

30ms of Cloudflare Wizardry with Code and Coffee: TLDR;

Bob Fornal & Lwin Maung



Senior Solutions Developer II, Leading EDJE, Inc.
Passionate about learning, testing, mentoring, speaking, and personal growth.



Microsoft MVP | Managing Director, Accolade Solutions
Full stack polyglot development consultant with passion for creating IoT, mobile, web, and cloud products with a flair for user experience and user interface design.

We Are Here ...

Lwin & Bob

- Found common excitement.



Cloudflare

- Worker Scripts
- Key Vaults
- D1 Database

Super Script: Built a tool for managing ...

- Redirects
- Split-tests
- Dynamic Content

This script handles millions of Worker Script hits per day.

Our Use Case

Non-Functional Requirements ...

- **Sub-60ms** change in page responsiveness (achieved ~36ms).

Functional Requirements ...

- Maintain Cookie for source IP.
- **Redirects** moved to central location (previously, separate WP managed lists).
- **Split-Tests** (2-6 destinations, percent driven).
- **Single-Param** page manipulations.
- **Multi-Param** page manipulations (phone numbers tied to marketing campaign via URL params).
- (future) **One-Armed Bandit Split-Testing**

Outcomes ...

- Increased revenue: **\$1.5M USD/week**.
- **Faster response** to Testing Results.
- Better **campaign management**.
- **Fixes to PROD faster**, pending future code fixes.

Worker Scripts

Thoughts ...

- Think "man-in-the-middle."
- Cannot be a dynamic page (i.e. SPA: Angular, React, Vue - SSR is fine).

Findings ...

- 60-second cache.
- Connect: Routes, DB, KV, Secrets (and more).
- Free 100,000 requests/day (paid, no limit).
- Module Pattern (`import/export`, multiple files).
- `ctx.waitUntil` (extend lifetime of worker).
- Cron Jobs.

Use-cases ...

- Manipulate DOM Elements.
- Add Cookies.
- Reroute/Redirect (301 & 302 Status Codes).
- API, keep in mind cache-buster, CORS, paging.
- Access to Domain, Path, and Parameters.

Deployment

- GitHub Actions (build process).
- Connect to Github via Settings (was able to not build).

Key Vaults

Findings ...

- Low Latency, Edge Compute.
- 1,200 rw/5-minutes (have to rethink large number of writes).

Data ...

- **key** (512B)
- **value** (5MB)
- **metadata** (1kB)

Use-cases ...

- Writing using a "named route."
- Lookup `{ [domain]: [KV-name] }`.
- Use `env[kv-name].get(...)`.

D1 Databases

D1 is Cloudflare's managed, serverless database with SQLite's SQL semantics, built-in disaster recovery, and Worker and HTTP API access.

Findings ...

- 500 MB (free), 10GB (paid)
- 100 Columns/Table, Unlimited Rows

Cloudflare Architecture

| Application | Rehost | Replatform | Refactor/Build |
|---|--|---|---|
| Components <ul style="list-style-type: none">• Frontend• Data storage• Backend | | <ul style="list-style-type: none">• Optimize content.• Data at the edge. | <ul style="list-style-type: none">• Fullstack apps and programmable platforms.• Data storage with low-latency availability. |
| Infrastructure <ul style="list-style-type: none">• Servers• Delivery• Security• Observability• AI Capabilities | <ul style="list-style-type: none">• Video & image transformation• Load balancing• Global DNS resolution• Protection: bots, DDoS, & zero-day attacks• Compliance visibility | <ul style="list-style-type: none">• Dynamic traffic management• Stop threats, malicious component, API attacks, & AI bots• Extend compliance visibility | <ul style="list-style-type: none">• Serverless development platform• AI image generation at the edge• Secure privileged access to dev infrastructure• Extend visibility across workloads• AI interference at edge• Secure and moderate LLM content |

Live Demo

<https://cloudflaretalk.com>



<https://cloudflaretalk.com>

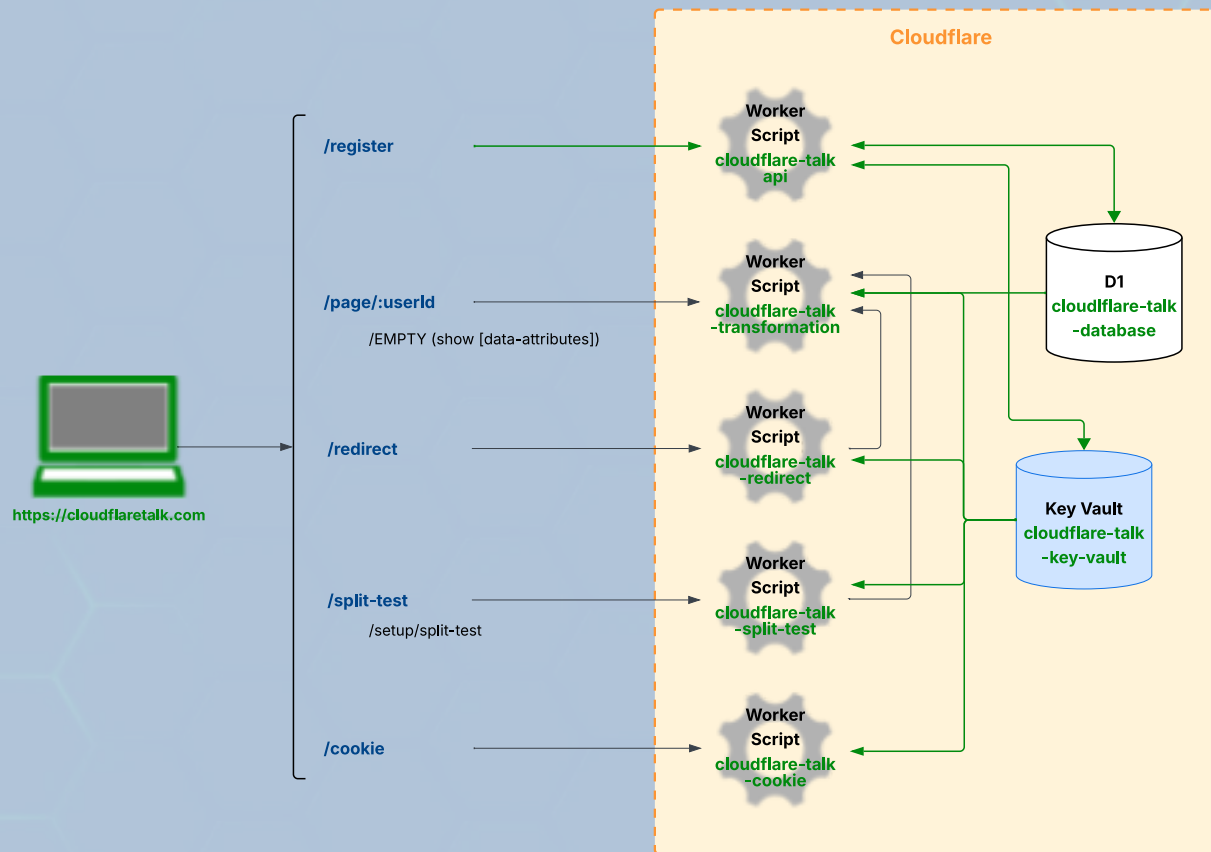
We will be examining...

1. Registering a User (uses API, KV, DB)
2. Empty Page (structure)
3. User Page (uses DB)
4. Refresh (uses API & Selection for Split Test)
5. Cookie (attached, uses DB)
6. Redirect (page to user's page, uses KV)
7. Split-Test (to selected pages, uses KV)



9 / 21





Registering a User

Step 01 ...

1. Registering a User (uses API, KV, DB)
2. Empty Page (structure)
3. User Page (uses DB)
4. Refresh (uses API & Selection for Split Test)
5. Cookie (attached, uses DB)
6. Redirect (page to user's page, uses KV)
7. Split-Test (to selected pages, uses KV)

Register


Cancel


User ID


bob-fornal


Email

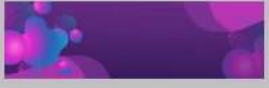
bob@code-squid.com














Banner Text

BOB'S BANNER TEXT

Save

View Empty Page

Step 02 ...

1. Registering a User (uses API, KV, DB)
2. Empty Page (structure)
3. User Page (uses DB)
4. Refresh (uses API & Selection for Split Test)
5. Cookie (attached, uses DB)
6. Redirect (page to user's page, uses KV)
7. Split-Test (to selected pages, uses KV)

Page

Back

| Data Point | Value |
|------------|---------------------------|
| User ID | [data-matcher = "userid"] |

[data-matcher = "image-path"]

[data-matcher = "header-text"]

| Data Point | Value |
|------------|-------------------------------|
| Email | [data-matcher = "email"] |
| City | [data-matcher = "city"] |
| State | [data-matcher = "state"] |
| Zipcode | [data-matcher = "zipcode"] |
| IP Address | [data-matcher = "ip-address"] |

Back

View User Page


Step 03 ...

1. Registering a User (uses API, KV, DB)
2. Empty Page (structure)
3. **User Page (uses DB)**
4. Refresh (uses API & Selection for Split Test)
5. Cookie (attached, uses DB)
6. Redirect (page to user's page, uses KV)
7. Split-Test (to selected pages, uses KV)

Page

[Back](#)

| Data Point | Value |
|------------|------------|
| User ID | bob-fornal |



BOB'S BANNER TEXT

| Data Point | Value |
|------------|---|
| Email | bob@code-squid.com |
| City | Grove City |
| State | Ohio |
| Zipcode | 43123 |
| IP Address | 2603:6011:2e00:8d8f:3c6e:f50a:9280:4c08 |

[Back](#)

List of Registered Users (Refresh)

Step 04 ...

1. Registering a User (uses API, KV, DB)
2. Empty Page (structure)
3. User Page (uses DB)
4. Refresh (uses API & Selection for Split Test)
5. Cookie (attached, uses DB)
6. Redirect (page to user's page, uses KV)
7. Split-Test (to selected pages, uses KV)

Dashboard: Cloudflare Talk

User ID: bob-fornal Page: EMPTY

Cookie Redirect Split-Test

Refresh

Once a second selection is made, the Split Test is stored.

| Select 2 | Details |
|----------|---|
| YOU! | User ID: bob-fornal IP: 2603:6011:2e00:8d8f:3c6e:f50a:9280:4c08 Banner: BOB'S BANNER TEXT |
| SELECT | User ID: talk-content-01 IP: 2603:6011:2e00:8d8f:3c6e:f50a:9280:4c08 Banner: EXCITING |
| SELECT | User ID: lwin-maung IP: 2603:6011:2e00:8d8f:3c6e:f50a:9280:4c08 Banner: COOL - LWIN! |

Refresh

Repository: <https://github.com/bob-fornal/talk--cloudflare-core>

Attached Cookie (screen)

Step 05 ...

1. Registering a User (uses API, KV, DB)
2. Empty Page (structure)
3. User Page (uses DB)
4. Refresh (uses API & Selection for Split Test)
5. Cookie (attached, uses DB)
6. Redirect (page to user's page, uses KV)
7. Split-Test (to selected pages, uses KV)

Cookie Page

Cookies can be seen using the Dev Tools
(Inspect > Application > Cookies).

Back

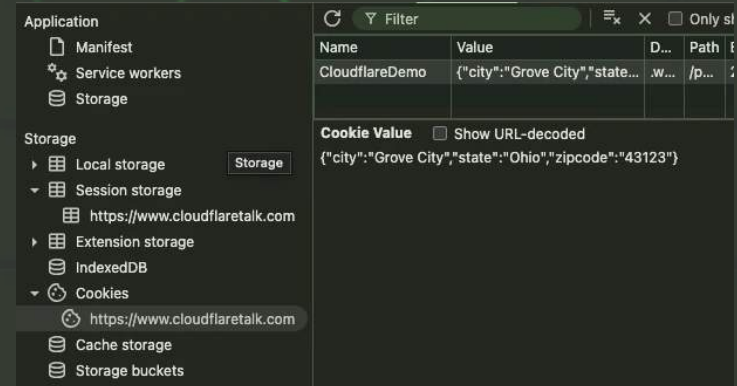
| Item | Value |
|---------|------------|
| City | Grove City |
| State | Ohio |
| Zipcode | 43123 |

Back

Attached Cookie (dev tools)

Step 05 ...

1. Registering a User (uses API, KV, DB)
2. Empty Page (structure)
3. User Page (uses DB)
4. Refresh (uses API & Selection for Split Test)
5. Cookie (attached, uses DB)
6. Redirect (page to user's page, uses KV)
7. Split-Test (to selected pages, uses KV)



Redirect

Step 06 ...

1. Registering a User (uses API, KV, DB)
2. Empty Page (structure)
3. User Page (uses DB)
4. Refresh (uses API & Selection for Split Test)
5. Cookie (attached, uses DB)
6. Redirect (page to user's page, uses KV)
7. Split-Test (to selected pages, uses KV)

Redirect Page

Issue: Redirect Not Active.

Back

Split-Test

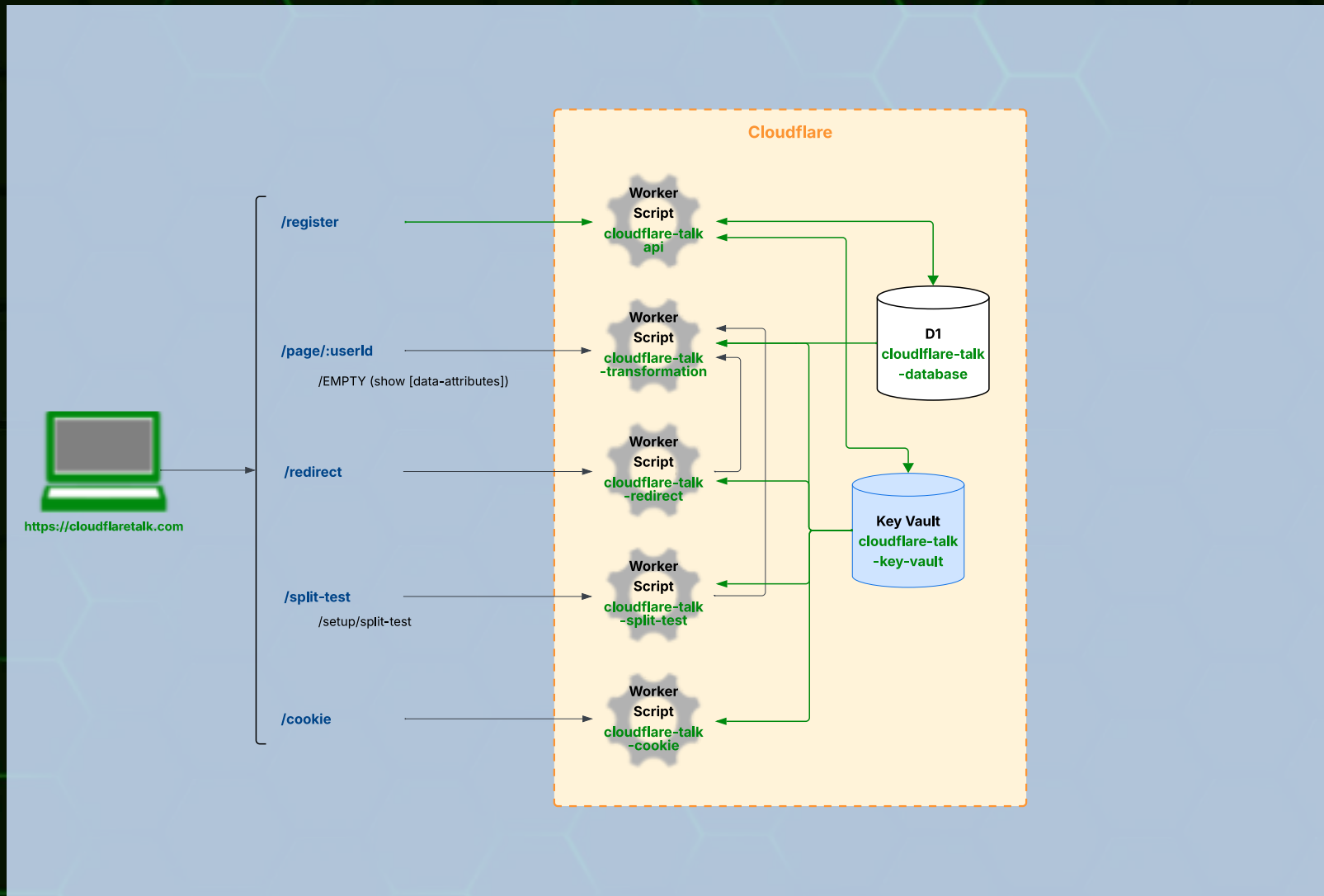
Step 07 ...

1. Registering a User (uses API, KV, DB)
2. Empty Page (structure)
3. User Page (uses DB)
4. Refresh (uses API & Selection for Split Test)
5. Cookie (attached, uses DB)
6. Redirect (page to user's page, uses KV)
7. Split-Test (to selected pages, uses KV)

Split-Test Page

Issue: Split-Test Not Active.

Back



Summary & Best Practices

Summary

- Cannot be **client-side** application.
- Manipulate DOM Elements.
- Add Cookies.
- Reroute/Redirect (**301 & 302 Status Codes**).
- **API**: keep in mind cache-buster, CORS, paging.
- Access to Domain, Path, and Parameters.
- GitHub Actions (build process) **VERSUS** Github via Settings (was able to not build).

Best Practices

- DEV Environment: Ensure it has paid access.
- Watch KV Read/Write Limits (use bulk where possible).
- Build processes add code (impacts run time).
- Cache-bust the 60-second cache.
- Use the module pattern (**import/export**)
- Use the extended worker lifetime (**ctx.waitUntil**)

