



用户指南

AWS 电信网络生成器



AWS 电信网络生成器: 用户指南

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

AWS TNB 是什么？	1
初次使用 AWS？	2
AWS TNB 适用于哪些企业？	2
为什么要使用 AWS TNB？	2
访问 AWS TNB	3
AWS TNB 定价	3
接下来做什么	4
工作原理	5
架构	5
集成	6
限额	6
概念	8
网络功能的生命周期	8
使用标准化接口	9
NF 包	9
NF 服务描述	10
管理和运营	12
网络服务描述文件	12
设置	15
报名参加 AWS	15
选择一个 AWS 地区	15
记下服务端点	16
(可选) 安装 AWS CLI	17
创建 IAM 用户	17
设置 AWS TNB 角色	17
开始使用	18
先决条件	18
创建功能包	19
创建网络包	19
创建和实例化网络实例	19
清理	20
功能包	21
创建	19
查看	22

下载包	23
删除包	23
网络包	25
创建	19
查看	26
下载	27
删除	27
网络	29
实例化	29
查看	30
更新	30
终止和删除	31
网络操作	33
查看	33
取消	33
TOSCA 参考	35
VNFD 模板	35
语法	35
拓扑模板	35
AWS.VNF	36
AWS.Artifacts.Helm	37
NSD 模板	38
语法	38
使用已定义的参数	39
VNFD 导入	39
拓扑模板	40
AWS.NS	40
AWS.Compute.EKS	42
AWS.compute.eks。 AuthRole	45
AWS.compute.eks ManagedNode	47
AWS.compute.eks SelfManagedNode	53
AWS.Compute。 PlacementGroup	59
AWS.Compute。 UserData	61
AWS. 联网。 SecurityGroup	62
AWS. 联网。 SecurityGroupEgressRule	64
AWS. 联网。 SecurityGroupIngressRule	66

AWS.Resource.Import	69
AWS.Networking.ENI	70
AWS.HookExecution	72
AWS. 联网。 InternetGateway	74
AWS. 联网。 RouteTable	76
AWS.Networking.Subnet	77
AWS.Deployment.VNFDeployment	80
AWS.Networking.VPC	82
AWS.Networking.NATGateway	83
AWS.Networking.Route	85
通用节点	86
AWS.HookDefinition.Bash	87
安全性	89
数据保护	89
数据处理	90
静态加密	90
传输中加密	90
互连网络流量隐私	91
Identity and Access Management	91
受众	91
使用身份进行身份验证	92
使用策略管理访问	94
AWS 电信网络构建器如何与 IAM 配合使用	96
基于身份的策略示例	102
故障排除	115
合规性验证	117
韧性	118
基础设施安全性	118
网络连接安全模型	119
IMDS 版本	119
监控	121
CloudTrail 日志	121
CloudTrail 中的 AWS TNB 信息	121
了解 AWS TNB 日志文件条目	122
部署任务	123
限额	126

文档历史记录	127
.....	CXXXi

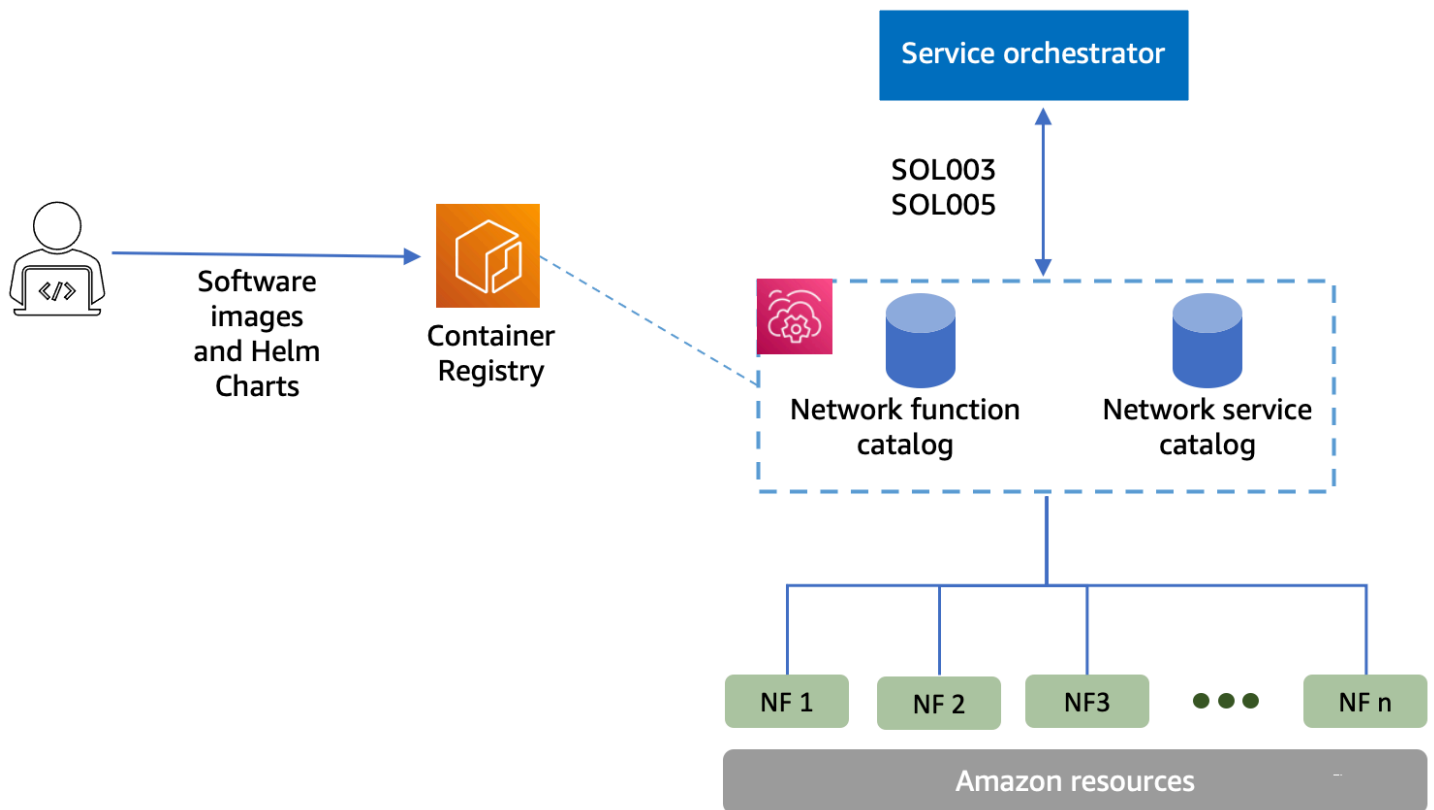
AWS 电信网络构建器是什么？

AWS 电信网络构建器 (AWS TNB) 是一项 AWS 服务，为通信服务提供商 (CSP) 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。

借助 AWS TNB，您可以使用容器化软件映像，在 AWS Cloud 中自动部署可扩展的安全 5G 网络。您不需要学习新技术、决定要使用的计算服务，也不需要了解如何预置和配置 AWS 资源。

相反，您只需要描述您的网络基础设施，并提供来自独立软件供应商 (ISV) 合作伙伴的网络功能软件映像。AWSTNB 与第三方服务编排工具和 AWS 服务集成，可自动预置必要的 AWS 基础设施、部署容器化网络功能以及配置联网和访问管理，从而创建全面运行的网络服务。

下图说明了 AWS TNB 和服务编排工具如何进行逻辑集成，以便使用基于欧洲电信标准协会 (ETSI) 的标准接口部署网络功能。



主题

- [初次使用 AWS ?](#)
- [AWS TNB 适用于哪些企业 ?](#)
- [为什么要使用 AWS TNB ?](#)

- [访问 AWS TNB](#)
- [AWS TNB 定价](#)
- [接下来做什么](#)

初次使用 AWS ?

如果不熟悉 AWS 产品和服务，请使用以下资源了解更多信息：

- [AWS 简介](#)
- [AWS 入门](#)

AWS TNB 适用于哪些企业？

AWS TNB 适用于希望利用 AWS Cloud 提供的成本效益、敏捷性和弹性的 CSP，而且该产品使 CSP 无需编写和维护用于设计、部署和管理网络服务的自定义脚本和配置。AWS TNB 会自动预置必要的 AWS 基础设施、部署容器化网络功能并配置联网和访问管理，以根据 CSP 定义的网络服务描述文件和 CSP 想要部署的网络功能创建可全面运行的网络服务。

为什么要使用 AWS TNB ?

以下是 CSP 应使用 AWS TNB 的一些原因：

帮助简化任务

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

与编排工具集成

AWS TNB 可与符合 ETSI 标准的常见第三方服务编排工具集成。

可扩展

您可以通过配置 AWS TNB 来扩展底层 AWS 资源，以满足流量需求，更高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 AWS 资源

AWS TNB 使您可以通过单个控制面板检查和监控支持您的网络的 AWS 资源，例如 Amazon VPC、Amazon EC2 和 Amazon 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 EC2 实例的实例类型) 时, 您能够以可重复且可扩展的方式跟踪更改。手动执行此操作需要管理网络状态, 创建和删除资源, 并注意所需更改的顺序。使用 AWS TNB 管理网络功能的生命周期时, 您只需要更改描述网络功能的网络服务描述文件。AWSTNB 随后将按正确的顺序自动执行所需的更改。

简化网络功能生命周期

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

访问 AWS TNB

可以使用以下任意接口创建、访问和管理 AWS TNB 资源 :

- AWS TNB 控制台 – 提供用于管理网络的 Web 界面。
- AWS TNB API – 提供用于执行 AWS TNB 操作的 RESTful API。有关更多信息, 请参阅 [AWS TNB API Reference](#)。
- AWS Command Line Interface (AWS CLI) – 提供了适用于各种 AWS 服务 (包括 AWS TNB) 的命令。它在 Windows、macOS 和 Linux 上受支持。有关更多信息, 请参阅 [AWS Command Line Interface](#)。
- AWS SDK – 提供特定于语言的 API 并完成许多连接细节工作。这些细节工作包括计算签名、处理请求重试和处理错误。有关更多信息, 请参阅 [AWS 软件开发工具包](#)。

AWS TNB 定价

AWS TNB 帮助通信服务提供商自动在 AWS 上部署和管理其电信网络。使用 AWS TNB 时, 您需要根据以下两个维度付费 :

- 托管的网络功能项目 (MNFI) 小时数。
- API 请求的数量。

将 AWS TNB 与其它 AWS 服务配合使用时，还会产生额外费用。有关更多信息，请参阅 [AWS TNB Pricing](#)。

若要查看您的账单，请转到 [AWS 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](#)。

如果您有关于 AWS 账单、账户和事件的问题，请[联系 AWS Support](#)。

AWS Trusted Advisor 是一项可用于帮助您优化 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 AWS TNB 的更多信息，请参阅以下主题：

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

AWS TNB 的工作原理

AWS TNB 与标准化的端到端编排工具和 AWS 资源集成，可运营完整的 5G 网络。

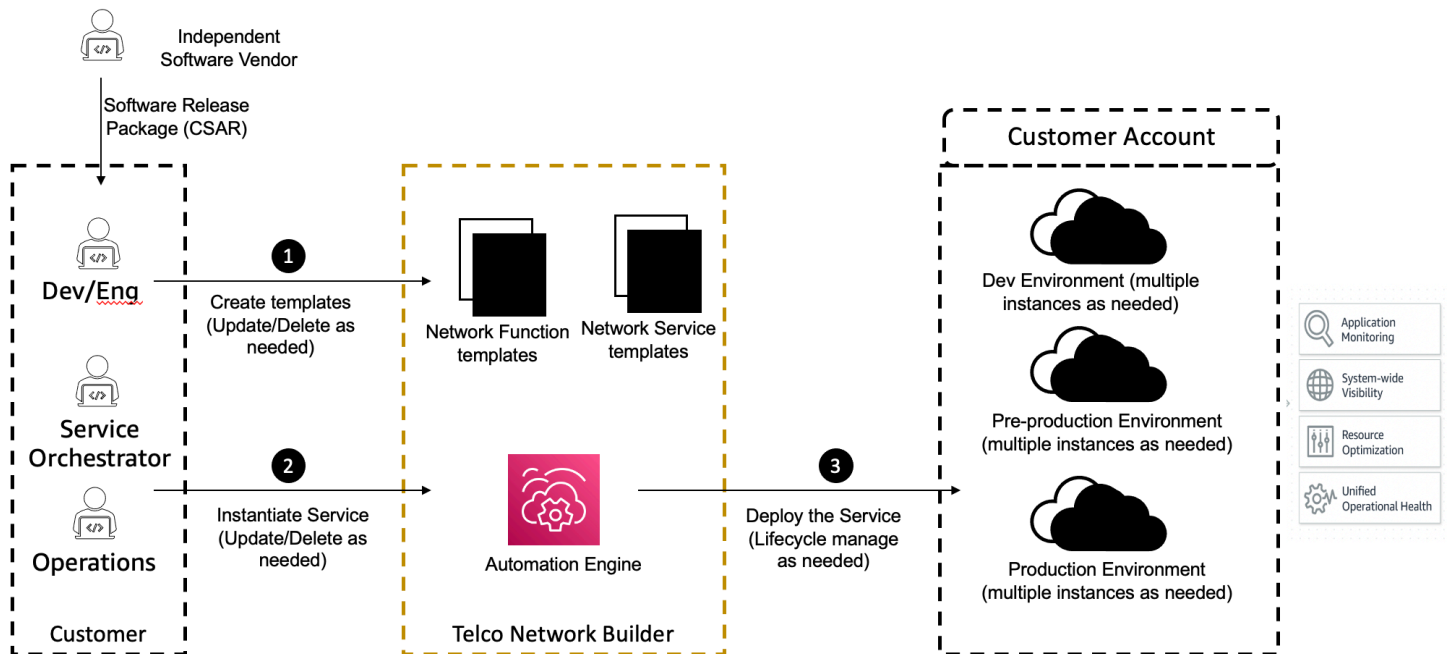
AWS TNB 允许您摄取网络功能包和网络服务描述文件（NSD），并为您提供运营网络的自动化引擎。您可以使用端到端编排工具并与 AWS TNB API 集成，也可以使用 AWS TNB SDK 构建自己的自动化流程。有关更多信息，请参阅 [AWS TNB 架构](#)。

主题

- [AWS TNB 架构](#)
- [与 AWS 服务 集成](#)
- [AWS TNB 资源限额](#)

AWS TNB 架构

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



与 AWS 服务 集成

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。AWSTNB 通过特定于电信的 API 与以下 AWS 服务集成，以创建全面运行的网络服务：

- Amazon Elastic Container Registry (Amazon ECR) ，用于存储独立软件供应商 (ISV) 网络功能构件。
- Amazon Elastic Kubernetes Service (Amazon EKS) ，用于设置集群。
- Amazon VPC ，用于网络结构。
- 使用 AWS CloudFormation 的安全组。
- AWS CodePipeline ，用于跨 AWS 区域、AWS Local Zones 和 AWS Outposts 部署目标。
- IAM ，用于定义角色。
- AWS Organizations ，用于控制对 AWS TNB API 的访问权限。
- AWS Health Dashboard 和 AWS CloudTrail ，用于监控运行状况和发布指标。

AWS TNB 资源限额

您的 AWS 账户对于每项 AWS 服务都具有默认限额（以前称为限制）。除非另有说明，否则，每个限额都特定于一个 AWS 区域。您可以请求提高某些限额，但不能请求提高所有限额。

要查看 AWS TNB 的限额，请打开[服务限额控制台](#)。在导航窗格中，选择 AWS 服务，然后选择 AWS TNB。

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

您的 AWS 账户具有以下与 AWS TNB 相关的限额。

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](#)
- [使用标准化接口](#)
- [适用于 AWS TNB 的网络功能包](#)
- [TNB 的网络功能服务描述符 AWS](#)
- [AWS TNB 的管理和运营](#)
- [TNB 的网络服务描述符 AWS](#)

网络功能的生命周期

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

规划

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

配置

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

实例化

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

生产

在网络功能的生命周期中，您会执行一系列生产操作，例如：

- 更新网络功能配置，例如，更新已部署的网络功能中的值。
- 更换或停用网络功能。

使用标准化接口

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

AWS TNB 支持以下规格。

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

适用于 AWS TNB 的网络功能包

使用 AWS TNB，您可以将符合 ETSI SOL001/SOL004 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网络功能的构件的云服务存档 (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。Amazon Elastic Container Registry (Amazon ECR) 可用作网络功能映像存储库。
- 其它文件 – 用于管理网络功能；例如，脚本和 Helm 图表。

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

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"
```

TNB 的网络功能服务描述符 AWS

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 (NSD)。可以按照 ETSI SOL007 的描述上传 YAML NSD 文件，该文件包括以下内容：

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

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

```
tosca_definitions_version: tnb_simple_yaml_1_0

SampleNFEKS:
  type: tosca.nodes.AWS.Compute.EKS
  properties:
    version: "1.27"
    access: "ALL"
    cluster_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleClusterRole"
  capabilities:
    multus:
      properties:
        enabled: true
  requirements:
    subnets:
      - Subnet01
      - Subnet02

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

```
- Subnet01
network_interfaces:
- ENI01
- ENI02
```

AWS TNB 的管理和运营

借 AWS 助 TNB，您可以使用符合 ETSI SOL003 和 SOL005 的标准化操作来管理您的网络。您可以使用 AWS TNB API 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。
- 管理网络功能包的版本。
- 管理 NSD 的版本。
- 检索有关您部署的网络功能的信息。

TNB 的网络服务描述符 AWS

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

NSD 文件分为以下几个部分：

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

```
tosca_definitions_version: tnb_simple_yaml_1_0
```

2. VNFD – NSD 包含要在其上执行生命周期操作的网络功能的定义。必须使用以下值来标识每个网络功能：

- 用于 `descriptor_id` 的唯一 ID。该 ID 必须与网络功能 CSAR 包中的 ID 相匹配。
- 用于 `namespace` 的唯一名称。该名称必须与唯一 ID 相关联，以便在整个 NSD YAML 文件中更容易引用，如以下示例所示。

```
vnfds:
```

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

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

```
topology_template:

  node_templates:

    SampleNS:
      type: toasca.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: toasca.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

设置 AWS TNB

通过完成本主题中描述的任务来设置 AWS TNB。

任务

- [报名参加 AWS](#)
- [选择一个 AWS 地区](#)
- [记下服务端点](#)
- [\(可选 \) 安装 AWS CLI](#)
- [创建 IAM 用户](#)
- [设置 AWS TNB 角色](#)

报名参加 AWS

当您注册 Amazon Web Services AWS 账户时，系统会自动注册所有服务 AWS，包括 AWS TNB。您只需为使用的服务付费。

如果您 AWS 账户 已有，请跳至下一个任务。如果您还没有 AWS 账户，请使用以下流程创建。

要创建 AWS 账户

1. 打开 <https://portal.aws.amazon.com/billing/signup>。
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](#)。

选择一个 AWS 地区

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

记下服务端点

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

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

区域名称	区域	端点	协议
南美洲 (圣保 罗)	sa-east-1	tnb.sa-east-1.amazonaws.com	HTTPS

(可选) 安装 AWS CLI

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

创建 IAM 用户

AWS Identity and Access Management (IAM) 是一项 Web 服务，可帮助您安全地控制对 AWS 资源的访问。创建 IAM 用户角色以使用短期凭证访问 AWS。

要创建角色，请按照《AWS IAM Identity Center User Guide》的 [Getting started](#) 中的说明操作。

您也可以通过在《AWS Command Line Interface 用户指南》[AWS IAM Identity Center 中配置 AWS CLI 要使用的来](#)配置编程访问权限。

设置 AWS TNB 角色

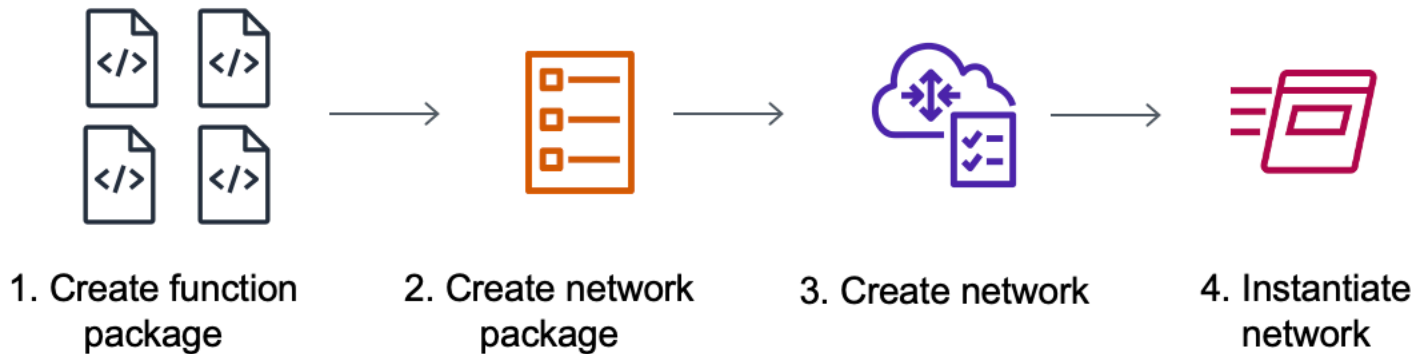
您必须创建 IAM 服务角色才能管理 AWS TNB 解决方案的不同部分。AWS TNB 服务角色可以代表您对其他 AWS 服务（例如 AWS CloudFormation AWS CodeBuild、以及各种计算和存储服务）进行 API 调用，以实例化和用于部署的资源。

有关 AWS TNB 服务角色的更多信息，请参阅[AWS TNB 的身份和访问管理](#)。

AWS TNB 入门

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

下图说明了部署过程：



任务

- [先决条件](#)
- [创建功能包](#)
- [创建网络包](#)
- [创建和实例化网络实例](#)
- [清理](#)

先决条件

在成功执行部署之前，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 SOL001/SOL004 标准的[网络功能 \(NF\) 软件包](#)。
- 符合 ETSI SOL007 的@@ [网络服务描述符 \(NSD\) 模板](#)。

您可以使用 T [AWS NB 示例包](#) [GitHub 网站](#)中的[示例函数包](#)或网络包。

创建功能包

创建功能包

1. 打开 AWS TNB 控制台，[网址为 https://console.aws.amazon.com/tnb/](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，然后将您的 CSAR 包作为 .zip 文件上传。
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

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”，然后将您的 NSD 作为 .zip 文件上传。
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

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. 选择您的 NSD。验证详细信息，然后选择下一步。

5. 选择创建网络实例。初始状态为 Created。
6. 选择网络实例的 ID，然后选择实例化。
7. 选择实例化网络。
8. 使用刷新图标跟踪您的网络实例的状态。

清理

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。

AWS TNB 功能包

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

任务

- [在 AWS TNB 中创建功能包](#)
- [在 AWS TNB 中查看功能包](#)
- [从 AWS TNB 下载功能包](#)
- [从 AWS TNB 删除功能包](#)

在 AWS TNB 中创建功能包

了解如何在 AWS TNB 网络功能目录中创建功能包。创建功能包是在 TNB 中创建网络的第一步。上传功能包后，需要创建一个网络包。

Console

使用控制台创建功能包

1. 打开 AWS TNB 控制台，地址：<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，然后上传 NF 的 CSAR 包。
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

使用 AWS CLI 创建功能包

1. 使用 [create-sol-function-package](#) 命令创建一个新的功能包：

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

2. 使用 [put-sol-function-package-content](#) 命令上传功能包内容。例如：

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

在 AWS TNB 中查看功能包

了解如何查看功能包的内容。

Console

使用控制台查看功能包

1. 打开 AWS TNB 控制台，地址：<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

使用 AWS CLI 查看功能包

1. 使用 [list-sol-function-packages](#) 命令列出功能包。

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

2. 使用 [get-sol-function-package](#) 命令查看有关功能包的详细信息。

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

从 AWS TNB 下载功能包

了解如何从 AWS TNB 网络功能目录下载功能包。

Console

使用控制台下载功能包

1. 打开 AWS TNB 控制台，地址：<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

使用 AWS CLI 下载功能包

使用 [get-sol-function-package-content](#) 命令下载功能包。

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

从 AWS TNB 删除功能包

了解如何从 AWS TNB 网络功能目录删除功能包。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，地址：<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

使用 AWS CLI 删除功能包

1. 使用 [update-sol-function-package](#) 命令禁用功能包。

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

2. 使用 [delete-sol-function-package](#) 命令删除功能包。

```
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
```

AWS TNB 网络包

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用于部署它们的 AWS 基础架构。

任务

- [在 AWS TNB 中创建网络包](#)
- [在 AWS TNB 中查看网络包](#)
- [从 AWS TNB 下载网络包](#)
- [从 AWS TNB 删除网络包](#)

在 AWS TNB 中创建网络包

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

网络包在创建了功能包后创建。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，地址：<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，然后上传您的 CSAR 包。
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

使用 AWS CLI 创建网络包

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-content](#) 命令上传网络包内容。例如：

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

在 AWS TNB 中查看网络包

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，地址：<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

使用 AWS CLI 查看网络包

1. 使用 [list-sol-network-packages](#) 命令列出网络包。

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

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

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


从 AWS TNB 下载网络包

了解如何从 AWS TNB 网络服务目录下载网络包。

Console

使用控制台下载网络包

1. 打开 AWS TNB 控制台，地址：<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

使用 AWS CLI 下载网络包

- 使用 [get-sol-network-package-content](#) 命令下载网络包。

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

从 AWS TNB 删除网络包

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

Console

使用控制台删除网络包

1. 打开 AWS TNB 控制台，地址：<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

使用 AWS CLI 删除网络包

1. 使用 [update-sol-network-package](#) 命令禁用网络包。

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

2. 使用 [delete-sol-network-package](#) 命令删除网络包。

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

AWS TNB 网络实例

网络实例是在 AWS TNB 中创建的可以部署的单个网络。

任务

- [使用 AWS TNB 实例化网络实例](#)
- [在 AWS TNB 中查看网络实例](#)
- [在 AWS TNB 中更新网络实例](#)
- [在 AWS TNB 中终止和删除网络实例](#)

使用 AWS TNB 实例化网络实例

网络实例需在创建了网络包之后创建。创建网络实例后，必须对其进行实例化。在实例化网络实例时，AWS TNB 会根据网络服务描述文件中的规范部署网络功能。

Console

使用控制台创建和实例化网络实例

1. 打开 AWS TNB 控制台，地址：<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. 选择您的 NSD。验证详细信息，然后选择下一步。
6. 选择创建网络实例。
7. 选择实例化。
8. 选择实例化网络。
9. 刷新以跟踪网络实例的状态。

AWS CLI

使用 AWS CLI 创建和实例化网络实例

1. 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"
```

2. 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}$
```

在 AWS TNB 中查看网络实例

了解如何查看网络实例。

Console

使用控制台查看网络实例

1. 打开 AWS TNB 控制台，地址：<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

使用 AWS CLI 查看网络实例

1. 使用 [list-sol-network-instances](#) 命令列出网络实例。

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

2. 使用 [get-sol-network-instance](#) 命令查看有关网络实例的详细信息。

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

在 AWS TNB 中更新网络实例

了解如何更新网络实例。

Console

使用控制台更新网络实例

1. 打开 AWS TNB 控制台，地址：<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. 在功能选项卡上，选择要更新的功能实例。
5. 选择更新。
6. 输入您的更新覆盖以确认更新。
7. 选择更新。
8. 刷新以跟踪网络实例的状态。

AWS CLI

使用 CLI 更新网络实例

使用 [update-sol-network-instance](#) 命令更新网络实例。

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

在 AWS TNB 中终止和删除网络实例

要删除网络实例，实例必须处于已终止状态。

Console

使用控制台终止和删除网络实例

1. 打开 AWS TNB 控制台，地址：<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

使用 AWS CLI 终止和删除网络实例

1. 使用 [terminate-sol-network-instance](#) 命令终止网络实例。

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

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

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

AWS TNB 网络操作

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

- [查看网络操作](#)
- [取消网络操作](#)

查看网络操作

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，地址：<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

使用 AWS CLI 查看网络操作

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，地址：<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

使用 AWS CLI 取消网络操作

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}$
```


AWS TNB 的 TOSCA 参考

云应用程序拓扑和编排规范 (TOSCA) 是一种声明性语法，通信服务提供商 (CSP) 使用它来描述基于云的 Web 服务的拓扑、其组件、关系以及管理这些服务的流程。CSP 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。然后，CSP 将模板上传到 AWS TNB，由后者聚合跨 AWS 可用区建立正常运行的 5G 网络所需的资源。

目录

- [VNFD 模板](#)
- [NSD 模板](#)
- [通用节点](#)

VNFD 模板

定义虚拟网络功能描述文件 (VNFD) 模板。

语法

```
tosca_definitions_version: tnb_simple_yaml_1_0

topology_template:

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

  node\_templates:
    SampleNode1: tosca.nodes.AWS.VNF
```

拓扑模板

node_templates

TOSCA AWS 节点。可能的节点如下：

- [AWS.VNF](#)

- [AWS.Artifacts.Helm](#)

AWS.VNF

定义 AWS 虚拟网络功能 (VNF) 节点。

语法

```
tosca.nodes.AWS.VNF:
  properties:
    descriptor\_id: String
    descriptor\_version: String
    descriptor\_name: String
    provider: String
  requirements:
    helm: String
```

属性

descriptor_id

描述文件的 UUID。

必需：是

类型：字符串

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

descriptor_version

VNFD 的版本。

必需：是

类型：字符串

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

descriptor_name

描述文件的名称。

必需：是

类型：字符串

provider

VNFD 的制作方。

必需：是

类型：字符串

要求

helm

定义容器构件的 Helm 目录。这是对 [AWS.Artifacts.Helm](#) 的引用。

必需：是

类型：字符串

示例

```
SampleVNF:
  type: toska.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 Helm 节点。

语法

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

属性

implementation

CSAR 包中包含 Helm 图表的本地目录。

必需：是

类型：字符串

示例

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

NSD 模板

定义网络服务描述文件 (NSD) 模板。

语法

```
tosca_definitions_version: tnb_simple_yaml_1_0

vnfds:
  - descriptor\_id: String
    namespace: String

topology_template:

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

  node\_templates:
    SampleNode1: tosca.nodes.AWS.NS
```

使用已定义的参数

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

以下示例演示了如何定义和使用参数：

```
tosca_definitions_version: tnb_simple_yaml_1_0

topology_template:

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

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

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

VNFD 导入

descriptor_id

描述文件的 UUID。

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 节点有：

- [AWS.NS](#)
- [AWS.Compute.EKS](#)
- [AWS.compute.eks。 AuthRole](#)
- [AWS.compute.eks ManagedNode](#)
- [AWS.compute.eks SelfManagedNode](#)
- [AWS.Compute。 PlacementGroup](#)
- [AWS.Compute。 UserData](#)
- [AWS. 联网。 SecurityGroup](#)
- [AWS. 联网。 SecurityGroupEgressRule](#)
- [AWS. 联网。 SecurityGroupIngressRule](#)
- [AWS.Resource.Import](#)
- [AWS.Networking.ENI](#)
- [AWS.HookExecution](#)
- [AWS. 联网。 InternetGateway](#)
- [AWS. 联网。 RouteTable](#)
- [AWS.Networking.Subnet](#)
- [AWS.Deployment.VNFDeployment](#)
- [AWS.Networking.VPC](#)
- [AWS.Networking.NATGateway](#)
- [AWS.Networking.Route](#)

AWS.NS

定义 AWS 网络服务 (NS) 节点。

语法

```
tosca.nodes.AWS.NS:
  properties:
    descriptor\_id: String
    descriptor\_version: String
    descriptor\_name: String
```

属性

descriptor_id

描述文件的 UUID。

必需：是

类型：字符串

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

descriptor_version

NSD 的版本。

必需：是

类型：字符串

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

descriptor_name

描述文件的名称。

必需：是

类型：字符串

示例

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

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

AWS.Compute.EKS

提供集群的名称、所需的 Kubernetes 版本以及允许 Kubernetes 控制平面管理 NF 所需资源的角色。AWS Multus 容器网络接口 (CNI) 插件已启用。您可以连接多个网络接口，并将高级网络配置应用于基于 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) 使用的属性。

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

enabled

指示是否启用默认 Multus 功能。

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

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

enabled

指示是否安装默认 Amazon EBS CSI 驱动程序。

必需：否

类型：布尔值

version

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

必需：否

类型：字符串

属性

version

集群的 Kubernetes 版本。 AWS Telco Network Builder 支持 Kubernetes 版本 1.23 到 1.29。

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

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: toska.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

```
toska.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-auth ConfigMap 的 IAM 角色的映射列表。

arn

IAM 角色的 ARN。

必需：是

类型：字符串

groups

要分配给 arn 中定义的角色角色的 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.eks ManagedNode

AWS TNB 支持 EKS 托管节点组，以自动为亚马逊 EKS Kubernetes 集群配置节点（Amazon EC2 实例）和进行生命周期管理。要创建 EKS 节点组，您必须通过提供亚马逊机器映像（AMI）的 ID 或 AMI 类型，为集群 Worker 节点选择 AMI。您还可为您的节点组提供用于 SSH 访问的 Amazon EC2 密钥对，以及扩缩属性。您的节点组必须与 EKS 集群相关联。您必须为 Worker 节点提供子网。

或者，您可以将安全组、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 Amazon EKS 托管节点组计算参数的属性，例如 Amazon EC2 实例类型和 Amazon EC2 实例 AMI。

ami_type

亚马逊 EKS 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

AMI 的 ID。

必需：否

类型：字符串

Note

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

instance_types

实例大小。

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 AWS TNB 会默认加密 Amazon EBS 根卷。

必需：否

默认：True


类型：布尔值

root_volume_encryption_key_arn

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

必需：否

类型：字符串

 Note

- 如果root_volume_encryption为 false，则不包含root_volume_encryption_key_arn。
- AWS TNB 支持对亚马逊 EBS 支持的 AMI 进行根卷加密。
- 如果 AMI 的根卷已加密，则必须包括root_volume_encryption_key_arn以便 AWS TNB 重新加密根卷。
- 如果 AMI 的根卷未加密，AWS TNB 将使用root_volume_encryption_key_arn对根卷进行加密。

如果不包括root_volume_encryption_key_arn，AWS TNB 将使用提供的默认密钥 AWS Key Management Service 对根卷进行加密。

- AWS TNB 不会解密加密的 AMI。

scaling

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

desired_size

此中的实例数量 NodeGroup。

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](#)，如果您将该[multus_role](#)属性包含在 [aws.compute.eks](#) 节点中，AWS TNB 将从该[multus](#)属性获得与 ENI 相关的权限。否则，AWS TNB 将从 [node_role](#) 属性中获取与 ENI 相关的权限。

必需：否

类型：列表

security_groups

一个 [AWS.Networking SecurityGroup](#)节点。

必需：否

类型：列表

placement_group

一个 [tosca.nodes. AWS.Compute. PlacementGroup](#)节点。

必需：否

类型：字符串

user_data

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

必需：否

类型：字符串

labels

节点标签列表。节点标签必须有名称和价值。使用以下标准创建标签：

- 名称和价值必须用分隔=。
- 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：[-, _, ., *, ?]
- 名称和价值必须以字母数字?、或*字符开头和结尾。

例如，myLabelName1=*NodeLabelValue1

必需：否

类型：列表

示例

```
SampleEKSMangedNode:
  type: tosa.nodes.AWS.Compute.EKSMangedNode
  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.eks SelfManagedNode

AWS TNB 支持 Amazon EKS 自我管理节点，以自动为亚马逊 EKS Kubernetes 集群配置节点（亚马逊 EC2 实例）和进行生命周期管理。要创建 Amazon EKS 节点组，您必须通过提供亚马逊机器映像（AMI）的 ID，为集群 Worker 节点选择 AMI。您也可以选择为 SSH 访问提供 Amazon EC2 密钥对。您还必须提供实例类型、所需大小，以及最小和最大大小。您的节点组必须与 Amazon EKS 集群相关联。您必须为 Worker 节点提供子网。

或者，您可以将安全组、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 Amazon EKS 自管理节点计算参数的属性，例如 Amazon EC2 实例类型和 Amazon EC2 实例 AMI。

ami_id

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

必需：是

类型：字符串

instance_type

实例大小。

必需：是

类型：字符串

key_pair

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 AWS TNB 会默认加密 Amazon EBS 根卷。

必需：否

默认：True

类型：布尔值

root_volume_encryption_key_arn

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

必需：否

类型：字符串

Note

- 如果 `root_volume_encryption` 为 `false`，则不包含 `root_volume_encryption_key_arn`。
- AWS TNB 支持对亚马逊 EBS 支持的 AMI 进行根卷加密。
- 如果 AMI 的根卷已加密，则必须包括 `root_volume_encryption_key_arn` 以便 AWS TNB 重新加密根卷。
- 如果 AMI 的根卷未加密，AWS TNB 将使用 `root_volume_encryption_key_arn` 对根卷进行加密。

如果不包括 `root_volume_encryption_key_arn`，AWS TNB AWS Managed Services 将使用加密根卷。

- AWS TNB 不会解密加密的 AMI。

scaling

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

`desired_size`

此中的实例数量 NodeGroup。

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](#)，如果您将该[multus_role](#)属性包含在 [aws.compute.eks](#) 节点中，AWS TNB 将从该[multus](#)属性获得与 ENI 相关的权限。否则，AWS TNB 将从 [node_role](#) 属性中获取与 ENI 相关的权限。

必需：否

类型：列表

security_groups

一个 [AWS.Networking SecurityGroup](#)节点。

必需：否

类型：列表

placement_group

一个 [tosca.nodes. AWS.Compute. PlacementGroup](#)节点。

必需：否

类型：字符串

user_data

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

必需：否

类型：字符串

labels

节点标签列表。节点标签必须有名称和值。使用以下标准创建标签：

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

例如，`myLabelName1=*NodeLabelValue1`

必需：否

类型：列表

示例

```
SampleEKSSelfManagedNode:
  type: toasca.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。

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

语法

```
tosca.nodes.AWS.Compute.PlacementGroup
properties:
  strategy: String
  partition\_count: Integer
  tags: List
```

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 – 将一个可用区内靠近的实例打包在一起。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 (HPC) 应用程序的典型特征。
- PARTITION – 将实例分布在不同的逻辑分区上，以便一个分区中的实例组不会与不同分区中的实例组共享相同的基础硬件。该策略通常为大型分布式和重复的工作负载所使用，例如，Hadoop、Cassandra 和 Kafka。
- SPREAD_RACK – 将一小组实例严格放置在不同的基础硬件上，以减少相关的故障。
- SPREAD_HOST – 只能与 Outpost 置放群组结合使用。将一小组实例严格放置在不同的基础硬件上以减少相关的故障。

partition_count

分区的数量。

必需：仅当 strategy 设置为 PARTITION 时才必需。

类型：整数

可能的值：1 | 2 | 3 | 4 | 5 | 6 | 7

tags

可以附加到置放群组资源的标签。

必需：否

类型：列表

示例

```
ExamplePlacementGroup:
  type: toscanodes.AWS.Compute.PlacementGroup
  properties:
    strategy: "PARTITION"
    partition_count: 5
    tags:
```

```
- tag_key=tag_value
```

AWS.Compute。 UserData

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

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

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

Note

- AWS TNB 支持大小不超过 7 KB 的用户数据脚本。
- 由 AWS 于 TNB 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: toasca.nodes.AWS.Compute.UserData
  properties:
    content_type: "text/x-shellscript"
    implementation: "./scripts/customUserData.sh"
```

AWS. 联网。 SecurityGroup

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

语法

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

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

AWS. 联网。 SecurityGroupEgressRule

AWS TNB 支持安全组出站规则，以自动配置可附加到 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 格式的 IPv4 地址范围。您必须指定允许出口流量的 CIDR 范围。

必需：否

类型：字符串

cidr_ipv6

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

必需：否

类型：字符串

description

出口 (出站) 安全组规则的描述。最多可以使用 255 个字符来描述该规则。

必需：否

类型：字符串

destination_prefix_list

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

必需：否

类型：字符串

from_port

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

必需：否

类型：整数

ip_protocol

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

必需：是

类型：字符串

to_port

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

必需：否

类型：整数

要求

security_group

要添加此规则的安全组的 ID。

必需：是

类型：字符串

destination_security_group

允许出口流量进入的目标安全组的 ID 或 TOSCA 参考。

必需：否

类型：字符串

示例

```
SampleSecurityGroupEgressRule:
  type: toscanodes.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. 联网。 SecurityGroupIngressRule

AWS TNB 支持安全组入口规则，以自动配置可附加到 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 格式的 IPv4 地址范围。您必须指定允许入口流量的 CIDR 范围。

必需：否

类型：字符串

cidr_ipv6

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

必需：否

类型：字符串

description

入口 (入站) 安全组规则的描述。最多可以使用 255 个字符来描述该规则。

必需：否

类型：字符串

source_prefix_list

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

必需：否

类型：字符串

from_port

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

必需：否

类型：整数

ip_protocol

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

必需：是

类型：字符串

to_port

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

必需：否

类型：整数

要求

security_group

要添加此规则的安全组的 ID。

必需：是

类型：字符串

source_security_group

源安全组的 ID 或 TOSCA 参考，将允许来自该安全组的入口流量。

必需：否

类型：字符串

示例

```
SampleSecurityGroupIngressRule:
  type: toska.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 AWS TNB 的资源类型。

必需：否

类型：列表

resource_id

导入到 AWS TNB 的资源 ID。

必需：否

类型：列表

示例

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

AWS.Networking.ENI

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

语法

```
toska.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。

允许的值：真、假

默认：True

必需：否

类型：布尔值

tags

要附加到资源的标签。

必需：否

类型：列表

要求

subnet

一个 [AWS.Networking.Subnet](#) 节点。

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: toska.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: toasca.nodes.AWS.HookExecution  
  requirements:  
    definition: SampleHookScript
```

```
vpc: SampleVPC
```

AWS. 联网。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

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

`dest_cidr`

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

必需：如果包含 `ipv6_dest_cidr` 属性，则为“否”。

类型：字符串

`ipv6_dest_cidr`

用于目标匹配的 IPv6 CIDR 块。

必需：如果包含 `dest_cidr` 属性，则为“否”。

类型：字符串

属性

tags

要附加到资源的标签。

必需：否

类型：列表

egress_only

一个特定于 IPv6 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: toscanodes.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: toska.nodes.AWS.Networking.InternetGateway
  properties:
    egress_only: true
  capabilities:
    routing:
      properties:
        ipv6_dest_cidr: "::/0"
  requirements:
    route_table: Free5GCPriateRouteTable
    vpc: Free5GCVPC
```

AWS. 联网。 RouteTable

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

语法

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

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

AWS.Networking.Subnet

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

语法

```
tosca.nodes.AWS.Networking.Subnet:
  properties:
    type: String
    availability\_zone: String
    cidr\_block: String
    ipv6\_cidr\_block: String
    ipv6\_cidr\_block\_suffix: String
    outpost\_arn: String
    tags: List
  requirements:
    vpc: String
    route\_table: String
```

属性

type

指示在此子网中启动的实例是否接收公有 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 Local Zones us-west-2-lax-1a。

必需：是

类型：字符串

cidr_block

子网的 CIDR 块。

必需：否

类型：字符串

ipv6_cidr_block

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

必需：否

类型：字符串

ipv6_cidr_block_suffix

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

如果包含此属性，请不要包含 ipv6_cidr_block。

必需：否

类型：字符串

outpost_arn

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

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

必需：否

类型：字符串

tags

要附加到资源的标签。

必需：否

类型：列表

要求

vpc

一个 [AWS.Networking.VPC](#) 节点。

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

```

AWS.Deployment.VNFDeployment

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

语法

```

toska.nodes.AWS.Deployment.VNFDeployment:
  requirements:
    deployment: String
    cluster: String
    vnfs: List
  interfaces:
    Hook:
      pre\_create: String
      post\_create: String

```

要求

deployment

一个 [AWS.Deployment.VNFDeployment](#) 节点。

必需：否

类型：字符串

cluster

一个 [AWS.Compute.EKS](#) 节点。

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.Networking.VPC

您必须为虚拟私有云 (VPC) 指定 CIDR 块。

语法

```
tosca.nodes.AWS.Networking.VPC:
  properties:
    cidr\_block: String
    ipv6\_cidr\_block: String
    dns\_support: String
    tags: List
```

属性

cidr_block

VPC 的 IPv4 网络范围，采用 CIDR 表示法。

必需：是

类型：字符串

ipv6_cidr_block

用于创建 VPC 的 IPv6 CIDR 块。

允许的值：AMAZON_PROVIDED

必需：否

类型：字符串

dns_support

指明在 VPC 内启动的实例是否可获得 DNS 主机名称。

必需：否

类型：布尔值

默认：false

tags

要附加到资源的标签。

必需：否

类型：列表

示例

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

AWS.Networking.NATGateway

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

语法

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

```
requirements:
  subnet: String
  internet\_gateway: String
properties:
  type: String
  eip\_allocation\_id: String
  tags: List
```

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

```
Free5GNatGateway01:
  type: toska.nodes.AWS.Networking.NATGateway
  requirements:
    subnet: Free5GSubnet01
    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

到目标资源的目标 IPv4 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](#) 节点参考。

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: toasca.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

定义要在 NSD 和 VNFD 中使用的节点。

- [AWS.HookDefinition.Bash](#)

AWS.HookDefinition.Bash

用 bash 定义 AWS HookDefinition。

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

示例：A123-45xYz=Example_789

必需：否

类型：列表

execution_role

挂钩执行的角色。

必需：是

类型：字符串

示例

```
SampleHookScript:
  type: tosa.nodes.AWS.HookDefinition.Bash
  properties:
    implementation: "./hooks/myhook.sh"
    environment_variables:
      - "variable01=value01"
      - "variable02=value02"
    execution_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleHookPermission"
```

AWS 电信网络生成器中的安全性

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

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

- 云安全 — AWS 负责保护在云中运行 AWS 服务的基础架构 AWS Cloud。AWS 还为您提供可以安全使用的服务。作为[AWS 合规计划](#)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、您的公司的要求以及适用的法律法规。

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

内容

- [AWS TNB 中的数据保护](#)
- [AWS TNB 的身份和访问管理](#)
- [AWS TNB 的合规性验证](#)
- [AWS TNB 中的弹性](#)
- [AWS TNB 中的基础设施安全](#)
- [IMDS 版本](#)

AWS TNB 中的数据保护

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设施上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅[数据隐私常见问题](#)。有关欧洲数据保护的信息，请参阅 AWS 安全性博客 上的 [AWS 责任共担模式和 GDPR](#) 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。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 与资源通信。AWS 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅 [《美国联邦信息处理标准 \(FIPS \) 第 140-2 版》](#)。

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 AWS 服务使用控制台、API 或 AWS SDK 与 AWS TNB 或其他人合作时。AWS CLI 在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

数据处理

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

静态加密

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

传输中加密

AWS TNB 使用传输层安全 (TLS) 1.2 保护传输中的所有数据。

您负责加密您的模拟代理与其客户端之间的数据。

互连网络流量隐私

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。

AWS TNB 的身份和访问管理

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

内容

- [受众](#)
- [使用身份进行身份验证](#)
- [使用策略管理访问](#)
- [AWS 电信网络构建器如何与 IAM 配合使用](#)
- [AWS 电信网络构建器基于身份的策略示例](#)
- [排除 AWS Telco Network Builder 身份](#)

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您在 AWS TNB 中所做的工作。

服务用户-如果您使用 AWS TNB 服务完成工作，则您的管理员会为您提供所需的凭证和权限。当您使用更多的 AWS TNB 功能来完成工作时，您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 AWS TNB 中的某个功能，请参阅 [排除 AWS Telco Network Builder 身份](#)。

服务管理员 — 如果您负责公司的 AWS TNB 资源，则可能拥有对 AWS TNB 的完全访问权限。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 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要详细了解您的公司如何将 IAM 与 AWS TNB 结合使用，请参阅[AWS 电信网络构建器如何与 IAM 配合使用](#)。

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策略来管理 AWS TNB 访问权限。要查看您可以在 IAM 中使用的基于身份的 AWS TNB 策略示例，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 IAM 用户身份或通过担任 AWS 账户根用户任 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 Identity Center) 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，您的管理员以前使用 IAM 角色设置了身份联合验证。当您使用联合访问 AWS 时，您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[签署 AWS API 请求](#)。

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《AWS IAM Identity Center 用户指南》中的[多重身份验证](#)和《IAM 用户指南》中的[在 AWS 中使用多重身份验证 \(MFA \)](#)。

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。有关要求您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的[需要根用户凭证的任务](#)。

联合身份

作为最佳实践，要求人类用户 (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 Center 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 IAM Identity 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 Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的[什么是 IAM Identity Center？](#)。

IAM 用户和群组

[IAM 用户](#)是您 AWS 账户内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，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[对于需要长期凭证的使用场景定期轮换访问密钥](#)。

[IAM 组](#)是一个指定一组 IAM 用户的身份。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，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人或应用程序关联，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[何时创建 IAM 用户（而不是角色）](#)。

IAM 角色

[IAM 角色](#)是您内部具有特定权限 AWS 账户的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过[切换角色在中临时担任 IAM 角色](#)。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的[使用 IAM 角色](#)。

具有临时凭证的 IAM 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。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。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 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。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）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的[IAM 角色与基于资源的策略有何不同](#)。

- 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，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 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以及 AWS 服务 向下游服务发出请求的请求。AWS 服务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 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅[转发访问会话](#)。
- 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 [IAM 角色](#)。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的[创建向 AWS 服务委派权限的角色](#)。
- 服务相关角色-服务相关角色是一种与服务相关联的服务角色。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 EC2 上运行的应用程序 — 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。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的[使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限](#)。

要了解是使用 IAM 角色还是 IAM 用户，请参阅《IAM 用户指南》中的[何时创建 IAM 角色 \(而不是用户\)](#)。

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 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅《IAM 用户指南》中的[JSON 策略概览](#)。

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。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关于您使用哪种方法执行操作。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 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（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[创建 IAM 策略](#)。

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和内联策略之间进行选择，请参阅《IAM 用户指南》中的[在托管策略与内联策略之间进行选择](#)。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service (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中使用 IAM 中的 AWS 托管策略。

访问控制列表 (ACL)

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

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。AWS WAF 要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的[访问控制列表 \(ACL\) 概览](#)。

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 - 权限边界是一个高级功能，用于设置基于身份的策略可以为 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 (SCP)-SCP 是 JSON 策略，用于指定组织或组织单位 (OU) 的最大权限。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。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 [SCP 的工作原理](#)。
- 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 电信网络构建器如何与 IAM 配合使用

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

可与 AWS Telco Network Builder 配合使用的 I

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
ACL	否
ABAC (策略中的标签)	是

IAM 功能	AWS TNB 支持
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 AWS TNB 和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的[AWS 服务](#)。

TNB 基于身份的策略 AWS

支持基于身份的策略	是
-----------	---

基于身份的策略是可附加到身份（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[创建 IAM 策略](#)。

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

TNB 基于身份的政策示例 AWS

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](#)

TNB 内部 AWS 基于资源的政策

支持基于资源的策略	否
-----------	---

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service (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。

要启用跨账户存取，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。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》中的[IAM 角色与基于资源的策略有何不同](#)。

AWS TNB的政策行动

支持策略操作	是
--------	---

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

JSON 策略的 Action 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限操作。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 T [AWS elco Network Builder 定义的操作](#)。

AWS TNB 中的策略操作在操作前使用以下前缀：

```
tnb
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

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

您也可以使用通配符（*）指定多个操作。例如，要指定以单词 List 开头的操作，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](#)

AWS TNB 的政策资源

支持策略资源 是

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

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 [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 ARN 的列表，请参阅《服务授权参考》中的 [AWS Telco Network Builder 定义的资源](#)。要了解您可以使用哪些操作来指定每种资源的 ARN，请参阅 [AWS Telco Network Builder 定义的操作](#)。

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

AWS TNB 的策略条件密钥

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，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。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密钥。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的 [AWS 全局条件上下文密钥](#)。

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

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

TNB 中的 AWS ACL

支持 ACL	否
--------	---

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

ABAC 和 TNB AWS

支持 ABAC (策略中的标签)	是
--------------------	---

基于属性的访问控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在中 AWS，这些属性称为标签。您可以向 IAM 实体 (用户或角色) 和许多 AWS 资源附加标签。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的 [条件元素](#) 中提供标签信息。

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

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的[什么是 ABAC ?](#)。要查看设置 ABAC 步骤的教程,请参阅《IAM 用户指南》中的[使用基于属性的访问权限控制 \(ABAC \)](#)。

在 AWS TNB 中使用临时证书

支持临时凭证 是

当你使用临时凭证登录时,有些 AWS 服务 不起作用。有关更多信息,包括哪些 AWS 服务 适用于临时证书,请参阅 IAM 用户指南中的[AWS 服务与 IAM 配合使用的信息](#)。

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《IAM 用户指南》中的[切换到角色 \(控制台 \)](#)。

您可以使用 AWS CLI 或 AWS API 手动创建临时证书。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 [IAM 中的临时安全凭证](#)。

TNB 的跨服务主体 AWS 权限

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以及 AWS 服务 向下游服务发出请求的请求。AWS 服务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 FAS 请求。在这种情况下,您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情,请参阅[转发访问会话](#)。

AWS TNB 的服务角色

支持服务角色 否

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

TNB 的 AWS 服务相关角色

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 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 AWS API 执行任务。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅 IAM 用户指南中的 [创建 IAM 策略](#)。

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

内容

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

策略最佳实践

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，请遵循以下准则和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的 AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 [AWS 托管策略](#) 或 [工作职能的 AWS 托管策略](#)。

- 应用最低权限 – 在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 [IAM 中的策略和权限](#)。
- 使用 IAM 策略中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 AWS CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 [IAM JSON 策略元素：条件](#)。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 [IAM Access Analyzer 策略验证](#)。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的 [配置受 MFA 保护的 API 访问](#)。

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

使用 AWS TNB 控制台

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

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

服务角色策略示例

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

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

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

AWS TNB 服务角色

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

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

以下代码显示了 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:UntagResource",
        "eks:DescribeCluster",
        "eks:ListNodegroups",
    ]
}

```

```

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

```

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

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 亚马逊 EKS 集群的 TNB 服务角色

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

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

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

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

- [AWS::IAM::Role](#)
- [选择堆栈模板](#)

AWS 亚马逊 EKS 节点组的 TNB 服务角色

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

以下示例显示了为 Amazon EKS 节点组策略创建 AWS TNB 服务角色的 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::Role](#)
- [选择堆栈模板](#)

AWS Multus 的 TNB 服务角色

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

以下示例显示了为 Multus 策略创建 AWS TNB 服务角色的 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::Role](#)
- [选择堆栈模板](#)

AWS 生命周期挂钩策略的 TNB 服务角色

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

Note

您的生命周期挂钩策略应基于您的生命周期挂钩尝试执行的操作。

以下示例显示了一个为生命周期挂钩策略创建 AWS TNB 服务角色的 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::Role](#)
- [选择堆栈模板](#)

允许用户查看他们自己的权限

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


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupForUser",
        "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 Telco Network Builder 身份

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

问题

- [我无权在 AWS TNB 中执行任何操作](#)
- [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 AWS TNB 中执行任何操作

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

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

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

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam : PassRole

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

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

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

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

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

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人 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 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 AWS TNB 是否支持这些功能，请参阅 [AWS 电信网络构建器如何与 IAM 配合使用](#)。

- 要了解如何提供对您拥有的资源的访问权限 AWS 账户，请参阅 [IAM 用户指南中的向您拥有 AWS 账户的另一个 IAM 用户提供访问权限](#)。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 [IAM 用户指南中的向第三方提供访问权限](#)。AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的 [为经过外部身份验证的用户 \(身份联合验证\) 提供访问权限](#)。
- 要了解使用角色和基于资源的策略进行跨账户存取之间的差别，请参阅《IAM 用户指南》中的 [IAM 角色与基于资源的策略有何不同](#)。

AWS TNB 的合规性验证

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 上构建 HIPAA 安全与合规性](#) — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

Note

并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅 [符合 HIPAA 要求的服务参考](#)。

- [AWS 合规资源](#) — 此工作簿和指南集可能适用于您的行业和所在地区。
- [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 提供了您内部安全状态的全面视图 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 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。
- [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。

AWS TNB 中的弹性

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。

AWS TNB 中的基础设施安全

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

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

- 传输层安全性协议 (TLS)。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密 (PFS) 的密码套件，例如 DHE (临时 Diffie-Hellman) 或 ECDHE (临时椭圆曲线 Diffie-Hellman)。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 委托人关联的秘密访问密钥来对请求进行签名。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 AWS TNB 的组件，包括：

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

Important

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

网络连接安全模型

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

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

您通过 AWS Management Console、AWS Command Line Interface (AWS CLI) 和 AWS SDK 与 AWS TNB 的连接已加密。

IMDS 版本

AWS TNB 支持利用实例元数据服务版本 2 (imdsv2) (一种面向会话的方法) 的实例。IMDSv2 包含比 IMDSv1 更高的安全性。有关更多信息，请参阅 [Add defense in depth against open firewalls](#),

[reverse proxies, and SSRF vulnerabilities with enhancements to the Amazon EC2 Instance Metadata Service](#)。

启动实例时，必须使用 IMDSv2。有关 imdsv2 的更多信息，请参阅亚马逊 EC2 用户指南[中的使用 imds v2](#)。

监控 AWS TNB

监控是保持 AWS TNB 和您的其它 AWS 解决方案的可靠性、可用性和性能的重要方面。AWS 提供了 AWS CloudTrail 来监控 AWS TNB、在出现错误时进行报告并适时自动采取措施。

可以使用 CloudTrail 捕获有关向 AWS API 发出的调用的详细信息。您可以将这些调用作为日志文件存储在 Amazon S3 中。可以使用这些 CloudTrail 日志确定已发出的调用、从中发出调用的源 IP 地址、调用的发出方、调用的发出时间等信息。

CloudTrail 日志包含有关 AWS TNB 的 API 操作调用的信息。这些日志还包含来自 Amazon EC2 和 Amazon EBS 等服务的 API 操作调用信息。

使用 AWS CloudTrail 记录 AWS 电信网络构建器 API 调用的日志

AWS 电信网络构建器已与 AWS CloudTrail 集成，后者作为一项服务提供 AWS TNB 中由用户、角色或 AWS 服务所执行的操作的记录。CloudTrail 将 AWS TNB 的所有 API 调用作为事件捕获。捕获的调用包含来自 AWS TNB 控制台和代码的 AWS TNB API 操作调用。如果您创建跟踪记录，则可以使 CloudTrail 事件（包括 AWS TNB 的事件）持续传送到 Amazon S3 桶。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台中的事件历史记录中查看最新事件。使用 CloudTrail 收集的信息，您可以确定向 AWS TNB 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其它详细信息。

要了解有关 CloudTrail 的更多信息，请参阅 [《AWS CloudTrail User Guide》](#)。

CloudTrail 中的 AWS TNB 信息

在您创建 AWS 账户时，将在该账户上启用 CloudTrail。当 AWS TNB 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其它 AWS 服务事件一同保存在事件历史记录中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅 [Viewing events with CloudTrail Event history](#)。

要实现 AWS 账户中的事件（包括 AWS TNB 的事件）的持续记录，请创建跟踪记录。通过跟踪记录，CloudTrail 可将日志文件传送到 Amazon S3 桶。预设情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 AWS 区域。此跟踪记录在 AWS 分区中记录所有区域中的事件，并将日志文件传送到您指定的 Amazon S3 桶。此外，您可以配置其它 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅下列内容：

- [Overview for creating a trail](#)

- [CloudTrail supported services and integrations](#)
- [Configuring Amazon SNS notifications for CloudTrail](#)
- [Receiving CloudTrail log files from multiple regions](#) 和 [Receiving CloudTrail log files from multiple accounts](#)

CloudTrail 记录所有 AWS TNB 操作，且 [AWS 电信网络构建器 API Reference](#) 中也介绍了这些操作。例如，对 `CreateSolFunctionPackage`、`CreateSolNetworkInstance` 和 `CreateSolNetworkPackage` 操作的调用会在 CloudTrail 日志文件中生成条目。

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

- 请求是使用根用户凭证还是 AWS Identity and Access Management (IAM) 用户凭证发出的。
- 请求是使用角色还是联合身份用户的临时安全凭证发出的。
- 请求是否由其它 AWS 服务发出。

有关更多信息，请参阅 [CloudTrail userIdentity element](#)。

了解 AWS TNB 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `CreateSolFunctionPackage` 操作。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE:example",
    "arn": "arn:aws:sts::111222333444:assumed-role/example/user",
    "accountId": "111222333444",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
```



```
        "arn": "arn:aws:iam::111222333444:role/example",
        "accountId": "111222333444",
        "userName": "example"
    },
    "webIdFederationData": {},
    "attributes": {
        "creationDate": "2023-02-02T01:42:39Z",
        "mfaAuthenticated": "false"
    }
}
},
"eventTime": "2023-02-02T01:43:17Z",
"eventSource": "tnb.amazonaws.com",
"eventName": "CreateSolFunctionPackage",
"awsRegion": "us-east-1",
"sourceIPAddress": "XXX.XXX.XXX.XXX",
"userAgent": "userAgent",
"requestParameters": null,
"responseElements": {
    "vnfPkgArn": "arn:aws:tnb:us-east-1:111222333444:function-package/
fp-12345678abcEXAMPLE",
    "id": "fp-12345678abcEXAMPLE",
    "operationalState": "DISABLED",
    "usageState": "NOT_IN_USE",
    "onboardingState": "CREATED"
},
"requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
"eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
"readOnly": false,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111222333444",
"eventCategory": "Management"
}
```

AWS TNB 部署任务

了解部署任务以有效监控部署并更快地采取行动。

下表列出了 T AWS NB 部署任务：

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

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
NetworkServiceInstantiation	名字没有变化	初始化网络服务。
SelfManagedNodesConfiguration	名字没有变化	使用 Amazon EKS 和 Kubernetes 控制面板启动自管理节点。

AWS 电信网络构建器服务限额

服务限额（也称为限制）是您的 AWS 账户使用的服务资源或操作的最大数量。有关更多信息，请参阅《Amazon Web Services 一般参考》中的 [AWS Service Quotas](#)。

以下是 AWS TNB 的服务限额。

名称	默认值	可调整	描述
持续并发执行的网络服务操作数量	每个支持的区域： 40 个	是	一个区域内可持续并发执行的网络服务操作的最大数量。
功能包数量	每个支持的区域： 200 个	是	一个区域内功能包的最大数量。
网络包数量	每个支持的区域： 40 个	是	一个区域内网络包的最大数量。
网络服务实例数量	每个支持的区域： 800 个	是	一个区域内网络服务实例的最大数量。

AWS TNB 用户指南的文档历史记录

下表描述了 AWS TNB 的文档版本。

变更	说明	日期
现有任务的新任务和新任务名称	有一项新任务可用。为清楚起见，自 2024 年 3 月 7 日起，一些现有任务有了新的名称。	2024 年 5 月 7 日
适用于集群的 Kubernetes 版本	AWS TNB 现在支持 Kubernetes 版本 1.29 来创建亚马逊 EKS 集群。	2024 年 4 月 10 日
Support 对网络接口的支持 security_groups	您可以将安全组附加到 AWS.networking.eni 节点。	2024 年 4 月 2 日
支持 Amazon EBS 根卷加密	您可以为亚马逊 EBS 根卷启用亚马逊 EBS 加密。 要启用 ，请在 aws.compute.eks 或 aws.Compute.ek ManagedNodes 节点中添加属性。 SelfManagedNode	2024 年 4 月 2 日
对节点的 Support labels	您可以在 aws.compute.eks 或 aws.Compute.ek ManagedNodes 节点中将节点标签附加到你的节点组。 SelfManagedNode	2024 年 3 月 19 日
Support 对网络接口的支持 source_dest_check	您可以通过 .networking.eni 节点指示是要启用还是禁用网络接口源/目标检查。	2024 年 1 月 25 日
对带有自定义用户数据的 Amazon EC2 实例的支持	您可以通过 AWS.Compute 启动带有自定义用户数据的 Amazon EC2 实例。 UserData 节点。	2024 年 1 月 16 日

对安全组的支持	AWS TNB 允许您导入安全组 AWS 资源。	2024 年 1 月 8 日
更新了 <code>network_interfaces</code> 的描述	当该 <code>network_interfaces</code> 属性包含在 <code>aws.compute.eks ManagedNode</code> 或 <code>aws.Compute.eks SelfManagedNode</code> 节点中时，AWS TNB 将从该属性（如果有）或该属性获得与 ENI 相关的权限。 <code>multus_role</code> <code>node_role</code>	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 <code>access</code> 属性设置为 <code>PRIVATE</code> 。	2023 年 12 月 11 日
适用于集群的 Kubernetes 版本	AWS TNB 现在支持 Kubernetes 版本 1.28 来创建亚马逊 EKS 集群。	2023 年 12 月 11 日
AWS TNB 支持置放群组	为 <code>AWS.Compute.EKSManagedNode</code> 和 <code>AWS.Compute.EKSSelfManagedNode</code> 节点定义添加了置放群组。	2023 年 12 月 11 日

[AWS TNB 增加了对 IPv6 的支持](#)

AWS TNB 现在支持使用 IPv6 基础设施创建网络实例。检查节点 [AWS.networking.vpc](#)、[.Networking.Subnet](#)、[AWS.Networking.InternetGateway](#)、[AWS.联网.SecurityGroupIngressRule](#)、[AWS.联网.SecurityGroupEgressRule](#)，以及用于 IPv6 配置的 [AWS.compute.eks](#)。我们还为 NAT64 配置添加了节点 [AWS.Networking.NATGateway](#) 和 [AWS.Networking.Route](#)。我们更新了 AWS Amazon EKS 节点组的 AWS TNB 服务角色和 TNB 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

[向 AWS TNB 服务角色策略添加了权限](#)

我们向 Amazon S3 的 AWS TNB 服务角色策略添加了权限 AWS CloudFormation，并允许启用基础设施实例化。

2023 年 10 月 23 日

[AWS TNB 在更多地区推出](#)

AWS TNB 现已在亚太地区（首尔）、加拿大（中部）、欧洲（西班牙）、欧洲（斯德哥尔摩）和南美洲（圣保罗）地区推出。

2023 年 9 月 27 日

[.compute.eks 的标签 SelfManagedNode](#)

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 imdsv2 的实例	启动实例时，必须使用 IMDSv2。	2023 年 8 月 14 日
更新了权限 MultusRoleInlinePolicy	MultusRoleInlinePolicy 现在包括 ec2:DeleteNetworkInterface 权限。	2023 年 8 月 7 日
适用于集群的 Kubernetes 版本	AWS TNB 现在支持 Kubernetes 版本 1.27 来创建亚马逊 EKS 集群。	2023 年 7 月 25 日
AWS.compute.eks。AuthRole	AWS TNB 支持 AuthRole 允许您向 Amazon EKS 集群添加 IAM 角色，aws-authConfigMap 以使用户可以使用 IAM 角色访问亚马逊 EKS 集群。	2023 年 7 月 19 日
AWS TNB 支持安全组。	添加了 AWS.Networking.SecurityGroup ， AWS.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 1.22 到 1.26 版本来创建亚马逊 EKS 集群。AWS TNB 不再支持 Kubernetes 版本 1.21。	2023 年 5 月 11 日
AWS.compute.eks SelfManagedNode	您可以在区域内、Local 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。