



FRONTEND & BEYOND · CALGARY

Zero Servers, Zero Dollars

Shipping real apps on Cloudflare



ABOUT ME



@gatezh

serge@yyc ~

```
$ curl -i https://gatezh.com/whoami
HTTP/2 200
server: cloudflare
cf-ray: 9a1f3c7e2b4d-YYC
{
  "name": "Serge Gatezh",
  "role": "Full Stack Developer",
  "based_in": "Calgary",
  "building_web_apps_since": 2018,
  "stack": ["React", "TypeScript", "Cloudflare"],
  "cares_about": [
    "low cognitive load",
    "sensible defaults",
    "secure design"
  ],
  "servers_i_manage": 0
}
```

That last line is what tonight is about. ↑

What I want you to leave with

01 Front-end friendly

The whole backend, no infrastructure to run.



Hosting
Pages



Functions
Workers



Database
D1



Storage
R2



Pipelines
Workflows



AI
AI Gateway



Anti-bot
Turnstile



Media
Images

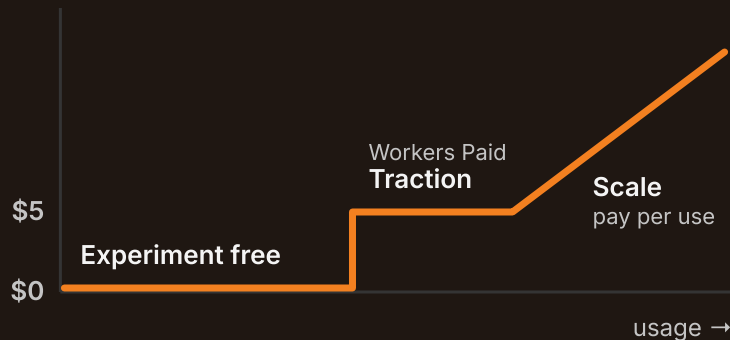


Access
Tunnel

If you can ship a React app, **you can ship this.**

02 Free tier = production

Experiment for free. If it picks up, it stays cheap.



Pay later, **if ever.** Illustrative shape, not to scale

LIFE BEFORE

"Just host it yourself," they said

```
worker_processes auto;
events { worker_connections 1024; }
http {
    server {
        listen 443 ssl;
        http2 on;
        server_name example.com;
        ssl_certificate      /etc/letsencrypt/live/example.com/fullchain.pem;
        ssl_certificate_key  /etc/letsencrypt/live/example.com/privkey.pem;
        ssl_protocols TLSv1.2 TLSv1.3;
        gzip on;
        gzip_types text/css application/javascript;
        location /api/ {
            proxy_pass http://app:3000;
            proxy_set_header Host $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }
    }
}
```

...plus `docker-compose.yml`, cert renewals, firewall rules, OS updates...

What I actually wanted

```
npx wrangler deploy
```

A million parameters, and a lot of ways to set them up wrong so it "works" badly. All I needed was the simple thing, **delivered**.

PROLOGUE

How it started

A personal website, and three homes for it.

GitHub Pages + Jekyll: it worked... kind of

WHY I MOVED ON

```
my-blog ~  
  
$ gem install jekyll bundler  
$ bundle install  
$ bundle exec jekyll serve  
...wait, which Ruby do I have?
```



Jekyll or nothing

Back then, Jekyll was the only generator GitHub Pages built for you.

Ruby, for a front-end dev

Gems, Bundler, Ruby versions: a toolchain I didn't know.

Slow-ish, a bit hacky

It worked. It just never felt quite right.

DECODING THE BUZZWORD

What's Jamstack?

J

JavaScript

The React app and the Hono Workers that make it dynamic.

A

APIs

Workers, D1, R2, Workflows, AI Gateway. Server-side logic as services.

M

Markup

Pre-built HTML from Hugo, served from the edge in milliseconds.

Pre-render what you can. Call APIs for the rest. **No server you have to babysit.**

Netlify + Hugo: great, until the fine print

  Found Hugo here: one binary, no Ruby. Smooth deploys. But on the free Starter plan:

Bandwidth

FREE

100

GB / month

→

PAID

\$55

per extra 100 GB

Pro (\$19 per member/month) included 1 TB.

Analytics

FREE

None

not included

→

PAID

\$9

/ month per site

Netlify Analytics was a paid add-on, per site.

Edge network

FREE

\$0

standard edge

→

PAID

Custom

Enterprise price

High-Performance Edge: more global PoPs, 99.99% SLA.



Then, on a podcast: "Cloudflare has the **fastest CDN**." Hmm.

THE PODCAST CLAIM

"Cloudflare has the **fastest CDN**"

So I fact-checked it.

330+

cities in 100+ countries

13,000+

networks directly interconnected

~50 ms

from 95% of the world's Internet-connected population

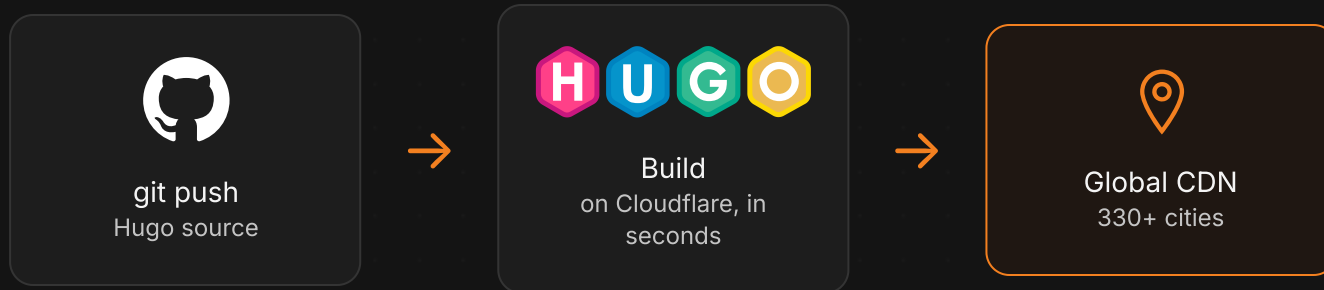
44%

of top networks where Cloudflare is fastest
(Cloudflare's own benchmark)

Verdict: true, with a footnote. It trades the top spot with Fastly on some networks. Where Cloudflare wins clearly is **reach**: more peering means fewer hops to users.



Pages: the static site that just works



Hugo is fast. The build pipeline is fast. The CDN is fast.
Plain HTML, delivered in milliseconds around the world.

Static requests: unlimited

500 builds / month

20,000 files / site

\$0

Workers: a contact form without a server

```
import { Hono } from 'hono'
import { Resend } from 'resend'

const app = new Hono<{ Bindings: Env }>()

app.post('/api/contact', async (c) => {
  const { name, email, message } = await c.req.parseBody()
  const resend = new Resend(c.env.RESEND_API_KEY)

  await resend.emails.send({
    from: 'Website <form@example.com>',
    to: 'me@example.com',
    replyTo: String(email),
    subject: `New message from ${name}`,
    text: String(message),
  })
  return c.redirect('/thanks')
})

export default app
```

A **serverless function** that only runs when a request comes in. Attached to the static Hugo site.



Hono for routing



Resend for email

Free tier

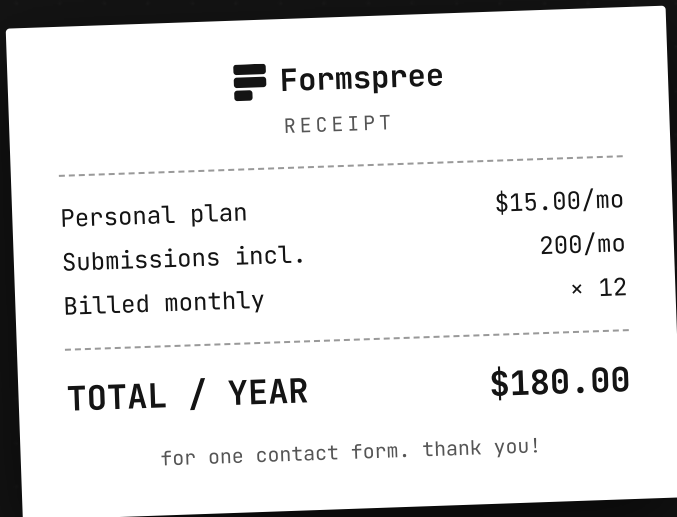
Workers: 100,000 requests / day

Resend: 100 emails / day

Resend: 3,000 / month

MEANWHILE, IN THE REAL WORLD

Then a friend told me: **people pay for forms**



» My Worker + Resend

\$0

3,000 emails / month free, and ~20 lines of code I own.

"I wish I had the problem of hitting that limit on my contact form."

There's a whole business in selling website forms. Bizarre.

Turnstile: spam protection, no CAPTCHA

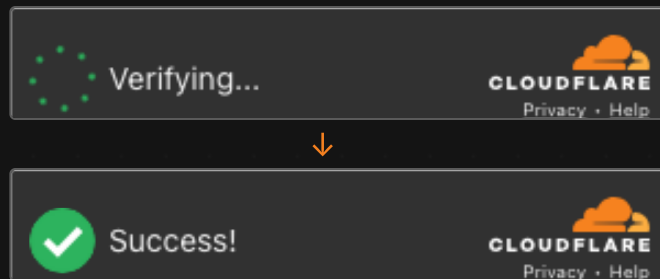
```
const CF = 'https://challenges.cloudflare.com'

app.post('/api/contact', async (c) => {
  const body = await c.req.parseBody()
  const res = await fetch(`${CF}/turnstile/v0/siteverify`, {
    method: 'POST',
    body: new URLSearchParams({
      secret: c.env.TURNSTILE_SECRET_KEY,
      response: String(body['cf-turnstile-response']),
    }),
  })
  const { success } = await res.json()
  if (!success) return c.text('Are you a robot?', 403)

  // ...send the email with Resend, like before
})
```

Form services sell spam filtering. Turnstile does it **for free**, and visitors never solve a puzzle.

What visitors see



On the page: one script tag + `<div class="cf-turnstile">`. In the Worker: verify the token (left). The widget alone doesn't protect the form.

Free plan

Unlimited challenges

Up to 20 widgets

Works on any site

PLOT TWIST

Pages → Workers

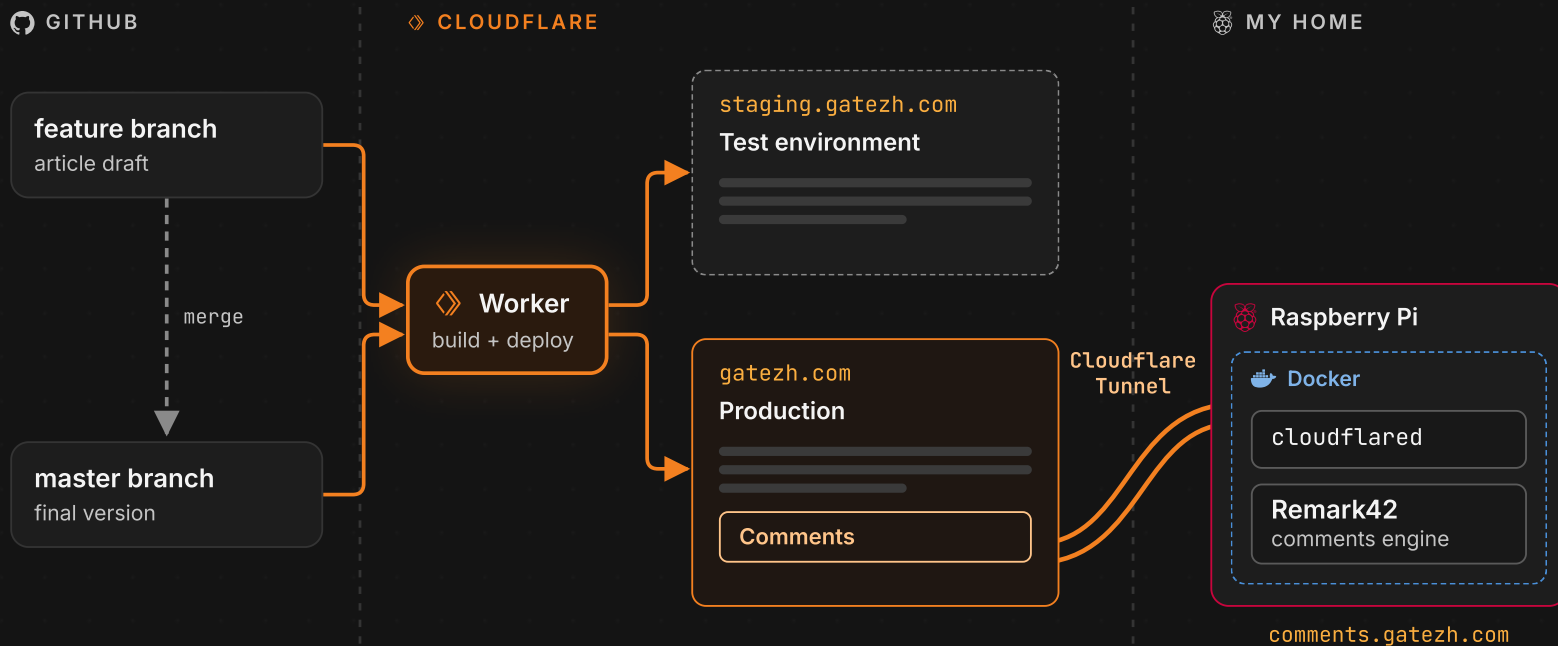
Cloudflare now steers new projects to **Workers**, which can serve static assets *and* run code, with a broader feature set than Pages.

So I moved my Hugo site, contact form included, from Pages to a Worker.

- Static asset requests: still free and unlimited
- Config is a little more work than Pages
- My hunch: Pages gets retired eventually

```
// wrangler.jsonc
{
  "name": "my-site",
  "main": "src/worker.ts",           // the contact form API
  "compatibility_date": "2026-10-01",
  "assets": {
    "directory": "./public",        // Hugo's build output
    "not_found_handling": "404-page" // Hugo's 404.html
  }
}
```

Tunnel: a Raspberry Pi online, **no open ports**



No public IP needed

Outbound-only: cloudflared dials out

No inbound ports to open

Access: put **staging.gatezh.com** behind a login

🔒 staging.gatezh.com

Get a login code emailed to you

me@gatezh.com

Send me a code



4

8

1

2

0

7

Illustrative: the email one-time PIN flow

Keep the copy out of search

A public duplicate of my blog has no business in Google's index.

A safe playground

Preview drafts and test changes before they ship.

Login: email one-time PIN

Stay signed in up to 1 month

No changes to the site itself

Heads-up: new Zero Trust accounts default to the Cloudflare identity provider. Add **One-time PIN** under Integrations → Identity providers.

Images: **one URL**, every size and format

gatezh.com/cdn-cgi/image/width=400,format=auto/uploads/photo.jpg

your domain

the magic prefix

what you want

the original (R2, any origin)

SAME IMAGE, MANY VISITORS

1st visitor	MISS	fetch original, resize, cache	+1
2nd visitor	HIT	served from the edge cache	+0
10,000th visitor	HIT	still the edge cache	+0

Billed: 1 unique transformation this month.

5,000 unique transformations / month, free

- `format=auto` serves AVIF or WebP per browser, and still counts as **one**
- Over the limit on Free? Cached images keep serving, new ones fail. **No surprise bill.**
- Paid: \$0.50 per 1,000 more

ACT TWO

Getting serious

From a personal site to real, product-sized web apps.

A whole **React app** on a Worker? Yes.

```
$ npm create cloudflare@latest -- my-react-app \
  --framework=react

src/    React SPA, built by Vite
worker/ API Worker for /api/*
```

```
// vite.config.ts
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
import { cloudflare } from '@cloudflare/vite-plugin'

export default defineConfig({
  plugins: [
    react(),
    cloudflare({ viteEnvironment: { name: 'worker' } }),
  ],
})
```

Dev = production runtime

`vite dev` runs your Worker inside **workerd**, the same runtime as production.

One deploy, one unit

The React build and the API ship together.

Static assets are free

Asset requests are **free and unlimited**. Only requests that hit your Worker code count.

SSR when you want it

Official support for React Router and TanStack Start.

A real web app needs auth and a database

THE DETOUR



"I need PostgreSQL." So: **Neon** + **Better Auth**.

- Outside my Cloudflare setup, so every query took an extra trip
- Free database went to sleep when idle
- I had to build a wake-up mechanism

THE SIGN



Reading Better Auth's release notes, again: **native D1 support** in v1.5 (March 2026). Just pass the binding:

```
betterAuth({ database: env.DB })
```

I revisited my requirements. Did I really need Postgres? **No.**

5M rows read / day

100k rows written / day

5 GB storage

The 10 ms CPU budget

10 ms

CPU time per request on Workers Free

5 min

max CPU per request on Workers Paid (30 s default)

I/O is free

time spent waiting on fetches and database calls doesn't count as CPU

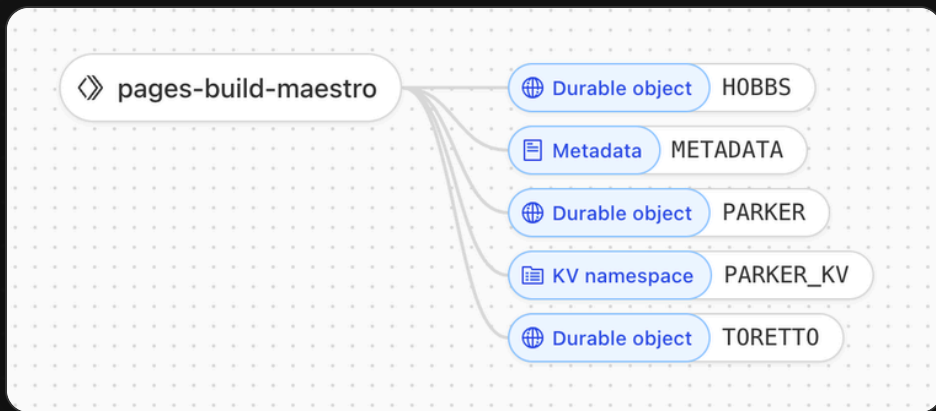
Password hashing is deliberately CPU-heavy. Better Auth's default, **script** in pure JavaScript, kept timing out on the free plan. The fix: Better Auth lets you plug in your own `hash` / `verify`, so I swapped in **PBKDF2-SHA256** from the Workers' built-in Web Crypto API (`crypto.subtle`).

Lesson: on the free plan, the first limit you feel is CPU per request, not traffic.

THE MAGIC PART

Bindings: no keys, no connection strings

DASHBOARD → YOUR WORKER → BINDINGS TAB



Cloudflare dashboard · Bindings view (Cloudflare Docs changelog)

Add a resource in config, and it shows up on `env`.
Cloudflare wires it behind the scenes.

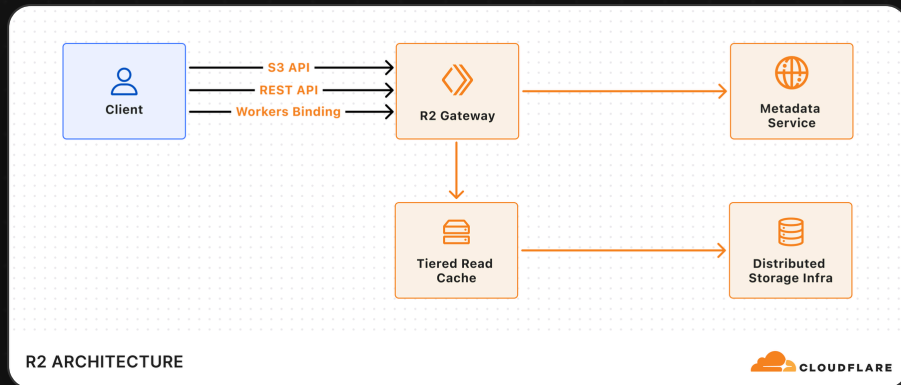
```
// D1
const user = await env.DB
  .prepare('SELECT * FROM users WHERE id = ?')
  .bind(id).first()

// R2
await env.FILES.put(`bills/${id}.pdf`, req.body)

// Workflows
await env.BILL_PIPELINE.create({ params: { id } })
```

"Add and connect external resources to your Worker without needing to manage permissions or API keys."

R2: file uploads with zero egress fees



S3-compatible. Use the AWS SDK, or just a binding. Download a file a million times, you still pay nothing for transfer.

50 GB stored + 1 TB downloaded per month

R2
\$0.60

Amazon S3
~\$84

Storage + egress only, request fees excluded. S3: us-east-1, \$0.023/GB-month, \$0.09/GB after the first 100 GB out. R2: \$0.015/GB-month above 10 GB.

10 GB-month free

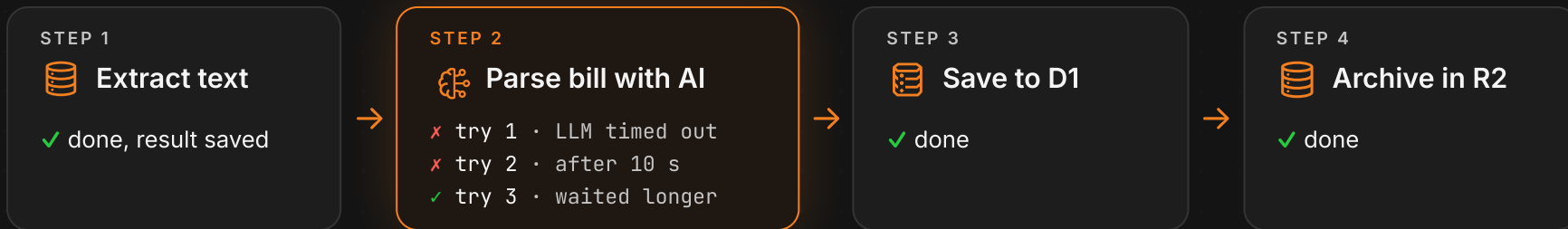
1M writes (Class A) / month

10M reads (Class B) / month

Egress: always \$0

Workflows: pipelines that survive failure

My bill pipeline is one Workflow with four steps. Here's a run where step 2 fails:



Finished steps stay finished

Step 1 isn't re-run when step 2 fails. Its result is saved.

Each step retries on its own

With the delay and exponential backoff you choose.

Minutes... or weeks

Steps can sleep, or pause for an event or an approval.

All the custom recovery logic I wrote? **Deleted.**

3,000 steps / day free

100 concurrent instances

up to 1,024 steps each

AI Gateway: for bills I've never seen

Known format → my own parser, no AI

Unknown format → an LLM, through AI Gateway



Workflow

step 2: parse bill



AI Gateway sits in the middle of every call

- Logs every request
- Tokens and cost, per model
- Caches repeated answers
- Rate limits
- Retries, or falls back to another model



Anthropic

bring your own key



Workers AI

Cloudflare-hosted models

← BACK TO THE WORKFLOW

```
{ "provider": "City Power",  
  "amount": 142.18, "due": "2026-11-05" }
```

JSON in my bill schema. Illustrative.

THE ONLY CODE CHANGE

```
baseUrl: `${gateway}/${account}/bills/anthropic`
```

Point the SDK at the gateway instead of the provider.

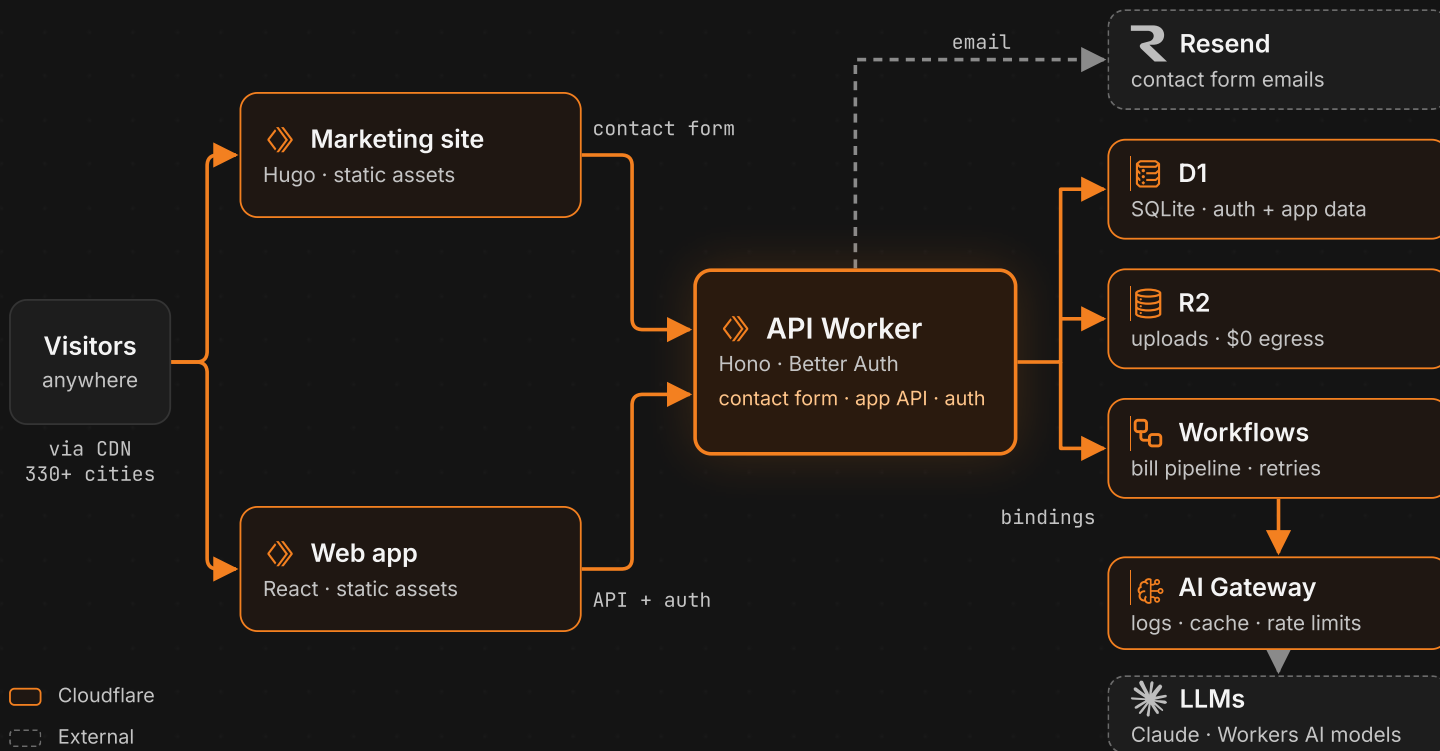
Analytics, caching, rate limiting: free

100,000 logs on Free

Workers AI: 10,000 neurons / day

WHERE I ENDED UP

The whole stack, one account, one bill: \$0



WHEN YOU OUTGROW FREE · WORKERS PAID IS \$5/MONTH

Free vs. Paid: **compute**

Requests

FREE

100k

per day

→

PAID

10M

per month included

Then \$0.30 per extra million

CPU time per request

FREE

10 ms

hard limit

→

PAID

5 min

max (30 s default)

30M CPU-ms per month included

Subrequests per request

FREE

50

→

PAID

10,000

Outbound fetch() and service calls from one request

Workers per account

FREE

100

→

PAID

500

Cron Triggers: 5 → 250 per account

Static asset requests: free and unlimited on both

Over the free limit: requests fail (Error 1027)

WHEN YOU OUTGROW FREE · WORKERS PAID IS \$5/MONTH

Free vs. Paid: **data**

D1 rows read

FREE

5M

per day



PAID

25B

per month included

Then \$0.001 per extra million

D1 rows written

FREE

100k

per day



PAID

50M

per month included

Then \$1.00 per extra million

D1 database size

FREE

500 MB

per database



PAID

10 GB

per database

Account storage: 5 GB free → up to 1 TB (\$0.75 / GB-month over 5 GB)

R2 storage

FREE

10 GB

per month



PAID

\$0.015

per GB-month over 10 GB

Egress always \$0 · R2 is billed separately from Workers Paid

WHEN YOU OUTGROW FREE · WORKERS PAID IS \$5/MONTH

Free vs. Paid: **workflows & AI**

Workflow steps

FREE

3,000

per day



PAID

500k

per month included

Then \$0.80 per extra 100,000

Concurrent workflow instances

FREE

100



PAID

50,000

Steps per workflow: 1,024 → 10,000 (up to 25,000)

Workers AI

FREE

10k

neurons per day



PAID

10k

neurons per day, then pay

Paid goes past the daily allocation at \$0.011 per 1,000 neurons

AI Gateway logs

FREE

100k

total



PAID

10M

per gateway

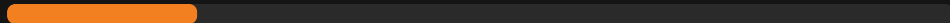
Analytics, caching, rate limiting: free on both

HOW FAR DOES FREE GO?

Napkin math: 100 daily active users

Worker requests

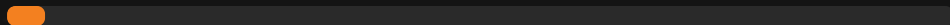
20,000 / 100,000 / day



20% of the free daily limit

D1 rows read

200,000 / 5,000,000 / day



4% of the free daily limit

D1 rows written

10,000 / 100,000 / day



10% of the free daily limit

Workflow steps

500 / 3,000 / day



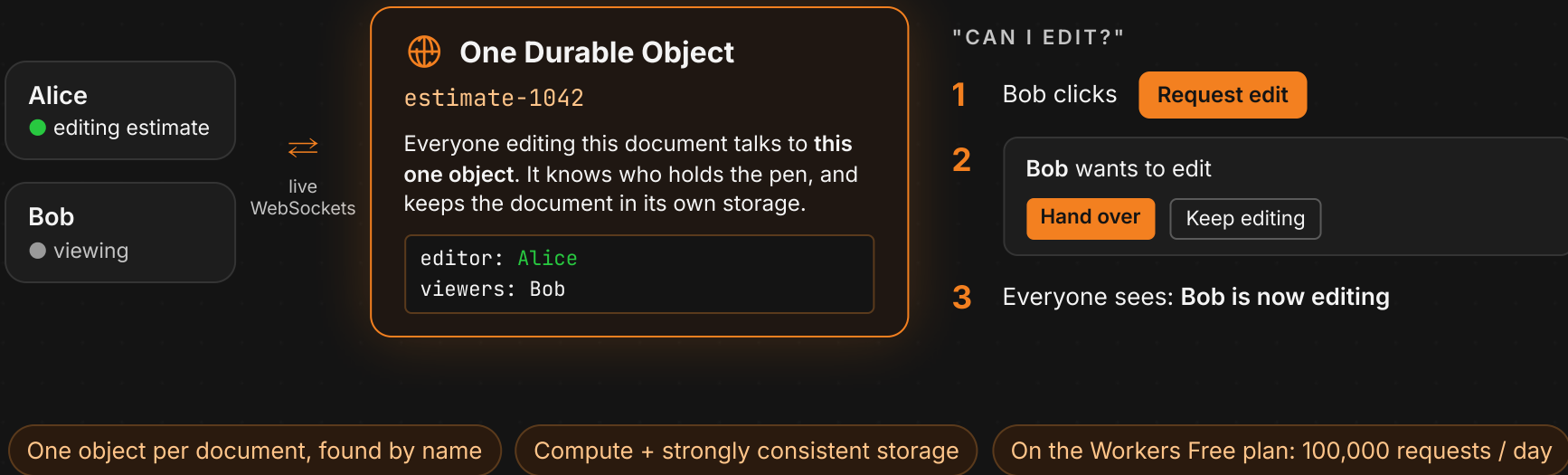
17% of the free daily limit

ASSUMPTIONS, PER USER PER DAY

- 200 requests (page loads + API calls)
- 2,000 rows read, 100 rows written
- 1 bill uploaded → 5 workflow steps

Comfortable headroom. The thing to watch is the **10 ms CPU** per request, not the volume.

Durable Objects: one editor at a time, **live**



BONUS 2 · BUILT FOR AI AGENTS

Let your **AI agent** drive Cloudflare

MCP Cloudflare MCP server

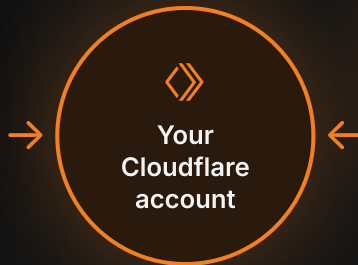
First

Point staging.gatezh.com at my new Worker

- search("dns records")
- execute(/* JS against a typed API */)
- ✓ Done.

2,500+ API endpoints

via just **2 tools**, in **~1,000 tokens** of context.



CLI the new **cf** command

Newest · beta

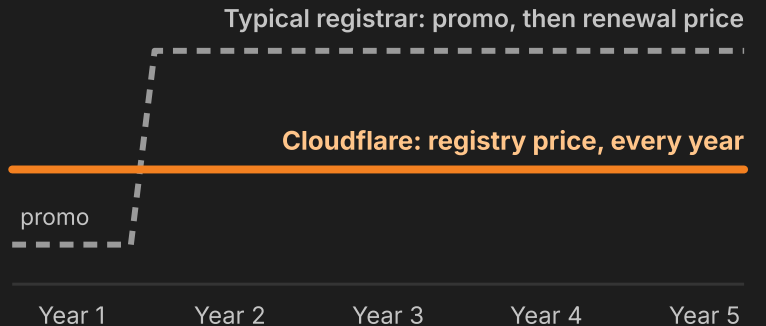
```
$ npm install --global cf
$ cf cli search "create D1 database"
$ cf schema d1 create
$ cf deploy
```

2,900+ commands

- Command search runs locally, no login
- Also **init**, **dev**, **deploy** for Workers
- Same commands for you and your agent

Domains at cost: **no renewal trap**

WHAT YOU PAY FOR THE SAME DOMAIN, YEAR AFTER YEAR



Illustrative shape, not real prices

\$0 markup. Registration, transfer and renewal at or below what registries and ICANN charge.

430+ top-level domains, from .com to .ai, .app and .dev

Free WHOIS redaction by default, one-click DNSSEC

New · beta

Registrar API: search, check the price and register a domain from code. It's in Cloudflare MCP by default, so **your AI agent can buy the domain too.**

One Cloudflare account per project

Simple naming

No prefixes on every Worker, bucket and database. `DB` is just `DB`.

Clean management

Separate dashboards, billing, access and API tokens. Easy to hand a project over.

Limits are per account

Free-tier allowances apply per account, so one busy project doesn't eat another's budget.

ACCOUNT

`personal-site`

`site`

`contact-form`

ACCOUNT

`web-app`

`app`

`DB`

`FILES`

ACCOUNT

`bills-app`

`app`

`DB`

`FILES`

`BILL_PIPELINE`

RECAP

One service at a time



Pages

static site



Workers

contact form, app



D1

database + auth



R2

file uploads



Workflows

bill pipeline



AI Gateway

unknown formats

01 Front-end friendly. No servers, no Nginx, no keys. Bindings do the wiring.

02 Free tier = production. Launch, learn, and pay \$5 only when you need to.

SOURCES

Every number in this deck

CLOUDFLARE PRICING & LIMITS

developers.cloudflare.com

[Workers pricing](#)

[Workers limits](#)

[D1 pricing](#)

[D1 limits](#)

[R2 pricing](#)

[Workflows pricing](#)

[Workflows limits](#)

[AI Gateway pricing](#)

[Workers AI pricing](#)

[Pages limits](#)

[Durable Objects pricing](#)

[Images pricing & caching](#)

CLOUDFLARE NETWORK & MORE

[Global network](#)

cloudflare.com

[Benchmarking edge network performance](#)

blog.cloudflare.com

[Migrate from Pages to Workers](#)

developers.cloudflare.com

[Turnstile plans](#)

developers.cloudflare.com

[React on Workers · Vite plugin](#)

developers.cloudflare.com

[Tunnel · Access sessions · one-time PIN](#)

developers.cloudflare.com

[cf CLI \(beta\) · Cloudflare MCP servers](#)

developers.cloudflare.com

[Cloudflare Registrar](#)

cloudflare.com

OTHER SERVICES

[Netlify pricing, 2022](#)

web.archive.org

[Netlify Analytics, 2021](#)

web.archive.org

[Resend pricing](#)

resend.com

[Formspree plans](#)

formspree.io

[Amazon S3 pricing](#)

aws.amazon.com

[Cloudflare Docs repo \(diagrams, icons\)](#)

github.com

[Iconify \(logos\)](#)

iconify.design

Thank you!

Questions?

 LET'S CONNECT

Scan to find me on LinkedIn

[linkedin.com/in/gatezh](https://www.linkedin.com/in/gatezh)

