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

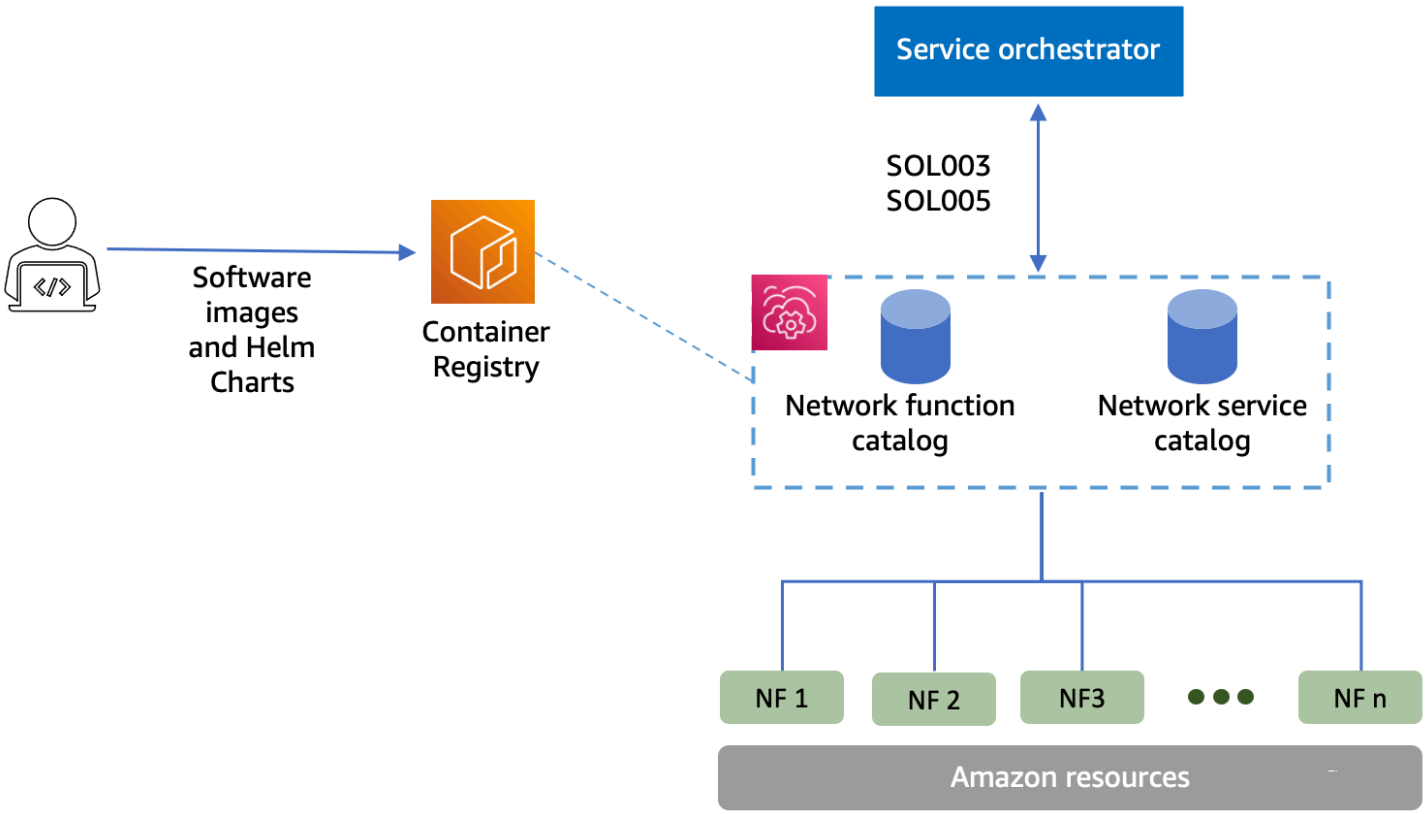
什么是 AWS 电信网络生成器？

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

借 AWS TNB助，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，您可以描述网络的基础架构，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，从而创建全面运行的网络服务。

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



主题

- 新手 AWS？
- AWS TNB 适用于哪些人？
- AWS TNB功能

- 正在访问 AWS TNB
- 的定价 AWS TNB
- 接下来做什么

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：

- AWS简介
- 入门 AWS

AWS TNB 适用于哪些人？

AWS TNB旨在利用所 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CSPs设计、部署和管理网络服务的自定义脚本和配置。 AWS TNB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要部署的网络功能创建全面运行的网络服务。CSP

AWS TNB功能

以下是 a CSP 想要使用的一些原因 AWS TNB：

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与流行的兼容第三方服务协调器集成。ETSI

可扩展

您可以配置 AWS TNB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，例如亚马逊VPCEC2、亚马逊和亚马逊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。当您使用 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 TNB然后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您的RANIMS、核心和网络应用程序。

正在访问 AWS TNB

您可以使用以下任何接口创建、访问和管理您的 AWS TNB资源：

- AWS TNB控制台-提供用于管理网络的 Web 界面。
- AWS TNBAPI— 提供RESTfulAPI用于执行 AWS TNB操作的。有关更多信息，请参阅AWS TNBAPI参考
- AWS Command Line Interface (AWS CLI) — 为一系列 AWS 服务提供命令，包括 AWS TNB。它在Windows、macOS 和 Linux 上受支持。有关更多信息，请参阅 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AWSSDKs。

的定价 AWS TNB

AWS TNB有助于CSPs自动部署和管理其电信网络 AWS。使用时，您需要为以下两个维度付费 AWS TNB：

- 按托管网络功能项目 (MNFI) 小时计算。
- 按API请求次数排列。

当您同时使用其他 AWS 服务时，还会产生额外费用。 AWS TNB有关更多信息，请参阅AWS TNB定价。

若要查看您的账单，请转到 AWS Billing and Cost Management 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 是一项可用于帮助优化 AWS 环境的成本、安全性和性能的服务。有关更多信息，请参阅 AWS Trusted Advisor。

接下来做什么

有关如何开始使用的更多信息 AWS TNB，请参阅以下主题：

- 正在设置 AWS TNB – 完成先决步骤。
- 入门 AWS TNB— 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)、用户平面功能 (UPF) 或完整的 5G 核心。

如何 AWS TNB运作

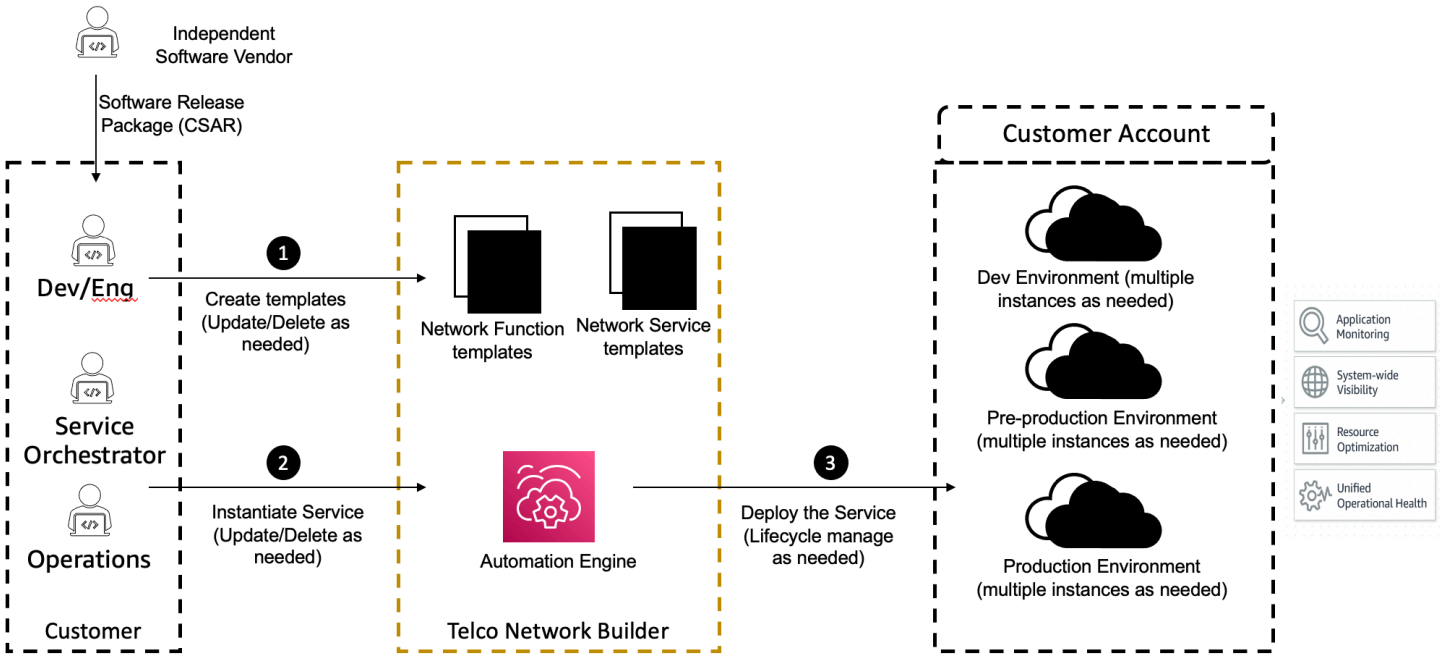
AWS TNB与标准化 end-to-end 协调器和 AWS 资源集成，可运行完整的 5G 网络。

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协调器并与之集成 AWS TNBAPIs，也可以 AWS TNBSDKs用来构建自己的自动化流程。有关更多信息，请参阅 AWS TNB建筑 。

主题

AWS TNB建筑

AWS TNB使您能够通过 AWS Management Console、 AWS CLI AWS TNBRESTAPI、和执行生命周期管理操作SDKs。这样可以利用不同的CSP角色，例如工程、运营和编程系统团队的成员。 AWS TNB您可以创建网络功能包并将其上传为云服务存档 (CSAR) 文件。该CSAR文件包含 Helm 图表、软件镜像和网络功能描述符 (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 的内容集成APIs，以创建全面运行的网络服务：

- Amazon Elastic Conta ECR iner Registry (AmazonISVs) 用于存储独立软件供应商 () 网络功能工件。
- Amazon Elastic Kubernetes Service （亚马逊EKS）来设置集群。
- Amazon VPC 用于网络结构。
- 使用安全组 AWS CloudFormation。
- AWS CodePipeline 对于跨区域的部署目标 AWS 区域，L AWS ocal Zones 和 AWS Outposts。
- IAM来定义角色。
- AWS Organizations 以控制对的访问权限 AWS TNBAPIs。
- AWS 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。您可以请求增加某些配额，但并非所有配额都能增加。

要查看的配额 AWS TNB，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《服务限额用户指南》中的请求提高限额。

您的 AWS 账户 配额与以下有关 AWS TNB。

资源限额	描述	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	描述	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

本主题介绍了一些基本概念，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软件包。
4. 用于 AWS TNB上传定义您的网络功能的CSAR软件包（例如 CU AMF 和UPF），并与持续集成和持续交付 (CI/CD) 管道集成，随着新的网络功能软件映像或客户脚本可用，该管道可以帮助您创建软件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. 使用这些信息来创建您的网络服务描述符 (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. 按照其中的定义对网络功能进行实例化（或配置），NSD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 TNBSDKsCLI、或启动操作，例如实例化网络功能或将网络功能升级到新版本。APIs

AWS TNB支持以下规格。

规范	发布版本	描述
ETSISOL001	v3.6.1	定义TOSCA基于允许的网络功能描述符的标准。
ETSISOL002	v3.6.1	围绕网络功能管理定义模型。
ETSISOL003	v3.6.1	定义网络功能生命周期管理的标准。
ETSISOL004	v3.6.1	定义网络功能包的CSAR标准。
ETSISOL005	v3.6.1	定义网络服务包和网络服务生命周期管理的标准。
ETSISOL007	v3.5.1	定义TOSCA基于允许的网络服务描述符的标准。

网络功能包适用于 AWS TNB

使用 AWS TNB，您可以将符合 ETSI SOL 001/ SOL 004 的网络功能包存储到函数目录中。然后，您可以上传包含描述您的网络功能的构件的 Cloud Service Archive (CSAR) 包。

- 网络功能描述文件 – 定义用于包加载和网络功能管理的元数据

- 软件映像 – 引用网络功能容器映像。亚马逊弹性容器注册表 (AmazonECR) 可以充当您的网络功能镜像存储库。
- 其它文件 – 用于管理网络功能；例如，脚本和 Helm 图表。

是由OASISTOSCA标准定义的软件包，包括符合规范的网络/服务描述符。CSAR OASIS TOSCA YAML有关所需YAML规范的信息，请参见TOSCA的参考 AWS TNB。

以下是网络功能描述文件的示例。

```
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"
```

的网络服务描述符 AWS TNB

AWS TNB存储有关您要部署的网络功能以及如何将其部署到目录中的网络服务描述符 (NSDs)。您可以按照 ETSI SOL 007 的说明上传YAMLNSD文件 (`vnfd.yaml`)，以包含以下信息：

- 您要部署的网络功能
- 联网指令
- 计算指令
- 生命周期挂钩（自定义脚本）

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

以下是NSD演示如何建模 AWS 服务的片段。网络功能将部署在 Kubernetes 版本 EKS 1.27 的亚马逊集群上。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
```

的管理和运营 AWS TNB

使用 AWS TNB，您可以使用符合 ETSI SOL 003 和 SOL 005 的标准化管理操作来管理您的网络。您可以使用 AWS TNBAPIs来执行生命周期操作，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。

的网络服务描述符 AWS TNB

网络服务描述符 (NSD) 是网络包中的一个`.yaml`文件，它使用该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 AWS 的基础架构。要定义NSD和配置您的底层资源和网络生命周期操作，您必须了解支持的NSDTOSCA架构 AWS TNB。

您的NSD文件分为以下几个部分：

1. TOSCA定义版本-这是NSDYAML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。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NSDYAML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
```

正在设置 AWS TNB

AWS TNB通过完成本主题中描述的任务进行设置。

任务

- 注册获取 AWS 账户
- 创建具有管理访问权限的用户
- 选择一个 AWS 地区
- 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

如果您没有 AWS 账户，请完成以下步骤来创建一个。

报名参加 AWS 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。您可以随时前往 https://aws.amazon.com/并选择“我的账户”，查看您当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。

2. 为您的 root 用户开启多重身份验证 (MFA)。

   有关说明，请参阅《用户指南》中的 “为 AWS 账户 root 用户（控制台）启用虚拟MFA设备” IAM。

创建具有管理访问权限的用户

1. 启用 “IAM身份中心”。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。

2. 在 IAM Identity Center 中，向用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 Ident IAM ity Center 用户登录URL，请使用您在创建 Ident IAM ity Center 用户时发送到您的电子邮件地址的登录信息。

  有关使用 Ident IAM ity Center 用户登录的帮助，请参阅《AWS 登录 用户指南》中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个遵循应用最低权限权限的最佳实践的权限集。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

选择一个 AWS 地区

要查看可用区域列表 AWS TNB，请参阅AWS 区域服务列表。要查看用于编程访问的终端节点列表，请参阅中的AWS TNB终端节点AWS 一般参考。

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，一些 AWS 服务还提供选定区域的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	
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	

区域名称	区域	端点	协议	
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 AWS TNB进行访问 AWS CLI。要开始使用，请参阅 AWS Command Line Interface 用户指南。有关命令的更多信息 AWS TNB，请参阅《AWS CLI 命令参考》中的 tnb。

设置 AWS TNB角色

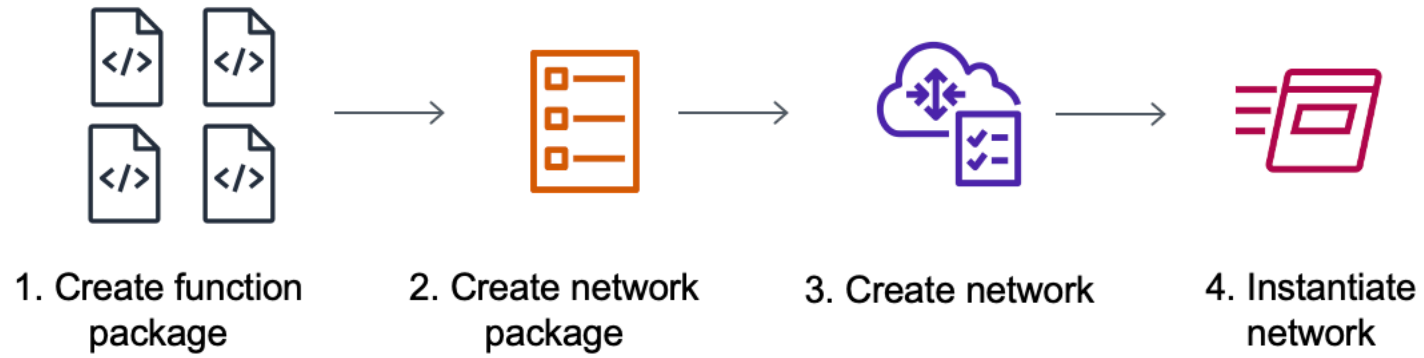
您必须创建IAM服务角色才能管理 AWS TNB解决方案的不同部分。 AWS TNB服务角色可以代表您API调用其他 AWS 服务（例如 AWS CloudFormation AWS CodeBuild、、以及各种计算和存储服务），以实例化和管理用于部署的资源。

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。

入门 AWS TNB

本教程演示 AWS TNB如何使用网络功能部署，例如集中式单元 (CU)、接入和移动管理功能 (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，您必须具备以下条件：

- B AWS usiness Support 计划。
- 通过IAM角色获得的权限。
- 符合 ETSI SOL 001/ 00 SOL 4 标准的网络功能 (NF) 软件包。
- 符合 ETSI SOL 007 的@@ 网络服务描述符 (NSD) 模板。

您可以使用 AWS TNB GitHub 网站示例包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。该CSAR文件包含 Helm 图表、软件镜像和网络功能描述符 (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.zip文件NSD作为文件上传。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

网络实例是在中创建的可以部署 AWS TNB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、 AWS TNB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以创建全面运行的网络服务时。

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包中的参数不变。
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。

的功能包 AWS TNB

函数包是CSAR（Cloud Service Archive）格式的.zip 文件，其中包含网络函数（ETSI标准电信应用程序）和功能包描述符，后者使用该TOSCA标准来描述网络功能应如何在网络上运行。

任务

- 在中创建函数包 AWS TNB
- 在中查看功能包 AWS TNB
- 从以下网址下载函数包 AWS TNB
- 从中删除函数包 AWS TNB

在中创建函数包 AWS TNB

了解如何在 AWS TNB网络函数目录中创建函数包。创建函数包是在中创建网络的第一步 AWS TNB。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
```

在中查看功能包 AWS TNB

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
```

从以下网址下载函数包 AWS TNB

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
```

从中删除函数包 AWS TNB

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
```

网络套餐适用于 AWS TNB

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

任务

- 在中创建网络包 AWS TNB
- 在中查看网络套餐 AWS TNB
- 从以下网址下载网络包 AWS TNB
- 从中删除网络包 AWS TNB

在中创建网络包 AWS TNB

网络包由网络服务描述符 (NSD) 文件（必填）和任何其他文件（可选）组成，例如特定于您需求的脚本。例如，如果您的网络包中有多个函数包，则可以使用NSD来定义哪些网络函数应在某些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.zip文件NSD作为文件上传。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

要创建网络包，请使用 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
```

在中查看网络套餐 AWS TNB

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
```

从以下网址下载网络包 AWS TNB

了解如何从网络服务目录下载 AWS TNB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
```

从中删除网络包 AWS TNB

了解如何从网络服务目录中删除 AWS TNB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
```

的网络实例 AWS TNB

网络实例是在中创建的可以部署 AWS TNB的单个网络。

任务

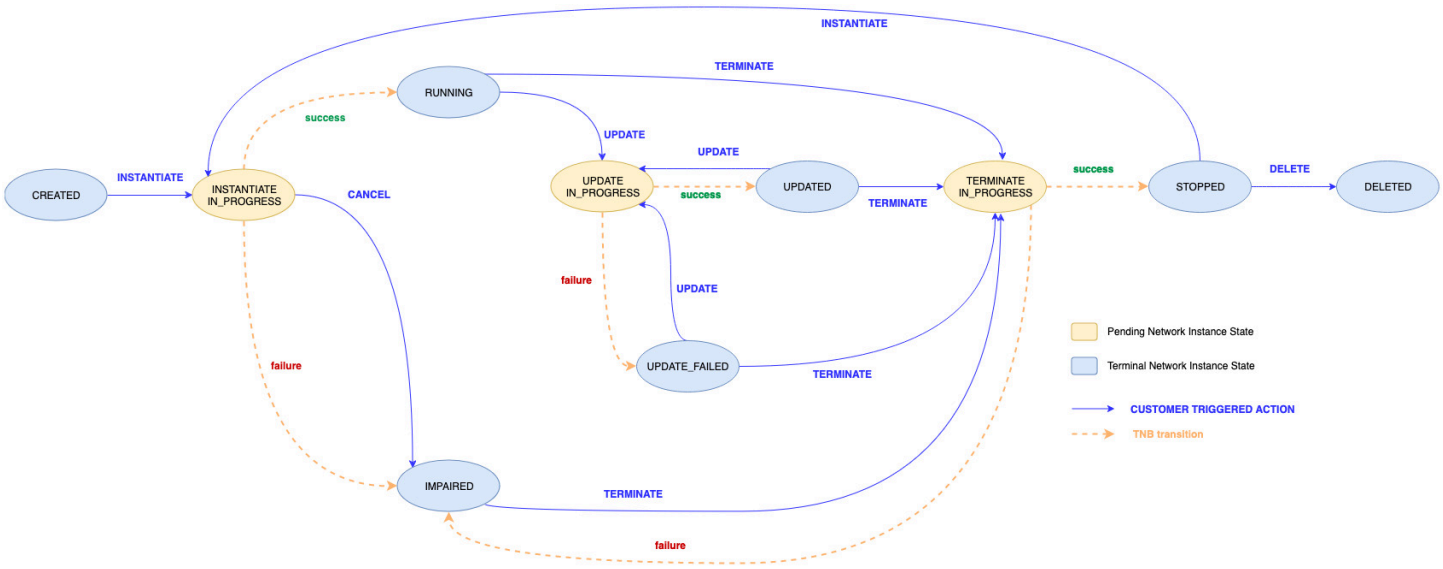
- 网络实例的生命周期操作
- 使用创建网络实例 AWS TNB
- 使用实例化网络实例 AWS TNB
- 更新中的函数实例 AWS TNB
- 更新中的网络实例 AWS TNB
- 在中查看网络实例 AWS TNB
- 终止并从中删除网络实例 AWS TNB

网络实例的生命周期操作

AWS TNB允许您使用与 ETSI SOL 003 和 SOL 005 相匹配的标准化管理操作轻松管理网络。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
- 查看网络详细信息和状态
- 终止网络

下图显示了网络管理操作：



使用创建网络实例 AWS TNB

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。

使用实例化网络实例 AWS TNB

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、 AWS TNB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以创建全面运行的网络服务时。

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包中的参数不变。
6. 选择实例化网络。

   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。

更新中的函数实例 AWS TNB

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 ...
```

更新中的网络实例 AWS TNB

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验证网络实例中的数量NSD和VNFD参数是否不超过 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	描述	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。工作节点未更新。	`EKSCluster:`
		`  type: tosca.nodes.AWS.Compute.EKS`
		`  properties:`
		`  version: "1.28"`
			`EKSC`
			`r:`
			
			`typ`
			`tos`
			`es.A`
			`mput`

参数	描述	示例：之前	示例：之后
			pro s: ver "1.

参数	描述	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`
			`EKSNoup0`
			`...`
			`sca`
			`pros:`
			`desize:`

参数	描述	示例：之前	示例：之后
			min max

参数	描述	示例：之前	示例：之后

亚马逊EBSCSI插件属性	您可以在您的亚马逊EKS集群上启用或禁用亚马逊EBSCSI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	描述	示例：之前	示例：之后
			ksbu "

参数	描述	示例：之前	示例：之后
VNF	您可以引用VNFs中的NSD并将它们部署到NSD使用VNFDeploymentTOSCA节点在中创建的集群。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 : "vn Upd VNF - des r_id "b7 839d -916 a166 " nam : "vn Add VNF ....

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

参数	描述	示例：之前	示例：之后

钩子	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	描述	示例：之前	示例：之后

参数	描述	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	描述	示例：之前	示例：之后
			pre e: *Hook*

参数	描述	示例：之前	示例：之后

钩子	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	描述	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	描述	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	描述	示例：之前	示例：之后
			pos te: eHoc

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。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，则网络实例将与您请求更新之前的状态保持不变，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\"}}"
```

在中查看网络实例 AWS TNB

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}$
```

终止并从中删除网络实例 AWS TNB

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}$
```

的网络运营 AWS TNB

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 AWS TNB网络操作
- 取消 AWS TNB网络操作

查看 AWS TNB网络操作

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 AWS TNB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}$
```

TOSCA的参考 AWS TNB

云应用程序的拓扑和编排规范 (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然后上传模板， 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

定义虚拟网络函数描述符 (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`

节TOSCA AWS 点。可能的节点如下：

- AWS.VNF

- AWS.Artifacts.Helm

AWS.VNF

定义 AWS 虚拟网络函数 (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`

描述符的。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`

的版本VNFD。

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`

的作者VNFD.

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

定义一个 AWS 头盔节点。

语法

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

属性

`implementation`

包中包含 Helm 图表的CSAR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

定义网络服务描述符 (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

当你想动态传递参数（例如VPC节点的CIDR块）时，你可以使用{ get_input: *input-parameter-name* }语法并在NSD模板中定义参数。然后在同一个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进口

`descriptor_id`

描述符的。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}`

`namespace`

唯一名称。

必需：是

类型：字符串

拓扑模板

`node_templates`

可能的TOSCA AWS 节点是：

- AWS.NS
- AWS.Compute。 EKS
- AWS.Compute。 EKS。 AuthRole
- AWS.Compute。 EKSManagedNode
- AWS.Compute。 EKSSelfManagedNode
- AWS.Compute。 PlacementGroup
- AWS.Compute。 UserData
- AWS. 联网。 SecurityGroup
- AWS. 联网。 SecurityGroupEgressRule
- AWS. 联网。 SecurityGroupIngressRule
- AWS.Resource.Import
- AWS. 联网。 ENI
- AWS.HookExecution
- AWS. 联网。 InternetGateway
- AWS. 联网。 RouteTable
- AWS.Networking.Subnet
- AWS. 部署。 VNFDeployment
- AWS. 联网。 VPC
- AWS. 联网。 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
```

属性

descriptor_id

描述符的。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

的版本NSD。

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

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 控制平面管理您的所需资源的角色。AWS NFs多容器网络接口 (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基于 Kubernetes 的网络功能。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) 用法的属性。

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。

必需：是

类型：布尔值

`multus_role`

Multus 网络接口管理角色。

必需：是

类型：字符串

`ebs_csi`

定义安装在亚马逊EBS集EKS群中的亚马逊容器存储接口 (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使用亚马逊EKS自行管理的节点。有关更多信息，请参阅《亚马逊EKS用户指南》中的 Amazon Elastic Block Store CSI 驱动程序。

`enabled`

表示是否安装了默认 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

属性

`version`

集群的 Kubernetes 版本。 AWS Telco Network Builder 支持 Kubernetes 版本 1.23 到 1.30。

必需：是

类型：字符串

可能的值：1.23 | 1.24 | 1.25 | 1.26 | 1.27 | 1.28 | 1.29 | 1.30

`access`

集群端点访问。

必需：是

类型：字符串

可能的值：`PRIVATE` | `PUBLIC` | `ALL`

`cluster_role`

集群管理角色。

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。

必需：是

类型：列表

示例

```
SampleEKS:
  type: tosca.nodes.AWS.Compute.EKS
  properties:
    version: "1.23"
    access: "ALL"
    cluster_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleRole"
    ip_family: "IPv6"
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
  capabilities:
    multus:
      properties:
        enabled: true
        multus_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/MultusRole"
    ebs_csi:
      properties:
        enabled: true
        version: "v1.16.0-eksbuild.1"
  requirements:
    subnets:
    - SampleSubnet01
    - SampleSubnet02
```

AWS.Compute。 EKS。 AuthRole

AuthRole 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
```

属性

`role_mappings`

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

`arn`

IAM角色ARN的。

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。

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 Amazon 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或AMI类型的 ID 为您的集群工作节点选择 Amazon 系统映像 (AMI)。
- 为您的节点组提供用于SSH访问和扩展属性的Amazon EC2 key pair。
- 确保您的节点组已与 Amazon EKS 集群关联。
- 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
    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

compute

定义亚马逊EKS托管节点组计算参数的属性，例如亚马逊EC2实例类型和亚马逊EC2实例AMIs。

ami_type

亚马逊EKS支持的AMI类型。

必需：是

类型：字符串

可能的值：AL2_x86_64 | AL2_x86_64_GPU | AL2_ARM_64 | CUSTOM | BOTTLEROCKET_ARM_64 | BOTTLEROCKET_x86_64 | BOTTLEROCKET_ARM_64_NVIDIA | BOTTLEROCKET_x86_64_NVIDIA

ami_id

的 ID AMI。

必需：否

类型：字符串

> **Note**
>
> 如果在模板中同时指定了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。

必需：是

类型：列表

`key_pair`

用于启用SSH访问的EC2密钥对。

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，则默认 AWS TNB会加密 Amazon EBS 根卷。

必需：否

默认：True

类型：布尔值

`root_volume_encryption_key_arn`

AWS KMS 钥ARN匙中的那个。 AWS TNB支持普通密钥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 false，则不包含`root_volume_encryption_key_arn`。
> - AWS TNB支持对由 Amazon EBS 支持的卷进行根卷加密AMI。
> - 如果AMI的根卷已经加密，则必须包含 f `root_volume_encryption_key_arn` or AWS TNB 才能重新加密根卷。
> - 如果AMI的根卷未加密，则 AWS TNB`root_volume_encryption_key_arn`使用加密根卷。

如果不包括root_volume_encryption_key_arn，则 AWS TNB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

scaling

定义亚马逊EKS托管节点组扩展参数的属性，例如所需的亚马逊EC2实例数量以及节点组中亚马逊EC2实例的最小和最大数量。

desired_size

此中的实例数量 NodeGroup。

必需：是

类型：整数

min_size

此中的最小实例数 NodeGroup。

必需：是

类型：整数

max_size

此中的最大实例数 NodeGroup。

必需：是

类型：整数

属性

node_role

附加到 Amazon EC2 实例的IAM角色的。ARN

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

要求

`cluster`

一个 AWS.Compute。 EKS节点。

必需：是

类型：字符串

`subnets`

一个 AWS.Networking.Subnet 节点。

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 AWS.C AWS TNB ompute 中包含了该`multus_role`属性，则会ENIs从该`multus`属性获取与的相关权限。 EKS节点。否则，ENIs从 node _role 属性中 AWS TNB获取与相关的权限。

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。用户数据脚本将传递给托管节点组启动的 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"
    scaling:
      properties:
        desired_size: 1
        min_size: 1
        max_size: 1
  properties:
    node_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleRole"
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
  requirements:
    cluster: SampleEKS
    subnets:
      - SampleSubnet
    network_interfaces:
      - SampleENI01
      - SampleENI02
    security_groups:
      - SampleSecurityGroup01
      - SampleSecurityGroup02
    placement_group: SamplePlacementGroup
    user_data: CustomUserData
    labels:
      - "sampleLabelName001=sampleLabelValue001"
      - "sampleLabelName002=sampleLabelValue002"
```

AWS.Compute。 EKSSelfManagedNode

AWS TNB支持亚马逊EKS自行管理节点，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：

- 通过提供集群工作节点的 ID 为集群工作节点选择 Amazon 系统映像 (AMI) AMI。
- 提供 Amazon EC2 密钥对以供SSH访问。
- 确保您的节点组已与 Amazon EKS 集群关联。
- 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
    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

compute

定义亚马逊EKS自行管理节点的计算参数的属性，例如亚马逊EC2实例类型和亚马逊实EC2例AMIs。

ami_id

用于启动实例的 AMI ID。 AWS TNB支持利用的实例IMDSv2。有关更多信息，请参阅 IMDS版本。

必需：是

类型：字符串

`instance_type`

实例大小。

必需：是

类型：字符串

`key_pair`

用于启用SSH访问的 Amazon EC2 密钥对。

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，则默认 AWS TNB会加密 Amazon EBS 根卷。

必需：否

默认：True

类型：布尔值

`root_volume_encryption_key_arn`

AWS KMS 钥ARN匙中的那个。 AWS TNB支持普通密钥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 false，则不包含`root_volume_encryption_key_arn`。
> - AWS TNB支持对由 Amazon EBS 支持的卷进行根卷加密AMI。
> - 如果AMI的根卷已经加密，则必须包含 f `root_volume_encryption_key_arn` or AWS TNB 才能重新加密根卷。

- 如果AMI的根卷未加密，则 AWS TNB`root_volume_encryption_key_arn`使用加密根卷。

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

scaling

定义亚马逊EKS自行管理节点的扩展参数的属性，例如所需的亚马逊EC2实例数量以及节点组中亚马逊EC2实例的最小和最大数量。

`desired_size`

此中的实例数量 NodeGroup。

必需：是

类型：整数

`min_size`

此中的最小实例数 NodeGroup。

必需：是

类型：整数

`max_size`

此中的最大实例数 NodeGroup。

必需：是

类型：整数

属性

`node_role`

附加到 Amazon EC2 实例的IAM角色的。ARN

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。

必需：是

类型：字符串

`subnets`

一个 AWS.Networking.Subnet 节点。

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 AWS.C AWS TNB ompute 中包含了`multus_role`属性，则会ENIs从`multus`属性获取与的相关权限。 EKS节点。否则，ENIs从 node _role 属性中 AWS TNB获取与相关的权限。

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。用户数据脚本将传递给由自管理节点组启动的 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"
    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。

当您启动新的 Amazon 时EC2instance，Amazon EC2 服务会尝试以这样的方式放置实例，即您的所有实例都分布在底层硬件上，以最大限度地减少相关故障。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
```

属性

`strategy`

用于放置 Amazon EC2 实例的策略。

必需：是

类型：字符串

可能的值：CLUSTER| PARTITION | SPREAD _ HOST | SPREAD _ RACK

- CLUSTER— 将实例紧密地打包在可用区内。这种策略使工作负载能够实现紧密耦合 node-to-node 通信所需的低延迟网络性能，这是高性能计算 () 应用程序的典型特征。HPC
- PARTITION— 将您的实例分布在逻辑分区中，这样一个分区中的实例组就不会与不同分区中的实例组共享底层硬件。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 — 在不同的底层硬件上放置一小组实例，以减少相关的故障。
- SPREAD_ HOST — 仅与 Outpost 置放群组一起使用。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，通过用户数据脚本向集群 AWS TNB提供 Amazon EC2 实例注册。如果还提供了自定义用户数据，则 AWS TNB合并两个脚本并将其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。
> - 由于 AWS TNB AWS 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 AWS CloudFormation 规则。

语法

```
tosca.nodes.AWS.Compute.UserData:
  properties:
```

```
    implementation: String
    content_type: String
```

属性

`implementation`

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

必需：是

类型：字符串

`content_type`

用户数据脚本的内容类型。

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. 网络。 SecurityGroup

AWS TNB支持安全组自动配置亚马逊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
```

属性

`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：._-:/()#,@[]+=&;{}!$*

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：._-:/()#,@[]+=&;{}!$*

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。

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. 网络。 SecurityGroupEgressRule

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

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
```

属性

`cidr_ip`

CIDR格式化IPv4的地址范围。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`

出口流量IPv6的地址范围（CIDR格式为）。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。有关托管前缀列表的更多信息，请参阅 Amazon VPC 用户指南中的托管前缀列表。

必需：否

类型：字符串

`from_port`

如果协议为TCP或UDP，则这是端口范围的起点。如果协议为ICMP或ICMPv6，则这是类型号。值为 -1 表示所有ICMP/ICMPv6类型。如果指定全部ICMP/ICMPv6类型，则必须指定全部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。

必需：是

类型：字符串

to_port

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

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引用。

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. 网络。 SecurityGroupIngressRule

AWS TNB支持安全组入口规则，可自动配置可附加到.Networking 的 Amazon EC2 安全组入口规则。AWS 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
```

属性

`cidr_ip`

CIDR格式化IPv4的地址范围。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`

入口流量IPv6的地址范围（CIDR格式为）。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。有关托管前缀列表的更多信息，请参阅 Amazon VPC 用户指南中的托管前缀列表。

必需：否

类型：字符串

`from_port`

如果协议为TCP或UDP，则这是端口范围的起点。如果协议为ICMP或ICMPv6，则这是类型号。值为 -1 表示所有ICMP/ICMPv6类型。如果指定全部ICMP/ICMPv6类型，则必须指定全部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。

必需：是

类型：字符串

`to_port`

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

必需：是

类型：字符串

`source_security_group`

允许来自其入口流量的源安全组的 ID 或TOSCA引用。

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
```

属性

`resource_type`

导入到的资源类型 AWS TNB。

必需：否

类型：列表

`resource_id`

导入到的资源 ID AWS TNB。

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. 网络。 ENI

网络接口是代表虚拟网卡VPC的逻辑网络组件。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，您可以为其连接网络接口，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
```

属性

device_index

设备索引必须大于零。

必需：是

类型：整数

source_dest_check

表示网络接口是否执行源/目的地检查。值为 `true` 表示已启用检查，`false` 表示已禁用检查。

允许值：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。

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。

必需：是

类型：字符串

vpc

一个 AWS.Networking VPC节点。

必需：是

类型：字符串

示例

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

```
    vpc: SampleVPC
```

AWS. 网络。 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

routing

在中定义路由连接的属性VPC。必须包括 dest_cidr 或 ipv6_dest_cidr 属性。

dest_cidr

用于目标匹配的方IPv4CIDR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

用于目标匹配的方IPv6CIDR块。

必需：如果包含 dest_cidr 属性，则为“否”。

类型：字符串

属性

`tags`

要附加到资源的标签。

必需：否

类型：列表

`egress_only`

一个IPv6特定的属性。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。

必需：是

类型：字符串

`route_table`

一个 AWS.Networking RouteTable节点。

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. 网络。 RouteTable

路由表包含一组名为路由的规则，用于确定来自您VPC或网关内子网的网络流量的定向位置。您必须将路由表与关联VPC。

语法

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

属性

`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。

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

子网是您的一个 IP 地址范围VPC，它必须完全位于一个可用区内。您必须为VPC子网指定、区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
```

属性

`type`

表示在此子网中启动的实例是否收到公有IPv4地址。

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。

必需：是

类型：字符串

`cidr_block`

子网的CIDR区块。

必需：否

类型：字符串

`ipv6_cidr_block`

用于创建IPv6子网的CIDR区块。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 创建的子网IPv6CIDR区块的 2 位十六进制后缀。VPC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`

子网 AWS Outposts 将在ARN其中创建。如果您想在上启动 Amazon EKS 自行管理节点，请将此属性添加到NSD模板中。 AWS Outposts有关更多信息，请参阅EKS《亚马逊EKS用户指南》 AWS Outposts中的 Amazon on。

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

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。

必需：是

类型：字符串

`route_table`

一个 AWS.Networking RouteTable节点。

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. 部署。 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。集群属性指定要托管您的EKS集群 NFs。vnfs 属性为您的部署指定网络功能。您还可以提供 pre_create 和 p ost_ creat e 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`

A AWS. 部署。 VNFDeployment节点。

必需：否

类型：字符串

`cluster`

一个 AWS.Compute。 EKS节点。

必需：是

类型：字符串

`vnfs`

一个AWS。 VNF节点。

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. 网络。 VPC

您必须为虚拟私有云指定一个CIDR区块 (VPC)。

语法

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

属性

`cidr_block`

的IPv4网络范围VPC，CIDR表示法。

必需：是

类型：字符串

`ipv6_cidr_block`

用于创建的IPv6CIDR方块VPC。

允许的值：AMAZON_PROVIDED

必需：否

类型：字符串

dns_support

表示是否在VPC获取DNS主机名中启动的实例。

必需：否

类型：布尔值

默认：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. 网络。 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，则AWS TNB会为您的账户分配一个弹性 IP 并将其与网关关联。

语法

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

```
  requirements:
    subnet: String
    internet_gateway: String
  properties:
    type: String
    eip_allocation_id: String
    tags: List
```

属性

subnet

AWS.Networking.Subnet 节点参考。

必需：是

类型：字符串

internet_gateway

AWS.Networking。 InternetGateway节点引用。

必需：是

类型：字符串

属性

type

指示网关是公有还是私有的。

允许的值：PUBLIC、PRIVATE

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
```

属性

`dest_cidr_blocks`

IPv4通往目标资源的目的地路由列表。

必需：是

类型：列表

成员类型：字符串

属性

`nat_gateway`

AWS.Networking。 NATGateway节点引用。

必需：是

类型：字符串

`route_table`

AWS.Networking。 RouteTable节点引用。

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
```

通用节点

为NSD和定义节点VNFD。

- 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**，以及以下正则表达式：`^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`

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。
- 不要在环境变量名称的开头使用以下 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 _。

例如：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"
```

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 AWS 的共同责任。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。作为AWS 合规计划合规计划合规计划合的一部分，第三方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时如何应用分担责任模型 AWS TNB。以下主题向您介绍如何进行配置 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您的 AWS TNB资源。

内容

中的数据保护 AWS TNB

分 AWS 担责任模型适用于 AWS 电信网络生成器中的数据保护。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架构上的内容的控制。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FAQ。 有关欧洲数据保护的信息，请参阅责任AWS 共担模型和AWS安全GDPR博客上的博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭据并使用 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。我们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)。
- 使用SSL/TLS与 AWS 资源通信。我们需要 TLS 1.2，建议使用 TLS 1.3。
- 使用API进行设置和用户活动记录 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 通过命令行界面或访问时需要 FIPS 140-3 经过验证的加密模块API，请使用端点。FIPS有关可用FIPS端点的更多信息，请参阅联邦信息处理标准 (FIPS) 140-3。

我们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）。这包括您使用 AWS TNB或以其他 AWS 服务 方式使用控制台时API、 AWS CLI、或 AWS SDKs。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URL向外部服务器提供，我们强烈建议您不要在中包含凭据信息，URL以验证您对该服务器的请求。

数据处理

当您关闭 AWS 帐户时，会将您的数据 AWS TNB标记为删除，并将其从任何使用中删除。如果您在 90 天内重新激活 AWS 帐户，则会 AWS TNB恢复您的数据。120 天后， AWS TNB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。

互联网络流量隐私

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 可以帮助管理员安全地控制对 AWS 资源的访问权限。IAM管理员控制谁可以通过身份验证（登录）和授权（拥有权限）使用 AWS TNB资源。IAM 无需支付额外费用即可使用。 AWS 服务

内容

受众

你使用 AWS Identity and Access Management (IAM) 的方式会有所不同，具体取决于你所做的工作 AWS TNB。

服务用户-如果您使用 AWS TNB服务完成工作，则管理员会为您提供所需的凭证和权限。当你使用更多 AWS TNB功能来完成工作时，你可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问中的功能 AWS TNB，请参阅排除 AWS 电信网络生成器身份和访问权限故障。

服务管理员-如果您负责公司的 AWS TNB资源，则可能拥有完全访问权限 AWS TNB。您的工作是确定您的服务用户应访问哪些 AWS TNB功能和资源。然后，您必须向IAM管理员提交更改服务用户权限的请求。查看此页面上的信息以了解的基本概念IAM。要详细了解贵公司如何IAM与配合使用 AWS TNB，请参阅AWS TNB如何使用 IAM。

IAM管理员-如果您是IAM管理员，则可能需要详细了解如何编写用于管理访问权限的策略 AWS TNB。要查看可在中使用的 AWS TNB基于身份的策略示例IAM，请参阅。AWS 电信网络构建器基于身份的策略示例

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 AWS 账户根用户、IAM用户身份或通过担任IAM角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM身份中心）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员之前使用IAM角色设置了联合身份。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM用户指南》中的API请求AWS 签名版本 4。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅用户指南中的多因素身份验证和AWS IAM Identity Center 用户指南IAM中的AWS 多因素身份验证。IAM

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以 root 用户身份登录的任务的完整列表，请参阅《用户指南》中的 “需要根用户凭据的IAM任务”。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 Ident IAM ity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户

和应用程序中使用。有关IAM身份中心的信息，请参阅什么是IAM身份中心？ 在《AWS IAM Identity Center 用户指南》中。

IAM 用户和组

IAM用户是您内部 AWS 账户 对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时证书，而不是创建拥有密码和访问密钥等长期凭证的IAM用户。但是，如果您有需要IAM用户长期凭证的特定用例，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM用户指南》中的定期轮换需要长期凭证的用例的访问密钥。

IAM群组是指定IAM用户集合的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可以拥有一个名为的组，IAMAdmins并授予该组管理IAM资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM用户指南》中的IAM用户用例。

IAM角色

IAM角色是您内部具有特定权限 AWS 账户 的身份。它与IAM用户类似，但与特定人员无关。要在中临时扮IAM演角色 AWS Management Console，可以从用户切换到IAM角色（控制台）。您可以通过调用 AWS CLI 或 AWS API操作或使用自定义操作来代入角色URL。有关使用角色的方法的更多信息，请参阅《IAM用户指南》中的代入角色的方法。

IAM具有临时证书的角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关用于联合身份验证的角色的信息，请参阅《IAM用户指南》中的为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为了控制您的身份在进行身份验证后可以访问的内容，Ident IAM ity Center 会将权限集关联到中的IAM角色。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的 权限集。
- 临时IAM用户权限-IAM 用户或角色可以代入一个IAM角色，为特定任务临时获得不同的权限。
- 跨账户访问-您可以使用IAM角色允许其他账户中的某人（受信任的委托人）访问您账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解角色和基于资源的跨账户访问策略之间的区别，请参阅IAM用户指南IAM中的跨账户资源访问权限。

- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在服务中拨打电话时，该服务通常会在 Amazon 中运行应用程序EC2或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS)-当您使用IAM用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS使用调用委托人的权限 AWS 服务以及 AWS 服务 向下游服务发出请求的请求。FAS只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出请求。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，请参阅转发访问会话。
  - 服务角色-服务IAM角色是服务代替您执行操作的角色。IAM管理员可以在内部创建、修改和删除服务角色IAM。有关更多信息，请参阅《IAM用户指南》 AWS 服务中的创建角色以向委派权限。
  - 服务相关角色-服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 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相关角色的权限。
- 在 Amazon 上运行的应用程序 EC2 — 您可以使用IAM角色管理在EC2实例上运行并发出 AWS CLI 或 AWS API请求的应用程序的临时证书。这比在EC2实例中存储访问密钥更可取。要为EC2实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建一个附加到该实例的实例配置文件。实例配置文件包含角色并允许在EC2实例上运行的程序获得临时证书。有关更多信息，请参阅IAM用户指南中的使用IAM角色向在 Amazon EC2 实例上运行的应用程序授予权限。

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。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以JSON文档 AWS 形式存储在中。有关JSON策略文档结构和内容的更多信息，请参阅《IAM用户指南》中的JSON策略概述。

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。

默认情况下，用户和角色没有权限。要授予用户对其所需资源执行操作的权限，IAM管理员可以创建IAM策略。然后，管理员可以将IAM策略添加到角色中，用户可以代入这些角色。

IAM无论您使用何种方法执行操作，策略都会定义该操作的权限。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或获取角色信息 AWS API。

基于身份的策略

基于身份的策略是可以附加到身份（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托管策略定义自定义IAM权限。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管策略或内联策略之间进行选择，请参阅《IAM用户指南》中的在托管策略和内联策略之间进行选择。

基于资源的策略

基于资源的JSON策略是您附加到资源的策略文档。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。

基于资源的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IAM中使用 AWS 托管策略。

访问控制列表 (ACLs)

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

Amazon S3 AWS WAF、和亚马逊VPC就是支持的服务示例ACLs。要了解更多信息ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界-权限边界是一项高级功能，您可以在其中设置基于身份的策略可以向IAM实体（IAM用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM用户指南》中的IAM实体的权限边界。
- 服务控制策略 (SCPs)-SCPs 是为中的组织或组织单位 (OU) 指定最大权限的JSON策略 AWS Organizations。 AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或

所有帐户。对成员账户中的实体（包括每个实体）的权限进行了SCP限制 AWS 账户根用户。有关 Organization SCPs s 和的更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。

- 会话策略 – 会话策略是当您以编程方式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。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。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。

AWS TNB如何使用 IAM

在使用管理IAM访问权限之前 AWS TNB，请先了解哪些IAM功能可供使用 AWS TNB。

IAM可以与 AWS Telco Network Builder 一起使用的功能

IAM功能	AWS TNB支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	是
策略条件键	是
ACLs	否
ABAC（策略中的标签）	是
临时凭证	是
主体权限	是
服务角色	否
服务相关角色	否

要全面了解大多数IAM功能 AWS TNB以及其他 AWS 服务是如何使用的，请参阅《IAM用户指南》IAM中与之配合使用的AWS 服务。

基于身份的策略 AWS TNB

支持基于身份的策略：是

基于身份的策略是可以附加到身份（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托管策略定义自定义IAM权限。

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。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可以在JSON策略中使用的所有元素，请参阅IAM用户指南中的IAMJSON策略元素参考。

基于身份的策略示例 AWS TNB

要查看 AWS TNB基于身份的策略的示例，请参阅。AWS 电信网络构建器基于身份的策略示例

内部基于资源的政策 AWS TNB

支持基于资源的策略：否

基于资源的JSON策略是您附加到资源的策略文档。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另一个账户中的IAM实体指定为基于资源的策略中的委托人。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时 AWS 账户，可信账户中的IAM管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM用户指南》IAM中的跨账户资源访问权限。

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Action元素描述了可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API操作同名。也有一些例外，例如没有匹配API操作的仅限权限的操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

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 AWS Telco Network Builder 定义的操作。

正在执行的策略操作在操作前 AWS TNB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。AWS 电信网络构建器基于身份的策略示例

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。

ResourceJSON策略元素指定要应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。最佳做法是，使用资源的 Amazon 资源名称 (ARN) 指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

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

要查看 AWS TNB资源类型及其列表ARNs，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，请参阅 AWS Telco Network Builder 定义的操作。ARN

要查看 AWS TNB基于身份的策略的示例，请参阅。AWS 电信网络构建器基于身份的策略示例

的策略条件密钥 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。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。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。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在资源上标有IAM用户的用户名时，您才能向IAM用户授予访问该资源的权限。有关更多信息，请参阅《IAM用户指南》中的IAM策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件键。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 AWS Telco 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基于身份的策略的示例，请参阅。AWS 电信网络构建器基于身份的策略示例

ACLs在 AWS TNB

支持ACLs：否

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

ABAC与 AWS TNB

支持ABAC（策略中的标签）：是

基于属性的访问控制 (ABAC) 是一种基于属性定义权限的授权策略。在中 AWS，这些属性称为标签。您可以将标签附加到IAM实体（用户或角色）和许多 AWS 资源。为实体和资源添加标签是的第一步。ABAC然后，您可以设计ABAC策略，允许在委托人的标签与他们尝试访问的资源上的标签匹配时进行操作。

ABAC在快速增长的环境中很有用，也有助于解决策略管理变得繁琐的情况。

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。

有关更多信息ABAC，请参阅《IAM用户指南》中的使用ABAC授权定义权限。要查看包含设置步骤的教程ABAC，请参阅IAM用户指南中的使用基于属性的访问控制 (ABAC)。

将临时证书与 AWS TNB

支持临时凭证：是

当你使用临时证书登录时，有些 AWS 服务 不起作用。有关其他信息，包括哪些 AWS 服务 适用于临时证书 AWS 服务 ，请参阅《IAM用户指南》IAM中的 “适用于临时证书”。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《用户指南》中的从IAM用户切换到IAM角色（控制台）。

您可以使用 AWS CLI 或手动创建临时证书 AWS API。然后，您可以使用这些临时证书进行访问 AWS。 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，请参阅中的临时安全证书IAM。

的跨服务主体权限 AWS TNB

支持转发访问会话 (FAS)：是

当您使用IAM用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS使用调用委托人的权限 AWS 服务以及 AWS 服务 向下游服务发出请求的请求。FAS只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出请求。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，请参阅转发访问会话。

AWS TNB 的服务角色

支持服务角色：否

服务IAM角色是服务代替您执行操作的角色。IAM管理员可以在内部创建、修改和删除服务角色IAM。有关更多信息，请参阅《IAM用户指南》 AWS 服务中的创建角色以向委派权限。

的服务相关角色 AWS TNB

支持服务相关角色：否

服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 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相关角色的权限。

AWS 电信网络构建器基于身份的策略示例

默认情况下，用户和角色无权创建或修改 AWS TNB资源。他们也无法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或来执行任务 AWS API。要授予用户对其所需资源执行操作的权限，IAM管理员可以创建IAM策略。然后，管理员可以将IAM策略添加到角色中，用户可以代入这些角色。

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

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

内容

策略最佳实践

基于身份的策略决定了某人是否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AWS 管策略或工作职能托管策略。

- 应用最低权限权限-使用IAM策略设置权限时，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。有关使用应用权限IAM的更多信息，请参阅《IAM用户指南》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-您可以在策略中添加条件以限制对操作和资源的访问权限。例如，您可以编写一个策略条件来指定所有请求都必须使用发送SSL。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 AWS CloudFormation。有关更多信息，请参阅《IAM用户指南》中的IAMJSON策略元素：条件。
- 使用 A IAM ccess Analyzer 验证您的IAM策略以确保权限的安全性和功能性 — A IAM ccess Analyzer 会验证新的和现有的策略，以便策略符合IAM策略语言 (JSON) 和IAM最佳实践。IAMAccess Analyzer 提供了 100 多项策略检查和可行的建议，可帮助您制定安全和实用的策略。有关更多信息，请参阅IAM用户指南中的使用 A IAMccess Analyzer 验证策略。
- 需要多重身份验证 (MFA)-如果您的场景需要IAM用户或 root 用户 AWS 账户，请打开MFA以提高安全性。要要求MFA何时调用API操作，请在策略中添加MFA条件。有关更多信息，请参阅《IAM用户指南》MFA中的使用进行安全API访问。

有关中最佳做法的更多信息IAM，请参阅《IAM用户指南》IAM中的安全最佳实践。

使用控制 AWS TNB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的 AWS TNB资源的详细信息 AWS 账户。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。

您无需为仅拨打 AWS CLI 或的用户设置最低控制台权限 AWS API。相反，只允许访问与他们尝试执行的API操作相匹配的操作。

服务角色策略示例

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

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

您可以使用服务创建服务角色及其权限策略。IAM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，您可以以管理员身份创建 AWS TNB服务角色并将其提供给 AWS TNB。此角色 AWS TNB允许调用其他服务，例如 Amazon Elastic Kubernetes Serv AWS CloudFormation ice Amazon Kubernetes Service，为您的网络配置所需的基础设施，并按照您的定义配置网络功能。NSD

我们建议您对 AWS TNB服务IAM角色使用以下角色和信任策略。在缩小此策略的权限范围时，请记住，对于从您的策略中解除的资源， AWS TNB可能会因访问被拒绝错误而失败。

以下代码显示了 AWS TNB服务角色策略：

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

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

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

```
                "ec2:DisassociateVpcCidrBlock",
                "ec2:ReleaseAddress",
                "ec2:UnassignIpv6Addresses",
                "ec2:DescribeImages",
                "eks:CreateCluster",
                "eks:ListClusters",
                "eks:RegisterCluster",
                "eks:TagResource",
                "eks:DescribeAddonVersions",
                "events:DescribeRule",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies",
                "iam:PassRole"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TNBAccessComputePerms"
        },
        {
            "Action": [
                "codebuild:BatchDeleteBuilds",
                "codebuild:BatchGetBuilds",
                "codebuild:CreateProject",
                "codebuild:DeleteProject",
                "codebuild:ListBuildsForProject",
                "codebuild:StartBuild",
                "codebuild:StopBuild",
                "events:DeleteRule",
                "events:PutRule",
                "events:PutTargets",
                "events:RemoveTargets",
                "s3:CreateBucket",
                "s3:GetBucketAcl",
                "s3:GetObject",
                "eks:DescribeNodegroup",
                "eks:DeleteNodegroup",
                "eks:AssociateIdentityProviderConfig",
                "eks:CreateNodegroup",
                "eks:DeleteCluster",
                "eks:DeregisterCluster",
                "eks:UpdateAddon",
                "eks:UpdateClusterVersion",
                "eks:UpdateNodegroupConfig",
                "eks:UpdateNodegroupVersion",
```

```
                "eks:DescribeUpdate",
                "eks:UntagResource",
                "eks:DescribeCluster",
                "eks:ListNodegroups",
                "eks:CreateAddon",
                "eks:DeleteAddon",
                "eks:DescribeAddon",
                "eks:DescribeAddonVersions",
                "s3:PutObject",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks",
                "cloudformation:UpdateStack",
                "cloudformation:UpdateTerminationProtection"
            ],
            "Resource": [
                "arn:aws:events:*:*:rule/tnb*",
                "arn:aws:codebuild:*:*:project/tnb*",
                "arn:aws:logs:*:*:log-group:/aws/tnb*",
                "arn:aws:s3:::tnb*",
                "arn:aws:eks:*:*:addon/tnb*/*/*",
                "arn:aws:eks:*:*:cluster/tnb*",
                "arn:aws:eks:*:*:nodegroup/tnb*/tnb*/*",
                "arn:aws:cloudformation:*:*:stack/tnb*"
            ],
            "Effect": "Allow",
            "Sid": "TNBAccessInfraResourcePerms"
        },
        {
            "Sid": "CFNTemplatePerms",
            "Effect": "Allow",
            "Action": [
              "cloudformation:GetTemplateSummary"
            ],
            "Resource": "*"
         },
        {
            "Sid": "ImageAMISSMPerms",
            "Effect": "Allow",
            "Action": [
                "ssm:GetParameters"
            ],
            "Resource": [
```

```
                "arn:aws:ssm:*::parameter/aws/service/eks/optimized-ami/*",
                "arn:aws:ssm:*::parameter/aws/service/bottlerocket/*"
            ]
        },
        {
            "Action": [
                "tag:GetResources"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TaggingPolicy"
        },
        {
            "Action": [
                "outposts:GetOutpost"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "OutpostPolicy"
        }
    ]
}
```

以下代码显示了 AWS TNB服务信任策略：

```
{
  "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 TNBAmazon EKS 集群的服务角色

当您在中创建 Amazon EKS 资源时NSD，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: 角色
- 选择堆栈模板

AWS TNBAmazon EKS 节点组的服务角色

在中创建 Amazon EKS 节点组资源时NSD，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 AWS 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: 角色
- 选择堆栈模板

AWS TNBMultus 的服务角色

当您在中创建 Amazon EKS 资源NSD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 AWS 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: 角色

- 选择堆栈模板

AWS TNB生命周期挂钩策略的服务角色

当你的NSD或网络函数包使用生命周期挂钩时，你需要一个服务角色来允许你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。

以下示例显示了为生命周期挂钩策略创建 AWS TNB服务角色的 AWS 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: 角色
- 选择堆栈模板

允许用户查看他们自己的权限

此示例说明如何创建允许IAM用户查看附加到其用户身份的内联和托管策略的策略。此策略包括在控制台上或使用或以编程方式完成此操作的 AWS CLI 权限。 AWS API

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

排除 AWS 电信网络生成器身份和访问权限故障

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

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 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，表明您无权执行某个操作，则您必须更新策略以允许执行该操作。

当mateojacksonIAM用户尝试使用控制台查看虚构*my-example-widget*资源的详细信息但没有虚构权限时，就会出现以下示例错误。tnb:*GetWidget*

```
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。

当名为的IAM用户marymajor尝试使用控制台在中执行操作时，会出现以下示例错误 AWS TNB。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 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，可以担任角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解是否 AWS TNB支持这些功能，请参阅AWS TNB如何使用 IAM。
- 要了解如何提供对您拥有的资源的访问权限，请参阅《IAM用户指南》中的 AWS 账户 向其他IAM用户提供访问权限。 AWS 账户
- 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
- 要了解如何通过联合身份验证提供访问权限，请参阅《用户指南》中的向经过外部身份验证的用户提供访问权限（联合身份验证）。IAM
- 要了解使用角色和基于资源的策略进行跨账户访问的区别，请参阅IAM用户指南IAM中的跨账户资源访问权限。

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。 AWS 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了部署以安全性和合规性为重点 AWS 的基准环境的步骤。
- 在 Amazon Web Services 上进行HIPAA安全与合规架构 — 本白皮书描述了各公司如何使用 AWS 来创建HIPAA符合条件的应用程序。

> **Note**
>
> 并非所有 AWS 服务 人都有HIPAA资格。有关更多信息，请参阅《HIPAA符合条件的服务参考》。

- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践， AWS 服务并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 这 AWS 服务 可以全面了解您的安全状态 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务 检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 可以帮助您满足各种合规性要求 PCIDSS，例如满足某些合规性框架规定的入侵检测要求。
- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

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、容错性和可扩展性。

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呼叫 AWS TNB通过网络进行访问。客户端必须支持以下内容：

- 传输层安全 (TLS)。我们需要 TLS 1.2，建议使用 TLS 1.3。
- 具有完美前向保密性的密码套件 ()，例如（Ephemeral Diffie-HellmanPFS）或（Elliptic C DHE urve Ephemeral Diffie-Hellman）。ECDHE大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与IAM委托人关联的私有访问密钥对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

下面是一些责任共担示例：

- AWS 负责保护支持的组件 AWS TNB，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
  - AWS TNB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、群组、角色和策略
  - 用于存储数据的 S3 存储桶 AWS TNB
  - 其他 AWS 服务 和您用来支持您通过配置的网络服务的资源 AWS TNB
  - 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您通过 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) 中的计算实例上运行。A 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。内部计算实例之间的通信VPCs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为在同一地区运行的所有客户提供的网络服务共享相同的VPC服务。通过 AWS TNB为不同客户配置的网络服务在同VPC一个客户中使用不同的计算实例。

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

您 AWS TNB通过 AWS Management Console、 AWS Command Line Interface (AWS CLI) 和的连接 AWS SDKs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包括比。更高的安全性IMDSV1。有关更多信息，请参阅通过对 Amazon EC2 实例元数据服务的增强，对开放的防火墙、反向代理和SSRF漏洞进行深度防御。

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

监控 AWS TNB

监控是维护和其他 AWS 解决方案的可靠性、可用性和性能的重要组成部分。 AWS TNB AWS AWS CloudTrail 提供监视 AWS TNB、报告问题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 日志包含有关号召性用API语的信息 AWS TNB。它们还包含来自亚马逊EC2和亚马逊等服务的号召性用语的信息EBS。API

使用记录 AWS Telco 网络生成器API呼叫 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 将所有API呼叫捕获 AWS TNB为事件。捕获的调用包括来自 AWS TNB控制台的调用和对 AWS TNBAPI操作的代码调用。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 AWS TNB、发出请求的 IP 地址、发出请求的时间以及其他详细信息。

每个事件或日记账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身份中心用户发出。
- 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，并且您自动可以访问 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 Management Console 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 Lak e 允许您对事件进行SQL基于查询的操作。 CloudTrail Lake 将基于行的格式的现有事件转换为 Apache JSON ORC 格式。ORC是一种列式存储格式，已针对快速检索数据进行了优化。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的顺序出现。

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 AWS TNB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	在亚马逊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	按照中的定义执行生命周期挂钩。NSD
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	从中注销 AWS 资源。 AWS TNB
-	InventoryRegistration	在中注册 AWS 资源 AWS TNB。
KubernetesClusterConfiguration	ClusterConfiguration	EKS AuthMap 按照中的定义，配置 Kubernetes 集群并向 Amazon 添加其他IAM角色。NSD

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	使用亚马逊EKS和 Kubernetes 控制平面启动自管理节点。
-	ValidateNetworkServiceUpdate	在更新网络实例之前运行验证。

的服务配额 AWS TNB

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。

以下是的服务配额 AWS TNB。

名称	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	一个区域中函数包的最大数量。
网络程序包	每个受支持的区域：40 个	是	一个区域内网络包的最大数量。
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

下表描述了的文档版本 AWS TNB。

变更	说明	日期
适用于集群的 Kubernetes 版本	AWS TNB现在支持 Kubernetes 版本 1.30 来创建亚马逊集群。EKS	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来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 AWS CloudFormation 操作：`cloudformation:UpdateStack`
	• 新的部署任务：`InfrastructureUpdate` 、`InventoryRegistration` 、`ValidateN`
		2024 年 7 月 30 日

	etworkServiceUpdate • API更新：GetSolNetworkOperationListSolNetworkOperations、和UpdateSolNetworkInstance	
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 来创建亚马逊集群。EKS	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。 EKSManagedNode或者 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。 EKSManagedNode或者 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 .Networking 来指示是要启用还是禁用网络接口源/目标检查。 AWS ENI节点。	2024 年 1 月 25 日

支持使用自定义用户数据的 Amazon EC2 实例	您可以通过 AWS.Compute 启动带有自定义用户数据的亚马逊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network_interfaces 属性包含在 AWS.Compute 中时。 EKSManagedNode或者 AWS.Compute。 EKSSelfManagedNode 节点，ENIs从multus_role 属性 AWS TNB获取与相关的权限（如果有），或者从node_role 属性中获取相关权限。	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 来创建亚马逊集群。EKS	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添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，AWS.网络。子网，. 联网。AWS InternetGateway，AWS.联网。 SecurityGroupIngressRule，AWS.联网。 SecurityGroupEgressRule，以及 AWS.Compute。 EKS用于IPv6配置。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 Amazon EKS 节点组的 AWS TNB AWS TNB服务角色和服务角色的IPv6权限。请参阅服务角色策略示例。	2023 年 11 月 16 日
为 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 AWS CloudFormation ，并允许启用基础设施实例化。	2023 年 10 月 23 日
AWS TNB已在更多地区推出	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 现在包含了`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 来创建亚马逊集群。EKS	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 Amazon EKS 集群。	2023 年 7 月 19 日
AWS TNB支持安全组。	添加了 AWS.Networking。SecurityGroup，AWS.联网。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 版本 1.22 到 1.26 来创建亚马逊集群。EKS 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《 AWS TNB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。