



AWS ホワイトペーパー

# AWS での 5G ネットワークの継続的な統合と 継続的な配信



# AWS での 5G ネットワークの継続的な統合と継続的な配信: AWS ホワイトペーパー

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは Amazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していない他のすべての商標は、それぞれの所有者の所有物であり、Amazon と提携、接続、または後援されている場合とされていない場合があります。

# Table of Contents

要約 .....	i
要約 .....	1
序章 .....	2
継続的インテグレーションと継続的デリバリー .....	4
継続的インテグレーション .....	4
継続的な配信とデプロイ .....	4
Infrastructure as Code .....	4
での CI/CD AWS .....	5
上の 5G ネットワーク AWS .....	8
5G ネットワークの CI/CD .....	9
CI/CD の詳細ステップ .....	11
ネットワーク設定 .....	12
インフラストラクチャのデプロイ .....	12
クラウドネイティブネットワーク関数のデプロイ .....	12
CNF 継続的デリバリー .....	14
セキュリティ .....	16
オペラビリティ .....	17
サードパーティーおよびオープンソースツールによる CI/CD オーケストレーション .....	20
Terraform .....	20
インフラストラクチャのデプロイ .....	21
ネットワーク関数のデプロイと設定 .....	22
テスト .....	24
CI/CD とオーケストレーション .....	26
結論 .....	27
寄稿者 .....	28
ドキュメントの改訂 .....	29
詳細情報 .....	30
頭字語 .....	31
注意 .....	33
.....	xxxiv

# AWS での 5G ネットワークの継続的インテグレーションと継続的デリバリー

公開日: 2021 年 3 月 8 日 ([ドキュメントの改訂](#))

## 要約

このホワイトペーパーでは、5G ネットワークの継続的インテグレーションと継続的デリバリー (CI/CD) と、Amazon Web Services (AWS) のツールとサービスを使用して 5G ネットワーク機能のデプロイとアップグレードを完全に自動化する方法について説明します。このホワイトペーパーでは、ネットワークのセットアップ、インフラストラクチャのデプロイ、クラウドネイティブネットワーク関数のデプロイ、ネットワーク関数の継続的な更新など、CI/CD for 5G ネットワーク関数のさまざまなステージについて詳しく説明します。また、テスト、オブザーバビリティ、オーケストレーションのためのオープンソースおよびサードパーティーツールとの統合に関する詳細も提供します。

このホワイトペーパーは、コミュニケーションサービスプロバイダー (CSPs) と独立系ソフトウェアベンダー (ISVs)。

# 序章

これまで、セルラーネットワーク内の新しいネットワークノードや新機能の開発、ラボとフィールドの統合テスト、本番環境へのデプロイは、ミッションとビジネスクリティカルな通信 (通信) サービスの安定性を確保するために数週間または数か月かかりました。デプロイの長いサイクルは、従来のネットワークノードのモノリシックアーキテクチャ、マルチベンダー環境、2G, 3G/4G モバイルネットワーク内のネットワークエンティティ間の多くの point-to-point インターフェイスによって引き起こされました。

[AWS による 5G Network Evolution](#) ホワイトペーパーで紹介したように、3GPP で標準化された 5G モバイルネットワークは、仮想化とコンテナ化によって実現されるクラウドネイティブアーキテクチャをサポートするようになりました。3GPP 具体的には、5G ネットワークはマイクロサービス、ステートレス、サービスベースのアーキテクチャの新しいパラダイムを導入し、サポートします。

この 5G アーキテクチャは、さまざまなネットワーク関数が、明確に定義されたインターフェイスと APIs を介して相互に通信する疎結合の独立したサービスとして動作できることを意味します。最も重要なのは、各ネットワーク関数を個別に更新できることです。この 5G のアーキテクチャシフトにより、CSPs、自動化を通じてテスト、セキュリティ要件、標準を維持しながら、ネットワーク機能の更新をより頻繁にロールアウトしやすくすることで、俊敏性と運用効率を向上させることができます。

CSP の新機能の統合とデプロイは、通常、ネットワーク関数ベンダーがコンテナベースのネットワーク関数の [Docker](#) イメージなどの新しいネットワーク関数ソフトウェアパッケージ、または [Kubernetes](#) アプリケーションケースの [Helm](#) チャートなどの新しい設定ファイルをリリースしたときに開始されます。(Helm チャートは、Kubernetes リソースの関連するセットを記述するファイルのコレクションです)。

5G ネットワーク関数のデプロイに CI/CD のパラダイムを使用するという考え方は注目を集めていますが、この考え方の実践的な実現は通信業界では課題となっています。

AWS は、システムの安定性とセキュリティを維持しながら、幅広い業界がソフトウェアの変更を迅速に開発およびロールアウトできるように、ソフトウェア配信用の新しい CI/CD ツールの開発に着手しました。これらのツールには、[CodeCommit](#)、[AWS CodeStar](#)、[CodePipeline](#)、[CodeBuild](#)、[CodeDeploy](#) などの一連の Software Development and Operations (DevOps) サービスが含まれます。

AWS は、[AWS Cloud Development Kit](#) (AWS CDK)、[AWS CloudFormation](#)、および [Terraform](#) などの API ベースのサードパーティーツールを使用して、Infrastructure as Code (IaC) の概念も促進しま

す。これらのツールを使用して、はネットワーク関数のデプロイプロセスをソースコード AWS としてに保存し、この IaC ソースコードを CI/CD パイプラインに維持して継続的な配信を実現 AWS できます。

このホワイトペーパーでは、5G ネットワーク関数のデプロイと更新に AWS IaC ツールと CI/CD ツールを活用する詳細なプロセスについて説明します。さらに、このホワイトペーパーでは、テスト、オブザーバビリティ、オーケストレーションのためのサードパーティーツールとの統合についても説明します。

AWS CI/CD ツールは 5G ネットワーク機能に制限されません。また、4G ネットワークのデプロイを自動化するためにも使用されます。これにより、CSPs 4G ネットワーク機能を迅速かつ効率的にデプロイおよび更新できます。ほとんどの 4G ネットワーク関数は、仮想ネットワーク関数 (VNF) ベースです。などの AWS CI/CD ツールセットを使用して 4G VNFs のデプロイを自動化し、4G ネットワークデプロイのスケールと時間効率を実現 AWS CloudFormation できます。

# 継続的インテグレーションと継続的デリバリー

## 継続的インテグレーション

継続的インテグレーション (CI) は、開発者が定期的にコードを [AWS CodeCommit](#) や [GitHub](#) などの中央リポジトリにプッシュするソフトウェアプロセスです。コードプッシュごとに自動ビルドがトリガーされ、その後にテストが実行されます。CI の主な目的は、コードの問題を早期に発見し、コード品質を向上させ、新しいソフトウェア更新の検証とリリースにかかる時間を短縮することです。

## 継続的な配信とデプロイ

継続的デリバリー (CD) は、アーティファクトがテスト環境、ステージング環境、本番環境にデプロイされるソフトウェアプロセスです。継続的デリバリーは完全に自動化することも、重要な時点で承認ステージを設定することもできます。これにより、リリース管理の承認など、デプロイ前に必要なすべての承認が確実に行われます。継続的デリバリーが正しく実装されている場合、デベロッパーには、標準化されたテストプロセスをパススルーしたデプロイ対応のビルドアーティファクトが常に用意されています。

継続的デプロイでは、開発者からの明示的な承認なしにリビジョンが本番環境に自動的にデプロイされるため、ソフトウェアリリースプロセス全体が自動化されます。これにより、製品ライフサイクルの早い段階で継続的な顧客フィードバックループが可能になります。

継続的デプロイでは、コミットされて自動テストに合格したすべての変更が自動的に本番環境にリリースされます。継続的デリバリーは、コミットされたすべての変更をリリースし、自動テストをすぐに本番環境に渡すことではなく、すべての変更を本番環境に移行する準備を整えることを意図しています。

## Infrastructure as Code

[AWS による 5G Network Evolution](#) ホワイトペーパーで説明されているように、IaC はアプリケーションとその環境の両方のプロビジョニングプロセスとライフサイクル管理を自動化するための主要なドライバーです。ネットワーク/IT 管理者と開発者の両方が、手動で実行されたステップに依存するのではなく、設定ファイルを使用してインフラストラクチャをインスタンス化できます。IaC は、これらの設定ファイルをソフトウェアコードとして扱います。これらのファイルを使用して、一連のアーティファクトを生成できます。つまり、運用環境を構成するコンピューティング、ストレージ、

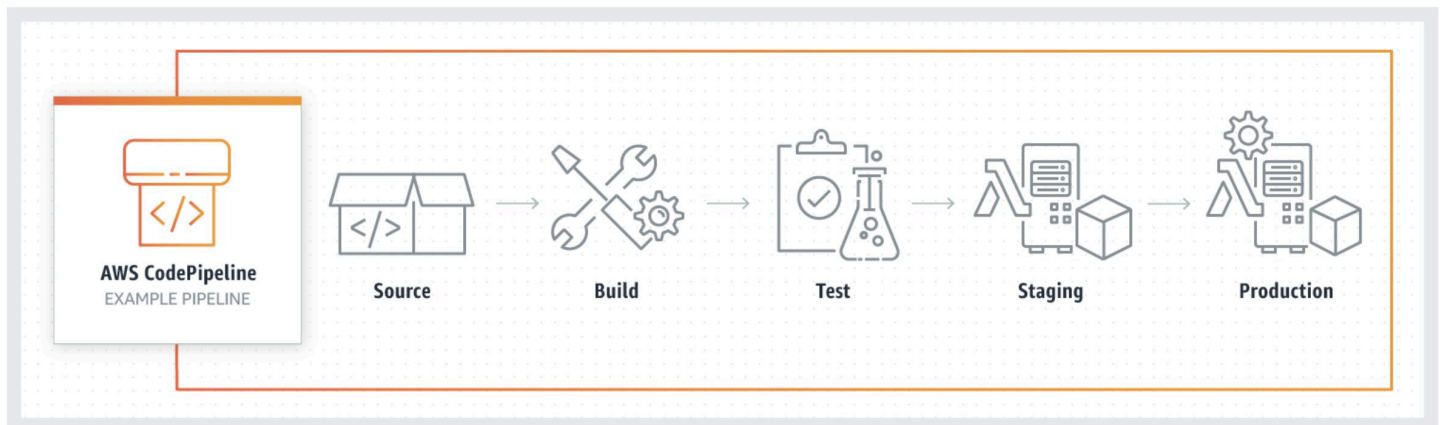
ネットワーク、アプリケーションサービスです。IaC は自動化による設定ドリフトを排除し、インフラストラクチャのデプロイのスピードと俊敏性を向上させます。

で Network Function Virtualization (NFV) を実装する場合 AWS、この IaC フレームワークはオーケストレーションの観点から価値をもたらします。Virtual Private Cloud (VPC) の作成からネットワーク関数のデプロイまで、すべてのステップをプログラムし、ソースコードとして管理し、のバージョン管理で維持できます [AWS CodeCommit](#)。

このネットワーク関数用の IaC フレームワークにより、反復可能で信頼性の高いインフラストラクチャとネットワーク関数の作成とデプロイが可能になります。これは、ネットワークスライス管理とサービスライフサイクル管理の end-to-end (E2E) オートメーションにまで拡張できます。は CloudFormation、Kubernetes AWS CDK 向けの やすべての AWS サービスの API 公開などのサービスを使用して AWS CDK、プログラム、説明、宣言的な方法でインフラストラクチャを作成、保守、デプロイするための包括的なツールセット AWS を提供します。

## での CI/CD AWS

CI/CD はパイプラインとして表現できます。ここでは、新しいコードが一端に送信され、一連のステージ (ソース、ビルド、テスト、ステージング、本番稼働) でテストされ、本番稼働用のコードとして公開されます。



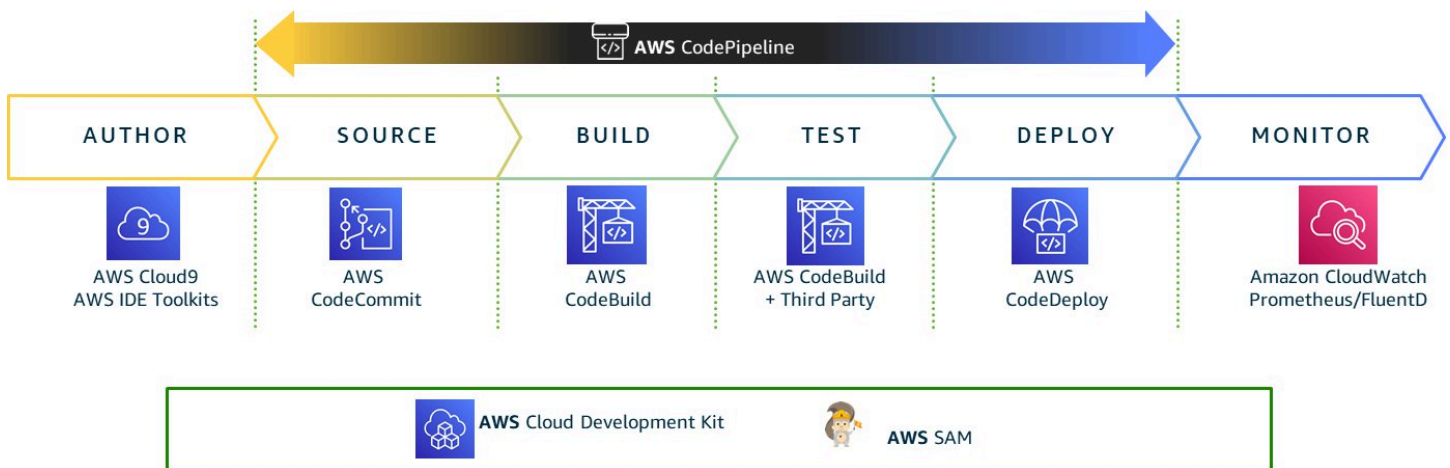
### CICD パイプラインの概要

CI/CD パイプラインの各ステージは、配信プロセスの論理単位として構造化されます。各ステージは、コードの特定の側面を検証するゲートとして機能します。コードがパイプラインを進むにつれて、コードの品質は後続の段階で高くなることを前提としています。これは、コードのより多くの側面が引き続き検証されるためです。初期段階で発見された問題は、コードがパイプラインを通過するのを停止します。テストの結果はすぐにチームに送信され、ソフトウェアがステージに合格しない場合、以降のビルドとリリースはすべて停止します。

AWS は、ソフトウェア開発とリリースサイクルを加速するための CI/CD 開発者ツールの完全なセットを提供します。は、定義されたリリースモデルに基づいて、コードが変更されるたびにリリースプロセスのビルド、テスト、デプロイフェーズ [AWS CodePipeline](#) を自動化します。これにより、機能と更新を迅速かつ確実に配信できます。

コードパイプラインは、他のサービスと統合できます。これらは、[Amazon Simple Storage Service](#) (Amazon S3) などの AWS サービスでも、GitHub などのサードパーティー製品でもかまいません。は、次のようなさまざまな開発および運用のユースケースに対処 [AWS CodePipeline](#) できます。

- を使用したコードのコンパイル、構築、テスト [AWS CodeBuild](#)
- クラウドへのコンテナベースのアプリケーションの継続的な配信
- ネットワークサービスまたは特定のクラウドネイティブネットワーク機能に必要なアーティファクト (記述子やコンテナイメージなど) のデプロイ前検証
- ベースラインテストと回帰テストを含む、コンテナ化されたネットワーク機能/仮想ネットワーク機能 (CNF/VNF) の機能テスト、統合テスト、パフォーマンステスト
- 信頼性とディザスタリカバリ (DR) テスト。



## AWS CICD パイプラインコンポーネント

AWS は、次の AWS デベロッパーツールを使用して CI/CD パイプラインをセットアップできます。

- [AWS CodeCommit](#)
- [AWS CodeBuild](#)
- [AWS CodePipeline](#)
- [AWS CodeDeploy](#)

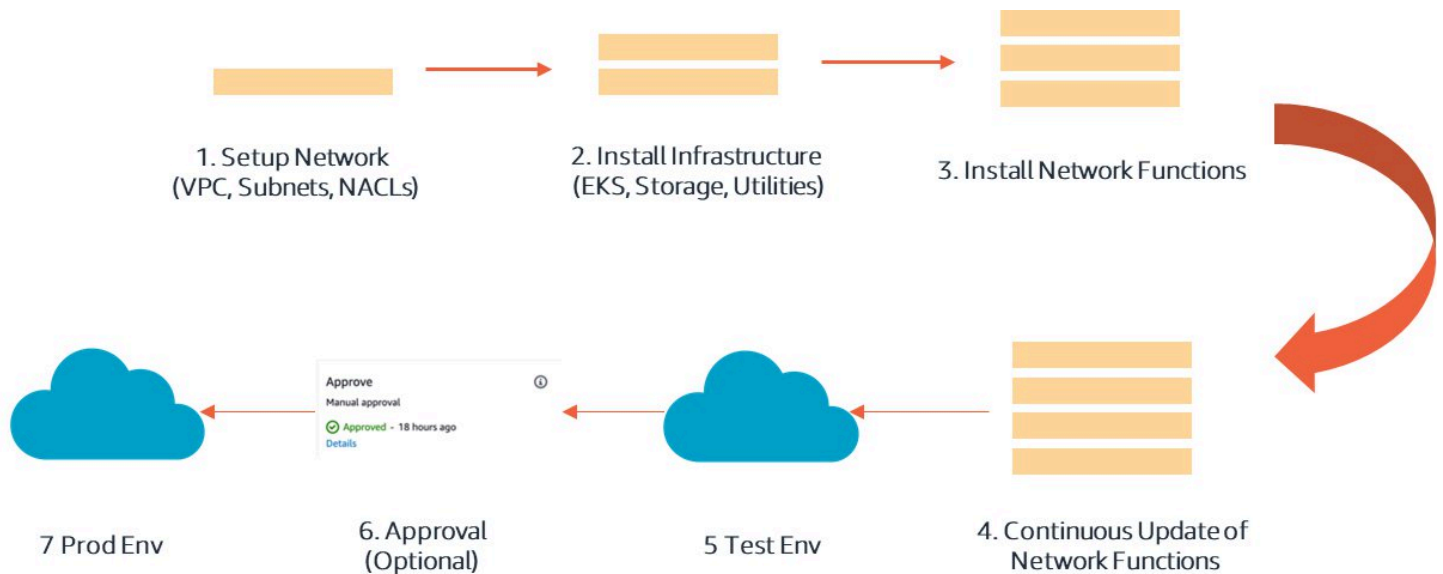
- [Amazon Elastic Container Registry](#)
- [AWS CodeStar](#)

CI/CD パイプラインの作成は、[AWS CDK](#)と [CloudFormation](#) を使用して自動化できます。NFV ドメインでは、この AWS ネイティブオートメーションを Management and Orchestration (MANO) フレームワークと CSP のサービスオーケストレーションフレームワークに統合できます。

CI/CD プロセスには、次のステップが含まれます。

- ネットワークのセットアップ – AWS CDK ネットワーク前提条件の作成 CloudFormation を開始します。
  - ネットワークスタック (VPC、サブネット、ネットワークアドレス変換 (NAT) ゲートウェイ、ルートテーブル、インターネットゲートウェイ)
- インフラストラクチャのデプロイ – AWS CDK 次のリソーススタックの作成 CloudFormation を開始します。
  - コンピューティングスタック ([Amazon Elastic Kubernetes Service](#) (Amazon EKS) クラスターの作成、EKS ワーカーノード、[AWS Lambda](#))
  - ストレージスタック (Amazon S3 バケット、[Amazon Elastic Block Store](#) (Amazon EBS) ボリューム、[Amazon Elastic File System](#) (Amazon EFS))
  - モニタリングスタック ([CloudWatch](#)、[Amazon OpenSearch Service](#) (OpenSearch Service))
  - セキュリティスタック ([AWS Identity and Access Management](#) (AWS IAM)、[Amazon Elastic Compute Cloud](#) (Amazon EC2) セキュリティグループ、VPC [ネットワークアクセスコントロールリスト](#) (NACLs))
- クラウドネットワーク関数 (CNF) のデプロイ – この段階では、CNF は [KubectI](#) および Helm チャートツールを使用して EKS クラスターにデプロイされます。このステージでは、CNFs するために必要な特定のアプリケーションやツール ([Prometheus](#) や [Fluentd](#) など) もデプロイします。CNFsは、Lambda 関数を介してデプロイすることも、[Fluentd](#) を使用してデプロイすることもできます AWS CodeBuild。
- 継続的な更新とデプロイ – これらは、アップグレードにつながるコンテナ/設定の変更の一部である変更をデプロイするために繰り返し実行される一連のステップです。CNF デプロイケースと同

様に、継続的な更新とデプロイは、[Amazon Elastic Container Registry](#) (Amazon ECR)、または [GitLab Webhooks](#) などのサードパーティソースシステムからトリガーして [AWS CodeCommit](#)、AWS サービスを使用して自動化できます。



## AWS CI/CD パイプラインのフロー図

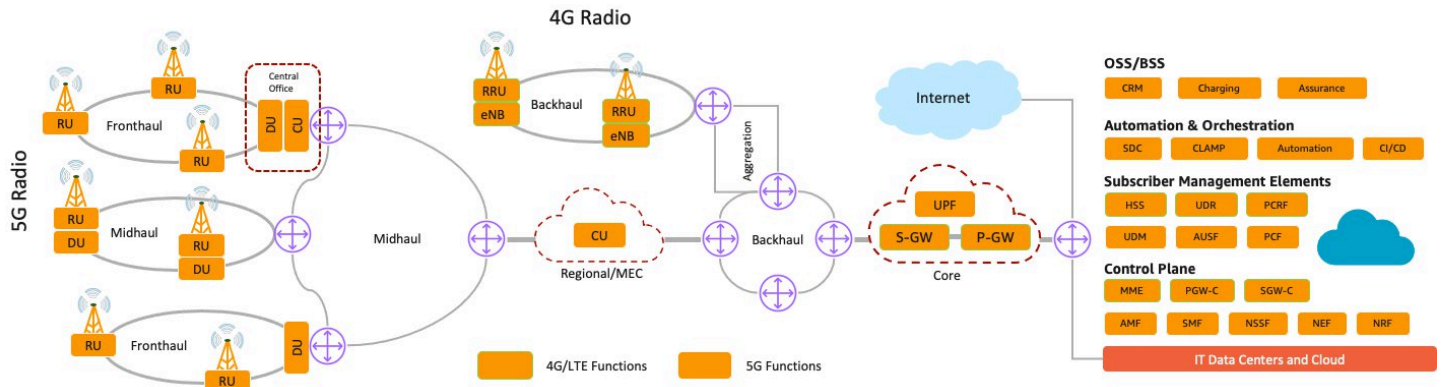
CI/CD パイプラインは [AWS CodePipeline](#) を使用して構築され、ソフトウェアのリリースに必要なステップをモデル化、視覚化、自動化する継続的デリバリーサービスを利用します。パイプラインのステージを定義することで、ソースコードリポジトリからコードを取得し、そのソースコードをリリース可能なアーティファクトに構築し、アーティファクトをテストして、本番環境にデプロイできます。これらのすべてのステージを正常に通過したコードのみがデプロイされます。必要に応じて、手動承認などの他の要件をパイプラインに追加して、承認された変更のみが本番環境にデプロイされるようにできます。

## 上の 5G ネットワーク AWS

5G ネットワークインフラストラクチャの一般的なモデルは、4G/5G 無線サイト、fronthaul/midhaul/backhaul ネットワーク、コアネットワークサイト、および通信/IT データセンターで構成されています。CSPs は、AWS サービスを使用して、初期投資コストを削減しながら、スケーラブルで柔軟な 5G ネットワークインフラストラクチャを作成できます。は、オペレーションサポートシステム/ビジネスサポートシステム (OSS/BSS) とコントロールプレーンコアネットワーク機能の大部分をホストするリージョンに仮想ネットワークオペレーションセンター (NOC) を実装するために AWS 使用できます。

AWS も活用できます。は、UPF (ユーザープレーン関数)、RAN 中央ユニット (CU)、マルチアクセスエッジコンピューティング (MEC) などの主にユーザープレーン関数をホストする [AWS Outposts](#) インスタンスのフリートを使用して、ローカル中央オフィス (CO) または分散データセンターを実装します。リファレンスアーキテクチャとでの 5G ネットワーク実装の利点の詳細については AWS、AWS での [5G Network Evolution](#) ホワイトペーパーで説明されています。

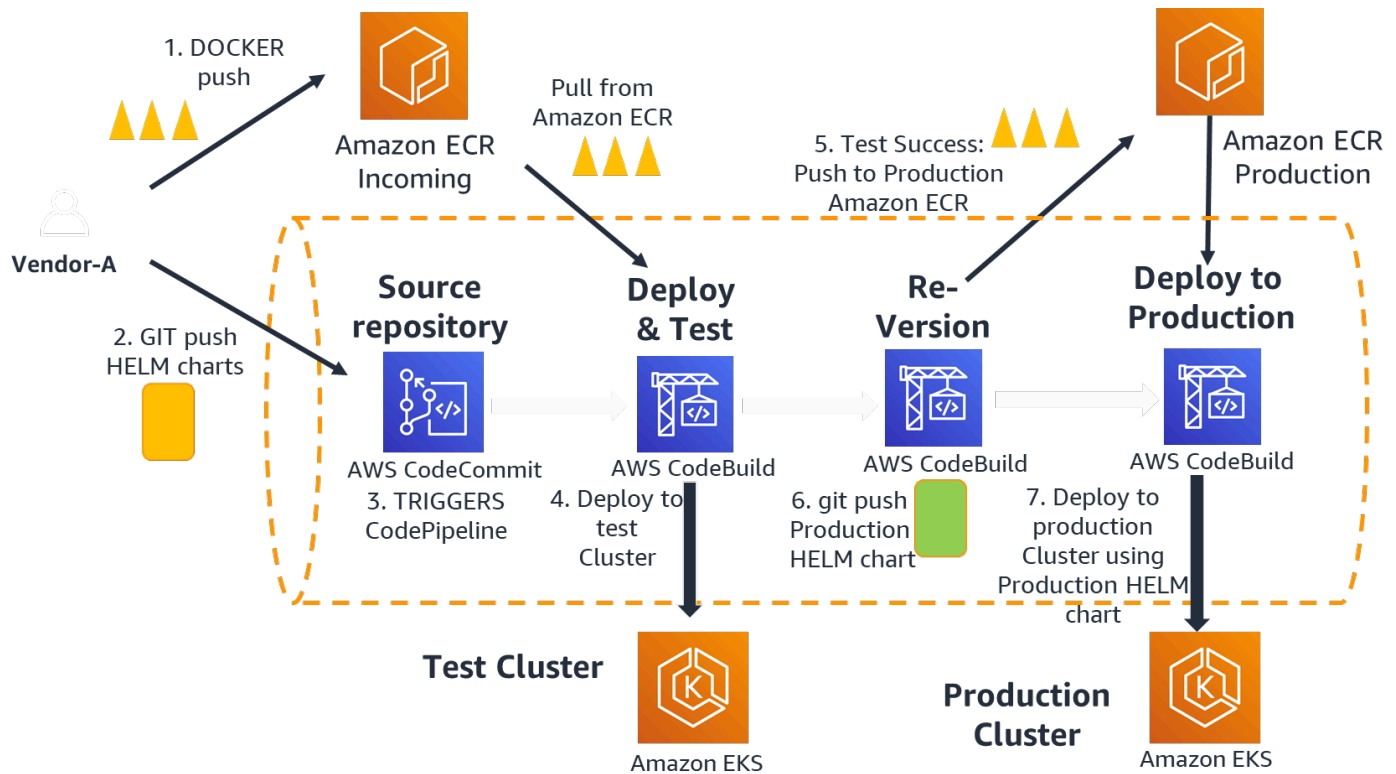
5G ネットワークを実装するとき、このホワイトペーパーの以下のセクションで紹介する AWS AWS CI/CD ツールを使用すると、5G ネットワーク機能のデプロイ、アップグレード、ライフサイクル管理を完全に自動化できます。



## 5G ネットワーク E2E アーキテクチャ

## 5G ネットワークの CI/CD

インフラストラクチャの設計構造は、宣言言語を使用してコード形式で保存されます。これにより、CSP は、必要に応じて同じ期待される動作でインフラストラクチャを繰り返し再現できます。コードはコードリポジトリに保持され、パイプラインはデプロイされたスタック (など) の更新を調整するように設定されています AWS CDK CloudFormation。AWS は、独立系ソフトウェアベンダー (ISV) 関数のアジャイルオンボーディングのためのコードとしてのインフラストラクチャ (IaC) を構築するのに役立ちます。



## コードパイプラインフロー

Helm チャートによるクラウドネイティブネットワーク関数設定の変更は、ネットワーク関数の自動 CI/CD パイプライン実行のトリガーと見なされます。

AWS CodeCommit を使用して設定ファイルを維持し、Amazon ECR を使用してコンテナイメージを保持できます。

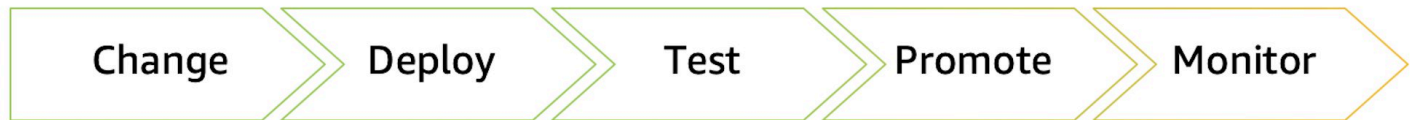
コードパイプラインのフロー図に示すように、ISV が新しいコード変更をコードリポジトリ (Helm チャート、設定ファイル、またはプロパティファイル) にプッシュすると、コードパイプラインがトリガーされます。コードパイプラインは ECR からイメージをプルし、Helm チャートを使用してアプリケーションをデプロイします。新しいアプリケーションテストは、サードパーティーのテスト自動化フレームワークと統合できます。その結果に基づいて、CSPs 本番デプロイを承認できます。

CodePipeline ソースステージは、設定ファイルの変更を検索します。ソースステージの有効なプロバイダーは、CodeCommit、Amazon S3、GitHub、または CloudFormation です。代替ソースシステムは、Lambda 関数を使用して Webhook を実装することで統合できます。これにより、Gitlab と間のイベント駆動型統合が可能になります AWS CodePipeline。詳細な実装ガイドについては、以下のリンクを参照してください。

- [GitLab を使用したウェブフック](#)

## • コンテナレジストリの統合

CI/CD パイプラインの設計では、テスト結果が期待に合致し、ベースラインに照らして検証された後、初期デプロイ、テスト、本番環境への昇格などの重要なデプロイステップを考慮する必要があります。パイプラインプロセスの各ステージでデータアーティファクトが提供されるため、比較やデータ駆動型の意味決定が可能になります。



### アプリケーションの CI/CD パイプラインステップ

各ステージは個別のタスクと見なすことができ、ネットワークサービスとクラウドネイティブなネットワーク機能をサポートするのに十分な検証とデプロイワークフローを組み込むことができます。実行タスクには、トラフィックジェネレーターやシミュレーターなどの追加のサードパーティーツールを組み込むことができ、end-to-endのネットワークサービス検証が可能になります。

AWS は、他の AWS サービスとネイティブに統合する高度な [AWS Step Function](#) (クラウドネイティブステートマシン) サービスを提供し、Jira やテストオートメーションフレームワークなどの外部システムと統合することもできます。

## CI/CD の詳細ステップ

CI/CD はパイプラインとして表現できます。パイプラインでは、新しいコードが一端に送信され、一連のステージ (ソース、ビルド、テスト、ステージング、本番稼働) でテストされ、本番稼働用のコードとして公開されます。

デプロイとテストの手順は次のとおりです。デプロイと設定は主に 4 つの主要セクションに分かれています。

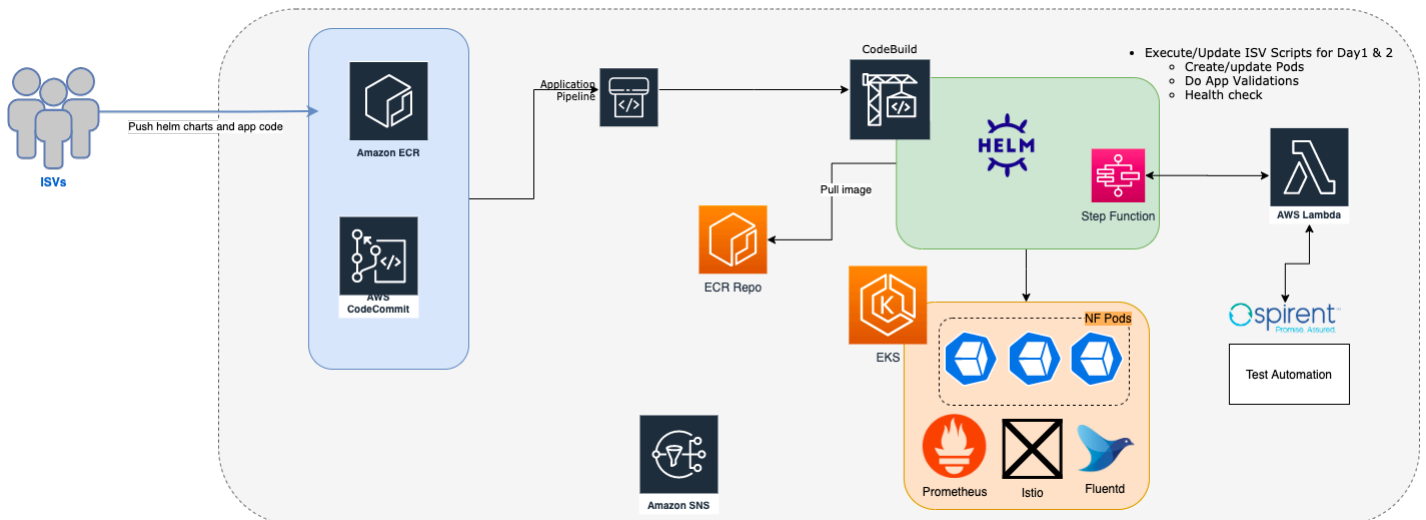
- ネットワーク設定
- インフラストラクチャのデプロイ
- クラウドネイティブネットワーク関数のデプロイ
- CNF 継続的デリバリー

## ネットワーク設定

インフラストラクチャの前提条件の設定に焦点を当てます。これには、VPC、ネットワーク、サブネット、NACLs の作成などが含まれます。ISVs と顧客の実装計画 (マルチテナンシーや静的割り当てと動的割り当てなど) を考慮して IP ネットワーク計画を設計します。このプランは、AWS CDK またはコーディングできません CloudFormation。このコードを実行すると、クラウドインフラストラクチャネットワークの前提条件がデプロイされます。

## インフラストラクチャのデプロイ

インフラストラクチャデプロイは、インフラストラクチャコンポーネントをプロビジョニングします。これには、EFS、EFS ワーカーノード、ELBs などの EKS EFS クラスタとサポートインフラストラクチャのスポーン、およびクラウドネイティブネットワーク機能の要件に従ったクラスタの設定が含まれます。CNF 要件に基づいて、は [Multus](#) インターフェイスを含むノード用の追加のネットワークインターフェイス AWS もデプロイします。ほとんどのデプロイおよび設定ステップは、アプリケーションの 1 回限りの作業であり、アプリケーションの更新として必要な場合にのみ更新されます。



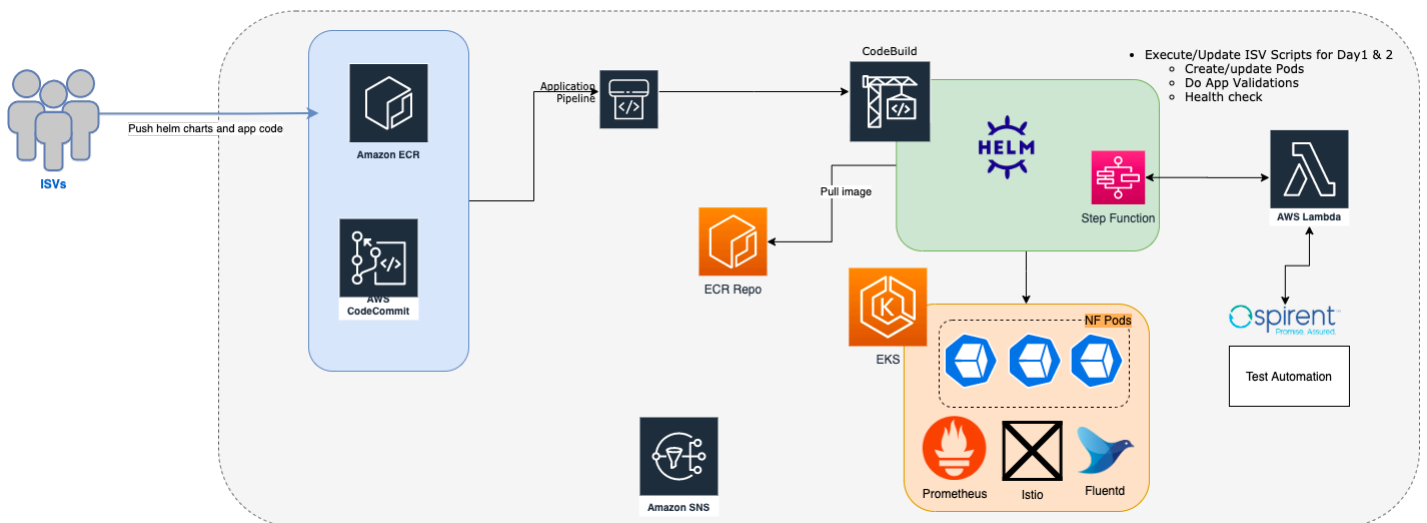
CDK を使用したインフラストラクチャのデプロイ

## クラウドネイティブネットワーク関数のデプロイ

CNF デプロイは、アプリケーションのデプロイに関するものです。CNF デプロイの一環として、アプリケーションの Helm チャートは CI/CD コードパイプラインを通じて実装されます。主に事前チェックと事後チェックを含む個々のアプリケーション固有のスクリプトを実行するコールバックが

組み込まれています。Helm チャートは、アプリケーションのニーズに応じて順番に実装され、デプロイの次のステップに進む前に Kubernetes PODs のステータスを確認します。多くの場合、ISVs は Helm チャートとサニティチェックを実行するラッパースクリプトを提供します。これらの ISV スクリプトは内部から呼び出されます AWS CodePipeline。このフェーズの一環として、Prometheus や Fluentd などのログ記録およびモニタリングエージェントは、アプリケーションのクラウドインフラストラクチャをログ記録およびモニタリングする Amazon CloudWatch に加えてデプロイされます。

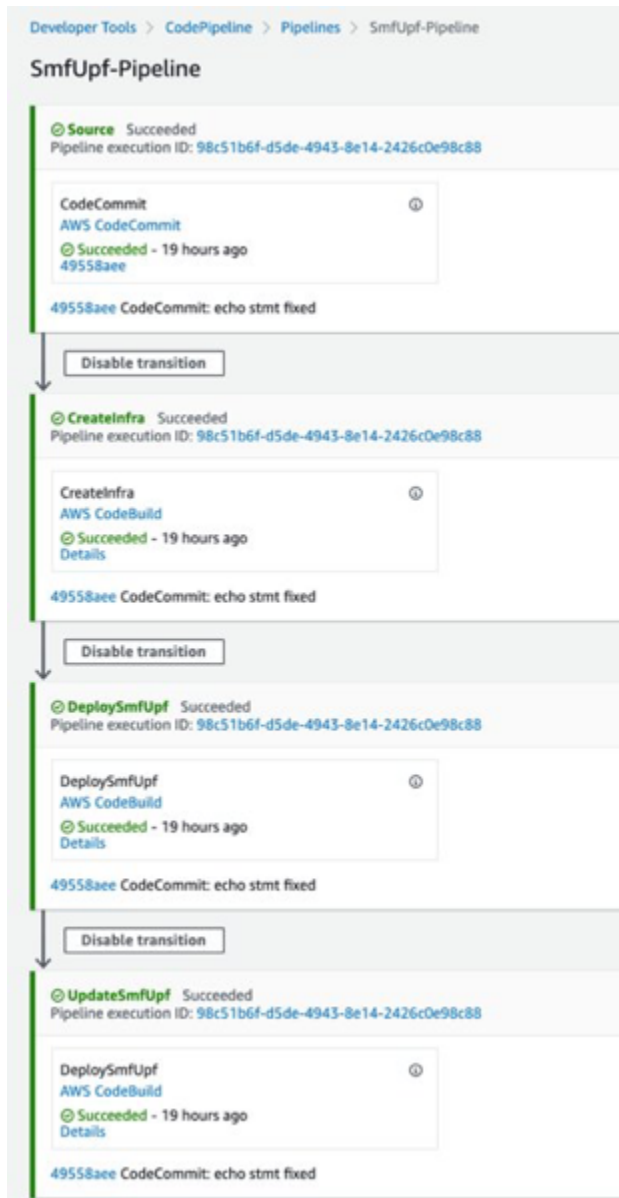
コードパイプラインは、サードパーティーのテスト自動化フレームワークと統合されています。コードパイプラインは、テスト自動化フレームワーク APIs を直接呼び出して、デプロイされたアプリケーションでテストを実行し、テスト結果をクエリして、結果を分析できます。これにより、アプリケーションのデプロイとテストが簡素化されます。



## アプリケーションのデプロイと更新

以下は、を介したユーザープレーン関数/セッション管理関数 (UPF/SMF) CNF のデプロイの例です AWS CodePipeline。

- CodeCommit、CodeBuild、CodePipeline を使用して CI/CD プロセス全体を自動化します。
- Infra の作成タスクとアプリケーションのインストールタスクは、パイプラインの一部として統合されます。
- FluentD エージェントと Prometheus エージェントは、Amazon CloudWatch ダッシュボードにインストールおよび作成されます。



UPF/SMF CNFs のデプロイ例

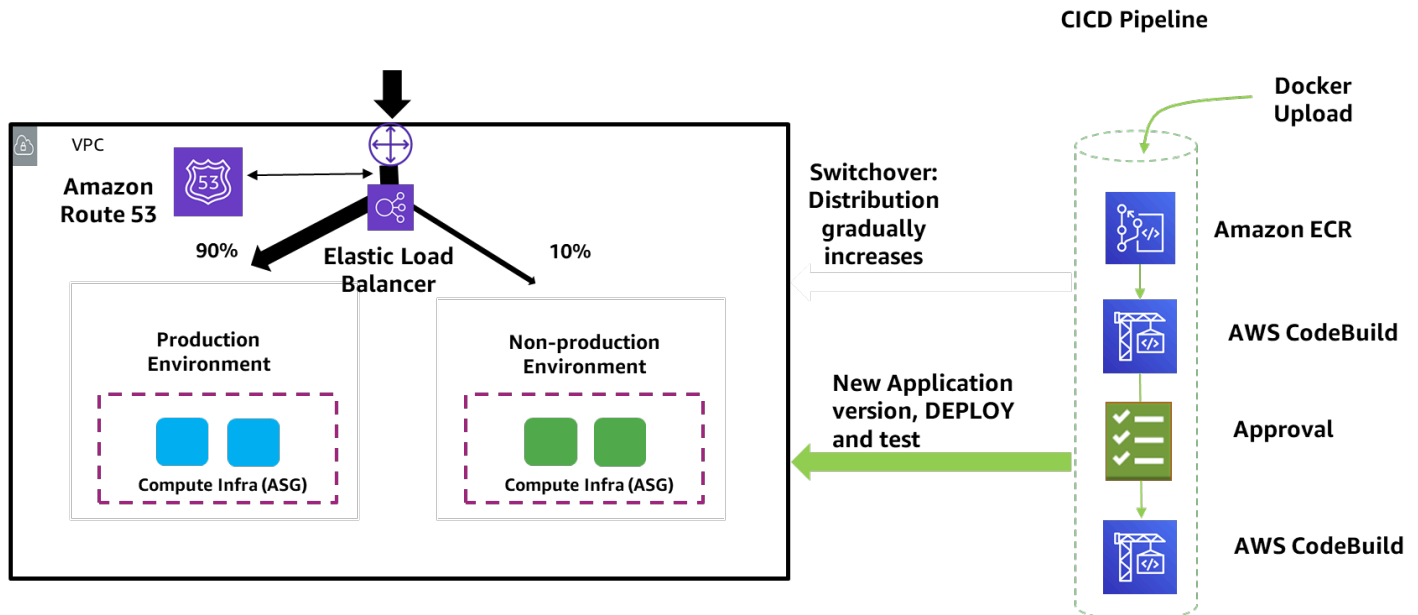
## CNF 継続的デリバリー

このステップは、アップグレードにつながるコンテナ/設定の変更の一部である変更をデプロイするために繰り返し実行される一連のステップで構成されます。CNF の継続的デリバリーはパイプラインを介して自動化され、個々のアプリケーションに固有です。は、標準の Helm チャート AWS を使用して特定の CNFs を更新します。コードパイプラインには、アプリケーションの更新ステータスの事前チェックと事後チェックがあります。更新された CI/CD パイプラインは、自動テストを実行するテスト自動化フレームワークとも統合されています。この抽象化により、ネットワーク関数をクリーンにデプロイできます。

CNF の継続的な配信とデプロイは、次のカテゴリに大別できます。

- アプリケーションのアップグレード — ほとんどのアプリケーションのアップグレードは、Kubernetes アプリケーション PODs。これらの更新は、コードパイプラインを通じて自動的に適用できます。ほとんどの CNFs アプリケーション PODs の複数のインスタンスを提供することで、インプレースアップグレードをサポートしています。複数のインスタンスでは、ローリングアップグレードアプローチが可能です。すべてのアプリケーション POD 変更が Helm アップグレードをサポートしているわけではありません。Pipelines はこれらのバリエーションを考慮し、必要に応じて Helm のインストール/削除を使用します。
- メジャーアップグレード — メジャーアップグレードは主にデータベーススキーマの変更です。この変更は、ダウンタイムを発生させることなく適用することはできません。これらの変更に対する標準的なアプローチは、アプリケーションを削除し、関連するポッドを再作成することです。プロセス中にアプリケーションが使用できない場合があります。アップグレードには、次のツールが使用されます。
  - [AWS CloudFormation](#) を使用すると、すべてのインフラストラクチャリソースを JSON テンプレートまたは YAML テンプレートで記述およびプロビジョニングできます。は、Lambda-backed カスタムリソースを通じて強力な拡張機能メカニズム CloudFormation を提供します。お客様は AWS リソース AWS CloudFormation を超えて拡張し、ハイブリッド環境のオンプレミスリソースなど、他の環境で必要なリソースをプロビジョニングできます。AWS CDK は、Python、TypeScript、JavaScript、Java、C# などの高レベルの使い慣れたプログラミング言語を使用してコードをビルドし、コードを低レベルの CloudFormation JSON 形式にコンパイルしてデプロイできます。
  - BlueGreen デプロイ — Blue/Green および Canary ベースのデプロイをテスト環境と本番環境で AWS サポートおよび推奨します。 [ブルー/グリーンデプロイを使用すると](#)、お客様は、含まれている環境で新しいアプリケーションバージョンをテストできます。本番トラフィックを切り替えるための簡単で適切な方法を提供します。 [Canary ベースのデプロイでは](#)、本番稼働用以外のグリーン環境を少数の本番稼働用トラフィックでテストして、本番稼働用トラフィックに起因する問題を検出できるようにすることで、この概念を拡張します。新しいアプリケーションバージョンは、内部のシミュレートされたテストトラフィックと少量の本番トラフィックに対してテストされるため、本番トラフィックを切り替える前にユーザーに自信を与えます。本番トラフィックは、スイッチオーバーが完了するまで徐々に増加します。実装には、加重 DNS と加重 ELB ターゲットグループが含まれます。

- 自動化は、Blue/Green および Canary ベースのデプロイステージ AWS CodePipeline で設定することで実現できます。承認ステージは、プロビジョニング中に最初に手動で実行できますが、後で完全に自動化する必要があります。テスト環境では、本番環境にデプロイする前に、常にロールバックアクションを使用してテストし、前方互換性と後方互換性を検証することをお勧めします。サービスマッシュを使用するクラスターへの Blue/Green デプロイは、サービスマッシュのエンドアプリケーションとルーティングゲートウェイによって提供されるサポートに依存し、正常な移行を実現します。
- [AWS Systems Manager](#) は統合されたユーザーインターフェイスを提供するため、CI/CD によってデプロイされたネットワーク関数で使われる複数の AWS サービスの運用データを表示できます。Systems Manager を使用すると、リソース全体の運用タスクを自動化できます AWS。



## Canary デプロイ

## セキュリティ

セキュリティは重要な要素です。以下は、アプリケーションをデプロイするときに CI/CD AWS プロセスが考慮するセキュリティステップのリストです。

- ソースベンダーに割り当てられた ECR リポジトリは、「プッシュ時のスキャン」フラグを有効にして設定されているため、Docker イメージのアップロードは直ちにセキュリティスキャンの

対象となります。既知の一般的な脆弱性と露出 (CVE) には、通知でフラグが付けられます。ECR とは別に、ベンダーがチャートを AWS CodeCommit リポジトリに配置すると、プレーンテキストではなく Secrets Manager で使用されるパスワードを暗号化するように求められます。

- アーティファクトの整合性 — パイプライン全体で使用されるアーティファクトは、保管中 (AWS マネージドキーを使用) か転送中 (SSL/TLS を使用) にかかわらず暗号化されます。
- ユーザーとロール — ユーザーまたはリソースに提供されるアクセス許可は、最小権限の原則に基づいています。異なるサービスのリソース間で運用している場合は、クロスロール信頼関係を設定する必要があります。たとえば、には Amazon EKS クラスターでコマンドを実行するアクセス許可 AWS CodeBuild が必要です。
- 監査 — が提供する監査機能は、 のサービスおよびユーザーオペレーション全体で各 API コール [AWS CloudTrail](#) を追跡し、過去のイベントの評価を可能にします。
- イメージ脆弱性スキャン — Amazon ECR にアップロードされた CNF イメージは、セキュリティの脆弱性を自動的にスキャンします。スキャン結果のレポートは で利用でき [AWS マネジメントコンソール](#)、API を使用して取得することもできます。その結果は、CNF イメージの置き換えなど、是正措置のために CSP オペレーターに送信できます。

セキュリティチェックは、パイプラインのさまざまな段階で行われ、新しくアップロードされたイメージが安全であり、必要なコンプライアンスチェックに準拠し、承認のために CSPs に通知を送信できるようにします。

- コンテナレジストリは、開いている CVE の脆弱性をスキャンします。
- 設定は、テスト段階で、既知の個人を特定できる情報 (PII) パターンである情報漏洩がないかチェックされ、予期しないオープン TCP/UDP ポートや DOS 脆弱性などの問題のコンプライアンスチェックルールがトリガーされます。
- 後方互換性と前方互換性は、アップグレード/ロールバックの安全性について検証されます。

アプリケーションとは別に、保管中か転送中にかかわらず、ステージ間でアーティファクトを暗号化して転送できるようにして、パイプラインセキュリティをプロビジョニングすることが重要です。

## オブザーバビリティ

AWS は、AWS デフォルトで にデプロイされている 5G CNFs のオブザーバビリティを有効にします。これは Amazon CloudWatch によって有効になります。CloudWatch は、クラウドリソースとアプリケーションを完全に可視化します。

Amazon CloudWatch には、このプロセス中に 4 つの主要なステップがあります。

1. 収集 — およびオンプレミスサーバーで実行されるすべての AWS リソース、アプリケーション、サービスからメトリクス AWS とログを収集します。
2. モニタリング — CloudWatch ダッシュボードを使用してアプリケーションとインフラストラクチャを視覚化し、ログとメトリクスを並行して関連付けてトラブルシューティングを行い、[CloudWatch アラーム](#) でアラートを設定します。
3. Act — [CloudWatch Events](#) とを使用して、運用上の変更への対応を自動化します [AWS Auto Scaling](#)。
4. 分析 — [CloudWatch Metric Math](#) を使用した最大 1 秒のメトリクス、拡張データ保持 (15 か月)、リアルタイム分析。

Amazon CloudWatch エージェントは、顧客の Kubernetes クラスターにインストールされます。エージェントは Prometheus [の設定](#)、検出、メトリクスプル機能をサポートし、すべての忠実度の高い Prometheus メトリクスとメタデータを [埋め込みメトリクス形式](#) (EMF) として [CloudWatch Logs](#) に強化して公開します。

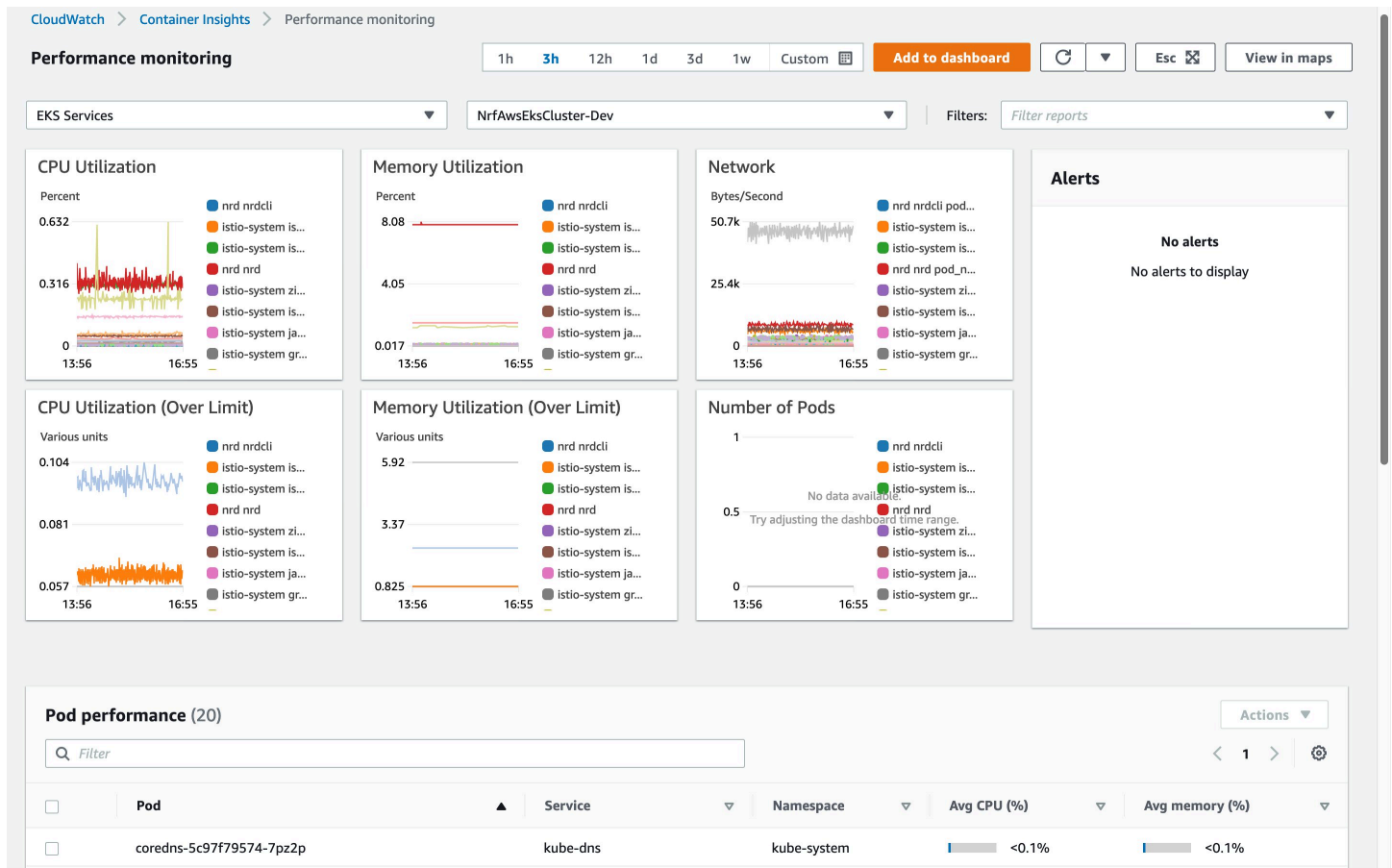
[Amazon CloudWatch Container Insights](#) は、コンテナ化されたアプリケーションからの Prometheus メトリクスの検出と収集を自動化します。ダッシュボードで視覚化された集約されたカスタム CloudWatch メトリクスを自動的に収集、フィルタリング、作成します。

各イベントは、完全に設定可能な厳選されたメトリクスディメンションのセットの CloudWatch カスタムメトリクスとしてメトリクスデータポイントを作成します。集約された Prometheus メトリクスを CloudWatch カスタムメトリクス統計として公開すると、パフォーマンスの問題や障害のモニタリング、アラーム、トラブルシューティングに必要なメトリクスの数が減少します。[CloudWatch Logs Insights クエリ言語](#) を使用して忠実度の高い Prometheus メトリクスを分析し、コンテナ化された環境のヘルスとパフォーマンスに影響を与える特定のポッドとラベルを分離することもできます。

AWS CloudTrail は、この可視性を提供し、サービス間のすべての API コールを記録します。[AWS Config](#) は、コンプライアンス検証の機能を提供します。AWS は、[AWS X-Ray](#) やなどのさまざまなサービスを使用して、アプリケーション、インフラストラクチャ、パイプラインのメトリクス、ログ、イベントの追加のモニタリングオプションをお客様に提供します [AWS CloudTrail](#)。

- AWS は、Prometheus、Fluentd などのオープンソースのメトリクスツールをネイティブに統合できます。
- [Prometheus メトリクス](#) は、詳細な分析のために Amazon CloudWatch または OpenSearch Service にさらに取り込むことができます。
- AWS は、さまざまなシステムからログを収集するための標準メカニズムとして fluentD を使用します。このプロジェクトでは、同じメカニズムが使用され、設定されます。

このメカニズムの設定方法の詳細については、[CloudWatch Logs にログを送信する DaemonSet として FluentD を設定する](#) を参照してください。



## Amazon CloudWatch モニタリングメトリクスの例

# サードパーティーおよびオープンソースツールによる CI/CD オーケストレーション

オーケストレーションレイヤーは IaC を使用して、5G ネットワーク機能の実行に必要な基盤となるインフラストラクチャをデプロイおよび設定します。このレイヤーは、モジュール式、ポータブル、再利用可能なように設計されている必要があります。

インフラストラクチャはクラウドネイティブのベストプラクティスに従っており、可用性が高く、冗長性があり、スケーラブルです。

前のセクションで説明したように、下線インフラストラクチャのデプロイは [AWS Cloud Development Kit \(AWS CDK\)](#) を使用して実現できます。これは、Hashicorp によって [Terraform](#) を使用して実現できます。

## Terraform

Terraform は、何百ものクラウドサービスを管理するための一貫したコマンドラインインターフェイス (CLI) ワークフローを提供する IaC ソフトウェアツールです。Terraform はクラウド APIs を宣言型設定ファイルに体系化します。

Terraform を使用したデプロイでは、CDK で使用されているのと同じ原則を使用します。このコードは、ベンダーの要件に従ってネットワークコンポーネントをカスタマイズして再利用できるモジュールで構成されています。

設定はすべてパラメータ化されているため、プロバイダーと ISV の推奨事項に従ってデプロイを完全に調整できます。

ネットワーク関数のデプロイは 2 つのフェーズに分かれています。

- 必要な AWS インフラストラクチャは、中央リポジトリを介して作成および管理されます。
- 設定とコードは GitHub リポジトリに一元的に保存されます。

前提条件が作成されると、前のステージで設定されたアプリケーションパイプラインを使用して、ネットワーク関数をデプロイする準備が整います。

# インフラストラクチャのデプロイ

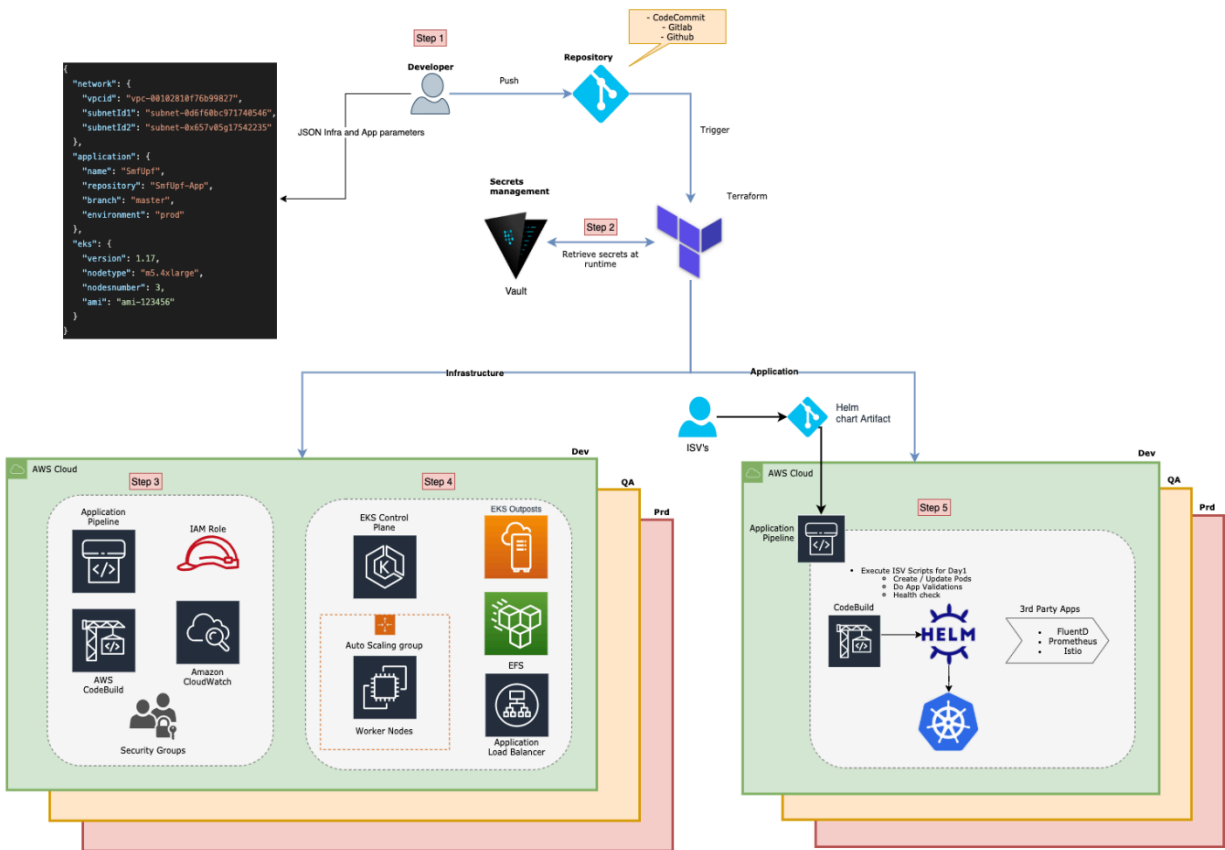
インフラストラクチャのデプロイには、ネットワーク関数を正常にデプロイして設定するためのすべての前提条件が含まれます。

このフェーズの一部として作成されたコンポーネントには、次のようなものがあります。

- ネットワーク — VPC、パブリックサブネットとプライベートサブネット、ルート、ロードバランサー
- コンピューティング — Kubernetes ( [Vmware Tanzu](#)、Amazon EKS、または AWS Outposts)、Amazon EC2 インスタンスのプライマリノードとワーカーノード、自動スケーリンググループ
- ストレージ — Amazon EFS、Amazon EBS、Amazon S3 バケット
- セキュリティ — [セキュリティグループ](#)
- パイプライン — CodePipeline、CodeBuild
- オブザーバビリティ — CloudWatch、Prometheus、FluentD

以下は、Terraform によってオーケストレーションされ、次の図で説明されているインフラストラクチャシーケンスです。

1. 開発者は、中央リポジトリに保存されている JSON ファイルに IaC コードを入力します。ファイルには、インスタンスサイズ、Kubernetes バージョン、ネットワーク情報、アプリケーションリポジトリの詳細など、必要なインフラストラクチャ設定に関する情報が含まれています。
2. 実行時に HashiCorp ポールトまたは [AWS Secrets Manager](#) からシークレットを取得します。
3. インフラストラクチャコンポーネント (ネットワーク、コンピューティング、ストレージ、セキュリティ) をデプロイして設定します。
4. ネットワーク関数ポッドをホストするワーカーノードを持つ Amazon EKS クラスタがデプロイされます。Amazon EKS は、データセンターに近接する必要があるワークロードをサポートするために [AWS Outposts](#) にデプロイすることもできます。
5. アプリケーションパイプラインが作成され、ネットワーク関数リポジトリの変更をリッスンするように設定されます。コードが設定されたリポジトリブランチにプッシュされるたびに、パイプラインはネットワーク関数の構築、テスト、デプロイを自動的にトリガーします。
6. ログとメトリクスを収集して一元化するオブザーバビリティツールは、すべてのノードにサービスとしてデプロイされ、[Grafana](#) または [OpenSearch Dashboards](#) で視覚化できるほぼリアルタイムのデータを提供します。



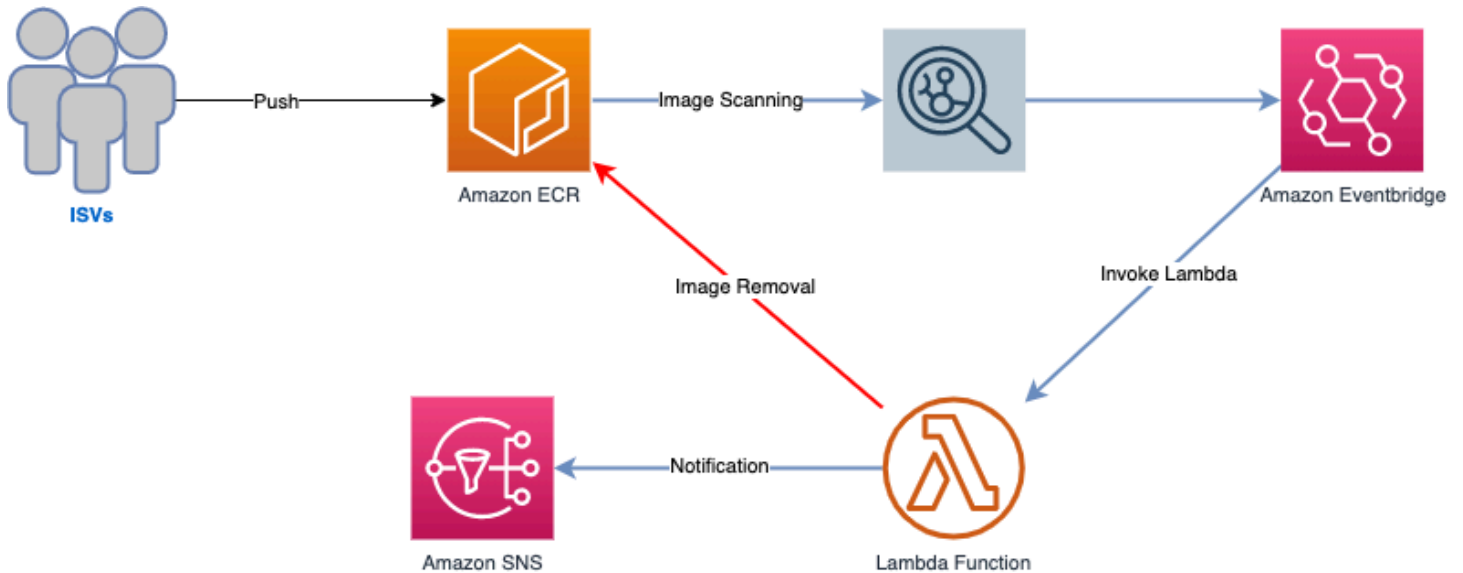
## ネットワーク関数のデプロイと設定

## ネットワーク関数のデプロイと設定

前のステージで作成されたパイプラインによりISVs とプロバイダーの両方がネットワーク機能のデプロイを分散および最適化できます。パイプラインが接続され、前述の図のステップ 1 の JSON ファイルで設定されているアプリケーションリポジトリの変更をリッスンします。

第三者によって公開されたイメージを検証するために、コンテナイメージ内のソフトウェアの脆弱性を特定するのに役立つ脆弱性スキャンソリューションがデプロイされ、設定されます。スキャンソリューションは、[Amazon ECR](#) にプッシュされたすべての新しいイメージを自動的にチェックします。ECR イメージスキャンの詳細については、「[イメージスキャン](#)」を参照してください。

次の図は、Image Vulnerability Scanning ソリューションのアーキテクチャを示しています。



### Image Vulnerability Scanning ソリューションのアーキテクチャ

アプリケーションパイプラインは、スキャン結果後のイメージの変更、またはリポジトリ内の直接の変更によってトリガーされるように設定できます。たとえば、新しい Helm イメージが作成された場合です。

次のリストは、次の図に示すように、ネットワーク関数を作成/アップグレードするシーケンスです。

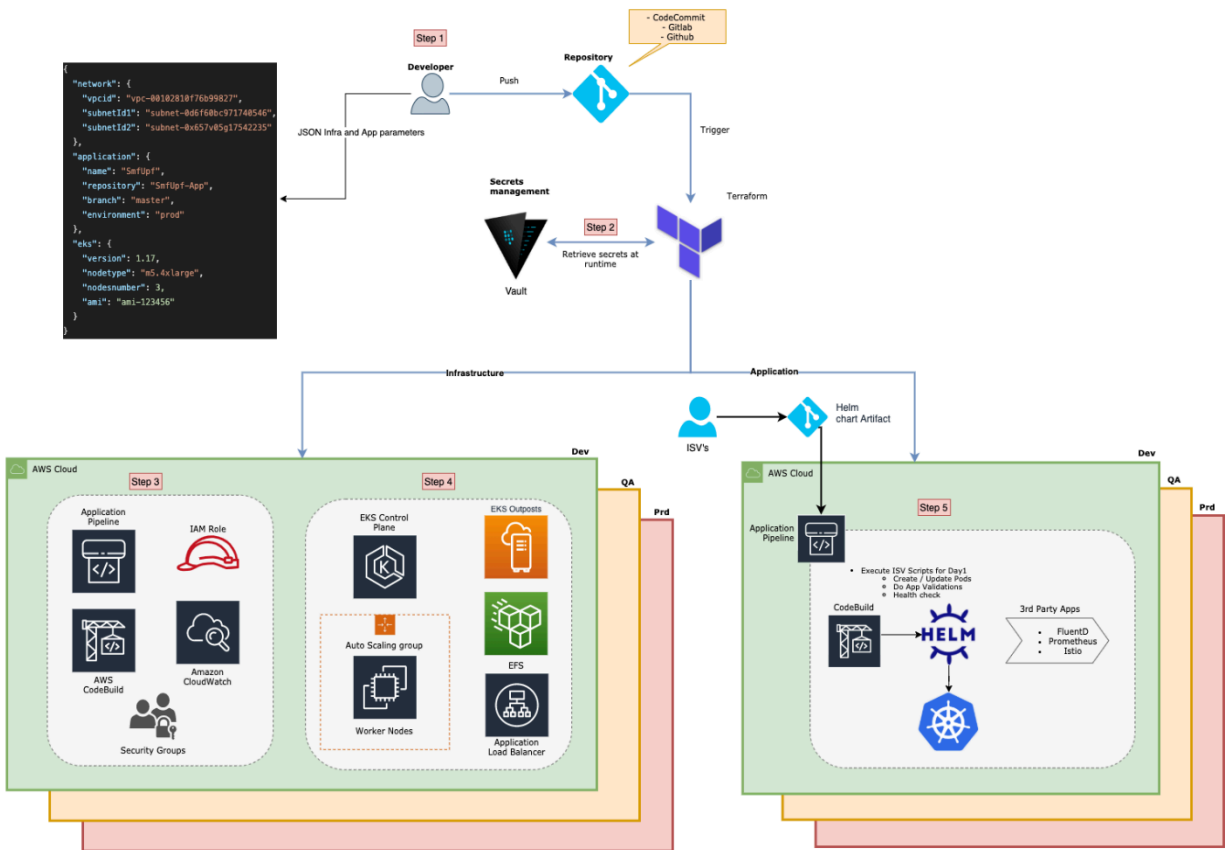
ISVs Amazon ECR に新しいイメージを発行します。(イメージが承認されると、アプリケーションパイプラインがトリガーされます。

CodePipeline は Amazon ECR から新しいイメージをプルし、CodeBuild を使用してイメージを Kubernetes にデプロイします。Helm コマンドを使用して、ネットワーク関数をアップグレードできます。

イメージがデプロイされると、サービスとしてのテスト (TaS) がトリガーされます。TaS は新しいデプロイを検証し、ストレスの下でネットワーク関数のパフォーマンスに関するデータとメトリクスを一元化します。

ログとメトリクスは OpenSearch と Grafana で収集され、一元化されます。[Datadog](#)、[Istio](#)、Prometheus などのサードパーティーも、追加のオブザーバビリティを提供するように設定できます。

ネットワークリソースを調整できる MANO をデプロイしてソリューションと統合することもできます。収集されたデータを使用して、ネットワークスライシングやサービス品質 (QoS) 自動スケーリングなどの自動アクションを実行します。



## アプリケーションパイプライン

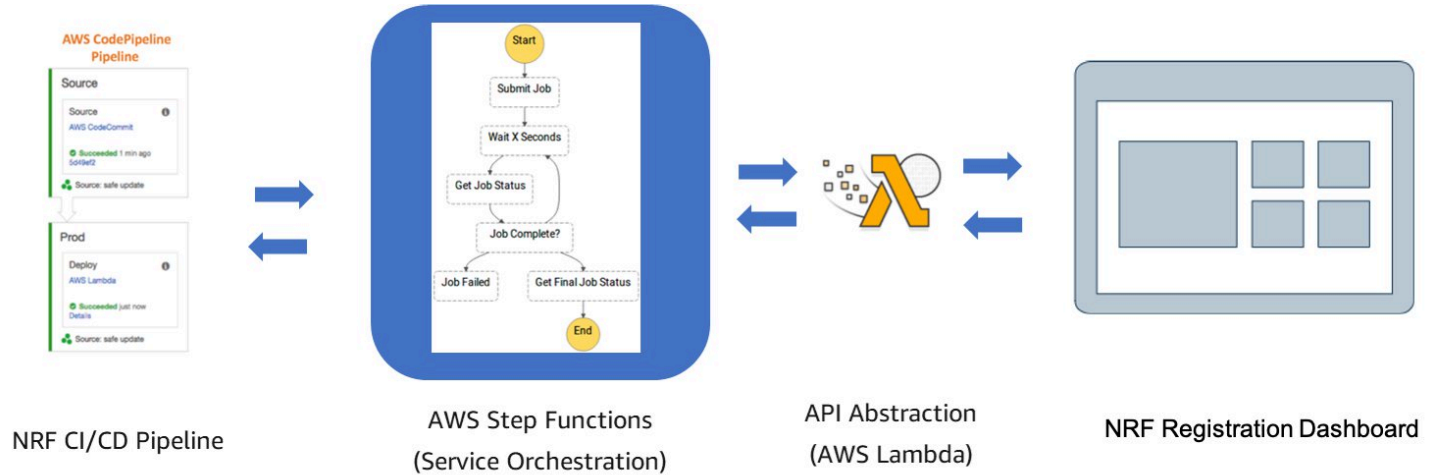
## テスト

通信固有のテスト自動化フレームワークは、コードパイプラインに統合できます。コードパイプラインはステップ関数と統合され、テスト自動化フレームワークとの統合を調整します。AWS ステップ関数は、複数の Lambda 関数を使用して、API コールを介してテスト自動化フレームワーク (サードパーティー製ツール) を呼び出します。ステップ関数は、最初にテスト ID を取得し、テストを実行し、テスト自動化フレームワークから結果を取得します。次に、ステップ関数は結果を分析してコードパイプラインに渡し、本番デプロイ用のアプリケーションを承認します。承認は、必要に応じてコードパイプラインで自動化または手動で維持できます。これは、CSPs がデプロイをテスト環境から本番環境に昇格させるための重要なステップです。統合に必要な高レベル APIs は、次のように分類されます。

- コンテキストの取得
- 特定のテストケースを実行する
- テストケースを停止する

- 結果の取得

外部 REST APIs呼び出しの複雑さは、AWS ステップ関数を使用してモデル化されます。これにより、標準コンストラクトは並列フローを呼び出し、結果を待機し、条件に基づいて分岐し、REST API をと統合できます AWS CodePipeline。



## テストフロー

## CI/CD とオーケストレーション

継続的インテグレーションと継続的デリバリーは、クラウドネイティブアーキテクチャとそれが 5G にどのように適用されるかに付随する全体的な自動化哲学の一部です。オーケストレーションは、この哲学のもう 1 つの側面であり、ネットワークで発生する変更に対して動的かつ反応的であることが期待されます。オーケストレーションと CI/CD は、正常なサービスを保証し、サービスの中断を最小限に抑えるために緊密に連携することが期待されます。CI/CD とオーケストレーションの統合は、次の 2 つの面で行われます。

- パッチとアップグレードをシステムに適用するには、ライブサービスの中断を最小限に抑える方法で管理およびオーケストレーションする必要があります。たとえば、オーケストレーションは、更新をロールアウトする最適なタイミングを動的に決定できます。
- CI/CD 対応オーケストレーションを使用すると、採用されたデプロイモデル戦略 (カナリア、線形、all-at-once) に基づいて、アップグレードのデプロイ中にトラフィックを移行できます。

通常、オーケストレーションソリューションは CI/CD パイプライン上で実行され、オーケストレーションがそれらのパイプラインにガバナンスフェーズを導入し、継続的なアップグレードサイクルに曝露できるようにします。

## 結論

CI/CD は、開発者やアプリケーションチームが数分で新しいアプリケーションコードをデプロイするための明確で効率的な方法を提供します。AWS には、AWS CodePipeline など、開発者が新しいコードの統合、テスト AWS CodeCommit AWS CodeBuild、デプロイに役立つ豊富なツール AWS CodeDeploy があります。このドキュメントでは、AWS サービスを使用して、新しいコードのデプロイを完了するために必要なさまざまなステップなど、5G ネットワーク関数を完全自動化された方法でデプロイするための CI/CD プロセスを作成する方法について説明しました。また、サードパーティーのテスト自動化フレームワークを CI/CD プロセスに統合する方法と、Terraform などのサードパーティーツールを使用する方法についても調べました。

## 寄稿者

本ドキュメントの寄稿者は次のとおりです。

- アマゾン ウェブ サービス、AWS Telecom、シニアコンサルタント、Hisham Elshaer
- Vara Prasad Talari、プリンシパルコンサルタント、AWS Telecom、Amazon Web Services
- Amazon Web Services、AWS Telecom、プリンシパルコンサルタント、Rabi Abdel
- アマゾン ウェブ サービス、共有配信、シニアコンサルタント、Franco Bontorin
- Pragtideep Singh、コンサルタント、共有配信、Amazon Web Services
- Subbarao Duggisetty、Cloud Infra Architect、グローバルアカウント、Amazon Web Services
- Young Jung、シニアパートナーソリューションアーキテクト、AWS Telecom、Amazon Web Services

# ドキュメントの改訂

このホワイトペーパーの更新に関する通知を受け取るには、RSS フィードにサブスクライブしてください。

変更	説明	日付
<a href="#">初版発行</a>	ホワイトペーパーの初回発行	2021 年 3 月 8 日

## 詳細情報

詳細については、次を参照してください。

- [AWS での継続的インテグレーションと継続的デリバリーの実践](#) (ホワイトペーパー)
- [AWS でのキャリアグレードモバイルパケットコアネットワーク](#) (ホワイトペーパー)
- [AWS による 5G ネットワークの進化](#) (ホワイトペーパー)

## 頭字語

- AMF — アクセスとモビリティ管理関数
- API — アプリケーションプログラミングインターフェイス
- AUSF — Authentication Server 関数
- BSS — ビジネスサポートシステム
- CDK — クラウド開発キット
- CI/CD — 継続的インテグレーションと継続的デリバリー
- CLI — コマンドラインインターフェイス
- CNF — クラウドネイティブまたはコンテナ化されたネットワーク関数
- CSP — 通信サービスプロバイダー
- CU — RAN 中央ユニット
- CVE — 一般的な脆弱性と露出
- DoS — サービス拒否
- DR — ディザスタリカバリ
- DU — RAN 分散ユニット
- E2E — エンドツーエンド
- ECR — Elastic Container Registry
- EFS — Elastic File System
- EKS — Elastic Kubernetes Service
- EPC — 進化したパケットコア
- IaC — Infrastructure as Code
- ISV — 独立系ソフトウェアベンダー
- MANO — 管理とオーケストレーション
- MEC — マルチアクセスエッジコンピューティング
- NACL — ネットワークアクセスコントロールリスト
- NAT — ネットワークアドレス変換
- NF — ネットワーク関数
- NFV — ネットワーク関数の仮想化
- NFVO — Network Function Virtualization Orchestrator

- NOC — ネットワークオペレーションセンター
- NRF — ネットワークリポジトリ関数
- OSS — オペレーションサポートシステム
- PII — 個人を特定できる情報
- QoS — サービスの品質
- RAN — 無線アクセスネットワーク
- SBI — サービスベースのインターフェイス
- SMF — セッション管理関数
- SSL — セキュアソケットレイヤー
- TaS — サービスとしてのテスト
- TCP — Transmission Control Protocol
- TLS — Transport Layer セキュリティ
- UDM — 統合データ管理
- UDP — ユーザーデータグラムプロトコル
- UPF — ユーザープレーン関数
- VIM — 仮想化インフラストラクチャマネージャー
- VNF — 仮想ネットワーク関数
- VPC — 仮想プライベートクラウド

## 注意

お客様は、本書に記載されている情報を独自に評価する責任を負うものとし、本書は、(a) 情報提供のみを目的とし、(b) AWS の現行製品と慣行について説明しており、これらは予告なしに変更されることがあり、(c) AWS およびその関連会社、サプライヤー、またはライセンサーからの契約上の義務や保証をもたらすものではありません。AWS の製品やサービスは、明示または黙示を問わず、一切の保証、表明、条件なしに「現状のまま」提供されます。お客様に対する AWS の責任は AWS 契約によって規定されています。また、本文書は、AWS とお客様との間の契約に属するものではなく、また、当該契約が本文書によって修正されることもありません。

© 2021 Amazon Web Services, Inc. or its affiliates. All rights reserved.

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。