AURANT TECHNOLOGIES

AX（AI×DX）コンサルが自社システムを大公開

コンサルファームが自ら実践する業務効率化の裏側　前編・後編 統合版

2026.03 | Aurant Technologies

kintone

Claude Code

内製化

n8n

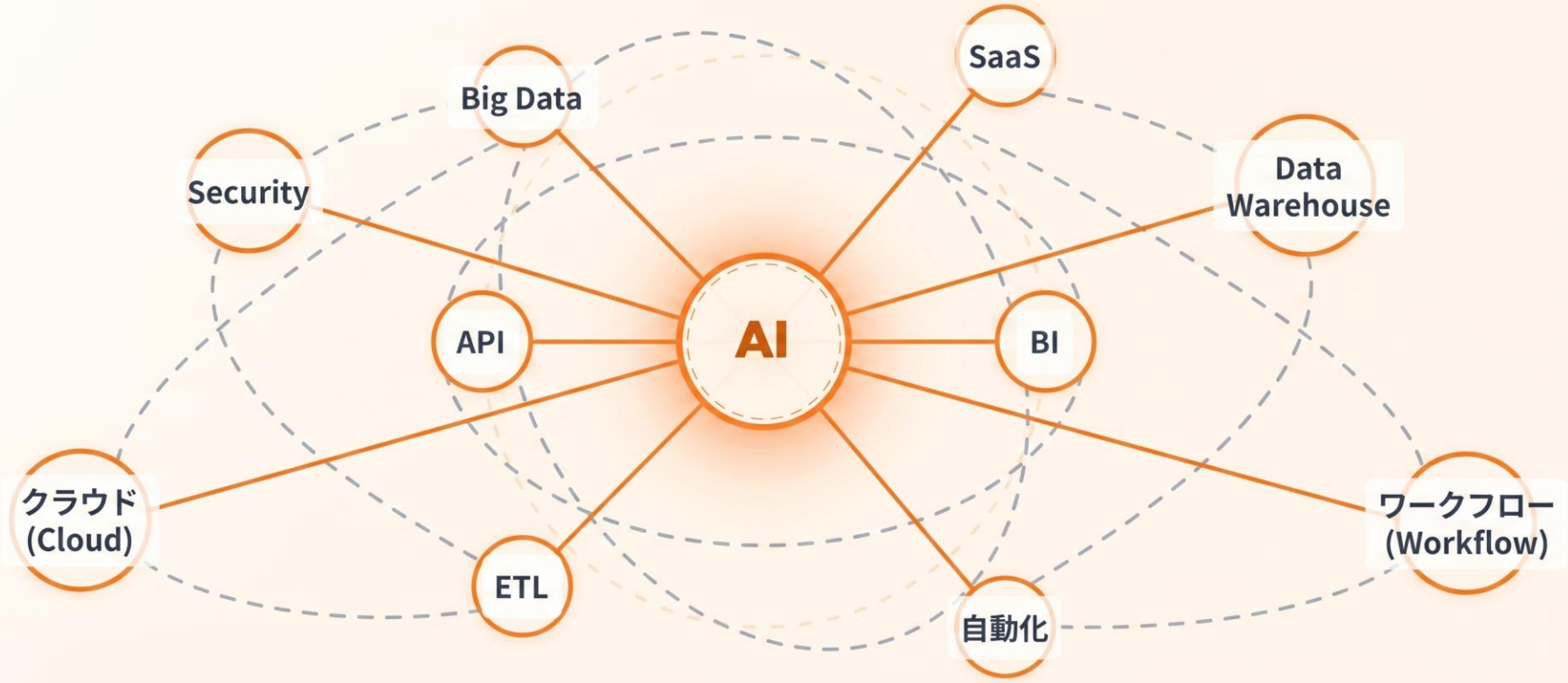
WebAPP

SaaS

01

会社紹介-
Our Profile

企業理念- Our Philosophy



Aurant（オーラント）は「新しい存在」を意味する造語。ラテン語の “aurum”（黄金）と “aura”（気配・存在感）“-ant”（～する者）を掛け合わせ、まだこの世に無かった価値を生み出す存在という意志を名前に刻んでいます。AIを使った、全く新しいビジネスを体感していただくことを理念として、サービスを提供します。

コンセプト- Our Concept

この10年、企業はSaaSを導入し続けてきました。その結果、SaaSの数だけデータのサイロが増え、ツール間を人手でつなぎ合わせる非効率が新たなボトルネックに。そして今、AIの進化がこの構造を根本から変えようとしています。

しかしAIはすべてを解決する魔法ではありません。AIに任せるべきところはAIに、AIでは届かない領域はコンサルティングと開発力で補完する。境界線の設計こそが、私たちの最大の価値です。

会社概要- Company Profile

項目	詳細
会社名	Aurant Technologies
所在地	東京都府中市武蔵台2丁目15
従業員数	50名
事業内容	・AI活用・業務自動化コンサルティング ・Salesforce / kintone / 会計システムの連携設計・実装 ・データ統合基盤の構築（可視化・分析基盤を含む） ・WebAPP開発・運用支援

メンバー紹介- Our Member

宮澤孝慈 Koji Miyazawa

慶應義塾大学でデータサイエンスを学び、学部卒業後は同大学の修士課程に進学。医療分野をはじめとした多様な領域で、ビッグデータを用いた実証検証や大企業とのPoCに携わる。学部在学中からはB2Cサービスの起業を通じてシステム開発にも取り組み、その実践的な経験を現在の事業にも活かしている。現在は、生成AI事業を含むシステム開発やデータサイエンスと並行して、kintone/salesforceを活用したデータ利活用支援や、業務改善を目的としたDX推進を得意

守高成悟 Seigo Moritaka

Webアプリ開発、データ基盤構築、業務のAI化を通じて、企業の抜本的な業務改善を支援するエンジニア・起業家。10歳からプログラミングに親しみ、大阪大学で情報工学・統計学・AI研究の学術的バックグラウンドを持つ。 フロントエンドからバックエンド、クラウドインフラ、データ活用まで一気通貫で手がけるフルスタックな技術力を武器に、現場で「使われ続ける」仕組みづくりを提供している。

三輪衛 Mamoru Miwa

慶應義塾大学在学中に創業1期目のベンチャー企業へ参画し、2年間にわたって新規事業部の立ち上げを主導。学生でありながらゼロから事業を創り上げる実践的な起業家経験を積む。その後、融資支援や企業財務分析を手がけながらバックオフィス全体の責任者として組織基盤の構築を牽引。事業開発から財務・管理領域まで幅広いキャリアを持つオールラウンドなビジネスプロフェッショナル。

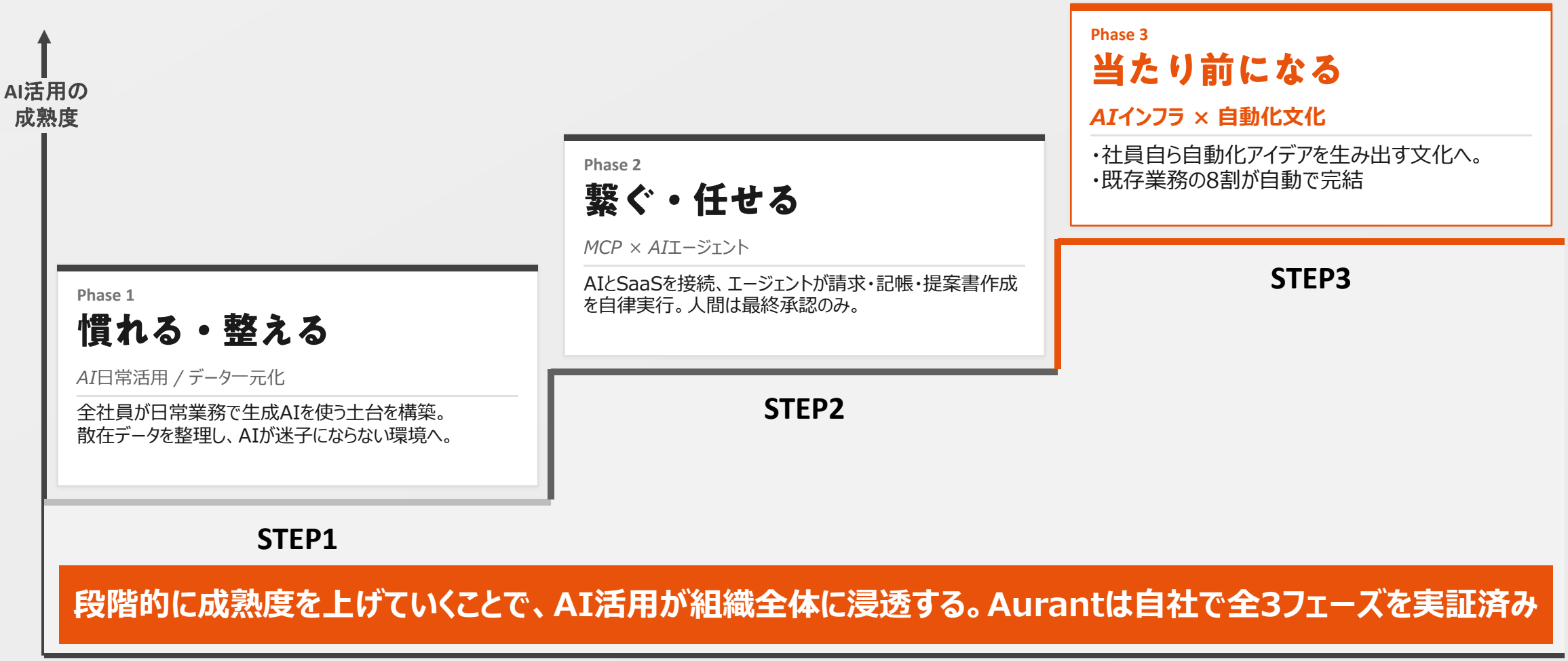
なぜAurantが選ばれるか？

「他社事例を借りる」でも「高額パッケージを売る」でもなく、「自社で動かしている実践知をそのまま提供する」

比較軸	一般的なEHコンサル	Aurant Technologies
会計・業務の専門知見	△ ITエンジニアのみで業務理解が浅い	◎ **CFOレポート自動化まで可能な会計・業務のプロ**
AI実装力	△ PoC止まり・本番稼働しない	◎ **本番稼働済み（自社で毎日動かしている）**
コスト感覚	○ 商材としてのAIを売っている	◎ **柔軟な設計可能・スケール時も跳ねない**
情報源（ノウハウ）	△ 他社の成功事例を借りる	◎ **今まさに自社で動かしている生きた実践知**
対応範囲	△ 単体ツールの導入のみ	◎ **戦略設計 → SaaS統合 → AI開発まで一気通貫**

要件が決まっていない状態でも、お気軽にご相談ください

AI組織実装の3フェーズ「慣れる→繋ぐ→当たり前になる」



02

弊社のAI・活用・基盤まで大公開

まずは効果から

SaaSと内製化を組み合わせ、Claude Codeを軸にAIを業務フローに組み込むことで「1人8倍の成果」を実現

01
SaaSの限界を認識
SaaS × 内製 × AI の設計

02
Claude Codeで変革
「自動化」から「仕組みを作る」へ

03
GitHub×n8nサイクル
個人の工夫を組織標準へ

SaaSの限界を認識し「SaaS × 内製 × AI」の設計を採用

お客様向けのシステム開発を行う際に、SaaS製品（kintone、Salesforce等）を中心に開発を行っていましたが。ユーザーフローへの当てはまりにくさ・システム間連携の不自由度・コストの増加・属人化・追従コストの課題に直面。各ツールを最低ライセンスで1ヶ月実際に使い込んで比較した上でフェーズに合った最適構成を選択した。

Claude Codeで「自動化」から「仕組みを作る」へ変革

汎用AIチャットは会話で終わるが、Claude Codeはコード実行・ファイル操作・外部連携まで完結。提案書生成・タスク自動化・記事化・財務自動化などのユースケースを実現。

Claude Codeで個人の自動化を突き進め、個人個人がエンジニアリングを行えるように。

GitHub × n8nで個人の工夫が組織標準のサイクルを構築

作業終了時にGitHub自動更新
→「神スキルズ」として蓄積。組織フローはn8nで標準化。このサイクルが全員のベースラインを継続的に引き上げる。

1人8倍の
生産性

この資料について

「SaaSだけでは足りない」と感じたことがある方へ。当社の実践・失敗・成果をそのまま公開します

01

経営者・経営企画

ツールコストが増え続けているのに業務が楽にならない、AIをどう導入すれば良いかわからない方に。

02

IT担当・情報システム

SaaSの連携設計・内製化の判断基準・GitHub文化の根付かせ方を知りたい方に。

03

コンサル・士業・業務改善

自社または顧客へのツール提案に使えるユースケース集として活用いただけます。

スキップガイド

📋 Part 1〜2（SaaSの限界・社内スタック）	ツール選定の判断基準を知りたい方はここから
🤖 Part 3〜4（AI活用・ユースケース）	Claude Code・n8n・MCPの説明から始まります。IT用語が不安な方も安心してください
📈 Part 5（成果・組織設計）	「どれだけ変わったか」を数字で確認したい方は直接こちらへ

SaaSだけでは届かない5つの壁

機能・連携・コスト・属人化が積み重なり、「SaaS＋内製WebAPP＋AI」の組み合わせへの移行判断に

01	フローとの不一致	機能は豊富でも自社業務フローに合わず操作が煩雑に。Salesforceでは入力項目が増え現場が嫌がる状態に。カスタマイズで追加費用が発生しても完全対応できないケースが頻発した。
02	連携の断絶と二重入力	複数SaaS間でデータが宙に浮き、二重入力が常態化。「Salesforceは受注だがkintoneはまだ商談中」という不一致が正確な予実管理を阻害した。ツールが増えるほど連携コストが指数的に増大。
03	ライセンスコストの膨張	Salesforce Enterpriseエディションは月額1.98万円/ユーザー～。コストを抑えようと一部のメンバーにしかアカウントを付与しない判断をすると、ライセンスなしメンバーがExcelで代理入力する逆効果が生まれる。
04	属人化リスク	設定した本人しか構造を把握できない「ブラックボックス」化が進行。担当者が離脱した際に「何がどうなっているか分からない」状態が発生。運用継続に外部ベンダーへの恒常的な依存が生まれた。
05	アップデート追従コスト	UIの変更・機能廃止・API仕様変更のたびに追従が必要。自動化が突然壊れる経験を複数回した。Backlogとkintoneの連携では手動転記が残り、どちらが最新か分からない状態が常態化した。

試して、手放して、学んだこと

実際に使い込み、費用・連携・定着の3観点で判断。失敗の言語化が次の選定精度を上げる

ツール	カテゴリ	手放した理由（調査・実経験）	代替の判断根拠
Salesforce	CRM	2025年8月値上げ後：Enterprise→21,000円、Professional→12,000円/ユーザー/月（25%増）。専任管理者不在の中小では改修のたびに外注コスト。中小企業からの移行相談が増加。入力項目多すぎで現場が定着しないケースが頻出	kintone：月額1,800円～/ユーザー。 ノーコードで現場改修可能。専任管理者不要
クラウドサイン	電子契約	送信1件あたり220円の従量課金。月100件で変動費2.2万円超。freeeとの連携不可で会計側に手動入力が残った	freeeサイン：月額固定5,980円～で50通まで無料。 freee会計と同エコシステムで仕訳連携が自動化
Backlog	PM	kintone工数データとの連携に手動転記が残り二重管理が常態化。「どちらが最新か」の不一致が予実管理を妨げた	kintone × 自社WebAPP（内製）：APIで打刻→ 予実突き合わせまで一本化
Microsoft Teams	コミュニケーション	外部サービス連携・自動通知の設計柔軟性でSlackが優位。kintone・n8nからの自動通知集約にTeamsは設計が複雑	Slack：n8n・kintoneからの通知を一本集約。 チャンネル設計で情報密度を制御
Tableau	可視化	Creatorライセンス月額約7～10万円/ユーザー。学習コスト高く、データ連携設計も複雑。コストに見合う機能差は限定的。中小では「用途2割でフルコスト」状態が多い	Apache Superset（OSS）：自社運用でゼロコスト。 Salesforce・kintone・freee全データを集約可能
マネーフォワード	会計	マネーフォワードは、ALLINONEの製品。その他、関連製品も一緒に導入しなければ業務効率が落ちる一方、コストがかさむため断念。freee会計は各種仕訳の分析機能が付随しており、より分析などに踏み込むことができる	freeeに一本化：同エコシステム（freee会計・freeeサイン・freee人事）でデータが完結
Gemini / ChatGPT	AI	汎用AIチャットは「テキストを出力して終わり」。実装・ファイル操作・外部連携まで完結しないため業務フロー組み込みに限界。Anthropic社内ではClaude Code活用でコード出力量が前年比200%増（2026年）	Claude / Claude Code：ターミナル統合・MCP対応・コード実行まで一気通貫

ツール詳細比較①：Salesforce vs kintone

Salesforceは、2025年8月に再値上げ。月額コストは最大10倍以上の差に。
中小企業での失敗パターンは「現場が入力しない→Excel逆戻り→ライセンスだけ払い続ける」の3段階が典型的

比較軸	Salesforce（2025年8月改定後）	kintone（スタンダード）	当社の判断
月額料金	Professional：12,000円/ユーザー/月 （2025年8月～25%値上げ） Enterprise：21,000円/ユーザー/月（同6%値上げ）	1,800円/ユーザー ～ （スタンダード）	最大10倍以上の差。初期フェーズでは費用対効果が合わない
現場の定着率	「入力項目が多くて面倒」が現場定着を阻む。導入後にExcelへ逆戻りし「高い仕組みを導入しただけ」に終わるケースが多発（複数調査で共通の失敗パターン）	直感的なUI。非IT部門でも使いやすく定着率が高い	「使われないシステム」を避けるため最優先の選定軸
専任管理者	ほぼ必須。変更・改修のたびに外部ベンダー依存になりやすい。担当者離脱でブラックボックス化が頻発	不要。ノーコードで現場が自ら改修可能	管理者コストを考慮すると実質コスト差はさらに拡大
カスタマイズ	ApexやVisualForceが必要。開発コストが恒常的に発生。「標準機能に業務を合わせる」発想が前提	ドラッグ&ドロップ。 JavaScriptで高度化も可能	初期フェーズは内製改修が可能なkintoneが現実的
フェーズ移行	IPO・大規模・高度な営業分析が必要になったら最適。5年以上使い込んだ企業では移行コストが高い	数百名規模まで対応可能。 その後Salesforceへ移行	規模・要件に応じて フェーズ移行の判断を行う

ツール詳細比較②：電子契約・可視化・AI

クラウドサインの従量課金・Tableauのライセンス費・AIの実装能力の差が乗り換えを決定的にした

電子契約：クラウドサイン → freeeサイン		
月額費用	従量課金：固定費0円＋220円/件送信	**固定費のみ：月額5,980円～で50通無料**
月100件の場合	変動費：2.2万円～（件数増でリニア増加）	固定費5,980円のみ（件数増でも一定）
会計連携	freeeへの直接連携なし。手動入力が残る	**freeeと同エコシステム。締結→仕訳まで自動**
可視化：Tableau → Apache Superset（OSS）		
ライセンス費	Creatorライセンス：月額約7～10万円/ユーザー	**オープンソース。自社運用でランニングコストゼロ**
データ接続	kintone・freee等との連携設計が複雑	REST API・直接DB接続で400種以上のデータソース対応
AI：Gemini / ChatGPT → Claude / Claude Code		
コード実行	会話出力のみ。コードは提示するが実行は手動	**ターミナル統合。書く・実行・修正・コミットまで完結**
外部連携	限定的（ブラウザ内のみ）	MCP対応。kintone・Slack・freee等を直接操作可能

ツール選定の判断基準

自社採用は「最低ライセンス×1ヶ月×Claude Code」で実戦検証。顧客提案はUI直感性と過去案件ユースケースでの機能テストで評価する

自社採用の判断基準

STEP 1

最低ライセンスで契約

フルライセンスは不要。最小コストで本番に近い使い方を確保する。評価のための1ヶ月分を割り切って払う。

STEP 2

1ヶ月間、実務に完全投入

最初の2週間は誰でも「使いにくい」と感じる。3〜4週間経て初めて「合う・合わない」の判断が可能になる。短縮すると正しい評価ができない。

STEP 3

Claude Codeと組み合わせて検証

API連携・他ツール接続・自動化の設計相性を確認。「素のツール」ではなく「フローに組み込んだとき」を評価する。これが当社の判断の核心。

顧客提案での評価基準

判断軸 01

UIの直感性

初めて見た人が10分で基本操作を覚えられるか。特にITリテラシーが多様な現場では最重要項目。「使われないシステム」を生まないための最優先確認事項。

判断軸 02

過去案件ユースケースで機能テスト

支援実績から抽出した業種別・課題別要件をツール上で実際に再現し、「どこまで対応できてどこからできないか」を具体的に検証する。

最終判断

自社で使い込んで本音で語れる

1ヶ月使い込んでいるからこそ「ここが強くてここが限界」と具体的に言える。これが「自社実践をそのまま提案に活かす」姿勢の核心。

03

社内スタック全体像と内製化の設計

ツール構成 ／ コミュニケーション設計 ／ 名刺管理 ／ kintone内製化 ／ バックオフィス

社内ツールスタック全体マップ

10カテゴリを「SaaSの強みを活かす領域」と「内製・カスタム領域」に分け設計。オレンジが内製・カスタム部分

コミュニケーション	ドキュメント管理	ファイル管理	顧客・名刺DB	PM・工数管理
Slack	Notion（移行中） Confluence GitHub	Google Drive	Eight Team × kintone 内製・カスタム	kintone × WebAPP 内製・カスタム

CRM	バックオフィス	可視化・経営	自動化・AI	Webサイト
kintone (過去：Salesforce)	現在： Freee 今後（フェーズ移行時） 勘定奉行 × バクラク	Github Apache Superset	Claude Code n8n GitHub 内製・カスタム	WordPress（自社カスタム） 内製・カスタム

コミュニケーション・ドキュメント管理の使い分け

「誰が主体か」「何を目的とするか」でツールを分担。Slackが全体をつなぎ、Notion/Confluence/GitHubがそれぞれの役割を持つ

Slack

全社通知ハブ

社内のやり取りと外部システム（kintone・AIエージェント・採用媒体など）からの自動通知を一本化。チャンネル設計でノイズを抑制。プロジェクト別・機能別の分離で「どこを見ればいいか」を常に明確に保つ。

→ コミュニケーション全般・自動通知の集約先

Notion（移行中）

非エンジニア向けナレッジ

提案書・プロジェクト振り返り・予実管理を蓄積。Notion AIで過去資料を横断検索でき、自然文（例：「株式会社〇〇案件100万円」）でダッシュボードに反映。ナレッジ層・運用層・メモ層の3層で設計。

→ 全社ナレッジ・予実管理・営業パイプライン

GitHub

エンジニア・開発の管理

コード・仕様書・設計ドキュメントを管理。作業終了時に自動でREADMEが生成・蓄積される（詳細はPart 4）。「過去に似たものないか」と検索する文化が車輪の再発明を防ぐ。

→ コード管理・技術ナレッジ・神スキルズの土台

Confluence

社内規定・全社共有文書

社内規定・全社共有が必要な重要ドキュメントを格納。Notionとの使い分け基準は「誰が主体か」。エンジニア主導はGitHub/Confluence、全社ナレッジはNotion。

→ 社内規定・全社向け重要文書

Notionの3層活用設計

ナレッジ・運用・メモの3層に分けることで「どこに何があるか分からない」問題を解消。Notion AIで横断検索・要約・ダッシュボード反映まで一気通貫

Layer 1

ナレッジ層

組織の記憶・蓄積

◦ 過去の提案書・プロジェクト振り返りをページとして蓄積

◦ Notion AIで横断検索：「〇〇業種に提案したことあったっけ」に即答

◦ 採用されなかった提案も含め、判断の文脈まで残す

実例

「kintone関連の過去提案を全部まとめて」→ Notion AIが数秒で要約

Layer 2

運用層

日常業務の回転

◦ 営業パイプライン・案件ステータス・週次予実管理シート

◦ 新規案件を自然文で入力→Notion AIがダッシュボードへ反映

◦ 「株式会社〇〇案件100万円」の入力だけで売上管理シートに自動更新

実例

入力の摩擦を下げることで「数字の更新を後回しにしない」状態を作る

Layer 3

メモ層

記録のハードルを下げる

◦ 商談メモ・アイデアの走り書き・ミーティングのアジェンダ

◦ 後から構造化するのはAIに任せ、記録することへの心理的障壁を下げる

◦ 「とにかく書いておく」文化がナレッジ蓄積の土台になる

実例

メモ→ナレッジ化のサイクルをAIが自動で橋渡しする

名刺管理の自動化：Eight Team × kintone 連携

スマホで名刺をスキャンするだけで顧客DBが自動更新。転職・役職変更も追従し、手入力ゼロで顧客データを常に最新に保つ

名刺受け取り	Eightで スキャン	Eight API で連携	kintone 顧客DB	更新自動 追従
営業現場・ 展示会など	スマホアプリで 読み取り・自動OCR	データ変換・ 自動転送	会社名・役職・ 連絡先が自動登録	後日メール送付 ・商談セッティング

kintoneが「現場起点の顧客マスタ」として機能し、後述するプロジェクト管理WebAPP・バックオフィス連携・AIエージェントの起点にもなる。名刺1枚のスキャンがCRM・請求・ナレッジまでつながる設計。

手入力ゼロ

スキャン→自動登録で転記ミスなし

常に最新

Eight側の転職情報が自動でDB反映

組織の資産

属人化しない顧客情報を全社で共有

プロジェクト管理・工数管理の内製化

「現場が続けられるUI」と「経営に使えるデータ」を同時に満たすためkintone × 自社WebAPPの内製構成を選択した

AS-IS 既製SaaSで挫折した理由

× 操作がしっくりこない——自社フロー独自の承認・ステータスに対応できない

× 他システムとつながらない——kintone工数データが手動転記になる

× 人数分のアカウント代がかさむ——プロジェクトが増えると全体像も掴みにくくなる

× 現場が入力を続けてくれない——UIが複雑で定着しない

TO-BE kintone × WebAPP 内製構成

打刻層（kintone）

現場スマホからワンタップ打刻。UIを極限まで絞り定着率を確保。建設業2024年問題の法的対応も充足。「打刻テーブル」と「案件テーブル」を分けて設計。

集計・予実層（WebAPP）

打刻データをAPIで取得し、ホワイトボード型の画面でプロジェクト全体を俯瞰。Salesforce受注金額と突き合わせて予実管理。「現場が触る層」と「集計に回す層」を分けることが運用継続の鍵。

AI予兆検知

工数対比でのコスト超過・採算悪化の予兆をClaudeが検知してSlack通知。「問題が起きてから対処」から「予兆を拾って先手を打つ」経営に変わった。

システム開発の第三の選択肢

AI（Claude Code） × WebAppで実現する、SaaSでもスクラッチ開発でもない第三の選択肢でDXを実現

SaaS・ローコードツールの悩み

導入は早いけど、自社の特殊な業務に合わない
人数が増えるとアカウント代が高すぎる

結局、エクセル管理　　ユーザー課金

スクラッチ開発の悩み

自社の業務にぴったり合うものを一から作ろう！
と見積もったら、初期費用が高い。時間もかかる、
出来上がったが、使いにくい

システム開発の長期化　　使いにくい

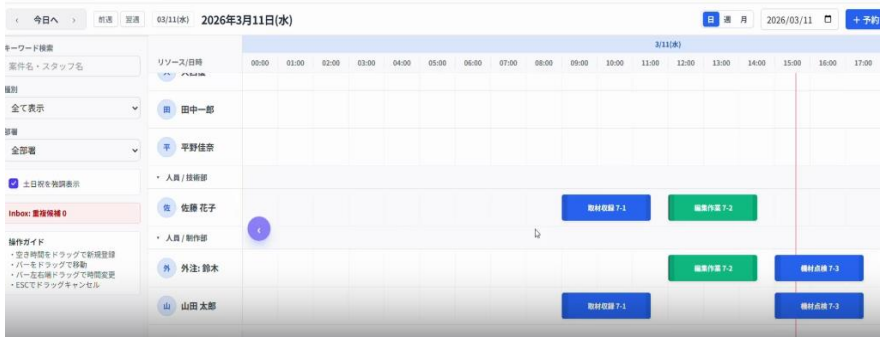
提案

Claude Code × Web APP

で実現する、システム開発
(固定料金、Good UI、)

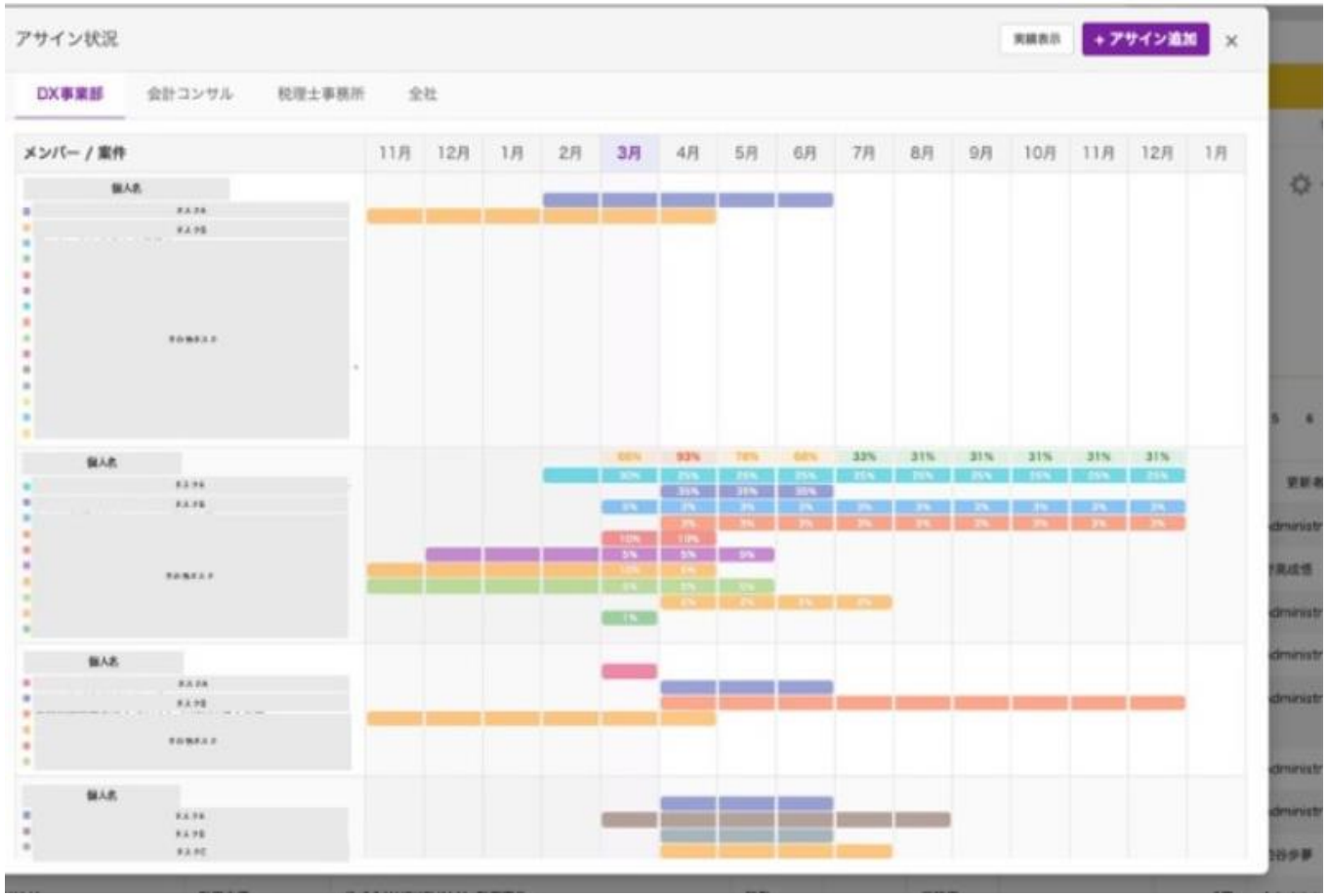
実際のイメージ（各種SaaS×WebAPP）

打刻・スケジュール・施設予約



会計レポート（BI）

プロジェクト・アサイン管理



バックオフィス（会計・請求）の構成

企業の成長フェーズに応じてシステムを切り替える。「スピード重視の黎明期」と「内部統制が必要な拡大期」で最適解が変わる

黎明期～中規模フェーズ（～数百名）

kintone × freee会計

kintone側で請求確定したデータをAPIでfreeeへ連携し、請求から消込までの流れを短く保つ構成。スピードと柔軟性のバランスが取りやすく、現場運用と経理処理を無理なく接続できる。

⚡ freeeサインで電子契約→freee会計で仕訳まで、同エコシステムで完結。

上場準備・大規模フェーズ（IPO・内部統制）

Salesforce × 勘定奉行 × バクラク

CRM→債権奉行→勘定奉行クラウドを連携軸に、現場入力はバクラクで吸収するハイブリッド構成。現場の入力負担を下げつつ、経理側へは承認済みの整ったデータだけを流す設計。

🏢 IPO準備・内部統制要件が強くなるフェーズで最適。

フェーズ移行の判断基準

1. 社員数が100名を超え、手動入力や確認作業が経理の負担になってきた
2. IPO準備・内部統制要件が強くなり、監査対応が必要になった
3. 複数の法人・会社間取引が発生し、freeeでの管理が複雑になってきた

kintone×freee会計

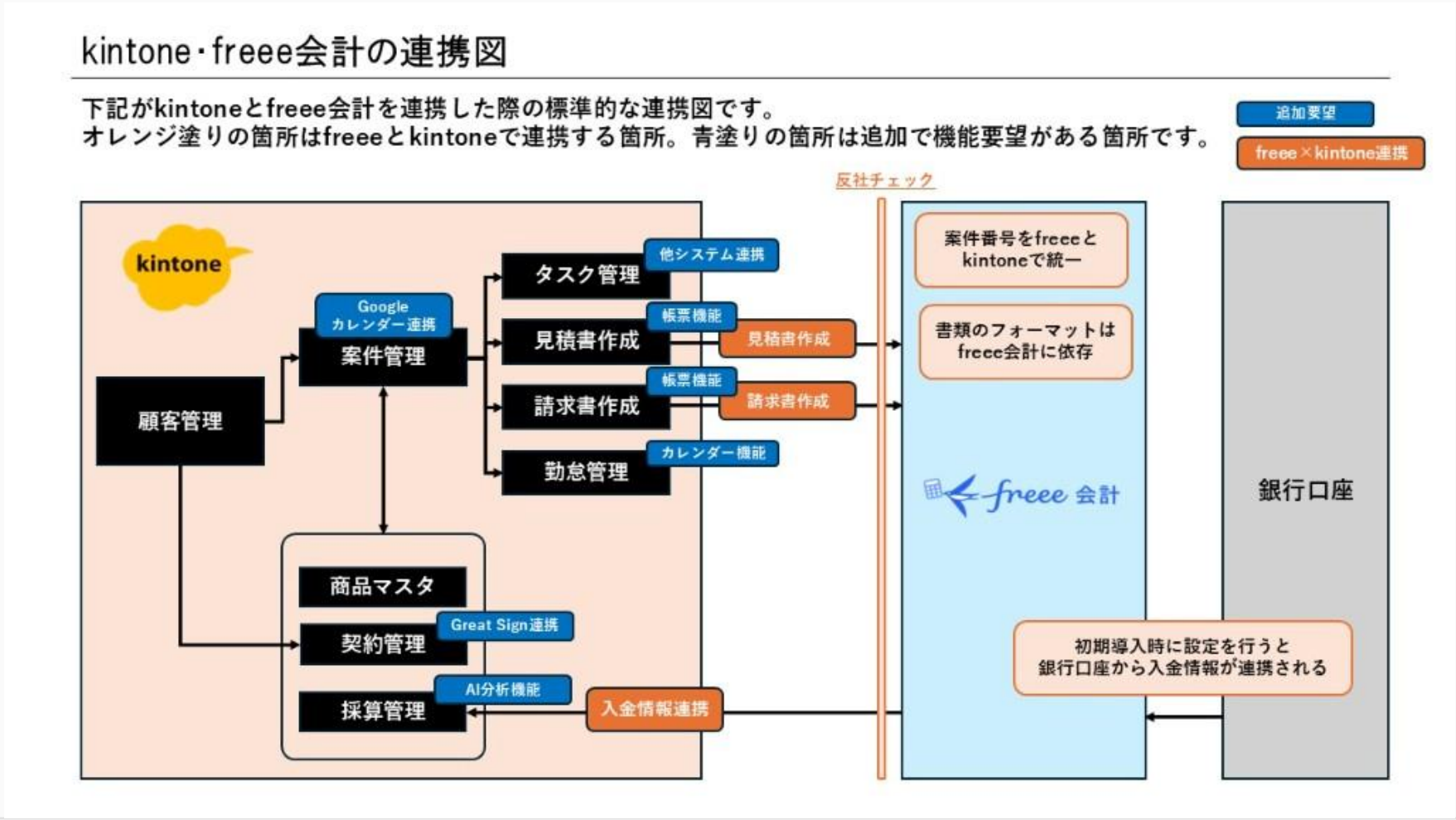
黎明期～中規模フェーズ（～数百名）

kintone × freee会計

kintone側で請求確定したデータをAPIでfreeeへ連携し、請求から消込までの流れを短く保つ構成。スピードと柔軟性のバランスが取りやすく、現場運用と経理処理を無理なく接続できる。

⚡ freeeサインで電子契約→freee会計で仕訳まで、同エコシステムで完結。

請求書抜け漏れ無し　**予実管理（PJ毎）**　**自動入金チェック**



Salesforce×勘定奉行×バクラク

上場準備・大規模フェーズ（IPO・内部統制）

Salesforce × 勘定奉行 × バクラク

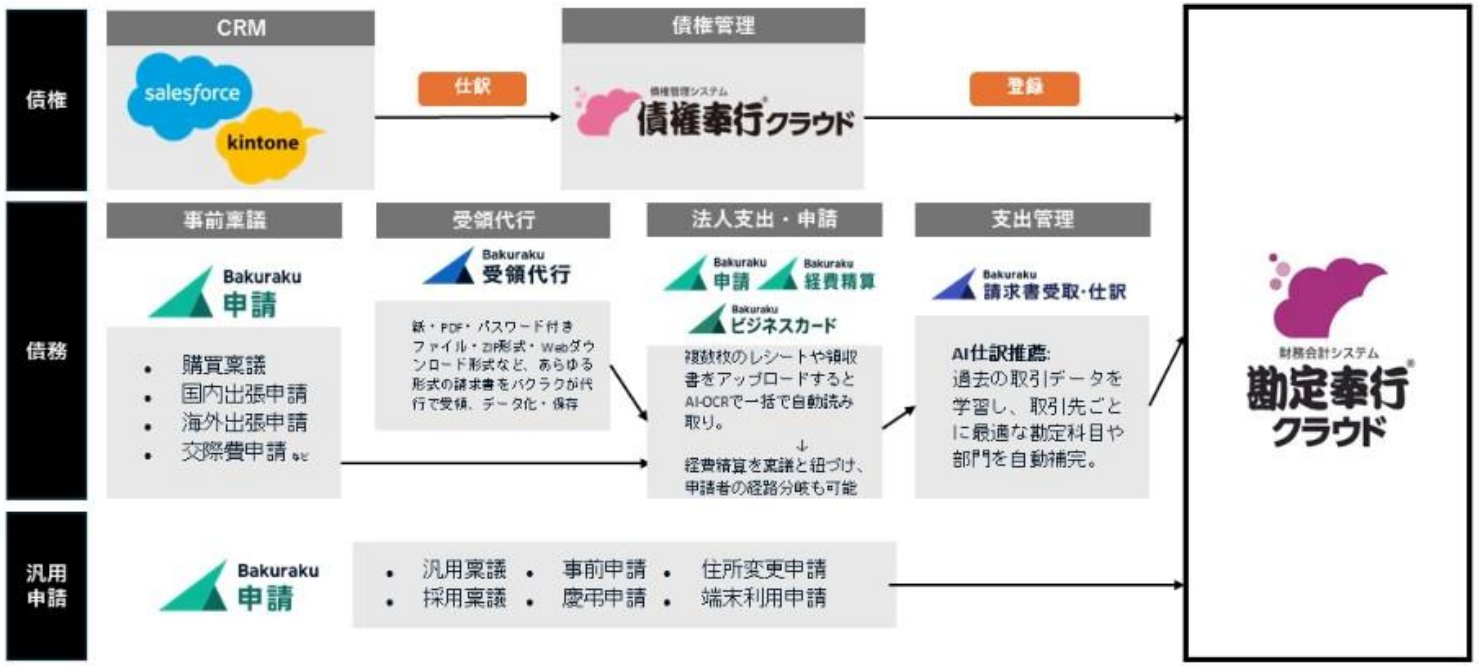
CRM→債権奉行→勘定奉行クラウドを連携軸に、現場入力はバクラクで吸収するハイブリッド構成。現場の入力負担を下げつつ、経理側へは承認済みの整ったデータだけを流す設計。

🏢 IPO準備・内部統制要件が強くなるフェーズで最適。

大量処理に対応　**大量の申請処理**　**複雑な会計処理**

バックオフィス（Salesforce×バクラク×勘定奉行クラウド）

CRMツールから「債権奉行クラウド」を経由し、「勘定奉行クラウド」へ仕訳データを連携するシステムフローを構築します。また、社内の申請や稟議などの各種ワークフロー業務については、「バクラク」を導入して運用を行います。



経営可視化：Notion AI × Apache Superset

日常の予実管理はNotion AI、全社横断の統合分析はApache Supersetと役割を分担。両者を組み合わせることで経営判断の速度を上げる

Notion AI：日常の予実・ナレッジ

自然文で入力→ダッシュボード反映

「株式会社〇〇 案件100万円」と入力するだけで売上管理シートやパイプライン、MRRのダッシュボードへ反映。入力の摩擦を下げることで営業側が数字更新を後回しにしない状態を作る。

ナレッジ横断検索

過去の提案書やプロジェクト資料に対して自然文で問い合わせ。「過去に〇〇業種で何をやったか」を短時間で拾い直せる。

Notionではやらないこと

大量のデータを一度に更新する処理や複雑な集計には向かない。大量データ処理・全社横断分析はSuperset側に任せる。用途を見極めて使うことが重要。

Apache Superset：全社統合可視化

全データを集約してリアルタイム可視化

Salesforce・kintone・WebAPP・freee・勘定奉行のデータを一元集約。予実・採算・工数をリアルタイムに可視化し、経営判断の速度を上げる。

OSS・ランニングコストゼロ

Tableauに比べてライセンス費が月額7〜10万円/ユーザーかかるのに対し、Supersetは自社運用でゼロコスト。機能差は小さく、当社の用途では十分以上。

Claude連携で「数字が教えてくれる」

財務KPIの変化をClaudeが自動検知し、Slackに文脈付きで通知。「数字を見る」から「数字が経営者に話しかける」状態へ変わった。

04

AI活用の土台

Claude Code ／ n8n ／ MCP をわかりやすく解説

AI活用の土台：3つの用語解説

Claude Code・n8n・MCPの3つを理解することで、当社がどのようにAIを業務フローに組み込んでいるかが分かる

Claude Code

クロード・コード

何か

Anthropicが開発したAIコーディングツール。ターミナル上で動き、コードを書いて・実行して・エラーを直して・外部サービスと連携するところまでを完結できる。

なぜ使うか

「AIに話しかけるとコードが動く」状態を実現。汎用AIチャットが「テキストを出力して終わる」のに対し、Claude Codeは実装まで完結する。コンサルが「提案して終わり」ではなく「動く仕組みを作れる」ようになる。

n8n

エヌエイトエヌ

何か

業務の自動化フローをノーコードで作れるオープンソースツール。「もしAアプリでXが起きたらBアプリにYをする」動作をプログラミングなしで設定できる。400種類以上のツールと接続可能。

なぜ使うか

Zapierに似ているが「自社サーバーで動かせる」ため機密データも安心。月額費用がかからない（自社インフラコストのみ）。Slack・kintone・freee・Google Drive等とシームレスに連携。

MCP

Model Context Protocol

何か

AIが外部のアプリやデータベースに直接アクセスするための「接続規格」。Anthropicが策定。Claude Codeに「手足」を持たせる仕組み。

なぜ使うか

MCPを使うことでClaude Codeがkintone・Slack・freeeなどを直接操作できるようになる。「AIに指示するだけで外部システムが動く」真のAI自動化を実現する。

Claude Code 5つのユースケース

「会話で終わる」を脱し、業務フローに直接組み込むことで提案・実装・発信・経営判断のサイクルを1人で回せるようになった

UC1

提案書
自動生成

商談メモ＋過去案件ナレッジ→提案書ドラフト。作成時間60〜70%削減

UC2

要件定義・
タスク自動化

議事録→要件構造化→タスク分解→kintoneへ担当者別自動登録。30分以内完了。

UC3

GitHub文化・
神スキルズ

作業終了時にGitHub自動更新。凄腕の暗黙知が組織資産に。再利用率約40%。

UC4

知見の
記事化

GitHub/Notion/商談履歴→記事構成・ドラフト生成。公開本数が約3倍に

UC5

財務情報×
経営自動化

財務KPI変化をClaudeが検知→文脈付きSlack通知。月次サマリー自動生成

05

ユースケース詳細

UC1 & UC2 提案書自動生成・タスク自動化フロー

商談後のヒアリングテキストをインプットすると、提案書ドラフト生成からkintoneへのタスク登録まで一気通貫で完結する

商談・ヒアリング	テキストインプット	課題構造化（Claude）	提案書ドラフト	タスク分解	kintone自動登録	人間が確認・調整
Google Meet Notion議事録	議事録・メモをClaude Codeに渡す	要件・制約をMECE整理	過去ナレッジ参照して生成	担当者・期限粒度を統一	担当者別に自動振り分け	精度確認と文脈合わせ

提案書の「0→80点」はClaude Codeが担当し、人間は「80→100点」の精度向上と文脈調整に集中する。
ナレッジが蓄積されるほど次の提案の精度が底上げされる。

60～70%	30分	ゼロ
提案書作成時間削減（ドラフト完成まで）	ヒアリング→タスク登録（以前は半日以上）	抜け漏れ・属人化（自動登録で担保）

UC3 GitHub文化と「神スキルズ」の仕組み

作業終了時にGitHubへ自動反映・README自動生成。凄腕コンサルタントの暗黙知が組織資産になり、全員のベースラインを底上げする

GitHubを使う3ルール

1 READMEを必ず書く

Claude Codeが自動生成。後から誰でも使える状態をゼロコストで担保。「後から追加しよう」は絶対にしない。

2 汎用部分と固有部分を分ける

再利用できる核を切り出し、個社対応はラッパーとして被せる設計。プラグイン化・製品化の土台にもなる。

3 コードレビューにClaude Codeを使う

APIキー平文記述などの初歩ミスを人間レビューの前に自動検知。セキュリティ・パフォーマンス・可読性を事前チェック。

コード再利用率 約40%（新規実装の4割が既存コードの流用・改修でカバー）

「神スキルズ」が生まれる仕組み

1. 凄腕コンサルタントが業務でコードや設計を作る
2. 作業終了時にGitHubへ自動コミット・README自動生成
3. 他のメンバーが「過去に似たものないか」でGitHub検索
4. 神スキルズを参照・流用・改善して業務に活用
5. 組織フローに昇格すべきものはn8nで標準化

📊 Anthropic社内データ：Claude Code Code Review導入後、コードレビューコメント付きPRの割合が16%→54%に向上。エンジニア1人あたりのコード出力量は前年比+200%

GitHubチェンジマネジメント：1ヶ月伴走で全員マストへ

「気を遣いすぎた失敗」から学んだ組織変革の方法論。最低公約数に合わせるのではなく、全員が上のレベルに上がれるよう設計する

Before 気を遣いすぎた失敗

「非エンジニアに合わせてGitHubを使わない設計にした」

→ コード共有が滞り、生産性の差が広がり続けた

→ 話が通じなくなり、組織が2層に分裂した

→「優しさ」が組織の足枷になっていた

After 方針転換：1ヶ月死ぬほど親切に

GitHubのコミット方法・READMEの書き方・レビューの受け方

質問があれば何度でも付き合う。怒らない・詰めない・諦めない

1ヶ月後は「マスト」として運用を切り替える。ラインを引く

メンバーも腹が決まった。「やれる」と分かった時点で加速した

変化を根付かせる4原則

1. 差は「技術力」ではなく「チャレンジできるか」。エンジニア出身かどうかは関係ない
2. 「最低公約数に合わせる」ではなく「全員が上のレベルに上がれるよう設計する」
3. 最初の1ヶ月だけ集中して伴走し、その後は同じ土台で走れるように引き上げる
4. ラインを引いた後は揺らがない。「マスト」として運用することで全員の腹が決まる

Claude Code×コードレビュー：業界の実データ

AI活用によりコード生成量が急増し、コードレビューがボトルネックに。Claude Code Code Reviewで「量×質」を同時に解決した事例が続々登場している

Anthropic社内データ（2026年3月 Code Review機能リリース時に公開）

200%

エンジニア1人あたりコード出力量（前年比増加（Claude Code活用後））

16%→54%

レビューコメントが付いたPR割合（Code Review導入前後）

3.2日→1.8日

PRの作成～マージまでの時間（コードレビューAI活用後）

国内企業の活用事例

LINEヤフー（2026年1月）

Claude Code × MCP でPRレビュー準備を自動化

育休復帰中のエンジニアが、Claude Code × MCP（Jira/Confluence連携）でPR内容の自動サマリを作成。週6時間のレビュー工数削減を実現。「まるでプロジェクト専用のドメインエキスパートが手元にいるような感覚」と語る。

週6時間削減

TOKIUM（エンジニア3人チーム）

AI活用で週30PR以上を少人数で回す開発体制

週30本以上のPRをエンジニア3人で回す開発体制をClaude Codeで実現。「朝10本のPR通知を見て『どれから手を付けよう』と悩んでいた時間がなくなった」。AIが先にレビューし、人間は設計判断に集中。

週30PR/3人体制

Uravation顧問先IT企業（エンジニア8名）

3ヶ月でコードレビュー指摘件数42%減・実装速度1.7倍

コードレビュー前にClaude Codeチェックを必須化。PRの作成～マージまでの平均時間が3.2日→1.8日に短縮。レビューの平均所要時間は45分→12分へ。AI非使用と比較して10倍以上の開発生産性向上を達成した事例も。

指摘件数42%減
実装速度1.7倍

UC4 知見の記事化——この資料自体がその実例

GitHub・Notion・商談履歴に蓄積した知見を参照させ、記事構成・ドラフト生成を自動化。人間が編集・磨くことで公開頻度が約3倍に

GitHub ナレッジ	Notion 提案履歴	商談・ 議事録
コード・設計・ 技術知見	過去案件・ ヒアリング内容	お客様からの 生の課題

↓ Claude Codeが統合・構成案を生成 ↓

Claude Codeが行うこと

- 読者ペルソナ・SEO観点での構成チェック
- 記事ドラフト生成（骨格＋各セクションの文章）
- 読者が疑問を持ちそうな箇所の補足提案
- 関連記事へのリンク候補の洗い出し

人間がやること（残り20%）

ファクトチェック・クライアント固有の文脈追加・語調の調整・最終確認。AIが「80点のドラフト」を作り、人間が「100点に磨く」分業で公開コストを大幅削減。

成果

記事公開本数が約3倍に増加。「書きたいけど時間がない」から「書ける体制になった」に変わった。なお、本記事（前編・後編）もこのワークフローで生まれています。

UC5 財務情報を基にした経営の自動化

freee / 勘定奉行のデータをApache Supersetに自動集約し、財務KPIの変化をClaudeが検知して文脈付きSlack通知を送る仕組みを構築

財務データ	freee / 勘定奉行 日次・月次
↓	
自動集約	Apache Superset リアルタイム
↓	
Claudeが分析	KPI変化・異常値 予兆を自動検知
↓	
Slack通知	文脈付きで 経営者に即時
↓	
意思決定	翌日の経営判断 に反映

Slackに届く通知のイメージ

📊 月次財務サマリー（自動生成）

A顧客プロジェクトの工数が先月比20%増加しています。受注金額は変わっていないため、このまま進むと採算悪化の可能性があります。追加費用の協議または作業範囲の見直しをご検討ください。

→ 詳細はダッシュボードで確認

経営判断にもたらした変化

- ⚡ 数字が出る翌日に動ける（月次終わりを待たない）
- 🔍 「予兆」の段階でリスクを検知できるようになった
- 📋 月次経営会議前に「数字の読み方の仮説」が手元にある

n8n：自動化の標準化と個人効率化の二層構造

すべての自動化をn8nに集約せず「組織フロー層」と「個人効率化層」に分けることで、自由と統制を両立する

組織フロー層（n8nで管理）

複数メンバー・部門をまたぐ処理 ／ セキュリティリスクが絡む情報の流れ ／ 会社の意思決定に影響するデータ連携

⚠️ 重要：標準化する前に「組織のフローとして正しいか」をちゃんと設計する。間違った自動化を標準化すると、間違いが高速で大量に起きる。

- 採用応募→通知→担当者振り分け
- 財務KPI変化→Slack通知（文脈付き）
- 請求書PDF→Drive振り分け→freee連携
- 月次サマリー→自動生成

個人効率化層（GitHubに自動蓄積）

個人の業務スタイルに合わせた自由な効率化。全員に同じやり方を強制せず、各自が最適解を発見 → 作業終了時にGitHubへ自動反映・README自動生成

- 提案書テンプレの自動生成スクリプト
- 採用応募→Slack通知の整形
- 議事録→特定フォーマット整形
- マニュアル自動生成(キャプチャ→説明文)
- kintone検索クエリのショートカット
- 個社最適の営業文面生成

非エンジニアがAIでコードを書くリスクと対策

AIで「動くコードが書ける」ようになった半面、セキュリティ・情報管理の考慮漏れリスクが増大。自社事故から学んだ3つの対策を徹底している

社内で実際に発生したヒヤリハット

CASE 1

APIキーの平文記述

Claude Codeで作ったスクリプトにAPIキーをコード内に直書き。GitHubにプッシュすれば外部漏洩のリスクがあった。「環境変数で管理する」という原則はエンジニアには常識でも、書き始めたばかりの人間には完全な盲点になる。

CASE 2

Slackへの機密情報の自動投稿（未遂）

社内情報共有の自動化スクリプトが「これは共有してはいけない情報では？」という内容を投稿しそうになった。自動化は便利だが、「何を・誰に・いつ共有するか」の判断をコードに委ねると、人間なら当然気づく文脈の問題を見逃す。

人間は本当に怖いものです（笑）

当社が導入した3つの対策

01 社内開発に限定

Claude Codeで作ったコードは本番環境へ直接デプロイしない。必ず社内での動作確認を経てから適用する。

02 コードレビューの徹底（二重チェック）

Claude Codeが初期レビュー（セキュリティ・情報漏洩リスク・エラーハンドリング）→人間エンジニアが最終確認。特に非エンジニアが書いたコードほど丁寧にレビューする。

03 テストケースを必ず書く

「動いた」で終わらせない。自動通知・自動投稿を含む処理は、想定外ケースのテストを済ませてから本番稼働させる。考慮漏れとテストケースの重要性を痛感した。

副産物：採用自動化で生まれた仕組み

Claude Codeで別の自動化を作る過程で「これも同じ仕組みでできるな」と気づいて横展開したもの。当初は想定していなかった成果

採用媒体から応募	Slackへ自動通知	kintone DB登録	面談後サマリー	オンボーディング
複数媒体を一元受信	返信が必要な候補者を整理	候補者情報を自動登録	会話内容をマネージャーに投稿	入社日に合わせ自動トリガー

自動化のコストが下がると、「やれたらいいな」がどんどん「やってみよう」に変わる。小さな副産物が積み上がり、気づけば組織全体の運用が変わっている。

📬 応募通知の自動整理

採用媒体からの応募が届くと、返信が必要な候補者の一覧が自動でまとめられ担当者にSlack通知。対応漏れがゼロになった。

💬 面談サマリーの自動投稿

候補者との面談後、会話の概要と重要ポイントがマネージャーのSlackチャンネルに自動投稿。全面談に同席しなくても判断に必要な情報がリアルタイムで届く。

🚀 オンボーディングの自動化

入社日に合わせ、必要なアカウント発行手順・初日スケジュール・確認事項がトリガー。担当者への通知と候補者へのウェルカムメッセージが自動で動く。

06

成果と組織設計
1人8倍が実現できる理由

1人8倍の生産性——定量的変化

「設計→実装→検証→発信→経営判断」のサイクルが1人で完結するようになり、ROIが合わなかった施策を「まず試す」判断が現実的になった

業務カテゴリ	導入前（時間）	AI（Claude Code等）による変化	導入後（時間）	備考
開発業務	100時間	▲ 80時間（80%減）	20時間	コーディング、 テスト作成等の自動化
採用業務	30時間	▲ 20時間（約67%減）	10時間	スカウト文面作成、 レジュメ要約の自動化
提案業務	20時間	▲ 10時間（50%減）	10時間	提案書の構成案、リサーチ、 資料化の効率化
提案活動	10時間	＋ 70時間（削減分を再投資）	80時間	顧客との対話、商談、 直接的な価値提供
合計	160時間	± 0時間	160時間	1人の月間労働時間は変わらず

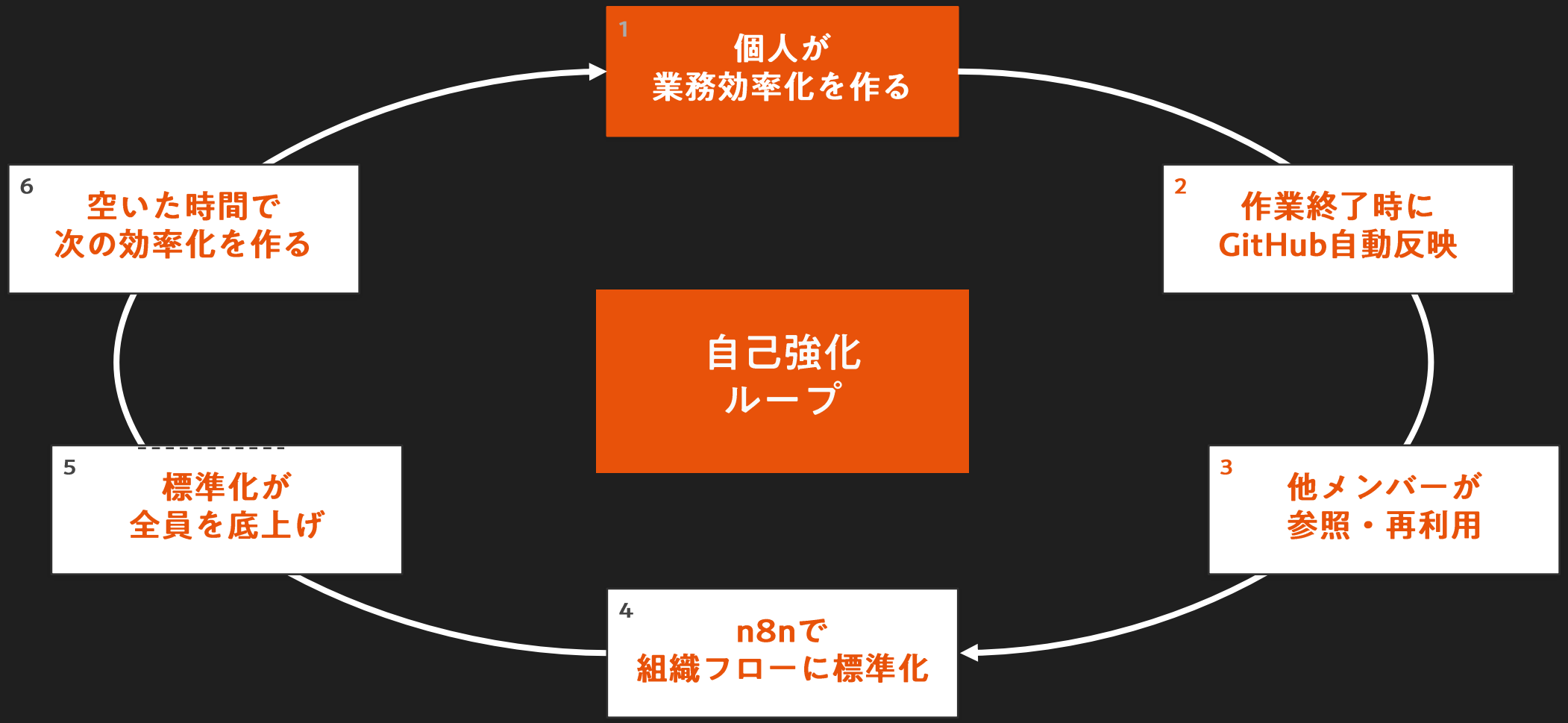
定性的変化：「経営判断」と「現場判断」の距離が縮まった

ツール導入そのものが目的ではない。生まれた時間で次の一手を打ち続ける運用設計こそが、DXの成果を分けた

	Before（導入前）	→	After（導入後）
	提案書1本を書くのに数日～数週間かかっていた	→	ドラフトが数時間で出る。人間は「磨き」に集中できる
	ヒアリングからタスク登録まで半日以上の手作業が必要	→	30分以内で完了。「抜け漏れ」の心配がなくなった
	コードを書くたびに「過去に似たものがないか」が分からない	→	GitHubで検索。神スキルズが車輪の再発明を防ぐ
	月次で財務を振り返るだけで、予兆を拾えなかった	→	KPIの変化を翌日に検知。先手を打てる経営に変わった
	記事を書く時間がなく、知見が外に出なかった	→	Claude Codeが下書きを作り、公開本数が3倍になった

組織全体のレベルが上がっていくサイクル

個人の工夫がGitHubに蓄積され、組織フローに昇格してn8nで標準化される。この自己強化ループが「少人数・高回転」を可能にする



「AIを使いこなせる人と使えない人の差が広がる」という課題への答えがこのサイクル。差は「固定された差」ではなく、神スキルズを通じて「後から追いつける差」になる。

AIを使いこなす組織の作り方

差は「技術力」ではなく「チャレンジできるか」。1ヶ月の集中伴走でGitHubを全員マストにした経験から、変化を根付かせる方法論が生まれた

差は「技術力」ではなく「チャレンジできるか」

使いこなせる人に共通するのは「まず試してみる」ができること。エンジニア出身かどうかは関係ない。「前のやり方への固執」がある人ほど変化が遅い。これは責めているのではなく、人間として自然な反応であり、だからこそ環境設計が重要。

「気を遣いすぎ」は相手への配慮ではない

非エンジニアに合わせてGitHubを使わない設計にしたことで、コード共有が滞り生産性の差が広がり、話が通じなくなった。「優しさのつもりが、変化を先延ばしにしているだけだった」。

「1ヶ月は死ぬほど親切に」の伴走設計

GitHubの使い方を、質問があれば何度でも付き合う。怒らない・詰めない・諦めない。1ヶ月後は「マスト」として運用を切り替える。ラインを引いたことでメンバーも腹が決まった。

「最低公約数に合わせる」から「全員が使えるようにする」へ

ツールのレベルを下げて全員が使えるようにするのではなく、全員が上のレベルに上がれるよう設計する。これが「組織レベルが上がるサイクル」の前提条件になる。

n8n × GitHub：自動化の全体アーキテクチャ

個人の効率化（GitHub）→組織の標準化（n8n）→全員の底上げというサイクルを、具体的なツール接続の観点から整理する

個人効率化（GitHub自動蓄積）	Claude Code / kintone API / Google Drive / freee API	個人が作ったスクリプト・自動化・プラグインが作業終了時に自動でGitHubへ。README自動生成で「後から誰でも使える」状態に。
組織標準化（n8n フロー管理）	Slack / kintone / freee / Google Drive / Gmail / GitHub	個人効率化の中で「組織フローに昇格すべきもの」だけをn8nで標準化。セキュリティリスクが絡む処理・複数部門をまたぐ処理はここで管理。
経営可視化（Apache Superset）	Salesforce / kintone / freee / 勘定奉行 / WebAPP	蓄積されたデータをSupersetで集約・可視化。Claudeが変化を検知してSlack通知→経営判断のループが日次で回る。

このアーキテクチャにより、個人の工夫が組織の標準になり、組織の標準が個人のベースラインを引き上げる自己強化ループが回り続ける。

自社実践→顧客提案への転用メカニズム

自分たちが使って効果を感じていない仕組みをお客様にお勧めすることはできない。だからこそ自社で試し、失敗し、最適化した

一般的なAIコンサル

△ 他社事例を借りる

「A社はこのツールで成功した」という事例を横展開。自社では実際に使ったことがない。

△ PoC止まり・本番稼働しない

概念実証（PoC）は作れるが、自社で毎日動かし続けている実績がない。

△ コスト感覚がない

実際に月額費用を払い、ランニングコストを肌で感じた経験がないため、リアルな比較ができない。

Aurant Technologies

◎ 今まさに自社で動かしている

今日この瞬間も動いているシステムの話をしている。失敗した経験も含めてリアルに伝えられる。

◎ 本番稼働済みの実践知

自社でClaude Code・n8n・kintone×WebAPPを毎日使い、数字で変化を確認している。

◎ コストの比較を自社で経験

Salesforce→kintone、クラウドサイン→freeeサイン等、実際に費用を払って比較した上で選定している。

今後の展望：現在構成はどこまでスケールできるか

今の構成は何人・何億円規模まで耐えられるか。フェーズ移行のトリガーと、スケール時に変わること・変わらないことを整理する

▼ 現在地

現在フェーズ

～数十名・数億円

変えないもの

- ✓ kintone × WebAPP
- ✓ Claude Code × n8n
- ✓ freee × freeeサイン
- ✓ Apache Superset（OSS）

拡大フェーズ

数十～100名規模

変えないもの

- ✓ Claude Code × n8n
- ✓ Apache Superset

追加・移行するもの

- → kintone → Salesforce追加（大口営業管理）
- → freee → 債権奉行連携（内部統制強化）

大規模フェーズ

100名～IPO準備

変えないもの

- ✓ Claude Code × MCP（変わらず中心）
- ✓ Apache Superset
- ✓ n8n

追加・移行するもの

- → Salesforce（全社CRM）
- → Salesforce × 勘定奉行 × バクラク
- → 監査対応・内部統制の強化

「Claude Code × n8n × MCP」の組み合わせはどのフェーズでも中心に残る。変わるのは周辺のSaaSであり、AI活用の設計思想そのものは変わらない。

よくある質問（Q&A）

自社実践を公開してから最もよく受ける質問と、当社の回答をそのままお伝えします

Q Claude Codeを使いこなすのに、技術的な素養はどれくらい必要ですか？

基本的なターミナル操作と「コードを読んで意図を確認できる」程度あれば十分です。コードを書く能力は不要です。ただし「動いたからOK」ではなく「なぜ動いているかを確認する習慣」は必須です。当社でも非エンジニアが使っており、最初の1ヶ月の伴走が鍵でした。

Q n8nの導入難易度はどれくらいですか？

ノーコード寄りのUIなので直感的に使えます。ただ、自社サーバーで動かす場合はインフラの知識が必要です。最初はクラウド版から始めることをお勧めします。重要なのはツールより「何を組織フローとして標準化するか」の設計です。

Q kintoneはどのくらいの規模まで使えますか？

数百名規模まではkintoneで十分な場合が多いです。それ以上になってくると、内部統制要件やデータ量の観点でSalesforceや基幹システムへの移行を検討するフェーズになります。

Q 今日話してくれた構成、全部自社で作ったんですか？

はい。kintoneのカスタマイズ、WebAPP、GitHub上のスクリプト群、n8nのフロー設計——これらは基本的に自社で内製しています。その経験があるからこそ「ここは内製した方がいい」「ここはSaaSで十分」という具体的な提案ができます。

Q AIリスクの話が出ましたが、社内のセキュリティポリシーはどう整備しましたか？

大きく3点です。コードは必ず社内でレビューしてから本番適用する、Claude Codeによる自動レビューを必須工程にする、情報の共有範囲をフロー設計の段階で明文化する。「動いた」で終わらせない文化が一番大事だと思っています。

AIを使わない場面——境界線の設計

AIが万能だという幻想を持つのは危険。「AIに任せる領域」と「人間が判断する領域」の境界線設計こそが、当社が最も大切にしていること

AIに任せること

- ✓ 提案書・要件定義・タスクの初期ドラフト生成
- ✓ コードの初期実装・レビュー・ドキュメント生成
- ✓ 財務KPIの変化検知と通知（パターン認識）
- ✓ 記事の構成案・ドラフト・SEOチェック
- ✓ 採用応募の通知整理・候補者DBへの自動登録
- ✓ 繰り返し作業の自動化（ルールが明確なもの）

人間が判断すること

- ◎ 最終的な提案内容・提案するかどうかの判断
- ◎ 何を組織フローとして標準化するかの設計
- ◎ 顧客との関係性・文脈に基づく交渉・説得
- ◎ 新しいユースケースの発見と「試してみよう」の決断
- ◎ AIが出した結果のファクトチェックと責任の所在
- ◎ 倫理的判断・機密情報の取り扱いポリシー

ご相談・お問い合わせ

「うちはどこから手をつければ良いか」が曖昧な場合も、現状のツールと業務フローから一緒に整理できます

こんな課題をお持ちの方へ

- SaaSコストが増え続けているのに業務が楽にならない
- AI活用の着手点が分からない・何から始めれば良いか迷っている
- kintoneやSalesforceを入れたが、うまく使いこなせていない
- Claude Code・n8nを試してみたいが、社内に知見がない
- 成長フェーズに合ったシステム構成を知りたい
- 自社の業務フローをAIで自動化できるか確認したい

まずは無料でご相談ください

現状のシステムコストの無駄を洗い出し、成長フェーズに合った最適なシステム構成をご提案します。

お問い合わせ（無料）

APPENDIX

自動化事例集

Claude Code事例 ／ n8n自動化事例 ／ kintone MCP対応 ／ 業界トレンド

Claude Code × freee / kintone：外部事例

Zenn・各社ブログで報告されている実践事例。Claude Codeと業務SaaSを組み合わせることで、専任エンジニアなしでも高度な自動化が実現できている

事例	使用ツール	実現した自動化	効果・数字
freee × Claude Code（個人事業主・月次経理）	Claude Code freee MCP	毎月の仕訳チェック・試算表分析・請求書照合を自然言語で自動化。人手なら件数に比例して時間が増えるが、AIは月100件でも数千件でも処理時間はほぼ一定	**30分→数分（月次確認作業）**
freee × Claude Code（ソロデザイナー）	Claude Code freee MCP	自然言語で損益計算書を取得し、174件の取引を一括処理。Studio CMSのAPI制約・freeeの限界と回避策まで一人で実装	**半日の経理作業を数時間に圧縮**
kintone × Claude MCP（見積書管理カスタマイズ）	Claude Desktop kintone MCP	見積書管理アプリのカスタマイズ要件をClaudeが分析・提案。「フィールド追加したい」と日本語で依頼するだけで実装	**開発工数大幅削減**
kintone × Cowork AI（日常業務指示）	Claude API kintone Plugin	「今月の売上を集計して」「〇〇商事の商談を進行中に更新して」と話しかけるだけでデータ検索・集計・レコード更新をAIが自律実行	**手作業なし日本語で完結**
開発50機能ERP構築（内装・建設業・1人）	Claude Code TypeScript/Next.js	kintoneとfreeeの組み合わせに限界を感じ、Claude Codeと2人で業務ERPをゼロから構築。問い合わせ→現調→見積→契約→施工→検査→請求→入金の全パイプライン	**月額4,000円50機能を3ヶ月で**
Zennチーム（エンジニア3人）オートパイロット開発	Claude Code GitHub	バグ修正・小規模機能追加・お知らせ機能開発などをClaude Codeに投げてオートパイロット化。「あとはやるだけ」という小さいタスクに分解することが近道	**開発速度大幅向上**

注目事例：kintoneを捨てて1人でERP50機能を3ヶ月で構築

内装・建設業の中小企業で働く「なんでも屋」が、Claude Codeと2人でゼロから業務ERPを構築。月額ランニングコスト4,000円で50機能以上が本番稼働（Zenn掲載・2026年3月）

なぜkintoneを捨てたのか

内装業の案件は「問い合わせ→現調→見積→契約→施工→検査→請求→入金」という長いパイプラインを通る。kintoneはこのフローに合わない。

× 無理にカスタマイズすると、現場が入力を嫌がって使わなくなる

× freeeとkintoneの組み合わせに限界を感じ始めた

× 「自分の業務に100%フィットするものを作った方が早い」と判断

技術構成

TypeScript + Next.js App Router + Tailwind CSS + Claude Code。「Claudeが最も正確にコードを生成できるスタック」として選定。Server Actionsが強力で、DB操作を含むバックエンド処理が1ファイルに完結。

成果と方法論

50+	3ヶ月	¥4,000
本番稼働中	構築期間	ランニングコスト

分業の「型」

何を作るか	→ 人間が決める
どう作るか	→ AIに投げる
検品・レビュー	→ git diffで確認

MEMORY.md（ドメイン知識＋ファイルマップ＋フィードバックを記載）が「会社固有の設計仕様書」として機能。新セッションでもClaude Codeがプロジェクト全体を即座に把握できる。

n8n × kintone 業種別自動化事例

記事・ブログから収集した実践事例。いずれも「ノーコード自動化」と「kintoneをデータハブとして活用」という共通パターンがある

📣 広告代理店	**商談管理・営業レポートの全自動化** Webフォームから問い合わせ → kintone「見込み客」アプリにレコード自動作成 → ステータス変更でSlack通知 → 受注で請求書発行システムに自動連携 → 月次でGoogleスプレッドシートに自動出力	効果：営業事務の作業時間が月40時間→5時間に削減。営業担当者は顧客対応に集中できるようになった
📦 EC・在庫管理	**マルチEC在庫の自動同期と発注書自動生成** 楽天・Amazon・自社サイトの在庫情報をn8nで15分ごとに取得 → kintone在庫管理アプリに自動反映 → 閾値を下回ったら発注書を自動生成・仕入先にメール送信 → 入荷完了で全EC在庫数を自動更新	効果：在庫切れによる機会損失が80%減少。各プラットフォームへの個別ログインと手動更新がゼロに
📄 経理・請求書処理	**請求書処理の全自動化（月200件超）** 請求書PDFがメールで届く → n8nがトリガー → AI-OCRで金額・取引先・日付を抽出 → 生成AIで整形してkintoneに自動登録 → freeeへ仕訳データを自動連携 → 承認者にSlack通知	効果：月間200件以上の請求書処理が自動化。経理部門の業務負荷が60%削減
⏰ 人事・勤怠管理	**勤怠データ収集・給与計算連携の自動化** kintone打刻アプリからn8nが勤怠データを取得 → 深夜残業・休日出勤など異常な打刻を自動検知 → 管理者にアラートメール送信 → 月末に給与計算システムへ自動連携	効果：人事担当者の月末業務が3日→半日に短縮。ミスも大幅に減少
🏭 製造業（kintone × n8n × AI）	**PDF書類OCR→kintone自動記入** kintoneに請求書・注文書の画像/PDFが添付される → Webhookでn8nが起動 → AI OCRで文字情報を抽出 → 生成AIで整形・正規化 → kintone REST APIでレコードを自動更新・Slackに通知	効果：「添付→転記作業がゼロ」を実現。紙書類の検索不能・転記ミス・手作業の三重苦を解消
👥 採用・HR（AI評価）	**履歴書の自動評価・スコアリング** Googleフォームで応募者が履歴書PDF送信 → n8n Webhookが起動 → PDF内テキストを抽出 → AIが職務記述書と照合してスコアリング・評価コメント生成 → 担当者にSlack通知・スプレッドシートに記録	効果：数百件の履歴書を数分で評価。主観的判断によるばらつきをなくし、評価の一貫性を確保

kintone 公式MCP対応とClaudeエコシステムの広がり

2025年、kintoneがClaude MCP公式サーバーを提供開始。「kintone × Claude」がノーコードで実現できる時代になり、周辺のツールやサービスが急速に増えている

kintone 公式MCPサーバー（2025年～）

何が変わったか

これまでkintoneとClaudeを連携するには独自のAPIコードを書く必要があった。公式MCPサーバーの提供により、.dxtファイルをドラッグ＆ドロップするだけで5分でClaudeからkintoneを直接操作できるようになった。

できること

kintoneのレコード検索・取得・登録・更新・削除、アプリ構造の分析、添付ファイルのダウンロードをClaude Desktop / Claude Codeから直接実行。「今月の受注一覧を出して」と日本語で話しかけるだけ。

セキュリティの考え方

APIトークン認証で最小権限を設定（読み取り専用/書き込み分離）。kintone側に実行ログ（日時・実行ID・操作内容）を残す設計が推奨されている。本番運用にはAPIトークン認証と権限分離が必須。

周辺エコシステムの広がり

1 Cowork AI for kintone

kintoneプラグインとして動くClaudeエージェント。自然言語でレコード検索・集計・更新・アプリ自動作成まで対応。無料・OSS（MIT）で公開中。

2 CData kintone MCP

CData社が提供するkintone × Claude Desktop連携。Salesforce・kintoneなど400種類以上のデータソースをMCPで接続できる企業向けソリューション。

3 n8n kintone連携テンプレート

Webhookノード経由でkintoneとn8nを接続するテンプレートが多数公開。生成AIとの組み合わせで問い合わせ自動分類・レコード自動更新が標準化されつつある。

4 freee MCP

freee公式MCPサーバーにより、Claude Codeから直接仕訳チェック・試算表取得・請求書発行が可能に。会計×AIの組み合わせが急速に普及中。

業界全体の潮流とAurantの立ち位置

n8nは2025年Series Cで約270億円を調達・評価額25億ドルに到達。Claude MCP対応ツールが急増中。「SaaS×AI自動化」は特殊スキルから標準インフラへ変わりつつある

T1

AIエージェント化の加速

汎用AIチャットから「自律的に動くAIエージェント」へのシフトが急速に進行。n8n × Claude Code × MCPの組み合わせが「AIに手足を持たせる」標準構成になりつつある。

Claude Codeプラグイン：
2026年4月時点で160種類以上

T2

SaaSエコシステムのMCP対応

kintone（公式MCP）・freee（freee-MCP）・Salesforce・HubSpot・Slackなどが相次いでMCPサーバーを提供。AIが直接SaaSを操作できる環境が整備されてきた。

n8n 接続ツール：
2025年時点で1,000種類超

T3

1人・小規模チームによる高度開発

エンジニア3人のZennチーム・1人の内装業担当者がERPを自作するなど「AIが人数の壁を壊している」事例が急増。投資対効果の概念が根本的に変わっている。

n8n Series C調達額：
約270億円（$180M）

Aurantの立ち位置

◎ 業界のトレンドに乗って「試している」のではなく、毎日自社で動かしながら「使えること・使えないこと」を体系化している

◎ n8n・Claude Code・kintone・freeeのいずれも自社の本番環境で稼働中。外部事例の検証と自社実践の差を埋め続けている

◎ お客様への提案は「他社事例を借りる」ではなく「今この瞬間自分たちで動かしている仕組みをそのまま提供する」

お問い合わせはこちら

info@aurant-technologies.com