AWS ホワイトペーパー

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

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

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: AWS ホワイトペーパー

Copyright © 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 のものではない製品またはサービスと関連付けてはならず、また、お客様に混乱を招くような形や Amazon の信用を傷つけたり失わせたりする形で使用することはできません。Amazon が所有しない商標はすべてそれぞれの所有者に所属します。所有者は必ずしも Amazon と提携していたり、関連しているわけではありません。また、Amazon 後援を受けているとはかぎりません。

Table of Contents

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 08 日 (改訂履歴)

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) について説明し、アマゾン ウェブ サービス (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をご紹介します。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なネットワーク機能のデプロイ、ネットワーク機能の継続的な更新など、5G ネットワーク機能のための CI/CD の各ステージについて、詳しく説明します。さらに、テスト、可観測性、オーケストレーションのためのオープンソースツールおよびサードパーティー製ツールとの統合についても詳細な情報を提供しています。

このホワイトペーパーは、通信サービスプロバイダー (CSP) と独立系ソフトウェアベンダー (ISV) を対象としています。

はじめに

従来、移動体通信ネットワークにおける新しいネットワークノードや新機能の開発、ラボ/フィールド統合テスト、本番環境へのデプロイには、ミッションクリティカルおよびビジネスクリティカルな電気通信サービスの安定性を確保するために数週間から数か月かかっていました。このようにデプロイに時間がかかるのは、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、および 2G、3G、および 4G モバイルネットワークのネットワークエンティティ間で使用する多数のポイントツーポイントインターフェイスが原因でした。

ホワイトペーパー「AWS での 5G ネットワークの進化」で紹介されているように、3GPP によって標準化された 5G モバイルネットワークでは、仮想化とコンテナ化によって実現するクラウドネイティブアーキテクチャがサポートされるようになりました。具体的には、5G ネットワークでは、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による新しいパラダイムを取り入れ、これをサポートしています。

この 5G アーキテクチャが持つ意味は、さまざまなネットワーク機能が、疎結合された個別のサービスとして動作し、明確に定義されたインターフェイスと API を介して相互に通信できるということです。最も重要なことは、各ネットワーク機能を個別に更新できる点です。この 5G アーキテクチャにおける転換により、CSP はテスト、セキュリティ要件、標準を自動化によって管理しつつ、ネットワーク機能の更新をより頻繁に、より簡単にロールアウトできます。

CSP 向けの新機能の統合とデプロイは、通常、ネットワーク機能ベンダーが新しいネットワーク機能ソフトウェアパッケージ (コンテナベースのネットワーク機能内の Docker イメージなど) や、新しい設定ファイル (Kubernetes アプリケーションケース内の Helm チャートなど) をリリースしたときに開始されます (Helm チャートは、Kubernetes リソースの関連セットを記述するファイルの集まりです)。

5G ネットワーク機能のデプロイに CI/CD のパラダイムを使用するという考え方に注目が集まりつつありますが、この考え方の実用化は長い間、通信業界での課題でした。

AWS はソフトウェア配信用の新しい CI/CD ツールをいち速く開発し、システムの安定性とセキュリティを維持しながら、ソフトウェアの変更を迅速に開発および展開できるよう幅広い業界を支援しています。これらのツールには、AWS CodeStar、CodeCommit、CodePipeline、CodeBuild、CodeDeploy などの DevOps (ソフトウェア開発および運用) サービスが含まれます。

また、AWS では、AWS Cloud Development Kit や AWS CloudFormation の他、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) の考え方を広めてい

ます。AWS ではこれらのツールを使用することで、ネットワーク機能のデプロイプロセスをソースコードとして AWS 内に保存し、この IaC ソースコードを CI/CD パイプラインに保持して継続的デリバリーを実現できます。

このホワイトペーパーでは、5G ネットワーク機能のデプロイおよび更新に AWS の IaC および CI/CD ツールを活用するための詳細なプロセスについて説明します。さらに、テスト、可観測性、オーケストレーションのためのサードパーティ製ツールとの統合についても取り上げます。

AWS の CI/CD ツールは 5G ネットワーク機能に限定されません。また、4G ネットワークのデプロイを自動化するためにも採用され、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るように CSP を支援しています。ほとんどの 4G ネットワーク機能は、仮想ネットワーク機能 (VNF) ベースです。AWS CloudFormation などの AWS CI/CD ツールセットを使用すると、4G VNF 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高めることができます。

継続的インテグレーションと継続的デリバリー (CI/CD)

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、AWS CodeCommit や GitHub などのセントラルリポジトリにデベロッパーが定期的にコードをプッシュするソフトウェアプロセスです。コードをプッシュするたび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てコードの品質を高め、新しいソフトウェアアップデートの検証とリリースにかかる時間を短縮することです。

継続的デリバリーと継続的デプロイ

継続的デリバリー (CD) は、アーティファクトをテスト環境、ステージング環境、および本番環境にデプロイす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、重要なポイントに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を確実に実施できます。継続的デリバリーが適切に実装されていれば、デベロッパーは、標準化されたテストプロセスに合格しデプロイ準備の整ったビルドアーティファクトが常に用意されている状態になります。

継続的デプロイでは、デベロッパーからの明示的な承認がなくても自動的にリビジョンが本番環境にデプロイされ、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を実現できます。

継続的デプロイでは、コミットされ自動テストに合格した変更はすべて、本番環境に自動的にリリースされます。継続的デリバリーは、コミットされ自動テストに合格したすべての変更を直ちに本番環境にリリースするのではなく、すべての変更を本番環境に移行できる状態にしておくことを目的としています。

Infrastructure as Code

ホワイトペーパー「AWS での 5G ネットワークの進化」に詳しく記載されているように、IaC はアプリケーションとその環境の両方で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重要な推進力です。手動で実行する手順に依存するのではなく、ネットワーク/IT 管理者とデベロッパーは、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 ではこれらの設定ファイルがソフトウェアコードとして扱われます。これらのファイルは、オペレーション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など、

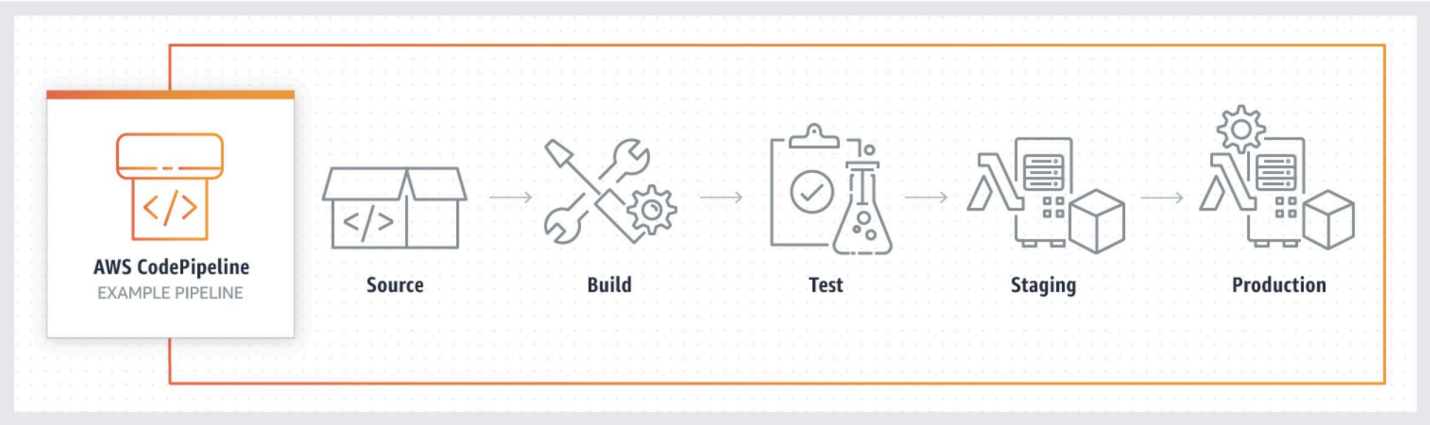
一連のアーティファクトを生成するために使用できます。IaC は、自動化によって設定ドリフトを排除し、インフラストラクチャのデプロイスピードと俊敏性を高めます。

AWS でのネットワーク機能仮想化 (NFV) 実装の場合、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。仮想プライベートクラウド (VPC) の作成からネットワーク機能のデプロイまで、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て、AWS CodeCommit のバージョン管理によって保守できます。

このネットワーク機能用 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機能を作成およびデプロイし、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エンドツーエンド (E2E) 自動化にまで拡張できます。AWS は、Kubernetes 向けの AWS CloudFormation、AWS CDK、AWS CDK などのサービスや、すべての AWS サービスの API 接続を使用して、プログラミング的、記述的、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を提供します。

AWS 上の CI/CD

CI/CD は、パイプラインとして考えることができます。パイプラインでは、一方の端で新しいコードが引き渡され、一連のステージ (ソース、ビルド、ステージング、本番) でテストされた後、本番環境で使用できるコードとして公開されます。



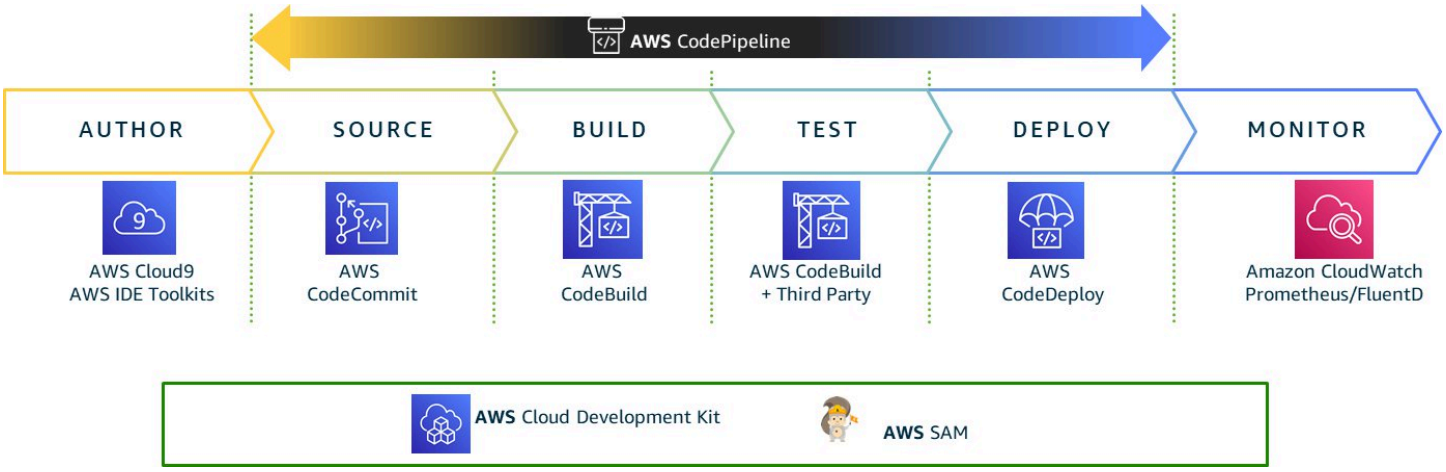
CICD パイプラインの概要

CI/CD パイプラインの各ステージは、デリバリープロセスの論理ユニットとして構造化されています。各ステージは、コードの特定の側面を精査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、より多くの側面が検証され続けるため、後のステージではコードの品質が高くなると想定されます。早い段階で問題が明らかになったコードについては、パイプラインでの進行が停止します。テストの結果は即座にチームに送信され、ソフトウェアがステージを正常に通過しなければ、それ以降のビルドやリリースはすべて停止されます。

AWS は、ソフトウェア開発とリリースサイクルを加速する CI/CD デベロッパーツール一式を提供しています。AWS CodePipeline 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、およびデプロイの各フェーズを自動的に実行します。これにより、機能やアップデートの迅速かつ信頼性の高い配信が可能になります。

コードパイプラインは他のサービスと統合できます。例えば、Amazon Simple Storage Service (Amazon S3) や、GitHub などのサードパーティー製品が対象になります。AWS CodePipeline 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応できます。

- AWS CodeBuild によるコードのコンパイル、ビルド、テスト
- クラウドに対する、コンテナベースのアプリケーションの継続的なデリバリー
- 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、コンテナイメージなど) のデプロイ前の検証
- コンテナ化ネットワーク機能/仮想ネットワーク機能 (CNF/VNF) の機能、統合、およびパフォーマンステスト (ベースラインテストと回帰テストを含む)
- 信頼性および災害対策 (DR) テスト



AWS CICD パイプラインコンポーネント

AWS では、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

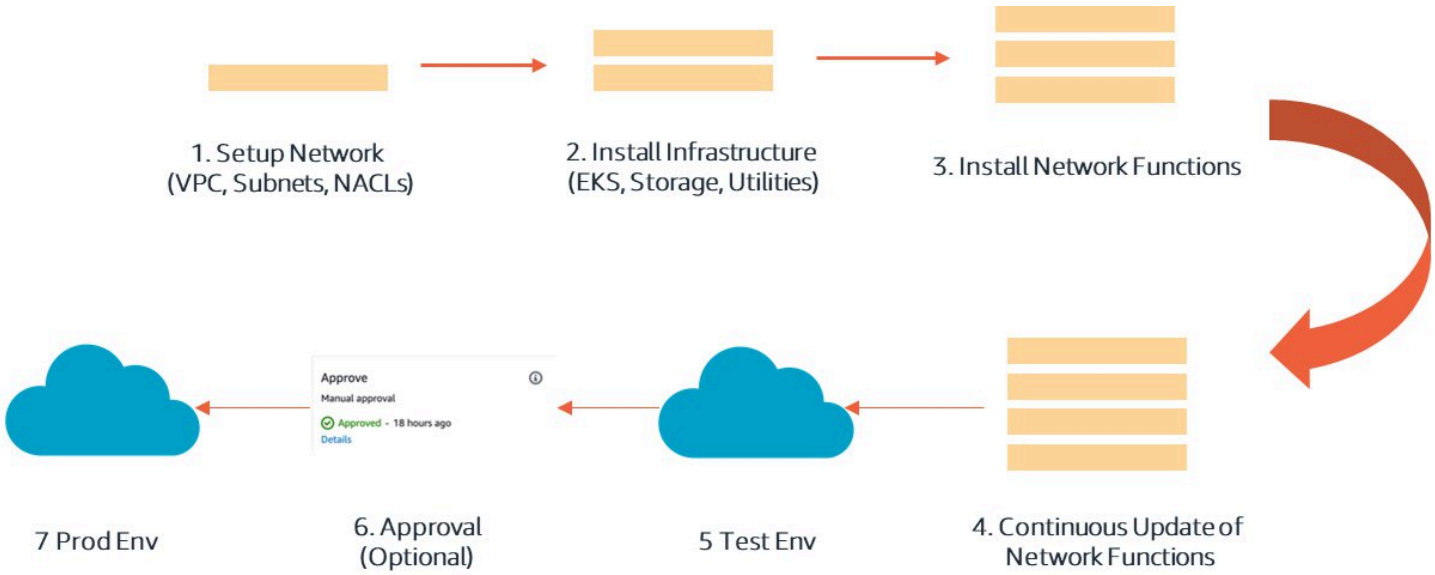
- AWS CodeDeploy
- Amazon Elastic Container Registry
- AWS CodeStar

CI/CD パイプラインの作成は、AWS CDK および AWS 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 MANO (管理およびオーケストレーション) 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、以下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 および AWS CloudFormation により、ネットワーク前提条件の作成が開始されます:
    - 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 および AWS CloudFormation により、次のリソーススタックの作成が開始されます:
    - 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 (Simple Storage Service (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 (NACL)
- クラウドネットワーク機能 (CNF) のデプロイ – このステージでは、Kubectl および Helm チャートツールを使用して CNF が EKS クラスターにデプロイされます。このステージでは、CNF の効率的な動作に必要な特定のアプリケーションやツール (Prometheus や Fluentd など) もデプロイされます。CNF は、Lambda 関数または AWS CodeBuild を使用してデプロイできます。

- 継続的な更新とデプロイ — コンテナ/設定への変更をデプロイしてアップグレードを行うために、反復的に実行される一連のステップです。CNF のデプロイの場合と同様、AWS CodeCommit、Amazon Elastic Container Registry (Amazon ECR)、またはサードパーティ製ソースシステム (GitLab Webhooks など) からのトリガーにより、AWS のサービスを使用して継続的な更新とデプロイを自動化できます。



AWS の CICD パイプラインフローの図

CI/CD パイプラインは、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て、AWS CodePipeline で構築します。パイプラインでステージを定義することにより、ソースコードリポジトリからコードを取得して、そのソースコードをビルドしてリリース可能なアーティファクトを作成し、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。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、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、手動承認など他の要件をパイプラインに追加できます。

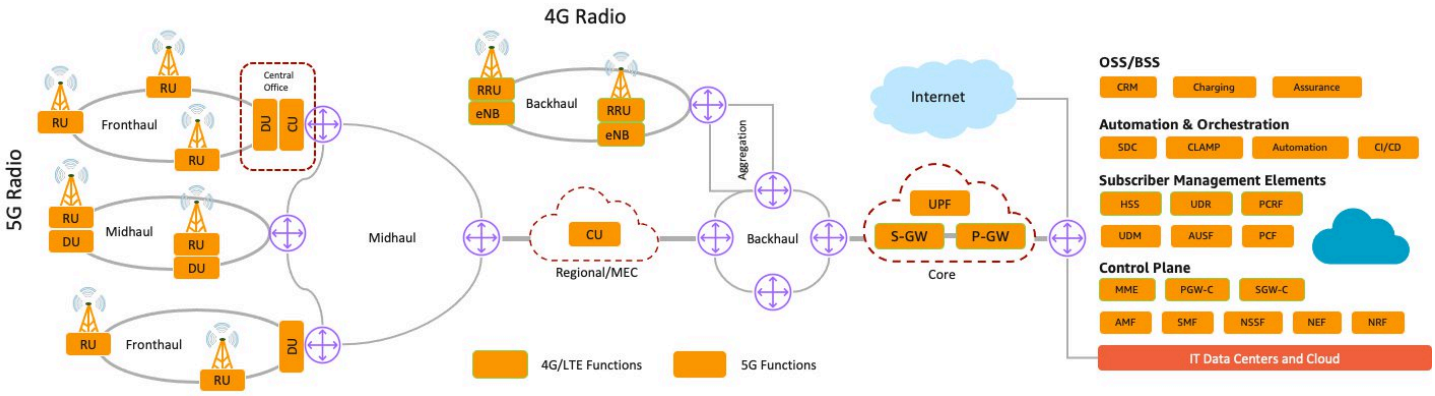
AWS 上の 5G ネットワーク

5G ネットワークインフラストラクチャの典型的なモデルは、4G/5G 無線サイト、フロントホール/ミッドホール/バックホールネットワーク、コアネットワークサイト、通信事業者/IT データセンターで構成されています。CSP は AWS のサービスを使用することで、先行投資のコストを抑えながら、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構築できます。AWS を使用すると、運用支援システム/ビジネス支援システム (OSS/BSS) とコントロールプレーンのコアネット

ワーク機能の大半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) を実装できます。

UPF (ユーザープレーン機能)、RAN 集約基地局 (CU)、マルチアクセスエッジコンピューティング (MEC) などのユーザープレーン機能を主にホストする AWS Outposts インスタンスのフリートを使用して、ローカルのセントラルオフィス (CO) または分散型データセンターを実装する場合にも、AWS を活用できます。AWS での 5G ネットワーク実装のリファレンスアーキテクチャと利点の詳細については、ホワイトペーパー「AWS での 5G ネットワークの進化」を参照してください。

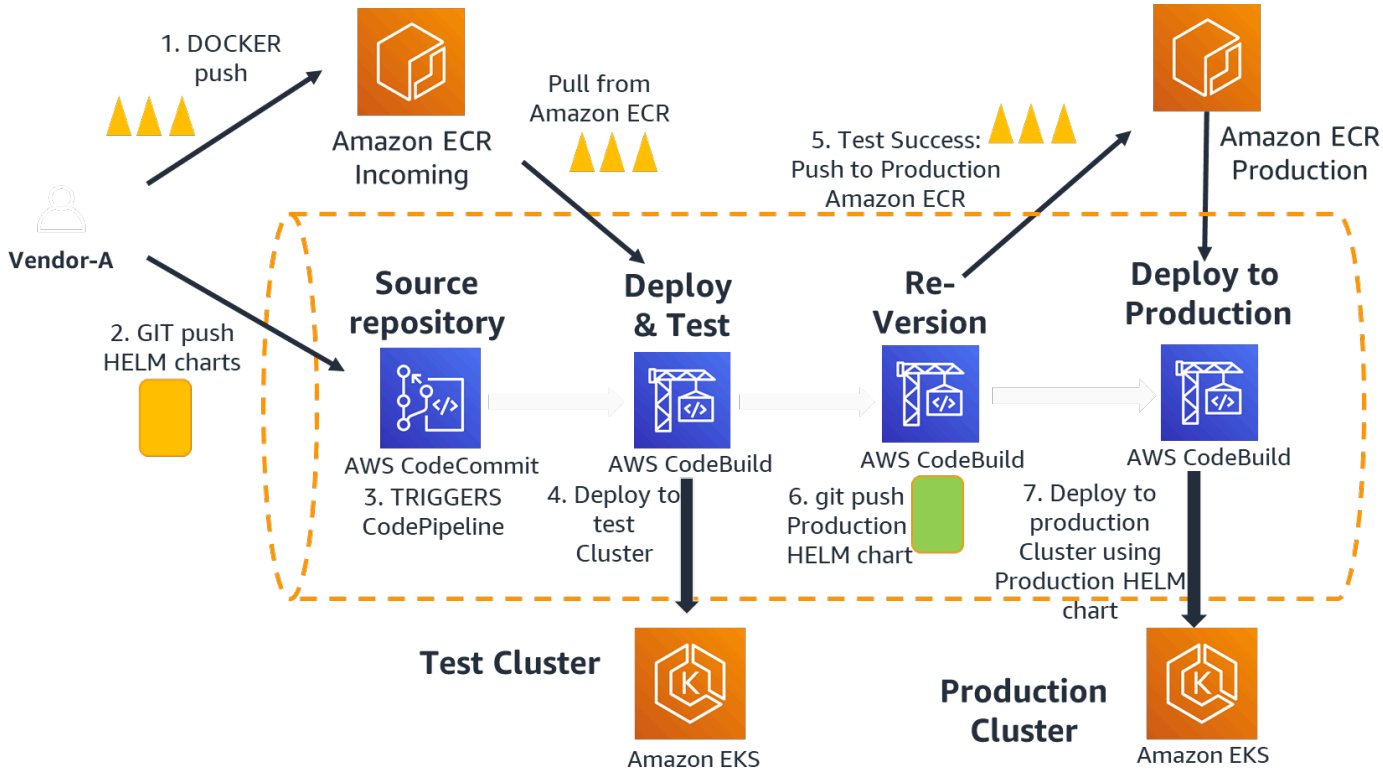
AWS に 5G ネットワークを実装する際には、このホワイトペーパーで紹介する AWS CI/CD ツールにより、5G ネットワーク機能のデプロイ、アップグレード、ライフサイクル管理を完全に自動化できます。



5G ネットワークの E2E アーキテクチャ

5G ネットワークにおける CI/CD

インフラストラクチャの設計構成は、宣言型言語を使用したコード形式で格納されます。これによって CSP は必要に応じ、求められる同じ動作を使用して、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で管理し、デプロイされたスタック (AWS CDK や AWS CloudFormation など) の更新をオーケストレートするようにパイプラインをセットアップします。AWS は、Infrastructure as Code (IaC) の構築を支援し、独立系ソフトウェアベンダー (ISV) 用機能を俊敏にオンボーディングできるよう後押しします。



コードの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なネットワーク機能の設定変更は、ネットワーク機能の自動 CI/CD パイプライン実行のトリガーと見なされます。

設定ファイルの管理には AWS CodeCommit を使用でき、コンテナイメージの保存には Amazon ECR を使用できます。

コードのパイプラインフロー図に示されているように、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へのトリガーが発生します。コードパイプラインは ECR からイメージを取得し、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。この結果に基づいて、CSP は本番環境へのデプロイを承認できます。

CodePipeline のソースステージでは、設定ファイル内の変更を探します。ソースステージに有効なプロバイダーは、CodeCommit、Simple Storage Service (Amazon S3)、GitHub、AWS CloudFormation です。代替ソースシステムを統合するには、Lambda 関数を使用して Webhook を実装します (これにより、Gitlab と AWS CodePipeline との間のイベント駆動型統合が可能になります)。詳細な実装ガイドについては、次のリンクを参照してください。

- Webhooks と GitLab
- 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、初期デプロイ、テストに続き、要件に沿ってテスト結果を調整し、ベースラインに対する検証後に本番環境へ昇格するという、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思決定が可能になります。

アプリケーションの CI/CD パイプラインのステップ

各ステージは個別のタスクと見なし、ネットワークサービスやクラウドネイティブのネットワーク機能をサポートするために適した検証および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やシミュレータなどのサードパーティーツールを追加で組み込むことができるため、エンドツーエンド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できる高度な AWS Step Function (クラウドネイティブステートマシン) サービスを提供しています。これには、Jira やテスト自動化フレームワークなどの外部システムと統合する機能も備わっています。

CI/CD ステップの詳細

CI/CD は、パイプラインとして考えることができます。パイプラインでは、一方の端で新しいコードが引き渡され、一連のステージ (ソース、ビルド、ステージング、本番) でテストされた後、本番環境で使用できるコードとして公開されます。

デプロイとテストのステップは次のとおりです。デプロイと設定は、主に次の 4 つの部分に分かれます。

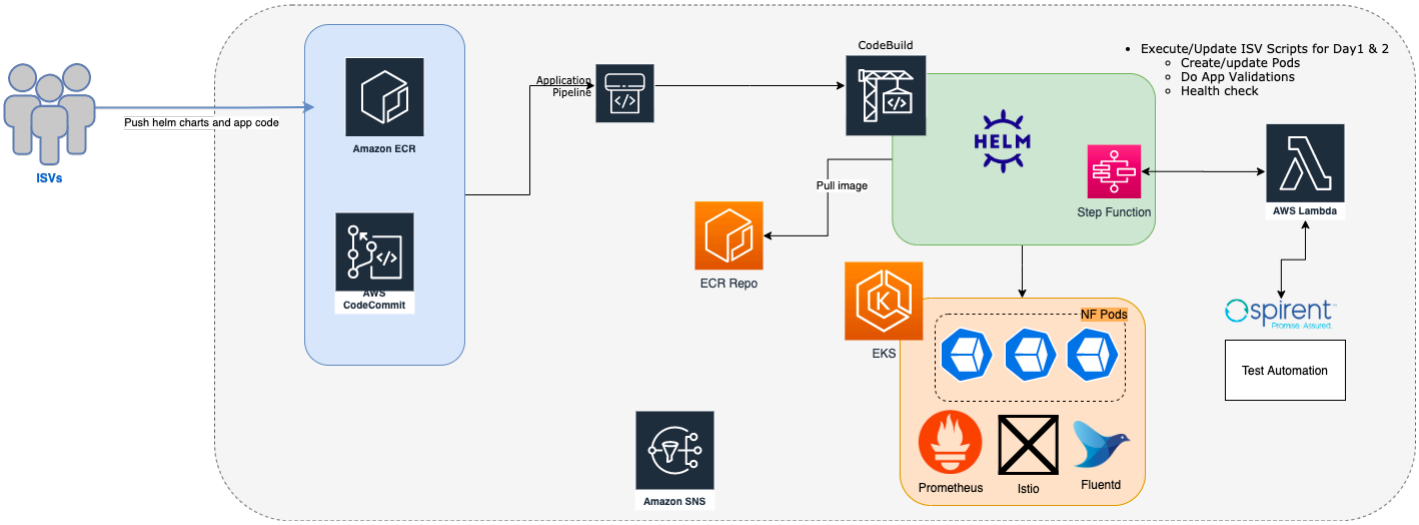
- ネットワークのセットアップ
- インフラストラクチャのデプロイ
- クラウドネイティブなネットワーク機能のデプロイ
- CNF の継続的デリバリー

ネットワークのセットアップ

インフラストラクチャの前提条件の設定に重点を置きます。これには、VPC、ネットワーク、サブネット、NACL などの作成が含まれます。ISV とお客様の実装計画 (マルチテナンシー、静的割り当て/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 AWS CloudFormation でコーディングできます。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のデプロイでは、あらゆるインフラストラクチャコンポーネントのプロビジョニングが行われます。これには、EKS クラスターおよびそれを支えるインフラストラクチャ (EFS、EKS ワーカーノード、ELB など) の生成と、クラウドネイティブのネットワーク機能の要件に従ったクラスターの設定が含まれます。CNF 要件に基づいて、Multus インターフェイスを含む追加のネットワークインターフェイスも AWS によってノードにデプロイされます。デプロイと構成のステップのほとんどは、アプリケーションごとに 1 回限り行う作業です。アプリケーションのアップデートとして必要な場合にのみ、更新が行われます。



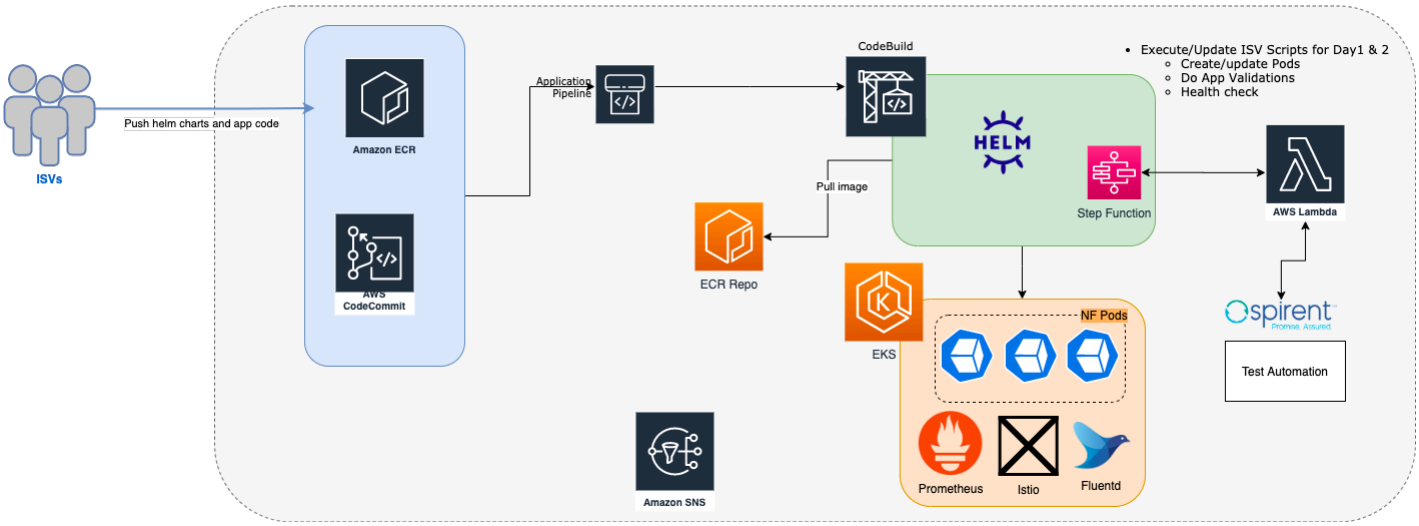
CDK によるインフラストラクチャのデプロイ

クラウドネイティブなネットワーク機能のデプロイ

CNF のデプロイは、アプリケーションのデプロイという形で行います。CNF のデプロイの一環として、CI/CD コードパイプラインを通じてアプリケーションの Helm チャートを実装します。アプリケーション固有のスクリプトを個別に実行するコールバックにより、主に事前チェックと事後チェッ

クが実行されます。Helm チャートはアプリケーションで必要となる順序で実装され、デプロイの次のステップに進む前に Kubernetes POD のステータスを確認します。多くの場合、ISV は Helm チャートとサニティチェックを実行するための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、AWS CodePipeline 内から呼び出されます。このフェーズでは、アプリケーションのクラウドインフラストラクチャのログ記録およびモニタリングを行う Amazon CloudWatch に加えて、Prometheus や Fluentd などのロギングエージェントおよびモニタリングエージェントがデプロイされます。

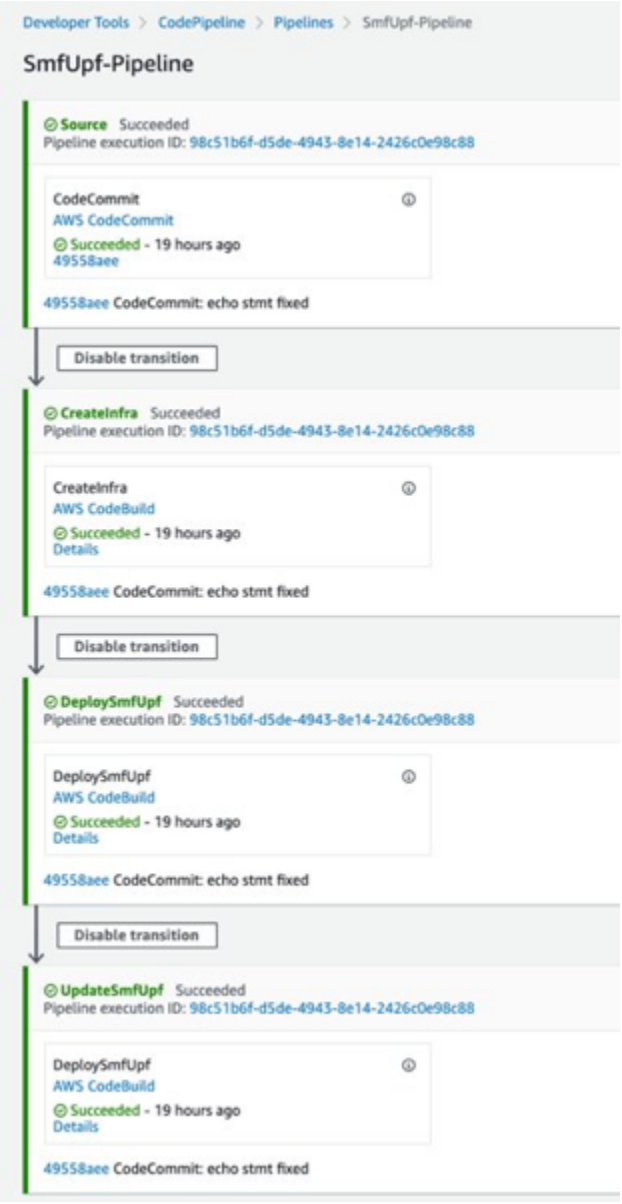
コードパイプラインは、サードパーティーのテスト自動化フレームワークと統合されます。コードパイプラインは、テスト自動化フレームワーク API を直接呼び出して、デプロイされたアプリケーションでのテストの実行、テスト結果の照会、結果の分析を行うことができます。これにより、アプリケーションのデプロイとテストを簡素化できます。



アプリケーションのデプロイと更新

AWS CodePipeline を介し、CNF としてユーザープレーン機能/セッション管理機能 (UPF/SMF) をデプロイする例を次に示します。

- CodeCommit、CodeBuild、および CodePipeline を使用して、完全な CI/CD プロセスを自動化します。
- インフラストラクチャ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をパイプラインの一部として統合します。
- Amazon CloudWatch ダッシュボードに、FluentD および Prometheus のエージェントをインストールして作成します。



UPF/SMF CNF のデプロイ例

CNF の継続的デリバリー

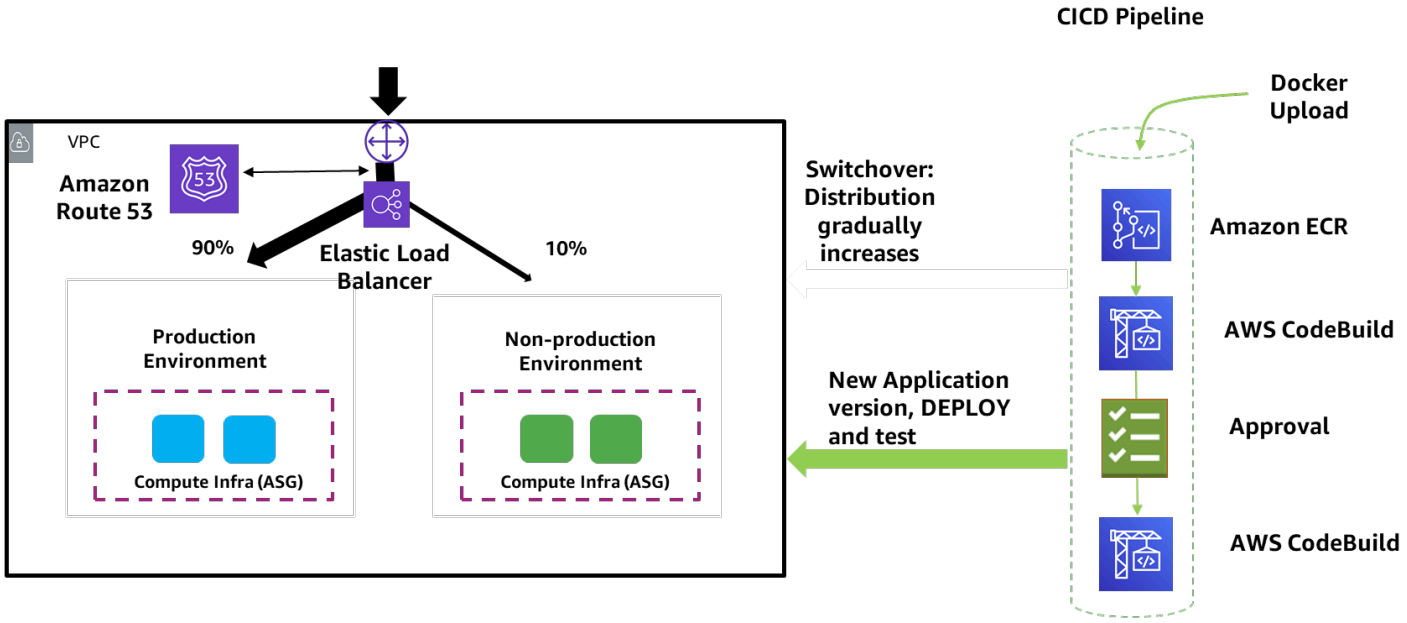
このステップは、コンテナ/設定への変更をデプロイしてアップグレードを行うために反復的に実行する一連の手順で構成されます。CNF の継続的デリバリーは個々のアプリケーションに固有であり、パイプラインによって自動化されます。AWS は標準の Helm チャートを使用して特定の CNF を更新します。コードパイプラインでは、アプリケーションの更新ステータスの事前チェックと事後チェックが行われます。更新された CI/CD パイプラインは、自動テストを実行するためのテスト自動化フレームワークにも統合されています。この抽象化により、ネットワーク機能のクリーンデプロイが可能になります。

CNF の継続的デリバリーとデプロイは、大まかに次のカテゴリに分類できます。

- 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アップグレードは Kuberbetes アプリケーション POD 内の変更です。このような更新は、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 CNF では、アプリケーション POD 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を使用すれば、ローリングアップグレードが可能になります。アプリケーション POD に対するすべての変更で Helm のアップグレードがサポートされるわけではありません。パイプライン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ずに適用することができません。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。このプロセスの実行中には、アプリケーションを使用できなくなる場合もあります。アップグレードには次のツールを使用します。

    - お客様は AWS CloudFormation を使用して、JSON または YAML テンプレートですべてのインフラストラクチャリソースを記述し、プロビジョニングを行うことができます。AWS CloudFormation は、Lambda でサポートされるカスタムリソースを通じて、強力な拡張メカニズムを提供します。お客様は AWSAWS CloudFormation リソースを超えて拡張し、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 を使用すると、デベロッパーが Python、TypeScript、JavaScript、Java、C# などの使い慣れた高レベルのプログラミング言語を使用してコードを構築し、そのコードを低レベルの AWS 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 — AWS では、テスト環境および本番環境における Blue/Green および canary ベースのデプロイをサポートし、推奨しています。Blue/Green デプロイを使用すると、自己完結型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。これにより、本番トラフィックを簡単かつ適切に切り替えることができます。canary ベースのデプロイでは、本番トラフィックによって生じる問題を明らかにするために、本番トラフィックのごく一部を使用して非本番の Green 環境をテスト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、内部で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でテストされるため、ユーザーは確信を得てから本番トラフィックに切り替えることができます。本番トラフィックは、切り替え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で AWS CodePipeline を設定することによって実現できます。承認ステージは、最初はプロビジョニング時に手動で進めることもできますが、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上位互換性と下位互換性を検証するテストを行うことをお勧めします。サービスメッシュを使用するクラスターでの Blue/Green デプロイでは、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を使用して、サービスメッシュでシステムダウンを発生させずに移行を実現します。
- AWS Systems Manager には、CI/CD でデプロイしたネットワーク機能に使用されている複数の AWS サービスの運用データを確認できるように、統一されたユーザーインターフェイスが備えられています。Systems Manager を使用すると、AWS リソース全体で運用タスクを自動化できます。



canary デプロイメント

セキュリティ

セキュリティは重要な要素です。アプリケーションをデプロイする際に AWS の CI/CD プロセスで考慮されるセキュリティステップの一覧を次に示します。

- ソース — ベンダーに割り当てられた ECR リポジトリでは、Docker イメージのアップロードが直ちにセキュリティスキャンの対象となるように、[プッシュ時にスキャン] フラグを有効に設定しま

す。既知の共通脆弱性識別子 (Common Vulnerabilities and Exposures: CVE) があれば通知されます。ECR とは別に、ベンダーが AWS CodeCommit リポジトリにチャートを配置する際には、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ベンダーに要求されます。

- 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、保管時 (AWS マネージドキーを使用) でも送信中 (SSL/TLS を使用) でも暗号化されます。
- IAM ユーザーおよびロール — ユーザーまたはリソースに付与されるアクセス許可は、最小特権の原則に基づいています。異なる複数のサービスのリソースにまたがって運用している場合は、IAM ロール間での信頼関係の設定が必要になることがあります。例えば AWS CodeBuild には、Amazon EKS クラスターでコマンドを実行するためのアクセス許可が必要です。
- 監査 — AWS CloudTrail によって提供される監査機能を使用すると、サービスおよびユーザー操作全体ですべての API コールを追跡し、過去のイベントを評価できます。
- イメージの脆弱性スキャン — Amazon ECR にアップロードされた CNF イメージについては、セキュリティの脆弱性がないか自動的にスキャンが行われます。スキャン結果のレポートは AWS Management Console に用意されており、API 経由で取得することもできます。その後、CNF イメージの置き換えなどの是正措置を受けるために、調査結果を 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ステージで実施され、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ていることを確認します。これにより、CSP に通知を送信して承認を受けることができます。

- コンテナレジストリは、オープンな CVE 脆弱性をスキャンします。
- テストステージでは、情報漏えい、個人を特定できる情報 (PII) の既知のパターンがないかチェックされ、予期せず開いている TCP/UDP ポートや DOS 脆弱性などの問題に対するコンプライアンスチェックルールがトリガーされます。
- アップグレード/ロールバックの安全性について、下位互換性と上位互換性が検証されます。

アプリケーション以外でも、保管時または転送中を問わず、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することで、パイプラインのセキュリティをプロビジョニングすることが重要です。

可観測性

AWS では、AWS にデフォルトでデプロイされている 5G CNF の可観測性が有効になります。これは Amazon CloudWatch によって実現されている機能です。CloudWatch によりは、クラウドリソースとアプリケーションの完全な可視化が可能になります。

このプロセスで、Amazon CloudWatch の処理には次の 4 つの主要なステップがあります。

1. 収集 — AWS とオンプレミスの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、メトリクスとログを収集します。
2. 監視 — CloudWatch ダッシュボードでアプリケーションとインフラストラクチャを視覚化し、トラブルシューティング用にログとメトリクスを並べて関連を示し、CloudWatch Alarms でアラートを設定します。
3. アクション — CloudWatch Events と AWS Auto Scaling を使用して、運用上の変更に対する対応を自動化します。
4. 分析 — CloudWatch Metric Math によるリアルタイム分析、最大 1 秒のメトリクス、データ保持期間の延長 (15 か月) が可能です。

Amazon CloudWatch エージェントが、お客様の Kubernetes クラスターにインストールされます。このエージェントは Prometheus の設定、検出、メトリクスのプル機能をサポートしており、忠実性の高いすべての Prometheus メトリクスおよびメタデータをエンリッチ化し、埋め込みメトリクスフォーマット (Embedded Metric Format: EMF) として CloudWatch Logs に公開します。

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集約される CloudWatch カスタムメトリクスの収集、フィルタリング、作成が自動的に行われます。

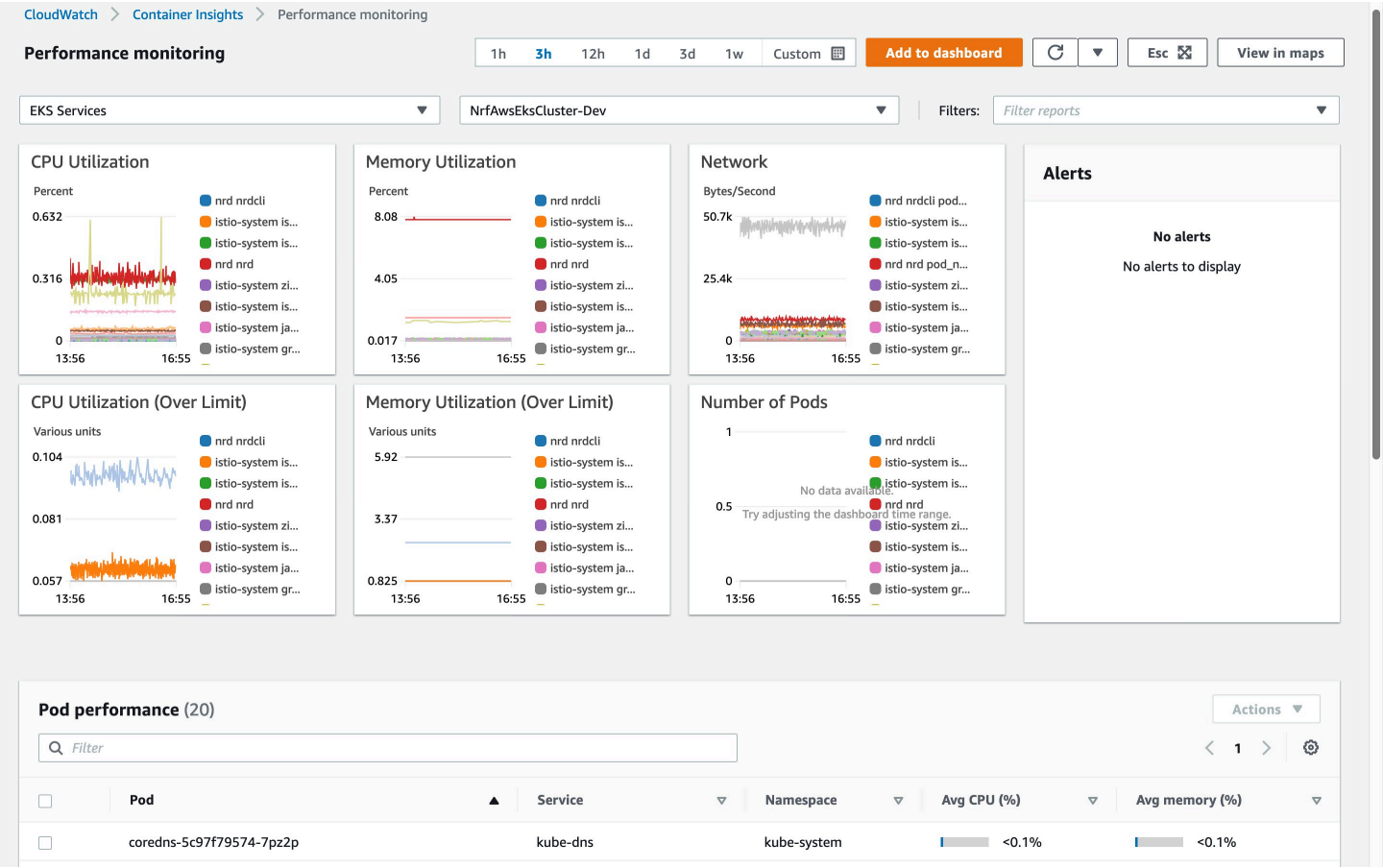
イベントごとに、CloudWatch カスタムメトリクスとしてメトリクスデータポイントが作成され、詳細な設定によってキュレートされる一連のメトリクスディメンションに使用されます。集約された Prometheus メトリクスを CloudWatch カスタムメトリクス統計として公開すると、パフォーマンスの問題や障害のモニタリング、アラーム、トラブルシューティングに必要なメトリクスの数が減ります。また、CloudWatch Logs Insights クエリ言語を使用して忠実性の高い Prometheus メトリクスを分析し、コンテナ化された環境の正常性とパフォーマンスに影響を与える特定のポッドと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、サービス全体ですべての API コールを記録します。AWS Config は、コンプライアンス検証のための機能を提供します。AWS では、AWS X-Ray や AWS

CloudTrail 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、イベントに関する追加のモニタリングオプションをお客様に提供しています。

- AWS では、Prometheus や 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統合で使用できます。
- Prometheus メトリクスを、Amazon CloudWatch または OpenSearch Service に取り込み、さらに詳しく分析することもできます。
- AWS では、さまざまなシステムからログを収集するための標準的なメカニズムとして FluentD を使用しています。このプロジェクトでも同じメカニズムが使用され、設定されています。

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



Amazon CloudWatch で監視されるメトリクスの例

サードパーティー製およびオープンソースのツールを使用した 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では 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、モジュール型かつポータブルで、再利用可能な設計する必要があります。

このインフラストラクチャは、可用性、冗長性、拡張性に優れ、クラウドネイティブのベストプラクティスに従っています。

これまでのセクションで説明したように、基盤となるインフラストラクチャのデプロイは AWS Cloud Development Kit を使用して実現できます。これは、Hashicorp の Terraform を使用することもできます。

Terraform

Terraform は、何百ものクラウドサービスを管理するために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オープンソースの IaC ソフトウェアツールです。Terraform では、クラウド API をコード化して宣言型設定ファイルを作成します。

Terraform でのデプロイには、CDK の場合と同じ原則を使用します。コードはモジュール型構成であり、ベンダーの要件に応じ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フルにカスタマイズできます。

ネットワーク機能のデプロイは、次の 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、セントラル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し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機能をデプロイする準備が整います。

インフラストラクチャデプロイ

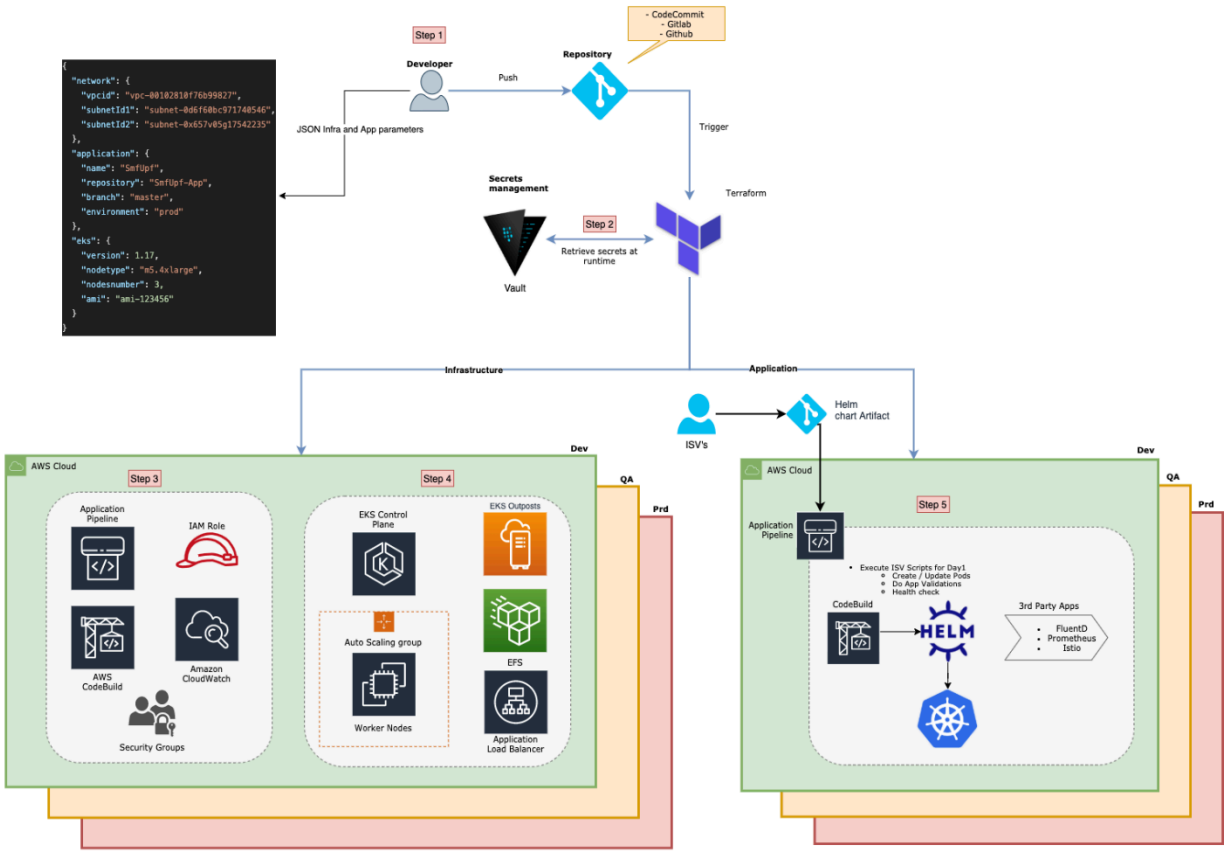
インフラストラクチャデプロイには、ネットワーク機能を正常にデプロイおよび設定するための前提条件がすべて含まれています。

このフェーズでは、次のようなコンポーネントが作成されます。

- ネットワーク — VPC、パブリック/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 (プライマリ/ワーカーノード)、Auto Scaling グループ
- ストレージ — Amazon EFS、Amazon EBS、Simple Storage Service (Amazon S3) バケット
- セキュリティ — IAM ロール、セキュリティグループ
- パイプライン — CodePipeline、CodeBuild
- 可観測性 — CloudWatch、Prometheus、FluentD

Terraform によってオーケストレートされるインフラストラクチャシーケンスを以下に示します。下の図を参照してください。

1. デベロッパーは、セントラルリポジトリに保存されている JSON ファイルに IaC コードを記述します。このファイルには、インスタンスのサイズ、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 Vault 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、設定します。
4. ネットワーク機能ポッドをホストするワーカーノードを含めて Amazon EKS クラスターがデプロイされます。Amazon EKS を AWS Outposts にデプロイして、データセンターへの近接性を必要とするワークロードをサポート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内の変更をリッスンするように設定されます。設定済みのリポジトリブランチにコードがプッシュされるたびに、パイプラインによってネットワーク機能のビルド、テスト、デプロイが自動的にトリガーされます。
6. ログとメトリクスを収集して一元化する可観測性ツールは、サービスとしてすべてのノードにデプロイされ、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。



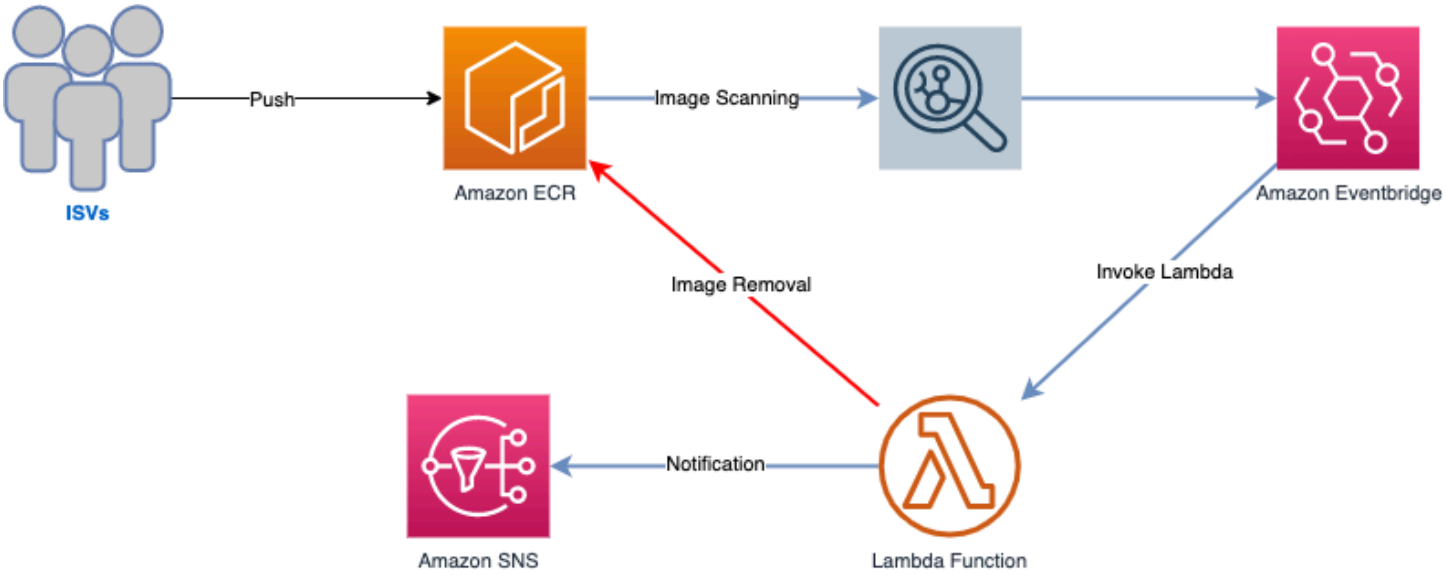
ネットワーク機能のデプロイと設定

ネットワーク機能のデプロイと設定

前のステージで作成されたパイプラインにより、ISV とプロバイダーはネットワーク機能のデプロイを分散して最適化できます。前の図のステップ 1 で JSON ファイル内で設定されたアプリケーションリポジトリにパイプラインが接続され、このリポジトリへの変更がリッスンされます。

サードパーティーによって公開されたイメージを精査するために、コンテナイメージ内のソフトウェアの脆弱性の特定を支援する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では、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調べます。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、イメージ脆弱性スキャンソリューションのアーキテクチャを示しています。



イメージ脆弱性スキャン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、スキャン結果後にイメージが変更された場合や、リポジトリ内で直接行われた変更によってトリガーされるように設定できます。例えば、新しい Helm イメージが作成されたときなどです。

ネットワーク機能を作成/アップグレードするシーケンスを以下に示します。下の図を参照してください。

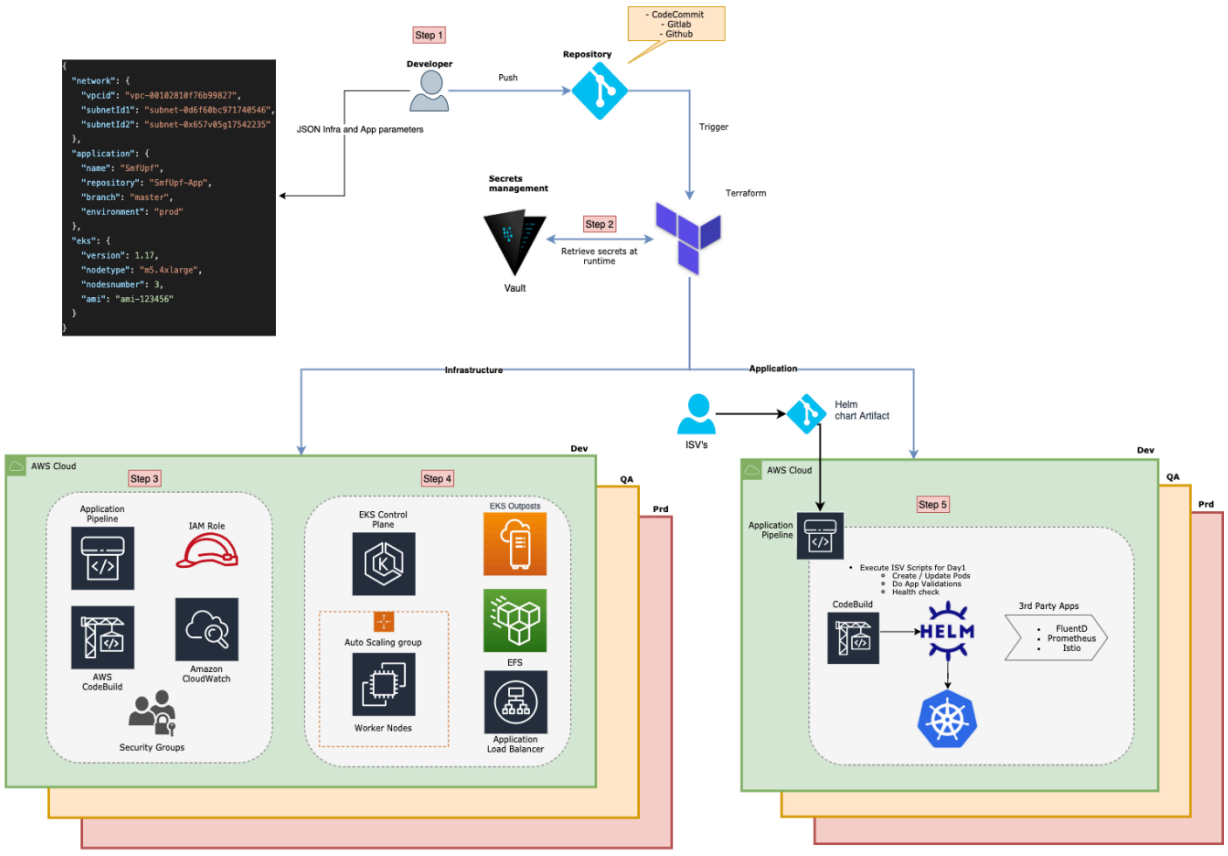
ISV が新しいイメージを Amazon ECR に発行します。イメージが承認されると、アプリケーションパイプラインがトリガーされます。

CodePipeline が 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。ネットワーク機能をアップグレードするには、Helm コマンドを使用できます。

イメージがデプロイされると、Test as Service (Tas) がトリガーされます。TaS が新しいデプロイを検証し、負荷がかかっているネットワーク機能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。追加の可観測性を提供できるように、Datadog、Istio、Prometheus などのサードパーティー製品を設定することもできます。

ネットワークリソースを調整できる MANO をデプロイし、ソリューションと統合することもできます。収集したデータは、ネットワークスライシングや Quality of Service (QoS) オートスケーリングなどの自動化されたアクションの実行に使用されます。



アプリケーションパイプライン

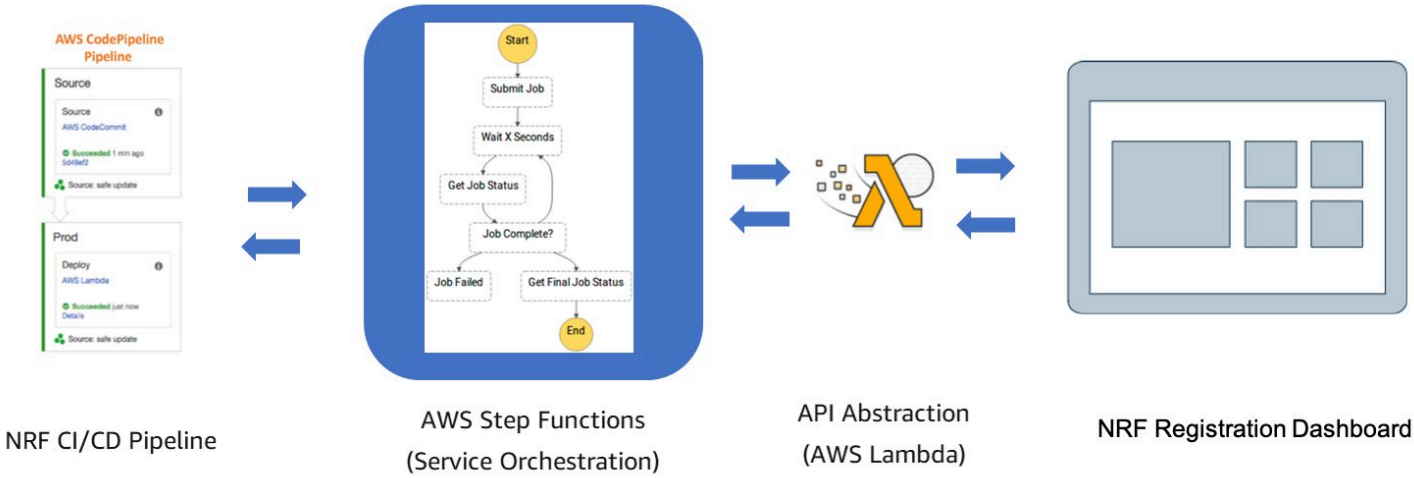
テスト

通信事業者固有のテスト自動化フレームワークをコードパイプラインに統合することもできます。コードパイプラインは、テスト自動化フレームワークとの統合をオーケストレートする Step Functions と統合されます。AWS Step Functions は複数の Lambda 関数を使用し、API コールを介してテスト自動化フレームワーク (サードパーティーツール) を呼び出します。Step Functions はまずテスト ID を取得し、テストを実行して、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 Step Functions は、本番環境にデプロイするアプリケーションの承認のために、結果を分析してコードパイプラインに渡します。承認は自動化することも、必要に応じてコードパイプラインで手動管理することもできます。これは、CSP がテスト環境から本番環境にデプロイを進めるうえで重要なステップです。統合に必要な高レベルの API 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の実行
- テストケースの停止

- 結果の取得

外部の REST API の呼び出しの複雑さは、AWS Step Functions を使用してモデル化できます。これを使用すると、標準コンストラクトで並列フローの呼び出し、結果の待機、条件に基づく分岐、および REST API と AWS CodePipeline との統合が可能になります。



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に不随するオートメーションの考え方と、それを 5G に適用する方法に含まれる手法です。オーケストレーションは、この考え方のもう 1 つの側面であり、ネットワーク上で発生するあらゆる変更に対して動的ですばやい対応が必要になります。サービスの安定性を保証し、サービスの中断を最小限に抑えるには、オーケストレーションと CI/CD の緊密な連携が必要になります。CI/CD とオーケストレーションの統合では、次の 2 つの点に注意します。

- システムへのパッチとアップグレードの適用は、稼働中のサービスの中断を最小限に抑えながら、管理およびオーケストレーションを行う必要があります。例えば、オーケストレーションによって、アップデート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 対応のオーケストレーションを使用すると、使用するデプロイモデル戦略 (canary、リニア、1 回にすべて) 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。

一般的な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によってパイプライン内にガバナンスフェーズが作られ、進行中のアップグレードサイクルへの接触が生じます。

まとめ

CI/CD は、デベロッパーやアプリケーションチームが新しいアプリケーションコードを数分でデプロイするための明確で効率的な手段を提供します。AWS には、AWS CodePipeline、AWS CodeCommit、AWS CodeBuild、AWS CodeDeploy など、デベロッパーが新しいコードの統合、テスト、デプロイを行う際に役立つ豊富なツールがあります。このドキュメントでは、AWS のサービスを使用して、完全に自動化された方法で 5G ネットワーク機能を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。これには、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も含まれていました。また、サードパーティー製のテストオートメーションフレームワークを CI/CD プロセスに統合する方法や、Terraform などのサードパーティー製ツールを使用する方法についても考察しました。

寄稿者

本書の作成における寄稿者

- 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
- アマゾン ウェブ サービス、AWS Telecom、プリンシパルコンサルタント、Vara Prasad Talari
- アマゾン ウェブ サービス、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、共有デリバリー、シニアコンサルタント、Franco Bontorin
- アマゾン ウェブ サービス、共有デリバリー、コンサルタント、Pragtideep Singh
- アマゾン ウェブ サービス、グローバルアカウント、クラウドインフラアーキテクト、Subbarao Duggisetty
- アマゾン ウェブ サービス、AWS Telecom、シニアパートナーソリューションアーキテクト、Young Jung

改訂履歴

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 フィードをサブスクライブしてください。

update-history-change	update-history-description	update-history-date
初版公開	ホワイトペーパーの初版公開	2021 年 3 月 8 日

詳細情報

詳細については、次の資料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の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 での 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 & モビリティ管理機能 (Access & Mobility Management Function)
- API — アプリケーションプログラミングインターフェイス (Application Programming Interface)
- AUSF — 認証サーバー機能 (Authentication Server Function)
- BSS — ビジネス支援システム (Business Support System)
- CDK — クラウド開発キット (Cloud Development Kit)
- CI/CD — 継続的インテグレーションと継続的デリバリー (Continuous Integration and Continuous Delivery)
- CLI — コマンドラインインターフェイス (Command Line Interface)
- 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機能 (Cloud-native or Containerized Network Function)
- CSP — 通信サービスプロバイダー (Communication Service Provider)
- CU — RAN 集約基地局 (RAN Central Unit)
- CVE — 共通脆弱性識別子 (Common Vulnerabilities and Exposures)
- DoS — サービス拒否 (Denial of Service)
- DR — 災害対策 (disaster recovery)
- DU — RAN リモート局 (RAN Distributed Unit)
- E2E — エンドツーエンド (end to end)
- ECR — Elastic Container Registry
- EFS — Elastic File System
- EKS — Elastic Kubernetes Service
- EPC — 進化したパケットコア (Evolved Packet Core)
- IaC — コードとしてのインフラストラクチャ (Infrastructure as Code)
- ISV — 独立系ソフトウェアベンダー (Independent Software Vendor)
- MANO — 管理およびオーケストレーション (Management and Orchestration)
- MEC — マルチアクセスエッジコンピューティング (Multi-Access Edge Computing)
- NACL — ネットワークアクセスコントロールリスト (Network Access Control List)
- NAT — ネットワークアドレス変換 (Network Address Translation)
- NF — ネットワーク機能 (Network Function)

- NFV — ネットワーク機能仮想化 (Network Function Virtualization)
- NFVO — ネットワーク機能仮想化オーケストレーター (Network Function Virtualization Orchestrator)
- NOC — ネットワークオペレーションセンター (Network Operations Centre)
- NRF — ネットワークリポジトリ機能 (Network Repository Function)
- OSS — 運用支援システム (Operations Support System)
- PII — 個人識別情報 (Personally Identifiable Information)
- QoS — サービス品質 (Quality of Service)
- RAN — 無線アクセスネットワーク (Radio Access Network)
- SBI — サービスベースインターフェイス (Service Based Interface)
- SMF — セッション管理機能 (Session Management Function)
- SSL — セキュアソケットレイヤー (Secure Sockets Layer)
- TA — サービスとしてのテスト (Test as Service)
- TCP — 伝送制御プロトコル (Transmission Control Protocol)
- TLS — トランスポートレイヤーセキュリティ (Transport Layer Security)
- UDM — 統合データ管理 (Unified Data Management)
- UDP — ユーザデータグラムプロトコル (User Datagram Protocol)
- UPF — ユーザープレーン機能 (User Plane Function)
- VIM — 仮想化インフラストラクチャマネージャー (Virtualized Infrastructure Manager)
- VNF — 仮想ネットワーク機能 (Virtual Network Function)
- VPC — 仮想プライベートクラウド (Virtual Private Cloud)

注意

お客様は、この文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ます。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、明示または暗示を問わず、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 契約により規定されます。本書は、AWS とお客様の間で締結されるいかなる契約の一部でもなく、その内容を修正するものでもありません。

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