aws

用户指南

AWS 电信网络生成器

AWS 电信网络生成器: 用户指南

Table of Contents

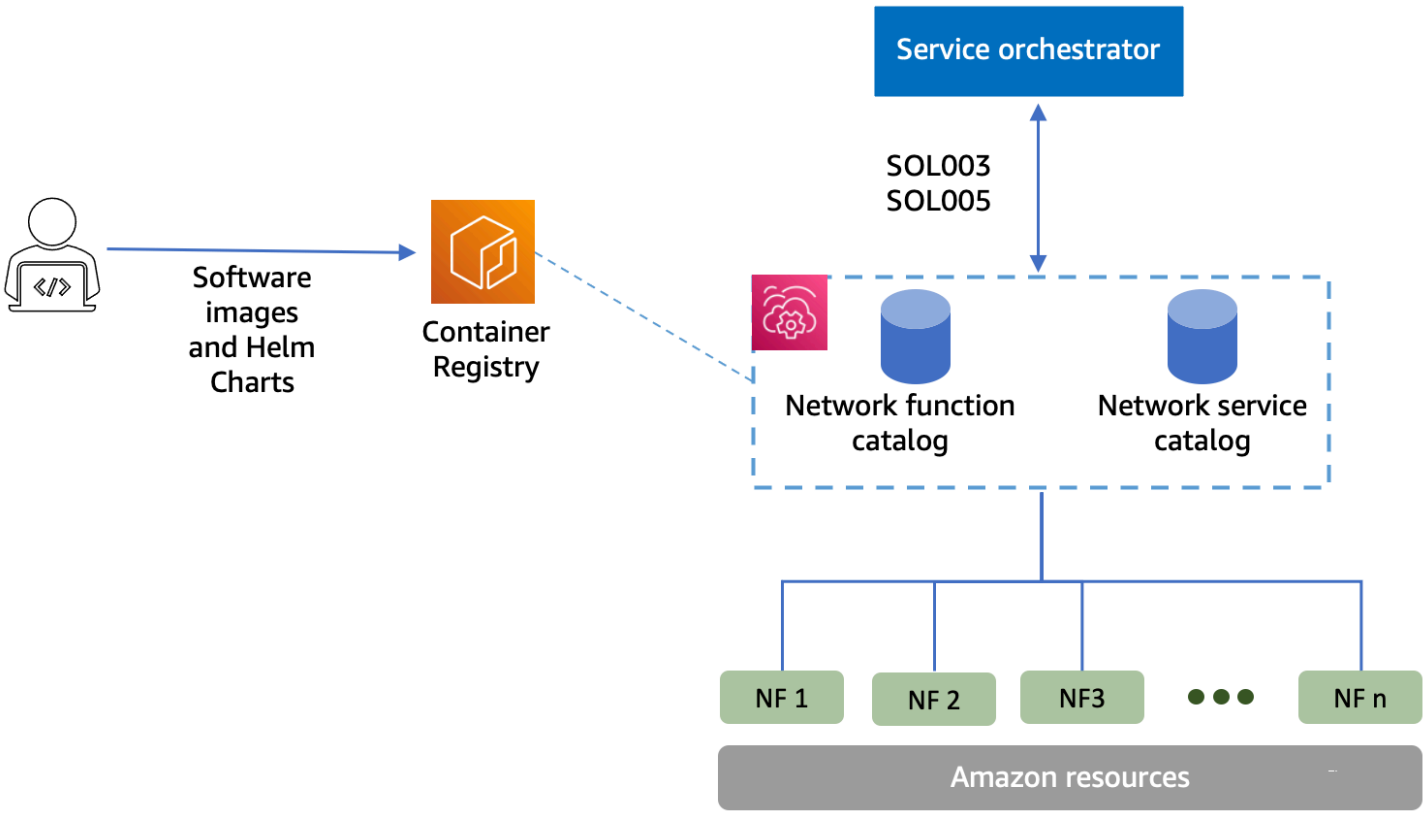
什么是 AWS 电信网络生成器？

AWS Telco AWS Network Builder (TNB) 是一项 AWS 服务，它为通信服务提供商 (CSPs) 提供了一种在基础设施上 AWS 部署、管理和扩展 5G 网络的有效方法。

借助 AWS TNB，您可以自动 AWS Cloud 使用网络映像部署可扩展且安全的5G网络。您无需学习新技术、决定使用哪种计算服务，也不需要知道如何预置和配置 AWS 资源。

相反，您需要描述您的网络基础架构，并提供来自独立软件供应商 (ISV) 合作伙伴的网络功能的软件映像。 AWS TNB 与第三方服务协调器和 AWS 服务集成，可自动配置必要的 AWS 基础架构、部署容器化网络功能以及配置网络和访问管理，从而创建全面运营的网络服务。

下图说明了 AWS TNB 和服务协调器之间的逻辑集成，以便使用基于欧洲电信标准协会 (ETSI) 的标准接口部署网络功能。



主题

- 新手 AWS？
- AWS TNB 是为谁准备的？
- AWS TNB 功能

新手 AWS？

如果您不熟悉 AWS 产品和服务，请通过以下资源开始了解更多信息：

- 简介 AWS
- 入门 AWS

AWS TNB 是为谁准备的？

AWS TNB 旨在 CSPs 利用所 AWS Cloud 提供的成本效益、敏捷性和弹性，而无需编写和维护用于设计、部署和管理网络服务的自定义脚本和配置。 AWS TNB 会自动配置必要的 AWS 基础架构，部署容器化网络功能，配置网络和访问管理，以根据 CSP 定义的网络服务描述符和 CSP 想要部署的网络功能创建完全可运行的网络服务。

AWS TNB 功能

以下是 CSP 想要使用 AWS TNB 的一些原因：

帮助简化任务

提高网络运营的效率，例如部署新服务、更新和升级网络功能以及更改网络基础设施拓扑。

与编排工具集成

AWS TNB 与符合 ETSI 标准的流行第三方服务协调器集成。

可扩展

您可以将 AWS TNB 配置为扩展底层 AWS 资源以满足流量需求，更有效地执行网络功能更新，推出网络基础设施拓扑更改，并将新 5G 服务的部署时间从几天缩短到几小时。

检查和监控资源 AWS

AWS TNB 允许您在单个控制面板上检查和监控支持您的网络的 AWS 资源，例如 Amazon VPC、Amazon 和 Amazon EC2 EKS。

支持服务模板

AWS TNB 允许您为所有电信工作负载（RAN、Core、IMS）创建服务模板。您可以创建新的服务定义、重复使用现有模板或与持续集成和持续交付（CI/CD）管道集成以发布新定义。

跟踪网络部署的变化

当您更改网络功能部署的底层配置（例如更改 Amazon 实例类型的 EC2 实例类型）时，您可以以可重复和可扩展的方式跟踪更改。手动执行此操作需要管理网络状态，创建和删除资源，并注意所需更改的顺序。使用 AWS TNB 管理网络功能的生命周期时，您只需要更改描述网络功能的网络服务描述符。 AWS 然后，TNB 将按正确的顺序自动进行所需的更改。

简化网络功能生命周期

您可以管理网络功能的第一个和所有后续版本，并指定何时升级。您也可以用同样的方式管理 RAN、Core、IMS 和网络应用程序。

正在访问 AWS TNB

您可以使用以下任何接口创建、访问和管理 AWS TNB 资源：

- AWS TNB 控制台 — 提供用于管理网络的 Web 界面。
- AWS TNB API — 提供用于执行 AWS TNB 操作 RESTful 的 API。有关更多信息，请参阅 AWS TNB API Reference。
- AWS Command Line Interface (AWS CLI) — 为包括 AWS TNB 在内的一系列 AWS 服务提供命令。它在 Windows、macOS 和 Linux 上受支持。有关更多信息，请参阅 用户指南。AWS Command Line Interface
- AWS SDKs— 提供特定语言 APIs 并完成许多连接细节。这些细节工作包括计算签名、处理请求重试和处理错误。有关更多信息，请参阅 AWS SDKs。

AWS TNB 的定价

AWS TNB帮助 CSPs 自动化其电信网络的部署和管理。 AWS使用 AWS TNB 时，您需要为以下两个维度付费：

- 托管的网络功能项目（MNFI）小时数。
- API 请求的数量。

与 AWS TNB一起使用其他 AWS 服务时，您还会产生额外费用。有关更多信息，请参阅 AWS TNB Pricing。

若要查看您的账单，请转到 AWS 账单与成本管理 控制台中的账单和成本管理控制面板。您的账单中包含了提供您的账单更多详情的使用情况报告的链接。有关 AWS 账户账单的更多信息，请参阅AWS 账户账单。

如果您对 AWS 账单、账户和活动有疑问，请联系 Su AWS pport。

AWS Trusted Advisor 是一项服务，你可以用它来帮助优化 AWS 环境的成本、安全性和性能。有关更多信息，请参阅 AWS Trusted Advisor。

接下来做什么

有关如何开始使用 AWS TNB 的更多信息，请参阅以下主题：

- 设置 AWS TNB – 完成先决步骤。
- AWS TNB 入门 – 部署您的第一个网络功能，例如集中式单元（CU）、访问和移动管理功能（AMF）、用户面功能（UPF）或完整的 5G 核心。

AWS TNB 是如何运作的

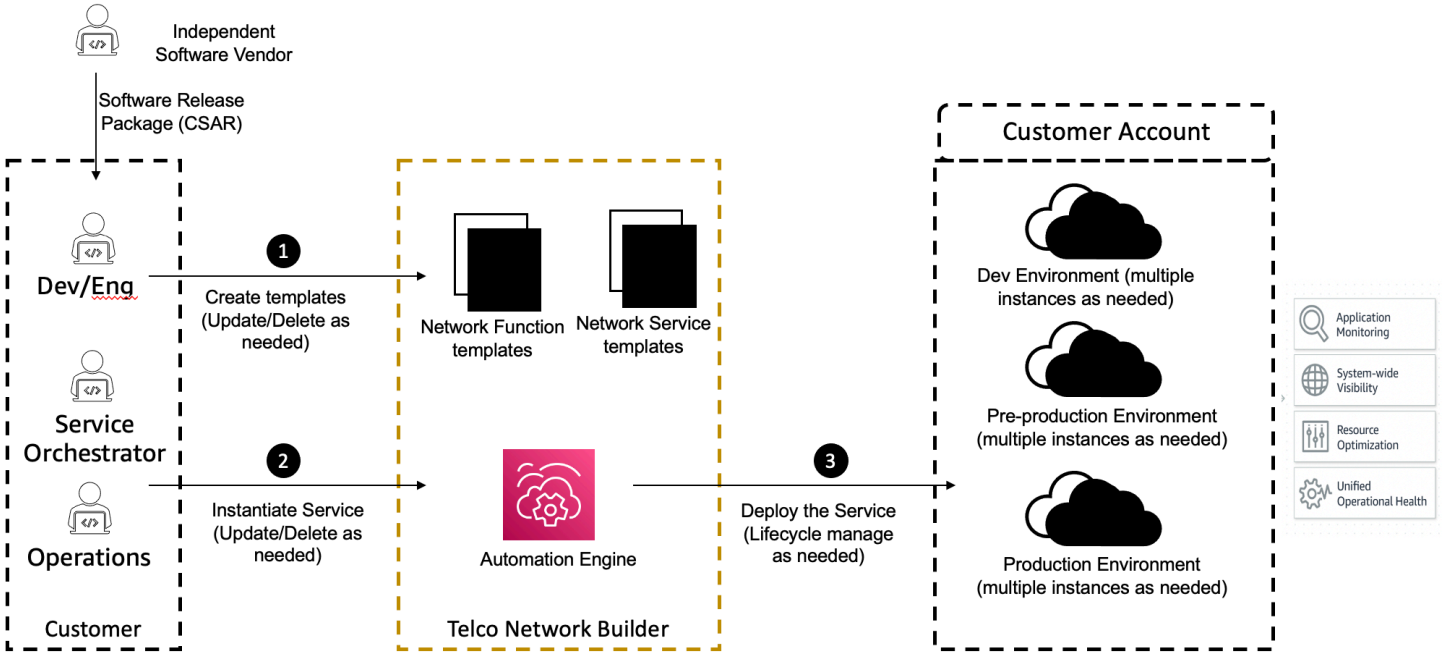
AWS TNB 与标准化 end-to-end协调器和 AWS 资源集成，可运行完整的 5G 网络。

AWS TNB 允许您摄取网络功能包和网络服务描述符 (NSDs)，并为您提供操作网络的自动化引擎。您可以使用您的 end-to-end协调器并与 AWS TNB 集成 APIs，也可以使用 T AWS NB SDKs 来构建自己的自动化流程。有关更多信息，请参阅 AWS TNB 架构。

主题

AWS TNB 架构

AWS TNB 使您能够通过 AWS 管理控制台、 AWS CLI、T AWS NB REST API 和执行生命周期管理操作。 SDKs这使不同的 CSP 角色（例如工程、运营和编程系统团队的成员）都可以利用 AWS TNB。您可以创建网络功能包并将其作为云服务存档（CSAR）文件上传。CSAR 文件包含 Helm 图表、软件映像和网络功能描述文件（NFD）。您可以使用模板来重复部署该功能包的多个配置。您可以创建网络服务模板，定义要部署的基础设施和网络功能。您可以使用参数覆盖，在不同的位置部署不同的配置。然后，您可以使用模板实例化网络，并在基础设施上 AWS 部署网络功能。 AWS TNB 为您提供部署的可见性。



与集成 AWS 服务

5G 网络由一组互连的容器化网络功能组成，这些功能部署在数千个 Kubernetes 集群中。 AWS TNB 与以下特定于电信 AWS 服务 的功能集成 APIs ，以创建全面运行的网络服务：

- Amazon Elastic Container Registry (Amazon ECR) Container Registry，用于存储独立软件供应商 ISVs () 网络功能工件。
- Amazon Elastic Kubernetes Service （ Amazon EKS ）来设置集群。
- Amazon VPC，用于网络结构。
- 使用安全组 CloudFormation。
- AWS CodePipeline 对于跨区域的部署目标 AWS 区域，L AWS ocal Zones 和 AWS Outposts。
- IAM，用于定义角色。
- AWS Organizations 控制对 AWS TNB APIs 的访问权限。
- Health Dashboard AWS CloudTrail 并监控运行状况和发布指标。

AWS TNB 资源配额

您的每个配额 AWS 账户 都有默认配额，以前称为限制 AWS 服务。除非另有说明，否则每个配额都特定于 AWS 区域。您只能请求提高某些配额，并非所有。

要查看 AWS TNB 的配额，请打开 S ervice Quotas 控制台。在导航窗格中，选择 AWS 服务，然后选择 AWS TNB。

要请求提高限额，请参阅《Service Quotas User Guide》中的 Requesting a quota increase。

您 AWS 账户 有以下与 AWS TNB 相关的配额。

资源限额	说明	默认 值	是否可调整？
网络服务实例数量	一个区域内网络服务实例的最大数量。	800	是
持续并发执行的网络服务操作数量	一个区域内可持续并发执行的网络服务操作的最大数量。	40	是

资源限额	说明	默认 值	是否可调整？
网络包数量	一个区域内网络包的最大数量。	40	是
功能包数量	一个区域内功能包的最大数量。	200	是

AWS TNB 概念

本主题介绍一些基本概念，可帮助您开始使用 AWS TNB。

内容

网络功能的生命周期

AWS TNB 在网络功能的整个生命周期中为您提供帮助。网络功能生命周期包括以下阶段和活动：

规划

1. 通过确定要部署的网络功能来规划您的网络。
2. 将网络功能软件映像放入容器映像存储库中。
3. 创建要部署或升级的 CSAR 包。
4. 使用 AWS TNB 上传定义您的网络功能的 CSAR 包（例如 CU AMF 和 UPF），并与持续集成和持续交付 (CI/CD) 管道集成，该管道可以帮助您在新的网络功能软件映像或客户脚本可用时创建 CSAR 包的新版本。

配置

1. 确定部署所需的信息，例如计算类型、网络功能版本、IP 信息和资源名称。
2. 使用这些信息创建网络服务描述文件（NSD）。
3. 载入 NSDs 定义您的网络功能和网络功能实例化所需的资源。

实例化

1. 创建网络功能所需的基础设施。
2. 按照 NSD 中的定义对网络功能进行实例化（或预置），然后开始传输流量。
3. 验证资产。

生产

在网络功能的生命周期中，您会执行一系列生产操作，例如：

- 更新网络功能配置，例如，更新已部署的网络功能中的值。
- 使用新的网络包和参数值更新网络实例。例如，更新网络包中的 Amazon EKS `version` 参数。

使用标准化接口

AWS TNB 与符合欧洲电信标准协会 (ETSI) 标准的服务协调器集成，使您能够简化网络服务的部署。服务协调器可以使用 AWS TNB SDKs、CLI 或 the APIs 来启动操作，例如实例化网络功能或将网络功能升级到新版本。

AWS TNB 支持以下规格。

规范	发布版本	描述
ETSI SOL001	v3.6.1	定义允许使用基于 TOSCA 的网络功能描述文件的标准。
ETSI SOL002	v3.6.1	围绕网络功能管理定义模型。
ETSI SOL003	v3.6.1	定义网络功能生命周期管理的标准。
ETSI SOL004	v3.6.1	定义网络功能包的 CSAR 标准。
ETSI SOL005	v3.6.1	定义网络服务包和网络服务生命周期管理的标准。
ETSI SOL007	v3.5.1	定义允许使用基于 TOSCA 的网络服务描述文件的标准。

功能包

使用 AWS TNB，您可以将符合 ETSI SOL001/SOL004 的函数包存储到函数目录中。然后，您可以上传包含描述您的虚拟网络功能的构件的云服务存档 (CSAR) 软件包。

- 虚拟网络功能描述符 — 定义软件包载入和虚拟网络功能管理的元数据。您必须将此文件命名为 `vnfd.yaml`。
- 软件镜像 — 引用虚拟网络功能容器镜像。Amazon Elastic Container Registry (Amazon ECR) 可以充当您的虚拟网络功能镜像存储库。

- 其他文件-用于管理虚拟网络功能；例如脚本和 Helm 图表。

CSAR 是一个由 OASIS TOSCA 标准定义的软件包，包括一个符合 OASIS TOSCA YAML 规范的网络/服务描述符。有关所需的 YAML 规范的信息，请参阅TNB 的 TOSCA 参考文献 AWS。

以下是虚拟网络函数描述符的示例。

```
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"
```

网络套餐

网络包是 CSAR（云服务存档）格式的.zip文件。它定义了您要部署的功能包以及要部署它们的基础 AWS 架构。

网络包包含以下文件：

- TOSCA 格式的网络描述符文件 (nsd.yaml)，如 ETSI SOL007 所述。

  该nsd.yaml文件包含对上传的函数包及其描述符 IDs的引用。
- 用户数据脚本（如果有）。
- 生命周期挂钩脚本（如果有）。

- 插件的values.yaml配置文件（如果有）。

AWS TNB 支持使用 TOSCA 语言对网络、服务和功能等资源进行建模的 ETSI 标准。 AWS TNB 以符合 ETSI 标准的服务 AWS 服务 协调器可以理解的方式对其进行建模，从而提高您的使用效率。

TNB 的网络服务描述符 AWS

网络服务描述文件（NSD）是网络包中的一个 .yaml 文件，它使用 TOSCA 标准来描述要部署的网络功能以及要在其上部署网络功能的 AWS 基础架构。要定义您的 NSD 并配置底层资源和网络生命周期操作，您必须了解 TNB 支持的 NSD TOSCA 架构。 AWS

NSD 文件分为以下几个部分：

1. TOSCA 定义版本 – 这是 NSD YAML 文件的第一行，包含版本信息，如以下示例所示。

```
tosca_definitions_version: tnb_simple_yaml_1_0
```

2. VNFD – NSD 包含要在其上执行生命周期操作的网络功能的定义。必须使用以下值来标识每个网络功能：
   - 用于 descriptor_id 的唯一 ID。该 ID 必须与网络功能 CSAR 包中的 ID 相匹配。
   - 用于 namespace 的唯一名称。该名称必须与唯一 ID 相关联，以便在整个 NSD YAML 文件中更容易引用，如以下示例所示。

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

3. 拓扑模板 – 定义要部署的资源、网络功能部署以及任何自定义脚本，例如生命周期挂钩。如以下示例所示。

```
topology_template:

  node_templates:

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

4. 其它节点 – 每个建模的资源都有属性和要求部分。属性部分描述了资源的可选或必备属性，例如版本。要求部分描述了必须作为参数提供的依赖项。例如，要创建 Amazon EKS 节点组资源，必须在 Amazon EKS 集群中创建该资源。如以下示例所示。

```
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
```

示例 NSD

以下是展示如何建模的 NSD 片段。 AWS 服务网络功能将部署在使用 Kubernetes 版本 1.27 的 Amazon EKS 集群上。应用程序的子网是 Subnet01 和 Subnet02。然后，您可以使用 Amazon 系统映像 (AMI)、实例类型和自动扩展配置为您的应用程序定义。 NodeGroups

```
tosca_definitions_version: tnb_simple_yaml_1_0

SampleNFEKS:
  type: tosca.nodes.AWS.Compute.EKS
  properties:
    version: "1.27"
    access: "ALL"
```

```
    cluster_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleClusterRole"
  capabilities:
    multus:
      properties:
        enabled: true
  requirements:
    subnets:
      - Subnet01
      - Subnet02

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

AWS TNB 的管理和运营

借 AWS 助 TNB，您可以使用符合 ETSI SOL003 和 SOL005 的标准化管理操作来管理您的网络。您可以使用 AWS TNB APIs 执行生命周期操作，例如：

- 实例化网络功能。
- 终止网络功能。
- 更新网络功能以覆盖 Helm 部署。

- 使用新的网络包和参数值更新实例化或更新的网络实例。
- 管理网络功能包的版本。
- 管理您的版本 NSDs.
- 检索有关您部署的网络功能的信息。

设置 AWS TNB

通过完成本主题中描述的任务来设置 AWS TNB。

任务

- 注册获取 AWS 账户
- 选择一个 AWS Region
- 记下服务端点
- （可选）安装 AWS CLI
- 设置 AWS TNB 角色

注册获取 AWS 账户

要开始使用 AWS，你需要一个 AWS 账户。有关创建的信息 AWS 账户，请参阅《AWS 账户管理 参考指南》 AWS 账户中的入门指南。

选择一个 AWS Region

要查看 AWS TNB 可用区域列表，请参阅AWS 区域服务列表。要查看用于编程访问的端点列表，请参阅中《AWS 一般参考》的 AWS TNB endpoints。

记下服务端点

要以编程方式连接到 AWS 服务，请使用终端节点。除标准 AWS 终端节点外，某些 AWS 服务还在选定区域提供 FIPS 终端节点。有关更多信息，请参阅 AWS service endpoint。

区域名称	区域	端点	协议	
美国东部（弗吉尼亚州北部）	us-east-1	tnb.us-east-1.amazonaws.com	HTTPS	

区域名称	区域	端点	协议
美国西部（俄勒冈州）	us-west-2	tnb.us-west-2.amazonaws.com	HTTPS
亚太地区（首尔）	ap-northeast-2	tnb.ap-northeast-2.amazonaws.com	HTTPS
亚太地区（悉尼）	ap-southeast-2	tnb.ap-southeast-2.amazonaws.com	HTTPS
加拿大（中部）	ca-central-1	tnb.ca-central-1.amazonaws.com	HTTPS
欧洲地区（法兰克福）	eu-central-1	tnb.eu-central-1.amazonaws.com	HTTPS
欧洲地区（巴黎）	eu-west-3	tnb.eu-west-3.amazonaws.com	HTTPS
欧洲（西班牙）	eu-south-2	tnb.eu-south-2.amazonaws.com	HTTPS
欧洲地区（斯德哥尔摩）	eu-north-1	tnb.eu-north-1.amazonaws.com	HTTPS
南美洲（圣保罗）	sa-east-1	tnb.sa-east-1.amazonaws.com	HTTPS

（可选）安装 AWS CLI

AWS Command Line Interface (AWS CLI) 为各种 AWS 产品提供命令，并在 Windows、macOS 和 Linux 上受支持。您可以使用访问 AWS TNB。 AWS CLI要开始使用，请参阅《AWS Command Line Interface 用户指南》。有关 AWS TNB 命令的更多信息，请参阅《AWS CLI 命令参考》中的 tnb。

设置 AWS TNB 角色

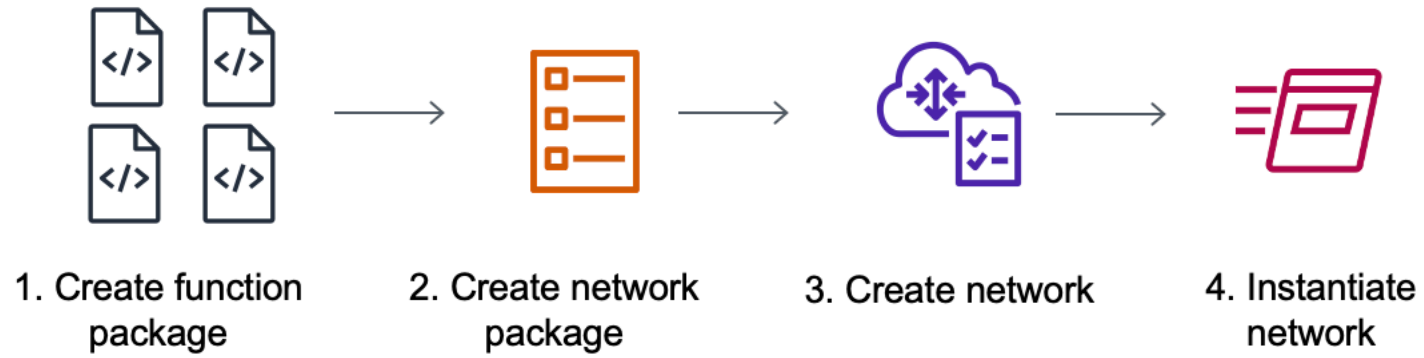
您必须创建 IAM 服务角色才能管理 AWS TNB 解决方案的不同部分。 AWS TNB 服务角色可以代表您对其他 AWS 服务（例如、 AWS CloudFormation AWS CodeBuild、以及各种计算和存储服务）进行 API 调用，以实例化和管理用于部署的资源。

有关 AWS TNB 服务角色的更多信息，请参阅的身份和访问管理 AWS TNB。

AWS TNB 入门

本教程演示如何使用 AWS TNB 部署网络功能，例如集中单元 (CU)、访问和移动管理功能 (AMF) 或 5G 用户平面功能 (UPF)。

下图说明了部署过程：



任务

- 先决条件
- 创建功能包
- 创建网络包
- 创建和实例化网络实例
- 清理

先决条件

在成功执行部署之前，必须具备以下条件：

- B AWS usiness Support 计划。
- 通过 IAM 角色获得的权限。
- 符合 ETSI SOL001/SOL004 标准的网络功能 (NF) 软件包。
- 符合 ETSI SOL007 的@@ 网络服务描述符 (NSD) 模板。

您可以使用 T AWS NB 示例包 GitHub 网站中的示例函数包或网络包。

创建功能包

网络功能包是云服务存档 (CSAR) 文件。CSAR 文件包含 Helm 图表、软件映像和网络功能描述文件（NFD）。

创建功能包

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择功能包。
3. 选择创建功能包。
4. 在上传函数包下，选择选择文件，然后将每个 CSAR 包作为.zip文件上传。您最多可以上传 10 个文件。
5. （可选）在 “标签” 下，选择 “添加新标签”，然后输入键和值。您可以使用标签来搜索和筛选资源或跟踪 AWS 成本。
6. 选择下一步。
7. 查看包详细信息，然后选择创建功能包。

创建网络包

网络包指定要部署的网络功能以及如何将其部署到目录中。

创建网络包

1. 在导航窗格中，选择网络包。
2. 选择创建网络包。
3. 在 “上传网络包” 下，选择 “选择文件”，然后将每个 NSD 作为.zip文件上传。您最多可以上传 10 个文件。
4. （可选）在 “标签” 下，选择 “添加新标签”，然后输入键和值。您可以使用标签来搜索和筛选资源或跟踪 AWS 成本。
5. 选择下一步。
6. 选择创建网络包。

创建和实例化网络实例

网络实例是在 AWS TNB 中创建的可以部署的单个网络。您必须创建网络实例并对其进行实例化。当您实例化网络实例时， AWS TNB 会配置必要的 AWS 基础架构，部署容器化网络功能，并配置网络和访问管理以创建全面运行的网络服务。

创建和实例化网络实例

1. 在导航窗格中，选择网络。
2. 选择创建网络实例。
3. 输入网络的名称和描述，然后选择下一步。
4. 选择网络套餐。验证详细信息并选择 “下一步”。
5. 选择创建网络实例。初始状态为 `Created`。

   将出现 “网络” 页面，显示`Not instantiated`处于状态的新网络实例。
6. 选择网络实例，选择操作和实例化。

   将出现 “网络实例化” 页面。
7. 查看详细信息并更新参数值。对参数值的更新仅适用于此网络实例。NSD 和 VNFD 软件包中的参数不会改变。
8. 选择实例化网络。

   将出现 “部署状态” 页面。
9. 使用刷新图标跟踪您的网络实例的部署状态。您也可以在 “部署任务” 部分启用自动刷新，以跟踪每个任务的进度。

清理

现在，您可以删除为本教程创建的资源。

清理资源

1. 在导航窗格中，选择网络。
2. 选择网络的 ID，然后选择终止。
3. 提示进行确认时，输入网络 ID，然后选择终止。
4. 使用刷新图标跟踪您的网络实例的状态。

5. （可选）选择网络，然后选择删除。

适用于 AWS TNB 的功能包

功能包是 CSAR（云服务存档）格式的.zip 文件，其中包含网络功能（ETSI 标准电信应用程序）和功能包描述文件，后者使用 TOSCA 标准来描述网络功能应如何在您的网络上运行。

任务

- 在 AWS TNB 中创建函数包
- 在 AWS TNB 中查看功能包
- 从 AWS TNB 下载功能包
- 从 AWS TNB 中删除一个函数包

在 AWS TNB 中创建函数包

了解如何在 AWS TNB 网络函数目录中创建函数包。创建函数包是在 AWS TNB 中创建网络的第一步。上传函数包后，您可以创建网络包。

Console

使用控制台创建功能包

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择功能包。
3. 选择创建功能包。
4. 选择 “选择文件”，然后将每个 CSAR 包作为.zip文件上传。您最多可以上传 10 个文件。
5. 选择下一步。
6. 查看包的详细信息。
7. 选择创建功能包。

AWS CLI

要使用创建函数包 AWS CLI

1. 使用create-sol-function-package命令创建新的函数包：

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

2. 使用 put-sol-function-package-conten t 命令上传函数包内容。例如：

```
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
```

在 AWS TNB 中查看功能包

了解如何查看功能包的内容。

Console

使用控制台查看功能包

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择功能包。
3. 使用搜索框找到功能包

AWS CLI

要使用查看函数包 AWS CLI

1. 使用list-sol-function-packages命令列出您的函数包。

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

2. 使用get-sol-function-package命令查看有关函数包的详细信息。

```
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
```

从 AWS TNB 下载功能包

了解如何从 AWS TNB 网络功能目录中下载功能包。

Console

使用控制台下载功能包

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在控制台左侧的导航窗格中，选择功能包。
3. 使用搜索框找到功能包
4. 选择功能包
5. 依次选择操作、下载。

AWS CLI

要使用下载函数包 AWS CLI

使用 get-sol-function-package-conten t 命令下载函数包。

```
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
```

从 AWS TNB 中删除一个函数包

了解如何从 AWS TNB 网络功能目录中删除功能包。要删除功能包，该功能包必须处于禁用状态。

Console

使用控制台删除功能包

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择功能包。
3. 使用搜索框找到功能包。

4. 选择功能包。
5. 依次选择操作、禁用。
6. 依次选择操作、删除。

AWS CLI

要删除函数包，请使用 AWS CLI

1. 使用update-sol-function-package命令禁用函数包。

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

2. 使用delete-sol-function-package命令删除函数包。

```
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
```

适用于 AWS TNB 的网络套餐

网络包是 CSAR（云服务存档）格式的.zip 文件。它定义了您要部署的功能包以及要部署它们的基础 AWS 架构。

网络包包含以下文件：

- TOSCA 格式的网络描述符文件 (nsd.yaml)，如 ETSI 所述。 SOL007

  该nsd.yaml文件包含对上传的函数包及其描述符 IDs的引用。
- 用户数据脚本（如果有）。
- 生命周期挂钩脚本（如果有）。
- 插件的values.yaml配置文件（如果有）。

任务

- 在 AWS TNB 中创建网络包
- 在 AWS TNB 中查看网络套餐
- 从 AWS TNB 下载网络包
- 从 AWS TNB 中删除网络包

在 AWS TNB 中创建网络包

网络包由网络服务描述文件（NSD，必需）和任何其它文件（可选，例如特定于您需求的脚本）组成。例如，如果您的网络包中有多个功能包，则可以使用 NSD 来定义哪些网络功能应在某些 VPCs子网或 Amazon EKS 集群中运行。

创建了功能包后再创建网络包。创建网络包后，您需要创建一个网络实例。

Console

使用控制台创建网络包

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择网络包。
3. 选择创建网络包。
4. 选择 “选择文件”，然后将每个 NSD 作为.zip文件上传。您最多可以上传 10 个文件。

5. 选择下一步。
6. 查看包的详细信息。
7. 选择创建网络包。

AWS CLI

要使用创建网络包 AWS CLI

1. 使用create-sol-network-package命令创建网络包。

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

2. 使用 put-sol-network-package-conten t 命令上传网络包内容。例如：

```
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
```

在 AWS TNB 中查看网络套餐

了解如何查看网络包的内容。

Console

使用控制台查看网络包

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择网络包。
3. 使用搜索框找到网络包。

AWS CLI

要查看网络套餐，请使用 AWS CLI

1. 使用list-sol-network-packages命令列出您的网络软件包。

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

2. 使用get-sol-network-package命令查看有关网络包的详细信息。

```
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
```

从 AWS TNB 下载网络包

了解如何从 AWS TNB 网络服务目录中下载网络包。

Console

使用控制台下载网络包

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择网络包。
3. 使用搜索框找到网络包
4. 选择网络包。
5. 依次选择操作、下载。

AWS CLI

要使用下载网络包 AWS CLI

- 使用 get-sol-network-package-conten t 命令下载网络软件包。

```
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
```

从 AWS TNB 中删除网络包

了解如何从 AWS TNB 网络服务目录中删除网络包。要删除网络包，该网络包必须处于禁用状态。

Console

使用控制台删除网络包

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择网络包。
3. 使用搜索框找到网络包
4. 选择网络包
5. 依次选择操作、禁用。
6. 依次选择操作、删除。

AWS CLI

要删除网络包，请使用 AWS CLI

1. 使用update-sol-network-package命令禁用网络包。

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

2. 使用delete-sol-network-package命令删除网络包。

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

AWS TNB 的网络实例

网络实例是在 AWS TNB 中创建的可以部署的单个网络。

任务

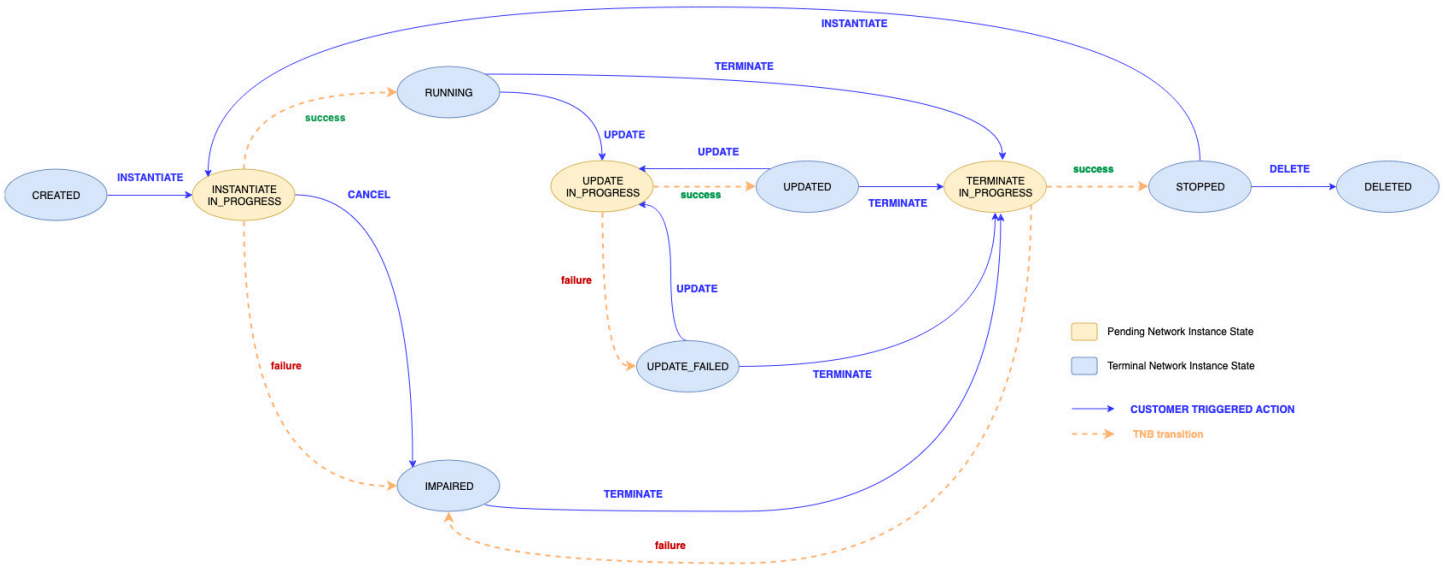
- 网络实例的生命周期操作
- 使用 AWS TNB 创建网络实例
- 使用 TNB 实例化网络实例 AWS
- 在 AWS TNB 中更新函数实例
- 在 AWS TNB 中更新网络实例
- 在 AWS TNB 中查看网络实例
- 终止并从 AWS TNB 中删除网络实例

网络实例的生命周期操作

AWS TNB允许您使用符合ETSI的标准化管理操作轻松管理网络，以及 SOL003 . SOL005 您可以执行以下生命周期操作：

- 创建网络
- 实例化网络
- 更新网络功能
- 更新网络实例
- 查看网络详情和状态
- 终止网络

下图显示了网络管理操作：



使用 AWS TNB 创建网络实例

网络实例需在创建了网络包之后创建。创建网络实例后，对其进行实例化。

Console

使用控制台创建网络实例

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择网络。
3. 选择创建网络实例。
4. 为实例输入名称和描述，然后选择下一步。
5. 选择网络包，验证详细信息，然后选择下一步。
6. 选择创建网络实例。

新的网络实例将显示在 “网络” 页面上。接下来，实例化这个网络实例。

AWS CLI

要使用创建网络实例 AWS CLI

- 使用create-sol-network-instance命令创建网络实例。

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

接下来，实例化这个网络实例。

使用 TNB 实例化网络实例 AWS

创建网络实例后，必须对其进行实例化。当您实例化网络实例时， AWS TNB 会配置必要的 AWS 基础架构，部署容器化网络功能，并配置网络和访问管理以创建全面运行的网络服务。

Console

使用控制台实例化网络实例

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择网络。
3. 选择要实例化的网络实例。
4. 选择 “操作”，然后选择 “实例化”。
5. 在实例化网络页面上，查看详细信息，也可以更新参数值。

   对参数值的更新仅适用于此网络实例。NSD 和 VNFD 软件包中的参数不会改变。
6. 选择实例化网络。

   此时将出现 “部署状态” 页面。
7. 使用刷新图标跟踪您的网络实例的部署状态。您也可以在 “部署任务” 部分启用自动刷新，以跟踪每个任务的进度。

   当部署状态更改为时Completed，网络实例即被实例化。

AWS CLI

要使用实例化网络实例 AWS CLI

1. 使用instantiate-sol-network-instance命令实例化网络实例。

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

2. 接下来，查看网络运行状态。

在 AWS TNB 中更新函数实例

实例化网络实例后，您可以更新网络实例中的函数包。

Console

使用控制台更新函数实例

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择网络。
3. 选择网络实例。只有当网络实例的状态为时，您才能对其进行更新Instantiated。

   将显示网络实例页面。
4. 从 “函数” 选项卡中，选择要更新的函数实例。
5. 选择更新。
6. 输入您的更新优先选项。
7. 选择更新。

AWS CLI

使用 CLI 更新函数实例

使用MODIFY_VNF_INFORMATION更新类型的update-sol-network-instance命令来更新网络实例中的函数实例。

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

在 AWS TNB 中更新网络实例

实例化网络实例后，您可能需要更新基础设施或应用程序。为此，您需要更新网络实例的网络包和参数值，然后部署更新操作以应用更改。

注意事项

- 您可以更新处于Instantiated或Updated状态的网络实例。
- 更新网络实例时，UpdateSolNetworkServiceAPI 使用新的网络包和参数值来更新网络实例的拓扑。
- AWS TNB 验证网络实例中 NSD 和 VNFD 参数的数量是否不超过 200。强制执行此限制是为了防止不良行为者通过错误或庞大的有效载荷来影响服务。

您可以更新的参数

在更新实例化的网络实例时，您可以更新以下参数：

参数	说明	示例：之前	示例：之后

亚马逊 EKS 集群版本	您可以将 Amazon EKS 集群控制平面version参数的值更新到下一个次要版本。您无法降级版本。	`EKSCluster:`
		`  type: tosca.nodes.AWS.Compute.EKS`
		`  properties:`
		`  version: "1.28"`
			`EKSC`
			`r:`
			`typ`
			`tos`
			`es.A`
			`mput`
			`pro`
			`s:`

参数	说明	示例：之前	示例：之后
			ver "1.

参数	说明	示例：之前	示例：之后

Amazon EKS Worker 节点	您可以更新EKSManagedNode kubernetes_version 参数的值以将您的节点组升级到较新的 Amazon EKS 版本，也可以更新ami_id参数以将您的节点组升级到最新 EKS 优化的 AMI。
	
	您可以更新的 AMI ID EKSSelfManagedNode 。AMI 的 Amazon EKS 版本必须与亚马逊 EKS 集群版本相同或最多低于 2 个版本。例如，如果亚马逊 EKS 集群版本为 1.31，则亚马逊 EKS AMI 版本必须为 1.31、1.30 或 1.29。
		`EKSManagedNodeGroup01:`
		`  ...`
		`  properties:`
		`  kubernetes_version: "  1.28"`
		`  EKSSelfManagedNode01:`
		`  compute:`
		`  compute:`
		`  properties:`
		`  ami_id: "ami-1231230LD "`
			`EKSM`
			`dNoc`
			`p01:`
			`...`
			`pro`
			`s:`
			`kub`
			`s_ve`
			`:`
			`"1.`
			`EKS`
			`nage`
			`01:`
			`com`

参数	说明	示例：之前	示例：之后
			com pro s: ami "am 3NEW

参数	说明	示例：之前	示例：之后

亚马逊 EKS 节点组	您可以根据计算需求添加或移除节点组。
	
	删除现有节点组并添加新节点组时，请确保新节点组与已删除的节点组不同 IDs ，否则该操作将被视为修改节点组，而不是删除和添加。请注意，对于现有节点组，只能更新有限的参数集。滚动浏览此表，查看可以更新哪些参数。
		`Free5GCEKSNode01:`
		`  type: tosca.nodes.AWS.Compute.EKSManagedNode`
		`  ...`
		`    scaling:`
		`      properties:`
		`        desired_size: 1`
		`        min_size: 1`
		`        max_size: 1`
		`  ...`
		`Free5GCEKSNode02 : # Deleted Nodegroup`
		`  type: tosca.nodes.AWS.Compute.EKSManagedNode`
		`  ...`
		`    scaling:`
		`      properties:`
		`        desired_size: 1`
		`        min_size: 1`
		`        max_size: 1`
		`  ...`
		`Free5GCEKSNode03 : # Deleted Nodegroup`
		`  type: tosca.nodes.AWS.Compute.EKSSelfManagedNode`
		`  ...`
		`    scaling:`
		`      properties:`
		`        desired_size: 1`
		`        min_size: 1`
		`        max_size: 1`
		`  ...`
			`Free`
			`SNod`
			`typ`
			`tos`
			`es.A`
			`mput`
			`Mana`
			`de`
			`...`
			`sca`
			`pro`
			`s:`
			`des`
			`ize:`
			`1`

参数	说明	示例：之前	示例：之后
			mir 1 max 1 ... *Free* *SNod* # New Nod typ tos es.A mput Self edNo ... sca

参数	说明	示例：之前	示例：之后

```
 pro
s:

 des
ize:
 1

 mir
 1

 max
 1

 ...
Free
SNod
 #
 New
 Nod
```

参数	说明	示例：之前	示例：之后
			typ tos es.A mput Mana de ... sca pro s: des ize: 1

参数	说明	示例：之前	示例：之后
			mir 1 max 1 ...

参数	说明	示例：之前	示例：之后

缩放属性	您可以更新EKSManagedNode 和 EKSSelfManagedNode TOSCA 节点的缩放属性。	`EKSNodeGroup01:`
		`  ...`
		`  scaling:`
		`  properties:`
		`  desired_size: 1`
		`  min_size: 1`
		`  max_size: 1`
			`EKSN`
			`oup0`
			`...`
			`sca`
			`pro`
			`s:`
			`des`
			`ize:`

参数	说明	示例：之前	示例：之后
			mir max

参数	说明	示例：之前	示例：之后

亚马逊 EBS CSI 插件属性	您可以在亚马逊 EKS 集群上启用或禁用 Amazon EBS CSI 插件。您也可以更改插件版本。	`EKSCluster:`
		`  capabilities:`
		`  ...`
		`  ebs_csi:`
		`  properties:`
		`  enabled: false`
			`EKSC`
			`r:`
			`cap`
			`ies:`
			`...`
			`ebs`
			`pro`
			`s:`
			`ena`
			`ver`
			`"v1`
			`e`

参数	说明	示例：之前	示例：之后
			ksbu "

参数	说明	示例：之前	示例：之后

根卷大小	您可以添加、移除或更新 EKSManaged Node 和 EKSSelf ManagedNode TOSCA 节点的根卷大小属性。	`Free5GCEKSNode01:`
		`  ...`
		`  capabilities:`
		`  compute:`
		`  properties:`
		`  root_volume_size: 50`
			`Free`
			`SNod`
			`...`
			`cap`
			`ies:`
			`com`
			`pro`
			`s:`

参数	说明	示例：之前	示例：之后
			roo me_s

参数	说明	示例：之前	示例：之后
VNF	您可以在 NSD VNFs 中引用，然后使用 VNFDeployment TOSCA 节点将它们部署到在 NSD 中创建的集群中。作为更新的一部分，您将能够向网络添加、更新和删除 VNFs 。	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 ....

参数	说明	示例：之前	示例：之后
			Sa mple elmD : typ tos es.A ploy VNFD ment rec nts: clu EKS r

参数	说明	示例：之前	示例：之后
			vnf - v leVN - v leVN

参数	说明	示例：之前	示例：之后

挂钩	要在创建网络函数之前和之后运行生命周期操作，请将pre_create 和post_create 挂钩添加到VNFDeployment 节点。
	
	在此示例中，PreCreateHook 挂钩将在实例化之前vnf3.SampleVNF3 运行，PostCreateHook 挂钩将在实例化之后vnf3.SampleVNF3 运行。
		vnfds:
		  - descriptor_id: "*43c012fa-2616-41a8-a833-0dfd4c5a049e* "
		    namespace: " *vnf1*"
		  - descriptor_id: "*64222f98-ecd6-4871-bf94-7354b53f3ee5* "
		    namespace: " *vnf2*"
		...
		SampleVNF1HelmDeploy:
		  type: tosca.nodes.AWS.Deployment.VNFDeployment
		  requirements:
		    cluster: EKSCluster
		    vnfs:
		      - *vnf1.SampleVNF1*
		      - *vnf2.SampleVNF2 // Removed during update*
			vnfd
			
			-
			des
			r_id
			"*43*
			2616
			-
			a833
			d4c5
			"
			
			nam
			:
			"*vn*
			
			-
			des
			r_id
			"*b7*
			839d
			-916
			a166
			"
			
			nam
			:
			"*vn*
			....
			S
			ampl
			Helm
			y:

参数	说明	示例：之前	示例：之后
			typ tos es.A ploy VNFD ment req nts: clu EKS r vnf - v leVN No cha to thi fur as the nam and uui rem

参数	说明	示例：之前	示例：之后
			the sam - v leVN New VNF as the nam , vnf was not pre y pre int s: Hoc pos te: eHoc

参数	说明	示例：之前	示例：之后
			pre e: *Hook*

参数	说明	示例：之前	示例：之后

挂钩	要在更新网络函数之前和之后运行生命周期操作，可以将pre_update 挂钩和post_update 挂钩添加到VNFDeployment 节点。
	
	在此示例中，PreUpdate Hook 将在更新之前运行vnf1.SampleVNF1 ，并在PostUpdateHook 更新到命名空间 vnf1 所指示uuid的vnf包之后vnf1.SampleVNF1 运行。
		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*
			vnfd
			
			-
			des
			r_id
			"*0e*
			bd87
			-
			b8a1
			4666
			"
			
			nam
			:
			"*vn*
			
			-
			des
			r_id
			"*64*
			ecd6
			-
			bf94
			4b53
			"
			
			nam
			:
			"*vn*
			...
			S
			ampl

参数	说明	示例：之前	示例：之后
			Helmy: typ tos es.A ploy VNFD ment req nts: clu EKS r vnf - v leVN A VNF upd as the uui cha for

参数	说明	示例：之前	示例：之后
			nam "vr - v leVN No cha to thi fur as nam and uui rem the sam int s: Hoo pre e: Hook

参数	说明	示例：之前	示例：之后
			pos te: *eHoc*

参数	说明	示例：之前	示例：之后
子网	您可以在网络中添加和删除子网。在删除子网之前，请确认该子网未被网络中的任何资源使用。	*Free5GCSubnet01* : #Deleted Subnet type: tosca.nodes.AWS.Networking.Subnet properties: type: "PUBLIC" availability_zone: { get_input: subnet_01_az } cidr_block: { get_input: subnet_01_cidr_block } requirements: route_table: Free5GCRouteTable vpc: Free5GCVPC	*Free* *bnet* #Ne Sub typ tos es.A twor Subn pro s: typ "PU ava ity_ { g : sub _az cid k: { g : sub _cid ck }

参数	说明	示例：之前	示例：之后
			rec nts: rou le: Fre uteT vpc Fre C

参数	说明	示例：之前	示例：之后
安全组	您可以在网络中添加和删除安全组。在删除安全组之前，请确认网络中的任何资源均未使用该安全组。	*Free5GCSecurityGroup01* : #Deleted Security Group type: tosca.nodes.AWS.Networking.SecurityGroup properties: description: "SecurityGroup for Free5GC cluster" name: "Free5GCSecurityGroup01" tags: - "Name=Free5GCEKSAdditionalSecurityGroup" requirements: vpc: Free5GCVPC *Free5GCSecurityGroupEgressRule01* : #Deleted Security Group Egress Node type: tosca.nodes.AWS.Networking.SecurityGroupEgressRule properties: ip_protocol: "tcp" from_port: 8000 to_port: 9000 description: "Egress Rule for free5GC cluster" cidr_ip : "172.10.10.1/24" requirements:	*Free* *curi* *up02* #Ne Sec Gro typ tos es.A twor Secu roup pro s: des on: "Se Grou for Fre clu nam "Fr ecur oup0 tag

参数	说明	示例：之前	示例：之后
		security_group: Free5GCSecurityGroup01 *Free5GCSecurityGroupIngressRule01* : #Deleted Security Group Ingress Node type: tosca.nodes.AWS.Networking.SecurityGroupIngressRule properties: ip_protocol: "tcp" from_port: 8000 to_port: 9000 description: "Ingress Rule for free5GC cluster" cidr_ip: "172.10.10.1/24" requirements: security_group: Free5GCSecurityGroup01	- "Na e5GC diti ecur oup" rec nts: vpc Fre C *Free curi upEg ule0* #Ne Sec Gro Egr Noc typ tos es.A twor Secu

参数	说明	示例：之前	示例：之后
			roup sRul pro s: ip_ ol: "to fro : 800 to_ 900 des on: "Eg Rul for fre clu cid "17 0.1/

参数	说明	示例：之前	示例：之后

```
 rec
nts:


 sec
grou
 Fre
curi
up02


Free
curi
upIr
Rule
 #Ne
 Sec
 Gro
 Ing
 Noc

 typ
 tos
es.A
twor
Secu
roup
ssRu

 pro
s:


 ip_
```

参数	说明	示例：之前	示例：之后
			ol: "to fro : 800 to_ 900 des on: "In Rul for fre clu cid "17 0.1/ rec nts: sec grou Fre

参数	说明	示例：之前	示例：之后
			curi up02

参数	说明	示例：之前	示例：之后
网络接口	您可以 ENIs 从网络中添加、修改和删除。	Free5GCENI01: #Modified ENI type: tosca.nodes.AWS.Networking.ENI properties: device_index: 2 requirements: subnet: *Free5GCENISubnet01* security_groups: - Free5GCSecurityGroup01 Free5GCENI02: #Modified ENI type: tosca.nodes.AWS.Networking.ENI properties: device_index: 3 source_dest_check: true requirements: subnet: Free5GCENISubnet01 *Free5GCENI04* : #Deleted ENI type: tosca.nodes.AWS.Networking.ENI properties: device_index: 4 source_dest_check: true requirements: subnet: Free5GCENISubnet01	Free I01: #Mo ENI typ tos es.A twor ENI pro s: dev dex: 2 req nts: sub *ISub* sec grou

参数	说明	示例：之前	示例：之后
			- Fre curi up01 Fre e5GC : #Mc ENI typ tos es.A twor ENI pro s: dev dex: 3 sou st_c tru rec nts: sub

参数	说明	示例：之前	示例：之后
			Fre ISub se grou - Fre curi up01 Free I03 #Ne ENI typ tos es.A twor ENI pro s: dev dex: 3 rec nts:

参数	说明	示例：之前	示例：之后
			sub Fre bnet sec grou - Fre curi up01

更新网络实例

Console

使用控制台更新网络实例

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择网络。
3. 选择网络实例。只有当网络实例的状态为Instantiated或时，您才能对其进行更新Updated。
4. 选择 “操作” 和 “更新”。

   将出现 “更新实例” 页面，其中包含当前基础设施中的网络详细信息和参数列表。
5. 选择新的网络套餐。

新网络包中的参数显示在 “更新的参数” 部分中。

6. （可选）在 “已更新的参数” 部分中更新参数值。有关您可以更新的参数值列表，请参阅[您可以更新的参数](#)。
7. 选择 “更新网络”。

   AWS TNB 验证请求并开始部署。此时将出现 “部署状态” 页面。
8. 使用刷新图标跟踪您的网络实例的部署状态。您也可以在 “部署任务” 部分启用自动刷新，以跟踪每个任务的进度。

   当部署状态更改为时`Completed`，网络实例即会更新。
9. - 如果验证失败，则网络实例将保持与您请求更新之前相同的状态——可以是`Instantiated`或`Updated`。
   - 如果更新失败，则会显示网络实例状态`Update failed`。为每项失败的任务选择链接以确定原因。
   - 如果更新成功，则会显示`Updated`网络实例状态。

AWS CLI

使用 CLI 更新网络实例

使用带有`UPDATE_NS`更新类型的update-sol-network-instance命令来更新网络实例。

```
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\"}}"
```

在 AWS TNB 中查看网络实例

了解如何查看网络实例。

Console

使用控制台查看网络实例

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择网络实例。

3. 使用搜索框找到网络实例。

AWS CLI

要查看网络实例，请使用 AWS CLI

1. 使用list-sol-network-instances命令列出您的网络实例。

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

2. 使用get-sol-network-instance命令查看有关特定网络实例的详细信息。

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

终止并从 AWS TNB 中删除网络实例

要删除网络实例，实例必须处于已终止状态。

Console

使用控制台终止和删除网络实例

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择网络。
3. 选择网络实例的 ID。
4. 选择终止。
5. 提示进行确认时，输入 ID，然后选择终止。
6. 刷新以跟踪网络实例的状态。
7. （可选）选择网络实例并选择删除。

AWS CLI

要终止和删除网络实例，请使用 AWS CLI

1. 使用terminate-sol-network-instance命令终止网络实例。

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

2. （可选）使用delete-sol-network-instance命令删除网络实例。

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

AWS TNB 的网络运营

网络操作是指对网络执行的任何操作，例如实例化或终止网络实例。

任务

- 查看 T AWS NB 网络运营情况
- 取消 T AWS NB 网络操作

查看 T AWS NB 网络运营情况

查看网络操作的详细信息，包括网络操作中涉及的任务和任务的状态。

Console

使用控制台查看网络操作

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择网络实例。
3. 使用搜索框找到网络实例。
4. 在 “部署” 选项卡上，选择网络操作。

AWS CLI

要查看网络操作，请使用 AWS CLI

1. 使用list-sol-network-operations命令列出所有网络操作。

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

2. 使用get-sol-network-operation命令查看有关网络操作的详细信息。

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

取消 T AWS NB 网络操作

了解如何取消网络操作。

Console

使用控制台取消网络操作

1. 打开 AWS TNB 控制台，网址为https://console.aws.amazon.com/tnb/。
2. 在导航窗格中，选择网络。
3. 选择网络的 ID 以打开其详细信息页面。
4. 在部署选项卡上，选择“网络操作”。
5. 选择取消操作。

AWS CLI

要取消网络操作，请使用 AWS CLI

使用cancel-sol-network-operation命令取消网络操作。

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

TNB 的 TOSCA 参考文献 AWS

云应用程序拓扑和编排规范 (TOSCA) 是一种声明性语法， CSPs 用于描述基于云的 Web 服务的拓扑、其组件、关系以及管理这些服务的流程。 CSPs 在 TOSCA 模板中描述连接点、连接点之间的逻辑链接以及诸如关联性和安全性之类的策略。 CSPs 然后将模板上传到 AWS TNB，该模板汇总了跨 AWS 可用区建立正常运行的 5G 网络所需的资源。

内容

VNFD 模板

定义虚拟网络功能描述文件（VNFD）模板。

语法

```
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
```

拓扑模板

`node_templates`

TOSCA 节点。 AWS 可能的节点如下：

- AWS.VNF

- AWS.Artifacts.Helm

AWS.VNF

定义 AWS 虚拟网络功能 (VNF) 节点。

语法

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

属性

`descriptor_id`

描述文件的 UUID。

必需：是

类型：字符串

模式：`[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}`

`descriptor_version`

VNFD 的版本。

必需：是

类型：字符串

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

`descriptor_name`

描述文件的名称。

必需：是

类型：字符串

`provider`

VNFD 的制作方。

必需：是

类型：字符串

要求

`helm`

定义容器构件的 Helm 目录。这是对 AWS.Artifacts.Helm 的引用。

必需：是

类型：字符串

示例

```
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

定义头 AWS 盔节点。

语法

```
tosca.nodes.AWS.Artifacts.Helm:
  properties:
    implementation: String
```

属性

`implementation`

CSAR 包中包含 Helm 图表的本地目录。

必需：是

类型：字符串

示例

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

网络服务描述符模板

定义网络服务描述文件（NSD）模板。

语法

```
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
```

使用已定义的参数

当您想要动态传递参数（例如 VPC 节点的 CIDR 块）时，可以使用 { get_input: *input-parameter-name* } 语法在 NSD 模板中定义参数。然后即可在同一 NSD 模板中重复使用该参数。

以下示例演示了如何定义和使用参数：

```
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 导入

`descriptor_id`

描述文件的 UUID。

是否必需：是

类型：字符串

模式：`[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}`

`namespace`

唯一名称。

是否必需：是

类型：字符串

拓扑模板

`node_templates`

可能的 TOSCA AWS 节点有：

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

AWS.NS

定义 AWS 网络服务 (NS) 节点。

语法

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

Properties

`descriptor_id`

描述文件的 UUID。

是否必需：是

类型：字符串

模式：`[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}`

`descriptor_version`

NSD 的版本。

是否必需：是

类型：字符串

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

`descriptor_name`

描述文件的名称。

是否必需：是

类型：字符串

示例

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

```
    descriptor_version: "1.0.0"
    descriptor_name: "Test NS Template"
```

AWS.Compute.EKS

提供集群的名称、所需的 Kubernetes 版本以及允许 Kubernetes 控制平面管理 NF 所需资源的角色。AWS Multus 容器网络接口（CNI）插件已启用。您可以连接多个网络接口，并将高级网络配置应用于 Kubernetes-based 网络功能。您还可以为集群指定集群端点访问权限和子网。

语法

```
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
```

功能

`multus`

可选。定义 Multus 容器网络接口（CNI）使用的属性。

如果您包含 `multus`，则指定 `enabled` 和 `multus_role` 属性。

`enabled`

指示是否启用默认 Multus 功能。

是否必需：是

类型：布尔值

`multus_role`

Multus 网络接口管理角色。

是否必需：是

类型：字符串

`ebs_csi`

定义安装在 Amazon EKS 集群中的 Amazon EBS 容器存储接口（CSI）驱动程序的属性。

启用此插件即可在 Local Zones 或 AWS 区域 L AWS ocal Zones 上 AWS Outposts使用 Amazon EKS 自我管理节点。有关更多信息，请参阅《Amazon EKS 用户指南》中的 Amazon Elastic Block Store CSI 驱动程序。

`enabled`

指示是否安装默认 Amazon EBS CSI 驱动程序。

必需：否

类型：布尔值

`version`

Amazon EBS CSI 驱动程序附加组件的版本。该版本必须与DescribeAddonVersions操作返回的版本之一相匹配。有关更多信息，请参阅 Amazon EKS API 参考DescribeAddonVersions中的

必需：否

类型：字符串

Properties

`version`

集群的 Kubernetes 版本。 AWS Telco Network Builder 支持 Kubernetes 版本 1.27 到 1.34。

是否必需：是

类型：字符串

可能的值：1.27 | 1.28 | 1.29 | 1.30 | 1.31 | 1.32 | 1.33 | 1.33 | 1.34

`access`

集群端点访问。

是否必需：是

类型：字符串

可能的值：PRIVATE | PUBLIC | ALL

`cluster_role`

集群管理角色。

是否必需：是

类型：字符串

`tags`

要附加到资源的标签。

必需：否

类型：列表

`ip_family`

表示集群中服务和容器组（pod）地址的 IP 系列。

允许的值：IPv4、IPv6

默认值：IPv4

必需：否

类型：字符串

要求

`subnets`

一个AWS。 Networking.Subnet节点。

是否必需：是

类型：列表

示例

```
SampleEKS:
  type: tosca.nodes.AWS.Compute.EKS
  properties:
    version: "1.26"
    access: "ALL"
    cluster_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleRole"
    ip_family: "IPv6"
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
  capabilities:
    multus:
      properties:
        enabled: true
        multus_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/MultusRole"
    ebs_csi:
      properties:
        enabled: true
        version: "v1.16.0-eksbuild.1"
  requirements:
    subnets:
    - SampleSubnet01
    - SampleSubnet02
```

AWS.Compute.EKS.AuthRole

AuthRole 允许您向 Amazon EKS 集群添加 IAM 角色，aws-authConfigMap以便用户可以使用 IAM 角色访问 Amazon EKS 集群。

语法

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

```
      groups: List
  requirements:
    clusters: List
```

Properties

`role_mappings`

定义需要添加到 Amazon EKS 集群 `aws-auth ConfigMap` 的 IAM 角色的映射列表。

`arn`

IAM 角色的 ARN。

是否必需：是

类型：字符串

`groups`

要分配给 `arn` 中定义的角色的 Kubernetes 组。

必需：否

类型：列表

要求

`clusters`

一个AWS。 Compute.EKS节点。

是否必需：是

类型：列表

示例

```
EKSAuthMapRoles:
    type: tosca.nodes.AWS.Compute.EKS.AuthRole
    properties:
```

```
        role_mappings:
        - arn: arn:aws:iam::${AWS::TNB::AccountId}:role/TNBHookRole1
          groups:
          - system:nodes
          - system:bootstrappers
        - arn: arn:aws:iam::${AWS::TNB::AccountId}:role/TNBHookRole2
          groups:
          - system:nodes
          - system:bootstrappers
    requirements:
         clusters:
         - Free5GCEKS1
         - Free5GCEKS2
```

AWS.Compute.EKSManagedNode

AWS TNB 支持 EKS 托管节点组，以自动为亚马逊 EKS Kubernetes 集群配置节点（Amazon EC2 实例）和进行生命周期管理。要创建 EKS 节点组，请执行以下操作：

- 通过提供 AMI 的 ID 或 AMI 类型，为您的集群工作节点选择亚马逊系统映像 (AMI)。
- 提供用于 SSH 访问的 Amazon EC2 密钥对以及节点组的扩展属性。
- 确保您的节点组与 Amazon EKS 集群相关联。
- 为工作节点提供子网。
- 或者，将安全组、节点标签和置放群组附加到您的节点组。

语法

```
tosca.nodes.AWS.Compute.EKSManagedNode:
  capabilities:
    compute:
      properties:
        ami_type: String
        ami_id: String
        instance_types: List
        key_pair: String
        root_volume_encryption: Boolean
        root_volume_encryption_key_arn: String
        root_volume_size: Integer
    scaling:
      properties:
```

```
        desired_size: Integer
        min_size: Integer
        max_size: Integer
  properties:
    node_role: String
    tags: List
    kubernetes_version: String
  requirements:
    cluster: String
    subnets: List
    network_interfaces: List
    security_groups: List
    placement_group: String
    user_data: String
    labels: List
```

功能

##

定义 Amazon EKS 托管节点组计算参数的属性，例如 Amazon EC2 实例类型和 Amazon EC2 实例 AMI。

`ami_type`

亚马逊 EKS-supported AMI 类型。

是否必需：是

类型：字符串

可能的值：`AL2_x86_64`| `AL2_x86_64_GPU` | `AL2_ARM_64` | `AL2023_x86_64` | `AL2023_ARM_64` | `AL2023_x86_64_NVIDIA` | `AL2023_x86_64_NEURON` | | `CUSTOM` | `BOTTLEROCKET_ARM_64` | `BOTTLEROCKET_x86_64` | `BOTTLEROCKET_ARM_64_NVIDIA` | `BOTTLEROCKET_x86_64_NVIDIA`

`ami_id`

AMI 的 ID。

必需：否

类型：字符串

> **Note**
>
> 如果模板中同时指定了ami_type和ami_id，则 AWS TNB 将仅使用该ami_id值来创建EKSManagedNode。

`instance_types`

实例大小。

是否必需：是

类型：列表

`key_pair`

用于启用 SSH 访问的 EC2 密钥对。

是否必需：是

类型：字符串

`root_volume_encryption`

为亚马逊 EBS 根卷启用亚马逊 EBS 加密。如果未提供此属性，则 AWS TNB 会默认加密 Amazon EBS 根卷。

必需：否

默认：True

类型：布尔值

`root_volume_encryption_key_arn`

密钥的 ARN。 AWS KMS AWS TNB 支持常规密钥 ARN、多区域密钥 ARN 和别名 ARN。

必需：否

类型：字符串

> **Note**
>
> - 如果root_volume_encryption为假，则不包括root_volume_encryption_key_arn。

- AWS TNB 支持 Amazon EBS-backed AMI 的根卷加密。
- 如果 AMI 的根卷已加密，则必须包括`root_volume_encryption_key_arn`以便 AWS TNB 重新加密根卷。
- 如果 AMI 的根卷未加密， AWS TNB `root_volume_encryption_key_arn` 将使用加密根卷。

  如果不包括`root_volume_encryption_key_arn`， AWS TNB 将使用提供的默认密钥 AWS Key Management Service 对根卷进行加密。
- AWS TNB 不会解密加密的 AMI。

`root_volume_size`

Amazon Elastic Block Store 根卷的大小 GiBs。

必需：否

默认值：20

类型：整数

可能的值：1 到 16,384

##

定义 Amazon EKS 托管节点组扩缩参数的属性，例如节点组中所需的 Amazon EC2 实例数量以及 Amazon EC2 实例的最少和最多数量。

`desired_size`

此中的实例数量 NodeGroup。

是否必需：是

类型：整数

`min_size`

此中的最小实例数 NodeGroup。

是否必需：是

类型：整数

`max_size`

此中的最大实例数 NodeGroup。

是否必需：是

类型：整数

Properties

`node_role`

附加到 Amazon EC2 实例的 IAM 角色的 ARN。

是否必需：是

类型：字符串

`tags`

要附加到资源的标签。

必需：否

类型：列表

`kubernetes_version`

托管节点组的 Kubernetes 版本。 AWS TNB 支持 Kubernetes 版本 1.27 到 1.34。请考虑以下事项：

- 指定`kubernetes_version`或`ami_id`。请勿指定此两者。
- `kubernetes_version`必须小于或等于 AWS。 Compute.EKSManagedNode 版本。
- 两者之间可能有 3 个版本的差异 AWS。 Compute.EKSManagedNode 版本和`kubernetes_version`。
- 如果未指定`kubernetes_version`或`ami_id`， AWS TNB 将使用该`AWS.Compute.EKSManagedNode`版本的最新 AMI 来创建 EKSManagedNode

必需：否

类型：字符串

可能的值：1.27 | 1.28 | 1.29 | 1.30 | 1.31 | 1.32 | 1.33 | 1.33 | 1.34

要求

`cluster`

一个AWS。 Compute.EKS节点。

是否必需：是

类型：字符串

`subnets`

一个AWS。 Networking.Subnet节点。

是否必需：是

类型：列表

`network_interfaces`

一个AWS。 Networking.ENI节点。确保将网络接口和子网设置为相同的可用区，否则实例化将失败。

设置后`network_interfaces`，如果您在节点中包含该属性，则 AWS TNB 将从该`multus_role`属性获取与 ENI 相关的权限。`multus` AWS.Compute.EKS否则， AWS TNB 将从 node_role 属性中获取与 ENI 相关的权限。

必需：否

类型：列表

`security_groups`

一个AWS。 Networking.SecurityGroup节点。

必需：否

类型：列表

`placement_group`

一个 tosca.nodes。AWS。 Compute.PlacementGroup节点。

必需：否

类型：字符串

`user_data`

一个 tosca.nodes。AWS。 Compute.UserData节点引用。用户数据脚本传递到由托管式节点组启动的 Amazon EC2 实例。将运行自定义用户数据所需的权限添加到传递给节点组的 node_role。

必需：否

类型：字符串

`labels`

节点标签列表。节点标签必须有名称和值。使用以下标准创建标签：

- 名称和值必须用分隔=。
- 名称和值的长度最多可为 63 个字符。
- 标签可以包含字母 (A-Z、a-z、)、数字 (0-9) 和以下字符：`[-, _, ., *, ?]`
- 名称和值必须以字母数字?、或*字符开头和结尾。

例如，`myLabelName1=*NodeLabelValue1`

必需：否

类型：列表

示例

```
SampleEKSManagedNode:
  type: tosca.nodes.AWS.Compute.EKSManagedNode
  capabilities:
    compute:
      properties:
        ami_type: "AL2_x86_64"
        instance_types:
          - "t3.xlarge"
        key_pair: "SampleKeyPair"
        root_volume_encryption: true
        root_volume_encryption_key_arn: "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        root_volume_size: 1500
    scaling:
      properties:
        desired_size: 1
        min_size: 1
```

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

AWS.Compute.EKSSelfManagedNode

AWS TNB 支持 Amazon EKS 自我管理节点，以自动为亚马逊 EKS Kubernetes 集群配置节点（亚马逊 EC2 实例）和进行生命周期管理。要创建 Amazon EKS 节点组，请执行以下操作：

- 通过提供 AMI 的 ID，为您的集群工作节点选择亚马逊系统映像 (AMI)。
- 提供用于 SSH 访问的 Amazon EC2 密钥对。
- 确保您的节点组与 Amazon EKS 集群相关联。
- 提供实例类型以及所需大小、最小和最大大小。
- 为工作节点提供子网。
- 或者，将安全组、节点标签和置放群组附加到您的节点组。

语法

```
tosca.nodes.AWS.Compute.EKSSelfManagedNode:
  capabilities:
```

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

功能

##

定义 Amazon EKS 自管理节点计算参数的属性，例如 Amazon EC2 实例类型和 Amazon EC2 实例 AMI。

`ami_id`

用于启动实例的 AMI ID。 AWS TNB 支持利用 imdsv2 的实例。有关更多信息，请参阅 IMDS 版本。

> **Note**
>
> 您可以更新的 AMI ID `EKSSelfManagedNode`。AMI 的 Amazon EKS 版本必须与亚马逊 EKS 集群版本相同或最多低两个版本。例如，如果亚马逊 EKS 集群版本为 1.31，则亚马逊 EKS AMI 版本必须为 1.31、1.30 或 1.29。

是否必需：是

类型：字符串

`instance_type`

实例大小。

是否必需：是

类型：字符串

`key_pair`

启用 SSH 访问的 Amazon EC2 密钥对。

是否必需：是

类型：字符串

`root_volume_encryption`

为亚马逊 EBS 根卷启用亚马逊 EBS 加密。如果未提供此属性，则 AWS TNB 会默认加密 Amazon EBS 根卷。

必需：否

默认：True

类型：布尔值

`root_volume_encryption_key_arn`

密钥的 ARN。 AWS KMS AWS TNB 支持常规密钥 ARN、多区域密钥 ARN 和别名 ARN。

必需：否

类型：字符串

> **Note**
>
> - 如果`root_volume_encryption`为假，则不包括`root_volume_encryption_key_arn`。
> - AWS TNB 支持 Amazon EBS-backed AMI 的根卷加密。

- 如果 AMI 的根卷已加密，则必须包括`root_volume_encryption_key_arn`以便 AWS TNB 重新加密根卷。
- 如果 AMI 的根卷未加密， AWS TNB `root_volume_encryption_key_arn` 将使用加密根卷。

  如果不包括`root_volume_encryption_key_arn`， AWS TNB AWS Managed Services 将使用加密根卷。
- AWS TNB 不会解密加密的 AMI。

`root_volume_size`

Amazon Elastic Block Store 根卷的大小 GiBs。

必需：否

默认值：20

类型：整数

可能的值：1 到 16,384

##

定义 Amazon EKS 自管理节点扩缩参数的属性，例如节点组中所需的 Amazon EC2 实例数量以及 Amazon EC2 实例的最少和最多数量。

`desired_size`

此中的实例数量 NodeGroup。

是否必需：是

类型：整数

`min_size`

此中的最小实例数 NodeGroup。

是否必需：是

类型：整数

`max_size`

此中的最大实例数 NodeGroup。

是否必需：是

类型：整数

Properties

`node_role`

附加到 Amazon EC2 实例的 IAM 角色的 ARN。

是否必需：是

类型：字符串

`tags`

要附加到资源的标签。标签将传播到资源创建的实例。

必需：否

类型：列表

要求

`cluster`

一个AWS。 Compute.EKS节点。

是否必需：是

类型：字符串

`subnets`

一个AWS。 Networking.Subnet节点。

是否必需：是

类型：列表

`network_interfaces`

一个AWS。 Networking.ENI节点。确保将网络接口和子网设置为相同的可用区，否则实例化将失败。

设置后`network_interfaces`，如果您在节点中包含该属性，则 AWS TNB 将从该`multus_role`属性获取与 ENI 相关的权限。`multus` AWS.Compute.EKS否则， AWS TNB 将从 node_role 属性中获取与 ENI 相关的权限。

必需：否

类型：列表

`security_groups`

一个AWS。 Networking.SecurityGroup节点。

必需：否

类型：列表

`placement_group`

一个 tosca.nodes。AWS。 Compute.PlacementGroup节点。

必需：否

类型：字符串

`user_data`

一个 tosca.nodes。AWS。 Compute.UserData节点引用。用户数据脚本传递到由自行管理的节点组启动的 Amazon EC2 实例。将执行自定义用户数据所需的权限添加到传递给节点组的 node_role。

必需：否

类型：字符串

`labels`

节点标签列表。节点标签必须有名称和值。使用以下标准创建标签：

- 名称和值必须用分隔=。
- 名称和值的长度最多可为 63 个字符。
- 标签可以包含字母 (A-Z、a-z、)、数字 (0-9) 和以下字符：`[-, _, ., *, ?]`

- 名称和值必须以字母数字?、或*字符开头和结尾。

例如，myLabelName1=*NodeLabelValue1

必需：否

类型：列表

示例

```
SampleEKSSelfManagedNode:
  type: tosca.nodes.AWS.Compute.EKSSelfManagedNode
  capabilities:
    compute:
      properties:
        ami_id: "ami-123123EXAMPLE"
        instance_type: "c5.large"
        key_pair: "SampleKeyPair"
        root_volume_encryption: true
        root_volume_encryption_key_arn: "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        root_volume_size: 1500
    scaling:
      properties:
        desired_size: 1
        min_size: 1
        max_size: 1
  properties:
    node_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleNodeRole"
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
  requirements:
    cluster: SampleEKSCluster
    subnets:
      - SampleSubnet
    network_interfaces:
      - SampleNetworkInterface01
      - SampleNetworkInterface02
    security_groups:
      - SampleSecurityGroup01
      - SampleSecurityGroup02
    placement_group: SamplePlacementGroup
```

```
    user_data: CustomUserData
    labels:
      - "sampleLabelName001=sampleLabelValue001"
      - "sampleLabelName002=sampleLabelValue002"
```

AWS.Compute.PlacementGroup

PlacementGroup 节点支持不同的策略来放置 Amazon EC2 实例。

在您启动新的 Amazon EC2 实例时，Amazon EC2 服务会尝试以某种方式放置实例，以便将所有实例分布在基础硬件上以最大限度减少相关的故障。您可以使用置放群组影响如何放置一组相互依赖的实例，从而满足您的工作负载需求。

语法

```
tosca.nodes.AWS.Compute.PlacementGroup
  properties:
    strategy: String
    partition_count: Integer
    tags: List
```

Properties

`strategy`

用于放置 Amazon EC2 实例的策略。

是否必需：是

类型：字符串

可能的值：CLUSTER | PARTITION | SPREAD_HOST | SPREAD_RACK

- CLUSTER – 将一个可用区内靠近的实例打包在一起。通过使用该策略，工作负载可以实现所需的低延迟网络性能，以满足高性能计算（HPC）应用程序通常使用的紧密耦合的节点到节点通信的要求。
- PARTITION – 将实例分布在不同的逻辑分区上，以便一个分区中的实例组不会与不同分区中的实例组共享相同的基础硬件。该策略通常为大型分布式和重复的工作负载所使用，例如，Hadoop、Cassandra 和 Kafka。
- SPREAD_RACK – 将一小组实例严格放置在不同的基础硬件上，以减少相关的故障。

- SPREAD_HOST – 只能与 Outpost 置放群组结合使用。将一小组实例严格放置在不同的基础硬件上以减少相关的故障。

`partition_count`

分区的数量。

必需：仅当 `strategy` 设置为 `PARTITION` 时才必需。

类型：整数

可能的值：1 | 2 |3 | 4 | 5 | 6 | 7

`tags`

可以附加到置放群组资源的标签。

必需：否

类型：列表

示例

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

AWS.Compute.UserData

AWS TNB 支持通过网络服务描述符 (NSD) 中的 UserData节点启动带有自定义用户数据的 Amazon EC2 实例。有关自定义用户数据的更多信息，请参阅 Amazon EC2 用户指南中的用户数据和 shell 脚本。

在网络实例化期间，AWS TNB 通过用户数据脚本向集群提供 Amazon EC2 实例注册。当还提供自定义用户数据时，AWS TNB 会合并两个脚本，并将它们作为多重脚本传递给 Amazon EC2。自定义用户数据脚本在 Amazon EKS 注册脚本之前运行。

要在用户数据脚本中使用自定义变量，请在左大括号 { 后面添加感叹号 !。例如，要在脚本中使用 `MyVariable`，请输入：`{!MyVariable}`

> **Note**
>
> - AWS TNB 支持大小不超过 7 KB 的用户数据脚本。
> - 由 AWS 于 TNB CloudFormation 用于处理和呈现multimime用户数据脚本，因此请确保脚本遵守所有 CloudFormation 规则。

语法

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

Properties

`implementation`

用户数据脚本定义的相对路径。格式必须是：`./scripts/script_name.sh`

是否必需：是

类型：字符串

`content_type`

用户数据脚本的内容类型。

是否必需：是

类型：字符串

可能的值：`x-shellscript`

示例

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

```
    implementation: "./scripts/customUserData.sh"
```

AWS.Networking.SecurityGroup

AWS TNB 支持安全组来自动配置 Amazon EC2 安全组，您可以将其附加到 Amazon EKS Kubernetes 集群节点组。

语法

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

Properties

`description`

安全组的描述。最多可以使用 255 个字符来描述该组。只能包含字母（A-Z 和 a-z）、数字 (0-9)、空格和以下特殊字符：._-:/() #、@ [] +=&; {}! $*

是否必需：是

类型：字符串

`name`

安全组的名称。该名称最多可使用 255 个字符。只能包含字母（A-Z 和 a-z）、数字 (0-9)、空格和以下特殊字符：._-:/() #、@ [] +=&; {}! $*

是否必需：是

类型：字符串

`tags`

可以附加到安全组资源的标签。

必需：否

类型：列表

要求

vpc

一个AWS。 Networking.VPC节点。

是否必需：是

类型：字符串

示例

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

AWS.Networking.SecurityGroupEgressRule

AWS TNB 支持安全组出站规则，以自动配置可附加到的 Amazon EC2 安全组出站规则。AWS Networking.SecurityGroup。请注意，您必须提供 cidr_ip/destination_security_group / destination_prefix_list 作为出口流量的目的地。

语法

```
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
```

Properties

`cidr_ip`

CIDR 格式的 IPv4 地址范围。您必须指定允许出口流量的 CIDR 范围。

必需：否

类型：字符串

`cidr_ipv6`

适用于出口流量的采用 CIDR 格式的 IPv6 地址范围。您必须指定目标安全组 (`destination_security_group` 或 `destination_prefix_list`) 或 CIDR 范围 (`cidr_ip` 或 `cidr_ipv6`)。

必需：否

类型：字符串

`description`

出口（出站）安全组规则的描述。最多可以使用 255 个字符来描述该规则。

必需：否

类型：字符串

`destination_prefix_list`

现有 Amazon VPC 托管式前缀列表的前缀列表 ID。这是与安全组关联的节点组实例的目标。有关托管式前缀列表的更多信息，请参阅《Amazon VPC 用户指南》中的托管式前缀列表。

必需：否

类型：字符串

`from_port`

如果协议是 TCP 或 UDP，则这是端口范围的起始端口。如果协议是 ICMP 或 ICMPv6，则这是类型编号。值为 -1 表示所有 ICMP/ICMPv6 类型。如果指定所有 ICMP/ICMPv6 类型，则必须指定所有 ICMP/ICMPv6 代码。

必需：否

类型：整数

`ip_protocol`

IP 协议名称（tcp、udp、icmp、icmpv6）或协议编号。使用 -1 可指定所有协议。当授权安全组规则时，指定 -1 或除 tcp、udp、icmp 或 icmpv6 以外的协议编号将允许所有端口上的流量，无论您指定的端口范围如何。对于 tcp、udp 和 icmp，您必须指定端口范围。对于 icmpv6，端口范围是可选的；如果您省略端口范围，则将允许所有类型和代码的流量。

是否必需：是

类型：字符串

`to_port`

如果协议是 TCP 或 UDP，则这是端口范围的终止端口。如果协议是 ICMP 或 ICMPv6，则这是代码。值为 -1 表示所有 ICMP/ICMPv6 代码。如果指定所有 ICMP/ICMPv6 类型，则必须指定所有 ICMP/ICMPv6 代码。

必需：否

类型：整数

要求

`security_group`

要添加此规则的安全组的 ID。

是否必需：是

类型：字符串

`destination_security_group`

允许出口流量进入的目标安全组的 ID 或 TOSCA 参考。

必需：否

类型：字符串

示例

```
SampleSecurityGroupEgressRule:
```

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

AWS.Networking.SecurityGroupIngressRule

AWS TNB 支持安全组入口规则，以自动配置可附加到的 Amazon EC2 安全组入口规则。 AWS Networking.SecurityGroup。请注意，您必须提供 cidr_ip/source_security_group /source_prefix_list 作为入口流量的来源。

语法

```
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
```

Properties

`cidr_ip`

CIDR 格式的 IPv4 地址范围。您必须指定允许入口流量的 CIDR 范围。

必需：否

类型：字符串

`cidr_ipv6`

适用于入口流量的采用 CIDR 格式的 IPv6 地址范围。您必须指定源安全组 (`source_security_group` 或 `source_prefix_list`) 或 CIDR 范围 (`cidr_ip` 或 `cidr_ipv6`)。

必需：否

类型：字符串

`description`

入口（入站）安全组规则的描述。最多可以使用 255 个字符来描述该规则。

必需：否

类型：字符串

`source_prefix_list`

现有 Amazon VPC 托管式前缀列表的前缀列表 ID。将允许与安全组关联的节点组实例从此来源接收流量。有关托管式前缀列表的更多信息，请参阅《Amazon VPC 用户指南》中的托管式前缀列表。

必需：否

类型：字符串

`from_port`

如果协议是 TCP 或 UDP，则这是端口范围的起始端口。如果协议是 ICMP 或 ICMPv6，则这是类型编号。值为 -1 表示所有 ICMP/ICMPv6 类型。如果指定所有 ICMP/ICMPv6 类型，则必须指定所有 ICMP/ICMPv6 代码。

必需：否

类型：整数

`ip_protocol`

IP 协议名称（tcp、udp、icmp、icmpv6）或协议编号。使用 -1 可指定所有协议。当授权安全组规则时，指定 -1 或除 tcp、udp、icmp 或 icmpv6 以外的协议编号将允许所有端口上的流量，无论您指定的端口范围如何。对于 tcp、udp 和 icmp，您必须指定端口范围。对于 icmpv6，端口范围是可选的；如果您省略端口范围，则将允许所有类型和代码的流量。

是否必需：是

类型：字符串

`to_port`

如果协议是 TCP 或 UDP，则这是端口范围的终止端口。如果协议是 ICMP 或 ICMPv6，则这是代码。值为 -1 表示所有 ICMP/ICMPv6 代码。如果指定所有 ICMP/ICMPv6 类型，则必须指定所有 ICMP/ICMPv6 代码。

必需：否

类型：整数

要求

`security_group`

要添加此规则的安全组的 ID。

是否必需：是

类型：字符串

`source_security_group`

源安全组的 ID 或 TOSCA 参考，将允许来自该安全组的入口流量。

必需：否

类型：字符串

示例

```
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

您可以将以下 AWS 资源导入 AWS TNB：

- VPC
- 子网
- 路由表
- 互联网网关
- 安全组

语法

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

Properties

`resource_type`

导入到 AWS TNB 的资源类型。

必需：否

类型：列表

`resource_id`

导入到 AWS TNB 的资源 ID。

必需：否

类型：列表

示例

```
SampleImportedVPC:
  type: tosca.nodes.AWS.Resource.Import
  properties:
```

```
    resource_type: "tosca.nodes.AWS.Networking.VPC"
    resource_id: "vpc-123456"
```

AWS.Networking.ENI

网络接口是 VPC 中代表虚拟网卡的逻辑联网组件。可以根据子网，自动或手动为网络接口分配 IP 地址。在子网中部署 Amazon EC2 实例后，您可以将网络接口连接到该实例，或者将网络接口与该 Amazon EC2 实例分离，然后重新连接到该子网中的另一个 Amazon EC2 实例。设备索引标识连接顺序中的位置。

语法

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

Properties

`device_index`

设备索引必须大于零。

是否必需：是

类型：整数

`source_dest_check`

表示网络接口是否执行 source/destination检查。值为 `true` 表示已启用检查，`false` 表示已禁用检查。

允许的值：真、假

默认：True

必需：否

类型：布尔值

`tags`

要附加到资源的标签。

必需：否

类型：列表

要求

`subnet`

一个AWS。 Networking.Subnet节点。

是否必需：是

类型：字符串

`security_groups`

一个AWS。 Networking.SecurityGroup节点。

必需：否

类型：字符串

示例

```
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

生命周期挂钩使您能够将自己的脚本作为基础设施和网络实例化的一部分来运行。

语法

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

功能

execution

运行挂钩脚本的挂钩执行引擎的属性。

type

挂钩执行引擎类型。

必需：否

类型：字符串

可能的值：CODE_BUILD

要求

definition

一个AWS。 HookDefinition.Bash节点。

是否必需：是

类型：字符串

vpc

一个AWS。 Networking.VPC节点。

是否必需：是

类型：字符串

示例

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

AWS.Networking.InternetGateway

定义 AWS Internet Gateway 节点。

语法

```
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
```

功能

##

定义 VPC 内路由连接的属性。必须包括 `dest_cidr` 或 `ipv6_dest_cidr` 属性。

`dest_cidr`

用于目标匹配的 IPv4 CIDR 块。此属性用于在 `RouteTable` 中创建路由，其值用作 `DestinationCidrBlock`。

必需：如果包含 ipv6_dest_cidr 属性，则为“否”。

类型：字符串

ipv6_dest_cidr

用于目标匹配的 IPv6 CIDR 块。

必需：如果包含 dest_cidr 属性，则为“否”。

类型：字符串

Properties

tags

要附加到资源的标签。

必需：否

类型：列表

egress_only

一个 IPv6-specific 财产。表示互联网网关是否仅用于出口通信。如果 egress_only 为 true，则必须定义 ipv6_dest_cidr 属性。

必需：否

类型：布尔值

要求

vpc

一个AWS。 Networking.VPC节点。

是否必需：是

类型：字符串

route_table

一个AWS。 Networking.RouteTable节点。

是否必需：是

类型：字符串

示例

```
Free5GCIGW:
  type: tosca.nodes.AWS.Networking.InternetGateway
  properties:
    egress_only: false
  capabilities:
    routing:
      properties:
        dest_cidr: "0.0.0.0/0"
        ipv6_dest_cidr: "::/0"
  requirements:
    route_table: Free5GCRouteTable
    vpc: Free5GCVPC
Free5GCEGW:
  type: tosca.nodes.AWS.Networking.InternetGateway
  properties:
    egress_only: true
  capabilities:
    routing:
      properties:
        ipv6_dest_cidr: "::/0"
  requirements:
    route_table: Free5GCPrivateRouteTable
    vpc: Free5GCVPC
```

AWS.Networking.RouteTable

路由表包含一组被称为路由的规则，决定了来自 VPC 中的子网或网关的网络流量将指向何处。您必须将路由表与 VPC 关联。

语法

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

```
    vpc: String
```

Properties

`tags`

要附加到资源的标签。

必需：否

类型：列表

要求

`vpc`

一个AWS。 Networking.VPC节点。

是否必需：是

类型：字符串

示例

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

AWS.Networking.Subnet

子网是 VPC 内的 IP 地址范围，必须完全位于一个可用区内。您必须为子网指定 VPC、CIDR 块、可用区和路由表。您还必须定义子网是私有还是公有。

语法

```
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
```

Properties

`type`

指示在此子网中启动的实例是否接收公有 IPv4 地址。

是否必需：是

类型：字符串

可能的值：PUBLIC | PRIVATE

`availability_zone`

子网的可用区。此字段支持 AWS 区域内的 AWS 可用区，例如us-west-2（美国西部（俄勒冈））。例如，它还支持可用区内的 L AWS ocal Zones us-west-2-lax-1a。

是否必需：是

类型：字符串

`cidr_block`

子网的 CIDR 块。

必需：否

类型：字符串

`ipv6_cidr_block`

用于创建 IPv6 子网的 CIDR 块。如果包含此属性，请不要包含 ipv6_cidr_block_suffix。

必需：否

类型：字符串

`ipv6_cidr_block_suffix`

通过 Amazon VPC 创建的子网的 IPv6 CIDR 块的 2 位十六进制后缀。采用以下格式：*2-digit hexadecimal*`::/`*subnetMask*

如果包含此属性，请不要包含 `ipv6_cidr_block`。

必需：否

类型：字符串

`outpost_arn`

将在其中创建子网 AWS Outposts 的 ARN。如果您希望在 AWS Outposts上启动 Amazon EKS 自管理节点，请将此属性添加到 NSD 模板中。有关更多信息，请参阅《Amazon EKS 用户指南》中的 AWS Outposts上的 Amazon EKS。

如果将此属性添加到 NSD 模板中，则必须将 `availability_zone` 属性的值设置为 AWS Outposts的可用区。

必需：否

类型：字符串

`tags`

要附加到资源的标签。

必需：否

类型：列表

要求

`vpc`

一个AWS。 Networking.VPC节点。

是否必需：是

类型：字符串

`route_table`

一个AWS。 Networking.RouteTable节点。

是否必需：是

类型：字符串

示例

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

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

AWS.Deployment.VNFDeployment

NF 部署是通过提供基础设施和与之关联的应用程序来建模的。cluster 属性指定托管 NF 的 EKS 集群。vnfs 属性为您的部署指定网络功能。您还可以提供 pre_create 和 post_create 类型的可选生命周期挂钩操作，来运行特定于您的部署的指令，例如调用库存管理系统 API。

语法

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

要求

`deployment`

一个AWS。 Deployment.VNFDeployment节点。

必需：否

类型：字符串

`cluster`

一个AWS。 Compute.EKS节点。

是否必需：是

类型：字符串

`vnfs`

一个 AWS.VNF 节点。

是否必需：是

类型：字符串

接口

挂钩

定义运行生命周期挂钩的阶段。

`pre_create`

一个AWS。 HookExecution节点。此挂钩在 VNFDeployment 节点部署之前运行。

必需：否

类型：字符串

`post_create`

一个AWS。 HookExecution节点。此挂钩在 VNFDeployment 节点部署之后运行。

必需：否

类型：字符串

示例

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

AWS.Networking.VPC

您必须为虚拟私有云（VPC）指定 CIDR 块。

语法

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

Properties

`cidr_block`

VPC 的 IIPv4 网络范围，采用 CIDR 表示法。

是否必需：是

类型：字符串

`ipv6_cidr_block`

用于创建 VPC 的 IPv6 CIDR 块。

允许的值：`AMAZON_PROVIDED`

必需：否

类型：字符串

`dns_support`

指明在 VPC 内启动的实例是否可获得 DNS 主机名称。

必需：否

类型：布尔值

默认值：`false`

`tags`

要附加到资源的标签。

必需：否

类型：列表

示例

```
SampleVPC:
  type: tosca.nodes.AWS.Networking.VPC
  properties:
    cidr_block: "10.100.0.0/16"
    ipv6_cidr_block: "AMAZON_PROVIDED"
    dns_support: true
```

```
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
```

AWS.Networking.NATGateway

您可以通过子网定义公有或私有 NAT 网关节点。对于公共网关，如果您不提供弹性 IP 分配 ID， AWS TNB 将为您的账户分配一个弹性 IP 并将其关联到网关。

语法

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

Properties

`subnet`

的AWS。 Networking.Subnet节点引用。

是否必需：是

类型：字符串

`internet_gateway`

的AWS。 Networking.InternetGateway节点引用。

是否必需：是

类型：字符串

Properties

`type`

指示网关是公有还是私有的。

允许的值：PUBLIC、PRIVATE

是否必需：是

类型：字符串

`eip_allocation_id`

表示弹性 IP 地址分配的 ID。

必需：否

类型：字符串

`tags`

要附加到资源的标签。

必需：否

类型：列表

示例

```
Free5GCNatGateway01:
  type: tosca.nodes.AWS.Networking.NATGateway
    requirements:
       subnet: Free5GCSubnet01
       internet_gateway: Free5GCIGW
    properties:
       type: PUBLIC
       eip_allocation_id: eipalloc-12345
```

AWS.Networking.Route

您可以定义一个路由节点，该节点将目标路由与 NAT 网关关联为目标资源，并将该路由添加到关联的路由表中。

语法

```
tosca.nodes.AWS.Networking.Route:
  properties:
```

```
    dest_cidr_blocks: List
  requirements:
    nat_gateway: String
    route_table: String
```

Properties

`dest_cidr_blocks`

到目标资源的目标 IPv4 路由列表。

是否必需：是

类型：列表

成员类型：字符串

要求

`nat_gateway`

的AWS。 Networking.NATGateway节点引用。

是否必需：是

类型：字符串

`route_table`

的AWS。 Networking.RouteTable节点引用。

是否必需：是

类型：字符串

示例

```
Free5GCRoute:
  type: tosca.nodes.AWS.Networking.Route
  properties:
    dest_cidr_blocks:
      - 0.0.0.0/0
```

```
      - 10.0.0.0/28
  requirements:
    nat_gateway: Free5GCNatGateway01
    route_table: Free5GCRouteTable
```

AWS.Store.SSMParameters

您可以通过 AWS TNB 创建 SSM 参数。您创建的 SSM 参数是在 SSM 中创建的，并以 AWS TNB 网络实例 ID 为前缀。当使用同一 NSD 模板实例化和升级多个实例时，这可以防止参数值被覆盖。

语法

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

Properties

参数

`name`

ssm 属性的名称。采用以下格式：`^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`

每个参数的名称必须少于 256 个字符。

是否必需：是

类型：字符串

`value`

ssm 属性的值。使用以下格式之一：

- 对于没有引用的值：`^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`
- 对于静态引用：`^\$\{[a-zA-Z0-9]+\.(properties|capabilities|requirements)(\.([a-zA-Z0-9\-_]+))+\}$`
- 对于动态引用：`^\$\{[a-zA-Z0-9]+\.(name|id|arn)\}$`

每个参数的值必须小于 4 KB。

是否必需：是

类型：字符串

`tags`

可以附加到 SSM 属性的标签。

必需：否

类型：列表

示例

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

通用节点

为 NSD 和 VNFD 定义节点。

- AWS。 HookDefinition.Bash

AWS.HookDefinition.Bash

定义一个 AWS HookDefinition in bash。

语法

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

属性

`implementation`

挂钩定义的相对路径。格式必须是：`./hooks/`*`script_name`*`.sh`

必需：是

类型：字符串

`environment_variables`

挂钩 bash 脚本的环境变量。使用以下格式：**envName**=**envValue**使用以下正则表达式模式：

- 对于没有引用的值：`^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`
- 对于静态引用：`^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=\$\{[a-zA-Z0-9]+\.(properties|capabilities|requirements)(\.([a-zA-Z0-9\-_]+))+\}$`
- 对于动态引用：`^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=\$\{[a-zA-Z0-9]+\.(name|id|arn)\}$`

确保 **envName**=**envValue** 的值符合以下标准：

- 不使用空格。
- **envName** 以字母（A-Z 或 a-z）或数字（0-9）开头。
- 环境变量名称的开头不使用以下 AWS TNB 保留关键字（不区分大小写）：
  - CODEBUILD
  - TNB
  - HOME
  - AWS
- 可以在 **envName** 和 **envValue** 中使用任意数量的字母（A-Z 或 a-z）、数字（0-9）和特殊字符 - 及 _。

- 每个环境变量（每个 **envName** =**envValue**）必须少于 128 个字符。

  示例：A123-45xYz=Example_789

  必需：否

  类型：列表

execution_role

  挂钩执行的角色。

  必需：是

  类型：字符串

示例

```
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"
```

中的安全性 AWS TNB

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在云中运行 AWS 服务的基础架构 AWS Cloud。 AWS 还为您提供可以安全使用的服务。 Third-party 作为AWS 合规计划合规计划合规计划合的一部分，审计师定期测试和验证我们安全的有效性。要了解适用于 AWS Telco Network Builder 的合规性计划，请参阅AWS 按合规计划划分的范围内AWS 服务按合规计划。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用 AWS TNB 时如何应用分担责任模型。以下主题向您展示如何配置 AWS TNB 以满足您的安全和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护 AWS TNB资源。

内容

中的数据保护 AWS TNB

责任共担模型 AWS 分适用于 AWS Telco Network Builder 中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题解答AWS条款。 有关欧洲数据保护的信息，请参阅通用数据保护条例（GDPR）中心。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》https://aws.amazon.com/compliance/fips/。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您 AWS 服务 使用控制台、API 或 AWS SDK 与 AWS TNB 或其他人合作时。 AWS CLI在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

数据处理

当您关闭 AWS 帐户时， AWS TNB会将您的数据标记为删除，并将其从任何用途中删除。如果您在 90 天内重新激活 AWS 帐户， AWS TNB 会恢复您的数据。120 天后， AWS TNB 将永久删除您的数据。 AWS TNB 还会终止您的网络并删除您的函数包和网络包。

静态加密

AWS TNB 始终对存储在服务中的所有静态数据进行加密，而无需进行任何其他配置。这种加密是通过自动进行的 AWS Key Management Service。

传输中加密

AWS TNB 使用传输层安全 (TLS) 1.2 保护传输中的所有数据。

您负责加密您的模拟代理与其客户端之间的数据。

Inter-network 交通隐私

AWS TNB 计算资源位于所有客户共享的虚拟私有云 (VPC) 中。所有内部 AWS TNB 流量都留在 AWS 网络内，不会通过互联网。您的模拟代理与其客户端之间的连接通过互联网路由。

的身份和访问管理 AWS TNB

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（有权限）使用 AWS TNB 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

内容

受众

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅问题排查 AWS 电信网络生成器身份和访问权限）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅操作方法 AWS TNB 与 IAM 合作）
- IAM 管理员：编写用于管理访问权限的策略（请参阅Identity-based 的策略示例 AWS 电信网络生成器）

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center （例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 用户指南》中的如何登录您的 AWS 账户。

对于编程访问，AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 根用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service ，或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限，建议使用。 AWS IAM Identity Center有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

IAM 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon EC2 上运行的应用程序非常有用。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

Identity-based 政策

Identity-based 策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

Identity-based 策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

Resource-based 政策

Resource-based 策略是您附加到资源的 JSON 策略文档。示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。您必须在基于资源的策略中指定主体。

Resource-based 策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《 IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略（SCP）– 指定 AWS Organizations中组织或组织单元的最大权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 资源控制策略（RCP）– 设置对账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略（RCP）。

- 会话策略 – 在为角色或联合用户创建临时会话时，作为参数传递的高级策略。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

操作方法 AWS TNB 与 IAM 合作

在使用 IAM 管理对 AWS TNB 的访问权限之前，请先了解有哪些 IAM 功能可用于 AWS TNB。

您可以搭配使用的 IAM 功能 AWS 电信网络生成器

IAM 功能	AWS TNB 支持
Identity-based 政策	是
Resource-based 政策	否
策略操作	是
策略资源	是
策略条件键	是
ACL	否
ABAC（策略中的标签）	是
临时凭证	是
主体权限	是
服务角色	否
Service-linked 角色	否

要全面了解 AWS TNB 和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS 服务。

Identity-based 的政策 AWS TNB

支持基于身份的策略：是

Identity-based 策略是您可以附加到身份（例如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

Identity-based 的策略示例 AWS TNB

要查看 AWS TNB 基于身份的策略的示例，请参阅。Identity-based 的策略示例 AWS 电信网络生成器

Resource-based 内在的政策 AWS TNB

支持基于资源的策略：否

Resource-based 策略是您附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

的政策行动 AWS TNB

支持策略操作：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。在策略中包含操作以授予执行关联操作的权限。

要查看 AWS TNB 操作列表，请参阅《服务授权参考》中的 T AWS elco Network Builder 定义的操作。

AWS TNB 中的策略操作在操作前使用以下前缀：

```
tnb
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

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

您也可以使用通配符（*）指定多个操作。例如，要指定以单词 `List` 开头的所有操作，包括以下操作：

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

要查看 AWS TNB 基于身份的策略的示例，请参阅。Identity-based 的策略示例 AWS 电信网络生成器

的政策资源 AWS TNB

支持策略资源：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

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

要查看 AWS TNB 资源类型及其 ARN 的列表，请参阅《服务授权参考》中的 AWS Telco Network Builder 定义的资源。要了解您可以使用哪些操作来指定每种资源的 ARN，请参阅 AWS Telco Network Builder 定义的操作。

要查看 AWS TNB 基于身份的策略的示例，请参阅。Identity-based 的策略示例 AWS 电信网络生成器

的策略条件键 AWS TNB

支持特定于服务的策略条件键：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

Condition 元素根据定义的条件指定语句何时执行。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

要查看 AWS TNB 条件密钥列表，请参阅《服务授权参考》中的 T AWS elco Network Builder 的条件密钥。要了解可以使用条件键的操作和资源，请参阅 AWS Telco Network Builder 定义的操作。

要查看 AWS TNB 基于身份的策略示例，请参阅。Identity-based 的策略示例 AWS 电信网络生成器

输入的 ACL AWS TNB

支持 ACL：否

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，但它们不使用 JSON 策略文档格式。

ABAC with AWS TNB

支持 ABAC（策略中的标签）：是

Attribute-based 访问控制 (ABAC) 是一种授权策略，它根据称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 AWS 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

要基于标签控制访问，您需要使用 aws:ResourceTag/*key-name*aws:RequestTag/*key-name* 或 aws:TagKeys 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时凭证与配合使用 AWS TNB

支持临时凭证：是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的临时安全凭证和使用 IAM 的。AWS 服务

Cross-service 的委托人权限 AWS TNB

支持转发访问会话（FAS）：是

转发访问会话 (FAS) 使用调用主体的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

的服务角色 AWS TNB

支持服务角色：否

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

Service-linked 的角色 AWS TNB

支持服务相关角色：否

服务相关角色是一种与服务相关联的 AWS 服务服务角色。该服务可以代替您执行操作。 Service-linked 角色出现在您的， AWS 账户 并且归服务所有。IAM 管理员可以查看但不能编辑服务关联角色的权限。

Identity-based 的策略示例 AWS 电信网络生成器

默认情况下，用户和角色无权创建或修改 AWS TNB 资源。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略（控制台）。

有关 AWS TNB 定义的操作和资源类型（包括每种资源类型的 ARN 格式）的详细信息，请参阅《服务授权参考》中的 T AWS elco Network Builder 的操作、资源和条件密钥。

内容

- 策略最佳实践
- 使用 AWS TNB 控制台
- 服务角色策略示例
- 允许用户查看他们自己的权限

策略最佳实践

Identity-based 策略决定是否有人可以在您的账户中创建、访问或删除 AWS TNB 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的AWS 托管策略或工作职能的AWS 托管策略。
- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性：IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 AWS TNB 控制台

要访问 AWS Telco Network Builder 控制台，您必须拥有一组最低权限。这些权限必须允许您列出和查看有关您 AWS 账户的 AWS TNB 资源的详细信息。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

服务角色策略示例

作为管理员，您拥有并管理 AWS TNB 根据环境和服务模板定义创建的资源。您必须将 IAM 服务角色附加到您的账户，以允许 AWS TNB 为您的网络生命周期管理创建资源。

IAM 服务角色允许 AWS TNB 代表您调用资源，以实例化和管理您的网络。如果您指定服务角色，AWS TNB 将使用该角色的凭证。

您可以使用 IAM 服务创建服务角色及其权限策略。有关创建服务角色的更多信息，请参阅 IAM 用户指南中的创建角色以向 AWS 服务委派权限。

AWS TNB 服务角色

作为平台团队的成员，您可以作为管理员创建 AWS TNB 服务角色并将其提供给 AWS TNB。此角色允许 AWS TNB 调用其他服务，例如 Amazon Elastic Kubernetes Ser CloudFormation vice Amazon Kubernetes Service，为您的网络配置所需的基础设施，并按照 NSD 中的定义配置网络功能。

我们建议您为 AWS TNB 服务角色使用以下 IAM 角色和信任策略。在缩小此策略的权限范围时，请记住， AWS TNB 可能会失败，因为 “访问被拒绝” 错误，无法访问您的策略。

以下代码显示了 T AWS NB 服务角色策略：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:GetCallerIdentity"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "AssumeRole"
        },
        {
            "Action": [
                "tnb:*"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TNBPolicy"
```

```
        },
        {
            "Action": [
                "iam:AddRoleToInstanceProfile",
                "iam:CreateInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile",
                "iam:TagInstanceProfile",
                "iam:UntagInstanceProfile"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "IAMPolicy"
        },
        {
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "eks.amazonaws.com",
                        "eks-nodegroup.amazonaws.com"
                    ]
                }
            },
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TNBAccessSLRPermissions"
        },
        {
            "Action": [
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateOrUpdateTags",
                "autoscaling:DeleteAutoScalingGroup",
                "autoscaling:DeleteTags",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeAutoScalingInstances",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:DescribeTags",
                "autoscaling:UpdateAutoScalingGroup",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
```

```
                "ec2:CreateLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteLaunchTemplateVersions",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:DeleteLaunchTemplate",
                "ec2:DeleteSecurityGroup",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeTags",
                "ec2:GetLaunchTemplateData",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:RunInstances",
                "ec2:AssociateRouteTable",
                "ec2:AttachInternetGateway",
                "ec2:CreateInternetGateway",
                "ec2:CreateNetworkInterface",
                "ec2:CreateRoute",
                "ec2:CreateRouteTable",
                "ec2:CreateSubnet",
                "ec2:CreateTags",
                "ec2:CreateVpc",
                "ec2:DeleteInternetGateway",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteRoute",
                "ec2:DeleteRouteTable",
                "ec2:DeleteSubnet",
                "ec2:DeleteTags",
                "ec2:DeleteVpc",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeInstances",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroupRules",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DetachInternetGateway",
                "ec2:DisassociateRouteTable",
                "ec2:ModifySecurityGroupRules",
                "ec2:ModifySubnetAttribute",
                "ec2:ModifyVpcAttribute",
```

```
                "ec2:AllocateAddress",
                "ec2:AssignIpv6Addresses",
                "ec2:AssociateAddress",
                "ec2:AssociateNatGatewayAddress",
                "ec2:AssociateVpcCidrBlock",
                "ec2:CreateEgressOnlyInternetGateway",
                "ec2:CreateNatGateway",
                "ec2:DeleteEgressOnlyInternetGateway",
                "ec2:DeleteNatGateway",
                "ec2:DescribeAddresses",
                "ec2:DescribeEgressOnlyInternetGateways",
                "ec2:DescribeNatGateways",
                "ec2:DisassociateAddress",
                "ec2:DisassociateNatGatewayAddress",
                "ec2:DisassociateVpcCidrBlock",
                "ec2:ReleaseAddress",
                "ec2:UnassignIpv6Addresses",
                "ec2:DescribeImages",
                "eks:CreateCluster",
                "eks:ListClusters",
                "eks:RegisterCluster",
                "eks:TagResource",
                "eks:DescribeAddonVersions",
                "events:DescribeRule",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TNBAccessComputePerms"
        },
        {
            "Resource": "*",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Condition": {
                "StringEquals": {
                   "iam:PassedToService": [
                       "ec2.amazonaws.com",
                       "eks.amazonaws.com",
                       "eks-nodegroup.amazonaws.com",
                       "events.amazonaws.com",
```

```
                    "autoscaling.amazonaws.com",
                    "codebuild.amazonaws.com"
                ]
            }
        }
    },
    {
        "Action": [
            "codebuild:BatchDeleteBuilds",
            "codebuild:BatchGetBuilds",
            "codebuild:CreateProject",
            "codebuild:DeleteProject",
            "codebuild:ListBuildsForProject",
            "codebuild:StartBuild",
            "codebuild:StopBuild",
            "events:DeleteRule",
            "events:PutRule",
            "events:PutTargets",
            "events:RemoveTargets",
            "s3:CreateBucket",
            "s3:GetBucketAcl",
            "s3:GetObject",
            "eks:DescribeNodegroup",
            "eks:DeleteNodegroup",
            "eks:AssociateIdentityProviderConfig",
            "eks:CreateNodegroup",
            "eks:DeleteCluster",
            "eks:DeregisterCluster",
            "eks:UpdateAddon",
            "eks:UpdateClusterVersion",
            "eks:UpdateNodegroupConfig",
            "eks:UpdateNodegroupVersion",
            "eks:DescribeUpdate",
            "eks:UntagResource",
            "eks:DescribeCluster",
            "eks:ListNodegroups",
            "eks:CreateAddon",
            "eks:DeleteAddon",
            "eks:DescribeAddon",
            "eks:DescribeAddonVersions",
            "s3:PutObject",
            "cloudformation:CreateStack",
            "cloudformation:DeleteStack",
            "cloudformation:DescribeStackResources",
```

```
                "cloudformation:DescribeStacks",
                "cloudformation:ListStackResources",
                "cloudformation:UpdateStack",
                "cloudformation:UpdateTerminationProtection",
                "ssm:PutParameter",
                "ssm:GetParameters",
                "ssm:GetParameter",
                "ssm:DeleteParameter",
                "ssm:AddTagsToResource",
                "ssm:ListTagsForResource",
                "ssm:RemoveTagsFromResource"
            ],
            "Resource": [
                "arn:aws:events:*:*:rule/tnb*",
                "arn:aws:codebuild:*:*:project/tnb*",
                "arn:aws:logs:*:*:log-group:/aws/tnb*",
                "arn:aws:s3:::tnb*",
                "arn:aws:eks:*:*:addon/tnb*/*/*",
                "arn:aws:eks:*:*:cluster/tnb*",
                "arn:aws:eks:*:*:nodegroup/tnb*/tnb*/*",
                "arn:aws:cloudformation:*:*:stack/tnb*",
                "arn:aws:ssm:*:*:parameter/tnb/*"
            ],
            "Effect": "Allow",
            "Sid": "TNBAccessInfraResourcePerms"
        },
        {
            "Sid": "CFNTemplatePerms",
            "Effect": "Allow",
            "Action": [
              "cloudformation:GetTemplateSummary"
            ],
            "Resource": "*"
         },
        {
            "Sid": "ImageAMISSMPerms",
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
                "arn:aws:ssm:*::parameter/aws/service/eks/optimized-ami/*",
                "arn:aws:ssm:*::parameter/aws/service/bottlerocket/*"
            ]
```

```
        },
        {
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TaggingPolicy"
        },
        {
            "Action": [
                "outposts:GetOutpost"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "OutpostPolicy"
        }
    ]
}
```

以下代码显示了 AWS TNB 服务信任策略：

JSON

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

```
      "Effect": "Allow",
      "Principal": {
        "Service": "codebuild.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "tnb.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS 亚马逊 EKS 集群的 TNB 服务角色

当您在 NSD 中创建 Amazon EKS 资源时，您需要提供 `cluster_role` 属性来指定将使用哪个角色来创建您的 Amazon EKS 集群。

以下示例显示了一个为 Amazon EKS 集群策略创建 AWS TNB 服务角色的 AWS CloudFormation 模板。

```
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"
```

有关使用 AWS CloudFormation 模板的 IAM 角色的更多信息，请参阅AWS CloudFormation 用户指南中的以下部分：

- AWS::IAM::Role
- 选择堆栈模板

AWS 亚马逊 EKS 节点组的 TNB 服务角色

当您在 NSD 中创建 Amazon EKS 节点组资源时，您需要提供 `node_role` 属性来指定将使用哪个角色来创建您的 Amazon EKS 节点组。

以下示例显示了为 Amazon EKS 节点组策略创建 AWS TNB 服务角色的 CloudFormation 模板。

```
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/*"
```

有关使用 AWS CloudFormation 模板的 IAM 角色的更多信息，请参阅AWS CloudFormation 用户指南中的以下部分：

- AWS::IAM::Role
- 选择堆栈模板

AWS Multus 的 TNB 服务角色

当您在 NSD 中创建 Amazon EKS 资源并希望将 Multus 作为部署模板的一部分进行管理时，必须提供 `multus_role` 属性以指定将使用哪个角色来管理 Multus。

以下示例显示了为 Multus 策略创建 AWS TNB 服务角色的 CloudFormation 模板。

```
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: "*"
```

有关使用 AWS CloudFormation 模板的 IAM 角色的更多信息，请参阅AWS CloudFormation 用户指南中的以下部分：

- AWS::IAM::Role
- 选择堆栈模板

AWS 生命周期挂钩策略的 TNB 服务角色

当您的 NSD 或网络功能包使用生命周期挂钩时，您需要一个服务角色来允许您创建执行生命周期挂钩的环境。

> **Note**
>
> 您的生命周期挂钩策略应基于您的生命周期挂钩尝试执行的操作。

以下示例显示了一个为生命周期挂钩策略创建 AWS TNB 服务角色的 CloudFormation 模板。

```
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"
```

有关使用 AWS CloudFormation 模板的 IAM 角色的更多信息，请参阅AWS CloudFormation 用户指南中的以下部分：

- AWS::IAM::Role
- 选择堆栈模板

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

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

问题排查 AWS 电信网络生成器身份和访问权限

使用以下信息来帮助您诊断和修复在使用 AWS TNB 和 IAM 时可能遇到的常见问题。

问题

- 我无权在以下位置执行操作 AWS TNB
- 我无权执行 iam：PassRole
- 我想允许我以外的人进入 AWS 账户 访问我的 AWS TNB 资源

我无权在以下位置执行操作 AWS TNB

如果您收到错误提示，指明您无权执行某个操作，则必须更新策略以允许执行该操作。

当 mateojackson IAM 用户尝试使用控制台查看有关虚构 *my-example-widget* 资源的详细信息，但不拥有虚构 tnb:*GetWidget* 权限时，会发生以下示例错误。

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

在此情况下，Mateo 的策略必须更新以允许其使用 tnb:*GetWidget* 操作访问 *my-example-widget* 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam：PassRole

如果您收到错误消息，提示您无权执行iam:PassRole操作，则必须更新您的策略以允许您将角色传递给 AWS TNB。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 marymajor 的 IAM 用户尝试使用控制台在 AWS TNB 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

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

在这种情况下，必须更新 Mary 的策略以允许她执行 iam:PassRole 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人进入 AWS 账户 访问我的 AWS TNB 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表（ACL）的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 AWS TNB 是否支持这些功能，请参阅操作方法 AWS TNB 与 IAM 合作。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

的合规性验证 AWS TNB

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

韧性在 AWS TNB

AWS 全球基础设施是围绕 AWS 区域 可用区构建的。 AWS 区域 提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络连接。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错能力和可扩展性。

有关 AWS 区域 和可用区的更多信息，请参阅AWS 全球基础设施。

AWS TNB 在您选择的 AWS 区域的虚拟私有云 (VPC) 中的 EKS 集群上运行您的网络服务。

中的基础设施安全 AWS TNB

作为一项托管服务， AWS Telco Network Builder 受到 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用通过网络访问 AWS TNB。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE（短暂的）或 ECDHE（椭圆曲线短暂的 Diffie-Hellman）。 Diffie-Hellman大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

下面是一些责任共担示例：

- AWS 负责保护支持 AWS TNB 的组件，包括：
  - 计算实例（也称为 Worker）
  - 内部数据库
  - 内部组件之间的网络通信
  - T AWS NB 应用程序编程接口 (API)
  - AWS 软件开发套件 (SDK)
- 您有责任保护自己对 AWS 资源和工作负载组件的访问权限，包括（但不限于）：
  - IAM 用户、组、角色和策略
  - 用于存储 TNB 数据的 S3 存储桶 AWS
  - 其他 AWS 服务 和您用来支持通过 AWS TNB 配置的网络服务的资源
  - 您的应用程序代码
  - 您通过 AWS TNB 配置的网络服务与其客户端之间的连接

> **Important**
>
> 您负责实施灾难恢复计划，该计划可以有效地恢复通过 AWS TNB 配置的网络服务。

网络连接安全模型

您通过 AWS TNB 配置的网络服务在位于您所选 AWS 区域的虚拟私有云 (VPC) 内的计算实例上运行。VPC 是 AWS 云中的虚拟网络，它按工作负载或组织实体隔离基础架构。VPC 内计算实例之间的通信保持在 AWS 网络内，不会通过互联网传输。一些内部服务通信会通过互联网进行并经过加密。通过 AWS TNB 为在同一区域运行的所有客户提供的网络服务共享同一 VPC。通过 AWS TNB 为不同客户配置的网络服务在同一 VPC 中使用不同的计算实例。

您的网络服务客户端与 AWS TNB 中的网络服务之间的通信通过互联网传输。 AWS TNB 不管理这些连接。您负责保护您的客户端连接。

您通过 AWS 管理控制台、 AWS Command Line Interface (AWS CLI) 和 AWS SDK 与 AWS TNB 的连接已加密。

IMDS 版本

AWS TNB 支持利用实例元数据服务版本 2 (imdsv2) (一种面向会话的方法) 的实例。IMDSv2 包含比 IMDSv1 更高的安全性。有关更多信息，请参阅 Add defense in depth against open firewalls, reverse proxies, and SSRF vulnerabilities with enhancements to the Amazon EC2 Instance Metadata Service。

启动实例时，必须使用 IMDSv2。有关 IMDSv2 的更多信息，请参阅《Amazon EC2 用户指南》中的使用 IMDSv2。

监控 AWS TNB

监控是维护 AWS TNB 和其他 AWS 解决方案的可靠性、可用性和性能的重要组成部分。 AWS AWS CloudTrail 提供监视 AWS TNB，在出现问题时进行报告，并在适当时自动采取行动。

CloudTrail 用于捕获有关拨打的呼叫的详细信息 AWS APIs。您可以将这些调用作为日志文件存储在 Amazon S3 中。您可以使用这些 CloudTrail 日志来确定拨打了哪个电话、呼叫来自哪个源 IP 地址、谁拨打了电话以及何时拨打了呼叫等信息。

CloudTrail 日志包含有关 AWS TNB 的 API 操作调用的信息。它们还包含来自亚马逊 EC2 和亚马逊 EBS 等服务调用 API 操作的信息。

使用记录 AWS 电信网络生成器 API 调用 AWS CloudTrail

AWS Telco Network Builder 与AWS CloudTrail一项服务集成，该服务提供用户、角色或角色所采取的操作的 AWS 服务记录。 CloudTrail 将 AWS TNB 的所有 API 调用捕获为事件。捕获的调用包括来自 AWS TNB 控制台的调用和对 T AWS NB API 操作的代码调用。使用收集的信息 CloudTrail，您可以确定向 AWS TNB 发出的请求、发出请求的 IP 地址、发出请求的时间以及其他详细信息。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是用户凭证发出的。
- 请求是否代表 IAM Identity Center 用户发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

CloudTrail 在您创建账户 AWS 账户 时在您的账户中处于活动状态，并且您可以自动访问 CloudTrail 活动历史记录。 CloudTrail 事件历史记录提供了过去 90 天中记录的管理事件的可查看、可搜索、可下载且不可变的记录。 AWS 区域有关更多信息，请参阅《AWS CloudTrail 用户指南》中的 “使用 CloudTrail 事件历史记录”。查看活动历史记录不 CloudTrail收取任何费用。

要持续记录 AWS 账户 过去 90 天内的事件，请创建跟踪或 CloudTrailLake 事件数据存储。

CloudTrail 步道

跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。使用创建的所有跟踪 AWS 管理控制台 都是多区域的。您可以通过使用 AWS CLI创建单区域或多区域跟踪。建议创建多区域跟踪，因

为您可以捕获账户 AWS 区域 中的所有活动。如果您创建单区域跟踪，则只能查看跟踪的 AWS 区域中记录的事件。有关跟踪的更多信息，请参阅《AWS CloudTrail 用户指南》中的为您的 AWS 账户创建跟踪和为组织创建跟踪。

通过创建跟踪，您可以免费将正在进行的管理事件的一份副本传送到您的 Amazon S3 存储桶，但会收取 Amazon S3 存储费用。 CloudTrail 有关 CloudTrail 定价的更多信息，请参阅AWS CloudTrail 定价。有关 Amazon S3 定价的信息，请参阅 Amazon S3 定价。

CloudTrail 湖泊事件数据存储

CloudTrail L@@ ak e 允许您对事件运行基于 SQL 的查询。 CloudTrail Lake 将基于行的 JSON 格式的现有事件转换为 Apache ORC 格式。ORC 是一种针对快速检索数据进行优化的列式存储格式。事件将被聚合到事件数据存储中，它是基于您通过应用高级事件选择器选择的条件的不可变的事件集合。应用于事件数据存储的选择器用于控制哪些事件持续存在并可供您查询。有关 CloudTrail Lake 的更多信息，请参阅AWS CloudTrail 用户指南中的使用 AWS CloudTrail Lake。

CloudTrail 湖泊事件数据存储和查询会产生费用。创建事件数据存储时，您可以选择要用于事件数据存储的定价选项。定价选项决定了摄取和存储事件的成本，以及事件数据存储的默认和最长保留期。有关 CloudTrail 定价的更多信息，请参阅AWS CloudTrail 定价。

AWS TNB 事件示例

事件代表来自任何来源的单个请求，包括有关所请求的 API 操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此事件不会按任何特定顺序出现。

以下示例显示了一个演示该`CreateSolFunctionPackage`操作 CloudTrail 的事件。

```
{
    "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"
}
```


有关 CloudTrail 录音内容的信息，请参阅《AWS CloudTrail 用户指南》中的CloudTrail录制内容。

AWS TNB 部署任务

了解部署任务以有效监控部署并更快地采取行动。

下表列出了 T AWS NB 部署任务：

在 2024 年 3 月 7 日之前开始的部署的任务名称	在 2024 年 3 月 7 日及之后开始的部署的任务名称	任务描述
AppInstallation	ClusterPluginInstall	在 Amazon EKS 集群上安装 Multus 插件。
AppUpdate	名字没有变化	更新已安装在网络实例中的网络功能。
-	ClusterPluginUninstall	在 Amazon EKS 集群上卸载插件。
ClusterStorageClassConfiguration	名字没有变化	在 Amazon EKS 集群上配置存储类别（CSI 驱动程序）。
FunctionDeletion	名字没有变化	从 AWS TNB 资源中删除网络函数。
FunctionInstantiation	FunctionInstall	使用 HELM 部署网络功能。
FunctionUninstallation	FunctionUninstall	从 Amazon EKS 集群中卸载网络功能。
HookExecution	名字没有变化	按照 NSD 中的定义执行生命周期挂钩。
InfrastructureCancellation	名字没有变化	取消网络服务。
InfrastructureInstantation	名字没有变化	代表用户置备 AWS 资源。
InfrastructureTermination	名字没有变化	取消配置通过 AWS TNB 调用的 AWS 资源。
-	InfrastructureUpdate	更新代表用户置备的 AWS 资源。
InventoryDeregistration	名字没有变化	从 AWS TNB 注销 AWS 资源。
-	InventoryRegistration	在 AWS TNB 中注册 AWS 资源。
KubernetesClusterConfiguration	ClusterConfiguration	按照 NSD 中的定义，配置 Kubernetes 集群并向 Amazon EKS AuthMap 添加其他 IAM 角色。

在 2024 年 3 月 7 日之前开始的部署的任务名称	在 2024 年 3 月 7 日及之后开始的部署的任务名称	任务描述
NetworkServiceFinalization	名字没有变化	最终确定网络服务并提供成功或失败状态更新。
NetworkServiceInstantiation	名字没有变化	初始化网络服务。
SelfManagedNodesConfiguration	名字没有变化	使用 Amazon EKS 和 Kubernetes 控制面板启动自管理节点。
-	ValidateNetworkServiceUpdate	在更新网络实例之前运行验证。

AWS TNB 的服务配额

服务配额，也称为限制，是您的 AWS 账户的最大服务资源或操作数量。有关更多信息，请参阅 Amazon Web Services 一般参考 中的 AWS 服务限额。

以下是 AWS TNB 的服务配额。

Name	默认值	可调整	描述
正在进行的并发网络服务操作	每个受支持的区域：40 个	是	一个区域内正在进行的并发网络服务操作的最大数量。
程序包函数	每个受支持的区域：200 个	是	一个区域中函数程序包的最大数量。
网络程序包	每个受支持的区域：40 个	是	网络程序包在一个区域内的最大数量。
网络服务实例	每个受支持的区域：800 个	是	一个区域中网络服务实例的最大数量。

AWS TNB 用户指南的文档历史记录

下表描述了 AWS TNB 的文档版本。

变更	说明	日期
Amazon EKS 节点组网络配置的更新	添加和删除子网和安全组。在网络 ENIs 中添加、修改和删除。有关更多信息，请参阅可以更新的参数。	2025 年 9 月 10 日
在现有集群中添加和删除 Amazon EKS 节点组	AWS TNB 现在支持添加新的节点组和从 Amazon EKS 集群中移除现有节点组。有关更多信息，请参阅可以更新的参数。	2025 年 6 月 4 日
根卷大小	您可以通过 AWS.Compute 中的`root_volume_size` 字段指定 Amazon EKS 工作节点的底层 Amazon EBS 根卷的大小。 EKSManaged节点和 AWS.Compute。 EKSSelfManagedNodeTOSCA 节点。	2025 年 5 月 19 日
脚本中的参考资源	您可以引用 AWS TNB 创建的资源，以便在生命周期挂钩脚本和用户数据脚本中对其进行配置。	2025 年 5 月 2 日
Amazon EKS 节点和托管节点组现在支持 Kubernetes 1.32 版。	AWS TNB 支持适用于.compute.eks 和 .Compute 的 Kubernetes 版本 1.32。AWSAWS EKSManaged节点。	2025 年 4 月 24 日
亚马逊 EKS 节点和托管节点组不再支持 Kubernetes 1.24 版	AWS TNB 不再支持 .compute.eks 和 .Compute 的 Kubernete	2025 年 4 月 17 日

	s 版本 1.24。AWSAWS EKSManaged节点。	
AL2023 AMI 支持 Amazon EKS 托管节点	AWS TNB 支持 AWS.Compute 的 AL2023 AMI 类型。 EKSManaged节点。	2025 年 4 月 17 日
亚马逊 EKS 节点和托管节点组不再支持 Kubernetes 1.23 版	AWS TNB 不再支持 .compute.eks 和 .Compute 的 Kubernetes 版本 1.23。AWSAWS EKSManaged节点。	2025 年 4 月 4 日
AMI ID 可以更新	现在，您可以在 `UpdateSolNetworkService` API 调用期间更新 ami_id 字段。	2025 年 3 月 31 日
Amazon EKS 节点和托管节点组现在支持 Kubernetes 1.31 版。	AWS TNB 支持适用于.compute.eks 和 .Compute 的 Kubernetes 版本 1.31。AWSAWS EKSManaged节点。	2025 年 2 月 18 日
.Comp@@ ute 的 Kubernetes 版本。 AWS EKSManaged节点	AWS TNB 支持 Kubernetes 1.23 到 1.30 版本来创建亚马逊 EKS 托管节点组。	2025 年 1 月 28 日
适用于集群的 Kubernetes 版本	AWS TNB 现在支持 Kubernetes 版本 1.30 来创建亚马逊 EKS 集群。	2024 年 8 月 19 日

AWS TNB 支持额外的操作来管理网络生命周期。	您可以使用新的网络包和参数值更新实例化或之前更新的网络实例。请参阅：
	• 生命周期运营
	• 更新网络实例
	• AWS TNB 服务角色示例：
	  • 添加以下 Amazon EKS 操作：`eks:UpdateAddon` `eks:UpdateClusterVersion` 、`eks:UpdateNodegroupConfig` 、、`eks:UpdateNodegroupVersion` 、`eks:DescribeUpdate`
	  • 添加以下 CloudFormation 操作：`cloudformation:UpdateStack`
	• 新的部署任务：`InfrastructureUpdate` 、`InventoryRegistration` 、`ValidateNetworkServiceUpdate`
	• API 更新：GetSolNetworkOperationListSolNetworkOperations 、和 UpdateSolNetworkInstance
		2024 年 7 月 30 日

现有任务的新任务和新任务名称	有一项新任务可用。为清楚起见，自 2024 年 3 月 7 日起，一些现有任务有了新的名称。	2024 年 5 月 7 日

适用于集群的 Kubernetes 版本	AWS TNB 现在支持 Kubernetes 版本 1.29 来创建亚马逊 EKS 集群。	2024 年 4 月 10 日
Support 对网络接口的支持 `security_groups`	您可以将安全组附加到 AWS.networking.eni 节点。	2024 年 4 月 2 日
支持 Amazon EBS 根卷加密	您可以为亚马逊 EBS 根卷启用亚马逊 EBS 加密。要启用，请在 AWS.Compute 中添加属性。 EKSManaged节点或 AWS.Compute。 EKSSelfManagedNode节点。	2024 年 4 月 2 日
对节点的 Support `labels`	您可以在 AWS.Compute 中将节点标签附加到您的节点组。 EKSManaged节点或 AWS.Compute。 EKSSelfManagedNode节点。	2024 年 3 月 19 日
Support 对网络接口的支持 `source_dest_check`	您可以通过.networ AWS king.eni 节点指示是要启用还是禁用网络接口 source/destination 检查。	2024 年 1 月 25 日
对带有自定义用户数据的 Amazon EC2 实例的支持	您可以通过 AWS.Compute 启动带有自定义用户数据的 Amazon EC2 实例。 UserData 节点。	2024 年 1 月 16 日
对安全组的支持	AWS TNB 允许您导入安全组 AWS 资源。	2024 年 1 月 8 日

更新了 network_interfaces 的描述	当该network_interfaces 属性包含在 AWS.Compute 中时。 EKSManaged节点或 AWS.Compute。 EKSSelfManagedNode节点， AWS TNB ENIs 从属性（如果有）或该multus_role 属性获得与之node_role 相关的权限。	2023 年 12 月 18 日
对私有集群的支持	AWS TNB 现在支持私有集群。要指示私有集群，请将 access 属性设置为 PRIVATE。	2023 年 12 月 11 日
适用于集群的 Kubernetes 版本	AWS TNB 现在支持 Kubernetes 版本 1.28 来创建亚马逊 EKS 集群。	2023 年 12 月 11 日
AWS TNB 支持置放群组	为 AWS.Compute.EKSManagedNode 和 AWS.Compute.EKSSelfManagedNode 节点定义添加了置放群组。	2023 年 12 月 11 日

AWS TNB 增加了对以下内容的支持 IPv6	AWS TNB 现在支持使用 IPv6 基础设施创建网络实例。检查节点AWS.networking.vpc、.Networking .Subnet、AWS.N etwork AWS InternetGateway，AWS. 网络。 SecurityGroupIngressRule，AWS. 网络。 SecurityGroupEgressRule，还有 AWS.compute.e ks 用于配置。 IPv6我们还添加了节点 AWS.Networking。 NATGateway以及 AWS.Networking.R oute 进行配置。 NAT64 我们更新了 AWS Amazon EKS 节点组的 AWS TNB 服务角色和 TNB 服务角色以获取权限。 IPv6请参阅服务角色策略示例。	2023 年 11 月 16 日
向 AWS TNB 服务角色策略添加了权限	我们向 Amazon S3 的 AWS TNB 服务角色策略添加了权限 CloudFormation ，并允许启用基础设施实例化。	2023 年 10 月 23 日
AWS TNB 在更多地区推出	AWS TNB 现已在亚太地区（首尔）、加拿大（中部）、欧洲（西班牙）、欧洲（斯德哥尔摩）和南美洲（圣保罗）地区推出。	2023 年 9 月 27 日
AWS.Compute 的标签。 EKSSelfManagedNode	AWS TNB 现在支持`AWS.Compute.EKSSelfManagedNode` 节点定义的标签。	2023 年 8 月 22 日

AWS TNB 支持利用的实例 IMDSv2	启动实例时，必须使用 IMDSv2。	2023 年 8 月 14 日
更新了的权限 MultusRoleInlinePolicy	MultusRoleInlinePolicy 现在包括`ec2:DeleteNetworkInterface` 权限。	2023 年 8 月 7 日
适用于集群的 Kubernetes 版本	AWS TNB 现在支持 Kubernetes 版本 1.27 来创建亚马逊 EKS 集群。	2023 年 7 月 25 日
AWS.compute.eks。 AuthRole	AWS TNB 支持 AuthRole 允许您向 Amazon EKS 集群添加 IAM 角色，`aws-authConfigMap` 以便用户可以使用 IAM 角色访问亚马逊 EKS 集群。	2023 年 7 月 19 日
AWS TNB 支持安全组。	添加了 AWS.Networking。SecurityGroup，AWS. 网络。SecurityGroupEgressRule，以及 AWS.Networking。SecurityGroupIngressRule到 NSD 模板。	2023 年 7 月 18 日
适用于集群的 Kubernetes 版本	AWS TNB 支持 Kubernetes 1.22 至 1.26 版本来创建亚马逊 EKS 集群。 AWS TNB 不再支持 Kubernetes 版本 1.21。	2023 年 5 月 11 日
AWS.Compute。 EKSSelfManagedNode	你可以在区域内、L AWS ocal Zones 和上创建自我管理的工作节点。 AWS Outposts	2023 年 3 月 29 日
初始版本	这是 T AWS NB 用户指南的第一个版本。	2023 年 2 月 21 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。