============================================================================================

NFT School

---------------------------------------------------------------------------------------------------

**Build a minting Service**

由于使用了**内容寻址**，IPFS 解决了这个潜在的灾难性问题。

向 IPFS 添加数据会生成**内容标识符** （content identifier ）(CID) 直接从数据本身派生，并链接到 IPFS 网络中的数据。

因为 CID 只能引用一段内容，所以我们知道没有人可以在不破坏链接的情况下替换或更改内容**。**

IPFS solves this potentially catastrophic problem thanks to its use of content addressing. Adding data to IPFS produces a content identifier (opens new window) (CID) that is directly derived from the data itself, and links to the data in the IPFS network. Because a CID can only ever refer to one piece of content, we know that nobody can replace or alter the content without breaking the link.

使用该 CID，只要网络上至少存在一个副本，任何人都可以从 IPFS 网络获取其数据的副本——即使原始提供者已经消失。这使得 CID 非常适合 NFT 存储。我们需要做的就是将 CID 放入一个ipfs://URI 中，例如ipfs://bafybeidlkqhddsjrdue7y3dy27pu5d7ydyemcls4z24szlyik3we7vqvam/nft-image.png，并且我们有一个从区块链到我们的令牌数据的不可变链接。

当然，在某些情况下，您**确实**希望在 NFT 发布后更改其元数据。那没问题！您只需要在智能合约中添加支持，以便在发行token后更新token的 URI。这将允许您将原始 URI 更改为新的 IPFS URI，同时仍然在区块链的交易历史中留下初始版本的记录。这通过让每个人都清楚地了解更改的内容、时间和人员来提供问责制。

使用 nft.storage，我们可以在 IPFS 上免费提供我们的 NFT 数据，并自动处理内容持久性！**值得注意的是，nft.storage 包括去中心化Filecoin**中的长期“冷存储”网络，以及使用IPFS 集群的“热存储” . 作为一个不错的奖励，您还可以获得一个 JavaScript客户端库 它可以帮助您将数据上传到 IPFS 并在一行代码中生成 NFT 元数据。

Of course, there may be some cases in which you **do** want to change the metadata for an NFT after it's been published. That's no problem! You'll just need to add support to your smart contract for updating the URI for a token after it's been issued. That will let you change the original URI to a new IPFS URI while still leaving a record of the initial version in the blockchain's transaction history. This provides accountability by making it clear to everyone what was changed, when, and by whom.

Using nft.storage, we can make our NFT data available on IPFS for free, with [content persistence](https://nftschool.dev/concepts/content-persistence/) automatically taken care of! Notably, nft.storage includes both long-term "cold storage" in the decentralized [Filecoin (opens new window)](https://filecoin.io) network, as well as "hot storage" using [IPFS Cluster (opens new window)](https://cluster.ipfs.io/). As a nice bonus, you also get a JavaScript [client library (opens new window)](https://www.npmjs.com/package/nft.storage) that helps you upload data to IPFS and generate NFT metadata all in one line of code.

============================================================================================

December 27, 2021

**How to create and deploy an ERC-721 (NFT)**

**Overview**

Digital collectibles compatible with the ERC-721 standard have become very popular since the launch of [Cryptokitties](https://www.cryptokitties.co/) and have moved forward towards mass adoption in recent months. This guide will cover creating and deploying our very own ERC-721 (NFT) using the [0xcert/ethereum-erc721](https://github.com/0xcert/ethereum-erc721) smart contract library.

**What is a Non-Fungible Token?**

Fungible means to be the same or interchangeable. For example, Ethereum tokens, all the members of a particular token class, have the same value. The same can be said of Cardano tokens. Fungible tokens are interchangeable 1:1.

With this in mind, NFTs are unique; each one is different. Every single token has unique characteristics and values. The types of things that can be NFTs are collectible cards, artworks, airplane tickets, etc. They are all clearly distinguishable from one another and are not interchangeable. Think of Non-Fungible Tokens (NFTs) as rare collectibles; each has unique characteristics, unusual attributes, and most times, its metadata.

**What is ERC-721?**

ERC stands for Ethereum Request for Comment, and 721 is the proposal identifier number. ERCs are application-level standards in the Ethereum ecosystem, they can be a smart contract standard for tokens such as [ERC-20](https://www.quiknode.io/guides/solidity/how-to-create-and-deploy-an-erc20-token), the author of an ERC is responsible for building consensus with the Ethereum community and once the proposal is reviewed and approved by the community it becomes a standard. You can track the recent ERC proposal [here](https://github.com/ethereum/EIPs/issues). ERC-721 was created to propose the functionality to track and transfer NFTs within smart contracts.

[ERC-721](https://eips.ethereum.org/EIPS/eip-721) is an open standard that describes how to build Non-Fungible tokens on EVM (Ethereum Virtual Machine) compatible blockchains; it is a standard interface for Non-Fungible tokens; it has a set of rules which make it easy to work with NFTs. NFTs are not only of ERC-721 type; they can also be [ERC-1155](https://eips.ethereum.org/EIPS/eip-1155) tokens.

The following are the set of functions and events defined in the ERC-721 standard:

ERC-721 defines some functions with compliance to ERC-20. This makes it easier for existing wallets to display simple token information.

**ERC20-like Functions:**

**name:**Used to define the token’s name, which other contracts and applications can identify.

**symbol:**Used to define token’s shorthand name or symbol.

**totalSupply:** This function is used to define the total number of tokens on the blockchain; the supply doesn’t have to be constant.

**balanceOf:** Returns number of NFTs owned by an address.

**Ownership Functions**

**ownerOf:**This function returns the address of the owner of a token. As each ERC-721 token is unique and non-fungible, they are represented on the blockchain by an ID. Other users, contracts, apps can use this ID to determine the owner of the token.

**approve:** This function grants or approves another entity the permission to transfer tokens on the owner’s behalf.

**takeOwnership:** This is an optional function that acts like a withdraw function since an outside party can call it to take tokens out of another user’s account. Therefore, takeOwnership can be used when a user has been approved to own a certain amount of tokens and wishes to withdraw said tokens from another user’s balance.

**transfer:** This is another type of transfer function; it allows the owner to transfer the token to another user, just like other digital tokens/coins.

**tokenOfOwnerByIndex:** This is an optional but recommended function. Each owner can own more than one NFT at the same time. Its unique ID identifies every NFT, and eventually, it can become difficult to keep track of IDs. So the contract stores these IDs in an array and the tokenOfOwnerByIndex function lets us retrieve this information from the array.

**Metadata Function**

**tokenMetadata:** This optional feature is an interface that lets us discover a token’s metadata or a link to its data.

**Events**

**Transfer:**This event is fired when the ownership of the token changes from one individual to another. It emits the information on which account transferred the token, which account received the token, and which token (by ID) was transferred.

**Approve:** This event is fired when a user approves another user to take ownership of the token, i.e., it is fired whenever approve function is executed. It emits the information on which account currently owns the token, which account is approved to take ownership of the token in the future, and which token (by ID) is approved to have its ownership transferred.

**Use cases of Non-Fungible Tokens (NFTs):**

**Digital art (or physical art):** Art pieces are the most popular use cases of NFTs. Digital art auctions were the first application of NFTs and continue to grow.

**Gaming:** Providing in-game purchases and collectibles of games.

**Real estate:** Tokenizing properties and smart contracts and carry buying and selling.

**Finance:** Financial instruments like loans, futures, and other responsibilities.

**Software titles:** Software licenses to ensure anti-piracy and privacy.

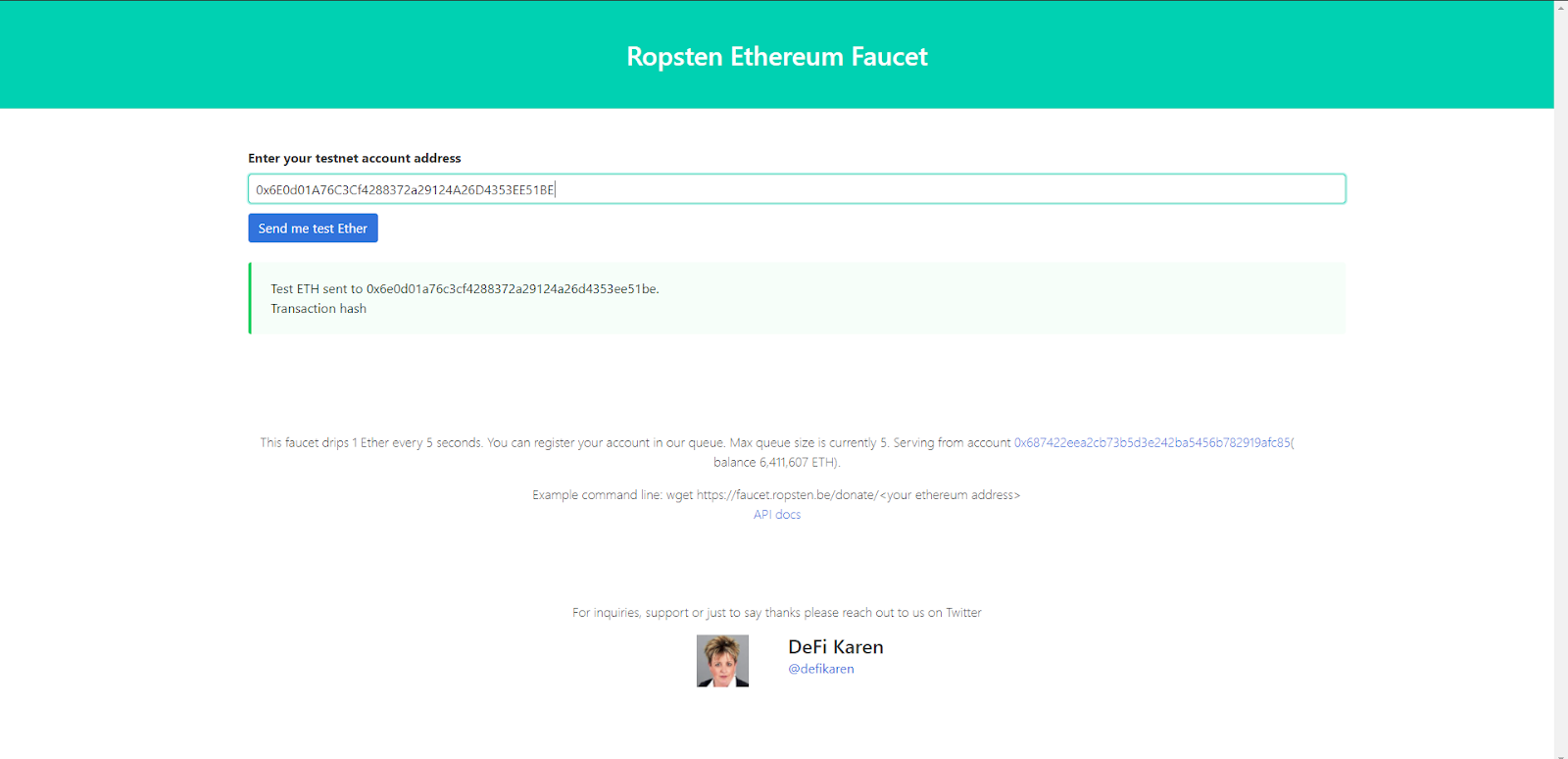
**Concert tickets/Sports match tickets:** To ensure that no fraud happens in ticket selling and fans can have a single place to view past experiences.

**KYC compliance:** Creating a token for a specific user’s KYC.

**Getting test ETH**

Now that we know what ERC-721 tokens are and how they work let’s see how we can build and deploy our own tokens.

We’ll deploy our contract on the Ropsten testnet. To get started, you will need the [Metamask](https://metamask.io/) browser extension to create an ETH wallet and some test ETH, which you can get by going to the [Ropsten faucet](https://faucet.ropsten.be/). You'll need to select Ropsten Test Network on your Metamask wallet and copy-paste the wallet address into the text field in the faucet, then click Send me test Ether.



**Adding Files to IPFS**

Before writing our NFT contract, we need to host our art for NFT and create a metadata file; for this, we’ll use [IPFS](https://ipfs.io/) - a peer-to-peer file storing and sharing distributed system. Download and install IPFS CLI  based on your Operating system by following the [installation guide in IPFS docs.](https://ipfs.io/#install)

Following are the steps for hosting the image and metadata file.

**Step 1**: Creating IPFS repo.

Start the IPFS repo by typing the following in a terminal/cmd window.

$ ipfs init

**Step 2**: Starting the IPFS daemon.

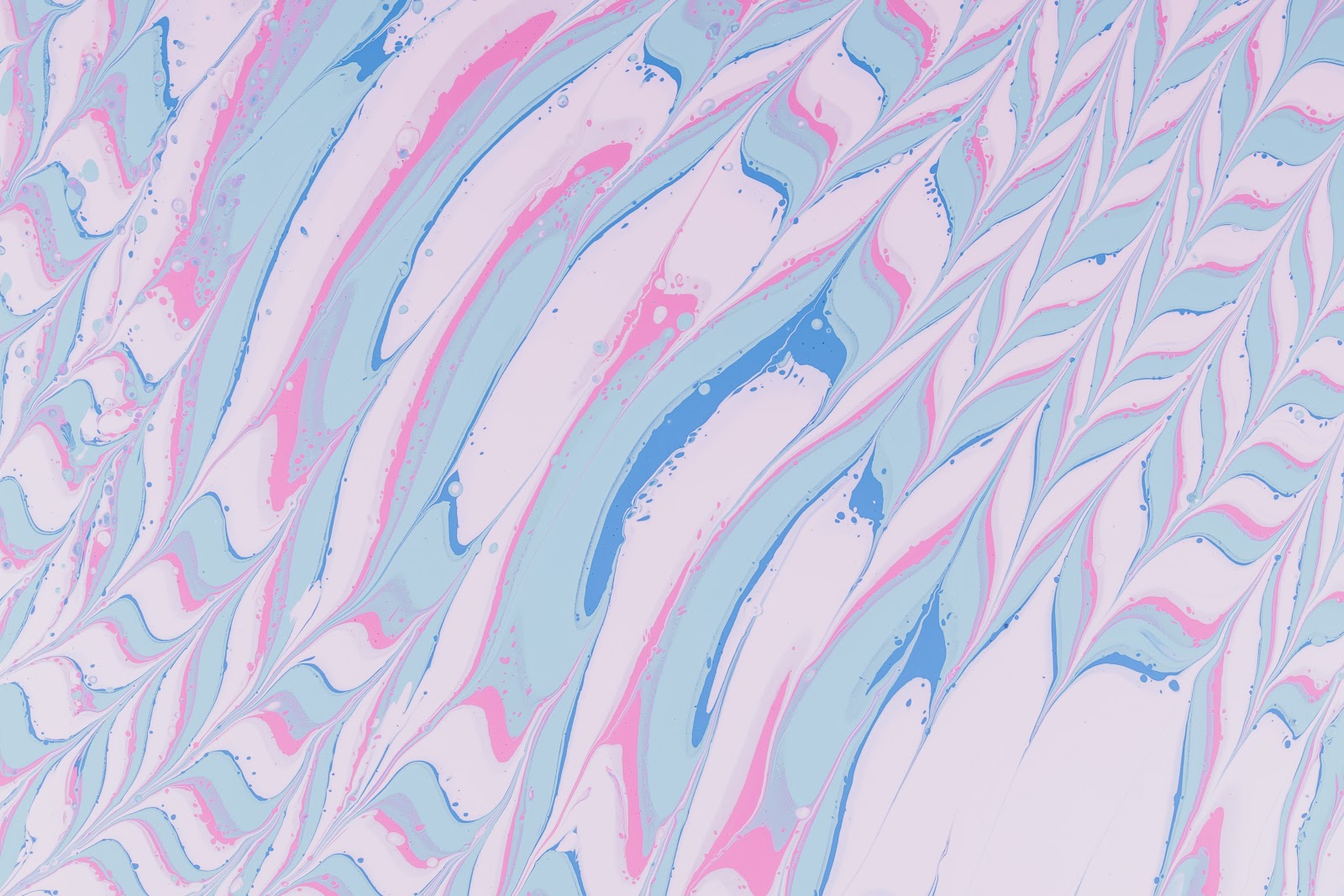
Start IPFS daemon, open a separate terminal/cmd window and type the following.

$ ipfs daemon

**Step 3**: Adding an image to IPFS

Go to the first terminal window and add the image to IPFS (art.png here).

$ ipfs add art.png



Copy the hash starting from Qm and add the “<https://ipfs.io/ipfs/>” prefix to it; it must look something like this <https://ipfs.io/ipfs/QmQEVVLJUR1WLN15S49rzDJsSP7za9DxeqpUzWuG4aondg>

**Step 4**: Adding JSON file to IPFS

Create a JSON file nft.json and save it in the same directory as the image.

JSON file format:

{

"name": "NFT Art",

"description": "This image shows the true nature of NFT.",

"image": "https://ipfs.io/ipfs/QmZzBdKF7sQX1Q49CQGmreuZHxt9sVB3hTc3TTXYcVZ7jC",

}

Now add the JSON file.

$ ipfs add nft.json

Copy the hash starting from Qm and add the “<https://ipfs.io/ipfs/>” prefix to it; it must look something like this <https://ipfs.io/ipfs/QmUFbUjAifv9GwJo7ufTB5sccnrNqELhDMafoEmZdPPng7>

Save this URL. We'll need this to mint our NFT.

**Creating our own token.**

For ease and security, we’ll use the [0xcert/ethereum-erc721](https://github.com/0xcert/ethereum-erc721) contract to create our NFT. With 0xcert/ethereum-erc721, we don’t need to write the whole ERC-721 interface. Instead, we can import the library contract and use its functions.

Head over to the Ethereum Remix IDE and make a new Solidity file, for example - nft.sol

Paste the following code into your new Solidity script:

X-License-Identifier: MIT

pragma solidity 0.8.0;

import "https://github.com/0xcert/ethereum-erc721/src/contracts/tokens/nf-token-metadata.sol";

import "https://github.com/0xcert/ethereum-erc721/src/contracts/ownership/ownable.sol";

contract newNFT is NFTokenMetadata, Ownable {

constructor() {

nftName = "Synth NFT";

nftSymbol = "SYN";

}

function mint(address \_to, uint256 \_tokenId, string calldata \_uri) external onlyOwner {

super.\_mint(\_to, \_tokenId);

super.\_setTokenUri(\_tokenId, \_uri);

}

}

contract newNFT is NFTokenMetadata, Ownable {

constructor() {

nftName = "Synth NFT";

nftSymbol = "SYN";

}

function mint(address \_to, uint256 \_tokenId, string calldata \_uri) external onlyOwner {

super.\_mint(\_to, \_tokenId);

super.\_setTokenUri(\_tokenId, \_uri);

}

}

Explanation of the code above:

Line 1: Specifying [SPDX license](https://spdx.org/licenses/) type, which is an addition after Solidity ^0.6.8. Whenever the source code of a smart contract is made available to the public, these licenses can help resolve/avoid copyright issues. If you do not wish to specify any license type, you can use a special value UNLICENSED or simply skip the whole comment (it won’t result in an error, just a warning).

Line 2: Declaring the solidity version.

Line 4-5: Importing 0xcert/ethereum-erc721 contracts.

Line 7: Starting our Contract named newNFT and mentioning it’s extending NFTokenMetadata and Ownable contracts.

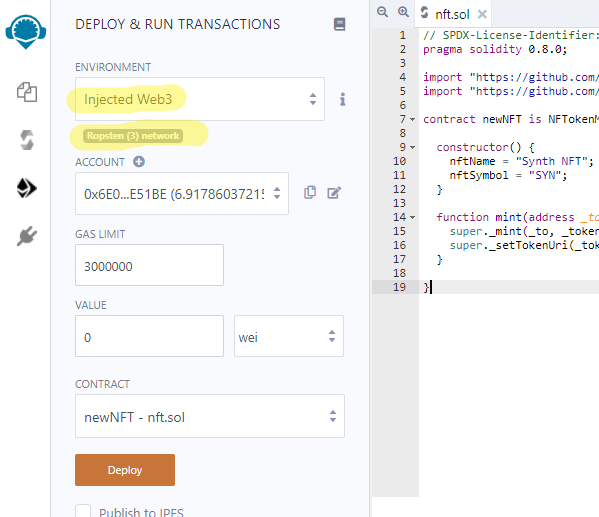
Line 9-12: Initializing the constructor and setting name, a symbol of our token.

Line 14: Declaring function mint with three arguments, variable \_to of type address which will store the address of the receiver of NFT token, variable \_tokenId of uint256 type which will hold the token id, variable \_uri of type string which will store the URI of the JSON file. Declaring mint as external means, it can be accessed from other smart contracts and outside the self scope.

Line 15: Minting token using the address of the receiver and token id.

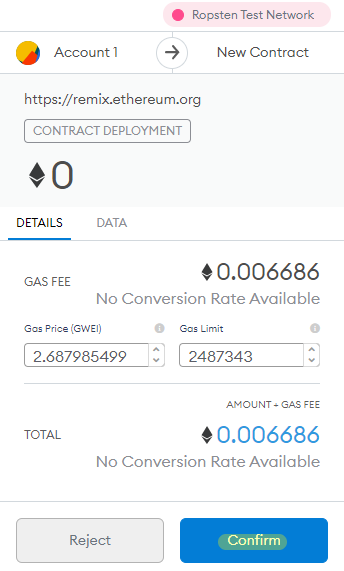
Line 16: Setting token URI using token id and URI of JSON file.

Compile the smart-contract and deploy it using*injected Web3* (make sure to select Ropsten testnet on Metamask before compiling the contract). Approve the transaction from metamask.



If you receive an error message before deployment, “This contract may be abstract,” make sure to select the appropriate contract under the Contract tab.

Confirm the transaction in Metamask



Now go to the “Deployed Contracts” section in Remix and expand the deployed contract. You’ll see a bunch of functions/methods. Expand the mint function and add the following details:

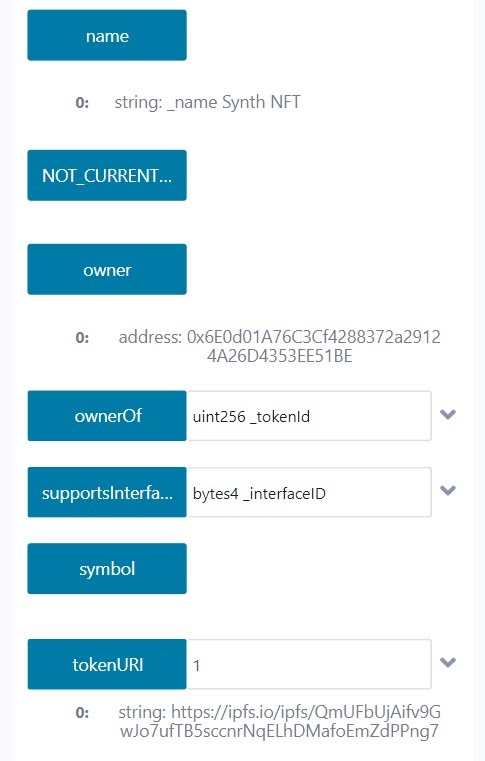
Add your Ropsten address in the \_to the field.

Enter any Big number value in the \_tokenid field (we suggest 1 since it’s the first).

Add URI of JSON file in the \_uri field, which we obtained in the previous section.

Click on transact and confirm the transaction from metamask. Now you have the token on the Ropsten chain.

You can check other details like name, symbol, owner, or tokenuri by entering the token id we mentioned earlier.



**Conclusion**

Congratulations on creating your very own NFT, help your artist friends put their artistic work on the Ethereum blockchain, or become an artist yourself. Checkout [0xcert/ethereum-erc721’s GitHub](https://github.com/0xcert/ethereum-erc721) for more examples.

Subscribe to our [newsletter](https://www.getrevue.co/profile/quiknode) for more articles and guides on Ethereum. If you have any feedback, feel free to reach out to us via [Twitter](https://twitter.com/QuickNode). You can always chat with us on our [Discord](https://discord.gg/ahckhyA) community server, featuring some of the coolest developers you’ll ever meet :)

Ready to try QuickNode?

============================================================================================

丁成 2022-01-04

在加密术语中，铸造是指使用高级加密技术对数字文件进行标记的过程。 代币化是一个过程，它成功地将数字商品/收藏品转换为可以存储在在线钱包中的代币。

到 2022 年，NFT 市场在一年内从 3.5 亿美元成倍增长到 230 亿美元的市场。

铸造 NFT 的特殊之处在于，用于标记数字商品的密码学在区块链中注册了所有者的姓名，因此，认可其对整个网络的排他性。

这种唯一标识允许用户独立拥有一个项目，而不必与任何其他人共享。 从长远来看，它还消除了复制伪造品的机会，以保持项目的稀有商完好无损。

**如何在 OpenSea 上铸造 NFT**

这是有关如何在最大的在线 NFT 交易市场 OpenSea 上成功创建 NFT 的分步指南。

首先，在 OpenSea 注册以开始铸造。 并将钱包地址链接到 OpenSea 以存储铸造的 NFT。

将钱包链接到 OpenSea 后，访问 OpenSea 仪表板并选择“我的收藏”。

单击“我的收藏”后，单击“创建”按钮开始铸造过程。

稍后，用户会发现创建 NFT 需要填写的不同部分，例如 URL、描述以及与类别和您的社交网络和 ID 相关的信息。

保存在“收藏”类别中制作的 NFT。 您可以将其保存或出售给出价最高的人。

=================================================================================

开放平台的免费铸币：

**OpenSea**

首先进入 OpenSea 官网：<https://opensea.io/>，并点击右上方的「Creat」

在编辑详细信息页面，你还可以添加相关网站链接、设置版税分成比例、设置可用代币种类等。

**Mintable**

首先打开 Mintable 官网：<https://mintable.app/>，点击「Mint an item」

律动注：以上两个平台「免费铸造NFT」其实是指先将作品数据托管在其服务器中，待作品售出再将其铸造成NFT的方法。请注意存储数据潜在风险，如数据丢失、损坏等。

如何制作 NFT

现在，制作 NFT 的最简单方法就是前往 Opensea、Rarible 或 Mintible 等平台，并按照他们的分步指南在他们的平台上进行部署即可。

你可以 100% 走这条路，但是你可能会被平台绑定。你无法实现无限的定制，或者真正利用 NFT 的任何优势。

如果你想用代码来实现，学习一些扎实的知识，并有能力以无限的创造力创造出一些东西，那么请继续阅读！

**如何进行无限定制的 NFT**

首先介绍NFT Brownie Mix。这是一个包含大量样板代码的工作仓库。

======================================================================================

发布于 2021-11-15 12:05

***02*NFT（非同质化代币）的特征**1.标准化NFT具有一些标准化功能，包括所有权、转让等。所有非同质化代币都有这些功能。**任何开发人员**都可以利用该功能来构建自己的NFT。2.通用性NFT是通用的，也就是说，任何想要使用NFT的应用都可以使用它。因为区块链是公开可访问的，而且每个人都可以阅读部署NFT的智能合约。3.流动性与加密资产相关的流动性市场非常庞大。人们可以根据自己的需求轻松地将它们换成现金或其他加密货币。因此，NFT具有很高的流动性。4.不变性区块链因不变性而闻名。NFT是通过智能合约实现的。这使得NFT不可变，即用户无法将其NFT更改为其他NFT。NFT的所有权将永久记录于区块链中，除非用户决定将其转让给其他用户。5.可编程NFT是通过智能合约实现的。NFT代币可被增强并包含其他复杂功能。***03*NFT（非同质化代币）的标准**1.ERC721它最初由**CryptoKitties**创造，也是代表非同质化数字资产的第一个标准。ERC721是可继承的Solidity智能合约标准。人们可以轻松地从OpenZeppelin库中继承，并将其用于编程基于NFT的项目。2.ERC1155它最早由Enjin提出。它使人们能够实现半同质化的代币。ERC1155允许拥有独特资产种类，而非拥有独特资产。这些资产与ID相连接。例如，一个ID可以有20个名为“Swords”的资产，而另一个ID可以有30个名为“Guns”的资产。3.元数据元数据用于定义单个代币的唯一特征。以加密猫为例，每只加密猫都有不同的颜色、形状、名称等。元数据通常以JSON形式表现4.链上在这种情况下，每个NFT都与各自的元数据链接，并存储在区块链中。最大的缺点是以太坊中的存储空间有限。5.链下在这种情况下，元数据存储在**中心化系统**（如AWS）或**去中心化网络**（如IPFS）中，并通过智能合约中的tokenURI参数链接到它们各自的NFT。***04*NFT（非同质化代币）的使用范围**1.收藏品NFT通过加密猫在收藏品领域首次亮相。在加密猫游戏流行之后，NFT成为Axie Infinity等许多其他收藏品领域的热门选择。2.游戏玩家花时间来获得游戏内奖励，但这些奖励无法在游戏外使用。然而基于区块链的NFT就解决了这一问题，因为其他游戏可以集成这些奖励。3.证书NFT可用于识别颁发给个人的唯一证书。通常，基于NFT的证书是不可转让的。这些证书存储在区块链中，因此无法篡改。

加密领域已涌现出了越来越多的NFT标准。它们的功能各异。NFT的实现范围仅限于开发人员的想象力。最后，**Wisdom Chain**已经升级了**合约脚本编程能力**，大家也可以在Wisdom Chain链上发行NFT代币。

=======================================================================================

**RaribleNFT:像稀有品这样的NFT项目-源码**

2021-03-13 08:25:06

:page\_with\_curl: 在IPFS上运行和上传的说明：
安装节点v12.10.0： curl -o-
https://raw.githubusercontent.com/nvm-sh/nvm/v0.37.2/install.sh | bash
nvm install 12.10.0 nvm alias default 12.10.0 重启终端
安装松露： npm i -g truffle
安装IPFS： npm i -g ipfs重新启动终端
运行IPFS节点： jsipfs daemon
获取项目目录（在新的终端窗口中）： git clone
https://github.com/dappuniversity/nft\_masterclass\_1.git
输入项目目录并安装依赖项： cd nft\_masterclass\_1 && npm i
进入松露开发者模式： truffle develop

=======================================================================================