ByPass CloudFlare

- CloudFlare ByPasses Options
- Passing AntiBot
- Finding the Origin IP Behind Cloudflare

CloudFlare ByPasses Options

It depends on what you mean by "bypass Cloudflare DNS" — here are the most common scenarios:

1. Reaching the Origin Server (bypassing Cloudflare's proxy/CDN)

If a site is behind Cloudflare's orange-cloud (proxied) DNS records, you want the real server IP:

- **Check non-proxied subdomains** – Sometimes `mail.example.com` or `ftp.example.com` point directly to the origin IP (grey cloud in the DNS panel).
- **DNS history lookups** – Services like **SecurityTrails**, **DNSDumpster**, or **ViewDNS.info** can show past A/CNAME records before Cloudflare was added.
- **SSL Certificate Transparency logs** – Check **crt.sh** or **Censys** for the domain; sometimes the origin IP appears in related certs.
- **Direct IP + Host header** – Once you find the origin IP, you can curl it:

  ```
  curl -H "Host: example.com" http://<origin-ip>
  ```

- **`dig`** / **`nslookup`** – Look for A records that *aren't* Cloudflare IPs (104.x, 172.67.x, 188.x, etc.).

2. Using a Different DNS Resolver (instead of 1.1.1.1)

If you just want to stop *using* Cloudflare's public DNS:

Provider	Address
Google	8.8.8.8 / 8.8.4.4
Quad9	9.9.9.9

Provider	Address
OpenDNS	208.67.222.222
Your ISP's default	(check router settings)

Change it in your OS settings, router DHCP config, or in a browser like Firefox (Settings → Network → DNS over HTTPS).

3. Getting Past Cloudflare's Anti-Bot / Challenge Pages

If you're hitting a "Checking your browser..." or 403 page:

- Use a **real, non-headless browser** with JS enabled.
- Clear cookies / try a different browser or private window.
- Wait a few seconds for the JS challenge to resolve.
- For **developers/owners**: adjust the Security Level, WAF rules, or use Cloudflare Turnstile / API tokens.
- For **scraping**: use a residential proxy + a real browser fingerprint (e.g., Playwright with stealth patches), or hit the site's **API endpoint** directly if one exists.

Which of these are you actually trying to solve? I can give more specific steps if you share a bit more context.

Passing AntiBot

Getting Past Cloudflare's Anti-Bot / Challenge Page — The Full Toolbox

Cloudflare layers **many** signals on top of each other. No single trick works universally; you often need to combine several. Below is the full stack, from easiest to most aggressive.

---

1. The Signal Stack Cloudflare Actually Inspects

Before diving into tricks, know what's being scored:

Layer	What it checks
IP reputation	Datacenter vs residential vs mobile, ASN, geo, prior abuse
TLS fingerprint (JA3 / JA4)	Cipher order, extensions, curve list, ALPN — unique per library/browser
HTTP/2 fingerprint	SETTINGS frame values, pseudo-header order, WINDOW_UPDATE size
HTTP headers	Which headers, their *order*, values (UA, Accept-Language, etc.)
JavaScript execution	The actual challenge: a PoW hash, DOM checks, or Turnstile
Browser fingerprint	Canvas, WebGL, AudioContext, fonts, screen size, plugins, `navigator.*`
Behavioural signals	Mouse movement, scroll, timing of clicks, focus events

Layer	What it checks
Cookie state	`cf_clearance`, `__cf_bm`, `__cf_chl_*` — tied to IP + UA + TLS
Rate / volume	How fast you're hitting, from how many IPs

A request that looks fine at layer 3 but has a Python-`requests` TLS fingerprint at layer 1 will get flagged.

---

2. Quick / Low-Effort Fixes

- **Just wait 5–10 s.** The JS challenge is often a proof-of-work that resolves automatically in a real browser.
- **Use a real, up-to-date browser** (Chrome, Firefox, Edge, Safari). Not a headless container, not an old one.
- **Disable ad-blockers / privacy extensions** (uBlock, Privacy Badger, etc.) that strip or reorder headers.
- **Enable JS, cookies, and third-party cookies** for the site.
- **Make sure your system clock is correct** (cert validation depends on it).
- **Try a different browser or an incognito window.**
- **Don't hammer the page.** Reload at most once or twice, then wait.

---

3. IP / Network Layer

This is often the #1 reason you keep getting challenged.

- **Avoid datacenter / cloud IPs** (AWS, GCP, DigitalOcean). Cloudflare flags them aggressively.
- **Use a residential proxy** (IPRoyal, Smartproxy, SOAX, Decodo, etc.) or a **mobile proxy**. They rotate IPs that look like real home/mobile users.
- **Avoid widely-used free VPNs** (Proton, Windscribe free tier, etc.) — Cloudflare has their ASN on a blocklist.
- **Geo-consistency:** your proxy IP's country should match your `Accept-Language`, timezone, and the ASN. A US IP sending `Accept-Language: de-DE` looks odd.
- **Stick to one IP per session.** Rapidly rotating IPs for the same "user" triggers the rate/volume check.

---

4. TLS & HTTP/2 Fingerprint (the "invisible" wall)

Even with a residential IP, `python-requests`, `curl`, or `Node fetch` have a **distinct TLS handshake** that a real browser doesn't. Cloudflare fingerprints this.

Tools that impersonate a real browser's TLS + HTTP/2 fingerprint:

Tool	Language	Notes
`curl_cffi`	Python	`pip install curl_cffi` – impersonates Chrome / Safari / Edge / Firefox TLS. Easiest
to start.		
`curl-impersonate`	C (CLI)	Patched `curl` binary that clones Chrome/Safari handshake byte-for-byte.
`tls-client` (bogdanfinn)	Go	Very popular in the scraping community. Supports cookie jars, proxy
rotation.		
`cycletls`	Go	Fork of tls-client, actively maintained.
`utls`	Go	Low-level; you build the handshake yourself.
`httpx` + `curl_cffi` backend	Python	Modern async HTTP with the right fingerprint.

Example (Python, curl_cffi):

```
from curl_cffi import requests

session = requests.Session(impersonate="chrome124")
session.proxies = {"https": "http://user:pass@residential-proxy:port"}

r = session.get("https://protected-site.com", headers={
    "Accept-Language": "en-US,en;q=0.9",
})
```

```
print(r.status_code, r.text[:200])
```

This alone clears a surprising number of 403 / challenge pages that vanilla `requests` can't.

---

5. The `cf_clearance` Cookie (the "master key")

When a real browser successfully passes the challenge, Cloudflare sets a **`cf_clearance`** cookie. It is:

- Tied to your **IP + User-Agent + TLS fingerprint**.
- Valid for roughly **30 min – a few hours**.
- Reusable for subsequent requests *without* re-solving the challenge.

Strategy: **solve once, reuse many**

1. Open the site in a **real Chrome** on a machine with a **residential IP**.
2. Let the challenge auto-resolve (or click Turnstile).
3. Extract the cookies:
   - DevTools → Application → Cookies, **or**
   - A browser extension (e.g., *Get cookies.txt*, *EditThisCookie*).
4. Replay those cookies in your script **with the same UA, same IP, same TLS fingerprint**.

```
# Reuse the clearance cookie
cookies = {
    "cf_clearance": "abcdef...",
    "__cf_bm": "123456...",
}
r = session.get("https://protected-site.com/data", cookies=cookies)
```

> ⚠ If the IP, UA, or TLS fingerprint changes even slightly, the cookie is rejected.

Automate the "solve once" step:

- **Flare Solver** – a small Chrome extension + REST API. You POST the URL, it opens the page in a real Chrome, waits for the challenge, and returns the cookies.
- **Playwright / Puppeteer** with a **real (non-headless) Chrome** + stealth plugin (see next section).
- Run a **real Chrome on a VPS** with an Xvfb display server and a residential proxy, drive it with Selenium/Playwright.

---

6. Headless Browser + Stealth (when you must script)

If you need a full browser (for dynamic content, SPAs, etc.):

Key stealth plugins / configs

- **Playwright**:

```
from playwright.sync_api import sync_playwright
from playwright_stealth import stealth_sync

with sync_playwright() as p:
    browser = p.chromium.launch(headless=False, args=[
        "--disable-blink-features=AutomationControlled",
        "--no-sandbox",
    ])
    page = browser.new_page(user_agent="...")
    stealth_sync(page)          # patches navigator.webdriver, etc.
    page.goto("https://protected-site.com")
    page.wait_for_timeout(8000) # give the challenge time
    # page.wait_for_selector("#content", timeout=15000)
```

- **Puppeteer (Node)**: use `puppeteer-extra` + `puppeteer-extra-plugin-stealth`.
- **Selenium**: use **`undetected-chromedriver`** (Python) which strips the `--enable-automation` flag and patches `navigator.webdriver`.

Things the stealth plugins patch:

- `navigator.webdriver = false`

- Removes `chrome.runtime` / `cdc_` artifacts
- Fixes `window.length` vs `window.frames.length` mismatch
- Patches `Intl.DateTimeFormat` timezone
- Removes automation-related CDP listeners

Extra hardening:

- **Don't run `headless=True`** if you can avoid it. Use `xvfb-run` or `--headless=new` (Chromium's new headless mode is much less detectable).
- Set a **real viewport**, **real screen size**, real **timezones**.
- Inject a small **userscript** (via Tampermonkey or `page.add_init_script`) to randomise Canvas / WebGL / AudioContext outputs.
- Add **human-like interaction**: random mouse moves, scroll a little, wait 2–4 s before clicking.
- **Don't load extensions** that leak (or use a consistent set every time).

---

7. Cloudflare Turnstile (the "new" captcha)

Turnstile replaced the old I-AM-HUMAN checkbox in many deployments. It's a small widget that:

- Runs a JS proof-of-work.
- Checks your fingerprint.
- In "managed" mode, may show no UI at all (invisible) or a checkbox.

How to handle it:

- In a **real browser**, it usually just resolves silently. Wait.
- In **Playwright/Puppeteer**, you can:
  - Wait for the iframe `challenges.cloudflare.com` to appear.
  - Click the checkbox if visible: `page.frame_locator("iframe[src*=challenges.cloudflare.com]").locator("input[type=checkbox]").click()`.
  - Wait for the `cf-turnstile-response` to be set.
- If it keeps failing, your **fingerprint or IP is being flagged** → go back to layers 3–4.

---

8. Alternative Endpoints (cheeky but effective)

Sometimes the main site is behind a heavy challenge, but:

- `api.example.com` – JSON API, lighter protection.
- `amp.example.com` – AMP pages.
- `example.com/feed` or `rss.xml` – XML feed.
- `sitemap.xml` – often served with less bot protection.
- The **Cloudflare Worker** that powers the site (if it's a SPA, the data comes from an API behind the Worker, which may have different rules).
- **CDN-cache paths**: if the page is cached at the edge, a conditional `If-None-Match` / `If-Modified-Since` request might get a 304 before the challenge.

Check the site's `robots.txt` and look for API docs.

---

9. If You Own / Control the Site

You (or the site owner) can simply **turn the dial down**:

- **Cloudflare Dashboard → Security → Bots → Bot Fight Mode**: toggle off, or lower the sensitivity.
- **Security → WAF**: add a rule to **skip the challenge** for your IP / IP range.
- **Security → Settings → Browser Integrity Check**: toggle off (lets you skip the JS check).
- **Use Cloudflare Access / Access Policies** with an auth token or JWT instead of the public challenge.
- **Create an API Token** and call the origin directly (bypassing the CDN) or through the Cloudflare API.
- **Add your bot to the "Verified Bot" list** (Cloudflare → Security → Bots → Verified Bots).
- **Exclude specific paths** (e.g., `/api/*`) from the managed challenge via a WAF rule or a Page Rule.

---

10. Nuclear / "For All Costs" Options

If nothing above works:

1. **Run a real Chrome on a residential-IP VPS** (e.g., a $5/month VPS in a datacenter + a residential proxy on the outbound), drive it with Playwright + stealth. This is what most scraping farms do.
2. **Use a managed scraping service** (Bright Data, Oxylabs, ScrapingBee, Apify, Zyte). They've already solved the fingerprint + IP + challenge dance and sell you the clean HTML/JSON.
3. **`curl-impersonate` + a real `cf_clearance` cookie**: the closest you can get to "invisible" without a full browser.
4. **Solve the PoW manually** (advanced): Cloudflare's old managed challenge is a hash-based proof-of-work. You can:
   - Intercept the JS with a breakpoint.
   - Extract the challenge parameters (`cf_chl_opt`).
   - Run the hash loop in your own code.
   - POST the result back and collect the `cf_clearance` cookie.
   - (This is what libraries like **`cloudscraper`** tried to do, but Cloudflare keeps rotating the algorithm, so it breaks periodically.)
5. **Use Cloudflare's own edge**: if you own the zone, add a **Cloudflare Worker** that sets a custom header or rewrites the response before it hits the challenge logic.

---

11. A Practical Decision Flowchart

```
GET request → 200?  → Done □
                → 403 / challenge page?
                    ├─ Are you on a datacenter IP?  → Switch to residential
                    ├─ Are you using raw requests/curl? → Switch to curl_cffi / tls-client
                    ├─ Is your UA / headers consistent with IP geo?  → Fix
                    ├─ Can you solve it once in a real browser and reuse the cookie? → Do that
                    ├─ Do you need dynamic content / a SPA?  → Playwright + stealth + residential proxy
                    ├─ Is it Turnstile?  → Real browser, wait, click checkbox if shown
                    └─ Still stuck?  → Managed scraping API, or ask the site owner to whitelist you
```

TL;DR: For most people, the combo that "just works" is:

> **Residential proxy + `curl_cffi` (or `tls-client`) + a real browser's UA + solve the challenge once in Chrome to grab `cf_clearance` + reuse the cookie.**

That covers ~90 % of cases. The remaining 10 % (heavy behavioural checks, Turnstile with visible UI, rate-limited APIs) need a full stealth-browser session or a paid scraping service.

Want me to write a concrete script for a specific target or toolchain?

Finding the Origin IP Behind Cloudflare

Finding the Origin IP Behind Cloudflare This is a well-known problem (sometimes called **"origin IP discovery"** or **"de-cloudflaring"**). Cloudflare's whole point is to hide it, so there's no single reliable method — but you can combine several.

---

1. Passive DNS / DNS History (most effective)

Before the site added Cloudflare, its A record pointed directly to the origin. DNS history services logged that.

Service	How
ViewDNS.info (dns view history)	Type the domain → "DNS History" tab
SecurityTrails	`securitytrails.com` → domain → "DNS Records" (has a history timeline)
RapidDNS (rapiddns.io)	Enter domain, shows all past A/CNAME records
Censys	Search the domain, look at historical IP associations
DNSTwist / whoisds.com	Similar passive-DNS lookups

> Look for an A record that is **not** a Cloudflare IP range (104.16–104.31.x, 172.64–172.71.x, 188.114.x, 190.93.x, 198.41.x, 198.97.x, 162.158.x, 131.0.72.x, 141.101.x, 173.245.x, etc.).

That non-Cloudflare IP from, say, 2019 is very likely still the origin (or was very recently).

---

2. Non-Proxied Subdomains

Site owners often only put the **main** domain (`example.com`, `www`) on the orange cloud. Other subdomains stay grey (direct):

```
# Quick manual check
for sub in mail ftp dev staging test api blog shop cdn assets media files docs app admin panel; do
  echo -n "${sub}.example.com → "
  dig +short A ${sub}.example.com
done
```

Or use a subdomain enumerator first (`subfinder`, `amass`, `assetfinder`) and then check each A record for a non-Cloudflare IP.

Common culprits: `mail.`, `ftp.`, `dev.`, `staging.`, `old.`, `legacy.`, `internal.`, `api.`, `ws.` (websockets)

If any of those resolve to a real IP (not a Cloudflare range), that's your origin.

---

3. SSL Certificate Transparency Logs

```
crt.sh?q=%25example.com
```

or search on **Censys** / **SSL Labs**.

Sometimes:

- A cert was issued to the **origin IP** directly.
- A cert is issued to a subdomain like `origin.example.com` that resolves to the real server.
- The cert's `subjectAltName` reveals a domain you can `dig` for a non-CF IP.

---

4. Shodan / Censys / FOFA

- **Shodan.io** → search `"example.com"` (it indexes Host headers, banners, certs).
- **Censys.io** → search by domain or by the cert hash.

- **FOFA.so** → same idea.

They often show the IP that *was* or *still is* answering with that domain's Host header, including before or outside Cloudflare.

---

5. Email & DNS Records That Bypass Cloudflare

Mail records are almost never proxied through Cloudflare:

```
dig +short MX example.com
dig +short TXT example.com    # SPF, DKIM, DMARC
dig +short NS example.com
```

The MX server's IP is frequently the **same server** (or same VPS) that hosts the website.

```
dig +short A mail.example.com   # or the MX hostname
```

Also check:

- **SPF record** (`v=spf1 include:...`) → the included domain's IP.
- **DKIM key** → sometimes references a hostname on the origin.
- **SRV records** (for XMPP, SIP, etc.) → often point to the origin.

---

6. Traceroute (long shot)

```
traceroute example.com
# or
mtr example.com
```

You'll see the path to Cloudflare's edge. After the CF hop, the next hop *should* be the origin's upstream ISP — and if you can identify that ASN/IP, you've narrowed it down. Not super reliable, but occasionally the last hop before "timeout" gives a hint.

---

7. Leaks in the Site Itself

Look at the page source / network tab in DevTools:

- **Hardcoded IPs** in JS, CSS, or HTML (`https://192.0.2.1/api/...`)
- **WebSocket URLs** pointing to a raw IP or non-CF domain
- **`X-Forwarded-For`**, `X-Real-IP`, or custom headers in API responses
- **CORS errors** in the console that reveal the origin
- **Redirect chains**: `example.com/old` → `http://192.0.2.1/...`
- **`Server` header** mismatch (e.g., Cloudflare says `cloudflare` but a sub-resource says `nginx/1.18` on a different IP)
- **Error pages** that leak the origin's hostname (Apache/Nginx default 502/503 pages)

---

8. WHOIS / RDAP (historical)

Check if the domain's registrar history or name-server history reveals a prior hosting provider. Less useful now, but sometimes the old NS pointed to a server whose IP you can trace.

---

9. Brute-Force / Ping Sweep (last resort)

If you've narrowed it to a /24 or a small range (e.g., you know it's an AWS `54.200.x.x` block):

```
for i in $(seq 1 254); do
  ping -c1 -W1 54.200.1.$i &
done
wait
```

Then `curl -H "Host: example.com" http://<ip>` on each live IP to see if it responds with the site's content.

---

10. If You Own / Control the Site

Open the **Cloudflare dashboard** → **Network** (or **DNS**) tab. The origin IP is right there, and you can also see it under **Overview → your zone's "origin" info**.

Practical "Try In This Order" List

1. **Passive DNS history** (ViewDNS / SecurityTrails) ← fastest, most likely to work
2. **Subdomain enumeration +** `dig` for non-CF IPs
3. **MX / SPF / DKIM** → resolve those hostnames
4. **crt.sh / Censys** for cert-based clues
5. **Shodan / FOFA** search by domain
6. **Inspect the site's JS / network traffic** for hardcoded IPs
7. **Traceroute** as a sanity check
8. **Ping sweep** if you've narrowed the range

Quick-Start Script (Python-ish)

```
DOMAIN="example.com"

echo "=== MX / SPF ==="
dig +short MX $DOMAIN
dig +short TXT $DOMAIN | grep spf

echo "=== Common subdomains ==="
for sub in mail ftp dev api blog shop ws cdn; do
  IP=$(dig +short A ${sub}.$DOMAIN)
  [[ -n "$IP" ]] && echo "${sub}.${DOMAIN} → $IP"
done

echo "=== Check if any IP is NOT Cloudflare ==="
# Cloudflare ranges (simplified): 104.16-31, 172.64-71, 188.114, 190.93, 198.41, 198.97, 162.158, 131.0.72, 141.101, 173.245
# If the IP doesn't match, it's likely the origin.
```

Want me to run through this for a **specific domain**? Give me the URL and I can walk through the steps.