**Web 3.0 Jumping Game**

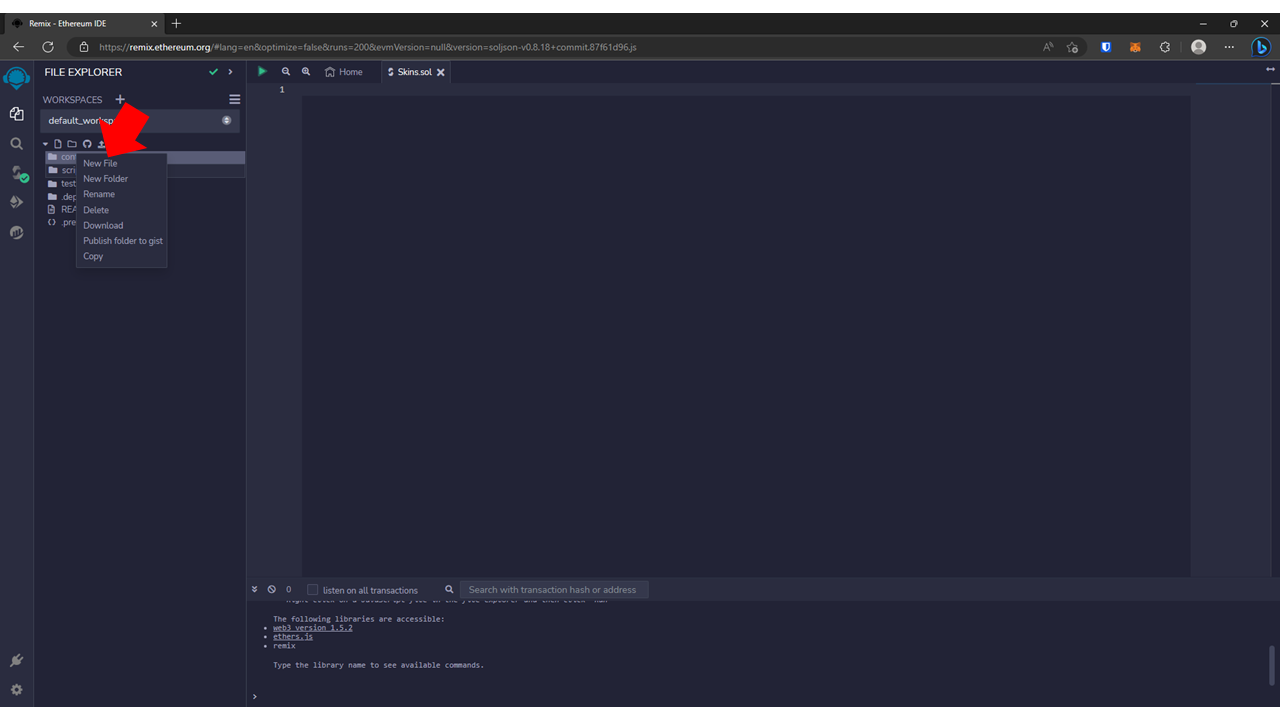
Welcome to our live coding session by 247codecamp! Today, we will be exploring Web 3.0 game development by building a jumping game that uses blockchain technology. Our game will feature a player character navigating through obstacles powered by Ethereum smart contracts and IPFS decentralized storage. Join us as we dive into the exciting world of decentralized game development!

Here's a quick overview of the tools we'll be using in this live coding session:

* MetaMask: a digital wallet for managing cryptocurrency and interacting with Ethereum blockchain applications.
* Remix IDE: an online integrated development environment for building and testing smart contracts on the Ethereum blockchain.
* CodeSandbox: a web-based code editor that allows you to develop, share, and collaborate on projects in real-time.
* OpenZeppelin: a library of reusable smart contracts for building decentralized applications on the Ethereum blockchain.
* GitHub: a platform for hosting and collaborating on code projects with features like version control, issue tracking, and pull requests.
* OpenSea: a marketplace for buying, selling, and discovering NFTs.
* Mumbai Polygonscan: a blockchain explorer for the Mumbai testnet, which is used for testing Ethereum applications.

Let’s get started!

1. Go to the Remix IDE (https://remix.ethereum.org) and create a new file under the "contracts" folder called Skins.sol.



1. Paste this code into the Skins.sol file.

// SPDX-License-Identifier: UNLICENSED

pragma solidity ^0.8.9;

import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";

import "@openzeppelin/contracts/utils/Counters.sol";

contract Skins is ERC721URIStorage {

using Counters for Counters.Counter;

Counters.Counter private \_tokenIds;

constructor() ERC721("Skins", "SKIN") {}

function mint(string memory tokenURI) public returns (uint256) {

\_tokenIds.increment();

uint256 newItemId = \_tokenIds.current();

\_mint(msg.sender, newItemId);

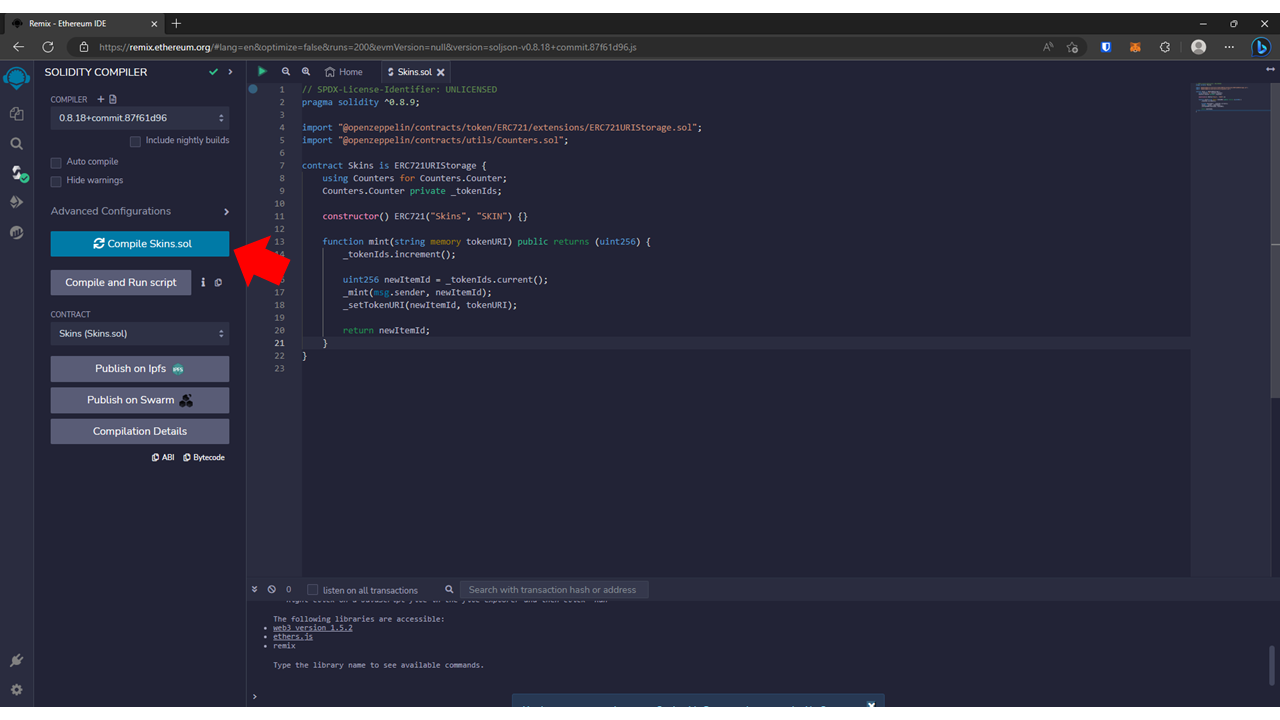
\_setTokenURI(newItemId, tokenURI);

return newItemId;

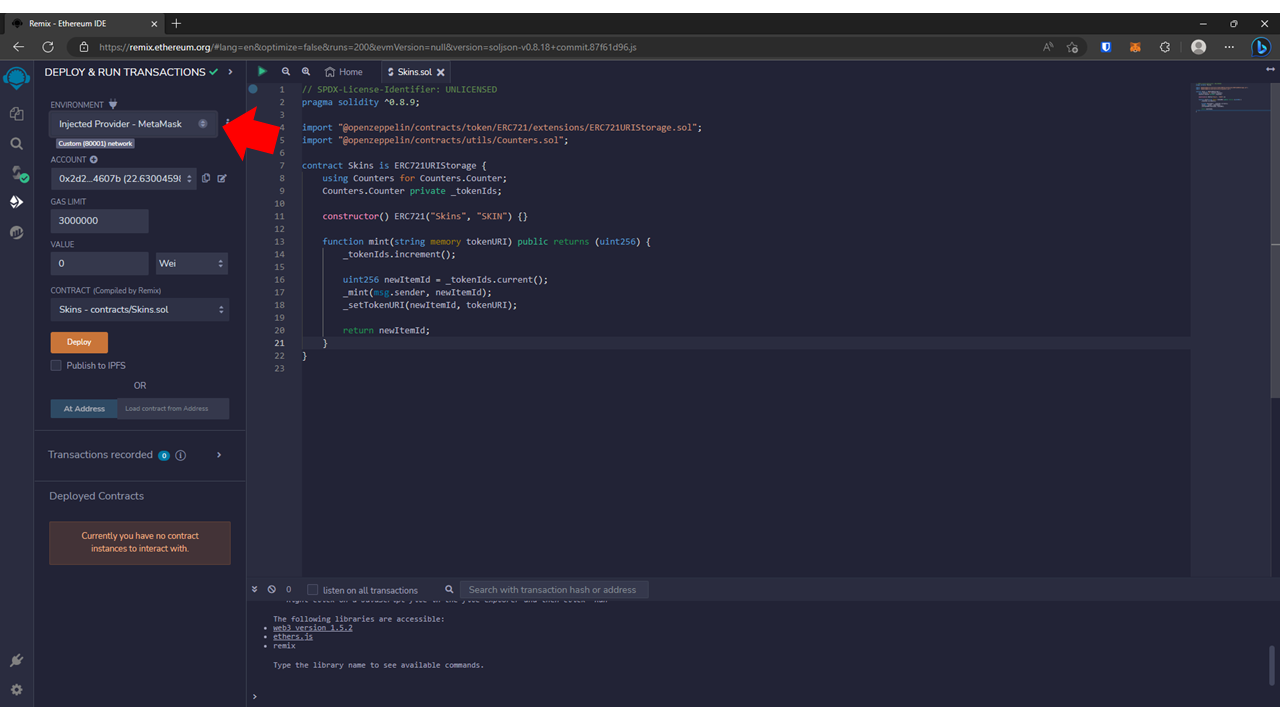
}

}

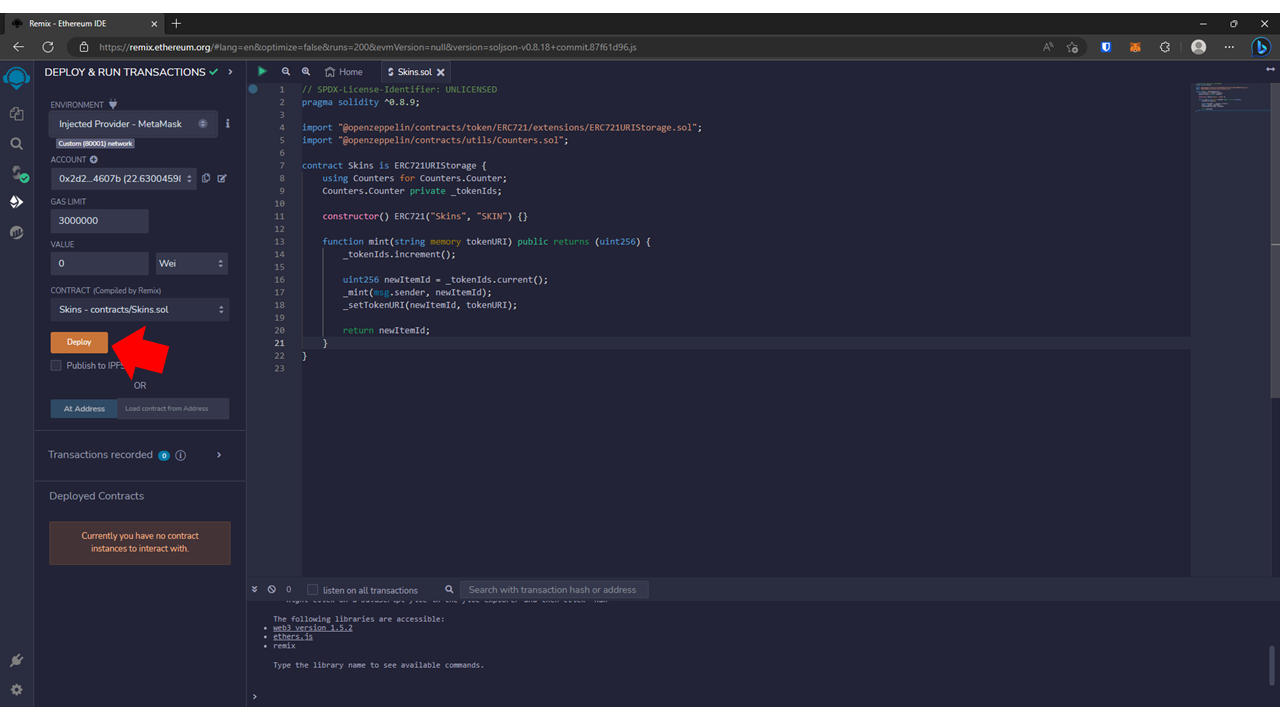
1. Go to the "Compile" tab and click "Compile Skins.sol". If there are no errors, you should be able to see a green checkmark on the "Compile" tab.



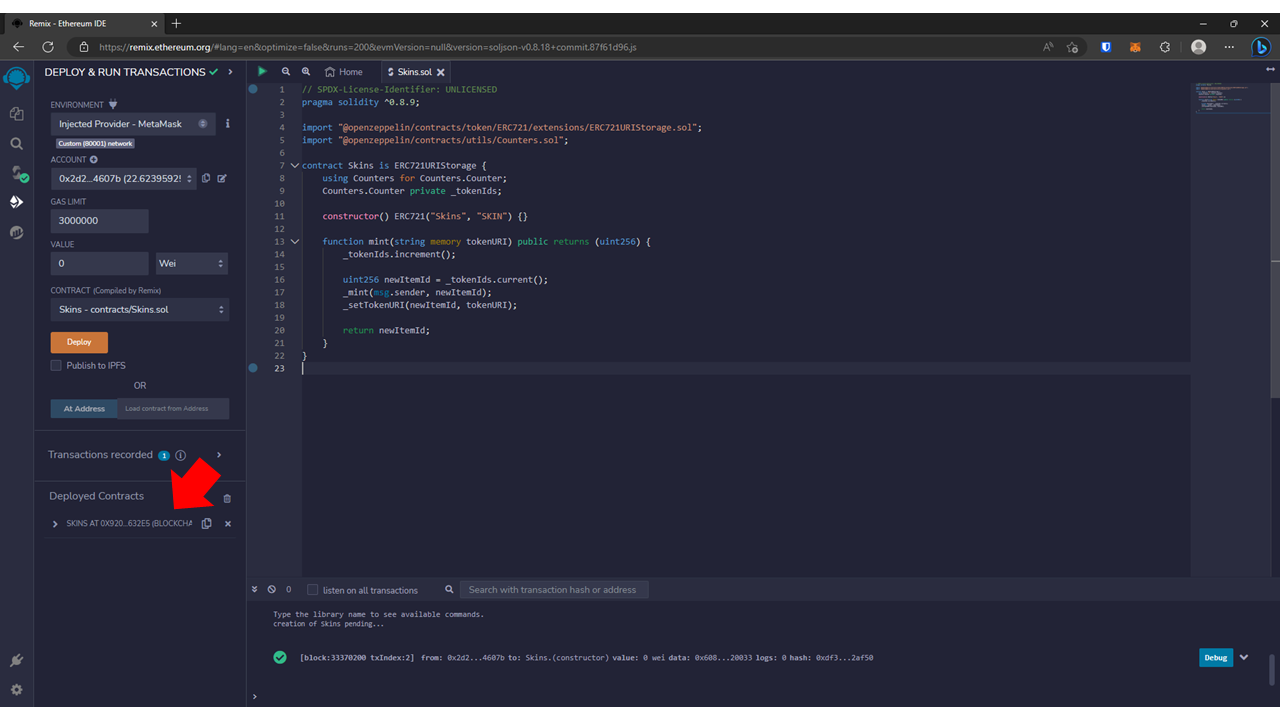
1. Go to the "Deployment" tab and change the environment to "Injected Provider - Metamask".



1. Click the "Deploy" button and wait for the MetaMask confirmation. Click "Confirm" on MetaMask to confirm the transaction.



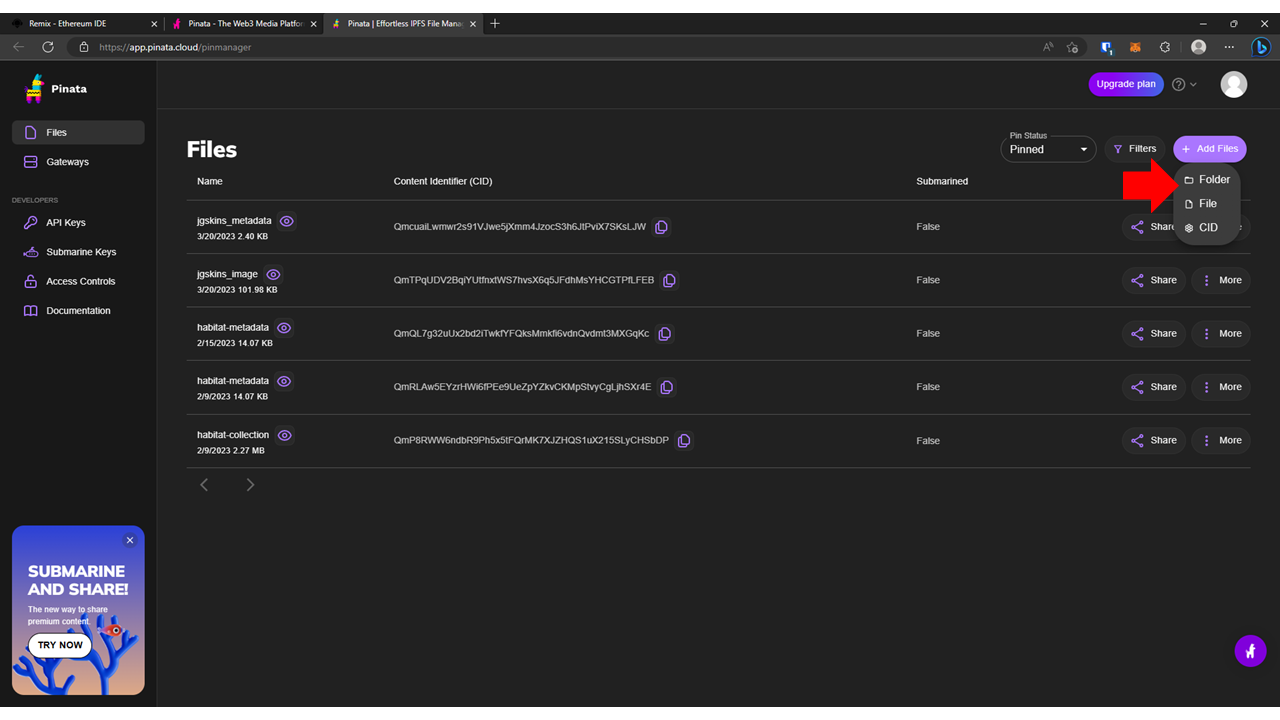
1. Wait for the transaction to finish.
2. Take note of the smart contract address of the deployed smart contract.



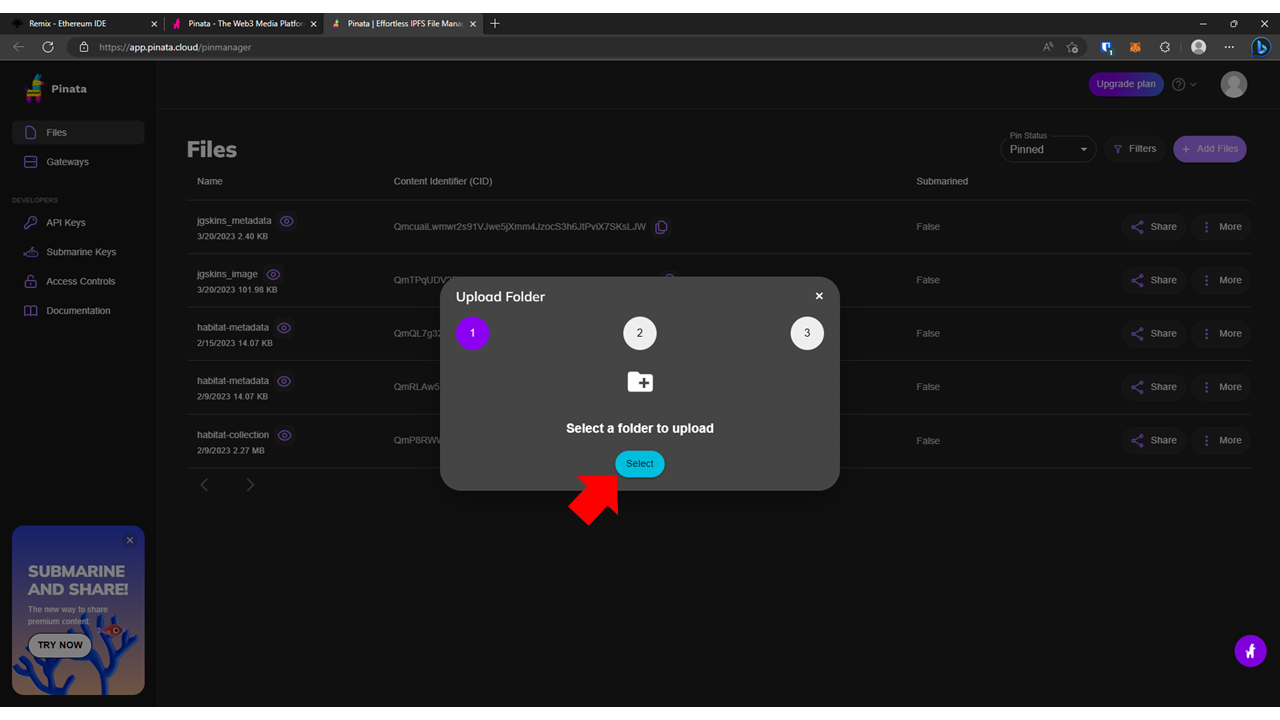
1. Prepare images for the skins in the game.



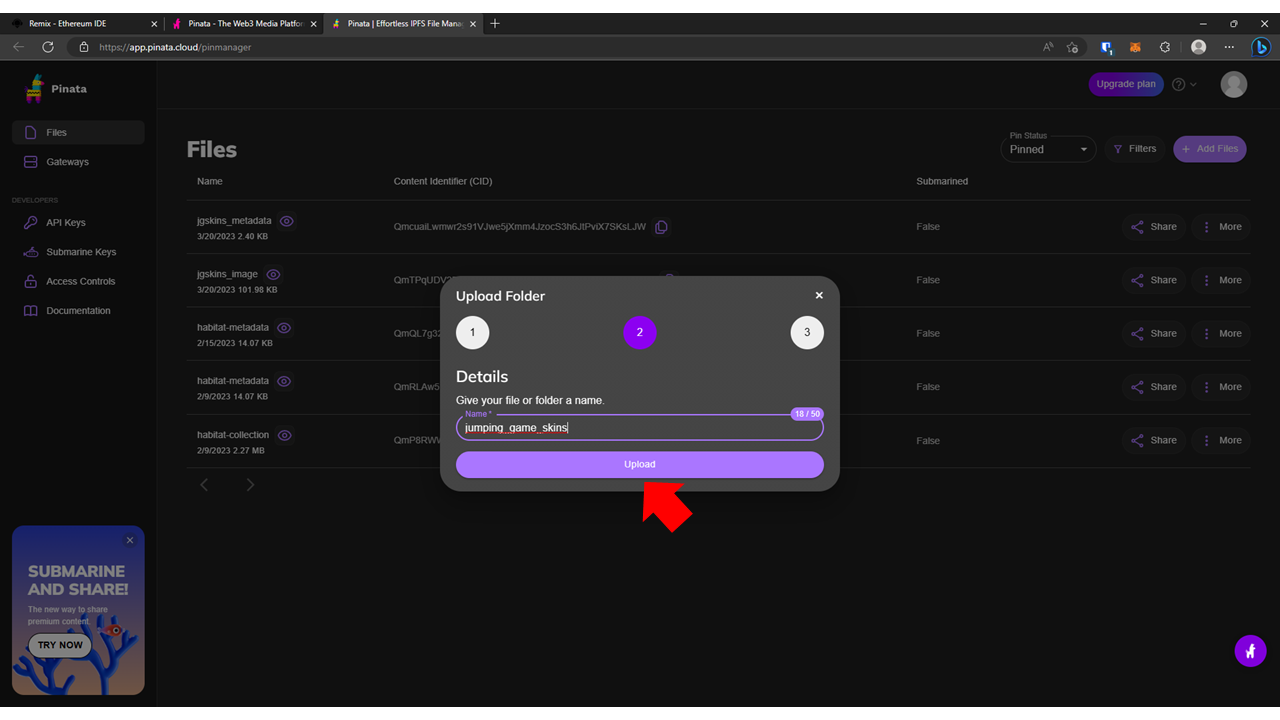
1. Go to Piñata (https://www.pinata.cloud/) and login or create an account.
2. On your account dashboard, click "Add Files" and select "Folder".



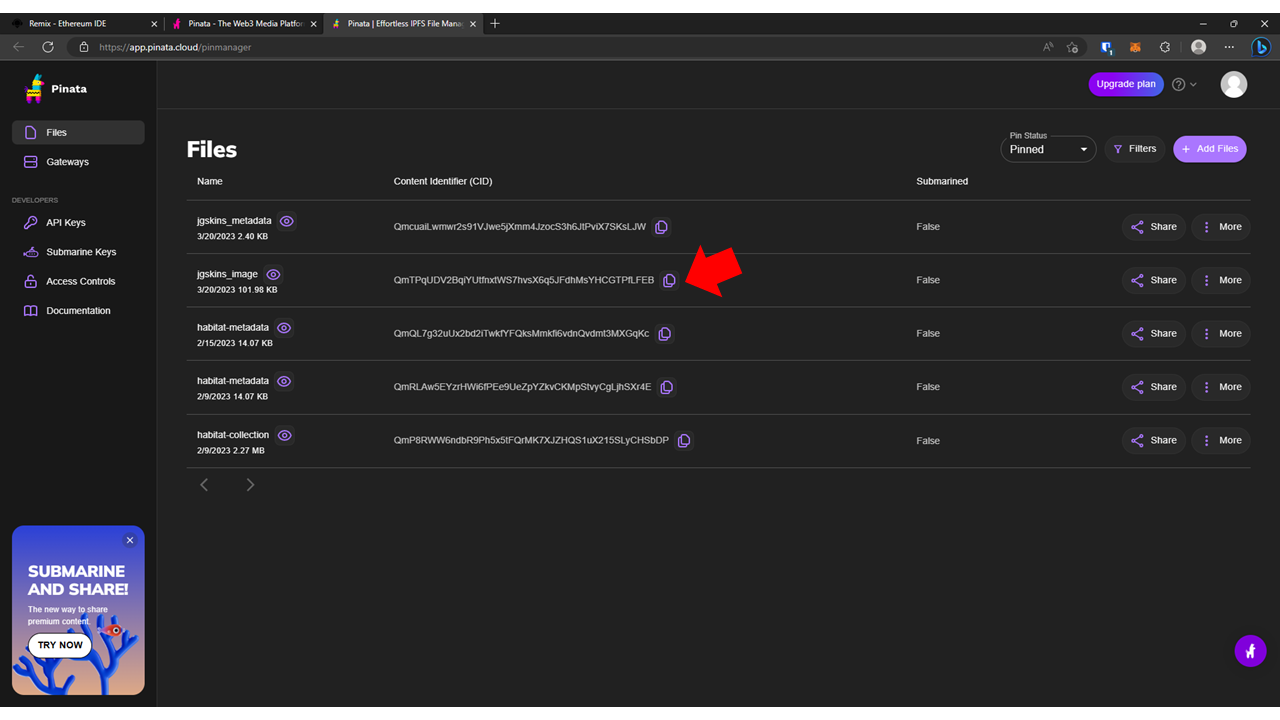
1. Choose a folder to upload that contains the images for the skins.



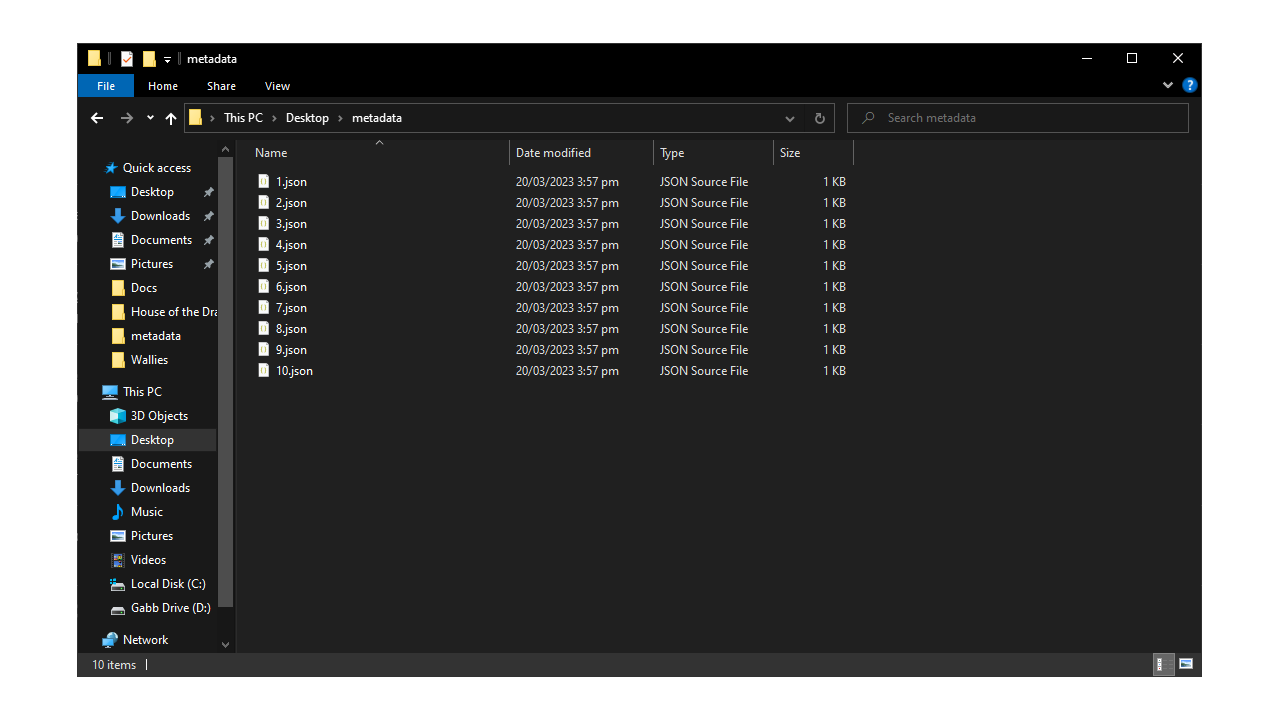
1. Give a distinct name to the folder you want to upload.



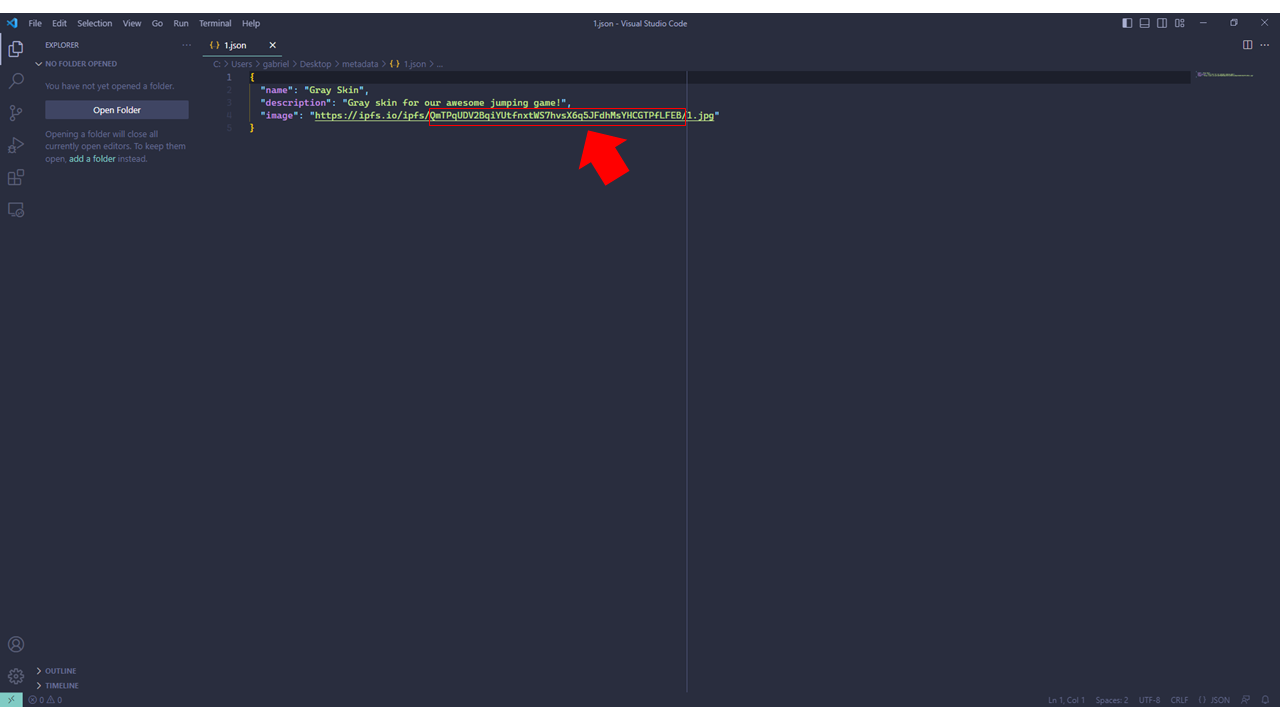
1. Once uploaded, take note of the Content Identifier (CID) of the folder.



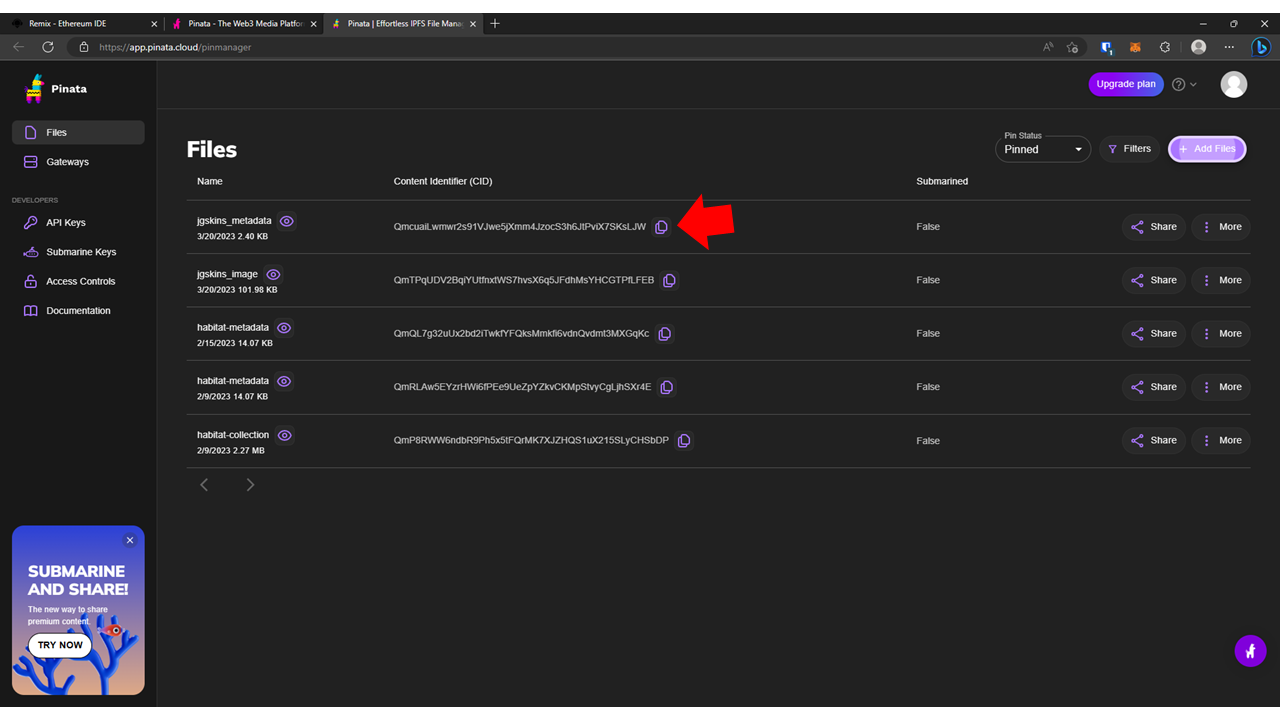
1. Let’s now create metadata for the images we have uploaded in the IPFS.



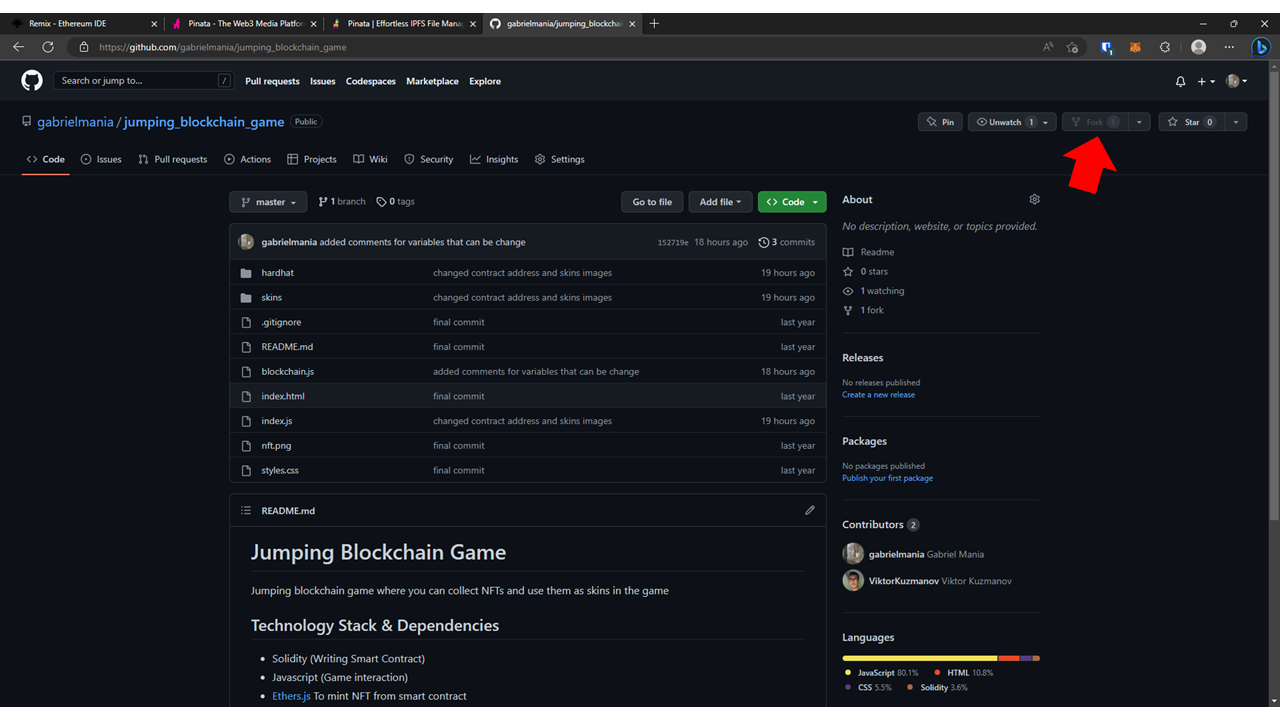
1. In the metadata files, edit the CID to the corresponding CID of the images you have uploaded in the IPFS. Make sure to also change the file name.



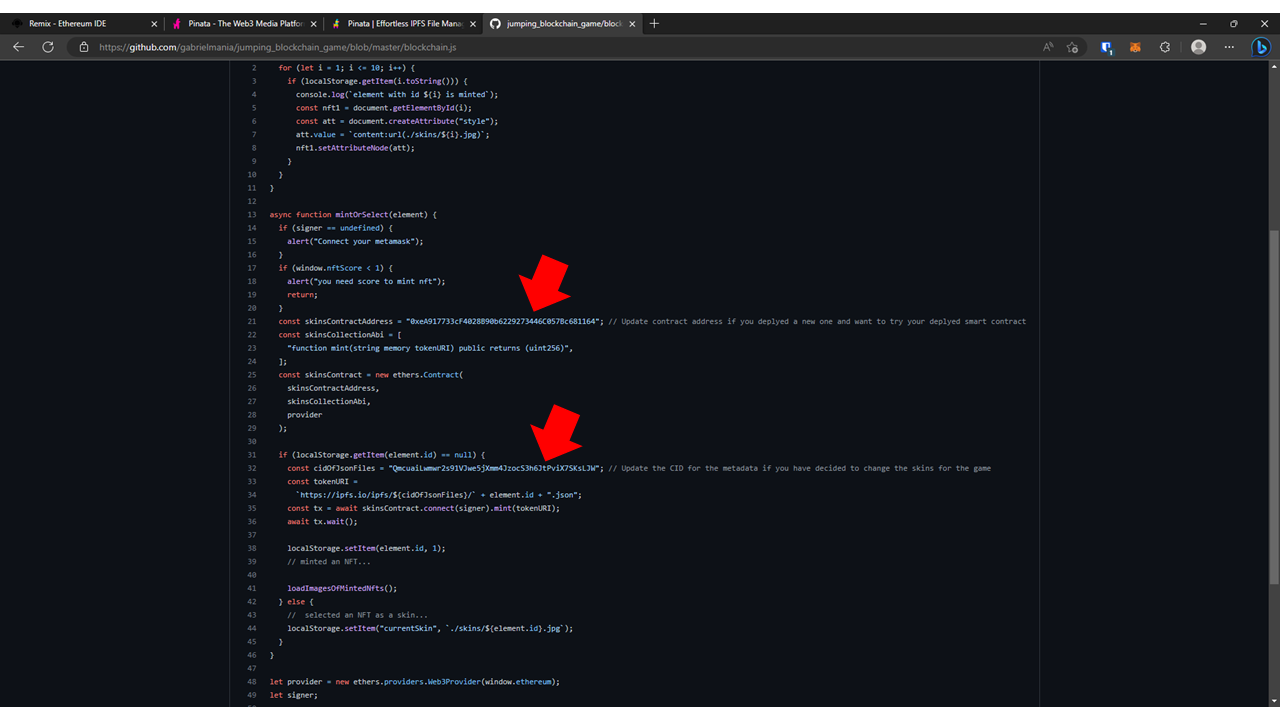
1. Once done with the editing of the metadata, we can now upload it to the IPFS via Piñata. The process is the same as uploading the images.
2. Take note of the CID of the metadata.



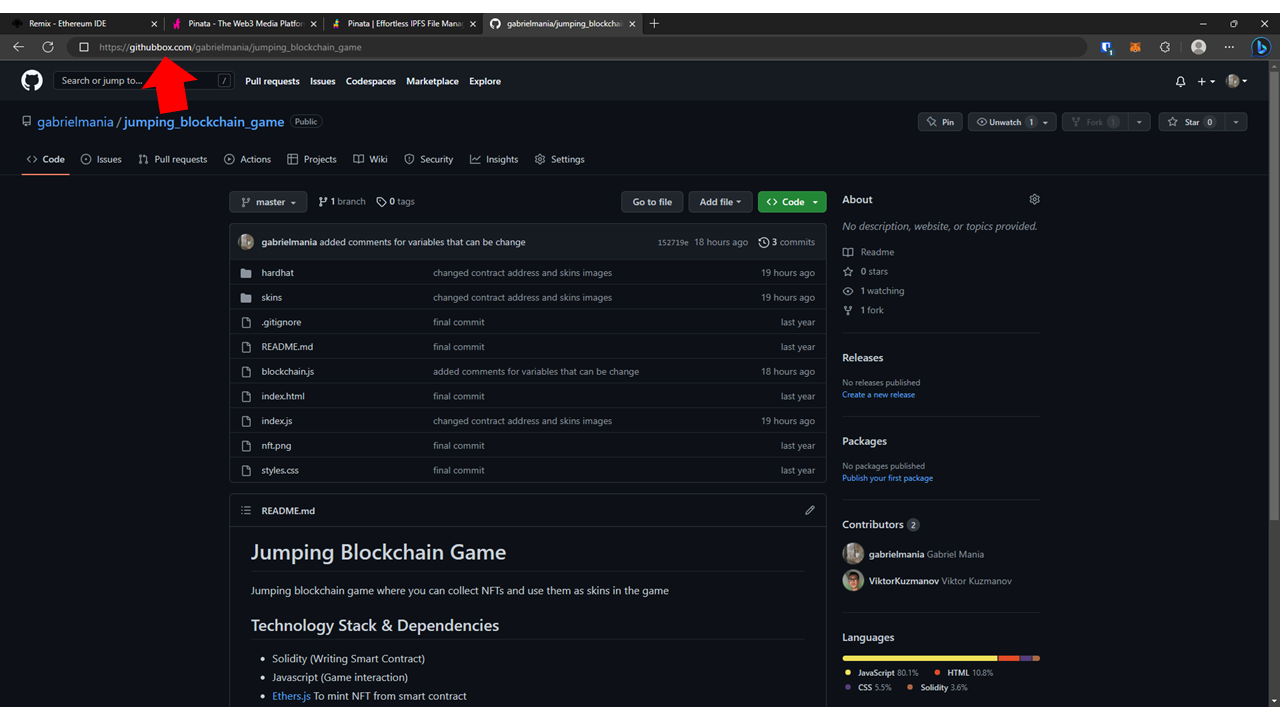
1. Now let’s proceed with the front-end code. Go to GitHub (https://github.com/) and login or create a new account.
2. Copy this link to the URL tab in your browser: https://github.com/gabrielmania/jumping\_blockchain\_game
3. Fork a copy of this repository to your GitHub account.



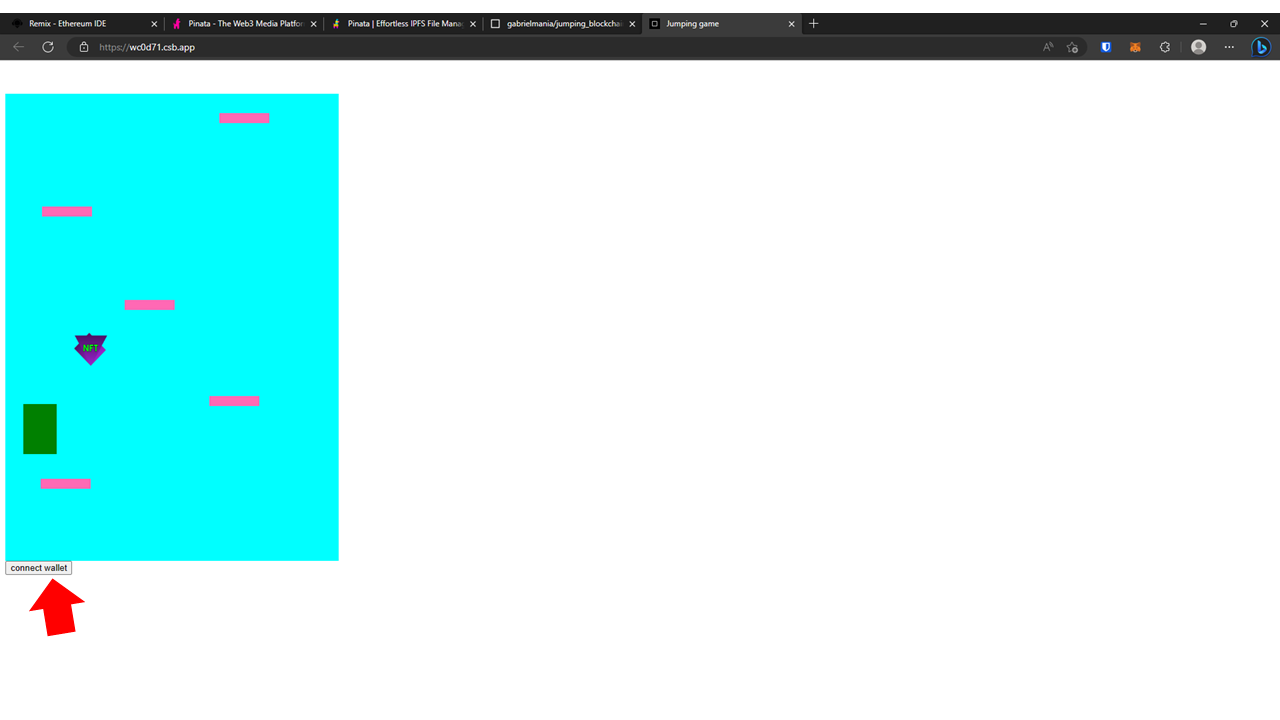
1. On line 21, change the contract address to the one you have deployed on the blockchain.
2. On line 23, change the CID to the CID of the metadata you have uploaded to the IPFS.



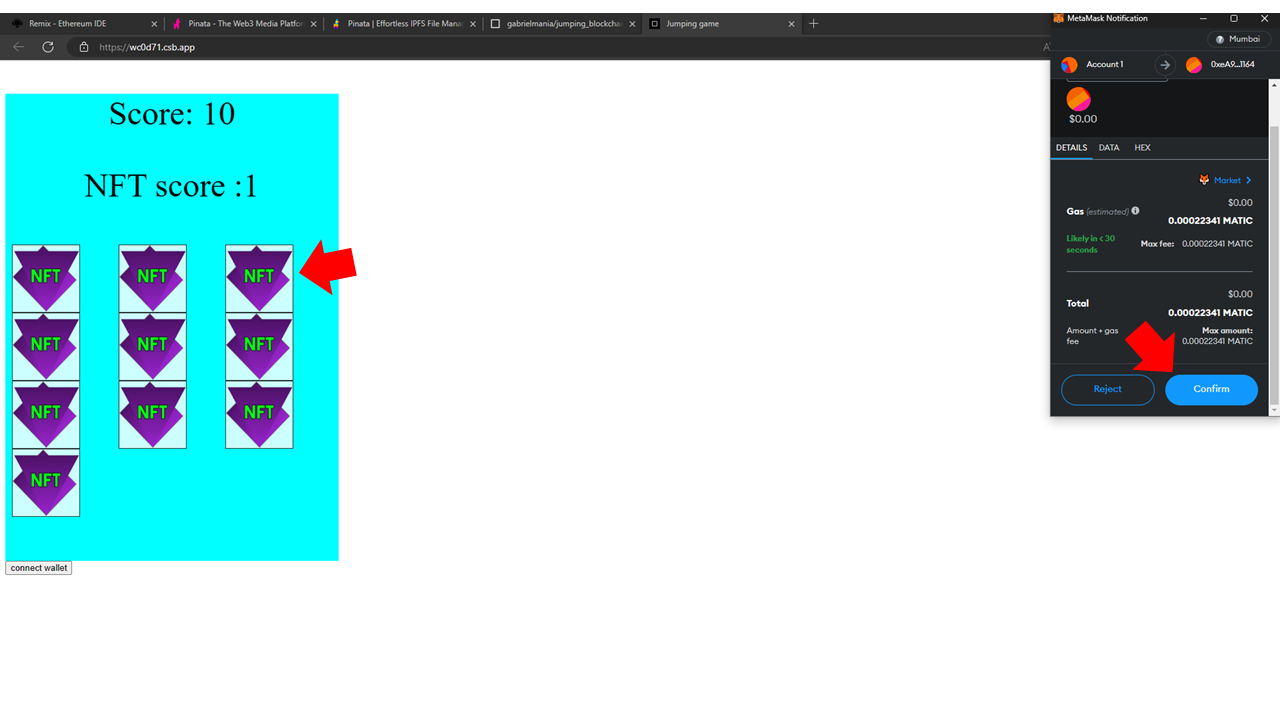
1. Go back to the main page of the repository. In the URL bar, add the word "box" to your URL after "github". Your final URL should look like this: <https://githubbox.com/gabrielmania/jumping_blockchain_game>



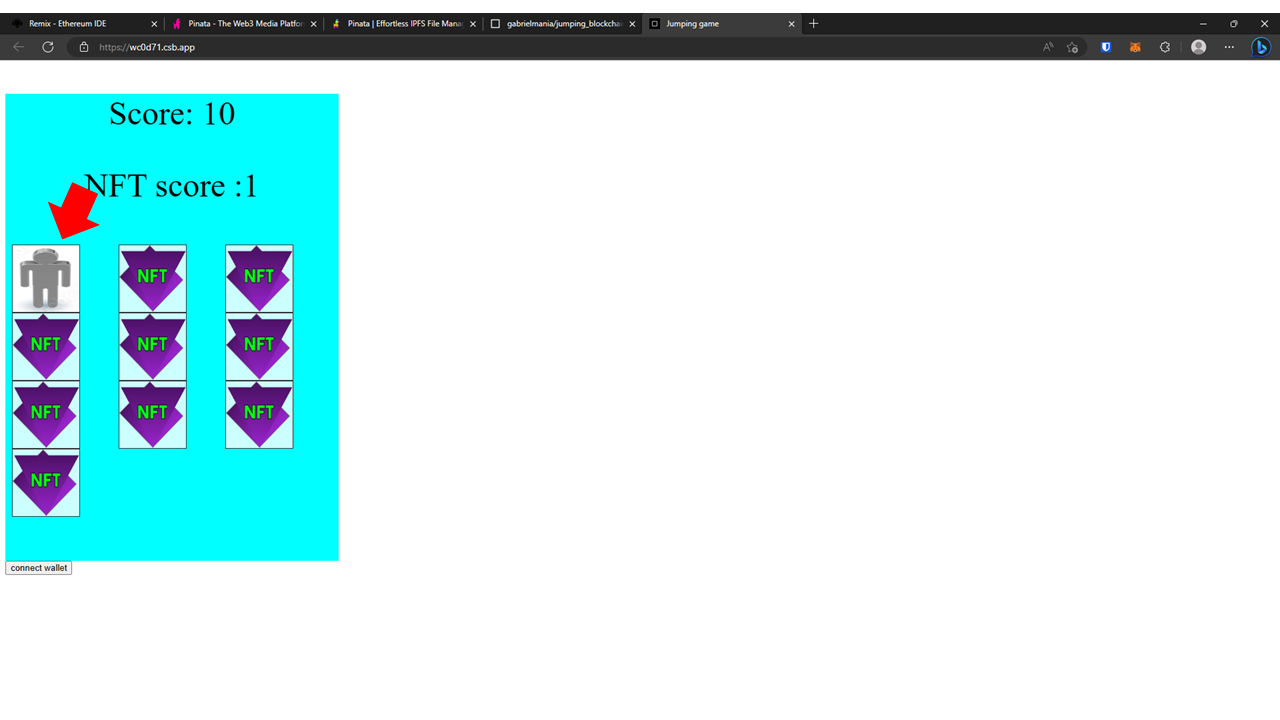
1. You will be directed to CodeSandbox with the code you have forked and edited.
2. You can click the pop-up icon to make it fullscreen, and you can now play the game!
3. Once you have scored some NFTs, you can connect your wallet by clicking the "Connect Wallet" button.



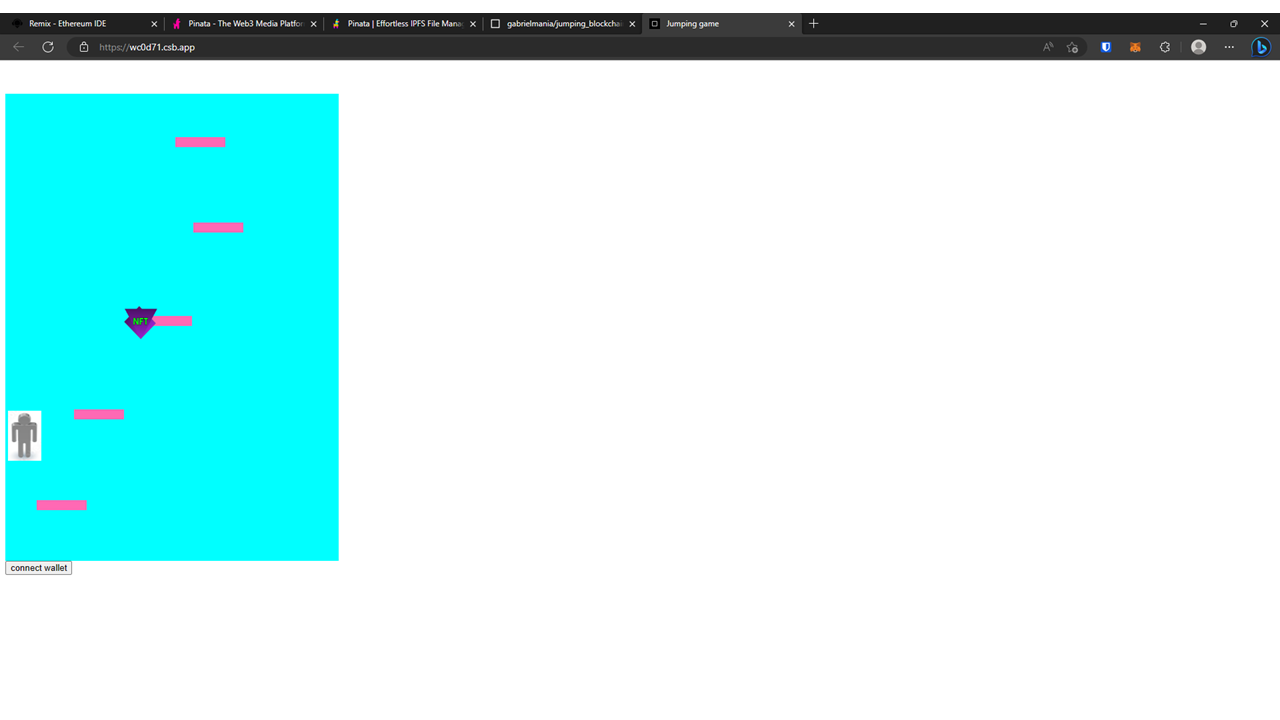
1. Click on an NFT that you want to mint, and then click "Confirm" on the MetaMask wallet to proceed with the minting of the skin.



1. Once the skin is minted, click on the skin to choose it as the skin and refresh the page to play the game again.



1. Collect as many NFTs as you can!



Congratulations to all the participants on successfully completing the live coding session and creating your own Web 3.0 Jumping Game! You all did an amazing job and should be proud of your accomplishment.

I encourage you to keep exploring and improving on your code. You have a solid foundation to build upon and can continue to add new features and functionality to your game. Don't be afraid to experiment and push the boundaries of what's possible.

Remember, the world of Web 3.0 development is constantly evolving and there are always new possibilities to explore. I encourage you to continue learning and exploring this exciting field. Keep up the great work!