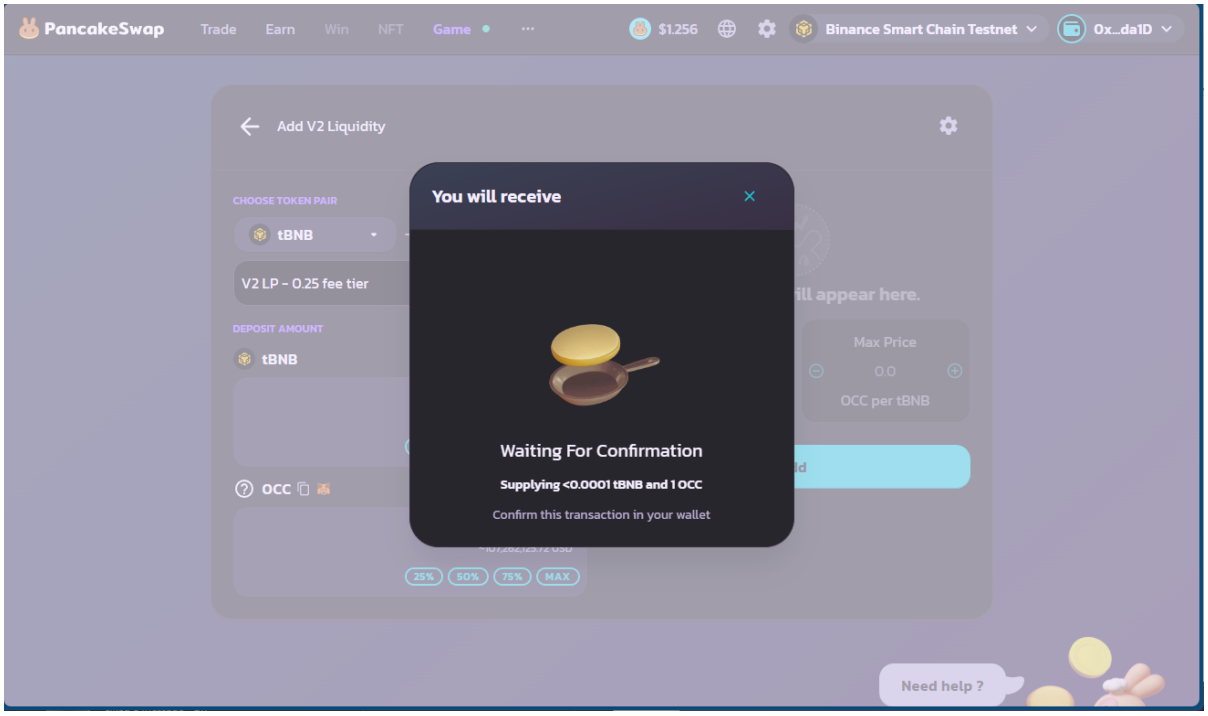
15. Click on **Enable OCC**
16. This will prompt your MetaMask. Just approve the transaction to make it push through.

17. Click on the **Add** button to **Confirm Supply**

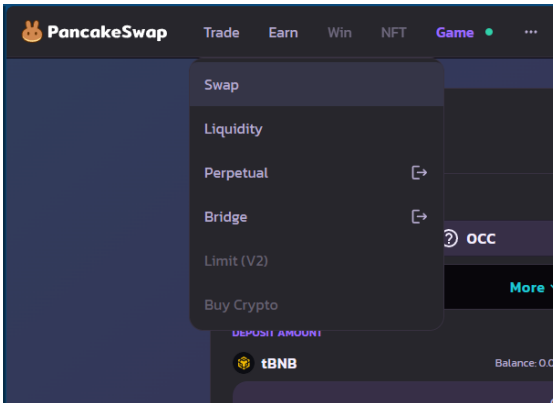




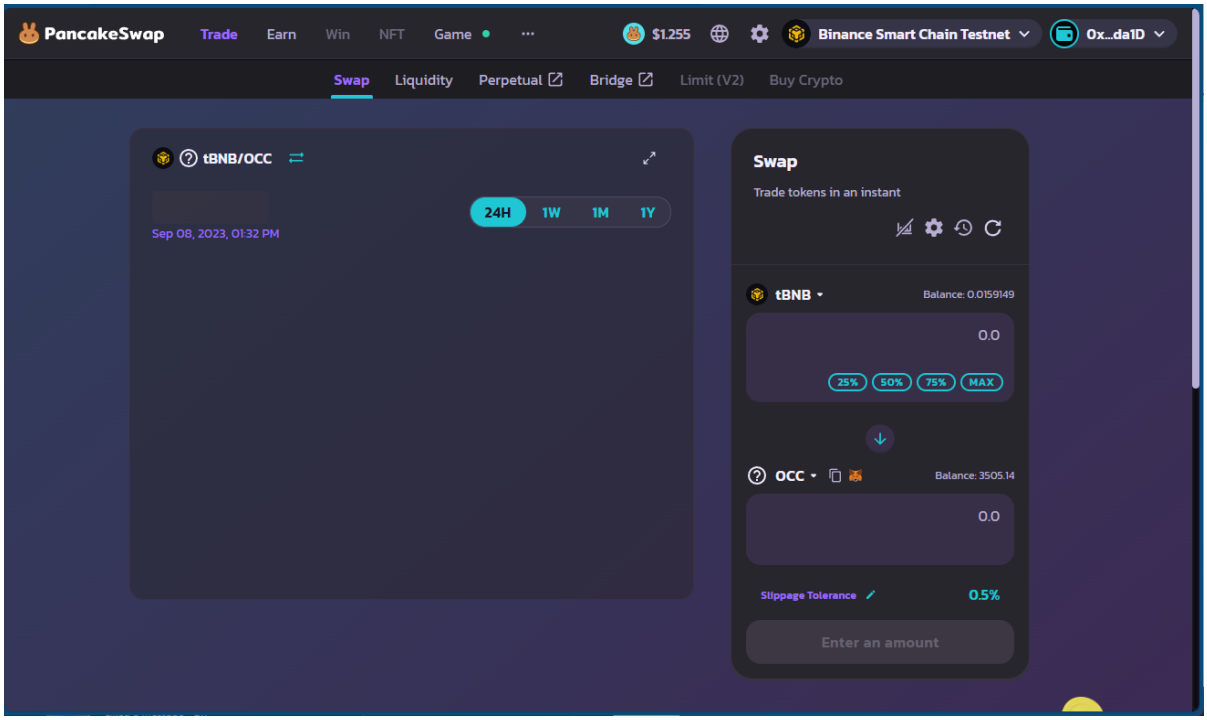
18. This will prompt you again to confirm via MetaMask



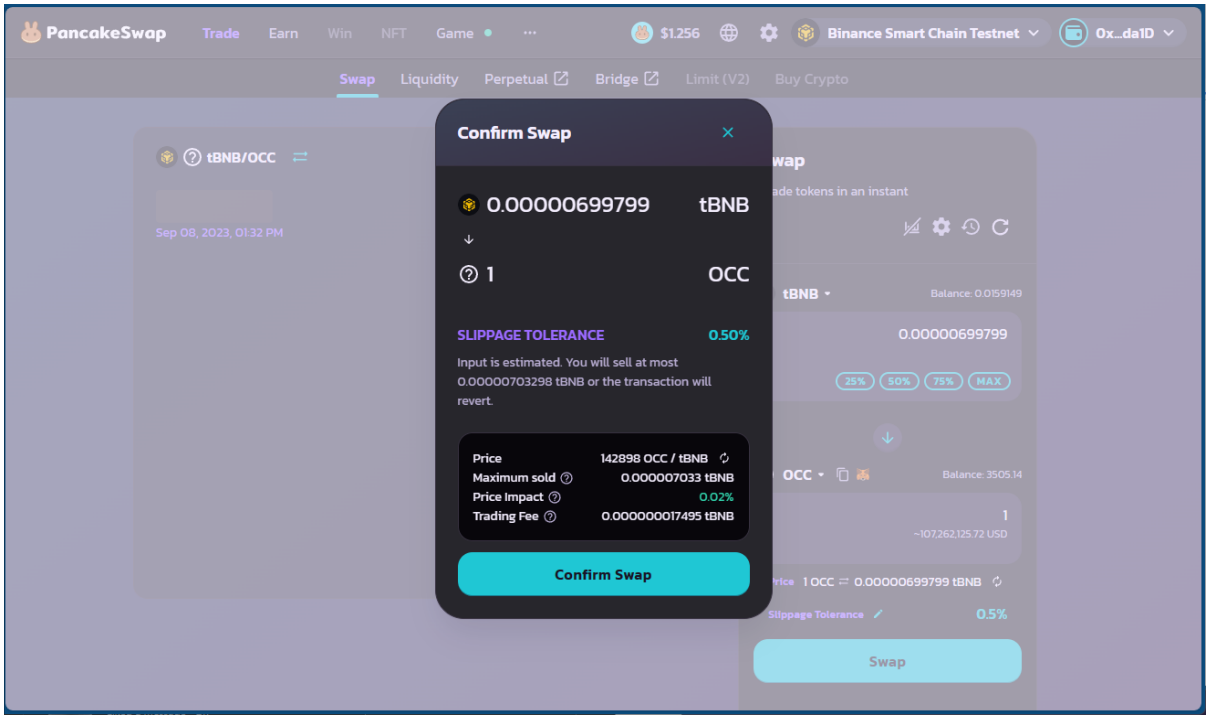
19. Notice that the value of your **OCC token** decreases because there is now that much tokens in circulation
20. To view the exchange rate, go to **Trade > Swap** from the upper navigation bar



21. Set the swap to be **tBNB** and **OCC**. This means that we are converting **tBNB** to **OCC**



22. Clicking on the **Swap** button will remove the **OCC** from circulation. MetaMask will ask you to confirm this.



23. Notice that now, your OCC balance will increase as less tokens are in circulation.
24. Congratulations, you have now mastered the basics of adding liquidity to your tokens!