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

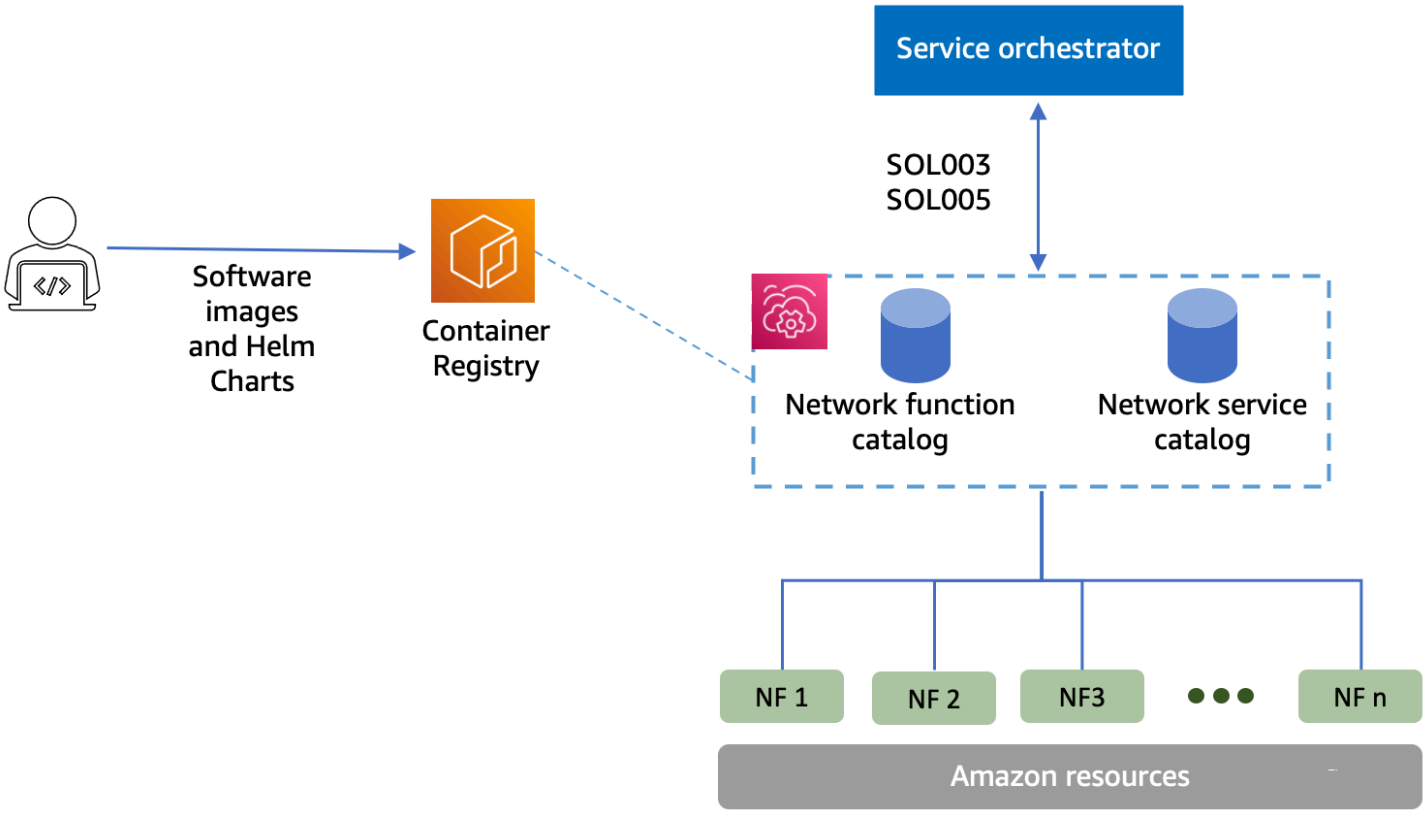
AWS Telco Network Builder とは

AWS Telco Network Builder (AWSTNB) は、通信サービスプロバイダー (CSP) が AWS インフラストラクチャで 5G ネットワークを効率的にデプロイ、管理、スケールできる方法を提供する AWS のサービスです。

AWS TNB を使用すると、コンテナ化されたソフトウェアイメージを自動化して使用し、スケーラブルで安全な 5G ネットワークを AWS クラウド にデプロイできます。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 リソースのプロビジョニングと構成方法を知ったりす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説明し、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てもらいます。AWSTNB は、サードパーティのサービスオーケストレーターや AWS の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、コンテナ化されたネットワーク機能をデプロイ、およびネットワークとアクセス管理を構成して、完全に機能するネットワークサービスを構築します。

次の図は、欧州電気通信標準化機構 (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的な統合を示しています。



トピック

AWS を使用するのは初めてですか?

AWS の製品やサービスを初めて使用する場合、詳細については、以下のリソースを参照してください。

- AWSへの概論
- AWS の開始方法

AWS TNB の対象者

AWS TNB は、ネットワークサービスの設計、デプロイ、管理のためのカスタムスクリプトや設定の作成や管理を行わずに、AWS クラウド が提供するコスト効率、俊敏性、伸縮性を活用したいと考えている CSP 向けです。AWSTNB 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、コンテナ化されたネットワーク機能をデプロイして、CSP が定義したネットワークサービス記述子と CSP がデプロイしたいネットワーク機能に基づいて、完全に機能するネットワークサービスを作成するためのネットワークとアクセス管理を設定します。

AWS TNB を使用する理由

CSP が AWS TNB の使用を検討する理由には、次のようなものがあります。

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サービスオーケストレーターと統合します。

スケーリング

トラフィックの需要に合わせて基盤となる AWS リソースをスケールしたり、ネットワーク機能の更新をより効率的に実行したり、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をロールアウトしたり、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したりするように AWS TNB を設定できます。

AWS リソースの検査とモニタリング

AWS TNB を使用すると、Amazon VPC、Amazon EC2、Amazon EKS など、ネットワークをサポートする AWS リソースを単一のダッシュボードで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) 用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機能のライフサイクルを管理する場合、ネットワーク機能を記述するネットワークサービス記述子に変更を加えるだけで済みます。その後、AWSTNB は必要な変更を自動的に正しい順序で行います。

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 にアクセスする

次のインターフェイスのいずれかを使用して、AWS TNB リソースの作成、アクセス、管理を行うことができます。

- AWS TNB コンソール — ネットワークを管理するための Web 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 を始めとする一連のさまざまな AWS のサービス用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」を参照してください。
- AWS SDK — 言語固有の API 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、CSP が AWS での通信ネットワークのデプロイと管理を自動化できるよう支援します。AWS TNB を利用する場合、次の 2 つの要素につい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、AWS TNB と併用して他の AWS サービスを利用すると、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 のアカウント請求の詳細については、「AWS のアカウント請求」を参照してください。

AWS 請求、アカウント、イベントについてご質問がある場合は、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、およびパフォーマンスの最適化に役立つサービスです。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 の使用を開始するための詳細については、次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

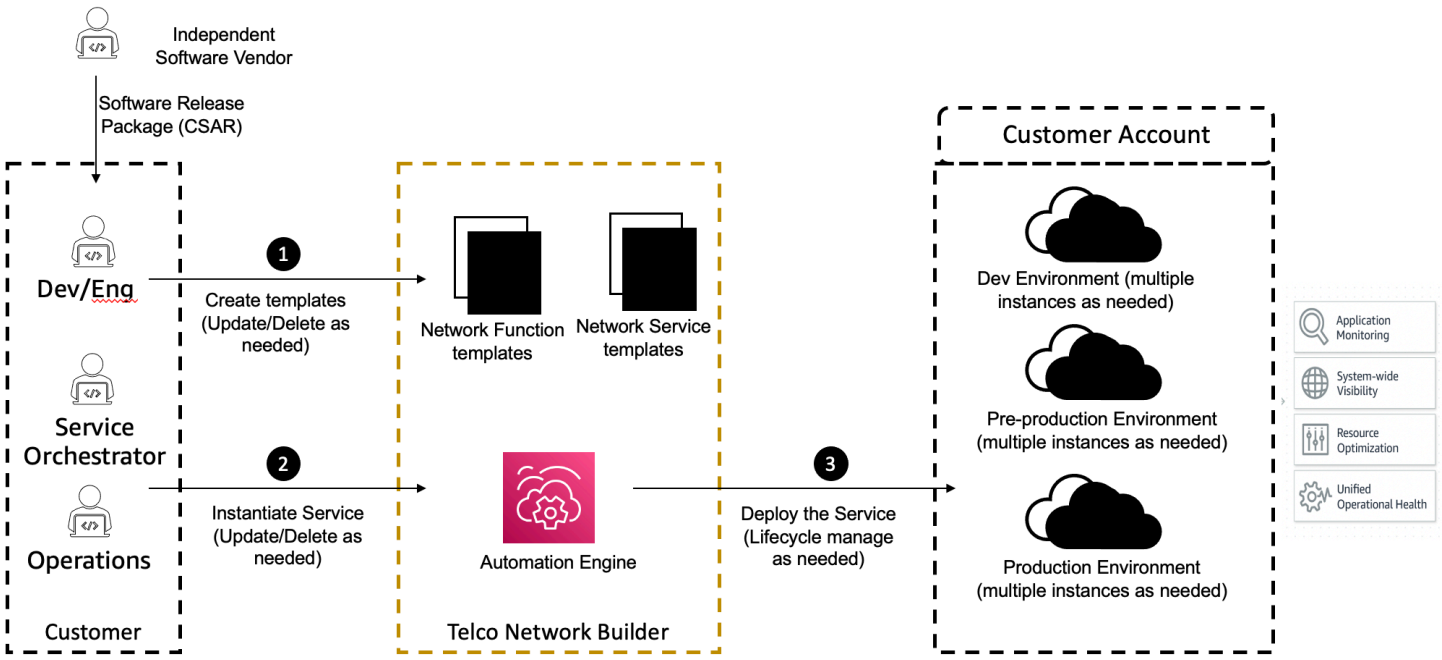
AWS TNB は標準化されたエンドツーエンドのオーケストレーターや AWS リソースと統合して、完全な 5G ネットワークを運用します。

AWS TNB では、ネットワーク機能パッケージとネットワークサービス記述子 (NSD) 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も提供されます。エンドツーエンドのオーケストレーターを使用して AWS TNB API と統合することも、AWS TNB SDK 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 のアーキテクチャ」を参照してください。

トピック

AWS TNB のアーキテクチャ

AWS TNB では、AWS Management Console、AWS CLI、AWS TNB REST API および SDK を使用してライフサイクル管理の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。AWSTNB を使用すると、デプロイを可視化できます。



AWS のサービスとの統合

5G ネットワークは、いくつもの Kubernetes クラスターにデプロイされている、相互接続され、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。AWSTNB は、通信専用の API として次の AWS のサービスと統合し、完全に機能するネットワークサービスを構築します。

- 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)。
- クラスターを設定する Amazon Elastic Kubernetes Service (Amazon EKS)。
- ネットワーキングコンストラクト用の Amazon VPC。
- AWS CloudFormation を使用するセキュリティグループ。
- AWS リージョン、AWS Local Zones、および AWS Outposts にわたるデプロイターゲット用の AWS CodePipeline。
- ロールを定義する IAM。
- AWS TNB API へのアクセス許可を制御する AWS Organizations。
- 健全性をモニタリングし、メトリクスを投稿する AWS Health Dashboard および AWS CloudTrail。

AWS TNB のリソースクォータ

AWS アカウント には、AWS のサービス ごとにデフォルトのクォータ (以前は制限と呼ばれたもの) があります。特に明記されていない限り、各クォータは AWS リージョン 固有です。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「Requesting a quota increase」(クォータ引き上げリクエスト) を参照してください。

お客様の 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	デフォルト値	調整可能?
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、ネットワーク機能を定義する CSAR パッケージ (CU AMF、UPF など) をアップロードし、新しいネットワーク機能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ったときに 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。

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。
- ネットワーク機能を置換または停止します。

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 SDKsCLI、または APIs を使用して、ネットワーク関数をインスタンス化したり、新しいバージョンにアップグレードしたりする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。
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。

の仕様	リリース	説明
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。

AWS TNB のネットワーク関数パッケージ

AWS TNB を使用すると、ETSI SOL001/SOL004 に準拠するネットワーク関数パッケージを関数カタログに保存できます。次に、ネットワーク機能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。

- ネットワーク機能記述子 — パッケージオンボーディングとネットワーク機能管理のためのメタデータを定義します
- ソフトウェアイメージ — ネットワーク機能コンテナイメージを参照します。Amazon Elastic Container Registry (Amazon ECR) は、ネットワーク機能イメージリポジトリとして機能します。
- 追加ファイル — スクリプトや Helm チャートなどのネットワーク機能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、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 仕様については、「」を参照してくださいAWS TNB 用 TOSCA リファレンス 。

ネットワーク機能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"
```

AWS TNB のネットワーク関数サービス記述子

AWS TNB は、デプロイするネットワーク機能、およびそれらをカタログにデプロイする方法に関するネットワークサービス記述子 (NSDs) を保存します。ETSI SOL007 で説明されているように YAML NSD ファイルをアップロードして、次のものを含めることができます。

- デプロイする必要がある NF
- ネットワーク手順
- コンピューティング手順
- ライフサイクルフック (カスタムスクリプト)

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方法でそれらをモデリング AWS のサービス することで、より効率的に を使用できます。

以下は、 の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、自動スケーリング設定を使用して、アプリケーションの を定義 NodeGroups 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。
- 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
```

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
- 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
- IAM ユーザーの作成
- AWS TNB ロールを設定する

にサインアップする AWS

Amazon Web Services にサインアップすると、 AWS アカウント は AWS TNB を含む AWSのすべてのサービスに自動的にサインアップされます。料金は、使用するサービスの料金のみが請求されます。

を AWS アカウント すでにお持ちの場合は、次のタスクに進んでください。 AWS アカウントをお持ちでない場合は、次に説明する手順に従ってアカウントを作成してください。

を作成するには AWS アカウント

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

   にサインアップすると AWS アカウント、 AWS アカウントのルートユーザーが作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行してください。

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

AWS サービスにプログラムで接続するには、 エンドポイントを使用します。標準 AWS エンドポイントに加えて、一部の AWS サービスでは、選択したリージョンで FIPS エンドポイントを提供しています。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	
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	

リージョン名	リージョン	エンドポイント	プロトコル	
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 製品用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

IAM ユーザーの作成

AWS Identity and Access Management (IAM) は、 AWS リソースへのアクセスを安全に制御するのに役立つウェブサービスです。短期の認証情報を使用して AWSにアクセスする IAM ユーザーロールを作成します。

ロールを作成するには、「AWS IAM Identity Center ユーザーガイド」の「はじめに」の手順に従います。

プログラムによるアクセスを設定するには、 AWS Command Line Interface ユーザーガイドの を使用する AWS CLI ように AWS IAM Identity Centerを設定します。

AWS TNB ロールを設定する

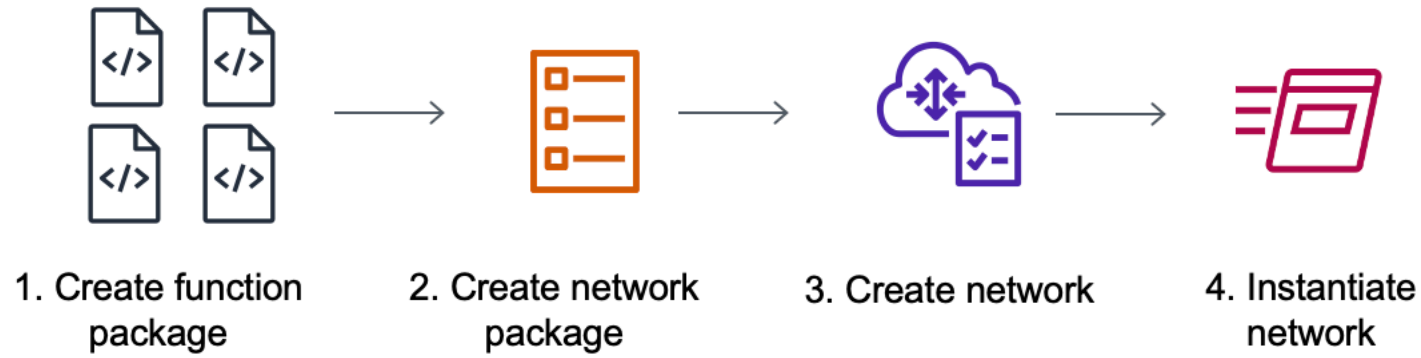
AWS TNB ソリューションのさまざまな部分を管理するには、IAM サービスロールを作成する必要があります。 AWS TNB サービスロールは、、 AWS CloudFormation AWS CodeBuild、およびさまざまなコンピューティングおよびストレージ AWS サービスなどの他のサービスに対して API コールを行い、デプロイ用のリソースをインスタンス化および管理できます。

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 TNB を使用して、集中型ユニット (CU)、アクセスとモビリティ管理機能 (AMF)、5G ユーザープレーン機能 (UPF) などのネットワーク機能を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前に、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 に準拠する Network Service Descriptor (NSD) テンプレート。 SOL007

AWS TNB 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 GitHub を使用できます。

関数パッケージを作成する

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. 「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、CSAR パッケージを.zipファイルとしてアップロードします。
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. [次へ] をクリックします。
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、NSD を.zipファイルとしてアップロードします。
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. [次へ] をクリックします。
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. NSD 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。
6. ネットワークインスタンスの ID を選択し、[インスタンス化] を選択します。
7. [ネットワークをインスタンス化] を選択します。
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ステータスを追跡します。

クリーンアップ

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 用関数パッケージ

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 ネットワーク機能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、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する必要があります。

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. [ファイルを選択] を選択し、NF の CSAR パッケージをアップロードします。
5. [Next] (次へ) をクリックします。
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

AWS CLI を使用して関数パッケージを作成するには

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:

```
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

AWS CLI 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. 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 ネットワーク機能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

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

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 ネットワーク機能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

AWS CLI を使用して関数パッケージを削除するには

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 用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、デプロイする関数パッケージとデプロイ先の AWS インフラストラクチャを定義します。

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。たとえば、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. [ファイルを選択] を選択し、CSAR パッケージをアップロードします。
5. [Next] (次へ) をクリックします。
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

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

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:

```
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

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

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

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

- 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

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

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、AWS TNB で作成されるデプロイ可能な単一のネットワークです。

タスク

- 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 でネットワークインスタンスを表示する
- 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を終了して AWS TNB から削除する

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

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、それを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 はネットワークサービス記述子の仕様に従ってネットワーク機能をデプロイします。

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し、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. [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. NSD を選択します。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。
7. [インスタンス化] を選択します。
8. [ネットワークをインスタンス化] を選択します。
9. 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。

AWS CLI

AWS CLI を使用してネットワークインスタンスを作成し、インスタンス化するには

1. 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"
```

2. 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}$
```

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

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

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. get-sol-network-instance コマンドを使用して、ネットワークインスタンスに関する詳細を表示します。

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

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を更新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の ID を選択します。
4. [関数] タブで、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. 更新のオーバーライドを入力して更新を確定します。
7. [更新] を選択します。
8. 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。

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

update-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 ...
```

ネットワークインスタンスを終了して AWS TNB から削除する

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

AWS CLI 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. 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

- 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をキャンセルする

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

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

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}$
```

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

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

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 用 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、CSP はテンプレートを AWS TNB にアップロードします。これにより、AWS アベイラビリティーゾーン全体で機能する 5G ネットワークを確立するために必要なリソースを統合します。

目次

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 仮想ネットワーク機能 (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

型: 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

型: 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

型: 文字列

`provider`

VNFD の作成者。

必須: はい

型: 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

型: 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

型: 文字列

例

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

NSD 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

型: 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

型: 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コンピューティング。PlacementGroup
- AWSコンピューティング。UserData
- AWSネットワーク。SecurityGroup
- AWSネットワーク。SecurityGroupEgressRule
- AWS.Networking。SecurityGroupIngressRule
- AWS.Resource.Import
- AWS.Networking.ENI
- AWS.HookExecution
- AWS.ネットワーク。InternetGateway
- AWS.ネットワーク。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

型: 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

型: 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

型: 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

`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

型: 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。

、 AWS ローカルゾーン AWS Outposts、または で Amazon EKS セルフマネージド型ノードを使用するには、このプラグインを有効にします 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と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DescribeAddonVersions」の「」を参照してください。

必須: いいえ

タイプ: String

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 1.23 から 1.29 をサポートしています。

必須: はい

型: 文字列

指定できる値: 1.23 | 1.24 | 1.25 | 1.26 | 1.27 | 1.28 | 1.29

`access`

クラスターエンドポイントのアクセス。

必須: はい

型: 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

型: 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: String

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: "1.23"
    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 では、IAM ロールを Amazon EKS クラスターに追加aws-authConfigMapして、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ようにします。

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

型: 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、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する EKS マネージドノードグループをサポートしています。EKS ノードグループを作成するには、AMI の ID または AMI タイプ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) を選択する必要があります。また、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も指定します。ノードグループは EKS クラスターに関連付けられている必要があります。ワーカーノードのサブネットを指定する必要があります。

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できます。

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
    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 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

型: 文字列

使用できる値: AL2_x86_64 | AL2_x86_64_GPU | AL2_ARM_64 | 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: String

> **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

型: 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

型: 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: String

> **Note**
>
> - `root_volume_encryption` が false の場合は、 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、 TNB AWS `root_volume_encryption_key_arn`の を含めてルートボリュームを再暗号化する必要があります。
> - 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 を使用してルートボリューム`root_volume_encryption_key_arn`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。

`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 のインスタンス数 NodeGroup。

必須: はい

タイプ: 整数

`min_size`

この のインスタンスの最小数 NodeGroup。

必須: はい

タイプ: 整数

`max_size`

この のインスタンスの最大数 NodeGroup。

必須: はい

タイプ: 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

型: 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

`cluster`

AWS.Compute.EKS ノード。

必須: はい

型: 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`プロパティを含めた場合、 AWS TNB は `multus`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。 ENIs 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。コンピューティングPlacementGroupノード。

必須: いいえ

タイプ: String

`user_data`

tosca.nodes AWS。Compute.UserData ノードリファレンス。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: String

`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。次の基準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*。

例えば、次のようになります: `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"
    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"
  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 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するために、Amazon EKS セルフマネージドノードをサポートしています。Amazon EKS ノードグループを作成するには、AMI の ID 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) を選択する必要があります。オプションで、SSH アクセス用の Amazon EC2 キーペアを指定します。また、インスタンスタイプ、必要なサイズ、最小サイズ、および最大サイズも指定する必要があります。ノードグループは Amazon EKS クラスターに関連付けられている必要があります。ワーカーノードのサブネットを指定する必要があります。

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できます。

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
    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。

必須: はい

型: 文字列

`instance_type`

インスタンスのサイズ。

必須: はい

型: 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

型: 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

型: 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: String

> **Note**
>
> - `root_volume_encryption` が false の場合は、 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、 TNB AWS `root_volume_encryption_key_arn`の を含めてルートボリュームを再暗号化する必要があります。
> - 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 を使用してルートボリューム`root_volume_encryption_key_arn`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。

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 のインスタンス数 NodeGroup。

必須: はい

タイプ: 整数

`min_size`

この のインスタンスの最小数 NodeGroup。

必須: はい

タイプ: 整数

`max_size`

この のインスタンスの最大数 NodeGroup。

必須: はい

タイプ: 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

型: 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

型: 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`プロパティを含めた場合、 AWS TNB は `multus`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。 ENIs 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。コンピューティングPlacementGroupノード。

必須: いいえ

タイプ: String

`user_data`

tosca.nodes AWS。Compute.UserData ノードリファレンス。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: String

`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。次の基準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*。

例えば、次のようになります: `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"
    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コンピューティング。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

型: 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) アプリケーションに典型的な密結合 node-to-node 通信に必要な低レイテンシーの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`

ターゲットパーティション数。

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コンピューティング。UserData

AWS TNB は、Network Service Descriptor (NSD) の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 UserData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介して 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 AWS 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 AWS 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

型: 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

型: 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

型: 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

型: 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

型: 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、. AWS Networking 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SecurityGroup。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: String

`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: String

`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: String

`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: String

`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

型: 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

型: 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

型: 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

型: 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: String

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.ネットワーク。SecurityGroupIngressRule

AWS TNB は、. AWS Networking 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SecurityGroup。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: String

`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: String

`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: String

`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: String

`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

型: 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

型: 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

型: 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

型: 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: String

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

タイプ: 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。

使用できる値: 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

型: 文字列

`security_groups`

AWS.Networking.SecurityGroup ノード。

必須: いいえ

タイプ: String

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: String

使用できる値: CODE_BUILD

要件

definition

AWS HookDefinition。Bash ノード。

必須: はい

型: 文字列

vpc

AWS.Networking.VPC ノード。

必須: はい

型: 文字列

例

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

```
    vpc: SampleVPC
```

AWS.ネットワーク。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」。

型: 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」。

型: 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

型: 文字列

`route_table`

AWS.Networking.RouteTable ノード。

必須: はい

型: 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ネットワーク。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

型: 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

型: 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、 (米国西部 us-west-2 (オレゴン)) など、 AWS リージョン内の AWS アベイラビリティーゾーンをサポートします。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

型: 文字列

`cidr_block`

サブネットの CIDR ブロック。

必須: いいえ

タイプ: String

`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: String

`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: String

`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: String

`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

型: 文字列

`route_table`

AWS.Networking.RouteTable ノード。

必須: はい

型: 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: String

`cluster`

AWS.Compute.EKS ノード。

必須: はい

型: 文字列

`vnfs`

AWS.VNF ノード。

必須: はい

型: 文字列

インターフェイス

Hooks

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: String

`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: String

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

型: 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: String

`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、それを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

型: 文字列

`internet_gateway`

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

必須: はい

型: 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

型: 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: String

`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

宛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

プロパティ

`nat_gateway`

AWS.Networking.NATGateway ノードのリファレンス。

必須: はい

型: 文字列

`route_table`

AWS.Networking.RouteTable ノードリファレンス。

必須: はい

型: 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
```

一般的なノード

NSD と VNFD で使用するノードを定義します。

- AWS.HookDefinition.Bash

AWS.HookDefinition.Bash

bash で AWS HookDefinition を定義します。

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

型: 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: **envName**=**envValue** と次の正規表現: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`

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。

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

型: 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 Telco Network Builder のセキュリティ

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 とユーザーの間で共有される責任です。責任共有モデルではこれを、クラウドのセキュリティ、およびクラウド内で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。また、 は、お客様が安全に使用できるサービス AWS も提供します。コンプライアンスAWS プログラムコンプライアンスプログラムコンプライアンス の一環として、サードパーティーの監査者は定期的にセキュリティの有効性をテストおよび検証。 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、お客様は、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 の使用時に責任共有モデルを適用する方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するために AWS TNB を設定する方法を示します。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 の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、 AWS Telco Network Builder 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。お客様は、このインフラストラクチャでホストされてい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、「データプライバシーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された記事「AWS 責任共有モデルおよび GDPR」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 Identity Center 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 は必須であり TLS 1.3 がお勧めです。
- 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。
- 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 などの高度なマネージド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 140-2 検証済みの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) 140-2」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は、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配置し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 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

データの処理

AWS アカウントを閉鎖すると、 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、 を通じて自動的に行われます 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 リソースの使用を承認する (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 AWS のサービス 使用できる です。

内容

対象者

AWS Identity and Access Management (IAM) の使用方法は、 AWS TNB で行う作業によって異なります。

サービスユーザー – AWS TNB サービスを使用してジョブを実行する場合、管理者から必要な認証情報とアクセス許可が与えられます。さらに多くの AWS TNB 機能を使用して作業を行う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするうえで役立ちます。 AWS TNB の機能にアクセスできない場合は、「AWS Telco Network Builder 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」を参照してください。

サービス管理者 – 社内の AWS TNB リソースを担当している場合は、通常、 TNB AWS へのフルアクセスがあります。サービスユーザーがどの AWS TNB 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で AWS TNB で IAM を使用する方法の詳細については、「」を参照してくださいAWS Telco Network Builder と IAM の連携方法。

IAM 管理者 – IAM 管理者は、 AWS TNB へのアクセスを管理するポリシーの作成方法の詳細について確認する場合があります。IAM で使用できる 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AWS Telco Network Builder のアイデンティティベースポリシーの例。

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。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けて認証（ にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center（ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーティッド ID の例です。フェデレーティッドアイデンティティとしてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 ユーザーガイド」の「 にサインインする方法 AWS アカウントAWS サインイン 」を参照してください。

AWS プログラムで にアクセスする場合、 は Software Development Kit (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。 AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM ユーザーガイドの API AWS リクエストの署名を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、 AWS では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させることをお勧めします。詳細については、『AWS IAM Identity Center ユーザーガイド』の「Multi-factor authentication」(多要素認証) および『IAM ユーザーガイド』の「AWSにおける多要素認証 (MFA) の使用」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、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。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。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、『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。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、一時的な認証情報を使用して にアクセスするための ID プロバイダーとのフェデレーションの使用を要求 AWS のサービス します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリのユーザー、または ID ソースを通じて提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID が にアクセスすると AWS アカウント、ロールが引き受けられ、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成することも、独自の ID ソース内のユーザーとグループのセットに接続して同期して、すべての AWS アカウント とアプリケーションで使用することも

できます。IAM Identity Center の詳細については、『AWS IAM Identity Center ユーザーガイド』の「What is IAM Identity Center?」(IAM Identity Center 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、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーション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「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する権限を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時的な認証情報が提供されます。詳細については、『IAM ユーザーガイド』の「IAM ユーザー (ロールではなく) の作成が適している場合」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 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。ロール を切り替える AWS Management Console ことで、 で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、 または AWS API AWS CLI オペレーションを呼び出すか、カスタム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「IAM ロールの使用」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッドアイデンティティに権限を割り当てるには、ロールを作成してそのロールの権限を定義します。フェデレーティッドアイデンティティが認証されると、そのアイデンティティはロールに関連付けられ、ロールで定義されている権限

が付与されます。フェデレーションの詳細については、『IAM ユーザーガイド』の「サードパーティーアイデンティティプロバイダー向けロールの作成」を参照してください。IAM アイデンティティセンターを使用する場合、権限セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。権限セットの詳細については、『AWS IAM Identity Center ユーザーガイド』の「権限セット」を参照してください。

- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、(ロールをプロキシとして使用する代わりに) ポリシーをリソースに直接アタッチできます。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違いについては、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。
- クロスサービスアクセス — 一部の は、他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの権限、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使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して でアクションを実行する場合 AWS、ユーザーはプリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。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。
  - サービスロール - 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。
  - 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、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。 AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、『IAM ユーザーガイド』の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して権限を付与する」を参照してください。

IAM ロールと IAM ユーザーのどちらを使用するかについては、『IAM ユーザーガイド』の「(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?」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、 AWS ID またはリソースにアタッチします。ポリシーは、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) AWS 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。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。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構造と内容の詳細については、「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 AWS JSON 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んなリソースにどんな条件でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。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。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの権限を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

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ポリシーは、IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 権限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 ポリシーの作成」を参照してください。

アイデンティティベースポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたは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「マネージドポリシーとインラインポリシーの比較」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。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制御できます。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は、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。

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。

アクセスコントロールリスト (ACL)

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権限を持つかをコントロールします。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。

Amazon S3、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、『Amazon Simple Storage Service デベロッパーガイド』の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティの許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) に対する最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、 AWS アカウント ビジネスが所有する複数の をグループ化して一元管理するサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、『AWS Organizations ユーザーガイド』の「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。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権限を理解するのがさらに難しくなります。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 Telco Network Builder 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、 TNB で使用できる IAM AWS 機能について学びます。

AWS Telco Network Builder で使用できる IAM の機能

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	Yes
リソースベースのポリシー	No
ポリシーアクション	Yes
ポリシーリソース	Yes
ポリシー条件キー	Yes
ACL	No
ABAC (ポリシー内のタグ)	はい
一時的な認証情報	Yes
プリンシパル権限	Yes
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を把握するには、「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ポリシーをサポートする	Yes

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 権限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 ポリシーの作成」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。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素については、「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AWS Telco Network Builder のアイデンティティベースポリシーの例。

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	No

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制御できます。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は、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、アカウント全体、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。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる にある場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリ

ンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーを追加する必要はありません。詳細については、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」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に対するサポート	はい

管理者は AWS JSON 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んなリソースにどんな条件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。ポリシーアクションの名前は通常、関連付けられた AWS API オペレーションと同じです。一致する API オペレーションのない権限のみのアクションなど、いくつかの例外があります。また、ポリシーに複数アクションが必要なオペレーションもあります。これらの追加アクションは、 依存アクション と呼ばれます。

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権限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、「サービス認証リファレンス」の AWS 「Telco Network Builder 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AWS Telco Network Builder のアイデンティティベースポリシーの例。

AWS TNB のポリシーリソース

ポリシーリソースに対するサポート	はい

管理者は AWS JSON 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どのような条件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。ステートメントには、Resource または NotResource要素を含める必要があります。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。これは、リソースレベルの権限と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限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」のAWS 「Telco Network Builder で定義されるリソース」を参照してください。 どのアクションで各リソースの ARN を指定できるかについては、AWS 「Telco Network Builder 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AWS Telco Network Builder のアイデンティティベースポリシーの例。

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	はい

管理者は AWS JSON 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んなリソースにどんな条件でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。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することで、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。

1 つのステートメントに複数の Condition 要素を指定するか、1 つの Condition 要素に複数のキーを指定すると、 AWS は AND 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、『IAM ユーザーガイド』の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートします。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、「サービス認証リファレンス」の AWS 「Telco Network Builder 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、AWS 「Telco Network Builder 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AWS Telco Network Builder のアイデンティティベースポリシーの例。

AWS TNB ACLs

ACL のサポート	No

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権限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 での ABAC

ABAC のサポート (ポリシー内のタグ)	はい

属性ベースのアクセス制御 (ABAC) は、属性に基づいてアクセス許可を定義するアクセス許可戦略です。では AWS、これらの属性はタグ と呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付け

は、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合に操作を許可するように ABAC ポリシーを設計します。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

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はいです。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 とは?」を参照してください。ABAC をセットアップするステップ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「属性に基づく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	はい

一部の AWS のサービス は、一時的な認証情報を使用してサインインすると機能しません。一時的な認証情報 AWS のサービス を使用する などの詳細については、IAM ユーザーガイドのAWS のサービス「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。例えば、会社の Single Sign-On (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ロールへの切り替え (コンソール)」を参照してください。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用して . AWS recommends にアクセスできます AWS。これは、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することを推奨しています。詳細については、「IAM の一時的セキュリティ認証情報」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

フォワードアクセスセッション (FAS) をサポート	はい

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。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	いいえ

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 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	いいえ

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、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

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

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。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできません。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。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

これらサンプルの 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 ポリシーの作成」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、「サービス認証リファレンス」のAWS 「 Telco Network Builder 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

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウントで誰かが AWS TNB リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。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する – ユーザーとワークロードにアクセス許可を付与するには、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して、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「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで権限を設定するときは、タスクの実行に必要な権限のみを付与します。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、最小特権権限とも呼ばれています。IAM を使用して権限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 でのポリシーと権限」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。例えば、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、 などの特定の を介してサービスアクションが使用される場合に 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 AWS CloudFormation。詳細については、IAM ユーザーガイドの [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素：条件) を参照してください。
- IAM Access Analyzer 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 Access Analyzer 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 ポリシーの検証」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、セキュリティを強化するために MFA を有効にします。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 保護 API アクセスの設定」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

AWS Telco Network Builder 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の定義に従って 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、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 の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として、管理者として AWS TNB サービスロールを作成して TNB AWS 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 AWS 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。このポリシーに対するアクセス許可をスコープダウンする場合、 AWS TNB 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。

```
{
    "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",
                "iam:PassRole"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TNBAccessComputePerms"
        },
        {
            "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:UntagResource",
                "eks:DescribeCluster",
                "eks:ListNodegroups",
                "eks:CreateAddon",
                "eks:DeleteAddon",
                "eks:DescribeAddon",
                "eks:DescribeAddonVersions",
                "s3:PutObject",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks",
                "cloudformation:UpdateTerminationProtection"
            ],
            "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*"
            ],
            "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、 TNB AWS サービス信頼ポリシーを示しています。

```
{
  "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 AWS 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 AWS 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 AWS 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": "*"
        }
    ]
}
```

AWS Telco Network Builder 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ない
- iam を実行する権限がありません。PassRole
- 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ない

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。サインイン資格情報を提供した担当者が管理者です。

iam を実行する権限がありません。PassRole

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権限が必要です。

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権限が必要です。Mary には、ロールをサービスに渡す権限がありません。

```
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いAWS Telco Network Builder 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) へのアクセス権限」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、「IAM ユーザーガイド」の「IAM ロールとリソースベースのポリシーとの相違点」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、コンプライアンスプログラムAWS のサービス による対象範囲内のコンプライアンスプログラムを参照し、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、「 でのレポートのダウンロード AWS Artifact」の」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。 は、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境 AWS を にデプロイする手順について説明します。
- アマゾン ウェブ サービスにおける HIPAA セキュリティとコンプライアンスのアーキテクチャ — このホワイトペーパーでは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法について説明します。

> **Note**
>
> すべて AWS のサービス HIPAA の対象となるわけではありません。詳細については、「HIPAA 対応サービスのリファレンス」を参照してください。

- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド — コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS のサービス 、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコントロールにマッピングするためのベストプラクティスをまとめています。
- 「 デベロッパーガイド」の「ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS のサービス 、 内のセキュリティ状態を包括的に確認できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、「Security Hub のコントロールリファレンス」を参照してください。
- Amazon GuardDuty – これにより AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。 GuardDuty は、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件への対応に役立ちます。
- AWS Audit Manager – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

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、高冗長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イルオーバーする

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が高く、フォールトトレラントで、スケーラブルです。

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 リージョンの仮想プライベートクラウド (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 AWS 環境を設計するには、「 Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

が AWS 公開している 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。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2 は必須で TLS 1.3 がお勧めです。
- 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。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

責任共有の例をいくつか示します。

- AWS は、以下を含む TNB AWS 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、以下を含む (ただしこれらに限定されない) 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。
  - IAM ユーザー、グループ、ロール、ポリシー
  - AWS TNB のデータを保存するために使用する 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 を通じ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 を通じ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 内で別々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 Management Console、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IMDSv22 の使用」を参照してください。 Amazon EC2

AWS TNB をモニタリングする

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、およびパフォーマンスを維持するうえで重要な部分です。AWS には、AWS TNB を監視したり、問題が発生したときに報告したり、必要に応じて自動アクションを実行したりするための AWS CloudTrail が用意されています。

CloudTrail を使用して、AWS API に対して実行された呼び出しに関する詳細情報を取得できます。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 TNB 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。

AWS CloudTrail による AWS Telco Network Builder API コールのログ記録

AWS Telco Network Builder は、AWS TNB のユーザー、ロール、または AWS のサービス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 AWS CloudTrail と統合されています。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。追跡を作成すると、AWS TNB のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的デリバリーを有効にすることができます。追跡を設定しない場合でも、CloudTrail コンソールの [Event history] (イベント履歴) で最新のイベントを表示できます。CloudTrail で収集した情報を使用して、AWS TNB に対して行ったリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

CloudTrail の詳細については、AWS CloudTrail ユーザーガイドを参照してください。

CloudTrail での AWS TNB 情報

CloudTrail は、アカウント作成時に AWS アカウント で有効になります。AWS TNB でアクティビティが発生すると、そのアクティビティは [イベント履歴] で他の AWS サービスのイベントとともに CloudTrail イベントに記録されます。最近のイベントは、AWS アカウント で表示、検索、ダウン

ロードできます。詳細については、「CloudTrail イベント履歴でのイベントの表示」を参照してください。

AWS TNB に関するイベントを含めた AWS アカウント内でのイベントの継続的な記録については、追跡を作成します。追跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョン に適用されます 証跡は、AWS パーティションのすべてのリージョンからのイベントをログに記録し、指定した Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集したイベントデータをより詳細に分析し、それに基づく対応するためにその他の AWS のサービスを設定できます。詳細については、次を参照してください。

- 「追跡を作成するための概要」
- CloudTrail がサポートされているサービスと統合
- CloudTrail の Amazon SNS 通知の設定
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての AWS TNB アクションは CloudTrail によってログに記録され、「AWS Telco Network Builder API リファレンス」に文書化されています。例えば、`CreateSolFunctionPackage`、`CreateSolNetworkInstance`、`CreateSolNetworkPackage`の各アクションを呼び出すと、CloudTrail ログファイルにエントリが生成されます。

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。

- リクエストが、ルート認証情報と AWS Identity and Access Management (IAM) ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーティッド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。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

AWS TNB ログファイルエントリの概要

「トレイル」は、指定した Simple Storage Service (Amazon S3) バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルには、単一か複数のログエントリ

があります。イベントはあらゆるソースからの単一のリクエストを表し、リクエストされたアクション、アクションの日時、リクエストのパラメータなどの情報が含まれます。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、特定の順序では表示されません。

次の例は、CreateSolFunctionPackageアクションを示す CloudTrail ログエントリです。

```
{
    "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"
}
```

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、 TNB AWS 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。
FunctionDeletion	名前に変更なし	AWS TNB リソースからネットワーク機能を削除します。
FunctionInstantiation	FunctionInstall	HELM を使用してネットワーク機能をデプロイします。

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
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た AWS リソースのプロビジョニングを解除します。
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。
KubernetesClusterConfiguration	ClusterConfiguration	Kubernetes クラスターを設定し、NSD で定義されている AuthMap ように Amazon EKS に 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。
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。

AWS Telco Network Builder の Service Quotas

サービスクォータ (制限とも呼ばれます) 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 Service Quotas を次に示します。

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	はい	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	はい	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	はい	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	はい	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
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は、AWS.Networking.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.EKS ManagedNode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.EKS ManagedNodeまたは AWS.Compute.EKSSelfManagedNode ノードのノー	2024 年 3 月 19 日

	ドグループにアタッチできます。	
ネットワークインターフェイスのサポート source_dest_check	AWS.Networking.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	.Compute AWS.UserData node を使用して、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、使用可能な場合は multus_role プロパティから、または ENIs に関連するアクセス許可を取得します。node_role	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 設定のノード AWS.Networking.VPC、AWS.Networking.Subnet、AWS.Networking.InternetGateway、AWS.Networking.SecurityGroupIngressRule、AWS.Networking.SecurityGroupEgressRule、.AWSCompute.EKS を確認します。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 アクセス許可の Amazon EKS ノードグループの AWS 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	Amazon S3 および の AWS TNB サービスロールポリシー AWS CloudFormation にアクセス許可を追加し、インフラストラクチャのインスタンス化を有効にしました。Amazon S3	2023 年 10 月 23 日

AWS TNB がより多くのリージョンで利用可能に	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
AWS.Compute.EKS のタグ SelfManagedNode	AWS TNB は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 は IMDSv2 を利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
のアクセス許可を更新しました MultusRoleInlinePolicy	に アクセスec2:DeleteNetworkInterface 許可が含まれる MultusRoleInlinePolicy 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 AuthRole 、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 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。	NSD テンプレートに AWS.Networking.SecurityGroup、AWS.Networking.SecurityGroupEgressRule、AWS.Networking.SecurityGroupIngressRule を追加しました。	2023 年 7 月 18 日
クラスター用の Kubernetes バージョン	AWS TNB は、Kubernetes バージョン 1.22 から 1.26 をサポートし、Amazon EKS クラスターを作成します。 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。