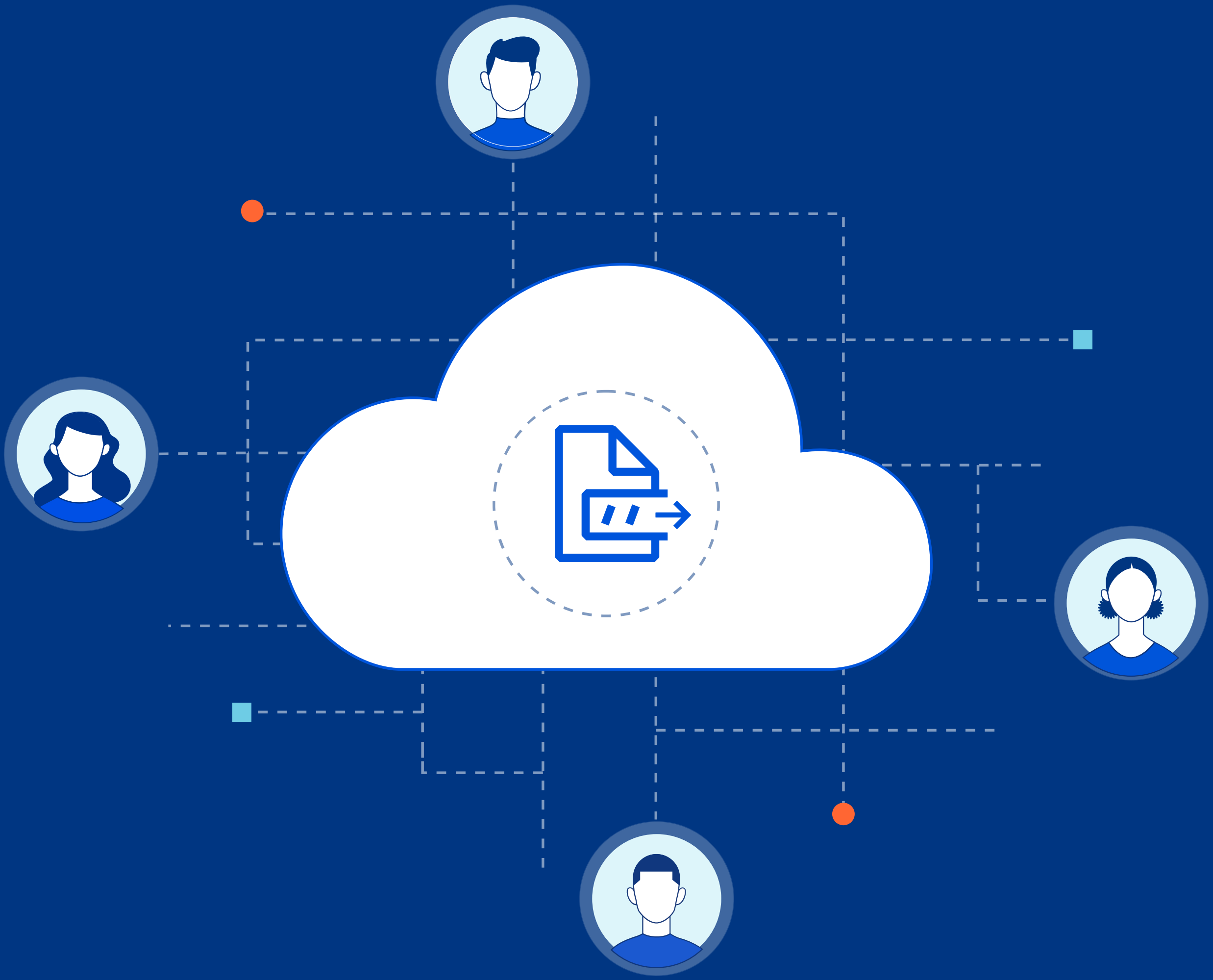
WHITEPAPER

CLOUDFLARE

Cloudflare IPFS Gateway



INDEX

Disclaimer

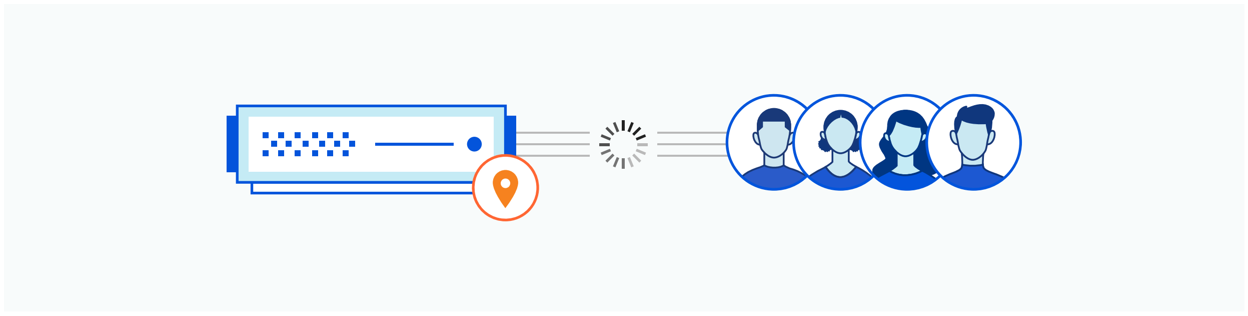
The architecture and respective behavior of Cloudflare IPFS Gateway as described in this document is what is intended for general availability (GA). As this document was written before GA, it is possible that some of the technical details, architecture, features or decisions on what is released may change prior to GA.

Overview

The Interplanetary File System (IPFS) is a protocol and peer-to-peer (P2P) distributed network for storing and sharing data; the nodes within an IPFS network contribute to a distributed file system. IPFS is a key component in enabling a decentralized, distributed version of the web (dWeb) as compared to today's centralized web model.

This whitepaper discusses the challenges of the current centralized web (Web 2.0), how IPFS helps alleviate these challenges and move towards a decentralized web (Web 3.0 or Web3), and the benefits of Cloudflare IPFS Gateway.

Challenges with the centralized web



The web as we know it today has developed around a host-centric model of end users (clients) requesting content from central repositories (servers on-premise or in the cloud). For example, a client (web browser on a computer or mobile device) will send an HTTP request to a web server listening for requests on a specific port. Once a request is received, the web server returns a response to the client. This current model presents several challenges, highlighted below.

- **Durability:** If the centralized server(s) providing the content/asset is unavailable or experiences an event where the content is lost, the content disappears from the Web.
- **Trust/Verification:** In today's Web 2.0, location-based addressing does not inherently enforce that users receive what they are requesting; additional measures must be taken to ensure the content received is what was requested and that it has not been tampered with.
- **Siloed Data:** Data tends to become siloed in centralized repositories and is not easily shared among applications and users. Although integrations and methods may exist in some cases to export data, with IPFS, there is no proprietary walled garden and reliance on support; the data is decentralized and anyone can use a different interface to access it.
- **Locality/Performance:** Content may be served from a centralized location not necessarily local or close to end users, thus resulting in higher latency. A centralized location requires additional resources and intervention to scale out as demand increases. Additionally, downloading content has latency tied to the 1:1 relationship between client and server.
- **Duplication:** Multiple people can publish the same content/file, creating duplication; further, if changes are made to the content, the entire content/file is republished.

Understanding dWeb/IPFS, benefits and use cases

Two central tenets of dWeb are:

1. There is no central entity responsible for storing and serving web content.
2. It is a trustless model with immutability and verification implicitly built in.

IPFS provides the storage layer for dWeb. It can be envisioned as a network of interconnected nodes that are able to communicate and share data with each other using the same protocols. This model aligns more to the original concept of the Web as a decentralized network of connected devices. IPFS exhibits the two central tenets of dWeb — decentralization and a trustless model, where verification is built in.

IPFS uses many different protocols for importing, naming, finding, and fetching data. A deep dive into all the associated protocols and how IPFS operates is outside the scope of this document, but a few important points to note:

- IPFS is a P2P network and thus inherently distributed; at any time a node on the network will be connected to a subset of other nodes on the network.
- IPFS references and retrieves data/content using content-based addressing via content identifiers (CIDs); a CID consists of a cryptographic hash, and every object stored in IPFS has a unique CID.
- Any content stored in IPFS is immutable; updating the content creates a new object with a new CID.
- Large content/objects are by default split into 256 kiB objects (note: the size is a configurable option up to 1 MiB). IPFS creates an empty IPFS object that links to all the pieces of the content. There are some inherent benefits to this chunking of data, in that the objects/blocks can spread across a greater number of hosts. Also, the chunking into multiple objects/blocks provides for native deduplication: the same blocks can be used for updated or different content where applicable.
- IPFS uses garbage collection to free disk space on IPFS nodes by deleting data it no longer needs. An IPFS node can protect data from garbage collection by pinning data. Pinning data tells IPFS to always keep a given object somewhere; by default this is the local node. There are also pinning services that can pin data, enabling the data to persist in IPFS without the user running IPFS nodes of their own for pinning.
- As discussed, content in IPFS is retrieved via CIDs. However, since content is immutable, as updates are made, new CIDs are created. Any updates to content changes the CID a user may want to provide for the latest update. There are solutions such as InterPlanetary Name System (IPNS), DNSLink, and Ethereum Name Service (ENS), which a user can use to map an IPFS hash/CID to a mutable address. A detailed discussion of each of these solutions is outside the scope of this paper.

The multiple benefits of IPFS and how it resolves the challenges of the centralized web are discussed in more detail below.

Decentralization / Durability

With IPFS, there is no central provider designated to serve content. Instead, each node is equal with any node capable of serving the content. Figure 1 and Figure 2 below show the difference between the current centralized Web and the dWeb model with IPFS.

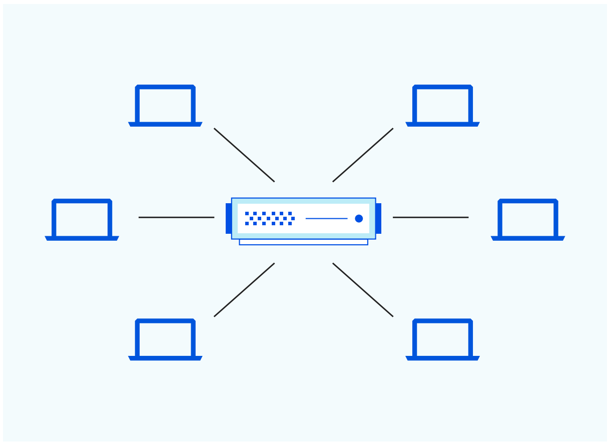


Figure 1: Centralized Web (Server-based)

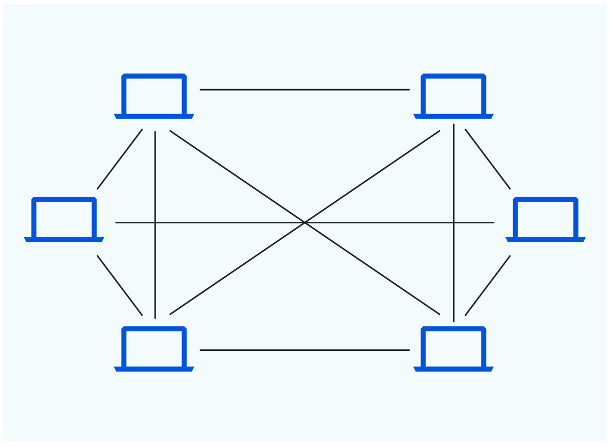


Figure 2: dWeb (P2P)

The benefits of the dWeb/P2P model are:

1. Anyone can publish content without having to rely on a single hosting provider. Instead of being locked in to one vendor, users have the flexibility to use multiple vendors and access the content with a consistent interface regardless of vendor. Users can use their own environment/nodes for pinning and persistence, use multiple free or commercial providers, or use a combination of their own nodes and pinning services.
2. Content is decentralized, so availability is increased because multiple nodes can serve the content. If one node goes offline, the other available IPFS nodes will still be able to serve the content.

Trust/verification

With HTTPS, users can have a secure encrypted connection to a site. However, since HTTP uses location-based addressing, there is no guarantee that users will receive the content they requested. If a server is hacked, users may receive tampered or malicious content. There is no inherent verification and users are explicitly trusting the provider.

With IPFS, trust and verification are implicitly built into the system using cryptography. When a file is added to IPFS, it is immutable and is referenced via a unique hash or CID. The cryptographic hashes associated with content stored in IPFS are the basis for content-based addressing. When a client makes a request, it requests the content knowing its CID. Also, when a naming service such as IPNS, DNSLink, or ENS is used, eventually the request does resolve to a CID for the requested object/content.

Figure 3 and Figure 4 below show the difference between location-based addressing as used today with HTTP and content-based addressing as used with IPFS.

Figure 3: Location-based addressing

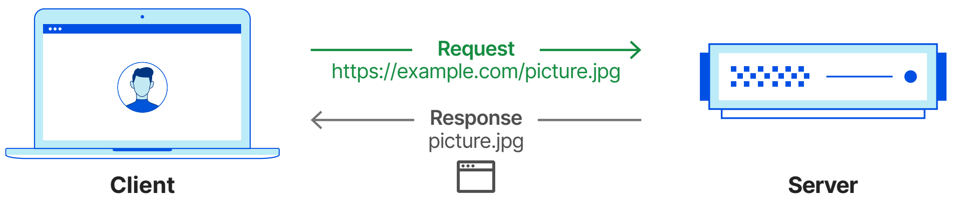
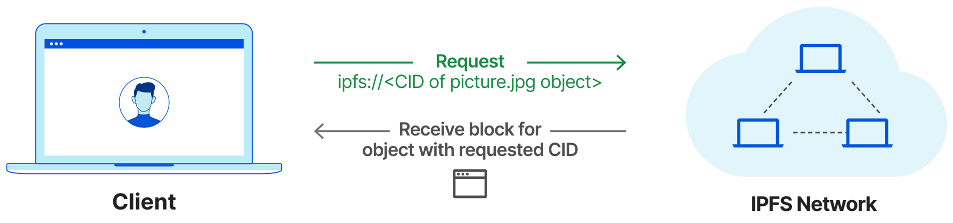


Figure 4: Content-based addressing



Location-based Addressing Example

Location-based addressing provides no guarantee that what a user receives is what they intended to receive.

http://example.com/picture.jpg

Content-based Addressing Example

Content-based addressing has a trustless model in which immutability and verification are inherently built in using hashes or CIDs to store and request data. Once content is stored, it cannot be changed. If any update is made, a new object and associated hash are created. The below example demonstrates accessing content on IPFS using the custom IPFS URL protocol.

The IPFS URL will have the following semantics: `ipfs://<CID>/<path>`

Below is an example of requesting an object via its CID.

`ipfs://QmRfGgnt2jokJP6Eh4GH7FbR3RPxmkn tKHyCXq5gNp8QuU`

No More Siloed Data

Today, application and user data is stored in centralized repositories behind closed walls of large private and public clouds. It is not easy to share application data across multiple environments. In addition, users have little control over their data as it is typically stored by one entity.

With IPFS, any data can be easily distributed and shared between different applications and end users because the model does not rely on central control.

Figure 5: Siloed data behind cloud provider

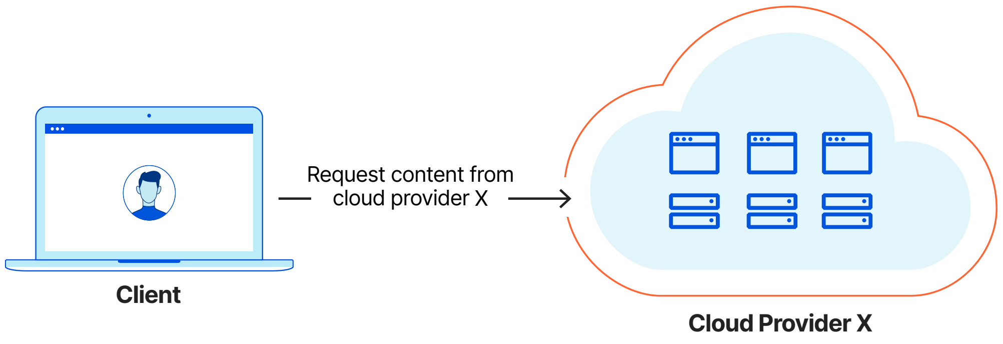
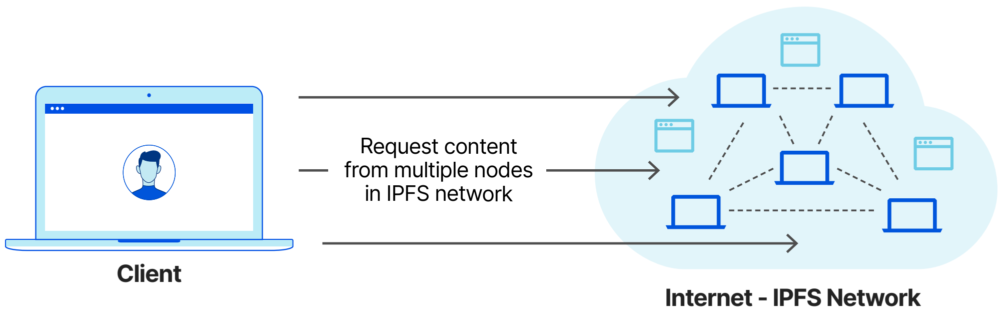


Figure 6: Data on IPFS network



Locality/Performance

IPFS also has performance benefits:

- Since IPFS is a decentralized network consisting of a distributed file system, there is no central entity becoming a choke point to serve all requests for content. Instead, any node in the P2P network that has the content cached can serve it.
- Unlike HTTP, which downloads files from one server at a time, P2P networks like IPFS retrieve pieces of data for the requested content from multiple nodes at the same time, which can result in reduced costs and latency. This also makes it possible to efficiently distribute high volumes of data without duplication. P2P protocols such as bitTorrent have long shown the performance benefits of leveraging multiple nodes simultaneously for retrieving content.

Figure 7 and Figure 8 below depict a file being downloaded in the centralized model versus a file being downloaded in a P2P model like IPFS.

Note that IPFS uses a different protocol called Bitswap to discover, request, and receive block(s) of data for a given content/object with a unique hash. In short, Bitswap is a message-based protocol consisting of a core module for exchanging blocks of data. It is responsible for directing the requesting and sending of blocks of data to and from other peers in the network. Each IPFS node sends a list to peers of blocks it wants to receive. Whenever a node receives a block, it checks if any of its peers want the block and sends it to them.

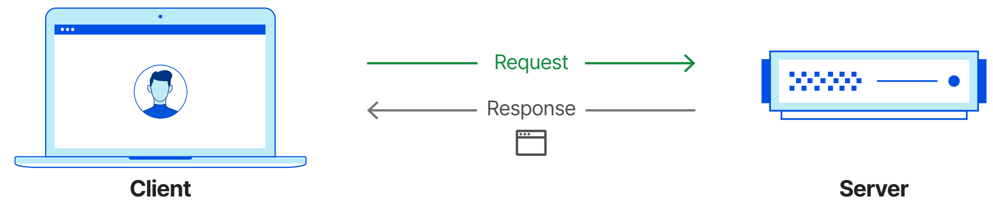


Figure 7: Centralized Web (server-based response)

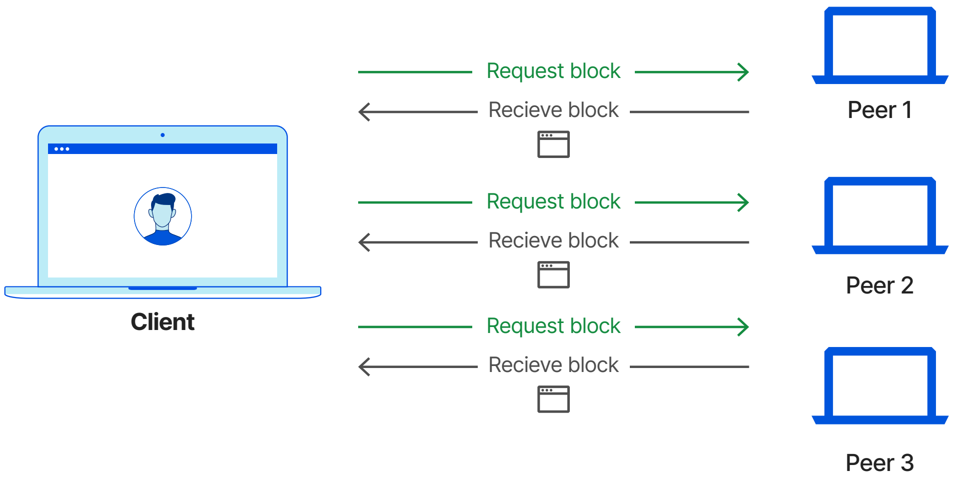


Figure 8: dWeb (P2P response)

Deduplication / Efficient Storage

IPFS inherently has deduplication built in, more efficiently using storage resources. There are several core actions IPFS takes here:

- IPFS does not store the same content twice — every content/object has a unique hash.
- IPFS chunks data into blocks, and as part of deduplication can leverage the same blocks if the content for those blocks has not changed. This is shown in the diagram below, where only the last block of updated content is uploaded.
- Content/data stored on IPFS is immutable, so an update involves the creation of new content/object. As mentioned previously, there is some inherent deduplication due to how data is chunked and stored in IPFS. Additionally, using an add-on or version control system (VCS), it is possible to implement a robust version control in which only changes are uploaded on any content updates; the IPFS architecture makes this possible.

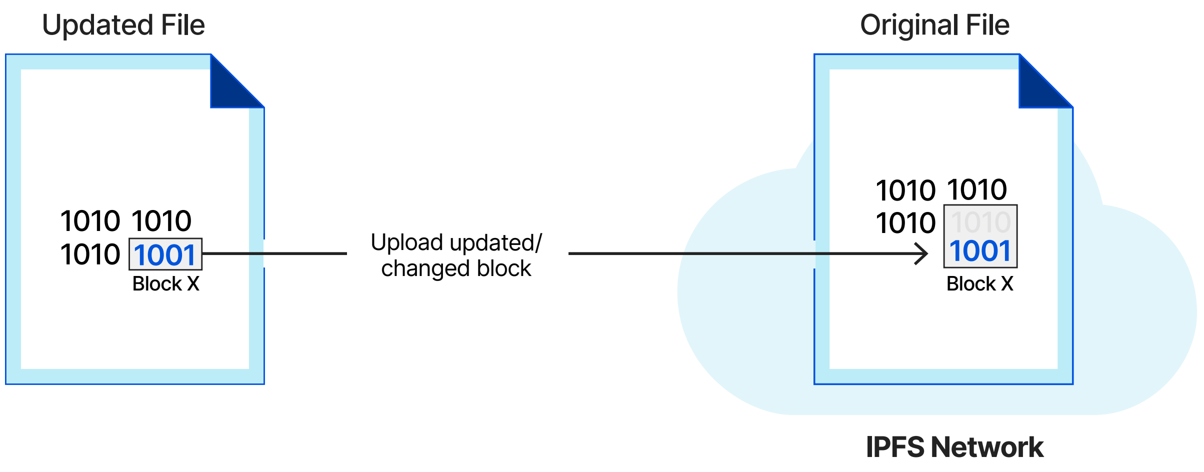


Figure 9: IPFS - Updated or changed block is uploaded

Why Cloudflare IPFS Gateway?

Today, IPFS has not yet reached the level of maturity of HTTP. For example, mainstream browsers do not yet support native IPFS networks and addressing. However, as described in this document, there are very clear benefits and advantages of using IPFS, and many companies have started to use it.

Cloudflare IPFS Gateway provides a bridge between the current Web 2.0 model and the new decentralized and trustless model of Web3. With Cloudflare IPFS Gateway, customers can start leveraging IPFS and make content on IPFS easily accessible via HTTP. One shortcoming of IPFS's content-based addressing is that hashes are difficult to remember. To help alleviate this shortcoming, customers can use Cloudflare DNS to map IPFS content to a domain name.

In short, Cloudflare IPFS Gateway allows customers to continue using HTTP while enjoying the benefits of IPFS. It allows for easy consumption for the larger user base using traditional web clients.

Another major benefit of Cloudflare IPFS Gateway is that it sits on Cloudflare's global Anycast edge network and inherits all associated benefits of reliability, security, caching, and performance.

Cloudflare IPFS Gateway Capabilities

Cloudflare IPFS Gateway provides the following capabilities:

1. Ability to easily retrieve IPFS content: Customers can easily access content on the IPFS network without having to deploy and secure their own IPFS nodes. Cloudflare's gateway leverages IPFS nodes on its own resilient and security-hardened network to retrieve IPFS content. Cloudflare IPFS gateway can be viewed as a cache in front of IPFS. The Cloudflare IPFS Gateway cannot be used to modify or remove content from the IPFS network.

2. Ability to serve IPFS content through custom domain names: IPFS supports using DNS to map easy-to-remember names to IPFS content. DNSLink is the protocol used to do so. Cloudflare supports DNSLink, allowing customers to serve IPFS content through their domain names.

3. Leverage CDN for IPFS content (caching, performance, reliability): When using the Cloudflare IPFS Gateway, customers get the additional benefit of using the Cloudflare CDN, which can cache IPFS content close to users, increasing overall performance.

4. Single pane of glass for IPFS Gateway and Cloudflare security, reliability, and performance capabilities/products: Customers can use IPFS Gateway and manage their full security model and additional Cloudflare reliability and performance capabilities/products through a single pane of glass.

Benefits of Cloudflare IPFS Gateway

Several benefits of using the Cloudflare IPFS Gateway are:

1. Ease of Access: An easy way to access content from the IPFS network that does not require installing and running any special software.

2. Security: Since users are not installing any software but instead leveraging the IPFS Gateway, the burden of security has been moved to Cloudflare, which uses its global Anycast infrastructure to provide enhanced security.

Cloudflare filters content of the HTTP interface similar to the current HTTP protection Cloudflare has been providing for years; this helps prevent distribution of malicious content.

Additionally, Cloudflare has implemented IPFS safemode — a node protection layer on top of the IPFS implementation used on all of its native IPFS nodes, providing the ability to prevent distribution of malicious content at the native IPFS level. More information on IPFS safemode can be found on Cloudflare's blog here.

3. No maintenance/monitoring: Since Cloudflare is providing a gateway to the IPFS network, there is nothing for customers to maintain. Cloudflare maintains and monitors security, reliability, and performance.

4. Reliability: Cloudflare's global Anycast network for the IPFS Gateway provides a high level of reliability and availability.

5. Performance: Because the IPFS Gateway is on Cloudflare's edge network consisting of data centers in over 100+ countries, customers receive the benefit of performance.

With IPFS content caching, content can be served from a data center closest to the user.

Cloudflare IPFS Gateway architecture and design

The Cloudflare IPFS Gateway provides an HTTP-accessible interface to IPFS, allowing users to easily retrieve IPFS content. The below diagram depicts the Cloudflare IPFS Gateway architecture.

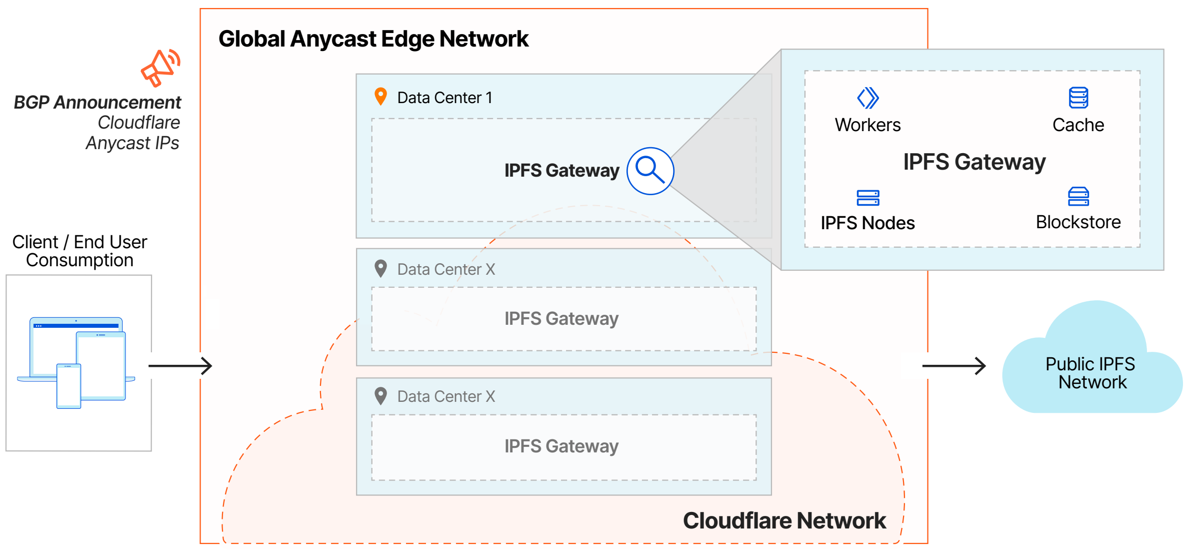


Figure 10: Cloudflare IPFS Gateway in every data center

There are several important items to note:

- Within the DNS for the respective domain, a CNAME will automatically be created that points to ipfs.cloudflare.com.
- The IPFS gateway knows what content to serve when it receives a request for a given domain by looking up the value of the DNSLink TXT record for the associated domain.
- The Cloudflare IPFS Gateway has a Cloudflare Workers code component that runs on servers globally across Cloudflare data centers. These data centers are across 100+ countries, reaching 95% of the Internet-connected population globally within 50 ms, while providing 100Tbps of network capacity and DDoS protection capability.
- IPFS nodes run in every Cloudflare data center; the Cloudflare Workers code makes API calls to the local IPFS nodes which in turn can access the local IPFS blockstore.
- IPFS nodes within the local data center peer with other IPFS nodes within Cloudflare's network; the public IPFS network can be reached as needed.
- Since Cloudflare's network leverages Anycast, users will always access the data center closest to them. Additionally, the Anycast network ensures high availability and automatically reroutes to another data center in case of problems.

Traffic Flow

Figure 11 describes the traffic flow as traffic hits the Cloudflare IPFS Gateway. A few important points to note:

- The CNAME record in the customer's DNS points to the Cloudflare IPFS Gateway URL. The request is sent to the closest data center and respective Cloudflare IPFS Gateway within Cloudflare's Anycast network.
- Cloudflare IPFS Gateway uses Cloudflare Workers and makes API calls to the local IPFS nodes. The local IPFS nodes request data from the local blockstore. If content is not cached in the local blockstore, content can be retrieved from other IPFS nodes within Cloudflare's network or the public IPFS network.
- Once the Cloudflare IPFS Gateway receives the response from the IPFS nodes, it will cache the content locally and respond to the initial request.

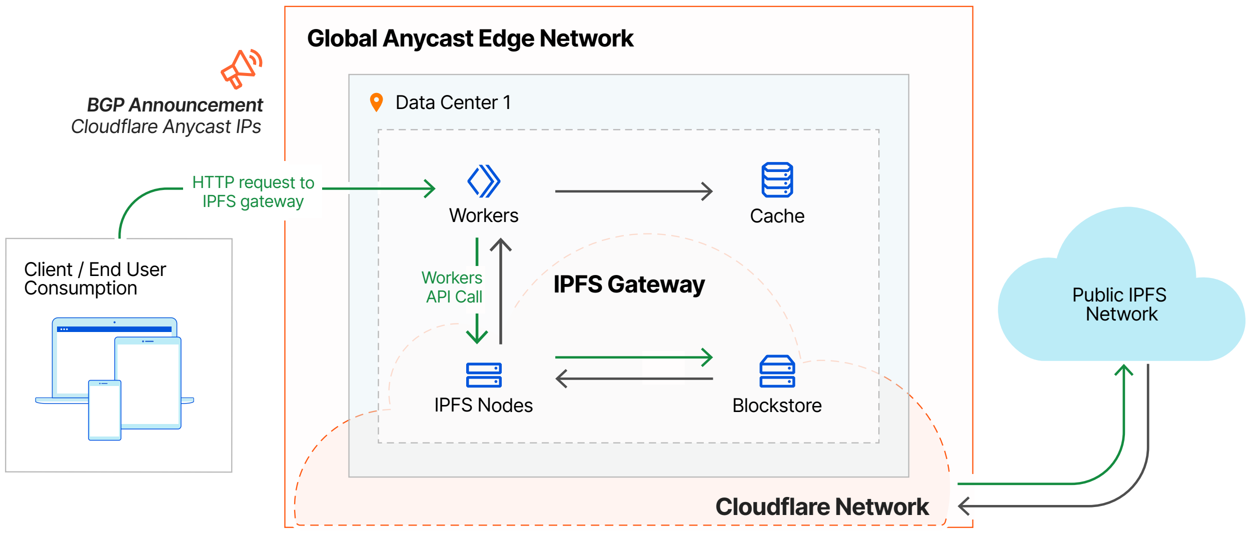


Figure 11: Traffic flow for Cloudflare IPFS Gateway

Caching IPFS Content

Cloudflare leverages its global network and caching infrastructure for caching IPFS content closest to the user and improving performance. As shown in the above diagram, IPFS Workers caches the IPFS content once received. Figure 12 demonstrates the traffic flow when a new request comes in for the same IPFS content. Note that Workers does not need to make another API call to the local IPFS nodes, because the IPFS content is already cached from the prior client request. This cuts latency and thus improves overall performance.

The Cloudflare IPFS Gateway leverages the Workers API to write to the Cloudflare cache. The maximum age of cached items is 1 day or 86,400 seconds. Upon receiving a content request, the Cloudflare IPFS Gateway will always check the DNSLink record and respective hash to ensure the cached content is the latest.

Workers API caches content in the Cloudflare Cache and the local IPFS node also caches content to its local blockstore. The maximum age of IPFS content in the blockstore is 30 days.

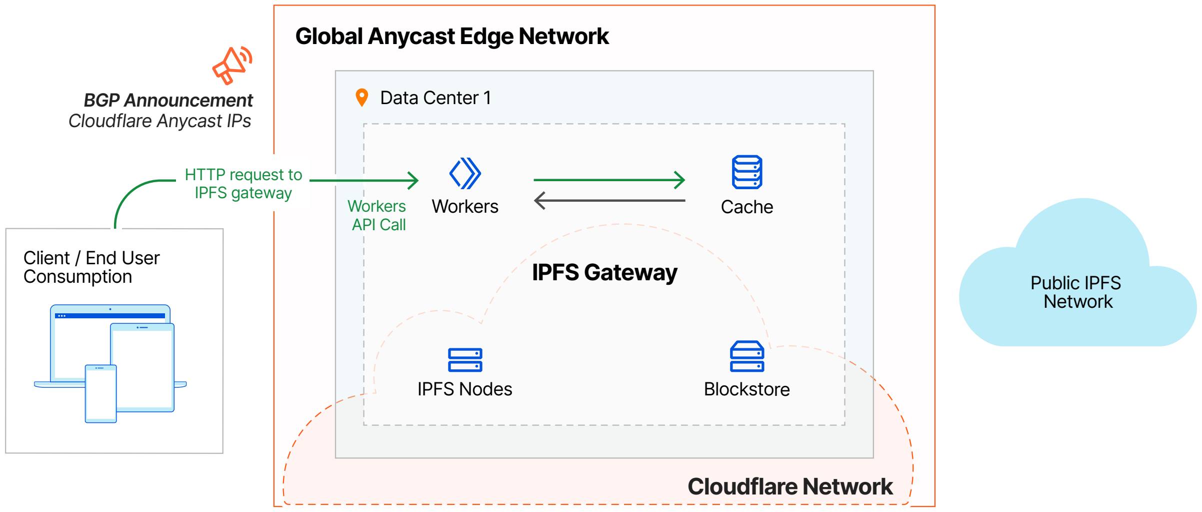


Figure 12: Cloudflare IPFS Gateway using local cache on second request for same content

Summary

IPFS is the storage layer of Web3. It resolves several challenges with the current Web 2.0 model by enabling a decentralized hosting and trustless model. Cloudflare IPFS Gateway bridges between Web 2.0 and Web3 by employing an IPFS gateway accessible via HTTP, allowing users to easily retrieve IPFS content and serve IPFS content through custom domains. With Cloudflare IPFS, customers do not need to deploy and monitor IPFS nodes in order to leverage IPFS. Additionally, customers benefit from the reliability, security, and performance of the Cloudflare network.