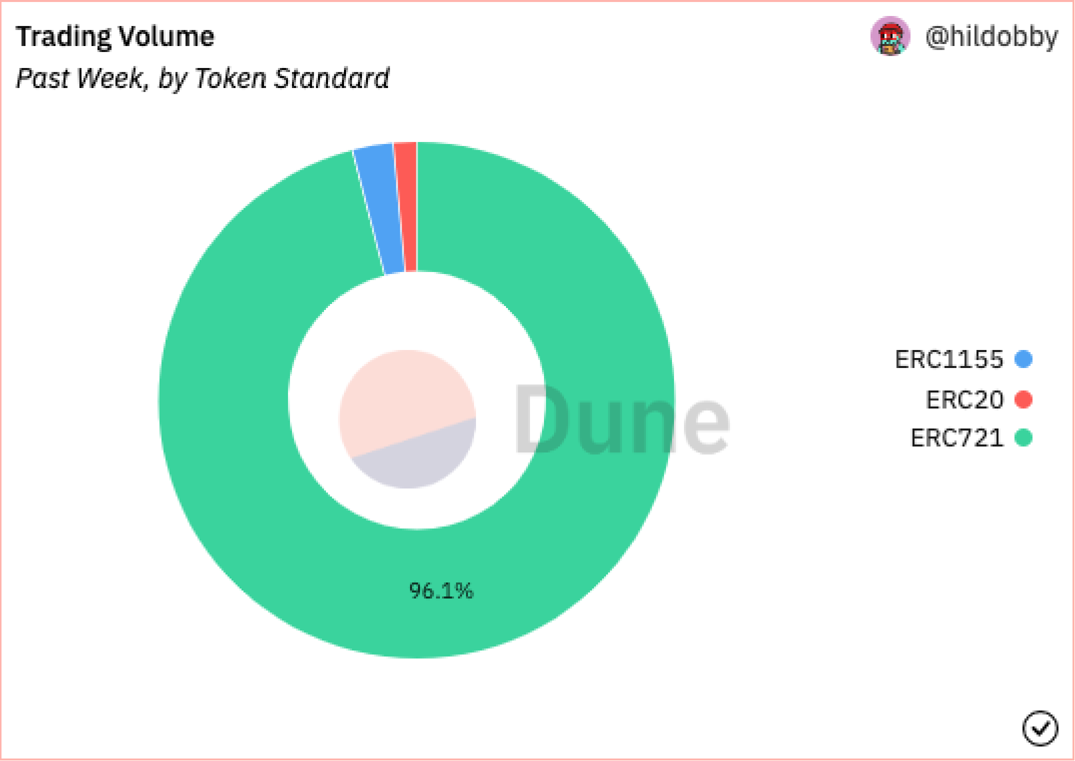
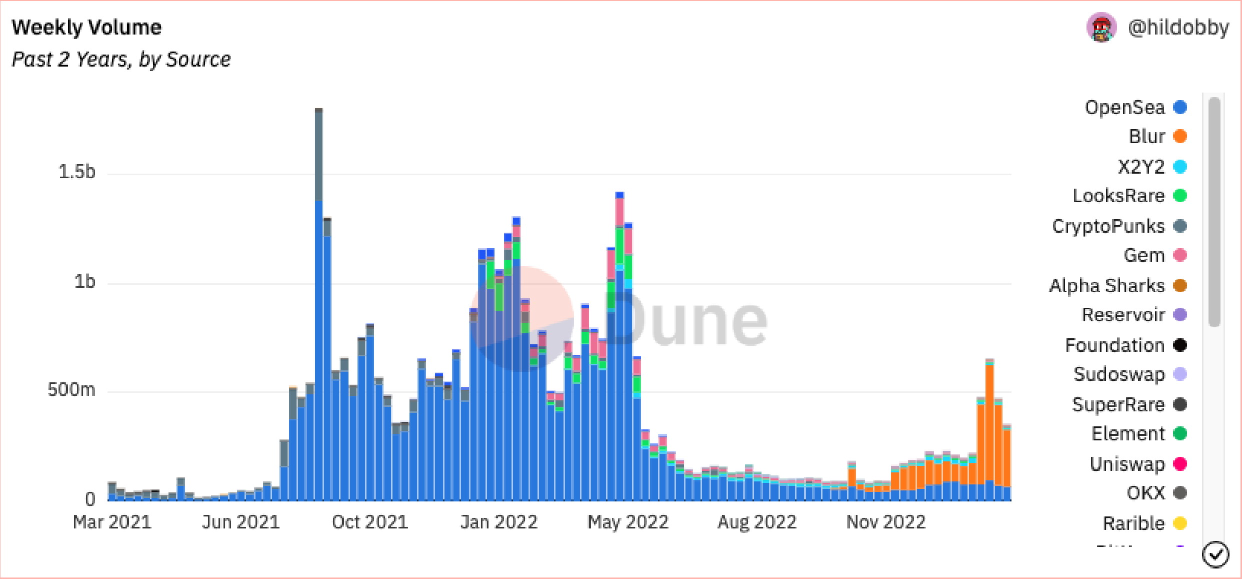
Lesson 6 - NFTs

NFTs - non fungible tokens

The essential point is that each token has a unique identity. Although they are not usually divisible, they could be.

They have been used to indicate ownership of digital assets, they now are also used as a means of access control, or an achievement in a game.





ERC721 Interface

See Documentation
and Open Zeppelin interface

```
interface IERC721 is IERC165

event Transfer(address indexed from, address indexed to,
uint256 indexed tokenId);
event Approval(address indexed owner, address indexed approved,
uint256 indexed tokenId);
event ApprovalForAll(address indexed owner, address indexed
operator, bool approved);
function balanceOf(address owner) external view returns
(uint256 balance);
function ownerOf(uint256 tokenId) external view returns
(address owner);
function safeTransferFrom(address from, address to, uint256
tokenId, bytes calldata data) external;
function safeTransferFrom(address from, address to, uint256
tokenId) external;
function transferFrom(address from, address to, uint256
tokenId) external;
function approve(address to, uint256 tokenId) external;
function setApprovalForAll(address operator, bool approved)
external;
function getApproved(uint256 tokenId) external view returns
```

```
(address operator);
function isApprovedForAll(address owner, address operator)
external view returns (bool);

}
```

ERC165

Example interface from Open Zeppelin

```
interface IERC165 {

function supportsInterface(bytes4 interfaceId) external view
returns (bool);

}
```

Preset Implementations from Open Zeppelin

See [ERC721PresetMinterPauserAutoId]()

```
...
constructor(string memory name, string memory symbol, string
memory baseTokenURI) ERC721(name, symbol) {

_baseTokenURI = baseTokenURI;

_setupRole(DEFAULT_ADMIN_ROLE, _msgSender());

_setupRole(MINTER_ROLE, _msgSender());

_setupRole(PAUSER_ROLE, _msgSender());

}

...
```

NFT Metadata

> NFTs are often used to indicate ownership of a digital asset, the digital asset would be large, so expensive to store on the blockchain, therefore it is usual for the digital asset to be stored elsewhere, and a link maintained in the ERC721 on the blockchain.
>
> The **metadata extension** is OPTIONAL for ERC-721 smart contracts.
> This allows your smart contract to be interrogated for its name and for details about the assets which your NFTs represent.
>
> In the above constructor you can see a storage variable being setup to hold the
> `_baseTokenURI` variable.
>
> ```
> interface ERC721Metadata /* is ERC721 */ {
>
>         function name() external view returns (string _name);
>         function symbol() external view returns (string _symbol);
>         function tokenURI(uint256 _tokenId) external view returns (string);
> }
> ```
>
> Example implementation of Token URI
> See Documentation
>
> ```
> function tokenURI(uint256 _tokenId) public view returns (string) {
>   return Strings.strConcat(
>       baseTokenURI(),
>       Strings.uint2str(_tokenId)
>   );
> }
> ```
>
> In addition to this, marketplaces on BSC support the following metadata
>
> ```
> {
> "name":"NFT Name",
> "description":"NFT Description",
> "image":"https://somedomain.com/pic/xxxx.jpg",
> ```

```
"external_url":"https://originalsite.io/2",
"attributes":[...]
}
```

Property	Description
name	Name of the item. Max 200 characters.
description	A human-readable description of the item. Markdown is supported. Max 500 characters.
image	This is the URL to the image of the item. It can be just about any type of image. A 350 x 350 image is recommended.
animation_url	This is the URL to a multi-media attachment for the item. The file extensions GLTF, GLB, WEBM, MP4, M4V, OGV, and OGG are supported, along with the audio-only extensions MP3, WAV, and OGA.
animation_type	This is the file format of the multi-media attachment provided from animation_url.
external_url	This is the URL that will appear below the asset's image on the marketplace and will allow users to leave the marketplace and view the item on your site.
attributes	These are the attributes for the item to describe the detail of the NFT. (see array below)

Example attributes

```
{
    "attributes":[
        {
            "trait_type":"Rarity Class",
            "value":"Normal"
        },
        {
            "trait_type":"Type",
            "value":"Male"
        },
        {
            "trait_type":"Face",
            "value":"Mole"
        },
        {
            "trait_type":"Beard",
            "value":"Chinstrap"
        },
        {
            "display_type":"boost_number",
            "trait_type":"Power",
            "value":"Power"
        },
        {
            "display_type":"boost_percentage",
            "trait_type":"Health Increase",
            "value":"20"
        },
        {
            "display_type":"number",
            "trait_type":"Generation",
            "value":"3"
        }
    ]
}
```

Open Zeppelin implementation including metadata

Description of ERC721URIStorage
Code

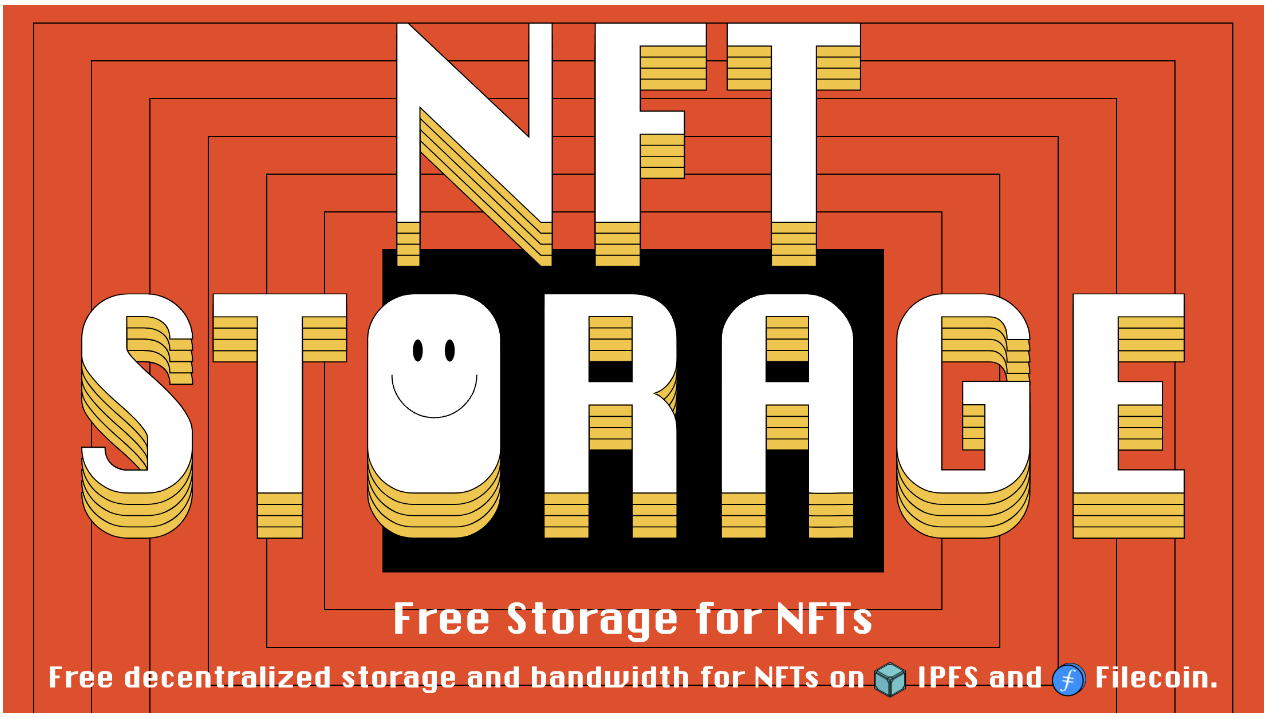
Storing Digital Assets

There are a number of decentralised storage providers available.

1. Filecoin
2. IPFS
3. NFT Storage

We will go into further details about decentralised storage in a later lesson.

NFT Storage

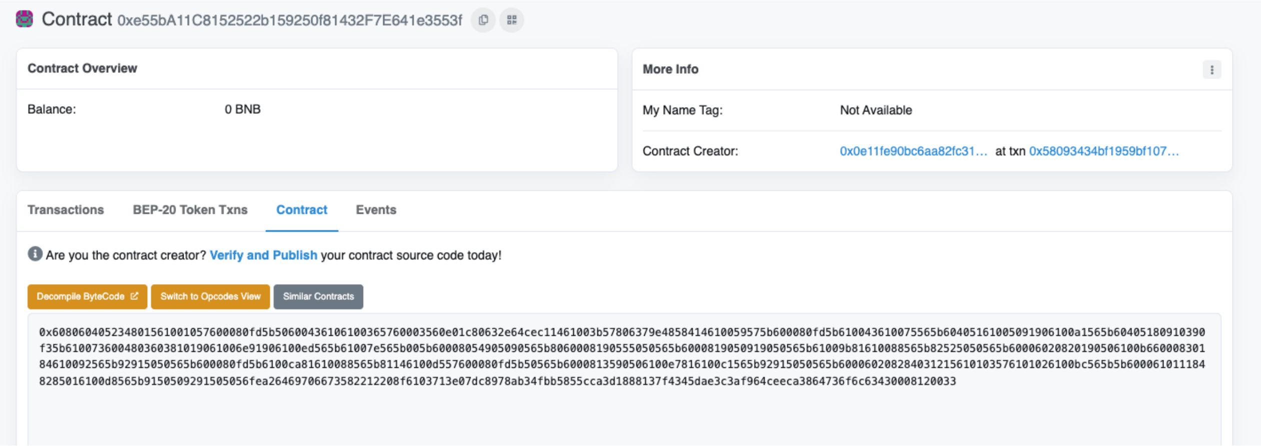

NFT STORAGE
Free Storage for NFTs
Free decentralized storage and bandwidth for NFTs on IPFS and Filecoin.

See Documentation

See their quick start guide
The steps are

1. Create an account
2. Upload your assets
3. Get an API key
4. Create a client using the nft.storage package and the client library

Verifying Contracts



Contract 0xe55bA11C8152522b159250f81432F7E641e3553f

Contract Overview

Balance:	0 BNB

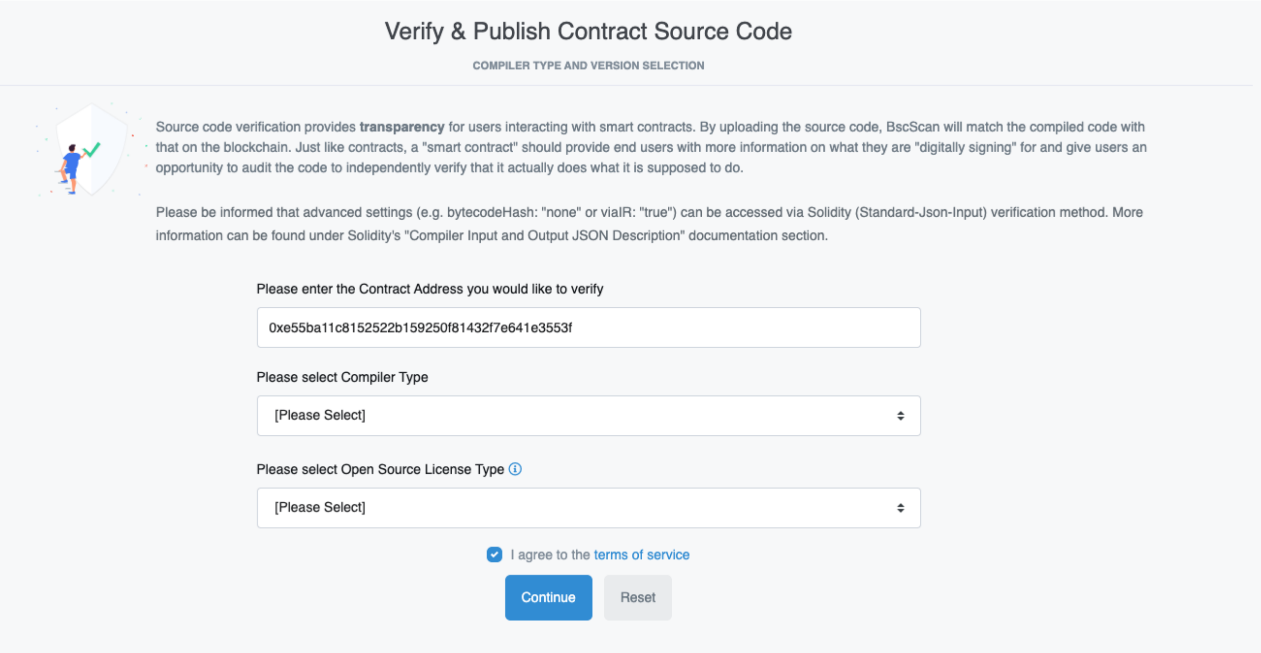
More Info

My Name Tag:	Not Available
Contract Creator:	0x0e11fe90bc6aa82fc31… at txn 0x58093434bf1959bf107…

Are you the contract creator? Verify and Publish your contract source code today!

```
0x608060405234801561001057600080fd5b50600436106100365760003560e01c80632e64cec11461003b57806379e4858414610059575b600080fd5b610043610075565b60405161005091906100a1565b60405180910390
f35b610073600480360381019061006e91906100ed565b61007e565b005b60008054905090565b8060008190555050565b600081905091905056 5b61009b81610088565b82525050565b6000602082019050610 0b660008301
84610092565b92915050565b600080fd5b6100ca81610088565b81146100d557600080fd5b50565b6000813590506100e7816100c1565b92915050565b600060208284031215610103576101026100bc565b5b600061011184
8285016100d8565b9150509291505056fea26469706673582212208f6103713e07dc8978ab34fbb5855cca3d1888137f4345dae3c3af964ceeca3864736f6c63430008120033
```

Click 'Verify and Publish'



Verify & Publish Contract Source Code

COMPILER TYPE AND VERSION SELECTION

Source code verification provides **transparency** for users interacting with smart contracts. By uploading the source code, BscScan will match the compiled code with that on the blockchain. Just like contracts, a "smart contract" should provide end users with more information on what they are "digitally signing" for and give users an opportunity to audit the code to independently verify that it actually does what it is supposed to do.

Please be informed that advanced settings (e.g. bytecodeHash: "none" or viaIR: "true") can be accessed via Solidity (Standard-Json-Input) verification method. More information can be found under Solidity's "Compiler Input and Output JSON Description" documentation section.

Please enter the Contract Address you would like to verify

0xe55ba11c8152522b159250f81432f7e641e3553f

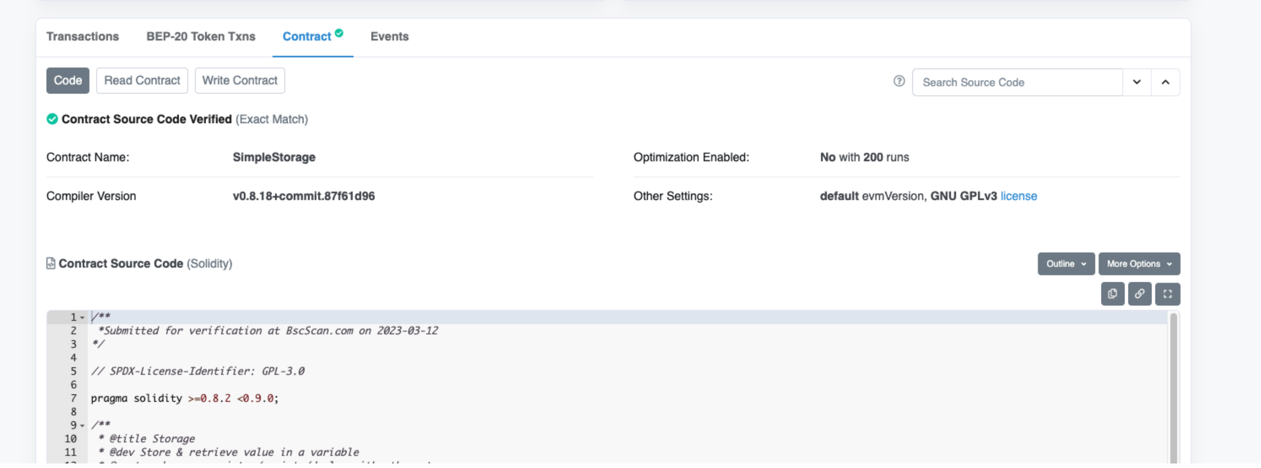
Please select Compiler Type

[Please Select]

Please select Open Source License Type

[Please Select]

☑ I agree to the terms of service



Contract Source Code Verified (Exact Match)

Contract Name:	SimpleStorage	Optimization Enabled:	No with 200 runs
Compiler Version	v0.8.18+commit.87f61d96	Other Settings:	default evmVersion, GNU GPLv3 license

Contract Source Code (Solidity)

```
/**
 *Submitted for verification at BscScan.com on 2023-03-12
*/

// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

/**
 * @title Storage
 * @dev Store & retrieve value in a variable
```