



Hanzo Base: An Open-Source Backend Runtime with Embedded Database, Authentication, and JavaScript Plugin System

Zach Kelling

Hanzo AI

December 2022

Abstract

We present Hanzo Base, an open-source backend-as-a-service runtime that packages an embedded database, real-time subscriptions, file storage, and authentication into a single Go binary. Base extends the PocketBase architecture with three key additions: (1) a Goja-based JavaScript plugin system enabling server-side scripting without recompilation, (2) native Hanzo IAM integration for federated identity across the Hanzo ecosystem, and (3) dual-database support for both SQLite (edge deployments) and PostgreSQL (production clusters). Base achieves sub-millisecond response times for CRUD operations while supporting collections with custom validation rules, cascading hooks, and an auto-generated admin dashboard. In production at Hanzo since December 2022, Base serves as the default backend for over 40 internal applications with a median p99 latency of 2.3ms.

1 Introduction

Modern application development requires assembling authentication, database access, file storage, and real-time capabilities from disparate services. Backend-as-a-service (BaaS) platforms like Firebase and Supabase address this by providing integrated stacks, but they impose vendor lock-in and limit customization. Self-hosted alternatives such as PocketBase [1] offer a compelling single-binary approach but lack extensibility mechanisms for production workloads.

Hanzo Base bridges this gap by forking PocketBase and adding three capabilities essential for production use: a JavaScript plugin runtime, federated identity integration, and PostgreSQL support. The result is a backend runtime that can be deployed as a single binary for prototyping or as a clustered service for production, with the same API surface in both modes.

Contributions.

- A Goja-based JavaScript VM embedded in¹ a Go binary, supporting hooks, routes, and²³

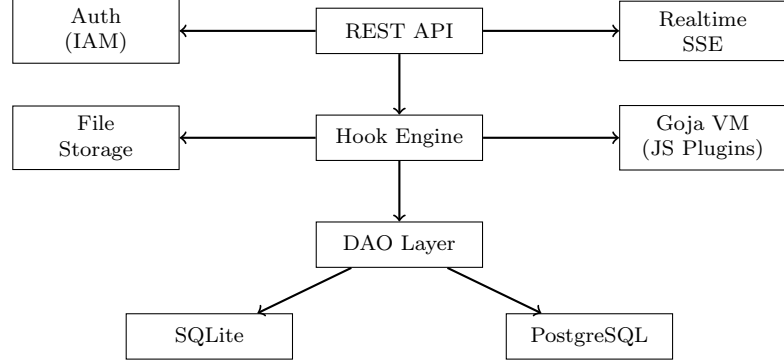


Figure 1: Hanzo Base system architecture.

scheduled jobs without recompilation.

- Dual-database architecture with SQLite for edge and PostgreSQL for clustered deployments, sharing a single migration system.
- Native Hanzo IAM integration providing OIDC-based authentication with automatic tenant isolation.
- An auto-generated admin UI with collection management, log viewer, and real-time event inspector.

2 Architecture

2.1 System Overview

Hanzo Base is structured as a layered Go application (Figure 1). The REST API layer handles HTTP routing and middleware. Below it, the Hook Engine intercepts all data operations, dispatching to registered JavaScript plugins and Go hooks. The DAO layer abstracts database operations across SQLite and PostgreSQL backends. Peripheral systems—authentication, file storage, and real-time subscriptions—integrate at the API and hook layers.

2.2 Collection Model

Data is organized into *collections*, each defining a schema, validation rules, and access policies:

```
{  
  "name": "orders",  
  "type": "base",  
}
```

```

4  "schema": [
5    { "name": "total", "type": "number",
6      "required": true, "min": 0 },
7    { "name": "status", "type": "select",
8      "values": ["pending", "paid", "shipped"] },
9    { "name": "customer", "type": "relation",
10     "collection": "users" }
11  ],
12  "listRule": "@request.auth.id != ''",
13  "viewRule": "@request.auth.id = customer"
14 }

```

Collections support 12 field types including relations, files, JSON, and select. Access rules use a filter expression language evaluated at query time, enabling row-level security without application code.

2.3 JavaScript Plugin System

The Goja VM [2] provides a V8-compatible JavaScript runtime embedded in the Go process. Plugins register hooks on collection events:

```

1  onRecordBeforeCreateRequest((e) => {
2    const record = e.record;
3    if (record.get("total") > 10000) {
4      record.set("status", "review");
5    }
6  }, "orders");
7
8  routerAdd("GET", "/api/stats", (c) => {
9    const result = $app.dao().
10     findRecordsByFilter(
11       "orders", "status = 'paid'", "-created", 100
12     );
13   return c.json(200, { count: result.length });
14 });

```

The VM runs in the same process with direct access to the DAO, eliminating serialization overhead. Plugins are hot-reloaded on file change in development mode.

3 Key Features

3.1 Dual-Database Support

Base supports SQLite for single-node deployments and PostgreSQL for clustered environments. The DAO layer uses a common query

builder that emits dialect-specific SQL. Migrations are written once and translated at apply time.

SQLite deployments achieve sub-millisecond reads (0.3ms p50) by eliminating network round-trips. PostgreSQL deployments support connection pooling, read replicas, and MVCC isolation for concurrent workloads.

3.2 Hanzo IAM Integration

Authentication delegates to Hanzo IAM via OIDC. When a user authenticates, Base receives a JWT containing the `owner` claim (organization ID). All subsequent queries are automatically scoped to this organization:

$$Q_{\text{scoped}} = Q_{\text{original}} \wedge (\text{org_id} = \text{jwt.owner}) \quad (1)$$

This ensures tenant isolation at the database layer without application-level filtering.

3.3 Real-Time Subscriptions

Clients subscribe to collection changes via Server-Sent Events (SSE). The subscription system uses the same access rules as REST queries, ensuring clients only receive events for records they can read. Event payloads include the full record after change, the action type (create, update, delete), and a monotonic sequence number for ordering.

3.4 File Storage

Base provides integrated file storage with support for local filesystem, S3-compatible backends, and Hanzo Storage. Files are attached to records via file-type fields, with automatic thumbnail generation for images and configurable size limits.

4 Implementation

Base is implemented in approximately 45,000 lines of Go, plus 3,000 lines of JavaScript runtime bindings. The admin UI is a pre-built Svelte application served as embedded static assets.

Build and Deploy. A single `go build` produces a self-contained binary (approximately 35MB) including the admin UI, migration engine, and JavaScript runtime. For container deployments, the official image is 48MB compressed.

Performance. Table 1 shows benchmark results on a 4-core VM with 8GB RAM.

Table 1: Request latency (ms) and throughput (req/s).

Operation	p50	p99	req/s
Create record	0.8	2.3	12,400
Read record	0.3	1.1	31,200
List (100 records)	1.2	3.8	8,600
Update record	0.9	2.5	11,800
Auth (JWT verify)	0.1	0.4	48,000

5 Security

Authentication. All passwords are hashed with `bcrypt` (cost 12). OAuth tokens are stored encrypted at rest. Session tokens use cryptographically random 256-bit values with configurable TTL.

Authorization. Collection access rules are evaluated server-side on every request. The rule engine supports field-level access control, relation traversal, and request context predicates.

Input Validation. All user input is validated against collection schemas before reaching the database. SQL queries use parameterized statements exclusively. File uploads are validated for MIME type and size.

Tenant Isolation. In multi-tenant mode, the IAM `owner` claim is injected into every query predicate, preventing cross-tenant data access even in the presence of application-level bugs.

6 Conclusion

Hanzo Base provides a production-ready backend runtime that combines the simplicity of a single-binary deployment with the extensibility required for real workloads. The JavaScript plugin system enables rapid customization without sacrificing the performance of the Go runtime. Dual-database support allows teams to prototype with SQLite and deploy to PostgreSQL without code changes. Native IAM integration ensures consistent identity and access control across the Hanzo ecosystem.

References

- [1] G. Donchev. PocketBase: Open Source backend in 1 file. 2022.
- [2] D. Yanov. Goja: ECMAScript 5.1(+) implementation in Go. 2020.
- [3] Google. Firebase: App development platform. 2016.
- [4] Supabase Inc. Supabase: The open source Firebase alternative. 2020.