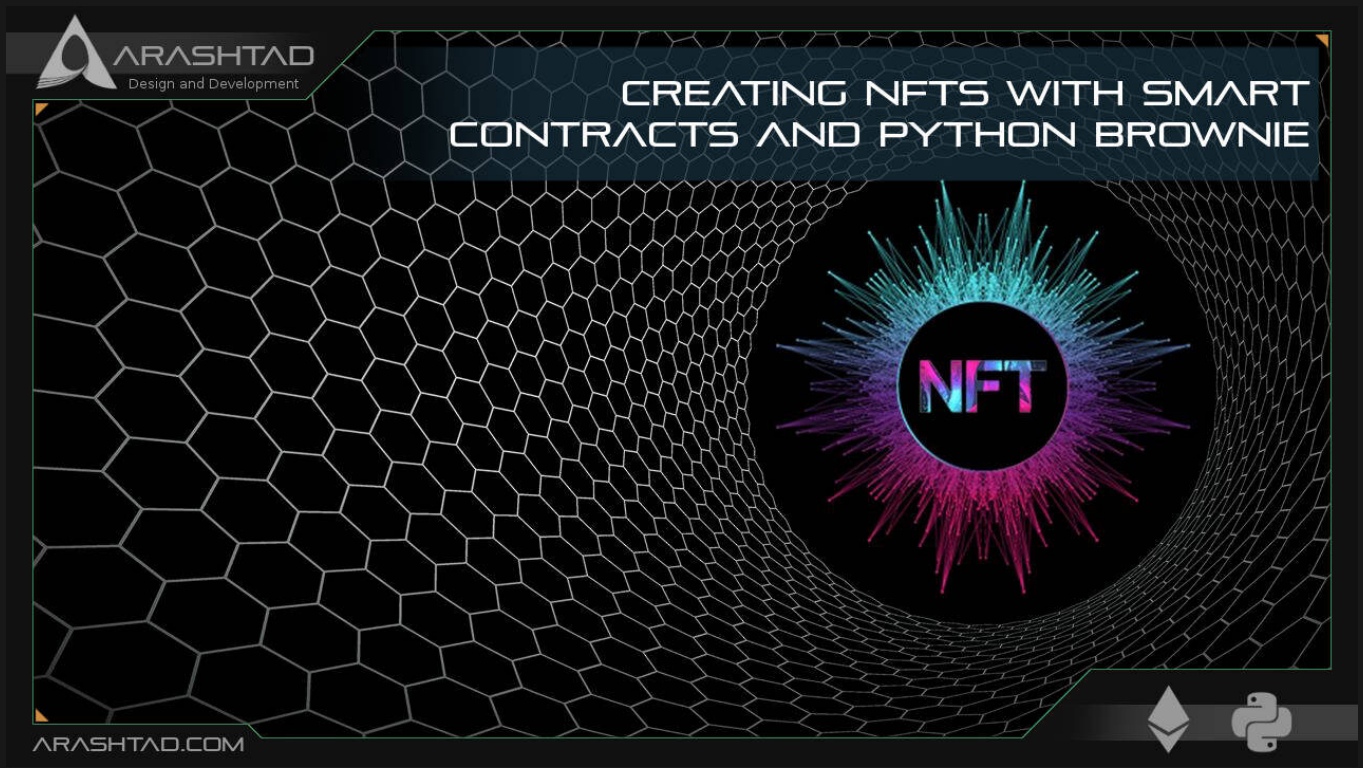
Creating NFTs with Smart Contracts and Python Brownie

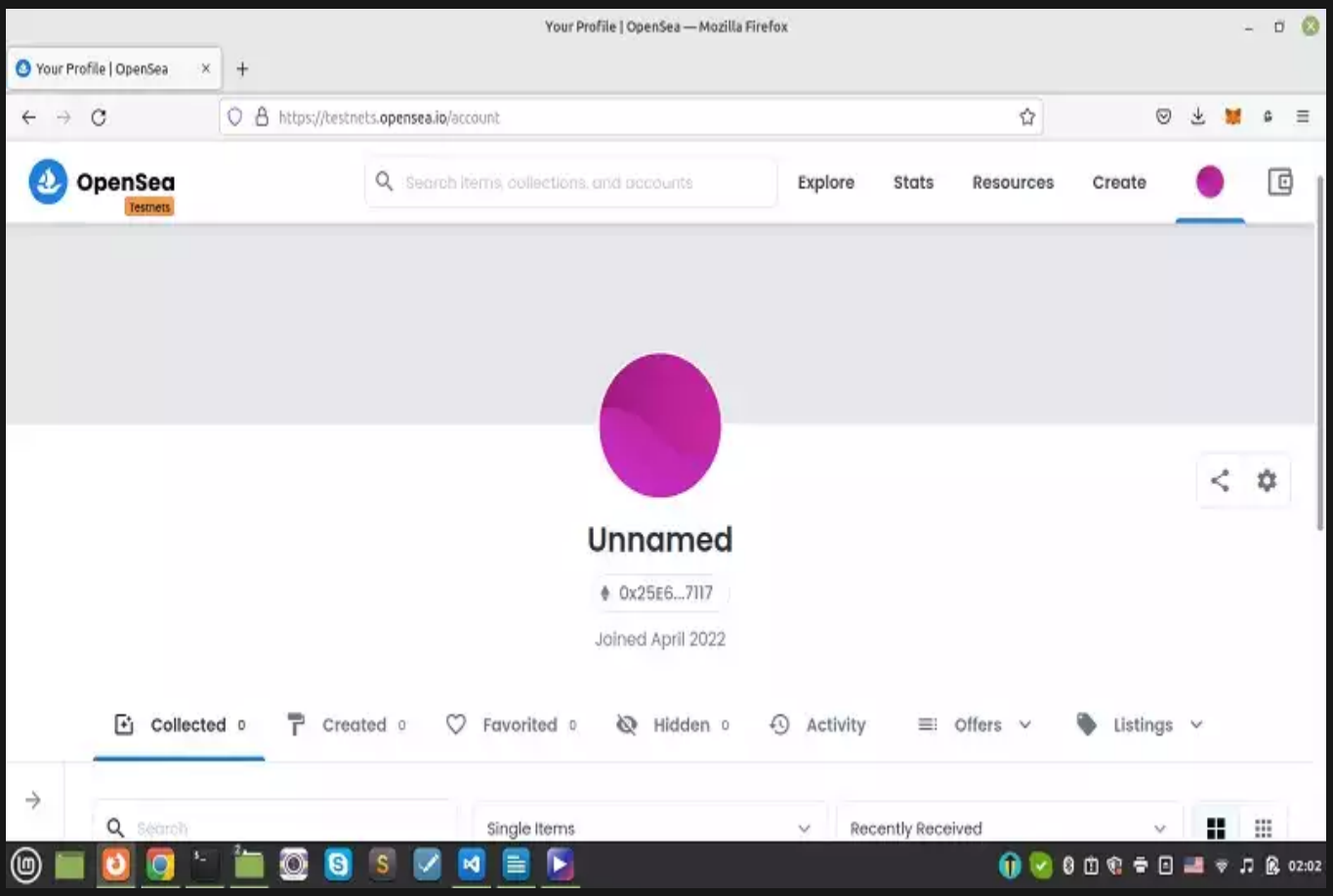
No comments



In our previous article about the Non-Fungible tokens, we explained about the methods of ERC-721. In this tutorial, we are going to use those methods to write the simple_collectible.sol smart contract and deploy it on the Rinkeby chain. In other words, we are going to create NFTs with Python Brownie tools. But before that, we also want to connect our test wallet to `https://testnets.opensea.io` in order to create an account in OpenSea which is a marketplace for NFTs.

Creating NFTs with Python Brownie

In this tutorial, we are going to use those methods to write the `simple_collectible.sol` smart contract and deploy it on the Rinkeby chain. But before that, we also want to connect our test wallet to `https://testnets.opensea.io` in order to create an account in OpenSea which is a marketplace for NFTs.



To begin writing our smart contract, we have 2 options:

1. Use Brownie not mix by typing the below command in the terminal: `brownie bake nft-mix` And we will see all the necessary files with scripts in them are created.

2. Start Brownie from scratch: `mkdir simpleNFTcd simpleNFTbrownie init` To better understand NFT smart contracts, we start from scratch. Now, let's begin writing the smart contract of our NFT:

```
// SPDX-License-Identifier: MIT
pragma solidity 0.6.6;

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

contract SimpleCollectible is ERC721 {
 uint256public tokenCounter;
 constructor ()public ERC721 ("Dogie", "DOG"){
  tokenCounter 0;
}
```

```
function createCollectible(string memory tokenURI) public
 returns (uint256){
  uint256 newTokenId = tokenCounter;
  _safeMint(msg.sender, newTokenId);
  _setTokenURI(newTokenId, tokenURI);
  tokenCounter = tokenCounter + 1;
return newTokenId;
 }
}
```

Notice that we have used `safeMint` and `setTokenURI` from `@openzeppelin/contracts/token/ERC721/ERC721.sol` that we had imported at the beginning of our contract and we should also add it to the dependencies of the `brownie-config.yaml` file. Now, let's deploy our contract using the below scripts at `deploy_and_create.py` file:

```
sample_token_uri =
"ipfs://Qmd9MCGtdVz2miNumBHDbvj8bigSgTwnr4SbyH6DNnpWdt?filename=0-PUG.json"

OPENSEA_URL = "https://testnets.opensea.io/assets/{}/{}"
def main():
    account = get_account()
    simple_collectible = SimpleCollectible.deploy({"from":account})
    tx = simple_collectible.createCollectible(sample_token_uri,{"from"
: account})
    tx.wait(1)
    print(f
"Awesome, you can{OPENSEA_URL.format(simple_collectible.address,
     simple_collectible.tokenCounter() - 1)}")
```

Also, don't forget the `helpful_scripts.py` file in the scripts folder:

```
from brownie import accounts, network, config, Contract
from web3 import Web3

LOCAL_BLOCKCHAIN_ENVIRONMENTS = ["hardhat", "development", "ganache",
"mainnet-fork"]

def get_account(index=None, id=None):
        if index:
return accounts[index]
if network.show_active() in LOCAL_BLOCKCHAIN_ENVIRONMENTS:
return accounts[0]
if id:
return accounts.load(id)
```

```
return accounts.add(config["wallets"]["from_key"])
```

The complete `brownie-config.yaml` file:

```
dependencies:
 - OpenZeppelin/openzeppelin-contracts@3.4.0
 - smartcontractkit/chainlink-brownie-contracts@1.1.1

compiler:
 solc:
  remappings:
   - '@openzeppelin=OpenZeppelin/openzeppelin-contracts@3.4.0'
   - '@chainlink=smartcontractkit/chainlink-brownie-contracts@1.1.1'
dotenv: .env
wallets:
 from_key: ${PRIVATE_KEY}
networks:
 development:
  keyhash:'0x2ed0feb3e7fd2022120aa84fab1945545a9f2ffc9076fd6156
fa96eaff4c1311'
  fee100000000000000000
 rinkeby:
  vrf_coordinator: '0xb3dCcb4Cf7a26f6cf6B120Cf5A875B7BBc655B'
  link_token: '0x01BE23585060835E02B77ef475b0Cc51aA1e0709'
  keyhash:'0x2ed0feb3e7fd2022120aa84fab1945545a9f2ffc9076fd6156
fa96eaff4c1311'
  fee100000000000000000 # 0.1
```

And our `.env` file:

```
export PRIVATE_KEY=' Paste your private key here'
export WEB3_INFURA_PROJECT_ID='INFURA ID'
export ETHERSCAN_TOKEN=' Your API Key goes here'
```

Now that we have set everything up, it is time to compile our project: `brownie compile` Result:

```
Brownie v1.18.1 - Python development framework for EthereumCompiling
contracts... Solc version: 0.6.6 Optimizer: Enabled Runs: 200 EVM
Version: Istanbul Generating build data... -
OpenZeppelin/openzeppelin-contracts@3.4.0/ERC165 -
OpenZeppelin/openzeppelin-contracts@3.4.0/IERC165 -
OpenZeppelin/openzeppelin-contracts@3.4.0/SafeMath -
OpenZeppelin/openzeppelin-contracts@3.4.0/ERC721 -
```

```
OpenZeppelin/openzeppelin-contracts@3.4.0/IERC721 -
OpenZeppelin/openzeppelin-contracts@3.4.0/IERC721Enumerable -
OpenZeppelin/openzeppelin-contracts@3.4.0/IERC721Metadata -
OpenZeppelin/openzeppelin-contracts@3.4.0/IERC721Receiver -
OpenZeppelin/openzeppelin-contracts@3.4.0/Address -
OpenZeppelin/openzeppelin-contracts@3.4.0/Context -
OpenZeppelin/openzeppelin-contracts@3.4.0/EnumerableMap -
OpenZeppelin/openzeppelin-contracts@3.4.0/EnumerableSet -
OpenZeppelin/openzeppelin-contracts@3.4.0/Strings - SimpleCollectible
Project has been compiled. Build artifacts saved at /home/mohamad/NFT-
Simple/build/contracts
```

And deploy our NFT smart contract:

```
brownie run scripts/deploy_and_create --network rinkeby
```

Result:

```
Brownie v1.18.1 - Python development framework for EthereumNftSimpleProject is the active project.Running 'scripts/deploy_and_create.py::main'...
Transaction sent: 0x4e328bf153e05a1fc06fd4060ca447ede90e7f8337e737c16508674cd6c7a951 Gas price: 1.000000061 gwei Gas limit: 2017295 Nonce: 67
SimpleCollectible.constructor confirmed Block: 10499268 Gas used: 1833905 (90.91%) SimpleCollectible deployed at: 0x657191536F5C1ec2EcfA1bD9bD4e14Ca8047F7bcTransaction sent: 0xfba6f759c7619d09a1339c975c51073846d2409c473c4f65889870cd9eae48a2 Gas price: 1.000000061 gwei Gas limit: 279109 Nonce: 68
SimpleCollectible.createCollectible confirmed Block: 10499269 Gas used: 253736 (90.91%)SimpleCollectible.createCollectible confirmed Block: 10499269 Gas used: 253736 (90.91%)Awesome, you can view your NFT at https://testnets.opensea.io/assets/0x657191536F5C1ec2EcfA1bD9bD4e14Ca8047F7bc
```

As you can see, we have successfully created our NFT and we can see it using the link given in the terminal.

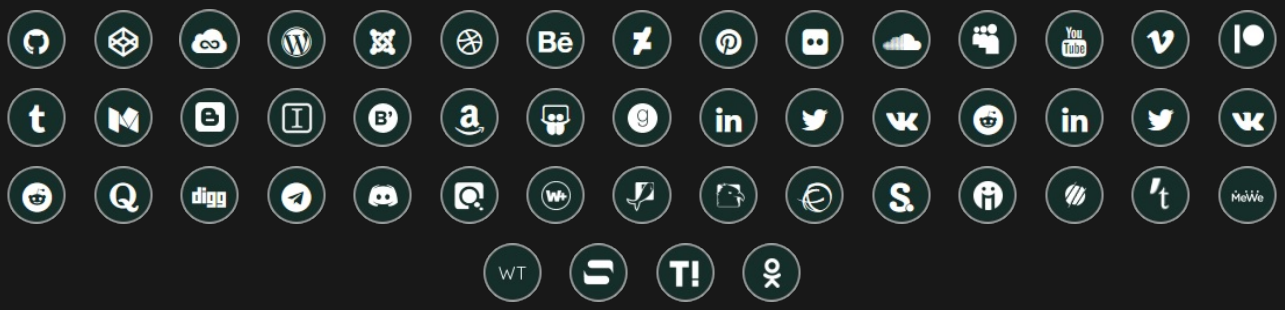
Wrapping Up

In this tutorial, we have managed to write the different parts of the simplest NFT project, from the smart contract to the deployments and dependencies, configurations, environment variables, and so on. In the end, we deployed the contract on the Rinkeby test network.

Join Arashtad Community

Follow Arashtad on Social Media

We provide variety of content, products, services, tools, tutorials, etc. Each social profile according to its features and purpose can cover only one or few parts of our updates. We can not upload our videos on SoundCloud or provide our eBooks on Youtube. So, for not missing any high quality original content that we provide on various social networks, make sure you follow us on as many social networks as you're active in. You can find out Arashtad's profiles on different social media services.



Get Even Closer!

Did you know that only one universal Arashtad account makes you able to log into all Arashtad network at once? Creating an Arashtad account is free. Why not to try it? Also, we have regular updates on our newsletter and feed entries. Use all these benefitial free features to get more involved with the community and enjoy the many products, services, tools, tutorials, etc. that we provide frequently.

SIGN UP | NEWSLETTER | RSS FEED