Netsujo 株式会社

PUBLIC / VERIFIED SNAPSHOT

Netsujo Agent OS

AI エージェント開発・運用実績

複数 AI を動かすだけでなく、
止める・確かめる・復旧する　までを自社の運用基盤として実装

運用で扱う 6 つの論点

01 再構築

02 選択

03 割当

04 実装

05 検証・判定

06 停止・復旧

GitHub snapshot: 2026-09-21 20:24 JST / Agent OS main: 804b786...

公開版

01 / EXECUTIVE SUMMARY

まず、数字で見る“ AI エージェント開発・運用実績”

2026-03-27 以降の運用記録と、 2026-09-21 20:24 JST の GitHub 再集計を分けて表示。
24 時間は全期間ではなく、制御判断を切り出した検証窓です。

約 5 か月
AI 開発・運用
3/27 → 9/7

540
Claude Code
sessions · 9/7

2,468
merged PR
同一 8 repos

5
本番確認
5 件 · 9/7

49
Workflows
9/7 実測

この資料が示すこと

約 5 か月の継続運用で、停止・再検証・失敗の検査化・本番境界まで実装。

まだ主張しない

- 顧客成果・ PMF
- Agent OS 単体売上
- 現在の連続稼働時間

歴史実測 / GitHub 現在値 / 24 時間の制御試験は、別々の証拠として扱います。

02 / WHY

AI を増やすほど、管理の仕事も増える

実装速度が上がるほど、人間が確認する項目も増えます。
「次に何をするか」「誰が担当するか」「完了したか」を何度も確かめる必要があります。

BEFORE

人間が AI の運用担当になる

- 進捗を見回って確認する
- 担当や競合を手で調整する
- CI とレビュー結果を見比べる
- チャットが切れた後の状況を手で復元する

AFTER

人間が担うのは、目的の定義と例外時の判断

- 正本から現在地を再構築する
- 競合しない仕事を選び、担当を割り当てる
- 実装後は CI と独立レビューで検証する
- 問題時は停止し、影響を照合してから復旧する

Agent OS が扱うのは「モデルの性能」ではなく、複数 AI を安全に運用する制御そのものです。

02 / ARCHITECTURE

Netsujo Agent OS は、 AI 開発を安全に進める制御基盤

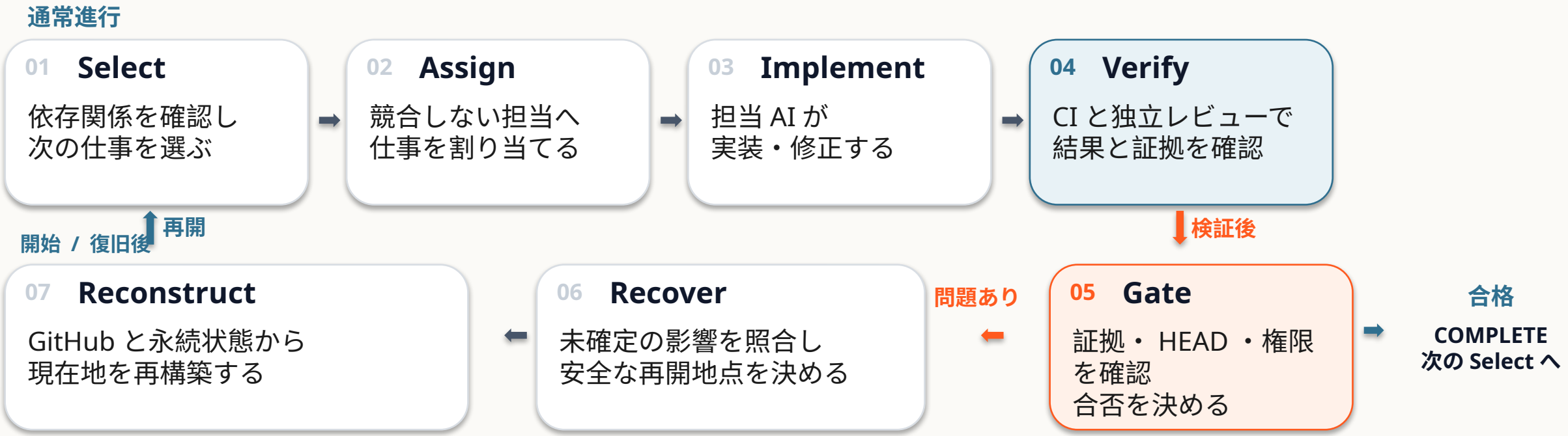
現在地の正本、担当、受け入れ証拠、停止条件、復旧地点を一つの制御基盤で管理します。

実装 AI と責任主体を同一視しない。責任・承認・証拠の境界を残したまま自動化します。

02 / CONTROL LOOP

仕事を選ぶ→担当を決める→実装→検証→判定の順で進む

Reconstruct は開始時と復旧時の入口です。 Gate で合格なら完了し、問題があれば停止して Recover→Reconstruct で再開します。



Gate は例外処理ではなく、 Verify の後に必ず通る判定点です。合格なら完了、問題があれば停止
→ Recover→Reconstruct で再開します。

02 / QUALITY & AUTHORITY

品質判定を、実装した AI の自己申告に任せない

実装する役割と、受け入れる役割を分ける。 CI が緑でも、それだけで完了とはみなしません。

1	**Machine Checks**	lint / typecheck / test / build / policy checks
2	**Independent QC**	実装者とは別の runtime ・文脈で評価
3	**Exact-head Evidence**	合格証拠を対象 HEAD へ固定
4	**Human Approval**	merge や Production 反映など、不可逆操作は人間が承認

実装した AI 自身に最終判定を任せない。これが品質設計の出発点です。

03 / HISTORICAL RUN

約 5 か月の運用で、 24 時間の制御判断を検証した

2026-09-08 13:16 → 09-09 13:16 の実運転記録。
約 5 か月の運用全体ではなく、停止・再検証・不採用・復旧の判断を確認した検証窓です。

STOP

CI が通っても QC HIGH なら停止

独立 QC を優先
CI だけで合格しない

RECHECK

HEAD が変わったら検証をやり直す

古い PASS を
使わない

DISCARD

速くても、効果が薄ければ捨てる

速度だけで
判断しない

RECOVER

チャットが切れても Live State から再開

重複を避けて
復旧する

24 時間の人間の実介入時間は未計測。現在までの連続稼働時間としては扱いません。

03 / HISTORICAL LEARNING

失敗を注意事項で終わらせず、次回の検査に変える

777

作業ログを集計

2026-08 集計

70.1%

汎用 AI に集中

専門 AI の使用記録 0 件

観測

専門 AI の使用記録 0 件

→

原因

実際の担当 ID へ未接続

→

修正

専門 AI を 3 役 新設して接続

→

固定

利用の偏りを検査に組み込む

AI 運用は導入して終わりではない。
実測し、運用設計そのものを改善します。

同じ失敗を二度注意したら、プロンプトではなく仕組みの欠陥として直します。

03 / OPERATING ASSETS

運用知を、再利用できる手順・役割・停止機構・検査へ変えた

2026-09-07 実測 / 意思決定ログ 44 件を確認

43 Skills

再利用できる AI 作業手順

26 Subagents

専門実行・レビュー役

9 Hooks

機械的に止める経路

49 Workflows

CI ・品質検査・運用自動化

20 Failure Cases

失敗事例（11 件は詳細記録）

17 Inspection Assets

失敗から作った検査資産

資産数は運用能力の規模であり、売上・顧客成果の実績値ではありません。

04 / DELIVERY PROOF

9/7 の本番確認 5 対象を、現在の公開状態まで再監査した

9/7 に本番確認した 5 対象（うち自社 SaaS 1 件）を、 2026-09-21 に現行 URL まで再監査。
4 対象は HTTP 200 。 Cyber Lab は Production 成功・認証保護中。

Netsujo SIGNAL	HTTP 200 / /services/signal
netsujo.jp	HTTP 200 / 公開トップ
みやこで IT	HTTP 200 / www へ転送
Netsujo Cyber Lab	Production 成功 / 認証保護中
蕎麦手打ち たか橋	HTTP 200 / canonical 一致

CURRENT STATUS

一般公開

4 対象 / HTTP 200
CTA ・ canonical 確認

認証保護

Cyber Lab / Production 成功
Vercel Authentication で非公開

「本番確認 5 件」の歴史実測と、現在の公開状態を分けて表示しています。

04 / HISTORICAL AUTOMATION

コードだけでなく、日常業務も AI で実行できるようにした

2026-09-07 時点の実装記録。反復作業を AI の実行系へつなぎ、不可逆操作は承認境界で分けました。

イベント下書き	connpass の下書きを生成
請求書登録	会計クラウドへ登録・ OCR
メール返信	返信案を生成
Search Console 週次監査	サイトマップ・ index ・ canonical ・ CTR を監査
Indexing API	対象ページの登録を実行

外部送信や本番反映など不可逆操作は、権限と承認を別に設けます。

04 / ENGINEERING SCALE

同じ 8 リポジトリで、 merged PR は 1,764 件から 2,468 件へ

2026-09-07 の baseline と同じ 8 リポジトリを GitHub で再集計。 2026-09-21 20:24 JST 時点で +704 件。

Repository	merged PR	現在の位置づけ
netsujo-web	1,395	旧主力 / 参照用
miyakodeit	311	コミュニティ
netsujo-site	252	コーポレート / 公開
netsujo-signal	236	プロダクト / 診断
orchestrator / agent-os	143 / 116	実行制御 / ルール正本

表外の netsujo-aio-seo 13 / netsujo-cyber-lab 2 を含む合計 2,468 件。

04 / REPOSITORY BOUNDARY

モノリスを分け、境界を暗黙の了解ではなく契約で固定した

2026-08-30 、公開面・ SaaS ・実行基盤を責務ごとに分離。
サイト側が SIGNAL の実装を直接 import できない境界を設けました。

netsujo-site
公開 Web サイト

netsujo-signal
SaaS

netsujo-orchestrator
実行基盤

境界で検査する

- 契約 ID version
- 生成元 repo commit SHA
- payload SHA-256
- 必須キー

食い違いがあれば、古い情報のまま進まず停止します。

コードだけでなく、状態・権限・証拠も同じ考え方で境界を固定します。

05 / CURRENT SOURCE STATE

Agent OS の現在地は「 LOCAL_VERSIONED 」

2026-09-21 20:24 JST 。 Agent OS は会社共通の役割・権限・知識・割当を独立 Git で管理・検証。
README 上も REMOTE_BACKED ではなく、 runtime adapter は Phase 2 まで未導入です。

CURRENT

LOCAL_VERSIONED

役割・権限・知識・割当を
会社共通の正本として
独立 Git で検証

NO
remote state

NO
runtime adapter

Phase 1
current

main 804b786...

README: LOCAL_VERSIONED
Phase 2 before adapter
source 更新 ≠ runtime 稼働

source の更新・ merge と runtime 稼働は、同じ状態として扱いません。

04 / CLIENT VALUE

自社で磨いた AI 開発・運用能力を、顧客環境へ実装する

完成品を一律販売するのではなく、顧客環境に合わせて運用を設計し実装します。

AI Agent 構築

業務を AI が実行できる単位に分解

複数 AI の役割設計

実装・調査・レビューを分担

品質ゲート

CI ・独立 QC ・証拠・承認

停止・復旧設計

結果不明や HEAD 変更後も安全に再開

AI 運用の社内定着

正本・ルール・検査を社内運用に定着

既存開発への導入

GitHub ・ CI ・ Vercel と接続

相談の入口： netsujo.jp/ai-development → AI 開発・運用について相談

05 / WHAT WE DO NOT CLAIM

信頼を落とす数字は載せない

実績資料では、数字の大きさより「根拠を示せること」を優先します。

× **Agent OS 単体の外販実績**	導入社数・売上は主張しない。社内 R&D の知見は支援に転用
× **未計測の人間介入時間**	GitHub 履歴だけでは正確に復元できないため載せない
× **R&D の利用者数・収益化**	研究・開発段階を提供実績として数えない
× **受託案件の顧客名・納品物**	守秘義務の範囲は公開しない
× **AI による無承認の本番変更**	不可逆操作は権限と承認条件を満たした経路だけで行う

公開物の目的は大きく見せることではなく、依頼先が能力と限界を正しく判断できる状態をつくることです。

05 / PROVENANCE

最新値にも、測定時刻と未到達の状態を残す

数字は対象範囲（ scope ）と測定方法が同じ場合だけ比較します。
過去値、現在の source 、 runtime の状態を分けて示します。

確認日	何を確認したか	数値・状態の扱い
2026-09-07	baseline： 同じ 8 repos ＋ AI 運用の歴史実測	1,764 PR · 540 sessions · 本番確認 5 件 9/7 実測として固定
2026-09-21	GitHub 20:24 JST · 同じ 8 repos の merged PR 数	2,468 件。同一 scope で比較可能、 +704 件
9/8→9/9	24 時間の制御判断を切り出した実運転	STOP / RECHECK / DISCARD / RECOVER を確認。 全運用期間・現在の連続稼働とは別

CURRENT **Agent OS main 804b786...** ORCH main d09f909...
runtime 稼働状態は別途確認

SOURCE **EV-2026-033 / EV-2026-034**
GitHub snapshot 2026-09-21 20:24 JST