Cloudflare OS: Setup and Connector Runbook

Companion to "We Deployed Cloudflare OS in Week One - The Real Read". Everything here was verified against the 5 August 2026 release by running it, not by reading about it. Where a fact comes from Cloudflare's own documentation it is attributed; where it comes from our setup log it is marked as ours.

1. What you are agreeing to

Both repositories - the core platform and the starter deployment - carry a plain Apache-2.0 licence with no additional rider. We read the LICENSE files rather than the announcement. There is no Commons Clause, no source-available restriction and no field-of-use limit, so forking, rebranding, self-hosting and commercially hosting it for a client are all permitted with nobody to ask.

The one live constraint is the standard trademark clause. You may say what your deployment is built on. You may not name your product after Cloudflare's, use their marks as your own, or imply endorsement.

> **Contribution is closed - this is the planning fact people miss**
> Cloudflare states it is not seeking outside code contribution. Small, trivially verified fixes are accepted; anything past roughly a dozen lines is closed with a pointer to that guideline. A connector of any real size belongs in your own repository. For larger proposals the open channel is a GitHub discussion, not a pull request.

2. Platform maturity - check before you commit

- Dynamic Workers: open beta since March 2026. No announced GA date.
- Durable Object Facets: open beta since April 2026. No announced GA date.
- The repository describes the current release as early access with "many rough edges".
- A fully managed dashboard version is stated as coming, with no date. Treat deployment labour as a depreciating asset and put effort into context, policy and connectors instead.

If your organisation's change-control process treats a beta dependency as a blocker, resolve that before a pilot rather than after. The app-building layer depends on both beta primitives.

3. Running it locally (about twenty minutes)

```
git clone https://github.com/cloudflare/cloudflare-os.git
cd cloudflare-os
pnpm run-local
# then open http://localhost:8787
```

One command installs, builds and serves the whole stack on wrangler and workerd. It hashes every tracked source file and skips straight to serving when nothing has changed, so repeat runs are fast.

- Ours ran on Node 22 without complaint, despite a newer version being suggested elsewhere in the project.
- First run downloads a pinned pnpm through corepack; allow a few minutes on a cold cache.
- Self-hosted deployments have unlimited AI usage by default and built-in username/password accounts. Nothing needs configuring to try it.

4. Model access and what it costs

Inference is routed through AI Gateway and the model is yours to choose. Cloudflare passes provider inference pricing through without markup, so in practice you bring your own provider key and pay the provider. Gateway core features are free; credits bought through unified billing carry a small fee.

- **`Required:`** Gateway mode requires an account ID and an API token with AI Gateway Run and Read permissions.
- The daily free allowance and credit top-up flow is opt-in and exists for running a public multi-user service. Leave it off for internal deployments.

> **The cost line that actually bites**
> Cloudflare's own charges for a small internal deployment are trivial - a paid Workers plan floor plus request and storage charges. The variable risk sits in Durable Object duration and per-app database writes, and above all in model tokens, which are billed by your provider and are invisible to any Cloudflare-side budget alert. Set a provider-side spend cap on day one. Note also that per-load billing for Dynamic Workers is currently waived during beta and will switch on; model your costs as though it already has.

5. Deploying to an account

- A paid Workers plan is required - Dynamic Workers is not on the free tier. There is no free-tier client demo beyond running it locally.
- The starter repository pins a release and holds your branding, sign-in, integrations, routes and upgrades.
- Account prerequisites span Workers, KV, R2 and Browser Rendering in addition to Durable Objects.
- Sign-in can be delegated to Cloudflare Access, which is the cheapest path if internal tools are already gated that way.

6. Writing a Gatekeeper - the shape of the work

A Gatekeeper is a Worker that mediates all access between an agent and one external service. It carries seven responsibilities: authorisation, a capability-shaped API, fine-grained resource granting, logging and approvals, caching, simulation of pending writes, and observer verification. The first three are enough for a working connector; the rest are a deliberate second pass.

The single most important decision is the API surface, because it is what the agent reads as documentation and what every later choice hangs off. Design one interface per logical resource rather than a god-object that takes an id on every call, and settle it before writing the implementation.

- Reads must authorise an observation before returning data to the caller.
- Writes must be submitted as actions and must not touch the service until approved.
- Simulate pending writes so later reads reflect them. This is what lets an agent keep working instead of stalling on each approval, and it is why users stop reflexively approving everything.

7. Observer verification - the step most integrations skip

When an app is shared, a colleague may see data the agent previously read. The framework makes you decide who is allowed to. The methods are mandatory; the code will not type-check without them. Pick a strategy per binding:

- Private only - always refuse. For resources too sensitive to share and with no per-person access oracle.
- Single-unit ACL check - the binding is one atomic resource; confirm the observer can read it. The common case.
- Data-set tracking - the binding spans sub-resources with different permissions and there is an oracle for each.
- Low stakes - no-ops, for personal services where any collaborator may observe.

Use data-set tracking only when both conditions hold: the binding really does span differently permissioned sub-resources, and you really can check each one against a given person. If a single permission covers everything, the single-unit check is correct and far simpler. Always run the check against the observer's own credentials, never the owner's.

8. Eight setup details that cost us time

- The development server auto-discovers connector packages by directory-name prefix and generates their configuration - which supersedes a manual wiring step still described in the project's authoring guide.
- Discovery happens at startup, so a newly added connector needs a server restart, not a hot reload.
- One type file in a connector must be a symlink rather than a copy; a copy fails in a way that does not obviously say so.
- The generated Worker type definitions file is large and machine-generated - generate it, never hand-edit it.
- The RPC validator skips files outside the package directory and says so; that warning is expected, not an error.
- Return an empty list from the auto-approvable-actions method unless you are certain a write is benign. It is the honest default for anything touching customer records.
- Duplicate the approval-queue handle before storing it in a session; parameter stubs are disposed when the call returns.
- Give created-but-unapproved records a clearly provisional identifier and document it, so an agent never shows one to a person as though it were real.

9. Connector API shapes worth knowing in advance

Every service has two or three quirks that consume most of the integration time. From writing a connector for a self-hosted CRM, the pattern generalises - look for these three before you start:

- Response envelopes. Payloads often nest under a key named for the operation rather than the path. Write one unwrapping helper with fallbacks instead of guessing per call.
- Composite fields. Names, emails, links, phone numbers and money are frequently objects, not scalars - and filters usually address the leaf while writes address the parent. Money is often stored in millionths.
- Join tables. Attaching a note or task to a record is commonly two calls, and reading them back needs an explicit depth parameter or the linked record does not ride along.

Also check the page-size ceiling. Asking for more than the maximum is often a hard error rather than a

silent clamp.

10. A short decision checklist

- Can we accept two open-beta primitives with no GA date in this system's change-control class?
- Do we have a paid Workers plan, and a provider-side spend cap on model tokens?
- Which systems do our agents need that have no connector, and who writes them - knowing it will not be upstreamed?
- For each connector, which reads are observations and which writes are actions?
- For each binding, which observer strategy applies, and is there an oracle to support it?
- Is the knowledge these agents will reason over organised, current and owned by us?

The last question is the one that decides whether any of this works, and it is the one the software cannot answer for you.

About this runbook

Written by Emerge Digital from a first-week deployment of Cloudflare OS and a connector built against it. Emerge Digital is not affiliated with or endorsed by Cloudflare, and holds no Cloudflare partner status. Product names are used only to say what the software is. Platform details reflect the 5 August 2026 release and the state of the beta primitives on that date - re-check them before committing to a plan.