AWS Marketplace Authentication Gateway + Metering Kit

Self-hosted Node.js gateway for SaaS products listed on AWS Marketplace

The Problem

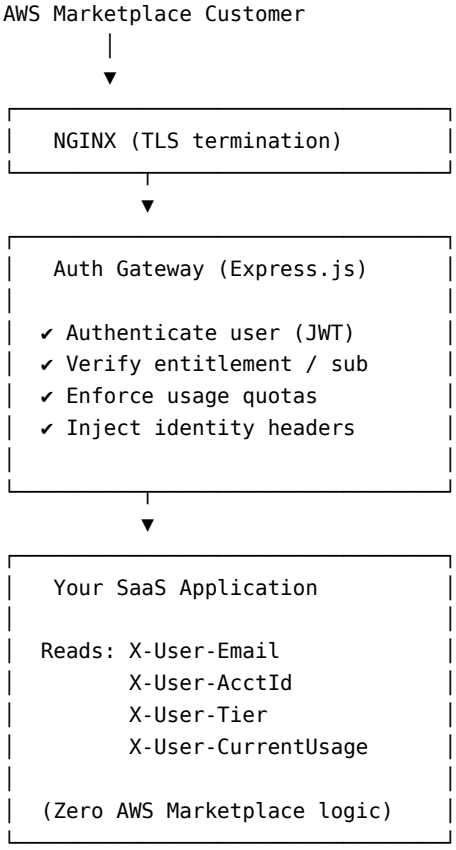
Every SaaS product listed on AWS Marketplace needs the same plumbing: customer onboarding via `ResolveCustomer`, user authentication, entitlement and subscription gating, and usage metering. Teams rebuild this from scratch each time — and getting metering semantics wrong means lost revenue or billing disputes.

The Solution

A production-tested, self-hosted Node.js gateway that sits between AWS Marketplace and your SaaS application. It handles the entire customer lifecycle — from first click in the AWS Marketplace console to hourly usage billing — so you can focus on your product.

This is the same code I have used in production to ship 4 AWS Marketplace SaaS products.

How It Works



Your upstream application receives authenticated, authorized requests with identity and billing context injected as HTTP headers. It needs **zero knowledge** of AWS Marketplace.

Features

Customer Onboarding

- **ResolveCustomer fulfillment** — Handles the AWS Marketplace POST redirect, validates the marketplace token, creates a tenant record, and redirects to admin signup
- **Subscription lifecycle via SQS** — Polls for `subscribe-success`, `unsubscribe-success`, and other lifecycle events; updates tenant status automatically
- **Race condition handling** — SQS events that arrive before a customer visits the registration page are reconciled at registration time

Authentication & User Management

- **JWT-based auth** — httpOnly, Secure, SameSite=Strict cookies (never exposed to JavaScript)
- **Multi-tenant org admin panel** — Invite users by email, assign roles (admin/user), remove users
- **Invitation flow** — Cryptographic token-based email invitations with configurable expiry
- **Password reset** — Token-based reset via email with one-time-use enforcement
- **Login audit trail** — Every login is recorded with IP and user agent

Entitlement & Subscription Gating

Two billing models, one toggle:

	Contract (PAYG=false)	Pay-As-You-Go (PAYG=true)
Entitlement check	AWS `GetEntitlements` API (cached)	None
Quota enforcement	Tier-based thresholds (monthly or lifetime)	None — billed by usage
Middleware chain	`auth → entitlement → balance → proxy`	`auth → subscription → proxy`
Tier config	JSON file (no code changes to add tiers)	N/A

Contract model tiers are configured in plain JSON:

```
{
  "tiers": [
    { "name": "free",       "threshold": 100,   "period": "monthly" },
    { "name": "basic",      "threshold": 1000,  "period": "monthly" },
    { "name": "enterprise", "threshold": 99999, "period": "lifetime" }
  ]
}
```

Usage Metering

- **Usage ingestion API** — `POST /usage` accepts events from your app (secured by API key)
- **Hourly BatchMeterUsage** — Aggregates and submits usage to AWS Marketplace automatically
- **Zero-usage heartbeats** — Sends 0-quantity records for active subscribers with no usage (AWS compliance)
- **Dimension auto-discovery** — Fetches metering dimensions from the AWS Marketplace Catalog at startup
- **Full audit trail** — Every metering submission (including failures) is recorded with the AWS `MeteringRecordId`
- **Backfill support** — Manual `meterForPeriod()` for resubmitting historical windows

Gateway / Reverse Proxy

- **Transparent proxying** — All non-gateway requests are forwarded to your upstream application
- **Identity injection** — `X-User-Email`, `X-User-AcctId`, `X-User-Tier`, `X-User-CurrentUsage` headers
- **Public path whitelist** — Configure paths (e.g., webhooks) that bypass authentication entirely
- **Original URL preservation** — Upstream sees the full original request path

Built-In Control Panel (Frontend)

- Admin dashboard — invite/remove users, manage roles
- User login, signup, password reset
- Profile page with role badges and password change
- Contact support form
- Built with SvelteKit 2 + Svelte 5 + Material UI

Security

- **httpOnly + Secure + SameSite cookies** — Auth tokens never accessible to client-side JavaScript
- **Single-use registration sessions** — Prevents replay attacks on the fulfillment flow
- **bcrypt password hashing** — With configurable cost factor
- **Startup validation** — Server refuses to start with missing or invalid configuration
- **NGINX hardening** — Blocks WebDAV, dotfiles, and database/config file extensions at the edge

Technology Stack

Layer	Technology
Backend	Node.js + Express 5
Database	SQLite (better-sqlite3, WAL mode)
Frontend	SvelteKit 2, Svelte 5, TypeScript, Material UI
Infrastructure	NGINX reverse proxy + systemd
AWS SDKs	Marketplace Metering, Entitlement Service, Catalog, SES, SQS (all v3)
Email	AWS SES with Pug HTML templates
Logging	Winston with daily rotation

What You Get

- Full source code (private repo access)
- Backend + frontend, production-ready
- NGINX configuration (dev + prod)
- systemd service file
- Database schema and creation scripts
- Jest + Vitest + Playwright test suites
- Environment configuration templates

What It Does NOT Do

- **SSO / OIDC** — Not included (available as optional add-on)
- **Stripe billing** — AWS Marketplace billing only
- **Multi-region deployment** — Single-instance deployment (you can replicate if needed)

Deployment

Runs in your VPC. No external SaaS dependencies. No data leaves your infrastructure (except AWS Marketplace API calls).

Requirements

- Node.js runtime
- NGINX (TLS termination)
- AWS account with Marketplace product listing
- AWS SES (for transactional emails)
- AWS SQS queue (for subscription lifecycle events)

Infrastructure footprint: Single server — SQLite means no database server to manage.

Who It's For

- **ISVs listing on AWS Marketplace** who don't want to spend weeks rebuilding fulfillment, auth, and metering
- **Teams shipping multiple Marketplace products** who want a reusable foundation
- **Solo founders and small teams** who need production-grade Marketplace integration without the overhead

Self-hosted. Source included. Battle-tested across 4 production AWS Marketplace SaaS products.