

Using Cloudflare @ Kobargo, A Case Study

When answers become cheap, the value lies in the question.

Our clients needed to share an internal redaction tool with specific teams. They didn't want to spin up new servers, roll out another VPN workflow, or expose sensitive infrastructure to the internet. So we built a redaction app that handles large, sensitive documents reliably and securely, then used Cloudflare to deliver it privately and without friction.

Workers gave us a serverless way to run this at the edge. Meaning it's close to users, fast to start, and quick to scale down when idle.

Good for performance, good for the environment.

What follows is how we met strict security, access, and operational requirements *without* building a traditional server stack.

The Requirements

The tool was meant for a defined subset of users. There was no on-prem infrastructure to host it, remote users needed secure access anyway, and exposing the app stack directly to the public internet wasn't an option. Apps with APIs, storage, and document-processing services have a big attack surface, and a basic login prompt isn't meaningful security.

Requiring username and password alone is not enough. OAuth and OIDC SSO help with identity at the application layer, but don't address network exposure or enforce access decisions before traffic hits the app. We still need to rate-limit and monitor endpoint interactions, and credential guessing can still lead to incident response conversations nobody wants to have.

The requirements boiled down to:

- Access limited to authorized users only
- Secure remote access without VPN dependency
- Identity-based controls that go beyond passwords or OAuth alone
- Edge-enforced protection against credential compromise
- Minimal operational overhead for monitoring and remediation

The Restrictions

These requirements instantly ruled out traditional hosting. No infrastructure, no client VPNs, no new hosts to manage, no public exposure of the origin. These pushed us toward Cloudflare Access Zero Trust Network Access (ZTNA).

The Solution

Cloudflare solved both access and infrastructure problems at once. We deployed the app on Workers, serverless code running across Cloudflare's global edge network, no servers to manage. Workers let us run the tool close to users with fast startup, automatic scaling, and no idle capacity burning resources. For ease of use we published it on a workers.dev URL (provided for free from Cloudflare), with the option to attach a custom domain later.

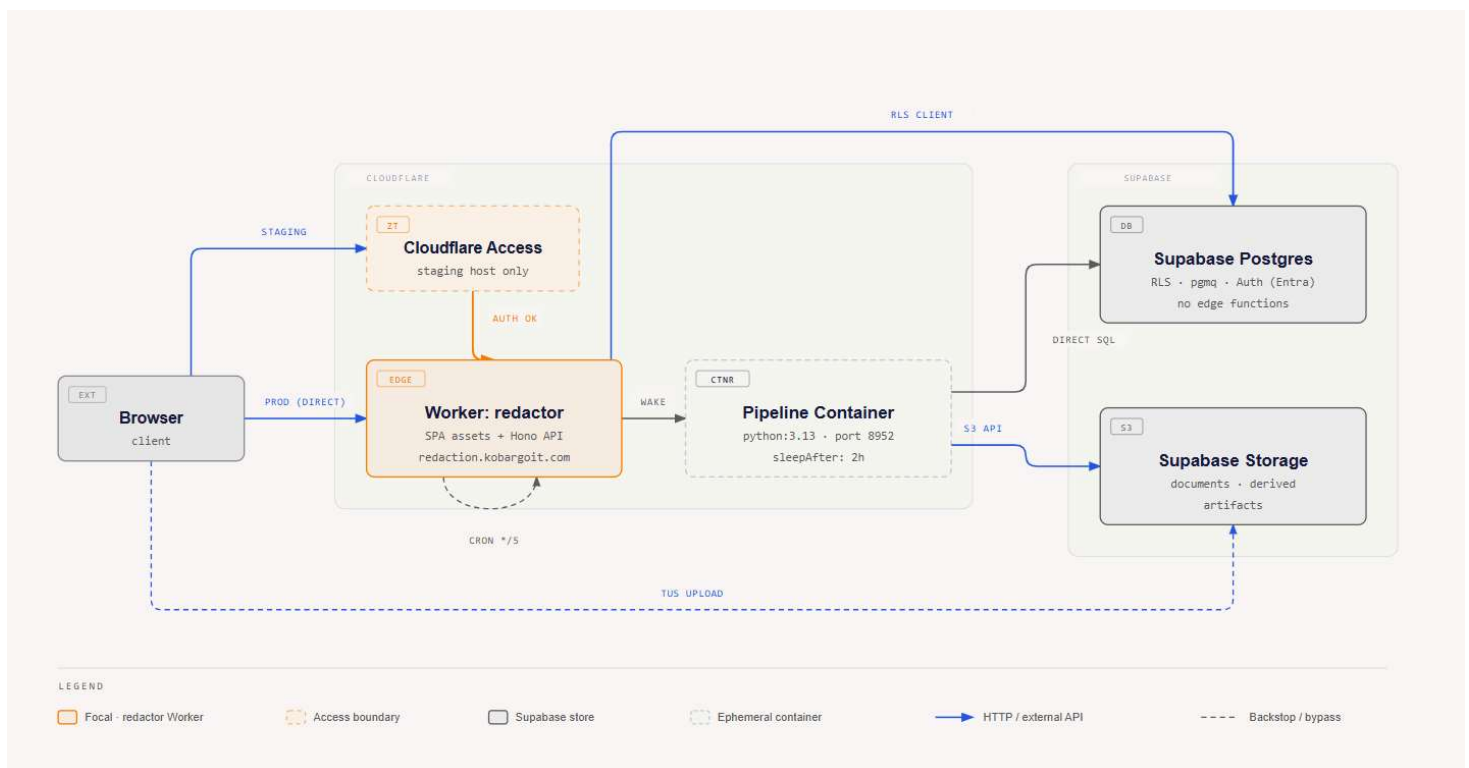
Next came access control. Cloudflare Access sits at the edge in front of the application and enforces identity-based policies before any request reaches the app. It integrates with our existing SSO/IdP via OAuth/OIDC, but crucially does this at the network edge rather than inside the app. From the user's perspective: a brief auth screen, then they're on the tool!

No extra passwords, no VPN client, no OTP emails or text messages.

The OCR pipeline, NER engine, and RegEx recognizers needed docker containers spinning up on demand for long documents. What used to require managed infrastructure or Kubernetes is now just configuring the app to run multiple instances that scale up during peak use and back down when idle. Cost tracks workload directly.

Here's how the requirements were met:

- Cloudflare Access handles identity-based access, enforced at the edge via SSO/IdP policies.
- Remote VPN-free access through browser-based SSO instead of tunnels or client VPNs.
- No new servers or public hosting. Workers and container workloads sit behind Cloudflare's edge with no directly exposed origin.
- Operational overhead dropped by consolidating access control, observability, and scaling into one platform.



Biggest Wins

Win	Who benefits	Why it matters	Cloudflare product / feature
Edge-hosted, serverless application	End users, dev team	Fast, reliable access with low latency, no servers to manage, runs close to users.	Cloudflare Workers (serverless edge compute).
Consolidated compute and network costs	Finance, ops, MSP leadership	One place to manage application, container, and network usage, simplifying cost tracking.	Workers, networking and observability tools.
Carbon-conscious, on-demand workloads	Clients, sustainability team	Compute scales with demand, reducing idle resources and associated emissions.	Workers' on-demand model and Cloudflare's cloud-based services.
Advanced analytics and tracing	Security, DevOps, support	Centralized visibility into requests, performance, and access events for faster troubleshooting.	Cloudflare dashboards, observability, Zero Trust logs.
Simple setup, deployment, management	Engineering, MSP operations	Less time spent on provisioning, patching, and VPN coordination; more time on features.	Workers deployment tooling and Access configuration.

Conclusion

Cloudflare Workers and Cloudflare Access delivered a high-trust redaction application that runs close to users, scales on demand, and stays private by default. All without adding traditional servers or VPN complexity. Identity is still central (SSO and OIDC remain part of it), but Cloudflare shifts enforcement to the edge, turning "just another login screen" into a network-level gate protecting APIs, storage, and processing workloads before traffic touches the app. For Kobargo and our clients: better performance, stronger security posture, lower operational overhead, and an architecture reusable for other sensitive internal tools beyond this one redaction platform.