FITE 1010 Introduction to financial technologies

Lab 3: Blockchain: Smart Contract and NFT

Deadline: November 14 23:59

Overview

Non-fungible token, NFT, is one of the most popular application of smart contract last year. At a very high level, most NFTs are part of the Ethereum blockchain. Ethereum's smart contracts store extra information that makes them work differently from, say, an ETH coin. NFTs can really be anything digital (such as drawings, music), but a lot of the current excitement is around using the technology to sell digital art.

In this lab, we will build our own smart contract and mint an NFT. You will be able to see the NFT in an NFT marketplace called OpenSea, and you can transfer it to your friend.

Objectives

After completing this lab, you will be able to:

- Understand what is a crypto wallet.
- Learn how to write, compile and deploy a smart contract.
- Understand what is NFT and ERC-721 contract
- Create an NFT and trade it in OpenSea.

Preparation: Virtual Machine

The preparation part is only for Window users. Mac users can directly use your terminal to finish this lab.

Please download the software below first:

1. Virtual Box 7.0.2 (https://www.virtualbox.org/wiki/Downloads)
2. Ubuntu 22.04.1 LTS (https://ubuntu.com/download/desktop).

Prepare a virtual machine in your own computer.

1. Install Virtual Box 7.0.2
2. Open Virtual Box → "New" button → Name: Your student ID. Select the type as "Linux." Type your name and select the Ubuntu (64 bit) option. Set memory = 4096MB(This is machine dependent. You can set it larger if it is allowed in your PC) in the expert mode. Next, keep the default setting (VDI, dynamic allocation) → "Continue." → "Create."
3. Select the name of the machine that you just created, and then click the green "Start" arrow. Open the Ubuntu .iso file that you downloaded

Install Ubuntu in the VM.

1. Install Ubuntu.
2. Choose "Normal Installation". Click "Continue."
3. Make sure "Erase disk and install Ubuntu" is selected. Click "Continue."
4. Follow instructions to select the region, language preferences, and your username and password. The computer will now install the Ubuntu Operating System. This process could take up to 10 minutes, but requires no effort on your part.

After successfully install Ubuntu, maybe you will find that the window is small. Please refer to another tutorial "how to resize VM's resolusion".

Exercise 1 Install Digital Wallet for Crypto – MetaMask

A crypto wallet is a place where you can securely keep your crypto. There are many different types of crypto wallets, but the most popular ones are hosted wallets, non-custodial wallets, and hardware wallets. In this lab, we will use a "non-custodial wallet" which doesn't rely on a third party — or a "custodian" — to keep your crypto safe. We will use a popular wallet called "MetaMask".

Metamask

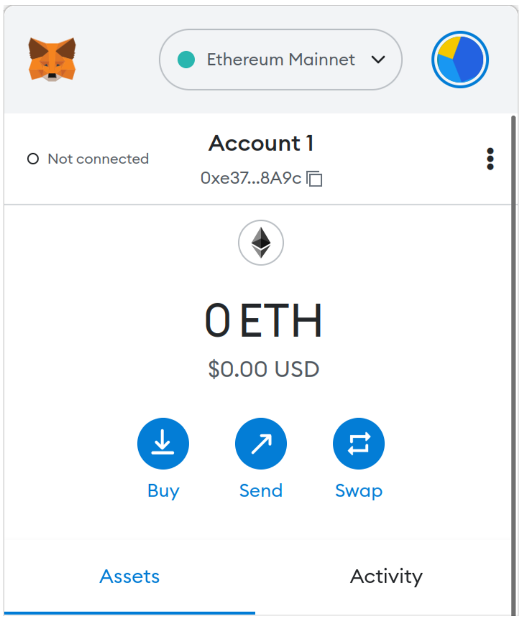
In your virtual machine (used in previous labs), open the browser Firefox.

Download the digital wallet app called "MetaMask" at https://metamask.io

Click the download button and select "Install MetaMask for Firefox".

After it is installed, create a wallet in MetaMask. You can select your own password. Choose “Remind me later” for secret recovery phrase.

After installation, you should see it in your browser (next to the address bar):



The number below “Account 1” is your Ethereum mainnet address generated by the wallet.

Question 1. (5 mark) Capture Screenshot 1 for your MetaMask wallet and write down your address.

Question 2. (10 mark) Your private key is securely stored in the wallet and it will not be displayed on the screen. How is your private key related to your address?

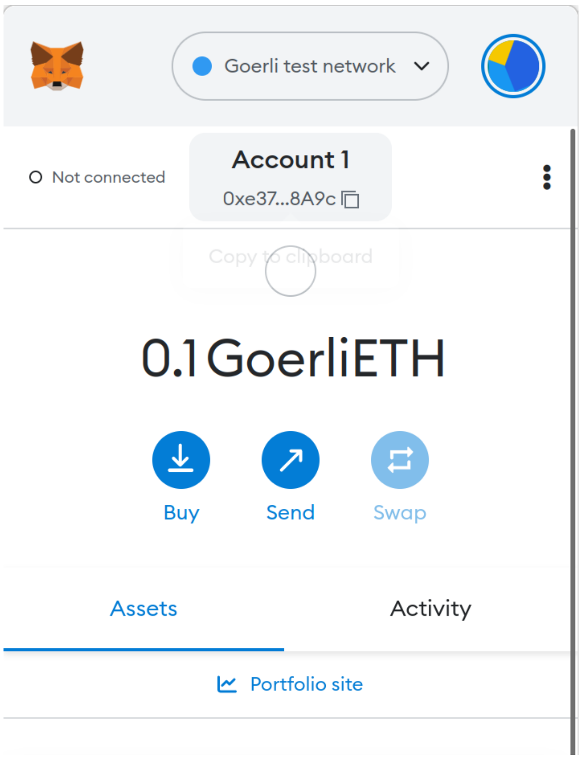
Question 3. (15 mark) Currently you have no money in your Ethereum mainnet address. Suggest three possible ways to get some ETH.

Goerli Test Network

Press the “Ethereum Mainnet” button → Show/hide test network → Turn on the “Show test network”.

Switch to “Goerli Test Network” and copy your address.

Get some ETH for your testnet address from https://goerlifaucet.com/ or any other website that you can find. You will need at least 0.02ETH to complete the lab. You should follow the instructions from those websites to get your ETH. You will be able to see it in your MetaMask afterward.



Question 4. (10 mark) Why do we need to use a test network to test our smart contract in this lab, instead of using the Ethereum mainnet?

Exercise 2 Writing a Smart Contract for NFT

A Non-Fungible Token (NFT) is used to identify something or someone in a unique way. This type of Token is perfect to be used on platforms that offer collectible items, access keys, lottery tickets, numbered seats for concerts and sports matches, etc.

The ERC-721 introduces a standard for NFT, in other words, this type of Token is unique and can have different value than another Token from the same Smart Contract, maybe due to its age, rarity or even something else like its visual. All NFTs have a uint256 variable called *tokenId*, so for any ERC-721 Contract, the pair *contract address*, *uint256 tokenId* must be globally unique.

In this exercise, we will create an NFT token of our own using the ERC-721 smart contract.

Remix IDE

Remix is an IDE (Integrated Development Environment) officially provided by Ethereum. It contains the full functionality of the compiler, executing smart contracts, publishing contracts, and more. No installation is required, you can just open https://remix.ethereum.org/ with your browser.

Pre-requisites for installation

Install NodeJS and npm if they are not installed. Below we use command for linux. Open the terminal and run:

```
sudo apt install nodejs
```

```
sudo apt install npm
```

If you use Mac, you can run:

```
brew install nodejs
```

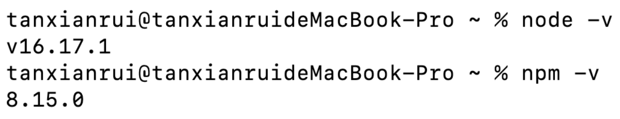
```
brew install npm
```

You can check if you have installed them successfully by:

```
node -v
```

```
npm -v
```

It is successful if you can see the version number:



```
tanxianrui@tanxianruideMacBook-Pro ~ % node -v
v16.17.1
tanxianrui@tanxianruideMacBook-Pro ~ % npm -v
8.15.0
```

OpenZeppelin

OpenZeppelin is an open-source smart contract repository with many smart contract templates, including ERC-721 smart contracts. By using the OpenZeppelin's ERC-721 template, we just need to extend our own specific functions.

Install OpenZeppelin by:

```
npm init -y
```

```
sudo npm install @openzeppelin/contracts
```

Remixd

Since Remix is an online IDE, we need to make some changes to link it to local storage (in case you need to save the file and update it later). We need to install an add-on called “remixd”.

Install Remixd by:

```
sudo npm install -g @remix-project/remixd
```

You can verify if you have installed it successfully.

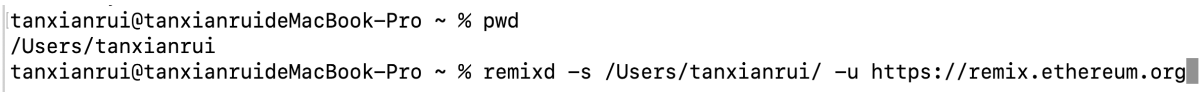
```
remixd -v
```

Create a folder “mynft” and link it.

```
mkdir mynft
```

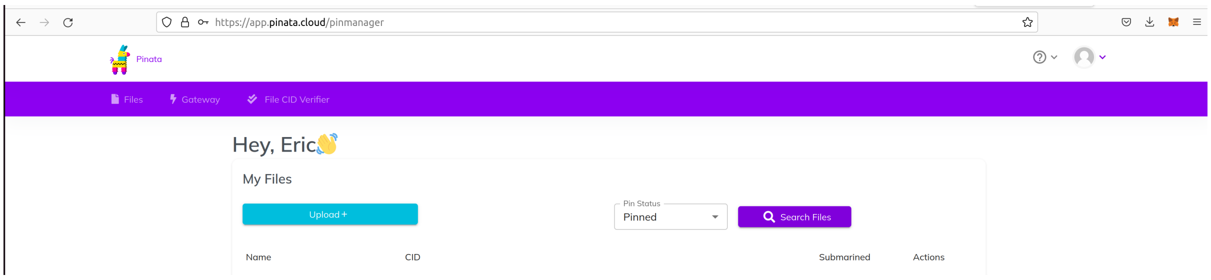
```
sudo remixd -s /(your VM name)/ -u https://remix.ethereum.org/
```

If you don’t know what VM name is, you can type **pwd** on your desktop and see it.



```
tanxianrui@tanxianruideMacBook-Pro ~ % pwd
/Users/tanxianrui
tanxianrui@tanxianruideMacBook-Pro ~ % remixd -s /Users/tanxianrui/ -u https://remix.ethereum.org
```

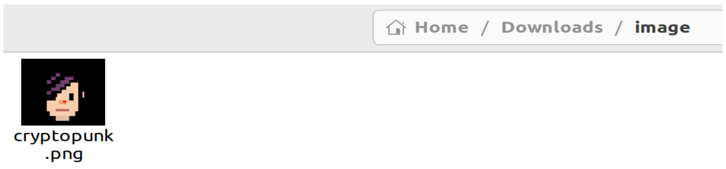
Creating metadata file with Pinata



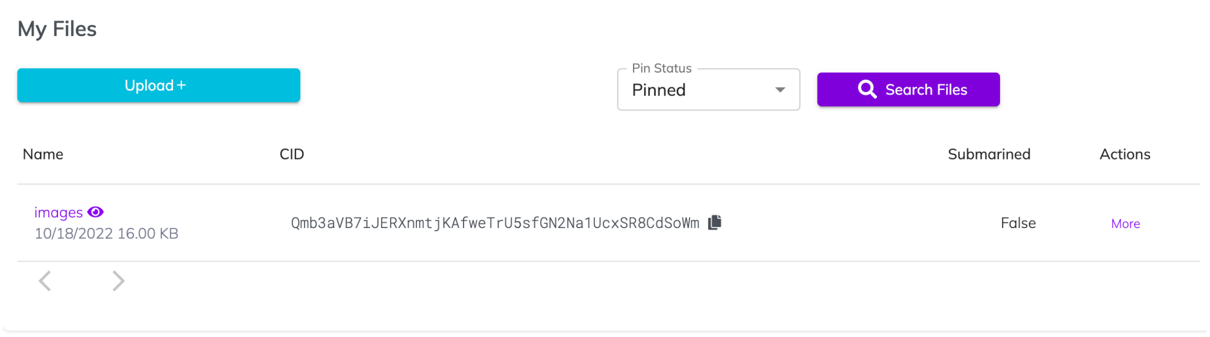
We will upload an image and the metadata for our NFT token in the pinata.

Draw/download your own small icon for your future NFT. You can use whatever image you like.

Create a folder locally and put the jpg there.



Then upload this folder to pinata.

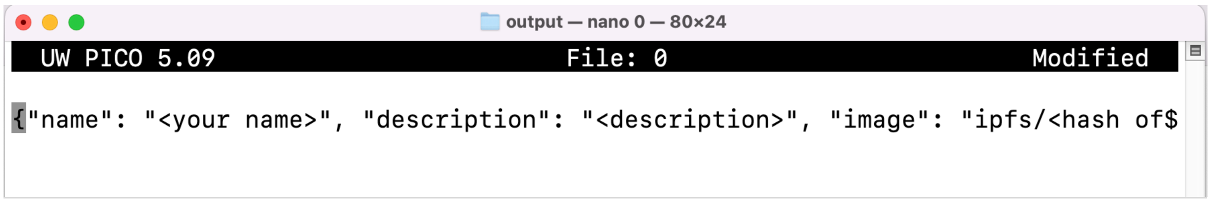


Name	CID	Submarined	Actions
images 10/18/2022 16.00 KB	Qmb3aVB7iJERXnmtjKAfweTrU5sfGN2Na1UcxSR8CdSoWm	False	More

Now create another folder called “json”, and create a JSON file called “0” in this folder, with the content(You can create the file by opening a terminal in this folder and typing “nano 0”):

```
{"name": "<your name>", "description": "<description>", "image": "ipfs/<hash of
the photo is put here>", "attributes": [{"trait_type": "Background", "value":
"black"}, {"trait_type": "skin", "value": "white"}, {"trait_type": "Eyes",
"value": "black"}, {"trait_type": "hair color", "value": "purple"}]}
```

ps: Try to place the content above in one line(remove the newline at the end of each line after you copy and paste)! Otherwise the photo may not be displayed later. For example, it should be:

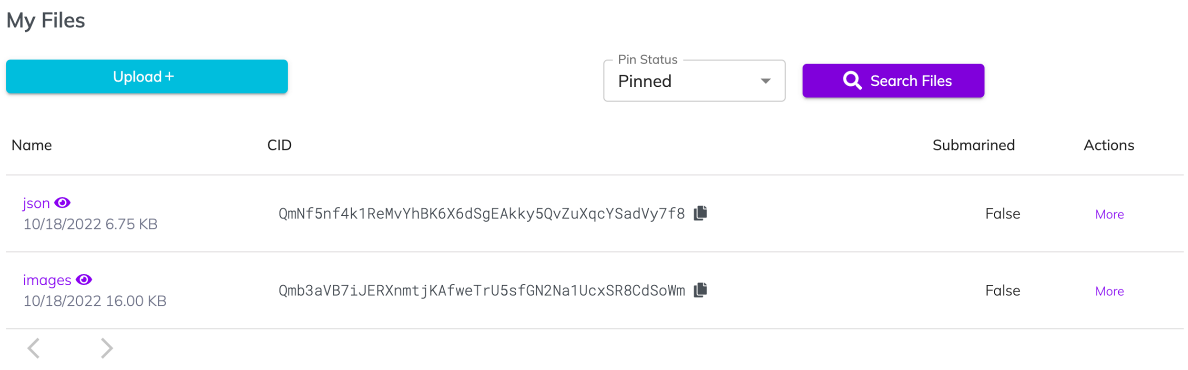


```
output — nano 0 — 80×24
  UW PICO 5.09                        File: 0                          Modified

{"name": "<your name>", "description": "<description>", "image": "ipfs/<hash of$
```

Feel free to change/add the `attributes` to match the description your icon. This file will be the Metadata file of your future NFT token. The `hash of the photo` can be found in pinata. If you image is png format, it should look like(assume the name of your image is 0): Qm***/0.png

Save the file “0”, and upload the “json” folder to pinata.

Copy the ipfs hash(CID) of the “json” folder (not the file “0” or the image).



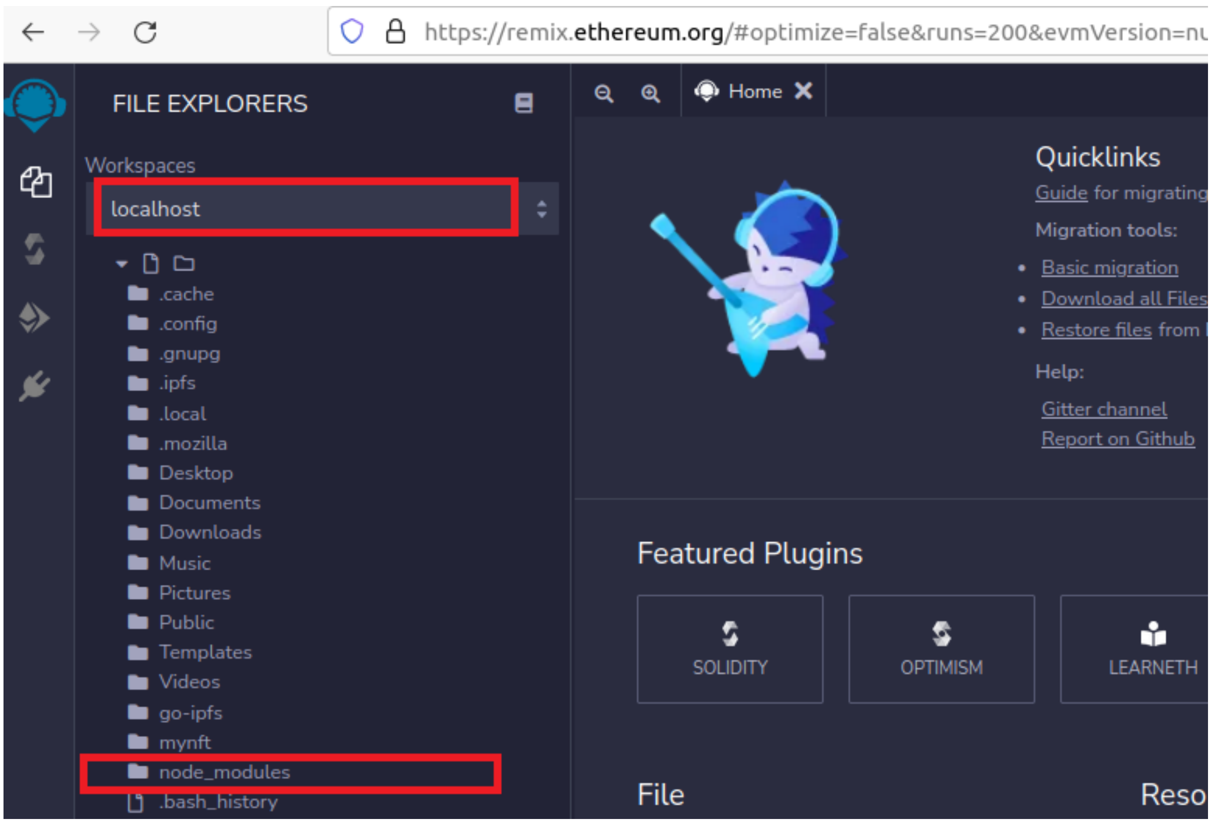
My Files

Name	CID	Submarined	Actions
json 10/18/2022 6.75 KB	QmNf5nf4k1ReMvYhBK6X6dSgEAkky5QvZuXqcYSadVy7f8	False	More
images 10/18/2022 16.00 KB	Qmb3aVB7iJERXnmtjKAfweTrU5sfGN2Na1UcxSR8CdSoWm	False	More

Writing Smart Contract

Open Remix IDE in Firefox: https://remix.ethereum.org/

In the button “Workspaces”, switch to “localhost”. The folder should contain node_modules installed by OpenZeppelin.



Right click “mynft” and create “FITE1010.sol”. Then open the “FITE1010.sol”.

Now you can create your first ERC721 contract by using the template below:

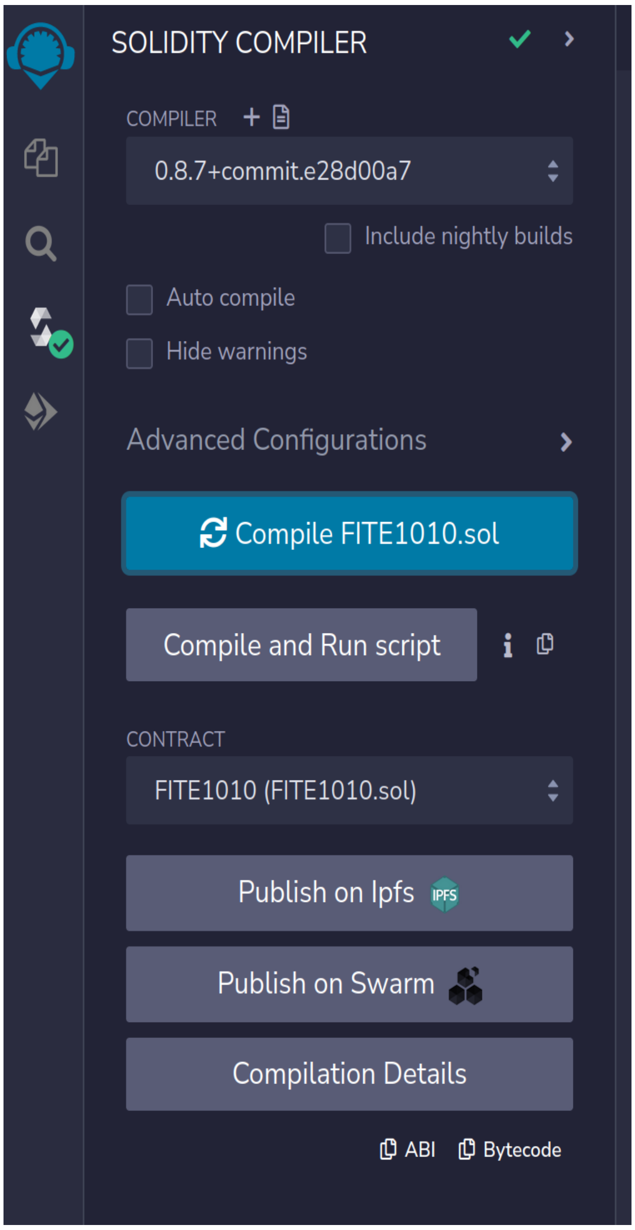
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

import
"../node_modules/@openzeppelin/contracts/token/ERC721/presets/ERC721PresetMint
erPauserAutoId.sol";

contract FITE1010 is ERC721PresetMinterPauserAutoId {

    constructor()
    ERC721PresetMinterPauserAutoId("<describe yourself here>", "<your name>", "
https://gateway.pinata.cloud/ipfs/<your ipfs folder address>/")
    {}

    // This allows the minter to update the tokenURI after it's been minted.
    // To disable this, delete this function.
    function setTokenURI(uint256 tokenId, string memory tokenURI) public {
        require(hasRole(MINTER_ROLE, _msgSender()), "web3 CLI: must have
minter role to update tokenURI");

        setTokenURI(tokenId, tokenURI);
    }
}
```

The above code is built using OpenZeppelin's ERC-721 contract. We simply set our NFT name, token code and token MetaData URI by the function input to ERC721PresetMinterPauserAutoId.

Question 5. (10 mark) What is the main difference between the ERC-721 contract and the standard ERC-20 contract?

Exercise 3 Compiling and Deploying a Smart Contract for NFT

After editing, you can switch to the Compile panel to compile. Be careful to choose the correct compiler version, the compiler version must be the same as the OpenZeppelin contract you are using, otherwise a compilation error will occur.



SOLIDITY COMPILER

COMPILER: 0.8.7+commit.e28d00a7

Include nightly builds

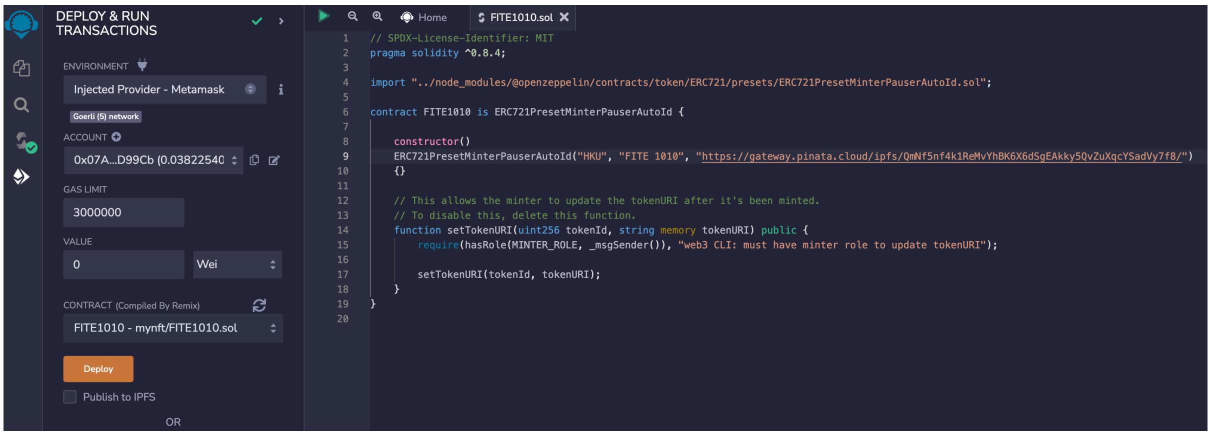
Auto compile

Hide warnings

CONTRACT: FITE1010 (FITE1010.sol)

Once the compilation is complete, it is ready to be deployed. Switch to the Deploy panel. At this point, we need to select [Injected Provider - metamask] in the Environment in the upper left corner, and MetaMask will pop up to link. If the MetaMask does not pop up, refresh the remix and try again.

There are many testnets for Ethereum, such as Ropsten, Rinkeby, Goerli, etc. To be able to test on OpenSea (an NFT trading platform) in the next excercise, we chose the Goerli testnet. To use the test network, remember to select Goerli Test Network in MetaMask.


DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - Metamask

Goerli (5) network

ACCOUNT: 0x07A...D99Cb (0.03822540

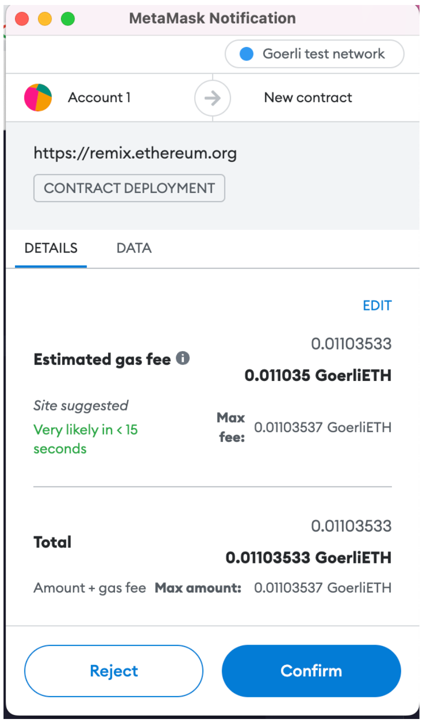
GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT (Compiled By Remix): FITE1010 - mynft/FITE1010.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

import "../node_modules/@openzeppelin/contracts/token/ERC721/presets/ERC721PresetMinterPauserAutoId.sol";

contract FITE1010 is ERC721PresetMinterPauserAutoId {

    constructor()
    ERC721PresetMinterPauserAutoId("HKU", "FITE 1010", "https://gateway.pinata.cloud/ipfs/QmNf5nf4k1ReMvYhBK6X6dSgEAkky5QvZuXqcYSadVy7f8/")
    {}

    // This allows the minter to update the tokenURI after it's been minted.
    // To disable this, delete this function.
    function setTokenURI(uint256 tokenId, string memory tokenURI) public {
        require(hasRole(MINTER_ROLE, _msgSender()), "web3 CLI: must have minter role to update tokenURI");

        setTokenURI(tokenId, tokenURI);
    }
}
```

After everything is ready, you can press [Deploy] to start publishing your smart contract, and MetaMask will pop up for payment confirmation.


MetaMask Notification

Goerli test network

Account 1 → New contract

https://remix.ethereum.org

CONTRACT DEPLOYMENT

DETAILS | DATA

EDIT

Estimated gas fee: 0.01103533 — **0.011035 GoerliETH**

Site suggested — Very likely in < 15 seconds — **Max fee:** 0.01103537 GoerliETH

Total: 0.01103533 — **0.01103533 GoerliETH**

Amount + gas fee — **Max amount:** 0.01103537 GoerliETH

After the deployment is completed, Remix will display the corresponding Transaction Hash. The corresponding Transaction can be seen on https://goerli.etherscan.io, and the ERC721 smart contract address we created can also be seen in it.

Question 6. (10 mark) Capture the screenshots of your deployed contract:

(1) in remix,

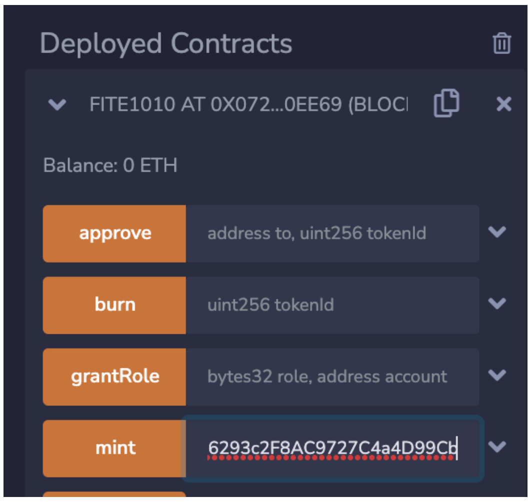
(2) in https://goerli.etherscan.io.

Question 7. (10 mark) Is there any change in your MetaMask? Why?

Exercise 4 Minting an NFT

After the contract is deployed, the next step is to Mint NFT.

Back to Remix, you can see that the smart contract we just released already exists in Deployed Contracts. After clicking on it, you can see that there is a function of mint in it. Now we can call this function to Mint NFT.



Enter your Goerli address in “address to” after the “mint” function, and this address will become the owner of the newly created NFT.

Question 8. (10 mark) Is there any change in your MetaMask? Why?

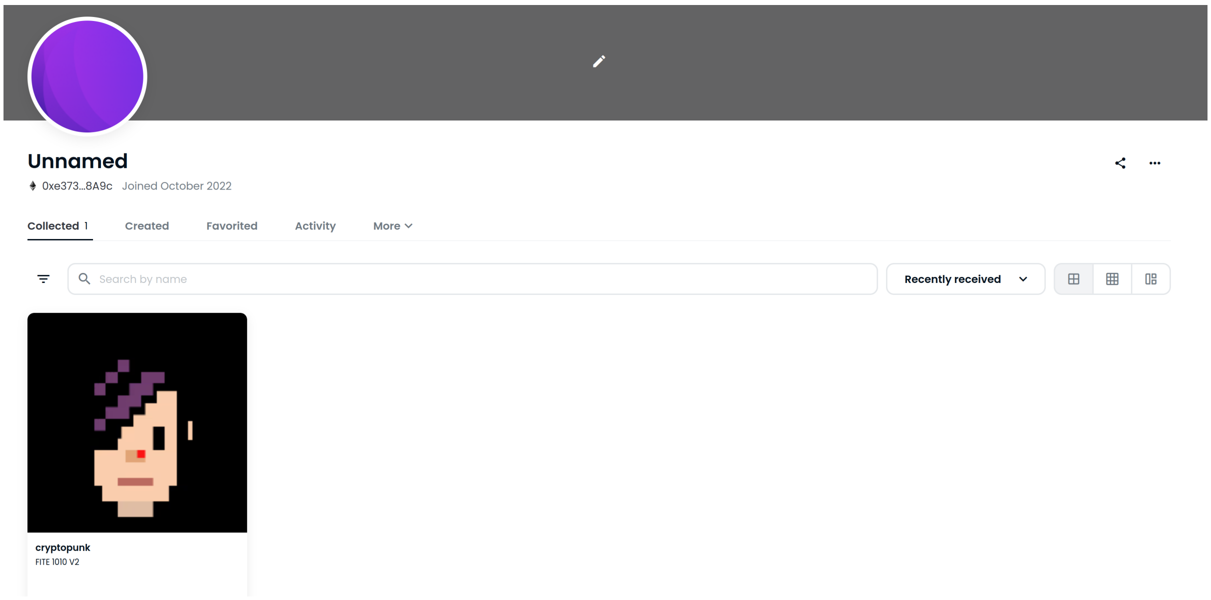
Exercise 5 Viewing your NFT in OpenSea

OpenSea (https://opensea.io) is a popular marketplace for NFT. But now we are issuing NFTs on the testnet, so we use the following URL instead:

https://testnets.opensea.io/

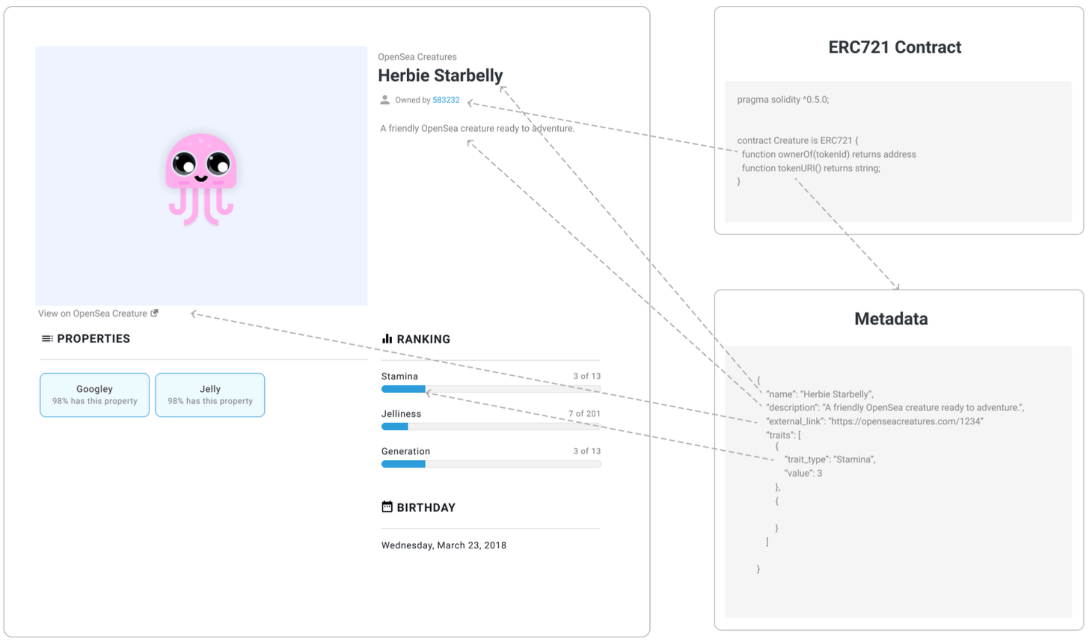
After entering, register an account with your own MetaMask address and you can see the NFT you just minted. MetaMask will automatically pop up when you register an account.

When you click your profile, you will see your NFT there.



The image and other attributes corresponding to the NFT are determined by the Metadata of the NFT. In order for the NFT to display the image and other attributes in OpenSea, the Metadata of the NFT needs to be specified.

According to the ERC-721 Metadata standard on OpenSea, OpenSea will call the tokenURI function in our smart contract and pass in the tokenID, and this function needs to return an HTTP or IPFS URL, which must return data in JSON format, such as in the JSON file in IPFS in Exercise 2, and this JSON defines various properties of our NFT.



```
pragma solidity ^0.5.0;


contract Creature is ERC721 {
  function ownerOf(tokenId) returns address
  function tokenURI() returns string;
}
```

```
{
  "name": "Herbie Starbelly",
  "description": "A friendly OpenSea creature ready to adventure.",
  "external_link": "https://openseacreatures.com/1234"
  "traits": [
    {
      "trait_type": "Stamina",
      "value": 3
    },
    {

    }
  ]

}
```

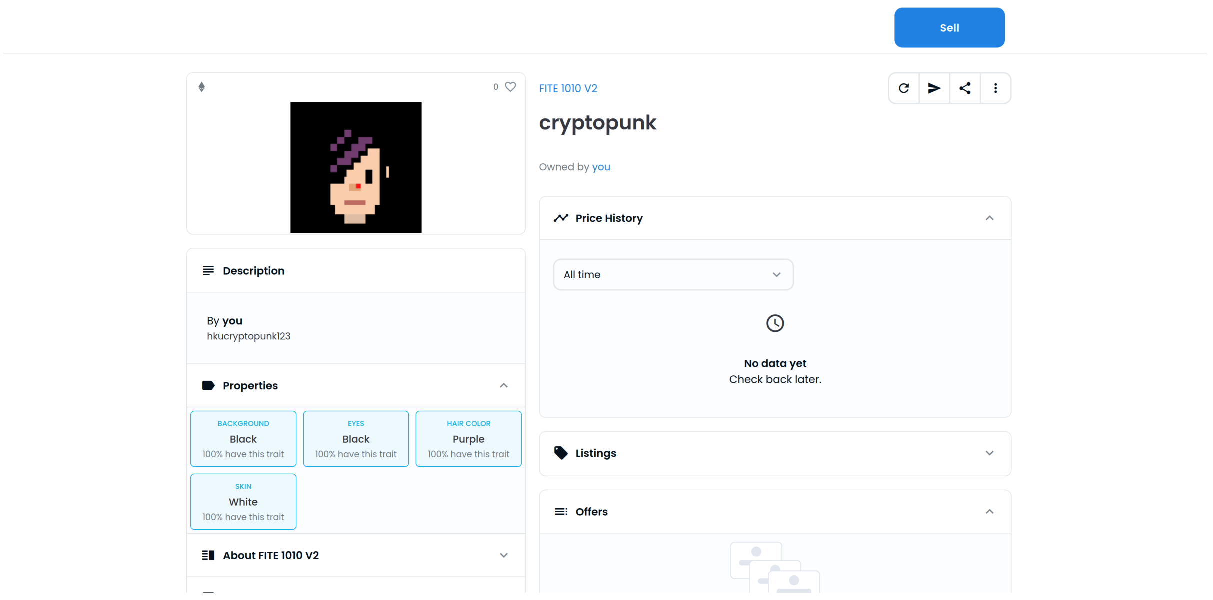
Taking the smart contract we wrote above as an example, notice that we have the following in the constructor:

```
ERC721PresetMinterPauserAutoId("<describe yourself here>",
"<your name>", "https://ipfs.io/ipfs/<your ipfs folder
address>/")
```

The base URI of our NFT is https://ipfs.io/ipfs/<your ipfs folder address>/. When you call the `tokenURI` function, our contract will return https://ipfs.io/ipfs/<your ipfs folder address>/<tokenID>. If this Metadata is set, the mint NFT will have attributes such as images.

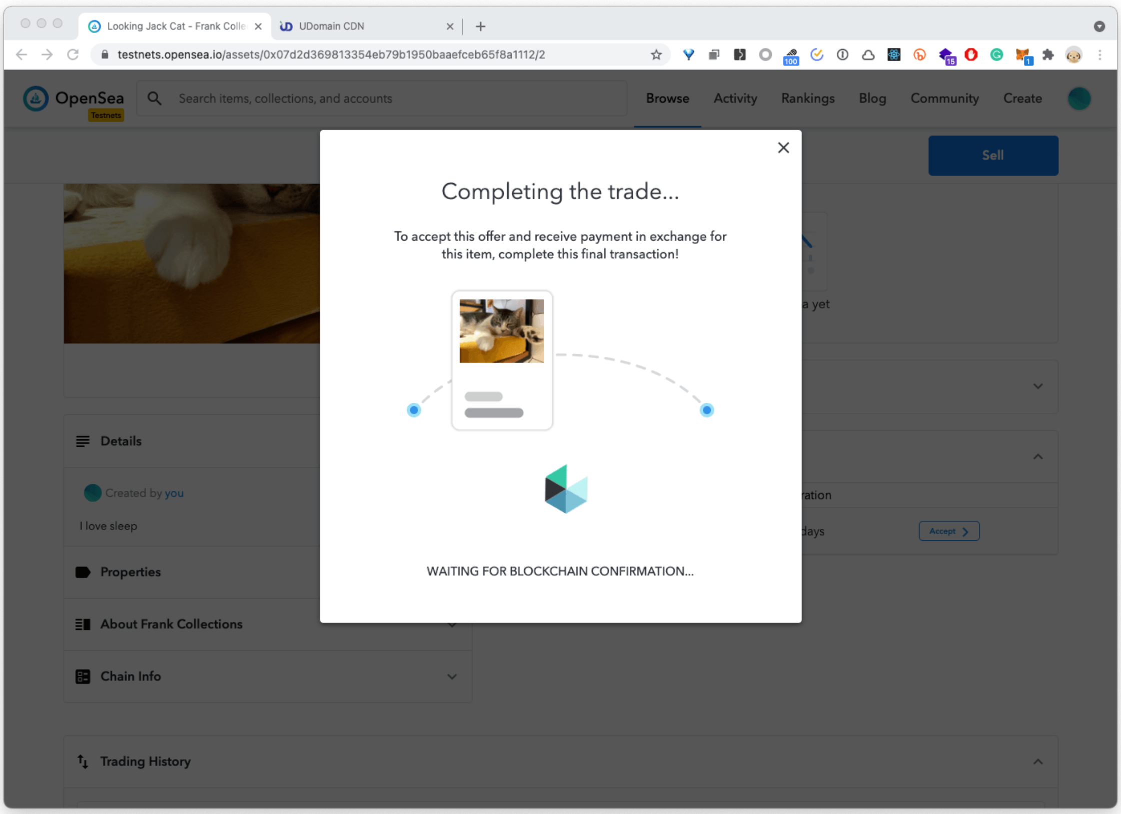
Click on the NFT you just created. You should see something like this:

Click on the reload button. You should be able to see your icon later (You may need to wait for 5-10 minutes).



Question 9. (10 mark) Capture the screenshot of your NFT icon and attributes in OpenSea.

Congratulations! You have created your own NFT. Of course, the NFT you built can also be traded on OpenSea.



Completing the trade...

To accept this offer and receive payment in exchange for this item, complete this final transaction!

WAITING FOR BLOCKCHAIN CONFIRMATION...

I love sleep

Question 10. (10 mark) Create another address in MetaMask (using the create account button in MetaMask) or ask your classmate for an address. Send the NFT token to him/her in OpenSea. Capture the screenshot of the transaction.

End of lab 4