**HOW TO CREATE A CRYPTOCURRENCY**

**A Guide by Jared Neil Hortaleza**

**Pre-requisites:**

**-** [**OCC - Metamask Installation and Mumbai TestNet Guide - Google Docs**](https://docs.google.com/document/d/1moUXqNjSUhvJP-OjZfF-rgqJ4rffI4l7sVeihj5JkYA/edit)

> *What is a Cryptocurrency?*

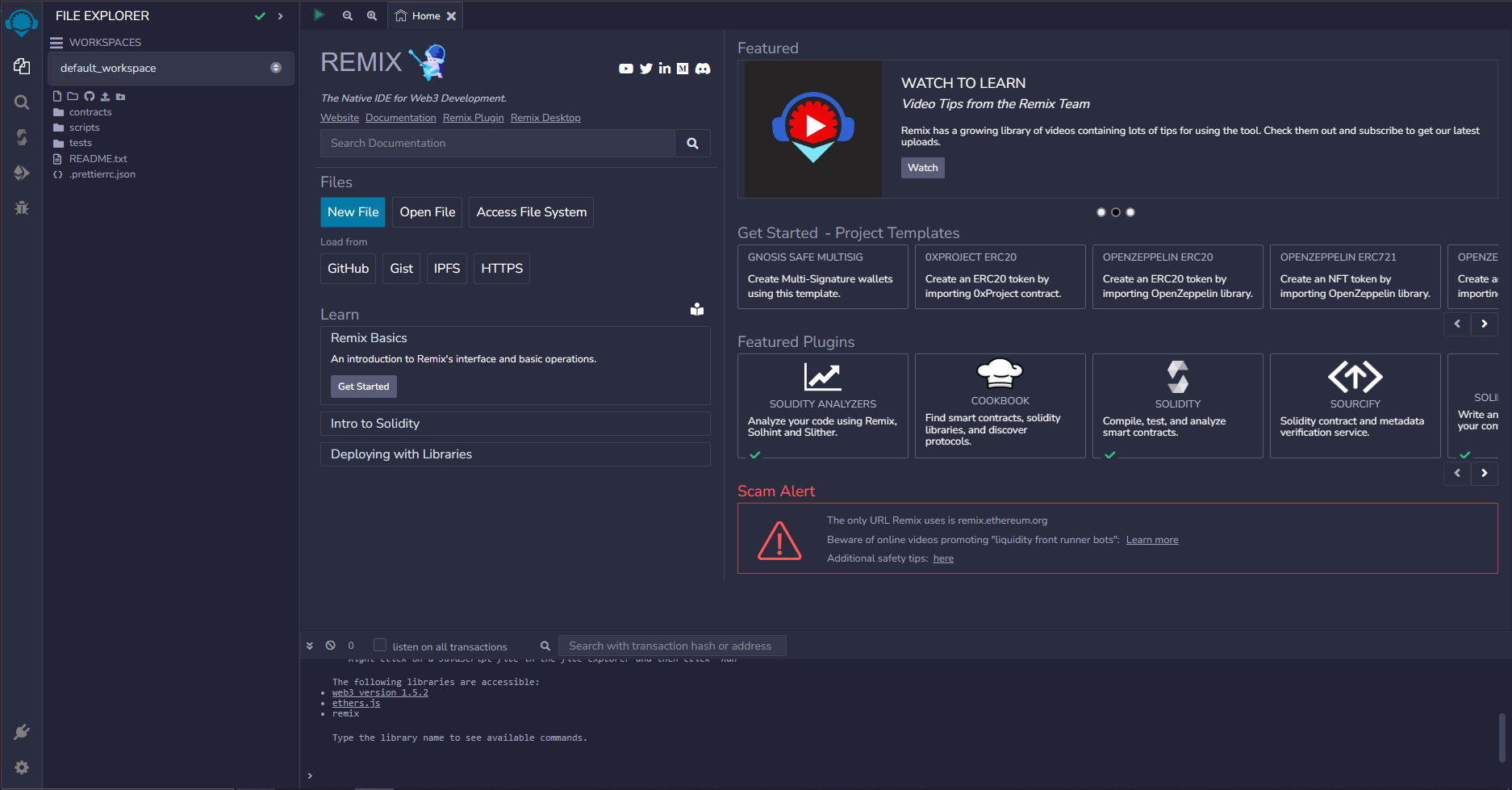
**A cryptocurrency, crypto-currency, or crypto** is a **digital currency** designed to work as a medium of exchange through a computer network that is **not reliant on any central authority**, such as a government or bank, to uphold or maintain it.

*> What is the Remix IDE?*

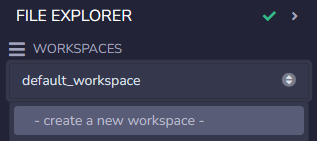
**Remix**, more commonly known as Remix IDE, is an **open-source Ethereum IDE** you can use to write, compile and debug **Solidity** code.

**A.) Make a cryptocurrency through Remix IDE.**

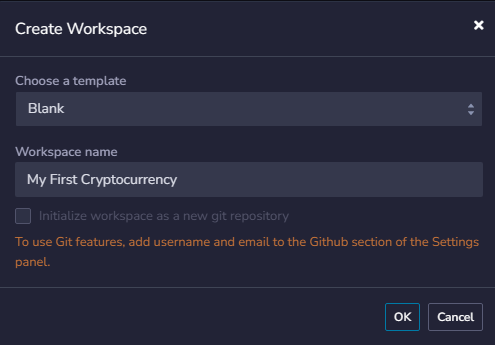
1.) Go to <https://remix.ethereum.org/>, and you will be redirected to this window.



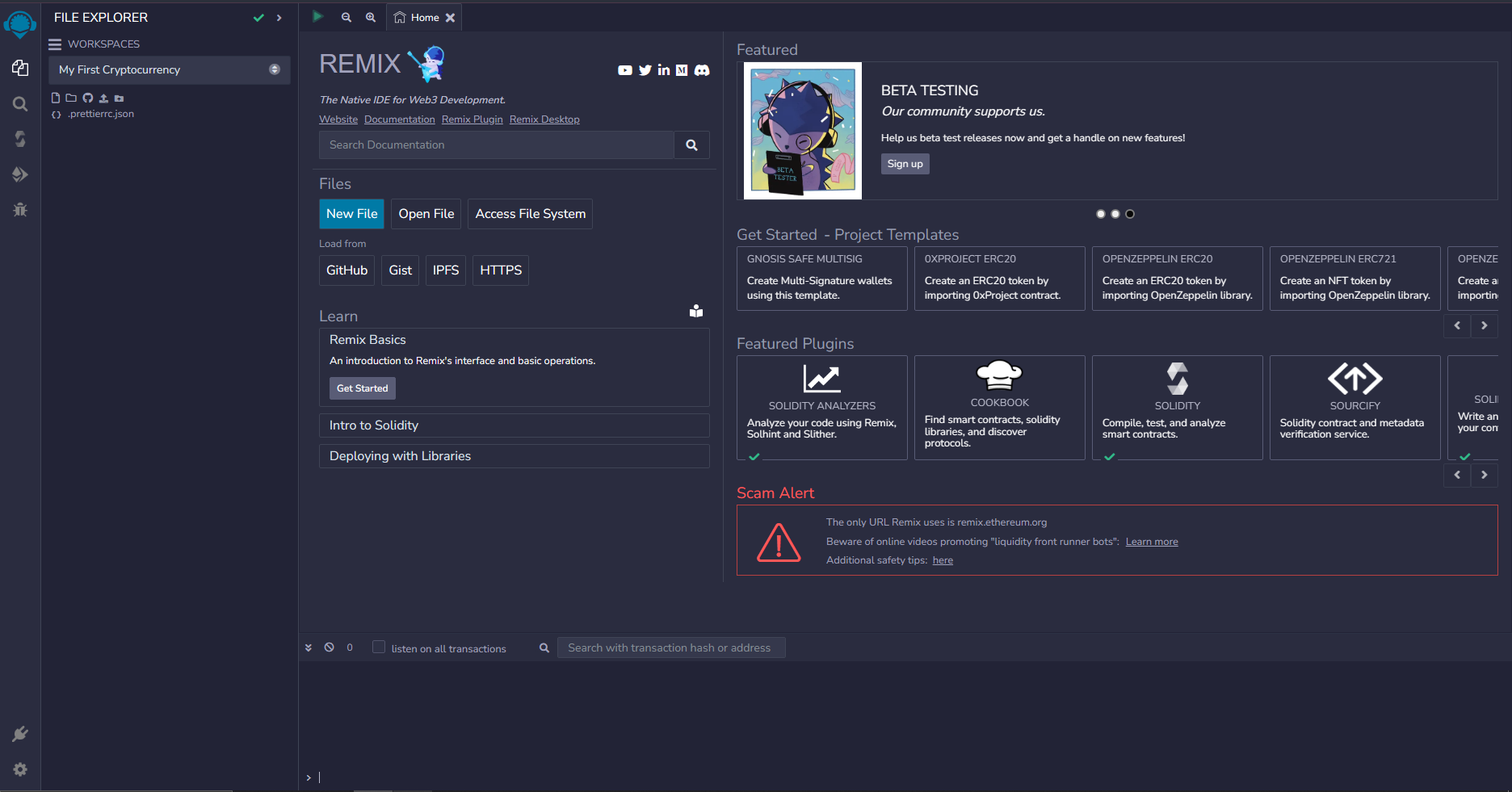
2.) Click on the **Workspaces** dropdown, and create a new workspace.



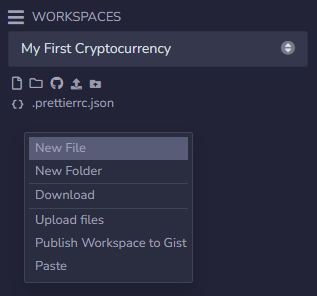
3.) Choose the **Blank** template and name your workspace **“My First Cryptocurrency”**.



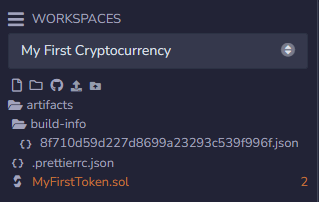
4.) Click on **OK**. You will be redirected to this window.



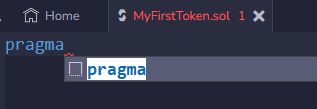
5.) Right click on the file explorer, and click on **New** **File**.



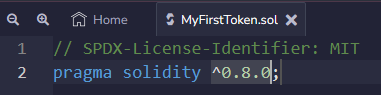
6.) Name the file **MyFirstToken.sol**. **Remix** will recognize this as a **Solidity file**, so you can make a smart contract.



7.) Type in **pragma** in the file, and **Click** on the autocomplete option or press **Tab**.



8.) It will then create two lines of code. Replace **version** with **^0.8.0** so we can specify that this works with Solidity from Version 0.8.0 and above.



*The first line indicates the license of the code. We use MIT for open-source and personal use.*

9.) Next, insert these lines of code after the first two:

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

These lines of code import functions necessary for creating your first token.

* **Ethereum Request for Comment 20** (ERC-20) is a technical standard used for **smart contracts** on the Ethereum Blockchain. ERC-20 guides the creation of new tokens on the Ethereum blockchain.
* **Ownable** is a contract module which provides a **basic access control mechanism**, where there is an account (an owner) that can be **granted exclusive access** to specific functions for a contract owner.



*Importing the ERC-20 and Ownable libraries from OpenZeppelin.*

10.) **Create a contract** by typing in:

* contract <insert your first token name> is ERC-20, Ownable {}

This line of code **declares a contract** which **uses the functions of ERC-20 and Ownable**, the libraries you imported previously.

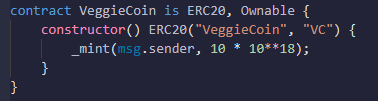


*Creating a contract named VeggieCoin which inherits ERC20 and Ownable.*

11.) Create a constructor to **mint (creating tokens)** the **initial supply** of the token.

* constructor() ERC20(“<your token name>”,”<your token symbol>”) {
* \_mint(msg.sender, 10 \* 10\*\*18);
* }

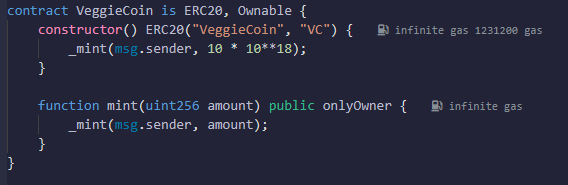
These lines of code indicate the **minting of 10^18 wei (10 of your token)**, and this will **run upon creation of the contract**. This acts as the **initial supply** of the Contract Owner.



12.) Add a minting function for the contract owner to add tokens to their account.

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

This function indicates that **only the owner of this contract** (onlyOwner) can enter an amount to mint (add tokens) to their own account.



13.) Create a burn function, for the user to burn (use) their tokens.

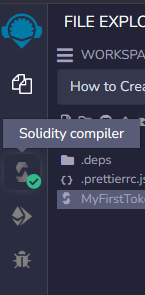
* function burn(uint256 amount) public onlyOwner {
* \_burn(msg.sender, amount);
* }

This function indicates that **only the owner of this contract** (onlyOwner) can enter an amount to burn (use tokens) from their own account.

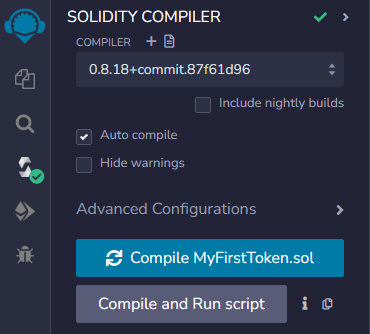


**B.) Deploy the smart contract.**

1.) Click on the **Solidity Compiler**.

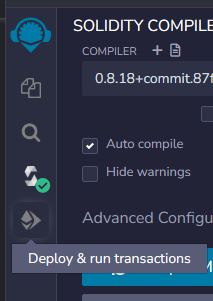


2.) Click on **“Compile MyFirstToken.sol”** to check for errors and get the code ready for deployment.

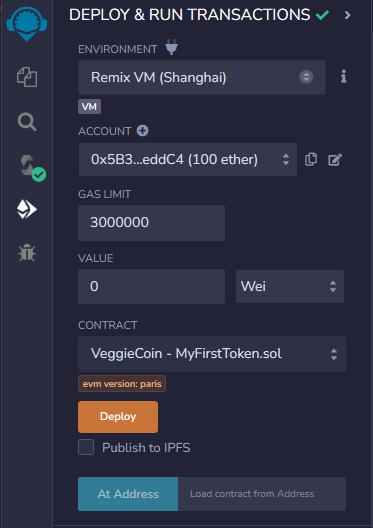


*You can also check the Auto compile option to compile every time the code changes.*

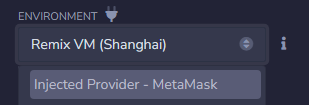
3.) Click on the button below the Compiler to go to Deploy & run transactions.



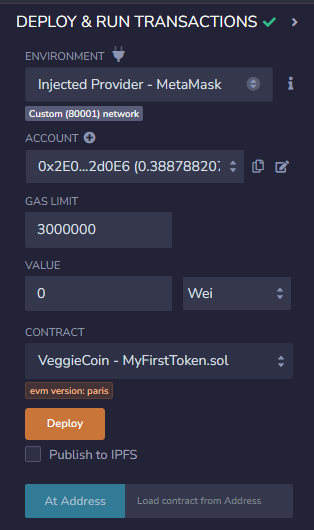
4.) You will be redirected to this window. Click on the **ENVIRONMENT** dropdown.



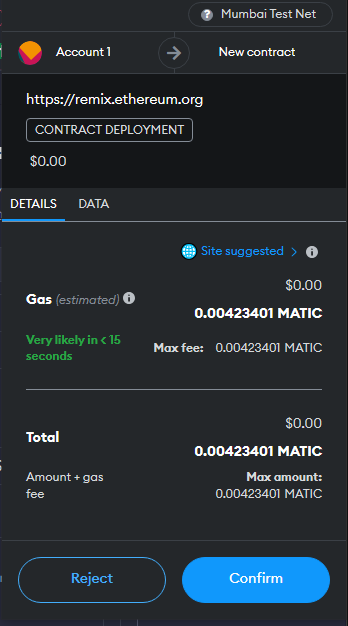
5.) Click on the **“Injected Provider - MetaMask”**. You will be sent a **connection request** via pop-up. **Approve** the request.



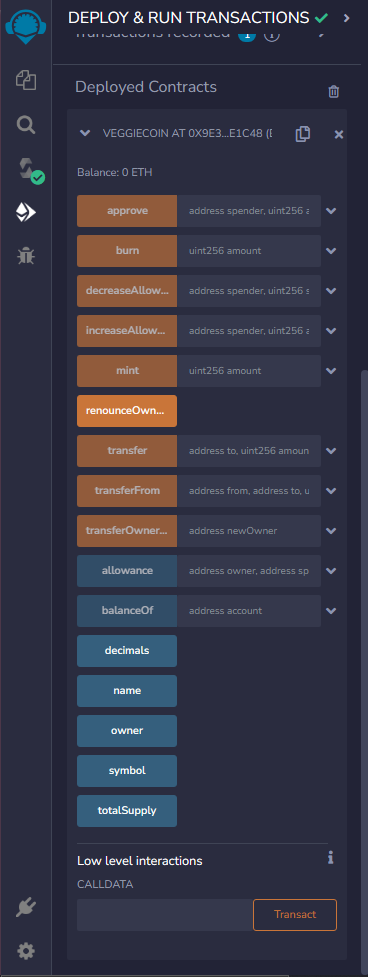
6.) Your window will now look like the image below with the account being your **MetaMask wallet address**. Click on **Deploy**.



7.) A confirmation window will pop up. **Confirm** the transaction.

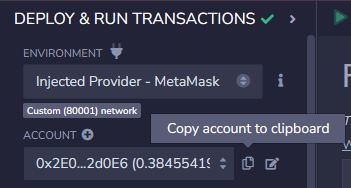


8.) You have now **deployed a contract to your own account**! Click on the **chevron icon** on the left of the contract to expand it and see the contract’s functions.

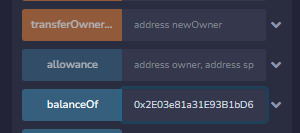


**C.) Send and burn your token.**

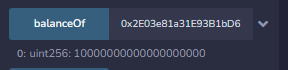
1.) Click on the Copy Icon (looks like two files) to copy your account in the Deploy & Run Transactions window.



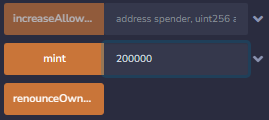
2.) Check how much token the account has currently. Scroll down to the deployed contract, and paste your account in the text box beside **balanceOf**.



3.) Since the constructor specified that contract owners start off with **10^18 wei** (10 tokens), the balance of your account will start off with 1,000,000,000,000,000,000.

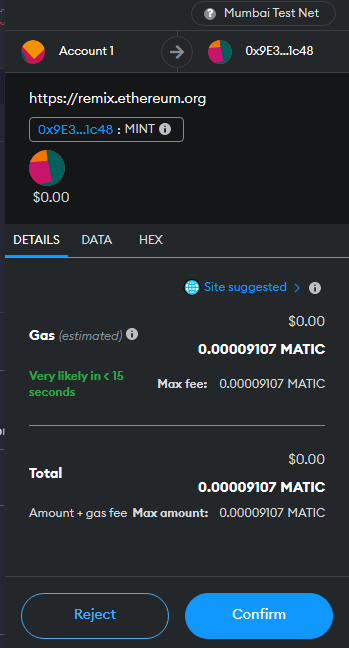


4.) To add to this balance, scroll up to the **mint** function in that same window, and specify an amount of wei you want to add to the account. Then, click on **mint**.

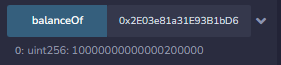


*Adding 200000 wei (0.0000000000002 of the token) to the account.*

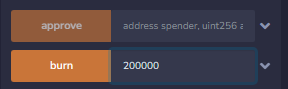
5.) A MetaMask prompt will appear. **Confirm** the transaction.



6.) Check the **balanceOf** of the account. It should now include the amount (in wei) that you specified.

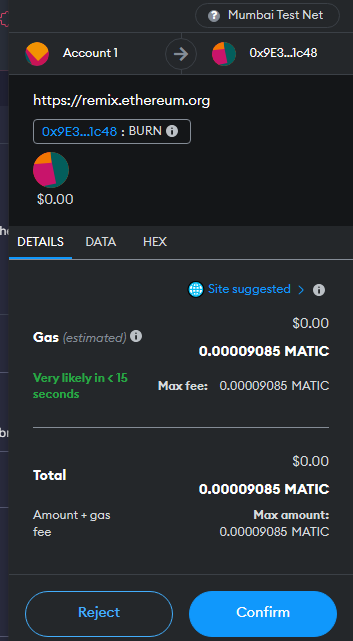


7.) To burn (use) your tokens, go to the **burn** function and specify how much (in wei) will be burned. Then, click on **burn**.

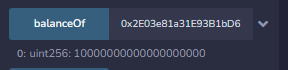


*Burning 200000 wei (0.0000000000002 of the token) from the account.*

8.) A MetaMask prompt will appear. **Confirm** the transaction.



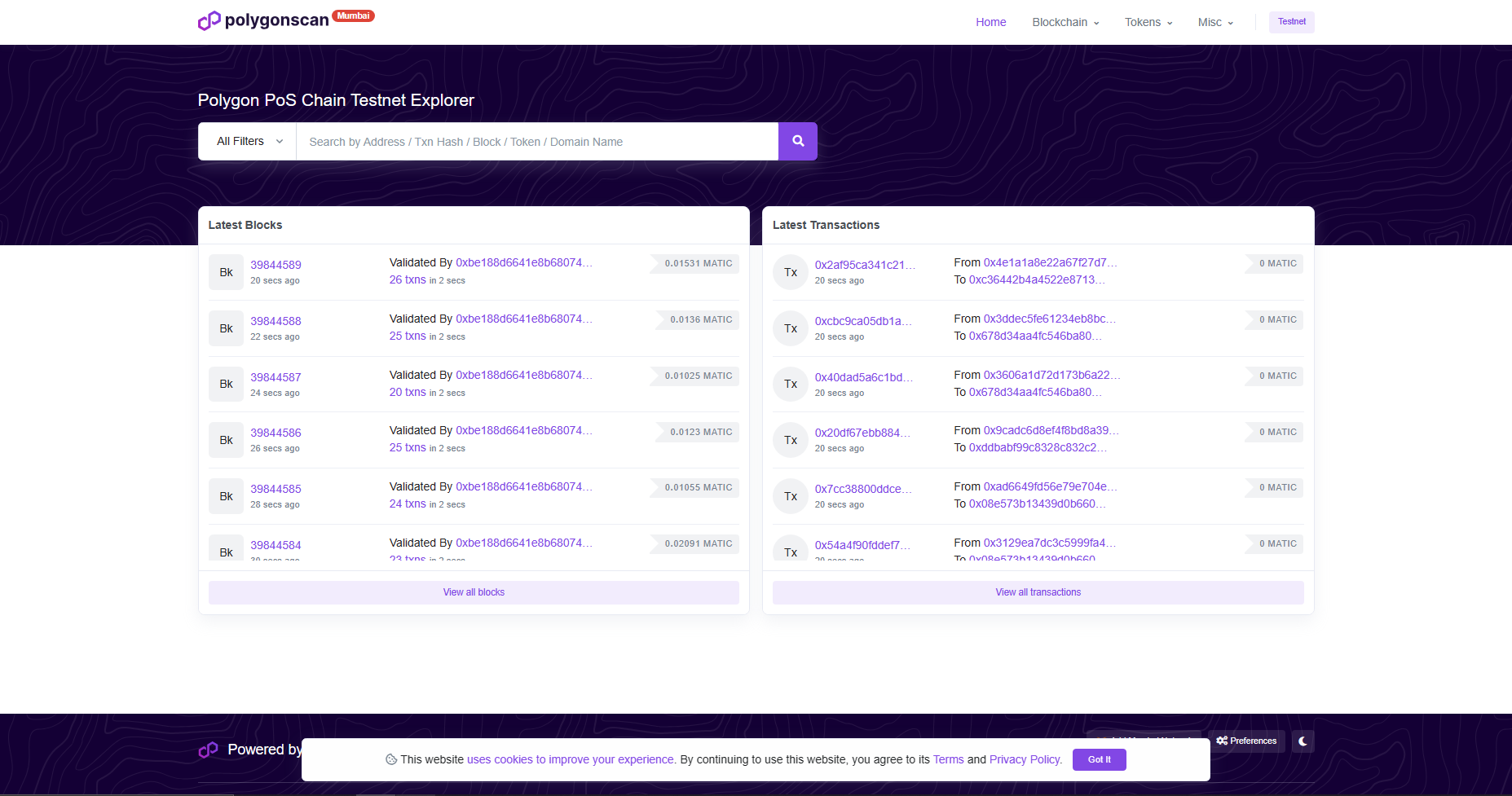
9.) Check the **balanceOf** of the account. It should now remove the amount (in wei) that you specified.



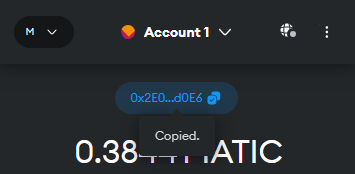
10.) You have successfully **minted** and **burned** your token! You can also check these transactions in Polygon Scan.

**D.) Check transactions in Polygon Scan.**

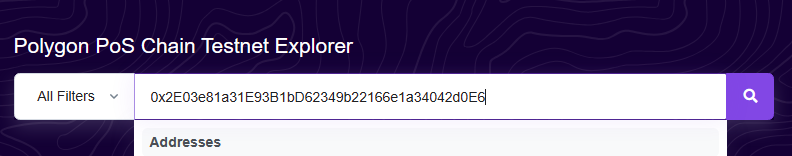
1.) Go to <https://mumbai.polygonscan.com/> to check your token transactions.



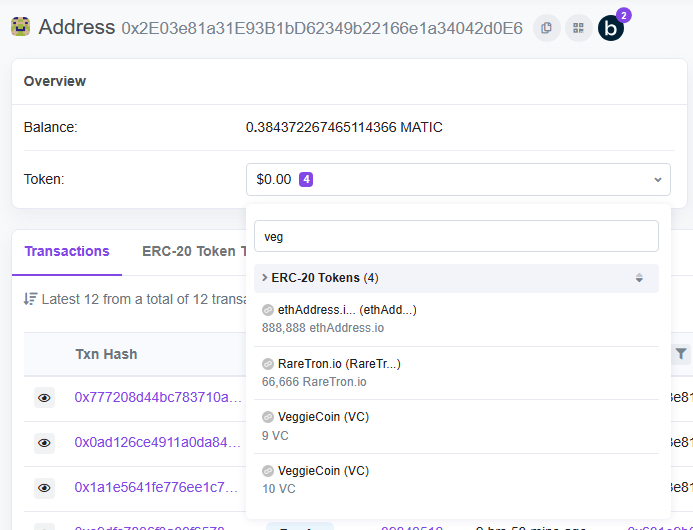
2.) Go to your MetaMask and click the **blue pill icon** to copy your wallet address.



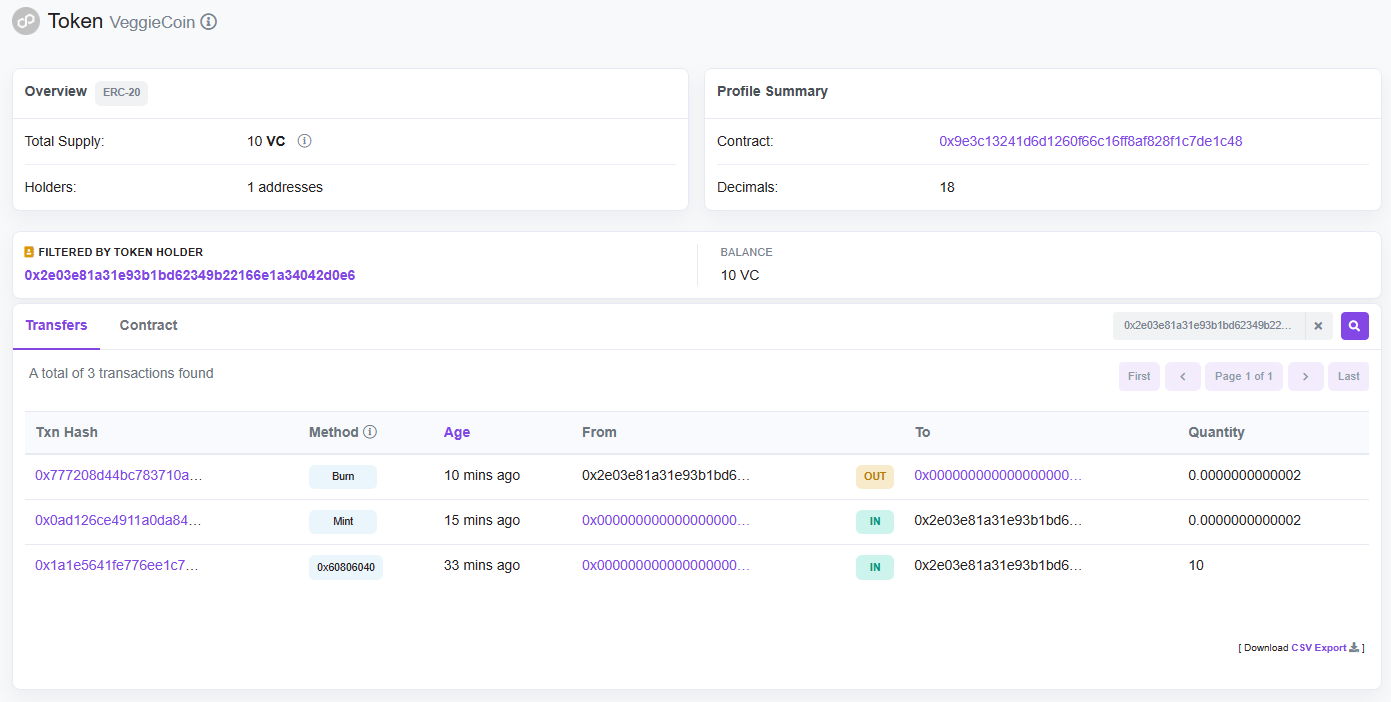
3.) Paste your address in the search bar. Click the search icon or press Enter on your keyboard.



4.) Click the **token** dropdown and select your token.



5.) You will now see the transactions you’ve made along with the **contract, total token supply, and holders**. Congratulations, your account is now a **test token holder**!

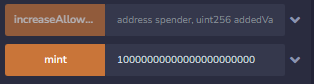


**E.) Add liquidity to your test token and importing the test token in MetaMask.**

> *What is liquidity?*

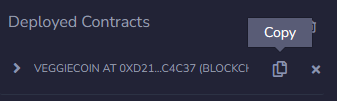
**Liquidity** refers to the **efficiency or ease** with which an asset or security can be converted into ready cash without affecting its market price. In crypto, it largely plays around exchange rates between different cryptocurrencies.

1.) Mint **10000** of your test tokens to your wallet.

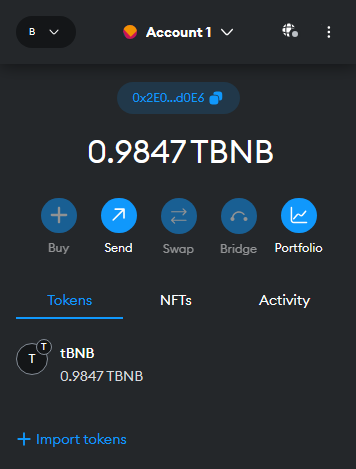


*Minting 10000000000000000000000 wei (10000 of the token) to the account.*

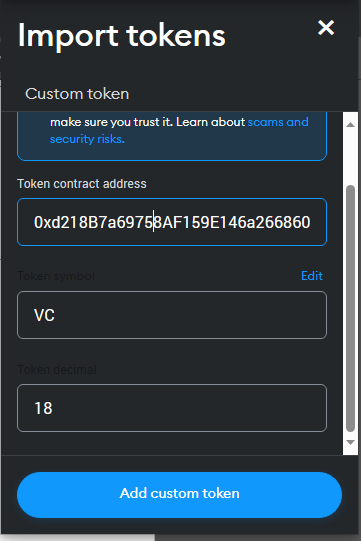
2.) Copy your deployed contract address.



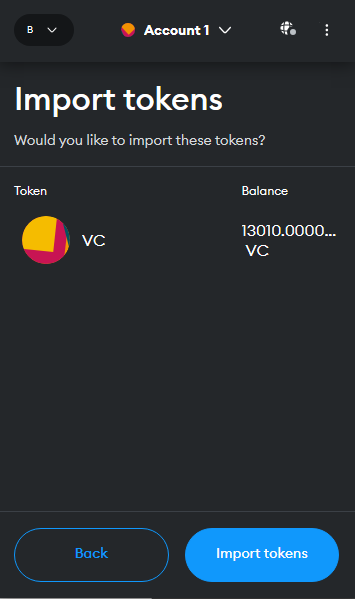
3.) Go to your MetaMask and click on **Import tokens**.



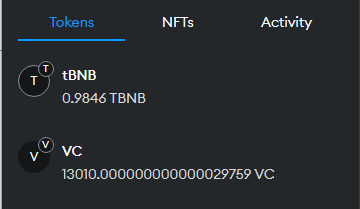
4.) Paste the copied address in the **Token contract address**. The rest of the fields will be auto-generated. Then, click on **Add custom token.**



5.) Go to your MetaMask and click on **Import tokens**.



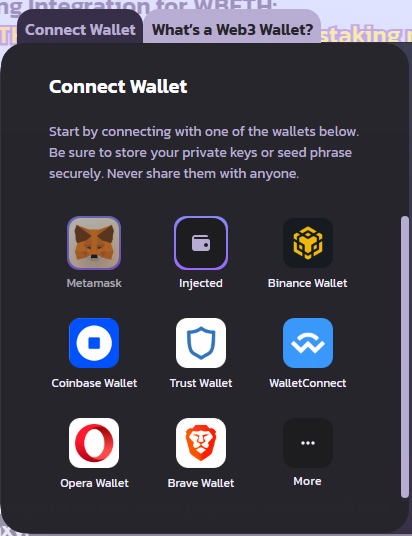
6.) You will now see your **test token** in your **MetaMask wallet**.



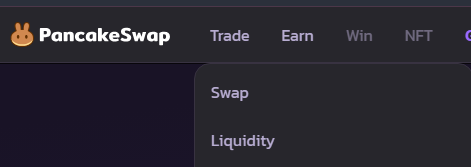
7.) Go to <https://pancakeswap.finance/?chain=bscTestnet> and click on **Connect Wallet**.



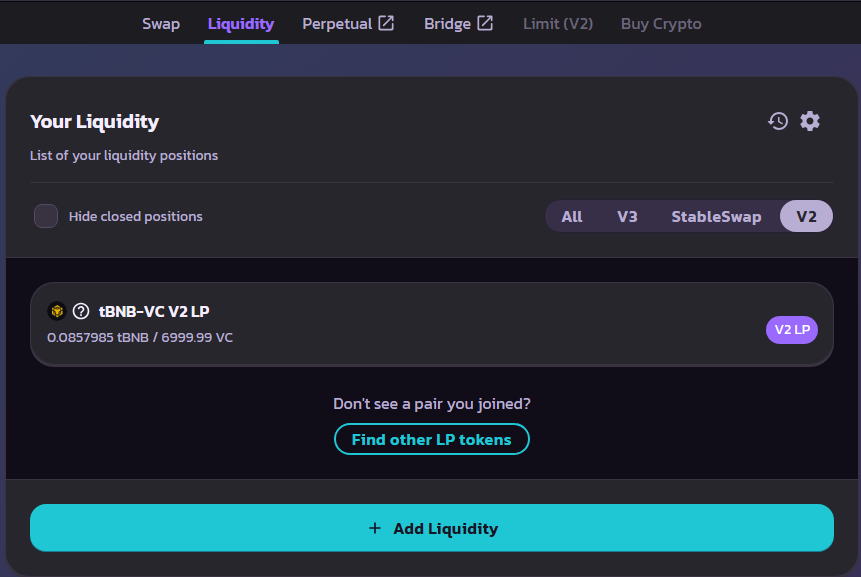
8.) Click on Metamask and **Approve** the connection to **PancakeSwap**.



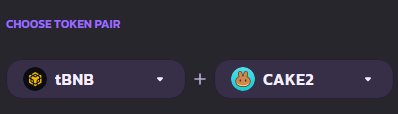
9.) Once your wallet has been connected, go to **Trade > Liquidity**



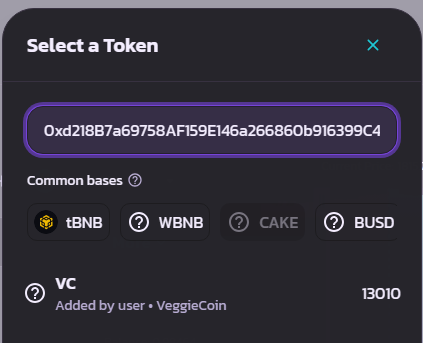
10.) Select **V2** then Click on **Add Liquidity**.



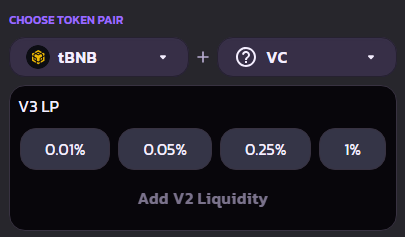
10.) Click on the **Token Dropdown** on the **right**.



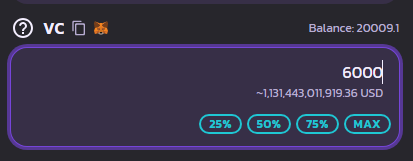
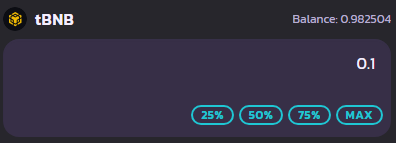
11.) Go back to Remix IDE and **copy your contract address again**, then **paste it** in the search bar. Then, select your token.



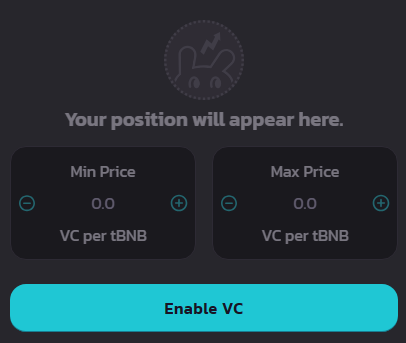
12.) Click on **Add V2 Liquidity**.



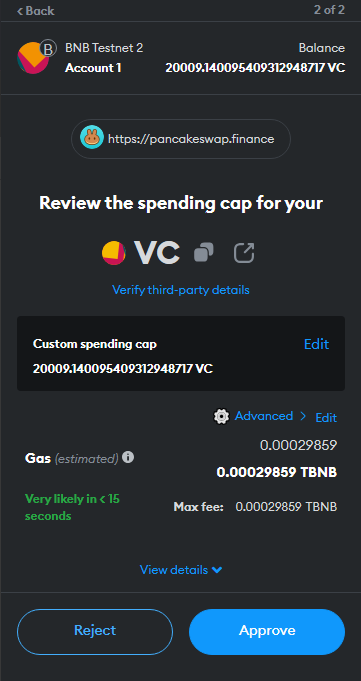
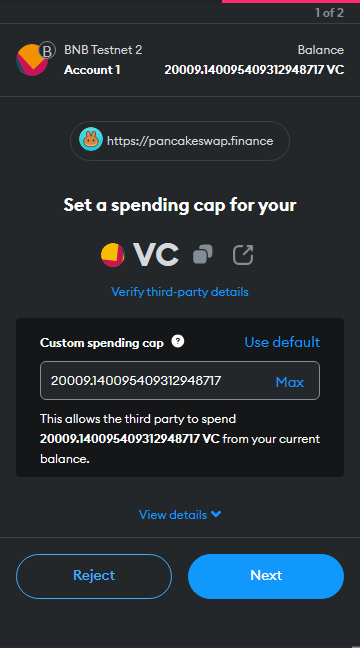
13.) Enter an amount (i.e. 0.1) in **tBNB** and an amount (i.e. 6000) in your **test token**.



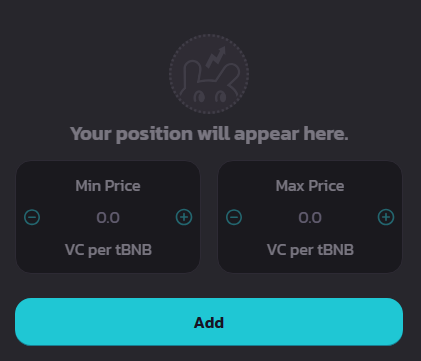
14.) Click on **Enable VC**.



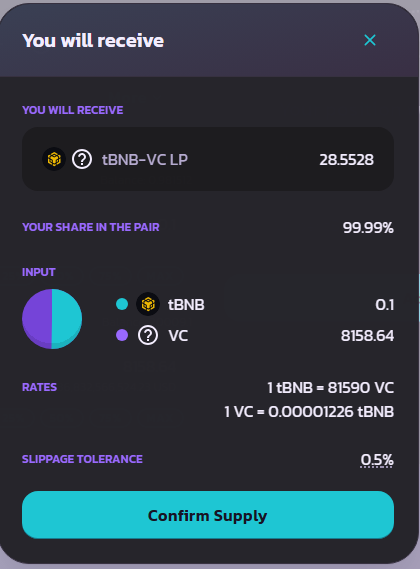
15.) MetaMask prompt will appear. Click on **Max** and click **Next**. Then, click **Approve**.

****

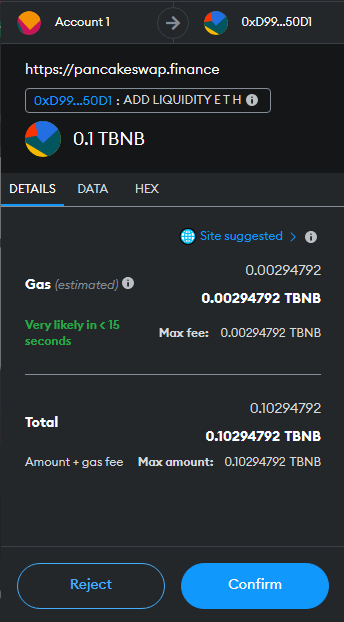
16.) Click on **Add**.



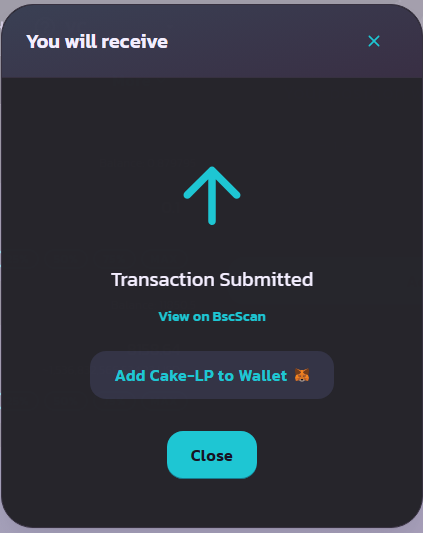
17.) Click on **Confirm Supply**.



18.) **Confirm** the transaction on the MetaMask pop-up.

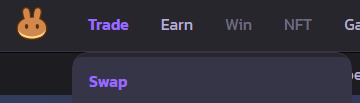


19.) **You have successfully added liquidity!** Click on **Close**.



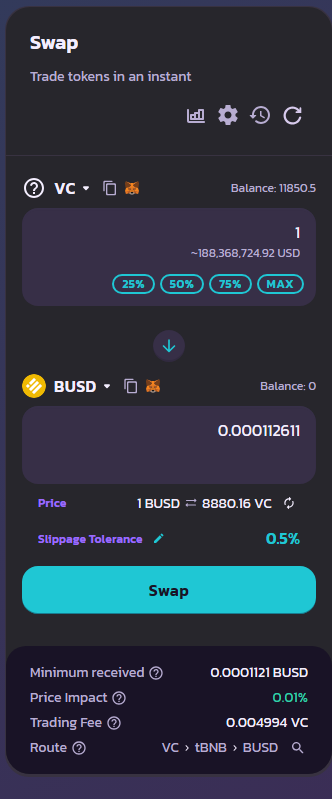
**F.) Check how much your test tokens are worth.**

1.) Go to **Trade > Swap**.



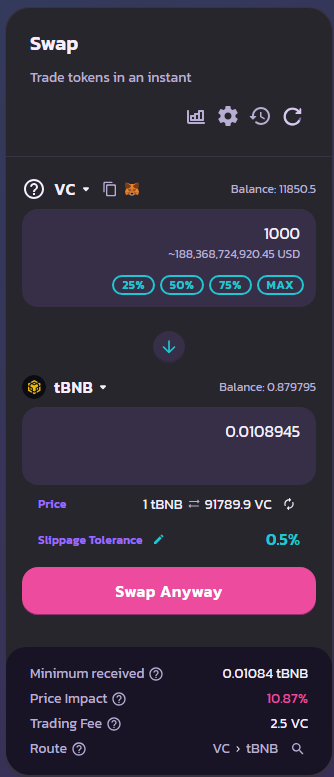
2.) Check how much your token is worth in **BUSD**. Make the first token **your test token** and the second token **BUSD**. Then, take note of **how much 1 of your test token** is worth.



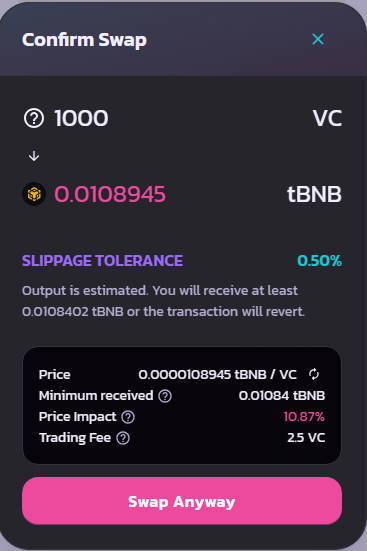


3.) Swap some of your token for some **tBNB**. Change the second token to **tBNB** and **enter an amount** that you’d want to swap. Then, click on **Swap Anyway**.

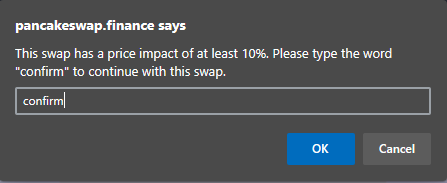




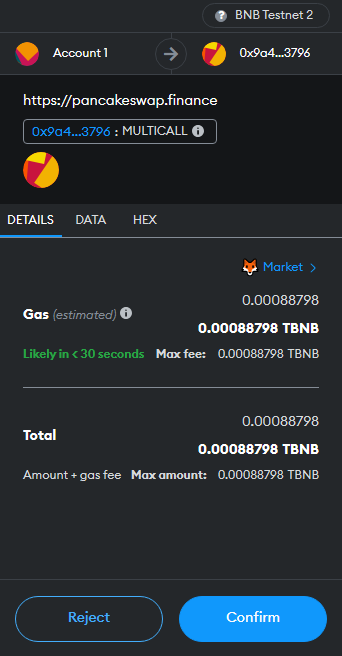
4.) Click on **Swap Anyway** again.



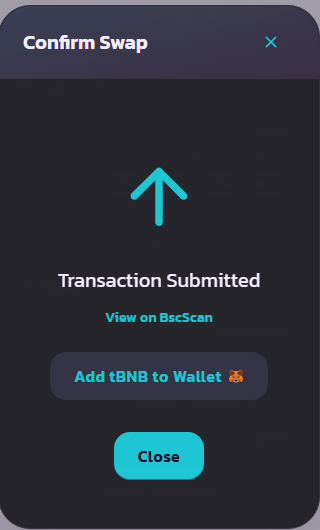
5.) Type **confirm**, then click **OK**.



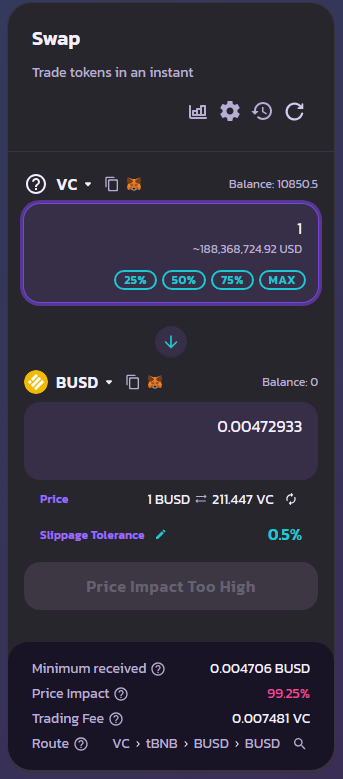
6.) **Confirm** the transaction on the MetaMask pop-up.



7.) **You traded your token for some tBNB!** Click on **Close**.



8.) Compare the price of one of your tokens in BUSD now. Make the first token **your test token** and the second token **BUSD**. Then, take note of **how much 1 of your test token** is worth.



9.) **You were able to increase the price of your token!** Now, you have a test token that is **currently in circulation** in the **Binance Smart Chain (BSC) Testnet**. **Congratulations!**