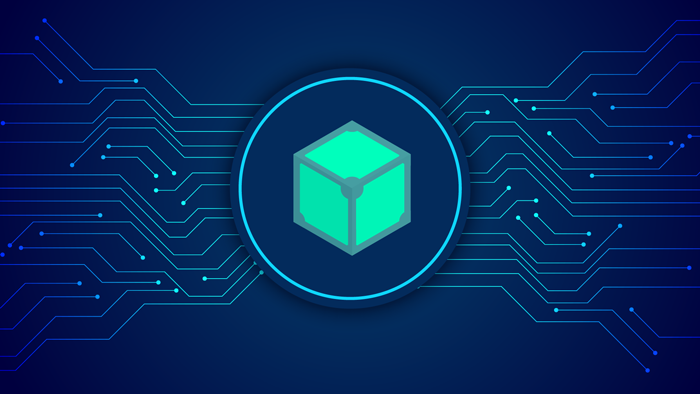
# **HOW TO SET UP DECENTRALIZED DATA STORAGE FOR NFTS USING IPFS?**

[Talk to our Consultant](https://www.leewayhertz.com/contact-us/)



Permanence and invariance are integral parts of the unique and irreplaceable NFT industry. Projected to extend a [market capitalization value](https://www.intelligenthq.com/nft-storage-preserving-the-data-of-the-multi-million-dollar-nft-industry/) of $710 Million in 2021, the non-fungible token (NFT) market is witnessing progressive growth. The fundamental defects in design fail to impart these characteristics for NFTS targeted for consumers and the growing library of new assets are minted as NFTs at risk of loss.

NFTs were meant to reside on the blockchain permanently. Due to frequent limitations of cost and space for data storage on blockchain, only the ownership record is stored with metadata linking the original NFT content. However, these links are fragile and users get directed to a specific location ( with HTTP protocol) instead of a specific asset. It implies that the content directed by the link is vulnerable to change or go offline ( due to 404 errors, rug pulls, broken links) at any time resulting in the permanent loss of original assets.

In NFT creation, blockchain is an excellent way to handle minting, bookkeeping, immutable metadata across numerous nodes. At the same time, it becomes very difficult to store a large amount of data on blockchain due to the heavy expenses for replicating the data across thousands of nodes in the network. As a result, the necessity to store and secure the off-chain NFT data marks the beginning and launch of IPFS (Interplanetary File System).

IPFS can help address these issues and ensure the permanence and accessibility of NFT data storage on the decentralized network. It is a peer-to-peer version-controlled filesystem and hypermedia protocol to store and retrieve the data. IPFS can uniquely identify each file in a global namespace for NFTs to link the NFT metadata for the stored digital asset with the content addressing feature. IPFS provides more persistence to data pinning as compared to centralized services like Dropbox or Google Drive.

The article discusses how IPFS augments a new way for persistent, permanent and decentralized NFT data storage by answering the significant questions stated below.

* [What is IPFS and how can it store NFT data?](https://www.leewayhertz.com/decentralized-data-storage-nfts/#What-is-IPFS-and-how-can-it-store-NFT-data)
* [What are the best practices to store NFT data on IPFS?](https://www.leewayhertz.com/decentralized-data-storage-nfts/#What-are-the-best-practices-to-store-NFT-data-on-IPFS)

## **What is IPFS and how can it store NFT data?**

IPFS is an open-source hypermedia protocol that allows peer-to-peer (p2p) decentralized data storage through:

* Simplified sharing
* Censorship resistance
* Easy retrieval

IPFS decides to move data around the network and find what you ask for using its content address with its file versioning advanced features and data structures.

The following three fundamental steps build upon each other constitute an entire IPFS ecosystem.

### **Step 1: Content Addressing through unique identification**

Once the users upload NFT data on IPFS, they receive an IPFS hash of the content, known as aCID. CIDs are unique identifiers or addresses of the NFT data used to reference the content irrespective of how and where it is stored. CIDs are created from the content. So, using CIDs to reference NFT data prevents issues like fragile links and rug pulls.

IPFS follows particular data-structure preferences, conventions, and IPLD from raw content to an IPFS address that uniquely identifies content on the IPFS network. The next step explores how links between content are embedded within that content address through a DAG data structure.

### **Step 2: Content Storage and Linking through DAGs**

IPFS provides decentralized data storage and retrieval ways to preserve NFT data for the long term. The permanence layer in IPFS uses cryptographic proofs to ensure the durability and persistence of the NFT data over time.

When it comes to NFT data linking, IPFS uses a Merkle DAG optimized for representing directories and files. It is possible to structure a Merkle DAG in numerous ways.

For building a Merkle DAG representation of your stored NFT data, IPFS initially splits it into *blocks*. Splitting it into blocks implies that different parts of the file can come from different sources and be authenticated quickly. Merkle DAGs also provides another important feature: if you have two similar files, parts of different Merkle DAGs are used to reference the same subset of data.

It results in transferring different versions of large datasets (such as genomics research or weather data) more conveniently. The reason behind it is that you only need to transfer the newly changed parts instead of creating entirely new files each time. Therefore, the entire NFT data gets linked and represented through Merkle DAG.

### **Step 3: Content Retrieval through Distributed hash tables (DHT)**

IPFS adopts a distributed hash table to find which peers host the NFT data. A hash table represents a database of keys to values. This hash table is split across all the peers in a distributed network where the libp2p handles connectivity and interactivity between peers.

The libp2p queries the DHT to know and find which peers store each of the blocks that make up the NFT data. After finding the content, you need to connect to that content (NFT data) and retrieve it.

IPFS uses the Bitswap module for this purpose to establish connections with peers and send a want list (a list of all blocks containing required NFT data). After receiving the blocks of requested content, they can be verified by hashing and comparing their CIDs. These CIDs also assist in block deduplication.

* **How does Libp2p support connection multiplexing?** It isn’t easy to establish the connection and maintain its expenses. With IPFS, Libp2p enables connection multiplexing among the peers with high interoperability and eliminates the need for setting up multiple connections for every service differently.The IPFS ecosystem creates CIDs to content and links it by generating IPLD Merkle DAGs. The content gets retrieved using a DHT provided by libp2p and then the user can download it using a multiplexed connection. All of this is held together by the middle of the stack, linked by unique identifiers.

## **LeewayHertz Decentralized Cloud Storage Solution**

**Get the best IPFS storage and pinning solution for your data**

[Learn more](https://www.leewayhertz.com/decentralized-cloud-storage/?utm_source=decentralized-data-storage-nfts-Insight&utm_medium=CTA)

## **What are the best practices to store NFT data on IPFS?**

With the popularity of NFT, it is important to adopt the following best practices for linking and storing NFT data on IPFS.

### **1. Different types of IPFS links for different use-cases**

There are different ways for referring to data on IPFS, suitable for different use cases. Some of them are listed below.

**CID (Content Identifier)**

CIDs is a self-description unique identifier for any content, regardless of where it is stored. They are stored and transferred over the network in a compact binary form. While displaying to users, they’re represented as strings of random-seeming characters in the following manner.

bafybeigdyrzt5sfp7udm7hu76uh7y26nf3efuylqabf3oclgtqy55fbzdi

To enable CIDv1 while using the IPFS command line, add the –cid-version=1 flag on executing the following code:

ipfs add --cid-version=1 ~/no-time-to-explain.jpeg

added bafkreigg4a4z7o5m5pwzcfyphodsbbdp5sdiu5bwibdw5wvq5t24qswula no-time-to-explain.jpeg

While using javascript, the following commands are used for the ipfs.add method:

const cid = await ipfs.add**({** content **}**, **{**

cidVersion: 1,

hashAlg: 'sha2-256'

**})**

On adding your data to IPFS, we get a CID. Then, it becomes easier to prepare a token’s metadata and mint the token on a blockchain. It is also important to convert CID to an IPFS URI for linking our content from a smart contract or inside our NFTs metadata.

* **URI (Uniform Resource Identifier)**

The IPFS URI Uniform Resource Identifier is used to specify particular content in a given context. The URI scheme identifies the context (appended to the URI as a prefix, followed by ://). Here, the IPFS’s URI scheme is just IPFS.The example of a full IPFS URI is given below.

ipfs://bafybeigdyrzt5sfp7udm7hu76uh7y26nf3efuylqabf3oclgtqy55fbzdi

IPFS URIs are the form of canonical representation for an IPFS link pointing to the file or directory. You can produce an IPFS URI by prefixing a CID string with the static string ipfs://.IPFS URI should be used to link from your smart contract to any external data stored with IPFS, comprising any metadata that describes and contextualizes the token. It is also advisable to use IPFS URI in the structured metadata of NFT while linking to different media assets stored on IPFS.It is also possible to include filenames inside the path component of an IPFS URI. For example, if you store the token’s metadata on IPFS wrapped in a directory, your URI may look like:

ipfs://bafybeibnsoufr2renqzsh347nrx54wcubt5lgkeivez63xvivplfwhtpym/metadata.json

**HTTP Gateway URL**

HTTP gateways offer interoperability for legacy browsers that fail to resolve IPFS URIs natively. These links are used in an application’s presentation layer without storing on a blockchain or inside NFT metadata.

For example:

https://dweb.link/ipfs/bafybeigdyrzt5sfp7udm7hu76uh7y26nf3efuylqabf3oclgtqy55fbzdi

Browsers with built-in support for IPFS can identify gateway links and resolve the content with the help of native IPFS protocols. Here, built-in support refers to IPFS Companion browser extension, or native support, provided by Brave. So, they automatically extract the CID from such links and load the data from IPFS according to user preferences.Gateway links are the medium to channelize interoperability, but they should not be the primary link to your NFTdata on IPFS. As long as the NFT data is stored on IPFS, the IPFS URI remains accessible. Gateway link may encounter failure if the operator losses connectivity.Hence, it is very important for developers while using gateway links to ensure that the gateway undergoes correct URL conventions. The following URL structures are acceptable in this case.

https://.tld/ipfs//path/to/subresource

https://.ipfs..tld/path/to/subresource

**How to address the content in various situations?**

Developers should adopt different link formats based on the type of context:

* + **On-Chain**

The NFT smart contract should return an IPFS URI to the assets and metadata confined to each token in the following way:

ipfs://bafybeibnsoufr2renqzsh347nrx54wcubt5lgkeivez63xvivplfwhtpym/metadata.json

It is important to create the IPFS URI before casting each token and locating it on the full URI on-chain. It indicates that smart contracts interfaces require URI and ipfs:// URI scheme makes it feasible for any app to see that the data remains available with IPFS.

* **Application**

It is advisable to link IPFS content through IPFS URI and HTTP gateway URL for user-facing applications. It is possible to generate the gateway links from a CID or IPFS as per the requirement.For example, an HTTP gateway URL targeting the public gateway at dweb.link is represented as

https://dweb.link/ipfs/bafybeigvafaks2bvivtv46n2z7uxszpvl25jhvzc6dbhnjjgjkbeia5jta/nft.mp4

Now, using CID as a subdomain, the same link is also represented as:

https://bafybeigvafaks2bvivtv46n2z7uxszpvl25jhvzc6dbhnjjgjkbeia5jta.ipfs.dweb.link/nft.mp4

Both links correspond to the following canonical IPFS URI:

ipfs://bafybeigvafaks2bvivtv46n2z7uxszpvl25jhvzc6dbhnjjgjkbeia5jta/nft.mp4

### **2. Linking metadata to its assets**

Metadata is an indispensable part of the NFT value. Most NFTs require structured metadata to specify the token’s properties. Therefore, it is important to store metadata as a JSON object encoded to a UTF-8 byte string.

For example, JSON metadata for an NFT is represented as:

**{**

"name": "No time to explain!",

"description": "I said there was no time to explain, and I stand by that.",

"image": "ipfs://bafybeict2kq6gt4ikgulypt7h7nwj4hmfi2kevrqvnx2osibfulyy5x3hu/no-time-to-explain.jpeg"

**}**

Depending upon specific use-cases, there are multiple ways to structure metadata for NFT. The above example uses the ERC-721 standard. Using ERC-721 and ERC-1155 standards can help you view NFTs using standard wallets and tools like block explorers.

It is better to use an IPFS URI to link to images, videos and other media rather than storing an HTTP gateway URL. If you need to use gateway URLs, they can be created in the presentation layer of your application.

Adding media assets to IPFS allows you to create the metadata to know the CID of images and other media for reference. The metadata stored on IPFS ensures the accessibility to retain the value of assets. Following are the steps that retain developers’ capabilities to include file names in their links (useful to user interactions) and ensure that metadata and assets can be referenced and accessed independently of each other.

### **Steps to access the linked metadata:**

**Step 1:** Create separate directories for assets and metadata.

**Step 2:** Add assets to newly created directories.

**Step 3:** Add the asset catalog to IPFS and note its CID.

**Step 4:** Generate metadata in your directory with CID reference assets for creating IPFS URIs. Make sure that the URI consists of the file name of CID and asset of the directory.

**Step 5:** Now, keep adding metadata directory to IPFS and note its CID.

**Step 6:** Generate an IPFS URI with the noted CID and store it to create an ownership record on the chain.

On completing the above steps, the metadata becomes accessible at:

ipfs://{metadata-directory-CID}/metadata-filename

and the asset becomes accessible at:

ipfs://{asset-directory-CID}/asset-filename

* + **How to preserve filenames with IPFS directories?**

While adding data to IPFS, it is possible to preserve human-readable filenames by wrapping them in a directory.

Set wrapwithDirectory option in javascript when callingipfs.add in the following manner:In javascript, you can set the wrapWithDirectory option when calling ipfs.add:

const cid = await ipfs.add**(**

**{** path: 'metadata.json', content: aJsonString **}**,

**{** wrapWithDirectory: true **}**

**)**

Adding files in a directory, ipfs.add command returns the CID of the directory object. You can also create a full IPFS URI to the file by adding the “/” character after the CID, followed by the filename, as shown in the following example.

ipfs://bafybeibnsoufr2renqzsh347nrx54wcubt5lgkeivez63xvivplfwhtpym/metadata.json

### **3. Data Persistence and Availability**

When NFT data is stored on IPFS, users can retrieve it from any IPFS node containing its copy. It results in easy data transfer and minimizes the load on a single server. On fetching the data, the user keeps its local copy to assist other users if they need it later. These copies are temporary and can be deleted until the user wants to pin the data.

Pinning a CID indicates IPFS that the data is crucial and should be eliminated when the node is close to its storage limit. When using IPFS to build a platform for storage, it is crucial to pin your NFT data to IPFS nodes. The robustness and high availability of these nodes ensure that they operate without the downtime and better performance. Adopting an IPFS cluster can also help you manage your cloud of IPFS nodes that coordinate to pin your platform data and offer it to the users.

Delegating the infrastructure to remote pinning services provides redundant, highly available storage for your IPFS data without any *vendor lock-in*. When your platform grows, it is possible to switch between pinning services or shift to another infrastructure as IPFS based data is addressed by CID, despite any location.

## **The roadmap ahead**

IPFS has acquired notable attention due to its persistence, permanence and pinning features. The surging need for data management systems and storage with easy retrieval and secured CIDs allows the users to adopt the best practices of IPFS.

With IPFS, NFT creators can get the benefits of decentralized, immutable and verifiable storage immediately. Creators can seamlessly push content into IPFS to create immutable links to the NFT’s metadata and ensure the content is stored in perpetuity.

Are you also looking to get started with IPFS for NFT data storage and leverage its potential for data replication and secure data sharing? [Get in touch](https://www.leewayhertz.com/contact-us/) with our experts to implement decentralized data storage for NFTs.