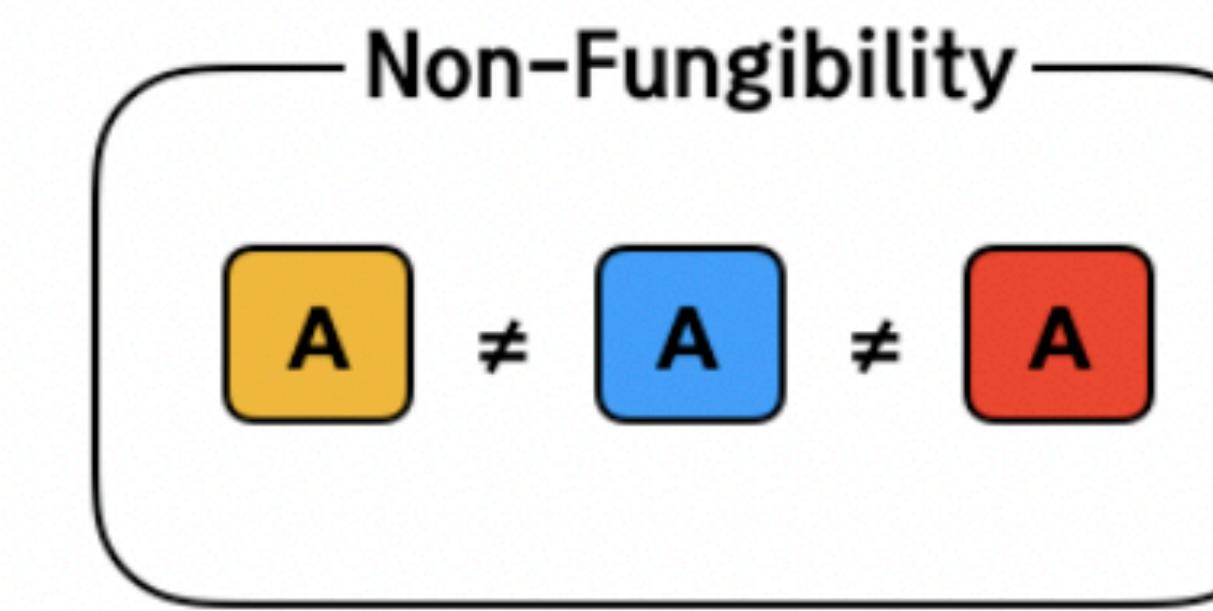
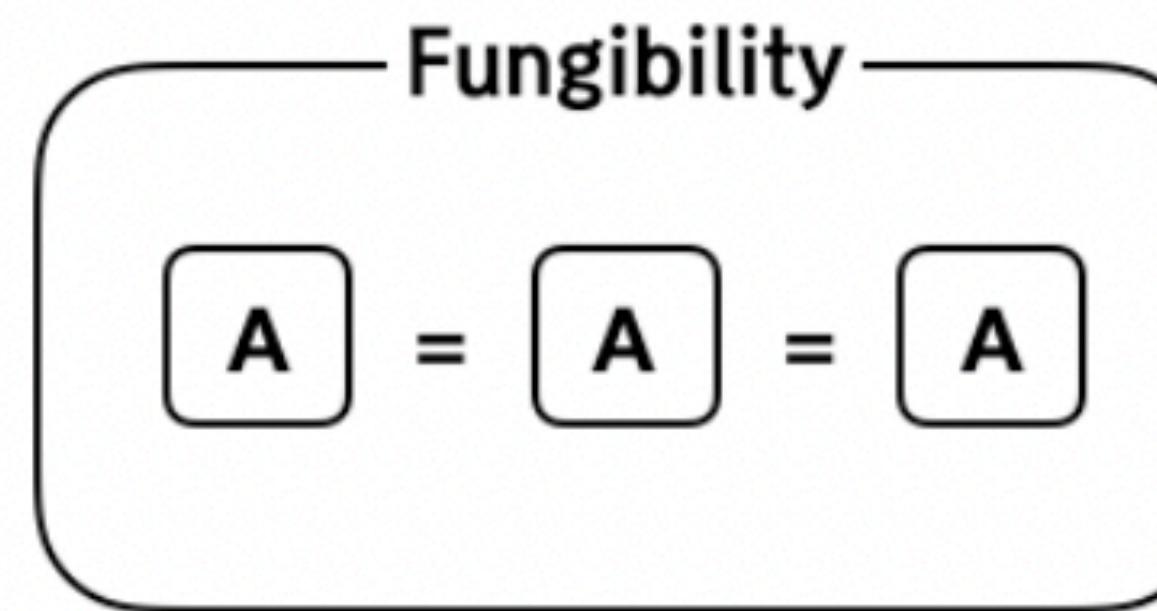


# Mint your own NFT collection

ERC721을 활용한 NFT 민팅 및 실습

# 대체 불가능한 토큰 (NFTs)

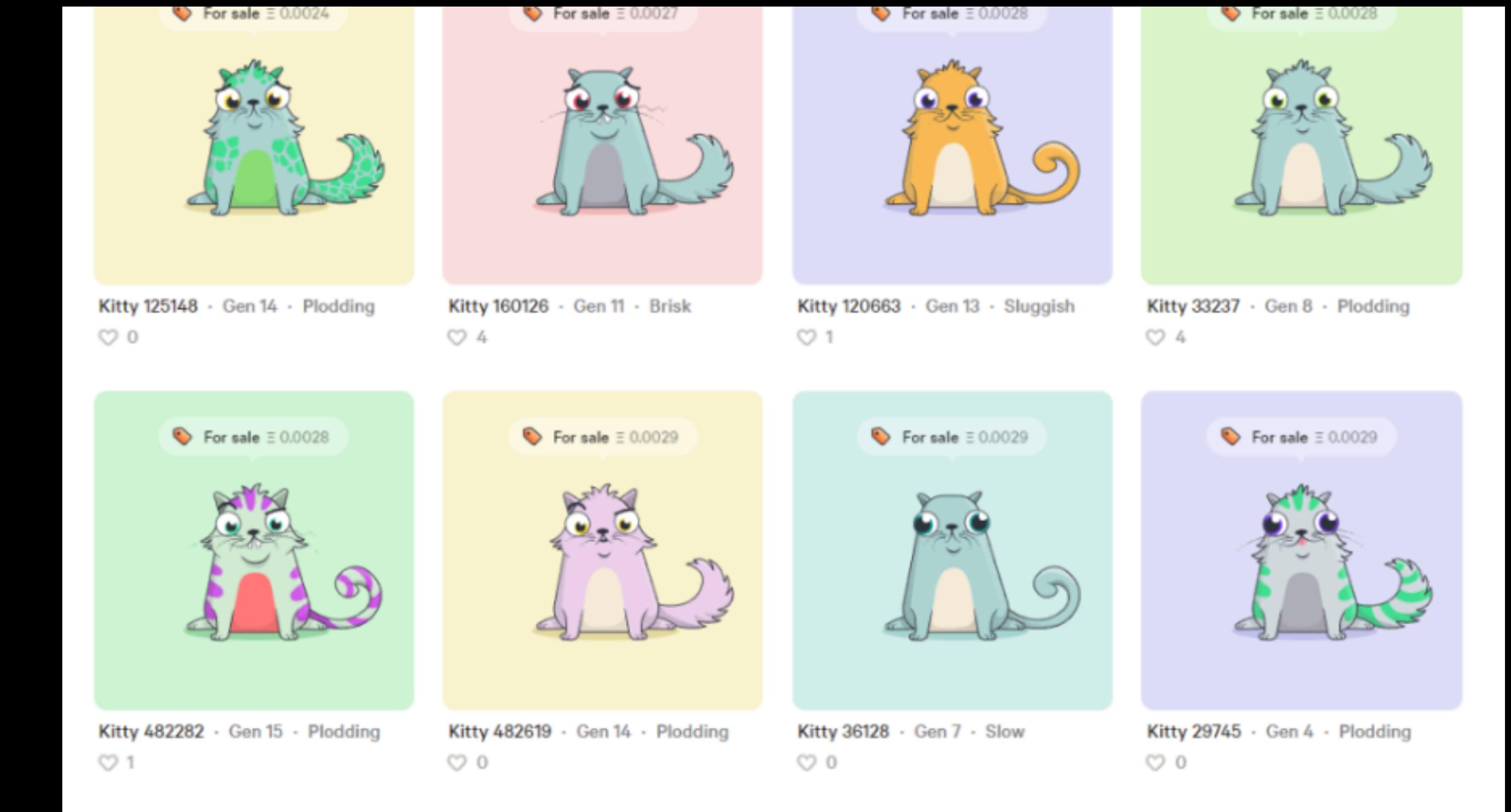
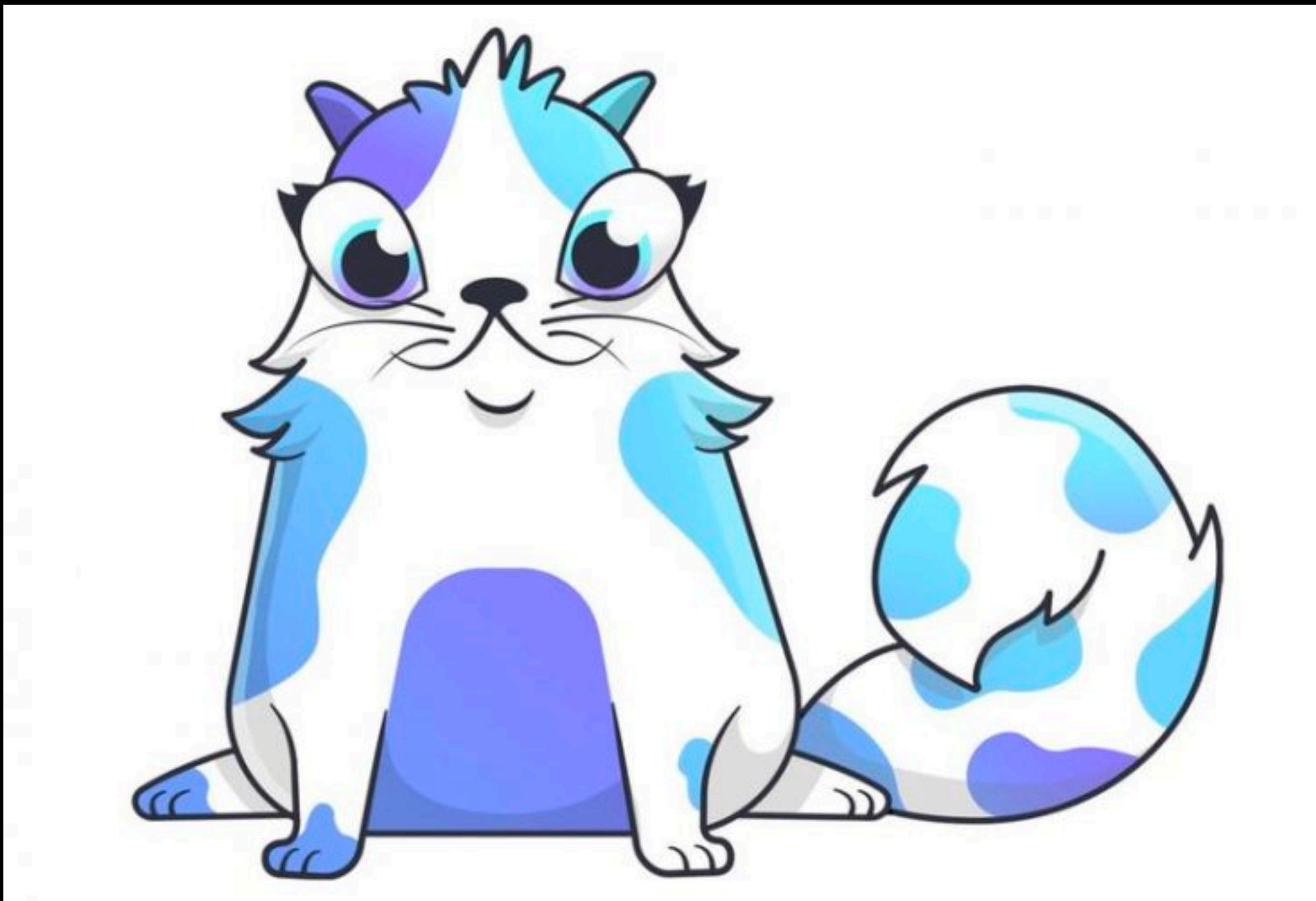
- Non-Fungibility (대체 불가능) : 남이 가진 1토큰과 내가 가진 1토큰은 서로 다르다!



실사용 예시: 크립토끼

# 대체 불가능한 토큰 (NFTs)

	Fungible Token	Non-Fungible Token
교환 가능성	상호 교환이 가능해 기존의 호가 기반의 거래소 적합	상호 교환이 불가능해 장외 거래소/OTC 적합 (e.g. AirSwap)
소수점 분할 여부	소수점 단위로 분할하여 거래하거나 소유할 수 있음	서로가 지니는 1토큰은 희소성을 띠므로 (일반적으로) 소수점 단위로 분할할 수 없음
기준 프로토콜	ERC20 (Ethereum), EOS.IO Default (EOS)	ERC721, EIP1155 (Ethereum), EOSIO.NFT (EOS)
사용 예시	유ти리티 토큰, Payment 기반 코인 (Bitcoin 등)	게임 아이템 (e.g. EOS Knights, CryptoKitties), KYC 인증서 등

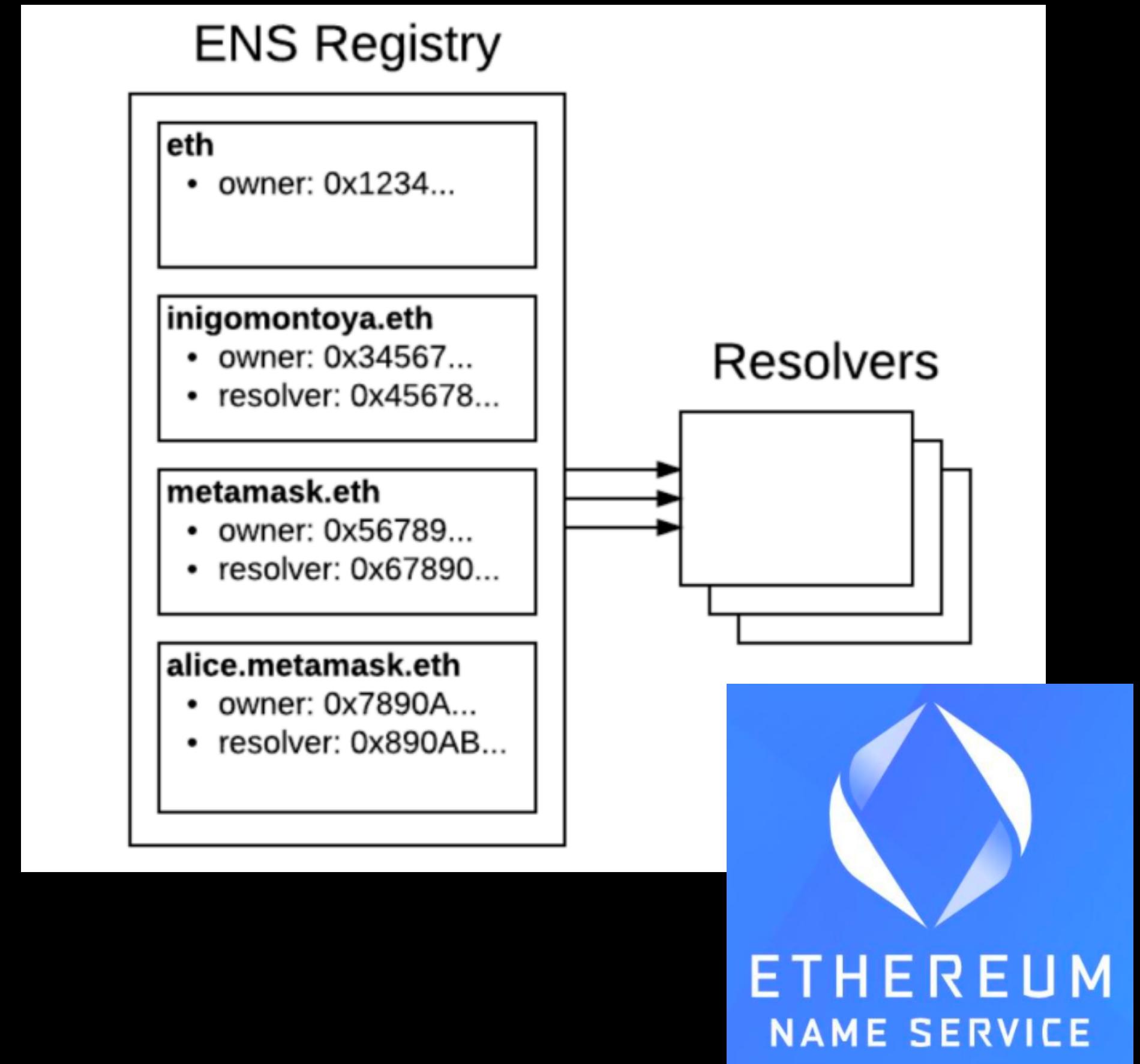
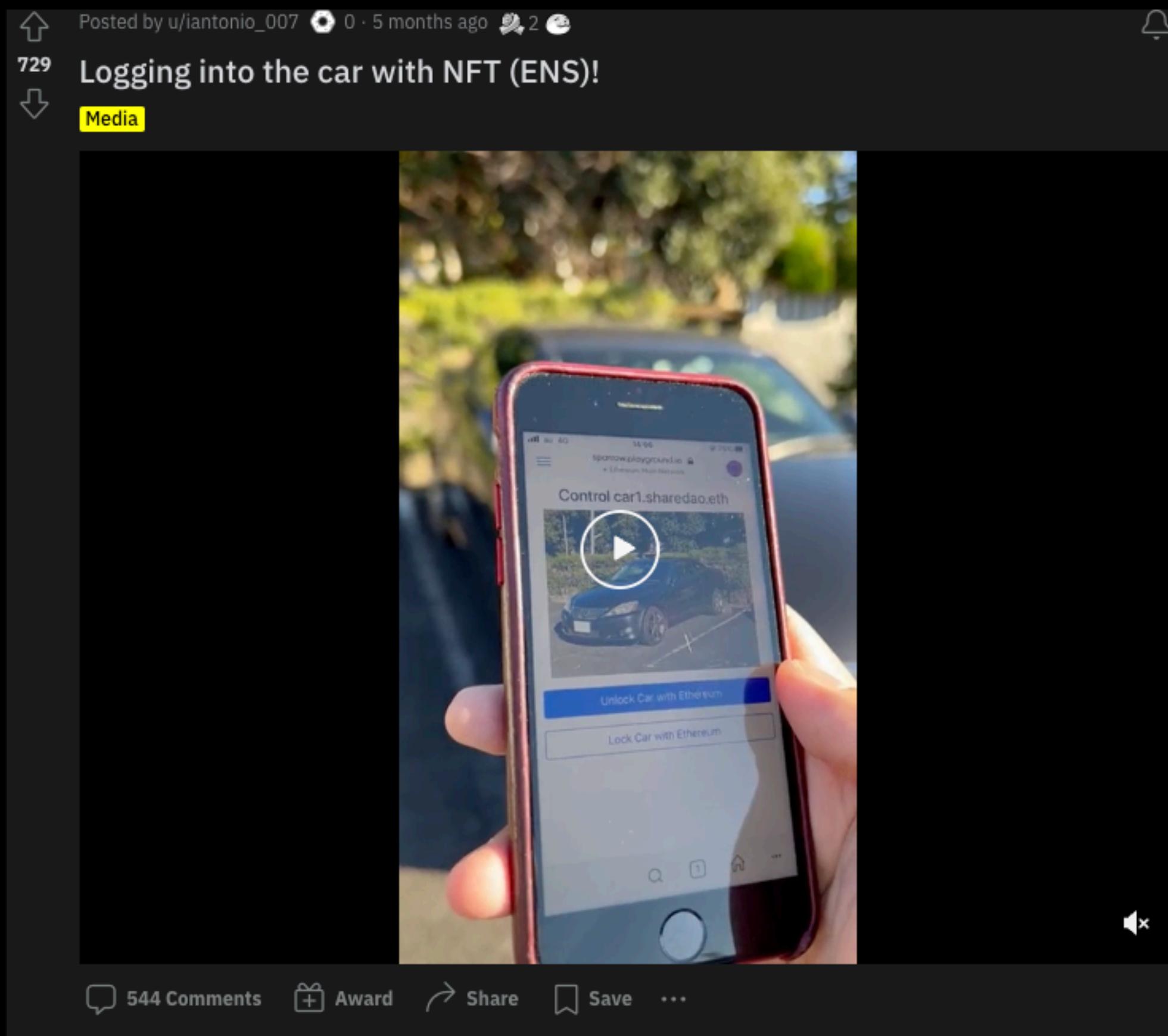


교배를 통해 태어난 고양이는 교배한 부모의 세대 중 높은 세대 +1 (2세대와 4세대 교배하면 자녀는 5세대가 됩니다) 이라는 세대값과 해당 개체만이 가지는 고유한 유전 형질을 얻게 됩니다. 이러한 유전 형질은 각각의 개체를 구분해주는 도구이자, 시장에서 얼마만큼의 가치를 지니는지를 측정하는 척도이기에 사실상 이 게임의 알파이자 오메가라고 할 수 있습니다.

<https://www.cryptokitties.co/catalogue>

그러나 유전자를 가지고 있으면 해서는 크게 쓸모가 없습니다. 형질을 겉으로 발현시켜 눈으로 확인할 수 있어야 비로소 가치가 올라가는 법인데요. 그러나 어떤 형질이 발현될지는 부모와 조상 개체들이 가지고 있는 특별한 유전자가 섞여 랜덤하게 정해지기에 플레이어 입장에서는 교배가 가능한 시간이 돌아올 때마다 접붙이기를 하는 것 외에는 특별히 손을 쓸 수 있는 것이 없습니다.

또한 설사 겉으로 형질이 드러나지 않더라도 잠재적으로는 특별한 유전자를 가지고 있을 수도 있는데, 평범한 싸구려 고양이 사이에서 희귀하고 비싼 품종이 태어나는 경우도 간혹 있다고 하는데요. 때문에 다른 수집형 게임들과 마찬가지로 여기에서도 운만 좋다면 싸구려 고양이들로도 대박을 노릴 수 있습니다.



[https://www.reddit.com/r/ethtrader/comments/r5imf0/logging\\_into\\_the\\_car\\_with\\_nft\\_ens/](https://www.reddit.com/r/ethtrader/comments/r5imf0/logging_into_the_car_with_nft_ens/)

<https://app.ens.domains/>

**Details**

Contract Address	0x495f...7b5e
Token ID	3539207056350968...
Token Standard	ERC-1155
Blockchain	Ethereum
Metadata	Centralized

**OpenSea API**

- API Overview
- Asset Model
- Event Model
- Account Model
- Collection Model
- Retrieve assets GET
- Retrieve events GET
- Retrieve collections GET
- Retrieve bundles GET
- Retrieve an asset** GET
- Retrieve listings GET
- Retrieve offers GET
- Retrieve a contract GET
- Retrieve a collection GET
- Retrieve collection stats GET

**OpenSea Listings**

- Getting started with listings
- Terminology
- Retrieve orders GET
- Retrive orders (Testnets) GET

**OpenSea Testnets API**

- Testnets API Overview
- Retrieve assets (Testnets) GET
- Retrive assets (Testnets) GET

**Retrieve an asset**

GET [https://api.opensea.io/api/v1/asset/{asset\\_contract\\_address}/{token\\_id}/](https://api.opensea.io/api/v1/asset/{asset_contract_address}/{token_id}/)

This endpoint is used to fetch information about a single NFT, based on its contract address and token ID. The response will contain an [Asset Object](#).

Requests to this endpoint require an API key. If you don't have an API key, please [request one](#).

**PATH PARAMS**

asset_contract_address	string required	0x06012c8cf97BE
Address of the contract for this NFT		
token_id	string required	1644426
Token ID for this item		

**QUERY PARAMS**

account_address	string	
Address of an owner of the token. If you include this, the response will include an <code>ownership</code> object that includes the number of tokens owned by the address provided instead of the <code>top_ownerships</code> object included in the standard response, which provides the number of tokens owned by each of the 10 addresses with the greatest supply of the token.		
include_orders	string	false
A flag determining if order information should be included in the response. The default value of this flag is false.		

**HEADERS**

X-API-KEY	string	
Your API key		

**LANGUAGE**

- Shell
- Node
- Ruby
- PHP
- Python
- ...

**CURL**

```
curl --request GET \
--url 'https://api.opensea.io/api/v1/asset/0x06012c8cf97BE/1644426'
```

**REQUEST**

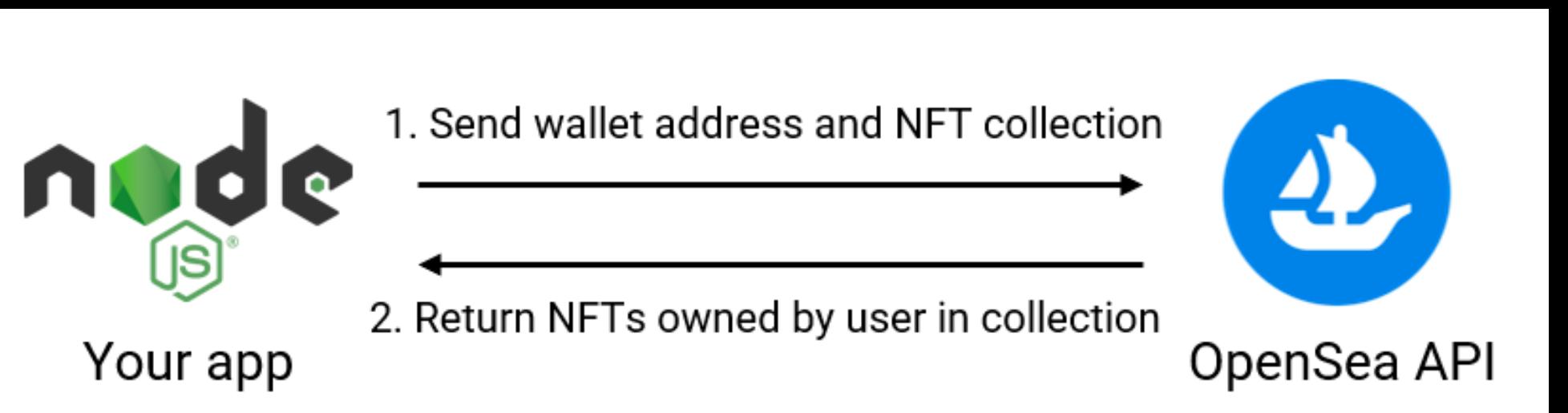
**Try It!**

**RESPONSE**

200 Log

```
1: {
2:   "id": 4493155,
3:   "num_sales": 0,
4:   "background_color": "faeefa",
5:   "image_url": "https://lh3.googleusercontent.com/qax",
6:   "image_preview_url": "https://lh3.googleusercontent",
7:   "image_thumbnail_url": "https://lh3.googleusercontent",
8:   "image_original_url": "https://img.cryptokitties.co",
9:   "animation_url": null,
10:  "animation_original_url": null,
11:  "name": "Vlad Whiptush",
12:  "description": "Hai! I'm Vlad Whiptush, a handsome !",
13:  "external_link": "https://www.cryptokitties.co/kitt",
14:  "asset_contract": {
15:    "address": "0x06012c8cf97bead5deae237070f9587f8e7",
16:    "asset_contract_type": "non-fungible",
17:    "created_date": "2018-01-23T04:51:38.832339",
18:    "name": "CryptoKitties",
19:    "nft_version": "1.0",
20:    "opensea_version": null,
21:    "owner": 463841,
22:    "schema_name": "ERC721",
23:    "symbol": "KITTY",
24:    "total_supply": null,
25:    "description": "CryptoKitties is a game centered",
26:    "external_link": "https://www.cryptokitties.co/","
```

- OpenSea, LooksRare 등은 블록체인 상에 올라와 있는 ERC721 NFT 컨트랙트의 내부 장부 (mapping 형태로 되어 있음) 와 통신하여 내가 원하는 NFT 토큰과 상호작용 할 수 있도록 하는 서비스 (컨트랙트, 프론트엔드 별도로 구현)
- 즉, OpenSea, LooksRare 등 NFT 거래소는 블록체인에 배포된 컨트랙트이고, 컨트랙트와 통신할 수 있는 프론트엔드 페이지 혹은 API를 함께 제공한다.
- OpenSea 클론코딩을 하는 프론트엔드 사이트를 만들 수 있는데 재미있는 점은, 백엔드 서버 쓰는 것마냥, 누구나 OpenSea 컨트랙트를 자신의 프론트엔드 사이트에 붙일 수 있다는 점이다.



<https://docs.opensea.io/reference/retrieving-a-single-asset>

Search by players, teams, and sets

FILTERS (1)

SORT BY Listing Date (Newest)

**MARCUS SMART**  
Steal - Jan 20 2021, Holo Icon (Series 2), BOS  
Legendary #/99 LE  
Lowest Ask **USD \$2200.00**  
15 listings

**TOBIAS HARRIS**  
Assist - Jan 16 2021, Holo Icon (Series 2), PHI  
Legendary #/99 LE  
Lowest Ask **USD \$2300.00**  
19 listings

**TIM HARDAWAY JR.**  
Layup - Jan 22 2021, Holo Icon (Series 2), DAL  
Legendary #/99 LE  
Lowest Ask **USD \$1925.00**  
18 listings

**DEJOUNTE MURRAY**  
Jump Shot - Jan 20 2021, Holo Icon (Series 2), SAS  
Legendary #/99 LE  
Lowest Ask **USD \$2300.00**  
19 listings

**STEPH CURRY**  
3 Pointer - Jan 23 2021, Holo Icon (Series 2), GSW  
Legendary #/99 LE  
Lowest Ask **USD \$15000.00**  
26 listings

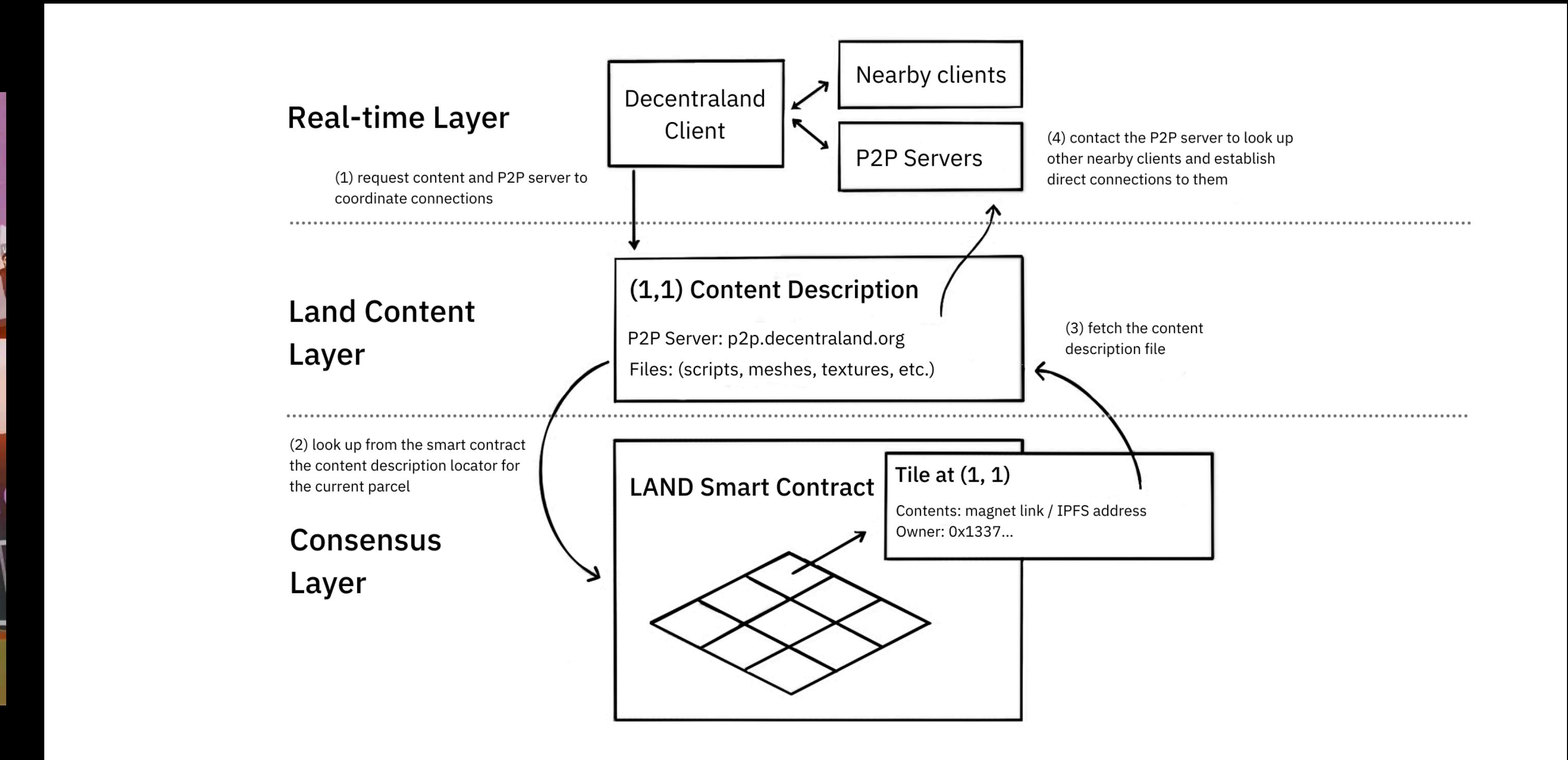
**T.J. WARREN**  
Layup - Dec 26 2020, Holo Icon (Series 2), IND  
Legendary #/99 LE  
Lowest Ask **USD \$1947.00**  
19 listings

**GORDON HAYWARD**  
Layup - Jan 24 2021, Holo Icon (Series 2), CHA  
Legendary #/99 LE  
Lowest Ask **USD \$5475.00**  
13 listings

**JERAMI GRANT**  
Dunk - Jan 18 2021, Holo Icon (Series 2), DET  
Legendary #/99 LE  
Lowest Ask **USD \$3260.00**  
22 listings

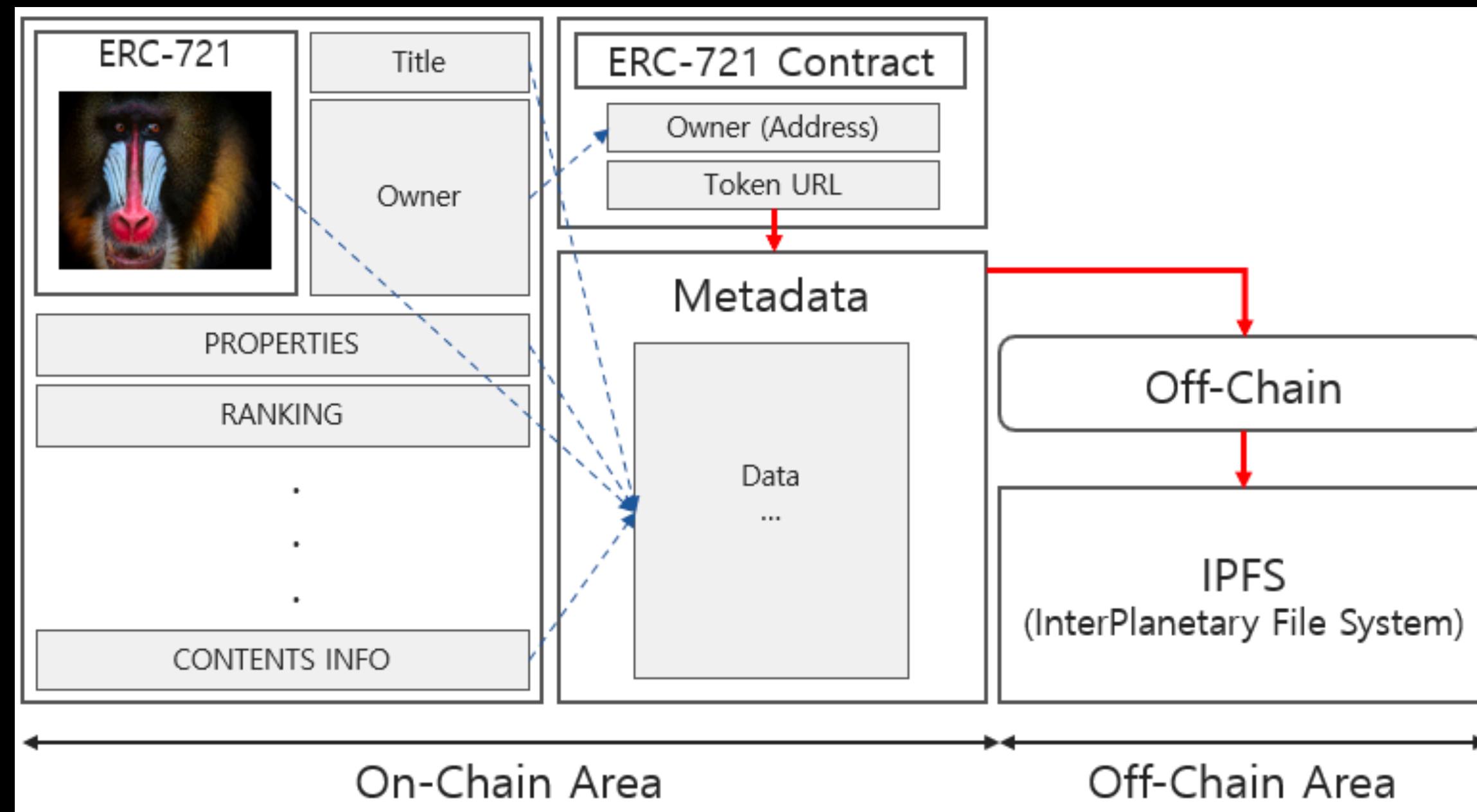
<https://nbatopshot.com/marketplace>

<https://dappradar.com/flow/collectibles/nba-topshot>



<https://www.readthegeneralist.com/briefing/decentraland>

# ERC721 Architecture



구분	주요요소	설명
On-Chain Area	ERC-721	- ERC-721 기반의 토큰 - 소유자 정보 - 컨텐츠 핵심 정보 TAG (이미지, 음악, 게임 등 다양한 컨텐츠 정보 TAG)
	ERC-721 Contract	- 소유자의 주소 정보 획득 - Metadata와 연결을 위한 URL 정보 획득
	Metadata	- 토큰의 컨텐츠에 대한 상세 TAG - 실제 데이터는 블록체인 내부에 저장하기에는 용량의 한계 존재
Off-Chain Area	Off-Chain	- 토큰의 실제 데이터를 외부에 저장하기 위한 기술
	IPFS	- 분산 환경에서 데이터를 저장하기 위한 파일 시스템

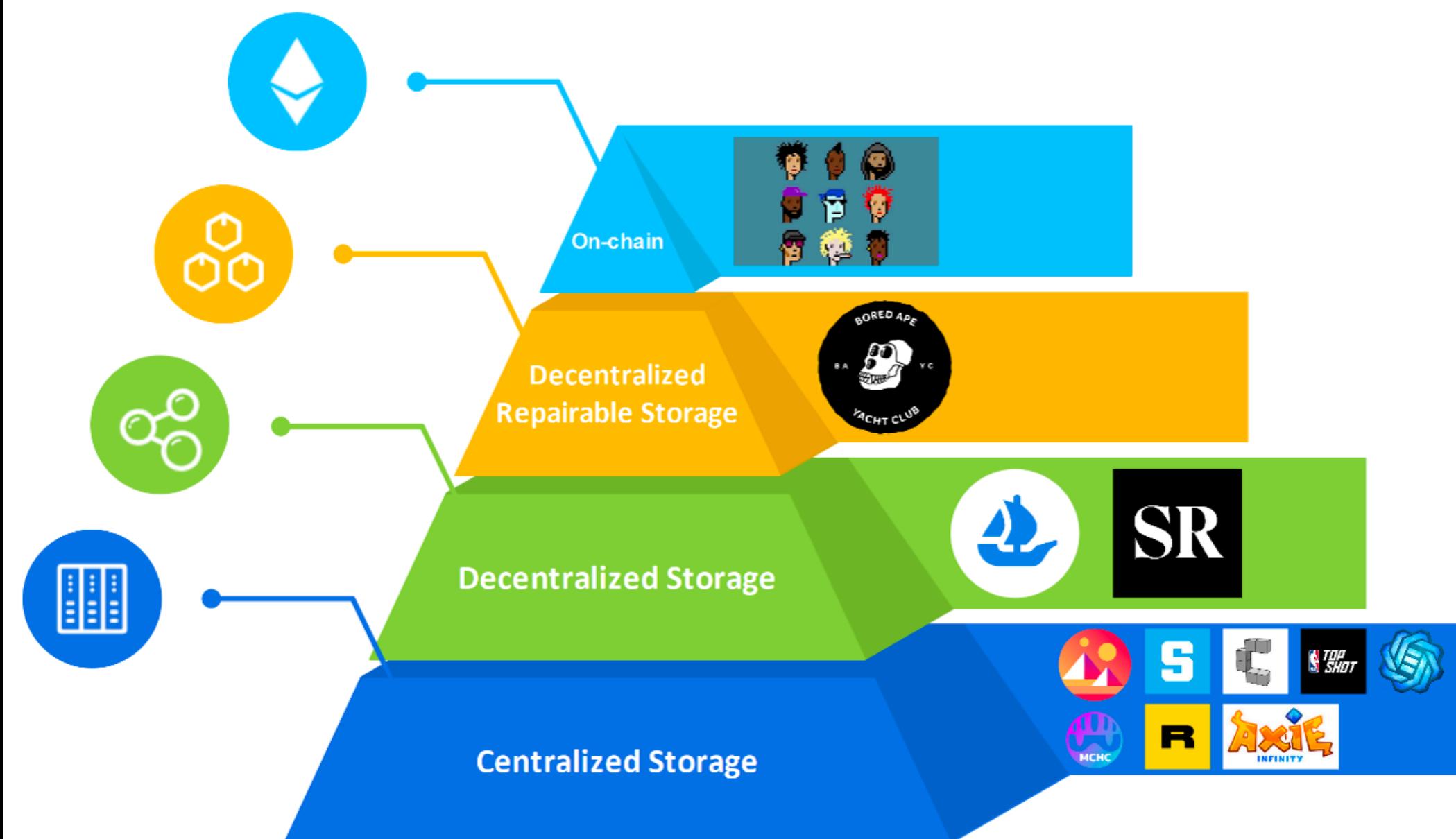
<https://itpenote.tistory.com/683>

<https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md>

# ERC721 NFT를 구현할 때 주의해야 할 점

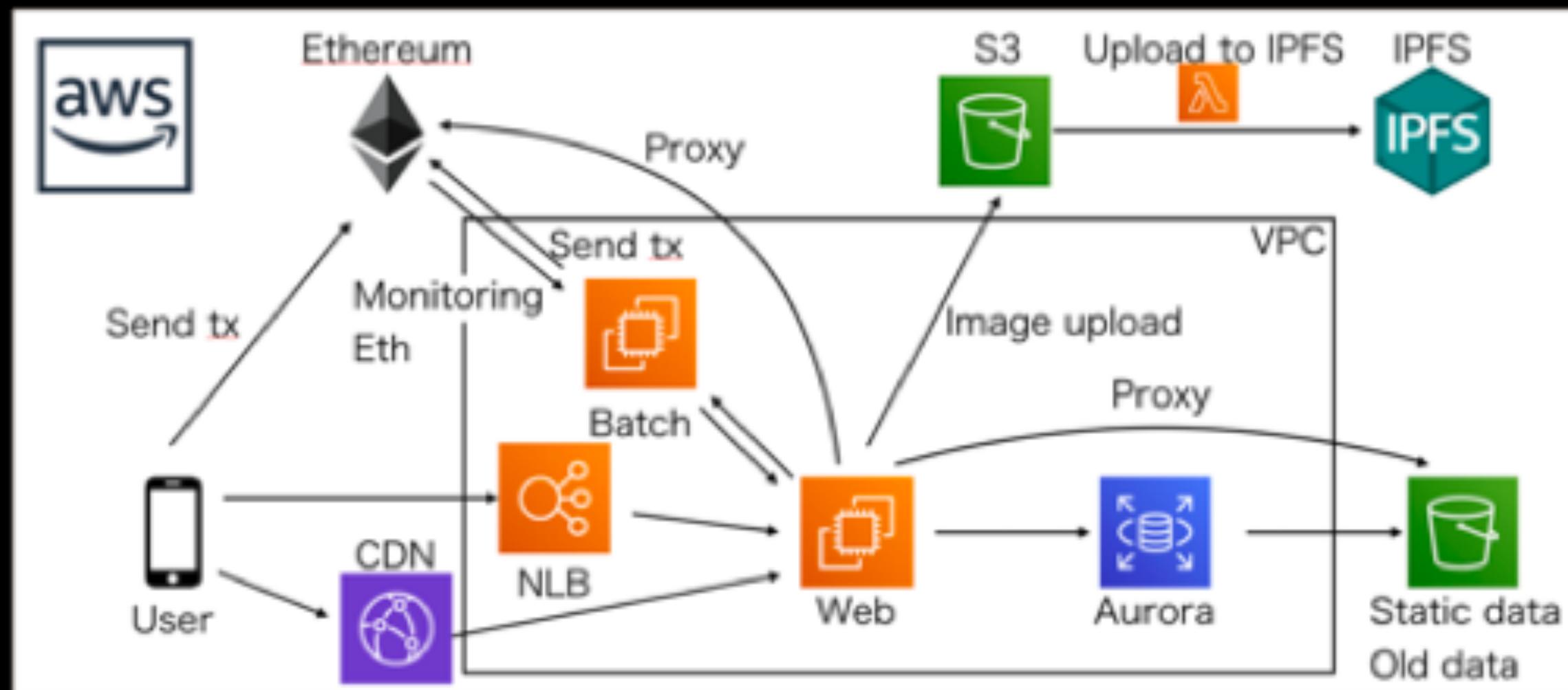
## Status of NFT storage

Metaverse, digital artworks, collectibles, games and platforms

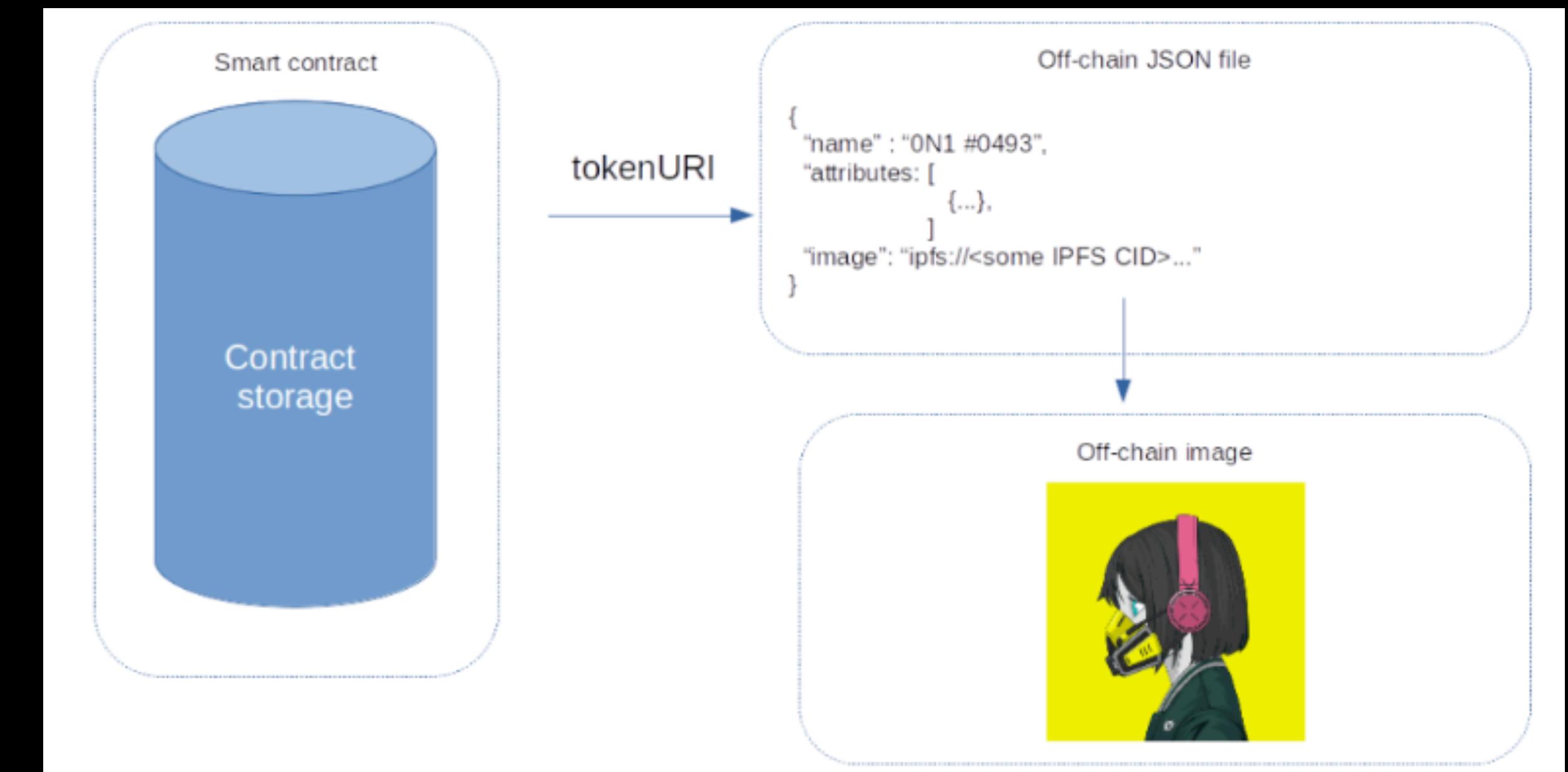


- 코드에서 `for/while loop`를 사용하지 마라. 가스비가 무한으로 올라가는 이슈가 있으므로, 필요한 경우에는 `for loop`을 활용하지 않고, 여러 트랜잭션을 프론트엔드 단에서 동시에 쓸 수 있는 multicall을 활용한다.
- 모든 메타데이터를 온체인상으로 올리기에는 너무 비싸다. 그래서 IPFS를 쓰거나, AWS 같은 오프체인 DB를 사용한다.
- NFT에서의 고유 식별은 uint256의 ID로 식별이 되기 때문에 단순하게 순차적으로 증가하는 패턴보다는 "Black box"로 취급해야 한다.

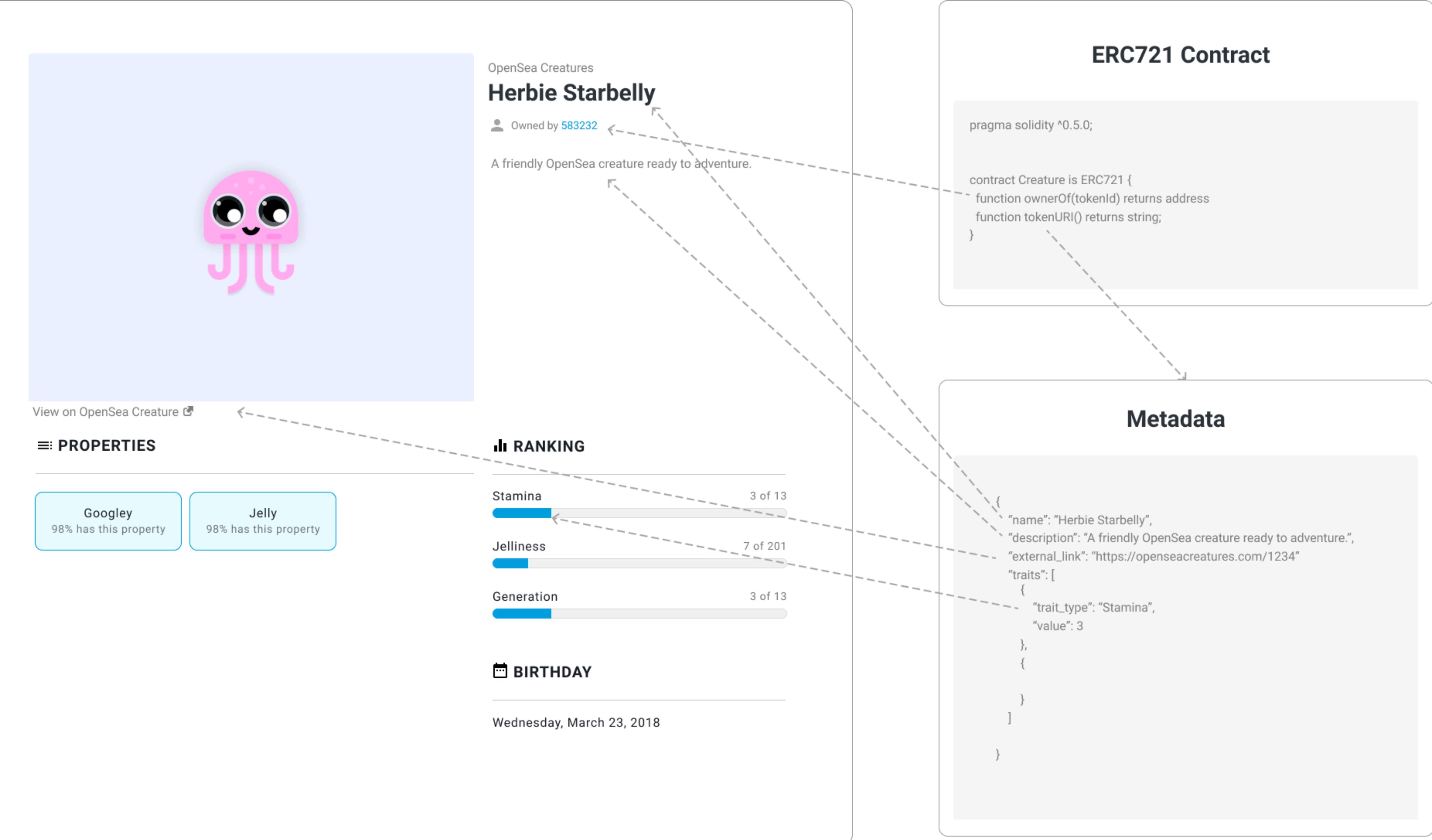
# How NFT stores off-chain data?



[https://aws.amazon.com/jp/blogs/startup/saoy2019\\_finalist\\_doublejump\\_tokyo/](https://aws.amazon.com/jp/blogs/startup/saoy2019_finalist_doublejump_tokyo/)



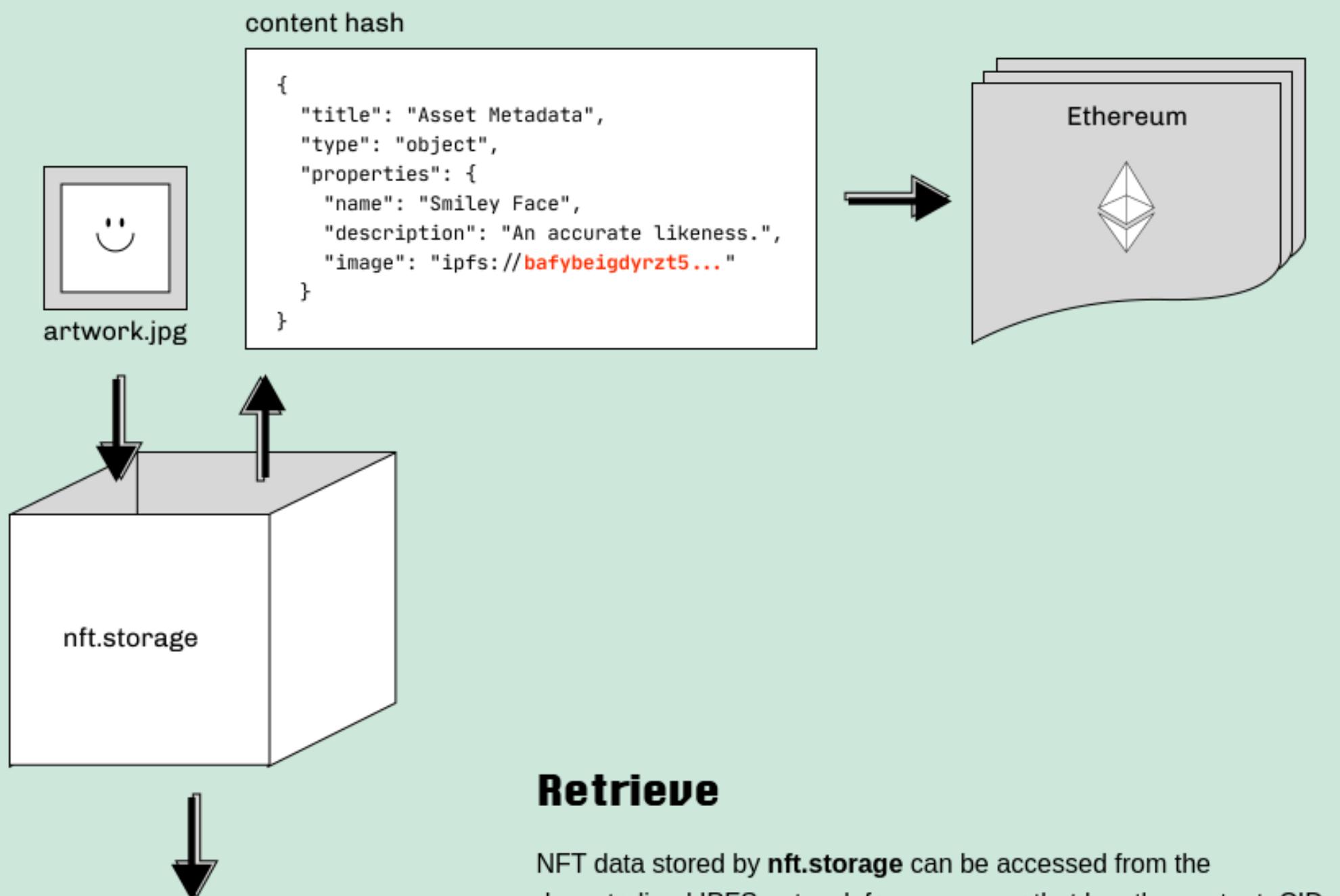
<https://leftasexercise.com/2021/10/03/using-nft-metadata-to-safely-store-digital-assets/>



## Store

Just upload your data and you'll receive an IPFS hash of the content (a CID) that can be used in **on-chain** NFT data as a pointer to the content.

Filecoin provides long term storage for the data ensuring that even if **nft.storage** is attacked or taken down the NFT data persists!



## Retrieve

NFT data stored by **nft.storage** can be accessed from the decentralized IPFS network from any peer that has the content. CIDs reference **immutable** content so you can be sure the content you access is the content referenced in the NFT.

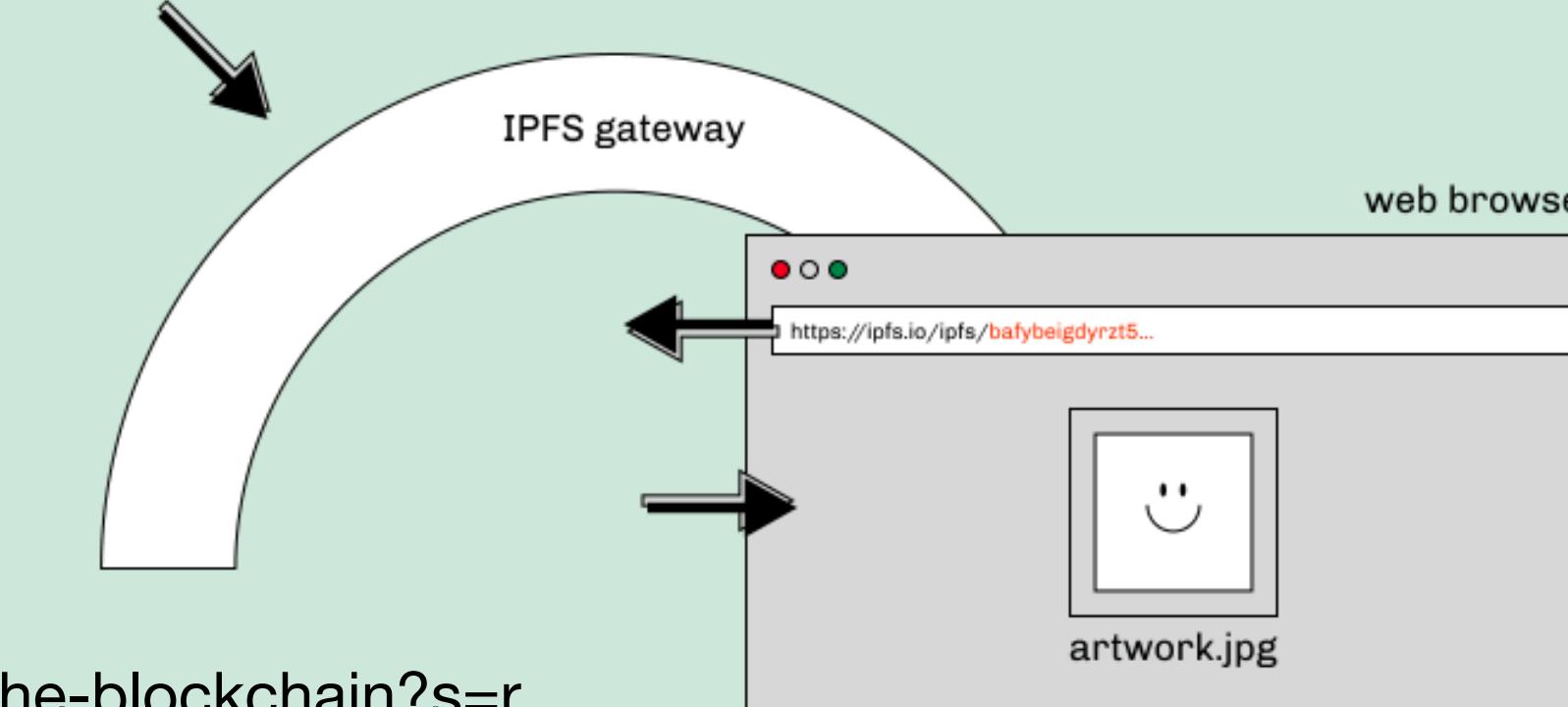
The data can be fetched directly in the browser using [Brave](#), or via a [public IPFS gateway](#), or by using [IPFS Desktop](#) or the [IPFS command line](#).

### command line

```
$ lotus client retrieve #...  
$ # OR  
$ ipfs cat bafybeigdyrzt5...
```

### IPFS gateway

### web browser



# Why NFT id should be “black-box” ?

Is there some standardization on how this ID is assigned?

1 The [ERC-721](#) standard explicitly states that there is no standard to assign the ID (except for the `uint256` datatype):

✓ While some ERC-721 smart contracts may find it convenient to start with ID 0 and simply increment by one for each new NFT, callers SHALL NOT assume that ID numbers have any specific pattern to them, and MUST treat the ID as a "black box".

e.g., do you really own this tweet, or do you only own it if you apply the tweet->token ID mapping

Token ownership does not mean that you own the underlying resource. It only means that you own the token (representing the resource).

Could you provide a few examples of how exactly is the token ID assigned to some (well-known) NFTs?

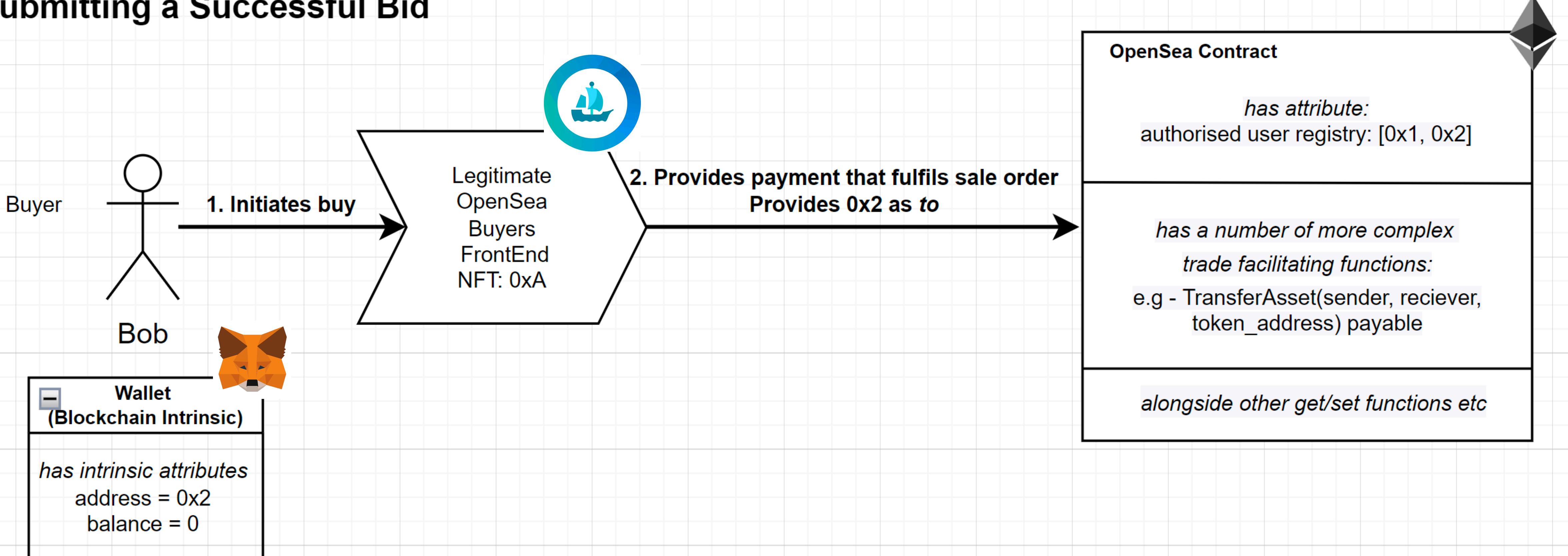
- CryptoKitties - [link](#), line 412, incrementing

```
uint256 newKittenId = kitties.push(_kitty) - 1;
```
- CryptoPunks - [link](#), lines 73 and 83, assigning ID set by the (authorized) caller

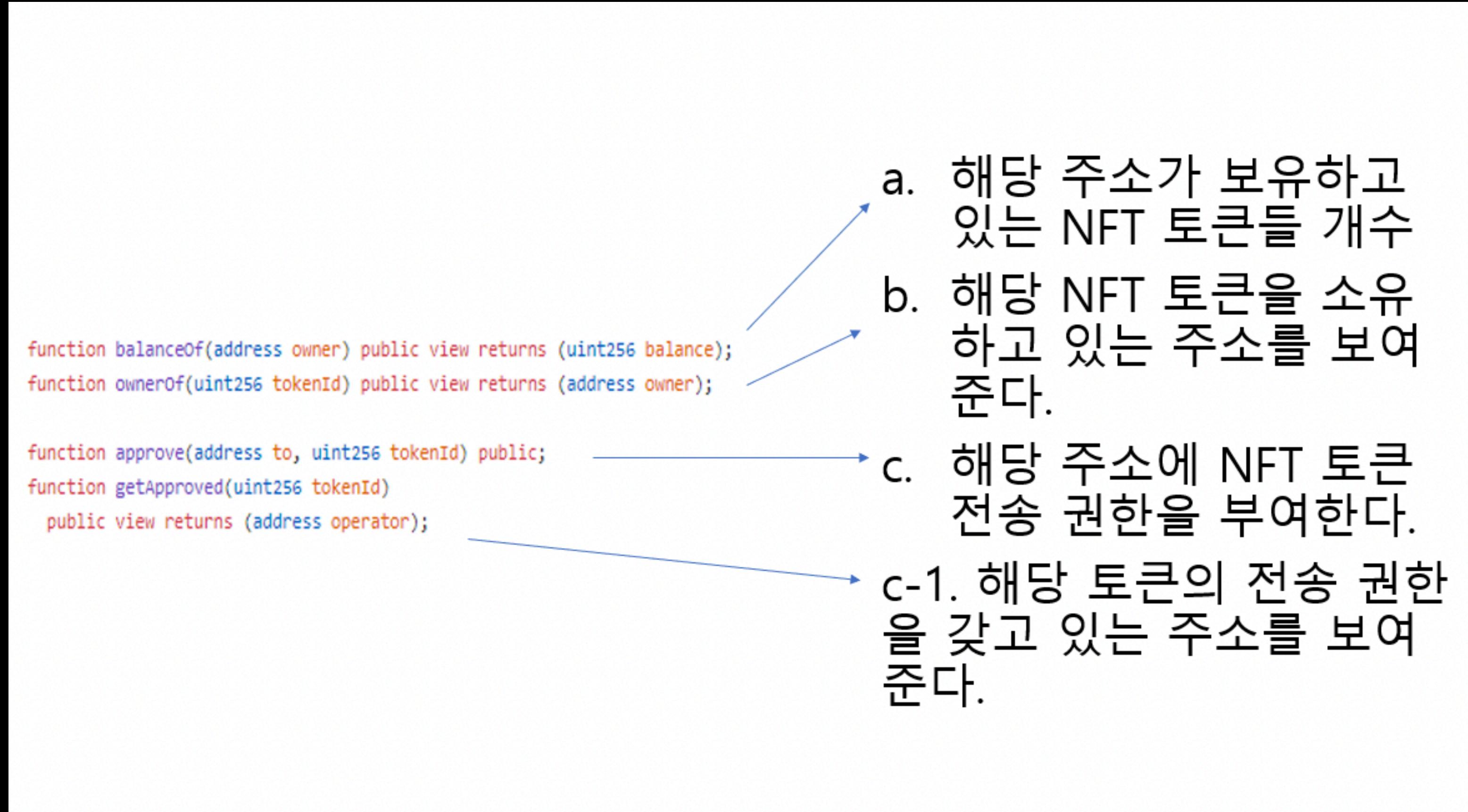
```
mapping (uint => address) public punkIndexToAddress;

function setInitialOwner(address to, uint punkIndex) {
    // ...
    punkIndexToAddress[punkIndex] = to;
```

## Submitting a Successful Bid



# ERC721 인터페이스 살펴보기



# ERC721 인터페이스 살펴보기

The diagram illustrates the ERC721 interface with annotations pointing to specific functions:

- d.** NFT 토큰 소유자가 해당 주소에게 모든 NFT 토큰에 대한 전송 권한을 부여(해제)한다.  
Annotation points to the `setApprovalForAll` function.
- d-1.** 'd'의 권한이 있는지 참/거짓을 보여준다.  
Annotation points to the `isApprovedForAll` function.
- e.** NFT 토큰 소유자로부터 해당 NFT 토큰을 다른 주소로 전송한다.  
Annotation points to the `transferFrom` function.
- f.** 전송받는(to) 주소가 erc721 토큰을 받을 수 있는지 체크하고 보낸다.  
Annotation points to the `safeTransferFrom` function.

```
function setApprovalForAll(address operator, bool _approved) public;
function isApprovedForAll(address owner, address operator)
    public view returns (bool);

function transferFrom(address from, address to, uint256 tokenId) public;
function safeTransferFrom(address from, address to, uint256 tokenId)
    public;

function safeTransferFrom(
    address from,
    address to,
    uint256 tokenId,
    bytes data
)
    public;
```

# ERC721 OpenZeppelin 구현체 살펴보기

```
contract ERC721 is Context, ERC165, IERC721, IERC721Metadata {
    using Address for address;
    using Strings for uint256;

    // Token name
    string private _name;

    // Token symbol
    string private _symbol;

    // Mapping from token ID to owner address
    mapping(uint256 => address) private _owners;

    // Mapping owner address to token count
    mapping(address => uint256) private _balances;

    // Mapping from token ID to approved address
    mapping(uint256 => address) private _tokenApprovals;

    // Mapping from owner to operator approvals
    mapping(address => mapping(address => bool)) private _operatorApprovals;
```

# ERC721 OpenZeppelin 구현체 살펴보기

```
abstract contract ERC721URIStorage is ERC721 {
    using Strings for uint256;

    // Optional mapping for token URIs
    mapping(uint256 => string) private _tokenURIs;

    /**
     * @dev See {IERC721Metadata-tokenURI}.
     */
    function tokenURI(uint256 tokenId) public view virtual override returns (string memory) {
        require(_exists(tokenId), "ERC721URIStorage: URI query for nonexistent token");

        string memory _tokenURI = _tokenURIs[tokenId];
        string memory base = _baseURI();

        // If there is no base URI, return the token URI.
        if (bytes(base).length == 0) {
            return _tokenURI;
        }
        // If both are set, concatenate the baseURI and tokenURI (via abi.encodePacked).
        if (bytes(_tokenURI).length > 0) {
            return string(abi.encodePacked(base, _tokenURI));
        }

        return super.tokenURI(tokenId);
    }

    function _setTokenURI(uint256 tokenId, string memory _tokenURI) internal virtual {
        require(_exists(tokenId), "ERC721URIStorage: URI set of nonexistent token");
        _tokenURIs[tokenId] = _tokenURI;
    }
}
```

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/extensions/ERC721URIStorage.sol>

# ERC721 OpenZeppelin 구현체 살펴보기

```
function transferFrom(
    address from,
    address to,
    uint256 tokenId
) public virtual override {
    //solhint-disable-next-line max-line-length
    require(_isApprovedOrOwner(_msgSender(), tokenId), "ERC721: transfer caller is not owner nor approved");

    _transfer(from, to, tokenId);
}

function _isApprovedOrOwner(address spender, uint256 tokenId) internal view virtual returns (bool) {
    require(_exists(tokenId), "ERC721: operator query for nonexistent token");
    address owner = ERC721.ownerOf(tokenId);
    return (spender == owner || isApprovedForAll(owner, spender) || getApproved(tokenId) == spender);
}
```

# ERC721 OpenZeppelin 구현체 살펴보기

```
function _transfer(  
    address from,  
    address to,  
    uint256 tokenId  
) internal virtual {  
    require(ERC721.ownerOf(tokenId) == from, "ERC721: transfer from incorrect owner");  
    require(to != address(0), "ERC721: transfer to the zero address");  
  
    _beforeTokenTransfer(from, to, tokenId);  
  
    // Clear approvals from the previous owner  
    _approve(address(0), tokenId);  
  
    _balances[from] -= 1;  
    _balances[to] += 1;  
    _owners[tokenId] = to;  
  
    emit Transfer(from, to, tokenId);  
  
    _afterTokenTransfer(from, to, tokenId);  
}
```

# ERC721 OpenZeppelin 구현체 살펴보기

```
function safeTransferFrom(
    address from,
    address to,
    uint256 tokenId,
    bytes memory _data
) public virtual override {
    require(_isApprovedOrOwner(_msgSender(), tokenId), "ERC721: transfer caller is not owner nor approved");
    _safeTransfer(from, to, tokenId, _data);
}
```

```
function _safeTransfer(
    address from,
    address to,
    uint256 tokenId,
    bytes memory _data
) internal virtual {
    _transfer(from, to, tokenId);
    require(_checkOnERC721Received(from, to, tokenId, _data), "ERC721: transfer to non ERC721Receiver implementer");
}
```

# ERC721 OpenZeppelin 구현체 살펴보기

```
function _checkOnERC721Received(
    address from,
    address to,
    uint256 tokenId,
    bytes memory _data
) private returns (bool) {
    if (to.isContract()) {
        try IERC721Receiver(to).onERC721Received(_msgSender(), from, tokenId, _data) returns (bytes4 retval) {
            return retval == IERC721Receiver.onERC721Received.selector;
        } catch (bytes memory reason) {
            if (reason.length == 0) {
                revert("ERC721: transfer to non ERC721Receiver implementer");
            } else {
                assembly {
                    revert(add(32, reason), mload(reason))
                }
            }
        }
    } else {
        return true;
    }
}
```

참고 ERC165 표준: byte가 있는 safeTransferFrom은 실제 유저의 주소(EOA)가 아닌 ERC721 NFT 스마트 컨트랙트 주소인지 확인하는 안전 장치다. 잘못 보내면 NFT가 소실된다.

# ERC721 OpenZeppelin 구현체 살펴보기

토큰을 대신 전송하는 사람이나 거래소 (OpenSea 등) 을 operator라고 하며, 토큰을 보유하고 있는 사람이 token id 와 operator의 address를 입력하면 operator에게 해당 토큰 거래를 허용하게 되는 것이다.  
getApproved는 해당 token id를 입력하면 그 토큰에 해당하는 operator를 반환해준다.

```
function _setApprovalForAll(
    address owner,
    address operator,
    bool approved
) internal virtual {
    require(owner != operator, "ERC721: approve to caller");
    _operatorApprovals[owner][operator] = approved;
    emit ApprovalForAll(owner, operator, approved);
}
```

```
function approve(address to, uint256 tokenId) public virtual override {
    address owner = ERC721.ownerOf(tokenId);
    require(to != owner, "ERC721: approval to current owner");

    require(
        _msgSender() == owner || isApprovedForAll(owner, _msgSender()),
        "ERC721: approve caller is not owner nor approved for all"
    );
    _approve(to, tokenId);
}

function _approve(address to, uint256 tokenId) internal virtual {
    _tokenApprovals[tokenId] = to;
    emit Approval(ERC721.ownerOf(tokenId), to, tokenId);
}
```

# ERC721 OpenZeppelin 구현체 살펴보기

```
interface ERC721Metadata {
    function name() external view returns (string _name);

    function symbol() external view returns (string _symbol);

    function tokenURI(uint256 _tokenId) external view returns (string);
}

/*
위 ERC721 Metadata 의 JSON 스키마 형태
{
    "title": "Asset Metadata",
    "type": "object",
    "properties": {
        "name": {
            "type": "string",
            "description": "Identifies the asset to which this NFT represents"
        },
        "description": {
            "type": "string",
            "description": "Describes the asset to which this NFT represents"
        },
        "image": {
            "type": "string",
            "description": "A URI pointing to a resource with mime type image/* represent"
        }
    }
}
*/
```

[https://velog.io/@yonaaaaaaa\\_a/ERC-721%EA%B3%BC-NFT](https://velog.io/@yonaaaaaaa_a/ERC-721%EA%B3%BC-NFT)

<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC721/extensions/IERC721Metadata.sol>

# Tip) Counters is a library.

using A for B 의 의미는 library A를 B 데이터 타입에 사용하겠다는 뜻. 라이브러리를 사용하여 검증된 방식의 가스비 절약 등의 효과를 누림

```
library Counters {
    struct Counter {
        uint256 _value; // default: 0
    }

    function current(Counter storage counter) internal view returns (uint256) {
        return counter._value;
    }

    function increment(Counter storage counter) internal {
        unchecked {
            counter._value += 1;
        }
    }

    function decrement(Counter storage counter) internal {
        uint256 value = counter._value;
        require(value > 0, "Counter: decrement overflow");
        unchecked {
            counter._value = value - 1;
        }
    }

    function reset(Counter storage counter) internal {
        counter._value = 0;
    }
}
```

```
contract Counter {
    using Counters for Counters.Counter;
    Counters.Counter clock;
    address public owner = msg.sender;

    constructor(uint256 _value) {
        owner = msg.sender;
        clock._value = _value;
    }

    modifier onlyOwner() {
        require(msg.sender == owner);
       _;
    }

    function current() external view returns (uint256) {
        return clock.current();
    }

    function increment() external onlyOwner returns (uint256) {
        clock.increment();
        return clock.current();
    }
}
```

# Tip) Event is for logging.

increaseWithEvent와 increaseWithoutEvent, count 모두 외부에서 호출 가능한 함수이고 withEvent와 withoutEvent는 동일한 기능 수행 (이벤트 emit만 차이)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

contract EventTestContract {

    mapping(address => uint256) private _counts;

    event Increase(address indexed addr, uint256 oldValue, uint256 newValue);

    function increaseWithEvent() public returns (bool) {
        uint256 oldValue = _counts[msg.sender];
        _increase();
        uint256 newValue = _counts[msg.sender];
        emit Increase(msg.sender, oldValue, newValue);
        return true;
    }

    function increaseWithoutEvent() public returns (bool) {
        _increase();
        return true;
    }

    function _increase() private {
        _counts[msg.sender] += 1;
    }

    function count() public view returns (uint256) {
        return _counts[msg.sender];
    }
}
```

The image displays two side-by-side screenshots of a blockchain transaction details interface, likely from a Ethereum wallet or explorer like MetaMask or Etherscan.

**Left Screenshot (increaseWithoutEvent Transaction):**

- Overview Tab:** Shows basic transaction details: [This is a Rinkeby Testnet transaction only], Transaction Hash, Status, Block, Timestamp, From, To, and Value.
- Logs Tab:** Shows zero logs.

**Right Screenshot (increaseWithEvent Transaction):**

- Overview Tab:** Shows basic transaction details: [This is a Rinkeby Testnet transaction only], Transaction Hash, Status (Success), Block (9088100, 2 Block Confirmations), Timestamp (22 secs ago, Aug-10-2021 02:15:30 AM), From, To, and Value (0 Ether (\$0.00)).
- Logs Tab:** Shows one log entry: increaseWithEvent, with the log value highlighted by a red box.

# Tip) Event is for logging.

increaseWithEvent와 increaseWithoutEvent, count 모두 외부에서 호출 가능한 함수이고 withEvent와 withoutEvent는 동일한 기능 수행 (이벤트 emit만 차이)

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

contract EventTestContract {

    mapping(address => uint256) private _counts;

    event Increase(address indexed addr, uint256 oldValue, uint256 newValue);

    function increaseWithEvent() public returns (bool) {
        uint256 oldValue = _counts[msg.sender];
        _increase();
        uint256 newValue = _counts[msg.sender];
        emit Increase(msg.sender, oldValue, newValue);
        return true;
    }

    function increaseWithoutEvent() public returns (bool) {
        _increase();
        return true;
    }

    function _increase() private {
        _counts[msg.sender] += 1;
    }

    function count() public view returns (uint256) {
        return _counts[msg.sender];
    }
}
```

Transaction Details

Overview	State
[ This is a Rinkeby Testnet transaction only ]	
⑦ Transaction Hash:	0x2dfa0612b77415cac461f8e04af13a4c9a...
⑦ Status:	<span>Success</span>
⑦ Block:	9088097 1 Block Confirmation
⑦ Timestamp:	① 15 secs ago (Aug-10-2021 02:14:45 AM)
⑦ From:	0x6f6eea573089b92252c76d8a221657168
⑦ To:	Contract 0x3fc272cbf070c9eeb4974399da...
⑦ Value:	0 Ether (\$0.00)

increaseWithoutEvent 호출 결과

# Tip) Event is for logging.-

increaseWithEvent의 경우 Data 부분에서 1에서 2로 증가한 모습을 알 수 있으며 increase 이벤트의 addr 변수를 indexed로 명시했으므로 Etherscan에서 검색이 가능 (로깅했으므로)  
Opcode 상 이벤트를 더 emit 한다는 것은 가스비의 추가 지불을 의미하지만 필요한 경우에는 이벤트 로깅이 필요함  
프론트엔드의 경우 event가 호출되면 어떠한 행동을 수행한다 라고 하는 WebHook과 같이 사용할 수 있음

Transaction Details	
Overview	Logs (1)
[ This is a Rinkeby Testnet transaction only ]	
⑦ Transaction Hash:	0x020b1d9074295a5bce203ceb73342ffff
⑦ Status:	<span>✓ Success</span>
⑦ Block:	9088100 <span>2 Block Confirmations</span>
⑦ Timestamp:	⌚ 22 secs ago (Aug-10-2021 02:15:30 AM)
⑦ From:	0x6f6eea573089b92252c76d8a22165716
⑦ To:	Contract <a href="#">0x3fc272cbf070c9eeb4974399d</a>
⑦ Value:	0 Ether (\$0.00)
increaseWithEvent 호출 결과	

<https://hichoco.tistory.com/entry/Solidity%EC%86%94%EB%A6%AC%EB%94%94%ED%8B%B0-%EC%9D%B4%EB%B2%A4%ED%8A%B8event-%EB%B0%A9%EC%B6%9Cemit-%ED%95%98%EA%B3%A0-%EC%95%88%ED%95%98%EA%B3%A0-%EC%B0%A8%EC%9D%B4>

# Minting NFT

```
pragma solidity ^0.8.1;

// We first import some OpenZeppelin Contracts.
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "hardhat/console.sol";

// We inherit the contract we imported. This means we'll have access
// to the inherited contract's methods.
contract MyEpicNFT is ERC721URIStorage {
    // Magic given to us by OpenZeppelin to help us keep track of tokenIds.
    using Counters for Counters.Counter;
    Counters.Counter private _tokenIds;

    // We need to pass the name of our NFTs token and its symbol.
    constructor() ERC721 ("SquareNFT", "SQUARE") {
        console.log("This is my NFT contract. Woah!");
    }

    // A function our user will hit to get their NFT.
    function makeAnEpicNFT() public {
        // Get the current tokenId, this starts at 0.
        uint256 newItemId = _tokenIds.current();

        // Actually mint the NFT to the sender using msg.sender.
        _safeMint(msg.sender, newItemId);

        // Set the NFTs data.
        _setTokenURI(newItemId, "blah");

        // Increment the counter for when the next NFT is minted.
        _tokenIds.increment();
    }
}
```

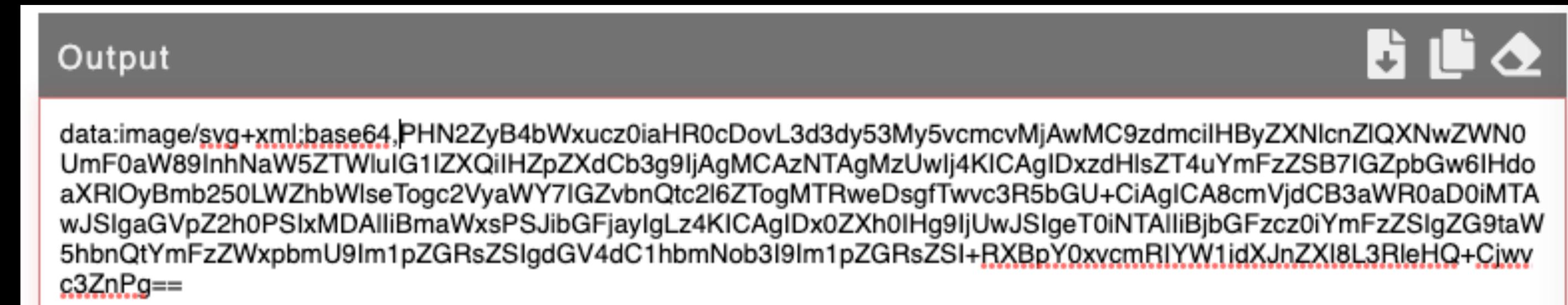
```
{
    "name": "Spongebob Cowboy Pants",
    "description": "A silent hero. A watchful protector.",
    "image": "https://i.imgur.com/v7U019j.png"
}
```

<https://jsonkeeper.com/>

# Storing Image on-chain is SVG

```
<svg xmlns="http://www.w3.org/2000/svg" preserveAspectRatio="xMinYMin meet" viewBox="0 0 350 350">
  <style>.base { fill: white; font-family: serif; font-size: 14px; }</style>
  <rect width="100%" height="100%" fill="black" />
  <text x="50%" y="50%" class="base" dominant-baseline="middle" text-anchor="middle">EpicLordHamburger
</svg>
```

<https://gist.github.com/jyphthemiracle/e7880633a614243dfbbc6eba725eda5d>



data:image/svg+xml;base64,INSERT\_YOUR\_BASE64\_ENCODED\_SVG\_HERE

<https://www.utilities-online.info/base64>

# Storing Image on-chain is SVG

```
{  
  "name": "EpicLordHamburger",  
  "description": "An NFT from the highly acclaimed square collection",  
  "image": "data:image/svg+xml;base64,PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHByZXNlcnZl  
}
```

```
_setTokenURI(newItemId, "data:application/json;base64,ewogICAgIm5hbWUiOiAiRXBpY0xvcmRIYW1idXJnZXIiLAogICAgImRlc2NyaXB0aW9uljoglkFul  
E5GVCBmcm9tIHRoZSBoaWdobHkgYWNjbGFpbWVkJHNxdWFyZSBjb2xsZWN0aW9uliwKICAgICJpbWFnZSI6ICJkYXRhO  
mltYWdIL3N2Zyt4bWw7YmFzZTY0LFBITjJaeUI0Yld4dWN6MGlhSFlwY0RvdkwzZDNkeTUzTXk1dmNtY3ZNaF3TUM5emRt  
Y2IJSEJ5WIhObGNuWmxRWE53WIdOMFVtRjBhVzg5SW5oTmFXNVpUV2x1SUcxbFpYUWIJSFpwWIhkQ2lzzlJakFnTUNB  
ek5UQWdNeIV3SWo0TkNpQWdJQ0E4YzNSNWJHVStMbUpoYzJVZ2V5Qm1hV3hzT2ICM2FHbDBaVHNnWm05dWRDMW  
1ZVzFwYkhrNkIlTmxjbWxtT3ICbWlyNTBMWE5wZW1VNklERTBjSGc3SUgwOEwzTjBIV3hsUGcwS0IDQWdJRHh5WIdOMEII  
ZHBaSFJvUFNJeE1EQWxJaUJvWldsbnFIUTIJakV3TUNVaUIHWnBiR3c5SW1Kc1IXTnJJaUF2UGcwS0IDQWdJRHgwWIho  
MEIIZzlJaIV3SINJZ2VUMGIOVEFsSWICamJHRnpjejBpWW1GelpTSWdaRzl0YVc1aGJuUXRZbUZ6WId4cGJtVTIJBTFwWkd  
Sc1pTSWdkR1Y0ZEMxaGJtTm9iM0k5SW0xcFpHUnNaU0krUlhCcFkweHZjbVJJWVcxaWRYSm5aWEk4TDNSbGVIUSTEUW  
84TDNOMlp6ND0iCn0=
```



data:application/json;base64,INSERT\_YOUR\_BASE64\_ENCODED\_JSON\_HERE

<https://www.utilities-online.info/base64>

<https://blog.simondlr.com/posts/flavours-of-on-chain-svg-nfts-on-ethereum>

# Randomly generate words on an image.

블록체인에서 랜덤함수를 구현하는 것은 불가능: 어느 노드에서 동일하게 실행해도, 같은 결과값을 내보내야 하기 때문 (*indeterministic*)

같은 의미에서 스마트 컨트랙트가 API 요청을 보낼 수 없다: 같은 트랜잭션을 여러 번 날렸을 때 같은 값이 보장되어야 하는데 200이 떨어질지 500이 떨어질지 모른다.

```
string[] firstWords = ["Fantastic", "Epic", "Terrible", "Crazy", "Wild", "Terrifying", "Spooky"];
string[] secondWords = ["Cupcake", "Pizza", "Milkshake", "Curry", "Chicken", "Sandwich", "Salad"];
string[] thirdWords = ["Naruto", "Sasuke", "Sakura", "Goku", "Gaara", "Minato", "Kakashi", "Madara"];
```

<https://gist.github.com/jyphthemiracle/0fa954c885c30d41b72c40481e49251a>

```
random(string(abi.encodePacked("FIRST_WORD", Strings.toString(tokenId))));
```

이것은 랜덤이 아니다: “FIRST\_WORD” 와 tokenId를 알고 있다면 누구나 이 함수로부터 같은 값을 받아올 수 있기 때문이다.

아래 코드를 참고하여 코딩해봅시다.

<https://gist.github.com/jyphthemiracle/daf9f5310310b8e00f49077525bc8e9e>

멀티체인 환경 경험하기: [AllThatNode](#)를 사용하여 같은 컨트랙트 ERC721 코드를 Ethereum Rinkeby 테스트넷과 Polygon 메인넷, 그리고 Avalanche 메인넷에 배포해봅시다.

PROTOCOLS

## All That Protocols



## Ethereum

[WEBSITE](#) [DOCS](#)

Ethereum is an open-source, decentralized blockchain computing platform for smart contracts and decentralized applications.

Create New Project

Go to Faucet

## Get NFT preview

Paste your encoded tokenURI below or add it as a query parameter "code" [example](#)

TokenURI code

```
data:application/json;base64,eyJuYW1lIjogIkEgZGV2ZWxvcGVyIGluIGFuIGVjb21tZXJjZSBzdGFydHVwIGlnbm9yaW5nIHN0YW5kLXVwIG1IZXRpbmdzIHdoaWxlIGxpc3RlbmluZyB0byB0aW
```

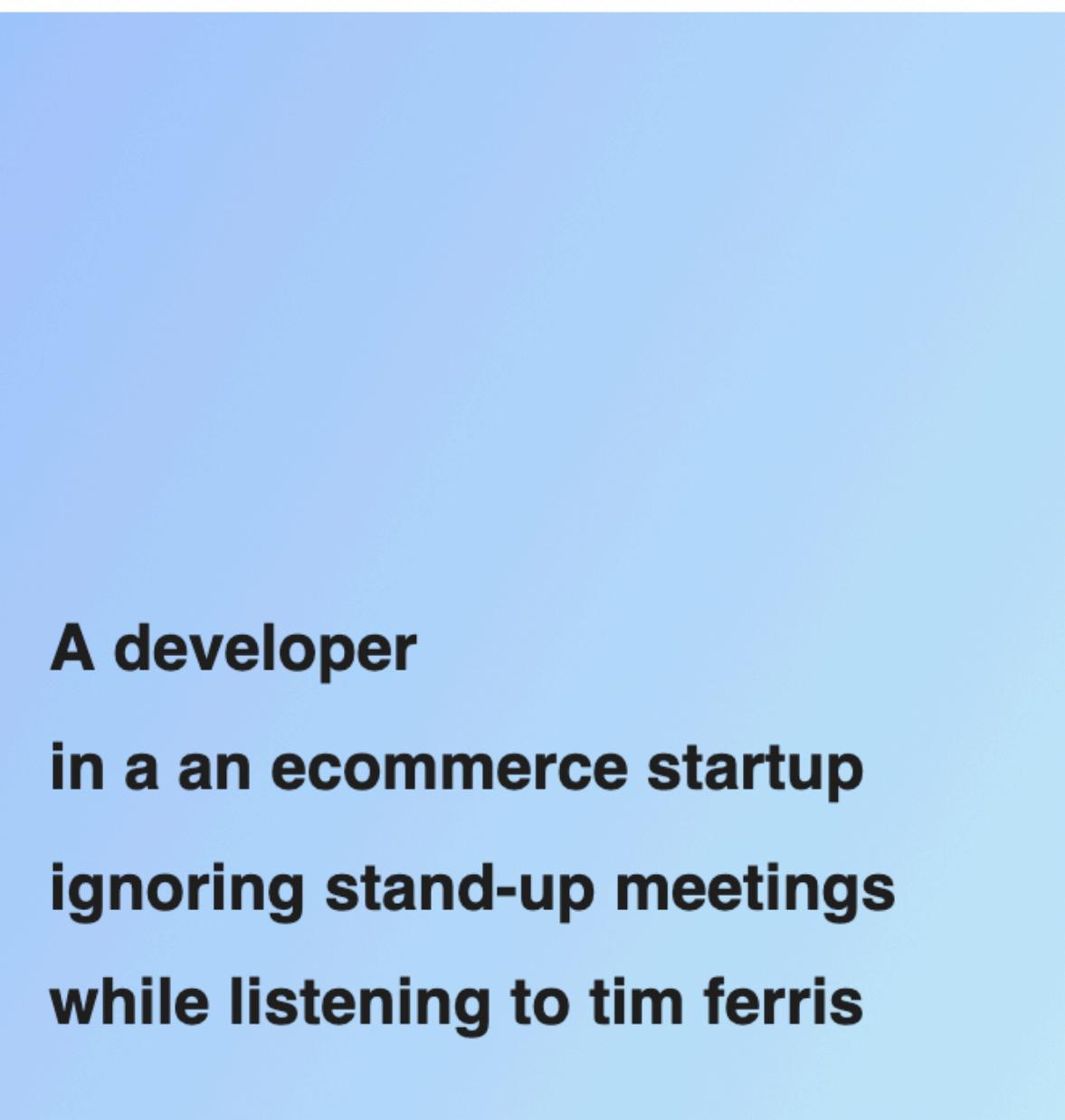
tokenURI

**name**

A developer in an ecommerce startup ignoring stand-up meetings while listening to tim ferris

**description**

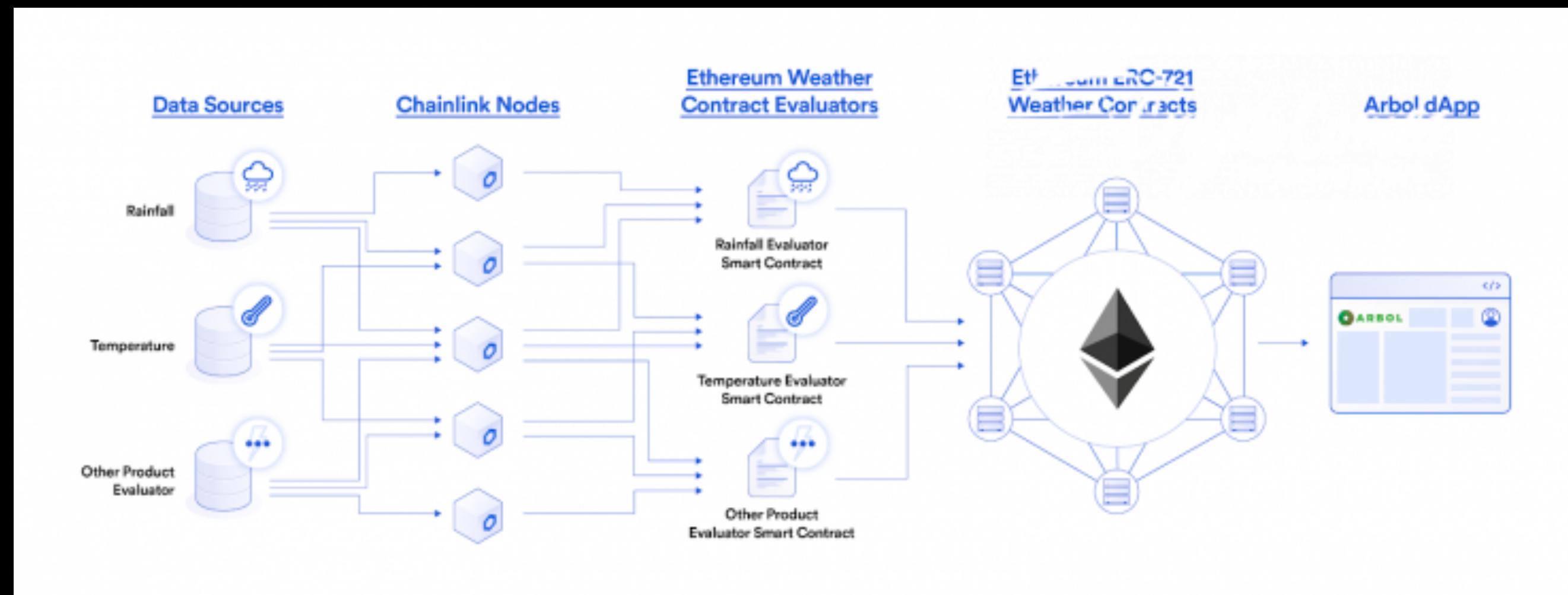
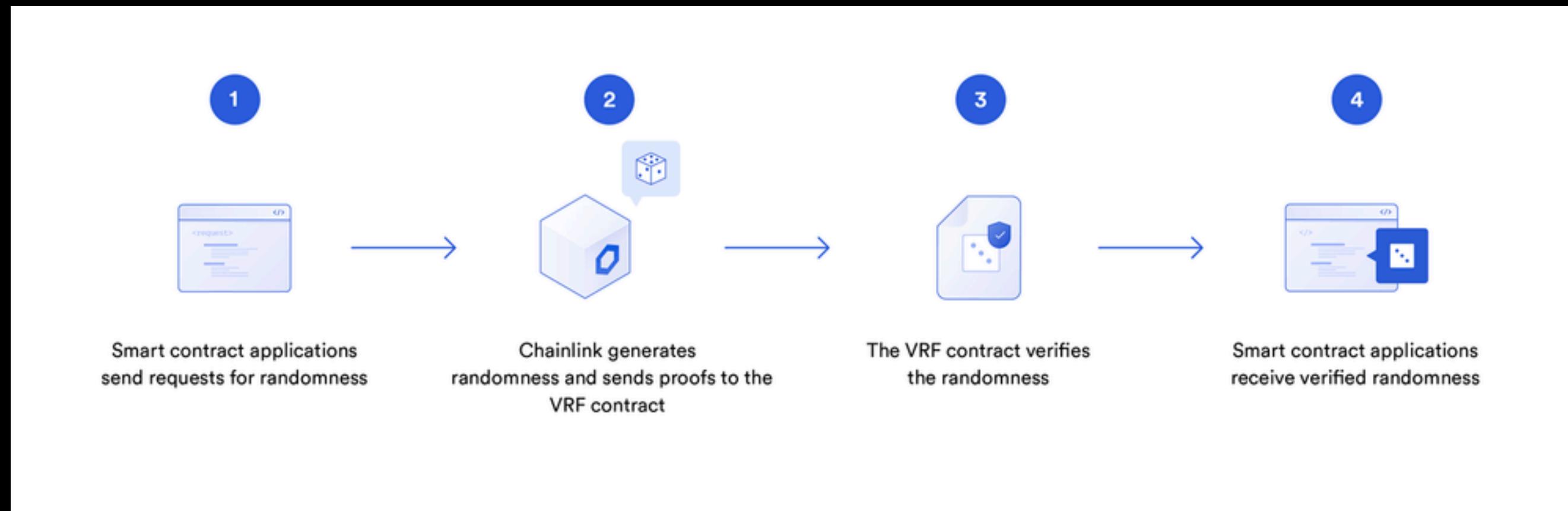
Just Bangalore TechBro things.



# Tip) Oracle Problem: 외부에서 누군가가 정보를 계속 주입해주어야 한다 (날씨, 주가 등..)

그리고 그 사람을 믿어야 한다: 만약 크리스마스에 눈이 내리면 ETH를 뿌리겠다는 컨트랙트가 있는데 내가 다 받아먹으려고 오라클 제공자가 거짓된 정보를 주입한다면?

Chainlink: 외부 블록체인에 날씨 정보, 주가 정보, 랜덤 엔트로피.. 등을 각각 블록체인에 배포한 컨트랙트로 쏴서 블록에 외부 정보를 주입해주는 별도의 메인넷 블록체인

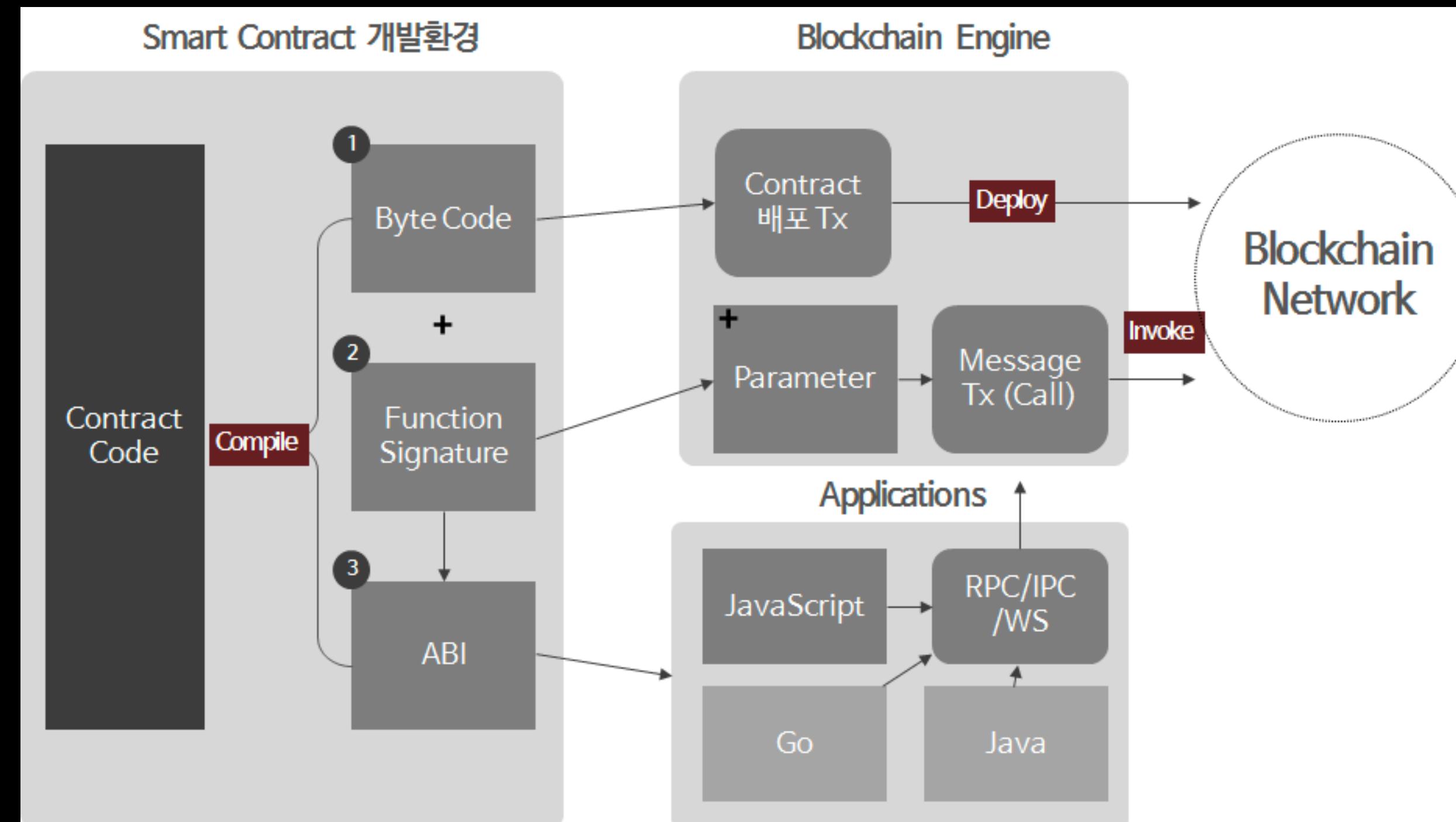


# React.js 프론트엔드 화면 구성하기

- 저한테 메타마스크 계정의 registered/deregistered 된 내용을 콘솔로 보여주세요!
- 아래 링크를 참고하여 코딩해주세요.
- <https://gist.github.com/jyptthemiracle/daf9f5310310b8e00f49077525bc8e9e>
- Tip) Provider는 Wallet 또는 ethers.js, web3.js와 같이 Ethereum 블록체인과 통신할 때 사용하는 API Node를 의미함
- Tip) Signer는 현재 import한 private key의 계정들 여러 개를 의미함. 이를 이용해 내가 “서명” 함으로서 내가 트랜잭션을 보낼 수 있음.

# What is ABI?

- ABI 파일은 주로 artifacts 폴더에 있음
- artifacts/contracts/MyEpicNFT.sol/MyEpicNFT.json
- 스마트 컨트랙트를 컴파일하면 바이트코드와 ABI 파일이 생성된다.
- 프론트엔드 페이지는 컨트랙트 주소와 ABI 파일만 알고 있으면 컨트랙트를 호출할 수 있다.



# 매우 주의할 것

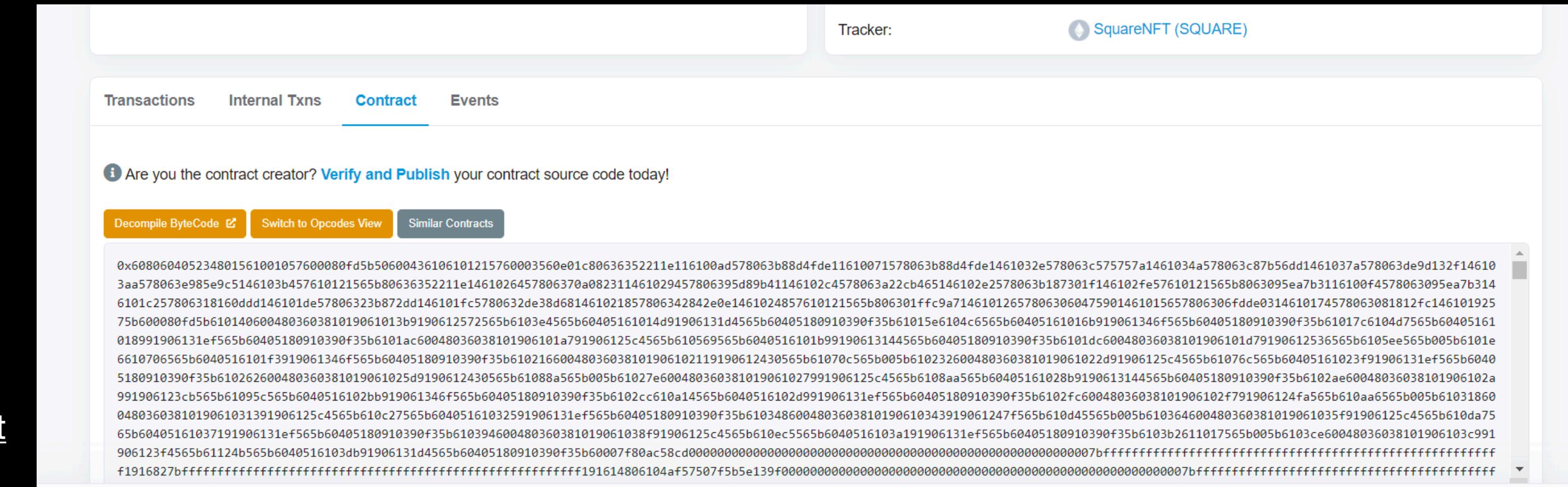
- 컨트랙트에 새로운 변경사항이 있다고 해보자. 그러면 아래 3가지를 동시에 진행해야 한다.
- `deploy.js`를 실행하여 다시 배포해야 한다.
- 프론트엔드 페이지 (`App.js`) 에 다시 배포한 컨트랙트 주소를 업데이트 해야 한다.
- 프론트엔드 페이지 (`App.js`) 에 다시 배포한 ABI 파일을 업데이트 해야 한다.
- 이렇게 해야 하는 이유는? 스마트 컨트랙트는 `immutable` (불변) 하기 때문이다.
- 즉, 컨트랙트를 새로 배포하면 기존의 NFT 데이터를 새 컨트랙트에서는 그대로 사용할 수 없다.

# Final Touches

- OpenSea Testnet에서 확인해보기: 컨트랙트 주소를 입력해서 Collection 클릭
- GitHub에 올릴 때 환경변수 설정하기: dotenv라는 라이브러리 활용하기
- IPFS로 업로드하기: <https://www.pinata.cloud/> 활용하기
- 웹 브라우저에는 [https://cloudflare-ipfs.com/ipfs/INSERT\\_YOUR\\_CID\\_HERE](https://cloudflare-ipfs.com/ipfs/INSERT_YOUR_CID_HERE)
- 컨트랙트에는 \_setTokenURI(itemId, “ipfs://INSERT\_YOUR\_CID\_HERE”)

# Final Touches

<https://docs.allthatnode.com/tutorials/a-simple-erc20-smart-contract>



- Etherscan에 컨트랙트 verify하기: 왜 해야 할까요?
    - 우리가 올린 컨트랙트 코드는 바이트코드 이므로, 사람이 읽을 수 없음
    - 내가 다루고자 하는 이 컨트랙트 또는 구매하고자 하는 NFT에 악의적인 코드가 있을지 없을지 알기 위해서는 Etherscan에 컨트랙트 코드를 올려야 함: verify 안된 컨트랙트는 무조건 스캠 의심할 것
    - Etherscan은 지금 내가 올리고자 하는 컨트랙트 코드와 실제 Ethereum 블록체인에 올라간 바이트 코드를 상호 비교하여 일치할 경우 verify를 찍어줌
    - GitHub를 통해 오픈소스로 컨트랙트 코드를 공개하지만, 실제로 바이트코드와 일치하는지 확인해주는 프론트엔드 페이지 역할을 하는 것이 Etherscan의 역할이다. 우리가 주로 Ethereum 트랜잭션을 보는데 사용되기도 하므로 사용성도 편리하다.

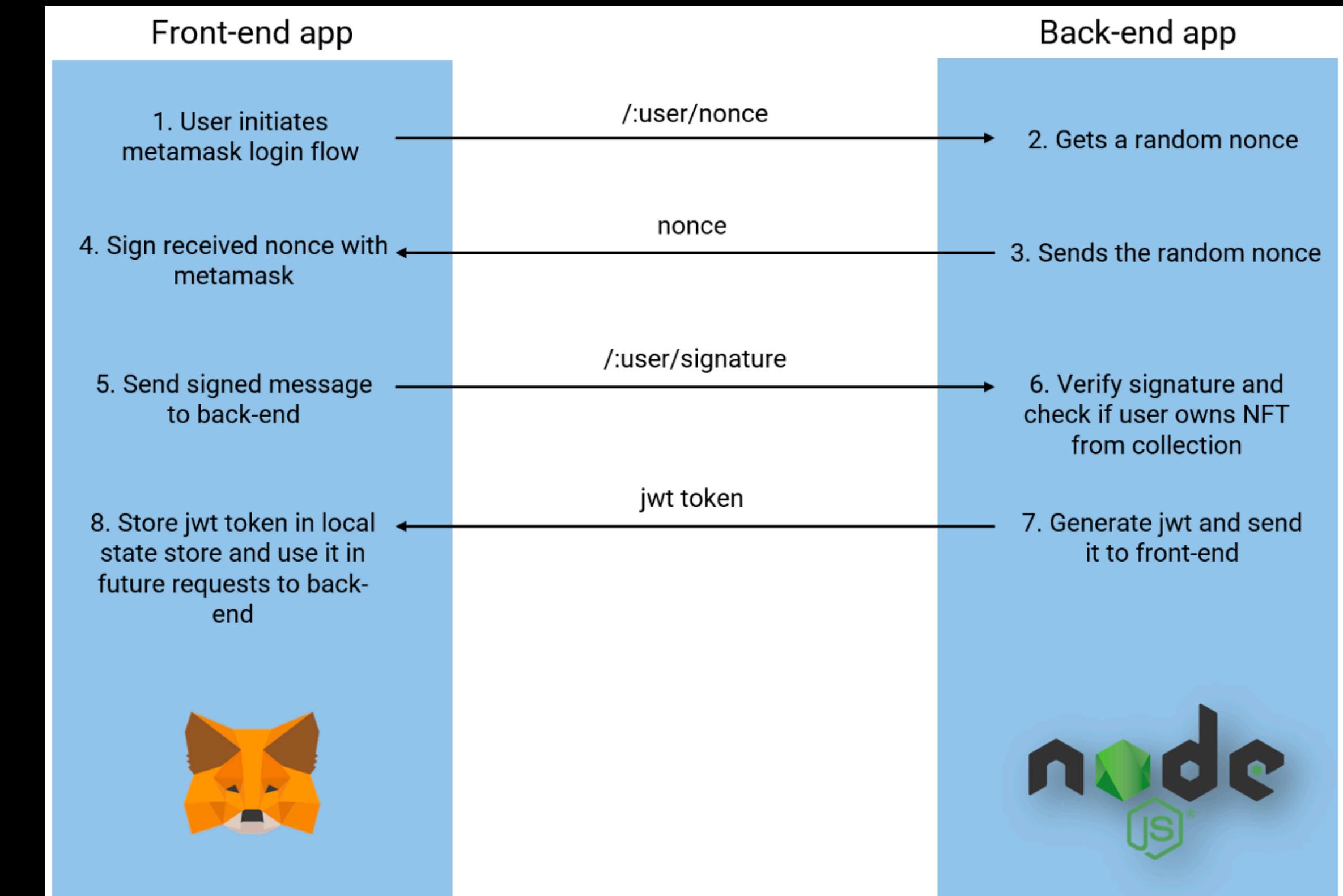
# 응용 가능 사례

NFT를 통한 웹사이트 로그인 기능 구현

collab.land를 활용한 NFT 디스코드 로그인 (NFT 소유자를 위한 Private DAO)

NFT를 활용한 cross-chain 브릿징 및 이를 활용한 담보 대출 등의 DeFi 금융 사례

... what else?



<https://duartefdias.medium.com/nft-login-with-nodejs-metamask-and-opensea-1ae3b49769bc>



<https://bridge.xp.network/>