ADVENTIVE PLATFORM ENGINEERING

Admin Dashboard Cloudflare Migration

Replacing legacy `adventive-admin` with a Cloudflare Workers API + Static Assets UI

Cloudflare Access · JumpCloud SAML · Hyperdrive · Hono · React + Vite

---

DOCUMENT	VERSION	DATE
Migration Plan	1.1	2026-04-30
OWNER	**STATUS**	**CONFIDENTIALITY**
Jeffrey Lambert	Phase 1 prototype shipped · Phase 2 in progress	Internal

00 — Context

Problem statement

Adventive's internal administrative interface is a single legacy application (`adventive-admin`, BitBucket-hosted, version 3.3.9) that handles two distinct responsibility domains:

1. **Customer service-level management** — operator tooling for managing customer accounts, tiers, entitlements, and configuration.
2. **Billing** — invoice generation, line-item rollup, Stripe invoice creation via API, custom invoice PDF rendering, and email delivery via Mailgun.

The application mixes API logic with presentation in a single codebase, does not reflect current Adventive brand guidelines, and has accumulated pain points typical of a long-lived internal tool. It is also the coupling point for a custom billing flow that the Stripe Billing Transition project will evaluate for replacement.

Repo analysis (April 2026) surfaced two findings that raise the urgency of migration beyond architectural debt alone:

- **Production AWS and Bitbucket credentials are hardcoded in source** (`application/config/adventive.php`) — a live secret-exposure risk in every clone and CI artifact.
- **CSRF protection is globally disabled** and **Duo 2FA is currently bypassed** — the admin's only active access control is JumpCloud username/password.

Business motivation

- **Architectural debt:** the admin is the last major internal surface not yet aligned with Adventive's API-first / Cloudflare Workers direction (Public API migration landed 2026-04; Cloudflare Tunnel landed prior).
- **Secret exposure:** AWS access keys, S3 secrets, and Bitbucket OAuth credentials are committed to source for all three environments. Remediation is blocked until secrets are moved to a managed store; this migration provides the natural forcing function.
- **Security posture:** CSRF disabled (`config.php:320`), Duo 2FA bypassed (latest auth commit), payment links secured only by MD5 hash (`Billing.php:561`). Cloudflare Access restores edge-enforced MFA and eliminates application-managed auth entirely.
- **Presentation coupling:** mixing presentation and API logic makes it hard to evolve either independently; a future mobile or partner-facing admin surface is effectively blocked.
- **Brand drift:** the visual layer predates the current brand system and reads as legacy to internal users.
- **Billing entanglement:** the admin owns both the billing UI and meaningful billing business logic. That logic is the scope-of-analysis input for the sibling Stripe Billing Transition project — separating presentation from API is a prerequisite for cleanly migrating billing.
- **Dependency risk:** `phpoffice/phpexcel` 1.7.9 (archived 2019, no security patches) is installed alongside its successor `phpspreadsheet`. CodeIgniter 3.1.* receives no active maintenance.

Scope

- **In scope:**
- Architecture audit of `adventive-admin` (routes, data model, integrations, auth). ✓ *Complete — see §01.*
- Serverless viability assessment for the API layer (Cloudflare Workers fit). ✓ *Complete — see §01.*
- Target architecture: separate API Worker + Cloudflare Pages SPA. ✓ *Decided.*
- UI refresh aligned with current Adventive brand guidelines.
- Auth/authz model: Cloudflare Access for operator identity, application RBAC for authorization. ✓ *Decided.*
- Migration plan with phased cutover and rollback. ✓ *See §02.*
- Documentation: full planning deliverable set → PDF → Claude Code handoff package.

- **Out of scope:**

- Billing business logic redesign — owned by `../Stripe Billing Transition/`. This project defines the *seam*; that project decides what lives on which side of it.
- Public-facing customer UI.
- Migration of the underlying customer / account / billing data stores (unless assessment surfaces a forced dependency — it did not).
- Partner-specific scheduled reports (`Reporting.php` , 25 methods) — deferred; evaluated for a dedicated analytics service in Phase 4.

- **Deferred:**

- Mobile or offline operator access.
- External partner/reseller admin surfaces.
- Feature additions beyond parity with the current admin.
- Directory provider choice for Cloudflare Access (JumpCloud vs. Google Workspace) — deferred per ADR.

Success criteria

1. Written and signed architecture plan matching the Adventive Engineering PDF standard. ✓ *This document.*
2. Clear API contract definition (OpenAPI via `@hono/zod-openapi`) that billing, service-level management, and any future consumer can build against.
3. Presentation layer demonstrably decoupled — can be rebuilt or replaced without touching the API.
4. Brand-guideline review sign-off on the new UI direction (reference: brandguide.adventive.com/dashboard).
5. Cutover plan with rollback path and defined success metrics (operator task completion time, error rate parity or better).
6. Implementation handoff package ready for Claude Code (CLAUDE.md, PLAN.md, kickoff doc — matching Public API pattern).

7. Production secrets removed from source and moved to AWS Secrets Manager or Cloudflare secrets before implementation begins.

Stakeholders

Role	Name	Interest
Owner	Jeffrey Lambert	Architecture, plan quality, delivery
Approver — UI	Jeffrey Lambert	Brand alignment, UX sign-off
Approver — API	Patrick	API contract, backend architecture sign-off
Approver — Billing UI	Jeffrey + Patrick	Joint sign-off on billing surface area of the admin
Informed	Adventive operations team (~10 operators)	End users of the new admin

Operator population and RBAC tiers

~10 operators total, across three permission tiers:

Tier	Count	Capabilities
Super-admin	2	All functionality across customer, service-level, billing, and tooling domains
Billing	TBD	Billing-scoped operations (invoices, payments, managed services, reports). Read access to accounts.
Read-only	TBD	Read access across all surfaces, no mutations

The two super-admins are the only tier with authority to destructive / cross-domain operations. Billing-tier operators can read all customer context but only write billing-domain records.

Design reference

Visual and interaction language target: **https://brandguide.adventive.com/dashboard** — the Adventive brand-guide dashboard concept. "Similar in design, refined for utility." Not a literal reskin; the admin's job is operator efficiency first.

Project sequencing

This project runs **first**, ahead of the Stripe Billing Transition migration phase. Rationale:

- The presentation/API separation is a prerequisite for cleanly migrating billing — you can't cleanly replace a subsystem that shares a codebase with its UI.
- The new admin will surface whatever billing decision gets made downstream (custom, Stripe-native, or hybrid), so it needs to be designed to tolerate that outcome without rework.
- Admin-first also front-loads the architectural work that the billing project will consume.
- The secret-exposure and auth-gap findings make this more urgent, not less.

The billing project can kick off in parallel at its viability-assessment phase, but its migration-plan phase should land after the admin API contract stabilizes.

Update — 2026-04-30: Phase 1 prototype shipped

A visually functional UI prototype is now in place at `~/Repositories/GitHub/Adventive/adventive-admin-ui/` and deploys as a Worker (Static Assets) to **`genesis-admin.adventive.dev`** for team review. The prototype establishes the brand language, navigation IA, and operator workflows the team will validate against; it runs entirely against in-memory mocks (no backend wiring) and is not yet behind Cloudflare Access.

What it covers (15 routes):

- **Dashboard** — operator landing with revenue KPIs, 12-month stacked-area trend, recent invoices, managed-services queue, quick links.
- **Accounts** — list with status filters and search; detail page with all 9 tabs (Settings, Users, Campaigns, Ad Units, Advertisers, Billing History, Permissions & Entitlements, Plan Info, Managed Services).
- **Billing** — Overview (KPIs + trend/breakdown chart toggle + collections progress + aging snapshot), Invoices, Account Revenue History (12-month per-account grid), Processing controls, Delinquent, Aging Summary.
- **Tools** — Preview Link Lookup, User Lookup, Ad Type Utilization, Ad Type Performance, Cognito Sync, Special Permissions catalog.

Stack chosen and proven: Vite 8 + React 19 + TypeScript + Tailwind 3 + shadcn-flavored primitives + TanStack Router (code-based) + TanStack Query + Recharts. Brand tokens are HSL CSS variables lifted directly from `brandguide.adventive.com/dashboard` (dark theme, primary purple `250 100% 70%`, radius `0.75rem`, Stripe-style accent palette, Inter font).

The prototype's mock data layer at `src/lib/mock/` is deliberately structured to mirror the API contract this project will define — replacing it with TanStack Query hooks against `adventive-admin-api` is the primary handoff between Phase 1 and Phase 2.

Update — 2026-04-30: Phase 2 locked decisions

Three open API/architecture questions were closed during the Phase 1 hand-off conversation and are now locked:

1. **API surface shape — hybrid REST + composite endpoints.** Strict REST per resource (`GET /accounts` , `/campaigns` , `/invoices` , `/users` , etc.) plus a small set of composite endpoints for heavy screens. The canonical example: `GET /accounts/:id?include=users,campaigns,adunits,advertisers,invoices,managedjobs,permissions,plan` returns the full Account Detail screen in one round-trip. Avoids the 9-call waterfall a strict-REST design would force on the legacy `/account/detail` tabbed page without over-fitting the API to the UI.
2. **Auth posture — Cloudflare Access JWT + Worker session for CSRF/idempotency.** Cloudflare Access at the edge handles authn (JumpCloud SAML + WebAuthn replaces the legacy LDAP + Duo flow entirely; the JumpCloud `Admin Dashboard` group becomes the Access policy). The Worker layers a lightweight signed-cookie session on top — KV-backed — that issues CSRF tokens for mutations, idempotency keys for billing actions, and seeds an audit trail. Defense-in-depth that matters specifically for the billing/processing surface.
3. **Billing scope — read-only in v1, mutations deferred to Stripe Billing Transition.** Read-only billing endpoints (overview, invoice list, account history, aging, delinquent) ship in admin-api Phase 2 so the prototype becomes review-able end-to-end. Mutating billing endpoints (test invoice generation, QuickBooks export, dunning sweep, year-end rollover) stay mocked in the UI until the Stripe Billing Transition project provides their replacement. Documented as a deliberate handoff to that project.

These three decisions, plus the existing ADRs for Cloudflare Access, the React SPA stack, and the no-CodeIgniter-retrofit deployment strategy, fully constrain the Phase 2 implementation. The `adventive-admin-api` Worker scaffold at `~/Repositories/GitHub/Adventive/adventive-admin-api/` reflects them in code.

Update — 2026-04-30: Phase 2 prerequisites surfaced by deeper analysis

A deep code analysis pass (Phase 2 kickoff) added findings beyond §01:

- **Auth replacement is cleaner than originally assumed.** Legacy uses JumpCloud LDAP bind + `Admin Dashboard` group check + Duo Universal Prompt 2FA. Cloudflare Access fully replaces all three layers with a single JumpCloud SAML/SCIM connector and WebAuthn second-factor. The new admin-api never ships with `Auth.php` , `Jc_auth` library, or the Duo SDK — entire surface area removed.
- **Four-database topology, not one.** Beyond `console` (Hyperdrive `DB_CONSOLE` already provisioned at `059838c4abb64a92a4aece2a6a533a29` for dev), the legacy admin reads from `billing` (invoicing/payments), `aggregate_ro` (KPI event metrics), and `data_warehouse` (Redshift, used only by Report.php). Phase 2 dev needs only `DB_CONSOLE` . Stg/prd planning must provision additional Hyperdrives for `billing` and `aggregate_ro` — captured in §03's environment matrix.
- **20+ snowflake customer reports do not migrate as-is.** `Report.php` (~5200 LOC) contains custom SQL for Sonoma, HighSnobiety, Pulpo, NPM, FarmJournal, Afar, Bridgetower, DoubleVerify, Plugshare, Atlantic, Boston Globe, SF Gate, etc. Each is a one-off CSV export with 3–5 join queries.

Recommendation: move to versioned templates or a BI tool; do not port to admin-api. Deferred to a Phase 4 analytics-service evaluation.

- **External microservice coupling.** Three internal HTTP services (`billing_service`, `account_service`, `asset_service`) are called from PHP via raw curl with no timeout, retry, or circuit-breaker. Phase 2 wraps these with proper resilience (timeout + exponential backoff + bulkhead). The `billing_service` boundary is co-owned with the Stripe Billing Transition project.
- **Hardcoded production secrets in `application/config/adventive.php`** — full inventory now visible: AWS IAM key/secret, Mailgun API key, Duo secret key, JumpCloud bind password, Bitbucket OAuth token. All require rotation at migration cutover; remediation is a hard prerequisite, not a Phase 2 task. The new admin-api has no equivalent secrets — Cloudflare Access replaces auth credentials, Hyperdrive replaces DB credentials at the binding layer, Mailgun and S3 calls are deferred to billing/billing_service.

These prerequisites are captured downstream: §02 details the new Worker's surface; §03 details the binding/secret topology; §05 captures the rotation and decommission plan.

01 — Architecture Assessment

Status: complete — populated from `adventive-admin` repo analysis, April 2026.

Current state

Attribute	Value
Application version	3.3.9 (composer.json)
Language	PHP ≥ 7.3
Framework	CodeIgniter 3.1.* (last release 3.1.13, 2022; no active maintenance)
Deploy target	EC2 (Linux) — direct server push via AWS CodeDeploy
CI/CD	Bitbucket Pipelines — three branch-triggered pipelines (development, staging, production)
Environments	`development` , `staging` , `production` (called "ops" in pipeline)
Authentication	JumpCloud LDAP (`application/libraries/Jc_auth.php`) + Duo 2FA (`duosecurity/duo_universal_php`) — **Duo currently disabled** (see pain points)
Data stores	MySQL: `console` DB (accounts, campaigns, plans), `billing` DB (invoices, profiles), `aggregate` DB (stats); AWS Redshift (impression events)
File storage	AWS S3 — invoice PDFs (`billing-*.adventivecdn.com`), override files (`override-*.adventivecdn.com`)
Email	Mailgun (`application/libraries/Mailgun.php`)
Integration endpoints	Internal `billing_service` API, AWS Cognito, AWS CodeDeploy, Bitbucket OAuth, DoubleVerify

Traffic profile: Internal only. Low volume, concentrated around month-end billing operations. Peak: ~10 concurrent operators.

Code inventory

Top-level folder tree

```
adventive-admin/
├── application/
│   ├── config/          (21 files — routes, database, adventive.php, mailgun.php, rest.php)
│   ├── controllers/     (12 controllers + 2 API sub-controllers)
│   ├── models/          (13 models, 11,172 total lines)
│   ├── libraries/       (9 libraries: Jc_auth, Mailgun, AmazonS3, AwsCognito,
│   │                     AwsAthena, AppDeploy, REST_Controller, Duo wrapper)
│   ├── views/           (56 view files across 11 directories)
│   ├── helpers/         (csv, download, file helpers)
│   ├── hooks/
│   ├── data/
│   └── cache/
├── www/                 (public web root, index.php)
├── tests/               (test directory — no test files found)
├── logs/
├── composer.json        (version 3.3.9, PHP ≥ 7.3)
├── appspec.yml          (AWS CodeDeploy — currently inactive, replaced by shell repo)
└── bitbucket-pipelines.yml
```

Route map (controllers → URL patterns)

Controller	URL prefix	Method count	Lines
`Dashboard.php`	`/dashboard`	1	42
`Auth.php`	`/auth`	4	205
`Account.php`	`/account`	10	460
`Billing.php`	`/billing`	18	960
`Campaign.php`	`/campaign`	3	111
`Report.php`	`/report`	22	445
`Reporting.php`	`/reporting`	25	997
`Tools.php`	`/tools`	14	451
`Override.php`	`/override`	8	167
`Chart.php`	`/chart`	4	98
`ManagedServiceJob.php`	`/managedservicejob`	2	104

Controller	URL prefix	Method count	Lines
`Utility.php`	`/utility` (CLI only)	2	58
`api/Campaign_Controller.php`	`/api/campaign`	2	117
`api/Managedservices_Controller.php`	`/api/managedservices`	2	99

Route config (`application/config/routes.php`) has minimal explicit routes — most routing is CodeIgniter's default `controller/method/param` convention.

Models → database tables

Model	File	Lines	Tables	Primary operation
`Account_model`	`Account_model.php`	1,321	`account` , `users` , `account_executive`	SELECT, UPDATE, INSERT
`Billing_model`	`Billing_model.php`	933	`billing_invoice` , `billing_invoice_account` , `billing_invoice_usage` , `billing_profiles` , `account_plan` , `account_plan_sub` , `account_plan_usage`	SELECT, INSERT, UPDATE
`Campaigns_model`	`Campaigns_model.php`	665	`campaign` , `client` , `ad_html5` , `ad_format`	SELECT, INSERT
`Report_model`	`Report_model.php`	**5,333**	`campaign` , `ad_html5` , `stats_aggregate_*` , `impressions` (Redshift)	SELECT (complex analytical)
`ManagedService_model`	`ManagedService_model.php`	505	`managed_service_jobs` , `managed_service_jobs_ads` , `managed_service_impression_log` , `managed_service_job_log`	SELECT, INSERT, UPDATE
`Stats_model`	`Stats_model.php`	1,286	`stats_*` aggregate tables	SELECT, INSERT
`Override_model`	`Override_model.php`	280	`override_file_descriptions` + S3	S3 + DB
`Dashboard_model`	`Dashboard_model.php`	121	`account`	SELECT

Model	File	Lines	Tables	Primary operation
`Advertiser_model`	`Advertiser_model.php`	112	`client`	SELECT, INSERT
`Tools_model`	`Tools_model.php`	298	various	SELECT
`Utility_model`	`Utility_model.php`	89	`account`, `api_keys`	UPDATE
`Pdo_model`	`Pdo_model.php`	124	(generic PDO wrapper for Redshift)	Generic
`Uuid_model`	`Uuid_model.php`	105	—	UUID generation utility

Views → operator workflow identification

Directory	File count	Operator surface
`views/account/`	12	Account list, detail, creation wizard, activation, cancellation
`views/billing/`	11	Billing dashboard, invoice list, processing forms, delinquent, managed services
`views/campaign/`	1	Campaign + ad list
`views/dashboard/`	1	Main dashboard
`views/report/`	5	Report listing, queue status
`views/tools/`	14	Lookup tools, ad types, benchmarks, permissions, deployment
`views/override/`	2	Override file editor + version viewer
`views/auth/`	2	Login form, Duo 2FA prompt
`views/common/`	7	Header, footer, sidebar, JS includes
`views/errors/`	10	Error pages

Third-party libraries (composer.json)

Package	Version	Status	Notes
`codeigniter/framework`	`3.1.*`	⚠ End-of-life	No active maintenance since 2022
`aws/aws-sdk-php`	`3.324.*`	✓ Active	S3, Cognito, Athena
`chriskacerguis/codeigniter-restserver`	`^3.1`	⚠ Community	Not officially maintained; REST framework for CI3
`phpoffice/phpexcel`	`1.7.9`	□ Abandoned	Archived 2019; no security patches. Used by `Billing::getMonthSummary`
`phpoffice/phpspreadsheet`	`1.25.*`	✓ Active	Successor to PHPExcel — **both installed simultaneously**
`mk-j/php_xlsxwriter`	`^0.39`	✓ Active	Third Excel library; three overlapping Excel dependencies
`duosecurity/duo_universal_php`	`^1.0`	✓ Active	Currently disabled in application
`paragonie/sodium_compat`	`^1.21`	✓ Active	Libsodium polyfill
`firebase/php-jwt`	`^6.4`	✓ Active	JWT support
`phan/phan`	`^1` (dev)	⚠ Outdated	Static analysis tool; current major is v5

Scheduled tasks / cron jobs

`Reporting.php` (997 lines) is the cron-facing controller. Its 25 public methods are called by server cron jobs via HTTP — **no auth check in the constructor** (the class extends `CI_Controller` without the `jc_auth->is_authenticated()` guard present in all other controllers). Each method fetches data from `Report_model`, generates a CSV/Excel file, and emails it to a partner contact via Mailgun.

Partner reports delivered on schedule:

Method	Schedule (implied)	Recipient
`runRefineryDaily`	Daily	Refinery Media
`runSfGateXMLWeekly`	Weekly	SF Gate
`runBridgetowerMonthly`	Monthly	BridgeTower Media

Method	Schedule (implied)	Recipient
`runHerCampusDaily`	Daily	Her Campus
`runHerCampusMonthlySite`	Monthly	Her Campus (per site)
`runRefineryMonthlySite`	Monthly	Refinery (per site)
`runSagaCityQuarterlySite`	Quarterly	Saga City
`runGoYuitMonthly`	Monthly	GoYuit
`runMansuetoDoubleVerifydaily`	Daily	Mansueto / DoubleVerify
`runMoveDoubleVerifydaily`	Daily	Move / DoubleVerify
`runFarmJournalMonthly`	Monthly	Farm Journal
`runMorganMurphyMonthly`	Monthly	Morgan Murphy Media
`runSonomaWeekly`	Weekly	Sonoma Media
`runHighSnobietyMonthly`	Monthly	HighSnobiety
`runAfarMonthly`	Monthly	Afar
`runNPRMonthly`	Monthly	NPR
`runAtlanticMonthly`	Monthly	The Atlantic
`runPulpoMonthly`	Monthly	Pulpo
`runPulpoWeekly`	Weekly	Pulpo
`runPulpoDaily`	Daily	Pulpo
`runPulpoPreviousDay`	Daily	Pulpo

`Utility.php` (CLI only) provides `resetApiRequestCounts` and `updateLiveCampaigns` — system maintenance tasks run from CLI, not HTTP.

Integration endpoints

Integration	Usage	Library / mechanism
Internal `billing_service` API	Invoice generation, payment processing, ACH info, remittance	`file_get_contents()` + curl POST (`Billing_model.php`)
AWS S3	Invoice PDFs, override files, report exports	`application/libraries/AmazonS3.php` (aws-sdk-php)
AWS Cognito	User sync, OAuth	`application/libraries/AwsCognito.php`
AWS Redshift	Impression analytics queries	`application/models/Pdo_model.php` (PDO direct)
AWS Athena	Events querying	`application/libraries/AwsAthena.php`
Mailgun	Invoice emails, report delivery	`application/libraries/Mailgun.php`
JumpCloud LDAP	Operator authentication	`application/libraries/Jc_auth.php`
Duo Security	2FA (currently disabled)	`duosecurity/duo_universal_php`
DoubleVerify	Ad verification report data	Direct HTTP calls in `Report_model.php`
Bitbucket OAuth	App deployment	`Tools.php` + curl
AWS CodeDeploy	Application deployments	`application/libraries/AppDeploy.php`

Domain decomposition

Domain	Current surface	Planned API seam	Stripe Billing Transition impact	Notes
Account / customer management	`Account.php` , `Account_model.php` , `Dashboard.php`	`admin-api / customers` , `/ dashboard`	None — accounts stay in legacy DB	Core operator workflow; cohort 1

Domain	Current surface	Planned API seam	Stripe Billing Transition impact	Notes
Service-level management	`Account.php` (settings), `ManagedServiceJob.php`, `Override.php`	`admin-api /service-levels`, `/managed-jobs`, `/overrides`	None	Cohort 2
Campaigns & ads	`Campaign.php`, `api/Campaign_Controller.php`, `Campaigns_model.php`	`admin-api /campaigns`, `/ads`	None	Cohort 3; ad copy is super-admin only
Billing — invoice view	`Billing.php` (view methods), `Billing_model.php`	`admin-api /invoices` → reads Stripe	Billing transitions to Stripe as SoR	Admin becomes a Stripe-read view
Billing — payment processing	`Billing::processPayment`	Replaced by Stripe Checkout / PaymentIntent	Stripe owns payment collection	Manual payment entry via Stripe Dashboard
Billing — invoice PDF + email	`billing_service` API → S3 → Mailgun	`invoice-renderer` Worker → R2 → Mailgun	Custom PDF retained (ADR)	Webhook-driven off `invoice.finalized`
Billing — month-end rollup	`Billing::getMonthSummary`, `generateInvoices`	Retired — Stripe accumulates InvoiceItems as-they-happen	Rollup job fully retired	Excel summary report may be reimplemented as a Stripe-read report
Operator auth / sessions	`Auth.php`, `Jc_auth.php`, CI sessions	Cloudflare Access (edge-terminated)	None	Directory choice deferred
Partner reports (scheduled)	`Reporting.php`, `Report_model.php`	Cloudflare Cron Trigger Workers	None	25 methods → Cron Triggers; deferred to Phase 4

Domain	Current surface	Planned API seam	Stripe Billing Transition impact	Notes
Partner reports (on-demand)	`Report.php`, `Report_model.php`	Deferred — dedicated analytics service evaluation	None	5,333-line god-model; not in admin-api scope
Tooling (deploy, Cognito)	`Tools.php`	`admin-api / tools`	None	App deployment UI; Cognito sync stays
Chart / metrics data	`Chart.php`	`admin-api / charts`	None	Simple aggregates off existing DB

Strengths

- **Auth is already isolated:** `Jc_auth.php` and `Duo` libraries are clean seams — dropping them for Cloudflare Access requires only removing the `jc_auth->is_authenticated()` guard in each controller constructor. No auth logic is scattered through controllers.
- **Models are well-separated from controllers:** The PHP model/controller split means DB queries can be read and ported without reverse-engineering view logic.
- **Billing DB is already separate:** `billing` database is physically separate from `console` — Stripe Billing Transition can take over the billing DB tables without schema conflict.
- **Deployment is environment-parameterized:** Bitbucket Pipelines to CodeDeploy pattern is understood; Cloudflare Workers + Pages CI/CD mirrors the same branch→environment model.
- **Mailgun is already library-isolated:** `application/libraries/Mailgun.php` is a clean seam — invoice-renderer Worker can call the same Mailgun API without touching the admin-api codebase.
- **No test debt to preserve:** the `tests/` directory is empty — no legacy test suite to port.

Pain points

P1 — CRITICAL: Production secrets hardcoded in source `application/config/adventive.php` contains AWS access keys, S3 secrets, and Bitbucket OAuth credentials for all three environments in plaintext. Production key `AKIAIAPRP6JG4QQK3YZA` is visible in the source history. The file comment reads *"to be removed upon inclusion of reading from aws-secretsmanager"* — never completed. Remediation is required before migration begins.

P2 — ⚠ CSRF protection globally disabled `application/config/config.php:320` — `'csrf_protection' => false`. Mutations (account creation, billing operations, settings changes) have no per-request token

protection. The admin relies entirely on session auth. Cloudflare Access + the new API's JWT validation replaces this at the architectural level, but the gap is live today.

P3 — ⚠ Duo 2FA currently bypassed Most recent auth commit: `config(auth): temporarily disable the DUO`. Current operator auth is JumpCloud username/password only — no second factor. Live until Cloudflare Access is in place.

P4 — ⚠ MD5-based payment link security `Billing.php:561` — `$payNowUrl .= md5($invoice->payment_customer_id . $invoice->inv_uuid . $invoice->acct_id)`. MD5 is cryptographically broken for integrity purposes. If `payment_customer_id` is known or guessable, the link can be forged. Replaced by Stripe Checkout Sessions in the target architecture.

P5 — `phpoffice/phpexcel` 1.7.9 — abandoned library Archived 2019. No security patches. `Billing::getMonthSummary` depends on it directly. Both PHPExcel and its successor `phpspreadsheet` are installed simultaneously — three Excel libraries total with `mk-j/php_xlsxwriter`. The month-end summary generator must migrate to `phpspreadsheet` (or be retired when Stripe owns the billing data).

P6 — `Report_model.php` at 5,333 lines — god-model 24 functions, each 100–300 lines of partner-specific SQL with hardcoded account IDs, campaign IDs, and date arithmetic. Analytical queries span Redshift and the MySQL `console` DB. This is not portable to a standard Hono CRUD route — it requires a dedicated analytics service decision. Recommend Phase 4 deferral with explicit evaluation.

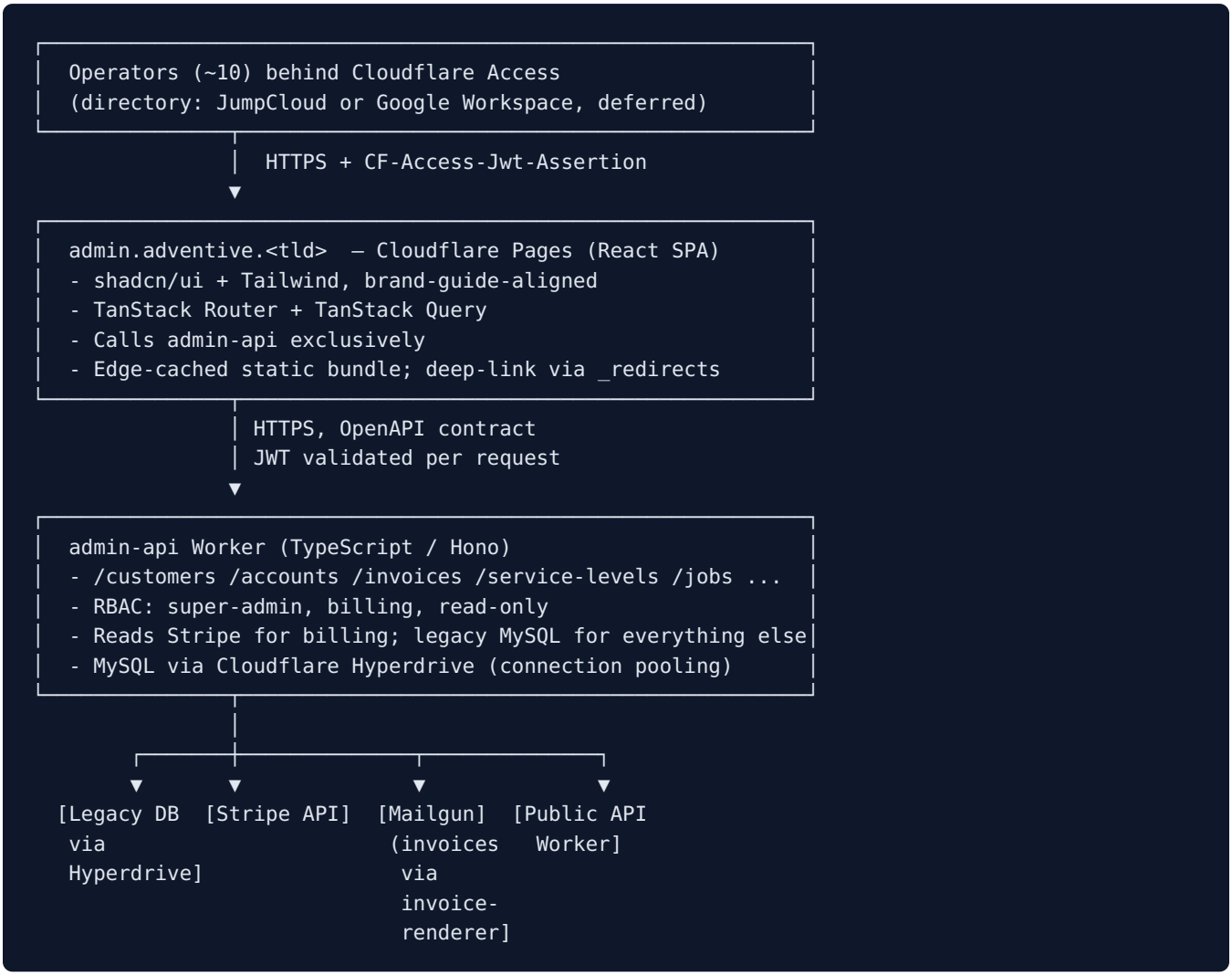
P7 — `Reporting.php` cron controller is unauthenticated HTTP The `Reporting` controller has no `jc_auth->is_authenticated()` guard. Any HTTP client that knows the route can trigger a report delivery. In the serverless architecture these become Cloudflare Cron Triggers — auth is handled by the Workers runtime, not application code.

P8 — Raw SQL strings in models `Billing_model.php` constructs SQL as interpolated strings in multiple places (e.g., lines 33, 86, 101, 148, 193, 256, 329, 347, 432). While values are mostly bound via CodeIgniter's query-builder binds, the pattern mixes interpolated and bound queries, making injection audits difficult. All queries rewritten as parameterized Hono/Drizzle queries in the target.

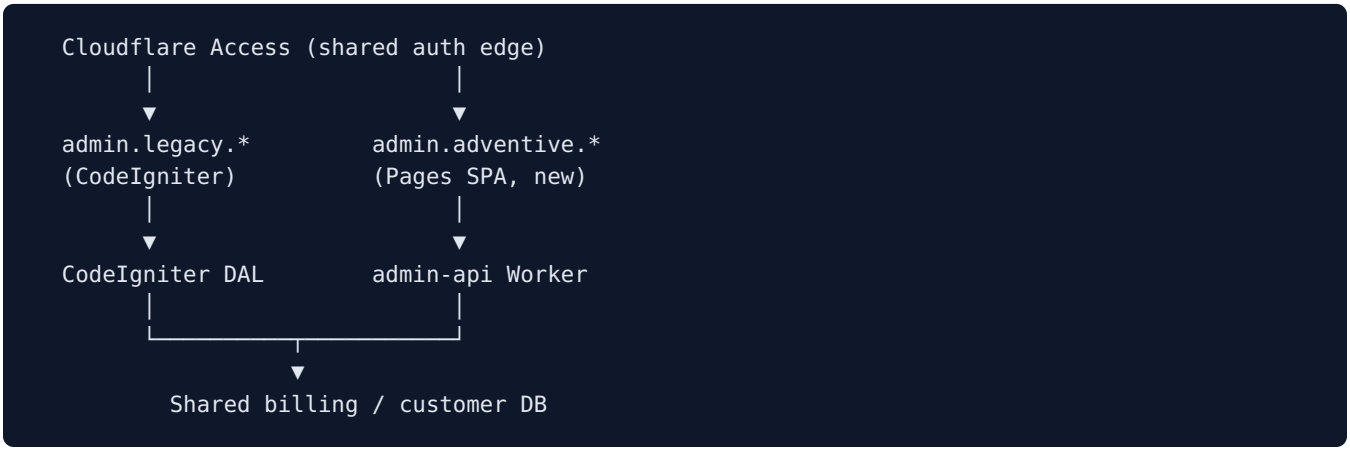
P9 — Deploy process requires SSH + AWS CLI in CI runner `bitbucket-pipelines.yml` bootstraps `aws-cli` and `openssh` in an Alpine container on each deploy, clones the private `shell` repo for deploy scripts, then calls proprietary `updateApp.sh`. This is fragile and environment-specific. Target: `wrangler deploy` from GitHub Actions — a one-command deploy with no shared-state scripts.

P10 — No test coverage `tests/` directory is empty. Zero test coverage. `phan/phan` (static analysis) is in `require-dev` at version 1 (current: 5) and shows no evidence of active use. Target architecture has unit tests (Vitest), contract tests (generated OpenAPI types), and Playwright E2E.

Target architecture (decided)



Parallel-run topology during transition:



Both admins read/write the same database during the transition. Operators migrate cohort-by-cohort; rollback is trivial (return to legacy URL).

Serverless viability assessment

Dimension	Current state	Serverless fit	Notes
Statefulness	PHP sessions, CI session library	□Excellent	Cloudflare Access JWT replaces sessions entirely; admin-api is stateless
Cold-start tolerance	Persistent PHP process (EC2)	□Good	Internal tool — ~10 operators; occasional cold starts are acceptable; Hono is sub-10ms startup
Request duration	Typical < 500ms; month-end Excel generation can run 5–30s	⚠ Caution	Excel/CSV generation moves to async (Queues) or client-side; Stripe-read views are fast
Memory footprint	PHP process ~64–128 MB typical	□Good	Hono Worker runs at ~10–20 MB; no full PHP runtime overhead
Dependency surface	10 composer packages; PHPExcel abandoned	□Better	npm ecosystem; no abandoned-library risk; Stripe SDK, Hono, Zod
PDF rendering	External `billing_service` generates PDF to S3	⚠ Constrained	Cloudflare Browser Rendering (leading candidate) or external service for invoice-renderer Worker; out of admin-api scope
Secrets / config	Hardcoded in `adventive.php` □	□Much better	Wrangler `secret` / Cloudflare environment variables; no committed secrets
Observability	Server logs on EC2; no structured metrics	□Better	Workers Analytics Engine + Logpush → existing log sink
Cost at current scale	EC2 instance + RDS (always-on)	□Lower	Workers: pay-per-request; Pages: free tier; Hyperdrive: ~$0.001/query at this scale
Scheduled jobs	Server cron → unauthenticated HTTP endpoints	□Better	Cloudflare Cron Triggers — authenticated, managed, observable

Recommendation: Proceed. Cloudflare Workers + Pages is a strong fit for this workload. The only constrained dimension is the billing PDF renderer — this is deliberately out of admin-api scope (handled by the invoice-renderer Worker in the Stripe Billing Transition project). All operator-facing request durations fit comfortably within the 30-second Worker limit with substantial headroom.

Operator workflow catalog

34 distinct operator-facing workflows, organized by cohort migration order:

#	Workflow	Controller::method	URL pattern	RBAC tier	Frequency
1	Login / 2FA	`Auth::login`	`/auth/login`	All	Daily
2	Dashboard overview	`Dashboard::index`	`/dashboard`	All	Daily
3	List/search accounts	`Account::index`	`/account`	All	Daily
4	View account detail	`Account::detail`	`/account/:id`	All	Daily
5	Create account	`Account::newAccount` + `validateNewAccount`	`/account/newAccount`	Super-admin	Weekly
6	Activate trial account	`Account::activateAccount`	`/account/activateAccount`	Super-admin	Weekly
7	Cancel account	`Account::cancelAccount`	`/account/cancelAccount/:id`	Super-admin	Rare
8	Unlock user	`Account::unlockUser`	`/account/unlockUser/:id`	Super-admin	Weekly
9	Toggle account settings	`Account::changeSetting`	`/account/changeSetting`	Super-admin	Weekly
10	Change account admin	`Account::updateAccountAdmin`	`/account/updateAccountAdmin`	Super-admin	Rare
11	User lookup by email	`Tools::userlookup`	`/tools/userlookup`	All	Daily
12	Preview link lookup	`Tools::lookup` + `previewSearch`	`/tools/lookup`	All	Weekly
13	Billing dashboard	`Billing::index`	`/billing`	Billing	Daily

#	Workflow	Controller::method	URL pattern	RBAC tier	Frequency
14	Invoice list / filter	`Billing::invoices`	`/billing/invoices`	Billing	Daily
15	Generate invoices (batch)	`Billing::generateInvoices`	`/billing/generateInvoices`	Billing	Monthly
16	Email invoice batch	`Billing::emailInvoice`	`/billing/emailInvoice`	Billing	Monthly
17	Send invoice reminder	`Billing::sendInvoiceReminder`	`/billing/sendInvoiceReminder`	Billing	Monthly
18	Process payment	`Billing::processPayment`	`/billing/processPayment`	Billing	Weekly
19	Update billing profile	`Billing::updateBillingProfile`	`/billing/updateBillingProfile`	Billing	Rare
20	View delinquent invoices	`Billing::viewDelinquentInvoices`	`/billing/viewDelinquentInvoices`	Billing	Monthly
21	Revenue history	`Billing::history`	`/billing/history`	Billing	Monthly
22	QuickBooks CSV export	`Billing::generateQuickbooksCSV`	`/billing/generateQuickbooksCSV`	Billing	Monthly
23	Commission report	`Billing::generateCommissionReport`	`/billing/generateCommissionReport`	Billing	Monthly
24	Month-end summary (Excel)	`Billing::getMonthSummary`	`/billing/getMonthSummary`	Billing	Monthly
25	Managed services billing overview	`Billing::managedServices`	`/billing/managedServices`	Billing	Monthly
26	Campaign list (account)	`api/Campaign_Controller::get`	`/api/campaign?resource=accountCampaigns`	All	Weekly

#	Workflow	Controller::method	URL pattern	RBAC tier	Frequency
27	Ad list (account)	`api/Campaign_Controller::get`	`/api/campaign?resource=accountAds`	All	Weekly
28	Copy ad/ template	`Campaign::copyAdOrTemplate`	`/campaign/copyAdOrTemplate`	Super-admin	Rare
29	Managed service jobs list	`api/Managedservices_Controller::get`	`/api/managedservices?resource=jobs`	Billing	Weekly
30	Create/edit managed service job	`ManagedServiceJob::create/updateManagedServiceJob`	`/managedservicejob/*`	Billing	Weekly
31	Override file versioning	`Override::*`	`/override/*`	Super-admin	Rare
32	Ad types / video benchmarks	`Tools::adtypes`, `videoBenchmarks`	`/tools/adtypes`	All	Rare
33	Special permissions management	`Tools::specialPermissions`	`/tools/specialPermissions`	Super-admin	Rare
34	Application deployment	`Tools::appDeploy`, `deploy`	`/tools/appDeploy`	Super-admin	Weekly

Out of scope for admin-api (deferred / separate): - Partner report downloads (22 methods in `Report.php`) — Phase 4 evaluation: dedicated analytics Worker or self-serve Stripe reporting - Partner report scheduled deliveries (25 methods in `Reporting.php`) — Cloudflare Cron Trigger Workers, separate from admin-api

Data model — billing tables fate

Table	Database	Fate	Owned by
`billing_invoice`	billing	**Retire** — Stripe Invoice becomes SoR	Stripe Billing Transition
`billing_invoice_account`	billing	**Retire** — Stripe InvoiceItem	Stripe Billing Transition

Table	Database	Fate	Owned by
`billing_invoice_usage`	billing	**Retire** — Stripe usage records	Stripe Billing Transition
`billing_profiles`	billing	**Partial retire** — address/ACH fields move to Stripe Customer; `stripe_customer_id` column stays	Stripe Billing Transition
`account_plan`	console	**Retain** during transition; long-term map to Stripe Products/Prices	Admin project (read-only in new admin)
`account_plan_sub`	console	**Retain** during transition; migrates to Stripe Subscription	Stripe Billing Transition
`account_plan_usage`	console	**Retain** during transition; migrates to Stripe Prices (graduated)	Stripe Billing Transition
`account`	console	**Retain** — customer record mirrors Stripe Customer via `stripe_customer_id`	Admin project
`users`	console	**Retain**	Admin project
`campaign`, `client`, `ad_html5`	console	**Retain**	Admin project
`managed_service_jobs`	console	**Retain**	Admin project
`override_file_descriptions`	console	**Retain** — file references migrate from S3 to R2	Admin project

Decisions logged

- **2026-04-23 — Operator auth: Cloudflare Access.** Directory integration (JumpCloud vs. Google Workspace) deferred. See `decisions/2026-04-23-cloudflare-access-for-operator-auth.md`.
- **2026-04-23 — UI stack: React + Vite + TypeScript SPA on Cloudflare Pages.** shadcn/ui + Radix + Tailwind + TanStack Query + TanStack Router. See `decisions/2026-04-23-ui-stack-react-spa.md`.
- **2026-04-23 — Deployment strategy: parallel run, cohort migration, no CodeIgniter retrofit.** See `decisions/2026-04-23-ship-together-no-codeigniter-retrofit.md`.
- **2026-04-23 — Data store: existing DB stays, admin-api connects directly via Hyperdrive during transition.** No data migration in this project scope.
- **2026-04-23 — Invoice PDF renderer retained (Stripe Billing Transition ADR).** Custom Handlebars renderer moves to a dedicated `invoice-renderer` Worker. See sibling project `decisions/2026-04-23-custom-invoice-pdf-retained.md`.

- **2026-04-23 — Stripe owns billing source of truth (Stripe Billing Transition ADR).** Month-end rollup retired. Admin billing surface becomes a Stripe-read view. See sibling project `decisions/2026-04-23-stripe-billing-scope.md`.

Open architectural questions

1. **Partner report surface** — 5,333-line `Report_model.php` with 24 hardcoded partner SQL queries. Options: (a) port to dedicated analytics Worker, (b) retire as partners migrate to Stripe self-serve, (c) thin wrapper exposing Redshift queries via admin-api. Needs evaluation in Phase 4.
2. **R2 vs. S3 for override files** — Current override files live in S3 (`override-*.adventivecdn.com`). Natural migration target is Cloudflare R2 (no egress fees, Workers-native). Decision needed before Phase 3 implementation.
3. **Cohort migration order (confirmed)** — Workflows 1–12 (accounts/lookup) → Workflows 13–25 (billing, coordinated with Stripe transition) → Workflows 26–31 (campaigns, managed services, overrides) → Workflows 32–34 + partner reports.
4. **App deployment tooling** — `Tools::appDeploy` currently orchestrates CodeDeploy via Bitbucket OAuth. Once the admin moves to Workers/Pages, deployment is via `wrangler deploy` from GitHub Actions. The legacy CodeDeploy tooling in the admin UI may be retired rather than ported.
5. **Month-end Excel summary** — `Billing::getMonthSummary` generates an Excel workbook for billing team review. After Stripe transition, the underlying data comes from Stripe. Should this be rebuilt as a Stripe-read report, or replaced by Stripe's native reporting? Needs Jeffrey + Patrick input.
6. **UI component library** — shadcn/ui + Radix + Tailwind confirmed as leading candidate. Formally validate against brandguide.adventive.com/dashboard before Phase 1 implementation begins.

02 — Code Updates / Migration Plan

Summary of changes

Greenfield replacement. Two new repositories (`adventive-admin-api` and `adventive-admin-ui`) ship in parallel with the existing CodeIgniter admin at `~/Repositories/BitBucket/adventive-admin`. Both read/write the same backing database during the transition via Cloudflare Hyperdrive (MySQL connection pooling). Operators migrate cohort-by-cohort. The CodeIgniter admin is not modified — it is decommissioned once parity is reached and a 30-day freeze window completes.

Before beginning implementation: production secrets (`application/config/adventive.php`) must be rotated and moved to AWS Secrets Manager or Cloudflare secrets. This is a prerequisite, not a Phase 1 task.

Target repository layout

```
adventive-admin-api/                (TypeScript / Hono on Cloudflare Workers)
├── src/
│   ├── index.ts                    # Hono app entrypoint + CF Access middleware
│   ├── auth/
│   │   ├── access.ts               # CF-Access-Jwt-Assertion validation
│   │   └── rbac.ts                 # Role derivation from Access email → super-admin/
billing/read-only
│   ├── routes/
│   │   ├── customers.ts            # GET/PATCH /customers, /customers/:id
│   │   ├── users.ts                # GET /customers/:id/users, POST /users/:id/unlock
│   │   ├── accounts.ts             # Account settings, activation, cancellation
│   │   ├── campaigns.ts            # GET /campaigns, /ads (per account)
│   │   ├── invoices.ts             # GET /invoices → reads Stripe
│   │   ├── billing-profiles.ts     # GET/PUT /billing-profiles/:account_id
│   │   ├── managed-jobs.ts         # CRUD /managed-jobs
│   │   ├── overrides.ts            # CRUD /overrides (R2-backed)
│   │   ├── service-levels.ts       # Account plan/tier management
│   │   ├── charts.ts               # GET /charts/* (account reg, revenue, ads)
│   │   ├── tools.ts                # User lookup, preview lookup, ad types
│   │   └── operators.ts            # Operator RBAC management
│   ├── db/
│   │   ├── client.ts               # Hyperdrive MySQL client
│   │   ├── queries/
│   │   │   ├── accounts.ts         # Port of Account_model queries
│   │   │   ├── campaigns.ts        # Port of Campaigns_model queries
│   │   │   ├── managed-jobs.ts     # Port of ManagedService_model queries
│   │   │   └── charts.ts           # Port of Dashboard_model + Chart queries
│   │   └── schema.ts               # Table name constants
│   ├── stripe/
│   │   └── client.ts               # Stripe SDK wrapper (invoice reads, customer reads)
│   ├── r2/
│   │   └── overrides.ts            # R2 operations for override file versioning
│   └── schemas/
│       └── *.ts                    # Zod schemas → @hono/zod-openapi
├── wrangler.toml
├── vitest.config.ts
└── test/
    ├── routes/
    └── auth/

adventive-admin-ui/                 (React + Vite on Cloudflare Pages)
├── src/
│   ├── main.tsx
│   ├── app/
│   │   ├── router.tsx              # TanStack Router config
│   │   └── query-client.ts         # TanStack Query config
│   ├── features/
│   │   ├── dashboard/              # Dashboard overview → replaces Dashboard.php view
│   │   ├── customers/              # Account list, detail, create, activate, cancel
│   │   ├── billing/                # Invoice list, payment, profiles, delinquent
│   │   ├── managed-jobs/           # Managed service job CRUD
│   │   ├── campaigns/              # Campaign + ad list, ad copy
│   │   ├── overrides/              # Override file versioning
│   │   └── tools/                  # User lookup, ad types, benchmarks, deploy
│   ├── api/
```

```
│   │   └── generated.ts            # Types generated from admin-api OpenAPI spec
│   ├── components/
│   │   ├── ui/                     # shadcn/ui primitives
│   │   └── brand/                  # Adventive brand overrides (colors, typography)
│   └── lib/
│       └── access.ts               # Read CF Access identity from /me endpoint
├── public/
│   └── _redirects                  # SPA fallback for CF Pages deep links
├── vite.config.ts
└── e2e/                            # Playwright
    └── flows/
        ├── account-management.spec.ts
        └── billing.spec.ts
```

Before / after

Before

- Monolithic CodeIgniter 3.1 app: PHP controllers render HTML views, same process handles DB access, billing rollup, invoice rendering, file I/O, and scheduled report delivery.
- Auth: JumpCloud LDAP → Duo 2FA → CI session. Currently: LDAP only (Duo disabled).
- Secrets: hardcoded in `application/config/adventive.php`.
- CSRF: globally disabled.
- Deploy: Bitbucket Pipelines → AWS CLI → CodeDeploy → EC2. Shared deploy-script repo (`shell`).
- Observability: EC2 server logs; no structured metrics or alerts.

After

- Three services: `admin-api` Worker (API), `admin-ui` Pages (presentation), `invoice-renderer` Worker (billing — Stripe Billing Transition scope).
- Auth: Cloudflare Access at the edge (JumpCloud or Google Workspace, deferred). Admin-api validates `CF-Access-Jwt-Assertion` on every request. No application-managed sessions.
- Secrets: Wrangler `secret` / Cloudflare environment variables. Nothing committed to source.
- CSRF: eliminated — stateless JWT architecture has no session to forge.
- Deploy: GitHub Actions → `wrangler deploy` (Worker) + Pages CI/CD (SPA). Single command per environment.
- Observability: Workers Analytics Engine + Logpush → existing log sink.

Migration phases

Phase	Scope	Gate to next
0	**Pre-work:** Rotate and remove hardcoded secrets from `adventive.php`. Move to AWS Secrets Manager. This is a prerequisite.	Secrets cleared from source history

Phase	Scope	Gate to next
1	Scaffold `admin-api` Worker (Hono skeleton, CF Access middleware, RBAC, stub endpoints) + `admin-ui` Pages (Vite + shadcn/ui + TanStack Router skeleton). Deploy to preview URLs. Cloudflare Access policy provisioned for all ~10 operators.	Preview deploys green; all operators can authenticate
2	Implement customer + account management domain (Cohort 1 workflows 1–12): account list, detail, create, activate, cancel, user lookup, unlock, settings. RBAC enforcement. Full test coverage.	Feature parity checklist for cohort 1 complete
3	**Cohort 1 migration** — account-management operators move to new admin. Legacy stays live. Collect feedback, iterate UI.	Cohort retention at new admin ≥ 2 weeks with no rollback
4	Implement billing read surface (Cohort 2 workflows 13–25): invoice list/detail from Stripe, billing profiles, delinquent view, revenue history, managed services. Co-develops with Stripe Billing Transition Phase B.	Billing parity checklist complete; Stripe SoR stable
5	**Cohort 2 migration** — billing-ops operators move to new admin.	Cohort retention ≥ 2 weeks
6	Implement campaigns, managed service jobs, override files (Cohort 3 workflows 26–31).	Parity checklist complete
7	**Cohort 3 migration** — remaining operators.	All operators on new admin
8	Phase 4 deferred items: tooling (deploy, Cognito sync, benchmarks), partner report evaluation.	All workflows covered or explicitly deferred
9	**Freeze legacy admin** (read-only mode). Start 30-day freeze window.	30 days clean
10	**Decommission** CodeIgniter admin. Archive repo.	—

File-by-file change list

Legacy files are **not modified** — they are inventoried so the replacement is feature-complete.

Legacy file	Lines	Replaced by	Risk	Notes
`application/controllers/Auth.php`	205	Cloudflare Access (no code equivalent)	Low	Auth moved entirely to edge; no application auth controller

Legacy file	Lines	Replaced by	Risk	Notes
`application/controllers/Dashboard.php`	42	`admin-api/src/routes/charts.ts` + UI dashboard feature	Low	Simple aggregation
`application/controllers/Account.php`	460	`admin-api/src/routes/customers.ts`, `accounts.ts`, `users.ts`	Medium	CRUD + account-service API call replicated in Worker
`application/controllers/Billing.php`	960	`admin-api/src/routes/invoices.ts`, `billing-profiles.ts` (Stripe reads)	High	Invoice generation retired; payment processing via Stripe; PDF stays in invoice-renderer Worker
`application/controllers/Campaign.php`	111	`admin-api/src/routes/campaigns.ts`	Medium	Ad copy curl call → internal Worker call
`application/controllers/Report.php`	445	Phase 4 evaluation — analytics Worker or Stripe reports	High	22 methods; do not port to admin-api
`application/controllers/Reporting.php`	997	Cloudflare Cron Trigger Workers (separate from admin-api)	High	25 scheduled-delivery methods → individual Cron Triggers
`application/controllers/Tools.php`	451	`admin-api/src/routes/tools.ts` (partial)	Medium	App deployment tooling may be retired; Cognito sync stays
`application/controllers/Override.php`	167	`admin-api/src/routes/overrides.ts` + R2	Low	S3 → R2 migration; same versioning model
`application/controllers/Chart.php`	98	`admin-api/src/routes/charts.ts`	Low	4 JSON endpoints; straightforward port

Legacy file	Lines	Replaced by	Risk	Notes
`application/controllers/ManagedServiceJob.php`	104	`admin-api/src/routes/managed-jobs.ts`	Low	CRUD; clean port
`application/controllers/Utility.php`	58	Cloudflare Cron Trigger Workers or retired	Low	CLI-only; `resetApiRequestCounts`, `updateLiveCampaigns`
`application/controllers/api/Campaign_Controller.php`	117	`admin-api/src/routes/campaigns.ts`	Low	DataTables pagination → standard pagination
`application/controllers/api/Managedservices_Controller.php`	99	`admin-api/src/routes/managed-jobs.ts`	Low	Same
`application/models/Account_model.php`	1,321	`admin-api/src/db/queries/accounts.ts`	High	All queries must be reproduced with full parity
`application/models/Billing_model.php`	933	`admin-api/src/stripe/client.ts` + residual DB queries	High	Billing queries largely retire; Stripe reads replace most
`application/models/Campaigns_model.php`	665	`admin-api/src/db/queries/campaigns.ts`	Medium	
`application/models/Report_model.php`	5,333	Phase 4 — analytics Worker	Critical	Do not attempt to port inline; 24 bespoke SQL functions
`application/models/ManagedService_model.php`	505	`admin-api/src/db/queries/managed-jobs.ts`	Medium	

Legacy file	Lines	Replaced by	Risk	Notes
`application/models/Stats_model.php`	1,286	Partially retire (billing stats → Stripe); remainder in analytics Worker	High	
`application/models/Override_model.php`	280	`admin-api/src/r2/overrides.ts`	Low	S3 ops → R2 ops
`application/models/Dashboard_model.php`	121	`admin-api/src/db/queries/charts.ts`	Low	
`application/config/adventive.php`	—	Cloudflare secrets + environment variables	Critical	**Secrets must be rotated before this file is removed from history**
`application/libraries/Jc_auth.php`	—	Cloudflare Access (no application code)	Low	Retired entirely
`application/libraries/Mailgun.php`	—	Retained in `invoice-renderer` Worker	Low	Not part of admin-api scope
`application/libraries/AmazonS3.php`	—	Cloudflare R2 SDK in overrides Worker	Low	S3 → R2
`application/libraries/AwsCognito.php`	—	Cognito SDK in tools route	Low	Thin wrapper retained
`application/libraries/AppDeploy.php`	—	Evaluate retirement vs. thin port	Low	CodeDeploy tooling; may be superseded by wrangler/Actions

New files (indicative)

File	Purpose
`admin-api/src/index.ts`	Hono app entrypoint; CF Access middleware; global error handler
`admin-api/src/auth/access.ts`	Validates `CF-Access-Jwt-Assertion`; extracts operator email
`admin-api/src/auth/rbac.ts`	Maps operator email → RBAC role from config; enforces per-route
`admin-api/src/db/client.ts`	Hyperdrive MySQL client factory
`admin-api/openapi.json`	Generated from `@hono/zod-openapi`; consumed by UI type generation
`admin-ui/src/api/generated.ts`	TypeScript types generated from OpenAPI spec
`admin-ui/public/_redirects`	CF Pages SPA fallback: `/* /index.html 200`
`.github/workflows/deploy-api.yml`	`wrangler deploy` on merge to main
`.github/workflows/deploy-ui.yml`	Pages CI/CD on merge to main
`admin-api/wrangler.toml`	Worker config: name, routes, Hyperdrive binding, R2 binding, secrets

Removed files (end of Phase 10)

- Entire `~/Repositories/BitBucket/adventive-admin` repository — archived, not deleted.

Dependencies

Added:

Package	Purpose
`hono`	HTTP framework for Workers
`@hono/zod-openapi`	OpenAPI spec generation — single source of truth for types
`zod`	Schema validation
`stripe`	Stripe SDK (Workers-compatible)
`react`, `react-dom`	UI framework

Package	Purpose
`@tanstack/react-query`	Server state management
`@tanstack/react-router`	Type-safe routing
`vite`	Build tool
`tailwindcss`	Utility CSS
`shadcn/ui` components + `@radix-ui/*`	UI component primitives
`vitest`	Unit testing
`@playwright/test`	E2E testing

Removed (end of Phase 10): PHP 7.3 runtime, CodeIgniter 3.1, all composer dependencies.

Breaking changes

- Admin URL changes during transition (operators informed by cohort).
- Operator auth moves from JumpCloud LDAP → Cloudflare Access. All ~10 operators must be provisioned in the Access policy before Cohort 1 migrates.
- API consumers (any automation scripts pointing at legacy admin URLs) must update to new contract.
- `billing_service` internal API is retired as Stripe Billing Transition completes — any external callers must be identified.

Schema-freeze policy during transition

While both admins share the same database, **no schema changes that break backward compatibility with the CodeIgniter admin** may be deployed. New columns (nullable, with defaults) are acceptable. Column drops, renames, and type changes are blocked until the CodeIgniter admin is decommissioned. This policy is in effect from Phase 1 until Phase 10.

Testing strategy

Layer	Tool	Scope
Unit — admin-api	Vitest	Route handlers, Zod schemas, RBAC logic, DB query builders
Unit — admin-ui	Vitest + Testing Library	Components, hooks, form validation

Layer	Tool	Scope
Contract	Generated OpenAPI types	UI type generation catches API drift at build time
Integration	admin-api against staging DB + Stripe test mode	Full request/response parity with legacy behavior
E2E	Playwright against Pages preview deploy	CF Access service token for automation; golden-path flows per cohort
Parity	Per-cohort checklist	Dogfooded alongside legacy admin before cohort migration
Rollback	Any cohort can return to legacy URL	No data migration risk — shared DB

Update — 2026-04-30: locked API contract decisions

API surface — hybrid REST + composite endpoints

Strict REST per resource for the primary surface. A small allow-list of composite endpoints is permitted on heavy screens to avoid request waterfalls without coupling the API to UI shape. The current allow-list:

- `GET /api/accounts/:id?include=users,campaigns,adunits,advertisers,invoices,managedjobs,permissions,plan` — Account Detail (replaces the 9-call waterfall a strict-REST design would force).

Future additions to the composite allow-list require an ADR. Default for all new endpoints is strict REST.

Auth — Cloudflare Access JWT + Worker session

Layer	Mechanism	What it protects
Edge	Cloudflare Access policy on `admin-api.*` and `admin.*` hostnames; JumpCloud SAML/SCIM IdP; WebAuthn second-factor	Authn — only members of JumpCloud `Admin Dashboard` group reach the Worker
Worker — authn verify	`accessAuth` middleware verifies `Cf-Access-Jwt-Assertion` against the Access JWKS (10-min cache); reads `email`, `groups`, `sub` from claims	Defense-in-depth in case a route bypasses Access (misconfig)
Worker — RBAC	`requireRole(...)` middleware factory; roles derived from JumpCloud groups (`Adventive Engineering` → super-admin, `Adventive Billing` → billing, `Admin Dashboard` → read-only)	Authz per route

Layer	Mechanism	What it protects
Worker — session	`ensureSession` middleware: random opaque token in Secure/HttpOnly/SameSite=Strict cookie; record (csrf, email, rotatedAt) in `SESSION_KV`; rotates every 8h or on email change	CSRF on mutations, idempotency on billing actions, audit-log seed
Worker — CSRF	`requireCsrf` middleware on POST/PUT/PATCH/DELETE; asserts `X-Csrf-Token` header equals the session record's csrf token	CSRF
Worker — idempotency	`requireIdempotency` on billing/processing mutations; client provides `Idempotency-Key` header; 24h replay cache in `IDEMPOTENCY_KV`	Double-submit protection on billing actions
Worker — concurrency	`requireIfMatch` on resource updates; client provides `If-Match` header with current ETag	Optimistic concurrency

Legacy auth components (`Auth.php` , `Jc_auth.php` LDAP library, Duo SDK, app-managed sessions, CSRF flag) have no equivalent in the new admin-api. They are removed entirely.

Billing scope — read-only in v1

The new admin-api ships these read-only billing endpoints in v1:

- `GET /api/billing/overview` — KPIs (MTD/QTD/YTD/LTD), 12-month revenue trend
- `GET /api/billing/invoices` — paginated invoice list with status filter
- `GET /api/billing/account-history` — 12-month per-account revenue grid
- `GET /api/billing/aging` — aging bucket summary + per-account breakdown
- `GET /api/billing/delinquent` — open invoices > 15 days past due

All five depend on a `DB_BILLING` Hyperdrive binding that is **not yet provisioned** in dev. The endpoints exist as scaffolds and return 503 with a clear message until the Stripe Billing Transition project provisions Hyperdrives for `billing` and `aggregate_ro` databases.

The following mutating billing endpoints are **deliberately not** in admin-api v1:

- Test invoice generation
- Real invoice generation / billing cycle run
- QuickBooks export
- Commission report generation
- Dunning sweep (delinquent reminders)
- Year-end rollover
- Payment processing (`Billing/processPayment`)
- Account creation (`Account/validateNewAccount`)

These are owned by the Stripe Billing Transition project. The UI keeps the Processing Controls page mocked (with a "Pending Stripe Billing Transition" indicator) until those endpoints are provided. Documented as a hand-off in §05.

Reports — not migrated

The legacy `Report.php` (~5200 LOC, 20+ customer-specific raw SQL queries) does not migrate to admin-api. The Reports surface is removed from the admin-ui in Phase 2. Customer-facing reports move to a future analytics-service evaluation (Phase 4) or to versioned BI templates.

Worker repository

Scaffold lives at `~/Repositories/GitHub/Adventive/adventive-admin-api/`. Initial commit established 2026-04-30. Layout:

```
src/
  index.ts                Hono app, middleware chain, error handler, OpenAPI doc
  types/env.ts            Bindings + Variables + Operator types
  auth/
    access.ts             Cloudflare Access JWT verification (RSASSA-PKCS1-v1_5 + JWKS cache)
    session.ts            Signed-cookie Worker session, CSRF, idempotency, If-Match
  lib/
    db.ts                 mysql2 connect/query helpers (disableEval:true)
    logger.ts             JSON log line emitter, level-gated
    errors.ts             ApiError + named factories
  schemas/
    common.ts             shared zod schemas (status, pagination, errors)
    accounts.ts           Account, AccountSummary, AccountUser, Campaign, includes
  routes/
    health.ts             /healthz
    me.ts                 /api/me
    accounts.ts           /api/accounts (list + composite detail)
    billing.ts            /api/billing/* (read-only; gated on DB_BILLING)
```

Wrangler dry-run for `dev` env passes. Worker name: `adv-svc-admin-api-dev`. Custom hostname: `admin-api.adventive.dev` (AAAA record pre-creation required per standing pattern).

Phase 1 → Phase 2 handoff

The UI prototype's `src/lib/mock/*` files are intentionally shaped to mirror the API contract. Replacing them with TanStack Query hooks against admin-api is the primary handoff. A typed API client generated from the Worker's `openapi.json` lives at `src/lib/api.ts` and replaces the mock module pattern once the API endpoints land.

03 — Deployment & Management

Deploy target

Service	Platform	Runtime	Region
`adventive-admin-api`	Cloudflare Workers	Cloudflare edge (global)	Smart placement — no pin needed
`adventive-admin-ui`	Cloudflare Pages	Static SPA + CDN	Global
`adventive-admin-api` (legacy DB)	Cloudflare Hyperdrive	MySQL connection pooler	Collocated with DB region

Production URLs (to be provisioned): - API: `https://admin-api.adventive.com` (Workers custom domain) - UI: `https://admin.adventive.com` (Pages custom domain) - Legacy admin (during transition): `https://admin-legacy.adventive.com` (existing EC2)

Both new services sit behind Cloudflare Access. Access policy enforces operator authentication before a single byte of the API or UI is reachable.

Environments

Env	API URL	UI URL	Trigger	Notes
Local	`http://localhost:8787`	`http://localhost:5173`	`wrangler dev` / `vite dev`	Uses `.dev.vars` for secrets; proxies to Hyperdrive staging binding
Preview	`admin-api-<branch>.adventive-admin.workers.dev`	CF Pages preview URL (per PR)	PR opened or commit pushed to non-main branch	Automated — CF Pages creates preview per commit; Worker preview via `wrangler deploy --env preview`

Env	API URL	UI URL	Trigger	Notes
Staging	`admin-api-staging.adventive.com`	`admin-staging.adventive.com`	Merge to `staging` branch	Shares staging DB with legacy admin-staging; all Cloudflare Access policies active
Production	`admin-api.adventive.com`	`admin.adventive.com`	Merge to `main` branch after staging sign-off	Requires explicit manual promotion step — no auto-deploy to prod

Promotion rules

```
feature branch → PR → preview deploy (automatic)
                  ↓ review + approve
                staging branch → staging deploy (automatic on merge)
                                 ↓ operator acceptance sign-off (cohort checklist)
                               main → production deploy (manual workflow_dispatch trigger)
```

Production deploys require: (1) staging deploy succeeded, (2) operator checklist signed off, (3) manual trigger by Jeffrey or Patrick.

Build & deploy pipeline

admin-api (Cloudflare Worker)

File: `.github/workflows/deploy-api.yml`

```
name: Deploy admin-api

on:
  push:
    branches: [staging, main]
  workflow_dispatch:
    inputs:
      environment:
        description: 'Target environment'
        required: true
        default: staging
        type: choice
        options: [staging, production]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: npm
          cache-dependency-path: admin-api/package-lock.json
      - run: npm ci
        working-directory: admin-api
      - run: npm run type-check
        working-directory: admin-api
      - run: npm run test
        working-directory: admin-api

  deploy:
    needs: test
    runs-on: ubuntu-latest
    environment: ${{ github.ref == 'refs/heads/main' && 'production' || 'staging' }}
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: npm
          cache-dependency-path: admin-api/package-lock.json
      - run: npm ci
        working-directory: admin-api
      - name: Deploy to Cloudflare Workers
        uses: cloudflare/wrangler-action@v3
        with:
          apiToken: ${{ secrets.CLOUDFLARE_API_TOKEN }}
          accountId: ${{ secrets.CLOUDFLARE_ACCOUNT_ID }}
          workingDirectory: admin-api
          command: deploy --env ${{ github.ref == 'refs/heads/main' && 'production' ||
'staging' }}
```

Local development:

```
cd admin-api
cp .dev.vars.example .dev.vars   # fill in staging secrets
wrangler dev --env staging       # proxies to staging Hyperdrive
```

Manual hotfix deploy:

```
cd admin-api
wrangler deploy --env staging    # staging
wrangler deploy --env production # production (requires CLOUDFLARE_API_TOKEN in local env)
```

admin-ui (Cloudflare Pages)

File: `.github/workflows/deploy-ui.yml`

```
name: Deploy admin-ui

on:
  push:
    branches: [staging, main]
  pull_request:
    branches: [staging, main]

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: npm
          cache-dependency-path: admin-ui/package-lock.json
      - run: npm ci
        working-directory: admin-ui
      - run: npm run type-check
        working-directory: admin-ui
      - run: npm run build
        working-directory: admin-ui
        env:
          VITE_API_BASE_URL: ${{ github.ref == 'refs/heads/main' && 'https://admin-
api.adventive.com' || 'https://admin-api-staging.adventive.com' }}
      - name: Deploy to Cloudflare Pages
        uses: cloudflare/pages-action@v1
        with:
          apiToken: ${{ secrets.CLOUDFLARE_API_TOKEN }}
          accountId: ${{ secrets.CLOUDFLARE_ACCOUNT_ID }}
          projectName: adventive-admin-ui
          directory: admin-ui/dist
          gitHubToken: ${{ secrets.GITHUB_TOKEN }}
          branch: ${{ github.ref_name }}
```

Local development:

```
cd admin-ui
cp .env.example .env.local  # VITE_API_BASE_URL=http://localhost:8787
npm run dev
```

OpenAPI type generation

Run after any admin-api schema change, before committing admin-ui changes:

```
# From repo root
cd admin-api && npm run generate:openapi   # emits openapi.json
cd ../admin-ui && npm run generate:types   # runs openapi-typescript against openapi.json
```

The generated `admin-ui/src/api/generated.ts` is committed to source. A CI check validates it is not stale.

Secrets & configuration

Rule: No secrets in source. No secrets in `wrangler.toml`. All environment-specific values use Wrangler secrets (Worker) or Cloudflare Pages environment variables (UI).

admin-api secrets

Secret name	Description	Set via	Rotation
`HYPERDRIVE_DATABASE_URL`	MySQL DSN (injected by Hyperdrive binding — not a manual secret)	`wrangler.toml` binding	Rotated at DB credential rotation
`STRIPE_SECRET_KEY`	Stripe API secret key (restricted key — invoice read, customer read only)	`wrangler secret put STRIPE_SECRET_KEY`	Annually or on compromise
`CF_ACCESS_AUD`	Cloudflare Access Application AUD tag (JWT audience validation)	`wrangler secret put CF_ACCESS_AUD`	On Access Application re-creation
`CF_ACCESS_TEAM_DOMAIN`	e.g. `adventive.cloudflareaccess.com`	`wrangler secret put CF_ACCESS_TEAM_DOMAIN`	On Access team domain change

Secret name	Description	Set via	Rotation
`RBAC_SUPER_ADMIN_EMAILS`	Comma-separated list of super-admin email addresses	`wrangler secret put RBAC_SUPER_ADMIN_EMAILS`	When operator list changes
`RBAC_BILLING_EMAILS`	Comma-separated list of billing-tier email addresses	`wrangler secret put RBAC_BILLING_EMAILS`	When operator list changes
`R2_OVERRIDE_BUCKET`	R2 bucket name for override files (injected by R2 binding — not a manual secret)	`wrangler.toml` binding	N/A

Setting secrets (run once per environment):

```
wrangler secret put STRIPE_SECRET_KEY --env staging
wrangler secret put STRIPE_SECRET_KEY --env production
# etc. for each secret
```

Local dev secrets (`.dev.vars` , never committed):

```
STRIPE_SECRET_KEY=sk_test_...
CF_ACCESS_AUD=...
CF_ACCESS_TEAM_DOMAIN=adventive.cloudflareaccess.com
RBAC_SUPER_ADMIN_EMAILS=jeffrey@adventive.com,patrick@adventive.com
RBAC_BILLING_EMAILS=...
```

admin-ui environment variables

UI has no secrets — it only knows the API base URL. Set in Cloudflare Pages dashboard:

Variable	Staging value	Production value
`VITE_API_BASE_URL`	`https://admin-api-staging.adventive.com`	`https://admin-api.adventive.com`

Legacy secret migration (Phase 0 prerequisite)

`application/config/adventive.php` contains production AWS keys (`AKIAIAPRP6JG4QQK3YZA`), S3 secret, and BitBucket OAuth credentials. These must be rotated and moved to AWS Secrets Manager before any other work begins. This is a blocker, not a Phase 1 task.

Rotation checklist (Phase 0): 1. Generate new AWS access key pair — deactivate old key in IAM console 2. Generate new Bitbucket OAuth credentials — revoke old tokens 3. Move all values to AWS Secrets Manager (or Cloudflare secrets if already accessible from the EC2 host) 4. Update CodeIgniter config to read from environment / Secrets Manager at runtime 5. Purge `adventive.php` secret values from source 6.

Rewrite git history to remove committed secrets (`git filter-repo` or BFG Repo Cleaner) 7. Invalidate all existing CI artifacts that may cache the old commit

---

Cloudflare Access configuration

Application definitions

Application name	URL pattern	Policy	Session duration
`Adventive Admin UI`	`https://admin.adventive.com/*`	Adventive Operators group	8 hours
`Adventive Admin API`	`https://admin-api.adventive.com/*`	Adventive Operators group + service token (for E2E)	8 hours
`Adventive Admin UI (Staging)`	`https://admin-staging.adventive.com/*`	Adventive Operators group	8 hours
`Adventive Admin API (Staging)`	`https://admin-api-staging.adventive.com/*`	Adventive Operators group + service token	8 hours

Policy — Adventive Operators group

```
Rule: Emails → [list of ~10 operator email addresses]
Identity provider: JumpCloud (or Google Workspace — deferred per ADR)
Require MFA: Yes
```

Service token (for E2E / Playwright automation)

Playwright tests authenticate to Access using a CF Access service token:

```
# Create the service token (one-time, per environment)
# Cloudflare dashboard: Access → Service Auth → Create Service Token
# Store client_id and client_secret in GitHub Actions secrets
# CF_ACCESS_CLIENT_ID, CF_ACCESS_CLIENT_SECRET
```

Playwright config sets `CF-Access-Client-Id` and `CF-Access-Client-Secret` headers in all requests.

Adding a new operator

1. Navigate to Cloudflare Zero Trust dashboard → Access → Access Groups → Adventive Operators
2. Add the operator's email address to the email list
3. Notify operator of the new admin URL and confirm they can authenticate

4. If adding a super-admin or billing-tier operator, also update `RBAC_SUPER_ADMIN_EMAILS` or `RBAC_BILLING_EMAILS` Wrangler secrets

Revoking operator access

1. Remove the operator's email from the Access group
2. If the operator has billing or super-admin RBAC tier, remove from the corresponding Wrangler secret and redeploy the Worker (`wrangler deploy --env production`)
3. Revoke any active Access sessions: Cloudflare dashboard → Access → Revoke User Sessions → enter email

---

Observability

Workers Analytics Engine

The admin-api Worker emits structured events to Workers Analytics Engine on every request:

```
// Captured per request
env.ANALYTICS.writeDataPoint({
  blobs: [route, method, operatorEmail, rbacTier],
  doubles: [responseStatus, durationMs],
  indexes: [route]
});
```

Dimensions available: route, HTTP method, operator email, RBAC tier, response status, request duration.

Dashboards (to be built in Cloudflare Analytics or Grafana): - Request rate by route (operator activity heatmap) - p50/p95/p99 duration by route - Error rate (4xx, 5xx) by route - Operator session activity (login frequency, last seen)

Logpush

Configure Logpush to forward Workers logs to the existing log sink (same destination as Public API Worker):

```
Cloudflare dashboard → Analytics → Logpush → Create job
Dataset: Workers trace events
Destination: [existing log sink — S3 or Splunk]
Filter: worker name = adventive-admin-api
```

Log entries include: timestamp, outcome (ok/exception/exceededCpu), cpuTime, wallTime, request URL, response status, CF-Ray ID.

Tail Workers (real-time debug)

```
wrangler tail adventive-admin-api --env production
wrangler tail adventive-admin-api --env staging --format pretty
```

Use tail Workers during cohort migrations to watch request patterns live.

Alerting

Alert	Threshold	Channel
Worker error rate > 5% (5-min window)	> 5% of requests returning 5xx	PagerDuty / Slack `#admin-alerts`
Worker CPU time approaching limit	p95 > 30ms (warning), > 45ms (critical)	Slack `#admin-alerts`
Hyperdrive connection failures	Any sustained DB error (> 3 consecutive failures)	PagerDuty
Access auth failures spike	> 10 failed Access checks in 5 minutes	Slack `#admin-alerts`
Pages build failure	Any failed build on `main` or `staging`	Slack `#deploy-notifications`

Status page

Add the admin Worker to the existing Adventive status page. External health check endpoint: `GET /health` → `200 {"status":"ok","version":"x.y.z"}`.

Cost model

All costs are on top of existing Cloudflare account (Workers Paid plan assumed — consistent with Public API project).

Cost driver	Unit	Estimated monthly usage	Estimated monthly cost
Workers Paid plan	$5/month flat	—	$5.00
Worker requests (admin-api)	$0.30 / million (after 10M free)	~50K requests/month (~10 operators × ~200 req/day × 25 days)	~$0 (well within free tier)

Cost driver	Unit	Estimated monthly usage	Estimated monthly cost
Worker CPU time	$0.02 / million ms (after 30M free)	~500K ms/month	~$0
Cloudflare Pages	Free (unlimited builds, 500 builds/month included)	~50 builds/month	$0
Hyperdrive	Included with Workers Paid	—	$0 (first 1M queries/ month free)
R2 storage (override files)	$0.015/GB/month	< 1 GB	< $0.02
R2 operations	$0.36 / million Class B	< 10K/month	< $0.01
Workers Analytics Engine	Free (included in Workers Paid)	—	$0
Logpush	Free for Workers	—	$0

Estimated total incremental cost: ~$5–6/month (dominated by Workers Paid plan flat fee, already paid for Public API).

Cost ceiling: At 10 operators with heavy usage (1,000 req/day each), monthly requests = 250K — still well within free request tier. This service will not require re-architecture based on cost at any realistic operator scale.

Rollback procedures

Worker rollback (admin-api)

Cloudflare Workers retains the 10 most recent deploys. Roll back in under 60 seconds:

```
# List available versions
wrangler deployments list --env production

# Roll back to a specific version
wrangler rollback [deployment-id] --env production

# Or roll back to the previous version (no ID needed)
wrangler rollback --env production
```

Verify rollback: `curl https://admin-api.adventive.com/health` → check version field.

Pages rollback (admin-ui)

```
# Via Cloudflare dashboard:
# Pages → adventive-admin-ui → Deployments → click previous deployment → "Rollback to this deployment"

# Or via Wrangler CLI:
wrangler pages deployment list --project-name adventive-admin-ui
wrangler pages deployment rollback [deployment-id] --project-name adventive-admin-ui
```

Cohort rollback (operator routing)

During the parallel-run period, rolling a cohort back to the legacy admin requires only redirecting the operator:

1. Send the operator the legacy admin URL (`https://admin-legacy.adventive.com`)
2. No data migration needed — both admins read/write the same DB
3. Document the rollback reason in the cohort migration log
4. Investigate the failure before re-attempting cohort migration

No code change or deployment is required for a cohort rollback.

Cohort migration playbook

Pre-migration checklist (for each cohort)

Before notifying any operator:

- [ ] Feature parity checklist for the cohort's workflows is signed off (Jeffrey + Patrick)
- [ ] Staging has been dogfooded by Jeffrey for at least 3 days
- [ ] All Vitest tests passing in CI
- [ ] E2E Playwright tests passing against staging
- [ ] Rollback procedure tested: confirm legacy admin URL still works
- [ ] Analytics Engine dashboard live and showing staging traffic
- [ ] Logpush forwarding verified to log sink
- [ ] Cloudflare Access policy verified: each operator on the list can authenticate
- [ ] RBAC secrets set correctly: super-admin and billing emails in Wrangler secrets

Cohort 1 migration — Account Management Operators

Scope: Workflows 1–12 (account list, detail, create, activate, cancel, user lookup, unlock, settings, service levels, override files, tools)

Target operators: Operators who primarily use account management and do not require billing access.

Migration day procedure:

```
1. Send Slack DM to cohort operators: "[New Admin] You're being migrated to the new
   Adventive Admin today. URL: https://admin.adventive.com — log in with your JumpCloud
   credentials through Cloudflare Access. Your old bookmarks still work at
   admin-legacy.adventive.com if you need them."

2. Monitor Workers Analytics Engine dashboard for 30 minutes post-notification:
   - Watch for 4xx or 5xx spikes
   - Confirm each operator shows up in the request log (first request = Access JWT validated)

3. Stand by in Slack for 2 hours. Answer questions. Fix anything that surfaces.

4. 48 hours later: check-in with operators. Note any reported issues.

5. At 2-week mark with no rollback requests: mark Cohort 1 as stable. Gate to Cohort 2 is
open.
```

Rollback trigger: Any operator reports a workflow they cannot complete that they could complete in the legacy admin. Immediately send them to `admin-legacy.adventive.com`, document the gap, and fix before re-attempting.

Cohort 2 migration — Billing Operators

Scope: Workflows 13–25 (invoice list, invoice detail, payment, billing profiles, delinquent accounts, revenue history, managed services list/CRUD)

Dependencies: Stripe Billing Transition Phase B must be stable before this cohort migrates. Coordinate timing with that project.

Migration day procedure: Same as Cohort 1. Additional monitoring: watch Stripe API call success rate from admin-api.

Special note: The billing surface is high-risk. Run Cohort 2 alongside the legacy admin for a minimum of 2 weeks, not just 2 weeks total. If Stripe Billing Transition is in the middle of a cutover when this cohort is ready, hold the cohort until Stripe-side is stable.

Cohort 3 migration — Remaining Operators

Scope: Workflows 26–34 (campaigns, managed service jobs, override files, tooling, ad types, benchmarks)

Migration day procedure: Same. After 2-week stable window, all operators are on the new admin.

Post-Cohort-3 freeze

After all operators have been stable on the new admin for 2 weeks:

1. Put legacy admin into read-only mode (disable all POST/PATCH routes in CodeIgniter controllers, or add a maintenance banner to the login page)
2. Start 30-day freeze window clock
3. At day 30: if no rollback requests, proceed to decommission

Decommission checklist (Phase 10)

- [ ] All operators confirmed on new admin ≥ 30 days with no rollback
- [ ] Legacy admin access logs show zero operator sessions in past 14 days
- [ ] Cron jobs (`Reporting.php` scheduled deliveries) migrated to Cloudflare Cron Trigger Workers
- [ ] `application/config/adventive.php` secrets already rotated (Phase 0 — verify)
- [ ] BitBucket repo archived (not deleted)
- [ ] EC2 instance terminated (coordinate with DevOps)
- [ ] DNS record for `admin-legacy.adventive.com` removed
- [ ] Cloudflare Access application for legacy admin removed
- [ ] CI/CD pipeline for legacy repo disabled

Update — 2026-04-30: UI deploy target changed to Worker + Static Assets

The UI is no longer deploying as a **Cloudflare Pages** project. It now deploys as a **Worker with Static Assets** (`assets = { directory = "./dist", not_found_handling = "single-page-application" }` in `wrangler.toml`).

Why: Cloudflare has converged on Workers as the modern unit of deployment for both static and dynamic. A Worker-with-Assets gives us one deployment unit, one wrangler config, and a clean upgrade path when Phase 2 adds server-side routes (OAuth callbacks, edge auth glue, JWT extraction for the typed API client) to the same Worker without changing the deploy target.

Updated deployment matrix

Service	Platform	Runtime	Region
`adv-ui-admin-{env}`	Cloudflare Workers (Static Assets)	Cloudflare edge (global)	Smart placement
`adv-svc-admin-api-{env}`	Cloudflare Workers	Cloudflare edge (global)	Smart placement
`DB_CONSOLE` , `DB_BILLING` , `DB_AGGREGATE`	Cloudflare Hyperdrive	MySQL connection pooler	Collocated with DB

Updated env URL matrix

Env	UI URL	API URL	UI Worker name	API Worker name
Local	`http://localhost:5173`	`http://localhost:8787`	`vite dev`	`wrangler dev --env dev`
Dev (preview)	`genesis-admin.adventive.dev`	`admin-api.adventive.dev`	`adv-ui-admin-dev`	`adv-svc-admin-api-dev`
Staging	`admin.adventivestg.com`	`admin-api.adventivestg.com`	`adv-ui-admin-stg`	`adv-svc-admin-api-stg`
Production	`admin.adventive.com`	`admin-api.adventive.com`	`adv-ui-admin-prd`	`adv-svc-admin-api-prd`

DNS prerequisite (per standing pattern)

For each Worker custom hostname, pre-create an `AAAA` record pointing to `100::` proxied through Cloudflare. Cloudflare does not auto-create DNS for Worker route bindings.

Zone	Records to pre-create
`adventive.dev`	`genesis-admin` AAAA → `100::` proxied; `admin-api` AAAA → `100::` proxied
`adventivestg.com`	`admin` AAAA → `100::` proxied; `admin-api` AAAA → `100::` proxied
`adventive.com`	`admin` AAAA → `100::` proxied; `admin-api` AAAA → `100::` proxied

Updated CI/CD

The UI's `.github/workflows/deploy-preview.yml` now runs `wrangler deploy --env dev` (was `wrangler pages deploy`). The API's deploy workflow follows the same pattern. Rollback for the UI is `wrangler rollback --env <env>` (Workers Versions) — not Pages Deployments.

Pages-specific bits removed

- `public/_redirects` — superseded by `not_found_handling = "single-page-application"` in wrangler config.
- `wrangler pages project create ...` — replaced by `wrangler deploy` (project provisions automatically on first deploy).
- Pages Deployments concept — replaced by Workers Versions (with the same shape: per-deploy URLs, atomic promote/rollback).

`public/_headers` is retained — Workers Static Assets honors the same Pages-format headers file.

Hyperdrive provisioning gap

Phase 2 dev binds only `DB_CONSOLE` (Hyperdrive `059838c4abb64a92a4aece2a6a533a29`). `DB_BILLING` and `DB_AGGREGATE` Hyperdrives are **not yet provisioned** in any environment. Endpoints that depend on those bindings throw 503 with a clear message until provisioned. Full provisioning needs to land before stg/prd promotion. Owner: platform team / Stripe Billing Transition project.

04 — Runbook

On-call quick reference

Role	Name	Contact
Primary on-call	Jeffrey Lambert	Slack: @jeffrey
Secondary on-call	Patrick	Slack: @patrick
Escalation (Cloudflare issues)	Cloudflare Support	dash.cloudflare.com → Support
Escalation (Stripe issues)	Stripe Support	dashboard.stripe.com → Support

Dashboards: - Workers Analytics Engine: Cloudflare dashboard → Workers & Pages → adventive-admin-api → Analytics - Real-time logs: `wrangler tail adventive-admin-api --env production --format pretty` - Cloudflare Access audit log: Cloudflare dashboard → Zero Trust → Logs → Access

Status check (quick):

```
curl -s https://admin-api.adventive.com/health | jq .
# Expected: {"status":"ok","version":"x.y.z"}
```

Day-to-day procedures

Add a new operator to Cloudflare Access

1. Open Cloudflare Zero Trust dashboard → Access → Access Groups → **Adventive Operators**
2. Click **Edit** → add the operator's email to the email list → **Save**
3. If the operator is super-admin tier: update the `RBAC_SUPER_ADMIN_EMAILS` Wrangler secret and redeploy `bash wrangler secret put RBAC_SUPER_ADMIN_EMAILS --env production # Enter new comma-separated list when prompted wrangler deploy --env production`
4. If the operator is billing tier: same procedure for `RBAC_BILLING_EMAILS`
5. Send operator the admin URL and confirm they can authenticate: `https://admin.adventive.com`
6. Verify first request appears in Workers Analytics Engine (shows operator email in request log)

Revoke operator access

1. Zero Trust dashboard → Access → Access Groups → Adventive Operators → **Edit** → remove the operator's email → **Save**
2. Revoke active sessions: Zero Trust → Logs → Access → search operator email → **Revoke session**

3. If operator had billing or super-admin RBAC: remove from Wrangler secret and redeploy (same commands as add, minus their email)
4. Confirm: attempt to access `https://admin.adventive.com` from the revoked operator's browser should redirect to Access login and then deny

Deploy a hotfix

```
# 1. Make the fix on a hotfix branch, open PR, get review
git checkout -b hotfix/description
# ... make changes, commit ...
git push origin hotfix/description

# 2. After PR approval, merge to staging first
git checkout staging && git merge hotfix/description && git push

# 3. Verify fix on staging
# ... test manually or run relevant Playwright spec ...

# 4. Merge to main and trigger production deploy
git checkout main && git merge staging && git push

# 5. Monitor: watch tail logs and Analytics Engine dashboard for 10 minutes post-deploy
wrangler tail adventive-admin-api --env production --format pretty
```

For UI-only hotfixes: same flow, but only admin-ui build/deploy runs (no Worker redeploy needed).

Roll back a Worker (admin-api)

```
# See available deployments
wrangler deployments list --env production

# Roll back to previous deployment
wrangler rollback --env production

# Or roll back to a specific deployment ID
wrangler rollback [deployment-id] --env production

# Verify
curl -s https://admin-api.adventive.com/health | jq .version
```

Roll back the UI (admin-ui)

Option 1 (dashboard): Cloudflare Pages → adventive-admin-ui → Deployments tab → click the deployment you want → **Rollback to this deployment**

Option 2 (CLI):

```
wrangler pages deployment list --project-name adventive-admin-ui
wrangler pages deployment rollback [deployment-id] --project-name adventive-admin-ui
```

Rotate a Worker secret

```
# Example: rotating Stripe key after a security incident
wrangler secret put STRIPE_SECRET_KEY --env production
# Enter the new key value when prompted; takes effect on next request

wrangler secret put STRIPE_SECRET_KEY --env staging
```

No Worker redeploy is required — secrets are fetched at request time. Confirm the new key works:

```
# Hit a Stripe-dependent endpoint (e.g., invoice list)
curl -s https://admin-api.adventive.com/invoices \
  -H "Authorization: Bearer [valid-operator-token]" | jq .
```

Rotate RBAC operator lists

If operators are added or removed without a corresponding Access Group change (e.g., role change within existing operators):

```
wrangler secret put RBAC_SUPER_ADMIN_EMAILS --env production
# Enter the full updated comma-separated list

wrangler secret put RBAC_BILLING_EMAILS --env production
# Enter the full updated list
```

No redeploy required.

Roll a cohort back to the legacy admin

If an operator cohort encounters a workflow blocker in the new admin:

1. Send affected operators the legacy admin URL: `https://admin-legacy.adventive.com`
2. Confirm they can log in (JumpCloud LDAP credentials; Duo bypass currently active)
3. Create a GitHub issue documenting: the workflow that failed, the operator who reported it, reproduction steps
4. Do not disable the new admin — other cohorts may still be using it
5. Fix the issue in a hotfix branch, test in staging, re-notify the cohort once resolved

No data migration is required in either direction — both admins share the same database.

Incident playbooks

Symptom: Admin UI shows blank page or 404 for all routes

Likely cause: Pages SPA routing fallback missing, or Pages deploy failed.

Diagnosis:

```
# Check Pages deployment status
wrangler pages deployment list --project-name adventive-admin-ui | head -5

# Check the _redirects file is present in the dist output
# Should contain: /* /index.html 200
```

Fix: - If `_redirects` missing: add `public/_redirects` with content `/* /index.html 200` and redeploy - If Pages deploy failed: check GitHub Actions workflow for build error, fix, and push again

Verification: Navigate to `https://admin.adventive.com/customers` — should render the customers list, not a 404.

Symptom: All API requests returning 401

Likely cause: CF Access JWT validation failing — wrong AUD tag, expired token, or Access policy misconfigured.

Diagnosis:

```
# 1. Check the CF_ACCESS_AUD secret is correct
wrangler secret list --env production
# Verify CF_ACCESS_AUD is present

# 2. Verify the AUD matches the Access Application
# Cloudflare dashboard → Access → Applications → Adventive Admin API → Application AUD

# 3. Check the Access audit log for the failing operator
# Zero Trust → Logs → Access → filter by operator email
```

Fix: - If AUD mismatch: update the `CF_ACCESS_AUD` secret to match the Access Application's AUD tag - If operator not in Access Group: add their email (see "Add a new operator" procedure) - If token expired: operator needs to re-authenticate at `https://admin.adventive.com` — Access handles re-authentication automatically on next visit

Verification: Operator navigates to the admin UI, is redirected to Access login, authenticates, and lands on the dashboard.

Symptom: API requests returning 500 on database operations

Likely cause: Hyperdrive connection failure, MySQL server unreachable, or schema mismatch.

Diagnosis:

```
# 1. Check real-time Worker logs for the error
wrangler tail adventive-admin-api --env production --format pretty

# 2. Look for: "connect ETIMEDOUT", "Access denied", "Unknown column"
# - ETIMEDOUT: DB host unreachable or Hyperdrive misconfigured
# - Access denied: DB credentials wrong or rotated without updating Hyperdrive
# - Unknown column: schema drift — a schema change was deployed that breaks this Worker

# 3. Test DB connectivity from a known-good host (EC2 → MySQL)
```

Fix: - ETIMEDOUT: Check MySQL server status on EC2; check Hyperdrive binding in wrangler.toml points to correct host - Access denied: Rotate and re-set DB credentials in Hyperdrive configuration (Cloudflare dashboard → Workers & Pages → Hyperdrive → edit binding) - Schema mismatch: Identify the schema change, either revert it or update the Worker query to handle both old and new schema (schema-freeze policy must not be violated during transition)

Verification: `curl -s https://admin-api.adventive.com/customers | jq .total` should return a number (not an error).

Symptom: Stripe-dependent endpoints (invoices, billing) returning errors

Likely cause: Stripe API key expired or revoked, Stripe API down, or rate limit hit.

Diagnosis:

```
# 1. Check Worker logs for Stripe error codes
wrangler tail adventive-admin-api --env production --format pretty
# Look for: stripe error codes (e.g., "authentication_failed", "rate_limit_error")

# 2. Verify key is valid
# Stripe dashboard → Developers → API keys → confirm restricted key for admin-api is active

# 3. Check Stripe status
# https://www.stripestatus.com
```

Fix: - Authentication failure: rotate `STRIPE_SECRET_KEY` (see "Rotate a Worker secret" procedure) - Rate limit: Stripe rate limits are very generous for invoice reads; if hit, investigate for a loop or runaway client. Implement request-level caching if needed. - Stripe outage: no fix — surface error to operators with a clear message; fall back to read-only view if possible

---

Symptom: Operator reports "Access Denied" when navigating to a route they should have access to

Likely cause: RBAC tier misconfigured — operator's email not in the correct Wrangler secret list.

Diagnosis:

```
# 1. Check which email is in the Access JWT by inspecting the request log
wrangler tail adventive-admin-api --env production --format pretty
# Look for the request from the operator and the email extracted from the JWT

# 2. Compare against current RBAC secrets
wrangler secret list --env production
# Verify RBAC_SUPER_ADMIN_EMAILS or RBAC_BILLING_EMAILS contains the operator's email
```

Fix:

```
# Update the relevant RBAC secret to include the operator's email
wrangler secret put RBAC_BILLING_EMAILS --env production
# Enter updated comma-separated list
```

No redeploy needed. Verify: operator refreshes the admin UI and retries the route.

Symptom: Workers Analytics Engine shows no data (blank dashboard)

Likely cause: Analytics binding misconfigured in `wrangler.toml`, or `writeDataPoint` calls not executing.

Diagnosis:

```
# Check that the ANALYTICS binding is defined in wrangler.toml
grep -A3 "analytics_engine_datasets" admin-api/wrangler.toml

# Check if Worker is emitting events by tailing and triggering a request
wrangler tail adventive-admin-api --env production --format pretty
```

Fix: If binding is missing or misconfigured, add it to `wrangler.toml` and redeploy. Analytics Engine data appears with a ~1-minute delay; wait before concluding there's an issue.

Symptom: Cloudflare Pages build failing

Likely cause: TypeScript errors, missing environment variables, or dependency issues.

Diagnosis: - Check GitHub Actions workflow run for the build step output - Common failures: `VITE_API_BASE_URL` not set in Pages environment, generated API types out of sync with openapi.json

Fix: - Set missing env var in Cloudflare Pages dashboard → Settings → Environment variables - Regenerate API types and commit: `cd admin-ui && npm run generate:types && git add src/api/generated.ts && git commit -m "chore: regenerate API types"`

Investigate a failed Access login

When an operator reports they cannot authenticate to the admin:

1. **Check Access audit log:** Zero Trust → Logs → Access → filter by operator email and time range
2. **Common outcomes to look for:** - `ALLOW` with a block at the application level → operator email not in Access Group - `BLOCK` with reason "Policy: Email not in list" → same; add email to group - `BLOCK` with reason "MFA required" → operator's identity provider session does not have MFA — operator needs to re-authenticate with MFA on their IDP account - `BLOCK` with reason "Revoked session" → manually revoked — investigate why; re-add if legitimate
3. **If none of the above:** Ask operator to clear browser cookies for `adventive.cloudflareaccess.com` and retry. If still failing, escalate to Cloudflare Support with the CF-Ray ID from the blocked request.

Decommission the CodeIgniter admin (Phase 10)

Perform only after the 30-day freeze window with zero operator rollbacks and zero legacy admin traffic.

```
Pre-decommission verification:
1. Pull legacy admin access logs from EC2: zero operator sessions in last 14 days
2. Confirm all Cron Trigger Workers are live and delivering scheduled reports
   (formerly Reporting.php — verify each report type is being generated)
3. Confirm override file S3 → R2 migration is complete
4. Confirm Phase 0 secret rotation is complete (adventive.php has no live credentials)

Shutdown sequence:
1. Disable legacy admin login page (add maintenance notice: "This system has been retired.
   Use https://admin.adventive.com")
2. Set all legacy admin routes to return 410 Gone
3. Let sit for 7 days — monitor for any automated callers hitting legacy URLs
4. Archive BitBucket repo: Settings → Archive repository
5. Terminate EC2 instance (coordinate with Patrick for infra sign-off)
6. Remove DNS record for admin-legacy.adventive.com
7. Remove Cloudflare Access application for legacy admin
8. Remove Bitbucket Pipelines deploy pipeline (or disable CI)
9. Update README.md in this planning folder: set Status = Decommissioned, Date = [today]
```

Known issues & workarounds

Issue	Workaround	Permanent fix
Schema-freeze policy blocks column renames	Use dual-write pattern: write to both old and new column names until legacy admin is decommissioned	Decommission legacy admin (Phase 10)

Issue	Workaround	Permanent fix
PHPExcel (abandoned) still used for month-end Excel export	Legacy admin continues to generate until Cron Trigger Worker replacement is live	Port month-end summary to phpspreadsheet or generate CSV via Cron Trigger Worker
Duo 2FA currently bypassed on legacy admin	Legacy admin only accessible during parallel-run; Cloudflare Access enforces MFA for new admin	Decommission legacy admin (Phase 10)
CF Access directory provider not yet chosen (JumpCloud vs. Google Workspace)	Use email allowlist as interim identity strategy	Resolve identity provider ADR before Phase 1 completes
Report_model.php (5,333 lines, 24 bespoke SQL functions) not ported	Partner reports continue to run from legacy Cron jobs during Phases 1–8	Phase 4: evaluate dedicated analytics Worker or Cloudflare Analytics product

05 — Appendix

References

Cloudflare documentation

Resource	URL	Relevance
Cloudflare Workers docs	https://developers.cloudflare.com/workers/	Primary runtime for admin-api
Cloudflare Pages docs	https://developers.cloudflare.com/pages/	Hosts admin-ui SPA
Cloudflare Access docs	https://developers.cloudflare.com/cloudflare-one/policies/access/	Operator authentication at edge
Cloudflare Hyperdrive docs	https://developers.cloudflare.com/hyperdrive/	MySQL connection pooling from Workers
Cloudflare R2 docs	https://developers.cloudflare.com/r2/	Override file storage (replaces S3)
Workers Analytics Engine	https://developers.cloudflare.com/analytics/analytics-engine/	Structured observability from Workers
Wrangler CLI docs	https://developers.cloudflare.com/workers/wrangler/	Deploy, dev, secret management
Workers Cron Triggers	https://developers.cloudflare.com/workers/configuration/cron-triggers/	Scheduled report delivery replacement
CF Access service tokens	https://developers.cloudflare.com/cloudflare-one/identity/service-tokens/	Playwright E2E auth

Framework and library documentation

Package	URL	Purpose
Hono	https://hono.dev	TypeScript HTTP framework for Workers
@hono/zod-openapi	https://hono.dev/snippets/zod-openapi	OpenAPI spec generation from Zod schemas
Zod	https://zod.dev	Runtime schema validation

Package	URL	Purpose
TanStack Router	https://tanstack.com/router	Type-safe SPA routing
TanStack Query	https://tanstack.com/query	Server-state management, caching
shadcn/ui	https://ui.shadcn.com	UI component primitives
Radix UI	https://www.radix-ui.com	Accessible headless component primitives
Tailwind CSS	https://tailwindcss.com	Utility-first CSS
Vitest	https://vitest.dev	Unit + integration testing
Playwright	https://playwright.dev	E2E testing
Stripe Node SDK	https://github.com/stripe/stripe-node	Stripe API (Workers-compatible)
openapi-typescript	https://openapi-ts.dev	UI type generation from OpenAPI spec

Related internal projects

Project	Path	Relationship
Public API Cloudflare Migration	`../Public API Cloudflare Migration/`	Predecessor project — auth model, deployment pipeline, observability approach mirror this project
Stripe Billing Transition	`../Stripe Billing Transition/`	Sibling project — billing surface area of this admin is co-designed with that project's decisions
Adventive Brand Guide	https://brandguide.adventive.com/dashboard	Visual and interaction language reference for admin-ui

Inputs

Files analyzed during this planning engagement:

File	Description	Source	Date
`application/controllers/Auth.php`	JumpCloud LDAP auth + Duo 2FA controller	adventive-admin repo	2026-04-23

File	Description	Source	Date
`application/controllers/Dashboard.php`	Dashboard aggregation controller	adventive-admin repo	2026-04-23
`application/controllers/Account.php`	Account CRUD, user management, service levels	adventive-admin repo	2026-04-23
`application/controllers/Billing.php`	Invoice generation, payment, managed services	adventive-admin repo	2026-04-23
`application/controllers/Campaign.php`	Campaign and ad management	adventive-admin repo	2026-04-23
`application/controllers/Report.php`	On-demand reporting (22 methods)	adventive-admin repo	2026-04-23
`application/controllers/Reporting.php`	Scheduled report delivery (25 cron methods)	adventive-admin repo	2026-04-23
`application/controllers/Tools.php`	Admin tooling: lookup, Cognito sync, deploy	adventive-admin repo	2026-04-23
`application/controllers/Override.php`	Override file versioning (S3)	adventive-admin repo	2026-04-23
`application/controllers/Chart.php`	Chart JSON endpoints	adventive-admin repo	2026-04-23
`application/controllers/ManagedServiceJob.php`	Managed service job CRUD	adventive-admin repo	2026-04-23
`application/controllers/Utility.php`	CLI utilities	adventive-admin repo	2026-04-23
`application/models/Account_model.php`	Account queries (1,321 lines)	adventive-admin repo	2026-04-23
`application/models/Billing_model.php`	Billing queries (933 lines)	adventive-admin repo	2026-04-23
`application/models/Campaigns_model.php`	Campaign queries (665 lines)	adventive-admin repo	2026-04-23
`application/models/Report_model.php`	Partner report queries (5,333 lines, 24 functions)	adventive-admin repo	2026-04-23

File	Description	Source	Date
`application/models/ManagedService_model.php`	Managed service job queries (505 lines)	adventive-admin repo	2026-04-23
`application/models/Stats_model.php`	Stats queries (1,286 lines)	adventive-admin repo	2026-04-23
`application/config/adventive.php`	Application config including hardcoded secrets	adventive-admin repo	2026-04-23
`application/config/config.php`	CI framework config (CSRF disabled at line 320)	adventive-admin repo	2026-04-23
`composer.json`	PHP dependency manifest	adventive-admin repo	2026-04-23
`application/views/`	56 view files across all controller domains	adventive-admin repo	2026-04-23

Open questions

The following items require a human decision before or during implementation. Items marked **BLOCKING** must be resolved before the indicated phase begins.

#	Question	Blocking phase	Owner	Notes
OQ-1	**Identity provider for Cloudflare Access: JumpCloud vs. Google Workspace?**	Phase 1 (before operators can authenticate)	Jeffrey	Deferred by ADR — but a decision must land before Phase 1 completes. Google Workspace is already in use for email; JumpCloud is the current LDAP provider.
OQ-2	**Secret rotation scope: which AWS keys are actively used, and by which processes?**	Phase 0 (prerequisite)	Jeffrey + Patrick	Hardcoded keys include AWS S3 and IAM credentials. Need a full inventory before rotation to avoid breaking the legacy admin during the parallel-run period.
OQ-3	**Cloudflare account ID and API token provisioning: who owns this?**	Phase 1	Patrick	The GitHub Actions workflows need `CLOUDFLARE_API_TOKEN` and `CLOUDFLARE_ACCOUNT_ID` as repository secrets.

#	Question	Blocking phase	Owner	Notes
OQ-4	**New GitHub repositories: where do they live?**	Phase 1	Jeffrey	The plan assumes `adventive/adventive-admin-api` and `adventive/adventive-admin-ui` on GitHub. Confirm organization name and visibility (private).
OQ-5	**Hyperdrive configuration: which DB host and credentials?**	Phase 1	Patrick	Hyperdrive needs a MySQL connection string pointing to the production/staging DB. Confirm the DB host is reachable from Cloudflare Workers (may require allowlisting Cloudflare IP ranges).
OQ-6	**Custom domain provisioning: admin.adventive.com and admin-api.adventive.com**	Phase 1 (for staging) / Phase 3 (for production)	Jeffrey	Both domains must be added to Cloudflare Pages / Workers as custom domains. DNS must point to Cloudflare.
OQ-7	**Partner report replacement strategy: dedicated analytics Worker, Cloudflare Analytics, or third-party?**	Phase 4	Jeffrey + Patrick	Report_model.php is 5,333 lines with 24 bespoke SQL query functions. This is the largest single scope item and cannot be ported inline. Requires a separate architectural decision.
OQ-8	**Stripe Billing Transition timing: when is Phase B stable?**	Phase 4 (Cohort 2 migration gate)	Jeffrey + Patrick	Cohort 2 (billing operators) should not migrate before Stripe Billing Transition Phase B is stable. Coordinate cutover timing.
OQ-9	**Invoice renderer: Cloudflare Browser Rendering vs. external container?**	Phase 4	Patrick	From Stripe Billing Transition: PDF rendering in Workers is constrained. Browser Rendering is the leading candidate but has cost and concurrency implications at invoice volume.
OQ-10	**Metering signal contract for Stripe Billing Transition:** format, transport, idempotency guarantees	Phase 4	Jeffrey + Patrick	The admin-api will surface billing data from Stripe, but the metering signal going *into* Stripe is owned by the Stripe Billing Transition project. The contract must be stable before billing surface is built in admin-api.

#	Question	Blocking phase	Owner	Notes
OQ-11	**Cognito sync tooling: retain or retire?**	Phase 8	Jeffrey	`application/libraries/AwsCognito.php` and the Cognito sync in `Tools.php` may be superseded by the Public API migration or another identity flow. Needs evaluation before tooling cohort is migrated.
OQ-12	**AppDeploy tooling: retain or retire?**	Phase 8	Jeffrey + Patrick	`application/libraries/AppDeploy.php` wraps AWS CodeDeploy. If the legacy EC2 deploy pipeline is retired before Phase 8, this tool becomes irrelevant. Track alongside the EC2 decommission timeline.

Glossary

Term	Definition
Access Group	A Cloudflare Zero Trust construct that defines a set of users (by email, IDP group, or other attribute) who can match a policy rule. The "Adventive Operators" Access Group contains all ~10 operator email addresses.
AUD tag	"Application Audience" — a unique identifier string attached to a Cloudflare Access Application. The admin-api Worker validates that incoming JWTs contain the correct AUD tag, preventing tokens issued for other Access Applications from being used against this API.
CF-Access-Jwt-Assertion	HTTP request header set by Cloudflare Access on every authenticated request. Contains a signed JWT with the operator's identity (email, identity provider, expiry). The admin-api reads this header to identify the operator without an application-managed session.
Cohort	A group of operators migrated together to the new admin in a single wave. Three cohorts are planned: (1) account management, (2) billing, (3) remaining workflows.
Cron Trigger Worker	A Cloudflare Workers feature that executes a Worker on a cron schedule. Used as the replacement for `Reporting.php`'s 25 scheduled report delivery methods.
Hyperdrive	Cloudflare's MySQL (and PostgreSQL) connection pooling and caching layer for Workers. Workers cannot hold persistent TCP connections; Hyperdrive maintains a connection pool on Cloudflare's infrastructure and proxies queries.

Term	Definition
RBAC	Role-Based Access Control. In this project: three tiers — super-admin (2 operators, all access), billing (billing domain read+write, account read), read-only (all surfaces, no mutations). Tier is derived at runtime from the operator's email address against Wrangler secret lists.
Schema-freeze policy	During the parallel-run period (Phases 1–9), no schema changes that break backward compatibility with the CodeIgniter admin are permitted. New nullable columns with defaults are allowed; column drops, renames, and type changes are blocked until the legacy admin is decommissioned.
Service token	A Cloudflare Access credential (client ID + client secret) that allows non-human callers (Playwright E2E tests, CI pipelines) to authenticate to an Access-protected application without a browser-based identity provider flow.
Smart placement	A Cloudflare Workers feature that routes Worker invocations to the data center geographically closest to the bound backend (in this case, the MySQL database). Reduces latency for database-heavy requests without manual region configuration.
Wrangler	Cloudflare's CLI tool for developing, deploying, and managing Workers and Pages projects. Used for deploys, secret management, tail logging, and local development.
Worker	A Cloudflare serverless function. The `admin-api` Worker handles all API requests; the `invoice-renderer` Worker (sibling project) handles PDF generation.
Stripe Billing Transition	The sibling project that migrates Adventive's billing pipeline from the custom admin-managed rollup to Stripe Billing as the system of record. The admin's billing surface is co-designed with this project's decisions.
Parallel run	The period during which both the legacy CodeIgniter admin and the new admin-api/admin-ui are live simultaneously, pointing at the same database. Operators migrate cohort-by-cohort; either admin can be used by any operator until their cohort is fully migrated.
R2	Cloudflare's S3-compatible object storage. Used in this project as the replacement for AWS S3 for override file versioning.
Override file	An operator-managed configuration file (stored in S3/R2) that overrides platform defaults for specific accounts. Versioned; operators can view history and roll back.
Managed service job	A manually-created billing line item attached to a customer account for managed services (e.g., campaign management fees). Separate from subscription-based charges.

Decisions log (cross-reference)

All formal decisions are in `decisions/`. Summary of locked-in decisions referenced throughout this document:

Decision	Summary	File
ADR-1	Operator auth = Cloudflare Access. Directory provider deferred.	`decisions/2026-04-XX-cloudflare-access-auth.md`
ADR-2	UI stack = React + Vite + TypeScript SPA on Cloudflare Pages	`decisions/2026-04-XX-ui-stack.md`
ADR-3	API stack = TypeScript + Hono on Cloudflare Workers, OpenAPI via @hono/zod-openapi	`decisions/2026-04-XX-api-stack.md`
ADR-4	Deployment = parallel run, cohort-by-cohort operator migration	`decisions/2026-04-XX-deployment-strategy.md`
ADR-5	No CodeIgniter retrofit — legacy app inventoried, not modified	`decisions/2026-04-XX-no-retrofit.md`
ADR-6	Data store = existing DB, no migration in scope	`decisions/2026-04-XX-data-store.md`
ADR-7	RBAC = super-admin (2), billing, read-only; ~10 operators	`decisions/2026-04-XX-rbac.md`

Change log

Version	Date	Author	Change
1.0	2026-04-23	Jeffrey Lambert	Initial planning deliverable — repo analysis complete; all chapters populated; PDF generated

06 — Phase 1 Prototype

Status: shipped 2026-04-30. Live preview at `genesis-admin.adventive.dev` (deploys as `adv-ui-admin-dev` Worker with Static Assets).

Purpose

Phase 1 ships a visually functional UI prototype before the API is built. The goals are pragmatic, not architectural:

1. Validate the brand language and information architecture with the operator team before Phase 2 commits to a backend contract.
2. Prove the chosen stack (Vite + React + TS + Tailwind + shadcn primitives + TanStack Router/Query) works end-to-end at Cloudflare Worker scale.
3. Surface the surfaces — give Phase 2 a concrete UI shape to reverse-engineer the API contract from, instead of speccing endpoints in the abstract.
4. Establish brand tokens, component primitives, and the layout shell that production will inherit verbatim.

The prototype is **not** a throwaway. It IS the v1 of `adventive-admin-ui`. Phase 2 swaps the data layer (`src/lib/mock/*` → TanStack Query hooks against `adventive-admin-api`) and adds Cloudflare Access; everything else carries forward.

What's in the box

15 routes, all rendering against in-memory mock fixtures:

Dashboard (`/`) — Operator landing page. Four KPI tiles (MTD/QTD/YTD/LTD revenue), a 12-month stacked-area revenue trend (Recharts), recent invoices list, managed-services queue snapshot, and a Quick Actions panel with deep links into open invoices, delinquent, Cognito sync, and user lookup.

Accounts list (`/accounts`) — Status-filter chips (active / trial / delinquent / paused / cancelled / all) with running counts, search box, exportable table with MRR, open balance, delinquent total, user count, activation date, and a per-row open link to detail.

Account detail (`/accounts/$accountId`) — Header KPI tiles (MRR, LTV, open invoices, delinquent) plus a tabbed surface with all 9 tabs the legacy `Account/detail` page exposes:

- Settings — Customer info, billing address, main contact, billing & payment terms, Adventive team
- Users — table of account users with role, created/activated/last-login dates
- Campaigns — list with advertiser, status, dates, ad-unit count, 30-day impressions and clicks
- Ad Units — list with format, advertiser, recipe UUID, status, 30-day impressions
- Advertisers — list with campaign count, created date
- Billing History — invoice list with status, dates, amount, PDF/CSV links
- Permissions & Entitlements — entitlement catalog with per-account override state
- Plan Info — tier, monthly price, contract dates, auto-renew, seats vs. seat limit, feature bundle

- Managed Services — job list with type, assignee, status, hours logged vs. budgeted, due date

Billing surfaces — Overview with KPIs + trend/breakdown chart toggle + collections progress bar + aging snapshot; Invoices list with status filters and search; Account Revenue History (12-month per-account grid with MTD, average, lifetime sales); Processing Controls (test invoice, real cycle, QB export, commission report, dunning sweep, year-end rollover) with destructive-action badges and last-run timestamps; Delinquent invoices with dunning batch and call-log actions; Aging Summary with bucketed bar chart and per-account aging matrix.

Tools — Preview Link Lookup (with QR + open buttons per result), User Lookup (cross-account search), Ad Type Utilization (horizontal bar chart + table with trend arrows), Ad Type Performance (impressions, viewability, CTR, video completion, engagement), Cognito Sync (with discrepancy callouts and per-row reconcile actions), Special Permissions catalog.

Stack — proven decisions

Concern	Choice	Why locked
Build	Vite 8	Already used elsewhere in the org; rolldown bundler
Framework	React 19	Aligns with existing engineering skills
Language	TypeScript	Required for OpenAPI-driven type generation in Phase 2
Styling	Tailwind 3 (NOT Tailwind 4)	shadcn/ui ecosystem still on v3; v4 migration deferred
Component primitives	shadcn-flavored on Radix	Direct lift from `brandguide.adventive.com/dashboard`
Routing	TanStack Router (code-based)	Type-safe, file-based optional, no build-step required for prototype
Server state	TanStack Query	Phase 2 swap target; mocks today, hooks tomorrow
Charts	Recharts	Used in dashboard, billing overview, aging
Icons	lucide-react	shadcn/ui's standing pair
Font	Inter (via rsms.me CDN)	Brand guide standard

Brand tokens

The prototype's tokens are HSL CSS variables lifted directly from `brandguide.adventive.com/dashboard` via DOM inspection. They live in `src/index.css` and Tailwind reads them through the `tailwind.config.ts` theme block.

```
--background: 230 25% 7%;        /* deep navy-black */
--foreground: 210 40% 98%;       /* near-white */
--primary:    250 100% 70%;      /* Adventive purple */
--card:       230 25% 10%;
--accent:     230 25% 15%;
--border:     230 25% 18%;
--radius:     0.75rem;
--stripe-purple: 250 100% 66%;
--stripe-pink:   330 100% 71%;
--stripe-blue:   217 91% 60%;
```

The dark theme is the default and only theme for now. A light variant is deferred until operator preference data is collected post-rollout.

Mock data layer — Phase 2 swap surface

Mock fixtures live in `src/lib/mock/`:

- `types.ts` — TypeScript types matching the API contract (Account, Invoice, Campaign, AdUnit, etc.)
- `accounts.ts` — 8 fully-realized accounts across all status tiers, with related users, campaigns, ad units, advertisers, managed-services jobs, permissions, and plan info
- `billing.ts` — invoices, KPIs, 12-month revenue trend, aging buckets, processing controls, account-history rows
- `tools.ts` — ad type utilization/performance, Cognito sync samples, preview link samples
- `operator.ts` — current operator and roster

Pages import from these modules directly today. Phase 2's swap pattern:

```
// Phase 1
import { accounts } from "@/lib/mock/accounts";
const data = accounts.filter((a) => a.status === "active");

// Phase 2
import { useAccounts } from "@/lib/api/hooks";
const { data, isLoading, error } = useAccounts({ status: "active" });
```

The mock module shape is intentionally aligned with the API contract — same field names, same types, same nullability. The swap is mechanical, not architectural.

Deploy

Worker name: `adv-ui-admin-dev`. Hostname: `genesis-admin.adventive.dev` (AAAA pre-creation required). Built with `npm run build`, deployed with `npm run deploy:dev`. Cloudflare Access is **not** yet enforced on the dev preview — the URL is unprotected during the team-review window. Access is added when the API lands and Phase 2 promotes to staging.

What Phase 2 inherits

- Layout shell (`src/components/admin/admin-shell.tsx` , `sidebar.tsx` , `top-bar.tsx` , `page-header.tsx`)
- All 15 route components — each one is the production view, just with mock data
- All shadcn-flavored primitives (`src/components/ui/*`)
- KpiCard, StatusPill — domain-specific composites that already encode brand and copy
- Brand tokens, font, CSS variables
- The OpenAPI-shape mock contract

What Phase 2 adds

- The typed API client at `src/lib/api/client.ts` (generated from `adventive-admin-api` 's `openapi.json`)
- TanStack Query hooks at `src/lib/api/hooks/*` replacing `src/lib/mock/*`
- Cloudflare Access integration on the UI Worker route
- Auth context provider that reads `/api/me` once on mount and caches the operator role for RBAC-conditional UI
- An "as-built" pass updating brand tokens from any review feedback that lands during the meeting

What's deliberately not in Phase 1

- Real authentication — Cloudflare Access defers to Phase 2 promotion
- Real data — every list, detail, KPI is a fixture
- Mutating actions — buttons render but trigger no network calls. The "Run April billing" button on the dashboard, for example, is decorative
- The full Reports surface — those endpoints are not in admin-api's scope and the corresponding UI is intentionally omitted (decision in §02)
- Dark/light theme toggle — dark only
- Mobile breakpoints below `lg` — sidebar disappears, but the prototype is desktop-first by design

Repo

Local: `~/Repositories/GitHub/Adventive/adventive-admin-ui/` — initial commit 2026-04-30. Pushed to GitHub by Jeffrey post-review.