RapidStart

ホワイトペーパー

OpenClawのビジネス展開

ローカルのエージェントから、安全で信頼でき、管理可能なビジネス基盤へ。

RAPIDSTART · 2026年8月

アーキテクチャ · デプロイ · セキュリティ
ガバナンス · 運用

要旨

エグゼクティブサマリー

OpenClawが魅力的なのは、際立ってオープンだからである。特定のクラウド、モデルプロバイダー、ID基盤、運用形態を組織に強制することなく、セルフホスト型のエージェントランタイムを提供する。この柔軟性こそが特長であり、同時に展開負担の源でもある。

ビジネス利用では、作業はインストールの先へ急速に広がる。組織は永続的なホストを用意し、IDと資格情報の境界を定義し、モデルと業務システムを接続し、エージェントに許す行為を制約し、プロンプトインジェクションや過剰な権限といったエージェント固有の脅威に備え、監視・更新・復旧の体制を確立しなければならない。

結論は実務的である。OpenClawを選ぶ組織は、その周囲の環境をシステムの一部として扱うべきだ。健全なデプロイパターンは、インフラ・セキュリティ・運用に関する判断を明示し、繰り返せるものを自動化し、ワークロード固有として残る部分を文書化し、環境を再構築可能な状態に保つ。

> **OpenClawが提供するのはランタイムである。本番運用への到達は、その周囲のアーキテクチャ、統制、運用実務によってもたらされる。**

4つの結論

01 **OpenClawが有力なのは**、完全マネージドのSaaS運用モデルよりも、ランタイムの所有、モデルの柔軟性、拡張性が重要な場合である。

02 **マネージド型エージェント基盤も有力な選択肢である**。プラットフォーム、ランタイム、運用負担の多くをベンダーに任せたい場合に適する。

03 **本番運用の成否を左右するのは**、エージェントランタイム自体と同等以上に、ID、権限、プロンプトインジェクション対策、オブザーバビリティ、復旧である。

04 **本当の到達点は**「OpenClawが動いている」ことではない。「予測可能にデプロイし、統制し、運用し、再構築できる」ことである。

01 OpenClawがビジネスにとって重要な理由

ビジネスAIにおける重要な変化は、モデルがより良い回答を返すようになったことではない。エージェントが常時稼働し、ツールを使い、コンテキストを保持し、複数のシステムにまたがって行動できるようになったことである。OpenClawは、セルフホスト型ゲートウェイ、チャネル、セッション、メモリ、ツール、スキル、マルチエージェントのルーティングを一つのランタイムに統合している。[1]

ビジネス上の利点	重要な理由
常時稼働エージェント	永続的なランタイムは、特定の従業員がチャット画面を開くことに依存せず、業務プロセスやチームに寄り添って動作できる。
業務システムへの操作	ツールとMCP接続により、エージェントは質問に答えるだけの存在から、CRM、ERP、Microsoft 365、APIを読み取り、それらに対して操作を実行する存在へと変わる。
モデルの柔軟性	ランタイムは複数のモデルプロバイダーやフェイルオーバー構成に対応して設定でき、単一のモデルがアプリケーションそのものになることを避けられる。[1]
ランタイムの制御	ランタイムをどこに置くか、どのように拡張するか、周辺アーキテクチャをどこまで自社で持つかを、組織が決定できる。

業務に沿って構築するエージェント

多くの企業には情報そのものは十分にある。摩擦が生じるのは、複数システムにまたがる情報を組み立て、次の一手を実行する段階である。エージェントは、コンテキストを収集し、それを踏まえて判断し、ツールを使って業務を前に進める結合層になり得る。これは全従業員に個人用チャットボットを配ることとは本質的に異なる。

> **長期にわたって稼働する業務エージェントでは、モデルはアプリケーションそのものではなく、エージェントの下に位置する基盤として扱えるようになりつつある。**

02 OpenClawの位置づけ

OpenClawの周辺にある選択肢は、直接の競合ではない。それぞれ異なるレイヤーに位置する。問うべきは、業務内容、統制要件、統合ニーズ、エンジニアリングへの意欲に対して、どの運用モデルが最も適合するかである。

選択肢	分類	強み	適する場面
Copilot Studio	マネージド型ローコードエージェント基盤	Microsoftネイティブの開発、コネクタ、ガバナンス、公開機能	マネージド基盤を望み、ランタイムの保有を最小限に抑えたい場合。[2]
Microsoft Foundry Agent Service	マネージド型エージェントホスティング	マネージドのエンドポイント、スケーリング、ID、オブザーバビリティを備えたプロンプト型およびホスト型エージェント	独自コードが必要でありながら、Microsoftによるマネージドホスティングとエンタープライズ統制を優先する場合。[3]
OpenClaw	セルフホスト型エージェントランタイム	ランタイムの所有、チャネル、ツール、スキル、モデルの柔軟性、オープンなアーキテクチャ	ランタイムと周辺アーキテクチャの選択を自社で保有したい場合。[1]
LangGraph / CrewAI	開発者向けフレームワーク	コードで実装するエージェントアプリケーションとオーケストレーションを最大限に制御できる	ランタイムを展開するのではなく、独自のエージェントアプリケーションを構築する場合。
Zapierなど	マネージド型AI自動化	SaaSとの迅速な接続とワークフロー自動化	ランタイムの所有よりも、SaaS自動化の網羅性が重要な場合。

マネージド基盤：ランタイムを持たない方がよい場合

マネージド型のエンタープライズ基盤が適切な場合もある。ローコードでの開発、Microsoftによる運用、ネイティブなガバナンスが主要要件であれば、Copilot Studioが強い。一方、ランタイムの所有、幅広い拡張性、モデルの柔軟性、周辺アーキテクチャを自ら形づくれることに価値を置くなら、OpenClawがより有力になる。問題はどちらの製品が優れているかではなく、組織がどの運用モデルを自ら保有したいかである。

マネージドホスティングと開発者向けフレームワーク

マネージド型のエージェントホスティングは、開発チームが独自コードを書きつつ、ホスティング、ID、オブザーバビリティの多くをプロバイダーに任せたい場合に有用である。LangGraphやCrewAIといった開発者向けフレームワークはスタックのより下層に位置し、エージェントアプリケーション自体をチームが構築する場合に適する。OpenClawは、独自のゲートウェイ、チャネル、スキル、運用モデルを備えた既存のセルフホスト型ランタイムである点で差別化される。[3][4][5]

03 OpenClawが適さない場合

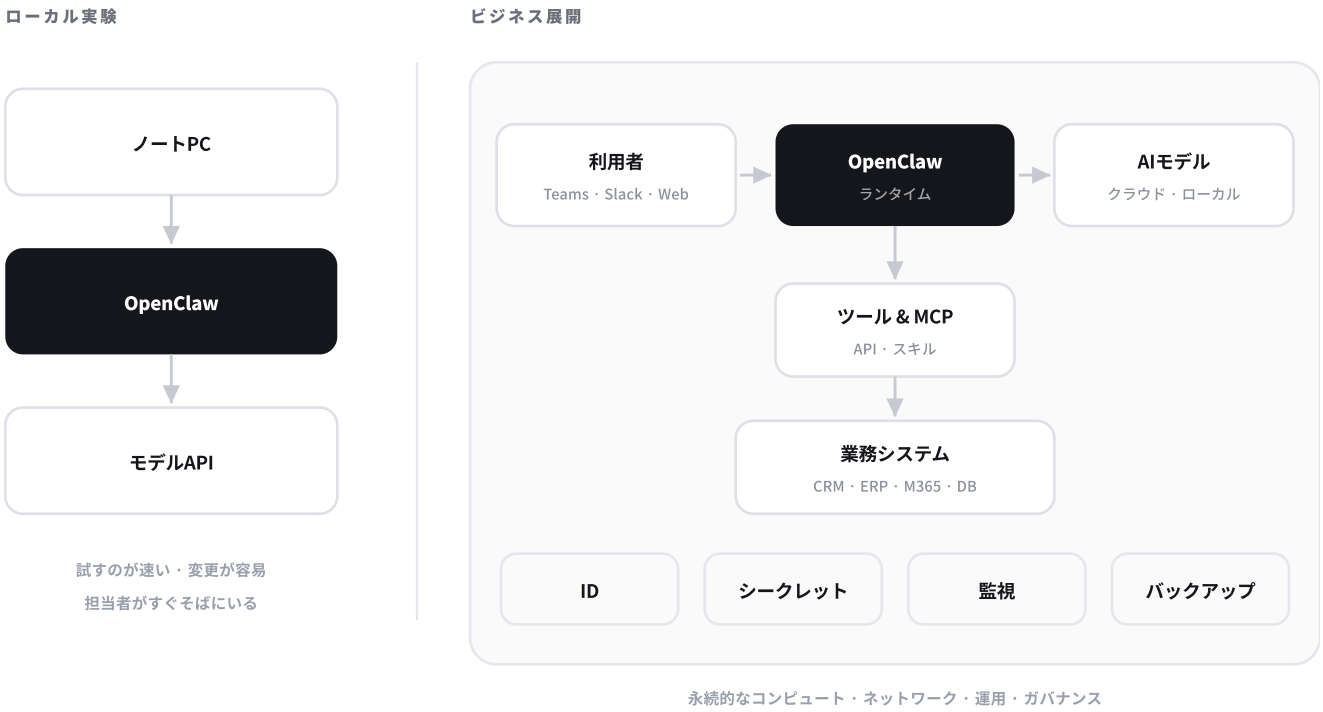
妥当なアーキテクチャ判断には、除外条件が含まれる。オープンであることや、ランタイムが強力であることだけを理由に、OpenClawが最適解になるわけではない。

除外条件	より適した対応
ベンダーが運用するランタイムとSLAを求めている	Copilot StudioやFoundry Agent Serviceのようなマネージド基盤の方が、運用面で適合する可能性がある。
課題が決定論的なワークフロー自動化である	Power Automate、Logic Apps、Zapierなどのワークフローツールの方が、簡潔で統制しやすい場合がある。
クラウド運用の担当者がいない	セルフホスト型ランタイムは、パッチ適用、監視、復旧、セキュリティの責任を生む。担当者がいないのであれば、その義務を作るべきではない。
自社チームでは満たせない厳格な規制要件がある	マネージドの統制機能、認証、サポート体制、監査可能性が要件に合致する基盤を選択する。
ランタイムの所有やモデルの柔軟性を必要としていない	OpenClawの主要な利点が、運用負担に見合わない可能性がある。
用途が単一の限定的なアシスタントである	汎用ランタイムよりも、マネージド型エージェントやアプリケーション組み込み機能の方が、明らかに簡潔になり得る。

> **オープンソースは戦略ではない。セルフホストに価値があるのは、それによって得られる統制が、生じる責任に見合う場合だけである。**

04 ローカルエージェントからビジネス基盤へ

OpenClawは最初の体験を意図的に容易にしている。開発や実験にはそれが優れている。しかしビジネス展開は、その一段手前から始まる。組織はまず、ランタイムが安定して存在できる永続的な環境を用意しなければならない。



クラウドVMが最も簡単なパターンであることは多いが、永続的なコンピュートを持てば、リージョン、OS、ストレージ、管理、再起動時の挙動、パッチ適用、監視、復旧といった選択が直ちに発生する。

> **開発者の問いは「動かせるか」である。ビジネスの問いは「既知の統制の下で、継続的かつ確実に動作する環境を用意できるか」である。**

05 本番展開のベースライン

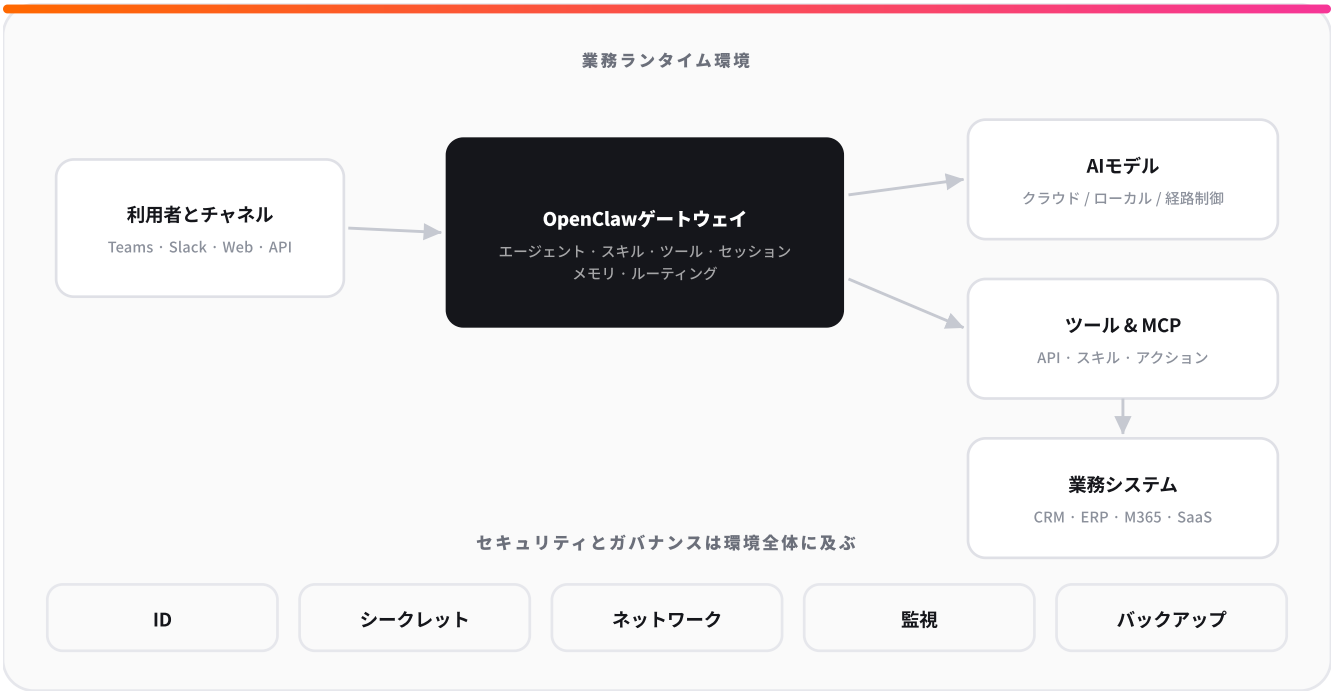
有用なデプロイガイドは、問いを並べるだけで終わってはならない。ここに示すベースラインは、OpenClaw環境をビジネス基盤として扱う前に、通常は意識的に決めておくべき統制と運用上の選択をまとめたものである。個々の製品やベンダーは変わり得るが、要件は変わらない。

判断項目	推奨ベースライン	理由
ランタイムのホスト	組織が保有する永続的なコンピュート	ランタイムを従業員の端末から切り離し、組織にとって明確な運用境界を与える。
組織のID	対応する場合は専用のワークロードIDを使用する[13]	エージェントを導入した個人の資格情報に本番アクセスを紐づけることを避ける。
リソースアクセス	プラットフォームが対応する場合はIDベースのアクセス[13]	長期間有効なシークレットへの依存を減らし、失効と監査可能性を高める。
シークレット	マネージドなシークレット保管。開発者端末に本番用シークレットファイルを置かない[15]	トークンや鍵を、アクセス制御・ローテーション・復旧手順を伴う管理対象資産として扱う。
モデルアクセス	承認済みのプロバイダーとモデル。リージョン、クォータ、フォールバック方針を文書化する[10][12]	モデル選定を開発者の好みではなく運用ポリシーとして扱う。
利用者接点	用途に適した、承認済みの業務チャネル	エージェントへのアクセスを、組織が統制できるコミュニケーション面の内側に留める。
運用	ヘルス、ログ、アラートの一元化と、文書化された再起動・更新手順	障害を可視化し、運用担当者に再現可能な対応手順を与える。
デプロイ	手作業で作り込んだ固有のVMではなく、スクリプト化された再現可能な構成[14]	再構築、2台目のインスタンス展開、復旧を現実的なものにする。

これらはベースラインとなるアーキテクチャ原則であり、普遍的な規程ではない。ネットワーク構成、個別の権限、保持期間、承認基準、復旧目標、承認済みサービスプロバイダーは、依然としてワークロードと組織に依存する。

06 ビジネス向けOpenClawのリファレンスアーキテクチャ

この図はリファレンスパターンであり、OpenClawが規定するアーキテクチャではない。



ランタイムの境界

周囲の環境、すなわちID、シークレット、ネットワーク、監視、復旧こそが、柔軟なランタイムを運用可能なシステムへと変える場所である。IDはエージェントが何者であるかを決める。シークレットはどの資格情報を使えるかを決める。ネットワークはどこに到達できるかを定める。監視は運用担当者が障害を認識できるかを左右する。バックアップと復旧は環境を元に戻せるかを決める。

OpenClawはその境界を組織が自由に形づくれるようにしている。Microsoft向けのデプロイパターンは、その選択の一定範囲をあらかじめ決めておくものである。

07 ID、シークレット そしてネットワーク

ID

エージェントを組織の主体として扱う

本番のエージェントを、導入した従業員個人の資格情報に紐づけてはならない。対象プラットフォームが対応していれば、組織が管理する専用のワークロードIDを使用する。外部システムによってはアプリケーション資格情報が必要な場合もあるが、それはライフサイクルの責任者、最小権限、失効手順を明確にした例外として管理すべきである。

シークレット

まず不要なシークレットをなくす

最良のシークレットは、そもそも必要としないシークレットである。IDベースのアクセスにより、保管しなければならない生の資格情報を減らせる。それでも必要なシークレットは、マネージドなシークレットストアに保管し、アクセスをランタイムのIDに限定し、ローテーションとインシデント対応の手順を文書化する。本番環境では、ハードコードされた資格情報や開発者端末のシークレットファイルを避ける。

ネットワーク

到達性はポリシーである

本番設計では、開発時の都合のよい既定値をそのまま引き継ぐのではなく、管理用の受信経路と外向き通信の到達範囲を意図的に定義する。不要な公開受信経路は設けず、プライベート接続、踏み台サービス、VPNなど統制された管理経路を用いる。外向き通信は、ワークロードが実際に必要とするモデルエンドポイント、チャネル、パッケージ配布元、業務システムに限定する。

> **IDはエージェントが何者かを定める。シークレットはどの扉を開けられるかを定める。ネットワークはどの扉に到達できるかを定める。**

08 業務システム統合は価値とリスクが交わる場所

エージェントがCRM、ERP、コラボレーション基盤、データベース、API、MCPサーバーにまたがって動作できるようになると、OpenClawの価値は急激に高まる。同時に、誤った判断がもたらす影響も大きくなる。ツールが使えることは、ポリシーではない。

機能	例	推奨する統制
読み取り	顧客、受注、文書のコンテキストを取得する	最小権限。データの範囲は利用者または処理上の必要性に従わせる。
分析	要約、分類、比較、例外検知	元データのコンテキストを保持し、生成された解釈を出所の裏付けなしに正式なデータとして扱わない。
提案	回答案の作成や次の行動の提案	実行前に人による確認が必要となる条件を定義する。
書き込み	CRMの更新やタスクの作成	書き込み可能なオブジェクトとフィールドを限定し、追跡可能なIDを用いる。
実行	API、ワークフロー、処理の起動	事前条件、上限、冪等性、承認ゲートを定義する。
外部への操作	メール送信、外部への投稿、顧客に見える状態の変更	より強い承認、監査証跡、迅速な失効手段を必須とする。

> **目指すべきは、あらゆる場面での最大限の自律性ではない。業務に見合った適切な自律性である。**

09 エージェント固有の脅威 インジェクションと混乱した代理人

権限管理だけでエージェントが安全になるわけではない。エージェントは正当にツールの利用を認可されていても、その権限を誤って行使するよう仕向けられ得る。NISTは、LLM統合アプリケーションが後から取得するコンテンツに埋め込まれた攻撃を含め、直接および間接のプロンプトインジェクションを生成AIシステムのリスクとして論じている。[7] OWASPは、LLMアプリケーションの主要リスクとしてプロンプトインジェクションと過剰な権限を挙げている。[8]

脅威	現れ方	統制パターン
直接プロンプトインジェクション	利用者がエージェントに対し、ポリシーを無視させたりツールを不正に使わせたりするよう指示する	システムレベルでのポリシー適用、許可リスト化されたツール、範囲を限定したID、重大な操作に対する承認ゲート。
間接プロンプトインジェクション	Webページ、文書、メール、取得データに悪意ある指示が仕込まれている	取得したコンテンツは信頼できないデータとして扱う。コンテンツと制御指示を分離し、ツール呼び出しと外向き通信を制限する。
混乱した代理人の挙動	エージェントは正当な権限を持つが、信頼できない情報源によって、その権限を誤った目的のために行使させられる	操作を認証済みの要求者やプロセスに紐づけ、意図とコンテキストを検証し、影響が大きい場合は承認を必須とする。
過剰な権限	エージェントが、タスクに必要な範囲を超えるツール、スコープ、自律性を持つ	最小権限、ツールスキーマの限定、書き込みと実行の境界の明示、可能な場合は短期間有効な資格情報。
不適切な出力処理	生成された出力が、コード、SQL、シェル、HTML、あるいは他のシステムへそのまま渡される	出力を検証・サニタイズし、可能な限り自由形式のコマンド実行ではなく型付きのツールインターフェースを用いる。

> **安全なエージェントアーキテクチャは、エージェントに何を許すかと、その判断に何が影響を与えてよいかの双方を統制しなければならない。**

10 モデル、信頼性 そして運用

モデル戦略

個人利用であれば好みのモデルを選べばよい。ビジネスにはポリシーが必要である。承認済みプロバイダー、地理的所在、プライバシー条件、レイテンシ、クォータ、フォールバック挙動、コスト管理がそれにあたる。Microsoft Foundryは統制されたモデルアクセス層を提供でき、その場合もOpenClawはランタイムであり続ける。[3][10][11] タスクによって異なるモデルが妥当となることもあり、業務プロセスを特定世代のモデルに固定すべきではない。

信頼性とオブザーバビリティ

業務プロセスがエージェントに依存する場合、運用担当者は、ゲートウェイが正常か、チャネルが接続されているか、モデルアクセスが失敗していないか、資格情報が失効していないか、そしてエージェントがどの重大な操作を試みたかを把握する必要がある。OpenClawはゲートウェイの診断機能とチャネルのプローブを提供するが、周囲の運用モデルには依然として責任者、アラート、対応手順が必要である。[6]

サイジングとコスト：見せかけの精度ではなく要因を押さえる

コスト要因	実務上の指針
VM / コンピュート	CPUとメモリは、チャネル、同時実行数、ローカルツール、そしてモデルをリモートで動かすかローカルで動かすかによって決まる。まずは控えめな汎用VMから始め、実際のワークロードで負荷試験を行ってから拡張する。
モデル利用	モデルがリモートの場合、通常は変動費の大半を占める。エージェント別・タスク別にトークンとリクエストを計測し、クォータを設定し、単純な処理は必要に応じて安価なモデルへ振り分ける。
ストレージ / ログ	会話状態、ログ、生成物は蓄積していく。VMを無限の容量とみなさず、保持期間を定めてストレージの増加を監視する。
運用	見えにくいコストは人的な運用負担である。パッチ適用、インシデント対応、更新の検証、アクセス権の棚卸しがそれにあたる。VM費用をわずかに削るよりも、自動化の方がこのコストを大きく減らす。

ここで具体的な価格を示していないのは意図的である。クラウドのリージョン、コンピュート種別、モデル単価は変化するためだ。デプロイパターンは、これらの変数を隠すのではなく、観測可能で差し替え可能なものにすべきである。

11 本番運用の準備度マトリクス

これは実務的な社内ゲートである。すべてのワークロードに同一の統制が必要なわけではないが、判断は明示されているべきである。

領域	実験段階	ビジネス展開	本番運用に耐える状態
コンピュート	ノートPC / 都度対応	永続的なホスト	既知のサイジング、責任者、再起動と復旧
ID	個人の資格情報	専用IDの構成	最小権限、ライフサイクル、監査可能性
シークレット	環境変数 / 設定ファイル	保護されたストレージ	保管庫で管理しローテーション可能。アクセス制御あり
ネットワーク	都合のよい既定値	受信 / 外向き通信の制限	文書化された管理経路とセグメント化
モデル	好みのモデル	承認済みプロバイダー / モデル	リージョン、クォータ、フォールバック、コスト方針
統合	開発者のトークン	範囲を限定した業務アクセス	読み取り / 書き込み / 実行の境界を明示
インジェクション対策	ほぼ未実施	プロンプトとツールのガードレール	信頼できないコンテンツの扱い、承認、外向き通信の統制
監視	目視確認	ヘルスチェック / ログ	アラート、責任者、対応手順
更新	都度対応	計画的	検証、ロールバック、リリース規律
復旧	ほぼ未実施	VM / 構成のバックアップ	検証済みの再構築・復旧手順
文書化	個人のメモ	デプロイ記録	再現可能な環境定義

> **エージェントが動くことは技術的な到達点にすぎない。運用できるエージェントこそがビジネス上の能力である。**

12 ライフサイクルと再現性は一つの論点である

本番環境は導入した時点で完成するのではない。プロビジョニングし、セキュリティを設定し、構成し、接続し、検証し、運用し、更新し、やがて再構築しなければならない。

これらのライフサイクル上の各段階は、デプロイ時の選択が一人のエンジニアの記憶ではなく記録として残されて初めて、確実なものになる。

再構築テスト

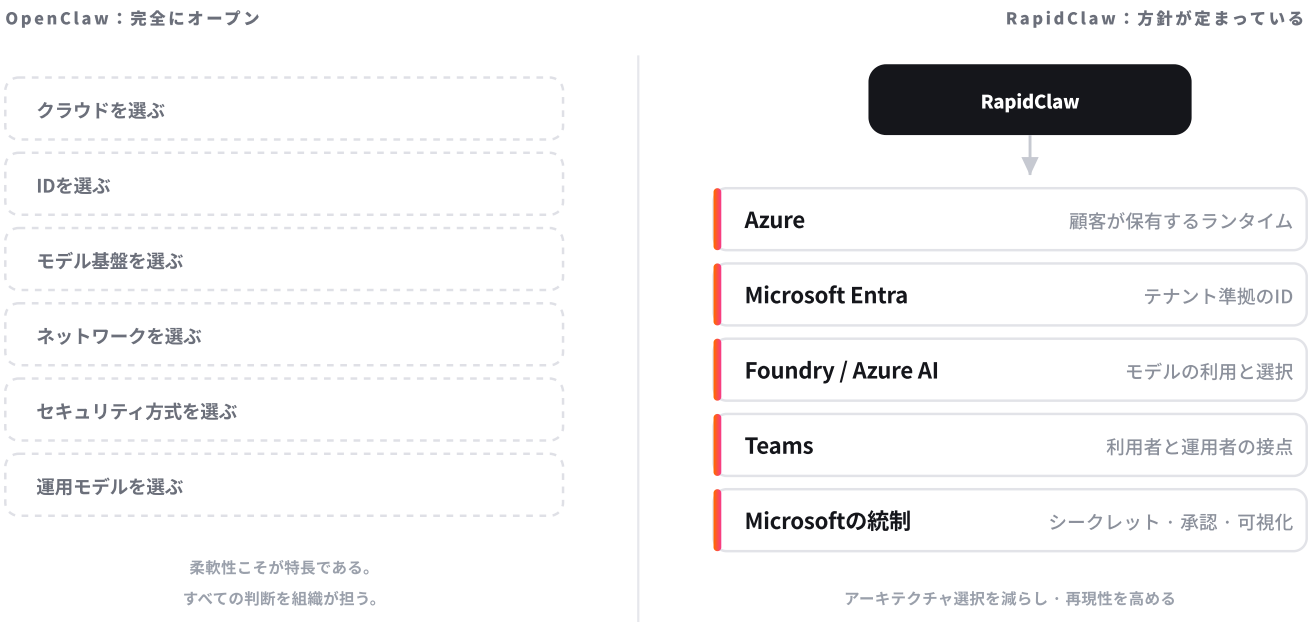
今夜そのVMが消失したとして、別の管理者が翌日に環境を再現できるだろうか。別のチームやリージョン向けに2つ目のインスタンスを展開できるだろうか。どの設定が意図的なもので、どれが過去の名残なのかを説明できるだろうか。Infrastructure as Code、スクリプト化された構成、デプロイ記録、統制された更新経路が、動いているマシンを再現可能な能力へと変える。

> **動いているVMは一つのインスタンスにすぎない。再現可能な環境こそが能力である。**

13 OpenClawから RapidClawへ

ここまでの内容は、クラウドやベンダーを問わず当てはまる。OpenClawは周辺アーキテクチャを意図的に開いたままにしている。RapidClawは、その開放性がMicrosoftで標準化された組織にとって展開負担となる地点から始まる。

RapidClawはOpenClawのフォークではない。ランタイムは引き続きOpenClawである。RapidClawは、Microsoftを基盤とする組織向けに、その周囲のデプロイと運用の方針を定めたパターンである。実際には、構築者に属人化した手作りのVMではなく、ガイド付きのデプロイによって顧客自身のAzureテナント内に環境全体を約20分で立ち上げることを意味する。[9]



判断領域	RAPIDCLAWの方針
クラウド境界	顧客が保有するAzure上のランタイム[9][14]
ID基盤	Microsoft Entraに準拠した組織ID[9][13]
モデル基盤	既定でMicrosoft Foundry / Azure AIの経路[3][10][12]
利用者接点	適切な場面ではMicrosoft Teams
統制	Microsoftに準拠したシークレット、アクセス、可視化、ガバナンスのパターン[9][11][15]
デプロイ	一度限りの手作業ではなく、ガイド付きで再現可能な構成[9][14]

14 RapidClawが変えるもの そして開いたままにするもの

RapidClawは、ある側面では意図的に狭く、別の側面では開いている。毎回作り直す必要のないMicrosoft環境でのデプロイ上の選択を標準化する一方、ワークロードに適したエージェント、スキル、ツール、業務システム接続については、OpenClaw自体を開いたままにしている。

境界	内容
RapidClawが標準化するもの	Azureへのデプロイ経路、MicrosoftのID統合、Foundry / Azure AIのモデル経路、Teamsとの接点、再現可能なセットアップと運用パターン。
顧客が引き続き決めるもの	どのエージェントを構築するか、どの業務システムとMCPサーバーを公開するか、読み取り・書き込み・実行の権限、承認ポリシー、保持期間、規制対象ワークロードの統制、業務上の責任分担。
RapidClawが行わないこと	OpenClawのフォークや置き換え、あらゆるクラウドとランタイム構成の同等サポート、ビジネスガバナンスの必要性の排除。

> **方針を定めたデプロイの価値は、あらゆる判断をなくすことにあるのではない。Microsoftを基盤とする組織が毎回ゼロから行う必要のない判断を取り除くことにある。**

結論

OpenClawの完全にオープンなアーキテクチャは、その魅力の大きな部分を占める。難しくなるのは、その柔軟性を信頼できるビジネス基盤へと変えなければならない段階である。永続的なコンピュート、ID、シークレット、ネットワーク、モデルポリシー、統合の境界、インジェクション対策、オブザーバビリティ、更新、そして復旧である。

すでにMicrosoftを基盤として運用している組織に対する、この展開ギャップへの当社の答えがRapidClawである。OpenClawをランタイムとして維持し、Microsoftアーキテクチャに関する選択を一度だけ行い、それを再現可能な運用パターンに変える。

参考文献

[1] OpenClaw公式ドキュメント、docs.openclaw.ai、2026年8月閲覧。
docs.openclaw.ai

[2] Microsoft Learn、「What is Microsoft Copilot Studio?」2026年8月3日更新。
learn.microsoft.com/en-us/microsoft-copilot-studio/fundamentals-what-is-copilot-studio

[3] Microsoft Learn、「What is Microsoft Foundry Agent Service?」2026年8月19日更新。
learn.microsoft.com/en-us/azure/foundry/agents/overview

[4] LangGraph公式ドキュメント、LangChain、2026年8月閲覧。
docs.langchain.com/oss/python/langgraph/overview

[5] CrewAI公式ドキュメント、2026年8月閲覧。
docs.crewai.com/en/introduction

[6] OpenClaw公式ドキュメント、ゲートウェイのランブックと診断、2026年8月閲覧。
docs.openclaw.ai/gateway

[7] NIST AI 600-1、「Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile」2024年7月、§2.9 Information Security。
nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf

[8] 「OWASP Top 10 for LLM Applications 2026」プロンプトインジェクションと過剰な権限、OWASP GenAI Security Project、2026年8月。
genai.owasp.org/resource/owasp-genai-llm-top-10-2026

[9] RapidStart、「RapidClaw for OpenClaw — Governed AI in Azure」2026年8月閲覧。
www.rapidstart.com/en/products/rapidclaw-for-openclaw

[10] Microsoft Learn、「Built-in policies for model deployment in Microsoft Foundry portal」2026年8月18日更新。
learn.microsoft.com/en-us/azure/foundry/how-to/model-deployment-policy

[11] Microsoft Learn、「Role-based access control for Microsoft Foundry」2026年7月3日更新。
learn.microsoft.com/en-us/azure/foundry/concepts/rbac-foundry

[12] Microsoft Learn、「Deployment types for Microsoft Foundry Models」2026年8月6日更新。
learn.microsoft.com/en-us/azure/foundry/foundry-models/concepts/deployment-types

[13] Microsoft Learn、「Managed identities for Azure resources」Microsoft Entra公式ドキュメント。
learn.microsoft.com/en-us/entra/identity/managed-identities-azure-resources/overview

[14] Microsoft Learn、「What is an Azure landing zone?」Cloud Adoption Framework、2026年8月26日更新。
learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone

[15] Microsoft Learn、「Understanding autorotation in Azure Key Vault」2026年4月10日更新。
learn.microsoft.com/en-us/azure/key-vault/general/autorotation

OpenClawはランタイムである。RapidClawはその周囲を固めるMicrosoftのデプロイパターンである。

RapidStartについて

RapidStartは、先進的なビジネステクノロジーの導入・運用・活用をより容易にすることを目指し、Microsoftを基盤とする業務アプリケーションとエージェント基盤を構築している。