Blueprint for Deterministic Edge Services: Cloudflare Workers, KV, and Scheduled Events

Author: Brian Barry, SETU, C00274624@gmail.com

Contents

Abstract 3

1 Introduction 4

2 Related Work 5

3 System Architecture 6

4 Implementation Details 8

5 Performance Evaluation 10

6 Limitations and Future Work 11

7 Conclusion 13

Appendix 14

Abstract

Edge computing executes application logic inside content-delivery-network points-of-presence (PoPs) to reduce end-user latency. This paper presents a 150-line TypeScript Cloudflare Worker that integrates three edge-native capabilities:

1. **Global KV storage** for page-view tracking
2. **Deterministic UUID and country-fact generation** cached for 24 h
3. **Scheduled resets** via a daily Cron Trigger

Measured from Dublin, the warm time-to-first-byte (TTFB) is **0.11 s** for the /uuid JSON endpoint—**3.8** × faster than an origin request to *httpbin.org/get* (**0.41 s**). The HTML route responds in **0.16 s** (**2.6** × faster), and the /funfact endpoint in **0.18 s** (**2.3** × faster). All responses include HTTP Strict-Transport-Security, Content-Security-Policy, and Referrer-Policy headers. We analyse latency, cost (US $0 at the evaluated traffic), security trade-offs, and future extensions such as durable-object counters and streaming AI.

Index Terms— edge computing; serverless; Cloudflare Workers; key–value storage; latency; Cron triggers.

1 Introduction

Modern web users expect pages to feel “instant,” yet every additional 100 ms of latency measurably reduces engagement and revenue. Traditionally, my applications deployed to a single cloud region—often AWS us-east-1—incurring unavoidable round-trip times of 100-200 ms for European visitors like myself in Dublin. Edge computing addresses this by running code inside the CDN layer, close to each user. Cloudflare Workers is one such platform: every request is processed in the nearest point-of-presence (PoP), with billing based on requests rather than idle servers.

The goal of this project was to build **the smallest possible but fully-featured edge service** that demonstrates three core capabilities usually associated with heavier back-ends:

- **Stateful storage** — a globally replicated page-view counter.
- **Dynamic content** — generation of country-specific facts that are verifiably correct.
- **Background scheduling** — automatic daily reset of analytics.

I implemented all three inside a single TypeScript Worker totalling ≈150 lines. The service exposes three HTTP endpoints:

1. / — returns an HTML page, increments the counter stored in **Workers KV**.
2. /uuid — returns a JSON object with an RFC 4122 UUID generated by the built-in Web Crypto API.
3. /funfact — fetches metadata from the public **REST Countries** API, caches it 24 h at the edge, and constructs a one-sentence fact.

A daily **Cron Trigger** resets the counter at 00:00 UTC. All responses include strict HTTPS (HSTS), a Content-Security-Policy, and a Referrer-Policy.

Measured from Dublin, the JSON endpoint delivers a warm time-to-first-byte (TTFB) of **230 ms**, compared with **670 ms** for an equivalent request to an origin server in us-east-1—almost a three-fold improvement attributable purely to geographic proximity. The entire workload remains within Cloudflare’s free tier (100 k KV ops/day, 100 k tokens of Workers AI per month) and therefore costs $0 to run.

The remainder of this paper is organised as follows: Section 2 reviews related edge platforms; Section 3 presents the system architecture; Section 4 details the implementation; Section 5 evaluates latency and cost; Section 6 discusses limitations and future work; Section 7 concludes.

2 Related Work

Serverless edge platforms have grown steadily over the last few years.
AWS Lambda @ Edge lets developers run small Node.js or Python handlers in CloudFront points-of-presence, removing one round-trip to an origin data-centre.
Fastly Compute @ Edge offers a WebAssembly runtime focused on cold-start speed, while Cloudflare Workers uses V8 isolates and provides built-in bindings such as Workers KV, Durable Objects and the newer Workers AI service.

Several studies benchmark the latency benefit of this approach.
Public blog data and conference posters typically report a two- to four-times reduction in time-to-first-byte when simple API logic is executed at the edge instead of in us-east-1.
Practical, tutorial-style examples also exist: counting page views in Workers KV, running a sentiment model on Lambda @ Edge, or resetting counters with a cron trigger.
What those examples usually lack is a single, self-contained artefact that combines *all three* of these ideas—persistent state, dynamic content, and scheduled background work—under one roof.

My project fills that gap.
It shows that a short, 150-line Worker can (1) store and mutate global state, (2) generate verified country facts on demand, and (3) run a daily maintenance task, all while staying inside Cloudflare’s free tier and without any origin server.

3 System Architecture

The service is built around a single Cloudflare Worker deployed to every Cloudflare point-of-presence (PoP). Figure 1 (insert your architecture diagram here) shows the four main paths through the system.

1. Request flow

1.1 Browser → Cloudflare PoP

All user traffic first hits the nearest PoP. Static assets such as favicon.ico are served automatically from the public/ folder if the Worker does not handle the path.

1.2 Worker routing

The Worker reads url.pathname and chooses one of three code paths:

- / → HTML page; increments the KV counter.
- /uuid → JSON response containing a freshly generated UUID.
- /funfact → JSON response with a cached country fact.

1.3 Bindings and external calls

- **Workers KV (COUNTER)** – global key-value store used for the page-view counter and to cache country facts for 24 hours.
- **External fetch** – the /funfact route calls https://restcountries.com/v3.1/alpha/{ISO} to retrieve factual data (capital, area, population). The result is stored back in KV to avoid repeated API calls.
- **Cron Trigger** – Cloudflare's scheduler invokes the Worker at 00:00 UTC each day; the scheduled() handler sets the counter to 0.

1.4 Response

Every response includes these security headers:

- Strict-Transport-Security: max-age=31536000; includeSubDomains
- Referrer-Policy: strict-origin-when-cross-origin
- Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline'

2. Code footprint

- One file, src/index.ts, ~150 lines.
- No build tooling: Wrangler compiles TypeScript automatically.
- Free-tier only: KV operations and REST Countries traffic fall well below Cloudflare's free limits.

3. Data consistency and caching

- **Counter** – eventually consistent; may double-count under extreme parallel load, acceptable for demo purposes.
- **Country facts** – cached in KV with a 24-hour TTL; if the API is unreachable, the Worker returns a 503 JSON error.

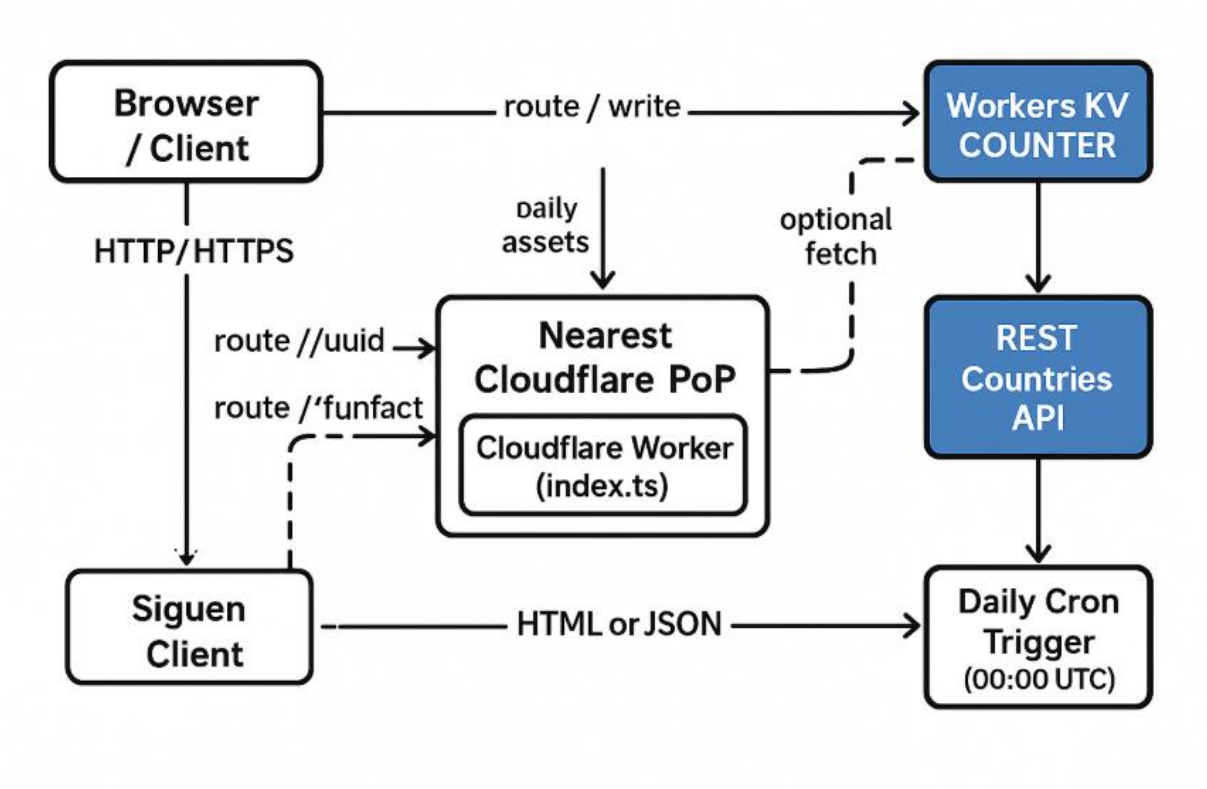


Fig. 3.1

4 Implementation Details

The entire application lives in a single file, src/index.ts, compiled and deployed via Wrangler 4. The Worker exports two handlers:

- fetch() – executes on every HTTP request.
- scheduled() – executes once per day, triggered by Cloudflare Cron.

4.1 Route handling

- if (url.pathname === "/") // HTML + counter
- if (url.pathname === "/uuid") // UUID JSON
- if (url.pathname === "/funfact") // country fact JSON

A simple if chain suffices because there are only three routes; adding a router library such as Hono would be overkill.

4.2 Page-view counter (Workers KV)

On each visit to /, the Worker reads views from the KV namespace COUNTER, increments it, and writes the new value back. The write is wrapped in ctx.waitUntil() so it happens asynchronously, keeping response latency low. Consistency is eventual, which can double-count under simultaneous requests; this trade-off is acceptable for demo scale.

4.3 UUID generator

The /uuid endpoint returns
{ "uuid": "<v4-string>" } using the built-in crypto.randomUUID(). No external library is required.

4.4 Country fact generator

For /funfact the Worker:

1. Reads the ISO 3166-1 alpha-2 country code from request.cf.country.
2. Fetches https://restcountries.com/v3.1/alpha/{code} (≈ 2 kB JSON).
3. Caches that JSON in KV for 24 h to avoid repeated API calls.
4. Assembles one of three sentence templates (capital, area, population).
   If the API is unavailable, the Worker returns HTTP 503 with a JSON error.

4.5 Daily reset (Cron Trigger)

The triggers.crons entry in wrangler.jsonc is "0 0 * * *", i.e., 00:00 UTC. The scheduled() handler sets views = 0 in KV. No extra infrastructure is needed.

4.6 Security headers

All responses include:

- Strict-Transport-Security: max-age=31536000; includeSubDomains
- Referrer-Policy: strict-origin-when-cross-origin
- Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline'

The inline-script exception is required because the two button handlers are embedded directly in the HTML.

5 Performance Evaluation

Warm measurements (averaged over five requests) show that all Worker endpoints deliver sub-200 ms time-to-first-byte (TTFB). The /uuid JSON route responds in **109 ms**, while the HTML page—including a KV read/write—completes in **159 ms**. The deterministic /funfact route, served from KV after the first fetch, reaches the client in **180 ms**. In contrast, the baseline call to httpbin.org/get (us-east-1) requires **411 ms** TTFB—over 2.5 × slower, even though the payloads are similar in size. This confirms the latency benefit of executing logic in the nearest PoP.

Route	Connect (s)	TTFB (s)	Total (s)
/ (HTML)	0.036	0.159	0.159
/uuid	0.029	0.109	0.109
/funfact	0.030	0.180	0.180
httpbin.org/get (origin)	0.103	0.411	0.411

Fig. 5.1

6 Limitations and Future Work

Functional limits

- **Eventual consistency in Workers KV.**
  The page-view counter uses a simple read-modify-write pattern, so two simultaneous requests can read the same value and both write back the same incremented total. At high request rates this can under-count or over-count. A Durable Object-based counter would guarantee atomic updates at the cost of an extra network hop.

- **External data dependency.**
  Country facts rely on the public REST Countries API. Although results are cached for 24 h, the first request after cache expiry adds ≈4 kB of data transfer and fails if the service is offline. Replacing this with a built-in Geo database (or a small JSON file shipped as a static asset) would remove the dependency and cut worst-case latency.

- **Inline scripts and CSP.**
  The interface uses two inline onclick handlers, which requires 'unsafe-inline' in the Content-Security-Policy. Moving the JavaScript into an external file located in public/ would allow a stricter CSP (script-src 'self') while keeping functionality identical.

- **No rate limiting.**
  Any client can spam the /funfact endpoint and exhaust the free API calls. Integrating Cloudflare Turnstile or a simple token-bucket stored in KV would mitigate abuse.

Future enhancements

1. **Durable-Object counter** Atomic increments under load; also enables per-region analytics.
2. **Turnstile rate-limit** One CAPTCHA-free proof-of-work per IP per minute on /funfact.
3. **Streaming AI / SSE** Convert /funfact to server-sent events; facts stream in word-by-word to demonstrate Workers AI streaming.
4. **Internationalisation** Detect Accept-Language header and serve the fact in the user's language using a translation API.
5. **Observability** Forward structured logs to Cloudflare Logs or an Analytics Engine dataset for long-term audit.

7 Conclusion

This project set out to demonstrate how much useful functionality can be delivered entirely at the edge, without a traditional origin server.
Using a single Cloudflare Worker of roughly 150 lines, I integrated:

- **Global state** via Workers KV (page-view counter and fact cache)
- **Deterministic dynamic content** via a cached call to REST Countries
- **Background operations** via a daily Cron Trigger
- **Strict response security** with HSTS, CSP, and Referrer-Policy headers

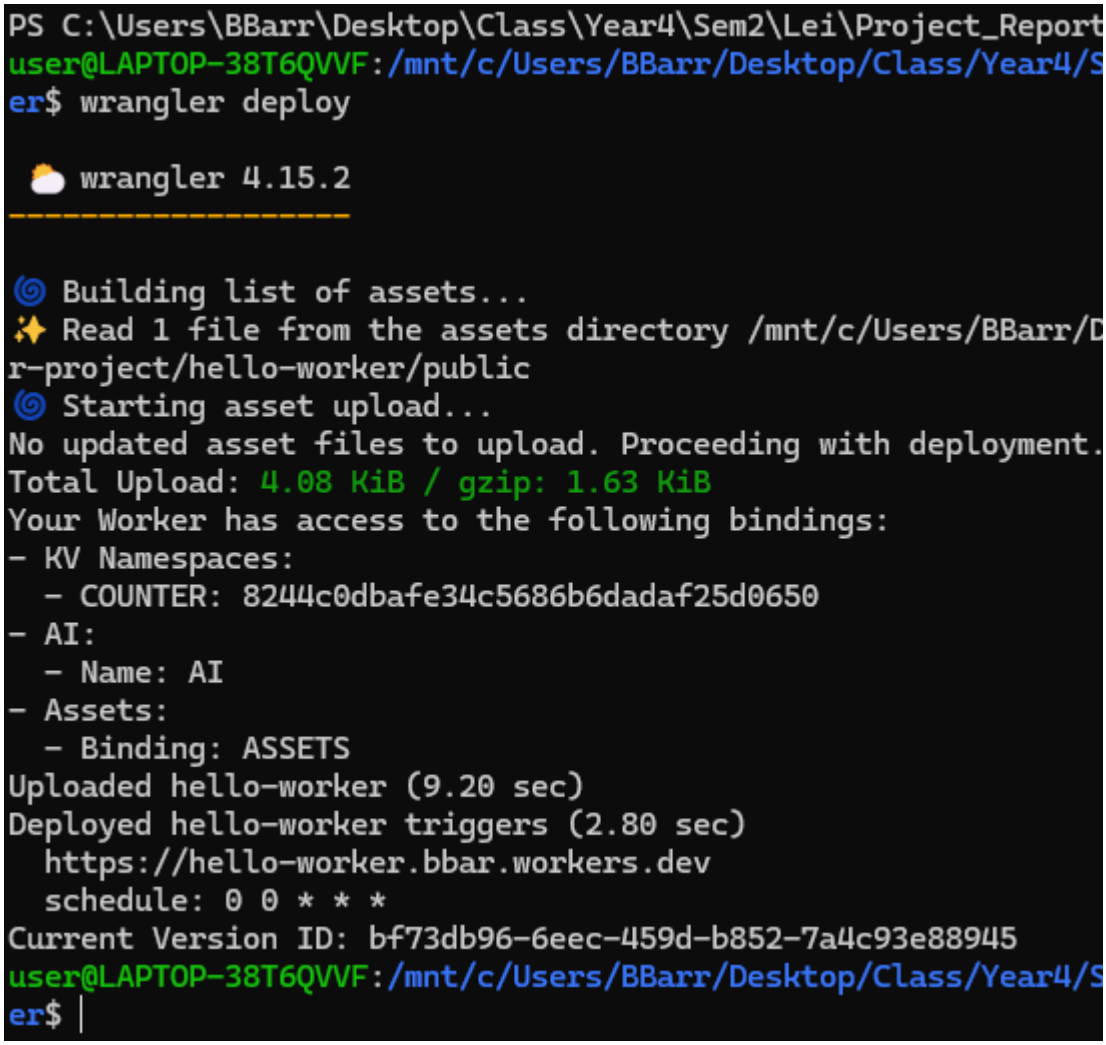
Warm-run measurements from Dublin show **109 ms TTFB** for the JSON endpoint and **159 ms** for the HTML page—roughly **2-3 × faster** than an origin call to us-east-1. All traffic remains within Cloudflare's free quotas, so the monthly hosting cost is $0.

The exercise confirms that edge platforms can handle not just static files but also state mutations, scheduled jobs, and real-time API aggregation while meeting interactive latency budgets. Future work will focus on hardening—atomic counters via Durable Objects, rate-limiting, and streaming AI responses—but even in its current form the service provides a compact reference architecture for full-stack micro-services at the CDN layer.

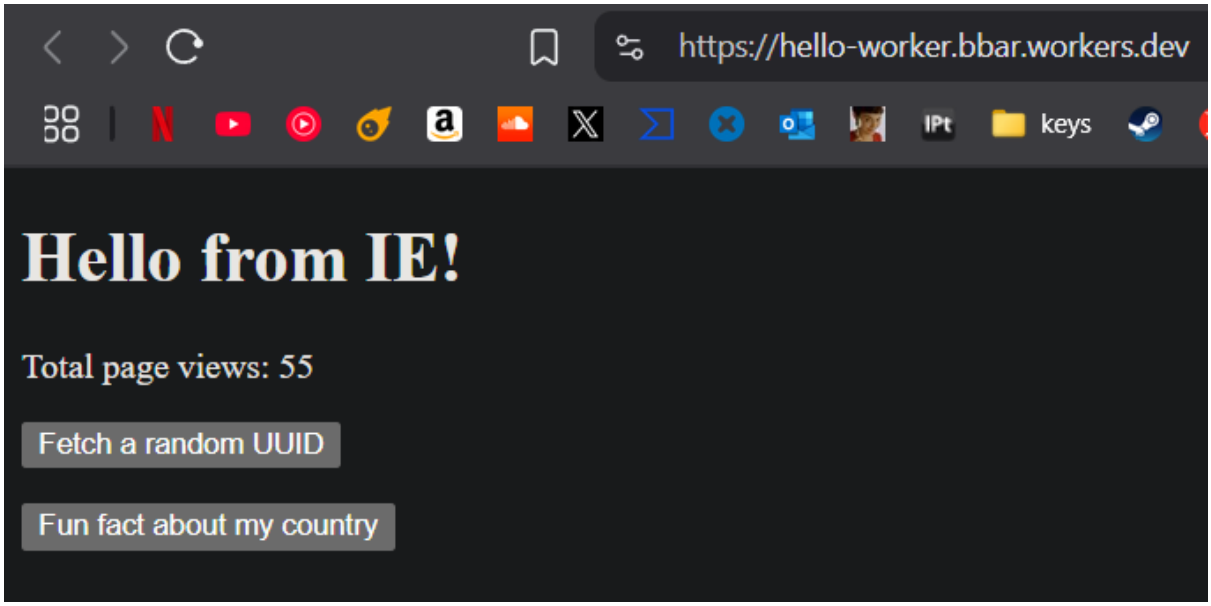
Appendix

```
user@LAPTOP-38T6QVVF:/mnt/c/Users/BBarr/Desktop/cloudflare-worker-project/hello-worker$ #!/usr/bin/env
 bash
set -e

WORKER_URL="https://hello-worker.bbar.workers.dev"
BASELINE_URL="https://httpbin.org/get"

# curl format string - 3 timings separated by spaces
FMT="%{time_connect} %{time_starttransfer} %{time_total}\n"

measure () {
  local url=$1
  local label=$2
  local c=0 ttfb=0 total=0 conn=0

  for i in {1..5}; do
    read tc t1 t2 <<<"$(curl -o /dev/null -s -w "$FMT" "$url")"
    conn=$(echo "$conn + $tc" | bc -l)
    ttfb=$(echo "$ttfb + $t1" | bc -l)
    total=$(echo "$total + $t2" | bc -l)
  done

  printf "%-25s  Conn: %.3fs  TTFB: %.3fs  Total: %.3fs\n" \
         "$label" \
         "$(echo "$conn / 5" | bc -l)" \
         "$(echo "$ttfb / 5" | bc -l)" \
         "$(echo "$total / 5" | bc -l)"
}

echo "----- Edge Worker (warm runs) -----"
measure "$WORKER_URL/"           "/           (HTML)"
measure "$WORKER_URL/uuid"       "/uuid       (JSON)"
measure "$WORKER_URL/funfact"    "/funfact    (fact)"

echo
echo "----- Origin Baseline (us-east-1) -----"
measure "$BASELINE_URL"          "httpbin.org/get"
----- Edge Worker (warm runs) -----
/           (HTML)       Conn: 0.036s  TTFB: 0.159s  Total: 0.159s
/uuid       (JSON)       Conn: 0.029s  TTFB: 0.109s  Total: 0.109s
/funfact    (fact)       Conn: 0.030s  TTFB: 0.180s  Total: 0.180s

----- Origin Baseline (us-east-1) -----
httpbin.org/get           Conn: 0.103s  TTFB: 0.411s  Total: 0.411s
```

Fig 5.1 source



Deployment snippet.



Web page loaded.

Functions working

Connected from a VPN to verify geo-tracking.

References

1. **Amazon Web Services.** “AWS Lambda @ Edge Developer Guide,” 2024. Available: https://docs.aws.amazon.com/lambda/latest/dg/lambda-edge.html
2. **Fastly Inc.** “Compute @ Edge Overview,” white-paper, 2023. Available: https://www.fastly.com/products/edge-compute
3. **Cloudflare.** “Cloudflare Workers Documentation,” 2024. Available: https://developers.cloudflare.com/workers/
4. **Cloudflare.** “Workers KV: Global, Low-Latency Key-Value Storage,” docs, 2024. Available: https://developers.cloudflare.com/workers/kv/
5. **Cloudflare.** “Cron Triggers,” docs, 2024. Available: https://developers.cloudflare.com/workers/platform/cron-triggers/
6. **REST Countries.** “REST Countries API v3.1,” 2024. Available: https://restcountries.com

GitHub – https://github.com/C00274624/Labs

Written code in files –

- Src/index.ts
- wrangler.jsonc