

■ // CLOUDFLARE OS • SETUP-ÜBERSICHT

Cloudflare OS in deinem eigenen Konto

Cloudflare hat sein internes KI-Betriebssystem als Open Source veröffentlicht. Diese Übersicht bringt dich in vier Schritten zum eigenen Deployment — mit beiden Deploy-Wegen, allen Voraussetzungen und den Prüfpunkten nach dem ersten Start.

2 DEPLOY-WEGE

4 SCHRITTE

VORAUSSETZUNGEN

OHNE API-TOKEN

EARLY-ACCESS-HINWEISE

Worum es geht: Cloudflare OS ist der KI-Workspace, den Cloudflare intern einsetzt — nach eigener Angabe arbeiten tausende Mitarbeiter täglich damit. Agenten recherchieren dort mit deinem Firmenwissen, schreiben Dokumente und bauen kleine Apps. Seit dem 26. August 2026 liegt der komplette Quellcode auf GitHub, und du kannst das System **in dein eigenes Cloudflare-Konto** deployen.

1 Die zwei Deploy-Wege

Weg A — 1-Klick-Deploy (*os.cloudflare.app/deploy*): deployt die aktuelle Version ohne lokalen Build in dein Konto und richtet Login samt Admin-E-Mails im Flow ein. Die App läuft danach auf einer workers.dev-Adresse. Site-Name, Logo, Akzentfarbe, Ansagen und Connectoren stellst du unter **/admin** um, ein Redeploy ist dafür nicht nötig.

Weg B — Starter-Repo (*github.com/cloudflare/cloudflare-os-starter*): für die eigene Domain, gepinnte Releases, eigene Integrationen (sog. Gatekeeper) und den E-Mail-Gatekeeper, der eine eigene Zone voraussetzt. Der Deploy umfasst sechs Worker; nur der Router bekommt eine öffentliche Route, alle anderen sind ausschließlich über Service-Bindings erreichbar.

VORHER ENTSCHIEDEN

Der Wechsel von A nach B ist ein manueller Migrationsfall: Worker-Namen, Storage-IDs, öffentliche URL und AI Gateway überträgst du einzeln, und jeder vergessene Wert schlägt still fehl. Wenn du produktiv planst, starte direkt mit Weg B.

2 Voraussetzungen für Weg B

- **Cloudflare-Konto** mit Workers, KV, R2, Browser Rendering und Dynamic Worker Loaders. Workers AI und AI Gateway kommen dazu, denn darüber läuft der Standard-Modellkatalog — **ohne API-Token**. Artifacts ist optional.
- **Lokal:** Node.js 24.19 oder neuer und pnpm 11.17.
- **Für die eigene Domain:** eine aktive Zone plus eine self-hosted Access-Applikation samt Audience-Tag. Zum reinen Ausprobieren geht auch hier eine workers.dev-Route.

3 Die vier Schritte

Schritt 1 — klonen und anmelden:

```
git clone --recurse-submodules https://github.com/cloudflare/cloudflare-os-starter
cd cloudflare-os-starter
pnpm install
pnpm --dir cloudflare-os install
pnpm exec wrangler login
```

Schritt 2 — *deployment.json* ausfüllen: Account-ID, Worker-Namen, Hostname, Access-Audience und Admin-E-Mails. Jede Option ist in der Datei direkt kommentiert. Ohne eigene Domain stellst du die Route auf `{ "workersDev": true }` und trägst die entstehende Adresse als *publicBaseUrl* ein.

Schritt 3 — prüfen und deployen:

```
pnpm check
pnpm deploy
```

Bleiben die Resource-Felder auf *null*, legt Wrangler die drei KV-Namespaces und den R2-Bucket automatisch an und verbindet sie bei späteren Deploys wieder.

Schritt 4 — */admin* öffnen: Site-Name, Logo und Akzentfarbe setzen. Branding braucht keinen Redeploy.

4 Nach dem Deploy prüfen

- 1 Hostname öffnen und bestätigen, dass der Access-Login mit der erwarteten Identität durchläuft — und dass dies die einzige öffentliche Route ist.
- 2 Unter **/admin** den Admin-Status kontrollieren und Context, Scheduler und Custom Gatekeeper auf disabled, optional oder enabled stellen.
- 3 Einen Agenten etwas für „in fünf Minuten“ planen lassen und abwarten, ob es ausgeführt wird — das testet den Scheduler von Ende zu Ende.
- 4 Die Logs von Router, Workshop und Error Reporter per *wrangler tail* gegenlesen.

5 Early Access — die Spielregeln

Cloudflare stuft das System selbst als Early-Access-Software ein. Für den Produktiv-Einsatz heißt das konkret:

- Upstream-Releases **pinnen** und Änderungen vor jedem Upgrade lesen — das Starter-Repo ist genau dafür gebaut.
- Die Trust Boundary vor jedem Produktiv-Upgrade verifizieren.
- Falls etwas schiefgeht: Rollback über die Deployment-History im Dashboard oder per *wrangler rollback*.

■ // AUF EINEN BLICK

Die Kurzfassung

Fünf Punkte, die das ganze Setup zusammenhalten.

- ✓ Zum Ausprobieren reicht der **1-Klick-Deploy** auf einer workers.dev-Adresse
- ✓ Produktiv gehst du über das **Starter-Repo**: eigene Domain, gepinnte Releases, eigene Gatekeeper
- ✓ Konto braucht Workers, KV, R2, Browser Rendering + Dynamic Worker Loaders; lokal **Node 24.19+ und pnpm 11.17**
- ✓ Der Modellkatalog läuft über **Workers AI + AI Gateway, ohne API-Token**
- ✓ Branding (Name, Logo, Farbe) änderst du unter **/admin**, ganz ohne Redeploy

■ // BLEIB DRAN

Open-Source-Releases, getestet statt gehypt.

Ich schaue mir solche Veröffentlichungen an, bevor ich sie empfehle — mit den echten Voraussetzungen und Grenzen. Folg mir für die nächste Einordnung.

@ki.borish

Fakten verifiziert am 27.08.2026 aus blog.cloudflare.com/cloudflare-os sowie den READMEs von [cloudflare/cloudflare-os](https://github.com/cloudflare/cloudflare-os) und [cloudflare/cloudflare-os-starter](https://github.com/cloudflare/cloudflare-os-starter). Cloudflare OS ist Early-Access-Software; Befehle und Optionen können sich mit neuen Releases ändern. Prüfe vor einem Produktiv-Deployment die aktuelle Dokumentation im Repo.