

Welcome everyone to this live-coding session on building a DeFi Slot Machine! Today, we're going to explore the exciting world of Decentralized Finance (DeFi) and learn about the key differences between DeFi and Centralized Finance (CeFi). If you're new to the concept of DeFi, you're in the right place. During this session, we'll walk you through the process of building a DeFi Slot Machine using Solidity, a programming language for developing smart contracts on the Ethereum blockchain.

To help you follow along, we've provided links to all the resources you'll need, including a GitHub repository with the code and a tutorial on building a simple DeFi Slot Machine. By the end of this session, you'll have a better understanding of the key differences between DeFi and CeFi, as well as the practical skills to build your own DeFi application. So, let's get started and dive into the exciting world of DeFi!

This guide assumes that you have the following tools:

* [Remix IDE](https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.17+commit.8df45f5f.js)
* [Metamask Wallet](https://metamask.io/)
* [Mumbai Test Token](https://faucet.polygon.technology/)
* Visual Studio Code

To create a DeFi Slot Machine using ChainLink VRF, follow these simple steps:

1. Go to Remix website and create a new file with ".sol" extension which is used for Solidity programming language.

Graphical user interface, application

Description automatically generated

1. Copy the source code of the Slot Machine from this repository: <https://github.com/devbfactor/slot-machine.github.io/blob/master/hardhat/contracts/SlotMachine.sol>

Graphical user interface, text, application, email

Description automatically generated

1. Paste the copied code into the new file that you created in Remix.

Text

Description automatically generated

1. Open the ChainLink VRF page at https://vrf.chain.link/mumbai to get a random number for the Slot Machine. Create and fund a subscription account by following these steps:
   1. Open MetaMask and set it to use the Mumbai testnet. The Subscription Manager detects your network based on the active network in MetaMask.

Graphical user interface, application, Teams

Description automatically generated

* 1. Check MetaMask to make sure you have testnet ETH and LINK on Mumbai. You can get testnet MATIC and LINK at faucets.chain.link.
  2. Open the Subscription Manager at vrf.chain.link.
  3. Open the Subscription Manager
  4. Click Create Subscription and follow the instructions to create a new subscription account. MetaMask opens and asks you to confirm payment to create the account on-chain. After you approve the transaction, the network confirms the creation of your subscription account on-chain.

Graphical user interface, application, Teams

Description automatically generated

* 1. After the subscription is created, click Add funds and follow the instructions to fund your subscription.

Graphical user interface, text, application, email

Description automatically generated  
  
For this example, a balance of 2 LINK is sufficient. MetaMask opens to confirm the LINK transfer to your subscription. After you approve the transaction, the network confirms the transfer of your LINK token to your subscription account.  
  
Graphical user interface, application

Description automatically generated

* 1. After you add funds, click Add consumer. A page opens with your account details and subscription ID.  
       
     Graphical user interface, application

     Description automatically generated
  2. Record your subscription ID, which you need for your consuming contract. You will add the consuming contract to your subscription later.

Text

Description automatically generated with medium confidence

You can always find your subscription IDs, balances, and consumers at vrf.chain.link.

1. After importing the ChainLink VRF contract, go back to the Remix IDE and compile your smart contract. Make sure there are no errors.

A screenshot of a computer

Description automatically generated with medium confidence

1. Once the compilation is successful, set the environment to "Injected Provider Metamask" in the "Environment" dropdown menu. This will connect your Remix IDE to your MetaMask wallet.

Graphical user interface, text, application, Teams

Description automatically generated

1. Inside the "Deploy & Run Transactions" section, enter your Subscription ID and the minimum bet or entry fee (e.g. "1234" and "100000000000000" wei).

A screenshot of a phone

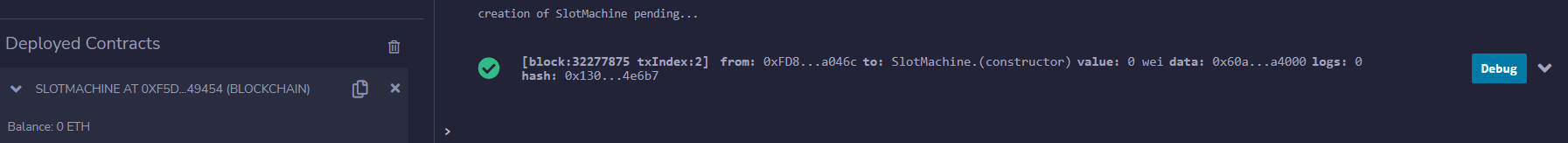
Description automatically generated with medium confidence

1. Click the "Deploy" button to deploy your smart contract on the blockchain.

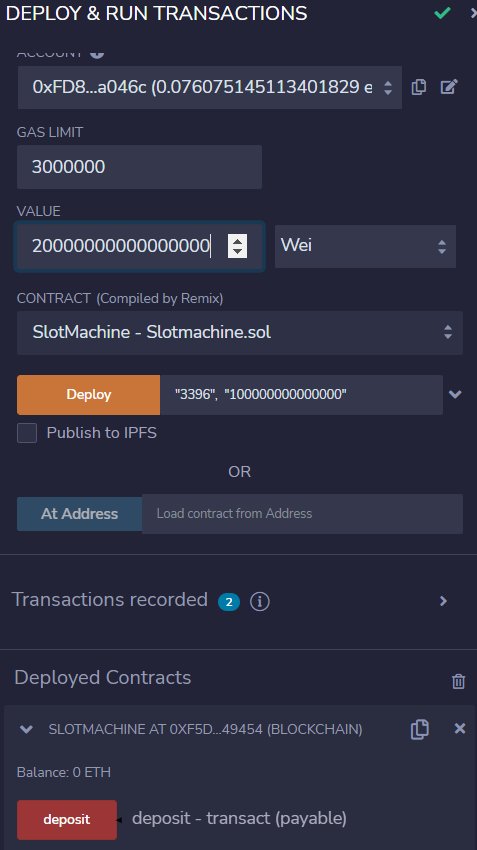
Graphical user interface, application

Description automatically generated

1. Once the deployment is successful, save the contract address for future reference.



1. Enter the minimum value and deposit an amount to set Prize. For this example, a balance of 20000000000000000 wei is sufficient. This Prize for the game will be stored and deposited inside of the Contract Address.



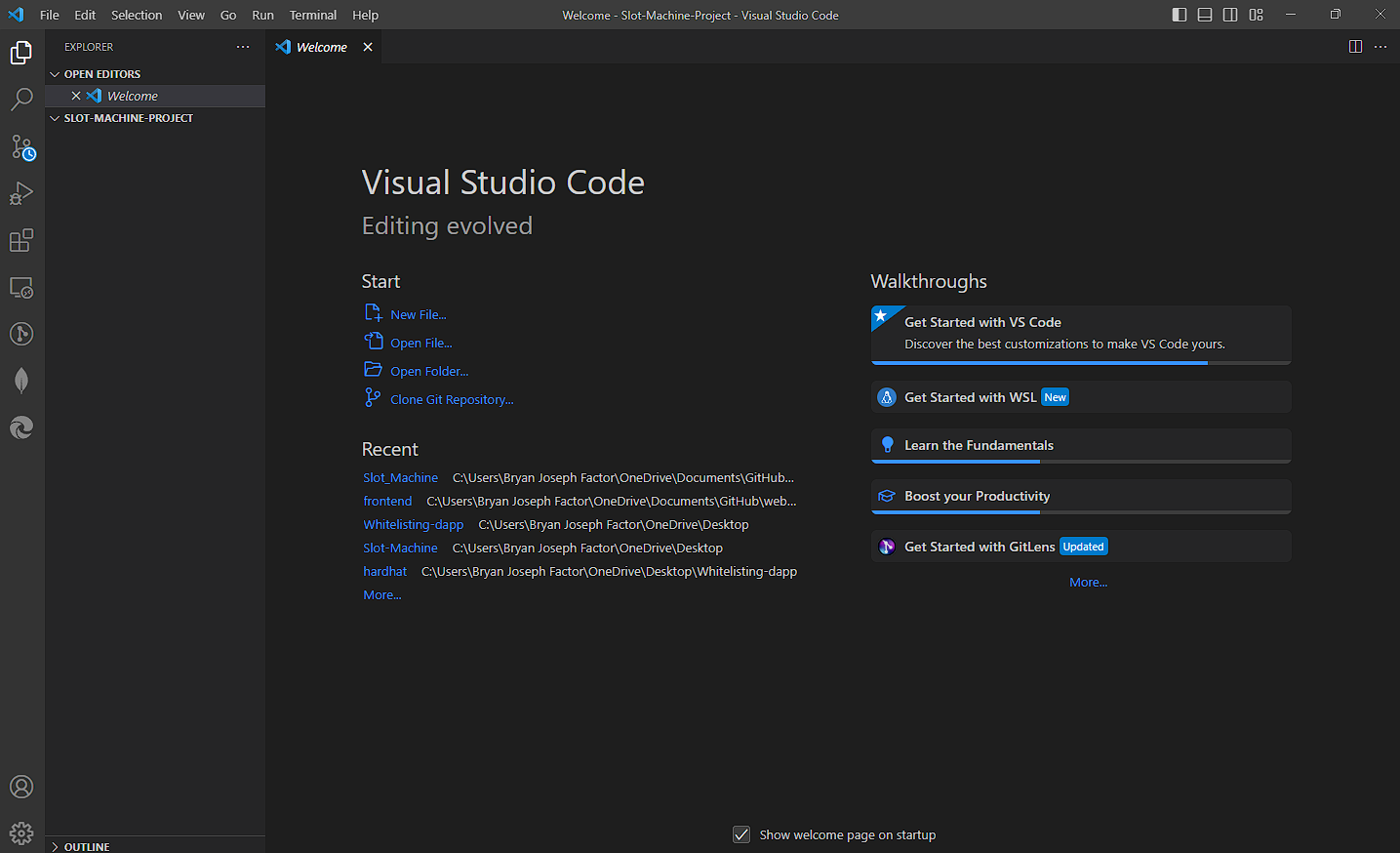
1. Go back to the "Compiler" tab and copy the ABI (Application Binary Interface) of your smart contract. This will be used in the front-end of your DeFi Slot Machine.

Graphical user interface, text, application

Description automatically generated

Great! Here are the remaining steps to create the frontend of your DeFi Slot Machine:

1. Open Visual Studio Code and create a new directory on your desktop.

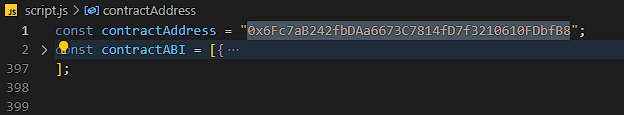


1. Inside the directory, create 3 new files: "index.html", "style.css", and "script.js".

Graphical user interface, text, application

Description automatically generated

1. Open this link: https://github.com/devbfactor/slot-machine.github.io/blob/master/index.html and copy the source code for the "index.html" file. Do not make any changes to the code.
2. Go back to Visual Studio Code and open the "index.html" file. Paste the copied source code from the repository.
3. Open this link: https://github.com/devbfactor/slot-machine.github.io/blob/master/style.css and copy the source code for the "style.css" file. Do not make any changes to the code.
4. Go back to Visual Studio Code and open the "style.css" file. Paste the copied source code from the repository. At this point, you now have the structure and design for your Slot Machine, but we still need to add the functionality.
5. Open this link: https://github.com/devbfactor/slot-machine.github.io/blob/master/script.js and copy the source code for the "script.js" file. Do not make any changes to the code.
6. Go back to Visual Studio Code and open the "script.js" file. Paste the copied source code from the repository.
7. Inside your "script.js" file, change the contract address and ABI to match the smart contract you just deployed. Make sure to include only the array part for the ABI.



1. Right-click on the "index.html" file, select "Copy Path", open a browser, and paste the path into the URL bar.

Graphical user interface

Description automatically generated

Congratulations! You have now successfully created a DeFi Slot Machine with a functional frontend.

Graphical user interface, application

Description automatically generated

-END-