v0.3.3

Walgo

Markdown to Web3, Simplified

A unified CLI and desktop application for building, optimizing, and deploying Hugo static sites to Walrus decentralized storage on the Sui blockchain.

Walgo Contributors · February 2026 · MIT License

github.com/selimozten/walgo · walgo.xyz

Table of Contents

1. Abstract 3
2. Introduction 3
3. Problem Statement 4
4. Solution 4
5. Architecture 5
6. Storage & Economics 6
7. AI Content Pipeline 6
8. Desktop Application 7
9. Roadmap 7
10. Conclusion 8
11. References 8

1. Abstract

Walgo is an open-source toolkit that unifies the entire static-site lifecycle — from content creation to decentralized deployment — into a single command-line interface and desktop application. By combining Hugo's static-site generation with Walrus decentralized storage on the Sui blockchain, Walgo enables developers, writers, and organizations to publish censorship-resistant websites without managing servers, DNS, or cloud infrastructure.

The toolkit provides automated optimization (HTML, CSS, JavaScript minification and Brotli compression), AI-powered content generation via multiple LLM providers, project management with versioned deployments, and a cross-platform desktop GUI built with Wails.

2. Introduction

The modern web relies on centralized infrastructure — cloud providers, CDNs, and domain registrars — creating single points of failure, censorship vectors, and ongoing operational costs. Static site generators like Hugo produce fast, secure websites, but deploying and maintaining them still requires server management or platform lock-in.

Walrus, built on the Sui blockchain, offers a decentralized storage solution using erasure coding to distribute data across a network of storage nodes. Content stored on Walrus is persistent, censorship-resistant, and requires only a one-time epoch-based payment rather than recurring hosting fees.

Walgo bridges these two technologies, providing a streamlined pipeline from Markdown content to globally accessible, permanently stored websites.

3. Problem Statement

Centralized Hosting Fragility

Traditional hosting depends on a small number of cloud providers. Service outages, policy changes, or legal pressure can take sites offline without the owner's consent. Recurring costs create perpetual dependency.

Complexity Barrier

Deploying to decentralized storage requires understanding blockchain wallets, gas mechanics, blob encoding, and erasure coding parameters. This technical barrier prevents mainstream adoption of Web3 publishing.

Fragmented Toolchain

Today's workflow requires separate tools for site generation, optimization, compression, deployment, and content management — each with its own configuration and learning curve.

4. Solution

Walgo solves these problems with a unified pipeline:

```
$ walgo quickstart my-site
```

This single command creates a Hugo site, installs a theme, generates sample content, builds with full optimization, and deploys to Walrus. The key capabilities include:

Feature	Description
Dual Deployment	On-chain (Sui wallet) and HTTP publisher modes
AI Content	Multi-provider LLM pipeline (OpenAI, Anthropic, Google, Groq)
Optimization	HTML/CSS/JS minification + Brotli compression
Project Management	Track deployments, versions, and object IDs
Desktop App	Cross-platform GUI via Wails (Go + React)
Interactive Wizard	8-step guided deployment for first-time users

5. Architecture

Walgo follows a layered architecture with clear separation of concerns:

Layer	Package	Responsibility
CLI	`cmd/`	Cobra commands, user interaction, flags
Hugo Engine	`internal/hugo/`	Site creation, build, serve, theme management
Optimizer	`internal/optimizer/`	HTML/CSS/JS minification
Walrus	`internal/walrus/`	Deployment, cost estimation, blob management
AI	`internal/ai/`	Content generation, planning, validation
Projects	`internal/projects/`	SQLite-backed deployment tracking
Desktop	`desktop/`	Wails GUI with React frontend
API	`pkg/api/`	Shared interface for CLI and desktop

All packages use dependency injection with testable wrappers (`execCommandContext` , `execLookPath`) instead of direct system calls, enabling comprehensive unit testing.

6. Storage & Economics

Walrus uses erasure coding to split each blob into shards distributed across storage nodes. This provides redundancy without full replication, reducing storage costs significantly.

Key economic properties:

Property	Value
Pricing Model	Per-epoch (Sui-based), one-time payment
Redundancy	Erasure coding (not full replication)
Typical Site Cost	~0.5–2 SUI for multi-epoch storage
Blob Size Limit	Configurable, split across shards

Walgo provides cost estimation via `walgo deploy --dry-run` and displays gas budgets before any on-chain transaction.

7. AI Content Pipeline

Walgo integrates with multiple LLM providers to generate, update, and validate Hugo content:

Command	Function
`walgo ai configure`	Interactive credential setup
`walgo ai pipeline`	Full site generation (plan + generate + validate)
`walgo ai generate`	Single content file generation
`walgo ai update`	Update existing content with AI
`walgo ai fix`	Validate and fix theme requirements

Supported providers: OpenAI (GPT-4), Anthropic (Claude), Google (Gemini), and Groq. The pipeline includes automatic front-matter validation, content structure checking, and theme-aware generation.

8. Desktop Application

The Walgo desktop application provides a graphical interface built with Wails (Go backend + React/TypeScript frontend). It exposes all CLI functionality through an intuitive UI including site creation, deployment, project management, system health monitoring, and AI content generation.

The desktop app communicates with the same Go packages used by the CLI, ensuring feature parity and consistent behavior across both interfaces.

9. Roadmap

Phase	Milestone
Phase 1	Core CLI — init, build, deploy, serve
Phase 2	Optimization pipeline — HTML/CSS/JS + Brotli
Phase 3	AI content generation — multi-provider support
Phase 4	Desktop application — Wails GUI
Phase 5	Theme ecosystem — biolink, whitepaper themes
Phase 6	Collaboration — multi-author, CI/CD integration

10. Conclusion

Walgo reduces decentralized web publishing from a multi-tool, multi-step process to a single command. By combining Hugo's mature static-site generation with Walrus's decentralized storage, it provides a practical path for mainstream adoption of Web3 publishing. The project is open-source under the MIT license and welcomes community contributions.

11. References

#	Reference
1	Sui Blockchain — https://sui.io
2	Walrus Protocol — https://walrus.xyz
3	Hugo Static Site Generator — https://gohugo.io
4	Brotli Compression — RFC 7932
5	Wails Framework — https://wails.io
6	Walgo Repository — https://github.com/selimozten/walgo