블록체인 기술 _ 이더리움 프로젝트 보고서

항공전자정보공학부 2020124184 차정은

0. 기본 설정

```
C:₩Users₩cygnu₩Downloads₩블록체인 기술_이더리움 2주차_프로젝트 폴더_2020124184_차정은>truffle unbox polygon
This directory is non-empty...
? Proceed anyway? (Y/n)
Starting unbox...
==================

? Proceed anyway? Yes
√ Preparing to download box
√ Downloading
contracts already exists in this directory...
? Overwrite contracts? Yes
migrations already exists in this directory...
? Overwrite migrations? Yes
test already exists in this directory...
? Overwrite test? Yes
truffle-config.js already exists in this directory...
? Overwrite truffle-config.js? Yes

Unbox successful, sweet!

Commands:

  Compile Contracts: truffle compile
```

Mumbai를 이용할 것이므로 polygon에 대한 프로젝트 생성

.env

```
1  MNEMONIC="marriage message kiwi boost swing prevent cycle describe blanket seven icon tent"
2
3  INFURA_PROJECT_ID="541ad13c6caa413da3a372d4f9c76220"
```

.env 파일 생성

```
26  const path = require('path');
27
28  module.exports = {  /**
29
30    * contracts_build_directory tells Truffle where to store compiled contracts
31    */
32    contracts_build_directory: path.join(__dirname, 'build/polygon-contracts'),
33    contracts_directory: path.join(__dirname, 'contracts/polygon'),
34
35    /**
```

Truffle-config.js 수정. 이전과 컴파일 할 폴더 위치가 다르고, 빌드할 위치도 다르기 때문.

```
1   const HDWalletProvider = require('@truffle/hdwallet-provider');
2   // create a file at the root of your project and name it .env --
3   // like the mnemomic and Infura project key below. Note: .env is
4   require('dotenv').config();
5   const mnemonic = process.env["MNEMONIC"];
6   const infuraProjectId = process.env["INFURA_PROJECT_ID"];
7
8   module.exports = {
9
10    /**
11    * contracts_build_directory tells Truffle where to store compi
12    */
13    contracts_build_directory: './build/polygon-contracts',
14
15    /**
16    * contracts_directory tells Truffle where the contracts you wa
17    */
18    contracts_directory: './contracts/polygon',
42      //polygon Infura testnet
43      mumbai: {
44        provider: () => new HDWalletProvider({
45          mnemonic: {
46            phrase: mnemonic
47          },
48          providerOrUrl:
49           "https://polygon-mumbai.infura.io/v3/" + infuraProjectId
50        }),
51        network_id: 80001,
52        confirmations: 2,
53        timeoutBlocks: 200,
54        skipDryRun: true,
55        chainId: 80001
56      }
57    },
```

Truffle-config.polygon.js 수정

1. MyNFT.sol 코드

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.10;
// 컴파일에 필요한 solidity 버전

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
// openzeppelin의 ERC721, Ownable, Counters, ERC721URIStorage 라이브러리 이용

contract MyNFT is ERC721, Ownable, ERC721URIStorage {
// 스마트 컨트랙트를 정의
// ERC721 (NFT 표준), Ownable (소유권 관리), ERC721URIStorage (NFT 메타데이터 저장) 상속
    using Counters for Counters.Counter;
    //  Counters 라이브러리를 Counters.Counter로 사용
    Counters.Counter private _tokenIds;
    // token의 id를 저장할 변수 선언
    struct Item { // token 구조체
        uint256 id;
        address owner;
        uint256 price;
    }
    mapping (uint256 => Item) private _items;
    // id에 item 매핑
    constructor() ERC721("MyNFT", "MNFT") {}
    // contract의 이름과 symbol 결정
    function _burn(uint256 tokenId) internal virtual override(ERC721, ERC721URIStorage) {
        super._burn(tokenId);
    } // 상속 받은 함수를 재정의

    function supportsInterface(bytes4 interfaceId) public view virtual override(ERC721, ERC721URIStorage) returns (bool) {
        // 주어진 interface를 사용하는지 확인
        return super.supportsInterface(interfaceId);
    } // 상속 받은 함수를 재정의

    function tokenURI(uint256 tokenId) public view virtual override(ERC721, ERC721URIStorage) returns (string memory) {
        // 주어진 토큰 ID에 연결된 메타데이터 URI를 반환
        return super.tokenURI(tokenId);
    }  // 상속 받은 함수를 재정의
    function mintNFT(address recipient, string memory tokenUriParam, uint256 price) public onlyOwner returns (uint256) {
    // 새로운 NFT 토큰을 발행하고, 해당 토큰의 URI와 가격을 설정
        // 전체 토큰 수를 증가 시킴
        _tokenIds.increment();
        // 새로 생성되는 토큰의 id 가져옴
        uint256 newItemId = _tokenIds.current();
        // 새 토큰 발행
        _mint(recipient, newItemId);
        // 새토큰 URI 설정
        _setTokenURI(newItemId, tokenUriParam);
        // 토큰 정보를 매핑한 item에 저장
        _items[newItemId] = Item(newItemId, recipient, price);
        // id 반환
        return newItemId;
    }

    function buyNFT(uint256 tokenId) public payable { //payable로 선언, ETH와 동반 호출 가능
    // token을 구매하는 함수
        // 토큰이 실제로 존재하는지 확인
        require(_exists(tokenId), "Error: wrong TokenId");
        // 전송된 이더의 양이 토큰의 가격과 일치하는지 확인
        require(msg.value >= _items[tokenId].price, "Error: the ETH value sent is not correct");
        // 구매할 토큰의 현재 소유자 주소
        address previousOwner = _items[tokenId].owner;
        // : 토큰을 현재 소유자로부터 호출자에게 전송
        _transfer(previousOwner, msg.sender, tokenId);
        // 이더를 현재 소유자에게 전송
        payable(previousOwner).transfer(msg.value);
        // 토큰의 소유자를 호출자로 설정
        _items[tokenId].owner = msg.sender;
    }
```

```
73      function getItem(uint256 tokenId) public view returns (uint256 id, address owner, uint256 price) {
74      // 특정 토큰의 정보를 가져오는 함수
75          require(_exists(tokenId), "Error: wrong TokenId");
76          //입력으로 token의 id 요구
77          Item memory item = _items[tokenId];
78          return (item.id, item.owner, item.price);
79          // 정상적인 입력을 받으면 해당 token의 item들을 반환
80      }
81  }
82
```

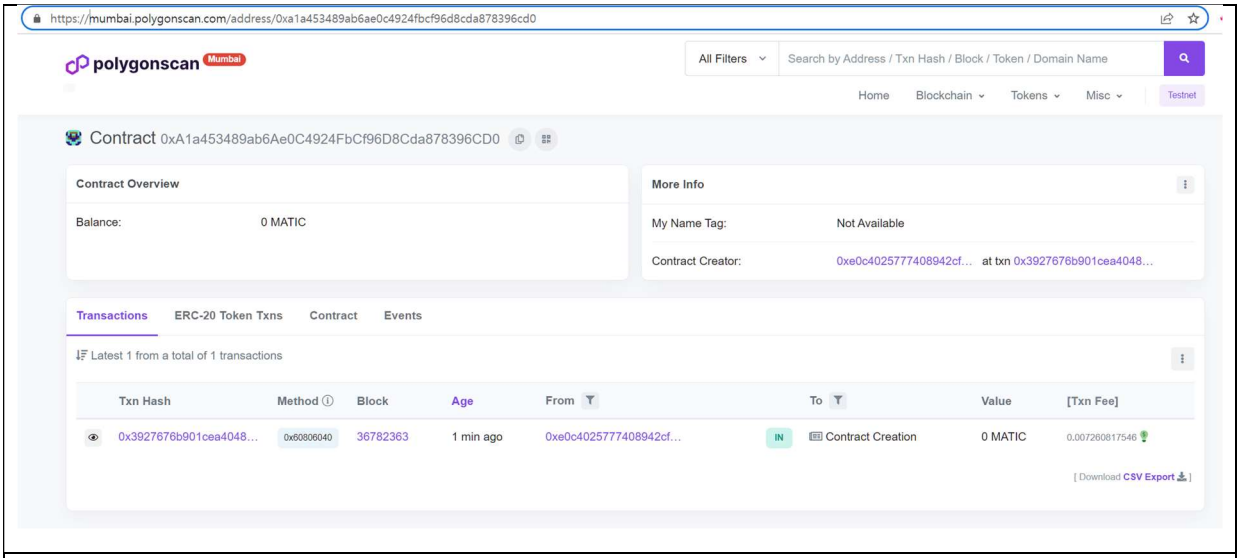
MyNFT.sol

2. 결과

```
> Artifacts written to C:\Users\cygnu\Downloads\블록체인\build\polygon-contracts
> Compiled successfully using:
   - solc: 0.8.10+commit.fc410830.Emscripten.clang


Starting migrations...
======================
> Network name:    'mumbai'
> Network id:      80001
> Block gas limit: 21176116 (0x1431f34)


2_deploy_contract.js
====================

   Deploying 'MyNFT'
   -----------------
   > transaction hash:    0x3927676b901cea40488689482ebefe6c01286fbf472d3b2c5dc0206ecf25d564
   > Blocks: 4            Seconds: 9
   > contract address:    0xA1a453489ab6Ae0C4924FbCf96D8Cda878396CD0
   > block number:        36782363
   > block timestamp:     1686625225
   > account:             0xE0C4025777408942cf322B8D8502396c5108aF7F
   > balance:             0.492739182453530768
   > gas used:            2904327 (0x2c5107)
   > gas price:           2.500000016 gwei
   > value sent:          0 ETH
   > total cost:          0.007260817546469232 ETH

   Pausing for 2 confirmations...

   ------------------------------
   > confirmation number: 1 (block: 36782367)
   > confirmation number: 3 (block: 36782369)
   > Saving artifacts
   -------------------------------------
   > Total cost:     0.007260817546469232 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.007260817546469232 ETH
```

Mumbai 배포 후 터미널 화면

https://mumbai.polygonscan.com/address/0xa1a453489ab6ae0c4924fbcf96d8cda878396cd0



배포한 CA의 주소 및 화면