zapier.georgelambert.org · Markdown indexes
VERAE TIME × ZAPIER

zappier-edge README (the billing/user platform)

site/packages/zappier/README.md

Zappier

Metered API platform: per-endpoint pricing, customer types with multipliers and monthly credits, a usage ledger, Stripe metered billing, purchase-order invoicing, a company admin console, a self-service customer portal, and a Zapier integration.

This is the **billing and user platform** (originally the standalone `zappier` git tree at `~/zappier`). Forgejo name: **zappier-edge**. It owns signup, TOTP, API keys, rate card, Stripe meter, PO invoices, admin users, and the customer portal. Verae middleware does not replace this.

Forgejo: https://git.georgelambert.org/marchon/zappier-edge
Catalog README: https://zapier.georgelambert.org/packages/zappier/README.pdf
Bring the whole system online: https://zapier.georgelambert.org/packages/verae-ops/GETTING-STARTED.pdf

Documentation (public PDFs)

- USER-MANUAL — operations & usage
- ACCOUNTING — invoices, PO billing, reports
- USER-MANAGEMENT — pricing, customer types
- CUSTOMER-PORTAL — signup, 2FA, reloads
- DEVELOPER
- WALKTHROUGH
- Markdown copies remain next to these files in `docs/`
- **Sample Zapier app:** `zapier-app/` — https://zapier.georgelambert.org/packages/zappier/zapier-app/README.pdf (if present) or the source tree in git

Surfaces

Surface	URL	Audience
Public API	`/v1/*`	API customers (`x-api-key`)
Interactive API docs	`/docs`	Integrating developers
Admin console	`/admin`	Company ops & accounting
Customer portal	`/portal`	End-user customers (signup, 2FA, billing)
Zapier app	`zapier-app/`	No-code users via Zapier

Pricing model

`openapi.yaml` defines the API surface; each `operationId` is a rate-card key. Endpoints carry **list prices** (seed: `src/pricing.ts` → `DEFAULT_RATE_CARD`). Customer types are **tier configs** (`DEFAULT_TIERS`) with a `multiplier` , a `monthlyCreditCents` quota, and an optional `defaultRule` for endpoints not on the card. Individual customers can carry a `multiplierOverride` . Billed price = `round(list price × multiplier)` ; usage up to the monthly credit is free. Pricing is editable at runtime in the admin console.

Seed rate card (list prices, cents per call)

Endpoint	Model	List price
`status`	free	0
`storage-list`	free	0
`transform`	fixed	4
`storage`	variable	10 + 1 per KB metadata + 50 per MB attached

Seed customer types

Tier	Multiplier	Monthly credit	Default rule (unlisted endpoints)
`free`	1.0	100 cents	none — call rejected with 403
`pro`	0.5	1000 cents	fixed 8 list → 4 billed
`business`	0.25	10000 cents	fixed 8 list → 2 billed

Adding a new API call = add it to `openapi.yaml`, then price it in the admin UI. Adding a customer type = create it in the admin UI. Variable pricing = base per call + metadata size (rounded up to KB) + attachment size (rounded up to MB), then the multiplier.

Quickstart

```
npm install
npm run dev
```

The server starts on port 3000. API docs at http://localhost:3000/docs, admin console at `/admin`, customer portal at `/portal`.

Deploying

```
npm ci && npm run build
node dist/index.js       # runs from ANY working directory
```

All runtime paths (SQLite default, `.env`, OpenAPI spec, static assets) resolve from the installation root, so the compiled server works under systemd, Docker, or cron regardless of cwd. `PORT` and `ZAPPIER_DB` remain environment-overridable.

Environment variables

Variable	Default	Purpose
`PORT`	`3000`	HTTP port the server listens on
`ZAPPIER_DB`	`<root>/zappier.db`	SQLite database file path
`ADMIN_KEY`	`admin-dev-key`	Admin UI / admin API key — **set a real secret in production**
`ADMIN_USER`	`admin`	Admin UI primary login username
`DEMO_ADMIN_USER`	`demo`	Admin UI demo login username
`DEMO_ADMIN_PASSWORD`	`$$$Adm1n###`	Demo login password — **override in production**
`STRIPE_SECRET_KEY`	*(none)*	Stripe secret key — billing job and portal reloads

Billing

Usage is reported to Stripe by a job (loads `STRIPE_SECRET_KEY` from `.env`):

```
npx ts-node src/jobs/report-usage.ts
```

The job sums each customer's usage since the first of the current month (UTC), applies the tier's monthly credit, and reports only the **delta** above what was already reported — re-runs are safe. Idempotency comes from three layers: a `billing_reports` ledger (cumulative cents per customer per month), an atomic `job_locks` run guard (1 h TTL), and a deterministic Stripe event `identifier` (`customer:period:billable`) that dedupes crash retries. It requires a Stripe meter named `zappier.api_cents` with Sum aggregation over the `value` field.

A Kimi cron job ("Zappier billing · report usage to Stripe") runs it daily at 06:17 America/New_York with a completion notification.

Purchase-order customers are invoiced manually from the admin console (**Invoices** tab); prepaid balances from the customer portal are drawn down automatically at invoice issue. See `docs/ACCOUNTING.md`.

Zapier app

The companion Zapier integration lives in `zapier-app/`:

```
cd zapier-app
npm install
npm test
```

To deploy it, create a Zapier developer account, run `zapier login`, then `zapier push` from the `zapier-app/` directory.

Testing

```
npm test                    # root API/service suite (jest, 165 tests)
cd zapier-app && npm test   # Zapier integration suite (mocha, 4 tests)
```