



InterPlanetary File System

1
language

Page [Discussion](#)

The [InterPlanetary File System](#) (IPFS) is a distributed hypermedia protocol, addressed by content and identities. IPFS enables the creation of completely distributed applications, datastores as well as websites and aims to make the web faster, safer, and more open.

It seeks to connect all computers which do peer-to-peer exchanges of the requested file transfers. In some ways this is similar to the original goals of the Web, but instead of using a client and server role model and location-based addressing, all computers in the network share the information about where which files are stored in a distributed fashion. This way a user does not need to know which server contains which files, but instead only needs to know which id the content has to receive it. The network locates the corresponding nodes and requests the data in a tamper-proof way. Files are stored in objects similar to the way git works, and the metadata storage and the file transfers have similarities to BitTorrent.

Files added to an IPFS node get chunked and hashed. The individual chunks are added to a [MerkleDAG](#) - a tree object - which gets stored alongside the chunks. The hash of the tree object is the tamper-proof content-id (CID). The node then publishes all hashes in the distributed hash-table of the network with its node-id. This way the content can be located by the CID to specific nodes which can provide the data on requests.

1 Installation

Install the [kubo](#) package or the [go-ipfs-git](#)^{AUR} package.

To start using IPFS you must first issue

```
$ ipfs init
```

as a user. This creates a `~/.ipfs` directory with all the necessary files in it.

Now you can start the IPFS daemon:

```
$ ipfs daemon
```

This starts your node, available via the `ipfs` cli, or the web interface on [localhost:5001/webui](#). Additionally, a local gateway goes up on localhost:8080 (the default port can be changed in `~/.ipfs/config`).

2 Using a service to start the daemon

For convenience, you can automate the startup of the IPFS daemon using the [Systemd/User](#) service included in [kubo](#). This ensures that the daemon starts when you log in and that it is restarted if it crashes.

You can [start/enable](#) `ipfs.service` as a [user unit](#).

If you want the service to always run regardless of the user session status, enable the system-wide service instead by [starting/enabling](#) `ipfs@username.service`.

2.1 Starting the service with a different command line

You may also want to limit the bandwidth IPFS uses by using [trickle](#)^{AUR} (c.f. [ipfs/go-ipfs#3429](#)). You can [edit](#) this [Systemd/User](#) service file to `$HOME/.config/systemd/user/go-ipfs.service`:

```
[Unit]
Description=InterPlanetary File System (IPFS) daemon (rate-limited via Trickle)
After=network-online.target
Wants=network-online.target
```

```
[Service]
ExecStart=/usr/bin/trickle -s -u 56 /usr/bin/ipfs daemon --routing=dhtclient
Restart=on-failure
```

```
[Install]
WantedBy=default.target
```

This will both start IPFS with trickle and pass the argument `--routing=dhtclient`. You may of course modify it as needed, or base your version on the package's `/usr/lib/systemd/user/ipfs.service`. [Start/enable](#) `go-ipfs.service` as a [user unit](#).

3 Firewall

IPFS requires port 4001 TCP and UDP. It attempts to configure port forwarding on the router using UPnP/IGD. Routers without UPnP/IGD support will need to do a manual port forward.

4 File sharing

To share a file using IPFS you need the daemon to be running.

```
$ ipfs add file
```

returns a hash. If someone shared this file via IPFS before, the hash would match that previous upload, making you the second source of the file.

To retrieve a file via the IPFS hash, use `ipfs cat` :

```
$ ipfs cat /ipfs/QmYwAPJzv5CZsnA625s3Xf2nemtYgPpHdWEz79ojWnPbdG/readme
```

You can pipe this into any other application, for example, to watch a video with [mpv](#):

```
$ ipfs cat QmWenbjgZnA6UguLtmUYayS6e7UQM7woB15zuEymSRRMoi | mpv -
```

Or you can download the file:

```
$ ipfs get QmWenbjgZnA6UguLtmUYayS6e7UQM7woB15zuEymSRRMoi
```

There is also an [ipget](#)^{AUR} utility, which acts like [wget](#) for IPFS. In addition, it includes a bootstrap node, so you will not have to have ipfs daemon running or installed to use it. To download a file:

```
$ ipget QmWenbjgZnA6UguLtmUYayS6e7UQM7woB15zuEymSRRMoi
```

You can share both files and folders. Folders should be shared recursively:


```
$ ipfs add -r folder
```

To view all the files and caches in a folder (if the hash is a folder):

```
$ ipfs ls QmYwAPJzv5CZsnA625s3Xf2nemtYgPpHdWEz79ojWnPbdG
```

Every file shared with network is accessible via the IPFS gateway on `localhost:8080` like this:

```
http://localhost:8080/ipfs/QmWenbjgZnA6UguLtmUYayS6e7UQM7woB15zuEymSRRMoi
```

There are public gateways, allowing users with no IPFS node running to access files on the network. For example, the official [website](#) .

5 Simple hosting with name resolution

In IPFS, files shared are never deleted, and with any change of a file its hash changes as well. This makes tasks such as website hosting difficult, as any changes to a webpage, for example to an `index.html`, would result in it having a different hash, and the old webpage would be still accessible from the old hash. It is one of a network's goals to store all the content persistently with full history. IPFS offers a name service you can use to generate persistent caches - IPNS. IPNS allows you to bind any hash to your node's unique ID, generated at initialization. You can view your ID like this:

```
$ ipfs id
```

And to bind any hash to it:

```
$ ipfs name publish HASH
```

This assigns the new hash generated by `ipfs add` after the file change to your node ID and hence makes the updated version of a folder/file accessible at the same address.

Note that when using IPNS the address has an `ipns` prefix instead of `ipfs`:

```
http://localhost:8080/ipns/QmPtMQErTfQMbzTMMpQh65cpk5y7D94WdYJurCeRqvXKmd/
```

6 Usage as makepkg DLAGENT

PKGBUILDS containing URIs pointing to IPFS resources need the **ipfs-dlagent**^{AUR} download agent installed and configured to be correctly built.

7 See also

- [IPFS homepage](#)
- [IPFS examples](#)
- [Awesome IPFS](#)
- [IPFS and pacman](#)
- [Arch Linux repository on IPFS](#)

Categories: [Clustered file systems](#) | [Peer-to-peer](#)

This page was last edited on 28 April 2023, at 03:35.

Content is available under [GNU Free Documentation License 1.3](#) or [later](#) unless otherwise noted.

[Privacy policy](#) [About](#) [Disclaimers](#)
[ArchWiki](#)