Developing an NFT Smart Contract on Blockchain

Prerequisites

You may create an NFT from scratch using the following platforms to start the project.

Openzzeplin Platform

OpenZeppelin is a popular open-source framework for writing secure and scalable smart contracts in the Solidity programming language. It provides a library of reusable and secure contract components that developers can use to build their own decentralized applications (dApps) on the Ethereum blockchain.

https://wizard.openzeppelin.com/#

Opensea Developer Platform

The OpenSea API helps developers build new experiences using NFTs and their marketplace data. They provide a set of endpoints that enable you to fetch ERC721 and ERC1155 token metadata as well as other core elements of their marketplace, including events, collection, listings, offers, and more.

https://docs.opensea.io/docs/deploy-an-nft-contract

Truffle Suite

Download **ganache** from Truffle Suite.

https://archive.trufflesuite.com/ganache/

Ganache is a personal blockchain for rapid Ethereum and Filecoin distributed application development. You can use Ganache across the entire development cycle; enabling you to develop, deploy, and test your dApps in a safe and deterministic environment.

Remix Platform

Remix is an online smart contract IDE

https://remix.ethereum.org/

Hardhat

Hardhat is an ethereum development platform for professionals

https://hardhat.org/

Alchemy

Alchemy Blockchain is a powerful platform that simplifies and enhances the development of blockchain applications. It provides developers with the tools and infrastructure to build, scale, and deploy blockchain solutions efficiently.

https://www.alchemy.com/

Alchemy Faucet provides free ETH for testing on Sepolia Textnet

https://www.alchemy.com/faucets/ethereum-sepolia

Infura

Infura provides the tools and infrastructure that allow developers to easily take their blockchain application from testing to scaled deployment - with simple, reliable access to Ethereum and IPFS.

https://app.infura.io/

Deployment Steps:

Install Dependencies:

Make sure you have **Node.js** and **npm** installed.

https://nodejs.org/en/download/package-manager

Use powershell to install Nodejs

```
# installs fnm (Fast Node Manager)
winget install Schniz.fnm

# download and install Node.js
fnm use --install-if-missing 20

# verifies the right Node.js version is in the environment
node -v # should print `v20.14.0`

# verifies the right NPM version is in the environment
npm -v # should print `10.7.0`
```

https://docs.npmjs.com/downloading-and-installing-node-js-and-npm

```
npm install -g npm
```

Then install Truffle and OpenZeppelin:

```
npm install -g truffle
npm install @openzeppelin/contracts
```

Initialize Truffle Project

```
truffle init
```

Creating the Smart Contract

Create a smart contract in a file named `nft.sol` (any name you like) under the `contracts` directory in your Truffle project.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract MyNFT is ERC721, Ownable {
    uint256 private _tokenIdCounter;
    constructor() ERC721("MyNFT", "MNFT") Ownable(msg.sender) {}

    function _baseURI() internal view virtual override returns (string memory)
{
        return "https://api.mynft.com/metadata/";
    }
    function mint(address to) public onlyOwner {
        uint256 tokenId = _tokenIdCounter;
        _tokenIdCounter += 1;
        _safeMint(to, tokenId);
    }
}
```

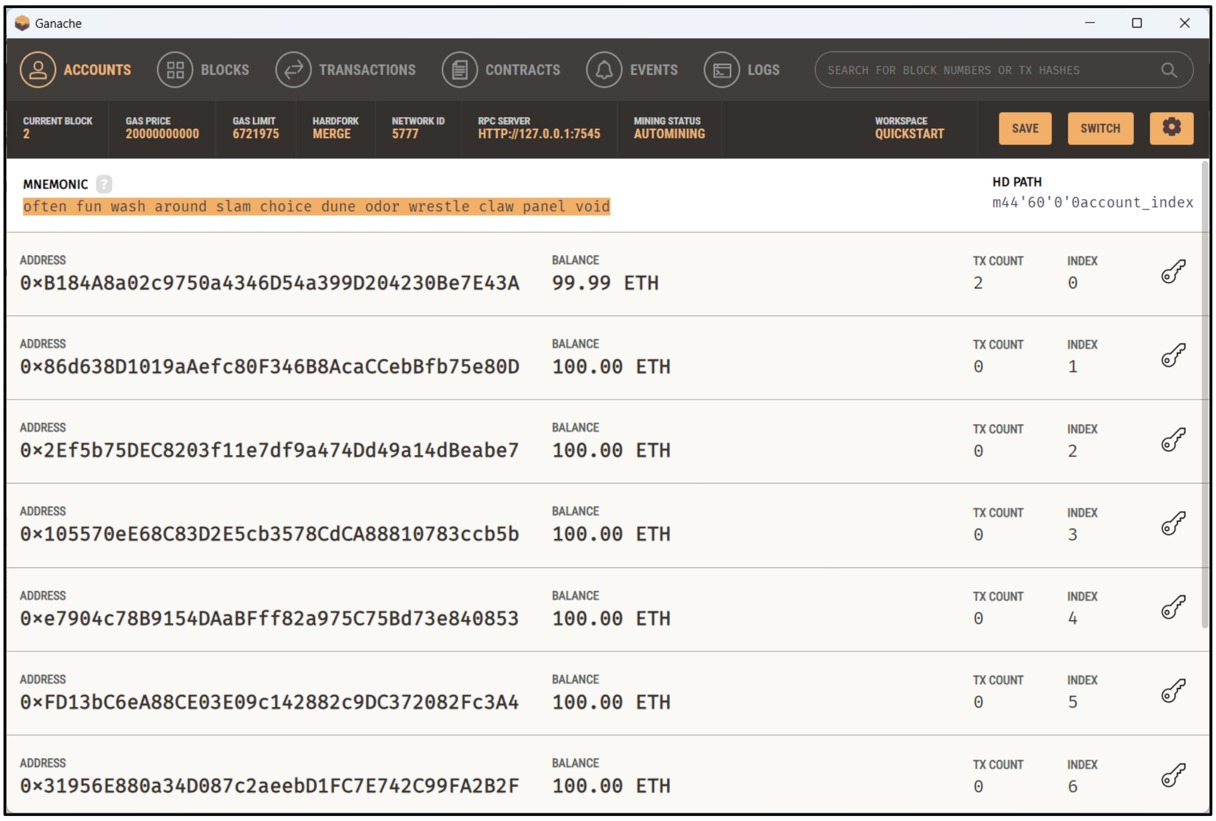
use version 0.8.19 because 0.8.20 has some issues

Configure the **truffle_config.js** file as follows:

```
Use this file to configure your truffle project. It's seeded with some
 * common settings for different networks and features like migrations,
 * compilation, and testing. Uncomment the ones you need or modify
 * them to suit your project as necessary.
 *
 * More information about configuration can be found at:
 *
 * https://trufflesuite.com/docs/truffle/reference/configuration
 *
 * Hands-off deployment with Infura
 * --------------------------------
 *
 * Do you have a complex application that requires lots of transactions to
deploy?
 * Use this approach to make deployment a breeze 🏖️:
```

```
 *

 * Infura deployment needs a wallet provider (like @truffle/hdwallet-provider)

 * to sign transactions before they're sent to a remote public node.

 * Infura accounts are available for free at 🔍: https://infura.io/register

 *

 * You'll need a mnemonic - the twelve word phrase the wallet uses to generate

 * public/private key pairs. You can store your secrets 🤐 in a .env file.

 * In your project root, run `$ npm install dotenv`.

 * Create .env (which should be .gitignored) and declare your MNEMONIC

 * and Infura PROJECT_ID variables inside.
 * For example, your .env file will have the following structure:
 * MNEMONIC = <Your 12 phrase mnemonic>
 * PROJECT_ID = <Your Infura project id>
 *
 * Deployment with Truffle Dashboard (Recommended for best security practice)
 * --------------------------------------------------------------------------
 *
 * Are you concerned about security and minimizing rekt status 🤔?
 * Use this method for best security:
 *
 * Truffle Dashboard lets you review transactions in detail, and leverages
 * MetaMask for signing, so there's no need to copy-paste your mnemonic.
 * More details can be found at 🔎:
 *
 * https://trufflesuite.com/docs/truffle/getting-started/using-the-truffle-dashboard/
 */


// require('dotenv').config();
// const { MNEMONIC, PROJECT_ID } = process.env;


// const HDWalletProvider = require('@truffle/hdwallet-provider');



module.exports = {

  /**

   * Networks define how you connect to your ethereum client and let you set
the

   * defaults web3 uses to send transactions. If you don't specify one truffle
   * will spin up a managed Ganache instance for you on port 9545 when you
   * run `develop` or `test`. You can ask a truffle command to use a specific
   * network from the command line, e.g
   *
   * $ truffle test --network <network-name>
   */

  networks: {
    // Useful for testing. The `development` name is special - truffle uses it
by default
    // if it's defined here and no other network is specified at the command
line.
    // You should run a client (like ganache, geth, or parity) in a separate
terminal
    // tab if you use this network and you must also set the `host`, `port`
and `network_id`
    // options below to some value.
    //
     ganache: {
      host: "127.0.0.1",     // Localhost (default: none)
     port: 7545,            // Standard Ethereum port (default: none)
     network_id: "*",       // Any network (default: none)
    },
    //
    // An additional network, but with some advanced options…
    // advanced: {
    //   port: 8777,             // Custom port
    //   network_id: 1342,       // Custom network
    //   gas: 8500000,           // Gas sent with each transaction (default: ~6700000)
    //   gasPrice: 20000000000,  // 20 gwei (in wei) (default: 100 gwei)
    //   from: <address>,        // Account to send transactions from (default: accounts[0])
    //   websocket: true         // Enable EventEmitter interface for web3 (default: false)
    // },
    //
    // Useful for deploying to a public network.
    // Note: It's important to wrap the provider as a function to ensure truffle uses a new provider every time.
    // goerli: {
    //   provider: () => new HDWalletProvider(MNEMONIC, `https://goerli.infura.io/v3/${PROJECT_ID}`),
    //   network_id: 5,       // Goerli's id

    //   confirmations: 2,    // # of confirmations to wait between deployments. (default: 0)

    //   timeoutBlocks: 200,  // # of blocks before a deployment times out (minimum/default: 50)

    //   skipDryRun: true     // Skip dry run before migrations? (default: false for public nets )

    // },

    //

    // Useful for private networks

    // private: {

    //   provider: () => new HDWalletProvider(MNEMONIC, `https://network.io`),

    //   network_id: 2111,   // This network is yours, in the cloud.
    //   production: true    // Treats this network as if it was a public net.
(default: false)


    // }
  },

  // Set default mocha options here, use special reporters, etc.
  mocha: {
    // timeout: 100000
  },

  // Configure your compilers
  compilers: {
    solc: {
      version: "0.8.19" // Fetch exact version from solc-bin (default:
truffle's version)
      // docker: true,        // Use "0.5.1" you've installed locally with
docker (default: false)
      // settings: {          // See the solidity docs for advice about
optimization and evmVersion
      //  optimizer: {
      //    enabled: false,
      //    runs: 200
      //  },
      //  evmVersion: "byzantium"
      // }
    }
  }



  // Truffle DB is currently disabled by default; to enable it, change
enabled:


  // false to enabled: true. The default storage location can also be


  // overridden by specifying the adapter settings, as shown in the commented
code below.


  //


  // NOTE: It is not possible to migrate your contracts to truffle DB and you
should


  // make a backup of your artifacts to a safe location before enabling this
feature.


  //
  // After you backed up your artifacts you can utilize db by running migrate
as follows:

  // $ truffle migrate --reset --compile-all

  //

  // db: {

  //   enabled: false,

  //   host: "127.0.0.1",

  //   adapter: {

  //     name: "indexeddb",

  //     settings: {

  //       directory: ".db"

  //     }

  //   }

  // }

};
```

Launch Ganache



CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
2	20000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	QUICKSTART

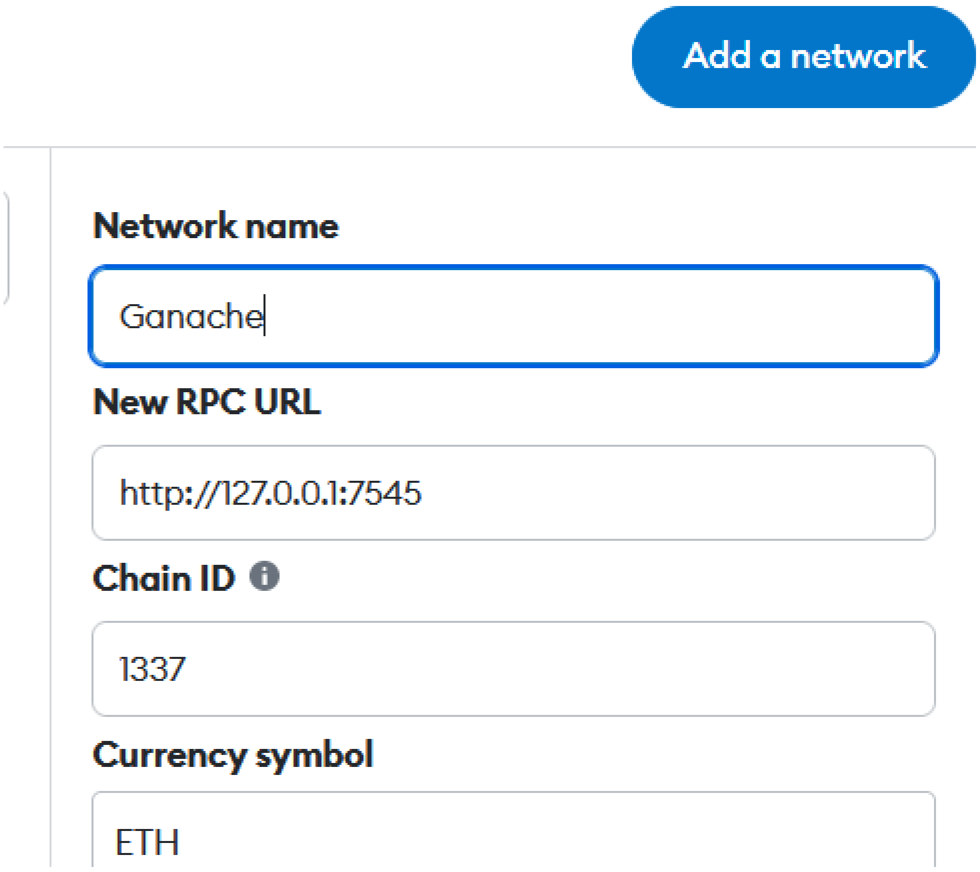
MNEMONIC
often fun wash around slam choice dune odor wrestle claw panel void

HD PATH
m44'60'0'0account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×B184A8a02c9750a4346D54a399D204230Be7E43A	99.99 ETH	2	0
0×86d638D1019aAefc80F346B8AcaCCebBfb75e80D	100.00 ETH	0	1
0×2Ef5b75DEC8203f11e7df9a474Dd49a14dBeabe7	100.00 ETH	0	2
0×105570eE68C83D2E5cb3578CdCA88810783ccb5b	100.00 ETH	0	3
0×e7904c78B9154DAaBFff82a975C75Bd73e840853	100.00 ETH	0	4
0×FD13bC6eA88CE03E09c142882c9DC372082Fc3A4	100.00 ETH	0	5
0×31956E880a34D087c2aeebD1FC7E742C99FA2B2F	100.00 ETH	0	6

Create a new Metamask account using the recovery phase here.

Set up the Ganache network



Network name

Ganache

New RPC URL

http://127.0.0.1:7545

Chain ID

1337

Currency symbol

ETH

Compile the Contract:

```
truffle compile
```

Deploy the Contract:

Create a migration script under the `migrations` directory (e.g., `2_deploy_contracts.js`):

```
const MyNFT = artifacts.require("MyNFT");

module.exports = function (deployer) {

  deployer.deploy(MyNFT);

};
```

Deploy the contract using the following command

```
truffle migrate --network ganache
```

The output

```
Compiling your contracts...

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

> Compiling .\contracts\MyNFT.sol

> Compiling .\contracts\MyNFT.sol

> Artifacts written to C:\Users\Pavilion\OneDrive\Documents\Blockchain\Blockchain
Courses\MSU\NFT\NFT App\build\contracts

> Compiled successfully using:

   - solc: 0.8.19+commit.7dd6d404.Emscripten.clang




Starting migrations...

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

> Network name:    'ganache'

> Network id:      5777

> Block gas limit: 6721975 (0x6691b7)
```

```
2_deploy_contracts.js

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


   Replacing 'MyNFT'

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

   > transaction hash:
0x1b2e8d3f5b98210419b81da6d2ccc504cf16d575fb6c38625cdcb06a6a90319c

   > Blocks: 0            Seconds: 0

   > contract address:    0x11153E7A7c1b673dD6Ca8E86aa29D6a0000228A5

   > block number:        1

   > block timestamp:     1717206971

   > account:             0xB184A8a02c9750a4346D54a399D204230Be7E43A

   > balance:             99.993367331875

   > gas used:            1965235 (0x1dfcb3)

   > gas price:           3.375 gwei

   > value sent:          0 ETH

   > total cost:          0.006632668125 ETH
   > Saving artifacts

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

   > Total cost:      0.006632668125 ETH



Summary

=======

> Total deployments:   1

> Final cost:          0.006632668125 ETH
```

You will notice ETH being spent to deploy the contract on Metamask and on Ganache.