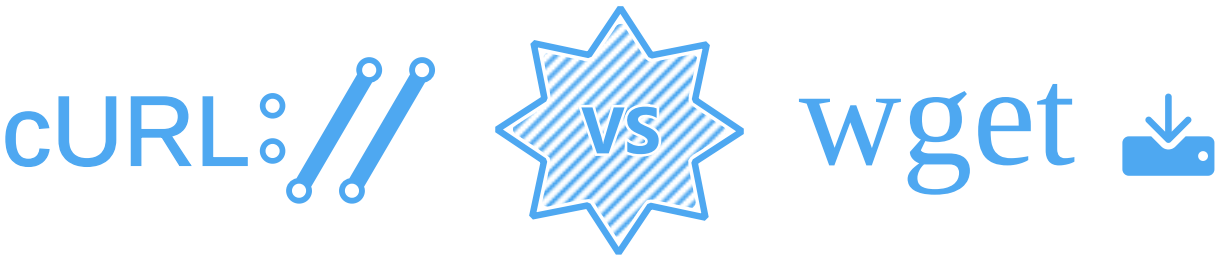


cURL vs Wget: Key Differences Explained

by Ziad Shamndy Oct 29, 2024 #cURL #HTTP #Tools    



In the world of web command-line tools, two names frequently come up: **cURL** and **wget**. Whether you're a web developer, system administrator or just want to automate your daily tasks you might wonder which tool should you choose — cURL or wget?

Both tools are commonly used for interacting with the web, though they serve slightly different purposes. cURL is a versatile web client designed for sending complex requests and interacting with a wide range of protocols, while wget specializes in downloading web files, making it ideal for simpler retrieval tasks.

This guide will dive into the key differences between these two powerful utilities and help you decide when to choose `cURL` and when to opt for `wget` .

What is cURL?

What is wget?

cURL vs wget – What's the Difference?

- 1. Availability
- 2. Speed and Scaling
- Using cURL for parallel downloads:
- 3. User Experience
- 4. Proxy Support
- 5. Blocking Bypass

Power Up with Scrapfly

FAQ

Summary

JOIN THE NEWSLETTER

Get monthly web scraping insights 👉

[Learn at ScrapFly Academy](#)

What is cURL?

cURL, short for "Client URL," is not only a highly adaptable command-line tool for transferring data via URLs but also a powerful HTTP library known as **libcURL**.

Libcurl library powers many HTTP client libraries in various programming languages, making cURL a fundamental building block in web communication. It's widely popular due to its support for multiple protocols such as HTTP, FTP, and SMTP.

Developers particularly favor cURL for its large feature set, including HTTP header manipulation, form handling, and SSL and proxy support. It is indispensable for making complex requests, interacting with APIs, and performing secure file transfers.

To give you some perspective, here are some tools and libraries powered by **libcURL**:

- **pycURL** and **curl_cffi** in Python
- **HttpClient** in C#/.NET
- **libcURL** and **Guzzle** for PHP
- **cURLHandle** in Java

These are just few examples which highlight how widely **libcURL** is used in HTTP communications across different languages and environments.

Basic cURL Example:

```
# Downloading a file with cURL, -o flag specifies the output file
cURL https://web-scraping.dev/assets/pdf/tos.pdf -o sample.pdf
```

This cURL command downloads a file from `https://web-scraping.dev/assets/pdf/tos.pdf` and saves it locally as `sample.pdf`.

What is wget?

wget, short for "World Wide Web Get," is a **GNU tool** primarily focused on downloading files from the web.

Unlike cURL, which is designed for versatility across a wide range of protocols, wget specializes in retrieving content recursively from websites (aka crawling). It excels at automating file retrieval over HTTP and FTP, making it ideal for users who need to download large sets of files or entire websites.

One of wget's standout features is its ability to **mirror** websites. In this context, "mirroring" refers to recursively downloading all the assets of a website, such as HTML pages, images, stylesheets, and scripts, and preserving the site structure. This is especially useful for creating backups, archiving websites, or offline browsing of web content.

Basic wget Example:

```
# Downloading a file with wget, -O flag specifies the output file
wget https://web-scraping.dev/assets/pdf/tos.pdf -O sample.pdf
```

This wget command downloads a file from `https://web-scraping.dev/assets/pdf/tos.pdf` and saves it locally as `sample.pdf`.

Now that we understand what each tool offers individually, we can compare them directly and see which one fits what use case.

cURL vs wget – What’s the Difference?

At first glance, both cURL and wget seem to perform similar tasks: transferring data over the web. However, there are key differences in how they function, their use cases, and their unique strengths. Let's break it down.

1. Availability

When it comes to availability, both cURL and wget are widely accessible, but they differ slightly in their distribution and protocol support. cURL is already comes installed with most operating systems while wget usually needs to be installed separately. See this table for more:

Feature	cURL	wget
Pre-installed	Windows, MacOS, Linux, BSD	Linux, BSD
Protocols	HTTP, HTTPS, FTP, FTPS, SMTP, POP3, IMAP, LDAP, SFTP, SCP, SMB, TELNET, GOPHER, MQTT	HTTP, HTTPS, FTP
Platforms	Linux, Windows, macOS, BSD,embedded systems	Linux, Windows, macOS, BSD

Protocols Supported by cURL:

cURL supports a wide range of protocols: HTTP, HTTPS, FTP, FTPS, SCP, SFTP, SMTP, POP3, IMAP, LDAP, SMB, TELNET, GOPHER, and MQTT. This extensive protocol support makes cURL a versatile tool for handling a variety of data transfer tasks and what makes `libcurl` so powerful.

By contrast, wget focuses on a smaller set of protocols, primarily HTTP, HTTPS, and FTP, making it more specialized in web file retrieval tasks.

2. Speed and Scaling

When comparing wget vs cURL in terms of speed, both tools excel in different areas.

- wget is optimized for **bulk downloading** and can recursively download entire websites. It’s excellent for large-scale downloads, like crawling and backing up web content.
- cURL, on the other hand, is optimized for making multiple, **individual requests**, and it send requests in parallel.

Here's a quick comparison overview:

Use Case	cURL	wget
Bulk Downloads	Not built for recursive downloads	Recursive download support (mirrors websites)
Single File Fetching	Efficient and quick	Suitable but slightly slower
Parallel Downloads	Supports parallel downloads with multiple requests	Not designed for parallel downloads

So while `cURL` is great for general web requests in parallel `wget` is better for buld downloads and crawling:

▼ Example: use wget for bulk downloads

```
# Download an entire website recursively
wget --mirror --convert-links --adjust-extension --page-requisites --no-parent
https://httpbin.dev/
```

This command will crawl the entire website at `https://httpbin.dev/` , adjusting all links for local browsing and ensuring any necessary files (like images or stylesheets) are downloaded as well. The output will produce a directory with HTML files and their dependencies, making it ideal for offline browsing or backup.

► Example Output

This results in a folder containing the entire site's content, ready for offline viewing.

Using cURL for parallel downloads:

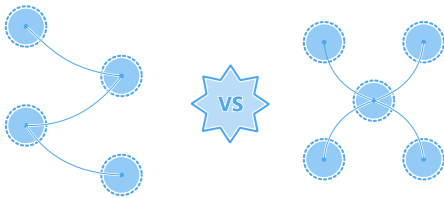
```
# Download multiple files in parallel using cURL
cURL -O https://httpbin.dev/file1.txt -O https://httpbin.dev/file2.txt
% Total      % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left  Speed
100  5023    0  5023    0     0  243k      0 --:--:-- --:--:-- --:--:--  243k
100  4024    0  4024    0     0  184k      0 --:--:-- --:--:-- --:--:--  184k
```

Here, cURL downloads two files (`file1.txt` and `file2.txt`) in parallel, saving them with their original filenames. Parallel downloads are an advantage when you need to fetch multiple individual files quickly. cURL is best suited for these kinds of tasks, where concurrency matters more than recursive downloading.

Parallel downloads allow cURL to fetch multiple files efficiently in one go.

Concurrency vs Parallelism

Related: for more on concurrency and parallelism in web requests see our introduction article



3. User Experience

The user experience varies depending on your needs. cURL offers more flexibility, but wget's simplicity makes it more accessible for straightforward tasks.

- cURL is highly customizable, making it ideal for developers and technical users who need fine control over HTTP requests, such as managing headers, cookies, and POST requests.
- wget is easier for users who just want to download files without worrying about other complexities.

Feature	cURL	wget
HTTP Method Control	Fine-grained control (GET, POST, PUT, etc.)	Limited to GET and POST

Feature	cURL	wget
HTTP Header Management	Supports adding/removing headers	Not designed for header manipulation
Recursive Downloads	Not natively supported	Full website mirroring capabilities
Ease of Use	Steeper learning curve	More user-friendly for basic tasks

For users who want cURL's flexibility but with a more modern interface, tools like [Curlie](#) are attempts to improve and simplify the cURL user experience.

Some Examples

cURL is great for interacting with APIs, where specific HTTP methods or headers are needed:

```
# Sending a POST request with cURL
cURL -X POST -d "username=user&password=pass" https://example.com/login
```

In this example, cURL is used to send a POST request to an API, including a data payload (username and password). It allows for fine control over the HTTP method.

wget is straightforward for downloading files:

```
# Downloading a file using wget
wget https://example.com/sample.pdf
```

This wget command downloads a file directly from a URL and saves it locally. wget is simpler to use for downloading files with minimal configuration.

4. Proxy Support

Proxy support is crucial when accessing restricted networks or scraping behind firewalls. Both tools offer proxy capabilities, but they differ in the types of proxies they support:

Proxy Support	cURL	wget
HTTP Proxy	✓	✓
HTTPS Proxy	✓	✓
SOCKS4 and SOCKS5 Proxy	✓	✗

[cURL](#) supports both HTTP/HTTPS and SOCKS4 and SOCKS5 proxies, which provides additional flexibility for web scraping or interacting with APIs behind different proxy configurations.

[wget](#) while supporting HTTP and HTTPS proxies, lacks support for SOCKS4 and SOCKS5 proxies, limiting its usability in some environments. Though, wget works with tools like [proxychains](#) which can add SOCKS proxy support.

5. Blocking Bypass

When it comes to bypassing blocks (like rate limits or web scraping restrictions), cURL generally has the upper hand.

`cURL` supports features such as proxying, user-agent spoofing, and cookie handling. With the help of tools like `curl-impersonate` which can mimic browser requests, helping bypass HTTP2 and TLS fingerprinting.

`wget` can handle basic blocking mechanisms like user-agent spoofing, but it doesn't have the full range of bypass techniques that cURL offers and can be easily identified by anti anti-bot system.

Here's a quick summary for more:

Blocking Feature	cURL	wget
User-Agent Spoofing	✓	✓
Proxy Support	✓	✓
Cookie Handling	Advanced cookie support	Basic cookie support
Anti-Fingerprinting	Supports cURL-impersonate for HTTP2 and TLS fingerpritsns	✗

Examples

To **bypass blocks with wget** start with setting User-Agent header that will bypass User-Agent based bot blocks:

```
# Using wget with a custom User-Agent
wget --user-agent="Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
https://httpbin.dev/headers
```

To **bypass blocks with cURL** you can also update the User-Agent header and import your existing cookies from a real web browser:

```
# Sending a request with a custom User-Agent and cookies using cURL
cURL -A "Mozilla/5.0 (Windows NT 10.0; Win64; x64)" --cookie cookies.txt
https://example.com
```

In this example, cURL uses a custom **User-Agent** string to simulate a real browser and sends cookies from a file (`cookies.txt`).

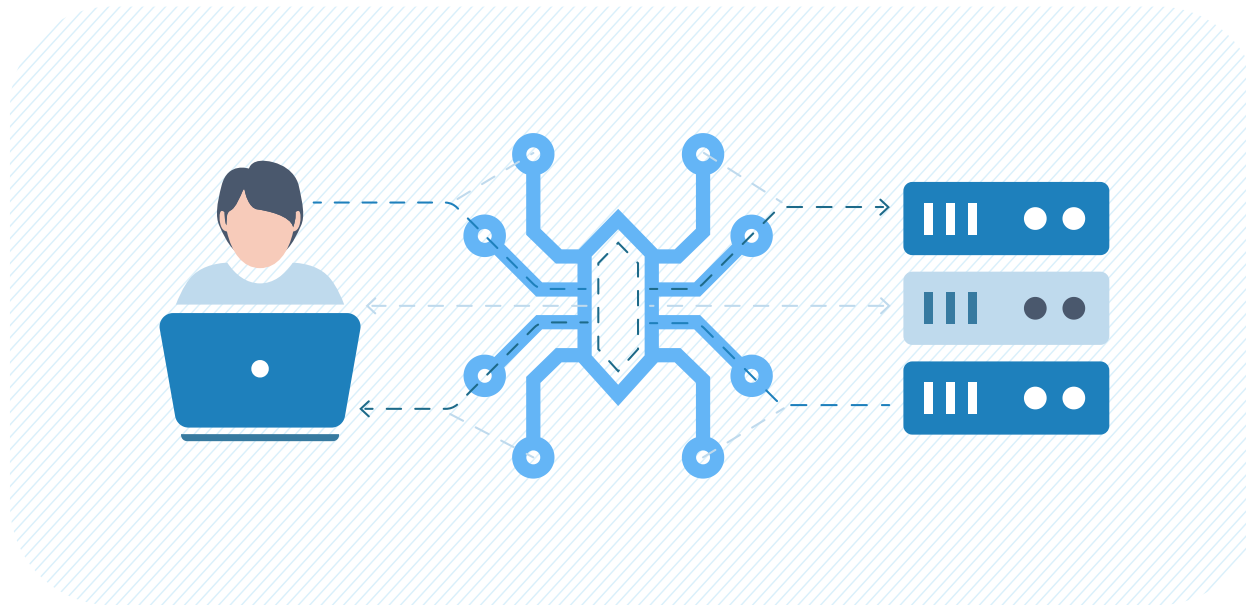
Use Curl Impersonate to scrape as Chrome or Firefox

Related: learn more how cURL impersonate prevents scraper blocking



Power Up with Scrapfly

While both cURL and wget are powerful tools for web interactions, they have their limitations when it comes to web scraping and web automation.



ScrapFly provides [web scraping](#), [screenshot](#), and [extraction](#) APIs for data collection at scale.

- [Anti-bot protection bypass](#) - scrape web pages without blocking!
- [Rotating residential proxies](#) - prevent IP address and geographic blocks.
- [JavaScript rendering](#) - scrape dynamic web pages through cloud browsers.
- [Full browser automation](#) - control browsers to scroll, input and click on objects.
- [Format conversion](#) - scrape as HTML, JSON, Text, or Markdown.
- [Python](#) and [Typescript](#) SDKs, as well as [Scrapy](#) and [no-code tool integrations](#).

FAQ

To wrap up this guide, here are answers to some frequently asked questions about cURL and wget.

Can wget send POST requests?

Yes, wget can send POST requests, but it is primarily designed for downloading files. It lacks the flexibility that cURL offers for handling multiple HTTP methods like PUT, DELETE, and PATCH, making cURL the preferred tool for API interactions or complex HTTP requests.

Which tool is better for web scraping: cURL or wget?

cURL is generally the better option for web scraping due to its advanced features like cookie handling, managing HTTP headers, and submitting complex forms. wget, on the other hand, is more suitable for scraping static sites or downloading entire webpages, especially when using its recursive downloading feature.

Is wget faster than cURL?

The speed depends on the task. wget is typically faster when downloading multiple files or entire websites recursively, as it's optimized for bulk downloads. cURL excels in handling single requests, making it faster for tasks like API calls or making several requests in parallel.

Summary

Choosing between cURL and wget ultimately depends on the specific task at hand:

`cURL` is ideal for interacting with APIs, managing complex HTTP requests, or bypassing restrictions. It supports a wide range of protocols, offering:

- Advanced control over headers and methods.

- Features like user-agent spoofing, cookie handling, and parallel downloads.
- Broad protocol support beyond just HTTP, making it versatile for developers and web scrapers.

wget excels in scenarios involving large-scale file downloads, especially when you need to:

- Recursively download entire websites or files.
- Mirror web content for offline use.
- Utilize a simple, efficient tool without needing extensive configuration.

By understanding these strengths, you can choose the right tool to match your specific requirements, whether for API handling, web scraping or web archiving.

Discover ScrapFly

Try ScrapFly for FREE!

Related Questions

- How To Use Proxy With cURL?

How to Solve the cURL (60) Error When Using Proxy?

How to Send a HEAD Request With cURL?

How To Send Multiple cURL Requests in Parallel?

How to Follow Redirects In cURL?

How To Send cURL POST Requests?

How to Copy as cURL With Brave?

How to Copy as cURL With Edge?
- What is The cURL (28) Error, Couldn't connect to server?

How to Set cURL Authentication - Full Examples Guide

How to Set User Agent With cURL?

How to Use cURL Config Files?

How To Download a File With cURL?

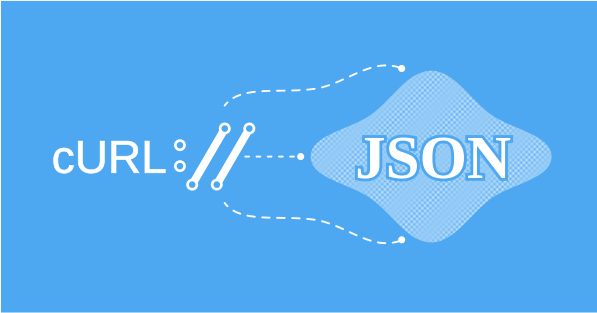
How to Copy as cURL With Safari?

How to Copy as cURL With Firefox?

More >

CURL HTTP TOOLS    

Related Posts




Mar 05, 2025

Guide to using JSON with cURL

Learn how to send JSON with `cURL` using files, inline data, environment variables, and `jq`. Includes real-world examples for Slack & Google Translate.

CURL



Dec 05, 2024

How to Ignore cURL SSL Errors

Learn to handle SSL errors in cURL, including using self-signed certificates. Explore common issues, safe practices.

CURL SSL



Nov 25, 2024

How to Use cURL to Download Files

Master file downloads with curl and discover advanced use cases.

[CURL](#)[TOOLS](#)

Company

- Careers
- Terms of service
- Privacy Policy
- Data Processing Agreement
- KYC Compliance
- Status

Integrations

- Zapier
- Make
- N8n
- LlamaIndex
- LangChain

Social



Tools

- Convert cURL commands to Python code
- JA3/TLS Fingerprint
- HTTP2 Fingerprint
- Xpath/CSS Selector Tester

Resources

- API Documentation
- Web Scraping Academy
- Is Web Scraping Legal?
- Web Scraping Tools
- FAQ

Learn Web Scraping

- Web Scraping with Python
- Web Scraping with PHP
- Web Scraping with Ruby
- Web Scraping with R
- Web Scraping with NodeJS
- Web Scraping with Python Scrapy
- How to Scrape without getting blocked tutorial
- Web Scraping with Python and BeautifulSoup
- Web Scraping with Nodejs and Puppeteer

- How To Scrape Graphql
- Best Proxies for Web Scraping
- Top 5 Best Residential Proxies

Usage

- What is Web Scraping used for?
- Web Scraping for AI Training
- Web Scraping for Compliance
- Web Scraping for eCommerce
- Web Scraping for Finance
- Web Scraping for Fraud Detection
- Web Scraping for Jobs
- Web Scraping for Lead Generation
- Web Scraping for News & Media
- Web Scraping for Real Estate
- Web Scraping for SERP & SEO
- Web Scraping for Social Media
- Web Scraping for Travel

© 2025 Scrapfly - The Best Web Scraping API For Developers