ユーザーガイド

AWS 通信ネットワークビルダー

AWS 通信ネットワークビルダー: ユーザーガイド

Table of Contents

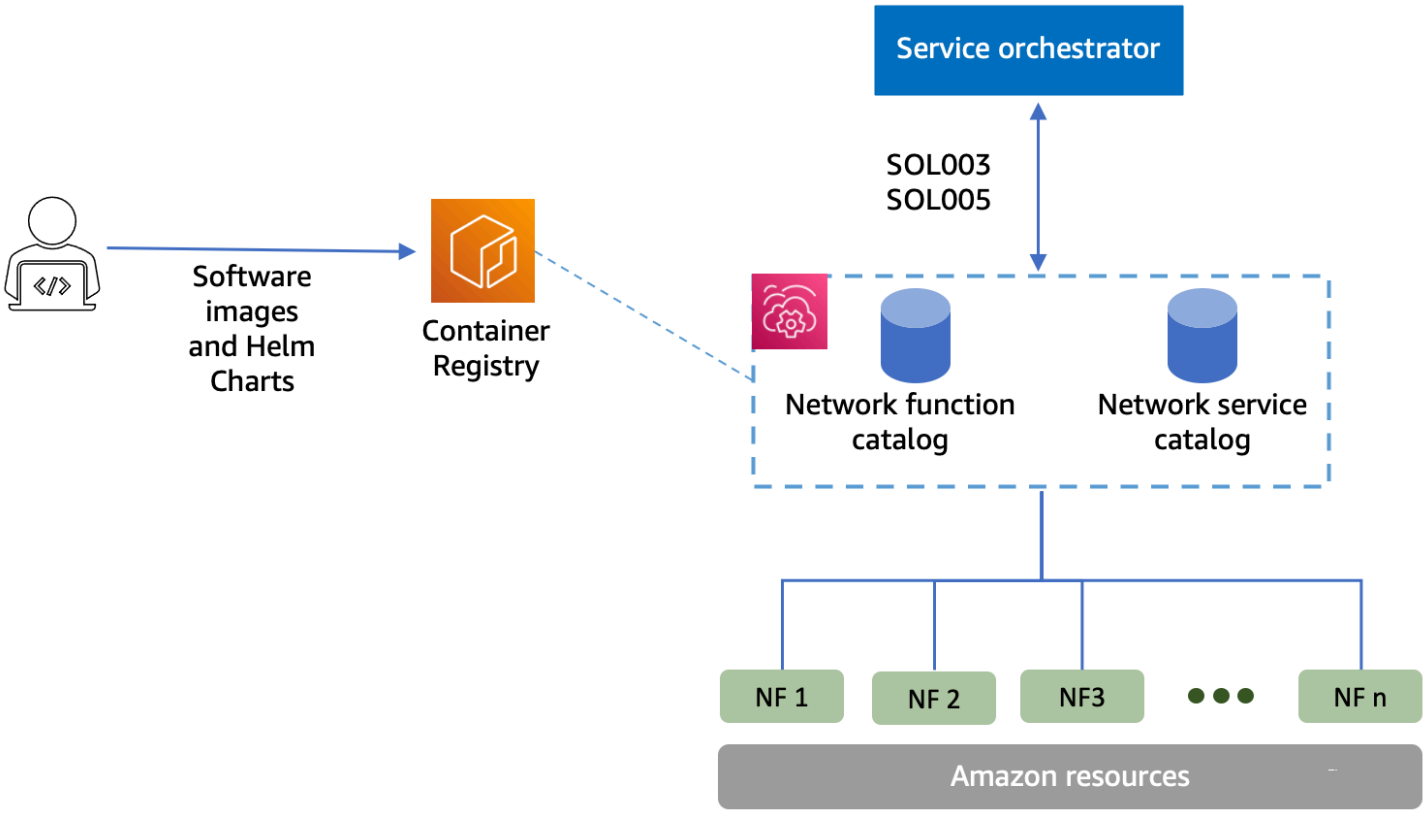
AWS Telco Network Builder とは

AWS Telco Network Builder (AWS TNB) は、通信サービスプロバイダー (CSPs) に 5G ネットワークをインフラストラクチャに AWS デプロイ、管理、スケーリングするための効率的な方法を提供する AWS サービスです。

AWS TNB では、ネットワークのイメージ AWS クラウド を使用して、スケーラブルで安全な 5G ネットワークを に自動でデプロイします。新しいテクノロジーを学習したり、使用するコンピューティングサービスを決定したり、 AWS リソースをプロビジョニングして設定する方法を知っている必要はありません。

代わりに、ネットワークのインフラストラクチャを記述し、独立系ソフトウェアベンダー (ISV) パートナーからネットワーク機能のソフトウェアイメージを提供します。 AWS TNB はサードパーティーのサービスオーケストレーターおよび AWS サービスと統合して、必要な AWS インフラストラクチャを自動的にプロビジョニングし、コンテナ化されたネットワーク機能をデプロイし、ネットワークとアクセス管理を設定して、完全に運用可能なネットワークサービスを作成します。

次の図は、欧州電気通信標準協会 (ETSI) ベースの標準インターフェイスを使用してネットワーク機能をデプロイするための AWS TNB とサービスオーケストレーターの論理統合を示しています。



トピック

初めて AWSですか？

AWS 製品やサービスを初めて使用する場合は、次のリソースから詳細を学んでください。

- の概要 AWS
- の開始方法 AWS

AWS TNB とは

AWS TNB は、コスト効率を活用しようとしている CSPs 向けです。 俊敏性、 設計するカスタムスクリプトと設定を記述して維持することなく、 AWS クラウド と の伸縮性を提供。 デプロイ、 ネットワークサービスの管理と管理を行います。 AWS TNB は必要な AWS インフラストラクチャを自動的にプロビジョニングします。 はコンテナ化されたネットワーク関数をデプロイします。 CSP で定義されたネットワークサービス記述子に基づいて完全に運用可能なネットワークサービスを作成するようにネットワークとアクセス管理を設定します。 および CSP がデプロイするネットワーク関数。

AWS TNB の機能

以下は、CSP が AWS TNB を使用する理由の一部です。

タスクの簡素化の支援

新しいサービスのデプロイ、ネットワーク機能の更新とアップグレード、ネットワークインフラストラクチャのトポロジーの変更など、ネットワークオペレーションの効率を高めます。

オーケストレーターとの統合

AWS TNB は、ETSI 準拠の一般的なサードパーティーサービスオーケストレーターと統合されています。

スケーリング

基盤となる AWS リソースをトラフィックの需要に合わせてスケーリングし、ネットワーク機能の更新をより効率的に実行し、ネットワークインフラストラクチャトポロジの変更をロールアウトし、新しい 5G サービスのデプロイ時間を数日から数時間に短縮するように AWS TNB を設定できます。

AWS リソースを検査およびモニタリングします

AWS TNB を使用すると、Amazon VPC、Amazon EC2、Amazon EKS など、単一のダッシュボードでネットワークをサポートする AWS リソースを検査およびモニタリングできます。

サービステンプレートのサポート

AWS TNB では、すべての通信ワークロード (RAN、Core、IMS) のサービステンプレートを作成できます。新しいサービス定義を作成したり、既存のテンプレートを再利用したり、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) パイプラインと統合して新しい定義を公開したりできます。

ネットワークデプロイへの変更の追跡

Amazon EC2 インスタンスタイプのインスタンスタイプを変更するなど、ネットワーク機能デプロイの基本的な設定を変更する場合、反復可能かつスケーラブルな方法でその変更を追跡できます。これを手動で行う場合は、ネットワークの状態の管理、リソースの作成および削除、必要な変更の順序についての注意が必要となります。 AWS TNB を使用してネットワーク関数のライフサイクルを管理する場合、ネットワーク関数を説明するネットワークサービス記述子のみを変更します。 AWS TNB は、必要な変更を正しい順序で自動的に行います。

ネットワーク機能のライフサイクルの簡素化

ネットワーク機能の最初のバージョンとそれ以降のすべてのバージョンを管理し、アップグレードのタイミングを指定できます。RAN、Core、IMS、ネットワークアプリケーションも同じ方法で管理できます。

AWS TNB へのアクセス

次のいずれかのインターフェイスを使用して、 AWS TNB リソースを作成、アクセス、管理できます。

- AWS TNB コンソール — ネットワークを管理するためのウェブインターフェイスを提供します。
- AWS TNB API — AWS TNB アクションを実行するための RESTful API を提供します。詳細については、「AWS TNB API リファレンス」を参照してください

- AWS Command Line Interface （ AWS CLI ） — AWS TNB を含む幅広い AWS サービスのコマンドを提供します。Windows、macOS、Linux でサポートされています。詳細については、AWS Command Line Interface ユーザーガイドをご参照ください。
- AWS SDKs – 言語固有の APIs を提供し、接続の詳細の多くを完了します。これらには、署名の計算、リクエストの再試行処理、エラー処理などを含みます。詳細については、AWS SDK を参照してください。

AWS TNB の料金

AWS TNB はCSPs が での通信ネットワークのデプロイと管理を自動化するのに役立ちます AWS。AWS TNB を使用する場合、次の 2 つのディメンションに対して料金が発生します。

- 管理対象ネットワーク機能項目 (MNFI) の時間。
- API リクエストの数。

また、他の AWS サービスを AWS TNB と組み合わせて使用すると、追加料金が発生します。詳細については、「AWS TNB の料金」を参照してください。

請求を表示するには、[AWS Billing and Cost Management console] (コンソール) の [Billing and Cost Management Dashboard] (請求およびコスト管理ダッシュボード) に移動します。請求書には、料金の明細が記載された使用状況レポートへのリンクが記載されています。 AWS アカウント請求の詳細については、AWS 「 アカウント請求」を参照してください。

AWS 請求、アカウント、イベントに関するご質問は、 AWS サポートにお問い合わせください。

AWS Trusted Advisor は、 AWS 環境のコスト、セキュリティ、パフォーマンスを最適化するために使用できるサービスです。詳細については、「AWS Trusted Advisor」を参照してください。

次のステップ

AWS TNB の使用を開始する方法の詳細については、以下のトピックを参照してください。

- AWS TNB のセットアップ – 前提条件の手順を完了します。
- AWS TNB の開始方法 – 集中型ユニット (CU)、アクセスおよびモビリティ管理機能 (AMF)、ユーザープレーン機能 (UPF)、または完全な 5G コアなど、最初のネットワーク機能をデプロイします。

AWS TNB の仕組み

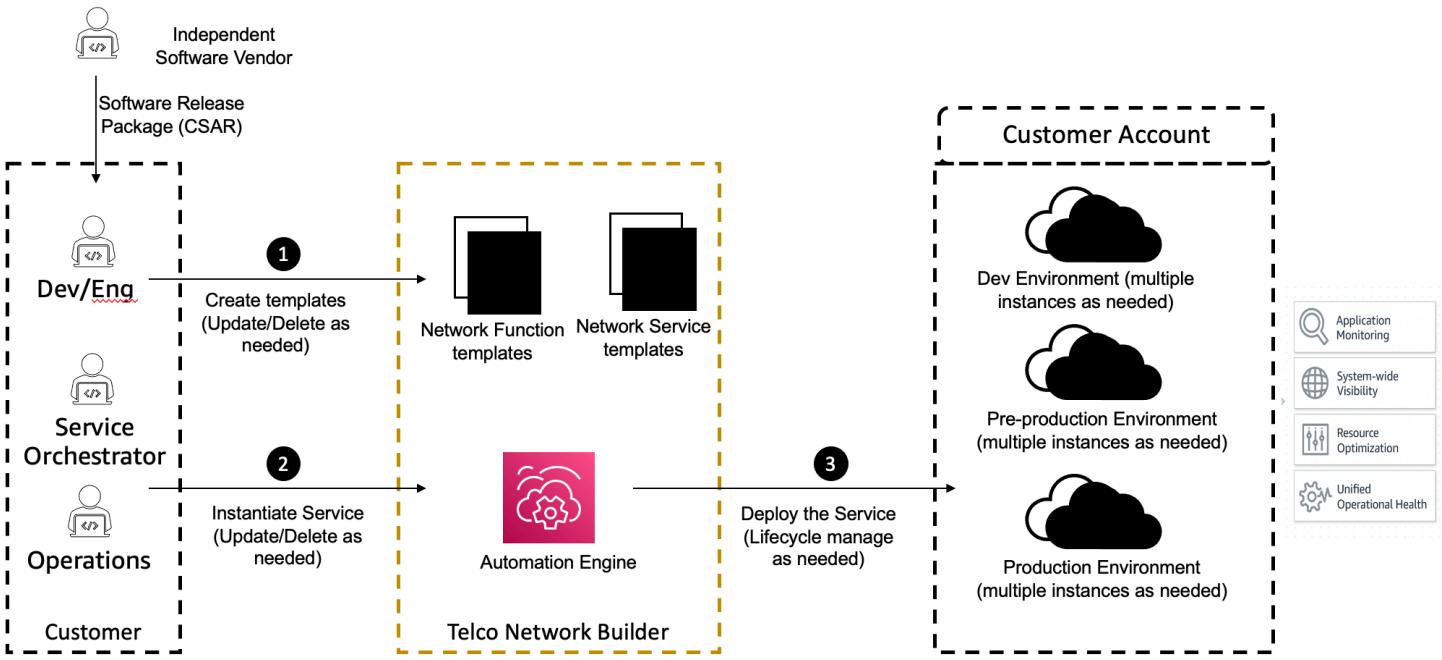
AWS TNB は、標準化されたend-to-endのオーケストレーターと AWS リソースと統合され、完全な 5G ネットワークを運用します。

AWS TNB を使用すると、ネットワーク関数パッケージとネットワークサービス記述子 (NSDs) を取り込むことができ、ネットワークを運用するための自動化エンジンが提供されます。end-to-endのオーケストレーターを使用して AWS TNB APIs と統合することも、 AWS TNB SDKs を使用して独自の自動化フローを構築することもできます。詳細については、「AWS TNB アーキテクチャ」を参照してください。

トピック

AWS TNB アーキテクチャ

AWS TNB では、 AWS マネジメントコンソール、 AWS CLI、 AWS TNB REST API、 SDKs を使用してライフサイクル管理オペレーションを実行できます。これにより、エンジニアリングチーム、オペレーションチーム、プログラマティックシステムチームのメンバーなど、さまざまな CSP ペルソナが AWS TNB を活用できるようになります。ネットワーク機能パッケージを Cloud Service Archive (CSAR) ファイルとして作成してアップロードします。CSAR ファイルには、Helm チャート、ソフトウェアイメージ、ネットワーク機能記述子 (NFD) が含まれています。テンプレートを使用して、そのパッケージの複数の設定を繰り返しデプロイできます。デプロイするインフラストラクチャとネットワーク機能を定義するネットワークサービステンプレートを作成します。パラメータオーバーライドを使用して、さまざまな設定を異なる場所にデプロイできます。その後、 テンプレートを使用してネットワークをインスタンス化し、ネットワーク機能を AWS インフラストラクチャにデプロイできます。 AWS TNB は、デプロイの可視性を提供します。



との統合 AWS のサービス

5G ネットワークは、数千の Kubernetes クラスターにデプロイされた相互接続されたコンテナ化されたネットワーク機能のセットで構成されています。 AWS TNB は、通信固有の APIs として以下 AWS のサービス と統合して、完全に運用可能なネットワークサービスを作成します。

- 独立系ソフトウェアベンダー (ISV) のネットワーク機能アーティファクトを保存する Amazon Elastic Container Registry (Amazon ECR)。
- Amazon Elastic Kubernetes Service クラスターをセットアップするための (Amazon EKS)。
- ネットワーキングコンストラクト用の Amazon VPC。
- を使用するセキュリティグループ CloudFormation。
- AWS CodePipeline 、 AWS ローカルゾーン AWS リージョン、および にまたがるデプロイターゲットの AWS Outposts。
- ロールを定義する IAM。
- AWS Organizations AWS TNB APIs。
- Health Dashboard および AWS CloudTrail を使用して、ヘルスをモニタリングし、メトリクスを投稿します。

AWS TNB リソースクォータ

AWS アカウント には、各 の制限と呼ばれるデフォルトのクォータがあります AWS のサービス。特に明記されていない限り、各クォータは に固有です AWS リージョン。一部のクォータについては引き上げをリクエストできますが、一部のクォータについてはリクエストできません。

AWS TNB のクォータを表示するには、Service Quotas コンソールを開きます。ナビゲーションペインで、[AWS のサービス] を選択し、次に [AWS TNB] を選択します。

クォータの引き上げをリクエストするには、「Service Quotas ユーザーガイド」の「クォータ引き上げリクエスト を参照してください。

AWS アカウント には AWS TNB に関連する次のクォータがあります。

リソースクォータ	説明	デフォルトの値	引き上げ可能?
ネットワークサービスインスタンス	1 つのリージョンのネットワークサービスインスタンスの最大数。	800	はい
継続的なネットワークサービスの同時オペレーション	1 つのリージョンで同時に進行中のネットワークサービスオペレーションの最大数。	40	はい
ネットワークパッケージ	1 つのリージョンのネットワークパッケージの最大数。	40	はい
関数パッケージ	1 つのリージョンの関数パッケージの最大数。	200	はい

AWS TNB の概念

このトピックでは、 AWS TNB の使用を開始するのに役立つ重要な概念について説明します。

内容

ネットワーク機能のライフサイクル

AWS TNB は、ネットワーク機能のライフサイクル全体を通じて役立ちます。ネットワーク機能のライフサイクルには、次の段階とアクティビティが含まれます。

計画

1. デプロイするネットワーク機能を特定してネットワークを計画します。
2. ネットワーク機能ソフトウェアイメージをコンテナイメージリポジトリに配置します。
3. CSAR パッケージを作成してデプロイまたはアップグレードします。
4. AWS TNB を使用して、ネットワーク関数を定義する CSAR パッケージ (CU AMF、UPF など) をアップロードし、CSAR パッケージの新しいバージョンを作成するのに役立つ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) パイプラインと統合します。新しいネットワーク関数ソフトウェアイメージまたはカスタマースクリプトが利用可能になります。

設定

1. コンピューティングタイプ、ネットワーク機能バージョン、IP 情報、リソース名など、デプロイに必要な情報を特定します。
2. この情報を使用してネットワークサービス記述子 (NSD) を作成します。
3. ネットワーク機能を定義する NSD と、ネットワーク機能のインスタンス化に必要なリソースを取り込みます。

インスタンス化

1. ネットワーク機能に必要なインフラストラクチャを作成します。

2. NSD で定義されているとおりにネットワーク機能をインスタンス化 (またはプロビジョニング) し、トラフィックの伝送を開始します。
3. アセットを検証します。

本番稼働

ネットワーク機能のライフサイクル中に、次のような生産オペレーションを完了します。

- ネットワーク機能の設定を更新します。例えば、デプロイされたネットワーク機能の値を更新します。
- 新しいネットワークパッケージとパラメータ値を使用してネットワークインスタンスを更新します。たとえば、ネットワークパッケージの Amazon EKS `version`パラメータを更新します。

標準化されたインターフェースの使用

AWS TNB は、欧州電気通信規格協会 (ETSI) 準拠のサービスオーケストレーターと統合されているため、ネットワークサービスのデプロイを簡素化できます。サービスオーケストレーターは AWS TNB SDKs、 CLI、または APIs を使用して、ネットワーク関数のインスタンス化や新しいバージョンへのアップグレードなどのオペレーションを開始できます。

AWS TNB は、次の仕様をサポートしています。

の仕様	リリース	説明
ETSI SOL001	v3.6.1	TOSCA ベースのネットワーク機能記述子を許可するための標準を定義します。
ETSI SOL002	v3.6.1	ネットワーク機能管理に関するモデルを定義します。
ETSI SOL003	v3.6.1	ネットワーク機能ライフサイクル管理の標準を定義します。
ETSI SOL004	v3.6.1	ネットワーク機能パッケージの CSAR 標準を定義します。

の仕様	リリース	説明
ETSI SOL005	v3.6.1	ネットワークサービスパッケージとネットワークサービスライフサイクル管理の標準を定義します。
ETSI SOL007	v3.5.1	TOSCA ベースのネットワークサービス記述子を許可するための標準を定義します。

関数パッケージ

AWS TNB を使用すると、ETSI SOL001/SOL004 に準拠する関数パッケージを関数カタログに保存できます。次に、仮想ネットワーク関数を説明するアーティファクトを含む Cloud Service Archive (CSAR) パッケージをアップロードできます。

- 仮想ネットワーク関数記述子 – パッケージオンボーディングと仮想ネットワーク関数管理のメタデータを定義します。このファイル `vnfd.yaml` には、名前を付ける必要があります。
- ソフトウェアイメージ – 仮想ネットワーク関数のコンテナイメージを参照します。Amazon Elastic Container Registry (Amazon ECR) は、仮想ネットワーク関数イメージリポジトリとして機能します。
- 追加のファイル – スクリプトや Helm チャートなど、仮想ネットワーク関数の管理に使用します。

CSAR は OASIS TOSCA 標準で定義されたパッケージであり、OASIS TOSCA YAML 仕様に準拠したネットワーク/サービス記述子が含まれています。必要な YAML 仕様の詳細については、「」を参照してくださいAWS TNB の TOSCA リファレンス。

仮想ネットワーク関数記述子の例を次に示します。

```
tosca_definitions_version: tnb_simple_yaml_1_0

topology_template:

  node_templates:

    SampleNF:
      type: tosca.nodes.AWS.VNF
      properties:
        descriptor_id: "SampleNF-descriptor-id"
```

```
        descriptor_version: "2.0.0"
        descriptor_name: "NF 1.0.0"
        provider: "SampleNF"
      requirements:
        helm: HelmChart

  HelmChart:
    type: tosca.nodes.AWS.Artifacts.Helm
    properties:
      implementation: "./SampleNF"
```

ネットワークパッケージ

ネットワークパッケージは、CSAR (Cloud Service Archive) 形式の.zipファイルです。デプロイする関数パッケージと、デプロイする AWS インフラストラクチャを定義します。

ネットワークパッケージには、次のファイルが含まれています。

- ETSI SOL007 で説明されている TOSCA 形式のネットワーク記述子ファイル (nsd.yaml)。

  nsd.yaml ファイルには、アップロードされた関数パッケージへの参照と記述子 IDsが含まれています。
- ユーザーデータスクリプトがある場合。
- ライフサイクルフックスクリプトがある場合。
- プラグインvalues.yamlの設定ファイルがある場合。

AWS TNB は、ネットワーク、サービス、関数などのリソースを TOSCA 言語でモデリングするための ETSI 標準をサポートしています。 AWS TNB を使用すると、ETSI 準拠のサービスオーケストレーターが理解できるようにモデリング AWS のサービス することで、 をより効率的に使用できます。

AWS TNB のネットワークサービス記述子

ネットワークサービス記述子 (NSD) は、TOSCA 標準を使用して、デプロイするネットワーク機能と、ネットワーク機能をデプロイする AWS のインフラストラクチャを記述するネットワークパッケージ内の .yaml ファイルです。NSD を定義し、基盤となるリソースとネットワークライフサイクルオペレーションを設定するには、 AWS TNB でサポートされている NSD TOSCA スキーマを理解する必要があります。

NSD ファイルは、次の各パートに分割されています。

1. TOSCA 定義バージョン — これは NSD YAML ファイルの最初の行で、次に示す例のようにバージョン情報が含まれています。

```
tosca_definitions_version: tnb_simple_yaml_1_0
```

2. VNFD — NSD には、ライフサイクルオペレーションを実行するネットワーク機能の定義が含まれています。各ネットワーク機能は次の値によって識別される必要があります。
   - descriptor_id の一意の ID。ID はネットワーク機能 CSAR パッケージの ID と一致する必要があります。
   - namespace の一意の名前。次の例に示すように、NSD YAML ファイル全体でより簡単に参照できるように、名前には固有の ID を関連付ける必要があります。

```
vnfds:
  - descriptor_id: "61465757-cb8f-44d8-92c2-b69ca0de025b"
    namespace: "amf"
```

3. トポロジーテンプレート — デプロイするリソース、ネットワーク機能のデプロイ、およびライフサイクルフックなどのカスタマイズされたスクリプトを定義します。以下の例ではこれを示しています。

```
topology_template:

  node_templates:

    SampleNS:
      type: tosca.nodes.AWS.NS
      properties:
        descriptor_id: "<Sample Identifier>"
        descriptor_version: "<Sample nversion>"
        descriptor_name: "<Sample name>"
```

4. 追加ノード — モデル化された各リソースには、プロパティと要件に関するセクションがあります。プロパティには、バージョンなど、リソースのオプション属性または必須属性が記述されています。要件には、引数として指定する必要がある依存関係が記述されています。例えば、Amazon EKS ノードグループリソースを作成するには、Amazon EKS クラスター内で作成する必要があります。以下の例ではこれを示しています。

```
SampleEKSNode:
```

```
  type: tosca.nodes.AWS.Compute.EKSManagedNode
  properties:
    node_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleRole"
  capabilities:
    compute:
      properties:
        ami_type: "AL2_x86_64"
        instance_types:
          - "t3.xlarge"
        key_pair: "SampleKeyPair"
    scaling:
      properties:
        desired_size: 1
        min_size: 1
        max_size: 1
  requirements:
    cluster: SampleEKS
    subnets:
      - SampleSubnet
    network_interfaces:
      - SampleENI01
      - SampleENI02
```

NSD の例

以下は、モデル化方法を示す NSD のスニペットです AWS のサービス。ネットワーク機能は Kubernetes バージョン 1.27 を搭載した Amazon EKS クラスターにデプロイされます。アプリケーションのサブネットは Subnet01 と Subnet02 です。その後、Amazon マシンイメージ (AMI)、インスタンスタイプ、および自動スケーリング設定を使用して、アプリケーションの NodeGroups を定義できます。

```
tosca_definitions_version: tnb_simple_yaml_1_0

SampleNFEKS:
  type: tosca.nodes.AWS.Compute.EKS
  properties:
    version: "1.27"
    access: "ALL"
    cluster_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleClusterRole"
  capabilities:
    multus:
      properties:
```

```
        enabled: true
  requirements:
    subnets:
      - Subnet01
      - Subnet02

SampleNFEKSNode01:
  type: tosca.nodes.AWS.Compute.EKSManagedNode
  properties:
    node_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleNodeRole"
  capabilities:
    compute:
      properties:
        ami_type: "AL2_x86_64"
        instance_types:
          - "t3.xlarge"
        key_pair: "SampleKeyPair"
    scaling:
      properties:
        desired_size: 3
        min_size: 2
        max_size: 6
  requirements:
    cluster: SampleNFEKS
    subnets:
      - Subnet01
    network_interfaces:
      - ENI01
      - ENI02
```

AWS TNB の管理とオペレーション

AWS TNB を使用すると、ETSI SOL003 および SOL005 に従って標準化された管理オペレーションを使用してネットワークを管理できます。 AWS TNB APIs、次のようなライフサイクルオペレーションを実行できます。

- ネットワーク機能のインスタンス化。
- ネットワーク機能の終了。
- ネットワーク機能の更新による Helm デプロイの上書き。
- 新しいネットワークパッケージとパラメータ値を使用して、インスタンス化または更新されたネットワークインスタンスを更新します。

- ネットワーク機能パッケージのバージョン管理。
- NSD のバージョン管理。
- デプロイされたネットワーク機能に関する情報の取得。

AWS TNB のセットアップ

このトピックで説明されているタスクを完了して AWS TNB を設定します。

タスク

- にサインアップする AWS アカウント
- AWS リージョンを選択する
- サービスエンドポイントに関する注記
- (オプション) をインストールする AWS CLI
- AWS TNB ロールを設定する

にサインアップする AWS アカウント

の使用を開始するには AWS、 が必要です AWS アカウント。の作成の詳細については AWS アカウント、「 AWS アカウント管理 リファレンスガイド」の「 の開始方法 AWS アカウント」を参照してください。

AWS リージョンを選択する

AWS TNB で利用可能なリージョンのリストを表示するには、AWS 「リージョンサービスリスト」を参照してください。プログラムによるアクセスが可能なエンドポイントのリストを表示するには、「AWS 全般のリファレンス」の「AWS TNB のエンドポイント 」を参照してください。

サービスエンドポイントに関する注記

プログラムで AWS サービスに接続するには、 エンドポイントを使用します。一部の AWS サービスは、標準 AWS エンドポイントに加えて、選択したリージョンで FIPS エンドポイントを提供します。詳細については、AWS サービスエンドポイントを参照してください。

リージョン名	リージョン	エンドポイント	プロトコル	
米国東部 (バージニア北部)	us-east-1	tnb.us-east-1.amazonaws.com	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
米国西部 (オレゴン)	us-west-2	tnb.us-west-2.amazonaws.com	HTTPS	
アジアパシフィック (ソウル)	ap-northeast-2	tnb.ap-northeast-2.amazonaws.com	HTTPS	
アジアパシフィック (シドニー)	ap-southeast-2	tnb.ap-southeast-2.amazonaws.com	HTTPS	
カナダ (中部)	ca-central-1	tnb.ca-central-1.amazonaws.com	HTTPS	
欧州 (フランクフルト)	eu-central-1	tnb.eu-central-1.amazonaws.com	HTTPS	
欧州 (パリ)	eu-west-3	tnb.eu-west-3.amazonaws.com	HTTPS	
欧州 (スペイン)	eu-south-2	tnb.eu-south-2.amazonaws.com	HTTPS	
欧州 (ストックホルム)	eu-north-1	tnb.eu-north-1.amazonaws.com	HTTPS	
南米 (サンパウロ)	sa-east-1	tnb.sa-east-1.amazonaws.com	HTTPS	

(オプション) をインストールする AWS CLI

AWS Command Line Interface (AWS CLI) は、さまざまな AWS 製品のコマンドを提供し、Windows、macOS、Linux でサポートされています。を使用して AWS TNB にアクセスできます AWS CLI。使用を開始する方法については『AWS Command Line Interface ユーザーガイド』を参照してください。 AWS TNB のコマンドの詳細については、「 コマンドリファレンス」の「tnb」を参照してください。 AWS CLI

AWS TNB ロールを設定する

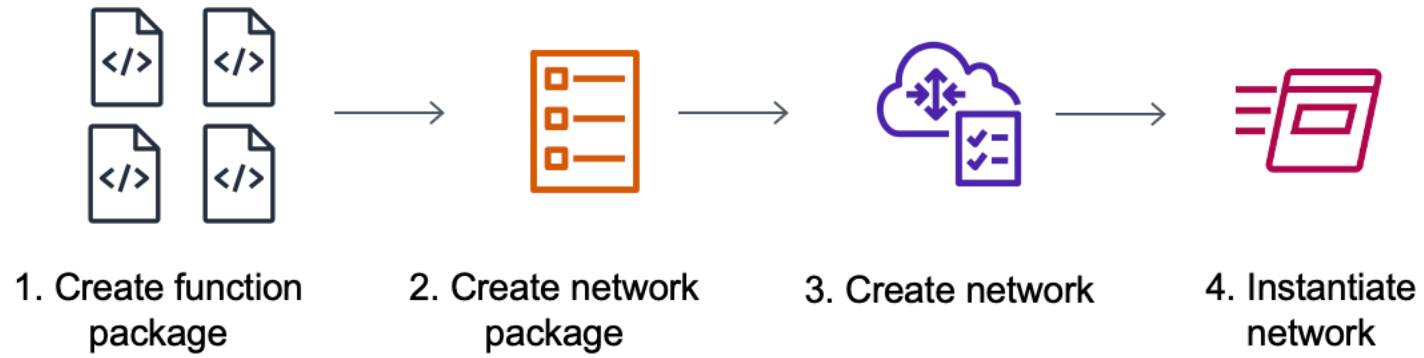
AWS TNB ソリューションのさまざまな部分を管理するには、IAM サービスロールを作成する必要があります。 AWS TNB サービスロールは、ユーザーに代わって AWS CloudFormation AWS CodeBuild、 やさまざまなコンピューティングおよびストレージサービスなどの他の AWS サービスに対して API コールを行い、デプロイ用のリソースをインスタンス化および管理できます。

AWS TNB サービスロールの詳細については、「」を参照してくださいAWS TNB の Identity and Access Management。

AWS TNB の開始方法

このチュートリアルでは、 AWS 集中型ユニット (CU)、アクセスおよびモビリティ管理関数 (AMF)、5G ユーザープレーン関数 (UPF) などのネットワーク関数を TNB を使用してデプロイする方法を示します。

以下の図は、そのデプロイプロセスを示したものです。



タスク

- 前提条件
- 関数パッケージを作成する
- ネットワークパッケージを作成する
- ネットワークインスタンスを作成してインスタンス化する
- クリーンアップ

前提条件

デプロイを正常に実行するには、次のものが必要です。

- AWS ビジネスサポートプラン。
- IAM ロールを介したアクセス許可。
- ETSI SOL001/SOL004 に準拠したネットワーク関数 (NF) パッケージ。
- ETSI SOL007 に準拠した Network Service Descriptor (NSD) テンプレート。

AWS TNB GitHub サイトのサンプルパッケージから、サンプル関数パッケージまたはネットワークパッケージを使用できます。 GitHub

関数パッケージを作成する

ネットワーク関数パッケージは、Cloud Service Archive (CSAR) ファイルです。CSAR ファイルには、Helm チャート、ソフトウェアイメージ、ネットワーク機能記述子 (NFD) が含まれています。

関数パッケージを作成するには

1. https://console.aws.amazon.com/tnb/://www.com」で AWS TNB コンソールを開きます。
2. ナビゲーションペインで、[関数パッケージ] を選択します。
3. [関数パッケージの作成] を選択します。
4. 関数パッケージのアップロード で、ファイルの選択 を選択し、各 CSAR パッケージを`.zip`ファイルとしてアップロードします。最大 10 個のファイルをアップロードできます。
5. (オプション) タグ で、新しいタグを追加 を選択し、キーと値を入力します。タグを使用して、リソースを検索およびフィルタリングしたり、 AWS コストを追跡したりできます。
6. [Next (次へ)] を選択します。
7. パッケージの詳細を確認し、[関数パッケージの作成] を選択します。

ネットワークパッケージを作成する

ネットワークパッケージは、デプロイするネットワーク関数と、それらをカタログにデプロイする方法を指定します。

ネットワークパッケージを作成するには

1. ナビゲーションペインで、[ネットワークパッケージ] を選択します。
2. [ネットワークパッケージの作成] を選択します。
3. 「ネットワークパッケージのアップロード」で「ファイルの選択」を選択し、各 NSD を`.zip`ファイルとしてアップロードします。最大 10 個のファイルをアップロードできます。
4. (オプション) タグ で、新しいタグを追加 を選択し、キーと値を入力します。タグを使用して、リソースを検索およびフィルタリングしたり、 AWS コストを追跡したりできます。
5. [Next (次へ)] を選択します。

6. [ネットワークパッケージの作成] を選択します。

ネットワークインスタンスを作成してインスタンス化する

ネットワークインスタンスは、デプロイできる AWS TNB で作成された単一のネットワークです。ネットワークインスタンスを作成してインスタンス化する必要があります。ネットワークインスタンスをインスタンス化すると、 AWS TPN は必要な AWS インフラストラクチャをプロビジョニングし、コンテナ化されたネットワーク機能をデプロイし、ネットワークとアクセス管理を設定して、完全に運用可能なネットワークサービスを作成します。

ネットワークインスタンスを作成してインスタンス化するには

1. ナビゲーションペインで [ネットワーク] を選択します。
2. [ネットワークインスタンスの作成] を選択します。
3. ネットワークの名前と説明を入力して [次へ] を選択します。
4. ネットワークパッケージを選択します。詳細を確認し、次へを選択します。
5. [ネットワークインスタンスの作成] を選択します。初期ステータスは、Created です。

   ネットワークページに、 Not instantiated状態の新しいネットワークインスタンスが表示されます。
6. ネットワークインスタンスを選択し、アクションとインスタンスを選択します。

   ネットワークのインスタンス化ページが表示されます。
7. 詳細を確認し、パラメータ値を更新します。パラメータ値の更新は、このネットワークインスタンスにのみ適用されます。NSD パッケージと VNFD パッケージのパラメータは変更されません。
8. [ネットワークをインスタンス化] を選択します。

   デプロイステータスページが表示されます。
9. 更新アイコンを使用して、ネットワークインスタンスのデプロイステータスを追跡します。デプロイタスクセクションで自動更新を有効にして、各タスクの進行状況を追跡することもできます。

クリーンアップ

このチュートリアル用に作成したリソースを削除できるようになりました。

リソースをクリーンアップするには

1. ナビゲーションペインで [ネットワーク] を選択します。
2. ネットワークの ID を選択し、[終了] を選択します。
3. 確認を求められたら、ネットワーク ID を入力して [終了] を選択します。
4. 更新アイコンを使用して、ネットワークインスタンスのステータスを追跡します。
5. (オプション) ネットワークを選択し、[削除] を選択します。

AWS TNB の関数パッケージ

関数パッケージは CSAR (Cloud Service Archive) 形式の.zip ファイルです。これにはネットワーク機能 (ETSI 標準の通信アプリケーション) と、TOSCA 標準を使用してネットワーク機能をネットワークでどのように実行するかを記述する関数パッケージ記述子が含まれています。

タスク

- AWS TNB で関数パッケージを作成する
- AWS TNB で関数パッケージを表示する
- AWS TNB から関数パッケージをダウンロードする
- AWS TNB から関数パッケージを削除する

AWS TNB で関数パッケージを作成する

AWS TNB ネットワーク関数カタログで関数パッケージを作成する方法について説明します。関数パッケージの作成は、 AWS TNB でネットワークを作成するための最初のステップです。関数パッケージをアップロードしたら、ネットワークパッケージを作成できます。

Console

コンソールを使用して関数パッケージを作成するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで、[関数パッケージ] を選択します。
3. [関数パッケージの作成] を選択します。
4. ファイルの選択を選択し、各 CSAR パッケージを.zipファイルとしてアップロードします。最大 10 個のファイルをアップロードできます。
5. [次へ] を選択します。
6. パッケージの詳細を確認します。
7. [関数パッケージの作成] を選択します。

AWS CLI

を使用して関数パッケージを作成するには AWS CLI

1. create-sol-function-package コマンドを使用して新しいファンクションパッケージを作成します。

```
aws tnb create-sol-function-package
```

2. put-sol-function-package-content コマンドを使用して、関数パッケージの内容をアップロードします。例えば、次のようになります。

```
aws tnb put-sol-function-package-content \
--vnf-pkg-id ^fp-[a-f0-9]{17}$ \
--content-type application/zip \
--file "fileb://valid-free5gc-udr.zip" \
--endpoint-url "https://tnb.us-west-2.amazonaws.com" \
--region us-west-2
```

AWS TNB で関数パッケージを表示する

関数パッケージの内容を表示する方法を説明します。

Console

コンソールを使用して関数パッケージを表示するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで、[関数パッケージ] を選択します。
3. 関数パッケージを検索するには、検索ボックスを使用します。

AWS CLI

を使用して関数パッケージを表示するには AWS CLI

1. list-sol-function-packages コマンドを使用して関数パッケージを一覧表示します。

```
aws tnb list-sol-function-packages
```

2. get-sol-function-package コマンドを使用して、関数パッケージに関する詳細を表示します。

```
aws tnb get-sol-function-package \
--vnf-pkg-id ^fp-[a-f0-9]{17}$ \
--endpoint-url "https://tnb.us-west-2.amazonaws.com" \
--region us-west-2
```

AWS TNB から関数パッケージをダウンロードする

AWS TNB ネットワーク関数カタログから関数パッケージをダウンロードする方法について説明します。

Console

コンソールを使用して関数パッケージをダウンロードするには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. コンソールの左側のナビゲーションペインで、[関数パッケージ] を選択します。
3. 関数パッケージを検索するには、検索ボックスを使用します。
4. 関数パッケージを選択する
5. [アクション]、[ダウンロード] の順に選択します。

AWS CLI

を使用して関数パッケージをダウンロードするには AWS CLI

get-sol-function-package-content コマンドを使用して、関数パッケージをダウンロードします。

```
aws tnb get-sol-function-package-content \
--vnf-pkg-id ^fp-[a-f0-9]{17}$ \
--accept "application/zip" \
--endpoint-url "https://tnb.us-west-2.amazonaws.com" \
--region us-west-2
```

AWS TNB から関数パッケージを削除する

AWS TNB ネットワーク関数カタログから関数パッケージを削除する方法について説明します。関数パッケージを削除するには、そのパッケージが無効状態になっている必要があります。

Console

コンソールを使用して関数パッケージを削除するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで、[関数パッケージ] を選択します。
3. 関数パッケージを検索するには、検索ボックスを使用します。
4. 関数パッケージを選択します。
5. [アクション]、[無効化] の順に選択します。
6. [アクション]、[削除] の順に選択します。

AWS CLI

を使用して関数パッケージを削除するには AWS CLI

1. update-sol-function-package コマンドを使用して、関数パッケージを無効にします。

```
aws tnb update-sol-function-package --vnf-pkg-id ^fp-[a-f0-9]{17}$ ---
operational-state DISABLED
```

2. delete-sol-function-package コマンドを使用して、関数パッケージを削除します。

```
aws tnb delete-sol-function-package \
--vnf-pkg-id ^fp-[a-f0-9]{17}$ \
--endpoint-url "https://tnb.us-west-2.amazonaws.com" \
--region us-west-2
```

AWS TNB のネットワークパッケージ

ネットワークパッケージは、CSAR (Cloud Service Archive) 形式の .zip ファイルです。デプロイする関数パッケージと、デプロイする AWS インフラストラクチャを定義します。

ネットワークパッケージには、次のファイルが含まれています。

- ETSI SOL007 で説明されている TOSCA 形式のネットワーク記述子ファイル (`nsd.yaml`)。

  `nsd.yaml` ファイルには、アップロードされた関数パッケージへの参照と記述子 IDsが含まれています。
- ユーザーデータスクリプトがある場合。
- ライフサイクルフックスクリプトがある場合。
- プラグイン`values.yaml`の設定ファイルがある場合。

タスク

- AWS TNB でネットワークパッケージを作成する
- AWS TNB でネットワークパッケージを表示する
- AWS TNB からネットワークパッケージをダウンロードする
- AWS TNB からネットワークパッケージを削除する

AWS TNB でネットワークパッケージを作成する

ネットワークパッケージは、ネットワークサービス記述子 (NSD) ファイル (必須) と、必要に応じて固有のスクリプトなどの追加ファイル (オプション) で構成されます。例えば、ネットワークパッケージに複数の関数パッケージが含まれている場合、NSD を使用して特定の VPC、サブネット、または Amazon EKS クラスターで実行するネットワーク機能を定義できます。

関数パッケージを作成したら、ネットワークパッケージを作成します。ネットワークパッケージを作成したら、ネットワークインスタンスを作成する必要があります。

Console

コンソールを使用してネットワークパッケージを作成するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。

2. ナビゲーションペインで、[ネットワークパッケージ] を選択します。
3. [ネットワークパッケージの作成] を選択します。
4. ファイルの選択を選択し、各 NSD を.zipファイルとしてアップロードします。最大 10 個のファイルをアップロードできます。
5. [次へ] を選択します。
6. パッケージの詳細を確認します。
7. [ネットワークパッケージの作成] を選択します。

AWS CLI

を使用してネットワークパッケージを作成するには AWS CLI

1. create-sol-network-package コマンドを使用してネットワークパッケージを作成します。

```
aws tnb create-sol-network-package
```

2. put-sol-network-package-content コマンドを使用して、ネットワークパッケージの内容をアップロードします。例えば、次のようになります。

```
aws tnb put-sol-network-package-content \
--nsd-info-id ^np-[a-f0-9]{17}$ \
--content-type application/zip \
--file "fileb://free5gc-core-1.0.9.zip" \
--endpoint-url "https://tnb.us-west-2.amazonaws.com" \
--region us-west-2
```

AWS TNB でネットワークパッケージを表示する

ネットワークパッケージの内容を表示する方法について説明します。

Console

コンソールを使用してネットワークパッケージを表示するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで、[ネットワークパッケージ] を選択します。
3. 検索ボックスを使用して、ネットワークパッケージを検索します。

AWS CLI

を使用してネットワークパッケージを表示するには AWS CLI

1. list-sol-network-packages コマンドを使用して、ネットワークパッケージを一覧表示します。

```
aws tnb list-sol-network-packages
```

2. get-sol-network-package コマンドを使用して、ネットワークパッケージに関する詳細を表示します。

```
aws tnb get-sol-network-package \
--nsd-info-id ^np-[a-f0-9]{17}$ \
--endpoint-url "https://tnb.us-west-2.amazonaws.com" \
--region us-west-2
```

AWS TNB からネットワークパッケージをダウンロードする

AWS TNB ネットワークサービスカタログからネットワークパッケージをダウンロードする方法について説明します。

Console

コンソールを使用してネットワークパッケージをダウンロードするには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで、[ネットワークパッケージ] を選択します。
3. 検索ボックスを使用して、ネットワークパッケージを検索します。
4. ネットワークパッケージを選択します。
5. [アクション]、[ダウンロード] の順に選択します。

AWS CLI

を使用してネットワークパッケージをダウンロードするには AWS CLI

- get-sol-network-package-content コマンドを使用して、ネットワークパッケージをダウンロードします。

```
aws tnb get-sol-network-package-content \
--nsd-info-id ^np-[a-f0-9]{17}$ \
--accept "application/zip" \
--endpoint-url "https://tnb.us-west-2.amazonaws.com" \
--region us-west-2
```

AWS TNB からネットワークパッケージを削除する

AWS TNB ネットワークサービスカタログからネットワークパッケージを削除する方法について説明します。ネットワークパッケージを削除するには、そのパッケージが無効状態になっている必要があります。

Console

コンソールを使用してネットワークパッケージを削除するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで、[ネットワークパッケージ] を選択します。
3. 検索ボックスを使用して、ネットワークパッケージを検索します。
4. ネットワークパッケージを選択します。
5. [アクション]、[無効化] の順に選択します。
6. [アクション]、[削除] の順に選択します。

AWS CLI

を使用してネットワークパッケージを削除するには AWS CLI

1. update-sol-network-package コマンドを使用して、ネットワークパッケージを無効にします。

```
aws tnb update-sol-network-package --nsd-info-id ^np-[a-f0-9]{17}$ --nsd-
operational-state DISABLED
```

2. delete-sol-network-package コマンドを使用して、ネットワークパッケージを削除します。

```
aws tnb delete-sol-network-package \
--nsd-info-id ^np-[a-f0-9]{17}$ \
```

```
--endpoint-url "https://tnb.us-west-2.amazonaws.com" \
--region us-west-2
```

AWS TNB のネットワークインスタンス

ネットワークインスタンスは、デプロイできる AWS TNB で作成された単一のネットワークです。

タスク

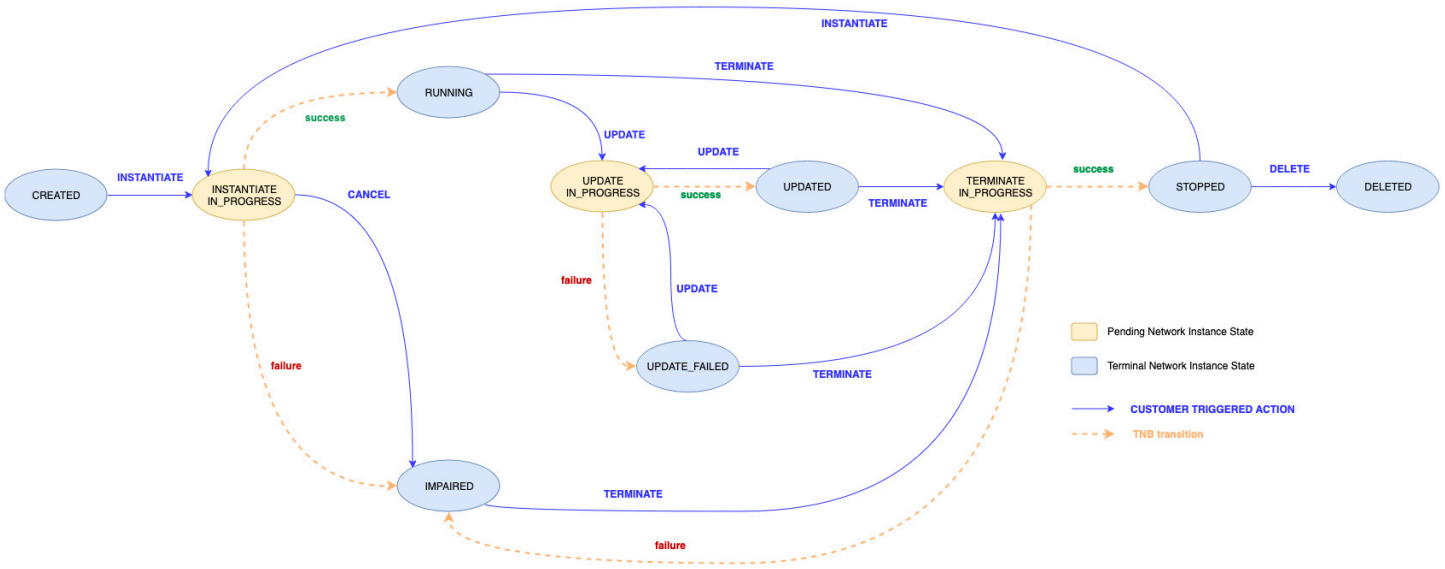
- ネットワークインスタンスのライフサイクルオペレーション
- AWS TNB を使用してネットワークインスタンスを作成する
- AWS TNB を使用してネットワークインスタンスをインスタンス化する
- AWS TNB で関数インスタンスを更新する
- AWS TNB でネットワークインスタンスを更新する
- AWS TNB でネットワークインスタンスを表示する
- AWS TNB からのネットワークインスタンスの終了と削除

ネットワークインスタンスのライフサイクルオペレーション

AWS TNB を使用すると、ETSI SOL003 および SOL005 とインラインで標準化された管理オペレーションを使用して、ネットワークを簡単に管理できます。次のライフサイクルオペレーションを実行できます。

- ネットワークを作成する
- ネットワークのインスタンス化
- ネットワーク関数を更新する
- ネットワークインスタンスを更新する
- ネットワークの詳細とステータスを表示する
- ネットワークを終了する

次の図は、ネットワーク管理オペレーションを示しています。



AWS TNB を使用してネットワークインスタンスを作成する

ネットワークインスタンスは、ネットワークパッケージの作成後に作成します。ネットワークインスタンスを作成したら、インスタンス化します。

Console

コンソールを使用してネットワークインスタンスを作成するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで [ネットワーク] を選択します。
3. [ネットワークインスタンスの作成] を選択します。
4. インスタンスの名前と説明を入力し、[次へ] を選択します。
5. ネットワークパッケージを選択し、詳細を確認し、次へを選択します。
6. [ネットワークインスタンスの作成] を選択します。

   新しいネットワークインスタンスがネットワークページに表示されます。次に、このネットワークインスタンスをインスタンス化します。

AWS CLI

を使用してネットワークインスタンスを作成するには AWS CLI

- create-sol-network-instance コマンドを使用してネットワークインスタンスを作成します。

```
aws tnb create-sol-network-instance --nsd-info-id ^np-[a-f0-9]{17}$ --ns-name
 "SampleNs" --ns-description "Sample"
```

次に、このネットワークインスタンスをインスタンス化します。

AWS TNB を使用してネットワークインスタンスをインスタンス化する

ネットワークインスタンスを作成したら、インスタンス化する必要があります。ネットワークインスタンスをインスタンス化すると、 AWS TNB は必要な AWS インフラストラクチャをプロビジョニングし、コンテナ化されたネットワーク機能をデプロイし、ネットワークとアクセス管理を設定して、完全に運用可能なネットワークサービスを作成します。

Console

コンソールを使用してネットワークインスタンスをインスタンス化するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで [ネットワーク] を選択します。
3. インスタンス化するネットワークインスタンスを選択します。
4. 「アクション」を選択し、「インスタンス化」を選択します。
5. ネットワークをインスタンス化するページで、詳細を確認し、オプションでパラメータ値を更新します。

   パラメータ値の更新は、このネットワークインスタンスにのみ適用されます。NSD パッケージと VNFD パッケージのパラメータは変更されません。
6. [ネットワークをインスタンス化] を選択します。

   デプロイステータスページが表示されます。
7. 更新アイコンを使用して、ネットワークインスタンスのデプロイステータスを追跡します。デプロイタスクセクションで自動更新を有効にして、各タスクの進行状況を追跡することもできます。

   デプロイステータスが に変わるとCompleted、ネットワークインスタンスがインスタンス化されます。

AWS CLI

を使用してネットワークインスタンスをインスタンス化するには AWS CLI

1. instantiate-sol-network-instance コマンドを使用して、ネットワークインスタンスをインスタンス化します。

```
aws tnb instantiate-sol-network-instance --ns-instance-id ^ni-[a-f0-9]{17}$ --
additional-params-for-ns "{\"param1\": \"value1\", \"param2\": \"value2\"}"
```

2. 次に、ネットワークオペレーションのステータスを表示します。

AWS TNB で関数インスタンスを更新する

ネットワークインスタンスがインスタンス化されたら、ネットワークインスタンスの関数パッケージを更新できます。

Console

コンソールを使用して関数インスタンスを更新するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで [ネットワーク] を選択します。
3. ネットワークインスタンスを選択します。ネットワークインスタンスは、その状態が の場合にのみ更新できますInstantiated。

   ネットワークインスタンスページが表示されます。
4. Functions タブから、更新する関数インスタンスを選択します。
5. [更新] を選択します。
6. 更新オーバーライドを入力します。
7. [更新] を選択します。

AWS CLI

CLI を使用して関数インスタンスを更新する

update-sol-network-instance コマンドと MODIFY_VNF_INFORMATION update タイプを使用して、ネットワークインスタンスの関数インスタンスを更新します。

```
aws tnb update-sol-network-instance --ns-instance-id ^ni-[a-f0-9]{17}$ --update-type
 MODIFY_VNF_INFORMATION --modify-vnf-info ...
```

AWS TNB でネットワークインスタンスを更新する

ネットワークインスタンスがインスタンス化されたら、インフラストラクチャまたはアプリケーションの更新が必要になる場合があります。そのためには、ネットワークインスタンスのネットワークパッケージとパラメータ値を更新し、更新オペレーションをデプロイして変更を適用します。

考慮事項

- Instantiated または Updated状態のネットワークインスタンスを更新できます。
- ネットワークインスタンスを更新すると、UpdateSolNetworkServiceAPI は新しいネットワークパッケージとパラメータ値を使用してネットワークインスタンスのトポロジを更新します。
- AWS TNB は、ネットワークインスタンス内の NSD および VNFD パラメータの数が 200 を超えないことを確認します。この制限は、サービスに影響を与える誤ったペイロードや巨大なペイロードを渡す不正なアクターから保護するために適用されます。

更新できるパラメータ

インスタンス化されたネットワークインスタンスを更新するときに、次のパラメータを更新できます。

パラメータ	説明	例: 前	例: 後
Amazon EKS クラスターバージョン	Amazon EKS クラスターコントロールプレーンversionパラメータの値を次のマイナーバージョンに更新できます。バージョンをダウングレードすることはできません。	<pre>EKSCluster: type: tosca.nod es.AWS.Compute.EKS properties: version: "1.28"</pre>	<pre>EKSC r: typ tos es.A mput</pre>

パラメータ	説明	例: 前	例: 後
			pro s: ver "1.

パラメータ	説明	例: 前	例: 後
Amazon EKS ワーカーノード	EKSManagedNode kubernetes_version パラメータの値を更新してノードグループを新しい Amazon EKS バージョンにアップグレードするか、 ami_idパラメータを更新してノードグループを最新の EKS 最適化 AMI にアップグレードできます。 の AMI ID を更新できますEKSSelfManagedNode 。AMI の Amazon EKS バージョンは、Amazon EKS クラスターバージョンと同じか、最大 2 つ前のバージョンである必要があります。例えば、Amazon EKS クラスターバージョンが 1.31 の場合、Amazon EKS AMI バージョンは 1.31、1.30、または 1.29 である必要があります。	<pre>EKSManagedNodeGroup01: ... properties: kubernete s_version: " 1.28" EKSSelfManagedNode 01: compute: compute: properties: ami_id: "ami-1231230LD "</pre>	<pre>EKSM dNoc p01: ... pro s: kub s_ve : "1. EKS nage 01: com</pre>

パラメータ	説明	例: 前	例: 後
			com pro s: ami "am 3NEW

パラメータ	説明	例: 前	例: 後
Amazon EKS ノードグループ	コンピューティングのニーズに応じて、ノードグループを追加または削除できます。 既存のノードグループを削除して新しいノードグループを追加するときは、新しいノードグループに削除されたノードグループとは異なる IDs があることを確認してください。そうしないと、オペレーションは削除や追加ではなくノードグループの変更として扱われます。既存のノードグループでは、更新できるパラメータのセットは限られています。このテーブルをスクロールして、更新できるパラメータを確認します。	<pre>Free5GCEKSNode01: type: tosca.nod es.AWS.Compute.EKS ManagedNode ... scaling: properties: desired_size: 1 min_size: 1 max_size: 1 ... *Free5GCEKSNode02* : # Deleted Nodegroup type: tosca.nod es.AWS.Compute.EKS ManagedNode ... scaling: properties: desired_size: 1 min_size: 1 max_size: 1 ... *Free5GCEKSNode03* : # Deleted Nodegroup type: tosca.nod es.AWS.Compute.EKS SelfManagedNode ... scaling: properties: desired_size: 1 min_size: 1 max_size: 1 ...</pre>	<pre>Free SNod typ tos es.A mput Mana de ... sca pro s: des ize: 1</pre>

パラメータ	説明	例: 前	例: 後
			mir 1 max 1 ... Free SNod # New Nod typ tos es.A mput Self edNo ... sca pro s:

パラメータ	説明	例: 前	例: 後
			des ize: 1 mir 1 max 1 ... Free SNod # New Nod typ tos es.A mput Mana de

パラメータ	説明	例: 前	例: 後
			... sca pro s: des ize: 1 mir 1 max 1

パラメータ	説明	例: 前	例: 後
			...

パラメータ	説明	例: 前	例: 後

スケーリングプロパティ	EKSManagedNode および EKSSelfManagedNode TOSCA ノードのスケーリングプロパティを更新できます。	`EKSNodeGroup01:`
		`  ...`
		`  scaling:`
		`  properties:`
		`  desired_size: 1`
		`  min_size: 1`
		`  max_size: 1`
			`EKSN`
			`oup0`
			`...`
			`sca`
			`pro`
			`s:`
			`des`
			`ize:`

パラメータ	説明	例: 前	例: 後
			mir max

パラメータ	説明	例: 前	例: 後

Amazon EBS CSI プラグインのプロパティ	Amazon EKS クラスターで Amazon EBS CSI プラグインを有効または無効にできます。プラグインのバージョンを変更することもできます。	`EKSCluster:`
		`  capabilities:`
		`  ...`
		`  ebs_csi:`
		`  properties:`
		`  enabled: false`
			`EKSC`
			`r:`
			`cap`
			`ies:`
			`...`
			`ebs`
			`pro`
			`s:`
			`ena`
			`ver`
			`"v1`
			`e`
			`ksbu`
			`"`

パラメータ	説明	例: 前	例: 後

ルートボリュームサイズ	EKSManagedNode ノードと EKSSelfManagedNode TOSCA ノードのルートボリュームサイズプロパティを追加、削除、または更新できます。	`Free5GCEKSNode01:`
		`  ...`
		`  capabilities:`
		
		`  compute:`
		`  properties:`
		`  root_volume_size: 50`
			`Free`
			`SNod`
			
			`...`
			
			`cap`
			`ies:`
			
			`com`
			
			`pro`
			`s:`

パラメータ	説明	例: 前	例: 後
			roo me_s

パラメータ	説明	例: 前	例: 後
VNF	NSD 内の VNFsを参照し、VNFDeployment TOSCA ノードを使用して NSD で作成されたクラスターにデプロイできます。更新の一環として、ネットワークに VNFsを追加、更新、削除できます。	vnfds: - descriptor_id: "43c012fa-2616-41a8-a833-0dfd4c5a049e " namespace: " vnf1" - descriptor_id: "64222f98-ecd6-4871-bf94-7354b53f3ee5 " namespace: "vnf2" // Deleted VNF ... SampleVNF1HelmDeploy: type: tosca.nodes.AWS.Deployment.VNFDeployment requirements: cluster: EKSCluster vnfs: - vnf1.SampleVNF1 - vnf2.SampleVNF2	vnfd - des r_id "59 79e9 - be53 2ad0 " nam : "vr Upd VNF - des r_id "b7 839d -916 a166 " nam : "vr Add VNF .... Sa mple

パラメータ	説明	例: 前	例: 後
			elmD : typ tos es.A ploy VNFD ment req nts: clu EKS r vnf

パラメータ	説明	例: 前	例: 後
			- w leVN - w leVN

パラメータ	説明	例: 前	例: 後
フック	ネットワーク関数の作成前と作成後にライフサイクルオペレーションを実行するには、 pre_create および post_create フックをVNFDeployment ノードに追加します。 この例では、 がvnf3.SampleVNF3 インスタンス化される前にPreCreateHook フックが実行され、 vnf3.SampleVNF3 がインスタンス化された後にPostCreateHook フックが実行されます。	vnfds: - descriptor_id: "43c012fa-2616-41a8-a833-0dfd4c5a049e " namespace: " vnf1" - descriptor_id: "64222f98-ecd6-4871-bf94-7354b53f3ee5 " namespace: " vnf2" ... SampleVNF1HelmDeploy: type: tosca.nodes.AWS.Deployment.VNFDeployment requirements: cluster: EKSCluster vnfs: - vnf1.SampleVNF1 - vnf2.SampleVNF2 // Removed during update	vnfd - des r_id "43 2616 - a833 d4c5 " nam : "vn - des r_id "b7 839d -916 a166 " nam : "vn .... S ampl Helm y: typ tos

パラメータ	説明	例: 前	例: 後
			es.A ploy VNFD ment rec nts: clu EKS r vnf - v leVN No cha to thi fur as the nam and uui rem the sam

パラメータ	説明	例: 前	例: 後
			- v leVN New VNF as the nam , vnf was not pre y pre int s: Hoo pos te: eHoo pre e: Hook

パラメータ	説明	例: 前	例: 後
フック	ネットワーク関数を更新する前と後にライフサイクルオペレーションを実行するには、 pre_update フックと post_update フックをVNFDeployment ノードに追加します。 この例では、 vnf1.SampleVNF1 は が更新された前にPreUpdateHook 実行され、 vnf1.SampleVNF1 が名前空間 vnf1 uuid用に更新された で示されるvnfパッケージに更新された後にPostUpdateHook 実行されます。	vnfds: - descriptor_id: "*43c012fa-2616-41a8-a833-0dfd4c5a049e* " namespace: " *vnf1*" - descriptor_id: "*64222f98-ecd6-4871-bf94-7354b53f3ee5* " namespace: " *vnf2*" ... SampleVNF1HelmDeploy: type: tosca.nodes.AWS.Deployment.VNFDeployment requirements: cluster: EKSCluster vnfs: - *vnf1.SampleVNF1* - *vnf2.SampleVNF2*	vnfc - des r_ic "*0e* *bd87* *-* *b8a1* *4666* " nam : "*vn* - des r_ic "*64* *ecd6* *-* *bf94* *4b53* " nam : "*vn* ... S ampl Helm y: typ

パラメータ	説明	例: 前	例: 後
			tos es.A ploy VNFD ment rec nts: clu EKS r vnf - v leVN A VNF upd as the uui cha for nam "vr - v

パラメータ	説明	例: 前	例: 後
			leVN No cha to thi fur as nam and uui rem the sam int s: Hoo pre e: Hook pos te: eHoo

パラメータ	説明	例: 前	例: 後
サブネット	ネットワークからサブネットを追加または削除できます。サブネットを削除する前に、サブネットがネットワーク内のどのリソースでも使用されていないことを確認します。	*Free5GCSubnet01* : #Deleted Subnet type: tosca.nodes.AWS.Networking.Subnet properties: type: "PUBLIC" availability_zone: { get_input: subnet_01_az } cidr_block: { get_input: subnet_01_cidr_block } requirements: route_table: Free5GCRouteTable vpc: Free5GCVPC	*Free* *bnet* #Ne Sub typ tos es.A twor Subn pro s: typ "PU ava ity_ { g : sub _az cid k: { g : sub _cid ck } req nts:

パラメータ	説明	例: 前	例: 後
			rou le: Fre uteT vpc Fre C

パラメータ	説明	例: 前	例: 後
セキュリティグループ	ネットワークからセキュリティグループを追加および削除できます。セキュリティグループを削除する前に、セキュリティグループがネットワーク内のリソースで使用されていないことを確認します。	*Free5GCSecurityGroup01* : #Deleted Security Group type: tosca.nodes.AWS.Networking.SecurityGroup properties: description: "SecurityGroup for Free5GC cluster" name: "Free5GCSecurityGroup01" tags: - "Name=Free5GCEKSAdditionalSecurityGroup" requirements: vpc: Free5GCVPC *Free5GCSecurityGroupEgressRule01* : #Deleted Security Group Egress Node type: tosca.nodes.AWS.Networking.SecurityGroupEgressRule properties: ip_protocol: "tcp" from_port: 8000 to_port: 9000 description: "Egress Rule for free5GC cluster" cidr_ip : "172.10.10.1/24" requirements: security_group: Free5GCSecurityGroup01	*Free* *curi* *up02* #Ne Sec Gro typ tos es.A twor Secu roup pro s: des on: "Se Grou for Fre clu nam "Fr ecur oup0 tag

パラメータ	説明	例: 前	例: 後
		Free5GCSecurityGroupIngressRule01 : #Deleted Security Group Ingress Node type: tosca.nodes.AWS.Networking.SecurityGroupIngressRule properties: ip_protocol: "tcp" from_port: 8000 to_port: 9000 description: "Ingress Rule for free5GC cluster" cidr_ip: "172.10.10.1/24" requirements: security_group: Free5GCSecurityGroup01	- "Name5GCditionecuroup" recnts: vpcFreC *Freecuriup Egule0* #NeSecGroEgrNod typtoses.Atwor Securoup sRul prcs:

パラメータ	説明	例: 前	例: 後
			ip_ ol: "to fro : 800 to_ 900 des on: "Eg Rul for fre clu cid "17 0.1/ req nts: sec grou Fre

パラメータ	説明	例: 前	例: 後
			curi up02 *Free* *curi* *upIr* *Rule* #Ne Sec Gro Ing Noc typ tos es.A twor Secu roup ssRu pro s: ip_ ol: "to fro : 800 to_ 900

パラメータ	説明	例: 前	例: 後
			des on: "In Rul for fre clu cid "17 0.1/ rec nts: sec grou Fre curi up02

パラメータ	説明	例: 前	例: 後
ネットワークインターフェイス	ネットワークから ENIs を追加、変更、削除できます。	Free5GCENI01: #Modified ENI type: tosca.nodes.AWS.Networking.ENI properties: device_index: 2 requirements: subnet: *Free5GCENISubnet01* security_groups: - Free5GCSecurityGroup01 Free5GCENI02: #Modified ENI type: tosca.nodes.AWS.Networking.ENI properties: device_index: 3 source_dest_check: true requirements: subnet: Free5GCENISubnet01 *Free5GCENI04* : #Deleted ENI type: tosca.nodes.AWS.Networking.ENI properties: device_index: 4 source_dest_check: true requirements: subnet: Free5GCENISubnet01	Free I01: #Mo ENI typ tos es.A twor ENI pro s: dev dex: 2 req nts: sub *ISub* sec grou - Fre

パラメータ	説明	例: 前	例: 後
			curi up01 Fre e5GC : #Mc ENI typ tos es.A twor ENI pro s: dev dex: 3 sou st_c tru req nts: sub Fre ISub se grou

パラメータ	説明	例: 前	例: 後
			- Fre curi up01 Free I03 #Ne ENI typ tos es.A twor ENI pro s: dev dex: 3 rec nts: sub Fre bnet sec grou

パラメータ	説明	例: 前	例: 後
			- Fre curi up01

ネットワークインスタンスの更新

Console

コンソールを使用してネットワークインスタンスを更新するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで [ネットワーク] を選択します。
3. ネットワークインスタンスを選択します。ネットワークインスタンスは、その状態が `Instantiated`または `Updated` の場合にのみ更新できます。
4. アクションと更新を選択します。

   インスタンスの更新ページに、現在のインフラストラクチャのネットワークの詳細とパラメータのリストが表示されます。
5. 新しいネットワークパッケージを選択します。

   新しいネットワークパッケージのパラメータは、パラメータの更新セクションに表示されます。
6. 必要に応じて、「パラメータの更新」セクションのパラメータ値を更新します。更新できるパラメータ値のリストについては、「」を参照してください更新できるパラメータ。
7. ネットワークの更新を選択します。

   AWS TNB はリクエストを検証し、デプロイを開始します。デプロイステータスページが表示されます。

8. 更新アイコンを使用して、ネットワークインスタンスのデプロイステータスを追跡します。デプロイタスクセクションで自動更新を有効にして、各タスクの進行状況を追跡することもできます。

   デプロイステータスが に変わるとCompleted、ネットワークインスタンスが更新されます。

9.
   - 検証が失敗した場合、ネットワークインスタンスは更新をリクエストする前と同じ状態のままになります。 Instantiatedまたは ですUpdated。
   - 更新が失敗した場合、ネットワークインスタンスの状態は と表示されますUpdate failed。失敗した各タスクのリンクを選択して、理由を特定します。
   - 更新が成功すると、ネットワークインスタンスの状態は と表示されますUpdated。

AWS CLI

CLI を使用して、ネットワークインスタンスを更新する

updateupdate-sol-network-instance コマンドと UPDATE_NS update タイプを使用して、ネットワークインスタンスを更新します。

```
aws tnb update-sol-network-instance --ns-instance-id ^ni-[a-f0-9]{17}$ --
update-type UPDATE_NS --update-ns "{\"nsdInfoId\":\"^np-[a-f0-9]{17}$\",
 \"additionalParamsForNs\": {\"param1\": \"value1\"}}"
```

AWS TNB でネットワークインスタンスを表示する

ネットワークインスタンスを表示する方法について説明します。

Console

コンソールを使用してネットワークインスタンスを表示するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで、[ネットワークインスタンス] を選択します。
3. 検索ボックスを使用して、ネットワークインスタンスを検索します。

AWS CLI

を使用してネットワークインスタンスを表示するには AWS CLI

1. list-sol-network-instances コマンドを使用して、ネットワークインスタンスを一覧表示します。

```
aws tnb list-sol-network-instances
```

2. getget-sol-network-instance コマンドを使用して、特定のネットワークインスタンスの詳細を表示します。

```
aws tnb get-sol-network-instance --ns-instance-id ^ni-[a-f0-9]{17}$
```

AWS TNB からのネットワークインスタンスの終了と削除

ネットワークインスタンスを削除するには、そのインスタンスが終了状態である必要があります。

Console

コンソールを使用してネットワークインスタンスを終了し、削除するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで [ネットワーク] を選択します。
3. ネットワークインスタンスの ID を選択します。
4. [Terminate] (終了) を選択します。
5. 確認を求められたら、ID を入力して [終了] を選択します。
6. 更新して、ネットワークインスタンスのステータスを追跡します。
7. (オプション) ネットワークインスタンスを選択し、[削除] を選択します。

AWS CLI

を使用してネットワークインスタンスを終了および削除するには AWS CLI

1. terminate-sol-network-instance コマンドを使用して、ネットワークインスタンスを終了します。

```
aws tnb terminate-sol-network-instance --ns-instance-id ^ni-[a-f0-9]{17}$
```

2. (オプション) delete-sol-network-instance コマンドを使用してネットワークインスタンスを削除します。

```
aws tnb delete-sol-network-instance --ns-instance-id ^ni-[a-f0-9]{17}$
```

AWS TNB のネットワークオペレーション

ネットワークオペレーションとは、ネットワークインスタンスのインスタンス化や終了など、ネットワークに対して行われる操作です。

タスク

- AWS TNB ネットワークオペレーションを表示する
- AWS TNB ネットワークオペレーションをキャンセルする

AWS TNB ネットワークオペレーションを表示する

ネットワークオペレーションに関係するタスクやタスクのステータスなど、ネットワークオペレーションの詳細を表示します。

Console

コンソールを使用してネットワークオペレーションを表示するには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで、[ネットワークインスタンス] を選択します。
3. 検索ボックスを使用して、ネットワークインスタンスを検索します。
4. デプロイタブで、ネットワークオペレーションを選択します。

AWS CLI

を使用してネットワークオペレーションを表示するには AWS CLI

1. list-sol-network-operations コマンドを使用して、すべてのネットワークオペレーションを一覧表示します。

```
aws tnb list-sol-network-operations
```

2. get-sol-network-operation コマンドを使用して、ネットワークオペレーションに関する詳細を表示します。

```
aws tnb get-sol-network-operation --ns-lcm-op-occ-id ^no-[a-f0-9]{17}$
```

AWS TNB ネットワークオペレーションをキャンセルする

ネットワークオペレーションをキャンセルする方法について説明します。

Console

コンソールを使用してネットワークオペレーションをキャンセルするには

1. https://console.aws.amazon.com/tnb/ で AWS TNB コンソールを開きます。
2. ナビゲーションペインで [ネットワーク] を選択します。
3. ネットワークの ID を選択して、その詳細ページを開きます。
4. [デプロイ] タブで、[ネットワークオペレーション] を選択します。
5. [オペレーションをキャンセル] を選択します。

AWS CLI

を使用してネットワークオペレーションをキャンセルするには AWS CLI

cancel-sol-network-operation コマンドを使用して、ネットワークオペレーションをキャンセルします。

```
aws tnb cancel-sol-network-operation --ns-lcm-op-occ-id ^no-[a-f0-9]{17}$
```

AWS TNB の TOSCA リファレンス

Topology and Orchestration Specification for Cloud Applications (TOSCA) は、CSP がクラウドベースの Web サービス、そのコンポーネント、関係、およびそれらを管理するプロセスのトポロジを記述するために使用する宣言構文です。CSP は、接続ポイント、接続ポイント間の論理リンク、アフィニティやセキュリティなどのポリシーを TOSCA テンプレートに記述します。次にCSPs はテンプレートを AWS TNB にアップロードし、ア AWS ベイラビリティーゾーン間で機能する 5G ネットワークを確立するために必要なリソースを合成します。

内容

VNFD テンプレート

仮想ネットワーク機能記述子 (VNFD) テンプレートを定義します。

構文

```
tosca_definitions_version: tnb_simple_yaml_1_0

topology_template:

  inputs:
    SampleInputParameter:
      type: String
      description: "Sample parameter description"
      default: "DefaultSampleValue"

  node_templates:
    SampleNode1: tosca.nodes.AWS.VNF
```

トポロジテンプレート

`node_templates`

TOSCA AWS ノード。使用できるノードは次のとおりです。

- AWS.VNF
- AWS.Artifacts.Helm

AWS.VNF

AWS 仮想ネットワーク関数 (VNF) ノードを定義します。

構文

```
tosca.nodes.AWS.VNF:
  properties:
    descriptor_id: String
    descriptor_version: String
    descriptor_name: String
    provider: String
  requirements:
    helm: String
```

プロパティ

`descriptor_id`

記述子の UUID。

必須: はい

タイプ: 文字列

パターン: `[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}`

`descriptor_version`

VNFD のバージョン。

必須: はい

タイプ: 文字列

パターン: `^[0-9]{1,5}\\.[0-9]{1,5}\\.[0-9]{1,5}.*`

`descriptor_name`

記述子の名前。

必須: はい

タイプ: 文字列

`provider`

VNFD の作成者。

必須: はい

タイプ: 文字列

要件

`helm`

コンテナアーティファクトを定義する Helm ディレクトリ。これは AWS.Artifacts.Helm への参照です。

必須: はい

タイプ: 文字列

例

```
SampleVNF:
  type: tosca.nodes.AWS.VNF
  properties:
    descriptor_id: "6a792e0c-be2a-45fa-989e-5f89d94ca898"
    descriptor_version: "1.0.0"
    descriptor_name: "Test VNF Template"
    provider: "Operator"
  requirements:
    helm: SampleHelm
```

AWS.Artifacts.Helm

AWS Helm ノードを定義します。

構文

```
tosca.nodes.AWS.Artifacts.Helm:
```

```
  properties:
    implementation: String
```

プロパティ

`implementation`

CSAR パッケージ内の Helm チャートを含むローカルディレクトリ。

必須: はい

タイプ: 文字列

例

```
SampleHelm:
  type: tosca.nodes.AWS.Artifacts.Helm
  properties:
    implementation: "./vnf-helm"
```

ネットワークサービス記述子テンプレート

ネットワークサービス記述子 (NSD) テンプレートを定義します。

構文

```
tosca_definitions_version: tnb_simple_yaml_1_0

vnfds:
  - descriptor_id: String
    namespace: String

topology_template:

  inputs:
    SampleInputParameter:
      type: String
      description: "Sample parameter description"
      default: "DefaultSampleValue"
```

```
  node_templates:
    SampleNode1: tosca.nodes.AWS.NS
```

定義済みのパラメータを使用する

VPC ノードの CIDR ブロックなどのパラメータを動的に渡す場合は、{ get_input: *input-parameter-name* } 構文を使用して NSD テンプレートでパラメータを定義できます。その後、同じ NSD テンプレートでパラメータを再利用します。

次のコード例は、パラメータを定義して使用する方法を示しています。

```
tosca_definitions_version: tnb_simple_yaml_1_0

topology_template:

  inputs:
    cidr_block:
      type: String
      description: "CIDR Block for VPC"
      default: "10.0.0.0/24"

  node_templates:
    ExampleSingleClusterNS:
      type: tosca.nodes.AWS.NS
      properties:
        descriptor_id: "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
         .....

    ExampleVPC:
      type: tosca.nodes.AWS.Networking.VPC
      properties:
        cidr_block: { get_input: cidr_block }
```

VNFD インポート

`descriptor_id`

記述子の UUID。

必須: はい

タイプ: 文字列

パターン: `[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}`

`namespace`

一意の名前。

必須: はい

タイプ: 文字列

トポロジテンプレート

`node_templates`

可能な TOSCA AWS ノードは次のとおりです。

- AWS.NS
- AWS.Compute.EKS
- AWS.Compute.EKS.AuthRole
- AWS.Compute.EKSManagedNode
- AWS.Compute.EKSSelfManagedNode
- AWS.Compute.PlacementGroup
- AWS.Compute.UserData
- AWS.Networking.SecurityGroup
- AWS.Networking.SecurityGroupEgressRule
- AWS.Networking.SecurityGroupIngressRule
- AWS.Resource.Import
- AWS.Networking.ENI
- AWS.HookExecution
- AWS.Networking.InternetGateway
- AWS.Networking.RouteTable
- AWS.Networking.Subnet
- AWS.Deployment.VNFDeployment

- AWS.Networking.VPC
- AWS.Networking.NATGateway
- AWS.Networking.Route

AWS.NS

AWS ネットワークサービス (NS) ノードを定義します。

構文

```
tosca.nodes.AWS.NS:
  properties:
    descriptor_id: String
    descriptor_version: String
    descriptor_name: String
```

プロパティ

`descriptor_id`

記述子の UUID。

必須: はい

タイプ: 文字列

パターン: `[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}`

`descriptor_version`

NSD のバージョン。

必須: はい

タイプ: 文字列

パターン: `^[0-9]{1,5}\\.[0-9]{1,5}\\.[0-9]{1,5}.*`

`descriptor_name`

記述子の名前。

必須: はい

タイプ: 文字列

例

```
SampleNS:
  type: tosca.nodes.AWS.NS
  properties:
    descriptor_id: "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    descriptor_version: "1.0.0"
    descriptor_name: "Test NS Template"
```

AWS.Compute.EKS

クラスターの名前、目的の Kubernetes バージョン、および Kubernetes コントロールプレーンが NFs に必要な AWS リソースを管理できるようにするロールを指定します。Multus コンテナネットワークインターフェイス (CNI) プラグインが有効になっています。複数のネットワークインターフェイスをアタッチし、Kubernetes ベースのネットワーク機能に高度なネットワーク設定を適用できます。また、クラスターエンドポイントのアクセスとクラスターのサブネットも指定します。

構文

```
tosca.nodes.AWS.Compute.EKS:
  capabilities:
    multus:
      properties:
        enabled: Boolean
        multus_role: String
    ebs_csi:
      properties:
        enabled: Boolean
        version: String
  properties:
    version: String
    access: String
    cluster_role: String
    tags: List
    ip_family: String
  requirements:
    subnets: List
```

機能

`multus`

オプション。Multus コンテナネットワークインターフェイス (CNI) の使用方法を定義するプロパティです。

`multus` を含めた場合は、`enabled` および `multus_role` の各プロパティを指定します。

`enabled`

デフォルトの Multus 機能が有効かどうかを示します。

必須: はい

タイプ: ブール値

`multus_role`

Multus ネットワークインターフェイス管理のロール。

必須: はい

タイプ: 文字列

`ebs_csi`

Amazon EKS クラスターにインストールされる Amazon EBS Container Storage Interface (CSI) ドライバーを定義するプロパティ。

このプラグインを有効にして、、 AWS ローカルゾーン AWS Outposts、または で Amazon EKS セルフマネージドノードを使用します AWS リージョン。詳細については、「Amazon EKS ユーザーガイド」の「Amazon Elastic Block Store CSI ドライバー」を参照してください。

`enabled`

デフォルトの Amazon EBS CSI ドライバーがインストールされているかどうかを示します。

必須: いいえ

型: ブール

`version`

Amazon EBS CSI ドライバーアドオンのバージョン。バージョンは、DescribeAddonVersions アクションによって返されるバージョンのいずれかに一致する必要があります。詳細については、「Amazon EKS API リファレンス」の「DescribeAddonVersions」を参照してください。

必須: いいえ

タイプ: 文字列

プロパティ

`version`

クラスターの Kubernetes バージョン。 AWS Telco Network Builder は、Kubernetes バージョン 1.27 から 1.34 をサポートしています。

必須: はい

タイプ: 文字列

指定できる値: 1.27 | 1.28 | 1.29 | 1.30 | 1.31 | 1.32 | 1.33 | 1.34

`access`

クラスターエンドポイントのアクセス。

必須: はい

タイプ: 文字列

使用できる値: `PRIVATE` | `PUBLIC` | `ALL`

`cluster_role`

クラスター管理のロール。

必須: はい

タイプ: 文字列

`tags`

このリソースにアタッチするタグ。

必須: いいえ

タイプ: リスト

`ip_family`

クラスター内のサービスアドレスとポッドアドレスの IP ファミリーを示します。

許可される値: IPv4、IPv6

デフォルト値: IPv4

必須: いいえ

タイプ: 文字列

要件

`subnets`

AWS.Networking.Subnet ノード。

必須: はい

タイプ: リスト

例

```
SampleEKS:
  type: tosca.nodes.AWS.Compute.EKS
  properties:
    version: "1.26"
    access: "ALL"
    cluster_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleRole"
    ip_family: "IPv6"
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
  capabilities:
    multus:
      properties:
        enabled: true
        multus_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/MultusRole"
    ebs_csi:
      properties:
```

```
        enabled: true
        version: "v1.16.0-eksbuild.1"
  requirements:
    subnets:
    - SampleSubnet01
    - SampleSubnet02
```

AWS.Compute.EKS.AuthRole

AuthRole を使用すると Amazon EKS クラスター aws-auth ConfigMap に IAM ロールを追加できるようになり、ユーザーは IAM ロールを使用して Amazon EKS クラスターにアクセスできます。

構文

```
tosca.nodes.AWS.Compute.EKS.AuthRole:
  properties:
    role_mappings: List
      arn: String
      groups: List
  requirements:
    clusters: List
```

プロパティ

role_mappings

Amazon EKS クラスター aws-auth ConfigMap に追加する必要がある IAM ロールを定義するマッピングのリスト。

arn

IAM ロールの ARN。

必須: はい

タイプ: 文字列

groups

arn で定義されているロールに割り当てる Kubernetes グループ。

必須: いいえ

タイプ: リスト

要件

`clusters`

AWS.Compute.EKS ノード。

必須: はい

タイプ: リスト

例

```
EKSAuthMapRoles:
    type: tosca.nodes.AWS.Compute.EKS.AuthRole
    properties:
        role_mappings:
        - arn: arn:aws:iam::${AWS::TNB::AccountId}:role/TNBHookRole1
          groups:
          - system:nodes
          - system:bootstrappers
        - arn: arn:aws:iam::${AWS::TNB::AccountId}:role/TNBHookRole2
          groups:
          - system:nodes
          - system:bootstrappers
    requirements:
         clusters:
         - Free5GCEKS1
         - Free5GCEKS2
```

AWS.Compute.EKSManagedNode

AWS TNB は EKS Managed Node グループをサポートし、Amazon EKS Kubernetes クラスターのノード (Amazon EC2 インスタンス) のプロビジョニングとライフサイクル管理を自動化します。EKS ノードグループを作成するには、以下を実行します。

- AMI の ID または AMI タイプを指定して、クラスターワーカーノードの Amazon マシンイメージ (AMI) を選択します。
- SSH アクセス用の Amazon EC2 キーペアと、ノードグループのスケーリングプロパティを指定します。
- ノードグループが Amazon EKS クラスターに関連付けられていることを確認します。

- ワーカーノードのサブネットを指定します。
- 必要に応じて、セキュリティグループ、ノードラベル、プレイスメントグループをノードグループにアタッチします。

構文

```
tosca.nodes.AWS.Compute.EKSManagedNode:
  capabilities:
    compute:
      properties:
        ami_type: String
        ami_id: String
        instance_types: List
        key_pair: String
        root_volume_encryption: Boolean
        root_volume_encryption_key_arn: String
        root_volume_size: Integer
    scaling:
      properties:
        desired_size: Integer
        min_size: Integer
        max_size: Integer
  properties:
    node_role: String
    tags: List
    kubernetes_version: String
  requirements:
    cluster: String
    subnets: List
    network_interfaces: List
    security_groups: List
    placement_group: String
    user_data: String
    labels: List
```

機能

`compute`

Amazon EKS マネージド型ノードグループのコンピューティングパラメータを定義するプロパティ (Amazon EC2 インスタンスタイプや Amazon EC2 インスタンス AMI など)。

`ami_type`

Amazon EKS がサポートする AMI タイプ。

必須: はい

タイプ: 文字列

指定できる値: `AL2_x86_64` | `AL2_x86_64_GPU` | `AL2_ARM_64` | `AL2023_x86_64` | `AL2023_ARM_64` | | `AL2023_x86_64_NVIDIA` | `AL2023_x86_64_NEURON` | `CUSTOM` | `BOTTLEROCKET_ARM_64` | `BOTTLEROCKET_x86_64` `BOTTLEROCKET_ARM_64_NVIDIA` | `BOTTLEROCKET_x86_64_NVIDIA`

`ami_id`

AMI の ID。

必須: いいえ

タイプ: 文字列

> **Note**
> テンプレートで `ami_type`と の両方が指定され`ami_id`ている場合、 AWS TNB は `ami_id`値のみを使用して を作成します`EKSManagedNode`。

`instance_types`

インスタンスのサイズ。

必須: はい

タイプ: リスト

`key_pair`

SSH アクセスを有効にする EC2 キーペア。

必須: はい

タイプ: 文字列

`root_volume_encryption`

Amazon EBS ルートボリュームの Amazon EBS 暗号化を有効にします。このプロパティが指定されていない場合、 AWS TNB はデフォルトで Amazon EBS ルートボリュームを暗号化します。

必須: いいえ

デフォルト: true

タイプ: ブール値

`root_volume_encryption_key_arn`

AWS KMS key. AWS TNB の ARN は、通常のキー ARN、マルチリージョンキー ARN、エイリアス ARN をサポートしています。

必須: いいえ

タイプ: 文字列

> **Note**
>
> - `root_volume_encryption` が false の場合、 を含めないでください`root_volume_encryption_key_arn`。
> - AWS TNB は、Amazon EBS-backed AMI のルートボリューム暗号化をサポートしています。
> - AMI のルートボリュームがすでに暗号化されている場合は、 AWS TTB `root_volume_encryption_key_arn`がルートボリュームを再暗号化するための を含める必要があります。
> - AMI のルートボリュームが暗号化されていない場合、 AWS TNB は `root_volume_encryption_key_arn` を使用してルートボリュームを暗号化します。
>
>   を含めない場合`root_volume_encryption_key_arn`、 AWS TNB は が提供するデフォルトキー AWS Key Management Service を使用してルートボリュームを暗号化します。
> - AWS TNB は暗号化された AMI を復号しません。

`root_volume_size`

Amazon Elastic Block Store ルートボリュームのサイズを GiBs。

必須: いいえ

デフォルト: 20

タイプ: 整数

指定できる値: 1 ~ 16,384

`scaling`

Amazon EKS マネージド型ノードグループのスケーリングパラメータ (必要な Amazon EC2 インスタンスの数、ノードグループ内の Amazon EC2 インスタンスの最小数と最大数など) を定義するプロパティ。

`desired_size`

この NodeGroup のインスタンスの数。

必須: はい

型: 整数

`min_size`

この NodeGroup のインスタンスの最小数。

必須: はい

型: 整数

`max_size`

この NodeGroup のインスタンスの最大数。

必須: はい

型: 整数

プロパティ

`node_role`

Amazon EC2 インスタンスにアタッチされた IAM ロールの ARN。

必須: はい

タイプ: 文字列

`tags`

リソースにアタッチするタグ。

必須: いいえ

タイプ: リスト

`kubernetes_version`

Managed Node group. AWS TNB の Kubernetes バージョンは、Kubernetes バージョン 1.27 から 1.34 をサポートしています。以下の点を考慮してください。

- `kubernetes_version` または を指定しますami_id。両方を指定することはできません。
- は、 AWS.Compute.EKSManagedNode バージョン以下kubernetes_versionである必要があります。
- AWS.Compute.EKSManagedNode バージョンと の間には 3 つのバージョンがありますkubernetes_version。
- `kubernetes_version` または が指定されていない場合ami_id、 AWS TNB はAWS.Compute.EKSManagedNode最新バージョンの AMI を使用して を作成します。EKSManagedNode

必須: いいえ

タイプ: 文字列

指定できる値: 1.27 | 1.28 | 1.29 | 1.30 | 1.31 | 1.32 | 1.33 | 1.34

要件

`cluster`

AWS.Compute.EKS ノード。

必須: はい

タイプ: 文字列

`subnets`

AWS.Networking.Subnet ノード。

必須: はい

タイプ: リスト

`network_interfaces`

AWS.Networking.ENI ノード。ネットワークインターフェイスとサブネットが同じアベイラビリティーゾーンに設定されていることを確認してください。異なる場合は、インスタンス化が失敗します。

を設定すると`network_interfaces`、AWS.Compute.EKS ノードに `multus_role`プロパティを含めると、 AWS TNB は `multus`プロパティから ENIs に関連するアクセス許可を取得します。それ以外の場合、 AWS TNB は node_role プロパティから ENI に関連するアクセス許可を取得します。

必須: いいえ

タイプ: リスト

`security_groups`

AWS.Networking.SecurityGroup ノード。

必須: いいえ

タイプ: リスト

`placement_group`

tosca.nodes.AWS.Compute.PlacementGroup ノード。

必須: いいえ

タイプ: 文字列

`user_data`

tosca.nodes.AWS.Compute.UserData ノードのリファレンス。ユーザーデータスクリプトは、マネージド型ノードグループによって起動される Amazon EC2 インスタンスに渡されます。カスタ

ムユーザーデータの実行に必要なアクセス許可を、ノードグループに渡される node_role に追加します。

必須: いいえ

タイプ: 文字列

`labels`

ノードラベルのリスト。ノードラベルには名前と値が必要です。次の条件を使用してラベルを作成します。

- 名前と値は で区切る必要があります=。
- 名前と値の長さはそれぞれ最大 63 文字です。
- ラベルには、文字 (A ~ Z、a ~ z)、数字 (0 ~ 9)、および次の文字を含めることができます。 `[-, _, ., *, ?]`
- 名前と値は、英数字、、?または * 文字で始まる必要があります。

例: `myLabelName1=*NodeLabelValue1`

必須: いいえ

タイプ: リスト

例

```
SampleEKSManagedNode:
  type: tosca.nodes.AWS.Compute.EKSManagedNode
  capabilities:
    compute:
      properties:
        ami_type: "AL2_x86_64"
        instance_types:
          - "t3.xlarge"
        key_pair: "SampleKeyPair"
        root_volume_encryption: true
        root_volume_encryption_key_arn: "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        root_volume_size: 1500
    scaling:
      properties:
```

```
        desired_size: 1
        min_size: 1
        max_size: 1
  properties:
    node_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleRole"
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
    kubernetes_version:
      - "1.30"
  requirements:
    cluster: SampleEKS
    subnets:
      - SampleSubnet
    network_interfaces:
      - SampleENI01
      - SampleENI02
    security_groups:
      - SampleSecurityGroup01
      - SampleSecurityGroup02
    placement_group: SamplePlacementGroup
    user_data: CustomUserData
    labels:
      - "sampleLabelName001=sampleLabelValue001"
      - "sampleLabelName002=sampleLabelValue002"
```

AWS.Compute.EKSSelfManagedNode

AWS TNB は Amazon EKS セルフマネージドノードをサポートし、Amazon EKS Kubernetes クラスターのノード (Amazon EC2 インスタンス) のプロビジョニングとライフサイクル管理を自動化します。Amazon EKS ノードグループを作成するには、次の手順を実行します。

- AMI の ID のいずれかを指定して、クラスターワーカーノードの Amazon マシンイメージ (AMI) を選択します。
- SSH アクセス用の Amazon EC2 キーペアを指定します。
- ノードグループが Amazon EKS クラスターに関連付けられていることを確認します。
- インスタンスタイプ、必要なサイズ、最小サイズ、最大サイズを指定します。
- ワーカーノードのサブネットを指定します。
- 必要に応じて、セキュリティグループ、ノードラベル、プレイスメントグループをノードグループにアタッチします。

構文

```
tosca.nodes.AWS.Compute.EKSSelfManagedNode:
  capabilities:
    compute:
      properties:
        ami_id: String
        instance_type: String
        key_pair: String
        root_volume_encryption: Boolean
        root_volume_encryption_key_arn: String
        root_volume_size: Integer
    scaling:
      properties:
        desired_size: Integer
        min_size: Integer
        max_size: Integer
  properties:
    node_role: String
    tags: List
  requirements:
    cluster: String
    subnets: List
    network_interfaces: List
    security_groups: List
    placement_group: String
    user_data: String
    labels: List
```

機能

compute

Amazon EKS セルフマネージド型ノードのコンピューティングパラメータを定義するプロパティ (Amazon EC2 インスタンスタイプや Amazon EC2 インスタンス AMI など)。

`ami_id`

インスタンスの起動に使用される AMI ID。 AWS TNB はIMDSv2 を利用するインスタンスをサポートします。詳細については、「IMDS バージョン」を参照してください。

> **Note**
>
> の AMI ID を更新できますEKSSelfManagedNode。AMI の Amazon EKS バージョンは、Amazon EKS クラスターバージョンと同じか、最大 2 つ前のバージョンである必要があります。例えば、Amazon EKS クラスターバージョンが 1.31 の場合、Amazon EKS AMI バージョンは 1.31、1.30、または 1.29 である必要があります。

必須: はい

タイプ: 文字列

`instance_type`

インスタンスのサイズ。

必須: はい

タイプ: 文字列

`key_pair`

SSH アクセスを有効にする Amazon EC2 キーペア。

必須: はい

タイプ: 文字列

`root_volume_encryption`

Amazon EBS ルートボリュームの Amazon EBS 暗号化を有効にします。このプロパティが指定されていない場合、 AWS TNB はデフォルトで Amazon EBS ルートボリュームを暗号化します。

必須: いいえ

デフォルト: true

タイプ: ブール値

`root_volume_encryption_key_arn`

AWS KMS key. AWS TNB の ARN は、通常のキー ARN、マルチリージョンキー ARN、エイリアス ARN をサポートしています。

必須: いいえ

タイプ: 文字列

> **Note**
>
> - `root_volume_encryption` が false の場合、 を含めないでくださいいroot_volume_encryption_key_arn。
> - AWS TNB は、Amazon EBS-backed AMI のルートボリューム暗号化をサポートしています。
> - AMI のルートボリュームがすでに暗号化されている場合は、 AWS TTB root_volume_encryption_key_arnがルートボリュームを再暗号化するための を含める必要があります。
> - AMI のルートボリュームが暗号化されていない場合、 AWS TNB は root_volume_encryption_key_arn を使用してルートボリュームを暗号化します。
>
> を含めない場合root_volume_encryption_key_arn、 AWS TNB は AWS Managed Services を使用してルートボリュームを暗号化します。
> - AWS TNB は暗号化された AMI を復号しません。

`root_volume_size`

Amazon Elastic Block Store ルートボリュームのサイズを GiBs。

必須: いいえ

デフォルト: 20

タイプ: 整数

指定できる値: 1 ~ 16,384

scaling

Amazon EKS セルフマネージド型ノードのスケーリングパラメータ (必要な Amazon EC2 インスタンスの数、ノードグループ内の Amazon EC2 インスタンスの最小数と最大数など) を定義するプロパティ。

`desired_size`

この NodeGroup のインスタンスの数。

必須: はい

型: 整数

`min_size`

この NodeGroup のインスタンスの最小数。

必須: はい

型: 整数

`max_size`

この NodeGroup のインスタンスの最大数。

必須: はい

型: 整数

プロパティ

`node_role`

Amazon EC2 インスタンスにアタッチされた IAM ロールの ARN。

必須: はい

タイプ: 文字列

`tags`

リソースにアタッチするタグ。タグは、リソースによって作成されたインスタンスに伝播されます。

必須: いいえ

タイプ: リスト

要件

`cluster`

AWS.Compute.EKS ノード。

必須: はい

タイプ: 文字列

`subnets`

AWS.Networking.Subnet ノード。

必須: はい

タイプ: リスト

`network_interfaces`

AWS.Networking.ENI ノード。ネットワークインターフェイスとサブネットが同じアベイラビリティーゾーンに設定されていることを確認してください。異なる場合は、インスタンス化が失敗します。

を設定すると`network_interfaces`、AWS.Compute.EKS ノードに `multus_role`プロパティを含めると、 AWS TNB は `multus`プロパティから ENIs に関連するアクセス許可を取得します。それ以外の場合、 AWS TNB は node_role プロパティから ENI に関連するアクセス許可を取得します。

必須: いいえ

タイプ: リスト

`security_groups`

AWS.Networking.SecurityGroup ノード。

必須: いいえ

タイプ: リスト

`placement_group`

tosca.nodes.AWS.Compute.PlacementGroup ノード。

必須: いいえ

タイプ: 文字列

`user_data`

tosca.nodes.AWS.Compute.UserData ノードのリファレンス。ユーザーデータスクリプトは、セルフマネージド型ノードグループによって起動された Amazon EC2 インスタンスに渡されます。カスタムユーザーデータの実行に必要なアクセス許可を、ノードグループに渡される node_role に追加します。

必須: いいえ

タイプ: 文字列

`labels`

ノードラベルのリスト。ノードラベルには名前と値が必要です。次の条件を使用してラベルを作成します。

- 名前と値は で区切る必要があります=。
- 名前と値の長さはそれぞれ最大 63 文字です。
- ラベルには、文字 (A ~ Z、a ~ z)、数字 (0 ~ 9)、および次の文字を含めることができます。 `[-, _, ., *, ?]`
- 名前と値は、英数字、、?または * 文字で始まる必要があります。

例: `myLabelName1=*NodeLabelValue1`

必須: いいえ

タイプ: リスト

例

```
SampleEKSSelfManagedNode:
  type: tosca.nodes.AWS.Compute.EKSSelfManagedNode
  capabilities:
    compute:
      properties:
        ami_id: "ami-123123EXAMPLE"
        instance_type: "c5.large"
        key_pair: "SampleKeyPair"
        root_volume_encryption: true
```

```
        root_volume_encryption_key_arn: "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        root_volume_size: 1500
    scaling:
      properties:
        desired_size: 1
        min_size: 1
        max_size: 1
  properties:
    node_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleNodeRole"
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
  requirements:
    cluster: SampleEKSCluster
    subnets:
      - SampleSubnet
    network_interfaces:
      - SampleNetworkInterface01
      - SampleNetworkInterface02
    security_groups:
      - SampleSecurityGroup01
      - SampleSecurityGroup02
    placement_group: SamplePlacementGroup
    user_data: CustomUserData
    labels:
      - "sampleLabelName001=sampleLabelValue001"
      - "sampleLabelName002=sampleLabelValue002"
```

AWS.Compute.PlacementGroup

PlacementGroup ノードは、Amazon EC2 インスタンスを配置するためのさまざまな戦略をサポートします。

新しい Amazon EC2 インスタンスを起動する場合、Amazon EC2 サービスは、相関性のエラーを最小限に抑えるために、すべてのインスタンスが基盤となるハードウェアに分散されるようにインスタンスを配置します。プレイスメントグループを使用することで、ワークロードのニーズに対応するために独立したインスタンスのグループのプレイスメントに影響を与えることができます。

構文

```
tosca.nodes.AWS.Compute.PlacementGroup
```

```
  properties:
    strategy: String
    partition_count: Integer
    tags: List
```

プロパティ

`strategy`

Amazon EC2 インスタンスを配置するために使用する戦略。

必須: はい

タイプ: 文字列

使用できる値: CLUSTER | PARTITION | SPREAD_HOST | SPREAD_RACK

- CLUSTER – アベイラビリティーゾーン内でインスタンスをまとめます。この戦略により、ワークロードは、ハイパフォーマンスコンピューティング (HPC) アプリケーションで典型的な緊密に組み合わされたノード間通信に必要な低レイテンシーネットワークパフォーマンスを実現できます。
- PARTITION – インスタンスを複数の論理パーティションに分散させ、1 つのパーティション内のインスタンスのグループが基盤となるハードウェアを別のパーティション内のインスタンスのグループと共有しないようにします。この戦略は、Hadoop、Cassandra、Kafka などの大規模な分散および複製ワークロードで一般的に使用されます。
- SPREAD_RACK – 相関性のエラーを減らすために、少数のインスタンスを基盤となるハードウェア全体に配置します。
- SPREAD_HOST – Outpost の配置グループとのみ使用できます。相関性のエラーを減らすために、少数のインスタンスを基盤となるハードウェア全体に配置します。

`partition_count`

パーティション数。

必須: `strategy` が `PARTITION` に設定されている場合のみ必須です。

タイプ: 整数

使用できる値: 1 | 2 |3 | 4 | 5 | 6 | 7

`tags`

配置グループリソースにアタッチできるタグ。

必須: いいえ

タイプ: リスト

例

```
ExamplePlacementGroup:
  type: tosca.nodes.AWS.Compute.PlacementGroup
  properties:
    strategy: "PARTITION"
    partition_count: 5
    tags:
      - tag_key=tag_value
```

AWS.Compute.UserData

AWS TNB は、Network Service Descriptor (NSD) の UserData ノードを介したカスタムユーザーデータを使用した Amazon EC2 インスタンスの起動をサポートしています。カスタムユーザーデータの詳細については、Amazon EC2 ユーザーガイド」の「ユーザーデータとシェルスクリプト」を参照してください。

ネットワークのインスタンス化中、 AWS TNB はユーザーデータスクリプトを使用して Amazon EC2 インスタンス登録をクラスターに提供します。カスタムユーザーデータも提供されると、 AWS TNB は両方のスクリプトをマージし、マルチミースクリプトとして Amazon EC2 に渡します。カスタムユーザーデータスクリプトは、Amazon EKS 登録スクリプトの前に実行されます。

ユーザーデータスクリプトでカスタム変数を使用するには、開く中括弧 { の後ろに感嘆符 ! を追加します。例えば、スクリプトで `MyVariable` を使用するには、「`{!MyVariable}`」のように入力します。

> **Note**
>
> - AWS TNB は、最大 7 KB のサイズのユーザーデータスクリプトをサポートします。
> - AWS TNB は CloudFormation を使用して`multimime`ユーザーデータスクリプトを処理およびレンダリングするため、スクリプトがすべての CloudFormation ルールに準拠していることを確認します。

構文

```
tosca.nodes.AWS.Compute.UserData:
  properties:
    implementation: String
    content_type: String
```

プロパティ

`implementation`

ユーザーデータスクリプト定義への相対パス。形式は `./scripts/script_name.sh` にする必要があります。

必須: はい

タイプ: 文字列

`content_type`

ユーザーデータスクリプトのコンテンツ型。

必須: はい

タイプ: 文字列

使用できる値: `x-shellscript`

例

```
ExampleUserData:
  type: tosca.nodes.AWS.Compute.UserData
  properties:
    content_type: "text/x-shellscript"
    implementation: "./scripts/customUserData.sh"
```

AWS.Networking.SecurityGroup

AWS TNB は、Amazon EKS Kubernetes クラスターノードグループにアタッチできる Amazon EC2 セキュリティグループのプロビジョニングを自動化するセキュリティグループをサポートしています。

構文

```
tosca.nodes.AWS.Networking.SecurityGroup
  properties:
    description: String
    name: String
    tags: List
  requirements:
    vpc: String
```

プロパティ

`description`

セキュリティグループの説明。グループの説明には最大 255 文字を使用できます。文字 (A ~ Z および a ~ z)、数字 (0 ~ 9)、スペースおよび特殊文字 (._-:/()#,@[]+=&;{}!$*) のみが使用できます。

必須: はい

タイプ: 文字列

`name`

セキュリティグループの名前。名前には最大 255 文字を使用できます。文字 (A ~ Z および a ~ z)、数字 (0 ~ 9)、スペースおよび特殊文字 (._-:/()#,@[]+=&;{}!$*) のみが使用できます。

必須: はい

タイプ: 文字列

`tags`

セキュリティグループリソースにアタッチできるタグ。

必須: いいえ

タイプ: リスト

要件

`vpc`

AWS.Networking.VPC ノード。

必須: はい

タイプ: 文字列

例

```
SampleSecurityGroup001:
  type: tosca.nodes.AWS.Networking.SecurityGroup
  properties:
        description: "Sample Security Group for Testing"
        name: "SampleSecurityGroup"
        tags:
          - "Name=SecurityGroup"
          - "Environment=Testing"
      requirements:
        vpc: SampleVPC
```

AWS.Networking.SecurityGroupEgressRule

AWS TNB は、.Networking AWS.SecurityGroup にアタッチできる Amazon EC2 セキュリティグループ Egress ルールのプロビジョニングを自動化するセキュリティグループ Egress ルールをサポートしています。エグレストラフィックの宛先として cidr_ip/destination_security_group/destination_prefix_list を指定する必要があることに注意してください。

構文

```
AWS.Networking.SecurityGroupEgressRule
  properties:
    ip_protocol: String
    from_port: Integer
    to_port: Integer
    description: String
    destination_prefix_list: String
    cidr_ip: String
    cidr_ipv6: String
  requirements:
    security_group: String
    destination_security_group: String
```

プロパティ

`cidr_ip`

CIDR 形式の IPv4 アドレス範囲。エグレストラフィックを許可する CIDR 範囲を指定する必要があります。

必須: いいえ

タイプ: 文字列

`cidr_ipv6`

出力トラフィック用の CIDR 形式の IPv6 アドレス範囲。対象セキュリティグループ (`destination_security_group` または `destination_prefix_list`) または CIDR 範囲 (`cidr_ip` または `cidr_ipv6`)。

必須: いいえ

タイプ: 文字列

`description`

Egress (送信) セキュリティグループルールの説明。ルールの説明には最大 255 文字を使用できます。

必須: いいえ

タイプ: 文字列

`destination_prefix_list`

既存の Amazon VPC マネージドプレフィックスリストのプレフィックスリスト ID。これは、セキュリティグループに関連付けられたノードグループインスタンスからの送信先です。詳細については、「Amazon VPC ユーザーガイド」の「マネージドプレフィックスリスト」を参照してください。

必須: いいえ

タイプ: 文字列

`from_port`

プロトコルが TCP または UDP の場合、これはポート範囲の始点になります。プロトコルが ICMP または ICMPv6 の場合、これはタイプ番号です。-1 の値はすべての ICMP/ICMPv6 タイプ

を示します。すべての ICMP/ICMPv6 タイプを指定した場合、すべての ICMP/ICMPv6 コードを指定する必要があります。

必須: いいえ

タイプ: 整数

`ip_protocol`

IP プロトコル名 (tcp、udp、icmp、icmpv6) またはプロトコル番号。-1 を使用してすべてのプロトコルを指定します。セキュリティグループルールを許可するときに、-1 または tcp、udp、icmp や icmpv6 以外のプロトコル番号を指定すると、指定したポート範囲に関係なく、すべてのポートでトラフィックが許可されます。tcp、udp、および icmp には、ポート範囲を指定する必要があります。icmpv6 の場合、ポート範囲はオプションです。ポート範囲を省略すると、すべてのタイプとコードのトラフィックが許可されます。

必須: はい

タイプ: 文字列

`to_port`

プロトコルが TCP または UDP の場合、これはポート範囲の終わりになります。プロトコルが ICMP または ICMPv6 の場合、これはコードです。-1 の値はすべての ICMP/ICMPv6 コードを示します。すべての ICMP/ICMPv6 タイプを指定した場合、すべての ICMP/ICMPv6 コードを指定する必要があります。

必須: いいえ

タイプ: 整数

要件

`security_group`

このルールを追加するセキュリティグループの ID。

必須: はい

タイプ: 文字列

`destination_security_group`

エグレストラフィックが許可される宛先セキュリティグループの ID または TOSCA リファレンス。

必須: いいえ

タイプ: 文字列

例

```
SampleSecurityGroupEgressRule:
      type: tosca.nodes.AWS.Networking.SecurityGroupEgressRule
      properties:
        ip_protocol: "tcp"
        from_port: 8000
        to_port: 9000
        description: "Egress Rule for sample security group"
        cidr_ipv6: "2600:1f14:3758:ca00::/64"
      requirements:
        security_group: SampleSecurityGroup001
        destination_security_group: SampleSecurityGroup002
```

AWS.Networking.SecurityGroupIngressRule

AWS TNB は、.Networking AWS.SecurityGroup にアタッチできる Amazon EC2 セキュリティグループイングレスルールのプロビジョニングを自動化するセキュリティグループイングレスルールをサポートしています。入力トラフィックのソースとして cidr_ip/source_security_group/source_prefix_list を指定する必要があることに注意してください。

構文

```
AWS.Networking.SecurityGroupIngressRule
  properties:
    ip_protocol: String
    from_port: Integer
    to_port: Integer
    description: String
    source_prefix_list: String
    cidr_ip: String
    cidr_ipv6: String
```

```
  requirements:
    security_group: String
    source_security_group: String
```

プロパティ

`cidr_ip`

CIDR 形式の IPv4 アドレス範囲。入力トラフィックを許可する CIDR 範囲を指定する必要があります。

必須: いいえ

タイプ: 文字列

`cidr_ipv6`

入力トラフィック用の CIDR 形式の IPv6 アドレス範囲。ソースセキュリティグループ (`source_security_group` または `source_prefix_list`) または CIDR 範囲 (`cidr_ip` または `cidr_ipv6`)。

必須: いいえ

タイプ: 文字列

`description`

入力 (受信) セキュリティグループルールの説明。ルールの説明には最大 255 文字を使用できます。

必須: いいえ

タイプ: 文字列

`source_prefix_list`

既存の Amazon VPC マネージドプレフィックスリストのプレフィックスリスト ID。このソースから、セキュリティグループに関連付けられているノードグループインスタンスがトラフィックを受信できます。詳細については、「Amazon VPC ユーザーガイド」の「マネージドプレフィックスリスト」を参照してください。

必須: いいえ

タイプ: 文字列

`from_port`

プロトコルが TCP または UDP の場合、これはポート範囲の始点になります。プロトコルが ICMP または ICMPv6 の場合、これはタイプ番号です。-1 の値はすべての ICMP/ICMPv6 タイプを示します。すべての ICMP/ICMPv6 タイプを指定した場合、すべての ICMP/ICMPv6 コードを指定する必要があります。

必須: いいえ

タイプ: 整数

`ip_protocol`

IP プロトコル名 (tcp、udp、icmp、icmpv6) またはプロトコル番号。-1 を使用してすべてのプロトコルを指定します。セキュリティグループルールを許可するときに、-1 または tcp、udp、icmp や icmpv6 以外のプロトコル番号を指定すると、指定したポート範囲に関係なく、すべてのポートでトラフィックが許可されます。tcp、udp、および icmp には、ポート範囲を指定する必要があります。icmpv6 の場合、ポート範囲はオプションです。ポート範囲を省略すると、すべてのタイプとコードのトラフィックが許可されます。

必須: はい

タイプ: 文字列

`to_port`

プロトコルが TCP または UDP の場合、これはポート範囲の終わりになります。プロトコルが ICMP または ICMPv6 の場合、これはコードです。-1 の値はすべての ICMP/ICMPv6 コードを示します。すべての ICMP/ICMPv6 タイプを指定した場合、すべての ICMP/ICMPv6 コードを指定する必要があります。

必須: いいえ

タイプ: 整数

要件

`security_group`

このルールを追加するセキュリティグループの ID。

必須: はい

タイプ: 文字列

`source_security_group`

入力トラフィックを許可するソースセキュリティグループの ID または TOSCA リファレンス。

必須: いいえ

タイプ: 文字列

例

```
SampleSecurityGroupIngressRule:
      type: tosca.nodes.AWS.Networking.SecurityGroupIngressRule
      properties:
        ip_protocol: "tcp"
        from_port: 8000
        to_port: 9000
        description: "Ingress Rule for free5GC cluster on IPv6"
        cidr_ipv6: "2600:1f14:3758:ca00::/64"
      requirements:
        security_group: SampleSecurityGroup1
        source_security_group: SampleSecurityGroup2
```

AWS.Resource.Import

次の AWS リソースを AWS TNB にインポートできます。

- VPC
- サブネット
- ルートテーブル
- インターネットゲートウェイ
- セキュリティグループ

構文

```
tosca.nodes.AWS.Resource.Import
  properties:
    resource_type: String
```

```
    resource_id: String
```

プロパティ

`resource_type`

AWS TNB にインポートされるリソースタイプ。

必須: いいえ

タイプ: リスト

`resource_id`

AWS TNB にインポートされるリソース ID。

必須: いいえ

タイプ: リスト

例

```
SampleImportedVPC:
  type: tosca.nodes.AWS.Resource.Import
  properties:
    resource_type: "tosca.nodes.AWS.Networking.VPC"
    resource_id: "vpc-123456"
```

AWS.Networking.ENI

ネットワークインターフェイスは、仮想ネットワークカードを表す VPC 内の論理ネットワーキングコンポーネントです。ネットワークインターフェイスには、サブネットに基づいて自動または手動で IP アドレスが割り当てられます。サブネットに Amazon EC2 インスタンスをデプロイしたら、そのインスタンスにネットワークインターフェイスをアタッチするか、その Amazon EC2 インスタンスからネットワークインターフェイスをデタッチして、そのサブネット内の別の Amazon EC2 インスタンスに再アタッチできます。デバイスインデックスは、アタッチの順番における位置を識別します。

構文

```
tosca.nodes.AWS.Networking.ENI:
```

```
  properties:
    device_index: Integer
    source_dest_check: Boolean
    tags: List
  requirements:
    subnet: String
    security_groups: List
```

プロパティ

`device_index`

デバイスインデックスはゼロより大きくする必要があります。

必須: はい

型: 整数

`source_dest_check`

ネットワークインターフェイスが送信元/送信先チェックを実行するかどうかを示します。`true` はチェックが有効であることを示し、`false` は無効であることを示します。

許可される値: true、false

デフォルト: true

必須: いいえ

型: ブール

`tags`

リソースにアタッチするタグ。

必須: いいえ

タイプ: リスト

要件

`subnet`

AWS.Networking.Subnet ノード。

必須: はい

タイプ: 文字列

`security_groups`

AWS.Networking.SecurityGroup ノード。

必須: いいえ

タイプ: 文字列

例

```
SampleENI:
  type: tosca.nodes.AWS.Networking.ENI
  properties:
    device_index: 5
    source_dest_check: true
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
  requirements:
    subnet: SampleSubnet
    security_groups:
      - SampleSecurityGroup01
      - SampleSecurityGroup02
```

AWS.HookExecution

ライフサイクルフックを使用すると、インフラストラクチャやネットワークのインスタンス化の一環として独自のスクリプトを実行できます。

構文

```
tosca.nodes.AWS.HookExecution:
  capabilities:
    execution:
      properties:
        type: String
  requirements:
    definition: String
```

```
    vpc: String
```

機能

execution

フックスクリプトを実行するフック実行エンジンのプロパティ。

type

フック実行エンジンのタイプ。

必須: いいえ

タイプ: 文字列

使用できる値: CODE_BUILD

要件

definition

AWS.HookDefinition.Bash ノード。

必須: はい

タイプ: 文字列

vpc

AWS.Networking.VPC ノード。

必須: はい

タイプ: 文字列

例

```
SampleHookExecution:
  type: tosca.nodes.AWS.HookExecution
  requirements:
    definition: SampleHookScript
```

```
    vpc: SampleVPC
```

AWS.Networking.InternetGateway

AWS インターネットゲートウェイノードを定義します。

構文

```
tosca.nodes.AWS.Networking.InternetGateway:
  capabilities:
    routing:
      properties:
        dest_cidr: String
        ipv6_dest_cidr: String
  properties:
    tags: List
    egress_only: Boolean
  requirements:
    vpc: String
    route_table: String
```

機能

routing

VPC 内のルーティング接続を定義するプロパティ。dest_cidr または ipv6_dest_cidr プロパティのいずれかを含める必要があります。

dest_cidr

ルーティング先の照合に使用する IPv4 CIDR ブロック。このプロパティは RouteTable でルートを作成するのに使用され、その値は DestinationCidrBlock として使用されます。

必須: ipv6_dest_cidr プロパティを含めた場合は「いいえ」。

タイプ: 文字列

ipv6_dest_cidr

ルーティング先の照合に使用する IPv6 CIDR ブロック。

必須: dest_cidr プロパティを含めた場合は「いいえ」。

タイプ: 文字列

プロパティ

`tags`

リソースにアタッチするタグ。

必須: いいえ

タイプ: リスト

`egress_only`

IPv6 固有のプロパティ。インターネットゲートウェイが出力通信専用かどうかを示します。`egress_only` が true の場合は、`ipv6_dest_cidr` プロパティを定義する必要があります。

必須: いいえ

型: ブール

要件

`vpc`

AWS.Networking.VPC ノード。

必須: はい

タイプ: 文字列

`route_table`

AWS.Networking.RouteTable ノード。

必須: はい

タイプ: 文字列

例

```
Free5GCIGW:
```

```
  type: tosca.nodes.AWS.Networking.InternetGateway
  properties:
    egress_only: false
  capabilities:
    routing:
      properties:
        dest_cidr: "0.0.0.0/0"
        ipv6_dest_cidr: "::/0"
  requirements:
    route_table: Free5GCRouteTable
    vpc: Free5GCVPC
Free5GCEGW:
  type: tosca.nodes.AWS.Networking.InternetGateway
  properties:
    egress_only: true
  capabilities:
    routing:
      properties:
        ipv6_dest_cidr: "::/0"
  requirements:
    route_table: Free5GCPrivateRouteTable
    vpc: Free5GCVPC
```

AWS.Networking.RouteTable

ルートテーブルには、VPC またはゲートウェイ内のサブネットからのネットワークトラフィックの経路を判断する、ルートと呼ばれる一連のルールが含まれます。ルートテーブルを VPC に関連付ける必要があります。

構文

```
tosca.nodes.AWS.Networking.RouteTable:
  properties:
    tags: List
  requirements:
    vpc: String
```

プロパティ

`tags`

このリソースにアタッチするタグ。

必須: いいえ

タイプ: リスト

要件

vpc

AWS.Networking.VPC ノード。

必須: はい

タイプ: 文字列

例

```
SampleRouteTable:
  type: tosca.nodes.AWS.Networking.RouteTable
  properties:
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
  requirements:
    vpc: SampleVPC
```

AWS.Networking.Subnet

サブネットは、VPC の IP アドレスの範囲で、全体が 1 つのアベイラビリティーゾーンに存在する必要があります。サブネットの VPC、CIDR ブロック、アベイラビリティーゾーン、およびルートテーブルを指定する必要があります。また、サブネットがプライベートかパブリックかを定義する必要もあります。

構文

```
tosca.nodes.AWS.Networking.Subnet:
  properties:
    type: String
    availability_zone: String
    cidr_block: String
    ipv6_cidr_block: String
```

```
    ipv6_cidr_block_suffix: String
    outpost_arn: String
    tags: List
  requirements:
    vpc: String
    route_table: String
```

プロパティ

`type`

このサブネットで起動されたインスタンスがパブリック IPv4 アドレスを受け取るかどうかを示します。

必須: はい

タイプ: 文字列

使用できる値: PUBLIC | PRIVATE

`availability_zone`

サブネットのアベイラビリティーゾーン。このフィールドは、 AWS リージョン内の AWS アベイラビリティーゾーン (米国西部 us-west-2 (オレゴン) など) をサポートします。また、 など、アベイラビリティーゾーン内の AWS ローカルゾーンもサポートしていますus-west-2-lax-1a。

必須: はい

タイプ: 文字列

`cidr_block`

サブネットの CIDR ブロック。

必須: いいえ

タイプ: 文字列

`ipv6_cidr_block`

IPv6 サブネットの作成に使用される CIDR ブロック。このプロパティを含める場合は、ipv6_cidr_block_suffix を含めないでください。

必須: いいえ

タイプ: 文字列

`ipv6_cidr_block_suffix`

Amazon VPC 上で作成されたサブネットの IPv6 CIDR ブロックの、2 桁の 16 進数のサフィックス。次の形式を使用します。*`2-digit hexadecimal`*`::/`*`subnetMask`*

このプロパティを含める場合は、`ipv6_cidr_block` を含めないでください。

必須: いいえ

タイプ: 文字列

`outpost_arn`

サブネット AWS Outposts が作成される の ARN。Amazon EKS セルフマネージド型ノードを AWS Outpostsで起動する場合は、このプロパティを NSD テンプレートに追加します。詳細については、「Amazon EKS ユーザーガイド」の「AWS Outpostsにおける Amazon EKS」を参照してください。

いこのプロパティを NSD テンプレートに追加する場合、`availability_zone` プロパティの値を AWS Outpostsのアベイラビリティーゾーンに設定する必要があります。

必須: いいえ

タイプ: 文字列

`tags`

リソースにアタッチするタグ。

必須: いいえ

タイプ: リスト

要件

`vpc`

AWS.Networking.VPC ノード。

必須: はい

タイプ: 文字列

`route_table`

AWS.Networking.RouteTable ノード。

必須: はい

タイプ: 文字列

例

```
SampleSubnet01:
  type: tosca.nodes.AWS.Networking.Subnet
  properties:
    type: "PUBLIC"
    availability_zone: "us-east-1a"
    cidr_block: "10.100.50.0/24"
    ipv6_cidr_block_suffix: "aa::/64"
    outpost_arn: "arn:aws:outposts:region:accountId:outpost/op-11223344EXAMPLE"
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
  requirements:
    vpc: SampleVPC
    route_table: SampleRouteTable

SampleSubnet02:
  type: tosca.nodes.AWS.Networking.Subnet
  properties:
    type: "PUBLIC"
    availability_zone: "us-west-2b"
    cidr_block: "10.100.50.0/24"
    ipv6_cidr_block: "2600:1f14:3758:ca00::/64"
  requirements:
    route_table: SampleRouteTable
    vpc: SampleVPC
```

AWS.Deployment.VNFDeployment

NF デプロイは、それに関連するインフラストラクチャとアプリケーションを提供することでモデル化されます。cluster 属性は、NF をホストする EKS クラスターを指定します。vnfs 属性は、デプロイのネットワーク機能を指定します。また、オプションで pre_create と post_create タイプのライフ

サイクルフックオペレーションを提供し、インベントリ管理システム API の呼び出しなど、デプロイ固有の命令を実行することもできます。

構文

```
tosca.nodes.AWS.Deployment.VNFDeployment:
  requirements:
    deployment: String
    cluster: String
    vnfs: List
  interfaces:
    Hook:
      pre_create: String
      post_create: String
```

要件

`deployment`

AWS.Deployment.VNFDeployment ノード。

必須: いいえ

タイプ: 文字列

`cluster`

AWS.Compute.EKS ノード。

必須: はい

タイプ: 文字列

`vnfs`

AWS.VNF ノード。

必須: はい

タイプ: 文字列

インターフェイス

フック

ライフサイクルフックが実行されるステージを定義します。

`pre_create`

AWS.HookExecution ノード。このフックは VNFDeployment ノードがデプロイされる前に実行されます。

必須: いいえ

タイプ: 文字列

`post_create`

AWS.HookExecution ノード。このフックは VNFDeployment ノードがデプロイされた後に実行されます。

必須: いいえ

タイプ: 文字列

例

```
SampleHelmDeploy:
  type: tosca.nodes.AWS.Deployment.VNFDeployment
  requirements:
    deployment: SampleHelmDeploy2
    cluster: SampleEKS
    vnfs:
      - vnf.SampleVNF
  interfaces:
    Hook:
      pre_create: SampleHook
```

AWS.Networking.VPC

仮想プライベートクラウド (VPC) の CIDR ブロックを指定する必要があります。

構文

```
tosca.nodes.AWS.Networking.VPC:
  properties:
    cidr_block: String
    ipv6_cidr_block: String
    dns_support: String
    tags: List
```

プロパティ

`cidr_block`

VPC の IPv4 ネットワーク範囲 (CIDR 表記)。

必須: はい

タイプ: 文字列

`ipv6_cidr_block`

VPC の作成に使用される IPv6 CIDR ブロック。

許可される値: `AMAZON_PROVIDED`

必須: いいえ

タイプ: 文字列

`dns_support`

VPC 内に起動されるインスタンスが DNS ホスト名を取得するかどうかを示します。

必須: いいえ

型: ブール

デフォルト: `false`

`tags`

このリソースにアタッチするタグ。

必須: いいえ

タイプ: リスト

例

```
SampleVPC:
  type: tosca.nodes.AWS.Networking.VPC
  properties:
    cidr_block: "10.100.0.0/16"
    ipv6_cidr_block: "AMAZON_PROVIDED"
    dns_support: true
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
```

AWS.Networking.NATGateway

パブリックまたはプライベート NAT ゲートウェイノードをサブネット上で定義できます。パブリックゲートウェイの場合、Elastic IP 割り当て ID を指定しない場合、 AWS TNB はアカウントに Elastic IP を割り当て、ゲートウェイに関連付けます。

構文

```
tosca.nodes.AWS.Networking.NATGateway:
  requirements:
    subnet: String
    internet_gateway: String
  properties:
    type: String
    eip_allocation_id: String
    tags: List
```

プロパティ

`subnet`

AWS.Networking.Subnet ノードのリファレンス。

必須: はい

タイプ: 文字列

`internet_gateway`

AWS.Networking.InternetGateway ノードのリファレンス。

必須: はい

タイプ: 文字列

プロパティ

`type`

ゲートウェイがパブリックかプライベートかを示します。

許可される値: `PUBLIC`、`PRIVATE`

必須: はい

タイプ: 文字列

`eip_allocation_id`

Elastic IP アドレスの割り当てを表す ID。

必須: いいえ

タイプ: 文字列

`tags`

このリソースにアタッチするタグ。

必須: いいえ

タイプ: リスト

例

```
Free5GCNatGateway01:
  type: tosca.nodes.AWS.Networking.NATGateway
    requirements:
       subnet: Free5GCSubnet01
       internet_gateway: Free5GCIGW
    properties:
```

```
        type: PUBLIC
        eip_allocation_id: eipalloc-12345
```

AWS.Networking.Route

送信先ルートをターゲットリソースとして NAT Gateway に関連付け、関連付けられたルートテーブルにルートを追加するルートノードを定義できます。

構文

```
tosca.nodes.AWS.Networking.Route:
  properties:
    dest_cidr_blocks: List
  requirements:
    nat_gateway: String
    route_table: String
```

プロパティ

`dest_cidr_blocks`

ターゲットリソースへの送信先 IPv4 ルートのリスト。

必須: はい

タイプ: リスト

メンバー型: 文字列

要件

`nat_gateway`

AWS.Networking.NATGateway ノードのリファレンス。

必須: はい

タイプ: 文字列

`route_table`

AWS.Networking.RouteTable ノードのリファレンス。

必須: はい

タイプ: 文字列

例

```
Free5GCRoute:
  type: tosca.nodes.AWS.Networking.Route
  properties:
    dest_cidr_blocks:
      - 0.0.0.0/0
      - 10.0.0.0/28
  requirements:
    nat_gateway: Free5GCNatGateway01
    route_table: Free5GCRouteTable
```

AWS.Store.SSMParameters

TNB を使用して SSM AWS パラメータを作成できます。作成する SSM パラメータは SSM で作成され、 AWS TTB ネットワークインスタンス ID がプレフィックスとして付けられます。これにより、同じ NSD テンプレートを使用して複数のインスタンスがインスタンス化およびアップグレードされた場合に、パラメータ値が上書きされるのを防ぎます。

構文

```
tosca.nodes.AWS.Store.SSMParameters
  properties:
    parameters:
      name: String
      value: String
    tags: List
```

プロパティ

パラメータ

name

ssm プロパティの名前。次の形式を使用します。^[a-zA-Z0-9]+[a-zA-Z0-9\-_]*[a-zA-Z0-9]+$

各パラメータの名前は 256 文字未満にする必要があります。

必須: はい

タイプ: 文字列

`value`

ssm プロパティの値。以下のいずれかの形式を使用します。

- 参照のない値の場合: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`
- 静的参照の場合: `^\$\{[a-zA-Z0-9]+\.(properties|capabilities|requirements)(\.([a-zA-Z0-9\-_]+))+\}$`
- 動的参照の場合: `^\$\{[a-zA-Z0-9]+\.(name|id|arn)\}$`

各パラメータの値は 4 KB 未満である必要があります。

必須: はい

タイプ: 文字列

`tags`

SSM プロパティにアタッチできるタグ。

必須: いいえ

タイプ: リスト

例

```
SampleSSM
    type: tosca.nodes.AWS.Store.SSMParameters
    properties:
        parameters:
            - name: "Name1"
              value: "Value1"
            - name: "EKS_VERSION"
              value: "${SampleEKS.properties.version}"
            - name: "VPC_ID"
              value: "${SampleVPC.id}
            - name: "REGION"
```

```
                value: "${AWS::Region}
        tags:
            - "tagKey=tagValue"
```

一般的なノード

NSD と VNFD のノードを定義します。

- AWS.HookDefinition.Bash

AWS.HookDefinition.Bash

で an AWS HookDefinition を定義しますbash。

構文

```
tosca.nodes.AWS.HookDefinition.Bash:
  properties:
    implementation: String
    environment_variables: List
    execution_role: String
```

プロパティ

`implementation`

フック定義への相対パス。形式は `./hooks/`*`script_name`*`.sh` にする必要があります。

必須: はい

タイプ: 文字列

`environment_variables`

フック Bash スクリプトの環境変数。次の形式を使用します。正規表現パターン**envName**=**envValue**は次のとおりです。

- 参照のない値の場合: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`
- 静的参照の場合: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=\$\{[a-zA-Z0-9]+\.(properties|capabilities|requirements)(\.([a-zA-Z0-9\-_]+))+\}$`

- 動的参照の場合: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=\$\{[a-zA-Z0-9]+\.(name|id|arn)\}$`

envName=envValue の値が次の基準を満たしていることを確認します。

- スペースは使用しません。
- **envName** の先頭には文字 (A ~ Z または a ~ z) または数字 (0 ~ 9) を使用します。
- 環境変数名の先頭に次の AWS TNB 予約キーワードを使用しないでください (大文字と小文字は区別されません)。
  - CODEBUILD
  - TNB
  - HOME
  - AWS
- **envName** と **envValue** には、任意の数の文字 (A ~ Z または a ~ z)、数字 (0 ~ 9)、および特殊文字 (- と _) を使用できます。
- 各環境変数 (各 **envName=envValue**) は 128 文字未満である必要があります。

例: `A123-45xYz=Example_789`

必須: いいえ

タイプ: リスト

`execution_role`

フック実行のロール。

必須: はい

タイプ: 文字列

例

```
SampleHookScript:
  type: tosca.nodes.AWS.HookDefinition.Bash
  properties:
    implementation: "./hooks/myhook.sh"
    environment_variables:
      - "variable01=value01"
      - "variable02=value02"
```

```
    execution_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleHookPermission"
```

AWS TNB のセキュリティ

でのクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャからメリットを得られます。

セキュリティは、 AWS お客様とお客様の間の責任共有です。責任共有モデルではこれをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS は、 で AWS サービスを実行するインフラストラクチャを保護する責任を担います AWS クラウド。 は、お客様が安全に使用できるサービス AWS も提供します。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。 AWS Telco Network Builder に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する AWS サービスによって決まります。また、ユーザーは、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、 AWS TNB を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。以下のトピックでは、セキュリティとコンプライアンスの目的を達成するように AWS TNB を設定する方法について説明します。また、 AWS TNB リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

内容

AWS TNB でのデータ保護

責任 AWS 共有モデル、Telco Network Builder AWS でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「Data Privacy FAQChina」を参照してください。 欧州におけるデータ保護に関する情報については、General Data Protection Regulation (GDPR) Center を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM アイデンティティセンターまたは AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須ですが、TLS 1.3 を推奨します。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して AWS TNB AWS CLIまたは他の AWS のサービス を使用する場合も同様です。AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

データの処理

AWS アカウントを閉じると、 AWS TNB はデータを削除対象としてマークし、あらゆる使用から削除します。90 日以内に AWS アカウントを再アクティブ化すると、 AWS TNB はデータを復元します。120 日後、 AWS TNB はデータを完全に削除します。 AWS TNB はネットワークを終了し、関数パッケージとネットワークパッケージも削除します。

保管中の暗号化

AWS TNB は、追加の設定を必要とせずに、保管中のサービスに保存されているすべてのデータを常に暗号化します。この暗号化は自動で実行されます AWS Key Management Service。

転送中の暗号化

AWS TNB は、Transport Layer Security (TLS) 1.2 を使用して転送中のすべてのデータを保護します。

シミュレーションエージェントとクライアント間のデータを暗号化するのはお客様の責任です。

ネットワーク間トラフィックのプライバシー

AWS TNB コンピューティングリソースは、すべてのお客様が共有する Virtual Private Cloud (VPC) にあります。すべての内部 AWS TNB トラフィックは AWS ネットワーク内にとどまり、インターネットを経由しません。シミュレーションエージェントとそのクライアント間の接続は、インターネット経由でルーティングされます。

AWS TNB の Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS TNB リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

内容

- AWS TNB と IAM の連携方法
- Telco Network Builder AWS のアイデンティティベースのポリシーの例
- Telco Network Builder AWS のアイデンティティとアクセスのトラブルシューティング

対象者

AWS Identity and Access Management (IAM) の使用方法は、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者にアクセス許可をリクエストします (「Telco Network Builder AWS のアイデンティティとアクセスのトラブルシューティング」を参照)。
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「AWS TNB と IAM の連携方法」を参照)
- IAM 管理者 - アクセスを管理するためのポリシーを作成します (「Telco Network Builder AWS のアイデンティティベースのポリシーの例」を参照)

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

(AWS IAM アイデンティティセンター IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「AWS アカウントにサインインする方法」を参照してください。

プログラムによるアクセスの場合、 は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「API リクエストに対するAWS 署名バージョン 4」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、すべての AWS のサービス および リソースへの完全なアクセス権を持つ AWS アカウント ルートユーザーと呼ばれる 1 つのサインインアイデンティティから始めます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー認証情報を必要とするタスクについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーテッドアイデンティティ

ベストプラクティスとして、人間のユーザーが一時的な認証情報 AWS のサービス を使用して にアクセスするには、ID プロバイダーとのフェデレーションを使用する必要があります。

フェデレーティッド ID は、エンタープライズディレクトリ、ウェブ ID プロバイダー、または ID ソースの認証情報 AWS のサービス を使用して Directory Service にアクセスするユーザーです。フェデレーテッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、 AWS IAM アイデンティティセンターをお勧めします。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の「IAM アイデンティティセンターとは」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することをお勧めします。詳細については、IAM ユーザーガイドの「ID プロバイダーとのフェデレーションを使用して にアクセスする必要がある AWS」を参照してください。

IAM グループは、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可を持つアイデンティであり、一時的な認証情報を提供します。ユーザーから IAM ロール (コンソール) に切り替えるか、 または API オペレーションを呼び出すことで、ロールを引き受けることができます。 AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーションに役立ちます。詳細については、IAM ユーザーガイド の IAM でのクロスアカウントリソースアクセス を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは、ID またはリソースに関連付けられたときにアクセス許可を定義します。 は、プリンシパルがリクエストを行うときにこれらのポリシー AWS を評価します。ほとんどのポリシーは JSON

ドキュメント AWS として に保存されます。JSON ポリシードキュメントの詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

アイデンティティベースのポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイド の カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーとインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

その他のポリシータイプ

AWS は、より一般的なポリシータイプによって付与されるアクセス許可の上限を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可境界」を参照してください。

- サービスコントロールポリシー (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成されるアクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

AWS TNB と IAM の連携方法

IAM を使用して AWS TNB へのアクセスを管理する前に、 AWS TNB で使用できる IAM 機能を確認してください。

Telco Network Builder で使用できる IAM AWS 機能

IAM 機能	AWS TNB サポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	なし
ポリシーアクション	あり
ポリシーリソース	あり
ポリシー条件キー	あり
ACL	なし
ABAC (ポリシー内のタグ)	あり

IAM 機能	AWS TNB サポート
一時的な認証情報	あり
プリンシパルアクセス権限	あり
サービスロール	いいえ
サービスリンクロール	いいえ

AWS TNB およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要については、IAM ユーザーガイドのAWS 「IAM と連携する のサービス」を参照してください。

AWS TNB のアイデンティティベースのポリシー

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。JSON ポリシーで使用できるすべての要素について学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

AWS TNB のアイデンティティベースのポリシーの例

AWS TNB アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいTelco Network Builder AWS のアイデンティティベースのポリシーの例。

AWS TNB 内のリソースベースのポリシー

リソースベースのポリシーのサポート: なし

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシー や Amazon S3 バケットポリシー があげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシー

を使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーで、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。詳細については、IAM ユーザーガイドのIAM でのクロスアカウントリソースアクセスを参照してください。

AWS TNB のポリシーアクション

ポリシーアクションのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

AWS TNB アクションのリストを確認するには、「サービス認可リファレンス」の「Telco Network Builder AWS で定義されるアクション」を参照してください。

AWS TNB のポリシーアクションは、アクションの前に次のプレフィックスを使用します。

```
tnb
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "tnb:CreateSolFunctionPackage",
      "tnb:DeleteSolFunctionPackage"
]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、`List` という単語で始まるすべてのアクションを指定するには次のアクションを含めます。

```
"Action": "tnb:List*"
```

AWS TNB アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいTelco Network Builder AWS のアイデンティティベースのポリシーの例。

AWS TNB のポリシーリソース

ポリシーリソースのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Resource JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。リソースレベルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

AWS TNB リソースタイプとその ARNs「Telco Network Builder AWS で定義されるリソース」を参照してください。 各リソースの ARN を指定できるアクションについては、「Telco Network Builder AWS で定義されるアクション」を参照してください。

AWS TNB アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいTelco Network Builder AWS のアイデンティティベースのポリシーの例。

AWS TNB のポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Condition 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの条件演算子を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

AWS TNB 条件キーのリストを確認するには、「サービス認可リファレンス」の「Telco Network Builder AWS の条件キー」を参照してください。条件キーを使用できるアクションとリソースについては、「Telco Network Builder AWS で定義されるアクション」を参照してください。

AWS TNB アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいTelco Network Builder AWS のアイデンティティベースのポリシーの例。

AWS TNB ACLs

ACL のサポート: なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするためのアクセス許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

AWS TNB を使用した ABAC

ABAC (ポリシー内のタグ) のサポート: あり

属性ベースのアクセス制御 (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグをアタッチし、プリンシパルのタグがリソースのタグと一致するときにオペレーションを許可するように ABAC ポリシーを設計できます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

AWS TNB での一時的な認証情報の使用

一時的な認証情報のサポート: あり

一時的な認証情報は、 AWS リソースへの短期的なアクセスを提供し、フェデレーションまたはスイッチロールの使用時に自動的に作成されます。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM ユーザーガイド」の「IAM の一時的な認証情報」および「AWS のサービス と IAM との連携」を参照してください。

AWS TNB のクロスサービスプリンシパル許可

転送アクセスセッション (FAS) のサポート: あり

転送アクセスセッション (FAS) は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする を使用します。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

AWS TNB のサービスロール

サービスロールのサポート: なし

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの AWS のサービスに許可を委任するロールを作成するを参照してください。

AWS TNB のサービスにリンクされたロール

サービスにリンクされたロールのサポート: なし

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

Telco Network Builder AWS のアイデンティティベースのポリシーの例

デフォルトでは、ユーザーとロールには AWS TNB リソースを作成または変更するアクセス許可はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。

これらのサンプルの JSON ポリシードキュメントを使用して IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーを作成する (コンソール)」を参照してください。

各リソースタイプの ARN の形式など、 AWS TNB で定義されるアクションとリソースタイプの詳細については、「サービス認可リファレンス」の「Telco Network Builder AWS のアクション、リソース、および条件キー」を参照してください。 ARNs

内容

- ポリシーに関するベストプラクティス
- AWS TNB コンソールの使用
- サービスロールポリシーの例
- 自分の権限の表示をユーザーに許可する

ポリシーに関するベストプラクティス

ID ベースのポリシーは、アカウント内で誰かが AWS TNB リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行 – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイド の AWS マネージドポリシー または ジョブ機能のAWS マネージドポリシー を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、IAM ユーザーガイド の IAM でのポリシーとアクセス許可 を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます CloudFormation。詳細については、IAM ユーザーガイド の IAM JSON ポリシー要素:条件 を参照してください。
- IAM アクセスアナライザー を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM アクセスアナライザー は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、IAM ユーザーガイド の IAM Access Analyzer でポリシーを検証する を参照してください。

- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイド の MFA を使用した安全な API アクセス を参照してください。

IAM でのベストプラクティスの詳細については、IAM ユーザーガイド の IAM でのセキュリティのベストプラクティス を参照してください。

AWS TNB コンソールの使用

Telco Network Builder AWS コンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、 の AWS TNB リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

サービスロールポリシーの例

管理者は、環境テンプレートとサービステンプレートで定義されているように AWS TNB が作成するリソースを所有および管理します。 AWS TNB がネットワークライフサイクル管理用のリソースを作成できるようにするには、アカウントに IAM サービスロールをアタッチする必要があります。

IAM サービスロールを使用すると、 AWS TNB はユーザーに代わって リソースを呼び出して、ネットワークをインスタンス化および管理できます。サービスロールを指定すると、 AWS TNB はそのロールの認証情報を使用します。

IAM サービスで、サービスロールと権限ポリシーを作成します。サービスロールの作成の詳細については、IAM ユーザーガイドの「 AWS サービスにアクセス許可を委任するロールの作成」を参照してください。

AWS TNB サービスロール

プラットフォームチームのメンバーは、管理者として AWS TNB サービスロールを作成して AWS TNB に提供できます。このロールにより、 AWS TNB は Amazon Elastic Kubernetes Service などの他の サービスを呼び出し CloudFormation 、ネットワークに必要なインフラストラクチャをプロビジョニングし、NSD で定義されているネットワーク機能をプロビジョニングできます。

AWS TNB サービスロールには、以下の IAM ロールと信頼ポリシーを使用することをお勧めします。このポリシーのアクセス許可をスコープダウンする場合は、ポリシーの対象から外れたリソースに対するアクセス拒否エラーで AWS TNB が失敗する可能性があることに注意してください。

次のコードは AWS TNB サービスロールポリシーを示しています。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:GetCallerIdentity"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "AssumeRole"
        },
        {
            "Action": [
                "tnb:*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TNBPolicy"
        },
        {
            "Action": [
                "iam:AddRoleToInstanceProfile",
                "iam:CreateInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile",
                "iam:TagInstanceProfile",
                "iam:UntagInstanceProfile"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "IAMPolicy"
        },
        {
            "Condition": {
```

```
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "eks.amazonaws.com",
                        "eks-nodegroup.amazonaws.com"
                    ]
                }
            },
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TNBAccessSLRPermissions"
        },
        {
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateOrUpdateTags",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DeleteTags",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:DescribeTags",
                "autoscaling:UpdateAutoScalingGroup",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteLaunchTemplateVersions",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:DeleteLaunchTemplate",
                "ec2:DeleteSecurityGroup",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeTags",
                "ec2:GetLaunchTemplateData",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:RunInstances",
                "ec2:AssociateRouteTable",
                "ec2:AttachInternetGateway",
                "ec2:CreateInternetGateway",
```

```
                "ec2:CreateNetworkInterface",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:CreateSubnet",
                "ec2:CreateTags",
                "ec2:CreateVpc",
                "ec2:DeleteInternetGateway",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteRoute",
                "ec2:DeleteRouteTable",
                "ec2:DeleteSubnet",
                "ec2:DeleteTags",
                "ec2:DeleteVpc",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeInstances",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroupRules",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DetachInternetGateway",
                "ec2:DisassociateRouteTable",
                "ec2:ModifySecurityGroupRules",
                "ec2:ModifySubnetAttribute",
                "ec2:ModifyVpcAttribute",
                "ec2:AllocateAddress",
                "ec2:AssignIpv6Addresses",
                "ec2:AssociateAddress",
                "ec2:AssociateNatGatewayAddress",
                "ec2:AssociateVpcCidrBlock",
                "ec2:CreateEgressOnlyInternetGateway",
                "ec2:CreateNatGateway",
                "ec2:DeleteEgressOnlyInternetGateway",
                "ec2:DeleteNatGateway",
                "ec2:DescribeAddresses",
                "ec2:DescribeEgressOnlyInternetGateways",
                "ec2:DescribeNatGateways",
                "ec2:DisassociateAddress",
                "ec2:DisassociateNatGatewayAddress",
                "ec2:DisassociateVpcCidrBlock",
                "ec2:ReleaseAddress",
                "ec2:UnassignIpv6Addresses",
```

```
                "ec2:DescribeImages",
                "eks:CreateCluster",
                "eks:ListClusters",
                "eks:RegisterCluster",
                "eks:TagResource",
                "eks:DescribeAddonVersions",
                "events:DescribeRule",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TNBAccessComputePerms"
        },
        {
            "Resource": "*",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Condition": {
                "StringEquals": {
                   "iam:PassedToService": [
                       "ec2.amazonaws.com",
                       "eks.amazonaws.com",
                       "eks-nodegroup.amazonaws.com",
                       "events.amazonaws.com",
                       "autoscaling.amazonaws.com",
                       "codebuild.amazonaws.com"
                   ]
                }
            }
        },
        {
            "Action": [
                "codebuild:BatchDeleteBuilds",
                "codebuild:BatchGetBuilds",
                "codebuild:CreateProject",
                "codebuild:DeleteProject",
                "codebuild:ListBuildsForProject",
                "codebuild:StartBuild",
                "codebuild:StopBuild",
                "events:DeleteRule",
                "events:PutRule",
```

```
                "events:PutTargets",
                "events:RemoveTargets",
                "s3:CreateBucket",
                "s3:GetBucketAcl",
                "s3:GetObject",
                "eks:DescribeNodegroup",
                "eks:DeleteNodegroup",
                "eks:AssociateIdentityProviderConfig",
                "eks:CreateNodegroup",
                "eks:DeleteCluster",
                "eks:DeregisterCluster",
                "eks:UpdateAddon",
                "eks:UpdateClusterVersion",
                "eks:UpdateNodegroupConfig",
                "eks:UpdateNodegroupVersion",
                "eks:DescribeUpdate",
                "eks:UntagResource",
                "eks:DescribeCluster",
                "eks:ListNodegroups",
                "eks:CreateAddon",
                "eks:DeleteAddon",
                "eks:DescribeAddon",
                "eks:DescribeAddonVersions",
                "s3:PutObject",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks",
                "cloudformation:ListStackResources",
                "cloudformation:UpdateStack",
                "cloudformation:UpdateTerminationProtection",
                "ssm:PutParameter",
                "ssm:GetParameters",
                "ssm:GetParameter",
                "ssm:DeleteParameter",
                "ssm:AddTagsToResource",
                "ssm:ListTagsForResource",
                "ssm:RemoveTagsFromResource"
            ],
            "Resource": [
                "arn:aws:events:*:*:rule/tnb*",
                "arn:aws:codebuild:*:*:project/tnb*",
                "arn:aws:logs:*:*:log-group:/aws/tnb*",
                "arn:aws:s3:::tnb*",
```

```
                "arn:aws:eks:*:*:addon/tnb*/*/*",
                "arn:aws:eks:*:*:cluster/tnb*",
                "arn:aws:eks:*:*:nodegroup/tnb*/tnb*/*",
                "arn:aws:cloudformation:*:*:stack/tnb*",
                "arn:aws:ssm:*:*:parameter/tnb/*"
            ],
            "Effect": "Allow",
            "Sid": "TNBAccessInfraResourcePerms"
        },
        {
            "Sid": "CFNTemplatePerms",
            "Effect": "Allow",
            "Action": [
              "cloudformation:GetTemplateSummary"
            ],
            "Resource": "*"
         },
        {
            "Sid": "ImageAMISSMPerms",
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:*::parameter/aws/service/eks/optimized-ami/*",
                "arn:aws:ssm:*::parameter/aws/service/bottlerocket/*"
            ]
        },
        {
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TaggingPolicy"
        },
        {
            "Action": [
                "outposts:GetOutpost"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "OutpostPolicy"
        }
```

```
    ]
}
```

次のコードは AWS TNB サービス信頼ポリシーを示しています。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "tnb.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS Amazon EKS クラスターの TNB サービスロール

NSD に Amazon EKS リソースを作成するときは、Amazon EKS クラスターの作成に使用されるロールを指定する `cluster_role` 属性を指定します。

次の例は、Amazon EKS クラスターポリシーの AWS TNB サービスロールを作成する AWS CloudFormation テンプレートを示しています。

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  TNBEKSClusterRole:
    Type: "AWS::IAM::Role"
    Properties:
      RoleName: "TNBEKSClusterRole"
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - eks.amazonaws.com
            Action:
              - "sts:AssumeRole"
      Path: /
      ManagedPolicyArns:
        - !Sub "arn:${AWS::Partition}:iam::aws:policy/AmazonEKSClusterPolicy"
```

AWS CloudFormation テンプレートを使用する IAM ロールの詳細については、 AWS CloudFormation ユーザーガイドの以下のセクションを参照してください。

- AWS::IAM::Role
- スタックテンプレートの選択

AWS Amazon EKS ノードグループの TNB サービスロール

NSD に Amazon EKS ノードグループリソースを作成するときは、Amazon EKS ノードグループの作成に使用されるロールを指定する node_role 属性を指定します。

次の例は、Amazon EKS ノードグループポリシーの AWS TNB サービスロールを作成する CloudFormation テンプレートを示しています。

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  TNBEKSNodeRole:
    Type: "AWS::IAM::Role"
    Properties:
      RoleName: "TNBEKSNodeRole"
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - "sts:AssumeRole"
      Path: /
      ManagedPolicyArns:
        - !Sub "arn:${AWS::Partition}:iam::aws:policy/AmazonEKSWorkerNodePolicy"
        - !Sub "arn:${AWS::Partition}:iam::aws:policy/AmazonEKS_CNI_Policy"
        - !Sub "arn:${AWS::Partition}:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly"
        - !Sub "arn:${AWS::Partition}:iam::aws:policy/service-role/
AmazonEBSCSIDriverPolicy"
      Policies:
        - PolicyName: EKSNodeRoleInlinePolicy
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - "logs:DescribeLogStreams"
                  - "logs:PutLogEvents"
                  - "logs:CreateLogGroup"
                  - "logs:CreateLogStream"
                Resource: "arn:aws:logs:*:*:log-group:/aws/tnb/tnb*"
```

```
        - PolicyName: EKSNodeRoleIpv6CNIPolicy
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action:
                  - "ec2:AssignIpv6Addresses"
                Resource: "arn:aws:ec2:*:*:network-interface/*"
```

AWS CloudFormation テンプレートを使用する IAM ロールの詳細については、 AWS CloudFormation ユーザーガイドの以下のセクションを参照してください。

- AWS::IAM::Role
- スタックテンプレートの選択

AWS Multus の TNB サービスロール

NSD に Amazon EKS リソースを作成し、デプロイテンプレートの一部として Multus を管理する場合は、Multus の管理にどのロールを使用するかを指定する `multus_role` 属性を指定する必要があります。

次の例は、Multus ポリシーの AWS TNB サービスロールを作成する CloudFormation テンプレートを示しています。

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  TNBMultusRole:
    Type: "AWS::IAM::Role"
    Properties:
      RoleName: "TNBMultusRole"
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Action:
              - "sts:AssumeRole"
          - Effect: Allow
            Principal:
```

```
            Service:
              - codebuild.amazonaws.com
          Action:
            - "sts:AssumeRole"
    Path: /
    Policies:
      - PolicyName: MultusRoleInlinePolicy
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: Allow
              Action:
                - "codebuild:StartBuild"
                - "logs:DescribeLogStreams"
                - "logs:PutLogEvents"
                - "logs:CreateLogGroup"
                - "logs:CreateLogStream"
              Resource:
                - "arn:aws:codebuild:*:*:project/tnb*"
                - "arn:aws:logs:*:*:log-group:/aws/tnb/*"
            - Effect: Allow
              Action:
                - "ec2:CreateNetworkInterface"
                - "ec2:ModifyNetworkInterfaceAttribute"
                - "ec2:AttachNetworkInterface"
                - "ec2:DeleteNetworkInterface"
                - "ec2:CreateTags"
                - "ec2:DetachNetworkInterface"
              Resource: "*"
```

AWS CloudFormation テンプレートを使用する IAM ロールの詳細については、 AWS CloudFormation ユーザーガイドの以下のセクションを参照してください。

- AWS::IAM::Role
- スタックテンプレートの選択

AWS ライフサイクルフックポリシーの TNB サービスロール

NSD またはネットワーク関数パッケージがライフサイクルフックを使用する場合、ライフサイクルフックを実行するための環境を作成できるサービスロールが必要です。

> **Note**
>
> ライフサイクルフックポリシーは、ライフサイクルフックが実行しようとしている内容に基づくものでなければなりません。

次の例は、ライフサイクルフックポリシーの AWS TNB サービスロールを作成する CloudFormation テンプレートを示しています。

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  TNBHookRole:
    Type: "AWS::IAM::Role"
    Properties:
      RoleName: "TNBHookRole"
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - codebuild.amazonaws.com
            Action:
              - "sts:AssumeRole"
      Path: /
      ManagedPolicyArns:
        - !Sub "arn:${AWS::Partition}:iam::aws:policy/AdministratorAccess"
```

AWS CloudFormation テンプレートを使用する IAM ロールの詳細については、 AWS CloudFormation ユーザーガイドの以下のセクションを参照してください。

- AWS::IAM::Role
- スタックテンプレートの選択

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Telco Network Builder AWS のアイデンティティとアクセスのトラブルシューティング

以下の情報は、 AWS TNB と IAM の使用時に発生する可能性がある一般的な問題の診断と修正に役立ちます。

問題

- AWS TNB でアクションを実行する権限がありません

- iam:PassRole を実行する権限がありません
- 自分の 以外のユーザーに AWS TNB リソース AWS アカウント へのアクセスを許可したい

AWS TNB でアクションを実行する権限がありません

アクションを実行する権限がないというエラーが表示された場合は、そのアクションを実行できるようにポリシーを更新する必要があります。

以下のエラー例は、mateojackson IAM ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細情報を表示しようとしているが、架空の tnb:*GetWidget* 権限がないという場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 tnb:GetWidget on resource: my-example-widget
```

この場合、Mateo のポリシーでは、*my-example-widget* アクションを使用して tnb:*GetWidget* リソースにアクセスすることを許可するように更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して AWS TNB にロールを渡すことができるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡すアクセス許可が必要です。

次のエラー例は、marymajor という IAM ユーザーがコンソールを使用して AWS TNB でアクションを実行しようとする場合に発生するものです。ただし、このアクションをサービスが実行するには、サービスロールから付与されたアクセス許可が必要です。Mary には、ロールをサービスに渡すアクセス許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに AWS TNB リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- AWS TNB がこれらの機能をサポートしているかどうかを確認するには、「」を参照してくださいAWS TNB と IAM の連携方法。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、「IAM ユーザーガイド」の「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイド の 外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可 を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、IAM ユーザーガイド の IAM でのクロスアカウントのリソースへのアクセス を参照してください。

AWS TNB のコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、AWS のサービス「コンプライアンスプログラムによる対象範囲内のコンプライアンス」を参照し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS 「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。を使用する際のコンプライアンス責任の詳細については AWS のサービス、AWS 「 セキュリティドキュメント」を参照してください。

AWS TNB の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。 は、低レイテンシー、高スループット、高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティーゾーン AWS リージョン を提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェールオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、フォールトトレランス、および拡張性が優れています。

AWS リージョン およびアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

AWS TNB は、選択した AWS リージョンの Virtual Private Cloud (VPC) で EKS クラスターで Network Service を実行します。

AWS TNB のインフラストラクチャセキュリティ

マネージドサービスである AWS Telco Network Builder は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して環境を AWS 設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS 公開された API コールを使用して、ネットワーク経由で AWS TNB にアクセスします。クライアントは次をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは Java 7 以降など、ほとんどの最新システムでサポートされています。

責任共有の例をいくつか示します。

- AWS は、以下を含む AWS TNB をサポートするコンポーネントを保護する責任があります。
  - コンピューティングインスタンス (ワーカーとも呼ばれます)
  - 内部データベース
  - 内部コンポーネント間のネットワーク通信
  - AWS TNB アプリケーションプログラミングインターフェイス (API)
  - AWS ソフトウェア開発キット (SDK)
- お客様は、 AWS リソースおよびワークロードコンポーネントへのアクセスを保護する責任があります。これには、以下が含まれます (ただし、これらに限定されません) 。
  - IAM ユーザー、グループ、ロール、ポリシー
  - AWS TNB のデータの保存に使用する S3 バケット
  - AWS TNB を通じてプロビジョニングしたネットワークサービスをサポートするために使用するその他の AWS のサービス および リソース
  - アプリケーションコード
  - AWS TNB を介してプロビジョニングしたネットワークサービスとクライアント間の接続

> **Important**
>
> AWS TNB を通じてプロビジョニングしたネットワークサービスを効果的に復旧できるディザスタリカバリプランを実装するのは、お客様の責任です。

ネットワーク接続セキュリティモデル

AWS TNB を通じてプロビジョニングするネットワークサービスは、選択した AWS リージョンにある Virtual Private Cloud (VPC) 内のコンピューティングインスタンスで実行されます。VPC は AWS クラウドの仮想ネットワークであり、ワークロードまたは組織エンティティごとにインフラストラクチャを分離します。VPC 内のコンピューティングインスタンス間の通信は AWS ネットワーク内にとどまり、インターネットを経由することはありません。一部の内部サービス通信はインターネットを経由し、暗号化されます。同じリージョンで実行されているすべてのお客様に AWS TNB を介してプロビジョニングされたネットワークサービスは、同じ VPC を共有します。異なる顧客向けに AWS TNB を介してプロビジョニングされたネットワークサービスは、同じ VPC 内で個別のコンピューティングインスタンスを使用します。

ネットワークサービスクライアントと AWS TNB のネットワークサービス間の通信はインターネットを経由します。 AWS TNB はこれらの接続を管理しません。クライアント接続を保護するのはお客様の責任です。

、 AWS Command Line Interface (AWS CLI) AWS マネジメントコンソール、および SDK を介した AWS TNB への接続は暗号化されます。 AWS SDKs

IMDS バージョン

AWS TNB は、セッション指向のメソッドであるインスタンスメタデータサービスバージョン 2 (IMDSv2) を利用するインスタンスをサポートしています。IMDSv2 には IMDSV1 よりも高いセキュリティが含まれています。詳細については、「Amazon EC2 インスタンスメタデータサービスの拡張により、オープンファイアウォール、リバースプロキシ、SSRF の脆弱性に対して多層防御を追加する」を参照してください。

インスタンスを起動するときは、IMDSv2 を使用する必要があります。IMDSv2 の詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータサービスを使用してインスタンスメタデータにアクセスする」を参照してください。

AWS TNB のモニタリング

モニタリングは、 AWS TNB およびその他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。 AWS は AWS 、 TNB を監視し、問題が発生したときに報告し、必要に応じて自動アクションを実行 AWS CloudTrail します。

CloudTrail を使用して、 AWS APIs。これらの呼び出しはログ ファイルとして Amazon S3 に保存できます。これらの CloudTrail ログを使用して、行われた呼び出し、呼び出し元のソース IP アドレス、呼び出し元、呼び出し時間などを判断できます。

CloudTrail ログには、 AWS TPN の API アクションの呼び出しに関する情報が含まれています。これらには、Amazon EC2 や Amazon EBS などのサービスからの API アクションの呼び出しに関する情報も含まれています。

を使用した AWS Telco Network Builder API コールのログ記録 AWS CloudTrail

AWS Telco Network Builder は、ユーザーAWS CloudTrail、ロール、または によって実行されたアクションを記録するサービスである と統合されています AWS のサービス。CloudTrail は AWS TNB のすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、 AWS TNB コンソールからの呼び出しと AWS TNB API オペレーションへのコード呼び出しが含まれます。CloudTrail で収集された情報を使用して、 AWS TPN に対するリクエスト、リクエスト元の IP アドレス、リクエスト日時などの詳細を確認できます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- ルートユーザーまたはユーザー認証情報のどちらを使用してリクエストが送信されたか。
- リクエストが IAM Identity Center ユーザーに代わって行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

CloudTrail は、アカウントを作成する AWS アカウント と でアクティブになり、CloudTrail イベント履歴に自動的にアクセスできます。CloudTrail の [イベント履歴] では、 AWS リージョンで過去

90 日間に記録された管理イベントの表示、検索、およびダウンロードが可能で、変更不可能な記録を確認できます。詳細については、「AWS CloudTrail ユーザーガイド」の「CloudTrail イベント履歴の使用」を参照してください。[イベント履歴] の閲覧には CloudTrail の料金はかかりません。

AWS アカウント 過去 90 日間のイベントの継続的な記録については、証跡または CloudTrail Lake イベントデータストアを作成します。

CloudTrail 証跡

証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。を使用して作成されたすべての証跡 AWS マネジメントコンソール はマルチリージョンです。 AWS CLIを使用する際は、単一リージョンまたは複数リージョンの証跡を作成できます。アカウント AWS リージョン 内のすべての でアクティビティをキャプチャするため、マルチリージョン証跡を作成することをお勧めします。単一リージョンの証跡を作成する場合、証跡の AWS リージョンに記録されたイベントのみを表示できます。証跡の詳細については、「AWS CloudTrail ユーザーガイド」の「AWS アカウントの証跡の作成」および「組織の証跡の作成」を参照してください。

証跡を作成すると、進行中の管理イベントのコピーを 1 つ無料で CloudTrail から Amazon S3 バケットに配信できますが、Amazon S3 ストレージには料金がかかります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。Amazon S3 の料金に関する詳細については、「Amazon S3 の料金」を参照してください。

CloudTrail Lake イベントデータストア

[CloudTrail Lake] を使用すると、イベントに対して SQL ベースのクエリを実行できます。CloudTrail Lake は、行ベースの JSON 形式の既存のイベントを Apache ORC 形式に変換します。ORC は、データを高速に取得するために最適化された単票ストレージ形式です。イベントは、イベントデータストアに集約されます。イベントデータストアは、高度なイベントセレクタを適用することによって選択する条件に基づいた、イベントのイミュータブルなコレクションです。どのイベントが存続し、クエリに使用できるかは、イベントデータストアに適用するセレクタが制御します。CloudTrail Lake の詳細については、 AWS CloudTrail ユーザーガイドのAWS CloudTrail 「Lake の使用」を参照してください。

CloudTrail Lake のイベントデータストアとクエリにはコストがかかります。イベントデータストアを作成する際に、イベントデータストアに使用する料金オプションを選択します。料金オプションによって、イベントの取り込みと保存にかかる料金、および、そのイベントデータストアのデフォルトと最長の保持期間が決まります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。

AWS TNB イベントの例

各イベントは任意の送信元からの単一のリクエストを表し、リクエストされた API オペレーション、オペレーションの日時、リクエストパラメータなどに関する情報を含みます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、イベントは特定の順序で表示されません。

次の例は、CreateSolFunctionPackage オペレーションを示す CloudTrail イベントを示しています。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:example",
        "arn": "arn:aws:sts::111222333444:assumed-role/example/user",
        "accountId": "111222333444",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111222333444:role/example",
                "accountId": "111222333444",
                "userName": "example"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-02-02T01:42:39Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-02-02T01:43:17Z",
    "eventSource": "tnb.amazonaws.com",
    "eventName": "CreateSolFunctionPackage",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "XXX.XXX.XXX.XXX",
    "userAgent": "userAgent",
    "requestParameters": null,
    "responseElements": {
        "vnfPkgArn": "arn:aws:tnb:us-east-1:111222333444:function-package/
fp-12345678abcEXAMPLE",
```

```
        "id": "fp-12345678abcEXAMPLE",
        "operationalState": "DISABLED",
        "usageState": "NOT_IN_USE",
        "onboardingState": "CREATED"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111222333444",
    "eventCategory": "Management"
}
```


CloudTrail レコードの内容については、「AWS CloudTrail ユーザーガイド」の「CloudTrail record contents」を参照してください。

AWS TNB デプロイタスク

デプロイタスクを理解することで、デプロイを効果的にモニタリングし、より迅速にアクションを実行できます。

次の表に、 AWS TPN デプロイタスクを示します。

2024 年 3 月 7 日より前に開始されたデプロイのタスク名	2024 年 3 月 7 日以降に開始されたデプロイのタスク名	タスクの説明
AppInstallation	ClusterPluginInstall	Multus プラグインを Amazon EKS クラスターにインストールします。
AppUpdate	名前に変更なし	ネットワークインスタンスにすでにインストールされているネットワーク機能を更新します。
-	ClusterPluginUninstall	Amazon EKS クラスターにプラグインをアンインストールします。
ClusterStorageClassConfiguration	名前に変更なし	Amazon EKS クラスターのストレージクラス (CSI ドライバー) を設定します。

2024 年 3 月 7 日より前に開始されたデプロイのタスク名	2024 年 3 月 7 日以降に開始されたデプロイのタスク名	タスクの説明
FunctionDeletion	名前に変更なし	AWS TNB リソースからネットワーク関数を削除します。
FunctionInstantiation	FunctionInstall	HELM を使用してネットワーク機能をデプロイします。
FunctionUninstallation	FunctionUninstall	Amazon EKS クラスターからネットワーク機能をアンインストールします。
HookExecution	名前に変更なし	NSD で定義されているライフサイクルフックを実行します。
InfrastructureCancellation	名前に変更なし	ネットワークサービスをキャンセルします。
InfrastructureInstantation	名前に変更なし	ユーザーに代わって AWS リソースをプロビジョニングします。
InfrastructureTermination	名前に変更なし	AWS TNB を介して呼び出される AWS リソースのプロビジョニングを解除します。
-	InfrastructureUpdate	ユーザーに代わってプロビジョニングされた AWS リソースを更新します。
InventoryDeregistration	名前に変更なし	AWS TNB から AWS リソースを登録解除します。
-	InventoryRegistration	AWS TNB に AWS リソースを登録します。
KubernetesClusterConfiguration	ClusterConfiguration	Kubernetes クラスターを設定し、NSD で定義されているように Amazon EKS AuthMap に追加の IAM ロールを追加します。
NetworkServiceFinalization	名前に変更なし	ネットワークサービスを確定し、成功または失敗のステータス更新を行います。

2024 年 3 月 7 日より前に開始されたデプロイのタスク名	2024 年 3 月 7 日以降に開始されたデプロイのタスク名	タスクの説明
NetworkServiceInstantiation	名前に変更なし	ネットワークサービスを初期化します。
SelfManagedNodesConfiguration	名前に変更なし	Amazon EKS と Kubernetes のコントロールプレーンを使用してセルフマネージド型のノードをブートストラップします。
-	ValidateNetworkServiceUpdate	ネットワークインスタンスを更新する前に検証を実行します。

AWS TNB のサービスクォータ

制限とも呼ばれるサービスクォータは、 AWS アカウントのサービスリソースまたはオペレーションの最大数です。詳細については、「Amazon Web Services 全般のリファレンス」の「AWS の Service Quotas」を参照してください。

AWS TNB のサービスクォータは次のとおりです。

名前	デフォルト	引き上げ可能	説明
継続的なネットワークサービスの同時オペレーション	サポートされている各リージョン: 40	可能	1 つのリージョンで同時に進行中のネットワークサービスオペレーションの最大数。
関数パッケージ	サポートされている各リージョン: 200	あり	1 つのリージョンの関数パッケージの最大数。
ネットワークパッケージ	サポートされている各リージョン: 40	可能	1 つのリージョンのネットワークパッケージの最大数。
ネットワークサービスインスタンス	サポートされている各リージョン: 800	あり	1 つのリージョンのネットワークサービスインスタンスの最大数。

AWS TNB ユーザーガイドのドキュメント履歴

次の表に、 AWS TNB のドキュメントリリースを示します。

変更	説明	日付
Amazon EKS ノードグループネットワーク設定の更新	サブネットとセキュリティグループを追加および削除します。ネットワークから ENIs を追加、変更、削除します。詳細については、「更新できるパラメータ」を参照してください。	2025 年 9 月 10 日
既存のクラスター内の Amazon EKS ノードグループの追加と削除	AWS TNB は、新しいノードグループの追加と Amazon EKS クラスターからの既存のノードグループの削除をサポートするようになりました。詳細については、「更新できるパラメータ」を参照してください。	2025 年 6 月 4 日
ルートボリュームサイズ	Amazon EKS ワーカーノードの基盤となる Amazon EBS ルートボリュームのサイズは、AWS.Compute.EKSManagedNode および AWS.Compute.EKSSelfManagedNode TOSCA ノードの `root_volume_size` フィールドを使用して指定できます。	2025 年 5 月 19 日
スクリプトのリファレンスリソース	AWS TNB によって作成されたリソースを参照して、ライフサイクルフックスクリプ	2025 年 5 月 2 日

	トとユーザーデータスクリプトで設定できます。	
Kubernetes バージョン 1.32 が Amazon EKS ノードとマネージドノードグループでサポートされるようになりました。	AWS TNB は、.AWS Compute.EKS および .AWS Compute.EKSManagedNode の Kubernetes バージョン 1.32 をサポートしています。	2025 年 4 月 24 日
Kubernetes バージョン 1.24 は Amazon EKS ノードとマネージドノードグループでサポートされなくなりました	AWS TNB は、.AWS Compute.EKS および .AWS Compute.EKSManagedNode の Kubernetes バージョン 1.24 をサポートしなくなりました。	2025 年 4 月 17 日
Amazon EKS マネージドノードの AL2023 AMI サポート	AWS TNB は、AWS.Compute.EKSManagedNode の AL2023 AMI タイプをサポートしています。	2025 年 4 月 17 日
Kubernetes バージョン 1.23 は Amazon EKS ノードとマネージドノードグループでサポートされなくなりました	AWS TNB は、.AWS Compute.EKS および .AWS Compute.EKSManagedNode の Kubernetes バージョン 1.23 をサポートしなくなりました。	2025 年 4 月 4 日
AMI ID を更新できます	`UpdateSolNetworkService` API コール中に ami_id フィールドを更新できるようになりました。	2025 年 3 月 31 日
Kubernetes バージョン 1.31 が Amazon EKS ノードとマネージド型ノードグループでサポートされるようになりました。	AWS TNB は、AWS.Compute.EKS および AWS.Compute.EKSManagedNode の Kubernetes バージョン 1.31 をサポートしています。	2025 年 2 月 18 日

for AWS.Compute.EKSManagedNode の Kubernetes バージョン	AWS TNB は、Kubernetes バージョン 1.23 ~ 1.30 をサポートし、Amazon EKS マネージド型ノードグループを作成します。	2025 年 1月 28 日
クラスター用の Kubernetes バージョン	AWS TNB は、Amazon EKS クラスターを作成するための Kubernetes バージョン 1.30 をサポートするようになりました。	2024 年 8 月 19 日

AWS TNB は、ネットワークライフサイクルを管理するための追加のオペレーションをサポートしています。	インスタンス化または以前に更新されたネットワークインスタンスを、新しいネットワークパッケージとパラメータ値で更新できます。以下を参照してください。 • ライフサイクルオペレーション • ネットワークインスタンスを更新する • AWS TNB サービスロールの例: • Amazon EKS アクション `eks:UpdateAddon` 、`eks:UpdateClusterVersion` 、`eks:UpdateNodegroupConfig` 、`eks:UpdateNodegroupVersion` 、 を追加します。 `eks:DescribeUpdate` • この CloudFormation アクションを追加します。 `cloudformation:UpdateStack` • 新しいデプロイタスク: `InfrastructureUpdate` 、`InventoryRegistration` 、`ValidateNetworkServiceUpdate`	2024 年 7 月 30 日

	• API 更新: GetSolNetworkOperation、ListSolNetworkOperations、UpdateSolNetworkInstance	
既存のタスクの新しいタスク名と新しいタスク名	新しいタスクを使用できます。2024 年 3 月 7 日現在、一部の既存のタスクにはわかりやすくするために新しい名前が付けられています。	2024 年 5 月 7 日
クラスター用の Kubernetes バージョン	AWS TNB は、Amazon EKS クラスターを作成するための Kubernetes バージョン 1.29 をサポートするようになりました。	2024 年 4 月 10 日
ネットワークインターフェイスのサポート `security_groups`	セキュリティグループを.Networking AWS.ENI ノードにアタッチできます。	2024 年 4 月 2 日
Amazon EBS ルートボリューム暗号化のサポート	Amazon EBS ルートボリュームの Amazon EBS 暗号化を有効にできます。有効にするには、AWS.Compute.EKSManagedNode または AWS.Compute.EKSSelfManagedNode ノードにプロパティを追加します。	2024 年 4 月 2 日
ノードのサポート `labels`	ノードラベルは、AWS.Compute.EKSManagedNode または AWS.Compute.EKSSelfManagedNode ノードのノードグループにアタッチできます。	2024 年 3 月 19 日

ネットワークインターフェイスのサポート source_dest_check	.Networking AWS.ENI ノードを介してネットワークインターフェイスの送信元/送信先チェックを有効または無効にするかどうかを指定できます。	2024 年 1 月 25 日
カスタムユーザーデータを使用した、Amazon EC2 インスタンスのサポート	AWS.Compute.UserData ノードを通じ、カスタムユーザーデータを使用して Amazon EC2 インスタンスを起動できます。	2024 年 1 月 16 日
セキュリティグループのサポート	AWS TNB では、セキュリティグループ AWS リソースをインポートできます。	2024 年 1 月 8 日
network_interfaces の説明を更新しました	network_interfaces プロパティが AWS.Compute.EKSManagedNode または AWS.Compute.EKSSelfManagedNode ノードに含まれている場合、 AWS TNB は multus_role プロパティまたは node_role プロパティから ENIsに関連するアクセス許可を取得します。	2023 年 12 月 18 日
プライベートクラスターのサポート	AWS TNB がプライベートクラスターをサポートするようになりました。プライベートクラスターを指定するには、access プロパティを PRIVATE に設定します。	2023 年 12 月 11 日

クラスター用の Kubernetes バージョン	AWS TNB は、Amazon EKS クラスターを作成するための Kubernetes バージョン 1.28 をサポートするようになりました。	2023 年 12 月 11 日
AWS TNB がプレイスメントグループをサポート	`AWS.Compute.EKSManagedNode` および `AWS.Compute.EKSSelfManagedNode` ノード定義の配置グループを追加しました。	2023 年 12 月 11 日

AWS TNB が IPv6 のサポートを追加	AWS TNB は、IPv6 インフラストラクチャを使用したネットワークインスタンスの作成をサポートするようになりました。IPv6 の設定については、AWS.Networking.VPC、AWS.Networking.Subnet、AWS.Networking.InternetGateway、AWS.Networking.SecurityGroupIngressRule、AWS.Networking.SecurityGroupEgressRule、および AWS.Compute.EKS の各ノードを確認してください。また、NAT64 の設定用の AWS.Networking.NATGateway および AWS.Networking.Route ノードも追加しました。IPv6 AWS アクセス許可の Amazon EKS ノードグループの TNB サービスロールと AWS TNB サービスロールを更新しました。「サービスロールポリシーの例」を参照してください。	2023 年 11 月 16 日
AWS TNB サービスロールポリシーにアクセス許可を追加しました	Amazon S3 および の AWS TNB サービスロールポリシー CloudFormation に、インフラストラクチャのインスタンス化を有効にするためのアクセス許可を追加しました。	2023 年 10 月 23 日

AWS より多くのリージョンで開始された TNB	AWS TNB が、アジアパシフィック (ソウル) 、カナダ (中部) 、欧州 (スペイン) 、欧州 (ストックホルム) 、南米 (サンパウロ) の各リージョンで利用可能になりました。	2023 年 9 月 27 日
for AWS.Compute.EKSSelfManagedNode のタグ	AWS TNB が`AWS.Compute.EKSSelfManagedNode` ノード定義のタグをサポートするようになりました。	2023 年 8 月 22 日
AWS TNB が IMDSv2 を活用するインスタンスをサポート	インスタンスを起動するときは、IMDSv2 を使用する必要があります。	2023 年 8 月 14 日
MultusRoleInlinePolicy のアクセス許可の更新	MultusRoleInlinePolicy に`ec2:DeleteNetworkInterface` のアクセス許可が含まれるようになりました。	2023 年 8 月 7 日
クラスター用の Kubernetes バージョン	AWS TNB は、Amazon EKS クラスターを作成するための Kubernetes バージョン 1.27 をサポートするようになりました。	2023 年 7 月 25 日
AWS.Compute.EKS.AuthRole	AWS TNB は、ユーザーが IAM ロールを使用して Amazon EKS クラスターにアクセスできる`aws-authConfigMap` ように、Amazon EKS クラスターに IAM ロールを追加できる AuthRole をサポートしています。	2023 年 7 月 19 日

AWS TNB はセキュリティグループをサポートしています。	AWS.Networking.SecurityGroup、AWS.Networking.SecurityGroupEgressRule、および AWS.Networking.SecurityGroupIngressRule を NSD テンプレートに追加しました。	2023 年 7 月 18 日
クラスター用の Kubernetes バージョン	AWS TNB は、Amazon EKS クラスターを作成するための Kubernetes バージョン 1.22 ~ 1.26 をサポートしています。AWS TNB は Kubernetes バージョン 1.21 をサポートしなくなりました。	2023 年 5 月 11 日
AWS.Compute.EKSSelfManagedNode	リージョン内、 AWS ローカルゾーン、および にセルフマネージド型ワーカーノードを作成できます AWS Outposts。	2023 年 3 月 29 日
初回リリース	これは TNB AWS ユーザーガイドの最初のリリースです。	2023 年 2 月 21 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。