

THE DEAL

Everything today fits inside a free tier. Here's the whole budget.

- **Hosting a static site or SPA** → ₹0 forever on Cloudflare. Static asset requests are free and unlimited , on every plan.
- **An app with an API** → ₹0 up to 100,000 requests/day. That's a real app with real users.
- **A database** → ₹0 up to 5 GB on Cloudflare D1, 1 GB on Neon.
- **A custom domain** → the domain is the only thing you actually have to buy (~₹800–1,200/year). TLS is free everywhere.
- **The escape hatch** → \$5/month (₹475) on Cloudflare Workers Paid covers almost anything a small project does.

Mindset: free tiers are not a trial. For a portfolio, a landing page, a side project or an internal tool, the free tier is the production platform.

TODAY'S REAL COST

₹0

Static hosting	free forever
API / functions	free tier
Database	free tier
TLS certificates	free
Domain name	₹ ~1k / yr

Note: a "free subdomain" (*.workers.dev, *.netlify.app, *.pages.dev) costs nothing at all — perfect for the first deploy.

Four layers. Every hosting question is really a question about one of them.



1 · BUILD

`npm run build`

Turns your source into plain files in `dist/` (or `build/` , `out/`). If this fails, no host on earth can help you. **Most "deploy" problems are actually build problems.**

2 · FILES

Static is the whole trick

A deployed site is just `index.html` , JS, CSS and images. Nothing runs on the server unless you ask for it. That's why hosting is free — it's file serving, not computing.

3 · HOST + CDN

Files on 300+ machines

The host copies your folder to edge servers worldwide. Indian users hit a Mumbai/Delhi PoP, not a server in Virginia. Free on every platform here.

4 · DNS + TLS

Your name, their machine

DNS points `yourdomain.com` at the host; TLS makes it `https` . Both free and automatic once DNS is right — this is where beginners lose an evening.

Diagnostic habit: when a deploy is broken, ask "which layer?" — build log green? files in the right folder? host serving them? domain resolving? You just cut the search space by 75%.

Pick your host by the shape of your project, not by the brand.

YOUR PROJECT	FREE PICK	WHY	ADD THIS ONLY IF...
Static portfolio, landing page, docs	Cloudflare Workers static assets	Asset requests are free and unlimited. No egress meter to get ambushed by.	never
SPA React/Vite, client-side routing	Cloudflare <code>not_found_handling: single-page-application</code>	Same free asset serving, one config key handles deep links.	you want PR previews → Netlify
Needs an API hide a key, webhooks, auth callback	Cloudflare Workers (100k req/day free)	Keep the secret server-side; ₹0 until real traffic.	Python/long jobs → Render/Railway
Needs a database	Cloudflare D1 or Supabase	D1 ships with Workers; Supabase gives DB + auth with no server code.	real Postgres → Neon free

Don't put a user-facing app on **Render Free** — it sleeps after 15 minutes idle and takes 30–60 s to wake. Someone opening your link from WhatsApp thinks the site is dead.

Don't ship a paying client's site on **Vercel Hobby** — that plan is limited to non-commercial personal use, and **Firebase free** silently disables your site after 10 GB/month.

What each free tier actually gives you — and how it fails.

PLATFORM	FREE TIER HEADLINE	FIRST LIMIT THAT BITES	PAID ENTRY
Cloudflare Workers + Pages	Static assets unlimited · 100k Worker req/ day · 10 ms CPU	Dynamic requests at 100k/day → Error 1027/429. Static: practically nothing.	\$5/mo (₹475)
Netlify	300 credits /month (hard cap)	~20 production deploys (15 credits each) — not traffic	\$9 Personal / \$20 Pro
Vercel	100 GB transfer · 1M invocations · 4 CPU-hrs	Bandwidth/CPU; functions default to iad1 (US)	\$20/mo (non-commercial free)
Firebase Hosting	10 GB storage · 10 GB/month transfer	Transfer cap → site disabled , no warning	Blaze PAYG
GitHub Pages	1 GB site · 100 GB/mo (soft)	Public repos only on free; no previews; no server	\$0 (Pro \$4 for private)

FAILS LOUDLY

Cloudflare · Netlify

You get 429s or a paused site. Annoying, honest, fixable — nothing is lost, nothing is billed behind your back.

FAILS QUIETLY

Firebase · Supabase

Firebase disables the site at the cap; Supabase pauses a free project after 7 days of low activity. Your app "just stops".

FAILS STUPIDLY

Render free · Railway

Sleeps after 15 min idle / deletes free Postgres at 30 days. Great for demos, wrong for anything someone depends on.

Start new projects on **Workers**. Pages still works — it's just no longer the default.

Cloudflare's own Pages page now opens with a warning box: "[Workers supports most Pages use cases... Start new projects with Workers.](#)" Pages isn't shut down and still gets fixes — but the tutorials that say "use Pages" are pre-2025.

What changed: static assets went GA on Workers (2025), and Workers gained `_headers`, `_redirects` and a 5-minute CPU ceiling on paid. So the "Pages-only" features moved over.

- **Free asset serving** — static requests aren't metered at all, on any plan
- **No cold starts** — V8 isolates, not containers. Nothing to wake up
- **One deploy** — `wrangler deploy` ships script + assets together

```
// wrangler.jsonc – the only file you need
{
  "name": "my-app",
  "compatibility_date": "2026-10-24",
  "assets": {
    "directory": "./dist",
    "not_found_handling":
      "single-page-application"
  }
}
```

Static-only project: no `main` script needed. Add `main` + a `fetch` handler the day you need an API.

Two ways in. Use Git for real projects, CLI for the first one.

PATH A · GIT-CONNECTED

Push to `main` → it builds itself

Dashboard → Workers & Pages → Create → connect GitHub → pick repo. Autoconfig detects the framework and opens a **PR** adding `wrangler.jsonc`, with a preview deploy. Merge → live.

```
# one-time: local build + first push
git add -A && git commit -m "deploy"
git push origin main
# → preview URL on the PR → merge → production
```

Free Pages builds: **500/month, one concurrent**, 20-min timeout. Workers Builds replace this for Workers projects.

PATH B · CLI (NO GIT)

Scaffold, run locally, ship

Best for "I have a folder and I want it online in four minutes".

```
# brand-new project
npm create cloudflare@latest -- my-app --framework=react
cd my-app
npx wrangler dev          # local, real runtime
npx wrangler login
npx wrangler deploy       # assets + code in one shot
```

```
# existing folder → Pages, no Git
npm run build
npx wrangler pages deploy ./dist
```

Drag-and-drop on Pages caps at **1,000 files**; the CLI handles 20,000 (25 MiB each).

Build image defaults (Pages v3): Node **22.16.0**, npm 10.9.2, pnpm 10.11.1, Bun 1.2.15. Override with a `NODE_VERSION` env var or a committed `.nvmrc`. v2 sunsets 23 Feb 2027.

Static hosting: unlimited. The meter starts when code runs.

FREE & UNLIMITED

Static asset requests + bandwidth

Not metered on any plan. A portfolio can be hugged to death by Reddit and you pay nothing. Only real caps: **20,000 files** (25 MiB each).

METERED · WORKERS FREE

100,000 requests/day · 10 ms CPU/request

Resets 00:00 UTC. Over the line you get `Error 1027` (or 429 on `run_worker_first` routes). CPU excludes waiting on fetch/DB — a typical Worker uses ~2 ms.

WHAT ACTUALLY BREAKS FIRST

SHAPE	WALL	SYMPTOM	FIX
Static site / SPA	nothing traffic-wise	—	stay free forever
App with API routes	100k req/day	1027 / 429 around ~70k page views/day	\$5/mo Workers Paid
Heavy SSR / image work	10 ms CPU	1102 Worker exceeded resource limits	\$5/mo (30M CPU-ms, 5 min max)
Busy Git workflow	500 builds/mo, 1 concurrent	queue stalls behind one build	Pro zone plans, or build locally

\$5/mo is the whole upgrade story: Workers Paid includes 10M requests and 30M CPU-ms. Note that the \$20 Cloudflare Pro zone plan is a different product and does nothing for your Worker limits.

Three doors in. Pick one and stop mixing them.

DRAG & DROP

app.netlify.com/drop

Runs your build **only** if you're logged in and it recognises the framework. Updates = re-drop the folder. **Use for:** a one-off demo, today's live deploy, nothing you'll edit again.

GIT-CONNECTED

The default for real projects

Connect GitHub → every push builds and deploys. Every PR gets a Deploy Preview URL. Instant rollback. **Use for:** anything you'll still be working on next week.

NETLIFY CLI

Scriptable deploys

`netlify deploy` uploads what's already built; it does **not** run your build unless you run `netlify build` first. **Use for:** monorepos, CI-less workflows, agents.

```
npm i -g netlify-cli
netlify login
netlify deploy           # draft URL – safe
netlify deploy --prod    # production
netlify deploy --allow-anonymous # no account; claim within 1 hour
```

Trap: manual/CLI deploys don't run your build command. If `dist/` is stale, you'll ship yesterday's code and swear at the CDN.

One netlify.toml fixes the SPA 404 — and makes your project reproducible.

```
# netlify.toml - commit this file

[build]
  base      = "apps/web/"    # dir w/ package.json
  command   = "npm run build"
  publish   = "dist/"        # relative to base

[build.environment]
  NODE_VERSION = "22"

[[redirects]]
  # SPA fallback - LAST rule
  from    = "/*"
  to      = "/index.html"
  status  = 200              # 200 = rewrite, 301 = loop
```

netlify.toml overrides the dashboard wherever both set the same key. Env-var **secrets** belong in the UI/CLI — toml values are build-time only and functions can't read them at runtime.

WHY DEEP LINKS 404

Netlify is a CDN, not a server

/dashboard isn't a file in dist/, so it 404s before your router ever loads. The status = 200 rewrite serves index.html instead — the URL stays, the app boots, the router takes over.

DEPLOY CONTEXTS

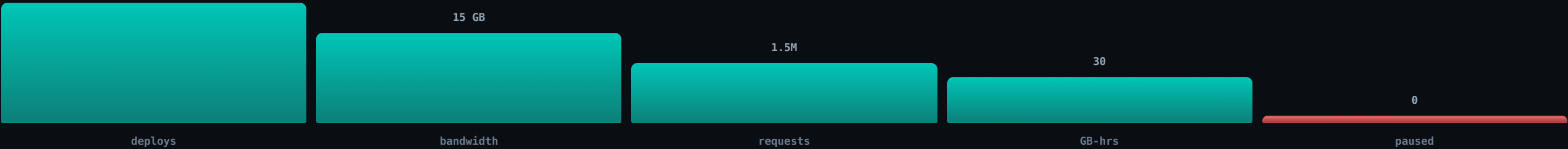
Different values per environment

[context.production], [context.deploy-preview], [context.branch-deploy] — each PR can point at a staging API instead of production.

Public-by-design: anything prefixed VITE_, NEXT_PUBLIC_ or REACT_APP_ is compiled into the browser bundle. Netlify's docs say it plainly — don't put secrets there.

Netlify's free tier is metered in credits now — and deploys are expensive.

Since **4 Sep 2025**²⁰ every new account is credit-based. Free = **300 credits/month, hard cap**. A production deploy costs **15 credits**.



All four are the same 300 credits — you don't get each. Spend them once.

The vibecoding failure mode: 20 pushes to `main` in a month burns the entire free plan before you have a single user. Previews and branch deploys are free (0 credits) — but production isn't.

PLAN	CREDITS	PRICE	≈ INR
Free	300 (hard cap)	\$0	₹0
Personal	1,000	\$9/mo	≈ ₹855
Pro	3,000	\$20/mo	≈ ₹1,899
Extra concurrency	—	\$40	≈ ₹3,798

At ₹95/USD. GST may apply at checkout. **Zero credits ⇒ every project on the account is paused** until the next cycle — there's no overage on Free.

Teach the pricing page, not the blog: the Sept 2025 launch post advertised Pro with 5,000 credits; today \$20 buys 3,000 and 5,000 costs \$33. Also gone: separate build-minute and function-invocation quotas.

Five platforms, ranked on the thing that matters first: the free tier.

PLATFORM	FREE TIER	COLD START	CUSTOM DOMAIN	BEST FOR
Cloudflare	unlimited static + 100k/day	none	free TLS	static, SPA, edge APIs
Netlify	300 credits (~20 deploys)	~200–800 ms	free TLS	best DX + PR previews
Vercel	non-commercial only	~250–500 ms	free TLS	Next.js specifically
Firebase	10 GB/mo then off	none (static)	free TLS	Firestore/Auth apps
GitHub Pages	1 GB, public repos	n/a	free TLS	OSS docs, portfolios

INDIA LATENCY

Cloudflare wins

Anycast with Indian PoPs — single-digit-ms TLS for Mumbai/Delhi. **Vercel functions default to iad1 (Virginia)**; set `"regions":["bom1"]` or every API call crosses the Atlantic.

EXIT COST

Static files are portable

Leaving a static host costs an afternoon of DNS. The real lock-in is features: Vercel ISR/middleware, Firebase Auth rules. Cloudflare + Docker/Postgres is the most portable.

REGION GAP

No India region here

Render and Railway offer Singapore (~40–70 ms) at best. Fine for admin tools, noticeable for real-time apps.

The five things that will break your first deploy — and the fix for each.

01
Deep link 404

Works when clicked, breaks when pasted. Netlify: `/* /index.html 200`. Vercel: a rewrite in `vercel.json`. **Cloudflare Pages already does this** — unless your generator emitted a `404.html`, which switches the fallback off.

02
Blank page after deploy

Almost always an env var added after the last build. Vite inlines `import.meta.env` at build time — the dashboard change does nothing until you **redploy**.

03
"Dev works, build fails"

Dev servers transpile only; `npm run build` runs `tsc` and type-checks. Unused vars and implicit `any` are invisible until CI. Fix: `"predev": "tsc --noEmit"` so it fails at home.

04
Wrong Node version

`crypto.hash is not a function`, `ERR_UNKNOWN_FILE_EXTENSION`. Hosts default differently (Pages 22.16, Vercel 24.x). Fix both ends: `"engines":{"node":"22.x"}` and a `.nvmrc` saying `22`.

05
Case-sensitive imports

`./Button` vs `button.tsx`: fine on Windows/macOS, fatal on Linux build machines. Rename the file to match the import exactly and check `git status` for case-only renames.

+
Honourable mentions

Wrong output dir (`dist` vs `build`), root directory unset in a monorepo, and committing a lockfile the host then ignores. All three are in the checklist later.

Your API key is probably in your JavaScript bundle. Check in 30 seconds.

The rule: anything the browser can read is public. AI tools happily write `VITE_OPENAI_API_KEY=sk-...` because it "works" — and then it ships to production inside a `.js` file.

```
# grep your live bundles for key shapes
curl -s https://yoursite.com \
  | grep -oE '/assets/[^"]+\.js' | sort -u \
  | while read f; do
    curl -s "https://yoursite.com$f" \
      | grep -oE 'sk-[A-Za-z0-9_-]{20,}|AIza[0-9A-Za-z_-]{35}|eyJhbGciOiJI[A-Za-z0-9]{20,},' \
      && echo "  ↑ leaked in $f"
  done
```

Or DevTools → Sources → `Ctrl/Cmd+Shift+F` → search `sk-`, `service_role`, `secret`.

ORDER MATTERS

Rotate, then fix, then redeploy

- 1. **Revoke/regenerate** at the provider — the bundle is cached on a CDN, so this is the only instant fix
- 2. **Move the call server-side** (a Worker/function); the new key gets no `VITE_` prefix
- 3. **Redeploy** + purge cache
- 4. Rewriting git history is cosmetic — do it, but it's not the fix

Public repos are scraped within minutes. If it was ever committed, assume it leaked — `git log -p | grep -i api_key`.

The good pattern: browser → your function (same origin) → provider. Key stays on the server, CORS disappears, and you can add rate limits later.

"It needs a backend" — the free ways to get one.

EASIEST

Hosted API

Supabase, Firebase, Airtable, a no-code API. Zero server code, auth and DB included. Best default for a vibecoded app.

MOST INSTRUCTIVE

Colocated functions

`api/hello.js` on Vercel, `netlify/functions/` on Netlify, a Worker `fetch` on Cloudflare. Same repo, same deploy, secret in env vars. "The same file becomes an endpoint."

ONLY WHEN NEEDED

Separate service

Render / Railway / Fly for long-running jobs, Python, websockets, or a real DB connection pool. Costs money or sleeps.

CORS, DECODED

It's not your frontend

No `'Access-Control-Allow-Origin'` header means the response lacked the header, so the browser refused to hand the body to your JS. And often the real bug is a 401/500 — error responses skip the CORS middleware, so you debug CORS while the actual problem is auth.

```
// Cloudflare Worker – the whole backend
export default {
  async fetch(request, env) {
    const url = new URL(request.url);
    if (url.pathname.startsWith("/api/"))
      return Response.json({ ok: true });
    return env.ASSETS.fetch(request);
  }
};
```

Best CORS fix of all: **don't have CORS**. Proxy `/api/*` through your own origin.

The last mile. Free TLS, three classic ways to lose an evening.

GOTCHA 1

Proxied (orange cloud) blocks certs

While your host issues a certificate, the DNS record must be **DNS-only (grey)**. Cloudflare proxying in front of Netlify/Vercel means they can't complete validation. Turn proxying on after the cert is active.

GOTCHA 2

Redirect loop = SSL mode

`ERR_TOO_MANY_REDIRECTS` is almost always Cloudflare **Flexible** SSL in front of an origin that already forces HTTPS. Fix: **Full (strict)**, or enforce HTTPS at the edge only.

GOTCHA 3

Apex vs www

Pick one canonical host and 301 the other. Apex can't be a CNAME — it needs Cloudflare/ALIAS or an A record. Don't set apex → www at the host and www → apex at DNS.

WHICH DNS RECORD?

Apex <code>example.com</code>	Cloudflare zone (nameservers) or ALIAS at your DNS host
Sub <code>www.</code> / <code>app.</code>	CNAME → <code>your-site.netlify.app</code> / <code>you.workers.dev</code>
External DNS + Workers	Not possible — Workers custom domains must be in a Cloudflare zone

```
# is DNS right yet?
dig +short learn.yourdomain.com
curl -sI https://learn.yourdomain.com | head -3
# exactly one 301 to the canonical host = healthy
```

Certificate still "pending" after 24 h ⇒ the record is wrong (or proxied). TLS itself is free everywhere here.

Databases and auth: what's actually free, and what quietly expires.

OPTION	FREE TIER	WATCH OUT
Cloudflare D1 SQLite	5M rows read/day · 100k written/day · 5 GB	Tied to Workers; SQLite semantics, not Postgres
Neon Postgres	1 GB/project, up to 20 GB total	HTTP driver works from serverless; scale-to-zero adds latency
Supabase	Postgres + auth + storage	Free project pauses after 7 days of low activity — the #1 "my app died" ticket. Ping it with a cron.
Clerk auth	50,000 monthly retained users	Fastest correct auth; costs scale with MAU

Never implement login by comparing passwords in client JS. If the check runs in the browser, the password is in the bundle — along with everyone else's.

Free-tier hygiene: add a tiny cron/uptime ping to anything that sleeps or pauses. Ten minutes of setup saves a week of "why is my app down".

Two minutes before every deploy.

Local truth

- `npm ci && npm run build` in a clean clone
- `npx tsc --noEmit` passes
- Lockfile committed (the host resolves versions from it)
- `engines.node` **and** `.nvmrc` set
- Root directory + output directory correct for this repo shape

Configuration truth

- Every build-time env var exists in the host — for **Production and Preview**
- `.env` in `.gitignore` ; `.env.example` committed
- Grep source for `VITE_` / `NEXT_PUBLIC_` / `REACT_APP_` + a key
- SPA fallback present, status **200**
- Nothing large in `public/` that could blow a 25 MiB limit

The five-second version: does it build clean, does it know its Node version, and did I leak a key? Those three catch most disasters.

Two minutes after every deploy — in strict order.

- Open an **incognito** window (no cached bundle lying to you)
- Hard-load a **deep link directly** : `site.com/dashboard` — don't click to it
- DevTools → Network: any 404s on `.js` / `.css` ? Console: red errors?
- Run the bundle grep from the secrets slide
- Test one **write** path, not just a read
- Read the function logs, not just the build log
- `curl -I https://www...` → exactly one 301 to canonical

DEFINITION OF DONE

Three URLs, all green

① the bare host URL · ② one deep link pasted cold · ③ the custom domain over https. If all three render, you shipped.

Anything else — a blank page, a console 404, a "pending" certificate — is a layer-3 problem you can now name.

Don't declare victory from the preview URL alone. Preview and production can differ in env vars — the single most common "it worked yesterday" cause.

NOW WE DO IT

Live deploy, then homework.

WHAT WE'LL DEPLOY TOGETHER

- **His project** , from local folder → public URL (CLI path, ~4 min)
- Break it on purpose: deep link 404 → one-line fix
- Then connect Git so every future push deploys itself
- Custom domain + TLS, watching the DNS propagate
- Grep the live bundle for leaked keys

Homework: ship two more projects this week — one static, one with a function. Then upgrade nothing. The point is that ₹0 holds.

IF HE REMEMBERS ONLY THREE THINGS

The whole session, compressed

- 1. **Match the host to the shape, not the brand.** Static or SPA → Cloudflare, because asset requests are free and unlimited and there's no egress meter to ambush you.
- 2. **Pick a free tier that fails loudly.** 429s and paused sites are honest. Silent disabling and 30-second cold starts are not.
- 3. **Budget ₹0, then ₹475.** Static is free forever; the day you add real backend traffic, \$5/month covers almost anything.

Docs: [Workers](#) · [Netlify](#) · [Vercel](#) — all facts here verified 24 Oct 2026.

Downloads, and what to do this week.

FILE	WHAT IT IS
cloudflare-hosting.zip 11 KB	Skill + preflight script for your AI agent. Drop into Hermes, <code>.claude/skills/</code> , or append to <code>AGENTS.md</code> .
SKILL.md	Read it raw — three modes (CLI / browser / walk-me-through), the verification gate, and the troubleshooting table.
ship-it-free.pdf	This deck, 21 pages, printable. Keep it offline for the next deploy.
Sidequest deck · PDF	The AI-coding-stack half: DeepSeek Harness, Z.ai GLM plan, ZCode, OpenCode — and what each actually costs.

THIS WEEK

Ship three things

- A **static** page — no build step, drag it up. Proves the concept in ten minutes.
- An **SPA** with routing — the deep-link fix is the whole lesson.
- One **function** — hide a key, learn what a backend actually is.

Then upgrade nothing. The point is that ₹0 holds.

Give the agent the skill, not the tutorial. Point your agent at `cloudflare-hosting.zip` and say "deploy this". It will ask you one question, then do the rest — and it can't declare victory without running the checks.