SuiMover

suimover.org

SuiMover Bootcamp Class#5 - NFT

Wayne Kuo

SuiMover

Agenda

suimover.org

SuiMover

suimover.org

Section 1

Sui Object NFT 基礎概念

常見的 Sui Object NFT

數位收藏

- PFP NFT
- SUI NS
- NFT Mint WL

Defi Position

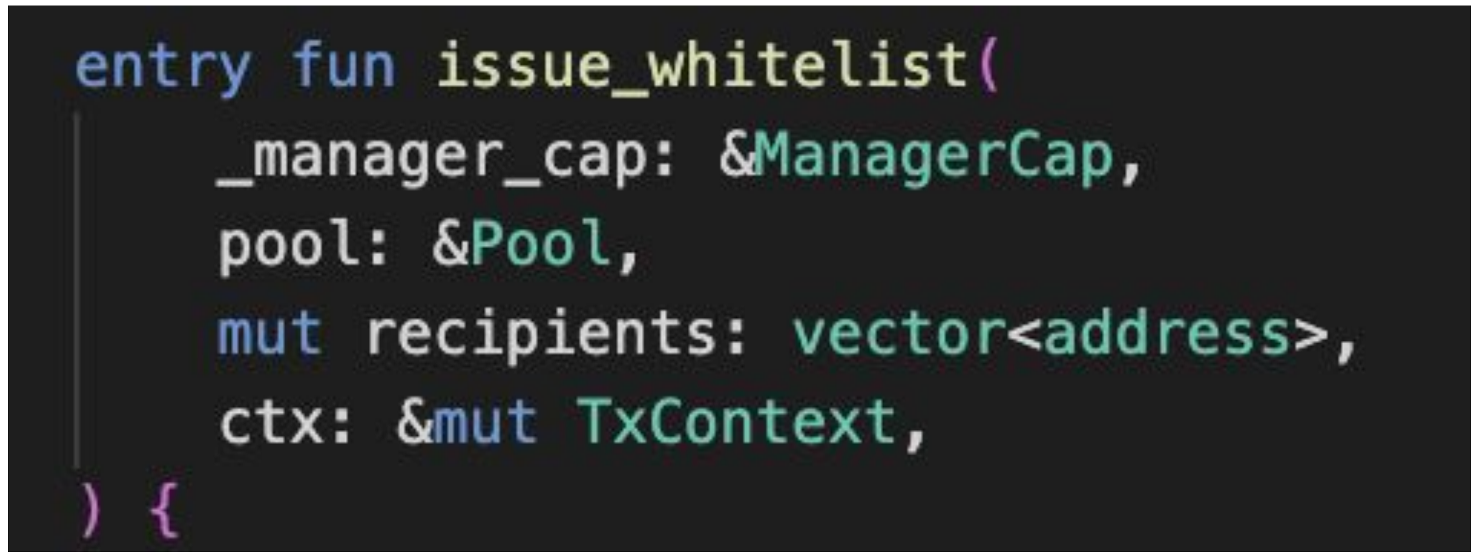
作為一種用戶的帳戶或收據存放

- Typus Deposit/Bid Receipt
- Bucket StakeProof
- Cetus LP
- Suilend

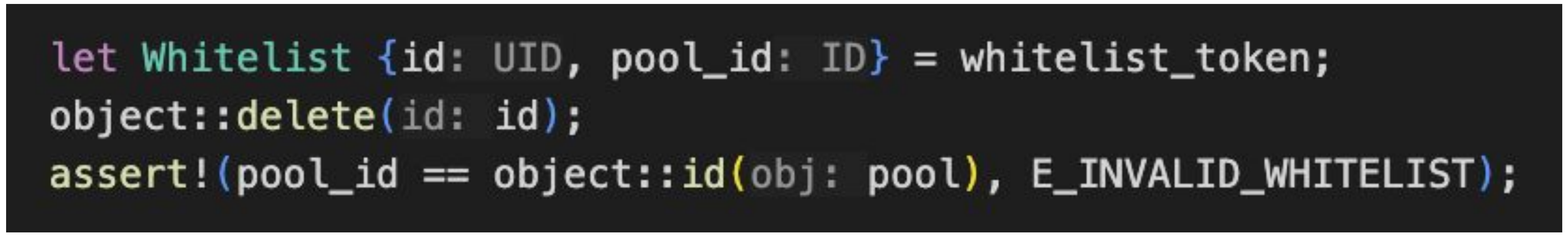
Capability

作為種驗證權限的方法

- UpgradeCap
- Publisher
- ManagerCap



```
entry fun issue_whitelist(
    _manager_cap: &ManagerCap,
    pool: &Pool,
    mut recipients: vector<address>,
    ctx: &mut TxContext,
) {
```



```
let Whitelist {id: UID, pool_id: ID} = whitelist_token;
object::delete(id: id);
assert!(pool_id == object::id(obj: pool), E_INVALID_WHITELIST);
```

Section 1

數位收藏

常見Fields

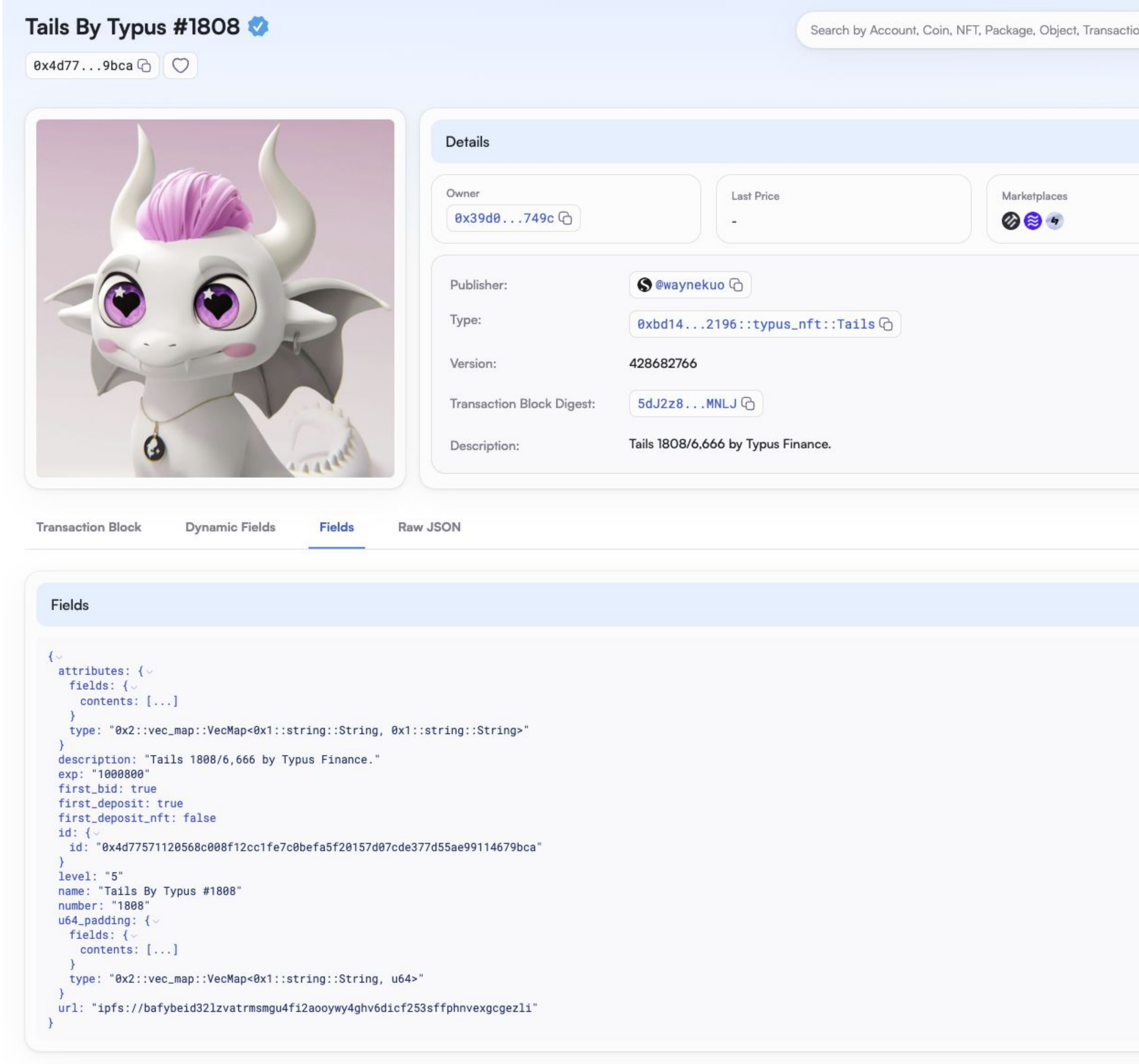
- name：NFT 名稱
- number：唯一的編號
- description：描述介紹
- attributes：各部位特徵

  通常會用這個來評估稀有度
- url：圖片連結

  去中心化的IPFS、中心化的GoogleDrive都可以）

除此之外也可以加一些特別的項目

像level, exp等，作為dynamic NFT 的升級的設計



Tails By Typus #1808

0x4d77...9bca

Search by Account, Coin, NFT, Package, Object, Transactio

Details

Owner	Last Price	Marketplaces
0x39d0...749c	-	

Publisher:	@waynekuo
Type:	0xbd14...2196::typus_nft::Tails
Version:	428682766
Transaction Block Digest:	5dJ2z8...MNLJ
Description:	Tails 1808/6,666 by Typus Finance.

Fields

```
{
  attributes: {
    fields: {
      contents: [...]
    }
    type: "0x2::vec_map::VecMap<0x1::string::String, 0x1::string::String>"
  }
  description: "Tails 1808/6,666 by Typus Finance."
  exp: "1000800"
  first_bid: true
  first_deposit: true
  first_deposit_nft: false
  id: {
    id: "0x4d77571120568c008f12cc1fe7c0befa5f20157d07cde377d55ae99114679bca"
  }
  level: "5"
  name: "Tails By Typus #1808"
  number: "1808"
  u64_padding: {
    fields: {
      contents: [...]
    }
    type: "0x2::vec_map::VecMap<0x1::string::String, u64>"
  }
  url: "ipfs://bafybeid32lzvatrmsmgu4fi2aooywy4ghv6dicf253sffphnvexgcgezli"
}
```

Section 1

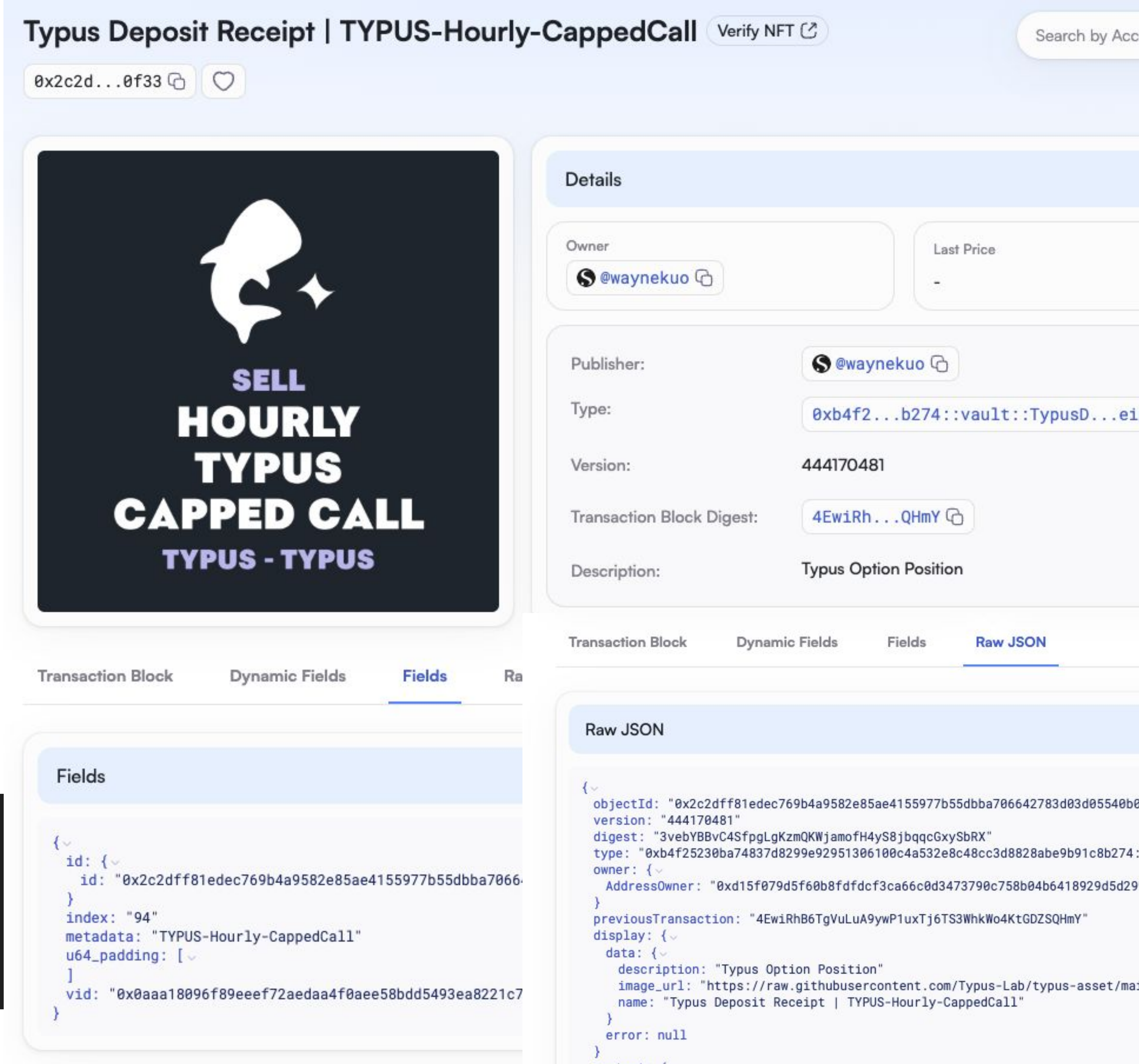
Defi Position

作為各個Defi項目的紀錄用途，會有的Fields就比較多變

以Typus為例：

- index：為存錢的vault的編號
  每個vault都有其對應的編號
- vid：為vault的object id可以直接找到那個vault並用此receipt去查找實際存了多少錢
- metadata：可以跟Display結合使用來改變顯示的名稱

```
public struct Leaderboard has key, store {
    id: UID,
    start_ts_ms: u64,
    end_ts_ms: u64,
    score: Table<address, u64>,
```



Typus Deposit Receipt | TYPUS-Hourly-CappedCall Verify NFT

Search by Acc

0x2c2d...0f33

SELL
HOURLY
TYPUS
CAPPED CALL
TYPUS - TYPUS

Details

Owner
@waynekuo

Last Price
-

Publisher: @waynekuo
Type: 0xb4f2...b274::vault::TypusD...ei
Version: 444170481
Transaction Block Digest: 4EwiRh...QHmY
Description: Typus Option Position

Transaction Block | Dynamic Fields | Fields | Ra

Fields

```
{
  id: {
    id: "0x2c2dff81edec769b4a9582e85ae4155977b55dbba7066
  }
  index: "94"
  metadata: "TYPUS-Hourly-CappedCall"
  u64_padding: [
  ]
  vid: "0x0aaa18096f89eeef72aedaa4f0aee58bdd5493ea8221c7
}
```

Transaction Block | Dynamic Fields | Fields | Raw JSON

Raw JSON

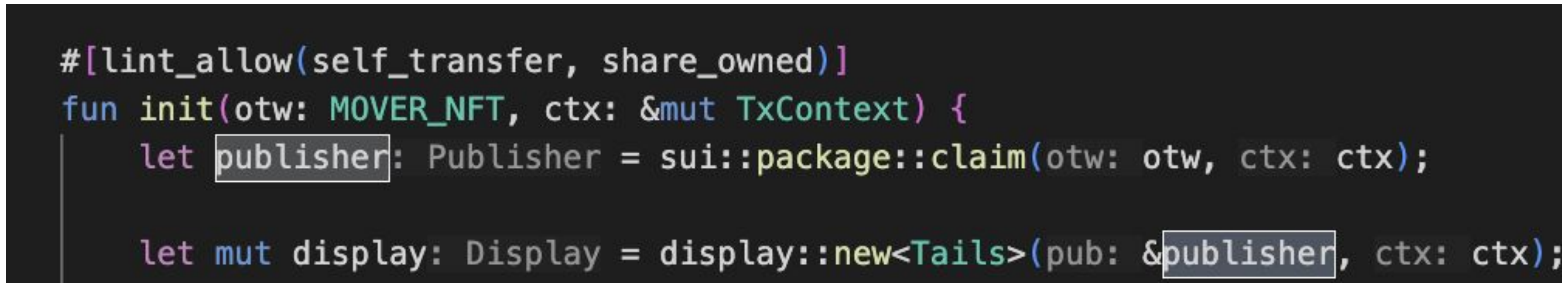
```
{
  objectId: "0x2c2dff81edec769b4a9582e85ae4155977b55dbba706642783d03d05540b0
  version: "444170481"
  digest: "3vebYBBvC4SfpgLgKzmQKWjamofH4yS8jbqqcGxySbRX"
  type: "0xb4f25230ba74837d8299e92951306100c4a532e8c48cc3d8828abe9b91c8b274:
  owner: {
    AddressOwner: "0xd15f079d5f60b8fdfdcf3ca66c0d3473790c758b04b6418929d5d29
  }
  previousTransaction: "4EwiRhB6TgVuLuA9ywP1uxTj6TS3WhkWo4KtGDZSQHmY"
  display: {
    data: {
      description: "Typus Option Position"
      image_url: "https://raw.githubusercontent.com/Typus-Lab/typus-asset/mai
      name: "Typus Deposit Receipt | TYPUS-Hourly-CappedCall"
    }
    error: null
  }
```

Section 1

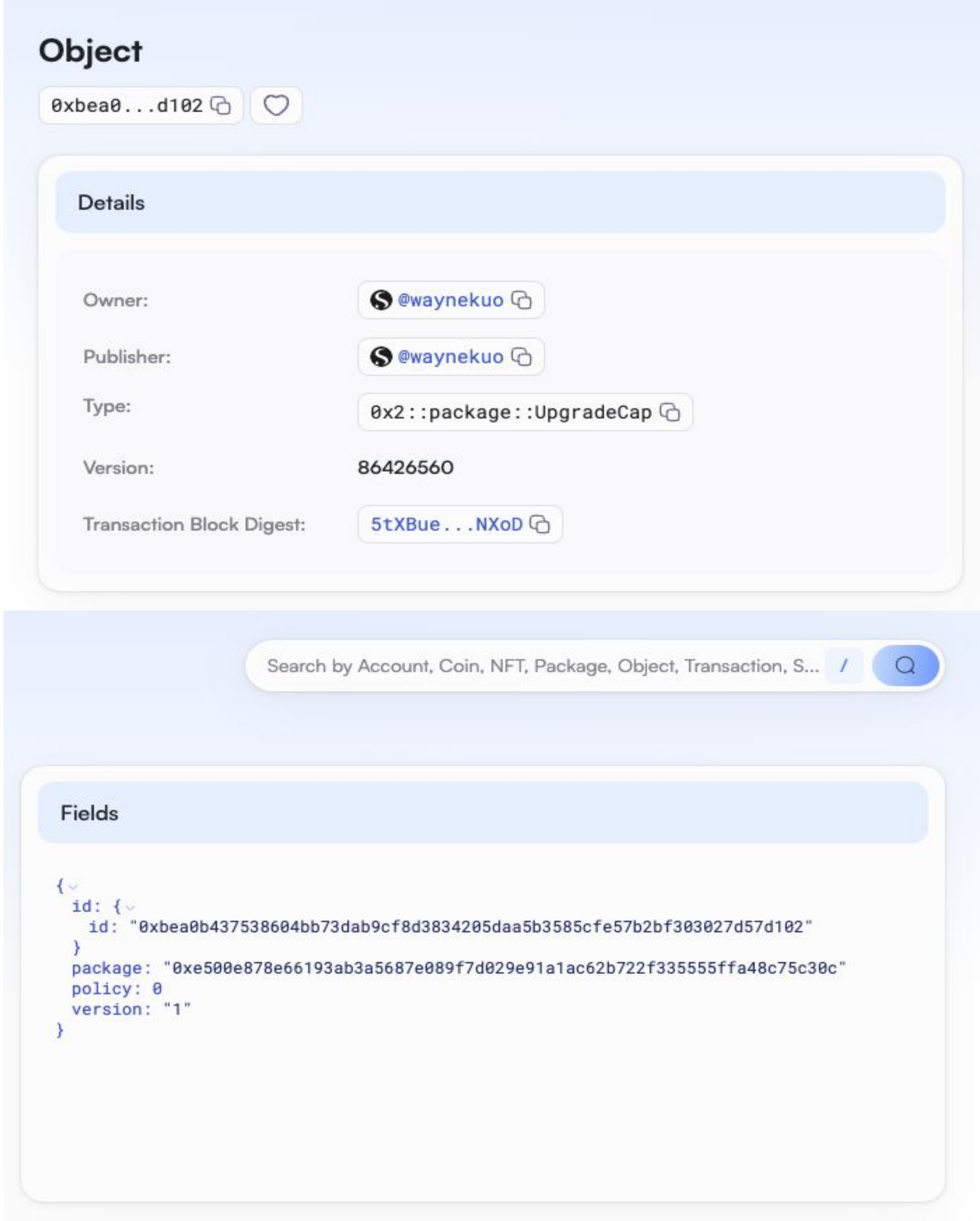
Capability

常見的Cap:

- UpgradeCap：合約升級的時候會用到，如右圖的0xbea0...可以用來升級0xe500...這個合約 ➜ sui client upgrade --upgrade-capability 的0xbea0...
- ManagerCap：可以當作合約的權限去使用，當擁有那個合約的ManagerCap才能夠呼叫特定的管理用function
- Publisher：用來驗證合約的Publisher，只有在第一次publish的時候會生成，在NFT的Display或是Kiosk的TransferPolicy都會需要用到



```
#[lint_allow(self_transfer, share_owned)]
fun init(otw: MOVER_NFT, ctx: &mut TxContext) {
    let publisher: Publisher = sui::package::claim(otw: otw, ctx: ctx);

    let mut display: Display = display::new<Tails>(pub: &publisher, ctx: ctx);
```



Object

0xbea0...d102

Details

Owner:	@waynekuo
Publisher:	@waynekuo
Type:	0x2::package::UpgradeCap
Version:	86426560
Transaction Block Digest:	5tXBue...NXoD

Search by Account, Coin, NFT, Package, Object, Transaction, S...

Fields

```
{
  id: {
    id: "0xbea0b437538604bb73dab9cf8d3834205daa5b3585cfe57b2bf303027d57d102"
  }
  package: "0xe500e878e66193ab3a5687e089f7d029e91a1ac62b722f335555ffa48c75c30c"
  policy: 0
  version: "1"
}
```

SuiMover

suimover.org

Section 2

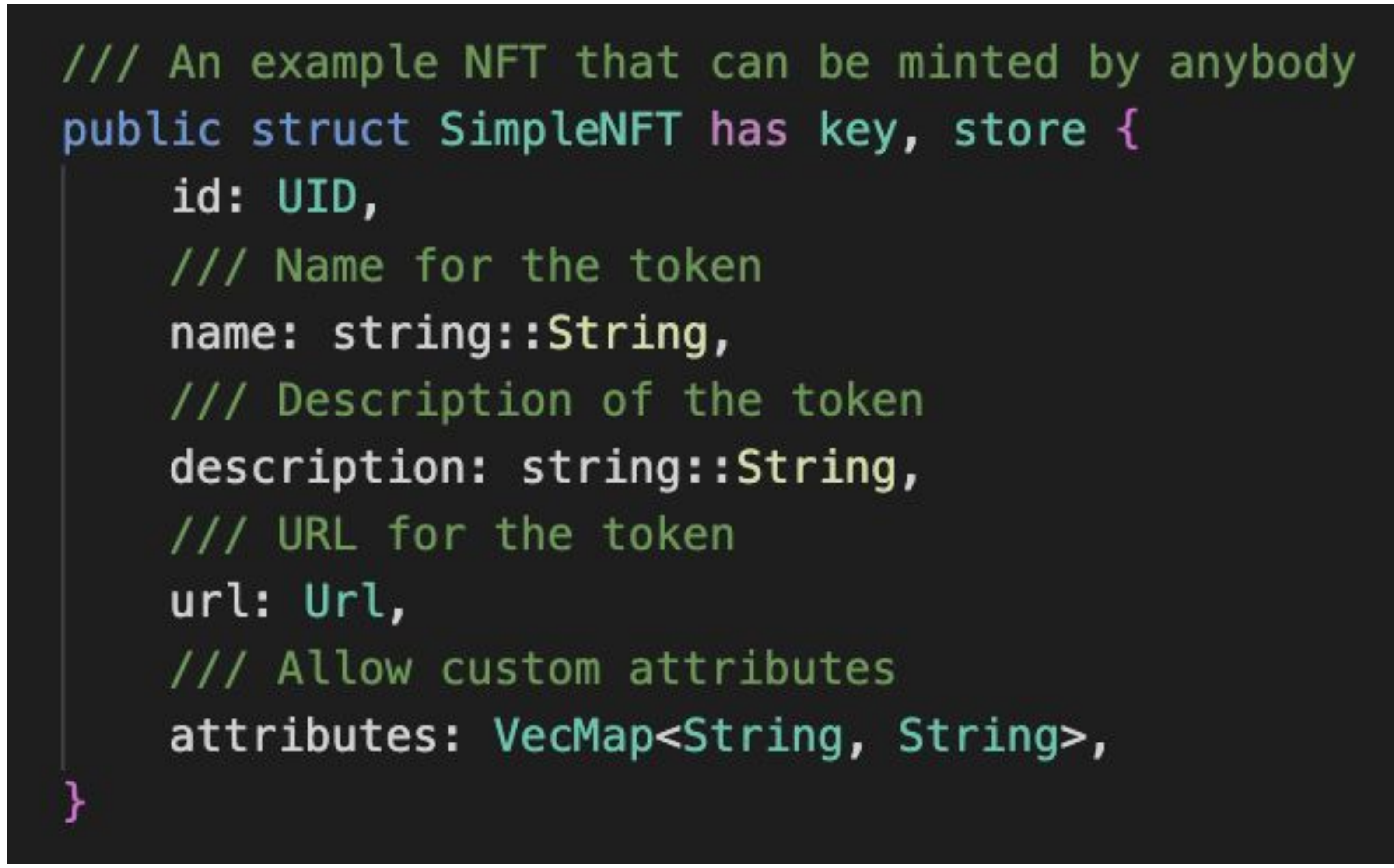
Sui Object NFT 程式碼

Section 2

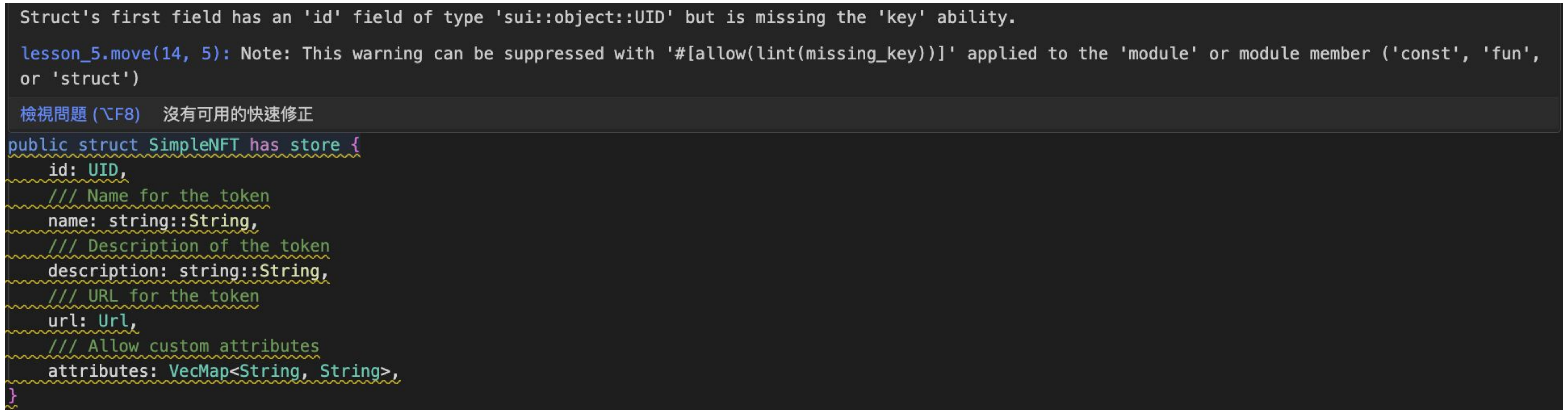
Object Struct

- **has key** - Globally unique IDs that define an object's ID in storage. Any Sui Object, that is a struct with the `key` ability, must have `id: UID` as its first field.

  有key就要有UID，但有UID不一定要有key
- **has store** - 才能被透過**此合約**以外的方式轉移



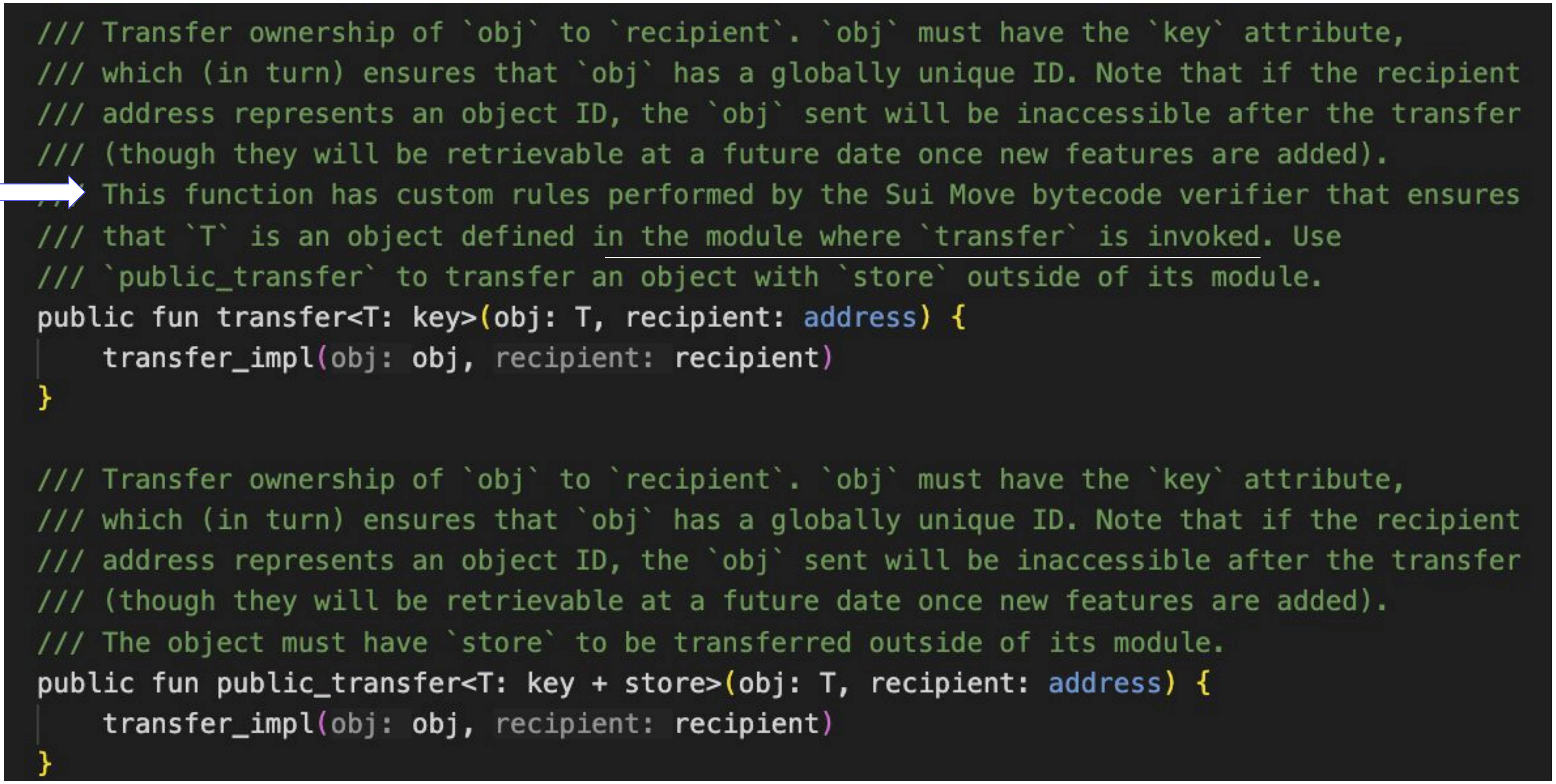
```
/// An example NFT that can be minted by anybody
public struct SimpleNFT has key, store {
    id: UID,
    /// Name for the token
    name: string::String,
    /// Description of the token
    description: string::String,
    /// URL for the token
    url: Url,
    /// Allow custom attributes
    attributes: VecMap<String, String>,
}
```



```
Struct's first field has an 'id' field of type 'sui::object::UID' but is missing the 'key' ability.

lesson_5.move(14, 5): Note: This warning can be suppressed with '#[allow(lint(missing_key))]' applied to the 'module' or module member ('const', 'fun', or 'struct')
```

```
public struct SimpleNFT has store {
    id: UID,
    /// Name for the token
    name: string::String,
    /// Description of the token
    description: string::String,
    /// URL for the token
    url: Url,
    /// Allow custom attributes
    attributes: VecMap<String, String>,
}
```

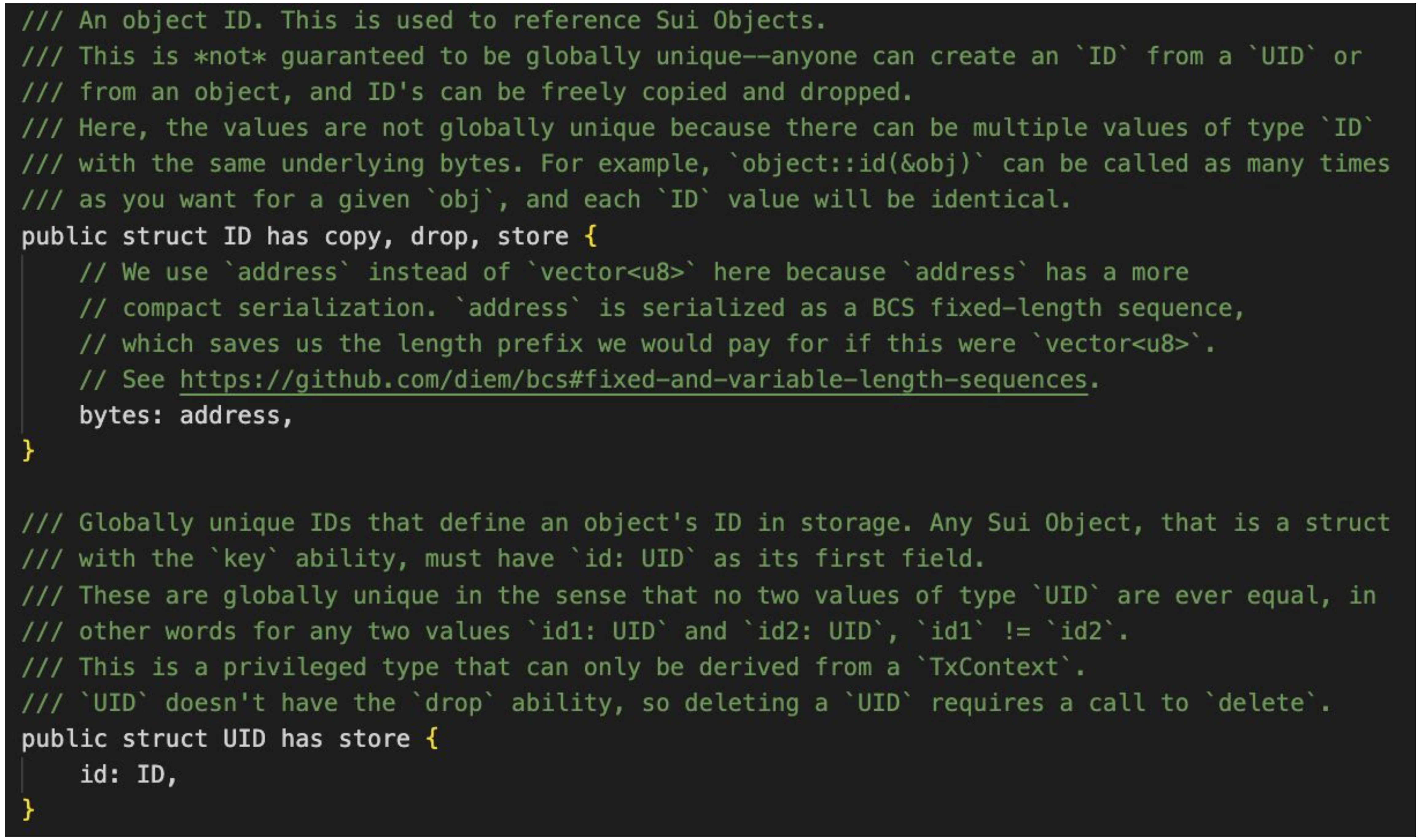
transfer vs public_transfer



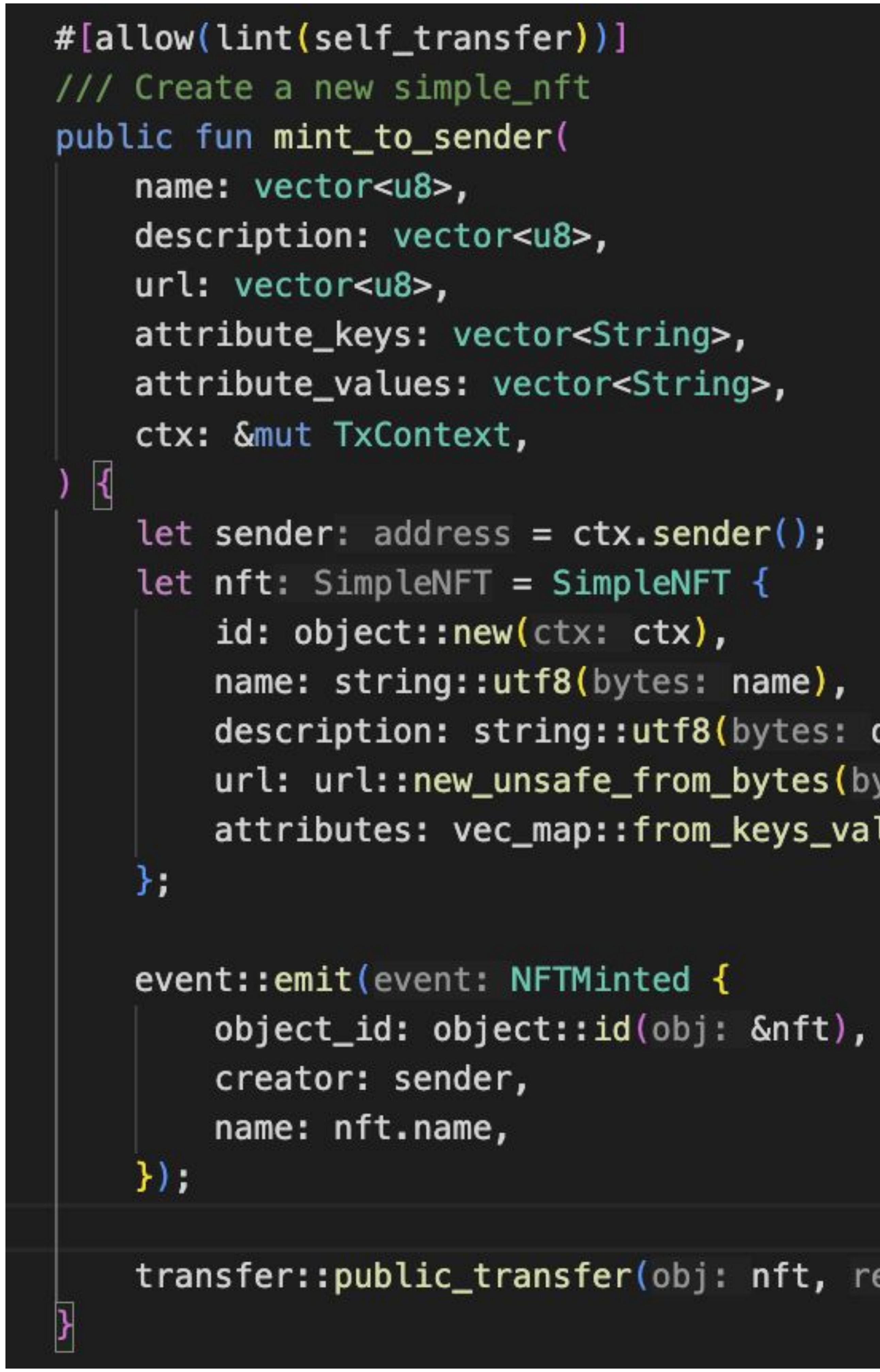
```
/// Transfer ownership of `obj` to `recipient`. `obj` must have the `key` attribute,
/// which (in turn) ensures that `obj` has a globally unique ID. Note that if the recipient
/// address represents an object ID, the `obj` sent will be inaccessible after the transfer
/// (though they will be retrievable at a future date once new features are added).
/// This function has custom rules performed by the Sui Move bytecode verifier that ensures
/// that `T` is an object defined in the module where `transfer` is invoked. Use
/// `public_transfer` to transfer an object with `store` outside of its module.
public fun transfer<T: key>(obj: T, recipient: address) {
    transfer_impl(obj: obj, recipient: recipient)
}

/// Transfer ownership of `obj` to `recipient`. `obj` must have the `key` attribute,
/// which (in turn) ensures that `obj` has a globally unique ID. Note that if the recipient
/// address represents an object ID, the `obj` sent will be inaccessible after the transfer
/// (though they will be retrievable at a future date once new features are added).
/// The object must have `store` to be transferred outside of its module.
public fun public_transfer<T: key + store>(obj: T, recipient: address) {
    transfer_impl(obj: obj, recipient: recipient)
}
```

Section 2

UID 唯一且不能被複製的 vs ID 可以複製



```
/// An object ID. This is used to reference Sui Objects.
/// This is *not* guaranteed to be globally unique--anyone can create an `ID` from a `UID` or
/// from an object, and ID's can be freely copied and dropped.
/// Here, the values are not globally unique because there can be multiple values of type `ID`
/// with the same underlying bytes. For example, `object::id(&obj)` can be called as many times
/// as you want for a given `obj`, and each `ID` value will be identical.
public struct ID has copy, drop, store {
    // We use `address` instead of `vector<u8>` here because `address` has a more
    // compact serialization. `address` is serialized as a BCS fixed-length sequence,
    // which saves us the length prefix we would pay for if this were `vector<u8>`.
    // See https://github.com/diem/bcs#fixed-and-variable-length-sequences.
    bytes: address,
}

/// Globally unique IDs that define an object's ID in storage. Any Sui Object, that is a struct
/// with the `key` ability, must have `id: UID` as its first field.
/// These are globally unique in the sense that no two values of type `UID` are ever equal, in
/// other words for any two values `id1: UID` and `id2: UID`, `id1` != `id2`.
/// This is a privileged type that can only be derived from a `TxContext`.
/// `UID` doesn't have the `drop` ability, so deleting a `UID` requires a call to `delete`.
public struct UID has store {
    id: ID,
}
```



```
#[allow(lint(self_transfer))]
/// Create a new simple_nft
public fun mint_to_sender(
    name: vector<u8>,
    description: vector<u8>,
    url: vector<u8>,
    attribute_keys: vector<String>,
    attribute_values: vector<String>,
    ctx: &mut TxContext,
) {
    let sender: address = ctx.sender();
    let nft: SimpleNFT = SimpleNFT {
        id: object::new(ctx: ctx),
        name: string::utf8(bytes: name),
        description: string::utf8(bytes: d
        url: url::new_unsafe_from_bytes(by
        attributes: vec_map::from_keys_val
    };

    event::emit(event: NFTMinted {
        object_id: object::id(obj: &nft),
        creator: sender,
        name: nft.name,
    });

    transfer::public_transfer(obj: nft, re
}
```

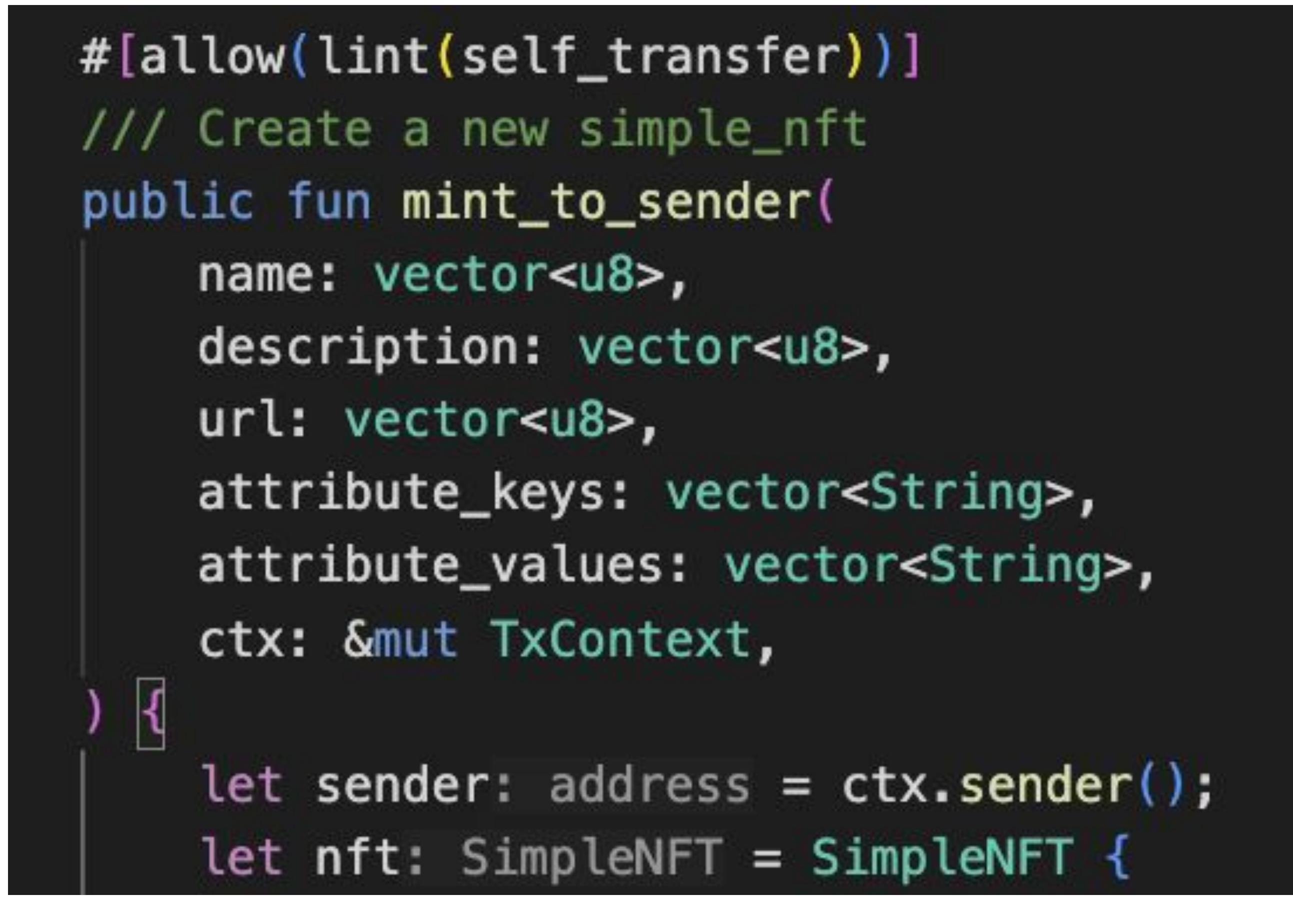
Section 2

Exercise 5: Mint Your Simple NFT

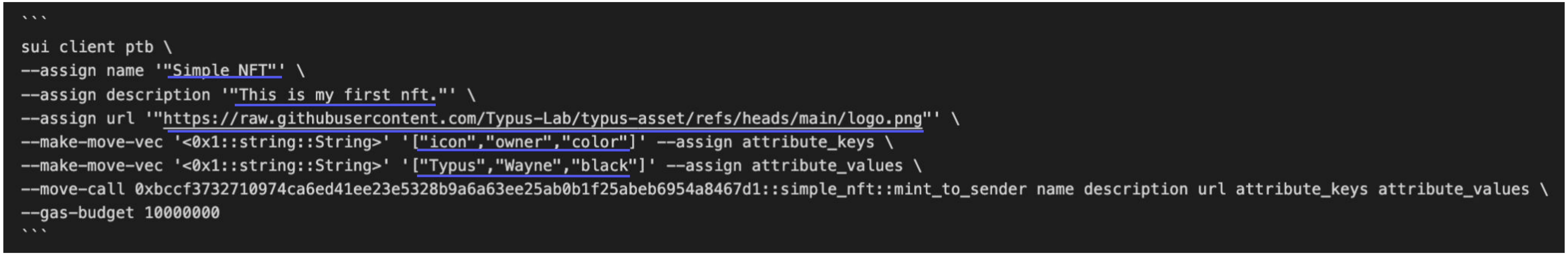
修改 name, description, url, attributes 並執行

mint 一個你專屬的NFT

https://github.com/SuiMover/Sui-Mover-2024-2/tree/main/Lesson5

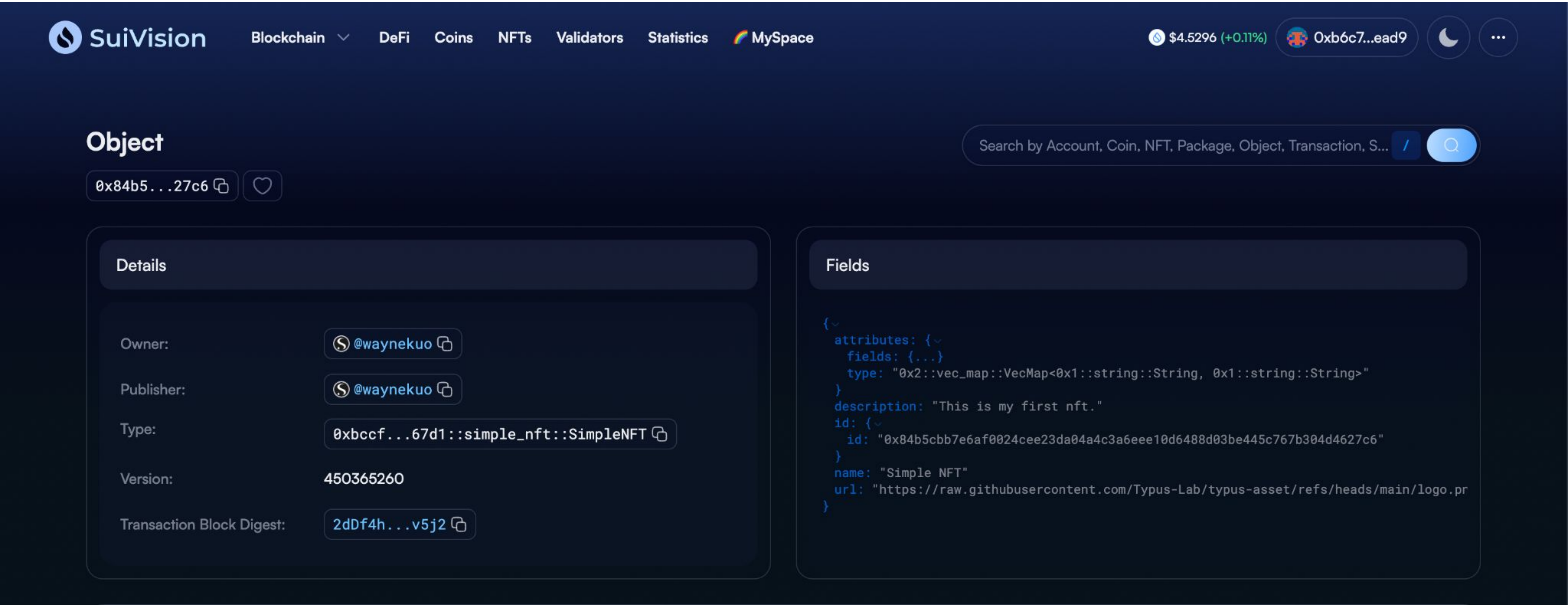


```
#[allow(lint(self_transfer))]
/// Create a new simple_nft
public fun mint_to_sender(
    name: vector<u8>,
    description: vector<u8>,
    url: vector<u8>,
    attribute_keys: vector<String>,
    attribute_values: vector<String>,
    ctx: &mut TxContext,
) {
    let sender: address = ctx.sender();
    let nft: SimpleNFT = SimpleNFT {
```



~~~
```
sui client ptb \
--assign name '"Simple NFT"' \
--assign description '"This is my first nft."' \
--assign url '"https://raw.githubusercontent.com/Typus-Lab/typus-asset/refs/heads/main/logo.png"' \
--make-move-vec '<0x1::string::String>' '["icon","owner","color"]' --assign attribute_keys \
--make-move-vec '<0x1::string::String>' '["Typus","Wayne","black"]' --assign attribute_values \
--move-call 0xbccf3732710974ca6ed41ee23e5328b9a6a63ee25ab0b1f25abeb6954a8467d1::simple_nft::mint_to_sender name description url attribute_keys attribute_values \
--gas-budget 10000000
```
~~~

Section 2



Object

0x84b5...27c6

Details

Owner: @waynekuo

Publisher: @waynekuo

Type: 0xbccf...67d1::simple_nft::SimpleNFT

Version: 450365260

Transaction Block Digest: 2dDf4h...v5j2

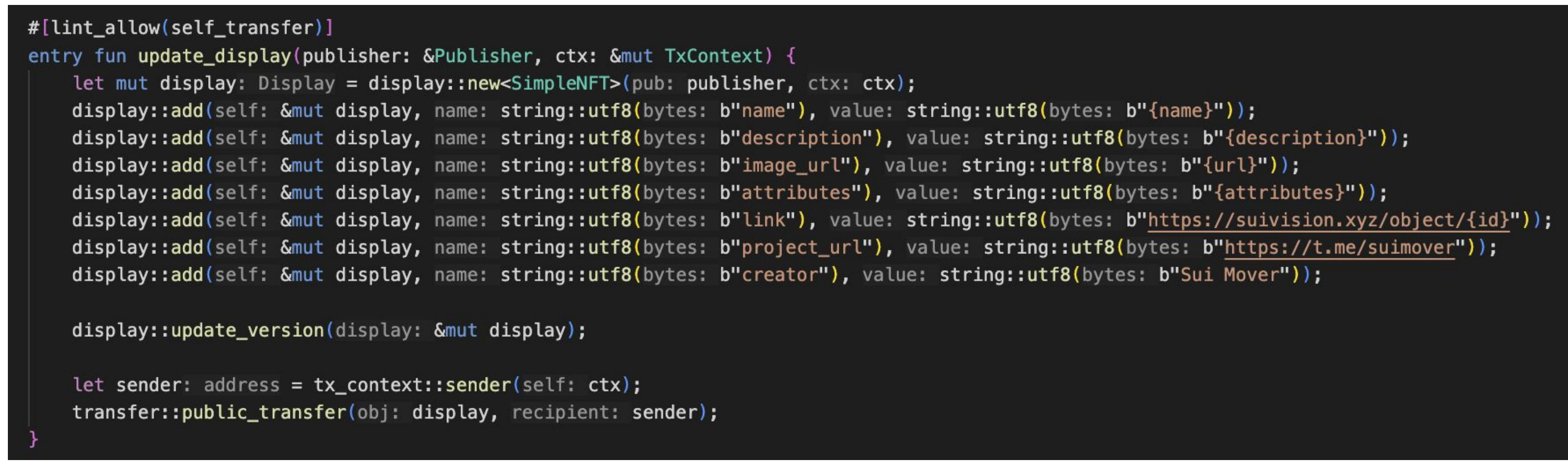
Fields

```
{
  attributes: {
    fields: {...}
    type: "0x2::vec_map::VecMap<0x1::string::String, 0x1::string::String>"
  }
  description: "This is my first nft."
  id: {
    id: "0x84b5cbb7e6af0024cee23da04a4c3a6eee10d6488d03be445c767b304d4627c6"
  }
  name: "Simple NFT"
  url: "https://raw.githubusercontent.com/Typus-Lab/typus-asset/refs/heads/main/logo.pr
}
```

Section 2

Display

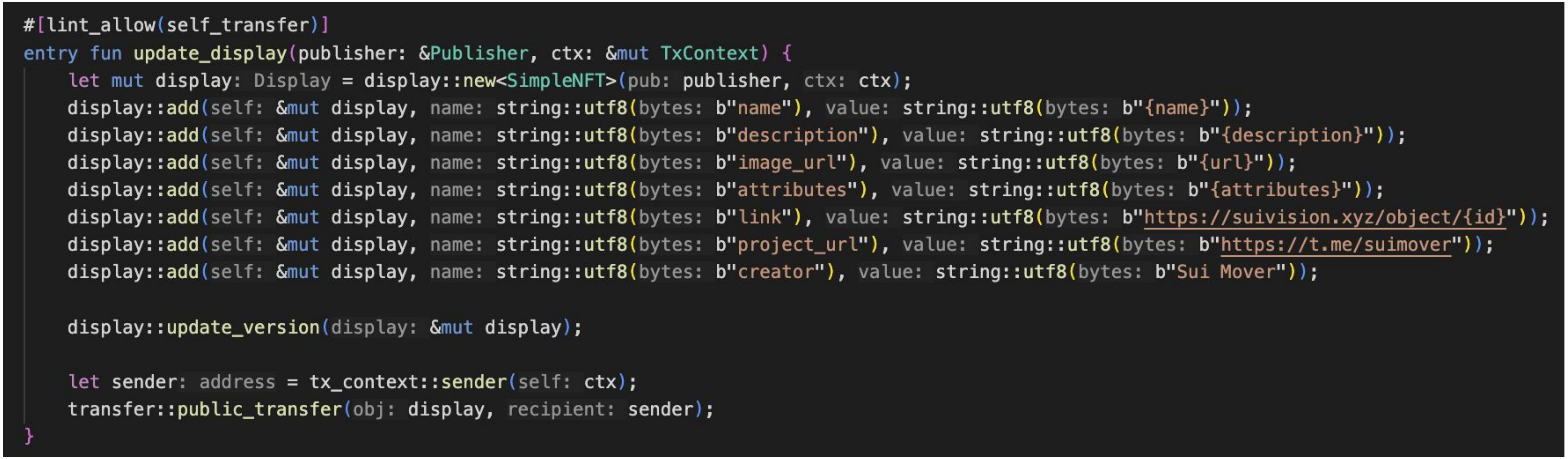
- **name** - A name for the object. The name is displayed when users view the object.
- **description** - A description for the object. The description is displayed when users view the object.
- **link** - A link to the object to use in an application.
- **image_url** - A URL or a blob with the image for the object.
- **thumbnail_url** - A URL to a smaller image to use in wallets, explorers, and other products as a preview.
- **project_url** - A link to a website associated with the object or creator.
- **creator** - A string that indicates the object creator.



```
#[lint_allow(self_transfer)]
entry fun update_display(publisher: &Publisher, ctx: &mut TxContext) {
    let mut display: Display = display::new<SimpleNFT>(pub: publisher, ctx: ctx);
    display::add(self: &mut display, name: string::utf8(bytes: b"name"), value: string::utf8(bytes: b"{name}"));
    display::add(self: &mut display, name: string::utf8(bytes: b"description"), value: string::utf8(bytes: b"{description}"));
    display::add(self: &mut display, name: string::utf8(bytes: b"image_url"), value: string::utf8(bytes: b"{url}"));
    display::add(self: &mut display, name: string::utf8(bytes: b"attributes"), value: string::utf8(bytes: b"{attributes}"));
    display::add(self: &mut display, name: string::utf8(bytes: b"link"), value: string::utf8(bytes: b"https://suivision.xyz/object/{id}"));
    display::add(self: &mut display, name: string::utf8(bytes: b"project_url"), value: string::utf8(bytes: b"https://t.me/suimover"));
    display::add(self: &mut display, name: string::utf8(bytes: b"creator"), value: string::utf8(bytes: b"Sui Mover"));

    display::update_version(display: &mut display);

    let sender: address = tx_context::sender(self: ctx);
    transfer::public_transfer(obj: display, recipient: sender);
}
```

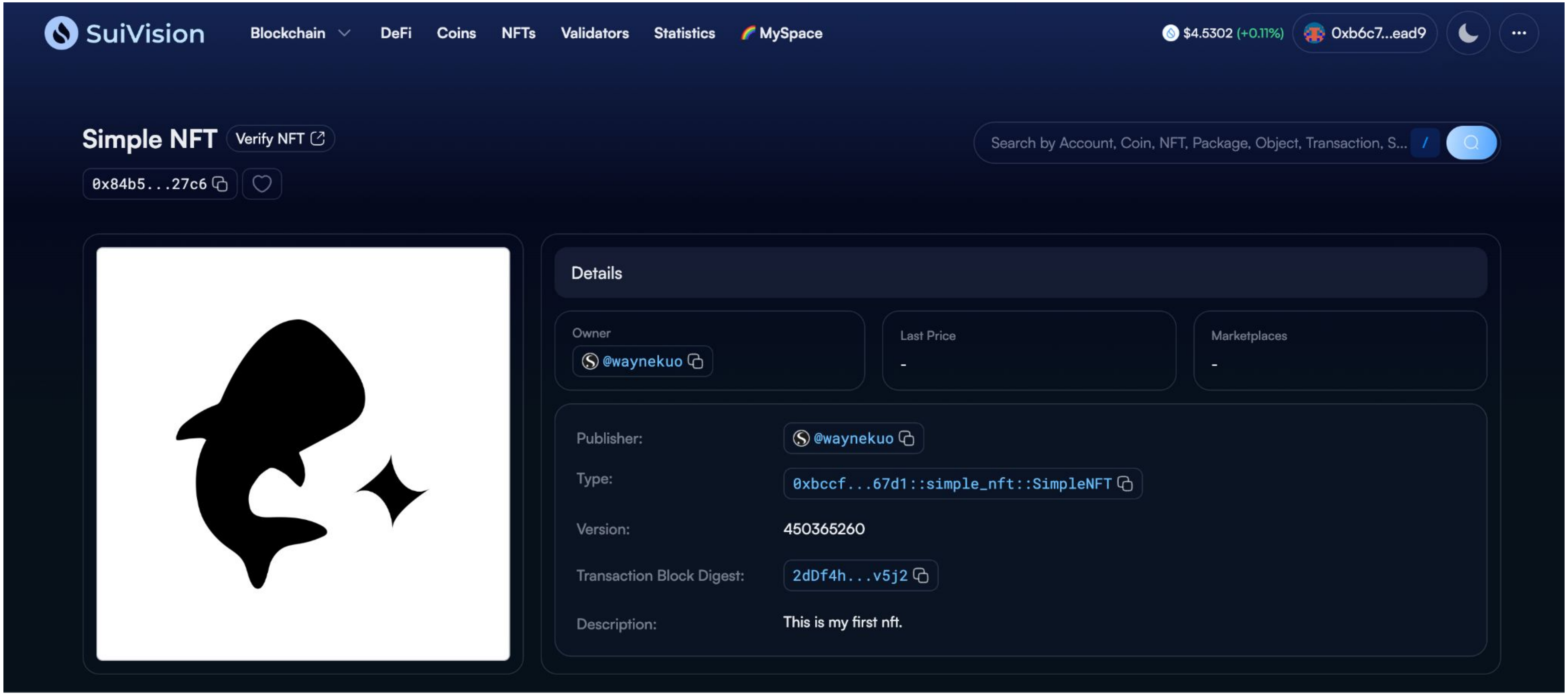
Section 2

Display



```
#[lint_allow(self_transfer)]
entry fun update_display(publisher: &Publisher, ctx: &mut TxContext) {
    let mut display: Display = display::new<SimpleNFT>(pub: publisher, ctx: ctx);
    display::add(self: &mut display, name: string::utf8(bytes: b"name"), value: string::utf8(bytes: b"{name}"));
    display::add(self: &mut display, name: string::utf8(bytes: b"description"), value: string::utf8(bytes: b"{description}"));
    display::add(self: &mut display, name: string::utf8(bytes: b"image_url"), value: string::utf8(bytes: b"{url}"));
    display::add(self: &mut display, name: string::utf8(bytes: b"attributes"), value: string::utf8(bytes: b"{attributes}"));
    display::add(self: &mut display, name: string::utf8(bytes: b"link"), value: string::utf8(bytes: b"https://suivision.xyz/object/{id}"));
    display::add(self: &mut display, name: string::utf8(bytes: b"project_url"), value: string::utf8(bytes: b"https://t.me/suimover"));
    display::add(self: &mut display, name: string::utf8(bytes: b"creator"), value: string::utf8(bytes: b"Sui Mover"));

    display::update_version(display: &mut display);

    let sender: address = tx_context::sender(self: ctx);
    transfer::public_transfer(obj: display, recipient: sender);
}
```

Section 2 SuiVision



Simple NFT

0x84b5...27c6

Details

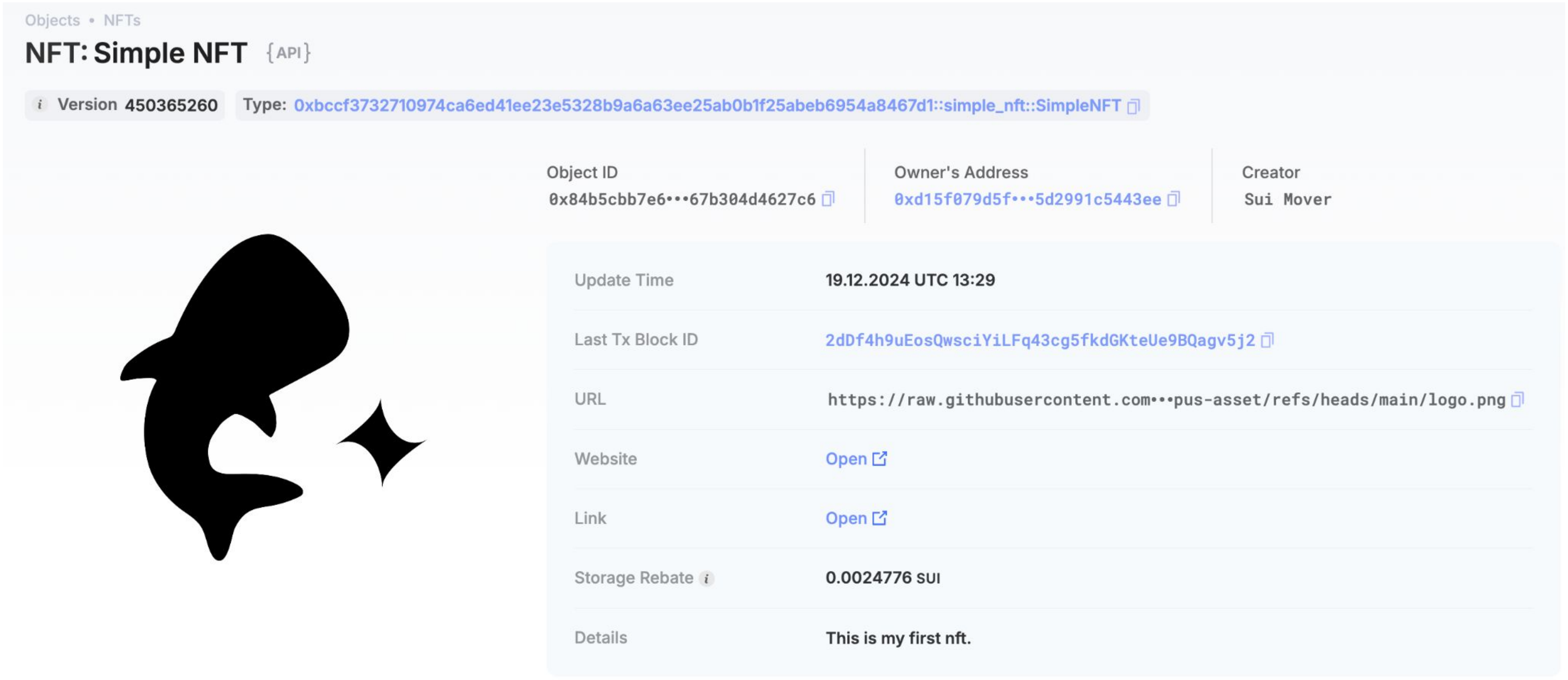
Owner
@waynekuo

Last Price
-

Marketplaces
-

Publisher:	@waynekuo
Type:	0xbccf...67d1::simple_nft::SimpleNFT
Version:	450365260
Transaction Block Digest:	2dDf4h...v5j2
Description:	This is my first nft.

Section 2 suiscan



Objects • NFTs

NFT: Simple NFT {API}

Version 450365260 Type: 0xbccf3732710974ca6ed41ee23e5328b9a6a63ee25ab0b1f25abeb6954a8467d1::simple_nft::SimpleNFT

Object ID	Owner's Address	Creator
0x84b5cbb7e6•••67b304d4627c6	0xd15f079d5f•••5d2991c5443ee	Sui Mover

Update Time	19.12.2024 UTC 13:29
Last Tx Block ID	2dDf4h9uEosQwsciYiLFq43cg5fkdGKteUe9BQagv5j2
URL	https://raw.githubusercontent.com•••pus-asset/refs/heads/main/logo.png
Website	Open
Link	Open
Storage Rebate	0.0024776 SUI
Details	This is my first nft.

SuiMover

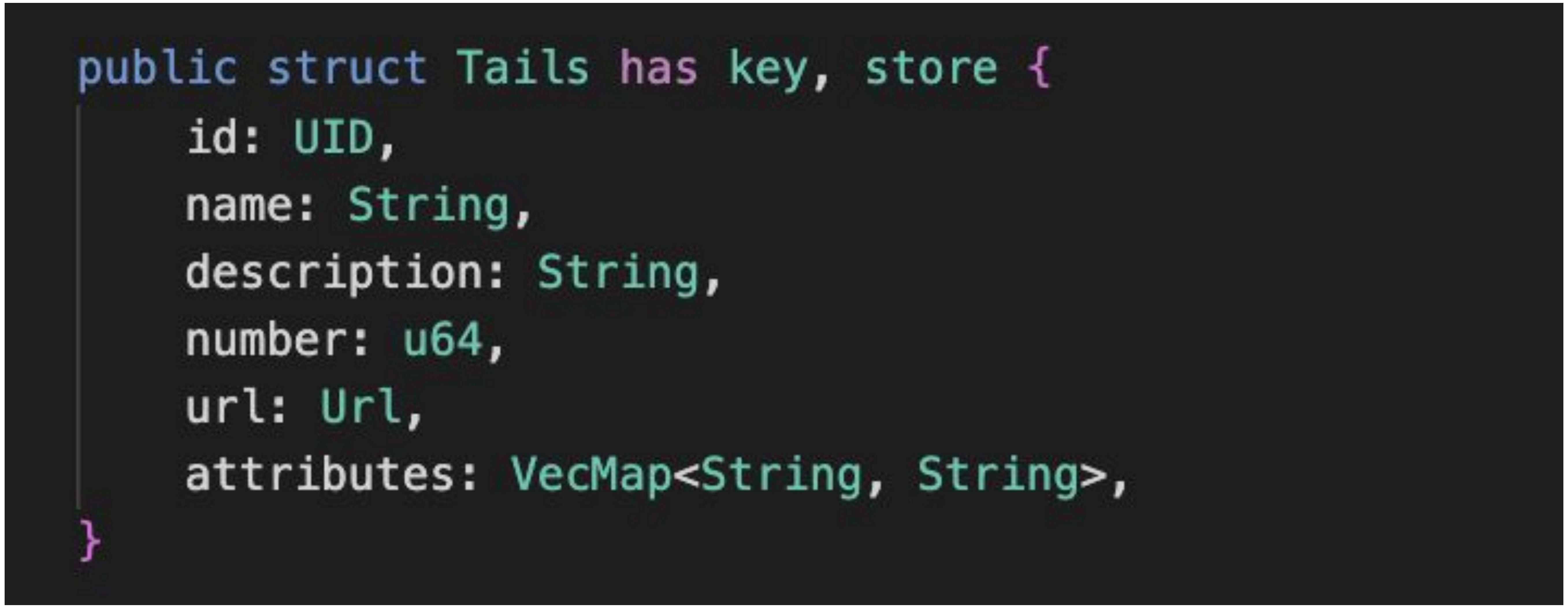
suimover.org

Section 3

NFT Project Example

Section 3

Define NFT struct

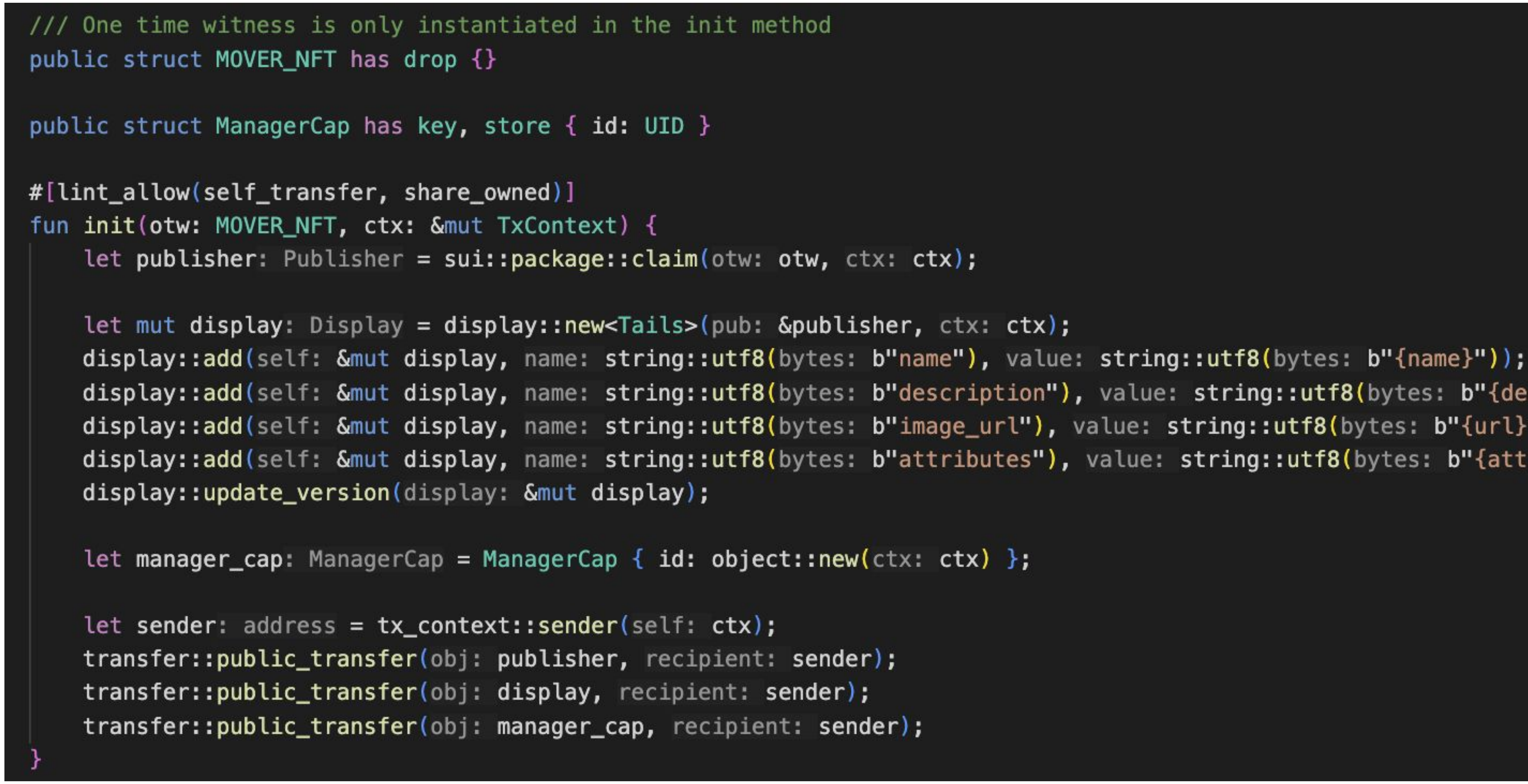

```
public struct Tails has key, store {
    id: UID,
    name: String,
    description: String,
    number: u64,
    url: Url,
    attributes: VecMap<String, String>,
}
```

Section 3

init

Key Points:

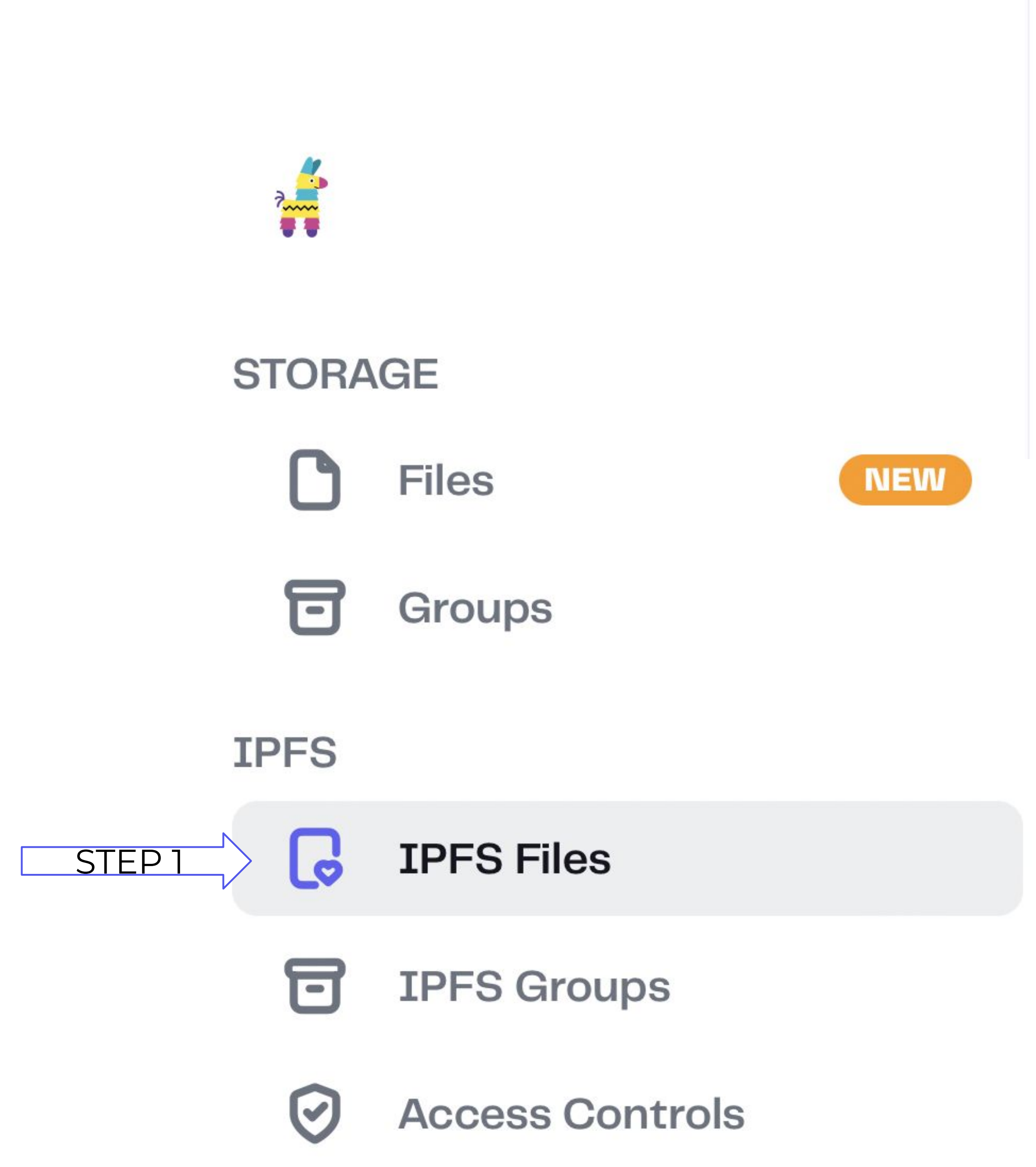
- one time witness
- publisher
- display
- ManagerCap

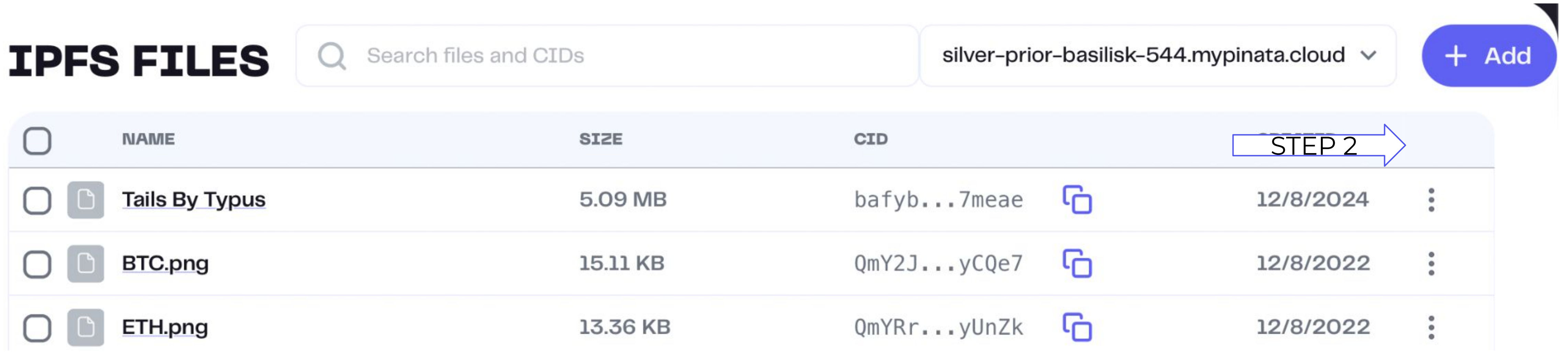


```
/// One time witness is only instantiated in the init method
public struct MOVER_NFT has drop {}

public struct ManagerCap has key, store { id: UID }

#[lint_allow(self_transfer, share_owned)]
fun init(otw: MOVER_NFT, ctx: &mut TxContext) {
    let publisher: Publisher = sui::package::claim(otw: otw, ctx: ctx);

    let mut display: Display = display::new<Tails>(pub: &publisher, ctx: ctx);
    display::add(self: &mut display, name: string::utf8(bytes: b"name"), value: string::utf8(bytes: b"{name}"));
    display::add(self: &mut display, name: string::utf8(bytes: b"description"), value: string::utf8(bytes: b"{de
    display::add(self: &mut display, name: string::utf8(bytes: b"image_url"), value: string::utf8(bytes: b"{url}
    display::add(self: &mut display, name: string::utf8(bytes: b"attributes"), value: string::utf8(bytes: b"{att
    display::update_version(display: &mut display);

    let manager_cap: ManagerCap = ManagerCap { id: object::new(ctx: ctx) };

    let sender: address = tx_context::sender(self: ctx);
    transfer::public_transfer(obj: publisher, recipient: sender);
    transfer::public_transfer(obj: display, recipient: sender);
    transfer::public_transfer(obj: manager_cap, recipient: sender);
}
```

Section 3

Upload Image on IPFS





NAME	SIZE	CID	
Tails By Typus	5.09 MB	bafyb...7meae	12/8/2024
BTC.png	15.11 KB	QmY2J...yCQe7	12/8/2022
ETH.png	13.36 KB	QmYRr...yUnZk	12/8/2022

步驟

1. 到Pinata註冊 https://app.pinata.cloud/ipfs/files
2. IPFS Files Add 上傳文件或數據
3. 獲取 CID 訪問文件

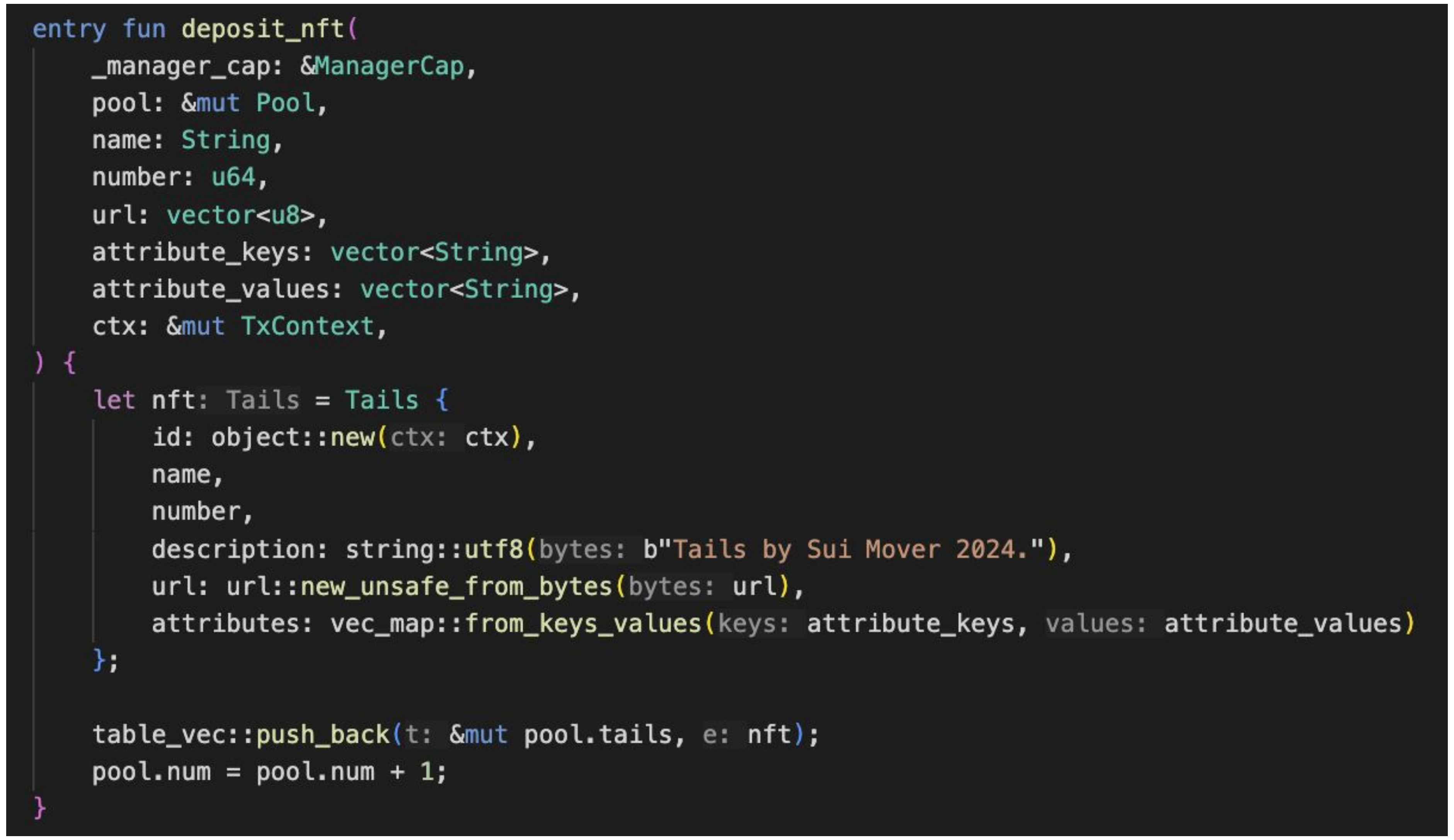
https://silver-prior-basilisk-544.mypinata.cloud/ipfs/bafybeietehky3bx4jwj7kw5kd7vk5ss7swisw2gb6ixxbspukdoh47meae

ipfs://bafybeietehky3bx4jwj7kw5kd7vk5ss7swisw2gb6ixxbspukdoh47meae

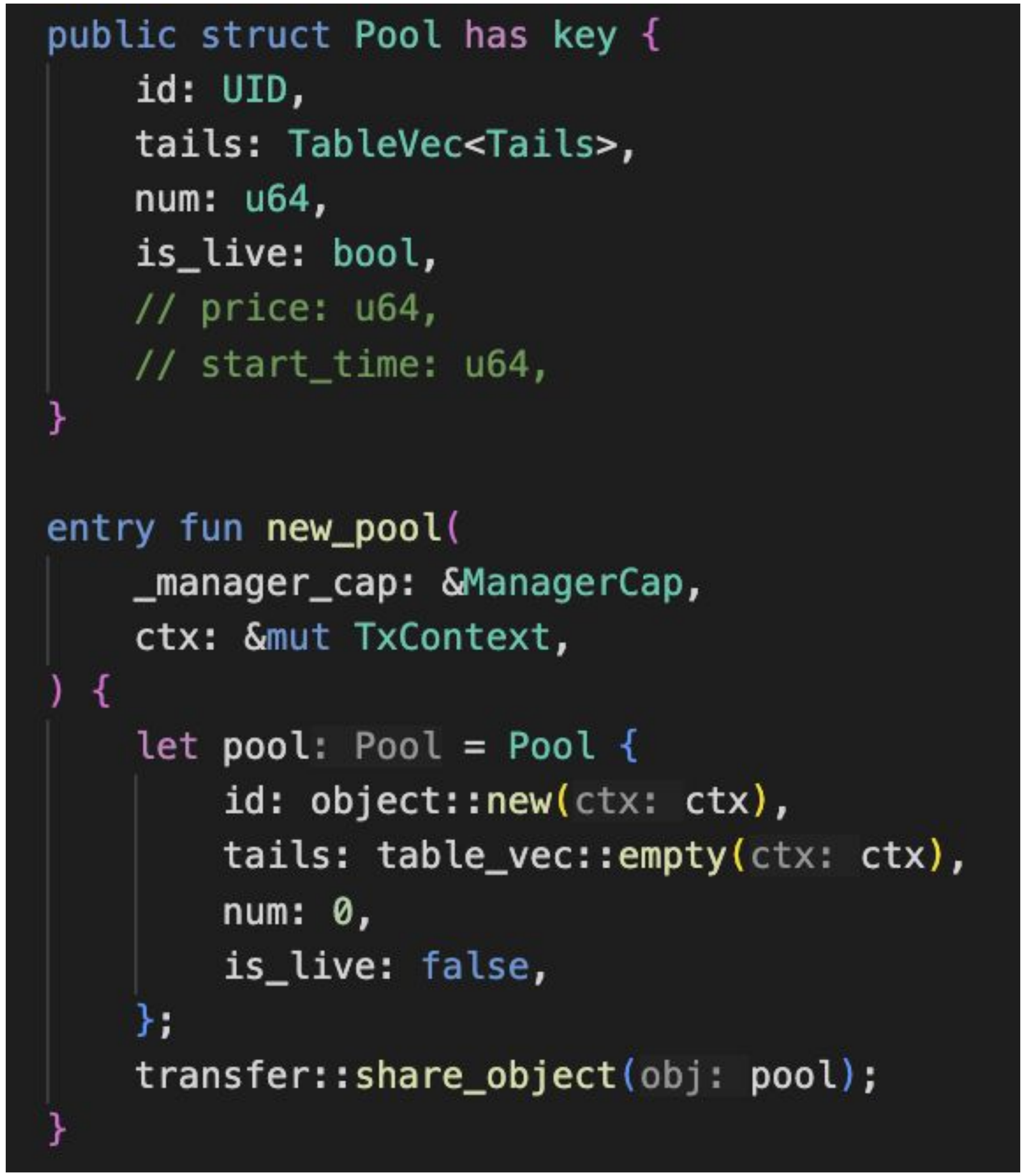
candy machine (nft pool)

Key Points:

- TableVec
- ManagerCap
- url ipfs://bafybeietehky3bx4jwj7kw5kd7vk5ss7swisw2gb6ixxbspukdoh47meae



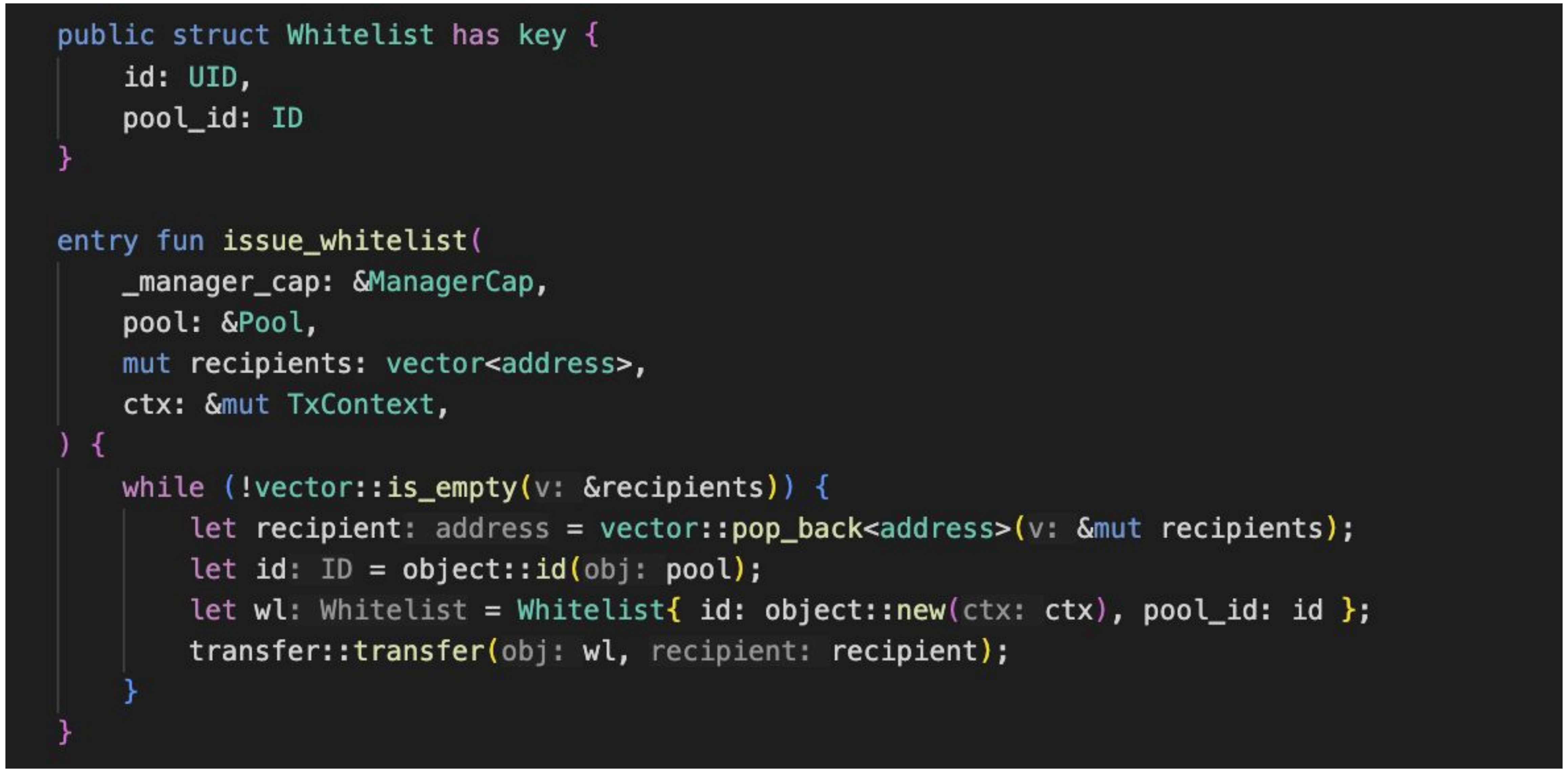
```
entry fun deposit_nft(
    _manager_cap: &ManagerCap,
    pool: &mut Pool,
    name: String,
    number: u64,
    url: vector<u8>,
    attribute_keys: vector<String>,
    attribute_values: vector<String>,
    ctx: &mut TxContext,
) {
    let nft: Tails = Tails {
        id: object::new(ctx: ctx),
        name,
        number,
        description: string::utf8(bytes: b"Tails by Sui Mover 2024."),
        url: url::new_unsafe_from_bytes(bytes: url),
        attributes: vec_map::from_keys_values(keys: attribute_keys, values: attribute_values)
    };

    table_vec::push_back(t: &mut pool.tails, e: nft);
    pool.num = pool.num + 1;
}
```



```
public struct Pool has key {
    id: UID,
    tails: TableVec<Tails>,
    num: u64,
    is_live: bool,
    // price: u64,
    // start_time: u64,
}

entry fun new_pool(
    _manager_cap: &ManagerCap,
    ctx: &mut TxContext,
) {
    let pool: Pool = Pool {
        id: object::new(ctx: ctx),
        tails: table_vec::empty(ctx: ctx),
        num: 0,
        is_live: false,
    };
    transfer::share_object(obj: pool);
}
```

Section 3

whitelist

- 

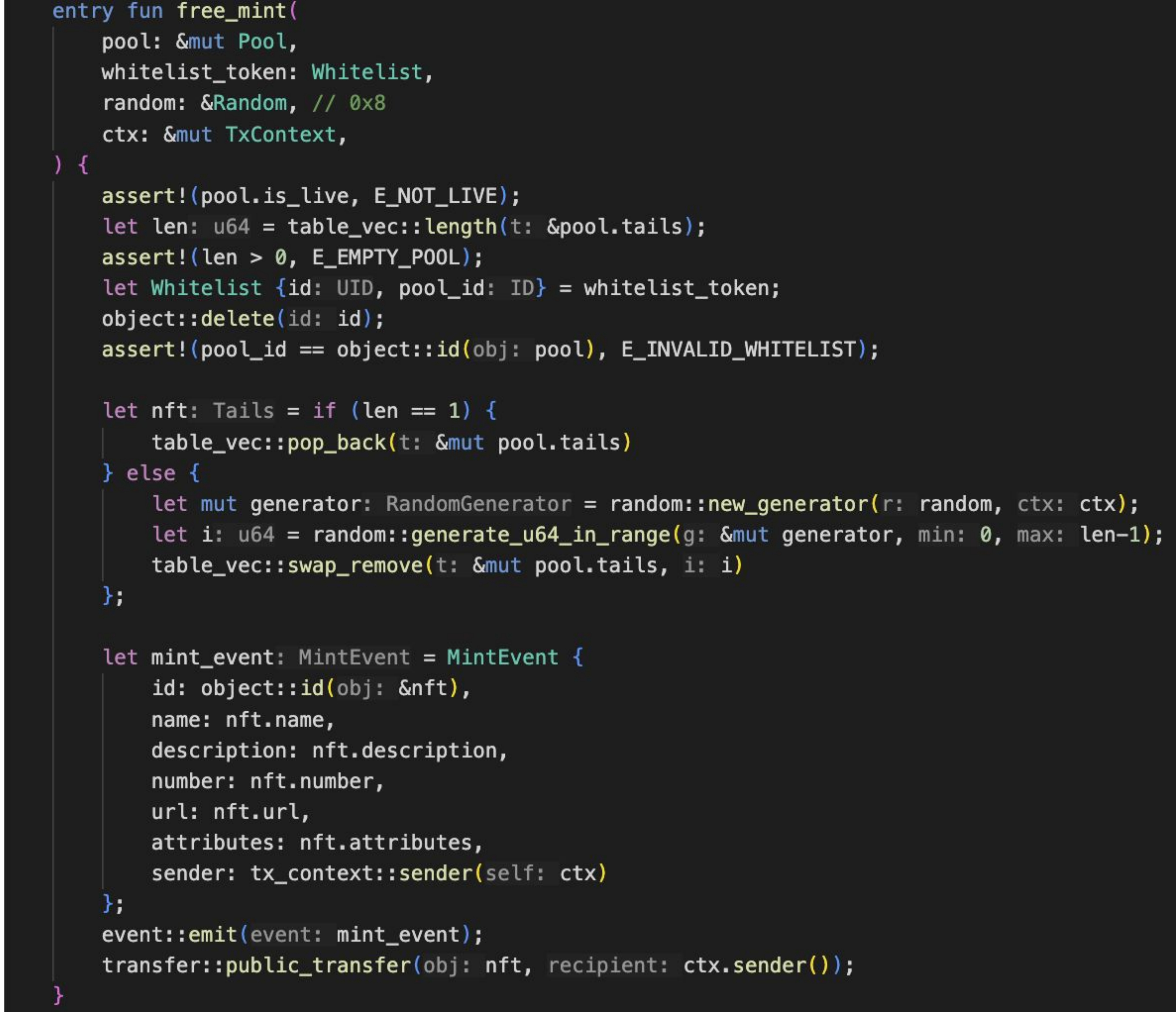
```
public struct Whitelist has key {
    id: UID,
    pool_id: ID
}

entry fun issue_whitelist(
    _manager_cap: &ManagerCap,
    pool: &Pool,
    mut recipients: vector<address>,
    ctx: &mut TxContext,
) {
    while (!vector::is_empty(v: &recipients)) {
        let recipient: address = vector::pop_back<address>(v: &mut recipients);
        let id: ID = object::id(obj: pool);
        let wl: Whitelist = Whitelist{ id: object::new(ctx: ctx), pool_id: id };
        transfer::transfer(obj: wl, recipient: recipient);
    }
}
```

Section 3

mint

Key Points:

- Whitelist
- Random



```
entry fun free_mint(
    pool: &mut Pool,
    whitelist_token: Whitelist,
    random: &Random, // 0x8
    ctx: &mut TxContext,
) {
    assert!(pool.is_live, E_NOT_LIVE);
    let len: u64 = table_vec::length(t: &pool.tails);
    assert!(len > 0, E_EMPTY_POOL);
    let Whitelist {id: UID, pool_id: ID} = whitelist_token;
    object::delete(id: id);
    assert!(pool_id == object::id(obj: pool), E_INVALID_WHITELIST);

    let nft: Tails = if (len == 1) {
        table_vec::pop_back(t: &mut pool.tails)
    } else {
        let mut generator: RandomGenerator = random::new_generator(r: random, ctx: ctx);
        let i: u64 = random::generate_u64_in_range(g: &mut generator, min: 0, max: len-1);
        table_vec::swap_remove(t: &mut pool.tails, i: i)
    };

    let mint_event: MintEvent = MintEvent {
        id: object::id(obj: &nft),
        name: nft.name,
        description: nft.description,
        number: nft.number,
        url: nft.url,
        attributes: nft.attributes,
        sender: tx_context::sender(self: ctx)
    };
    event::emit(event: mint_event);
    transfer::public_transfer(obj: nft, recipient: ctx.sender());
}
```

Reference

- https://docs.sui.io/standards/display

SuiMover

THANK YOU

PRESENTATION TEMPLATE

suimover.org

End Slide