組織のコードツールとしてのインフラストラクチャの選択

AWS 規範ガイダンス

AWS 規範ガイダンス: 組織のコードツールとしてのインフラストラクチャの選択

Table of Contents

組織のコードツールとしてのインフラストラクチャの選択

Amazon Web Services (寄稿者)

2026 年 2 月 (ドキュメント履歴)

Infrastructure as Code (IaC) は、一連の設定ファイルを通じてアプリケーションのインフラストラクチャをプロビジョニングおよび管理するプロセスです。IaC は、新しい環境を再現可能で信頼性が高く、一貫性のあるものにするため、インフラストラクチャを一元的に管理し、リソースを標準化し、スケールを迅速に行えるように設計されています。これは、バージョン管理、継続的インテグレーション、継続的デプロイなど、アジャイルおよび DevOps プラクティスの主要なコンポーネントです。

Infrastructure as Code (IaC) ツールの選択は、組織にとって戦略的決定と見なされます。この決定は、会社のインフラストラクチャ、アプリケーション、サービスを構築するすべてのチームに影響します。各ツールには長所と短所があるため、one-size-fits-allはありません。

以前は、インフラストラクチャの管理とプロビジョニングは手動プロセスであり、エラーが発生していました。IaC は、コードを通じてこれらのタスクを合理化し、インフラストラクチャをデプロイするための信頼性の高いソリューションとなっています。IaC ツールは、開発者がプログラミング言語を使用してインフラストラクチャを定義およびデプロイできるようにします。これにより、ビジネスの俊敏性が向上するだけでなく、成長とイノベーションのスピードも加速します。さらに、IaC ではデプロイ前にコードをスキャンできるため、IaC はセキュリティを大幅に向上させ、インフラストラクチャの信頼性と安全性を検証します。最終的に、適切な IaC ツールは技術的な決定だけでなく、ビジネスの全体的な成功に直接影響する戦略的決定でもあります。

このガイドでは、リソースのプロビジョニング AWS に使用できる 5 つの異なる IaC ツール、AWS CloudFormation、 AWS Serverless Application Model (AWS SAM) AWS Cloud Development Kit (AWS CDK)、HashiCorp Terraform、Pulumi について説明します。これらのツールを比較し、チーム、組織、クラウド人材のニーズを満たすツールを選択するプロセスを説明します。重要なのは、選択した IaC ツールを組織の目標とデベロッパーのスキルセットに合わせることです。たとえば、チームが JavaScript に習熟している場合は、開発ワークフローを最適化するため、 AWS CDK TypeScript をプライマリ IaC ツールとして選択できます。

IaC を実装することの利点

組織に適した IaC ツールを選択することで、次の利点が得られます。

- スピード — IaC の目標の 1 つは、手動プロセスを排除することで、より迅速に作業を行うことです。コードベースのアプローチにより、短時間でより多くの作業を簡単に行うことができます。各 IT 環境の手動セットアップは非常にコストがかかり、ハードウェアとソフトウェアのセットアップには専用のエンジニアとアーキテクトが必要です。スピードを上げることで、組織は変化する顧客ニーズや市場状況に迅速に適応できるようになります。また、IaC コードを 1 回記述し、同じコードを数百の環境にデプロイすることも、手動で作成するのに必要な時間の一部で実行できます。
- スケーラビリティ – IaC を使用すると、既存のインフラストラクチャにリソースを簡単に追加できます。アップグレードは迅速にプロビジョニングされるため、ピーク使用期間中に迅速に拡張できます。例えば、オンラインサービスを実行している組織は、ブラックフライデーなどの需要の高い時間帯にユーザーの需要に追いつくようにスケールアップできます。
- セキュリティの強化 – IaC は組織のセキュリティとコンプライアンスの有効化に優れています。組織は、IaC に対して自動テストを実行することで、ベースラインのセキュリティポリシーと設定を設定し、パイプラインを通じて適用できます。IaC がコンプライアンスルールに違反すると、パイプラインは失敗し、開発者にフィードバックを自動的に返します。これにより、安全でないインフラストラクチャの作成を防ぐことができます。
- 一貫性 — 一貫性は IaC のもう 1 つの重要な利点です。複数のエンジニアが設定を手動でデプロイしている場合、不整合は避けられません。時間の経過とともに、同じ環境を追跡して再現することが困難になります。これらの不整合は、多くの場合、開発環境、QA 環境、本番環境の重要な違いにつながります。
- 効率 – IaC は開発ライフサイクル全体の効率と生産性を向上させます。プログラマーはサンドボックス環境を作成し、単独で開発します。オペレーションでは、セキュリティテスト用のインフラストラクチャを迅速にプロビジョニングできます。QA エンジニアは、テスト中に本番環境の正確なコピーを持っています。デプロイの準備ができたら、デベロッパーはインフラストラクチャとコードの両方を 1 ステップで本番環境にプッシュします。また、より高いレベルのコラボレーションとチームワークを実現することもできます。チームはアプリケーションの安全なパターンを作成し、それらのテンプレートやコンストラクトを他のチームと共有できるため、重複する作業を減らすことができます。
- コストの削減 — IaC はソフトウェア開発のコストを削減します。環境を手動でセットアップするためにリソースを費やす必要はありません。料金は、アクティブに使用しているリソースに対してのみ発生するため、不要なオーバーヘッドはありません。

- 信頼性の向上 – インフラストラクチャが大きい場合、リソースの設定ミスやサービスのプロビジョニングが間違った順序で簡単に行えます。IaC では、リソースは常に宣言されたとおりにプロビジョニングおよび設定されます。リソースを手動で作成するとエラーが発生しやすいため、IaC を使用するときにインフラストラクチャが意図したとおりに作成されるという高い信頼性が得られます。
- 設定ドリフト検出 — 環境をプロビジョニングした設定が実際の環境と一致しなくなった場合に、設定ドリフトが発生します。多くの IaC ツールは、ドリフトを検出して修正するのに役立ちます。例えば、誰かがリソースを手動で誤って変更した場合、IaC ツールを使用して変更内容を検出し、意図した状態に復元できます。
- 実験 — IaC は新しいインフラストラクチャのプロビジョニングを非常に迅速かつ簡単にするため、多くの時間とリソースを費やすことなく実験的な変更を行い、テストできます。結果を気に入ったら、新しいインフラストラクチャを本番環境にすばやくスケールアップできます。

の IaC ツールの確認 AWS クラウド

このガイドでは、 リソースのプロビジョニング AWS に使用できる 5 つの異なる IaC ツールについて説明します。これらのツールを比較し、チーム、組織、クラウド人材のニーズを満たすツールを選択するプロセスを説明します。重要なのは、選択した IaC ツールを組織の目標とデベロッパーのスキルセットに合わせることです。

このセクションでは、各ツールの仕組みと、その利点と欠点について詳しく説明します。例えば、一部のツールは でのみ使用でき AWS クラウド、マルチクラウドやハイブリッドクラウドのデプロイには適していません。また、特殊なプログラミング言語に関する知識が必要で、一般的なプログラミング言語をサポートしているものもあります。各ツールの利点と欠点を評価し、組織に最適なものは何かを検討してください。

このセクションでは、以下の IaC ツールについて説明します。

- AWS CloudFormation
- AWS Serverless Application Model (AWS SAM)
- AWS Cloud Development Kit (AWS CDK)
- HashiCorp Terraform
- プルミ

IaC ツール AWS CloudFormation としての の使用

AWS CloudFormation は AWS のサービス 、テンプレートファイルを使用して AWS リソースのプロビジョニングを自動化する です。デプロイするすべての AWS リソースを記述するテンプレートを作成し、それらのリソースを CloudFormation プロビジョニングして設定します。

CloudFormation テンプレートは JSON または YAML を使用して記述されます。 CloudFormation スタックは、テンプレートで定義されているリソースの実装です。 CloudFormation スタックは、、プログラムで CloudFormation SDK AWS マネジメントコンソール、または AWS Command Line Interface () を使用して管理できますAWS CLI。の CloudFormation 仕組みの詳細については、CloudFormation ドキュメントの「 のAWS CloudFormation 概念」および「 の AWS CloudFormation 仕組み」を参照してください。

を使用する利点 CloudFormation：

- CloudFormation 変更セットを使用すると、実行中のスタックに変更をデプロイする前に、その変更をプレビューできます。変更セットは、既存のスタックで実行中のリソースに対して提案された変更をまとめたものです。これにより、デプロイ前に競合や意図しない結果を特定できます。例えば、Amazon Relational Database Service (Amazon RDS) データベースインスタンスの名前を変更すると、 CloudFormation は新しいデータベースを作成し、古いデータベースを削除します。既にバックアップしていない限り、古いデータベース内のデータは失われます。変更セットを生成すると、変更によってデータベースが置き換えられ、スタックを更新する前にそれに応じて計画できるようになります。
- 変更セットのデプロイ中にエラーが発生すると、 は既知の最後の動作状態に自動的に CloudFormation ロールバックします。
- CloudFormation スタックセットを使用して、複数の AWS アカウント と にリソースをデプロイできます AWS リージョン。
- AWS::*、Alexa::*、および Custom::* の名前空間では、 をリソースプロバイダー CloudFormation で使用しても追加料金はかかりません。このような場合、手動でプロビジョニングしたリソースと同様に、プロビジョニングした AWS リソースに対してのみ料金が発生します。
- CloudFormation は 状態を管理します。つまり、 CloudFormation は基盤となる のサービス呼び出しを に AWS 行い、 CloudFormation テンプレートで定義されているリソースをプロビジョニングして設定します。
- CloudFormation には、設定ドリフトを検出して修正するためのツールが用意されています。詳細については、 CloudFormation ドキュメントの「スタックとリソースに対するアンマネージド型設定変更の検出」を参照してください。
- CloudFormation を使用してカスタムリソース を作成できます。カスタムプロビジョニングロジックは、スタックを作成、更新、または削除するたびに CloudFormation 実行されるテンプレートに記述できます。
- CloudFormation は、CloudFormation レジストリ を使用したサードパーティーアプリケーションリソースのモデリング、プロビジョニング、管理をサポートします。
- CloudFormation は、 CloudFormation 管理への既存のリソースのインポートをサポートします。

を使用することの欠点 CloudFormation：

- JSON または YAML 構文に慣れていない場合は、ある程度の慣れが必要です。JSON は人間が読み取れるように設計されておらず、インラインコメントを行うことはできません。YAML を使用すると、コメントを作成でき、読みやすくなります。ただし、その構文はタブとスペースに基づいているため、インデントミスを簡単に行うことができます。
- CloudFormation はマルチクラウドデプロイをサポートしていません。

- 再利用可能なコンストラクトやその他のモジュール化されたコード AWS Cloud Development Kit (AWS CDK)を作成するには、 などの高レベルの実装を使用する必要があります。

AWS Serverless Application Model （ AWS SAM) を IaC ツールとして使用する

AWS Serverless Application Model (AWS SAM) は、 を拡張するツールキットです AWS CloudFormation。サーバーレスアプリケーションをより迅速に作成できるように設計された追加機能が含まれています。テンプレートを AWS SAM デプロイすると、定義されたリソースを作成 CloudFormation するために に変換されます。 AWS SAM は、AWS SAM テンプレート仕様と AWS SAM コマンドラインインターフェイス (AWS SAM CLI) の 2 つの部分で構成されます。テンプレートでは構文を直接使用できます CloudFormationが AWS SAM 、 はサーバーレス開発の高速化に特に重点を置いた独自の構文 AWS SAM を提供します。この短縮構文により、Amazon API Gateway などのサーバーレスリソース、 AWS Lambda、および AWS Step Functions リソースの IaC の定義を最適化できます。 Amazon API Gateway AWS SAM CLI は、 AWS Lambda 関数をローカルでテストし、継続的インテグレーションおよび継続的デリバリー (CI/CD) パイプラインを作成し、コマンドを実行してサーバーレスアプリケーションをデプロイするのに役立つ機能を含むデベロッパーツールです。

を使用する利点 AWS SAM :

- AWS SAM には、 と同じ利点があります CloudFormation。
- と比較すると CloudFormation、 AWS SAM を使用して、 でバックアップされた Amazon API Gateway などのサーバーレスアプリケーションとリソースをより簡単に作成できます AWS Lambda。
- AWS SAM CLI を使用して、 AWS Lambda 関数をローカルでテストできます。デバッグモードで Lambda 関数をローカルで呼び出すと、その関数にデバッガーをアタッチできます。デバッガーを使用すると、コードを 1 行ずつステップスルーする、さまざまな変数の値を確認する、および他のアプリケーションと同じように問題を修正することができます。

を使用することの欠点 AWS SAM :

- AWS SAM には、 と同じ欠点があります CloudFormation。
- AWS SAM は の外部では使用できません AWS。

を IaC ツール AWS CDK として使用する

AWS Cloud Development Kit (AWS CDK) は、使い慣れたプログラミング言語を使用してクラウドアプリケーションリソースを定義できるオープンソースのソフトウェア開発フレームワークです。は、 JavaScript、 TypeScript、Python、Java、C#、Go AWS CDK をサポートしています。は、 を介して安全かつ繰り返し可能な方法でリソースを AWS CDK プロビジョニングします AWS CloudFormation。 AWS CDK コードを合成すると、結果は CloudFormation テンプレートになります。 AWS CDK は、リソースの定義プロセスを簡素化する高レベルの抽象化 AWS を提供します。

はコンストラクト の概念 AWS CDK を使用します。コンストラクトは、Amazon Simple Storage Service (Amazon S3) バケットなど、1 つ以上の CloudFormationリソースとその設定を表すアプリケーション内のコンポーネントです。コンストラクトを構成およびカスタマイズして、より複雑なインフラストラクチャを作成できます。詳細については、 AWS CDK ドキュメントの「ビルドレベル」を参照してください。 は、デベロッパーが記述したコードに基づいて CloudFormation テンプレート AWS CDK を生成します。これにより、 CloudFormation テンプレートを手動で作成する必要がなくなります。多くの組織は、他のソフトウェアライブラリと同様に、コミュニティ内でコンストラクトをカスタマイズ、共有、再利用します。コンストラクトを共有すると、デベロッパーはコードを迅速に作成し、デフォルトでベストプラクティスを組み込むことができます。

AWS CDK の側面は、組織が特定の範囲内のすべてのコンストラクトに標準を適用するのに役立ちます。この側面では、タグを追加するなどしてコンストラクトを変更できます。または、コンストラクトの状態について何かを検証することもできます。

AWS CDK を使用すると、デベロッパーは既存のプログラミングスキルと知識を使用してクラウドインフラストラクチャを定義できます。使い慣れたプログラミング言語を使用することで、デベロッパーは専門知識を適用して AWS リソースを記述できるため、アプリケーション開発からインフラストラクチャプロビジョニングへの移行が容易になります。また、 AWS CDK は AWS インフラストラクチャの作成を迅速化することもできます。これにより、 CloudFormation テンプレートを手動で記述するよりも開発ライフサイクルが短縮されます。

を使用する利点 AWS CDK :

- は、よく知られているプログラミング言語 AWS CDK をサポートしています。
- 汎用言語を使用すると、 for-loops、オブジェクト、強力なタイプ、その他のプログラミング手法などの論理コンストラクトを使用できます。これにより、デベロッパーは簡潔で エラーのない方法でインフラストラクチャを宣言できます。このアプローチにより、統合開発環境 (IDE) と関連ツールを使用して、多数のリソースの宣言に伴う複雑さを管理することもできます。

- AWS CDK コンストラクトは共有可能で、ガバナンスとコンプライアンスの要件を満たすのに役立ちます。
- AWS CDK コンストラクトは開発にかかる時間と労力を削減できます。詳細については、「Construct Library API Reference」を参照してください。
- AWS CDK は に基づいています CloudFormation。 CloudFormation とその概念に精通している場合は、 AWS CDK 概念を理解しやすくなります。
- AWS CDK は、ユニットテストとスナップショットテストの実行に役立ちます。
- 機能が でネイティブにサポートされていない場合は AWS CDK、レベル 1 コンストラクトと raw オーバーライド を使用できます。または、 API を直接呼び出すCloudFormation カスタムリソースを使用することもできます。
- CloudFormation スタックを削除することで、リソースを効率的にクリーンアップできます。

を使用することの欠点 AWS CDK：

- には、各 にブートストラップされた環境 AWS CDK が必要です AWS アカウント。ブートストラップは、リソースをデプロイするすべての環境で実行する必要がある 1 回限りのアクションです。
- AWS CDK は、 でのみ IaC をデプロイするために使用できます AWS クラウド。

の IaC ツールとしての Terraform の使用 AWS クラウド

HashiCorp Terraform は、クラウドインフラストラクチャの管理に役立つコードとしてのインフラストラクチャ (IaC) ツールです。Terraform を使用すると、クラウドリソースとオンプレミスリソースの両方を、バージョン管理、再利用、共有できる設定ファイルで定義できます。その後、一貫したワークフローを使用して、ライフサイクルを通じてすべてのインフラストラクチャをプロビジョニングおよび管理できます。

デベロッパーは、Terraform 言語 と呼ばれる高レベルの設定言語を使用します。Terraform 言語のネイティブで低レベルの構文は、HashiCorp設定言語 (HCL) です。Terraform 言語は、人間が読み書きしやすいように設計されています。Terraform 言語を使用して、クラウドまたはオンプレミスインフラストラクチャの目的の終了状態を記述します。次に、Terraform はその終了状態に到達するための計画を生成し、その計画を実行してインフラストラクチャをプロビジョニングします。

Terraform を使用する利点：

- Terraform はプラットフォームに依存しません。これは、任意のクラウドサービスプロバイダーで使用できます。インフラストラクチャは、 および他の多くのクラウドプロバイダー間で設定、テスト AWS 、デプロイできます。組織が複数のクラウドプロバイダーを使用している場合、Terraform はクラウドインフラストラクチャを管理するための単一の統一された一貫性のあるソリューションになります。マルチクラウドサポートの詳細については、Terraform ウェブサイトの「マルチクラウドプロビジョニング」を参照してください。
- Terraform はエージェントレスです。マネージドインフラストラクチャにソフトウェアをインストールする必要はありません。
- Terraform モジュールは、コードを再利用し、DRY (Don't Repeat Yourself) の原則に従うための強力な方法です。例えば、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、Amazon Elastic Block Store (Amazon EBS) ボリューム、および論理的にグループ化されたその他のリソースを含むアプリケーションに特定の設定があるとします。この設定またはアプリケーションの複数のコピーを作成する必要がある場合は、コード全体を複数回コピーするのではなく、リソースを Terraform モジュールにパッケージ化し、モジュールの複数のインスタンスを作成できます。これらのモジュールは、設定の整理、カプセル化、再利用に役立ちます。また、一貫性を提供し、ベストプラクティスを確保します。
- Terraform は、インフラストラクチャのドリフトを検出および管理できます (Terraform ブログ記事) 。例えば、Terraform によって管理されるリソースが Terraform の外部で変更された場合、Terraform CLI を使用してドリフトを検出し、目的の状態に戻すことができます。

Terraform を使用することの欠点：

- クラウドプロバイダーに関連する新機能や新しいリソースのサポートは利用できない場合があります。
- Terraform は、 のような状態を自動的に管理しません AWS CloudFormation。デフォルトではローカルファイルに保存されますが、Amazon S3 バケットまたは Terraform Enterprise を介してリモートに保存することもできます。
- Terraform 状態には、データベースパスワードなどの機密データが含まれている可能性があり、セキュリティ上の懸念が生じる可能性があります。ステートファイルを暗号化し、リモートで保存し、ファイルバージョニングを有効にして、読み取りおよび書き込みオペレーションに最小権限を使用するのがベストプラクティスです。詳細については、「 AWS Secrets Manager と HashiCorp Terraform を使用した機密データの保護」を参照してください。
- 2023 年 8 月、Hashicorp は Mozilla Public License の下でオープンソースとしてライセンスされなくなると発表しました。代わりに、ビジネスソースライセンス の下でライセンスされるようになりました。

の IaC ツールとしての Pulumi の使用 AWS クラウド

Pulumi はオープンソースのコードとしてのインフラストラクチャツールです。TypeScript、JavaScript、Python、Go、.NET、Java、YAML などの既存の一般的なプログラミング言語をサポートしています。また、ネイティブエコシステムを使用して、Pulumi SDK を介してクラウドリソースとやり取りします。ダウンロード可能な CLI、ランタイム、ライブラリ、ホスト型サービスは連携して、クラウドインフラストラクチャのプロビジョニング、更新、管理を行います。

Pulumi SDK を使用して、コンテナ、サーバーレス関数、ホストされたサービス、インフラストラクチャを使用するクラウドソフトウェアを任意のクラウドに作成およびデプロイできます。

オプションで、Pulumi クラウドと Pulumi をペアリングできます。Pulumi Cloud は、状態とシークレットを保存するマネージドサービスであり、Pulumi インフラストラクチャのデプロイを管理します。

Pulumi を使用する利点:

- Pulumi は、50 を超えるクラウドおよび Software as a Service (SaaS) プロバイダーからのインフラストラクチャを提供しています。
- クラウドの複雑さを軽減するように設計された、完全で一貫性のあるインターフェイスを提供します。

Pulumi を使用することの欠点:

- Pulumi にはサポートを提供するアクティブなコミュニティがありますが、Terraform コミュニティよりも小さくなっています。
- Pulumi の学習曲線は急勾配です。

IaC ツールの選択

では、どのツールを選択すればよいですか？

非常に多くの異なるツールオプションとさまざまなビジネス要件があるため、 one-size-fits-all アプローチはありません。このガイドで説明されている各ツールの利点と欠点に加えて、ビジネス要件と運用モデルに関する以下の推奨事項を検討してください。

- 最小限の依存関係または依存関係でサーバーレス AWS ソリューションを管理またはデプロイする場合は、 AWS Serverless Application Model （ AWS SAM) が適切なオプションである可能性があります。と同じ機能がすべて備わっています AWS CloudFormation。また、サーバーレスアプリケーションのテストと へのデプロイも簡素化されます AWS クラウド。
- インフラストラクチャを で完全に管理している場合は AWS、 AWS CloudFormation と が適切なオプション AWS Cloud Development Kit (AWS CDK) です。 out-of-the-box 状態管理を提供し、新しい機能や AWS リソースをネイティブに使用できます。
- マルチプロバイダーユーティリティ、特にマルチクラウドまたはハイブリッドクラウドのインフラストラクチャを管理する場合は、プラットフォームに依存しないため、Terraform が適している可能性があります。Terraform では、さまざまなプラグインを使用でき、エンタープライズサポートオプションを備えた大規模なコミュニティがあります。
- ベストプラクティスを含むトップダウンディストリビューションがあり、一般的なプログラミング言語を使用して再利用可能なモジュールを作成、公開、配布するオーケストレーションがある場合は、 が適している AWS CDK 可能性があります。
- 組織が高レベルのリスクを許容でき、マルチクラウドまたはハイブリッドクラウド環境をサポートする必要がある場合は、Pulumi の使用を検討してください。

次のステップ

このガイドでは、 AWS リソースとインフラストラクチャの構築、デプロイ、管理に使用できるいくつかの Infrastructure as Code (IaC) ツールの利点と欠点について説明しました。選択したツールに基づいて、以下のリソースにアクセスして開始することをお勧めします。

AWS Cloud Development Kit (AWS CDK)

- AWS CDK 入門ワークショップ

AWS CloudFormation

- AWS CloudFormation ラボ
- AWS CloudFormation ワークショップ

HashiCorp Terraform

- AWS Terraform ワークショップ
- CDK for Terraform リポジトリ (GitHub)

プルミ

- Pulumi リポジトリ (GitHub)
- AWS Pulumi ワークショップによるモダナイゼーション

リソース

AWS ドキュメント

- AWS CDK で を使用して IaC プロジェクト TypeScript を作成するためのベストプラクティス (AWS 規範ガイダンス)
- cdk-nag ルールパックを使用して AWS CDK アプリケーションまたは CloudFormation テンプレートのベストプラクティスを確認する (AWS 規範ガイダンス)
- DevOps の 入門 AWS: コードとしてのインフラストラクチャ (AWS ホワイトペーパー)
- 「AWS CloudFormation ドキュメント」

その他のドキュメント

- Infrastructure as Code の開始方法 (ウェブサイト) HashiCorp
- HashiCorp Terraform ドキュメント
- Pulumi ドキュメント
- Pulumi: Infrastructure as Code とは (Pulumi ウェブサイト)

ツール

- cdk-nag (GitHub)
- cfn-nag (GitHub)
- Terratest

寄稿者

オーサリング

- Siamak Heshmati、シニア DevOps アーキテクト、 AWS
- Mason Cahill、シニア DevOps コンサルタント、 AWS
- Sandip Gangapadhyay、シニア DevOps アーキテクト、 AWS
- シニアリードコンサルタント、Sandeep Gawande AWS
- Rajneesh Tyagi、リードコンサルタント、 AWS

のレビュー

- David Howerton、シニアエンゲージメントマネージャー、 AWS
- Steven Guggenheimer、シニアクラウドアプリケーションアーキテクト、 AWS

テクニカルライティング

- 、 AbouHarbシニアテクニカルライター、 AWS

ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。今後の更新に関する通知を受け取る場合は、RSS フィードをサブスクライブできます。

変更	説明	日付
Pulumi の更新	Pulumi の章を更新しました。	2026 年 2 月 17 日
初版発行	—	2024 年 2 月 23 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。