HashiCorp

DX を加速する クラウド運用モデルと プラットフォームチーム

HashiCorp スタックを活用した セキュアなワークフローの標準化

ホワイトペーパー

＊本ホワイトペーパーはオリジナル版を日本語訳したものです。

内容

エグゼクティブサマリ

クラウドは、顧客に新たな価値を提供する組織にとって、今や標準の選択肢となっています。成功する企業は、クラウドサービスを定着させるためのフレームワークであるクラウド運用モデルを活用することで、俊敏性、信頼性、セキュリティを最大限に高め、優れたビジネス成果を実現しています。

しかし、それは最初の一歩に過ぎません。最も成熟した組織は、人材、プロセス、ツールを細かく調整して、全社的なクラウド導入の拡大を中心となって支援するプラットフォームチームを構築しています。

プラットフォームチームは、クラウド運用モデルから最大限の利益を引き出すうえで重要な役割を担います。プラットフォームチームのエンジニアが、組織全体の開発者が利用する共有インフラやランタイムなどのサービスを提供するからです。プラットフォームチームが効果的に活動できれば、クラウド導入のための標準化されたワークフロー、コンプライアンスに対応したゴールデンイメージ、SoR（Systems of Record）を提供するクラウド運用モデルを構築できます。そしてこれが、生産性の向上、リリース頻度の増加、安定性の改善、リスクの低減、コストの最適化につながります。

はじめに

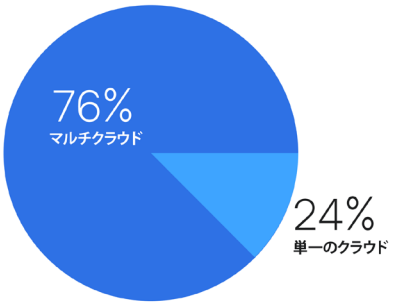
組織がクラウドを効果的に導入し、マルチクラウド環境で成功するためには、クラウド運用モデルが不可欠です。このホワイトペーパーでは、クラウド運用モデルの構成要素と、アプリケーションデリバリを工業化するうえでプラットフォームチームが中心となって果たす主な役割について説明します。また、人々の働き方、従うプロセス、使用するツールを標準化するための実証済みの事例についても採り上げます。

プラットフォームチームには、クラウドインフラやその他の共有サービスのプロビジョニング、運用、管理を行うエンジニアが在籍します。そのエンジニアが、組織のどこからでも必要に応じて利用できる高度に自動化されたプラットフォームを構築して運用します。開発者はセルフサービスのプロセスを介してそういったプラットフォームの機能を活用することで、新しい環境や新しいサービスインスタンスをすばやく簡単に構築できるようになります。プラットフォームチームの役割は、プラットフォームの安定性、回復力、効率性、安全性を維持することです。また、プラットフォームチームがもたらす確かなサービス基盤を強化することで、組織のアプリケーション開発チームが新しい機能をすばやく作成し、ユーザーに短期間でリリースできるようになります。

長期的には、パフォーマンスに優れたプラットフォームチームが開発者やセキュリティエンジニア、そしてビジネスリーダーからのフィードバックに基づいてプロセスを改善し、信頼性を向上させ、魅力的な新機能を追加できるようになります。

マルチクラウドの導入で生まれたプラットフォームチーム

2021 年 HashiCorp のクラウド戦略の現状調査によれば、調査対象者の 76% が複数のクラウドを利用しています。また、組織の規模が大きくなるほど、複数のクラウドを利用する傾向が高くなっています。



当然ながら、マルチクラウドを採用しているのは主に大企業ですが、その差は時間の経過とともに縮まっています。

組織規模別のマルチクラウドの導入状況

60%	76%	90%
小規模企業	**中規模企業**	**大企業**
（従業員数 100 人未満）	（従業員数 101 ～ 5,000 人）	（従業員数 5,000 人超）

出典: 2021 年クラウド戦略の現状調査

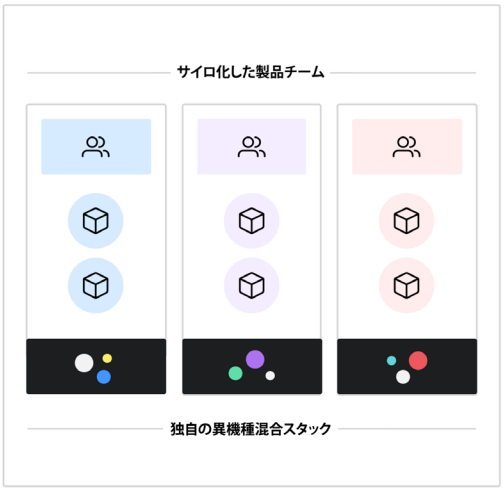
有機的な導入、多様化の検討、合併と買収など、さまざまな理由から複数のクラウドを導入する組織が増えています。マルチクラウドへの移行を背景として、大規模な組織では 2 つの新しい動きが広がっています。

1. **プラットフォームチームとプラットフォームマインドセットへの転換**。開発者と運用チームとの間の摩擦を軽減するために、一連のインフラサービスを標準化します。これにより、中心となる小規模なプラットフォームエンジニアチームに適切なツールを提供し、開発者エクスペリエンスを API、ドキュメント、サポートによって向上させることができます。
2. **「Platform-as-a-Product（製品としてのプラットフォーム）」の実践**。従来の IT プロジェクトは開始日と終了日が決まっています。しかし、クラウドプラットフォームは違います。クラウドプラットフォームは製品であり、終わりがくることはありません。継続的なタスクとして、バックログの管理、定期的な機能のリリース、関係者への最新のロードマップの提供などがあります。

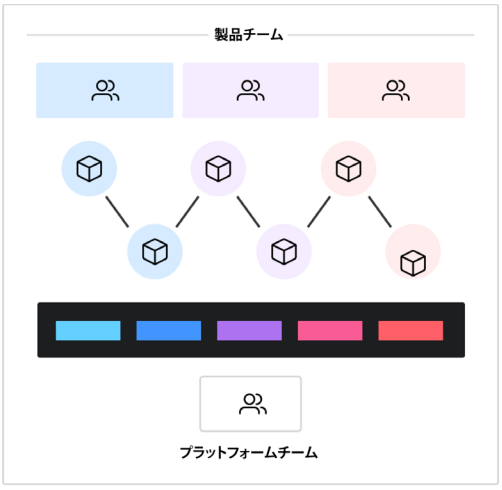
プラットフォームチームとプラットフォームマインドセットへの転換

プラットフォームチームは厳選された標準 API を使用して、マルチクラウドアーキテクチャについて開発チームが抱える複雑さという問題を取り除きます。こうすることで、定評のあるサービスの利用が可能になり、各チームはカスタムコードの作成に集中できるようになります。

プラットフォームチームは組織の力を大幅に強化する重要な役割を果たすことができます。目標は、セキュリティ要件やコンプライアンス要件など、組織のクラウドに関する一連のベストプラクティスを共有プラットフォームにしっかりと組み込むことです。このモデルが運用効率を改善し、開発者の生産性を向上させます。



組織では、しばしばクラウドの使用がサイロ化します。このような場合、各製品チームが独自のプロセス、ツール、アーキテクチャを使用しています。独自のテクノロジスタックを維持するために発生する継続的なコストと複雑さは甚大です。



プラットフォームチームを運用する組織は、規模が拡大しても効率を向上させることができます。プラットフォームチームがプロビジョニング、セキュリティ、ネットワークなどの標準サービスを提供することで、製品チームがアプリケーション開発に集中できるようになるからです。

Platform-as-a-Product（製品としてのプラットフォーム）の実践

組織は自社のクラウドプラットフォームを製品として実行する必要があります。これはユーザー中心設計の基本原則です。クラウドプラットフォーム「製品」を構築する際の目標は、そのプラットフォーム上で動作するサービスを構築するチームのニーズを理解することです。導入と成功を促進するために、プラットフォームには実証できる価値が必要です。

製品としてのプラットフォームの構築は、従来の IT プロジェクトの実行とは異なります。プロジェクトごとの開始日や終了日ではなく、プラットフォームに必要なのは反復的な製品開発です。プラットフォームチームはビジネスや技術の関係者からのフィードバックを絶えず取り入れ、そのフィードバックに基づいて新機能や拡張機能を定期的にリリースする必要があります。

最大限の効果を得るために、プラットフォームチームは 9 つの重要な分野に対する実用的なアプローチに焦点を当てる必要があります。

1. **信頼性の定義と測定:** 開発者がプラットフォームチームを信頼することは非常に重要であり、信頼は時間をかけて構築されるものです。信頼性に優れた予測可能な一連のサービスがあれば、開発者、ひいては経営陣がビジネスの正念場においてもプラットフォームの信頼性と拡張性に自信を持つことができます。期待される成果には、アプリケーションの稼働時間やパフォーマンスに関する目標を達成し、それを上回ることなどが含まれます。測定手段としては、従来のサービス品質保証（SLA）が役に立ちます。また、長期的には、サービスレベル目標やエラーバジェットなどの高度なサイト信頼性エンジニアリング（SRE）を実践することで、より大きな効果を得られる可能性があります。

. また、長期的には、サービスレベル目標やエラーバジェットなどの高度なサイト信頼性エンジニアリング（SRE）を実践することで、より大きな効果を得られる可能性があります。

2. **継続的な労力削減と自動化の改善:** 可能な限り自動化に努める必要があります。自動化は一貫性を高め、労力を削減し、開発者の生産性を向上させます。自動化の第一歩は、特定のワークフローを完了するために必要な手順を理解することです。エンジニアリングチームと話し合うことで、特定のプロセスのブループリントを作成できます。そこから、プラットフォームチームはタスクの自動化を進めることで、デプロイの頻度を高めたり、市場投入までの期間を短縮したりできます。各ワークフローを自動化することで、あとから新たな成果を実現するために必要となる労力が少なくなります。

3. **教育の実施とプラットフォーム導入の促進:** 成功する企業は、ネットワーク、セキュリティ、インフラ、バッキングサービスなどを処理する機能的なプラットフォームを構築して管理する傾向にあります。しかし、プラットフォームとその構成要素に関するセルフサービスポータルやドキュメントの提供は、この取り組みにおいて重要でありながら見落とされている部分です。配慮の行き届いた「開発者エクスペリエンス」を提供するプラットフォームエンジニアは、高い意識を持って、新しいアプリケーションやサービスの提供を加速する標準パターンの使用を促進します。

4. **セキュリティとコンプライアンスを組み込む:** セキュリティ、コンプライアンス、暗号化、監査などの情報セキュリティに関する問題は、自動化およびリファレンスアーキテクチャを綿密に実装し、プラットフォームの基盤で対処すべきです。これにより、たとえば、プラットフォームチームは重大な脆弱性に対応するために、最新のセキュリティパッチを適用した標準化されたゴールデンイメージを展開できます。また、開発者ではなくエンジニアが、シークレット管理およびクレデンシャルをローテーションするためのワークフローを自動的に実行するように構成します。こうすることで、各チームはスタック全体ではなく、特定のサービスに焦点を絞ってセキュリティとコンプライアンスの確保に取り組むことができます。その結果、開発者の生産性が向上します。

5. **サポートを通じた社内コミュニティの構築:** 特定のプラットフォームやテクノロジの使用を徹底するだけでは不十分です。クラウド運用モデルを効果的に推進するためには、プラットフォームチームが適切なワークロードとユースケースに対応するプラットフォームの使用をサポートし、促す必要があります。さらには、新しい機能を文書化し、最新のロードマップを共有することも重要です。プラットフォームチームは社内のミーティングやハッカソンに参加して開発者と交流することで、彼らの目標や課題を知ることができます。こうした取り組みによってチーム間で共感や信頼が生まれ、プラットフォームの導入が促進されます。

6. **優れた開発者エクスペリエンスの提供:** 開発者を説得し、開発チームの競合プラットフォームではなく、組織のプラットフォームを導入してもらう必要があるかもしれません。開発者が強く望むもの、つまり面倒な作業を最小限に抑えた迅速な導入オプションを提供することで、開発者を惹き付けることができます。実用的な初期設定、一般的なユースケースの規範的ガイダンス、製品化までの道筋、チュートリアル、コードサンプルなどによってプラットフォーム導入のハードルを下げ、優れた開発者エクスペリエンスを実現します。複雑な組織では、管理面での面倒を取り除くことが、開発者に正当なプラットフォームサービスを利用してもらう強力な要因となります。開発者自身がそのメリットを理解すれば、組織内の他のチームの説得にも力を貸してくれるでしょう。

7. **実用的な標準化を図る:** 組織が利用できるクラウドサービスは数多くあります。プラットフォームチームはこれらの製品を評価し、実用的な「サービスカタログ」を開発者がいつでも確認できるようにしておく必要があります。プラットフォームチームはビジネスの関係者とともに、サポート対象のサービスをわかりやすくリストにまとめることで、イノベーションと持続可能性のバランスをとることができます。これを適切に行えば、開発者は必要なツールとサービスを手に入れることができ、プラットフォームチームは不要なサービスの管理に費やす運用上のオーバーヘッドを回避できます。

8. **「チャージバック」と「ショーバック」でコストを最適化:** プラットフォームチームは、クラウドをより効率的に利用するモデルの構築を支援する必要があります。共有リソース、自動化、標準化を幅広く活用することで、たとえばサーバーの無秩序な増加を抑えることができます。コストが最適化されるのは、実際にいくら使っているのかを一人一人のユーザーが理解したときです。そこで、プラットフォームチームはリソースのタグ付けを行い、使用したサービスを特定のチームや事業部門に関連付ける必要があります。この方法は、コストの透明性を高めたり（「ショーバック」）、社内の予算配分を引き下げたり（「チャージバック」）するのに使用できます。プラットフォームチームが明確で透明性の高い利用状況レポートを提供すれば、予算配分やコストの最適化に関する議論がはるかに容易になります。

9. **目標とビジネス成果の一致:** 最後に、プラットフォームチームはサービスを提供する側であるため、組織のプラットフォームアプローチをビジネス成果と一致させる必要があります。その成否は、リリース頻度、プラットフォームの安定性と信頼性、セキュリティワークフローの効率性、コストの最適化など、成果重視の指標で評価されるべきです。

クラウド運用モデルによるプラットフォームチームのワークフローの標準化

クラウド運用モデルは、インフラ、セキュリティ、ネットワーク、アプリケーションなど、スタックのすべてのレイヤのチームに影響を与えます。また、クラウドに対する組織の成熟度が増すほど、IT のスピードがさらに上がります。そのため、企業はプラットフォームチームを作り、各レイヤでのアプリケーションの迅速な提供、強固なセキュリティ体制、運用の効率化の実現に必要な動的サービスの提供を目指しているのです。

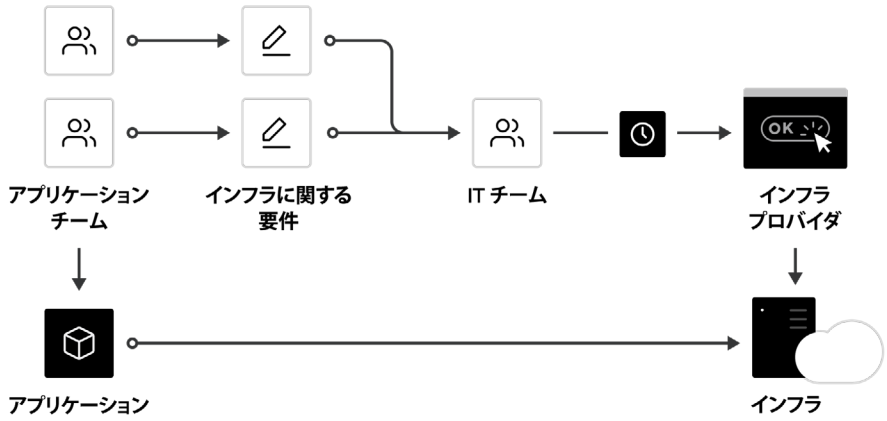
HashiCorp がこれまで目にしてきた組織の成功事例に基づいて、ここではクラウド運用モデルを導入し、インフラ、セキュリティ、ネットワーク、アプリケーションの各レイヤのプラットフォームに適用するためのベストプラクティスをいくつかご紹介します。

HashiCorp Terraform によるインフラのプロビジョニングの標準化

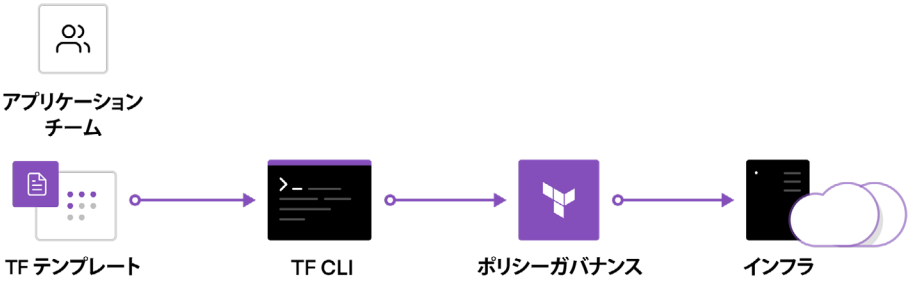
クラウドプラットフォーム運用の基礎となるのはインフラのプロビジョニングです。HashiCorp Terraform は、世界で非常に人気のあるクラウドプロビジョニング製品です。Terraform を導入すれば、さまざまなプロバイダが提供するクラウドや一般的なソフトウェアアプリケーションを使って、あらゆるサービスのインフラをプロビジョニングできます。

プラットフォームチームは、インフラプロビジョニングのための共有サービスを構築するために、再現可能な Infrastructure as Code の実装から始める必要があります。その後、コンプライアンスとガバナンスのワークフローを階層化して、適切な制御を確保します。

Terraform 導入前



Terraform 導入後



自動化されたプロビジョニングワークフローがなければ、リソースに関するリクエストのレビューと承認を手動で行わなければなりません。HashiCorp Terraform なら、プラットフォームチームがすべてのリクエストを事前に承認されたテンプレートを使用して自動的に処理することで、アプリケーションチームが必要に応じてインフラにアクセスできるようになります。

再現可能な Infrastructure as Code

インフラプロビジョニングのための共有サービスに関する最初の目標は、再現可能な Infrastructure as Code を提供できるようにすることです。これによって製品チームは、CI/CD ワークフローの範囲内で使い慣れたツールを使用して、リソースのプランニングとプロビジョニングを行うことができます。理想はこのプロビジョニングを適切に機能させ、プロビジョニングに関する作業を開発チームから取り除くことです。

プラットフォームチームはテンプレートとして動作する Terraform モジュールを作成して、1 つ以上のクラウドプラットフォームのサービス構成を指定できます。Terraform を使用すれば広く普及している大半の構成管理ツールと連携させることができ、基盤となるリソースのプロビジョニングを実施した後にきめ細かなプロビジョニングを行うことができます。最終的には、ほかの多数のソフトウェアベンダによるサービスを使ってテンプレートを拡張し、エージェント監視システム、アプリケーションパフォーマンス管理（APM）システム、セキュリティツール、DNS、ダッシュボードなどを導入できます。また、一度定義すれば、テンプレートを使って必要に応じて自動的にプロビジョニングすることも可能です。このようにして、Terraform はリソースをプロビジョニングするチーム間の共通言語、そして共通のワークフローとなり、プラットフォームやプラットフォーム機能の拡張に役立ちます。

クラウドエコシステムは HashiCorp Terraform を標準としています。2,000 社以上の Terraform プロバイダが、一貫したプロビジョニングワークフローを導入できるようプラットフォームチームを支援しています。

セルフサービスインフラの提供に取り組むプラットフォームチームにとっては、テンプレート作成プロセスとプロビジョニングプロセスを切り離すことで、アプリケーションの本稼働までに要する時間を短縮できます。事前承認されたテンプレートを使っている限り、開発者が稼働の承認を待つ必要がないためです。

コンプライアンスと管理

ほとんどのプラットフォームチームは、作成したインフラの種類、その使用法と使用するチームに対してポリシーを適用する必要があります。Policy as Code フレームワークである HashiCorp の Sentinel を使用すれば、チームの全体的なワークフローを変更せずにコンプライアンスとガバナンスを確保できます。Sentinel 自体もコードで定義されているので、プラットフォームモデルを運用するにあたり共同作業や理解を促すことが可能です。

Policy as Code を採用しない場合、変更の承認にチケットベースのレビュープロセスを使用することになります。このため、開発者はインフラがプロビジョニングされるまで少なくとも数週間待たされることになり、これがボトルネックとなる可能性があります。Policy as Code を活用すると、ポリシーの定義とポリシーの適用が切り離されるため、プラットフォームチームはこの問題を解決できます。

プラットフォームチームはポリシーをコード化して、セキュリティ、コンプライアンス、運用上のベストプラクティスをサービスプロビジョニング全体に対して適用する必要があります。「シフトレフト」アプローチによって自動化すれば、変更内容をコンプライアンスに基づいて確実に適用し、手作業でのレビューによるボトルネックが発生しません。

HashiCorp Packer によるイメージの自動構築

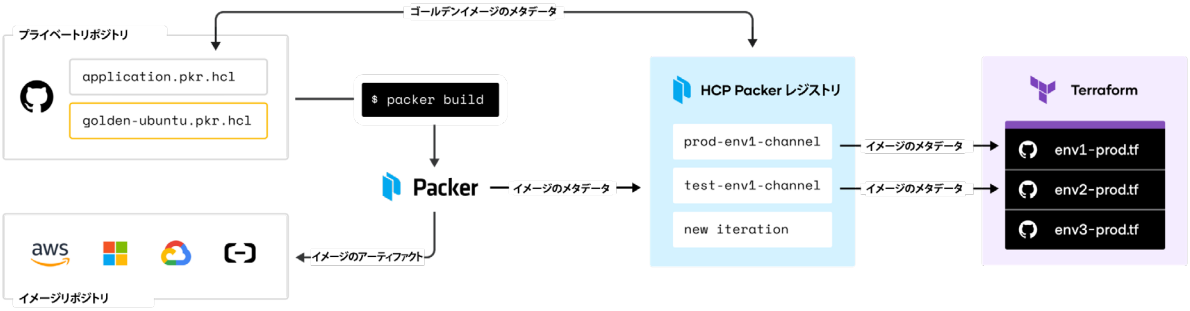
HashiCorp Packer は、単一のソーステンプレートで複数のクラウドに対して同一のマシンイメージを作成できるオープンソースツールです。一般的なユースケースは「ゴールデンイメージ」の作成であり、プラットフォームチームはこれを使用してクラウドインフラを標準化できます。

Packer を活用すると、Docker イメージやクラウドサービスプロバイダで使用するイメージなど、あらゆるタイプのマシンイメージの作成を自動化できます。Packer で作成したイメージの多くは、Terraform でプロビジョニングのワークフローを開始する入力データとして使われます。

自動化の規模拡大を目指すのであれば、HCP Packer レジストリが役立ちます。HCP Packer は、Packer のイメージファクトリと Terraform のイメージ展開の橋渡しをするクラウドサービスです。このツールチェーンにより、イメージの作成、管理、利用を、セキュリティチームとプラットフォームチームが連携して一元的に実行できるようになります。

自動化の規模拡大を目指すのであれば、HCP Packer レジストリが役立ちます。HCP Packer は、Packer のイメージファクトリと Terraform のイメージ展開の橋渡しをするクラウドサービスです。このツールチェーンにより、イメージの作成、管理、利用を、セキュリティチームとプラットフォームチームが連携して一元的に実行できるようになります。

HCP Packer レジストリには、組織のゴールデンイメージに関するメタデータが保存されます。具体的には、メタデータの作成日時、クラウド上の保存場所、イメージビルドに関連付けられた git commit（ある場合）などの情報です。レジストリを使用すると、Packer ビルドが生成するゴールデンイメージに関する情報を追跡し、テスト環境および本番環境に適切なイメージを明確に選定できます。その後、選定したゴールデンイメージをインフラのプロビジョニングワークフローに使用します。



HashiCorp Vault によるシークレット管理とデータ保護

動的なクラウドインフラでは、ホストベースのアイデンティティではなくアプリケーションベースのアイデンティティを使用します。また、複数のクラウドで、明確な境界がなく信頼性が低いネットワーク、またはゼロトラストネットワークを使用します。

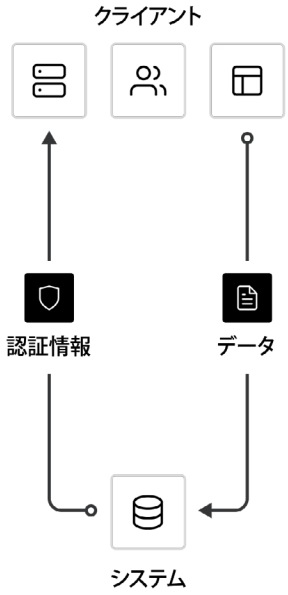
従来のセキュリティの世界では、信頼性の高い社内ネットワークが想定されていました。その結果、外部に対しては強固なセキュリティ体制がとられる一方、内部のセキュリティ体制は緩いものでした。最新の「ゼロトラスト」アプローチでは、内部に対しても強力なセキュリティを適用します。そのためには、ユーザーとアプリケーションの両方に対して認証を確実に行い、具体的にはシークレットの読み取りと機密データの処理を実行するための権限を明確に付与して、厳しく監査を実施する必要があります。

HashiCorp Vault を使用すると、プラットフォームチームはトークン、パスワード、証明書、暗号化キーを安全に保管してこれらへのアクセスを厳密に制御できます。これにより、マシンとアプリケーションの包括的なシークレット管理ソリューションが実現します。このほかにも、Vault は保管中のデータと転送中のデータの保護に役立ちます。Vault では、暗号化を行うための高レベルの API を公開しており、開発者は暗号化キーを提供することなく機密データを保護できます。また、Vault は認証局のように機能して、一時的な証明書を動的に発行し、SSL/TLS を利用して通信を保護します。さらに、Vault では異なるプラットフォーム間のアイデンティティの違いを吸収します。たとえば、Azure Active Directory（AAD）と AWS Identity and Access Management（AWS IAM）でプラットフォームの境界を越えてアプリケーションを利用できます。

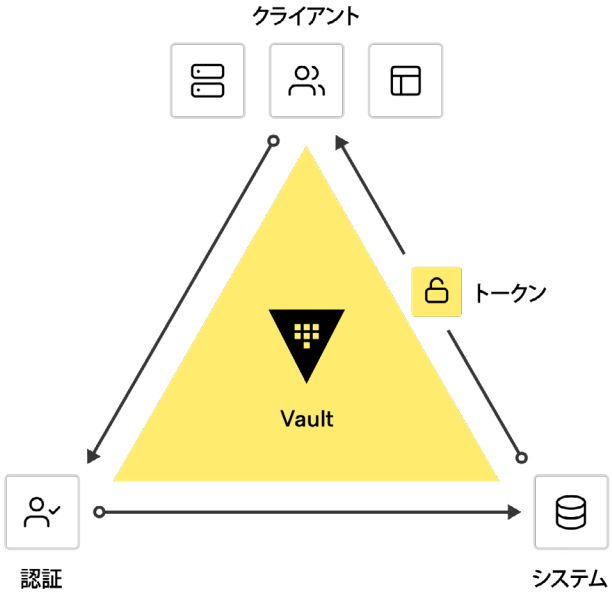
Vault は、証券取引所、大手金融機関、ホテルチェーンなど、さまざまな業界のプラットフォームチームに広く利用されており、クラウド運用モデルのセキュリティを提供しています。

プラットフォームチームは、セキュリティのための共有サービスを提供するために、シークレット管理サービスを一元化する必要があります。そして、証明書とキーのローテーション、転送中および保管中のデータの暗号化などにこの基盤を使用し、より高度な Encryption as a Service ユースケースを提供しなければなりません。これにより、セキュリティの考慮事項がプラットフォームに組み込まれるため、製品チームは提供された API に接続するだけで、企業のセキュリティ基準を満たしたサービスを構築できるようになります。

Vault 導入前



Vault 導入後

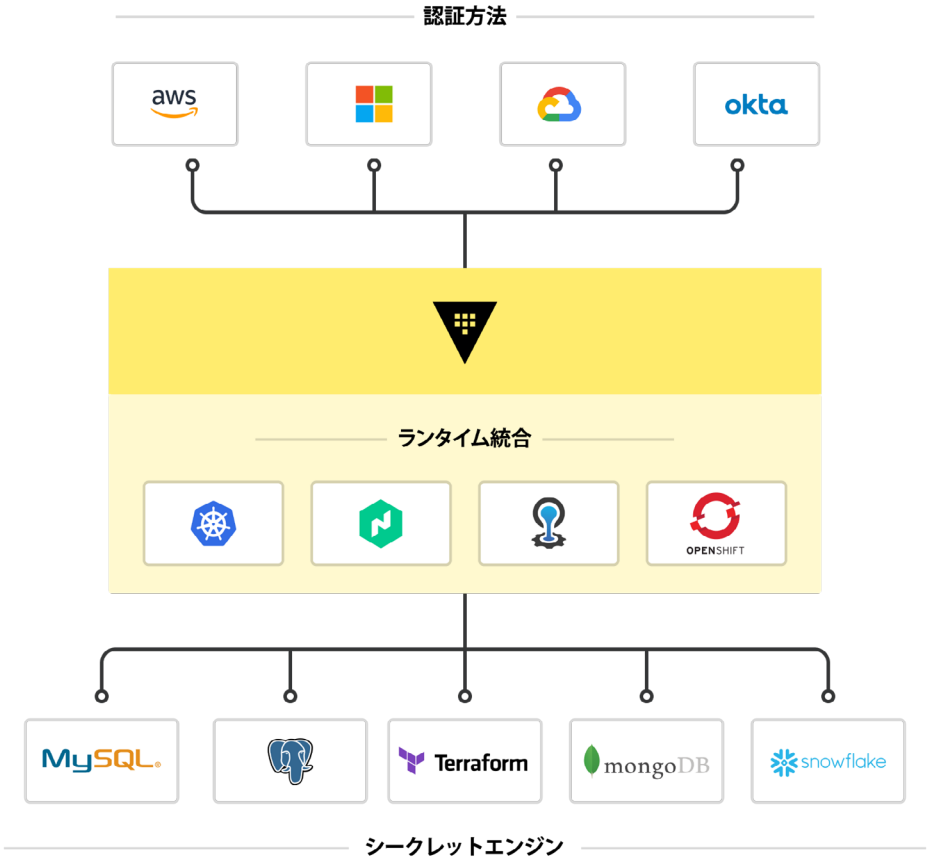


信頼性の高い従来のデプロイには、プログラムを使用してパスワードやその他の機密情報を保護する方法がありません。しかし、HashiCorp Vault を導入すれば、ユーザーとマシンの両方のワークフローを自動化し、クレデンシャルへのアクセスを一元管理できます。

シークレット管理

クラウドセキュリティの第一歩はシークレット管理の実現です。これには、中心となるストレージを使用し、アクセス制御、シークレットの動的な配布を行います。ただし、サービスやリソースの認証とアクセスには、静的な IP アドレスに依存するのではなく、AAD や AWS IAM などのアイデンティティベースのアクセスシステムと連携させることがきわめて重要です。

Vault では、ポリシーを使ってアプリケーションでの認証方法、認証で使用するクレデンシャル、監査の実施方法をコード化します。また、Vault は、クラウドのアイデンティティおよびアクセス管理プラットフォーム、Kubernetes、Active Directory、その他の SAML ベースの認証システムといった信頼性の高いさまざまなアイデンティティプロバイダと連携させることができます。Vault を使用すれば、信頼性の高いアプリケーションのソースとユーザーアイデンティティに基づいて、シークレットやシステムを一元的に管理し、これらにアクセスすることも可能です。



HashiCorp Vault は、アイデンティティに関する一般的なソースと連携し、
信頼できるアイデンティティブローカとして大規模に運用できます。

プラットフォームチームは、一貫性があり、監査とセキュリティで保護されたワークフローを使用してあらゆるシステムのシークレットを要求できる共有サービスを構築する必要があります。このような共有サービスは、セキュリティを高めるだけでなく、開発者の生産性向上にもつながります。また、開発者は、侵害されたシークレットを参照している処理を手動で検索する必要がなくなります。Vault でシークレットの更新と失効のワークフローを自動化して、シークレットを修復できるからです。

Encryption as a Service

前述に加えて、保管中と転送中のアプリケーションデータも暗号化する必要があります。Vault は Encryption as a Service としてキー管理と暗号化を実行できる一貫性のある API を提供します。このためプラットフォームチームは、Vault と連携させるだけで複数の環境でデータを保護できるようになります。

Vault を Encryption as a Service の基盤として使用することで、証明書とキーのローテーションなど、プラットフォームチームとセキュリティエンジニアが直面する複雑な問題を解決できます。また、キーを一元管理したり、複数のクラウドとデータセンターに対して転送中のデータと保管中のデータを簡単に暗号化したりできます。これによって、高価なハードウェアセキュリティモジュール（HSM）に要するコストを削減し、プラットフォームの利用者全体で一貫したセキュリティワークフローと暗号化標準を確立し、生産性を向上させることができます。

多くの企業では開発者にデータを暗号化するよう指示していますが、通常はそのための「方法」が提供されず、開発者は暗号についての理解が不十分なままカスタムソリューションを構築しなければなりません。プラットフォームチームは Vault を使用して開発者にシンプルな API を提供し、連携するセキュリティチームは必要に応じてポリシー制御とライフサイクル管理のための API を使用できます。

高度なデータ保護

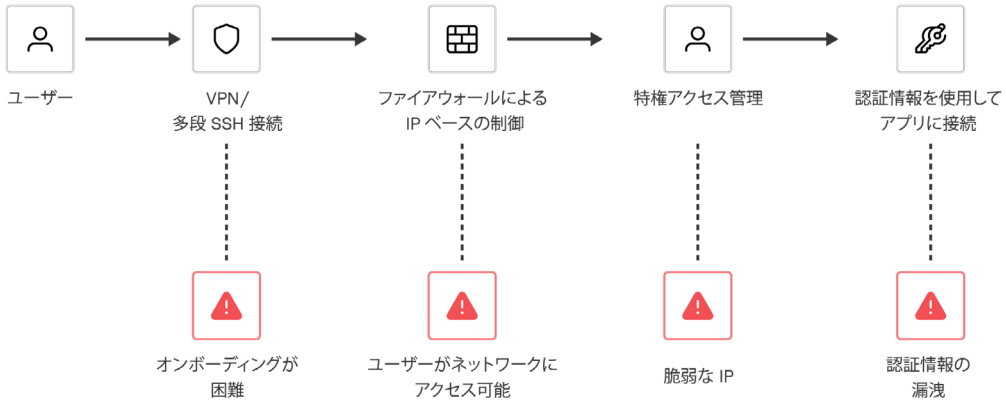
多くの場合、プラットフォームチームはストレージに保管しているデータを暗号化するなど、暗号化が必要なオンプレミスのサービスおよびアプリケーションの保守とサポートを引き続き行う必要があります。しかし、このような方法でこれらのアプリケーションを改善することはできないかもしれません。なぜなら、暗号化キーを管理するために必要なカスタムエンジニアリングには、かなりの手間がかかるためです。解決策は、キー管理のタスクを外部プロバイダに委託することです。Vault の高度なデータ保護機能を使用すれば、インフラと Vault 間を安全に接続し、高度な暗号化キーとその運用および管理を制御しながら連携させることができます。たとえば透過的なデータ暗号化（TDE）を使用して、MySQL、MongoDB、PostgreSQL などのデータベース内のデータを保護します。

高度なデータ保護機能を使用すれば、プラットフォームチームはデータマスキングなどのデータのトークン化を行い、クレジットカード番号や銀行口座情報などの機密データを保護できます。そのため、この機能はデータコンプライアンス（PCI DSS、HIPAAなど）のセキュリティ要件が厳しい組織で広く利用されています。

HashiCorp Boundary によるアクセスの保護と管理

現代におけるアイデンティティのセキュリティ原則は、Vault が対応するマシン間のアクセスワークフローの枠を超えて拡大しています。アイデンティティは、ユーザーとマシン間のやり取りを保護するうえでも同様に重要な役割を果たします。

ユーザーのアクセスを保護する従来のソリューションでは、SSH キーや VPN クレデンシャルの配布、Bastion ホストの配置およびそれぞれの管理が必要でした。このアプローチには、クレデンシャルが無秩序に増加するリスクがあるうえ、運用するためにはかなりの手作業が必要です。よくある例が、手間を減らすために SSH キーを無期限に設定し、それによってユーザーがネットワークやシステム全体に無期限にアクセスできるようになるというケースです。



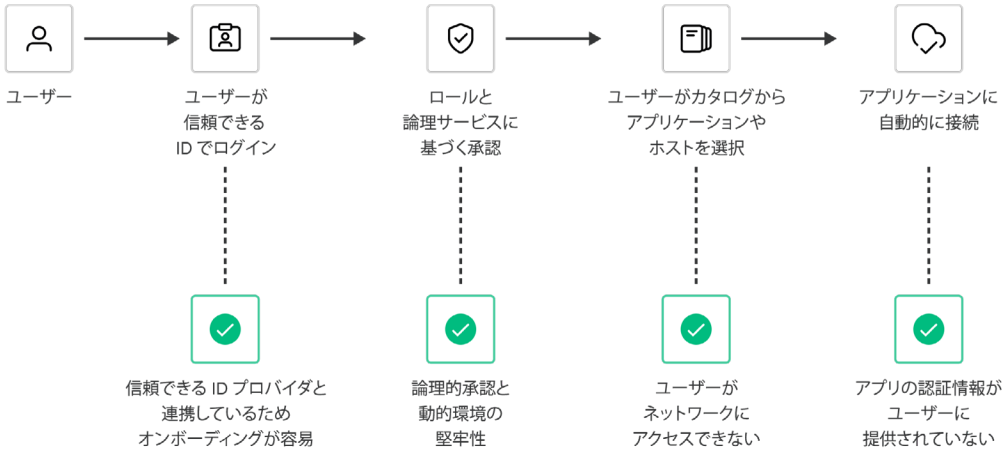
従来のアクセスワークフローは手作業で行われるため、システムに複数の脆弱性をもたらします。

次の一般的なケースについて考えてみます。ある開発者が本番環境のアプリにアクセスし、問題をトラブルシューティングしようとしています。従来のアクセスモデルでは、このアプリは既知の仮想 IP アドレスでプライベートネットワーク（または VPC）にデプロイされています。また、アクセスは IP レベルで構成されています。

管理者が該当のアプリケーションへのアクセスを構成するためには、多くのアクセスレイヤを経由する必要があり、以下のレイヤごとに独自の構成が必要になります。

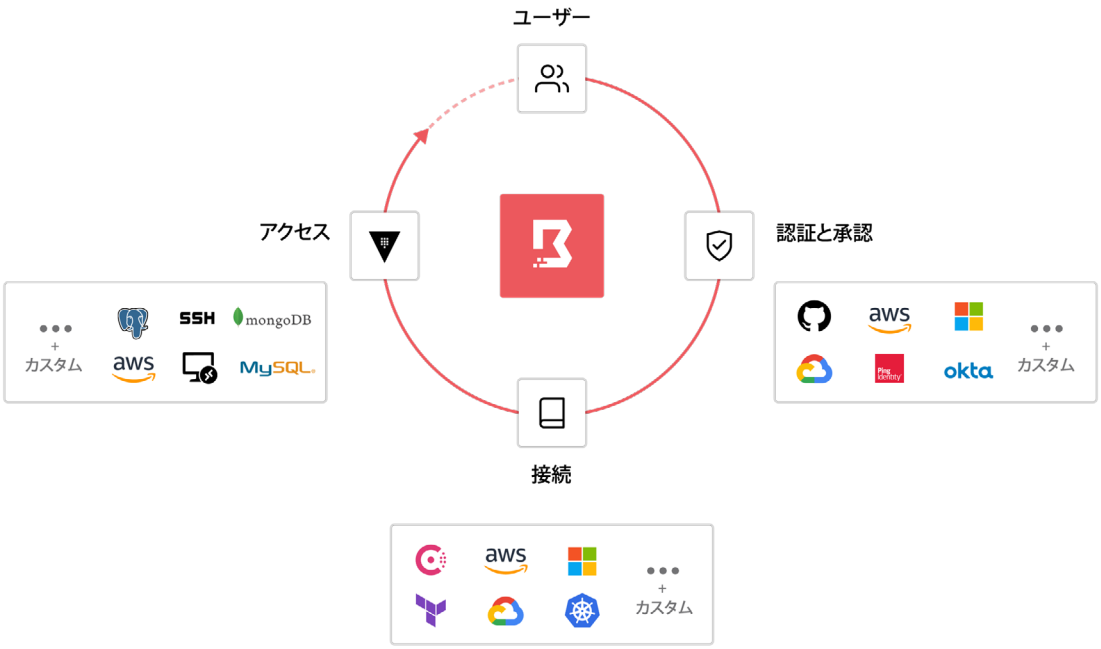
- 詳細なアクセス制御ができない企業の VPN/プライベートネットワークレイヤ
- 企業ネットワークとデータセンターネットワークにまたがる Bastion ホスト/踏み台サーバーレイヤ
- 論理アイデンティティベースではなく、クライアント/ターゲット IP レベルでアクセス許可を実施するネットワークファイアウォールレイヤ（またはその他のロケーションベース制御）

クラウドインフラの世界には、一時的なクラウドリソースや動的 IP アドレスといった性質があるため、この原理は破綻します。ユーザーとマシン間のアクセスを提供する最新のアプローチでは、ユーザーアイデンティティの検証が必要です。さらに、エンドシステムへのアクセスおよび権限のプロビジョニングを自動化する必要があります。



自動化された最新のアクセスワークフローでは、その基盤としてアイデンティティを使用します。

HashiCorp Boundary は、クラウド運用モデルでのアクセスに関する最新の課題を解決します。Boundary は、信頼できるアイデンティティに基づくきめ細かな認証で、アプリケーションや重要なシステムへのアクセスを簡単に保護できる、安全なリモートアクセスソリューションです。Boundary なら、クラウド、ローカルデータセンター、信頼性が低いネットワークにまたがるアクセスを、基盤となるネットワークを公開することなく管理できます。



HashiCorp Boundary を利用すれば、ユーザーのアイデンティティに基づいて、
あらゆるシステムにどこからでも簡単かつ安全にリモートアクセスできます。

ユーザーは任意のアイデンティティプロバイダ（AAD、Okta、AWS IAM など）を使用して Boundary で認証できます。その後、各ユーザーには動的な一連のターゲットに対してアクションを実行するための厳格な権限が付与され、Vault などのクレデンシャル管理ソリューションから提供されるクレデンシャルを使用してこれらのターゲットに接続するための Just-In-Time アクセスが許可されます。

このモデルは、以下を利用して動的なインフラに安全にアクセスできるため、プラットフォームチームの共有サービスアプローチに適しています。

- **アイデンティティベースのアクセス制御:** プラットフォームエンジニアは、ユーザーやアプリケーションの特権セッション（TCP、SSH、RDP など）への Just-In-Time アクセスを効率化できます。また、チームは拡張可能なロールベースのアクセス制御を使用して、アクセス許可を制御できます。
- **アクセスの自動化:** プラットフォームチームは、Boundary の完全にインストルメント化された Terraform プロバイダ、REST API、CLI、SDK を使用して、リソース、アイデンティティ、アクセス制御の境界をコードとして定義できます。また、リソースをプロビジョニングする際の、新しいリソースの検出や既存ポリシーの適用を自動化することもできます。
- **セッションの可視化:** セキュリティエンジニアは、Boundary で確立された各特権セッションの監視と管理を行えます。セッションログはさまざまな分析ツールにエクスポートすることが可能です。

プラットフォームの機能がどのように進化しても、Boundary はあらゆるシステムやアプリケーションへの安全なリモートアクセスをサポートできます。

HashiCorp Consul によるアプリケーションの安全な接続

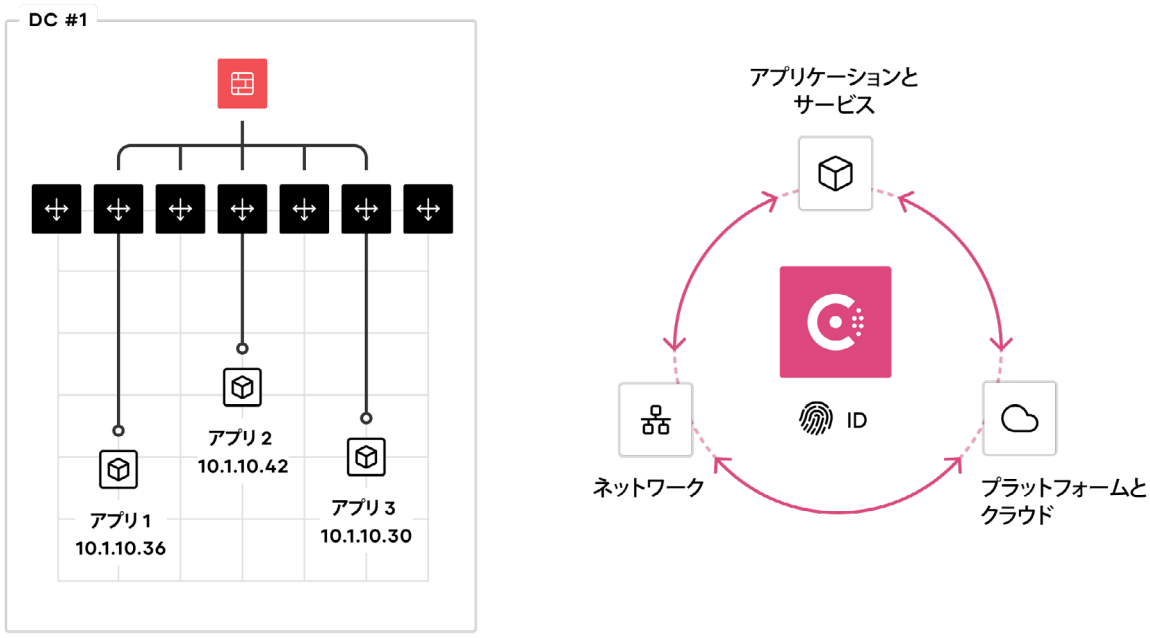
クラウドでのネットワーク構築は、クラウド運用モデルを導入しようとするプラットフォームチームにとって非常に困難な課題の 1 つです。エンジニアは、動的な IP アドレスをうまく辿り、マイクロサービス実装における East-West 型のトラフィックの急増を考慮し、不明確なネットワーク境界に対応する必要があります。

HashiCorp Consul を利用すれば、プラットフォームチームは検出したサービスに安全に接続し、マルチクラウドネットワークを管理できます。Consul は、サービスネットワークをグローバルに運用するために広く導入されています。

ネットワークサービスは、サービス ID に基づいて一元的に提供されるべきです。こうすることで、プラットフォームチームがサービスディスカバリのために一元的なサービスレジストリを作成できるようにする必要があります。共通のレジストリを利用することで、実行中のサービスの「マップ」を取得し、そのサービスが存在する場所、および実行中のサービスの現在の動作状態を取得できます。また、プログラムからレジストリに対してクエリを実行してサービスを検出したり、API ゲートウェイ、ロードバランサ、ファイアウォールなどの重要なネットワークミドルウェアコンポーネントの設定の自動化を促進したりできます。サービスメッシュアプローチをとると、特にマルチクラウド環境やマルチデータセンター環境で、ネットワークトポロジをシンプルにできます。

Consul 導入前

Consul 導入後



従来のネットワーク構築では、ホストに静的 IP アドレスを手作業で設定していました。
現代のクラウドアーキテクチャチームとプラットフォームチームは HashiCorp Consul を活用することで、
自動化されたサービス中心のアプローチを採用できます。

サービスディスカバリ

クラウド運用モデルのネットワークの開始点となるのが共通のサービスレジストリです。このレジストリは、実行中のサービス、そのサービスが存在するネットワーク上の場所、および実行中のサービスの現在の動作状態について示す、最新状態のディレクトリとして機能します。ネットワーク構築の従来のアプローチは、長寿命のアプリケーションでは静的 IP の使用を前提とし、ロードバランサとファイアウォールを組み合わせて通信を制御するというものでした。サービスのネットワーク上の位置を追跡するには、多くの場合、一貫性のない手動のプロセスや、スプレッドシート、ロードバランサのダッシュボード、構成ファイルなどのツールが必要です。

Consul では、個々のサービスはプログラムを使用して登録され、他のサービスから検出できるように DNS および API インターフェイスが提供されています。また、ヘルスチェックと連携させて各サービスインスタンスの動作状態を監視します。このため、プラットフォームチームは利用可能な各インスタンスに優先順位を付け、Consul を活用して不健全なサービスインスタンスにトラフィックをルーティングしないようにすることができます。

Consul と既存の North-South 型のトラフィックを管理する従来のロードバランサなどのサービスや Kubernetes などの分散型コンテナオーケストレーションを連携させれば、クラウドプラットフォーム全体で一貫したレジストリと検出サービスを提供できます。

ネットワークインフラの自動化

次のステップでは、ネットワークを自動化することで、既存のネットワークインフラの運用上の複雑さを軽減します。Consul を使用すると、Terraform と連携させることでこれらのネットワーク運用を自動化し、事前に定義されたタスクに基づいて変更を反映できます。サービスのネットワーク上の位置や構成に変更があるたびに、手作業やチケットベースのプロセスを通じてロードバランサを再構成したりファイアウォールのルールを適用したりする必要はありません。自動化を行うには、静的ベースのアプローチよりもはるかに拡張性の高い非常に動的なインフラを構築し、ネットワークインフラデバイスがサービスレジストリからサービスの変更をサブスクライブできるようにします。

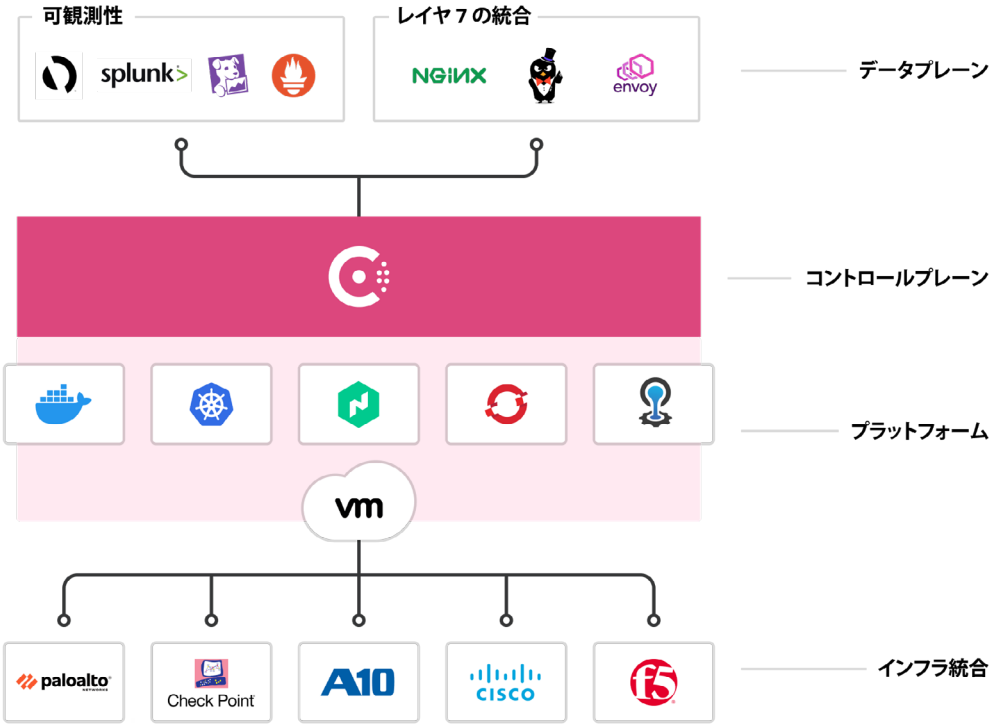
このように構成することで、Consul が検出したイベントの変更に基づいて Terraform が構成を実行し、一般的なタスクの依存関係や障害を取り除くことができます。製品チームはアプリケーションを個別にデプロイし、プラットフォームチームは Consul を利用することで製品チームが必要とするダウンストリームの自動化に対処できます。自動化により、サービスの終了時にファイアウォールのポートを適切に閉じることができるため、サービスのライフサイクル全体を通じてメリットがあります。

サービスメッシュと API ゲートウェイによるゼロトラストネットワーク

企業がマイクロサービスベースやクラウドネイティブのアプリケーションを拡張すると、East-West 型のトラフィックが急増し、より大きく、より動的なインフラが必要になります。 また、これらのマイクロサービスへの外部クライアントからのアクセスを制御することで、North-South 型のトラフィックも徐々に増加します。その結果、単一の障害点となるコストの高いネットワークミドルウェアデバイスが不必要に増え続け、プラットフォームチームに多大な運用上のオーバーヘッドをもたらすことになります。

Consul が提供する分散型サービスメッシュでは、ルーティングや承認などのネットワーク機能を、ミドルウェアを使用せずにネットワークのエンドポイントにプッシュします。これによってネットワークトポロジが簡素化されて管理が容易になり、East-West 型のトラフィックパス内にコストが高いミドルウェアデバイスを設置する必要がなくなります。そして、サービス間通信の信頼性と拡張性が向上します。Consul の API Gateway を導入することで、単一の一元化されたコントロールプレーンで処理する North-South 型のトラフィックを一貫して制御し、セキュリティを確保できます。

Consul はサイドカープロキシ（Envoy などのプロキシ）および個々のサービスインスタンスと連携できる API 駆動型のコントロールプレーンです。サイドカープロキシは分散型のデータプレーンを提供します。この 2 つのプレーンを組み合わせて、すべてのサービス間通信の認証、承認、暗号化が確実に行われるゼロトラストネットワークモデルを実現します。このセキュリティ体制は、自動的な相互 TLS 暗号化とアイデンティティベースの承認によって確保します。プラットフォームチームは、適切な関係者と協力して、（IP アドレスではなく）論理サービスによるセキュリティポリシーを定義し、開発者に最小限の特権アクセスを提供できます。



HashiCorp Consul は、単一のコントロールプレーンで、サービスネットワークのための幅広いエコシステムを実現します。

Consul と Vault を統合することで、コントロールプレーンとデータプレーンの両方で証明書の自動ローテーションを実行する、PKI と証明書の一元管理を行えます。さらに、Consul を Kubernetes 環境に統合すれば、キーやトークンなどの機密データを Vault 内に保管することもできます。 このアプローチをとると、ネイティブの Kubernetes シークレットを使用するよりもリスクが低くなります。

Consul サービスメッシュは、あらゆるクラウド環境のあらゆるランタイムで、サービス接続のセキュリティを確保します。この一貫性のあるデータプレーンにより、開発者やプラットフォームチームは異機種混合環境や抽象化の枠を越えてサービスを接続できます。さらに、Consul は管理パーティションを備えたマルチテナントに対応しています。この機能により、複数のデプロイを単一のコントロールプレーンで管理でき、各テナントの自律性と分離性を維持しながら、一貫した管理とガバナンスを実現できます。

HashiCorp Waypoint によるアプリケーションのデプロイの標準化

組織は新しい「SoE（Systems of Engagement）」を構築し、アプリケーション環境の大部分を刷新しようとしています。刷新が予定されているアプリケーション環境の範囲は、アーキテクチャの種類、開発フレームワーク、ビジネス上の重要領域など多岐にわたります。

刷新を進める中で、開発者はコンテナ、スケジューラ、YAML ファイル、サーバーレスなどがもたらす数多くの複雑さに直面する可能性があります。一方で、この複雑さにはメリットもあります。たとえば、マイクロサービスアーキテクチャを導入すれば、革新的な成果を得られる可能性があります。ただし、1 つのサービスを本番環境にデプロイできるようになるまでには、相当の時間を要すると考えられます。

また、大規模な組織に共通する課題として、異なるツールを異なる環境に導入することが挙げられます。開発者は、たとえば Kubernetes には Docker と kubectl、VM には HashiCorp Packer と Terraform、各サーバーレスプラットフォームにはカスタム CLI を使用します。開発者やオペレータなど個人にとって、このように多種多様なツールや環境を使用することは、その習得が課題となります。チームで考えると、一貫性、複雑さ、セキュリティ面での課題が生じます。

ビルド、デプロイ、リリース

クラウド運用モデルを採用すると、HashiCorp Waypoint を利用して本番環境にデプロイする際のゴールデンワークフローを確立することで、さまざまなツールや環境を統合できます。Waypoint は、ランタイム全体でビルド、デプロイ、リリースを行うための最新ワークフローを提供します。

HashiCorp Waypoint を利用することで、開発者はアプリケーションのビルド、デプロイ、リリースを一元的に行えるようになります。

Waypoint には、簡単に使えてあらゆるアプリケーションをデプロイできる 1 つのコマンド「waypoint up」が用意されています。このワークフローは、Kubernetes、HashiCorp Nomad、Amazon EC2、Google Cloud Run など、あらゆるプラットフォームで一貫しています。Waypoint はプラグインによる拡張が可能で、ビルド/デプロイ/リリースの任意のロジックに対応します。プラットフォームチームは、GitHub Actions、GitLab CI/CD、Jenkins、CircleCI などの継続的なデプロイワークフローに Waypoint を導入できます。

Waypoint の logs や exec などのコマンドを活用すると、デプロイ後にその検証やデバッグを行うことができます。

Waypoint がプラットフォームチームにもたらす影響

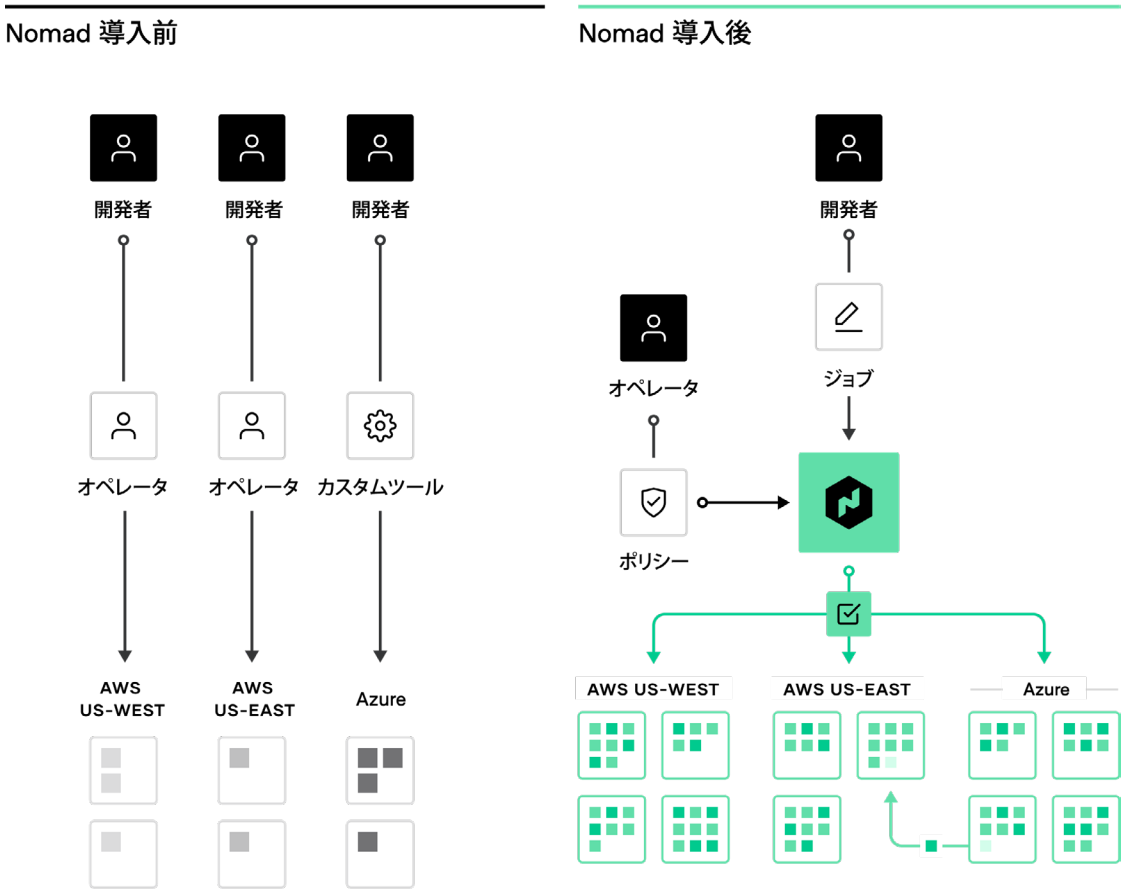
プラットフォームチームにとって最も大きな課題の 1 つは、多くの開発チーム、情報セキュリティ担当者、およびコンプライアンス監査のニーズに合った本番環境への単一経路を提供することです。各開発チームは、フレームワーク、バッキングサービス、ターゲットランタイム、デプロイの頻度、運用上の考慮事項などを独自に組み合わせてサービスを構築しています。また、セキュリティ要件やコンプライアンス要件も満たさなければなりません。

プラットフォームチームは、Waypoint と Terraform、Vault、Boundary、Consul を連携させることで、コンプライアンス、セキュリティ、ネットワークの必須要件を組み込みながら、クラウドインフラ上で一貫したアプリケーションデリバリを実現できます。

HashiCorp Nomad によるワークロードのオーケストレーションの標準化

アプリケーションレイヤでは、新しいアプリケーションの配信が増える一方で、従来のアプリケーションもより柔軟に管理する必要があります。HashiCorp Nomad は従来のアプリケーションと最新のアプリケーションの両方をデプロイおよび管理できる柔軟なオーケストレータです。長期にわたって実行されるサービスから一時的に利用するバッチジョブ、システムエージェントにいたるまで、あらゆる種類のワークロードに対応します。

アプリケーションデリバリのための共有サービスのメリットを得るために、プラットフォームチームは Nomad と Terraform、Vault、Consul を連携させる必要があります。これらのソリューションを連携させることで、クラウドインフラ上で一貫したアプリケーションデリバリを実現しながら、コンプライアンス、セキュリティ、ネットワークの必須要件も満たすことができます。

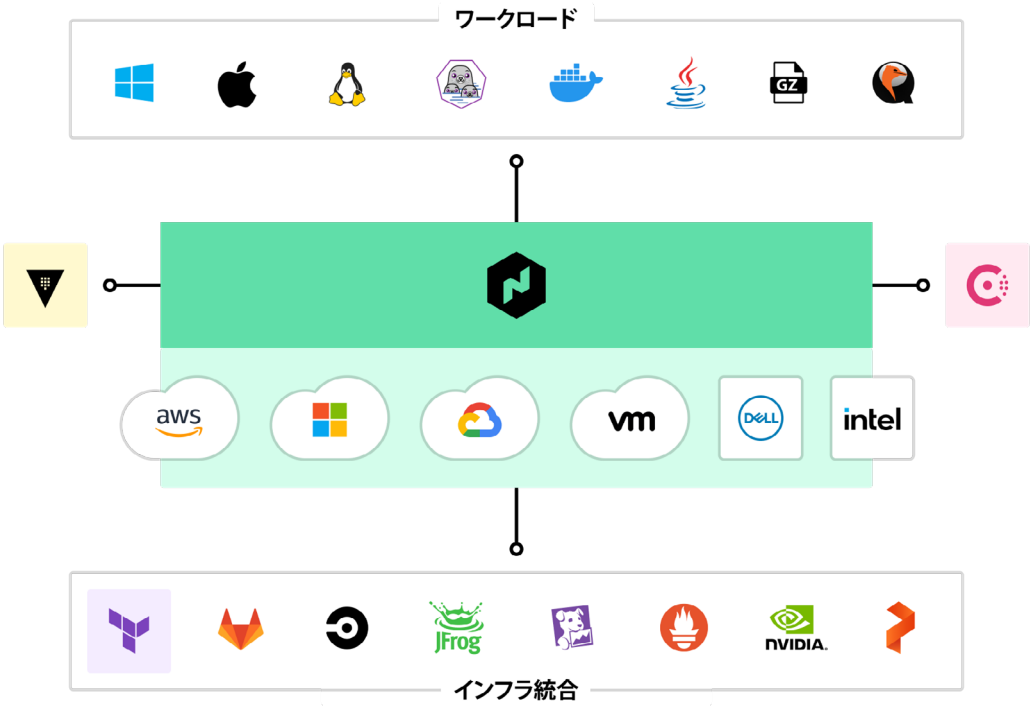


開発チームは独自のツールチェーンやデプロイプロセスを構築することが多く、結果的に運用効率が低下する場合があります。Nomad を使用すれば、Nomad の単一ジョブ仕様を標準化し、単一のワークフローを使用して、さまざまな種類のアプリケーションをさまざまなクラウドで実行できます。

多種多様なワークロードのオーケストレーション

今日、新しいワークロードの多くは、Kubernetes やその他のランタイムにデプロイするために、コンテナでパッケージングされています。ただし、レガシーワークロードの多くはこれらのプラットフォームに移行できません。また、今後のサーバーレスアプリケーションも同様でしょう。Nomad は仮想マシンからスタンドアロンのバイナリ、コンテナにいたるすべてのワークロードをデプロイするための一貫したプロセスを提供します。また、これらすべてのワークロードに対し、リリースの自動化、アップグレード戦略、バイナリパッキング、レジリエンスといった中核となるオーケストレーション機能のメリットを活用できます。

通常、最新のアプリケーションはコンテナ内で構築されます。Nomad は、こうした最新のアプリケーションに対し、あらゆる環境で一貫したワークフローを大規模に提供します。Nomad ではオーケストレーションとスケジューリングのシンプルさと有効性が重視されています。そのため、コンテナのワークロードだけを運用して対処するのに専門的なスキルが要求される Kubernetes のようなスケジューラの複雑さを取り除けます。



HashiCorp Nomad は、すでに使用されているワークフローやテクノロジと統合できます。

Nomad を既存の CI/CD ワークフローと連携させると、従来のワークロードと最新のワークロードの両方に対応できるようにアプリケーションを高速で自動的にデプロイできます。

ハイパフォーマンスコンピューティング

Nomad は大規模なクラスタでも遅延を抑えてアプリケーションをスケジューリングするよう設計されています。このことは、大規模なバッチジョブを処理するプラットフォームチームにとって不可欠です。ほとんどのハイパフォーマンスコンピューティング（HPC）ワークロードについても同じことが言えます。「2 Million Container Challenge」という試みで、Nomad は全世界の 10 の AWS リージョンで、200 万の Docker コンテナを 22 分でスケジューリングできました。その平均速度は 1 秒あたり約 1,500 コンテナでした。いくつかの企業では、Nomad のさらに大規模なデプロイも行われています。

Nomad を使用すれば、ハイパフォーマンスアプリケーションで API を使用して動的に容量を確保し、Apache Spark などのデータ分析アプリケーションでのリソース共有を容易に効率化できます。スケジューリングでの遅延が低く、結果をすばやく利用でき、アイドル状態の無駄なリソースを最小限に抑えられます。

マルチデータセンターでのワークロードのオーケストレーション

Nomad はマルチリージョン、ハイブリッドクラウドに対応するよう設計されており、一貫したワークフローであらゆるワークロードをデプロイできます。Nomad のオーケストレーション機能とスケジューリング機能を活用すれば、グローバルアプリケーションを複数のデータセンターに、あるいはクラウドの境界を越えて展開できます。この製品はインフラ、セキュリティ、ネットワークの各リソースに対応し、アプリケーションの正常なデプロイをサポートします。

まとめ: 人材、プロセス、ツール

最新のマルチクラウド環境の価値を最短で生み出すには、共通のクラウド運用モデルとプラットフォームチームの力を、人材、プロセス、ツールという 3 つの側面から組み合わせる必要があります。最適な結果を得るためには、それぞれの側面で独自の転換が必要です。

- **人材: サイト信頼性エンジニアリング（SRE）の実践への転換**
  - 社内データセンターの管理と単一のクラウドベンダーの利用から得られた専門知識を再利用し、あらゆる環境に一貫して適用する。
  - DevSecOps などのアジャイルな手法を採り入れ、短命化と分散化が進むシステムを継続的に提供する。
  - IT 担当者がコードを使って自動化し、運用をソフトウェアの問題のように扱えるようにする。
- **プロセス: セルフサービスへの転換**
  - プラットフォームチームを配置し、アプリケーションデリバリを高速化する共有サービスを提供する。リスクを最小限に抑えてより迅速にソフトウェアを提供できるようにする。
  - インフラ、セキュリティ、ネットワーク、およびその他の機能領域のCoE（Centers of Excellence）を確立し、セルフサービスで機能を提供できるようにする。
  - 開発チームが利用できる共通のサービスを集めた「サービスカタログ」を標準化し、フィードバックに応じて新しい機能を随時追加していく。
- **ツール: 動的な環境への転換**
  - 短命化と分散化が進むインフラとアプリケーションに対応するツールを使用する。また、特定のテクノロジに縛られず、重要なワークフローに対応できるツールを使用する。
  - ポリシーとガバナンスを適用するためのツールを提供して、デリバリのスピードとコンプライアンスとのバランスを取り、セルフサービス環境でのリスクを管理する。
  - プラットフォーム自体にセキュリティ要件とコンプライアンス要件を組み込むことで、本番環境でのデプロイを加速する。

クラウド運用モデルで
引き出される価値

これらを考え合わせると、デジタルトランスフォーメーションの成果の最大化を目指す企業にとって、共通のクラウド運用モデルの導入という変化を受け入れるべきであることは明らかです。企業の IT 部門は、コストの最適化を重視した ITIL ベースの制御から脱却し、スピードの最適化を重視するセルフサービスを実現する組織へと進化しています。

プラットフォームチームは共有サービスをクラウドの各レイヤに提供し、製品チームが新しいビジネス価値と顧客価値を迅速に届けられるよう支援することで、この進化をサポートできます。クラウドインフラのレイヤごとにソリューションを提供し、プラットフォームチームがクラウド運用モデルへの移行を成功へと導けるよう、ぜひ HashiCorp の各種ツールをご活用ください。

HashiCorp について

HashiCorp はマルチクラウドインフラ自動化ソフトウェアのトップ企業です。HashiCorp のソフトウェアを使用すれば、一貫したワークフローを導入し、インフラのプロビジョニング、セキュリティ、ネットワーク、アプリケーションのデプロイなど、クラウドを自動化するための SoR（Systems of Record）を構築できます。HashiCorp の製品ポートフォリオには、Vagrant™、Packer™、Terraform®、Vault™、Consul、Nomad™、Boundary、Waypoint™ などがあります。HashiCorp の製品は、オープンソース、エンタープライズ、およびマネージドクラウドサービスとして提供されます。HashiCorp は本社をサンフランシスコに構えていますが、世界各地に戦略的に分散する形で、従業員のほとんどがリモートワークで働いています。詳しくは、hashicorp.com をご覧いただくか、HashiCorp の Twitter @HashiCorp をフォローしてください。

スタートアップチェックリスト

クラウド運用モデルによるプラットフォームチームのワークフローの標準化

- [ ] Terraform Cloud にサインアップ
- [ ] HashiCorp Cloud Platform にサインアップ
- [ ] HashiCorp Terraform によるインフラのプロビジョニングの標準化
  - [ ] 再現可能な Infrastructure as Code の確立
  - [ ] コンプライアンスおよび管理のワークフローの自動化
- [ ] HashiCorp Packer によるイメージの自動構築
- [ ] HashiCorp Vault によるシークレット管理とデータ保護
  - [ ] シークレット管理の自動化の確立
  - [ ] Encryption as a Service の実装
  - [ ] 高度なデータ保護の導入
- [ ] HashiCorp Boundary によるアクセスの保護と管理
- [ ] HashiCorp Consul によるアプリケーションの安全な接続
  - [ ] サービスディスカバリパターンの確立
  - [ ] ネットワークインフラの自動化の実装
  - [ ] サービスメッシュと API ゲートウェイによるゼロトラストネットワークの導入
- [ ] HashiCorp Waypoint によるアプリケーションのデプロイの標準化
- [ ] HashiCorp Nomad によるワークロードのオーケストレーションの標準化

Platform-as-a-Product（製品としてのプラットフォーム）の実践

- ☐ 信頼性の定義と測定
- ☐ 継続的な労力削減と自動化の改善
- ☐ 教育の実施とプラットフォーム導入の促進
- ☐ セキュリティとコンプライアンスを組み込む
- ☐ サポートを通じた社内コミュニティの構築
- ☐ 優れた開発者エクスペリエンスの提供
- ☐ 実用的な標準化を図る
- ☐ 「チャージバック」と「ショーバック」によるコスト最適化の奨励
- ☐ 目標とビジネス成果の一致

HashiCorp

Copyright © 2022