ProxyLook

FREE EBOOK · 2026 EDITION

The Proxy Playbook

How to choose, set up and scale proxies without wasting money or getting blocked.



12
chapters, from first request to production

4
proxy types compared plainly

0
sign-ups, gates or affiliate links

proxylook.com/ebooks

FREE EBOOK · 2026 EDITION

The Proxy Playbook

How to choose, set up and scale proxies without wasting money or getting blocked.

Published by ProxyLook · proxylook.com · First published November 2026

ProxyLook is an independent directory of proxy networks and antidetect browsers. This book is free. Share it with anyone, unchanged and with credit to ProxyLook.

Written and edited by the ProxyLook editorial team. Provider details come from the ProxyLook directory as it stood in November 2026.

Prices, pools and features change often. This book links to live figures on proxylook.com instead of printing numbers that go stale. Vendor-stated figures are labelled as such. The book contains no affiliate links.

Nothing here is legal advice. Laws and platform rules differ by country and by service. Check the rules that apply before acting.

© 2026 ProxyLook. Product names belong to their owners.

START HERE

How to use this book

Most proxy problems are not proxy problems. They are a mismatch between the job, the type of IP and the way the traffic is sent. This book is about closing that gap.

It is written for the person who has to make proxies work: the developer wiring them into a crawler, the growth lead buying a plan, the analyst whose price tracker keeps returning the wrong country. It starts from zero and ends in production.

The book has four parts. **Understand** explains what a proxy changes, the four types of IP, the protocols and how rotation works. **Choose** matches proxies to real jobs, takes pricing apart and gives a scorecard for picking a provider. **Build** has working setup code and covers browsers, antidetect tools and phones. **Run** deals with blocks, cost control, security and the law.

91	12	17	24	22
PAGES	CHAPTERS	DIAGRAMS	TABLES	CODE EXAMPLES

Read it front to back once. After that it works as a reference: every chapter ends with key takeaways, and the appendix collects the glossary, the scorecard, the checklists and a status-code table for the day something breaks.

One thing this book leaves out on purpose is a price list. Proxy prices move every month, so the book explains how pricing works and links to live figures at proxylook.com/pricing.

CONTENTS

What is inside

PART I · UNDERSTAND

01 **What a Proxy Really Does** 6

The request path, with and without a proxy / What the website sees, and what stays the same / Forward and reverse proxies / Proxy, VPN or Tor / Anonymity levels live in the headers / The jobs proxies do

02 **The Four Proxy Types** 12

Four types, one question: who owns the IP / The trust ladder / How the IPs are sold / Where residential IPs come from, and why consent matters / What each type costs / Checking what an IP really is

03 **Protocols, Ports and Authentication** 18

HTTP proxies and the CONNECT tunnel / What the proxy can and cannot see / HTTPS proxies, SOCKS5 and the HTTP/3 gap / Two ways to authenticate / Anatomy of a proxy URL / DNS leaks

04 **Rotation, Sessions and Geo-Targeting** 25

One address, many exits: the backconnect gateway / Rotate per request, or hold a sticky session / Sessions and targeting in the username / Geo-targeting: country, state, city, ZIP, ASN / Static IP lists versus gateways / Sticky and rotating in Python

PART II · CHOOSE

05 **Matching Proxies to the Job** 31

Start cheap, escalate on evidence / Playbooks for collecting data / Playbooks for verification, accounts and AI / A decision table for everything else / Signs of a mismatch

06 **How Proxy Pricing Works** 37

Five ways to bill for a proxy / What list prices look like / Cost per successful request / Bandwidth math / Hidden costs / Getting value from a trial

07 **Choosing a Provider** 43

A 12-point scorecard / Pool size, handled with care / Ethical sourcing and KYC / Ten example networks / A seven-day pilot / Red flags

PART III · BUILD

08 **Setup in Code** 49

Credentials and a first request / Python: requests and httpx / Node.js with undici / Browsers: Playwright and Puppeteer / Scrapy / Verifying the exit and fixing errors

09 **Browsers, Antidetect Tools and Phones** 58

Where the proxy setting lives / Firefox, Chrome and extensions / Antidetect browsers: one profile, one proxy / Phones: Wi-Fi settings and cloud phones / Leaks that give the real identity away / Checking the setup with ProxyLook's free tools

PART IV · RUN

10 **Staying Unblocked** 65

What a website checks / How a block actually shows up / Symptom, likely cause, first fix / The escalation ladder / When a scraper API or unblocker makes sense / No setup is permanent

11 **Scaling and Cost Control** 72

Concurrency versus politeness / Retries with backoff and jitter / Not paying for the same bytes twice / Tiered routing: cheap first, expensive on failure / The four metrics worth tracking / Budgets and alerts

12 **Security, Law and Ethics** 78

Why free public proxies are risky / Credentials and TLS / The legal questions to ask first / Robots.txt and rate etiquette / Residential IPs: sourcing and responsible use / A 30-day action plan

REFERENCE

A **Appendix** 85

Glossary / Provider scorecard / Pre-launch checklist / HTTP status quick reference / Troubleshooting order / ProxyLook's free tools

CHAPTER 01

What a Proxy Really Does

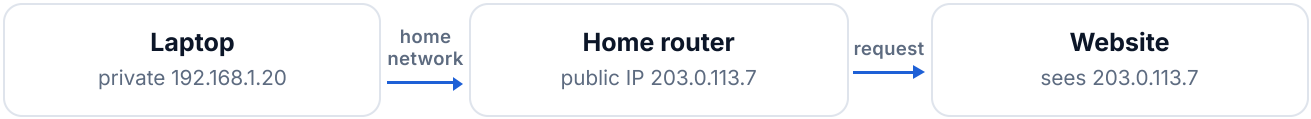
How a request travels with and without a proxy, what the website can still see, and where proxies sit next to VPNs and Tor.

- The request path through a proxy
- Proxy, VPN and Tor compared
- What changes and what stays visible
- Transparent, anonymous and elite proxies

A price-tracking script in Chicago asks a retailer for a product page. Sent directly, the request arrives from a home broadband address in Illinois. Sent through a proxy in Frankfurt, the same request arrives from a German address, and nothing else about it has changed.

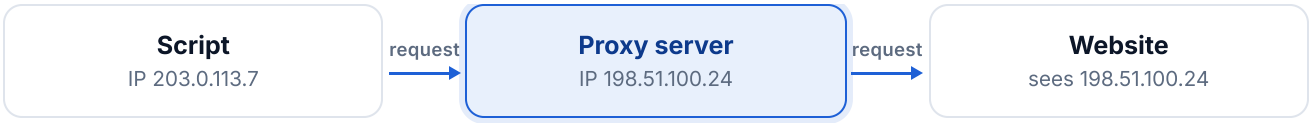
The request path, with and without a proxy

Every request carries a return address, because the website has to send the page back somewhere. On a direct connection, that address is the public IP of the network the device sits on. A laptop at home has a private address that only works inside the house. The router swaps it for the one public IP the internet provider assigned.



- A direct connection. The website logs the router's public IP.

A proxy is a server that makes the request on the client's behalf. The client hands the request to the proxy. The proxy opens its own connection to the website, fetches the response and passes it back. The return address the website records is now the proxy's.



The same request through a proxy. The website logs the proxy's IP.

One connection has become two. On the first, the proxy sees the client's real IP. On the second, the website sees only the proxy. That split is the entire product. The rest of this book is about which IP the proxy lends, and for how long.

> **A proxy changes the return address on a request, and nothing else.**

What the website sees, and what stays the same

The retailer can look up 198.51.100.24 in an IP database before it builds the page. Three facts come back, and each one feeds a decision about whether to serve, challenge or block.

- **The IP and its history.** Blocklists and reputation scores follow an address around. An IP that sent abuse last week is still paying for it.
- **The ASN.** This is the network that announces the IP: a cloud host, a home internet provider or a mobile carrier. Chapter 2 explains why this label matters.
- **The location.** The database returns a country, a region and usually a city. Prices, search results and available content follow from it.

Any browser can show the same three facts for its own connection on ProxyLook's What Is My IP page. Loading it once directly and once through a proxy makes the difference obvious.

The harder lesson is what a proxy leaves alone. First-time buyers often expect a new IP to make them a new visitor. It does not.

Changed by a proxy

- The IP address the site logs
- The ASN and the network type behind it
- The country, region and city the site assumes

Not changed by a proxy

- Cookies and logged-in accounts
- The browser fingerprint: user agent, screen, fonts, canvas, time zone, language
- The TLS fingerprint of the client software
- Request speed and timing

A cookie set yesterday identifies the same visitor today from any IP. An account is the same account wherever it logs in from. A browser reports the same screen size, fonts and time zone through every proxy. A Frankfurt IP paired with a Chicago time zone and US English is a mismatch a site can read in

one page load.

The TLS fingerprint catches scripts in particular. When a client opens an HTTPS connection, its first message lists ciphers and extensions in an order typical of that software. That message travels through the proxy untouched. A Python script that claims to be Chrome in its user agent still shakes hands like a Python script.

> **WATCH OUT** **Browsers can leak around the proxy**
>
> WebRTC, the browser feature behind video calls, can reveal the device's real IP even when page traffic uses a proxy. Run the WebRTC leak test with the proxy switched on. If a home IP shows up, fix that before anything else.

Forward and reverse proxies

Every proxy in this book is a forward proxy. It stands in front of clients and speaks to websites for them. A reverse proxy does the opposite job. It stands in front of a website's servers and speaks to visitors for them.

Nginx balancing load across ten application servers is a reverse proxy. So is Cloudflare. The visitor never chooses it and usually cannot tell it is there. The distinction is practical: the system that inspects and scores an incoming request is often a reverse proxy. Many requests pass through both kinds, a forward proxy on the way out and a reverse proxy on the way in.

Proxy, VPN or Tor

All three put another machine's IP on the request. They differ in scope, in encryption and in how much control the operator gets over the exit.

	Proxy	VPN	Tor
Covers	One app, or one request	The whole device by default	Tor Browser, or apps set up for it
Added encryption	None by default	Device to VPN server	Layered across three relays
Exit IP	Chosen by type, country, city	One of the provider's servers	A volunteer relay picked by the network
Many IPs at once	Yes, one per request if needed	No, one tunnel at a time	No

	Proxy	VPN	Tor
Built for	Automation and location testing	Personal privacy on untrusted networks	Anonymity from network observers

How proxies, VPNs and Tor differ in scope, encryption and control.

A VPN suits a person on hotel Wi-Fi who wants everything encrypted. A proxy suits a script that needs a few hundred different German IPs in one afternoon, each on its own connection. Tor suits someone who must hide from the network itself and accepts slow pages in return. The list of Tor exit relays is public, so many sites challenge or block that traffic on sight.

The missing encryption on a proxy sounds worse than it is. On an HTTPS site the content is already encrypted between client and website, and the proxy relays bytes it cannot read. Chapter 3 covers the details.

Anonymity levels live in the headers

Free proxy lists label every entry transparent, anonymous or elite. The labels describe which HTTP headers the proxy adds. Here is a plain HTTP request as a website receives it from a transparent proxy.

HTTP · REQUEST HEADERS

```
GET /product/123 HTTP/1.1
Host: example.com
User-Agent: python-requests/2.32.3
Accept: */*
Via: 1.1 proxy-01
X-Forwarded-For: 203.0.113.7
```

The last two lines were added by the proxy. `Via` announces that a proxy handled the request. `X-Forwarded-For` hands over the client's real IP, which defeats the purpose for anyone trying to change it.

Level	X-Forwarded-For	Via	What the site learns
Transparent	Client's real IP	Present	A proxy is in use, and who is behind it
Anonymous	Absent, or the proxy's IP	Present	A proxy is in use, but not who is behind it
Elite	Absent	Absent	Nothing from the headers

The three header-level anonymity labels. Elite is also sold as high anonymity.

Two caveats keep these labels in proportion. First, they apply to plain HTTP only. On HTTPS the proxy relays an encrypted tunnel and cannot insert a header, so the website receives none. Second, elite describes headers and nothing more. An elite proxy on a flagged IP range is still on a flagged IP range.

TIP **A quick header check**

Send one plain HTTP request through the proxy to a page that echoes request headers back. If `Via`, `X-Forwarded-For` or `Forwarded` appears in the echo, the proxy is not elite. Paid networks normally pass this test, but it takes a minute to confirm.

The jobs proxies do

Proxies began as office plumbing. A Squid server cached pages and filtered outbound traffic for a whole company, and many still do. The commercial proxy market grew from a different need: seeing the web as someone else, somewhere else, sees it.

Price and stock monitoring

Retailers show different prices by country and sometimes by city. E-commerce teams collect each version from a local IP.

Search rank tracking

Search results depend on location. SEO tools request the same query from many cities to see each local page.

Ad verification

Advertisers check that a campaign appears in the right country, on the right sites, without fraud. That requires viewing ads as a local visitor.

Brand protection

Counterfeit listings and phishing pages often hide from the brand's own office IPs. Investigators browse from ordinary consumer addresses.

Localization testing

QA teams confirm that a visitor in Brazil gets the Brazilian storefront, currency and cookie banner. A proxy stands in for the plane ticket.

Data collection at scale

Market research and AI training pipelines fetch millions of public pages. Spreading requests across many IPs keeps the load from any one address low.

Account management is the seventh job. Social media and marketplace teams give each account its own steady IP, so that ten accounts do not share one address. Running multiple accounts can breach a platform's terms of service, and the operator carries that risk.

KEY IDEA **Legal questions differ by data type**

Collecting public pages, collecting personal data, scraping behind a login and breaking a site's terms of service each raise a different legal question. A proxy changes none of the answers. This book is not legal advice.

Each job leans on a different kind of IP. The ProxyLook directory lists more than 200 providers, and the next chapter explains the four proxy types they sell.

KEY TAKEAWAYS

- A proxy makes the request on the client's behalf, so the website logs the proxy's IP, ASN and location.
- Cookies, logins, the browser fingerprint, the TLS fingerprint and request timing all pass through a proxy unchanged.
- Forward proxies act for clients, while reverse proxies such as Cloudflare act for websites and often do the screening.
- A VPN encrypts one device's traffic, Tor hides the user from the network, and a proxy supplies controllable exit IPs.
- Transparent, anonymous and elite describe HTTP headers only, and say nothing about the reputation of the IP.
- Multi-accounting can breach platform terms, and scraping raises legal questions that a proxy does not settle.

CHAPTER 02

The Four Proxy Types

Where datacenter, ISP, residential and mobile IPs come from, how websites rank them, what each costs and how to verify what was bought.

- Datacenter, ISP, residential and mobile IPs
- Shared, dedicated and rotating pools
- How sites classify an IP by ASN
- Rough price ranges and IP checks

Two requests reach a ticketing site one second apart. One comes from an IP owned by a cloud hosting company, the other from a phone on a mobile carrier. The site challenges the first and serves the second, and it made that choice before reading a single header.

Four types, one question: who owns the IP

Every public IP belongs to a network, and every network has an autonomous system number, or ASN. Comcast has ASNs. So do Amazon Web Services and T-Mobile. IP databases tag each ASN with a category such as hosting, fixed-line ISP, mobile carrier, business or education.

A website looks up the visitor's IP and gets back a line such as `AS64500, Example Hosting LLC, hosting`. The last word does the damage. Shoppers do not browse from server racks, so a hosting ASN suggests automation. A home broadband or carrier ASN suggests a person. The four proxy types are four answers to the same lookup, and the buyer is paying to change that one word.

Datacenter

IPs leased from hosting companies and clouds, served from machines in a rack. They are fast, cheap and sold in bulk, and the ASN says hosting.

ISP (static residential)

IPs registered to consumer internet providers but hosted on data center servers. They stay fixed, run fast and look like home broadband to an ASN lookup.

Residential

IPs of real home devices whose owners share bandwidth through an app. Traffic exits through a living-room connection, so speed varies and peers come and go.

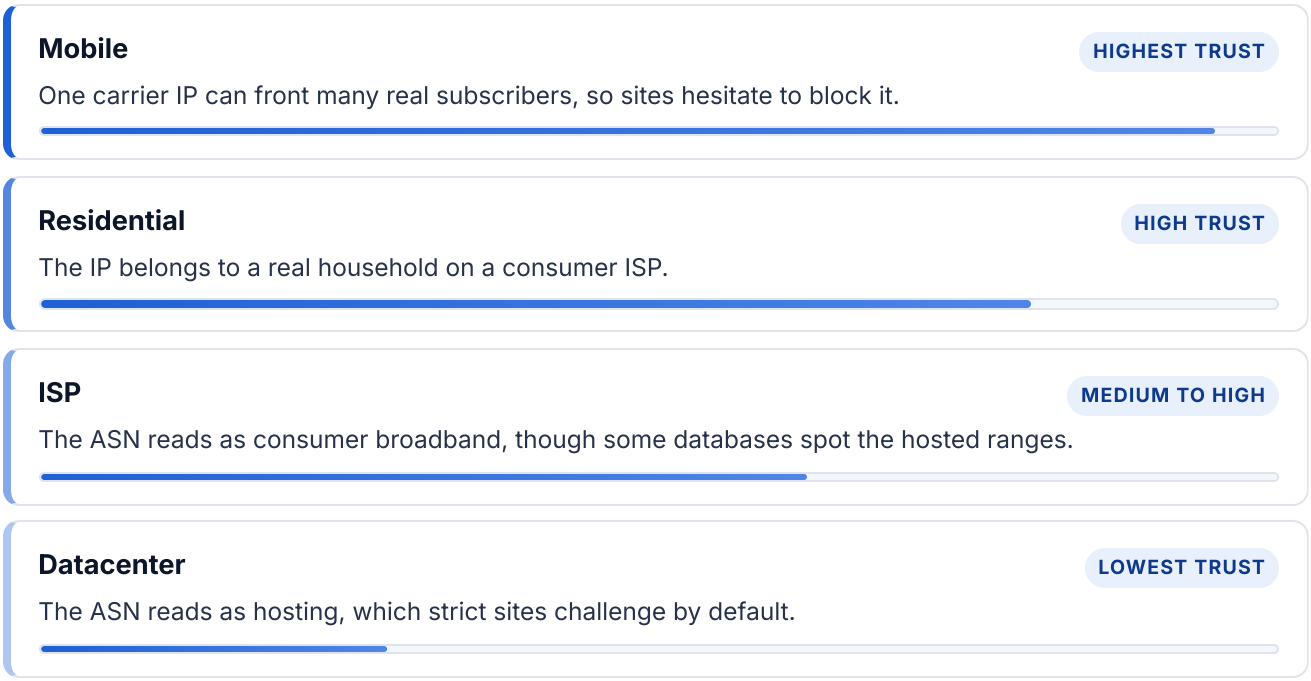
Mobile

IPs assigned by cellular carriers to phones and 4G or 5G modems. Carriers place many subscribers behind one public IP, so blocking it blocks real customers.

The names overlap in confusing ways. ISP proxies and static residential proxies are the same product under two labels. A rotating datacenter pool and a rotating residential pool are used the same way and differ only in where the IPs come from.

The trust ladder

Sites do not treat the four types equally. The order below is the default starting position of each type on a strict website.



Default trust by proxy type, highest first. The meters show relative order, not measurements.

The ladder describes a starting point, not a result. A residential IP that hammered a site an hour ago carries that history. A clean datacenter IP works well on the many sites that run no strict bot screening. Cost climbs with trust, so the practical rule is to start at the bottom rung and climb only when blocks appear.

Type	Speed	Trust	Cost	Best use
Datacenter	Fastest	Lowest	Lowest	Lenient sites, APIs, bulk crawling
ISP	Fast	Medium to high	Moderate, per IP	Long logged-in sessions on one IP
Residential	Variable	High	Higher, per GB	Strict sites, city-level local data
Mobile	Variable, often slowest	Highest	Highest	Mobile-first apps and social platforms

A qualitative comparison of the four proxy types.

Account work appears in two of those rows, and it comes with a caveat. Running multiple accounts can breach a platform's terms of service, and the operator carries that risk whatever the proxy type.

How the IPs are sold

Shared, dedicated or rotating pool

The same IP type can be packaged three ways, and the package changes both price and risk.

- **Shared.** Several customers use the same IP at once. It is the cheapest option. The risk is a neighbor who gets the IP blocked on the very site the buyer needs.
- **Dedicated.** One customer holds the IP for the rental period. Its reputation on a target then depends on that customer's behavior alone.
- **Rotating pool.** The buyer gets no specific IPs, only a gateway that draws from a large pool. Billing is usually by traffic. Chapter 4 covers gateways.

Datacenter and ISP proxies come in all three packages. Residential and mobile proxies are almost always pools, because nobody can promise that one household's device stays online.

IPv4 or IPv6

IPv4 has about 4.3 billion addresses and the world ran out of fresh ones years ago. Every IPv4 proxy sits on an address somebody leases, which is why per-IP pricing exists. IPv6 is the opposite. A single /64 subnet holds more addresses than the whole IPv4 internet, so IPv6 proxies are cheap.

Two catches come with the low price. Websites know IPv6 addresses cost nothing, so they tend to score a whole subnet as one visitor. A thousand addresses from one /64 can count as one. The second catch is reach: a site that publishes no IPv6 address cannot be visited from one.

SHELL · DIG

```
# Does the target accept IPv6? Ask DNS for its AAAA records.
dig AAAA example.com +short

# One or more IPv6 addresses printed: IPv6 proxies can reach it.
# Nothing printed: the site is IPv4 only. Buy IPv4 for this target.

# Repeat for every hostname the job touches, including API hosts.
dig AAAA api.example.com +short
```

Where residential IPs come from, and why consent matters

A residential network is built from other people's devices. Someone installs a free app on a phone, laptop or smart TV. The app includes a bandwidth-sharing component. While the device sits idle, requests from the provider's customers leave through that home connection. Sourcing follows one of three models.

1. **An SDK inside third-party apps.** The app shows an opt-in screen and offers something in return, such as fewer ads or a premium feature.
2. **A dedicated bandwidth-sharing app.** The device owner installs it on purpose and is paid for the traffic carried.
3. **No real consent.** Malware, hijacked routers or a clause buried in terms nobody reads. Reputable providers stay away from this, and buyers should too.

Consent matters to the buyer for practical reasons as well as ethical ones. Traffic sent through a device whose owner never agreed creates legal and reputational exposure for whoever sent it. A pool built on hijacked devices can also vanish overnight when the malware is cleaned up.

- [ ] Ask how peers are recruited, and ask to see the consent screen.
- [ ] Ask whether peers are paid or compensated, and how they opt out.
- [ ] Ask whether the provider vets its own customers and blocks abusive use.
- [ ] Treat a vague or missing answer as the answer.

KEY IDEA **Pool size is a vendor-stated number**

Oxylabs states 177M+ IPs and Bright Data states 150M+. These are the vendors' own totals. A job depends on something narrower: how many peers are online in the target city at the moment of the request. Only a trial shows that.

What each type costs

Proxies are billed in several units: per GB, per IP, per port or thread, per request, or as flat daily and monthly plans. The two common ones are per GB for rotating pools and per IP for static addresses.

Product	Billing unit	Rough entry-level price
Rotating residential	Per GB	Roughly $1 to $5, less at volume
Rotating mobile	Per GB	Roughly $1 to $8.50
Rotating or shared datacenter	Per GB	Roughly $0.10 to $1.40
Dedicated datacenter	Per IP per month	Roughly $1 to $2.50
Static ISP	Per IP per month	Roughly $1 to $3.50

- Entry-level list prices in the ProxyLook directory, compiled November 2026. They change often.

Treat the table as orientation only. Live figures for each provider sit on the ProxyLook pricing page. The billing unit matters as much as the number. Per-GB plans punish heavy pages full of images and video. Per-IP plans reward heavy traffic through a few addresses.

A quick estimate prevents surprises. A job that fetches 100,000 pages at 2 MB each moves about 200 GB. Blocking images, fonts and video in the scraper cuts that figure, and the bill, directly. Put the monthly total in GB into the cost calculator before choosing a plan. Scraper APIs and unblockers sit outside the table. They are priced per successful request or per monthly plan, and the live pages carry the figures.

Checking what an IP really is

A label on a pricing page is a claim. An ASN lookup is evidence. Verifying a new proxy takes a few minutes with two free ProxyLook tools.

1. **Find the exit IP.** Set a browser to use the proxy and open What Is My IP. It shows the public IP, location, ISP and ASN that websites see.
2. **Look up the network type.** Paste the IP into the proxy checker. It returns the country, ASN, ISP and a network type of residential, datacenter or mobile.
3. **Compare with the order.** An ISP proxy whose ASN comes back as hosting is a datacenter proxy to any site using similar data.

4. **Sample more than one.** Check several IPs from a static list, or several fresh sessions from a rotating pool, at different times of day.

WATCH OUT **One database is one opinion**

The network type comes from an IP database, and databases disagree at the edges. A target site may use a different one. Treat a clean result as a good sign and a bad result as a reason to ask the provider for a replacement.

KEY TAKEAWAYS

- The four proxy types differ in who owns the IP, and websites read that from the ASN category.
- Default trust runs mobile, residential, ISP, then datacenter, and cost follows the same order.
- Start with the cheapest type that works on the target and climb the ladder only when blocks appear.
- Shared IPs are cheap but inherit a neighbor's behavior, while dedicated IPs carry only the buyer's own.
- Residential pools run on other people's devices, so ask every provider how consent is obtained.
- Verify the ASN and network type of a new proxy before building a job around it.

CHAPTER 03

Protocols, Ports and Authentication

How HTTP, HTTPS and SOCKS5 proxies carry traffic, what each can see, how to authenticate, and how to read and test a proxy URL.

- HTTP proxies and the CONNECT tunnel
- Passwords versus IP allowlisting
- SOCKS5, UDP and remote DNS
- Proxy URL anatomy and curl tests

A provider's dashboard hands over one line: `gate.example-proxy.com:7000:USERNAME:PASSWORD`. Whether that line works in curl, in Python or in a browser depends on three things it does not say: the protocol, the meaning of the port and the way the client proves who it is.

HTTP proxies and the CONNECT tunnel

An HTTP proxy handles two kinds of traffic in two different ways. For a plain HTTP page, the client sends the whole request to the proxy, full URL included. The proxy reads it, fetches the page and returns it. The proxy can log, cache or rewrite everything.

For an HTTPS page, the client asks for a tunnel instead. It sends one short request with the `CONNECT` method, naming a host and a port. The proxy opens a TCP connection to that host and answers 200. From then on it copies bytes in both directions without understanding them.

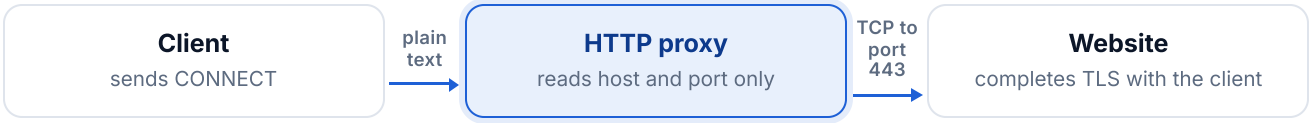
HTTP · CONNECT EXCHANGE

```
CONNECT example.com:443 HTTP/1.1
Host: example.com:443
Proxy-Authorization: Basic VVNFUk5BTUU6UEFTU1dPUkQ=

HTTP/1.1 200 Connection established

[TLS handshake between client and example.com]
[encrypted requests and responses]
```

The first three lines are the only part the proxy can read. The TLS handshake that follows runs between the client and the website, end to end. This is why a proxy whose URL starts with `http://` carries HTTPS sites without trouble. The scheme describes how the client talks to the proxy, not what travels inside.



A CONNECT tunnel. The proxy relays an encrypted session it cannot open.

What the proxy can and cannot see

The difference between the two modes decides how much trust a proxy operator needs. On plain HTTP the operator sees everything. On HTTPS the operator sees where the traffic goes and how much of it there is.

Item	Plain HTTP target	HTTPS target
Hostname and port	Visible	Visible, from the CONNECT line
Full URL and path	Visible	Hidden
Headers and cookies	Visible, and can be changed	Hidden
Page content and form data	Visible, and can be changed	Hidden
Timing and volume of traffic	Visible	Visible

What a forward proxy can read for plain HTTP and for HTTPS traffic.

Two practical rules follow. Never send a login or an API key to a plain HTTP address through a proxy run by an unknown operator. And be wary of any proxy that asks the client to install a certificate or switch off TLS verification. That request means the proxy intends to decrypt the traffic.

HTTPS proxies, SOCKS5 and the HTTP/3 gap

HTTPS proxies

Look again at the CONNECT exchange. The credentials sit in a `Proxy-Authorization` header, encoded in Base64. That is an encoding, not encryption. Anyone on the same coffee-shop Wi-Fi can decode it, along with the hostname.

An HTTPS proxy closes that gap. The client first opens a TLS connection to the proxy itself, then sends CONNECT inside it. The proxy URL starts with `https://`. Not every provider offers a TLS port and not every client supports one, so test before relying on it.

SOCKS5

SOCKS5 works one level lower. The client says "connect to this host on this port" and then sends raw bytes. The proxy never parses HTTP, so it adds no headers and carries any TCP protocol: web pages, mail, a database session. SOCKS5 proxies have three properties worth knowing.

- **No encryption of its own.** SOCKS5 passwords cross the network in clear text. HTTPS inside the tunnel is still protected.
- **UDP on paper.** The protocol defines a UDP relay. Many commercial proxies do not switch it on, so ask before building on it.
- **A DNS choice.** The client can resolve the hostname itself or let the proxy do it. In URL form, `socks5://` resolves locally and `socks5h://` resolves at the proxy.

For ordinary HTTPS scraping the website cannot tell the two protocols apart. HTTP proxies have the widest tool support and make the sensible default. SOCKS5 earns its place when the traffic is not HTTP or the tool speaks nothing else.

The HTTP/3 caveat

HTTP/3 runs on QUIC, and QUIC runs on UDP. A CONNECT tunnel carries TCP only. A browser behind an HTTP proxy therefore falls back to HTTP/2 or HTTP/1.1. Scripts do the same. Sites accept this, because many office networks block QUIC too. A newer standard for proxying UDP exists, but it should not be assumed in a commercial proxy product.

> **WATCH OUT** **UDP can slip past the tunnel**
>
> Where software forces a whole machine's traffic into a proxy, UDP is sometimes left out. QUIC and WebRTC packets then leave from the real IP. Disable HTTP/3 in the client or block outbound UDP on port 443, and run the WebRTC leak test in any browser profile.

Two ways to authenticate

A paid proxy must know which customer to bill. It learns that in one of two ways. With username and password, the client sends credentials on every new connection. With IP allowlisting, the customer registers a source IP in the dashboard, and the proxy accepts anything arriving from it.

Credentials travel with the code, so they work from a laptop, a server or a container that started ten seconds ago. They also carry settings: most providers read country and session options from the username, as Chapter 4 shows. The cost is a secret that can leak into logs and repositories.

An allowlist keeps secrets out of the code and suits tools that cannot send proxy credentials. Selenium driving Chrome is the usual example. The weakness is the IP itself. An allowlist breaks the moment the source IP changes, and it lets everyone behind the same office IP spend the plan.

Browsers add one trap. Chrome has long been unable to send a username and password to a SOCKS5 proxy. In a Chromium-based browser, use an HTTP proxy with credentials, or a SOCKS5 port opened by allowlist.

If...	Then...
The job runs on one server with a fixed IP	Allowlist that IP and drop the credentials
Workers autoscale, or run on laptops and home broadband	Use username and password
The tool has no field for proxy credentials	Allowlist, or run from a fixed-IP server
Targeting options go in the username	Use username and password
Several people share one office IP	Give each person separate credentials

Choosing between credentials and IP allowlisting.

One status code saves hours of confusion. A `407 Proxy Authentication Required` comes from the proxy and means wrong or missing credentials. A 403 that arrives inside the tunnel comes from the website and is a different problem.

Anatomy of a proxy URL

Most libraries want the dashboard's four fields rearranged into a URL: `http://USERNAME:PASSWORD@gate.example-proxy.com:7000`. Each part has one job.

Part	Example	What it controls
Scheme	`http://`	Protocol spoken to the proxy: http, https, socks5 or socks5h
Username	`USERNAME`	The account, often with targeting options appended
Password	`PASSWORD`	The secret. Special characters need percent-encoding
Host	`gate.example-proxy.com`	A gateway hostname, or one static proxy IP
Port	`7000`	The service: protocol, product or session behavior

The five parts of a proxy URL.

The port deserves a second look. By convention Squid listens on 3128, many HTTP proxies use 8080 and SOCKS uses 1080. Commercial gateways ignore convention and give ports meaning: one for rotating IPs, a range for sticky sessions, another for SOCKS5. Speaking HTTP to a SOCKS port produces a hang or a garbled error, never a clear message. The password causes the other classic failure. An `@` or `:` inside it must be written as `%40` or `%3A`.

Curl is the fastest way to test a URL before it goes into code. The first command below prints the exit IP. The second shows the CONNECT exchange and any 407. ProxyLook's proxy checker also writes a curl test command for any IP and port pasted into it.

SHELL · CURL

```
# HTTP proxy, HTTPS target. Prints the exit IP as JSON.
curl -x http://USERNAME:PASSWORD@gate.example-proxy.com:7000 \
     --max-time 30 https://httpbin.org/ip

# Verbose mode: shows the CONNECT request and the proxy's reply.
curl -v -x http://USERNAME:PASSWORD@gate.example-proxy.com:7000 \
     --max-time 30 -o /dev/null https://example.com
```

The SOCKS5 form differs in one flag. Use the provider's SOCKS5 port where it has a separate one.

SHELL · CURL

```
# SOCKS5 with remote DNS: the proxy resolves the hostname.
curl --socks5-hostname gate.example-proxy.com:7000 \
     --proxy-user USERNAME:PASSWORD \
     --max-time 30 https://httpbin.org/ip

# The same request written as a proxy URL.
curl -x socks5h://USERNAME:PASSWORD@gate.example-proxy.com:7000 \
     --max-time 30 https://httpbin.org/ip
```

TIP **Check the environment first**

Curl, Python requests and most command-line tools read the `http_proxy` and `https_proxy` environment variables. That helps when a tool has no proxy setting of its own. It also explains traffic that uses a proxy nobody remembers configuring, so look in the shell profile.

DNS leaks

A scraper in Chicago uses a German SOCKS5 proxy, written as `socks5://`. Before each request, the machine asks its local DNS server for the target's address. Only then does it hand the proxy an IP to connect to. That lookup is a DNS leak.

The leak has three costs. The local network and its DNS provider learn every hostname the job touches. The answer is tuned for Chicago, so a site that steers visitors by DNS hands a German exit its nearest US servers. And in a browser, a site can compare the resolver's location with the IP's location and notice the gap.

- [ ] HTTP and HTTPS proxies: the hostname goes to the proxy, which resolves it. No action needed.
- [ ] curl: use `--socks5-hostname` or a `socks5h://` URL, never plain `--socks5`.
- [ ] Python requests: run `pip install "requests[socks]"` and write the scheme as `socks5h://`.
- [ ] Firefox: tick "Proxy DNS when using SOCKS v5" in the connection settings.
- [ ] Verify: capture traffic on port 53 during a run. Target hostnames there mean a leak.

KEY TAKEAWAYS

- An HTTP proxy carries HTTPS through a CONNECT tunnel and sees only the hostname, port, timing and volume.
- On plain HTTP targets the proxy can read and change everything, so keep secrets off them.
- SOCKS5 carries any TCP protocol, adds no encryption and needs the socks5h form for remote DNS.
- Proxied traffic falls back from HTTP/3 to HTTP/2 or HTTP/1.1 because CONNECT tunnels carry TCP only.
- Use credentials when source IPs change or options ride in the username, and allowlisting for fixed servers.
- Test every new proxy URL with curl first, and read a 407 as a message from the proxy.

CHAPTER 04

Rotation, Sessions and Geo-Targeting

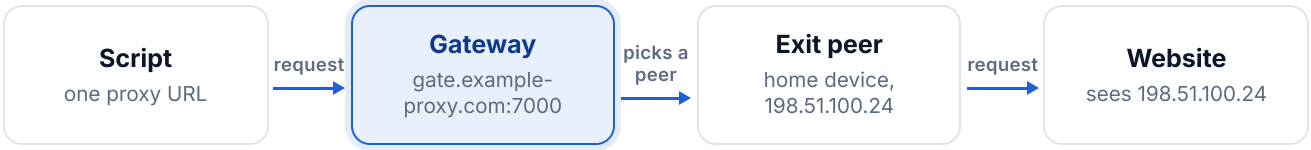
How a backconnect gateway hands out IPs, when to rotate or hold a session, and how location targeting trades precision for pool size.

- How a backconnect gateway works
- Country, city, ZIP and ASN targeting
- Rotating versus sticky sessions
- Sticky and rotating code in Python

A script sends three requests to the same proxy address, `gate.example-proxy.com:7000`. The target site logs three different visitors: one in Ohio, one in Texas, one in Oregon. The script changed nothing between requests. The gateway did.

One address, many exits: the backconnect gateway

Residential and mobile IPs cannot be handed out as a list. The devices behind them switch off, change networks and come back with new addresses. Providers solve this with a backconnect gateway: one stable hostname in front of a pool that never stops moving.



- A backconnect gateway. The script talks to one address and exits through whichever peer the gateway picks.

The client connects to the gateway and never learns the exit IP in advance. The gateway reads the username, picks a peer that matches and relays the traffic. Each peer keeps a connection open back to the provider, which lets the gateway reach devices that sit behind home routers.

Two consequences follow. Every request takes an extra hop through a consumer connection, so latency is higher and less even than on a datacenter proxy. Every choice about the exit, location and lifetime included, is made through the proxy credentials.

Rotate per request, or hold a sticky session

A gateway offers two main behaviors. In rotating mode, each new connection gets a fresh IP from the pool. In sticky mode, the client names a session and the gateway keeps sending that session through the same peer.

One detail trips up many first scripts. Rotation happens per connection, not per HTTP request. A client that keeps a connection alive and reuses it will hold the same exit IP for as long as that connection lasts. A scraper that expects a new IP on every page must open a new connection for every page.

A third mode sits between the two: timed rotation. The gateway holds one IP for a set interval and then swaps it, with no session ID involved. Dedicated mobile ports often work this way, and many add a link that forces a new IP on demand.

How long a sticky session lasts

A sticky session ends in one of three ways. The provider's time limit runs out, anywhere from a minute to several hours depending on the product. The client stops using the session ID. Or the peer disappears, because a phone left its Wi-Fi or a laptop lid closed.

The third case arrives without warning. Some gateways return an error, and others quietly continue on a new IP. Code that depends on one IP should check for a change and restart the flow, not carry on halfway through a login. Restarting means a new session ID and a clean cookie jar, from step one.

If...	Then...
The job fetches thousands of unrelated pages	Rotate on every connection
A flow spans several steps: login, search, cart	Hold a sticky session for the whole flow
The site binds a cookie or token to the IP that received it	Reuse the same session with that cookie
The target limits requests per IP	Rotate, and still pace each IP politely
An account must keep one IP for days or weeks	Buy a static ISP proxy, not a sticky session

Choosing between rotation, sticky sessions and static IPs.

The last row matters for account work. A sticky session is built for minutes or hours. No residential peer can promise next week. Running multiple accounts can also breach a platform's terms of service, and the operator carries that risk.

Sessions and targeting in the username

Most gateways read their instructions from the proxy username. The account name comes first, followed by options. The lines below show the idea.

TEXT · ILLUSTRATIVE USERNAMES

```
# Illustrative only. Exact syntax comes from the provider's docs.

USERNAME                              rotate, any country
USERNAME-country-us                   rotate, United States only
USERNAME-country-us-session-a1b2c3    hold one US exit IP
USERNAME-country-us-session-x9y8z7    hold a second US exit IP
USERNAME-country-de-city-berlin       rotate, Berlin only
```

The session ID is a string the client invents. The gateway has never seen `a1b2c3` before, so it assigns a peer and remembers the pairing. The same ID returns the same IP until the session ends. A new ID gets a new IP.

That makes parallel identities cheap. Fifty session IDs through one gateway give fifty held IPs, each with its own cookie jar in the client. Dropping an ID and inventing another is how a script asks for a fresh IP on demand.

Traffic is metered at the gateway on per-GB plans. Blocked pages, retries and abandoned sessions all count toward the bill, which is one more reason to match the mode to the job.

WATCH OUT **Syntax differs by vendor**

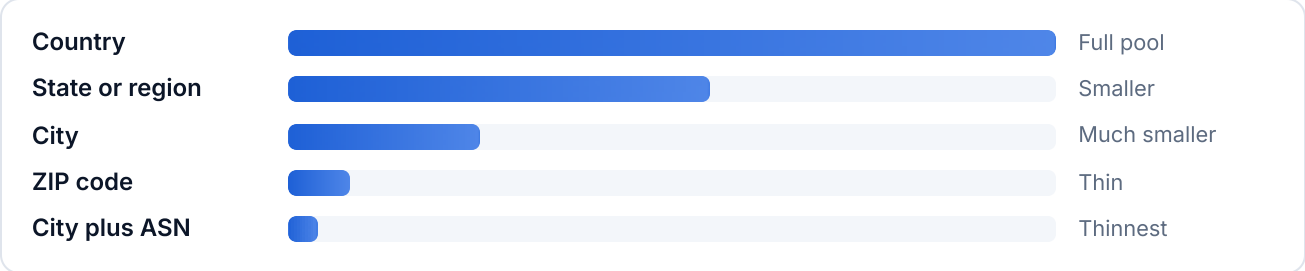
Separators, parameter names and the order of options change from one provider to the next. Some use dedicated ports or hostnames in place of username options. Copy the exact format from the provider's documentation or its dashboard URL builder.

Geo-targeting: country, state, city, ZIP, ASN

A German retailer shows German prices to German IPs. A search engine shows different local results in Dallas and in Houston. Geo-targeting lets a job pick the exit location to match, at five levels of precision.

- **Country.** The default level, and enough for prices, catalogs and most content checks.
- **State or region.** Useful where laws, taxes or offers differ inside a country.
- **City.** Needed for local search results, maps listings and delivery checks.
- **ZIP or postal code.** Offered by few providers, for store-level stock and local advertising.
- **ASN or carrier.** Picks the network, not the place. Ad verification on one mobile carrier is the typical job.

Each filter can only remove peers from the pool. Bright Data, Decodo and Oxylabs each state coverage of 195 countries, by their own count. A country count says little about how many peers are online in one mid-sized city on a weekday morning.



Illustrative, not measured: every added filter leaves fewer usable IPs.

A thin pool shows up in three ways. The same IPs come around again sooner, so per-IP rate limits bite earlier. More requests fail because no matching peer is free. And some providers charge extra for city or ZIP targeting, or keep it for higher plans.

The rule is to target at the widest level the data requires. Country-level prices do not need a city. When a narrow pool runs dry, providers differ: some fail the request and some fall back to a wider area without saying so. Open What Is My IP through the proxy to see where an exit really lands.

KEY IDEA **Location is an estimate**

A provider places an IP in Dallas using one geolocation database. The target site may use another that says Fort Worth. City-level IP location is approximate everywhere, so allow for a share of near misses in any city-targeted job.

Static IP lists versus gateways

Datacenter and ISP proxies usually arrive in the other format: a file of fixed addresses such as `198.51.100.24:8000`, one per line. The two formats shift work between the provider and the client.

Static list

Each IP stays the same until the plan ends or the IP is replaced. The client code does the rotating: it picks from the list, tracks which IPs a target has blocked and rests them.

Gateway

One hostname covers the whole pool. The provider rotates, replaces dead peers and applies targeting. The client cannot know an exit IP ahead of time.

A static list is the right choice when the IP itself has value: a long-lived account, or a partner system that allowlists known addresses. A gateway is the right choice when the IP is disposable and volume matters. ProxyLook's rotating proxy page lists providers that sell gateway access.

Static lists need some care that gateways hide. Ask for IPs spread across several subnets, because sites often block a whole range when a few addresses in it misbehave. Keep a per-IP record of blocks and response times, and return dead addresses to the provider for replacement.

Sticky and rotating in Python

The first example holds one exit IP across three requests. The session ID in the username pins the IP. The `requests.Session` object keeps cookies, so the site sees one consistent visitor.

PYTHON · REQUESTS

```python
import uuid
import requests

URL = "https://httpbin.org/ip"
GATEWAY = "gate.example-proxy.com:7000"
session_id = uuid.uuid4().hex[:8]
user = f"USERNAME-country-us-session-{session_id}"
proxy = f"http://{user}:PASSWORD@{GATEWAY}"
proxies = {"http": proxy, "https": proxy}

with requests.Session() as s:
    for _ in range(3):
        r = s.get(URL, proxies=proxies, timeout=30)
        r.raise_for_status()
        print(r.json()["origin"])  # the same IP three times
```

The second example rotates. It has no session ID, and it avoids a shared `Session` on purpose. Each call to `requests.get` opens a new connection, so the gateway assigns a new peer each time.

PYTHON · REQUESTS

```python
import requests

URL = "https://httpbin.org/ip"
GATEWAY = "gate.example-proxy.com:7000"
proxy = f"http://USERNAME-country-us:PASSWORD@{GATEWAY}"
proxies = {"http": proxy, "https": proxy}

for _ in range(3):
    # A new connection per call, so a new exit IP per call.
    r = requests.get(URL, proxies=proxies, timeout=30)
    r.raise_for_status()
    print(r.json()["origin"])  # normally three different IPs
```

Run both before writing anything larger. If the sticky script prints different IPs, the username format is wrong for that provider. If the rotating script prints one IP three times, the plan or port is set to sticky by default. Either result is cheaper to find in a ten-line test than in a job that has already spent its traffic.

KEY TAKEAWAYS

- A backconnect gateway is one address in front of a moving pool, and the username tells it what to do.
- Rotation happens per connection, so a reused connection keeps its exit IP.
- A sticky session holds an IP for minutes or hours and can end early when the peer goes offline.
- Accounts that need the same IP for weeks belong on static ISP proxies, not sticky sessions.
- Every geo filter shrinks the pool, so target at the widest level the data requires.
- Username syntax is vendor-specific, so copy it from the provider's documentation and verify the exit IP.

CHAPTER 05

Matching Proxies to the Job

Eight short playbooks that pair common jobs with a proxy type, a rotation style and the pitfall to avoid, plus a ladder for escalating only when needed.

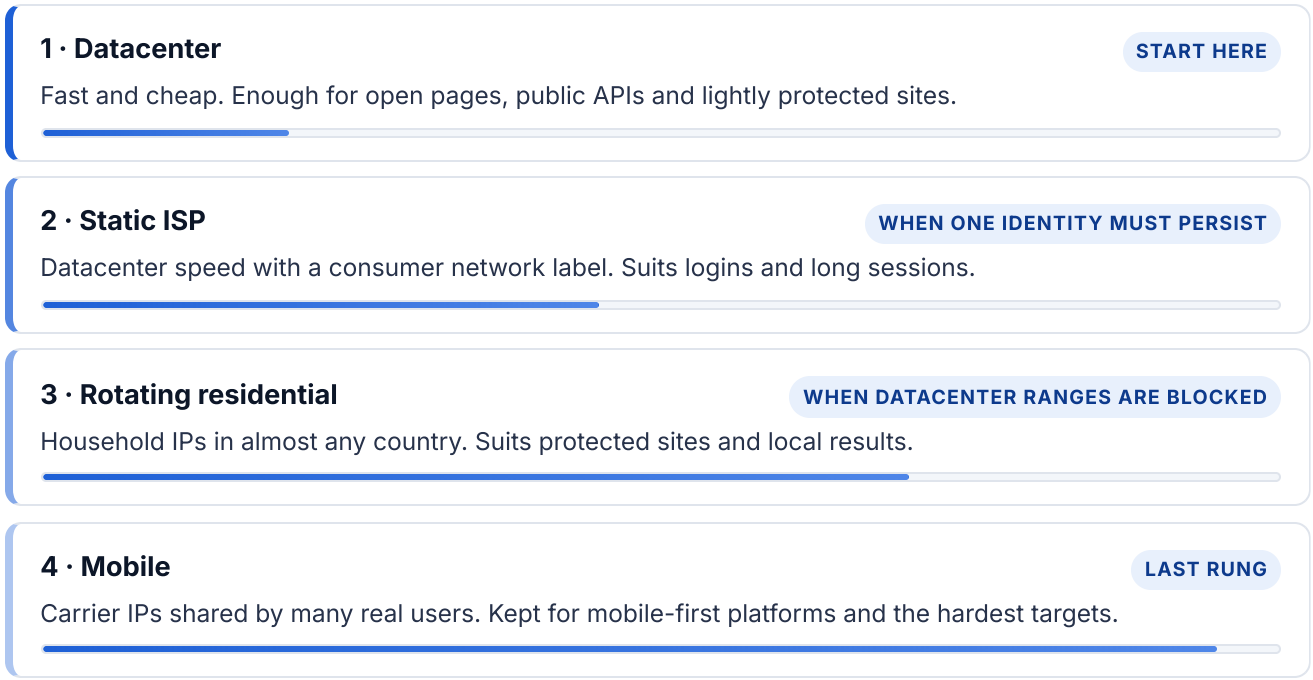
- Starting cheap and escalating on evidence
- An if/then decision table
- Eight playbooks by use case
- Signs of a mismatched setup

One team checks 5,000 keyword rankings each morning and wants thousands of short-lived IPs. Another collects air fares through a five-step search and wants one address that holds still for ten minutes. Both say they need proxies.

Start cheap, escalate on evidence

A product catalog that loads through a shared datacenter IP gains nothing from a mobile IP except a larger bill. Chapter 2 ranked the four proxy types by trust, and cost climbs in the same order. The right choice is the cheapest type that completes the job.

In practice that rule is a ladder. Start on the first rung, run a real sample, and climb only when the results demand it.



The escalation ladder, first rung at the top. The meter is a rough guide to trust and cost, not a measured value.

The ladder is a default order, not a law. Rungs two and three solve different problems. A static ISP address is the step up when one identity must stay put for weeks. Rotating residential is the step up when a site blocks datacenter ranges outright, or when the job needs many locations.

Rule out the free fixes first

A blocked request does not always mean the wrong proxy type. A bare HTTP client with default headers gets blocked on residential IPs too. Chapter 10 covers each fix in detail. The short version is five steps, in this order.

1. **Run a sample.** Send a few hundred real requests to the real target. Record how many return the expected content.
2. **Fix the client.** Send realistic headers, keep cookies where the site expects them, and slow the request rate.
3. **Fix the rotation.** Switch between per-request rotation and sticky sessions to match how the site tracks a visit.
4. **Fix the location.** Use exit IPs in a country the site actually serves.
5. **Climb one rung.** If success is still below target, move up one proxy type and rerun the same sample.

EXAMPLE **Escalate per target, not per project**

Picture a price tracker that covers ten stores. Nine load cleanly through datacenter IPs, and one large marketplace returns CAPTCHAs. Only that store's requests move to residential, so the bill rises for one target instead of ten.

Playbooks for collecting data

The four jobs below pull public pages at volume. They differ in how hard the targets push back and in how much location matters.

Web scraping

- **Proxy:** datacenter first, rotating residential for sites that block datacenter ranges.
- **Rotation:** a new IP per request for independent pages, sticky for pagination tied to a cookie.
- **Pitfall:** paying residential rates for pages a datacenter IP fetches without trouble.

SEO and SERP tracking

- **Proxy:** rotating residential with country and city targeting, or a SERP API billed per request.
- **Rotation:** a new IP for every query.
- **Pitfall:** results change with location, language and device. A ranking checked from the wrong city is a wrong ranking.

Price monitoring and e-commerce

- **Proxy:** datacenter for small stores, residential for large marketplaces with strong bot defenses.
- **Rotation:** per request for product pages, sticky when a delivery postcode or currency must persist.
- **Pitfall:** prices, stock and shipping vary by country. An exit in the wrong market records the wrong price.

Travel fare collection

- **Proxy:** residential in the country where the ticket would be sold.
- **Rotation:** one sticky session per search, from first query to final fare.
- **Pitfall:** rotating mid-search breaks the session. Script-heavy booking pages also eat per-GB traffic.

One pattern runs through all four. Independent pages rotate, and multi-step flows stick. A scraper reading 10,000 unrelated product pages can use 10,000 different IPs. A fare search that carries a session token from page one to page five cannot.

> **The target's defenses pick the proxy type, and the shape of the task picks the rotation.**

ProxyLook keeps a page of matching providers for each of these jobs, including web scraping, SERP tracking and travel fares.

WATCH OUT **Collection is not one legal question**

Public pages, personal data, content behind a login and a site's terms of service each raise different legal questions. The answers also change by country. Nothing in this book is legal advice.

Playbooks for verification, accounts and AI

The next four jobs care less about volume. Each one has to look like a specific kind of visitor in a specific place.

Ad verification

- **Proxy:** residential or mobile in the exact market where the campaign runs.
- **Rotation:** sticky for one page visit, then a new IP for the next check.
- **Pitfall:** IP location databases disagree. Confirm the exit city before trusting what the ad server showed.

Social media management

- **Proxy:** one static ISP or mobile IP per account, in the account's home country.
- **Rotation:** none. The same account keeps the same address.
- **Pitfall:** rotating under a logged-in account, or stacking many accounts on one IP, ties them together.

Market research

- **Proxy:** residential spread across the countries being studied, datacenter for open sources.
- **Rotation:** per request for breadth, sticky when browsing a local site as one visitor.
- **Pitfall:** thin coverage. A network can be large overall and still hold few IPs in a small market.

AI agents and training data

- **Proxy:** sticky residential or ISP for agents that browse, datacenter first for bulk crawls.
- **Rotation:** one session per agent task, per-request rotation for crawls.
- **Pitfall:** an agent that changes IP mid-task loses its session. Full browsers also multiply traffic per page.

AI work is two jobs sharing one name. An agent that compares products or fills in forms behaves like one user and needs one stable session. A training-data crawl behaves like a scraper and follows the scraping playbook, with extra care over licensing and personal data.

WATCH OUT **Multiple accounts carry platform risk**

Managing multiple accounts on one platform can breach that platform's terms of service, even when each account belongs to a real client. A separate IP per account lowers the chance of accounts being linked. It does not make the practice permitted, and the operator carries the risk.

Account work also depends on browser fingerprints, cookies and behavior, which a proxy does not touch. ProxyLook's companion book, The Multi-Account Handbook, covers that side in full.

A decision table for everything else

Plenty of jobs fit no named playbook. A compliance team checking how its own site renders abroad is one example. The conditions below usually settle the choice.

If...	Then...
The page loads the same for any visitor, with no blocks	Shared datacenter, new IP per request
An account must look like one person for weeks	One static ISP or mobile IP per account
Results change by country or city	Residential with geo-targeting, and verify the exit location
The task spans several linked pages	A sticky session that outlasts the whole flow
Datacenter IPs meet blocks or CAPTCHAs on the first request	Step up to rotating residential
The platform is mobile-first or app-based	Mobile IPs from a carrier in the target country
Pages are heavy and billing is per GB	Block images and media, or price a per-IP plan
Residential still fails with a well-built client	Price a scraper API or unblocker billed per success

Conditions that decide the proxy type and rotation style when no playbook fits.

Two rows can apply at once. A logged-in account that also needs city-level results takes the stricter answer for each part: a static IP, located in that city. For anything the table leaves open, the ProxyLook assistant takes a plain description of a job and suggests matching providers from the directory.

Signs of a mismatch

A setup that fit at launch can drift as targets and volumes change. The signs below are worth a monthly look at the usage dashboard.

- [ ] Residential traffic is going to a site that never blocked datacenter IPs.
- [ ] A logged-in account shows a new IP on every request.
- [ ] Per-GB traffic is spent on images, fonts and video the job never reads.
- [ ] The exit country differs from the market the data is meant to describe.
- [ ] Success fell after a volume increase, and nobody lowered the request rate per IP.
- [ ] Mobile IPs are in use out of habit, not because a test demanded them.

Stepping down is as valid as stepping up. Sites change their defenses in both directions, and a target that needed residential IPs last year may accept datacenter IPs today. Rerun the original sample on a cheaper rung every few months. If it passes, the cheaper rung is the right one.

KEY TAKEAWAYS

- Start with the cheapest proxy type that completes the job, and climb one rung at a time on evidence.
- Fix headers, request rate, rotation and location before paying for a higher proxy type.
- Independent pages suit per-request rotation, while multi-step flows and logins need sticky or static IPs.
- Location is part of the data for SERP, price, fare and ad checks, so verify the exit location.
- Managing multiple accounts can breach platform terms, and no proxy removes that risk from the operator.

CHAPTER 06

How Proxy Pricing Works

How proxies are billed, what list prices roughly look like, and how to turn any plan into one comparable number: cost per successful request.

- Five billing models compared
- Cost per successful request
- Rough price ranges by proxy type
- Bandwidth math and hidden costs

A residential plan is sold by the gigabyte, but a scraper has no use for gigabytes. It needs pages that load with the right content. Most pricing mistakes sit in the gap between the unit on the invoice and the unit the job needs.

Five ways to bill for a proxy

Providers in the ProxyLook directory bill in five ways. The model matters as much as the price, because each one rewards a different shape of traffic.

Model	What is metered	Usually sold with	Suits	Watch for
Per GB	Traffic through the proxy	Rotating residential and mobile	Light pages, many IPs	Heavy pages and browser traffic
Per IP	Each address, per month	Dedicated datacenter, static ISP	Logins and long sessions	Traffic caps, limits on swapping IPs
Per port or thread	Concurrent connections	Rotating pools, mobile ports	Steady, bandwidth-heavy work	A fixed ceiling on parallel requests
Per request	Successful responses	Scraper APIs and unblockers	Hard targets, small teams	Surcharges for JavaScript rendering
Flat "unlimited"	Days or months of access	Unlimited-traffic plans	Very high bandwidth	Speed caps and fair-use clauses

- The five billing models seen in the ProxyLook directory.

Two jobs show the difference. A QA team keeps full browser sessions open on 30 regional versions of a site all day. That is heavy traffic over few addresses, so paying per IP wins. A crawler touches a million light pages through as many IPs as it can get. That is light traffic over many addresses, so paying per GB wins.

An "unlimited" plan always has a limit somewhere. It may be the number of threads, the speed of each connection, the size of the pool or a fair-use clause. The plan is still good value when that limit sits above what the job needs.

What list prices look like

Chapter 2 introduced the ranges below, and every calculation in this chapter starts from them. They are entry-level list prices in the ProxyLook directory at the time of writing. They are rough, they change often, and volume commitments push them lower. Live figures are on the ProxyLook pricing page.

Product	Billing unit	Rough entry-level range
Rotating residential	Per GB	Roughly $1 to $5
Rotating mobile	Per GB	Roughly $1 to $8.50
Rotating or shared datacenter	Per GB	Roughly $0.10 to $1.40
Dedicated datacenter IPs	Per IP per month	Roughly $1 to $2.50
Static ISP (static residential) IPs	Per IP per month	Roughly $1 to $3.50
Scraper APIs and unblockers	Per successful request or monthly plan	See live pages

Rough entry-level list prices by product. Ranges change often and are not quotes.

Two things stand out. At the bottom of each range, datacenter traffic costs about a tenth of residential traffic. That gap is why the ladder in Chapter 5 starts with datacenter IPs. The ranges are also wide. The top of the residential range is five times the bottom, and a headline price does not explain the spread.

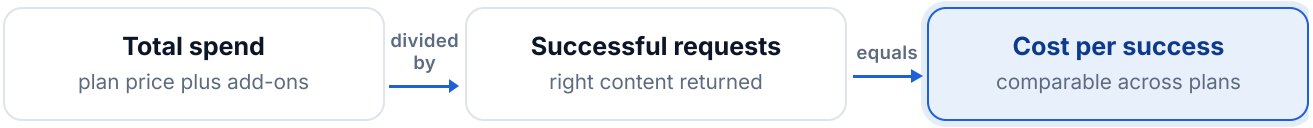
The entry price is usually the highest rate a provider charges. Larger commitments and promotions lower it, and current offers are listed on the deals page.

Cost per successful request

Price per GB and price per IP cannot be compared with each other. Neither says how often a request comes back with the right content. One number puts every plan on the same scale.

KEY IDEA **The formula**

Cost per successful request = total spend ÷ requests that returned the right content. For a per-GB plan, total spend is the price per GB × the gigabytes used, including traffic burned by failures and retries.



- Any billing model reduces to one comparable figure.

The example below uses made-up round numbers to show the arithmetic. It is an illustration, not a benchmark of any provider. Assume every attempt transfers 500 KB whether it succeeds or fails. One gigabyte, taken as 1,000 MB, then pays for 2,000 attempts.

Plan	Price per GB	Success rate	Good pages per GB	Cost per 1,000 good pages
Plan A	$2.00	50%	1,000	$2.00
Plan B	$3.00	90%	1,800	$1.67

- Illustration only, with invented round numbers. Not measured data and not a benchmark.

Plan B costs 50 percent more per gigabyte and about a sixth less per usable page. The plan with the lower price is the expensive one. It also needs nearly twice as many attempts to finish the same job, which means slower runs and more load on the target.

Real traffic is messier than the example. A block page usually weighs less than a full page, so failures cost less than successes. The only reliable inputs are measured ones: total spend and a count of good responses from a trial.

WATCH OUT **A 200 is not a success**

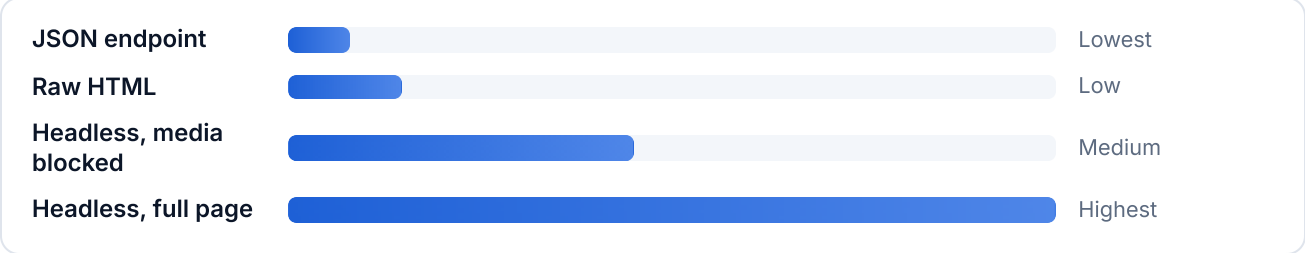
Many sites answer a suspected bot with a normal status code and a CAPTCHA or an empty shell. Count a request as successful only when the response contains something the real page has, such as a price or a result list.

The same figure works for other models. For a per-IP plan, divide the monthly fee by the good requests that IP delivered. For a scraper API, the price is already per success, so the task is to check how the vendor defines success.

Bandwidth math

Per-GB budgets start with one multiplication: pages per month times average traffic per page, plus an allowance for retries. Take 100,000 product pages a month as an example. Fetched as raw HTML at 200 KB each, they use 20 GB. Loaded in a headless browser at 2 MB each, the same pages use 200 GB. The weights are made up for the example. The direction is not: a full browser load moves more data than a single HTML request.

The reason is simple. An HTTP client downloads one document. A browser downloads that document and then everything it references: images, style sheets, scripts, fonts, video and analytics calls. The scripts then request more.



Relative traffic per page by fetch method. Bar lengths are illustrative, not measurements.

Four habits keep the meter down:

- Fetch the JSON endpoint behind a page when the page loads its data from one.
- Block images, fonts and media in the browser when the job only reads text.
- Cache anything that does not change between runs.
- Measure instead of guessing. Run 100 pages, read the usage meter, then multiply.

Many providers meter traffic in both directions, so request headers and uploads count too. Chapter 11 returns to these savings at scale. ProxyLook's proxy cost calculator turns a monthly traffic figure in GB into an estimated spend.

Hidden costs

The headline rate is rarely the whole bill. Six items deserve a line in any comparison sheet.

Failed requests

A per-GB plan bills bytes, not outcomes. Block pages, CAPTCHAs and half-finished downloads all count.

Minimum top-ups

Prepaid wallets often set a minimum deposit. A low rate per GB can sit behind a large first payment.

Traffic expiry

Prepaid traffic may expire after a set period. Unused gigabytes then disappear.

Rollover

Subscription traffic usually resets at renewal. Some plans carry unused traffic forward and others do not.

Paid add-ons

City, ZIP or carrier targeting, extra sub-users and higher concurrency can carry a surcharge or need a higher tier.

Taxes and fees

List prices often exclude VAT or sales tax. Some payment methods add a processing fee.

None of these is a trick. Each is stated somewhere in the pricing page or the terms. The work is to read them before the first payment, not after the first invoice.

Getting value from a trial

Trials come as free traffic, a low-cost starter pack or a refund window. Whatever the form, a trial exists to produce the two inputs of the formula.

1. **Read the limits first.** Trial accounts may cap traffic, restrict targeting or ask for identity checks. Refunds often lapse above a usage threshold.
2. **Test the real target.** Fetching a test page proves the credentials work and nothing else.
3. **Define success in advance.** Pick an element only the real page contains and check for it in every response.
4. **Log four numbers.** Record requests sent, successful requests, gigabytes used and typical response time.
5. **Test at production shape.** Use real concurrency, real locations and the hours the job will run.
6. **Compute cost per 1,000 successes.** Use the plan price that would apply after the trial, add-ons included.

Run the same script against two or three providers and the decision usually makes itself. ProxyLook's compare page helps build that shortlist before any money moves.

KEY TAKEAWAYS

- Proxies are billed per GB, per IP, per port or thread, per request, or as flat plans with limits elsewhere.
- List prices are rough and change often, so check live pricing before budgeting.
- Cost per successful request is total spend divided by responses with the right content.
- Headless browsers can use many times the traffic of raw HTML requests for the same data.
- Expiry, rollover rules, minimum top-ups and paid add-ons belong in every price comparison.
- A trial is only useful when it measures real targets at production volume.

CHAPTER 07

Choosing a Provider

A 12-point scorecard, a seven-day pilot plan and a red-flag list for picking a proxy provider on evidence instead of homepage claims.

- A 12-point provider scorecard
- Ethical sourcing and KYC
- Reading pool-size claims with care
- A seven-day pilot plan

Open ten provider homepages and the claims blur together: tens of millions of IPs, nearly every country, high success rates. None of those lines says how a network behaves against one buyer's targets, in that buyer's locations, at that buyer's volume. A scorecard and a one-week pilot answer that question.

A 12-point scorecard

The scorecard turns a vague impression into a number. Score each row from 1 (poor) to 5 (excellent), then double the rows that matter most for the job. A scraping team might double coverage and pricing fit. A team managing logins might double session control and proxy types.

Criterion	What to check	A good sign
1. Pool and coverage	IPs in the countries and cities the job needs	Per-country figures, not one global total
2. Proxy types	Which of the four types are sold	The type needed now, plus the next rung up
3. Targeting	Country, region, city and carrier options	The needed level is included in the plan price
4. Session control	Sticky duration and rotation settings	Session length can be set per connection
5. Pricing fit	Billing model against the shape of the traffic	Low cost per successful request in a pilot
6. Trial and refund	How to test before committing	Trial or refund terms stated up front

Criterion	What to check	A good sign
7. Sourcing transparency	Where residential and mobile IPs come from	A public page on consent and opt-out
8. Documentation	Setup guides and code samples	Public docs, readable without an account
9. Support	Channels, hours and speed of reply	A technical answer within hours during the pilot
10. Dashboard and API	Usage stats, sub-users and automation	Usage broken down by day and by target
11. Compliance and KYC	Acceptable-use policy and customer checks	Clear rules and a real verification step
12. Reputation	Track record and independent feedback	A named company with years of operation

- The 12-point provider scorecard. Score each row 1 to 5 and weight by use case.

Rows 1 to 6 can be scored from a pricing page and a trial. Rows 7 to 12 take reading and a few emails. Skipping the second half is how teams end up with a cheap plan from a provider that disappears mid-project. The appendix has a compact worksheet to copy for each provider on the shortlist.

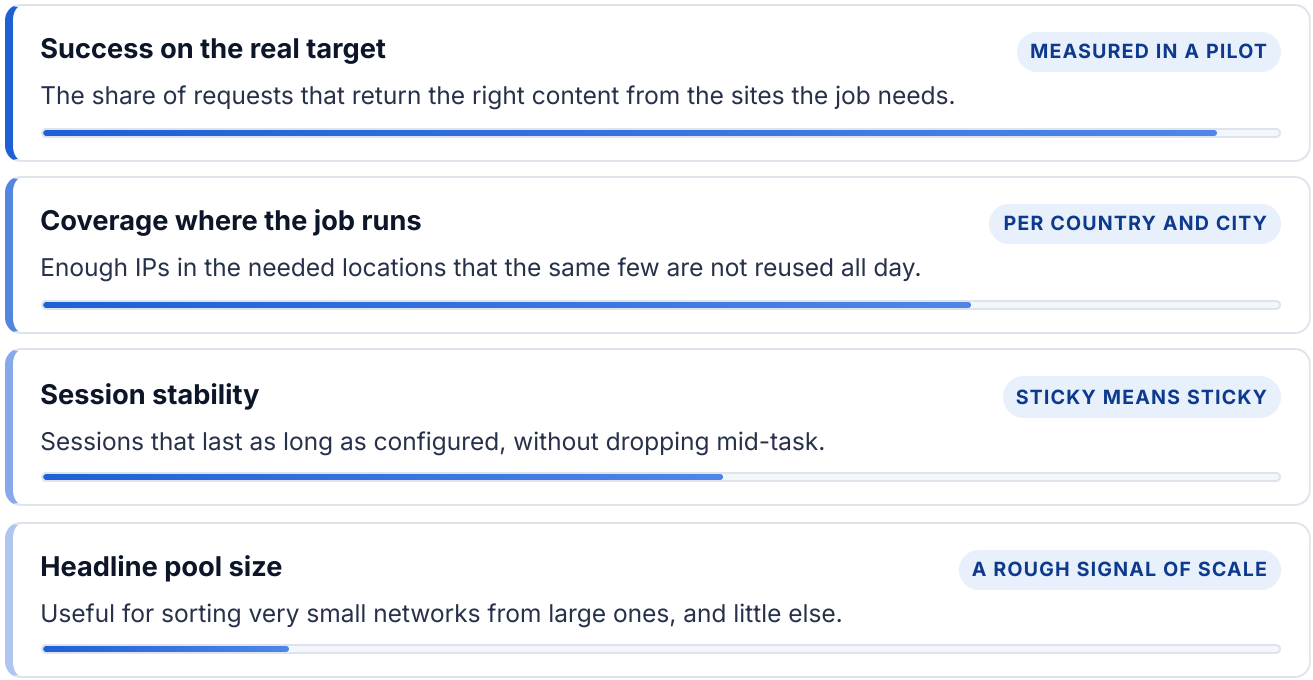
Pool size, handled with care

Pool size is the number buyers quote most and understand least. Every figure is vendor-stated, and no outside body audits it. Three problems follow.

First, vendors count differently. One counts every IP seen over a month. Another counts IPs online at a given moment. Residential devices come and go, so the two methods give very different totals for the same network.

Second, networks overlap. Resellers draw on the same upstream pools, and one household device can sit in several networks at once. Adding up the claims of ten vendors does not give ten separate pools.

Third, a global total says nothing about one place. A job that needs IPs in Lisbon is served by the IPs in Lisbon. A hundred million addresses elsewhere do not help.



What to weigh when comparing networks, most useful first. The meters show relative weight, not measurements.

How clean the IPs are matters more than how many exist. An address that other customers hammered against the same retail site yesterday arrives already flagged. No vendor publishes that figure, which is one more reason to run a pilot.

Ethical sourcing and KYC

Chapter 2 explained that residential and mobile pools are built from real people's devices, and why their consent matters. For a buyer comparing vendors the question becomes practical: how to score rows 7 and 11 from the outside.

Row 7 is scored on what a provider publishes. The useful evidence is a public page. It should say how devices join the network, what the owner sees before agreeing, and how to leave.

Scores 5

A public sourcing page, a visible consent screen, a stated opt-out and a named abuse contact.

Scores 3

General statements about "ethical" IPs, with real detail supplied only when a buyer asks.

Scores 1

Nothing published, and questions to sales get a vague reply or none.

Row 11 looks in the other direction, at how the provider screens its own customers. Customer identity checks, known as KYC, can mean a company email, an ID check or a short call. Some providers also block sensitive targets, such as banking or government sites, until a use case is approved.

The checks slow signup down, and buyers often resent them. They are still a good sign. A provider with no checks sells the same pool to anyone, and the abuse that follows lands on IPs every other customer shares. A provider that asks what the traffic is for is policing its own network.

> **TIP** **Ask in writing**
>
> Send the sourcing questions to sales or support by email before buying. A clear written answer is worth more than a badge on a homepage. A vague answer is information too.

Ten example networks

The table lists ten networks as examples drawn from the ProxyLook directory. It is in alphabetical order and it is not a ranking. Pool figures are vendor-stated and are reproduced as published.

Network	Proxy types	Vendor-stated pool
Bright Data	Residential, datacenter, ISP, mobile	150M+ IPs
Decodo (formerly Smartproxy)	Residential, datacenter, ISP, mobile	125M+ IPs
IPRoyal	Residential, datacenter, ISP, mobile	32M+ IPs
NetNut	Residential, datacenter, mobile	85M+ residential, 5M+ mobile
NodeMaven	Residential, ISP, mobile	30M+ residential, 250K+ mobile
Oxylabs	Residential, datacenter, ISP, mobile	177M+ IPs
ProxyEmpire	Residential, datacenter, ISP, mobile	30M+ residential, 4M+ mobile
Proxy-Seller	Residential, datacenter (IPv4 and IPv6), ISP, mobile	20M+ residential
SOAX	Residential, ISP, mobile	155M+ IPs
Webshare	Residential, datacenter	80M+ residential, 30M+ datacenter

- Ten example networks in alphabetical order. Not a ranking. Pool sizes are vendor-stated.

The pool column makes the earlier point by itself. Some vendors publish one total and others split residential from mobile, so the figures do not line up. The types column is more useful. Six of the ten list all four proxy types, which lets a team climb the ladder without changing vendor.

The full list of more than 200 providers is at proxylook.com/all-providers. ProxyLook publishes how it scores them on its methodology page, and the compare page puts candidates side by side.

A seven-day pilot

A pilot is a small, timed experiment with a pass mark agreed in advance. One working week is enough to separate two or three candidates.

1. **Day 1: write the brief.** List targets, locations, daily volume and session needs. Set a pass mark for success rate and for cost per 1,000 good pages.
2. **Day 2: shortlist and sign up.** Use the scorecard to cut the list to two or three. Open trials and complete any verification.
3. **Day 3: smoke test.** Confirm authentication, exit IP and exit location for every country the job needs.
4. **Day 4: run at low volume.** Send a few hundred requests per target with the same script for each provider. Log success, response time and traffic used.
5. **Day 5: run at production shape.** Use full concurrency at the hours the job will run. Watch for success falling as volume rises.
6. **Day 6: test sessions and support.** Hold sticky sessions for as long as the job requires. Send support one real technical question and time the reply.
7. **Day 7: score and decide.** Compute cost per 1,000 successful requests, fill in the scorecard and file the raw logs.

The raw logs matter later. When success drops six months on, they show what normal looked like.

Red flags

One of these signs alone is a reason to ask questions. Several together are a reason to walk away.

- [ ] No explanation of where residential or mobile IPs come from.
- [ ] A pool figure with no country breakdown and no definition of what is counted.
- [ ] No trial, no refund and a large minimum purchase.
- [ ] "Unlimited" plans with no published fair-use terms.

- [ ] Marketing that promises accounts or scrapers will never be detected or banned.
- [ ] No company name, address or terms of service on the site.
- [ ] Documentation hidden until after payment.
- [ ] Support that answers sales questions quickly and technical questions never.

KEY TAKEAWAYS

- Score providers on 12 criteria and weight the rows that matter for the job.
- Pool sizes are vendor-stated, counted in different ways, and overlap between networks.
- Success on the real target and coverage in the needed locations outweigh headline pool size.
- Consent-based sourcing and customer verification protect the buyer as well as device owners.
- A seven-day pilot with a pass mark agreed in advance beats any homepage claim.
- The ten networks named here are examples, not a ranking, and the full list is at /all-providers.

CHAPTER 08

Setup in Code

Working proxy setups for curl, Python, Node.js, Playwright, Puppeteer and Scrapy, plus a script that verifies the exit IP and a table of common errors.

- Credentials in environment variables
- Verifying the exit IP
- Python, Node.js and browser examples
- Common errors and their fixes

A proxy setup works when a request to `https://api.ipify.org` returns an address that is not the machine's own. Every example in this chapter ends with that check. Each one reads its credentials from the environment, sets a timeout and leaves TLS verification on.

Credentials and a first request

A proxy URL carries a username and a password. Pasted into a script, it ends up in version control, logs and screenshots. Environment variables keep the secret in one place, outside the code.

SHELL · ENVIRONMENT

```
# proxy.env: add this file name to .gitignore
export PROXY_HOST="gate.example-proxy.com:7000"
export PROXY_USER="USERNAME"
export PROXY_PASS="PASSWORD"
export PROXY_URL="http://$PROXY_USER:$PROXY_PASS@$PROXY_HOST"

# Load the variables into the current shell:
#   source proxy.env
```

HTTP clients take the full URL. Browsers want the host and the credentials as separate fields, so the file defines both forms. A password containing `@`, `:` or `/` must be percent-encoded before it goes into a URL.

Many providers read targeting options from the username, in a form such as `USERNAME-country-us-session-a1b2c3`. The exact syntax differs by vendor and comes from the provider's documentation.

Chapter 3 introduced curl as the quickest test of a proxy URL. The same commands read from the environment here.

SHELL · CURL

```
# HTTP proxy. Give up after 30 seconds in total.
curl -x "$PROXY_URL" --max-time 30 https://api.ipify.org

# SOCKS5, with DNS resolved by the proxy instead of locally.
curl --socks5-hostname "$PROXY_HOST" \
     --proxy-user "$PROXY_USER:$PROXY_PASS" \
     --max-time 30 https://api.ipify.org

# Print the proxy handshake when something fails.
curl -v -x "$PROXY_URL" --max-time 30 https://api.ipify.org
```

The SOCKS5 port is often different from the HTTP port, so check the dashboard first. The `-v` output includes the encoded credentials. Trim it before pasting it into a support ticket.

Python: requests and httpx

A production `requests` script differs from a demo in three ways: a reused Session, a timeout on every call, and retries with backoff.

PYTHON · REQUESTS

```python
import os

import requests
from requests.adapters import HTTPAdapter
from urllib3.util.retry import Retry

proxy_url = os.environ["PROXY_URL"]
proxies = {"http": proxy_url, "https": proxy_url}

retry = Retry(
    total=3,
    backoff_factor=1,
    status_forcelist=[429, 500, 502, 503, 504],
    allowed_methods=["GET", "HEAD"],
)
session = requests.Session()
session.mount("http://", HTTPAdapter(max_retries=retry))
session.mount("https://", HTTPAdapter(max_retries=retry))

resp = session.get(
    "https://api.ipify.org", proxies=proxies, timeout=(10, 30)
)
resp.raise_for_status()
print(resp.text)
```

The timeout tuple allows 10 seconds to connect and 30 seconds to read. The retry policy waits longer after each failure and honors a `Retry-After` header from the target. Chapter 11 adds jitter for larger jobs. Passing `proxies` on the call is deliberate. Proxies stored on the Session object can be overridden by an `HTTPS_PROXY` variable in the environment.

For concurrent work, `httpx` offers an async client. Current versions take a single `proxy` argument, and the older `proxies` argument was removed in 0.28. A semaphore caps how many requests run at once.

PYTHON · HTTPX

```python
import asyncio
import os

import httpx

PROXY_URL = os.environ["PROXY_URL"]
TIMEOUT = httpx.Timeout(30.0, connect=10.0)


async def exit_ip(client, limit):
    async with limit:
        resp = await client.get("https://api.ipify.org")
        resp.raise_for_status()
        return resp.text


async def main():
    limit = asyncio.Semaphore(3)
    async with httpx.AsyncClient(proxy=PROXY_URL, timeout=TIMEOUT) as c:
        ips = await asyncio.gather(*(exit_ip(c, limit) for _ in range(6)))
    print(ips)


asyncio.run(main())
```

The six results may show fewer than six addresses. As Chapter 4 explained, a rotating gateway assigns the exit IP when a connection opens, and the client reuses open connections. Here three connections serve six requests.

Node.js with undici

Node's built-in `fetch` does not take a proxy URL directly. The `undici` package supplies a `ProxyAgent` that is passed as the dispatcher. Importing `fetch` from the same package keeps the two versions matched.

JAVASCRIPT · UNDICI

```
// npm install undici, then save as check.mjs
import { fetch, ProxyAgent } from "undici";

const dispatcher = new ProxyAgent(process.env.PROXY_URL);

try {
  const res = await fetch("https://api.ipify.org", {
    dispatcher,
    signal: AbortSignal.timeout(30_000),
  });
  if (!res.ok) throw new Error(`Target answered ${res.status}`);
  console.log(await res.text());
} finally {
  await dispatcher.close();
}
```

The `.mjs` extension matters because the file uses top-level `await`. `AbortSignal.timeout` cancels the request after 30 seconds. Closing the dispatcher lets the process exit cleanly.

Browsers: Playwright and Puppeteer

Automated browsers split the proxy into a server address and a pair of credentials. Playwright accepts all three at launch.

PYTHON · PLAYWRIGHT

```
import os

from playwright.sync_api import sync_playwright

proxy = {
    "server": "http://" + os.environ["PROXY_HOST"],
    "username": os.environ["PROXY_USER"],
    "password": os.environ["PROXY_PASS"],
}

with sync_playwright() as p:
    browser = p.chromium.launch(proxy=proxy)
    page = browser.new_page()
    resp = page.goto("https://api.ipify.org", timeout=30_000)
    print(resp.text())
    browser.close()
```

Playwright takes the same proxy object on `browser.new_context()`, which gives each context its own exit. Puppeteer works differently. The server goes in a Chromium launch flag, and the credentials are supplied per page.

JAVASCRIPT · PUPPETEER

```
// npm install puppeteer, then save as check.mjs
import puppeteer from "puppeteer";

const browser = await puppeteer.launch({
  args: [`--proxy-server=${process.env.PROXY_HOST}`],
});
try {
  const page = await browser.newPage();
  await page.authenticate({
    username: process.env.PROXY_USER,
    password: process.env.PROXY_PASS,
  });
  const res = await page.goto("https://api.ipify.org", {
    timeout: 30_000,
  });
  console.log(await res.text());
} finally {
  await browser.close();
}
```

Chromium does not support SOCKS5 proxies that need a password, so browser automation should use the HTTP endpoint. A full page load also uses far more traffic than a plain HTTP call, as Chapter 6 showed.

WATCH OUT **Selenium with Chrome**

Chrome does not accept proxy credentials on the command line, so Selenium cannot pass them in the proxy flag. The dependable route is IP allowlisting: add the machine's public IP in the provider dashboard and connect without a password.

Scrapy

Scrapy reads the proxy from each request's `meta`. A small downloader middleware sets it in one place for the whole project.

PYTHON · SCRAPY

```python
# myproject/middlewares.py
import os


class ProxyMiddleware:
    def process_request(self, request, spider=None):
        request.meta["proxy"] = os.environ["PROXY_URL"]


# myproject/settings.py
DOWNLOADER_MIDDLEWARES = {
    "myproject.middlewares.ProxyMiddleware": 350,
}
DOWNLOAD_TIMEOUT = 30
RETRY_TIMES = 3
CONCURRENT_REQUESTS_PER_DOMAIN = 4
```

Priority 350 runs the middleware ahead of Scrapy's built-in `HttpProxyMiddleware`, which then moves the credentials into a `Proxy-Authorization` header. Replace `myproject` with the real project name. Start with low per-domain concurrency and raise it while watching the error rate.

Verifying the exit and fixing errors

A script can finish without errors and still send traffic from the machine's own IP. A misspelled key in the `proxies` dictionary is silently ignored, for example. The check below compares the direct address with the proxied one.

PYTHON · REQUESTS

```python
import os

import requests

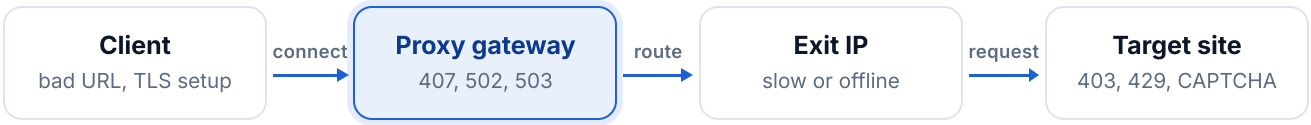
CHECK = "https://api.ipify.org"
proxy_url = os.environ["PROXY_URL"]
proxies = {"http": proxy_url, "https": proxy_url}

direct = requests.get(CHECK, timeout=10).text
exits = {
    requests.get(CHECK, proxies=proxies, timeout=30).text
    for _ in range(5)
}

print("direct :", direct)
print("exits  :", sorted(exits))
assert direct not in exits, "traffic is not using the proxy"
```

A rotating plan should print several exit addresses. A sticky or static plan should print one. If the assertion fails, the proxy settings are not being applied. ProxyLook's free proxy checker and IP lookup give a second opinion from the browser.

When a request fails, the first task is to find where. A proxied request passes four points, and each one fails in its own way.



Four places a proxied request can fail. The error usually says which one.

For HTTPS targets, a proxy-side failure arrives as a failed tunnel. curl reports that the CONNECT tunnel failed, and `requests` raises `ProxyError`. A status code inside a normal response came from the target.

Symptom	Likely cause	Fix
407 Proxy Authentication Required	Wrong credentials, unencoded password or IP not allowlisted	Recheck the username string and percent-encode the password
Connection timeout	Wrong host or port, a firewall, or a dead exit	Test with curl, open the port, retry on a new session
TLS or certificate error	An `https://` scheme on a plain HTTP proxy port	Use `http://` in the proxy URL; never disable verification

Symptom	Likely cause	Fix
502 or 503 from the gateway	No exit available for the requested location	Loosen the targeting, retry, check provider status
403 from the target	The site rejected the IP or the client	Fix headers, slow down, then change IP or proxy type
429 from the target	Too many requests from one IP	Lower concurrency, back off, honor `Retry-After`

Common proxy errors, where they come from and what to change.

One TLS case needs a note. Some unblocker products re-sign traffic and require the vendor's CA certificate to be installed. Install the certificate as the vendor documents. Turning verification off hides the error and removes the protection. Blocks from the target are the subject of Chapter 10, and ProxyLook's companion book, Scraping Without Blocks, goes further.

KEY TAKEAWAYS

- Keep proxy credentials in environment variables and out of version control.
- Set a timeout on every request and retry with backoff on 429 and 5xx responses.
- HTTP clients take a full proxy URL, while browsers take the server and credentials separately.
- Confirm the exit IP with a call to api.ipify.org before pointing any script at a real target.
- A failed tunnel points at the proxy, and a 403 or 429 in a normal response points at the target.

CHAPTER 09

Browsers, Antidetect Tools and Phones

Where to set a proxy on desktops, browsers, antidetect profiles and phones, and how to stop the rest of the device giving the real identity away.

- System-wide versus per-app proxy settings
- Android, iOS and cloud phones
- Per-profile proxies in antidetect browsers
- WebRTC, DNS and timezone leaks

A buyer types a proxy into the Windows settings page, opens Chrome and sees a German IP. A curl command in the terminal on the same machine still reports the office address. The proxy works. It was set in a place that only some programs read.

Where the proxy setting lives

A proxy can be set at four levels, and each one covers less of the machine than the one before. Setting it at the wrong level is a common reason traffic leaves through the wrong IP.

System-wide — THE WHOLE DEVICE, IN THEORY

Set once in the operating system. Only apps that ask the system for its proxy will use it.

Per app — ONE PROGRAM

Firefox, curl and most scraping tools carry their own proxy setting.

Per profile — ONE BROWSER IDENTITY

An antidetect browser binds a different proxy to each profile.

Per request — ONE CONNECTION

Code picks a proxy for each request, as Chapter 8 showed.

- The four places a proxy can be set, from widest scope to narrowest.

On Windows the system setting sits under Settings, Network & internet, Proxy. On macOS it sits under System Settings, Network, the active connection, Details, Proxies. Chrome, Edge and Safari read these panels. Many command-line tools do not. They look for the `HTTP_PROXY` and `HTTPS_PROXY` environment variables, or for a flag such as `curl -x`.

A system proxy is a preference, not a tunnel. Nothing forces an app to honor it. A device that must push every packet through another network needs a VPN.

KEY IDEA **Check from inside the app**

After setting a proxy at any level, load an IP-check page from the same program that will do the work. A correct answer in the browser says nothing about what a script or a desktop app is sending.

Firefox, Chrome and extensions

Firefox keeps its own proxy settings under Settings, General, Network Settings. The panel can follow the system, take a manual HTTP or SOCKS proxy, or load a PAC file. That makes Firefox the simplest way to run one proxied browser beside a normal one.

Two details matter. The "Proxy DNS when using SOCKS v5" box from Chapter 3 should stay ticked. The panel also has no fields for a SOCKS username and password, so SOCKS5 in Firefox works best with IP allowlisting. HTTP proxies prompt for credentials on first use.

Chrome has no proxy panel of its own. Its settings page opens the operating system's panel instead. A proxy for Chrome alone takes one of two routes:

- **A launch flag.** Starting Chrome with `--proxy-server` and a separate `--user-data-dir` gives a second, proxied instance.
- **An extension.** Proxy-switcher extensions use Chrome's proxy API to set a proxy for one Chrome profile. Pick a publisher with a track record, because the extension can see every URL.

SHELL · CHROME

```
# macOS: a second Chrome with its own profile and proxy
"/Applications/Google Chrome.app/Contents/MacOS/Google Chrome" \
  --user-data-dir="$HOME/chrome-proxy-de" \
  --proxy-server="gate.example-proxy.com:7000"

# Linux: the same idea
google-chrome --user-data-dir="$HOME/chrome-proxy-de" \
  --proxy-server="gate.example-proxy.com:7000"
```

The flag cannot carry a username and password. Chrome asks for them in a prompt, or the provider allowlists the machine's IP. Chrome also does not support SOCKS5 with a username and password, so use the provider's HTTP port or allowlisting.

Antidetect browsers: one profile, one proxy

An agency manages 12 client ad accounts from one laptop. In a normal browser those accounts share cookies, a fingerprint and an IP. In an antidetect browser each account gets a profile with its own storage, its own fingerprint and its own proxy.

The proxy field is only part of what these tools add:

- **Isolated storage.** Cookies, local storage and cache never cross between profiles.
- **A stable fingerprint.** Each profile reports the same canvas, WebGL, font, screen and user-agent values every time it opens.
- **Location that follows the IP.** Many tools can set timezone, language and geolocation from the proxy's exit address.
- **Team and automation features.** Profiles can be shared with colleagues or driven through an API.

Tool	Engine	Worth knowing
Multilogin	Chromium core "Mimic"	Also offers Android cloud phones. Has a free plan.

Tool	Engine	Worth knowing
AdsPower	SunBrowser (Chromium) and FlowerBrowser (Firefox)	Built-in RPA and a window synchronizer.
GoLogin	Chromium core "Orbita"	Free plan available.
Kameleo	"Chroma" (Chromium) and "Junglefox" (Firefox)	Supports Selenium, Puppeteer and Playwright.
Octo Browser	Chromium based	API access.
Camoufox	Firefox fork	Free, open source, built for Playwright automation.

- Six tools from the ProxyLook directory, with attributes as listed there.

The full antidetect browser list covers the rest. Pair each profile with one steady address: a static ISP proxy or a long sticky residential session. An account that logs in from a new city every ten minutes looks wrong, however good the fingerprint is. ProxyLook's companion book, The Multi-Account Handbook, covers account work in depth.

WATCH OUT **Multi-accounting and platform rules**

Running several accounts can breach a platform's terms of service, and the operator carries that risk, including bans and lost balances. An antidetect browser is built to stop accounts being linked through browser signals. It cannot guarantee that they never are.

Phones: Wi-Fi settings and cloud phones

Android and iOS both accept a proxy per Wi-Fi network in the system settings, and only an HTTP proxy.

Android

Open the Wi-Fi network's settings, choose Advanced options and set Proxy to Manual, then enter host and port. There are no credential fields. Android itself notes that the browser uses the proxy but other apps may not.

iOS

Open Settings, Wi-Fi, tap the info button beside the network, then Configure Proxy and Manual. Enter the server and port, and switch on Authentication to add a username and password. The setting does not cover cellular data.

Menu names vary by phone maker and system version. Two gaps remain on both systems: SOCKS5, and apps that ignore the Wi-Fi proxy. A proxy client app that registers as a local VPN closes both, because it captures all traffic on the device.

Cloud phones solve a different problem. TikTok, Instagram and similar platforms are built app-first, and an app can read far more device detail than a web page can. A cloud phone is an Android device hosted in a data center and controlled remotely. Each one has its own device identity and its own proxy setting.

- **GeeLark** offers Android cloud phones plus browser profiles, with metered running time.
- **DuoPlus** offers Android cloud phones only, opened from a browser tab. It charges a per-device fee plus metered running time, and comes from the team behind AdsPower.
- **Multilogin** offers Android cloud phones alongside its browser profiles.

Mobile or residential proxies suit cloud phones best. A phone app that arrives from a server-farm IP tells two different stories at once. Metered running time also means that a phone left open keeps costing money.

Leaks that give the real identity away

A proxied browser can still reveal the machine behind it. Three leaks are worth checking every time.

WebRTC

WebRTC powers video calls in the browser. To set up a call it asks a STUN server, over UDP, for the device's public address. An HTTP proxy does not carry UDP, so that request can leave directly, and a script on the page can read the real IP. Firefox can switch WebRTC off in `about:config`. Chrome needs an extension or a managed policy. Antidetect browsers usually offer a WebRTC setting per profile.

DNS

Chapter 3 covered the SOCKS5 lookup problem and the `socks5h://` fix for scripts. Browsers and antidetect profiles on SOCKS5 need the same care. A DNS leak test run inside the profile should show a resolver near the proxy, not the local internet provider.

Timezone and language

A page can read the browser's timezone and language list with two lines of JavaScript. A Frankfurt IP with a Chicago clock and English-only language headers does not add up. Set both to match the exit location. In Playwright that is `new_context(locale="de-DE", timezone_id="Europe/Berlin")`.

One rule sits behind all three fixes. Every signal a site can read should describe the same person in the same place.

Signal	How a site reads it	Should agree with
IP location	The connection itself	The account's usual country and city
Timezone	JavaScript date settings	IP location
Language	Accept-Language header and browser language list	IP country or account history
WebRTC address	STUN lookup run by the page	The proxy IP, or no address at all
DNS resolver	Lookups for test hostnames	The proxy's region
User agent	Request header and JavaScript	The real engine, system and screen size
Network type	ASN lookup on the IP	The device: a carrier IP for a phone app

- Identity signals that should agree with each other.

Checking the setup with ProxyLook's free tools

Run this check inside each new profile or device before logging in to anything.

1. **Confirm the exit IP.** Open What Is My IP. The IP, location, ISP and ASN should belong to the proxy, not the local connection.
2. **Test for a WebRTC leak.** Open the WebRTC Leak Test. Any public address other than the proxy's is a leak.
3. **Read the user agent.** Paste it into the User-Agent Parser. The browser, engine, system and device should match what the profile claims to be.
4. **Look up a new proxy before assigning it.** The Proxy Checker returns the country, ASN, ISP and network type for a proxy IP.
5. **Check the fit.** The Compatibility Checker suggests which proxy type fits a target site and use case.

Repeat the first two steps after every browser update or proxy change. A leak that appears in month three does the same damage as one on day one.

KEY TAKEAWAYS

- A system proxy only reaches apps that choose to read it, so confirm the exit IP from inside each program.
- Firefox has its own proxy panel, while Chrome needs the system setting, a launch flag or an extension.
- Antidetect browsers and cloud phones give each account separate storage, a separate fingerprint and a separate proxy.
- WebRTC, local DNS and a mismatched timezone or language can expose the real location behind a working proxy.
- Multi-accounting can breach platform terms, and no tool removes that risk for the operator.

CHAPTER 10

Staying Unblocked

How websites decide a visitor is automated, how a block really shows up, and a cheapest-first order for fixing it.

- The seven layers a website checks
- Symptom, cause and fix table
- Hard blocks and soft blocks
- An escalation ladder by cost

A price scraper runs clean for three weeks. On a Monday every request still returns 200, and every price field in the output is empty. Nothing failed loudly. The site stopped trusting the visitor and began serving a page without the data.

What a website checks

Bot detection is a stack of separate checks, and a request has to pass every one a site runs. Some happen before the server reads a single header. Others build up over days of account activity.

1 · IP reputation	Network type, past abuse, and how many clients share the range.
2 · Request rate and pattern	Requests per IP and per session, URL order, clockwork timing.
3 · TLS and HTTP/2 fingerprint	The handshake reveals the client library behind the request.
4 · Headers	User-Agent, Accept-Language, header order, anything a browser would send.
5 · Browser and JavaScript fingerprint	Canvas, WebGL, fonts, and automation flags such as navigator.webdriver.
6 · Behavior	Mouse movement, scrolling, time on page, the path through the site.
7 · Account-level signals	Account age, login history, linked devices, activity that repeats across accounts.

- The layers a website can check, from the network up to the account.

A proxy changes layer 1 and helps with layer 2 by spreading requests across addresses. It does nothing for the other five. That explains a familiar complaint: premium residential IPs that still get blocked on the first request.

> **A proxy changes the IP address and nothing else about the request.**

Layer 3 catches many scripts. Python's `requests` library opens a TLS connection that looks nothing like Chrome's. If the User-Agent header then claims to be Chrome, the two layers disagree, and that mismatch is easy to flag. Detection vendors hash the handshake into fingerprints with names such as JA3 and JA4. HTTP/2 settings and header order give a second fingerprint of the same kind.

Layers 5 and 6 only apply when the site runs JavaScript checks, which a plain HTTP client never executes. On those sites the missing script result is itself the signal. Layer 7 matters for anything logged in, and it is where the profile hygiene from Chapter 9 pays off.

How a block actually shows up

Blocks rarely announce themselves. A scraper that only counts 403s will miss most of the forms below.

403 Forbidden

A flat refusal. On the first request it usually points at the IP range or the client fingerprint.

429 Too Many Requests

A rate limit. It often carries a Retry-After header that says how long to wait.

CAPTCHA or challenge page

A puzzle or a "checking the browser" screen. It can arrive with a 200, 403 or 503 status.

Redirect

A 302 to a login page, a consent wall, the home page or another country's store.

Soft block

A 200 with wrong or missing data: empty result lists, blank prices, stale pages, fewer items than a real visitor sees.

Silence

Timeouts and reset connections. The server drops the request without an answer.

Soft blocks do the most damage because the pipeline keeps running and bad data flows into reports. Chapter 6 made the point for billing: a 200 is not a success. The same check belongs in the code. Every response should be tested for something a good page always contains, such as a product title or a price element.

PYTHON · REQUESTS

```python
import requests

PROXY = "http://USERNAME:PASSWORD@gate.example-proxy.com:7000"
MARKERS = ("captcha", "unusual traffic", "access denied")

def classify(r):
    body = r.text.lower()
    if r.status_code in (403, 429):
        return "blocked"
    if r.history and "login" in r.url:
        return "redirected"
    if any(m in body for m in MARKERS):
        return "challenge"
    if 'class="price"' not in body:
        return "soft-block"
    return "ok"

r = requests.get("https://example.com/item/42",
                 proxies={"http": PROXY, "https": PROXY}, timeout=30)
print(r.status_code, len(r.content), classify(r))
```

The marker strings and the price selector are placeholders. Each target needs its own, taken from a real good page and a real blocked page. Log the label with every request so that success rate means valid data.

Symptom, likely cause, first fix

Most blocks fall into a small set of patterns. Chapter 8 covered errors from the proxy side. This table covers what the target sends back.

Symptom	Likely cause	First fix
403 on the very first request	Flagged IP range or TLS mismatch	Use a browser-like client, then another IP type
403 after a few dozen pages	Rate or pattern rule tripped	Lower concurrency and rotate per session
429	Too many requests per IP or session	Back off, honor Retry-After, widen the pool
CAPTCHA on every page	Low IP trust or automation flags	Residential or mobile IP, real browser

Symptom	Likely cause	First fix
Redirect to login	Lost cookies, or content needs an account	Keep cookies on a sticky session
200 with empty or odd data	Soft block or wrong exit country	Validate content and check geo-targeting
Timeouts and resets	Dead exit node or silent drop	Retry on a new IP with a timeout
407	Proxy credentials or allowlist	Fix at the provider; the site never saw it

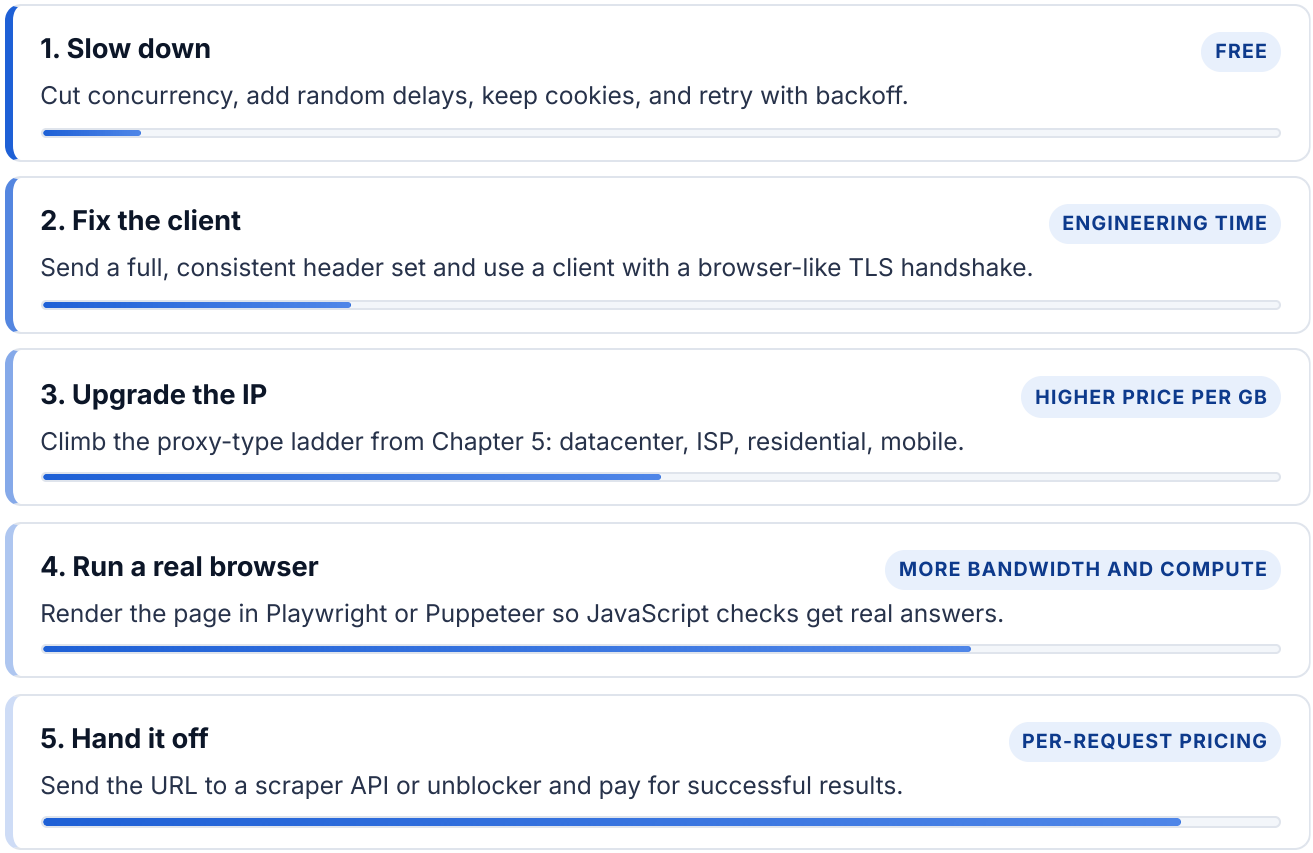
Common block symptoms with the cause to suspect first.

When the cause is unclear, separate the IP from the client. Load the target in a normal browser through the same proxy, then compare. The Proxy Checker also shows whether an address is registered as datacenter, residential or mobile.

> **EXAMPLE** **A two-minute diagnosis**
>
> A script gets a 403 on its first request through a residential IP. A normal browser on the same proxy loads the page. The IP is fine and the client is the problem, so a pricier pool would change nothing.

The escalation ladder

Chapter 5 gave a ladder of proxy types. This one is wider, because the IP is only one of the things that can be changed. Fixes have very different prices. Slowing down is free. Residential traffic runs roughly $1 to $5 per GB at entry level. Datacenter traffic runs roughly $0.10 to $1.40. Live figures are on the pricing page. A headless browser multiplies both bandwidth and compute. Climb one rung at a time, rerun the same URL sample, and stop at the first rung that works.



An escalation ladder. The meter shows relative cost per page, not a measured figure.

For rung 2, open-source clients such as `curl_cffi` copy a real browser's TLS and HTTP/2 handshake. They stay as light as a plain HTTP library. On sites that run no JavaScript checks, that one change can remove the need for rungs 3 and 4.

The order of rungs 3 and 4 can swap. A target that needs JavaScript to show any content needs a browser whatever the IP. For the IP rung, the web scraping page in the directory lists providers for that job.

When a scraper API or unblocker makes sense

A scraper API or unblocker takes a URL and returns the page. The vendor picks the IP, sets the fingerprint, renders JavaScript where needed and retries on failure. These products are priced per successful request or per monthly plan. Where billing is per success, failed attempts are the vendor's cost.

If...	Then...
The target stays blocked after rungs 1 to 4	Buy the unblocker and stop spending engineering time

If...	Then...
The team is small and the data is the product	A scraper API removes proxy and browser maintenance
The target is easy and volume is high	Raw datacenter proxies will usually cost far less
The job needs a logged-in session or multi-step flows	Keep control with proxies and a real browser

Choosing between raw proxies and a managed unblocker.

Many teams end up with both: raw proxies for the easy majority of targets and an unblocker for the stubborn few. The proxy API providers page lists vendors that sell this kind of product. ProxyLook's companion book, Scraping Without Blocks, goes deeper on each detection layer.

No setup is permanent

Detection systems change without notice. A configuration that returned clean data last month can start failing this week, and no proxy type, browser or vendor is immune to that. Mobile IPs get challenged. Unblockers return errors on hard targets.

> **WATCH OUT** **Treat blocks as a signal**
>
> A block is also the site saying it does not want this traffic. Before climbing the ladder, check that the job is one the operator is entitled to run. Chapter 12 covers the legal and ethical questions.

The practical answer is monitoring. Track the share of valid pages per target every day and alert on a drop. Keep the fixed URL sample ready for a quick diagnosis. The next chapter turns those numbers into a cost model.

KEY TAKEAWAYS

- Websites can check seven layers, and a proxy only changes the first one.
- A TLS handshake that disagrees with the User-Agent can get a script blocked on its first request.
- Soft blocks return a 200 with wrong data, so validate page content on every response.
- Escalate from free fixes to expensive ones, one rung at a time, against a fixed URL sample.
- A scraper API or unblocker suits hard targets and small teams, while raw proxies stay cheaper for easy ones.
- No proxy, browser or vendor makes a scraper immune to blocks, so monitor valid-data rate daily.

CHAPTER 11

Scaling and Cost Control

How to grow a proxy workload without growing the bill at the same rate: polite concurrency, smart retries, caching, tiered routing and four metrics to watch.

- Concurrency limits and politeness
- Caching and blocking heavy resources
- Retries with backoff and jitter
- Tiered routing, metrics and budgets

A scraper that retries instantly, downloads unchanged pages again and loads every product photo can burn a month of residential traffic in a week. None of those three habits improves the data. The biggest savings usually come from engineering, not from a cheaper plan.

Concurrency versus politeness

Concurrency is the number of requests in flight at once. Politeness is the load placed on one website. The two get confused because a rotating pool hides the problem. Take 500 parallel requests through 500 different IPs. Each address looks gentle, yet the target's servers still answer all 500.

Set limits per domain, not per job. A crawl across 2,000 small shops can run hundreds of connections in total while holding each shop to two or three. A crawl of one large marketplace can go higher, but it should rise in steps.

1. **Start low.** Begin with two to five concurrent requests per domain and a short random delay between them.
2. **Raise in steps.** Double the limit, run for several minutes, and compare success rate and latency with the previous step.
3. **Stop at the first warning.** Rising response times, 429 responses or new CAPTCHA pages mean the last step was too far. Drop back one level.

4. **Write the limit down.** Store the working number per domain in configuration so the next run starts from it.

Scrapy has this built in: `CONCURRENT_REQUESTS_PER_DOMAIN`, `DOWNLOAD_DELAY` and the AutoThrottle extension, which adjusts delay from observed latency. Provider plans add their own ceiling. Some bill per port or thread, so the concurrency limit is part of the price.

Retries with backoff and jitter

A failed request is worth retrying, but not immediately and not forever. An instant retry hits the same rate limit again and pays for the bandwidth twice. Exponential backoff doubles the wait after each failure. Jitter makes each wait random, so a hundred workers that failed together do not all return in the same second.

PYTHON · REQUESTS

```python
import random
import time
import requests

PROXY = "http://USERNAME:PASSWORD@gate.example-proxy.com:7000"
PROXIES = {"http": PROXY, "https": PROXY}
RETRY_ON = {408, 429, 500, 502, 503, 504}

def fetch(url, attempts=5, base=1.0, cap=30.0):
    for attempt in range(attempts):
        if attempt:
            ceiling = min(cap, base * 2 ** attempt)
            time.sleep(random.uniform(0, ceiling))
        try:
            r = requests.get(url, proxies=PROXIES, timeout=30)
        except requests.RequestException:
            continue
        if r.status_code not in RETRY_ON:
            return r
    raise RuntimeError(f"gave up on {url}")

print(fetch("https://example.com").status_code)
```

The function waits up to 2 seconds before the second attempt, then up to 4, 8 and 16, and gives up after five tries. Three rules keep retries cheap:

- **Retry only what can recover.** Timeouts, 429 and 5xx responses qualify. A 404 will still be a 404, and a 407 means the proxy rejected the credentials.
- **Honor Retry-After.** When a 429 or 503 carries that header, wait at least that long. The example leaves this out to stay short.
- **Cap the total.** Track retries as a share of all requests. A rising share usually means a block in progress, and backoff alone will not fix it.

A 403 is a special case. Repeating it from the same IP rarely helps. With a rotating gateway each retry leaves from a new address, so one or two more attempts are reasonable before escalating.

Not paying for the same bytes twice

The cheapest request is the one never sent. Chapter 6 showed how page weight drives a per-GB bill and named the habits that cut it. This section covers the mechanics.

Deduplicate and cache

Normalize URLs before queueing them, so that `?ref=home` and `?ref=search` versions of one product count once. Store every fetched page on disk with a timestamp, and give each page type a lifetime. A price may need a daily refresh. A product description can wait a month. During development, work from cached pages instead of the live site. Scrapy's `HTTPCACHE_ENABLED` setting does this in one line.

Ask whether the page changed

Many servers return an `ETag` or `Last-Modified` header. Sending it back as `If-None-Match` or `If-Modified-Since` lets the server answer 304 with no body when nothing changed. Sitemaps with modification dates serve the same purpose for whole sections of a site.

Block heavy resources in headless browsers

A rendered page pulls in images, fonts and video that the scraper never reads. In Playwright, route interception aborts those requests before they use any proxy bandwidth.

PYTHON · PLAYWRIGHT

```python
from playwright.sync_api import sync_playwright

BLOCKED = {"image", "media", "font"}

def handle(route):
    if route.request.resource_type in BLOCKED:
        route.abort()
    else:
        route.continue_()

with sync_playwright() as p:
    browser = p.chromium.launch(proxy={
        "server": "http://gate.example-proxy.com:7000",
        "username": "USERNAME",
        "password": "PASSWORD",
    })
    page = browser.new_page()
    page.route("**/*", handle)
    page.goto("https://example.com", timeout=30000)
    print(page.title())
    browser.close()
```

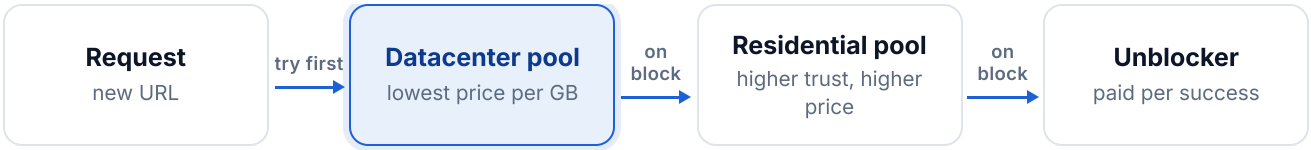
Check the output after adding the filter, because some pages break when resources are missing. A browser that fetches no images can also look unusual to a strict target. Leave scripts and XHR calls alone, since they usually carry the data.

> **TIP** **Cache the failures too**
>
> Record every 404 and permanent redirect alongside the good pages. A crawler that forgets them asks for the same dead URLs on every run and pays for each answer.

Tiered routing: cheap first, expensive on failure

Datacenter traffic lists at roughly $0.10 to $1.40 per GB and rotating residential at roughly $1 to $5 per GB at entry level. Live figures are on the pricing page. Sending every request through the expensive pool pays the premium even on pages that a datacenter IP would have fetched without trouble.

Chapter 5 applied "start cheap, escalate on evidence" to a whole job. Tiered routing applies it to each request. The router tries the cheap pool first and moves up only when a request fails for a block-related reason.



Tiered routing. Each request starts on the cheapest pool and escalates only after a block.

Four rules make the router work in practice:

- **Escalate on block signals only.** A 403, a 429 after backoff, a challenge page or a failed content check moves the request up. A 404 does not.
- **Remember what worked.** Keep a per-domain record of the lowest tier that succeeds, and start there next time.
- **Probe downward.** Send a small share of traffic to the cheaper tier now and then. Targets relax as well as tighten.
- **Limit the climb.** A block moves a request up one tier, once. A URL that fails on every tier goes to a review queue.

The directory pages for datacenter proxies and residential proxies list networks for the first two tiers. Buying both from one provider is convenient. Buying from two gives a fallback when one has an outage.

The four metrics worth tracking

Dashboards fill up fast. Four numbers, recorded per target domain and per proxy tier, answer most cost questions.

Metric	How to calculate it	What a bad reading means
Success rate	Valid pages divided by requests sent	Blocks, including soft blocks
Cost per 1,000 successful requests	Proxy spend divided by valid pages, times 1,000	Waste from retries, heavy pages or the wrong tier
p95 latency	Time within which 95% of requests finish	Slow exit nodes or quiet throttling
GB per day	Bytes through the proxy, from logs or the dashboard	A runaway job or a plan about to run out

Four metrics that cover reliability, cost, speed and consumption.

Chapter 6 introduced cost per successful request as the way to compare plans. In production it becomes a daily reading, because it folds price, page weight and failure rate into one figure. When it rises, one of the other three metrics will show why.

All four come from one log line per request. It needs the timestamp, target domain, proxy tier, status code, bytes received, elapsed time and content-check result. Scripts that log only failures cannot calculate any of them.

The proxy cost calculator estimates monthly spend from bandwidth and price per GB before a job exists to measure.

Budgets and alerts

Usage-based billing has no natural ceiling. A loop that requests the same URL all night is billed like real work. A few guardrails turn a surprise invoice into a short alert.

- [ ] Give every project its own sub-user or API key so traffic can be traced to its source.
- [ ] Set a daily GB cap per job, and stop the job when it is reached.
- [ ] Alert when GB per day climbs well above the trailing weekly average.
- [ ] Alert when success rate falls below the normal range for a target.
- [ ] Pause a job automatically when success rate stays low for several minutes.
- [ ] Review cost per 1,000 successful requests each month, per target.

The automatic pause matters most. Failed requests still use bandwidth, so a blocked scraper left running pays full price for nothing. Many provider dashboards offer per-user traffic limits. Check for them during the trial, along with what happens when a prepaid balance runs out mid-job.

KEY TAKEAWAYS

- Limit concurrency per domain, raise it in steps, and drop back at the first sign of throttling.
- Retry only recoverable failures, with exponential backoff, jitter and a hard cap on attempts.
- Deduplicate URLs, cache responses and block images, fonts and media to cut bandwidth.
- Route each request through the cheapest pool first and escalate only on a block.
- Cost per 1,000 successful requests is the fairest way to compare proxy setups.
- Daily caps, alerts and an automatic pause stop a broken job from draining the budget.

CHAPTER 12

Security, Law and Ethics

How to keep proxy credentials and traffic safe, which legal questions to ask before collecting data, and a 30-day plan that puts the whole book to work.

- Why free public proxies are risky
- Legal questions, robots.txt and etiquette
- Credential hygiene and TLS verification
- A 30-day action plan

A free proxy list promises thousands of working IPs at no cost. Someone pays for those servers, and the list does not say who or why. Every request sent through one passes across a stranger's machine first.

Why free public proxies are risky

A proxy sits in the middle of the connection by design. Chapter 3 set out what that position lets an operator see. With a known company, a contract and a reputation at stake, the exposure is acceptable. With an anonymous open proxy, it is the whole problem.

It can read plain HTTP

On an unencrypted site the operator sees every URL, cookie and form field, and can change the page on its way back.

It sees where traffic goes

HTTPS hides the content but not the hostname. The operator still learns which sites are visited and when.

Its owner may not know

Some open proxies are misconfigured servers that were never meant to be shared. Using them means using someone's machine without permission.

It is already burned

A public list is public to websites too. The addresses are shared by everyone and blocked widely.

A free trial from a paid provider is a different thing: a named company, published terms and a support desk. The ProxyLook directory lists more than 200 proxy providers, and entry-level datacenter traffic costs little enough that a free list saves almost nothing.

Credentials and TLS

Proxy credentials are spending power. Anyone who holds them can run traffic on the account's bill, and whatever they do traces back to the account holder. Chapter 8 moved the proxy URL into an environment variable. The next place it tends to leak is the logs, where a full URL prints the password in clear text.

PYTHON · STANDARD LIBRARY

```python
import os
from urllib.parse import urlsplit

def masked(proxy_url):
    """Return the proxy URL with the password hidden, for logs."""
    password = urlsplit(proxy_url).password
    if not password:
        return proxy_url
    return proxy_url.replace(f":{password}@", ":***@", 1)

proxy = os.environ["PROXY_URL"]
print("using proxy", masked(proxy))
```

Masking is one habit among several.

- [ ] Keep credentials in environment variables or a secret manager, never in source files.
- [ ] Add `.env` files to `.gitignore` and turn on secret scanning in the repository.
- [ ] Create one sub-user per project or person, so that one leak exposes one budget.
- [ ] Rotate passwords on a schedule, and at once when someone leaves or a password shows up in a log.
- [ ] Use IP allowlisting for servers with fixed addresses, and remove old entries.
- [ ] Mask the password before logging a proxy URL or pasting an error into a ticket.

Allowlisting has a second benefit. A username and password sent to an `http://` gateway travel unencrypted between the client and the gateway. On a shared network anyone on the path can read them. An allowlisted IP sends no password at all.

Always verify TLS

A certificate error through a proxy is a warning, and the tempting fix is to switch verification off. Doing so removes the one protection that stops a proxy operator reading passwords and session cookies. With verification on, a proxy can carry HTTPS traffic but cannot open it.

Chapter 8 noted the one exception: unblocker products that re-sign traffic and need the vendor's certificate installed. That is a deliberate trade, and it means the vendor can read what passes through. Trust the certificate in that one client only, and never send logins or payment details through such a product.

The legal questions to ask first

> **WATCH OUT** **Not legal advice**
>
> Nothing in this book is legal advice. Laws differ by country, courts have reached different results on similar facts, and the details of a project matter. This section sets out the questions a team should be able to answer before taking the rest to a lawyer.

Proxies are legal tools in most places. The legal questions attach to what is collected and how, and they are not one question. Four kinds of data call for very different levels of care.

Content behind a login — MOST CARE

Access depends on an account and the terms accepted to create it.

Personal data on public pages — HIGH CARE

Names, profiles and posts tied to people. Privacy laws apply even when the page is public.

Creative content — MODERATE CARE

Articles, photos and videos. Analyzing them differs from copying and republishing them.

Public facts — LEAST CARE

Prices, stock levels and specifications that any visitor can see without an account.

Kinds of data ranked by the care they call for. The meter is a guide, not a legal rating.

The table turns those categories into questions to answer in writing before a project starts.

Question	Why it matters	Safer habit
Is the page public or behind a login?	Logged-in access rests on an account and its terms	Stay on pages any visitor can load
Does the data identify people?	Privacy laws such as GDPR cover public personal data	Skip it, or collect the minimum for a stated purpose
What do the terms of service say?	A breach can bring bans or a contract claim	Read them and record how they were accepted
Is the content creative work?	Copyright protects text, images and video	Extract facts; do not republish the work
Where are the people and the operator?	Privacy and database laws apply by location	List the jurisdictions before collecting

Five legal questions for any data collection project.

A few points deserve more than a table cell. The GDPR covers information about identifiable people in the EU, and it can reach operators based elsewhere. California's CCPA and a growing list of national laws set their own rules, and they do not all treat public data the same way. Under the GDPR, storing personal data creates duties: a lawful reason, limited retention, and a way to answer deletion requests.

Terms of service matter most where they were clearly accepted, as with an account signup. Multi-accounting usually falls here. It can breach a platform's terms, and the operator carries the risk of bans and claims.

Copyright generally protects expression, not facts. A price is a fact. The review text next to it is someone's writing. Some jurisdictions also protect databases as such. The rules for training AI models on collected content are still being argued in courts and legislatures.

Robots.txt and rate etiquette

The file at `/robots.txt` states which paths a site asks crawlers to stay out of. The format is an internet standard, RFC 9309. It is a request, not a lock, and its legal weight varies. Ignoring it still looks bad in any later dispute, and following it costs little. Python's standard library reads it with `urllib.robotparser`.

Etiquette goes beyond that file:

- **Keep the load small.** A crawl should never slow the site for real visitors. Small sites need gentler limits than large ones.
- **Back off when asked.** A 429 or 503 is a request to slow down, not a puzzle to route around.
- **Prefer the official route.** Where a site offers an API, a data feed or a sitemap, use it.
- **Take only what the project needs.** Fewer pages and fewer fields mean less cost, less risk and less to secure.

> **KEY IDEA** **A simple test**
>
> Would the project be comfortable if the target site's engineers read its logs? A crawl that is slow, narrow and limited to public facts usually passes. One that hides behind rotating IPs to pull personal profiles does not.

Residential IPs: sourcing and responsible use

Chapters 2 and 7 covered how residential and mobile pools are built and which sourcing questions to ask. The short version: buy only from networks whose device owners opted in, are compensated and can leave. The other half of the responsibility sits with the buyer. Every request leaves through a real person's home connection or phone, and the target sees that person's address.

- [ ] Send nothing through a residential IP that would be a problem if traced to its owner.

- [ ] Keep volume per IP modest. The bandwidth belongs to someone, and on mobile it may be metered.
- [ ] Stay inside the provider's acceptable use policy, including its list of restricted targets.
- [ ] Use datacenter IPs where they work, and keep residential IPs for the jobs that need them.
- [ ] Keep the provider's written sourcing answers on file and ask again each year.

The residential proxies page in the directory is a starting point for a shortlist. The sourcing questions still have to go to each vendor directly.

A 30-day action plan

The plan below turns the book into four weeks of work for one team and one project.

1. **Week 1: define and pilot.** Write down targets, countries, pages per day and the fields needed. Answer the five legal questions. Shortlist two or three providers with the compare page or the ProxyLook assistant, then run the seven-day pilot from Chapter 7.
2. **Week 2: build and verify.** Move credentials into environment variables with one sub-user per project. Confirm the exit IP with What Is My IP and run the WebRTC Leak Test in every browser profile. Add a content check to every response.
3. **Week 3: harden and cut waste.** Add per-domain concurrency limits, backoff with jitter, caching and resource blocking. Start recording success rate, cost per 1,000 successful requests, p95 latency and GB per day.
4. **Week 4: govern.** Set daily caps and alerts. Rotate passwords and clean the allowlist. Write one page on what is collected, why, and how long it is kept. File the provider's sourcing answers, and set a monthly review date.

The appendix holds a scorecard worksheet for week 1 and a pre-launch checklist for week 2.

KEY TAKEAWAYS

- A free public proxy puts an unknown operator in the middle of every request.
- Proxy credentials belong in environment variables or a secret manager, with one sub-user per project.
- TLS verification stays on, because it is what stops a proxy reading encrypted traffic.
- Public facts, personal data, logged-in content and creative work raise different legal questions, and none of this is legal advice.
- Robots.txt, gentle request rates and careful use of residential IPs are part of running proxies responsibly.
- Thirty days is enough to define a project, verify the setup, cut waste and put controls in place.

APPENDIX

Appendix

Reference pages to keep open while working: a glossary, a provider scorecard, a pre-launch checklist, HTTP status codes and ProxyLook's free tools.

- Glossary of 24 proxy terms
- Provider scorecard worksheet
- Pre-launch checklist
- HTTP status codes behind a proxy

These pages are for use, not for reading through. Print the scorecard and the checklist, and keep the status table near the terminal.

Glossary

Term	Meaning
Allowlisting	Authorizing a client by its IP address instead of a password.
Antidetect browser	Browser that gives each profile its own fingerprint, storage and proxy.
ASN	Number that identifies the network owning an IP range.
Backconnect gateway	One hostname that forwards each request to a changing exit IP.
Backoff	Waiting longer after each failed attempt before retrying.
Cloud phone	Remote Android device with its own identity and proxy setting.
Concurrency	Number of requests in flight at the same time.
Datacenter proxy	IP hosted in a data center: fast, cheap, easier to flag.
Exit node	The device or server whose IP the target sees.
Fingerprint	Set of browser or client traits that can identify a visitor.
Geo-targeting	Choosing the country, region or city of the exit IP.
ISP proxy	Static IP on a server, registered to a consumer internet provider.

Term	Meaning
Jitter	Random variation in delays so clients do not retry together.
Mobile proxy	IP assigned by a mobile carrier to a device.
p95 latency	Time within which 95% of requests finish.
Residential proxy	IP assigned by a home internet provider to a household.
Rotation	Changing the exit IP per request or on a timer.
Scraper API	Service that fetches a URL and returns the page or data.
SOCKS5	Proxy protocol that carries any TCP traffic, and optionally UDP.
Soft block	A 200 response that carries wrong or missing data.
Sticky session	Keeping the same exit IP for a set time.
TLS fingerprint	Signature of a client's TLS handshake, such as JA3.
Unblocker	Managed proxy that handles fingerprints, retries and challenges itself.
WebRTC leak	Browser revealing the real IP through a WebRTC lookup.

- Twenty-four terms used in this book.

Provider scorecard

These are the 12 criteria from Chapter 7, scored 1 to 5. Copy the worksheet once per provider and fill it in during the pilot.

Criterion	What to check	Score (1 to 5)
1. Pool and coverage	IPs in the countries and cities the job needs	
2. Proxy types	The type needed now, plus the next rung up	
3. Targeting	Country, region, city and carrier options in the plan	
4. Session control	Sticky duration and rotation settings per connection	
5. Pricing fit	Cost per 1,000 successful requests in the pilot	
6. Trial and refund	Trial or refund terms stated before payment	
7. Sourcing transparency	Written answer on consent and opt-out for device owners	
8. Documentation	Public setup guides and working code samples	

Criterion	What to check	Score (1 to 5)
9. Support	Speed and quality of the reply to a technical question	
10. Dashboard and API	Usage by day and target, sub-users, traffic limits	
11. Compliance and KYC	Acceptable use policy and a real customer check	
12. Reputation	A named company with a track record	

Provider scorecard worksheet. The maximum unweighted score is 60.

TIP **Weight what matters**

Before adding up, double the three or four rows that matter most for the job. A team that needs ten static IPs should not let coverage in 195 countries decide the result.

Pre-launch checklist

Run through this list before a new scraper, profile or automation goes live.

- [ ] Exit IP, country and ASN confirmed from inside the app that does the work.
- [ ] No WebRTC or DNS leak in any browser profile.
- [ ] Timezone and language match the exit location.
- [ ] Credentials come from environment variables, and nothing secret is in the repository.
- [ ] Every request has a timeout, and TLS verification is on.
- [ ] Retries use backoff with jitter and a cap on attempts.
- [ ] Every response is checked for expected content, not only for status.
- [ ] Concurrency is limited per domain.
- [ ] A daily GB cap and a success-rate alert are in place.
- [ ] The legal questions are answered in writing, and robots.txt has been read.

HTTP status quick reference

A 407 always comes from the proxy, never from the target. A 502 or 503 can come from either. For HTTPS targets a proxy-side error shows up as a failed tunnel, while the target's own error arrives inside a normal response. Providers also add their own error codes, which their documentation explains.

Code	Name	Behind a proxy it usually means	What to do
200	OK	Success, or a soft block with wrong data	Validate the content
301 / 302	Redirect	A normal move, or a push to login or CAPTCHA	Check the final URL
403	Forbidden	Target refused the IP or the client fingerprint	Change the client, then the IP type
404	Not Found	Page is gone; now and then a disguised block	Do not retry; spot-check in a browser
407	Proxy Authentication Required	Wrong credentials, or the client IP is not allowlisted	Fix credentials or the allowlist
408	Request Timeout	Slow exit node or slow target	Retry with backoff
429	Too Many Requests	Rate limit per IP or per session	Slow down; honor Retry-After
500	Internal Server Error	Target failed while handling the request	Retry a few times
502	Bad Gateway	Proxy or exit node could not reach the target	Retry on a new IP
503	Service Unavailable	Target overloaded, a challenge page, or no exit available	Back off and inspect the body

- Common status codes and what they usually mean behind a proxy.

Troubleshooting order

When requests fail, work from the bottom of the connection upward. Each layer is quicker and cheaper to check than the one after it.

1 · Credentials	A 407 or refused connection: check username, password, port and allowlist.
2 · Connectivity	Timeouts: test the gateway alone with curl and a 30-second limit.
3 · Exit IP	Confirm the country, ASN and network type match the order.
4 · Rate	429s or blocks after some pages: lower concurrency and add backoff.
5 · Client fingerprint	403 on the first request: fix headers and TLS, or use a real browser.
6 · IP type	Still blocked: move to ISP, residential or mobile, then an unblocker.

The order in which to check a failing proxy setup.

ProxyLook's free tools

All six are free and run in the browser.

- What Is My IP shows the public IP, location, ISP and ASN a website sees.
- Proxy Checker returns the country, ASN, ISP and network type for a proxy IP.
- WebRTC Leak Test checks whether the browser exposes the real IP through WebRTC.
- User-Agent Parser breaks a user-agent string into browser, engine, system and device.
- Compatibility Checker suggests which proxy type fits a target site and use case.
- Proxy Cost Calculator estimates monthly spend from bandwidth and price per GB.

KEY TAKEAWAYS

- Score providers during a pilot on real targets, not on list price or pool size.
- Run the pre-launch checklist before every new scraper, profile or automation.
- A 407 comes from the proxy, and a 200 can still be a block.
- Troubleshoot in order: credentials, connectivity, exit IP, rate, client fingerprint, IP type.

KEEP GOING

Where to go next

A book can explain the method. The numbers live on the site, where they stay current.

Everything below is free and needs no account.

Provider directory

Every proxy network ProxyLook lists, with filters for type, price and location.

`proxylook.com/all-providers`

Live pricing

Current entry prices by proxy type, side by side.

`proxylook.com/pricing`

Free tools

Proxy checker, IP lookup, WebRTC leak test and a cost calculator.

`proxylook.com/tools`

AI Proxy Finder

Describe the job and get a shortlist drawn from the directory.

`proxylook.com/assistant`

The Multi-Account Handbook

Antidetect browsers, cloud phones and proxies, working together.

`proxylook.com/ebooks/multi-account-handbook`

Scraping Without Blocks

A developer’s guide to proxies, fingerprints and reliable data collection.

`proxylook.com/ebooks/scraping-without-blocks`

ProxyLook

Independent research on proxy networks and antidetect browsers. Free to read, free to share.

proxylook.com