aws

ユーザーガイド

AWS Telco Network Builder

AWS Telco Network Builder: ユーザーガイド

Table of Contents

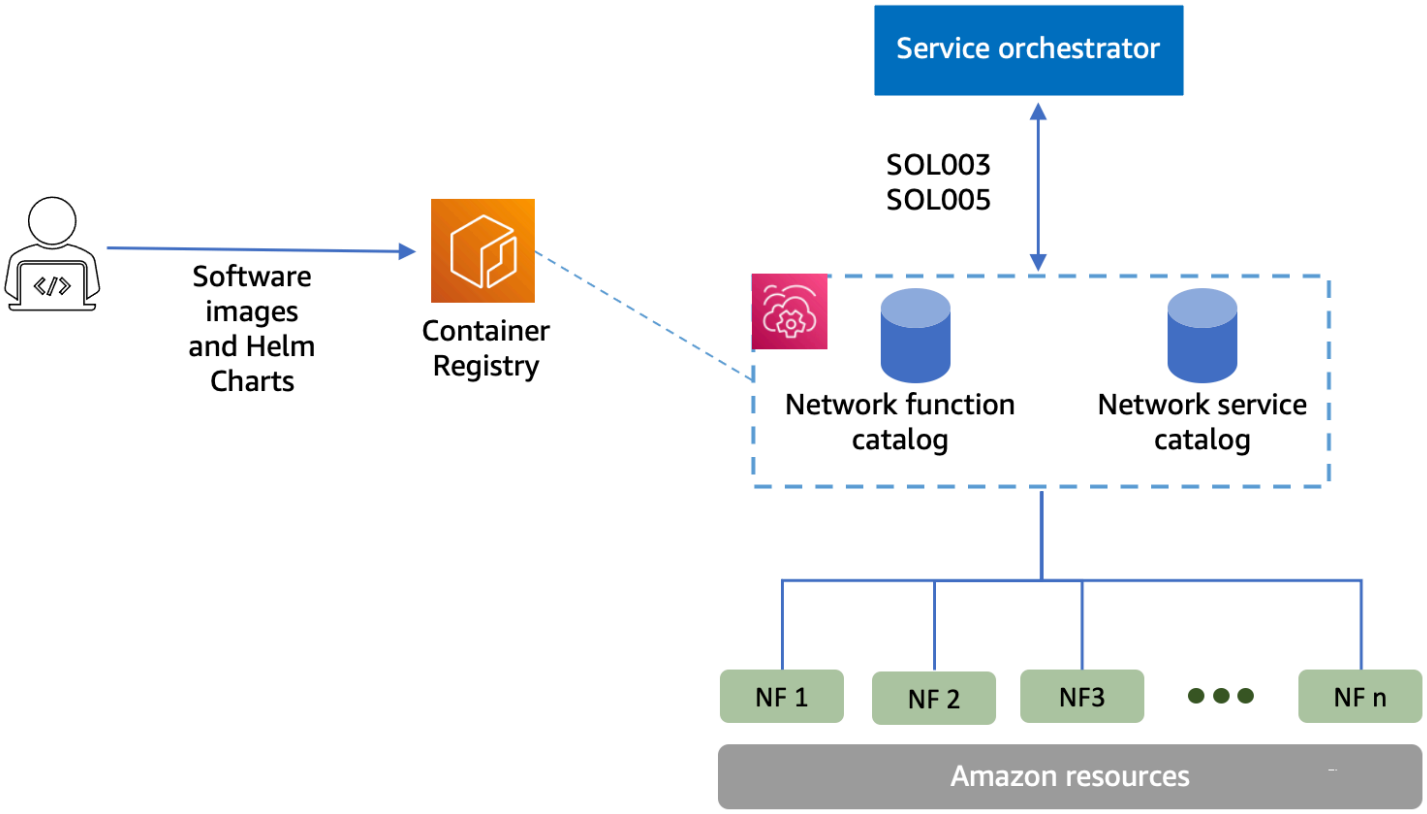
AWS Telco Network Builder とは

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

を使用すると AWS TNB、ネットワークのイメージ AWS クラウド を使用して、スケーラブルで安全な 5G ネットワークを自動的に 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 リソースをプロビジョニングして設定する方法を理解し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記述し、独立したソフトウェアベンダー (ISV) パートナーからネットワーク機能のソフトウェアイメージを提供します。 AWS TNB は、サードパーティーのサービスオーケストレーターと AWS のサービスを統合して、必要な AWS インフラストラクチャを自動的にプロビジョニングし、コンテナ化されたネットワーク機能をデプロイし、ネットワークとアクセス管理を設定して、完全に運用可能なネットワークサービスを作成します。

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



トピック

は初めて AWSですか？

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

- AWSへの概論
- の開始方法 AWS

AWS TNB の対象

AWS TNB は、ネットワークサービスを設計、デプロイ、管理するためのカスタムスクリプトと設定を記述および維持することなく、 AWS クラウド が提供するコスト効率、俊敏性、および弾力性を活用CSPsしたいと考えています。 AWS TNB は、 に必要な AWS インフラストラクチャを自動的にプロビジョニングし、コンテナ化されたネットワーク機能をデプロイし、ネットワークとアクセス管理を設定して、 CSPがデプロイCSPしたいネットワークサービス記述子とネットワーク機能に基づいて、完全に運用可能なネットワークサービスを作成します。

AWS TNB 機能

以下は、 が CSPを使用する理由の一部です AWS TNB。

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に準拠する一般的なサードパーティーのサービスオーケストレーターと統合されています。

スケーリング

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

AWS リソースを検査およびモニタリングする

AWS TNB では、Amazon 、Amazon 、Amazon などの単一のダッシュボードでネットワークをサポートする AWS リソースを検査VPCEC2およびモニタリングできますEKS。

サービステンプレートのサポート

AWS TNB では、すべてのテレコムワークロード (RAN、コア、) のサービステンプレートを作成できますIMS。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 インスタンスEC2タイプのインスタンスタイプを変更するなど、ネットワーク関数のデプロイの基盤となる設定を変更すると、反復可能でスケーラブルな方法で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。を使用して AWS TNBネットワーク関数のライフサイクルを管理する場合、ネットワーク関数を説明するネットワークサービス記述子にのみ変更を加えます。 AWS TNB は、必要な変更を正しい順序で自動的に行います。

ネットワーク機能のライフサイクルの簡素化

ネットワーク機能の最初のバージョンとそれ以降のすべてのバージョンを管理し、アップグレードのタイミングを指定できます。RAN、コア、IMS、および 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。

アクセス AWS TNB

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

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

- AWS Command Line Interface （ AWS CLI ） — を含む幅広い AWS サービスのセットにコマンドを提供します AWS TNB。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 SDKs – 言語固有の を提供しAPIs、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、AWSSDKs「」を参照してください。

の料金 AWS TNB

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

- マネージドネットワーク関数項目 (MNFI) 時間別。
- API リクエストの数別。

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

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

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

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

次のステップ

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

- のセットアップ AWS TNB – 前提条件の手順を完了します。
- の開始方法 AWS TNB – 集中ユニット (CU)、アクセスとモビリティ管理機能 ()、ユーザープレーン機能 (UPF) AMF、または完全な 5G Core などの最初のネットワーク関数をデプロイします。

の AWS TNB仕組み

AWS TNB は、標準化された end-to-end オーケストレーターと 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。

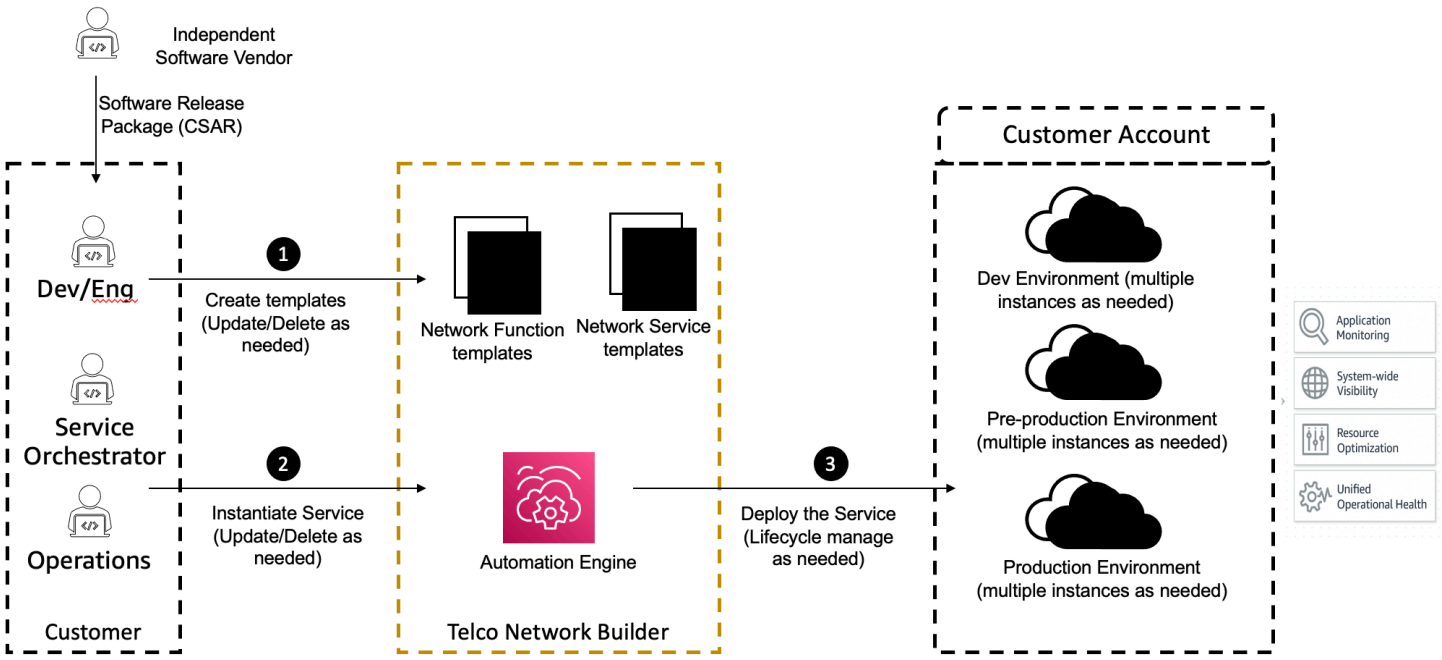
AWS TNB では、ネットワーク関数パッケージとネットワークサービス記述子 (NSDs) を取り込むことができ、ネットワークを運用するためのオートメーションエンジンが提供されます。オーケストレーターを使用して end-to-end と AWS TNB統合することもAPIs、 SDKs を使用して AWS TNB独自のオートメーションフローを構築することもできます。詳細については、「AWS TNB アーキテクチャ」を参照してください。

トピック

- AWS TNB アーキテクチャ
- との統合 AWS のサービス
- AWS TNB リソースクォータ

AWS TNB アーキテクチャ

AWS TNB では、 AWS Management Console、 AWS CLI、 AWS TNBAPI、および REST を使用してライフサイクル管理オペレーションを実行できますSDKs。これにより、エンジニアリング、オペレーション、プログラム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 を利用できるようになります AWS TNB。ネットワーク関数パッケージを 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、および Network Function Descriptor () が含まれていますNFD。テンプレートを使用して、そのパッケージの複数の設定を繰り返しデプロイできます。デプロイするインフラストラクチャとネットワーク機能を定義するネットワークサービステンプレートを作成します。パラメータオーバーライドを使用して、さまざまな設定を異なる場所にデプロイできます。その後、 テンプレートを使用してネットワークをインスタンス化し、ネットワーク機能を AWS インフラストラクチャにデプロイできます。 AWS TNB は、デプロイの可視性を提供します。



との統合 AWS のサービス

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

- 独立系ソフトウェアベンダー (ECR) ネットワーク関数アーティファクトを保存する Amazon Elastic Container Registry (Amazon ISVs) 。
- Amazon Elastic Kubernetes Service (Amazon EKS) でクラスターを設定します。
- Amazon VPC for Networking コンストラクト。
- を使用するセキュリティグループ AWS CloudFormation。
- AWS CodePipeline AWS リージョン、 AWS ローカルゾーン、および 全体のデプロイターゲット。 AWS Outposts
- IAM ロールを定義します。
- AWS Organizations は、 へのアクセス AWS TNBを制御しますAPIs。
- AWS Health Dashboard および AWS CloudTrail を使用して、ヘルスをモニタリングし、メトリクスを投稿します。

AWS TNB リソースクォータ

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

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

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

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

リソースクォータ	説明	デフォルト値	引き上げ可能?
ネットワークサービスインスタンス	1 つのリージョンのネットワークサービスインスタンスの最大数。	800	可能
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	可能
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	可能
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、 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 AWS TNB。

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. デプロイまたはアップグレードするCSARパッケージを作成します。
4. を使用して AWS TNB、ネットワーク関数を定義するCSARパッケージ (CU 、 などUPF) をアップロードしAMF、新しいネットワーク関数ソフトウェアイメージまたはカスタマースクリプトが利用可能になったときに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統合と継続的配信 (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. 情報を使用して、ネットワークサービス記述子 () を作成しますNSD。
3. ネットワーク関数と、ネットワーク関数がインスタンス化するために必要なリソースNSDsを定義する取り込み。

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. で定義されているネットワーク関数をインスタンス化 (またはプロビジョニング) NSDし、トラフィックの伝達を開始します。
3. アセットを検証します。

本番稼働

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

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

標準化されたインターフェースの使用

AWS TNB は、European Telecommunications Standards Institute (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SDKs、、CLI、または を使用して AWS TNB、ネットワーク関数を新しいバージョンにインスタンス化またはアップグレードするなどのオペレーションAPIsを開始できます。

AWS TNB では、次の仕様がサポートされています。

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関数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	ネットワーク関数パッケージの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。

のネットワーク関数パッケージ AWS TNB

を使用すると AWS TNB、SOL001/00SOL4 ETSI に準拠した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、ネットワーク関数を説明するアーティファクトを含む Cloud Service Archive (CSAR) パッケージをアップロードできます。

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

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"
```

のネットワークサービス記述子 AWS TNB

AWS TNB には、デプロイするネットワーク関数と、カタログへのデプロイ方法に関するネットワークサービス記述子 (NSDs) が保存されます。SOL007 で説明されているように、YAMLNSDファイル (vnfd.yaml) ETSI をアップロードして、次の情報を含めることができます。

- デプロイするネットワーク関数
- 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 は、 ETSI準拠のサービスオーケストレーターが理解できるようにモデリング AWS のサービス することで、 をより効率的に使用できるようにします。

以下は、 をモデル化する方法NSDを示す のスニペットです AWS のサービス。ネットワーク関数は、Kubernetes バージョン 1.27 の 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
```

の管理とオペレーション AWS TNB

を使用すると AWS TNB、003 および ETSI SOL00SOL5 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。を使用して AWS TNBAPIs、次のようなライフサイクルオペレーションを実行できます。

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

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

のネットワークサービス記述子 AWS TNB

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関数とAWS 、ネットワーク関数をデプロイする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。を定義NSD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、 でサポートされているNSDTOSCAスキーマを理解する必要がありますAWS TNB。

NSD ファイルは、次の部分に分かれています。

1. TOSCA 定義バージョン – これはNSDYAMLファイルの最初の行であり、次の例に示す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 – 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関数の定義NSD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、ネットワーク関数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、名前はNSDYAMLファイル全体でより簡単に参照できるように、一意の ID に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 Node Group リソースを作成するには、Amazon EKS クラスター内に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
```

のセットアップ AWS TNB

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

タスク

- にサインアップする AWS アカウント
- 管理アクセスを持つユーザーを作成する
- AWS リージョンを選択する
- サービスエンドポイントに関する注記
- (オプション) をインストールする AWS CLI
- AWS TNB ロールのセットアップ

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. https://portal.aws.amazon.com/billing/サインアップ 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 は、サインアッププロセスが完了した後に確認 E メールを送信します。/ に移動し、マイアカウント を選択すると、いつでも現在のアカウントアクティビティを表示https://aws.amazon.comし、アカウントを管理できます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、 のセキュリティを確保し AWS アカウントのルートユーザー、 を有効にして管理ユーザーを作成し AWS IAM Identity Center、日常的なタスクにルートユーザーを使用しないようにします。

のセキュリティ保護 AWS アカウントのルートユーザー

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS Management Consoleとして にサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドのルートユーザーとしてサインインするを参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、「 ユーザーガイド」の AWS アカウント 「ルートユーザー (コンソール) の仮想MFAデバイスの有効化」を参照してください。 IAM

管理アクセスを持つユーザーを作成する

1. IAM Identity Center を有効にします。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。
2. IAM Identity Center で、ユーザーに管理アクセスを許可します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、 AWS IAM Identity Center ユーザーガイドの「デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM Identity Center ユーザーでサインインするには、IAMIdentity Center ユーザーの作成時に E メールアドレスにURL送信されたサインインを使用します。

  IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン「 ユーザーガイド」のAWS 「 アクセスポータルへのサインイン」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM Identity Center で、最小権限のアクセス許可を適用するベストプラクティスに従うアクセス許可セットを作成します。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。
2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの参加」を参照してください。

AWS リージョンを選択する

で使用可能なリージョンのリストを表示するには AWS TNB、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用のエンドポイントのリストを表示するには、「」のAWS TNB「エンドポイント」を参照してくださいAWS 全般のリファレンス。

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	ap-northeast-2	tnb.ap-northeast-2.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	
ク (ソウル)				
アジアパシフィック (シドニー)	ap-southeast-2	tnb.ap-southeast-2.amazonaws.com	HTTPS	
カナダ (中部)	ca-central-1	tnb.ca-central-1.amazonaws.com	HTTPS	
欧州 (フランクフルト)	eu-central-1	tnb.eu-central-1.amazonaws.com	HTTPS	
欧州 (パリ)	eu-west-3	tnb.eu-west-3.amazonaws.com	HTTPS	
欧州 (スペイン)	eu-south-2	tnb.eu-south-2.amazonaws.com	HTTPS	
欧州 (ストックホルム)	eu-north-1	tnb.eu-north-1.amazonaws.com	HTTPS	
南米 (サンパウロ)	sa-east-1	tnb.sa-east-1.amazonaws.com	HTTPS	

(オプション) をインストールする AWS CLI

AWS Command Line Interface （ AWS CLI) は、幅広い AWS 製品セットのコマンドを提供し、Windows、macOS 、および Linux でサポートされています。を使用して にアクセスできます AWS TNB 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。のコマンドの詳細については AWS TNB、 AWS CLI コマンドリファレンスの「tnb」を参照してください。

AWS TNB ロールのセットアップ

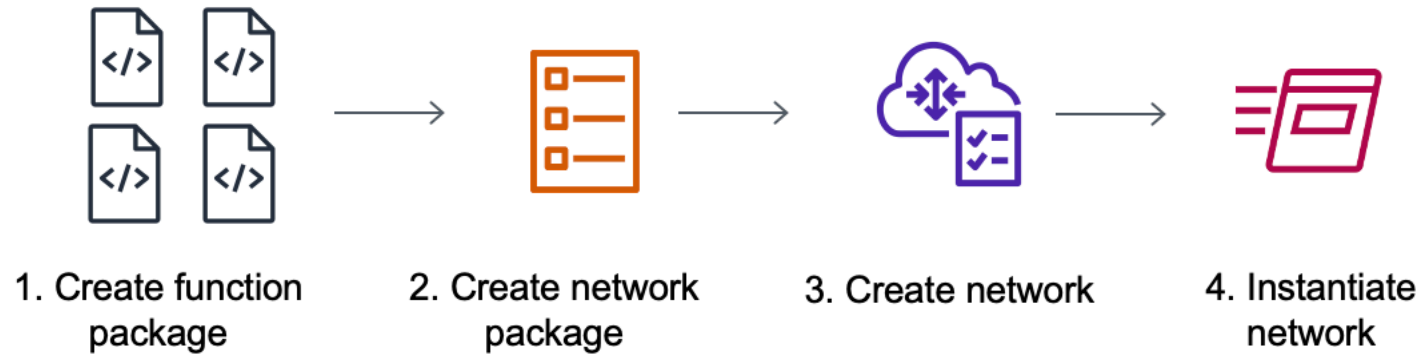
ソリューションの AWS TNBさまざまな部分を管理するには、IAMサービスロールを作成する必要があります。 AWS TNB サービスロールは、 AWS CloudFormationや AWS CodeBuild、ユーザーに代わってさまざまなコンピューティングサービスやストレージサービスなどの他の AWS サービスに対してAPI呼び出しを行い、デプロイ用のリソースをインスタンス化および管理できます。

サービスロールの詳細については、 AWS TNB「」を参照してくださいのアイデンティティとアクセスの管理 AWS TNB。

の開始方法 AWS TNB

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

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、以下が必要です。

- 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。
- SOL001/00SOL4 に準拠したネットワーク関数 (NF) パッケージ。 ETSI
- SOL007 ETSI に準拠した Network Service Descriptor (NSD) テンプレート。

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 AWS TNB GitHub を使用できます。

関数パッケージを作成する

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

関数パッケージを作成するには

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

ネットワークパッケージを作成する

ネットワークパッケージは、デプロイするネットワーク関数と、カタログへのデプロイ方法を指定します。

ネットワークパッケージを作成するには

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

6. [ネットワークパッケージの作成] を選択します。

ネットワークインスタンスを作成してインスタンス化する

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

ネットワークインスタンスを作成してインスタンス化するには

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

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

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

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

クリーンアップ

このチュートリアル用に作成したリソースを削除できるようになりました。

リソースをクリーンアップするには

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

の関数パッケージ AWS TNB

関数パッケージは、ネットワーク関数 CSAR (ETSI標準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関数をネットワーク上で実行する方法を記述する関数パッケージ記述子を含む (Cloud Service Archive) 形式の .zip ファイルです。

タスク

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

で関数パッケージを作成する AWS TNB

ネットワーク関数カタログで AWS TNB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、 でネットワークを作成する最初のステップです AWS TNB。関数パッケージをアップロードしたら、ネットワークパッケージを作成できます。

Console

コンソールを使用して関数パッケージを作成するには

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

AWS CLI

を使用して関数パッケージを作成するには AWS CLI

1. create-sol-function-package コマンドを使用して、新しい関数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
```

で関数パッケージを表示する AWS TNB

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

AWS CLI

を使用して関数パッケージを表示するには AWS CLI

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

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

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

```
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
```

から関数パッケージをダウンロードする AWS TNB

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

AWS CLI

を使用して関数パッケージをダウンロードするには AWS CLI

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

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

から関数パッケージを削除する AWS TNB

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

AWS CLI

を使用して関数パッケージを削除するには AWS CLI

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

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

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

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

のネットワークパッケージ AWS TNB

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

タスク

- 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 AWS TNB

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、ニーズに固有のスクリプトなどの追加のファイル (オプション) で構成されます。例えば、ネットワークパッケージに複数の関数パッケージがある場合は、 NSD を使用して、特定の VPCs、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関数を定義できます。

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. でコンソールを開きます AWS TNBhttps://console.aws.amazon.com/tnb/。
2. ナビゲーションペインで、[ネットワークパッケージ] を選択します。
3. [ネットワークパッケージの作成] を選択します。
4. ファイルの選択を選択し、それぞれを.zipファイルNSDとしてアップロードします。最大 10 個のファイルをアップロードできます。
5. [Next (次へ)] を選択します。
6. パッケージの詳細を確認します。
7. [ネットワークパッケージの作成] を選択します。

AWS CLI

を使用してネットワークパッケージを作成するには AWS CLI

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

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

2. put-sol-network-package-content コマンドを使用して、ネットワークパッケージコンテンツ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
```

でネットワークパッケージを表示する AWS TNB

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

AWS CLI

を使用してネットワークパッケージを表示するには AWS CLI

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

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

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

```
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
```

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

AWS CLI

を使用してネットワークパッケージをダウンロードするには AWS CLI

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

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

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

AWS CLI

を使用してネットワークパッケージを削除するには AWS CLI

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

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

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

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

のネットワークインスタンス AWS TNB

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

タスク

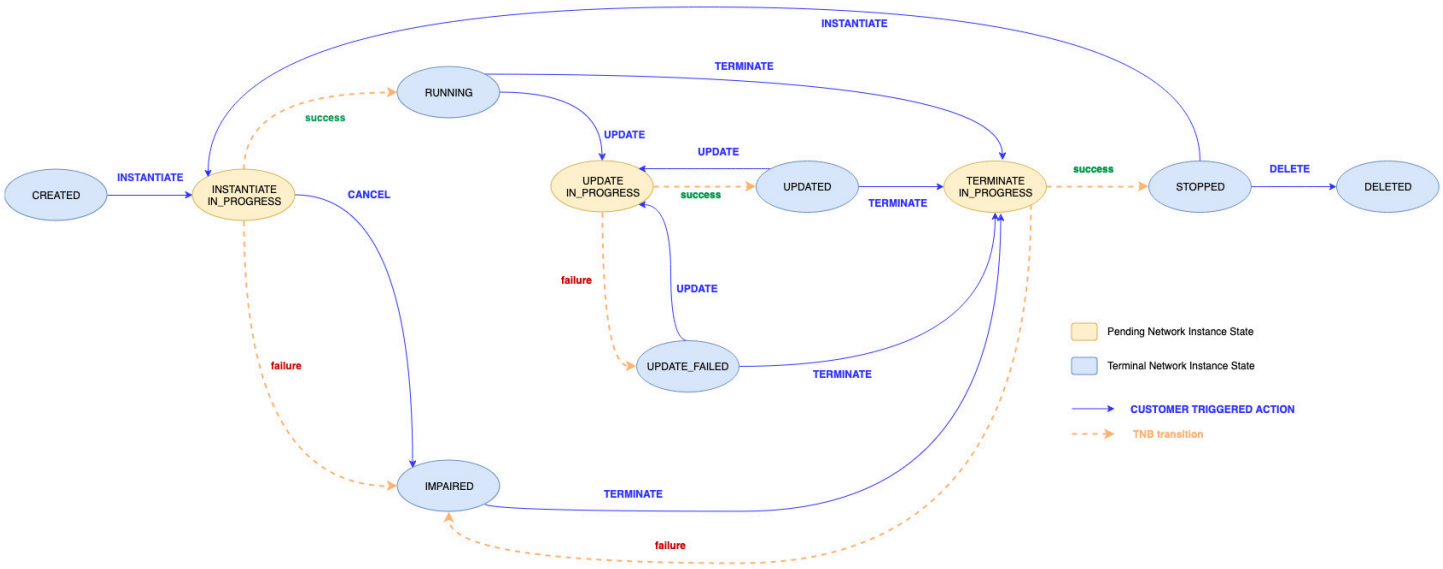
- ネットワークインスタンスのライフサイクルオペレーション
- を使用してネットワークインスタンスを作成する AWS TNB
- を使用してネットワークインスタンスをインスタンス化する AWS TNB
- で関数インスタンスを更新する AWS TNB
- でネットワークインスタンスを更新する AWS TNB
- でネットワークインスタンスを表示する AWS TNB
- からネットワークインスタンスを終了および削除する AWS TNB

ネットワークインスタンスのライフサイクルオペレーション

AWS TNB では、003 および ETSI SOL00SOL5 とインラインで標準化された管理オペレーションを使用して、ネットワークを簡単に管理できます。次のライフサイクルオペレーションを実行できます。

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

次の図は、ネットワーク管理オペレーションを示しています。



を使用してネットワークインスタンスを作成する AWS TNB

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

Console

コンソールを使用してネットワークインスタンスを作成するには

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

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

AWS CLI

を使用してネットワークインスタンスを作成するには AWS CLI

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

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

次に、このネットワークインスタンスをインスタンス化します。

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

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

Console

コンソールを使用してネットワークインスタンスをインスタンス化するには

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

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

   デプロイステータスページが表示されます。

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

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

AWS CLI

を使用してネットワークインスタンスをインスタンス化するには AWS CLI

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

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

2. 次に、ネットワークオペレーションのステータスを表示します。

で関数インスタンスを更新する AWS TNB

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

Console

コンソールを使用して関数インスタンスを更新するには

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

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

AWS CLI

を使用して関数インスタンスCLIを更新する

MODIFY_VNF_INFORMATION 更新タイプで 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の関数インスタンスを更新します。

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

でネットワークインスタンスを更新する AWS TNB

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

考慮事項

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

更新できるパラメータ

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

パラメータ	説明	例: 前	例: After
Amazon EKSクラスターバージョン	Amazon EKSクラスターコントロールプレーンversionパラメータの値を次のマイナーバージョンに更新できます。	EKSCluster: type: tosca.nodes.AWS.Compute.EKS properties:	EKSCr:

パラメータ	説明	例: 前	例: After
	バージョンをダウングレードすることはできません。ワーカーノードは更新されません。	`version: "1.28"`	typ tos es.A mput pro s: ver "1.

パラメータ	説明	例: 前	例: After

スケーリングプロパティ	EKSManagedNode および EKSSelfManagedNode TOSCAノードのスケーリングプロパティを更新できます。	`EKSNodeGroup01:`
		`  ...`
		`  scaling:`
		`  properties:`
		`  desired_size: 1`
		`  min_size: 1`
		`  max_size: 1`
			`EKSNodeGroup01:`
			`  ...`
			`  scaling:`
			`  properties:`
			`  desired_size: 2`
			`  min_size: 1`
			`  max_size: 1`

パラメータ	説明	例: 前	例: After
			mir max

パラメータ	説明	例: 前	例: After

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

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

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

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

パラメータ	説明	例: 前	例: After

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

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

パラメータ	説明	例: 前	例: After

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

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

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

ネットワークインスタンスの更新

Console

コンソールを使用してネットワークインスタンスを更新するには

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

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

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

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

   デプロイステータスが に変わると`Completed`、ネットワークインスタンスが更新されます。
9. - 検証が失敗した場合、ネットワークインスタンスは、更新をリクエストする前と同じ状態のままになります。 `Instantiated`または ですUpdated。
   - 更新に失敗すると、ネットワークインスタンスの状態は を表示しますUpdate failed。失敗した各タスクのリンクを選択して、理由を決定します。
   - 更新が成功すると、ネットワークインスタンスの状態は を表示しますUpdated。

AWS CLI

を使用してネットワークインスタンスCLIを更新する

UPDATE_NS 更新タイプで 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 UPDATE_NS --update-ns "{\"nsdInfoId\":\"^np-[a-f0-9]{17}$\",
 \"additionalParamsForNs\": {\"param1\": \"value1\"}}"
```

でネットワークインスタンスを表示する AWS TNB

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

AWS CLI

を使用してネットワークインスタンスを表示するには AWS CLI

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

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

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

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

からネットワークインスタンスを終了および削除する AWS TNB

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

AWS CLI

を使用してネットワークインスタンスを終了および削除するには AWS CLI

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

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

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

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

のネットワークオペレーション AWS TNB

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

ネットワークオペレーションを表示する AWS TNB

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. でコンソールを開きます AWS TNBhttps://console.aws.amazon.com/tnb/。
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. Deployments タブで、ネットワークオペレーションを選択します。

AWS CLI

を使用してネットワークオペレーションを表示するには AWS CLI

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

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

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

```
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 AWS TNB

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

AWS CLI

を使用してネットワークオペレーションをキャンセルするには AWS CLI

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

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

TOSCA のリファレンス AWS TNB

クラウドアプリケーションのトポロジーとオーケストレーション仕様 (TOSCA) は、 CSPsを使用して、クラウドベースのウェブサービス、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ーを記述する宣言構文です。CSPs は、接続ポイント、接続ポイント間の論理リンク、およびTOSCAテンプレート内のアフィニティやセキュリティなどのポリシーを記述します。CSPs 次に、ア AWS ベイラビリティーゾーン間で機能する 5G ネットワークを確立するために必要なリソースを合成する テンプレート AWS TNBをアップロードします。

内容

VNFD テンプレート

仮想ネットワーク関数記述子 (VNFD) テンプレートを定義します。

構文

```
tosca_definitions_version: tnb_simple_yaml_1_0

topology_template:

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

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

トポロジテンプレート

`node_templates`

TOSCA AWS ノード。使用できるノードは次のとおりです。

- AWS.VNF
- AWS.Artifacts.Helm

AWS.VNF

AWS 仮想ネットワーク関数 (VNF) ノードを定義します。

構文

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

プロパティ

`descriptor_id`

記述子UUIDの 。

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`

のバージョンVNFD。

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`

の作成者VNFD。

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"
```

ネットワークサービス記述子テンプレート

ネットワークサービス記述子 (NSD) テンプレートを定義します。

構文

```
tosca_definitions_version: tnb_simple_yaml_1_0

vnfds:
  - descriptor_id: String
    namespace: String

topology_template:

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

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

定義済みのパラメータを使用する

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

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UUIDの 。

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.コンピューティング。EKS
- AWS.コンピューティングEKS..AuthRole
- AWS.コンピューティング。EKSManagedNode
- AWS.コンピューティング。EKSSelfManagedNode
- AWS.コンピューティング。PlacementGroup
- AWS.コンピューティング。UserData
- AWS.ネットワーキング。SecurityGroup
- AWS.ネットワーキング。SecurityGroupEgressRule
- AWS.ネットワーキング。SecurityGroupIngressRule
- AWS.Resource.Import
- AWS.ネットワーキング。ENI
- AWS.HookExecution
- AWS.ネットワーキング。InternetGateway
- AWS.ネットワーキング。RouteTable
- AWS.Networking.Subnet
- AWS.デプロイ。VNFDeployment

- AWS.ネットワーキング。VPC
- AWS.ネットワーキング。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UUIDの 。

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`

のバージョンNSD。

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.コンピューティング。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 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NFs。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) の使用を定義するプロパティ。

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

型: ブール値

multus_role

Multus ネットワークインターフェイス管理のロール。

必須: はい

型: 文字列

ebs_csi

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

このプラグインを有効にして、、 AWS ローカルゾーン AWS Outposts、または で Amazon EKSセルフマネージドノードを使用します AWS リージョン。詳細については、「Amazon 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。 EKS

enabled

デフォルトの Amazon EBSCSI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: ブール値

`version`

Amazon EBSCSI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 EKSAPIリファレンスDescribeAddonVersions」の「」を参照してください。

必須: いいえ

型: 文字列

プロパティ

`version`

クラスターの Kubernetes バージョン。 AWS Telco Network Builder は、Kubernetes バージョン 1.23 から 1.30 をサポートしています。

必須: はい

型: 文字列

可能な値: 1.23 | 1.24 | 1.25 | 1.26 | 1.27 | 1.28 | 1.29 | 1.30

`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

型: 文字列

要件

subnets

AWS.Networking.Subnet ノード。

必須 : はい

タイプ: リスト

例

```
SampleEKS:
  type: tosca.nodes.AWS.Compute.EKS
  properties:
    version: "1.23"
    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.コンピューティングEKS..AuthRole

AuthRole を使用すると、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 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aws-authConfigMap。

`arn`

IAM ロールARNの 。

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.コンピューティング。EKSManagedNode

AWS TNB は、Amazon 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 Managed Node EKS グループをサポートしています。EKS Node グループを作成するには、以下を実行します。

- AMI または AMIタイプの 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) を選択します。
- SSH アクセス用の Amazon EC2キーペアとノードグループのスケーリングプロパティを指定します。
- ノードグループが Amazon EKSクラスターに関連付けられていることを確認します。

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

構文

```
tosca.nodes.AWS.Compute.EKSManagedNode:
  capabilities:
    compute:
      properties:
        ami_type: String
        ami_id: String
        instance_types: List
        key_pair: String
        root_volume_encryption: Boolean
        root_volume_encryption_key_arn: String
    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 インスタンスタイプや Amazon EC2インスタンス など、Amazon EC2 EKSマネージド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AMIs。

`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`

の IDAMI。

必須: いいえ

型: 文字列

> **Note**
>
> テンプレートで `ami_type`と の両方`ami_id`が指定されている場合、 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 キーARNの 。 AWS TNB は、通常のキー ARN、マルチリージョンキー ARN、およびエイリアス をサポートしますARN。

必須: いいえ

型: 文字列

> **Note**
>
> - `root_volume_encryption` が false の場合、 を含めないでくださいいroot_volume_encryption_key_arn。
> - AWS TNB は、Amazon EBSでバックアップされた AMIのルートボリューム暗号化をサポートします。
> - AMIのルートボリュームが既に暗号化されている場合は、ルートボリュームを再暗号化するには、 `root_volume_encryption_key_arn` の AWS TNB を含める必要があります。
> - AMIのルートボリュームが暗号化されていない場合、 AWS TNBは `root_volume_encryption_key_arn` を使用してルートボリュームを暗号化します。
>
>   を含めない場合root_volume_encryption_key_arn、 AWS TNBは によって提供されたデフォルトキー 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 を復号しませんAMI。

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を定義するプロパティ。例えば、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です。

`desired_size`

この のインスタンス数 NodeGroup。

必須: はい

型: 整数

`min_size`

この の最小インスタンス数 NodeGroup。

必須: はい

型: 整数

`max_size`

この 内のインスタンスの最大数 NodeGroup。

必須: はい

型: 整数

プロパティ

`node_role`

Amazon EC2インスタンスにアタッチされているIAMロールARNの 。

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

必須：いいえ

タイプ: リスト

`security_groups`

AWS.Networking.SecurityGroup ノード。

必須：いいえ

タイプ: リスト

`placement_group`

tosca.nodes AWS。Compute.PlacementGroup ノード。

必須: いいえ

型: 文字列

`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

型: 文字列

`labels`

ノードラベルのリスト。ノードラベルには名前と値が必要です。次の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 * 文字で開始および終了する必要があります。

例えば、`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.コンピューティング。EKSSelfManagedNode

AWS TNB は、Amazon 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 Amazon EKS 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、以下を実行します。

- 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) を選択します AMI。
- SSH アクセス用の Amazon EC2キーペアを指定します。
- ノードグループが Amazon EKSクラスターに関連付けられていることを確認します。
- インスタンスタイプと、必要なサイズ、最小サイズ、最大サイズを指定します。
- ワーカーノードのサブネットを指定します。

- 必要に応じて、セキュリティグループ、ノードラベル、およびプレイスメントグループをノードグループにアタッチします。

構文

```
tosca.nodes.AWS.Compute.EKSSelfManagedNode:
  capabilities:
    compute:
      properties:
        ami_id: String
        instance_type: String
        key_pair: String
        root_volume_encryption: Boolean
        root_volume_encryption_key_arn: String
    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 インスタンスタイプや Amazon EC2 EC2インスタンス など、Amazon EKSセルフマネージド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AMIs。

`ami_id`

インスタンスの起動に使用される AMI ID。 AWS TNB は、 を活用するインスタンスをサポートしますIMDSv2。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 キーARNの 。 AWS TNB は、通常のキー ARN、マルチリージョンキー ARN、およびエイリアス をサポートしますARN。

必須: いいえ

型: 文字列

> **Note**
>
> - `root_volume_encryption` が false の場合、 を含めないでください`root_volume_encryption_key_arn`。

- AWS TNB は、Amazon EBSでバックアップされた AMIのルートボリューム暗号化をサポートします。
- AMIのルートボリュームが既に暗号化されている場合は、ルートボリュームを再暗号化するには、 `root_volume_encryption_key_arn` の AWS TNB を含める必要があります。
- AMIのルートボリュームが暗号化されていない場合、 AWS TNB は `root_volume_encryption_key_arn` を使用してルートボリュームを暗号化します。

  を含めない場合`root_volume_encryption_key_arn`、 AWS TNB は AWS Managed Services を使用してルートボリュームを暗号化します。
- AWS TNB は暗号化された を復号しませんAMI。

scaling

Amazon インスタンスの希望数、ノードグループEC2内の Amazon EC2 インスタンスの最小数と最大数など、Amazon EKSセルフマネージドノードのスケーリングパラメータ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

型: 整数

`min_size`

この のインスタンスの最小数 NodeGroup。

必須: はい

型: 整数

`max_size`

この 内のインスタンスの最大数 NodeGroup。

必須: はい

型: 整数

プロパティ

`node_role`

Amazon EC2インスタンスにアタッチされているIAMロールARNの 。

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 TNB.AWSCompute.EKS ノードに `multus_role` プロパティを含めると、 は `multus`プロパティENIsから に関連するアクセス許可を取得しま

す。それ以外の場合、 は node_role プロパティENIsから に関連するアクセス許可 AWS TNBを取得します。

必須：いいえ

タイプ: リスト

`security_groups`

AWS.Networking.SecurityGroup ノード。

必須：いいえ

タイプ: リスト

`placement_group`

tosca.nodes AWS。Compute.PlacementGroup ノード。

必須: いいえ

型: 文字列

`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

型: 文字列

`labels`

ノードラベルのリスト。ノードラベルには名前と値が必要です。次の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 * 文字で開始および終了する必要があります。

例えば、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ています。

新しい Amazon を起動するとEC2instance、Amazon EC2サービスは、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して、相関する障害を最小限に抑えるようにインスタンスを配置しようとします。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

使用可能な値: CLUSTER | PARTITION | SPREAD_HOST | SPREAD_RACK

- CLUSTER – は、アベイラビリティーゾーン内でインスタンスを密接にまとめます。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 – は、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異なる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、インスタンスを論理パーティションに分散します。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 – インスタンスの小さなグループを個別の基盤となるハードウェアに配置し、相関する障害を減らします。
- SPREAD_HOST – Outpost プレイスメントグループでのみ使用されます。相関性の工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。

型: 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 () の UserDataノードを介したカスタムユーザーデータを使用した Amazon EC2インスタンスの起動をサポートしますNSD。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クラスターに Amazon EC2インスタンス登録を提供します。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 に渡しますEC2。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 のユーザーデータスクリプトをサポートします。
> - AWS CloudFormation を使用してmultimimeユーザーデータスクリプトを処理しレンダリングするため AWS TNB、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.ネットワーク。SecurityGroup

AWS TNB は、Amazon Kubernetes クラスターノードグループにアタッチできる Amazon EC2 Security Groups EKS 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.ネットワーキング。SecurityGroupEgressRule

AWS TNB は、. AWS Networking にアタッチできる Amazon EC2 Security Group Egress Rules のプロビジョニングを自動化するためのセキュリティグループ出力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。出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

型: 文字列

`cidr_ipv6`

出力トラフィックの CIDR 形式のIPv6アドレス範囲。送信先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) を指定する必要がありますcidr_ipv6。

必須: いいえ

型: 文字列

`description`

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

必須: いいえ

型: 文字列

`destination_prefix_list`

既存の Amazon VPCマネージドプレフィックスリストのプレフィックスリスト ID。これは、セキュリティグループに関連付けられたノードグループインスタンスからの送信先です。マネージドプレフィックスリストの詳細については、「Amazon 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。 VPC

必須: いいえ

型: 文字列

`from_port`

プロトコルが TCPまたは の場合UDP、これはポート範囲の開始です。プロトコルが ICMPまたは の場合ICMPv6、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 タイプを指定する場合は、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 の場合UDP、これはポート範囲の終わりです。プロトコルが ICMPまたは の場合ICMPv6、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 タイプを指定する場合は、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`

送信トラフィックが許可される送信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

型: 文字列

例

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

AWS.ネットワーキング。SecurityGroupIngressRule

AWS TNB は、. AWS Networking にアタッチできる Amazon EC2 Security Group Ingress Rules 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。進入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

型: 文字列

`cidr_ipv6`

イングレストラフィックCIDRの形式の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) を指定する必要がありますcidr_ipv6。

必須: いいえ

型: 文字列

`description`

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

必須: いいえ

型: 文字列

`source_prefix_list`

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

必須: いいえ

型: 文字列

`from_port`

プロトコルが TCPまたは の場合UDP、これはポート範囲の開始です。プロトコルが ICMPまたは の場合ICMPv6、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 タイプを指定する場合は、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 の場合UDP、これはポート範囲の終わりです。プロトコルが ICMPまたは の場合ICMPv6、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 タイプを指定する場合は、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`

進入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

型: 文字列

例

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

AWS.Resource.Import

次の AWS リソースを にインポートできます AWS TNB。

- 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`

にインポートされるリソースタイプ AWS TNB。

必須：いいえ

タイプ: リスト

`resource_id`

にインポートされるリソース ID AWS TNB。

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.ネットワーキング。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VPCを表す 内の論理ネットワークコンポーネントです。ネットワークインターフェイスには、サブネットに基づいて自動または手動で IP

アドレスが割り当てられます。サブネットに Amazon EC2インスタンスをデプロイしたら、そのインスタンスにネットワークインターフェイスをアタッチするか、その Amazon EC2インスタンスからネットワークインターフェイスをデタッチして、そのサブネット内の別の Amazon EC2 インスタンスに再アタッチできます。デバイスインデックスは、アタッチの順番における位置を識別します。

構文

```
tosca.nodes.AWS.Networking.ENI:
  properties:
    device_index: Integer
    source_dest_check: Boolean
    tags: List
  requirements:
    subnet: String
    security_groups: List
```

プロパティ

`device_index`

デバイスインデックスはゼロより大きくする必要があります。

必須: はい

型: 整数

`source_dest_check`

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

許可される値: true、false

デフォルト: true

必須 : いいえ

型: ブール値

`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

型: 文字列

例

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

AWS.HookExecution

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

構文

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

機能

execution

フックスクリプトを実行するフック実行エンジンのプロパティ。

type

フック実行エンジンのタイプ。

必須: いいえ

型: 文字列

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.ネットワーキング。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`

内のルーティング接続を定義するプロパティVPC。`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`

送信先一致に使用されるIPv4CIDR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

送信先一致に使用されるIPv6CIDR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: ブール値

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.ネットワーキング。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がどこに送られるかを決定する、ルートと呼ばれる一連のルールが含まれています。ルートテーブルをに関連付ける必要がありますVPC。

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 内の IP アドレスの範囲でありVPC、1 つのアベイラビリティーゾーン内に完全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、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アドレスを受信す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、 AWS リージョン内の AWS アベイラビリティーゾーン (`us-west-2`米国西部 (オレゴン) など) をサポートします。また、 など、アベイラビリティーゾーン内の AWS ローカルゾーンもサポートしていますus-west-2-lax-1a。

必須: はい

型: 文字列

`cidr_block`

サブネットのCIDRブロック。

必須: いいえ

型: 文字列

`ipv6_cidr_block`

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

必須: いいえ

型: 文字列

`ipv6_cidr_block_suffix`

Amazon 経由で作成されたサブネットのIPv6CIDRブロックの 2 桁の 16 進サフィックスVPC。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

型: 文字列

`outpost_arn`

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

このプロパティを NSD テンプレートに追加する場合は、`availability_zone`プロパティの値を 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 AWS Outposts。

必須: いいえ

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

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.デプロイ。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。クラスター属性は、 をホストするEKSクラスターを指定しますNFs。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、pre_create および post_create タイプのオプションのライフサイクルフックオペレーションを指定して、インベントリマネジメントシステム を呼び出すなど、デプロイに固有の手順を実行することもできますAPI。

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

型: 文字列

`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

型: 文字列

`post_create`

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

必須: いいえ

型: 文字列

例

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

AWS.ネットワーキング。VPC

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

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`

のIPv4ネットワーク範囲をVPC表記でCIDR示します。

必須: はい

型: 文字列

`ipv6_cidr_block`

の作成に使用されるIPv6CIDRブロックVPC。

許可される値: `AMAZON_PROVIDED`

必須: いいえ

型: 文字列

`dns_support`

VPC get DNS hostnames でインスタンスが起動されたかどうかを示します。

必須：いいえ

型: ブール値

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.ネットワーキング。NATGateway

サブネット上で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

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

例

```
Free5GCNatGateway01:
  type: tosca.nodes.AWS.Networking.NATGateway
```

```
    requirements:
       subnet: Free5GCSubnet01
       internet_gateway: Free5GCIGW
    properties:
       type: PUBLIC
       eip_allocation_id: eipalloc-12345
```

AWS.Networking.Route

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

構文

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

プロパティ

`dest_cidr_blocks`

ターゲットリソースへの送信先IPv4ルートのリスト。

必須：はい

タイプ: リスト

メンバー型: 文字列

プロパティ

`nat_gateway`

AWS.Networking.NATGateway ノードリファレンス。

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 および のノードを定義しますVNFD。

- AWS.HookDefinition.バッシュ

AWS.HookDefinition.Bash

AWS HookDefinition で を定義しますbash。

構文

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

プロパティ

`implementation`

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

必須: はい

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。
- 以下の AWS TNB予約キーワード (大文字と小文字は区別されません) を使用して環境変数名を開始しないでください。
  - 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"
```

のセキュリティ AWS TNB

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 、最もセキュリティに敏感な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 とユーザーの間で責任を共有します。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS 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があります AWS クラウド。 は、安全に使用できるサービス AWS も提供します。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラム の一環として、当社のセキュリティの有効性を定期的にテストおよび検証。 AWS Telco Network Builder に適用されるコンプライアンスプログラムの詳細については、AWS 「コンプライアンスプログラムによる対象範囲内のサービスコンプライアンス」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、 を使用する際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 AWS TNB。次のトピックでは、セキュリティとコンプライアンスの目的を満たすように を設定する AWS TNB方法を示します。また、 リソースのモニタリングと保護に役立つ他の AWS サービスの使用方法についても AWS TNB説明します。

内容

でのデータ保護 AWS TNB

責任 AWS 共有モデル 、 AWS Telco Network Builder 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FAQ」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 「 責任共有モデル」とGDPRAWS 「セキュリティブログ」のブログ記事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 () 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IAM。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。1.2 が必要でTLS、1.3 TLS をお勧めします。
- で APIとユーザーアクティビティのログ記録を設定します AWS CloudTrail。証 CloudTrail 跡を使用して AWS アクティビティをキャプチャする方法については、AWS CloudTrail 「 ユーザーガイド」の CloudTrail 「証跡の操作」を参照してください。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な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 AWS を介して にアクセスするときに FIPS 140-3 検証済みの暗号化モジュールが必要な場合はAPI、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、「連邦情報処理標準 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、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。これには、コンソール、、またはを使用して AWS のサービス API AWS CLIまたは他の を操作する AWS TNB場合も含まれます 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。URL を外部サーバーに提供する場合は、そのサーバーへのリクエストを検証URLするために認証情報を に含めないことを強くお勧めします。

データの処理

AWS アカウントを閉じると、 AWS TNB はデータを削除対象としてマークし、そのデータを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 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仮想プライベートクラウド (VPC) に存在します。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。

のアイデンティティとアクセスの管理 AWS TNB

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、誰にリソースの使用 AWS TNBを許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

内容

- AWS Telco Network Builder のアイデンティティベースポリシーの例
- AWS Telco Network Builder の ID とアクセスのトラブルシューティング

対象者

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

サービスユーザー – サービスを使用して AWS TNBジョブを実行する場合、管理者は必要な認証情報とアクセス許可を提供します。より多くの AWS TNB機能を使用して作業を行う際には、追加のアクセス許可が必要になる場合があります。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。の機能にアクセスできない場合は AWS TNB、「」を参照してくださいAWS Telco Network Builder の ID とアクセスのトラブルシューティング。

サービス管理者 – 社内のリソースを担当 AWS TNBしている場合は、 へのフルアクセスがある可能性があります AWS TNB。サービスユーザーがどの機能やリソースにアクセスする必要があるかAWS TNBを判断するのはお客様の仕事です。その後、サービスユーザーのアクセス許可を変更するリクエストをIAM管理者に送信する必要があります。このページの情報を確認して、 の基本概念を理解しますIAM。会社IAMで を使用する方法の詳細については AWS TNB、「」を参照してくださいと の仕組み AWS TNB IAM。

IAM 管理者 - IAM管理者の場合は、 へのアクセスを管理するためのポリシーの作成方法の詳細を知りたい場合があります AWS TNB。で使用できるアイデンティティベースのポリシーの例 AWS TNBを表示するにはIAM、「」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

認証は、アイデンティティ認証情報 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) する必要があります。

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 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 ID としてサインインすると、管理者は以前に IAMロールを使用して ID フェデレーションをセットアップしていました。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、 AWS サインイン ユーザーガイドの「 へのサインイン方法 AWS アカウント」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、 はソフトウェア開発キット (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化して署名します。 AWS ツールを使用しない場合は、自分でリクエストに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に署名する方法の詳細については、 IAM ユーザーガイドのAWS API「リクエストの署名バージョン 4」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、 では、アカウントのセキュリティを高めるために多要素認証 (MFA) を使用する AWS 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「多要素認証」とIAM「 ユーザーガイド」のAWS 「多要素認証IAM」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 ID 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してアクセスします。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「ルートユーザーの認証情報を必要とするタスク」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、 では、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アクセスするために ID プロバイダーとのフェデレーション AWS のサービス の使用を要求します。

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、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 を使用してアクセスするすべてのユーザーのユーザーです。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、それらはロールを引き受け、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したり、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 ユーザーとは、1 人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 ID 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持つIAMユーザーを作成する代わりに、一時的な認証情報に依存することをお勧めします。ただし、IAMユーザーとの長期的な認証情報を必要とする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ユーザーのコレクションを指定する ID 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、という名前のグループがありIAMAdmins、そのグループにIAMリソースを管理するアクセス許可を付与できます。

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、ロールでは一時認証情報が提供されます。詳細については、「 ユーザーガイド」のIAM「ユーザーのユースケース」を参照してください。 IAM

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 ID です。ユーザーと似ていますがIAM、特定の人物には関連付けられていません。でIAMロールを一時的に引き受けるには AWS Management Console、ユーザーからIAMロール (コンソール) に切り替えることができます。またはオペレーションを AWS CLI AWS API呼び出すか、カスタム を使用してロールを引き受けることができますURL。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「ロールを引き受ける方法」を参照してください。

IAM 一時的な認証情報を持つ ロールは、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 ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールの詳細については、 IAM ユーザーガイドの「サードパーティー ID プロバイダーのロールの作

成」を参照してください。IAM Identity Center を使用する場合は、アクセス許可セットを設定します。ID が認証された後にアクセスできる内容を制御するために、IAMIdentity Center はアクセス許可セットを のロールに関連付けますIAM。アクセス許可セットの詳細については、「AWS IAM Identity Center ユーザーガイド」の「アクセス許可セット」を参照してください。

- 一時的なIAMユーザーアクセス許可 – IAM ユーザーまたはロールは、特定のタスクに対して異なるアクセス許可を一時的に引き受けるIAMロールを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、別のアカウントの誰か (信頼できるプリンシパル) が自分のアカウントのリソースにアクセスすることを許可できます。クロスアカウントアクセスを許可する主な方法は、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、 (プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。クロスアカウントアクセスの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「 のクロスアカウントリソースアクセスIAM」を参照してください。
- クロスサービスアクセス — 他の の機能 AWS のサービス を使用するものもあります AWS のサービス。例えば、 サービスで呼び出しを行う場合、そのサービスが Amazon でアプリケーションを実行EC2したりAmazon S3にオブジェクトを保存したりするのが一般的です。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンク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権限が必要です。FAS リクエストを行う際のポリシーの詳細については、「アクセスセッションの転送」を参照してください。
  - サービスロール – サービスロールは、ユーザーに代わってアクションを実行するためにサービスが引き受けるIAMロールです。IAM 管理者は、 内からサービスロールを作成、変更、削除できますIAM。詳細については、IAM「 ユーザーガイド」の「 にアクセス許可を委任するロールの作成 AWS のサービス」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アクセス許可を表示することはできますが、編集することはできません。

- Amazon で実行されているアプリケーション EC2 – 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インスタンス内にアクセスキーを保存するよりも望ましいです。EC2インスタンスに AWS ロールを割り当て、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 ロールが含まれており、EC2インスタンスで実行されているプログラムが一時的な認証情報を取得できるようにします。詳細については、IAM「 ユーザーガイド」のIAM「ロールを使用して Amazon EC2インスタンスで実行されているアプリケーションにアクセス許可を付与する」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には AWS 、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 AWS 、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 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) 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、「 ユーザーガイド」のJSON「ポリシーの概要」を参照してください。 IAM

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

デフォルトでは、ユーザーやロールに権限はありません。必要なリソースに対してアクションを実行するアクセス許可をユーザーに付与するには、IAM管理者はIAMポリシーを作成できます。その後、管理者はIAMポリシーをロールに追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行に使用する方法に関係なく、アクションのアクセス許可を定義します。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 からロール情報を取得できますAPI。

アイデンティティベースのポリシー

ID ベースのポリシーは、IAMユーザー、ユーザーのグループ、ロールなどの ID にアタッチできるJSONアクセス許可ポリシードキュメントです。これらのポリシーは、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。ID 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「カスタマーマネージドポリシーを使用したカスタムIAMアクセス許可の定義」を参照してください。

アイデンティティベースのポリシーは、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。マネージドポリシーは、 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選択」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、リソースにアタッチする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の例としては、IAMロール信頼ポリシー と Amazon S3 バケットポリシー があります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、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IAMでは、 から AWS 管理ポリシーを使用することはできません。

アクセスコントロールリスト (ACLs)

アクセスコントロールリスト (ACLs) 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アクセス許可を持っているかを制御します。ACLs 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されません。

Amazon S3、および Amazon VPCは AWS WAF、 をサポートするサービスの例ですACLs。の詳細についてはACLs、「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) 概要」を参照してください。

その他のポリシータイプ

AWS は、追加の低頻度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がIAMエンティティ (IAMユーザーまたはロール) に付与できる最大アクセス許可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。アクセス許可の境界の詳細については、IAM「 ユーザーガイド」のIAM「エンティティのアクセス許可の境界」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) の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、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ためのサービス AWS アカウント 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 (SCPs) をアカウントのいずれかまたはすべてに適用できます。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SCPを制限します AWS アカウントのルートユーザー。Organizations と の詳細については SCPs、AWS Organizations 「 ユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- 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、「 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。 IAM

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。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、IAM「 ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

と の仕組み AWS TNB IAM

IAM を使用して へのアクセスを管理する前に AWS TNB、 で使用できるIAM機能について説明します AWS TNB。

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

IAM 機能	AWS TNB サポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	なし
ポリシーアクション	あり
ポリシーリソース	Yes
ポリシー条件キー	可能
ACLs	不可
ABAC (ポリシーのタグ)	可能
一時的な認証情報	あり
プリンシパル権限	あり
サービスロール	いいえ
サービスリンクロール	不可

および他の AWS のサービスがほとんどのIAM機能とどのように連携するか AWS TNBの概要については、 IAM ユーザーガイドのAWS 「 と連携する のサービスIAM」を参照してください。

のアイデンティティベースのポリシー AWS TNB

アイデンティティベースのポリシーのサポート: あり

ID ベースのポリシーは、IAMユーザー、ユーザーのグループ、ロールなどの ID にアタッチできるJSONアクセス許可ポリシードキュメントです。これらのポリシーは、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。ID 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「カスタマー管理ポリシーによるカスタムIAMアクセス許可の定義」を参照してください。

IAM ID ベースのポリシーでは、許可または拒否されたアクションとリソース、およびアクションが許可または拒否され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IAMJSON「ポリシー要素リファレンス」を参照してください。

のアイデンティティベースのポリシーの例 AWS TNB

ID ベースのポリシーの例 AWS TNB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。

内のリソースベースのポリシー AWS TNB

リソースベースのポリシーのサポート: なし

リソースベースのポリシーは、リソースにアタッチする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の例としては、IAMロール信頼ポリシー と Amazon S3 バケットポリシー があります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、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、リソースベースのポリシーのプリンシパルとして、別のアカウントのアカウントまたはIAMエンティティ全体を指定できます。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さらに付与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「 のクロスアカウントリソースアクセスIAM」を参照してください。

のポリシーアクション AWS TNB

ポリシーアクションのサポート: あり

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

JSON ポリシーの Action要素は、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を記述します。ポリシーアクションは通常、関連付けられた AWS APIオペレーションと同じ名前です。一致するAPIオペレーションがないアクセス許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数の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。

アクションのリスト AWS TNB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。

の AWS TNB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*"
```

ID ベースのポリシーの例 AWS TNB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。

のポリシーリソース AWS TNB

ポリシーリソースのサポート: あり

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

Resource JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。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、リソースレベルの許可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": "*"
```

リソースタイプとその のリスト AWS TNBを確認するにはARNs、「サービス認証リファレンス」のAWS 「Telco Network Builder で定義されるリソース」を参照してください。各リソースARNの を指定できるアクションについては、AWS 「Telco Network Builder で定義されるアクション」を参照してください。

ID ベースのポリシーの例 AWS TNB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。

のポリシー条件キー AWS TNB

サービス固有のポリシー条件キーのサポート: あり

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

Condition 要素 (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 要素を指定する場合、または 1 つの Condition 要素に複数のキーを指定する場合、 AWS では 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ユーザー名でタグ付けされている場合にのみ、リソースにアクセスするアクセス許可をIAMユーザーに付与できます。詳細については、「 ユーザーガイド」のIAM「ポリシー要素: 変数とタグ」を参照してください。 IAM

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、 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。 IAM

条件キーのリスト AWS TNB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。

ID ベースのポリシーの例 AWS TNB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。

ACLs の AWS TNB

をサポートACLs： なし

アクセスコントロールリスト (ACLs) 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アクセス許可を持っているかを制御します。ACLs 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ていません。

ABAC と AWS TNB

サポート ABAC (ポリシーのタグ): はい

属性ベースのアクセスコントロール (ABAC) は、属性に基づいてアクセス許可を定義する認証戦略です。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、の最初のステップですABAC。次に、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する場合に、オペレーションを許可するABACポリシーを設計します。

ABAC は、急速に成長している環境で役立ち、ポリシー管理が面倒になる状況に役立ちます。

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

の詳細についてはABAC、「 ユーザーガイド」の「 ABAC認可によるアクセス許可の定義」を参照してください。 IAM を設定する手順を含むチュートリアルを表示するにはABAC、 IAM ユーザーガイドの「属性ベースのアクセスコントロールを使用する (ABAC) 」を参照してください。

での一時的な認証情報の使用 AWS TNB

一時的な認証情報のサポート: あり

一時的な認証情報を使用してサインインすると機能 AWS のサービス しない場合があります。一時的な認証情報 AWS のサービス を使用する方法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 を使用する方法IAM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、会社のシングルサインオン (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。ロールの切り替えの詳細については、 IAM ユーザーガイドの「ユーザーからIAMロールへの切り替え (コンソール)」を参照してください。

AWS CLI または を使用して、一時的な認証情報を手動で作成できます AWS API。その後、これらの一時的な認証情報を使用して にアクセスできます AWS。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「」の「一時的なセキュリティ認証情報IAM」を参照してください。

のクロスサービスプリンシパルアクセス許可 AWS TNB

転送アクセスセッションをサポート (FAS): 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と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストを使用します。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権限が必要です。FAS リクエストを行う際のポリシーの詳細については、「アクセスセッションの転送」を参照してください。

AWS TNB のサービスロール

サービスロールのサポート: なし

サービスロールは、ユーザーに代わってアクションを実行するためにサービスが引き受けるIAMロールです。IAM 管理者は、 内からサービスロールを作成、変更、削除できますIAM。詳細については、IAM「 ユーザーガイド」の「 にアクセス許可を委任するロールの作成 AWS のサービス」を参照してください。

のサービスにリンクされたロール AWS TNB

サービスにリンクされたロールのサポート: なし

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、 サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示することはできますが、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リソースを作成または変更 AWS TNBするアクセス許可がありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または を使用してタスクを実行することはできません AWS API。必要なリソースに対してアクションを実行するアクセス許可をユーザーに付与するには、IAM管理者はIAMポリシーを作成できます。その後、管理者はIAMポリシーをロールに追加し、ユーザーはロールを引き受けることができます。

これらのポリシードキュメント例を使用して IAM ID ベースのJSON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 (コンソール) 」を参照してください。

各リソースタイプの の形式など AWS TNB、 で定義されるアクションとARNs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。

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、誰かがアカウント内のリソースを作成、アクセス、または削除 AWS TNB できるか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、最小権限のアクセス許可に移行 – ユーザーとワークロードへのアクセス許可の付与を開始するには、多くの一般的なユースケースのアクセス許可を付与するAWS管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有のAWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM「 ユーザーガイド」の「 管理AWS ポリシー」または ジョブ機能の 管理ポリシーを参照してください。 AWS
- 最小権限のアクセス許可を適用する – IAMポリシーでアクセス許可を設定する場合、タスクの実行に必要なアクセス許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAMを使用してアクセス許可を適用する方法の詳細については、IAM「 ユーザーガイド」の「 のポリシーとアクセス許可IAM」を参照してください。
- IAM ポリシーの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、アクションと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 を使用して送信する必要があることを指定できますSSL。などの特定の を通じてサービスアクションが使用されている場合 AWS のサービス、条件を使用して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、「 ユーザーガイド」のIAMJSON「ポリシー要素: 条件」を参照してください。 IAM
- IAM Access Analyzer を使用してIAMポリシーを検証し、安全で機能的なアクセス許可を確保する – IAM Access Analyzer は、ポリシーがポリシー言語 (JSON) とIAMベストプラクティスに準拠するように、新規および既存のIAMポリシーを検証します。IAM Access Analyzer には、安全で機能的なポリシーの作成に役立つ 100 を超えるポリシーチェックと実用的な推奨事項が用意されています。詳細については、IAM「 ユーザーガイド」のIAM「Access Analyzer でポリシーを検証する」を参照してください。
- 多要素認証が必要 (MFA) – 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、 をオンにMFAしてセキュリティを強化します。API オペレーションが呼び出されるMFAタイミングを要求するには、ポリシーにMFA条件を追加します。詳細については、「 ユーザーガイド」の「 によるセキュアAPIアクセスMFA」を参照してください。 IAM

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「 のセキュリティのベストプラクティスIAM」を参照してください。 IAM

コンソールの使用 AWS TNB

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、 内のリソースの詳細を AWS TNB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 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 AWS API。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にのみアクセスを許可します。

サービスロールポリシーの例

管理者は、環境テンプレートとサービステンプレートで定義されているように、 が AWS TNB作成するリソースを所有および管理します。ネットワークライフサイクル管理用のリソースの作成を許可する AWS TNB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、ユーザーに代わってリソースを呼び出して、ネットワークをインスタンス化および管理できます。サービスロールを指定すると、 AWS TNBはそのロールの認証情報を使用します。

サービスでサービスロールとそのアクセス許可ポリシーを作成しますIAM。サービスロールの作成の詳細については、IAM「 ユーザーガイド」のAWS 「サービスへの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、 に提供できます AWS TNB。このロールにより AWS TNB、 は Amazon Elastic Kubernetes Service などの他の サービスに呼び出しを行い 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、 で定義されているようにネットワーク機能をプロビジョニングできますNSD。

AWS TNB サービスIAMロールには、次の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は、ポリシーからスコープさ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 AWS TNB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:UpdateAddon",
                "eks:UpdateClusterVersion",
                "eks:UpdateNodegroupConfig",
                "eks:UpdateNodegroupVersion",
                "eks:DescribeUpdate",
                "eks:UntagResource",
                "eks:DescribeCluster",
                "eks:ListNodegroups",
                "eks:CreateAddon",
                "eks:DeleteAddon",
                "eks:DescribeAddon",
                "eks:DescribeAddonVersions",
                "s3:PutObject",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks",
                "cloudformation:UpdateStack",
                "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、 AWS TNB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 TNB Amazon EKSクラスターのサービスロール

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

次の例は、Amazon EKSクラスターポリシーのサービスロールを作成する AWS CloudFormation AWS TNB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::ロール
- スタックテンプレートの選択

AWS TNB Amazon EKSノードグループのサービスロール

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

次の例は、Amazon EKSノードグループポリシーのサービスロールを作成する AWS CloudFormation AWS TNB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::ロール
- スタックテンプレートの選択

AWS TNB Multus のサービスロール

で Amazon EKSリソースを作成しNSD、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_role` 属性を指定して、Multus の管理に使用するロールを指定する必要があります。

次の例は、Multus ポリシーのサービスロールを作成する AWS CloudFormation AWS TNB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::ロール
- スタックテンプレートの選択

AWS TNB ライフサイクルフックポリシーのサービスロール

NSD またはネットワーク関数パッケージで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環境を作成するための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サービスロールを作成する AWS CloudFormation AWS TNB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::ロール
- 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、IAMユーザーがユーザー ID にアタッチされているインラインポリシーとマネージドポリシーを表示できるようにするポリシーを作成する方法を示します。このポリシーには、コンソールで、または AWS CLI または 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 AWS API。

```
{
    "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 ID とアクセスのトラブルシューティング

以下の情報は、 および の使用 AWS TNB時に発生する可能性のある一般的な問題を診断して修正するのに役立ちますIAM。

問題

- でアクションを実行する権限がありません AWS TNB
- iam を実行する権限がありません。PassRole
- 自分の外部のユーザーに自分の AWS TNBリソース AWS アカウント へのアクセスを許可したい

でアクションを実行する権限がありません AWS TNB

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、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。

次のエラー例は、mateojacksonIAMユーザーがコンソールを使用して架空の*my-example-widget*リソースの詳細を表示しようとしたときに、架空のtnb:*GetWidget*アクセス許可がない場合に発生します。

```
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* リソースへのアクセスを許可するように更新する必要があります。

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

iam 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、 にロールを渡すことができるようにポリシーを更新する必要があります AWS TNB。

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、 という名前のIAMユーザーがコンソールを使用して marymajor でアクションを実行しようとすると発生します AWS TNB。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。メアリーには、ロールをサービスに渡す許可がありません。

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

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

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

自分の外部のユーザーに自分の 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 (ACLs) をサポートするサービスでは、これらのポリシーを使用して、リソースへのアクセスをユーザーに許可できます。

詳細については、以下を参照してください。

- がこれらの機能をサポートしているかどうか AWS TNBについては、「」を参照してくださいとの仕組み AWS TNB IAM。
- 所有 AWS アカウント している 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。
- サードパーティー にリソースへのアクセスを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「外部認証されたユーザーへのアクセスを提供する (ID フェデレーション)」を参照してください。
- クロスアカウントアクセスにロールとリソースベースのポリシーを使用する違いについては、IAM ユーザーガイドの「 のクロスアカウントリソースアクセスIAM」を参照してください。

のコンプライアンス検証 AWS TNB

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、AWS のサービス「コンプライアンスプログラムによるスコープ」の」の「」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することができます AWS Artifact。詳細については、「」の AWS Artifact」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、データの機密性、会社のコンプライアンス目的、および適用される法律と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。

- Amazon Web Services HIPAA のセキュリティとコンプライアンスのためのアーキテクチャ – 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**
>
> すべての AWS のサービス がHIPAA対象となるわけではありません。詳細については、HIPAA「対象サービスリファレンス」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、お客様の業界とロケーション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 ()、国際標準化機構 (ISO) などPCI) 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。
- AWS Config 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、リソース設定が内部プラクティス、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 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 を検出できます。 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出要件を満たすことでDSS、 PCI などのさまざまなコンプライアンス要件に対応するのに役立ちます。
- 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、リスクと規制や業界標準へのコンプライアンスの管理を簡素化できます。

でのレジリエンス AWS TNB

AWS グローバルインフラストラクチャは AWS リージョン を中心に構築されており、アベイラビリティーゾーン. は、低レイテンシー、高スループット、冗長性の高いネットワークに接続されている

複数の物理的に分離および分離された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) 内のEKSクラスターで Network Service を実行します。

でのインフラストラクチャセキュリティ AWS TNB

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 「 Cloud Security 」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。

AWS 公開されたAPI呼び出しを使用して、ネットワーク経由で にアクセスします AWS TNB。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) 。1.2 が必要でTLS、1.3 TLS をお勧めします。
- (DHE工フェメラルディフィ-ヘルマンPFS) や (エリプティックカーブ工フェメラルディフィ-ヘルマン) など、完全なフォワードシークレット ECDHE () を持つ暗号スイート。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。

さらに、リクエストは、アクセスキー ID とプリンIAM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 AWS TNB、以下を含む、 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 リソースと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
  - データの保存に使用する S3 バケット AWS TNB
  - を通じてプロビジョニングしたネットワークサービスをサポートするために使用するその他の AWS のサービス および リソース AWS TNB
  - アプリケーションコード
  - を介して AWS TNB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**
>
> 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責任があります AWS TNB。

ネットワーク接続セキュリティモデル

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 AWS TNB、選択した AWS リージョンにある仮想プライベートクラウド (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 、クラウド内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別にインフラストラクチャを分離します。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VPCs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顧客に対して を通じて AWS TNBプロビジョニングされたネットワークサービスは、同じ を共有しますVPC。異なる顧客向けに を通じて AWS TNBプロビジョニングされたネットワークサービスは、同じ 内で個別のコンピューティングインスタンスを使用しますVPC。

でのネットワークサービスクライアントとネットワークサービス間の通信は、インターネット AWS TNBを通過します。 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 Management Console、 AWS Command Line Interface （ AWS CLI ）、および AWS SDKs を介した への接続 AWS TNBは暗号化されます。

IMDS バージョン

AWS TNB は、セッション指向のメソッドである Instance Metadata Service バージョン 2 (IMDSv2) を活用するインスタンスをサポートします。IMDSv2 には、 よりも高いセキュリティが含まれていますIMDSV1。詳細については、「Amazon EC2 Instance Metadata Service の機能強化による、オープンファイアウォール、リバースプロキシ、SSRF脆弱性に対する詳細な防御の追加」を参照してください。

インスタンスを起動するときは、 を使用する必要がありますIMDSv2。の詳細についてはIMDSv2、「Amazon ユーザーガイド」の「 の使用IMDSv2」を参照してください。 EC2

モニタリング AWS TNB

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

を使用して CloudTrail 、 に対して行われた呼び出しに関する詳細情報をキャプチャします AWS APIs。これらの呼び出しはログ ファイルとして Amazon S3 に保存できます。これらの CloudTrail ログを使用して、どの呼び出しが行われたか、呼び出し元の送信元 IP アドレス、呼び出しを行ったユーザー、呼び出しが行われた日時などの情報を確認できます。

CloudTrail ログには、 のAPIアクションへの呼び出しに関する情報が含まれています AWS TNB。また、Amazon EC2や Amazon 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EBS。

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

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

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

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

CloudTrail アカウントを作成し、 CloudTrail イベント履歴 に自動的にアクセスできる AWS アカウント と、 は でアクティブになります。 CloudTrail イベント履歴には、過去 90 日間の で記録され

た管理イベントの表示可能、検索可能、ダウンロード可能、およびイミュータブルなレコードが表示されます AWS リージョン。詳細については、AWS CloudTrail 「 ユーザーガイド」の CloudTrail 「イベント履歴の操作」を参照してください。イベント履歴を表示するための料金は発生しません CloudTrail。

AWS アカウント 過去 90 日間のイベントを継続的に記録するには、証跡または CloudTrail Lake イベントデータストアを作成します。

CloudTrail 証跡

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

証跡を作成 CloudTrail することで、進行中の管理イベントのコピーを から Amazon S3 バケットに無料で配信できますが、Amazon S3 ストレージ料金が発生します。 CloudTrail 料金の詳細については、AWS CloudTrail 「料金」を参照してください。Amazon S3 の料金に関する詳細については、「Amazon S3 の料金」を参照してください。

CloudTrail Lake イベントデータストア

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

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

AWS TNB イベントの例

イベントは、任意の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APIオペレーション、オペレーションの日付と時刻、リクエストパラメータなどに関する情報が含まれます。 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、イベントは特定の順序で表示されません。

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

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

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


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

AWS TNB デプロイタスク

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

次の表に、 AWS TNB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	Amazon EKSクラスターに Multus プラグインを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クラスターのプラグインをアンインストールします。
ClusterStorageClassConfiguration	名前に変更なし	Amazon EKSクラスターのストレージクラス (CSI ドライバー) を設定します。

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

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

のサービスクォータ AWS TNB

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

以下は、 のサービスクォータです AWS TNB。

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

ユーザーガイドの AWS TNBドキュメント履歴

次の表に、 のドキュメントリリースを示します AWS TNB。

変更	説明	日付
クラスター用の Kubernetes バージョン	AWS TNB では、Amazon EKSクラスターを作成するために Kubernetes バージョン 1.30 がサポートされるようになりました。	2024 年 8 月 19 日
AWS TNB は、ネットワークライフサイクルを管理するための追加のオペレーションをサポートします。	インスタンス化されたネットワークインスタンスまたは以前に更新されたネットワークインスタンスを、新しいネットワークパッケージとパラメータ値で更新できます。以下を参照してください。 • ライフサイクルオペレーション • ネットワークインスタンスを更新する • AWS TNB サービスロールの例： • Amazon EKSアクションを追加します。`eks:UpdateAddon` 、`eks:UpdateClusterVersion` 、`eks:UpdateNodegroupConfig` 、`eks:UpdateNodegroupVersion` 、	2024 年 7 月 30 日

	`eks:DescribeUpdate` • この AWS CloudFormation アクションを追加します。 `cloudformation:UpdateStack` • 新しいデプロイタスク: `InfrastructureUpdate` 、`InventoryRegistration` 、`ValidateNetworkServiceUpdate` • API 更新: GetSolNetworkOperation、ListSolNetworkOperations 、および UpdateSolNetworkInstance	
既存のタスクの新しいタスク名と新しいタスク名	新しいタスクを使用できます。2024 年 3 月 7 日現在、一部の既存のタスクにはわかりやすくするために新しい名前が付けられています。	2024 年 5 月 7 日
クラスター用の Kubernetes バージョン	AWS TNB では、Amazon EKSクラスターを作成するための Kubernetes バージョン 1.29 がサポートされるようになりました。	2024 年 4 月 10 日
ネットワークインターフェイスのサポート `security_groups`	セキュリティグループを 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.EKSManagedNode または AWS.Compute.EKSSelfManagedNode ノードにプロパティを追加します。	2024 年 4 月 2 日
ノードのサポート `labels`	ノードラベルは、. AWSCompute.EKSManagedNode または AWS.Compute.EKSSelfManagedNode ノードのノードグループにアタッチできます。	2024 年 3 月 19 日
ネットワークインターフェイスのサポート `source_dest_check`	.Networking AWS ENI. ノードを介して、ネットワークインターフェイスのソース/宛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、利用可能な場合は multus_role プロパティENIsから、または node_role プロパティからに関連するアクセス許可を取得します。	2023 年 12 月 18 日
プライベートクラスターのサポート	AWS TNB がプライベートクラスターをサポートするようになりました。プライベートクラスターを指定するには、access プロパティを PRIVATE に設定します。	2023 年 12 月 11 日
クラスター用の Kubernetes バージョン	AWS TNB では、Amazon EKSクラスターを作成するための Kubernetes バージョン 1.28 がサポートされ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 のサポートを追加 IPv6	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。ノード AWS.Networking.VPC 、AWS.Networking.Subnet 、AWS.Networking.InternetGateway、AWS.Networking.SecurityGroupIngressRule、AWS.Networking.SecurityGroupEgressRule、および AWS.Compute.EKS IPv6の設定を確認します。また、NAT64ノード AWS.Networking.NATGateway と AWS.Networking.Route を設定に追加しました。アクセス IPv6許可の Amazon EKSノードグループのサービスロールとサービスロールを更新 AWS TNBしました AWS TNB。「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
サービスロールポリシーにアクセス許可を追加しました AWS TNB	Amazon S3 のサービスロールポリシーに AWS TNB および のアクセス許可を追加して AWS CloudFormation 、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。	2023 年 10 月 23 日

AWS TNB が他のリージョンで起動	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 のタグ。EKSSelfManagedNode	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、 を活用するインスタンスをサポートします。 IMDSv2	インスタンスを起動するときは、 を使用する必要がありますIMDSv2。	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 では、Amazon EKSクラスターを作成するための Kubernetes バージョン 1.27 がサポートされるようになりました。	2023 年 7 月 25 日
AWS.コンピューティング EKS..AuthRole	AWS TNB AuthRole は、ユーザーがIAMロールを使用して Amazon EKSクラスターにアクセスできる`aws-authConfigMap` ように、Amazon EKSクラスターにIAMロールを追加できるようにします。	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 では、Kubernetes バージョン 1.21 はサポートされなくなりました。	2023 年 5 月 11 日
AWS.コンピューティング。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 ユーザーガイドの最初のリリース AWS TNB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。