

Sino Control Hub on Cloudflare Containers

The hub as one Cloudflare container behind a small Worker on your domain, with a database server outside Cloudflare

Sino Privacy Vault (<https://sinosecure.app>)

You need a database server outside Cloudflare. A Cloudflare container loses its disk whenever it restarts, so the hub cannot keep a SQLite file there, and Cloudflare's own D1 has no JDBC driver. Any Postgres, MySQL or MariaDB the container can reach over the internet with TLS works, for example Neon, Supabase, Aiven, or your own.

These steps have not yet been checked against a live Cloudflare account. Containers is a newer Cloudflare product and its configuration may change; tell us anything that differs.

1. The Database

Make a `sino_hub` database and user on your provider, with TLS required. The address is `jdbc:postgresql://<host>:5432/sino_hub?sslmode=require` (for MySQL or MariaDB, `jdbc:mysql://<host>:3306/sino_hub?sslMode=verify-full`). Allow connections from the internet, since Cloudflare's egress addresses are not fixed, and rely on the password and TLS.

2. The Worker Project

```
npm create cloudflare@latest sino-hub -- --type=hello-world --lang=ts
cd sino-hub && npm install @cloudflare/containers
```

`wrangler.jsonc`:

```
{
  "name": "sino-hub",
  "main": "src/index.ts",
  "compatibility_date": "2026-10-01",
  "containers": [{
    "class_name": "SinoHub",
    "image": "registry.example.com/sino-hub:<version>",    // the hub
image, pushed to a registry Cloudflare can pull from
    "instance_type": "standard-1",                        // at least 1
GiB of memory
    "max_instances": 1
  }],
  "durable_objects": { "bindings": [{ "class_name": "SinoHub", "name":
"HUB" }] },
  "migrations": [{ "tag": "v1", "new_sqlite_classes": ["SinoHub"] }],
  "routes": [{ "pattern": "hub.example.com", "custom_domain": true }]
}
```

src/index.ts:

```

import { Container, getContainer } from "@cloudflare/containers";

export class SinoHub extends Container<Env> {
  defaultPort = 8080;
  // Keep the hub running: its timers deliver audit events and mirror
  recovery copies.
  sleepAfter = "720h";

  constructor(ctx: DurableObjectState, env: Env) {
    super(ctx, env);
    this.envVars = {
      SINO_HUB_MASTER_KEY: env.SINO_HUB_MASTER_KEY,
      SINO_HUB_DATABASE_URL: env.SINO_HUB_DATABASE_URL,
      SINO_HUB_DATABASE_USER: env.SINO_HUB_DATABASE_USER,
      SINO_HUB_DATABASE_PASSWORD: env.SINO_HUB_DATABASE_PASSWORD,
      SINO_HUB_TRUST_PROXY: "true",
    };
  }
}

export default {
  async fetch(request: Request, env: Env): Promise<Response> {
    // One hub: every request goes to the same container. The client's
    address is passed on for rate limits,
    // replacing anything the client sent.
    const headers = new Headers(request.headers);
    headers.set("X-Forwarded-For", request.headers.get("CF-Connecting-
IP") ?? "");
    return getContainer(env.HUB, "hub").fetch(new Request(request, {
      headers }));
  },
};

```

3. Secrets and Deploy

```
openssl rand -base64 32 | npx wrangler secret put SINO_HUB_MASTER_KEY
npx wrangler secret put SINO_HUB_DATABASE_URL          #
jdbc:postgresql://<host>:5432/sino_hub?sslmode=require
npx wrangler secret put SINO_HUB_DATABASE_USER
npx wrangler secret put SINO_HUB_DATABASE_PASSWORD
npx wrangler deploy
```

Keep a copy of the master key outside Cloudflare, because a Worker secret cannot be read back. Without the key the database cannot be opened.

4. Claim the Hub

Open `https://hub.example.com` once to start the container. Then find the setup code in the container's logs, in the Cloudflare dashboard under **Workers & Pages** → **sino-hub** → **Containers**, or with `npx wrangler tail`. Claim the hub with your passkey and sign its certificate in the Sino app (*Deployment Guide*, section 3, step 5). This address is permanent: every device keeps it.

5. Things to Know

- A container can be restarted at any time, for example by Cloudflare's maintenance. The hub keeps everything in its database, so a restart costs a few seconds. Until an owner claims the hub, each start prints a new setup code.
- `max_instances: 1` and the fixed name `"hub"` keep a single hub. Don't route by location.
- A PC's streamed answer (a line every 20 seconds, for up to 3 minutes) should pass through the Worker as it comes. If something on the way holds it back, PCs notice within 10 seconds and use held waits of 25 seconds instead, by themselves.
- `GET /healthz` answers 200 while the hub's database does, for any monitor you point at the hub.