Benutzer-Leitfaden

AWS Telco Network Builder

AWS Telco Network Builder: Benutzer-Leitfaden

Table of Contents

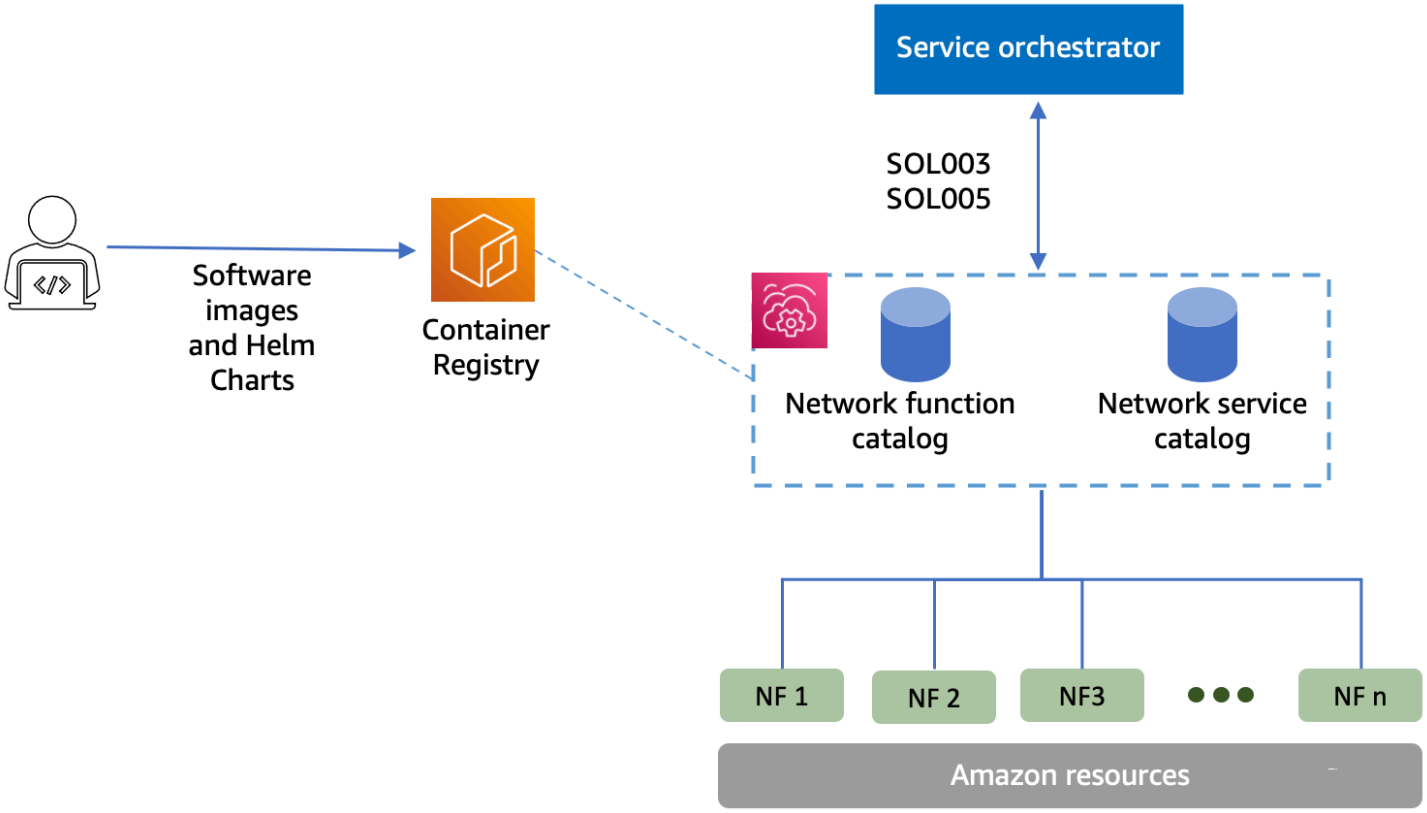
Was ist AWS Telco Network Builder?

AWS Telco Network Builder (AWS TNB) ist ein AWS Dienst, der Anbietern von Kommunikationsdiensten (CSPs) eine effiziente Möglichkeit bietet, 5G-Netzwerke in der Infrastruktur bereitzustellen, zu verwalten und zu skalieren. AWS

Mit AWS TNB stellen Sie skalierbare und sichere 5G-Netzwerke bereit, AWS Cloud indem Sie ein Abbild Ihres Netzwerks auf automatisierte Weise verwenden. Sie müssen sich nicht mit neuen Technologien vertraut machen, entscheiden, welchen Rechendienst Sie verwenden möchten, oder wissen, wie AWS Ressourcen bereitgestellt und konfiguriert werden.

Stattdessen beschreiben Sie die Infrastruktur Ihres Netzwerks und stellen die Software-Images der Netzwerkfunktionen von Ihren ISV-Partnern (Independent Software Vendor) zur Verfügung. AWS TNB lässt sich in Service-Orchestratoren und AWS Dienste von Drittanbietern integrieren, um automatisch die erforderliche AWS Infrastruktur bereitzustellen, containerisierte Netzwerkfunktionen bereitzustellen und Netzwerk- und Zugriffsmanagement zu konfigurieren, um einen voll funktionsfähigen Netzwerkdienst zu schaffen.

Das folgende Diagramm veranschaulicht die logischen Integrationen zwischen AWS TNB und Service Orchestrator zur Bereitstellung von Netzwerkfunktionen mithilfe von Standardschnittstellen, die auf dem European Telecommunications Standards Institute (ETSI) basieren.



Themen

Neu bei? AWS

Wenn Sie noch keine Erfahrung mit AWS Produkten und Dienstleistungen haben, können Sie sich anhand der folgenden Ressourcen weiterbilden:

- Einführung in AWS
- Erste Schritte mit AWS

Für wen ist AWS TNB gedacht?

AWS TNB möchte die Vorteile der Kosteneffizienz, Flexibilität und Flexibilität nutzen, die es AWS Cloud bietet, ohne benutzerdefinierte Skripte und Konfigurationen für CSPs den Entwurf, die Bereitstellung und Verwaltung von Netzwerkdiensten schreiben und verwalten zu müssen. AWS TNB stellt automatisch die erforderliche AWS Infrastruktur bereit, stellt containerisierte Netzwerkfunktionen bereit und konfiguriert das Netzwerk- und Zugriffsmanagement, um voll funktionsfähige Netzwerkdienste zu erstellen, die auf den vom CSP definierten Netzwerkdienstbeschreibungen und den Netzwerkfunktionen basieren, die der CSP bereitstellen möchte.

AWS TNB-Funktionen

Im Folgenden sind einige der Gründe aufgeführt, aus denen ein CSP TNB verwenden AWS möchte:

Hilft bei der Vereinfachung von Aufgaben

Sorgen Sie für mehr Effizienz im Netzwerkbetrieb, z. B. durch die Bereitstellung neuer Dienste, die Aktualisierung und Aufrüstung von Netzwerkfunktionen und die Änderung der Netzwerkinfrastrukturtopologien.

Lässt sich in Orchestratoren integrieren

AWS TNB lässt sich in beliebte ETSI-konforme Service-Orchestratoren von Drittanbietern integrieren.

Waagen

Sie können AWS TNB so konfigurieren, dass die zugrunde liegenden AWS Ressourcen skaliert werden, um den Verkehrsbedarf zu decken, Aktualisierungen der Netzwerkfunktionen effizienter durchzuführen, Änderungen der Netzwerkinfrastrukturtopologie einzuführen und die Bereitstellungszeit neuer 5G-Dienste von Tagen auf Stunden zu reduzieren.

Inspiziert und überwacht Ressourcen AWS

AWS Mit TNB können Sie die AWS Ressourcen, die Ihr Netzwerk unterstützen, auf einem einzigen Dashboard überprüfen und überwachen, z. B. Amazon VPC EC2, Amazon und Amazon EKS.

Unterstützt Service-Vorlagen

AWS Mit TNB können Sie Dienstvorlagen für alle Telekommunikations-Workloads (RAN, Core, IMS) erstellen. Sie können eine neue Servicedefinition erstellen, eine vorhandene Vorlage

wiederverwenden oder sie in eine CI/CD-Pipeline (Continuous Integration and Continuous Delivery) integrieren, um eine neue Definition zu veröffentlichen.

Verfolgt Änderungen an Netzwerkbereitstellungen

Wenn Sie die zugrunde liegende Konfiguration einer Netzwerkfunktionsbereitstellung ändern, z. B. den Instance-Typ eines EC2 Amazon-Instance-Typs ändern, können Sie die Änderungen wiederholbar und skalierbar verfolgen. Um dies manuell zu tun, müssten Sie den Status des Netzwerks verwalten, Ressourcen erstellen und löschen und auf die Reihenfolge der erforderlichen Änderungen achten. Wenn Sie AWS TNB verwenden, um den Lebenszyklus Ihrer Netzwerkfunktion zu verwalten, nehmen Sie nur die Änderungen an Ihren Netzwerkdienstbeschreibungen vor, die die Netzwerkfunktion beschreiben. AWS TNB nimmt dann automatisch die erforderlichen Änderungen in der richtigen Reihenfolge vor.

Vereinfacht den Lebenszyklus von Netzwerkfunktionen

Sie können die erste und alle nachfolgenden Versionen einer Netzwerkfunktion verwalten und angeben, wann ein Upgrade durchgeführt werden soll. Sie können auch Ihre RAN-, Core-, IMS- und Netzwerkanwendungen auf die gleiche Weise verwalten.

Zugreifen auf AWS TNB

Sie können Ihre AWS TNB-Ressourcen über eine der folgenden Schnittstellen erstellen, darauf zugreifen und sie verwalten:

- AWS TNB-Konsole — Bietet eine Weboberfläche für die Verwaltung Ihres Netzwerks.
- AWS TNB-API — Stellt eine RESTful API zur Durchführung von AWS TNB-Aktionen bereit. Weitere Informationen finden Sie in der AWS TNB-API-Referenz
- AWS Command Line Interface (AWS CLI) — Stellt Befehle für eine Vielzahl von AWS Diensten bereit, einschließlich AWS TNB. Es wird unter Windows, MacOS und Linux unterstützt. Weitere Informationen finden Sie im AWS Command Line Interface -Benutzerhandbuch.
- AWS SDKs— Stellt sprachspezifisch bereit APIs und vervollständigt viele Verbindungsdetails. Dazu gehören das Berechnen von Signaturen, das Behandeln von Wiederholungsversuchen und das Behandeln von Fehlern. Weitere Informationen finden Sie unter AWS SDKs.

Preise für TNB AWS

AWS TNB hilft bei der CSPs Automatisierung der Bereitstellung und Verwaltung ihrer Telekommunikationsnetze auf. AWS Sie zahlen für die folgenden beiden Dimensionen, wenn Sie AWS TNB verwenden:

- Nach Stunden für verwaltete Netzwerkfunktionen (Managed Network Function Item, MNFI).
- Nach Anzahl der API-Anfragen.

Außerdem fallen zusätzliche Gebühren an, wenn Sie andere AWS Dienste in Verbindung mit AWS TNB nutzen. Weitere Informationen finden Sie unter AWS TNB-Preise.

Um Ihre Rechnung anzuzeigen, navigieren Sie zum Fakturierungs- und Kostenverwaltungs-Dashboard in der AWS Fakturierung und Kostenmanagement -Konsole. Ihre Abrechnung enthält Links zu Nutzungsberichten mit zusätzlichen Details zu Ihrer Abrechnung. Weitere Informationen zur AWS Kontoabrechnung finden Sie unter AWS Kontoabrechnung.

Wenn Sie Fragen zu AWS Abrechnung, Konten und Veranstaltungen haben, wenden Sie sich an den AWS Support.

AWS Trusted Advisor ist ein Service, mit dem Sie die Kosten, die Sicherheit und die Leistung Ihrer AWS Umgebung optimieren können. Weitere Informationen finden Sie unter AWS Trusted Advisor.

Was kommt als Nächstes

Weitere Informationen zu den ersten Schritten mit AWS TNB finden Sie in den folgenden Themen:

- AWS TNB einrichten— Führen Sie die erforderlichen Schritte aus.
- Erste Schritte mit AWS TNB— Stellen Sie Ihre erste Netzwerkfunktion bereit, z. B. eine zentrale Einheit (CU), eine Access and Mobility Management Function (AMF), eine Benutzerebenenfunktion (UPF) oder einen kompletten 5G-Core.

So AWS funktioniert TNB

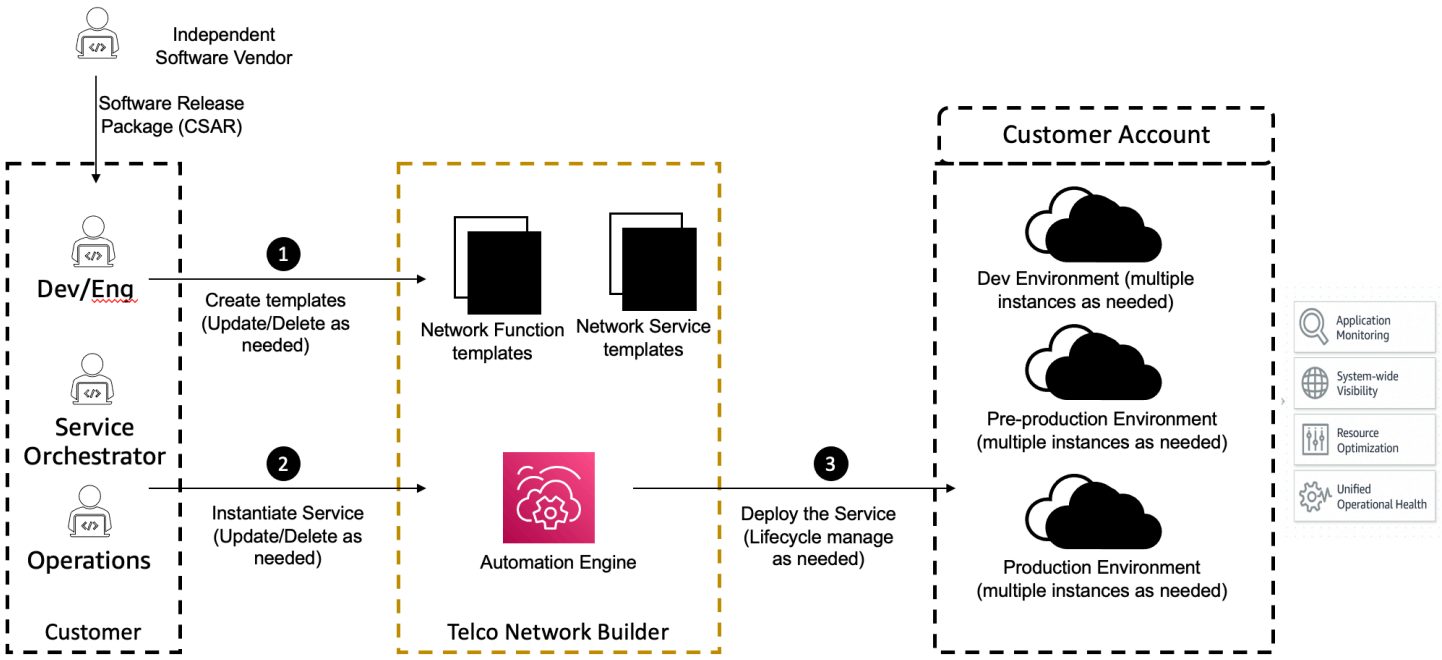
AWS TNB integriert sich in standardisierte end-to-end Orchestratoren und AWS Ressourcen, um vollständige 5G-Netzwerke zu betreiben.

AWS TNB ermöglicht Ihnen die Aufnahme von Netzwerkfunktionspaketen und Netzwerkdienstbeschreibungen (NSDs) und stellt Ihnen die Automatisierungs-Engine für den Betrieb Ihrer Netzwerke zur Verfügung. Sie können Ihren end-to-end Orchestrator verwenden und in TNB integrieren oder AWS TNB verwenden APIs, um Ihren eigenen Automatisierungsablauf zu AWS erstellen SDKs . Weitere Informationen finden Sie unter AWS TNB-Architektur.

Topics

AWS TNB-Architektur

AWS TNB bietet Ihnen die Möglichkeit, Lebenszyklusmanagement-Operationen über die AWS-Managementkonsole, AWS CLI, AWS TNB-REST-API und durchzuführen. SDKs Auf diese Weise können die verschiedenen CSP-Personas, z. B. Mitglieder der Teams für Technik, Betrieb und Programmatic System, die Vorteile von TNB nutzen. AWS Sie erstellen ein Netzwerk-Funktionspaket und laden es als Cloud Service Archive (CSAR) -Datei hoch. Die CSAR-Datei enthält Helm-Diagramme, Software-Images und einen Network Function Descriptor (NFD). Sie können Vorlagen verwenden, um wiederholt mehrere Konfigurationen dieses Pakets bereitzustellen. Sie erstellen Netzwerkdienstvorlagen, die die Infrastruktur und die Netzwerkfunktionen definieren, die Sie bereitstellen möchten. Sie können Parameterüberschreibungen verwenden, um verschiedene Konfigurationen an verschiedenen Orten bereitzustellen. Anschließend können Sie mithilfe der Vorlagen ein Netzwerk instanziieren und Ihre Netzwerkfunktionen in der Infrastruktur bereitstellen. AWS AWS TNB bietet Ihnen den Überblick über Ihre Bereitstellungen.



Integration mit AWS-Services

Ein 5G-Netzwerk besteht aus einer Reihe miteinander verbundener containerisierter Netzwerkfunktionen, die in Tausenden von Kubernetes-Clustern eingesetzt werden. AWS TNB lässt sich AWS-Services als Telekommunikationsspezifikation mit folgenden Komponenten integrieren, um einen voll funktionsfähigen Netzwerkdienst zu schaffen APIs :

- Amazon Elastic Container Registry (Amazon ECR) zum Speichern von Netzwerkfunktions-Artefakten von Independent Software Vendors (ISVs).
- Amazon Elastic Kubernetes Service (Amazon EKS), um Cluster einzurichten.
- Amazon VPC für Netzwerkkonstrukte.
- Sicherheitsgruppen verwenden. CloudFormation
- AWS CodePipeline für Bereitstellungsziele in verschiedenen AWS-Regionen AWS Local Zones und AWS Outposts.
- IAM, um Rollen zu definieren.
- AWS Organizations um den Zugriff auf AWS APIs TNB zu kontrollieren.
- Health Dashboard und AWS CloudTrail zur Überwachung des Zustands und zur Veröffentlichung von Kennzahlen.

AWS TNB-Ressourcenkontingente

Ihr AWS-Konto verfügt über Standardkontingente, die früher als Limits bezeichnet wurden, für jedes AWS-Service Kontingent. Sofern nicht anders angegeben, ist jedes Kontingent spezifisch für ein AWS-Region. Sie können Erhöhungen für einige Kontingente beantragen aber nicht für alle Kontingente.

Um die Kontingente für AWS TNB anzuzeigen, öffnen Sie die Service Quotas Quotas-Konsole. Wählen Sie im Navigationsbereich AWS TNB aus und wählen AWS-ServicesSie es aus.

Informationen zur Erhöhung eines Kontingents finden Sie unter Anfordern einer Kontingenterhöhung im Service-Quotas-Benutzerhandbuch.

Ihr AWS-Konto hat die folgenden Kontingente in Bezug auf AWS TNB.

Ressource nkontingent	Description	Standardwert	Anpassbar?
Netzwerkd ienstinstanzen	Die maximale Anzahl von Netzwerkd ienstinstanzen in einer Region.	800	Ja
Gleichzeitiger laufender Netzwerkd ienstbetrieb	Die maximale Anzahl gleichzeitiger laufender Netzwerkdienstoperationen in einer Region.	40	Ja
Netzwerk-Pakete	Die maximale Anzahl von Netzwerkp aketen in einer Region.	40	Ja
Funktionspakete	Die maximale Anzahl von Funktions paketen in einer Region.	200	Ja

AWS TNB-Konzepte

In diesem Thema werden grundlegende Konzepte beschrieben, die Ihnen den Einstieg in die Verwendung von AWS TNB erleichtern sollen.

Inhalt

Lebenszyklus einer Netzwerkfunktion

AWS TNB unterstützt Sie während des gesamten Lebenszyklus Ihrer Netzwerkfunktionen. Der Lebenszyklus von Netzwerkfunktionen umfasst die folgenden Phasen und Aktivitäten:

Planung

1. Planen Sie Ihr Netzwerk, indem Sie die bereitzustellenden Netzwerkfunktionen identifizieren.
2. Speichern Sie die Software-Images für Netzwerkfunktionen in einem Container-Image-Repository.
3. Erstellen Sie die CSAR-Pakete, die bereitgestellt oder aktualisiert werden sollen.
4. Verwenden Sie AWS TNB, um das CSAR-Paket hochzuladen, das Ihre Netzwerkfunktion definiert (z. B. CU AMF und UPF), und integrieren Sie es in eine CI/CD-Pipeline (Continuous Integration and Continuous Delivery), die Ihnen helfen kann, neue Versionen Ihres CSAR-Pakets zu erstellen, sobald neue Netzwerkfunktions-Softwareimages oder Kundenskripte verfügbar sind.

Konfiguration

1. Identifizieren Sie die für die Bereitstellung erforderlichen Informationen, z. B. den Berechnungstyp, die Version der Netzwerkfunktion, IP-Informationen und Namen der Ressourcen.
2. Verwenden Sie diese Informationen, um Ihren Network Service Descriptor (NSD) zu erstellen.
3. Ingest NSDs , die Ihre Netzwerkfunktionen und die Ressourcen definieren, die für die Instanziierung der Netzwerkfunktion erforderlich sind.

Instanziierung

1. Erstellen Sie die Infrastruktur, die für die Netzwerkfunktionen erforderlich ist.
2. Instanziieren (oder stellen Sie sie bereit), wie in ihrer NSD definiert, und beginnen Sie mit der Übertragung des Datenverkehrs.
3. Validieren Sie die Ressourcen.

Produktion

Während des Lebenszyklus der Netzwerkfunktion werden Sie Produktionsvorgänge abschließen, wie z. B.:

- Aktualisieren Sie die Konfiguration der Netzwerkfunktion, indem Sie beispielsweise einen Wert in der bereitgestellten Netzwerkfunktion aktualisieren.
- Aktualisieren Sie die Netzwerkinstanz mit einem neuen Netzwerkpaket und neuen Parameterwerten. Aktualisieren Sie beispielsweise den Amazon `version` EKS-Parameter im Netzwerkpaket.

Verwenden Sie standardisierte Schnittstellen

AWS TNB lässt sich in ETSI (European Telecommunications Standards Institute) integrierte Service-Orchestratoren integrieren, sodass Sie die Bereitstellung Ihrer Netzwerkdienste vereinfachen können. Service-Orchestratoren können AWS TNB SDKs, die CLI oder die verwenden, um Operationen APIs zu initiieren, z. B. die Instanziierung oder Aktualisierung einer Netzwerkfunktion auf eine neue Version.

AWS TNB unterstützt die folgenden Spezifikationen.

Spezifikation	Veröffentlichung	Description
ETSI SOL001	v3.6.1	Definiert Standards für die Zulassung von TOSCA-basierten Netzwerkfunktionsdeskriptoren.
ETSI SOL002	v3.6.1	Definiert Modelle rund um das Netzwerkfunktionsmanagement.
ETSI SOL003	v3.6.1	Definiert Standards für das Lebenszyklusmanagement von Netzwerkfunktionen.

Spezifikation	Veröffentlichung	Description
ETSI SOL004	v3.6.1	Definiert CSAR-Standards für Netzwerkf unktionspakete.
ETSI SOL005	v3.6.1	Definiert Standards für das Netzwerk-Servicepa ket und das Lebenszyklusmanagement von Netzwerkdiensten.
ETSI SOL007	v3.5.1	Definiert Standards für die Zulassung von TOSCA-basierten Netzwerkdienstdeskriptoren.

Funktionspaket

Mit AWS TNB können Sie Funktionspakete, die ETSI SOL001/SOL004 entsprechen, in einem Funktionskatalog speichern. Anschließend können Sie Cloud Service Archive (CSAR) -Pakete hochladen, die Artefakte enthalten, die Ihre virtuelle Netzwerkfunktion beschreiben.

- Deskriptor für virtuelle Netzwerkfunktionen — Definiert Metadaten für das Onboarding von Paketen und die Verwaltung virtueller Netzwerkfunktionen. Sie müssen dieser Datei einen Namen geben. `vnfd.yaml`
- Software-Images — Verweist auf Container-Images für virtuelle Netzwerkfunktionen. Amazon Elastic Container Registry (Amazon ECR) kann als Repository für virtuelle Netzwerkfunktions-Images dienen.
- Zusätzliche Dateien — werden zur Verwaltung der virtuellen Netzwerkfunktion verwendet, z. B. für Skripts und Helm-Diagramme.

Das CSAR ist ein Paket, das durch den OASIS TOSCA-Standard definiert ist und einen Netzwerk-/Dienstdeskriptor enthält, der der OASIS TOSCA YAML-Spezifikation entspricht. Hinweise zur erforderlichen YAML-Spezifikation finden Sie unter. TOSCA-Referenz für TNB AWS

Im Folgenden finden Sie ein Beispiel für einen Funktionsdeskriptor für virtuelle Netzwerke.

```
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"
```

Netzwerkpaket

Ein Netzwerkpaket ist eine `.zip` Datei im CSAR-Format (Cloud Service Archive). Es definiert die Funktionspakete, die Sie bereitstellen möchten, und die AWS Infrastruktur, auf der Sie sie bereitstellen möchten.

Das Netzwerkpaket enthält die folgenden Dateien:

- Eine Netzwerkdeskriptordatei (`nsd.yaml`) im TOSCA-Format, wie von ETSI SOL007 beschrieben.

  Die `nsd.yaml` Datei enthält Verweise auf hochgeladene Funktionspakete mit ihrem Deskriptor. IDs
- Benutzerdatenskripten, falls vorhanden.
- Lifecycle-Hook-Skripten, falls vorhanden.
- `values.yaml`Konfigurationsdateien der Plugins, falls vorhanden.

AWS TNB unterstützt ETSI-Standards für die Modellierung von Ressourcen wie Netzwerk, Dienst und Funktion in der TOSCA-Sprache. AWS TNB macht die Nutzung für Sie effizienter, AWS-Services indem es sie so modelliert, dass Ihr ETSI-konformer Service Orchestrator sie verstehen kann.

Netzwerkservice-Deskriptoren für TNB AWS

Ein Network Service Descriptor (NSD) ist eine `.yaml` Datei in einem Netzwerkpaket, die den TOSCA-Standard verwendet, um die Netzwerkfunktionen, die Sie bereitstellen möchten,

und die AWS Infrastruktur, auf der Sie die Netzwerkfunktionen bereitstellen möchten, zu beschreiben. Um Ihre NSD zu definieren und Ihre zugrunde liegenden Ressourcen und Netzwerklebenszyklusoperationen zu konfigurieren, müssen Sie das von TNB unterstützte NSD-TOSCA-Schema verstehen. AWS

Ihre NSD-Datei ist in die folgenden Teile unterteilt:

1. TOSCA-Definitionsversion — Dies ist die erste Zeile Ihrer NSD-YAML-Datei und enthält die Versionsinformationen, wie im folgenden Beispiel gezeigt.

```
tosca_definitions_version: tnb_simple_yaml_1_0
```

2. VNFD — Die NSD enthält die Definition der Netzwerkfunktion, auf der Lebenszyklusoperationen ausgeführt werden sollen. Jede Netzwerkfunktion muss anhand der folgenden Werte identifiziert werden:
   - Eine eindeutige ID für`descriptor_id`. Die ID muss mit der ID im CSAR-Paket der Netzwerkfunktion übereinstimmen.
   - Ein eindeutiger Name für`namespace`. Der Name muss mit einer eindeutigen ID verknüpft werden, damit er in Ihrer NSD-YAML-Datei leichter referenziert werden kann, wie im folgenden Beispiel gezeigt.

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

3. Topologievorlage — Definiert die bereitzustellenden Ressourcen, die Bereitstellung von Netzwerkfunktionen und alle benutzerdefinierten Skripts, wie z. B. Lifecycle-Hooks. Dies wird im folgenden Beispiel veranschaulicht.

```
topology_template:

  node_templates:

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

4. Zusätzliche Knoten — Jede modellierte Ressource hat Abschnitte für Eigenschaften und Anforderungen. Die Eigenschaften beschreiben optionale oder obligatorische Attribute für eine Ressource, z. B. die Version. Die Anforderungen beschreiben Abhängigkeiten, die als Argumente angegeben werden müssen. Um beispielsweise eine Amazon EKS Node Group-Ressource zu erstellen, muss sie innerhalb eines Amazon EKS-Clusters erstellt werden. Dies wird im folgenden Beispiel veranschaulicht.

```
SampleEKSNode:
  type: tosca.nodes.AWS.Compute.EKSManagedNode
  properties:
    node_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleRole"
  capabilities:
    compute:
      properties:
        ami_type: "AL2_x86_64"
        instance_types:
          - "t3.xlarge"
        key_pair: "SampleKeyPair"
    scaling:
      properties:
        desired_size: 1
        min_size: 1
        max_size: 1
  requirements:
    cluster: SampleEKS
    subnets:
      - SampleSubnet
    network_interfaces:
      - SampleENI01
      - SampleENI02
```

Beispiel NSD

Das Folgende ist ein Ausschnitt aus einem NSD, der zeigt, wie man modelliert. AWS-Services Die Netzwerkfunktion wird auf einem Amazon EKS-Cluster mit Kubernetes Version 1.27 bereitgestellt. Die Subnetze für die Anwendungen sind Subnet01 und Subnet02. Anschließend können Sie das NodeGroups für Ihre Anwendungen mit einem Amazon Machine Image (AMI), einem Instance-Typ und einer Autoscaling-Konfiguration definieren.

```
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
```

Verwaltung und Betrieb für TNB AWS

Mit AWS TNB können Sie Ihr Netzwerk mithilfe standardisierter Verwaltungsoperationen gemäß ETSI SOL003 und SOL005 verwalten. Sie können die AWS TNB verwenden, um Lebenszyklusoperationen durchzuführen APIs , wie z. B.:

- Instanziieren Ihrer Netzwerkfunktionen.
- Beenden Ihrer Netzwerkfunktionen.
- Aktualisierung Ihrer Netzwerkfunktionen, um Helm-Bereitstellungen außer Kraft zu setzen.
- Aktualisierung einer instanziierten oder aktualisierten Netzwerkinstanz mit einem neuen Netzwerkpaket und neuen Parameterwerten.
- Verwaltung von Versionen Ihrer Netzwerkfunktionspakete.
- Verwaltung von Versionen Ihrer NSDs.
- Abrufen von Informationen zu Ihren bereitgestellten Netzwerkfunktionen.

AWS TNB einrichten

Richten Sie AWS TNB ein, indem Sie die in diesem Thema beschriebenen Aufgaben ausführen.

Aufgaben

- Melden Sie sich an für ein AWS-Konto
- Erstellen eines Benutzers mit Administratorzugriff
- Wählen Sie eine Region AWS
- Notieren Sie sich den Service-Endpunkt
- (Optional) Installieren Sie AWS CLI
- Richten Sie TNB-Rollen ein AWS

Melden Sie sich an für ein AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/die Anmeldung.
2. Folgen Sie den Online-Anweisungen.

   Während der Anmeldung erhalten Sie einen Telefonanruf oder eine Textnachricht und müssen einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Benutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Du kannst jederzeit deine aktuellen Kontoaktivitäten einsehen und dein Konto verwalten, indem du zu https://aws.amazon.com/gehst und Mein Konto auswählst.

Erstellen eines Benutzers mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS-Managementkonsolealls Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung -Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Benutzers mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center -Benutzerhandbuch.
2. Gewähren Sie einem Administratorbenutzer im IAM Identity Center Benutzerzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie IAM-Identity-Center-Verzeichnis im Benutzerhandbuch unter Benutzerzugriff mit der Standardeinstellung konfigurieren.AWS IAM Identity Center

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Access-Portal.

Weiteren Benutzern Zugriff zuweisen

1. Erstellen Sie im IAM-Identity-Center einen Berechtigungssatz, der den bewährten Vorgehensweisen für die Anwendung von geringsten Berechtigungen folgt.

   Anweisungen hierzu finden Sie unter Berechtigungssatz erstellen im AWS IAM Identity Center - Benutzerhandbuch.
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Eine genaue Anleitung finden Sie unter Gruppen hinzufügen im AWS IAM Identity Center - Benutzerhandbuch.

Wählen Sie eine Region AWS

Eine Liste der verfügbaren Regionen für AWS TNB finden Sie in der Liste der AWS regionalen Dienste. Eine Liste der Endpunkte für den programmatischen Zugriff finden Sie unter AWS TNB-Endpunkte in der. Allgemeine AWS-Referenz

Notieren Sie sich den Service-Endpunkt

Um programmgesteuert eine Verbindung zu einem AWS Dienst herzustellen, verwenden Sie einen Endpunkt. Zusätzlich zu den AWS Standardendpunkten bieten einige AWS Dienste FIPS-Endpunkte in ausgewählten Regionen. Weitere Informationen finden Sie unter AWS -Service-Endpunkte.

Name der Region	Region	Endpunkt	Protocol (Protokoll)	
USA Ost (Nord-Virginia)	us-east-1	tnb.us-east-1.amazonaws.com	HTTPS	

Name der Region	Region	Endpunkt	Protocol (Protokoll)	
USA West (Oregon)	us-west-2	tnb.us-west-2.amazonaws.com	HTTPS	
Asien-Pazifik (Seoul)	ap-northeast-2	tnb.ap-northeast-2.amazonaws.com	HTTPS	
Asien-Pazifik (Sydney)	ap-southeast-2	tnb.ap-southeast-2.amazonaws.com	HTTPS	
Kanada (Zentral)	ca-central-1	tnb.ca-central-1.amazonaws.com	HTTPS	
Europa (Frankfurt)	eu-central-1	tnb.eu-central-1.amazonaws.com	HTTPS	
Europa (Paris)	eu-west-3	tnb.eu-west-3.amazonaws.com	HTTPS	
Europa (Spanien)	eu-south-2	tnb.eu-south-2.amazonaws.com	HTTPS	
Europa (Stockholm)	eu-north-1	tnb.eu-north-1.amazonaws.com	HTTPS	
Südamerika (São Paulo)	sa-east-1	tnb.sa-east-1.amazonaws.com	HTTPS	

(Optional) Installieren Sie AWS CLI

Die AWS Command Line Interface (AWS CLI) bietet Befehle für eine Vielzahl von AWS Produkten und wird unter Windows, MacOS und Linux unterstützt. Sie können mit dem auf AWS TNB zugreifen. AWS CLI Informationen zu den ersten Schritten finden Sie im AWS Command Line Interface - Benutzerhandbuch. Weitere Informationen zu den Befehlen für AWS TNB finden Sie unter tnb in der AWS CLI Befehlsreferenz.

Richten Sie TNB-Rollen ein AWS

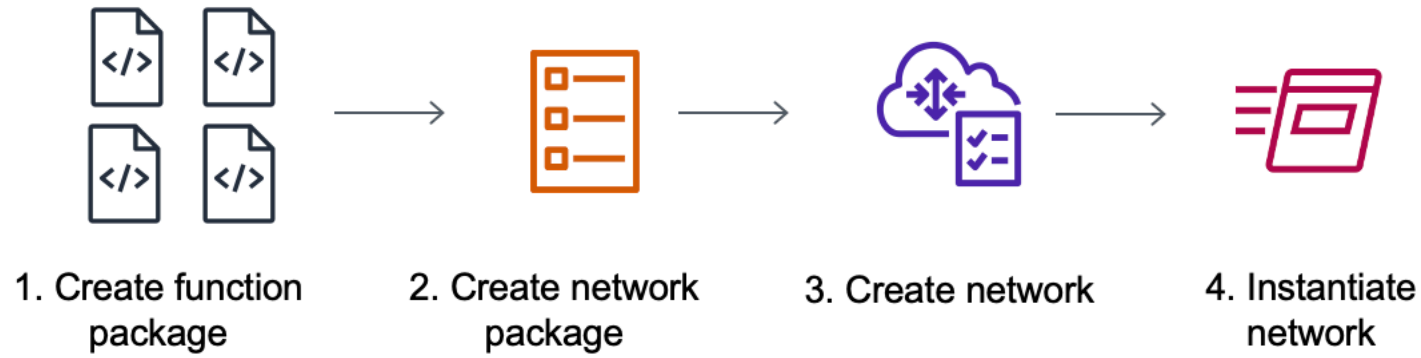
Sie müssen eine IAM-Servicerolle erstellen, um verschiedene Teile Ihrer AWS TNB-Lösung zu verwalten. AWS TNB-Servicerollen können in Ihrem Namen API-Aufrufe an andere AWS Dienste wie AWS CloudFormation AWS CodeBuild, und verschiedene Rechen- und Speicherdienste richten, um Ressourcen für Ihre Bereitstellung zu instanziieren und zu verwalten.

Weitere Informationen zur AWS TNB-Servicerolle finden Sie unter. Identitäts- und Zugriffsmanagement für TNB AWS

Erste Schritte mit AWS TNB

Dieses Tutorial zeigt, wie Sie AWS TNB verwenden, um eine Netzwerkfunktion bereitzustellen, z. B. die Centralized Unit (CU), die Access and Mobility Management Function (AMF) oder die 5G User Plane Function (UPF).

Das folgende Diagramm veranschaulicht den Bereitstellungsprozess:



Aufgaben

- Voraussetzungen
- Erstellen Sie ein Funktionspaket
- Erstellen Sie ein Netzwerkpaket
- Erstellen und instanziieren Sie eine Netzwerkinstanz
- Bereinigen

Voraussetzungen

Bevor Sie eine erfolgreiche Bereitstellung durchführen können, müssen Sie über Folgendes verfügen:

- Ein AWS Business Support Plan.
- Berechtigungen durch IAM-Rollen.
- Ein Network Function (NF) -Paket, das ETSI SOL001/SOL004 entspricht.
- NSD-Vorlagen (Network Service Descriptor), die ETSI SOL007 entsprechen.

Sie können ein Beispielfunktionspaket oder ein Netzwerkpaket von der Website Beispielpakete für TNB verwenden. AWS GitHub

Erstellen Sie ein Funktionspaket

Ein Netzwerk-Funktionspaket ist eine Cloud Service Archive (CSAR) -Datei. Die CSAR-Datei enthält Helm-Diagramme, Software-Images und einen Network Function Descriptor (NFD).

Um ein Funktionspaket zu erstellen

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich die Option Funktionspakete aus.
3. Wählen Sie Funktionspaket erstellen.
4. Wählen Sie unter Funktionspaket hochladen die Option Dateien auswählen aus und laden Sie jedes CSAR-Paket als `.zip` Datei hoch. Sie können maximal 10 Dateien hochladen.
5. (Optional) Wählen Sie unter Tags die Option Neues Tag hinzufügen aus und geben Sie einen Schlüssel und einen Wert ein. Sie können Tags verwenden, um Ihre Ressourcen zu suchen und zu filtern oder Ihre AWS Kosten zu verfolgen.
6. Wählen Sie Weiter.
7. Überprüfen Sie die Paketdetails und wählen Sie dann Funktionspaket erstellen.

Erstellen Sie ein Netzwerkpaket

Ein Netzwerkpaket gibt die Netzwerkfunktionen an, die Sie bereitstellen möchten, und gibt an, wie Sie sie im Katalog bereitstellen möchten.

Um ein Netzwerkpaket zu erstellen

1. Wählen Sie im Navigationsbereich Netzwerkpakete aus.
2. Wählen Sie Netzwerkpaket erstellen aus.
3. Wählen Sie unter Netzwerkpaket hochladen die Option Dateien auswählen aus und laden Sie jede NSD als `.zip` Datei hoch. Sie können maximal 10 Dateien hochladen.
4. (Optional) Wählen Sie unter Tags die Option Neues Tag hinzufügen aus und geben Sie einen Schlüssel und einen Wert ein. Sie können Tags verwenden, um Ihre Ressourcen zu suchen und zu filtern oder Ihre AWS Kosten zu verfolgen.

5. Wählen Sie Weiter.
6. Wählen Sie Netzwerkpaket erstellen.

Erstellen und instanziieren Sie eine Netzwerkinstanz

Eine Netzwerkinstanz ist ein einzelnes Netzwerk, das in AWS TNB erstellt wurde und bereitgestellt werden kann. Sie müssen eine Netzwerkinstanz erstellen und instanziieren. Wenn Sie eine Netzwerkinstanz instanziieren, stellt AWS TNB die erforderliche AWS Infrastruktur bereit, stellt containerisierte Netzwerkfunktionen bereit und konfiguriert das Netzwerk- und Zugriffsmanagement, um einen voll funktionsfähigen Netzwerkdienst zu erstellen.

Um eine Netzwerkinstanz zu erstellen und zu instanziieren

1. Wählen Sie im Navigationsbereich Netzwerke aus.
2. Wählen Sie Netzwerkinstanz erstellen aus.
3. Geben Sie einen Namen und eine Beschreibung für das Netzwerk ein und wählen Sie dann Weiter.
4. Wählen Sie ein Netzwerkpaket aus. Überprüfen Sie die Details und wählen Sie Weiter.
5. Wählen Sie Netzwerkinstanz erstellen. Der Ausgangszustand ist`Created`.

   Die Seite Netzwerke wird angezeigt, auf der die neue Netzwerkinstanz im `Not instantiated` Status angezeigt wird.
6. Wählen Sie die Netzwerkinstanz aus, klicken Sie auf Aktionen und dann auf Instanziieren.

   Die Seite Netzwerkinstanziierung wird angezeigt.
7. Überprüfen Sie die Details und aktualisieren Sie die Parameterwerte. Aktualisierungen der Parameterwerte gelten nur für diese Netzwerkinstanz. Die Parameter in den NSD- und VNFD-Paketen ändern sich nicht.
8. Wählen Sie Netzwerk instantiieren.

   Die Seite mit dem Bereitstellungsstatus wird angezeigt.
9. Verwenden Sie das Aktualisierungssymbol, um den Bereitstellungsstatus Ihrer Netzwerkinstanz zu verfolgen. Sie können die automatische Aktualisierung auch im Bereich Bereitstellungsaufgaben aktivieren, um den Fortschritt der einzelnen Aufgaben zu verfolgen.

Bereinigen

Sie können jetzt die Ressourcen löschen, die Sie für dieses Tutorial erstellt haben.

So bereinigen Sie Ihre Ressourcen

1. Wählen Sie im Navigationsbereich Netzwerke aus.
2. Wählen Sie die ID des Netzwerks und dann Terminate aus.
3. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie die Netzwerk-ID ein und wählen Sie dann Terminate.
4. Verwenden Sie das Aktualisierungssymbol, um den Status Ihrer Netzwerkinstanz zu verfolgen.
5. (Optional) Wählen Sie das Netzwerk aus und wählen Sie Löschen.

Funktionspakete für AWS TNB

Ein Funktionspaket ist eine ZIP-Datei im CSAR-Format (Cloud Service Archive), die eine Netzwerkfunktion (eine ETSI-Standard-Telekommunikationsanwendung) und einen Funktionspaketdeskriptor enthält, der den TOSCA-Standard verwendet, um zu beschreiben, wie die Netzwerkfunktionen in Ihrem Netzwerk ausgeführt werden sollen.

Aufgaben

- Erstellen Sie ein Funktionspaket in TNB AWS
- Ein Funktionspaket in AWS TNB anzeigen
- Laden Sie ein Funktionspaket von AWS TNB herunter
- Löscht ein Funktionspaket aus TNB AWS

Erstellen Sie ein Funktionspaket in TNB AWS

Erfahren Sie, wie Sie ein Funktionspaket im AWS TNB-Netzwerkfunktionskatalog erstellen. Das Erstellen eines Funktionspakets ist der erste Schritt zur Erstellung eines Netzwerks in AWS TNB. Nachdem Sie ein Funktionspaket hochgeladen haben, können Sie ein Netzwerkpaket erstellen.

Console

Um ein Funktionspaket mit der Konsole zu erstellen

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich die Option Funktionspakete aus.
3. Wählen Sie Funktionspaket erstellen.
4. Wählen Sie Dateien auswählen und laden Sie jedes CSAR-Paket als `.zip` Datei hoch. Sie können maximal 10 Dateien hochladen.
5. Wählen Sie Weiter aus.
6. Überprüfen Sie die Paketdetails.
7. Wählen Sie Funktionspaket erstellen.

AWS CLI

Um ein Funktionspaket mit dem zu erstellen AWS CLI

1. Verwenden Sie den create-sol-function-packageBefehl, um ein neues Funktionspaket zu erstellen:

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

2. Verwenden Sie den Befehl put-sol-function-package-content, um den Inhalt des Funktionspakets hochzuladen. Beispiel:

```
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
```

Ein Funktionspaket in AWS TNB anzeigen

Erfahren Sie, wie Sie den Inhalt eines Funktionspakets einsehen können.

Console

So zeigen Sie ein Funktionspaket mit der Konsole an

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich die Option Funktionspakete aus.
3. Verwenden Sie das Suchfeld, um das Funktionspaket zu finden

AWS CLI

Um ein Funktionspaket anzuzeigen, verwenden Sie AWS CLI

1. Verwenden Sie den list-sol-function-packagesBefehl, um Ihre Funktionspakete aufzulisten.

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

2. Verwenden Sie den get-sol-function-packageBefehl, um Details zu einem Funktionspaket anzuzeigen.

```
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
```

Laden Sie ein Funktionspaket von AWS TNB herunter

Erfahren Sie, wie Sie ein Funktionspaket aus dem AWS TNB-Netzwerkfunktionskatalog herunterladen.

Console

So laden Sie ein Funktionspaket über die Konsole herunter

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich auf der linken Seite der Konsole die Option Funktionspakete aus.
3. Verwenden Sie das Suchfeld, um das Funktionspaket zu finden
4. Wählen Sie das Funktionspaket
5. Wählen Sie Aktionen, Herunterladen.

AWS CLI

Um ein Funktionspaket herunterzuladen, verwenden Sie AWS CLI

Verwenden Sie den Befehl get-sol-function-package-content, um ein Funktionspaket herunterzuladen.

```
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
```

Löscht ein Funktionspaket aus TNB AWS

Erfahren Sie, wie Sie ein Funktionspaket aus dem AWS TNB-Netzwerkfunktionskatalog löschen. Um ein Funktionspaket zu löschen, muss sich das Paket in einem deaktivierten Zustand befinden.

Console

Um ein Funktionspaket mit der Konsole zu löschen

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich die Option Funktionspakete aus.
3. Verwenden Sie das Suchfeld, um das Funktionspaket zu finden.
4. Wählen Sie ein Funktionspaket.
5. Wählen Sie Aktionen, deaktivieren.
6. Wählen Sie Aktionen, Löschen aus.

AWS CLI

Um ein Funktionspaket mit dem zu löschen AWS CLI

1. Verwenden Sie den update-sol-function-packageBefehl, um ein Funktionspaket zu deaktivieren.

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

2. Verwenden Sie den delete-sol-function-packageBefehl, um ein Funktionspaket zu löschen.

```
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
```

Netzwerkpakete für AWS TNB

Ein Netzwerkpaket ist eine ZIP-Datei im CSAR-Format (Cloud Service Archive). Es definiert die Funktionspakete, die Sie bereitstellen möchten, und die AWS Infrastruktur, auf der Sie sie bereitstellen möchten.

Das Netzwerkpaket enthält die folgenden Dateien:

- Eine Netzwerkdeskriptordatei (`nsd.yaml`) im TOSCA-Format, wie von ETSI beschrieben. SOL007

  Die `nsd.yaml` Datei enthält Verweise auf hochgeladene Funktionspakete mit ihrem Deskriptor. IDs
- Benutzerdatenskripten, falls vorhanden.
- Lifecycle-Hook-Skripten, falls vorhanden.
- `values.yaml`Konfigurationsdateien der Plugins, falls vorhanden.

Aufgaben

- Erstellen Sie ein Netzwerkpaket in TNB AWS
- Ein Netzwerkpaket in AWS TNB anzeigen
- Laden Sie ein Netzwerkpaket von AWS TNB herunter
- Löscht ein Netzwerkpaket aus TNB AWS

Erstellen Sie ein Netzwerkpaket in TNB AWS

Ein Netzwerkpaket besteht aus einer NSD-Datei (Network Service Descriptor) (erforderlich) und allen zusätzlichen Dateien (optional), z. B. Skripten, die auf Ihre Bedürfnisse zugeschnitten sind. Wenn Ihr Netzwerkpaket beispielsweise mehrere Funktionspakete enthält, können Sie mithilfe der NSD definieren, welche Netzwerkfunktionen in bestimmten VPCs Subnetzen oder Amazon EKS-Clustern ausgeführt werden sollen.

Erstellen Sie ein Netzwerkpaket, nachdem Sie Funktionspakete erstellt haben. Sobald Sie ein Netzwerkpaket erstellt haben, müssen Sie eine Netzwerkinstanz erstellen.

Console

Um ein Netzwerkpaket mit der Konsole zu erstellen

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich Netzwerkpakete aus.
3. Wählen Sie Netzwerkpaket erstellen aus.
4. Wählen Sie Dateien auswählen und laden Sie jede NSD als `.zip` Datei hoch. Sie können maximal 10 Dateien hochladen.
5. Wählen Sie Weiter aus.
6. Überprüfen Sie die Paketdetails.
7. Wählen Sie Netzwerkpaket erstellen.

AWS CLI

Um ein Netzwerkpaket mit dem zu erstellen AWS CLI

1. Verwenden Sie den create-sol-network-packageBefehl, um ein Netzwerkpaket zu erstellen.

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

2. Verwenden Sie den Befehl put-sol-network-package-content, um den Inhalt eines Netzwerkpakets hochzuladen. Beispiel:

```
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
```

Ein Netzwerkpaket in AWS TNB anzeigen

Erfahren Sie, wie Sie den Inhalt eines Netzwerkpakets anzeigen können.

Console

So zeigen Sie ein Netzwerkpaket mit der Konsole an

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich Netzwerkpakete aus.
3. Verwenden Sie das Suchfeld, um das Netzwerkpaket zu finden.

AWS CLI

Um ein Netzwerkpaket anzuzeigen, verwenden Sie AWS CLI

1. Verwenden Sie den list-sol-network-packagesBefehl, um Ihre Netzwerkpakete aufzulisten.

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

2. Verwenden Sie den get-sol-network-packageBefehl, um Details zu einem Netzwerkpaket anzuzeigen.

```
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
```

Laden Sie ein Netzwerkpaket von AWS TNB herunter

Erfahren Sie, wie Sie ein Netzwerkpaket aus dem AWS TNB-Netzwerkdienstkatalog herunterladen.

Console

So laden Sie ein Netzwerkpaket über die Konsole herunter

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich Netzwerkpakete aus.
3. Verwenden Sie das Suchfeld, um das Netzwerkpaket zu finden
4. Wählen Sie das Netzwerkpaket aus.
5. Wählen Sie Aktionen, Herunterladen.

AWS CLI

Um ein Netzwerkpaket herunterzuladen, verwenden Sie AWS CLI

- Verwenden Sie den Befehl get-sol-network-package-content, um ein Netzwerkpaket herunterzuladen.

```
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
```

Löscht ein Netzwerkpaket aus TNB AWS

Erfahren Sie, wie Sie ein Netzwerkpaket aus dem AWS TNB-Netzwerkdienstkatalog löschen. Um ein Netzwerkpaket zu löschen, muss sich das Paket im deaktivierten Zustand befinden.

Console

Um ein Netzwerkpaket mit der Konsole zu löschen

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich Netzwerkpakete aus.
3. Verwenden Sie das Suchfeld, um das Netzwerkpaket zu finden
4. Wählen Sie ein Netzwerkpaket
5. Wählen Sie Aktionen, deaktivieren.
6. Wählen Sie Aktionen, Löschen aus.

AWS CLI

Um ein Netzwerkpaket mit dem zu löschen AWS CLI

1. Verwenden Sie den update-sol-network-packageBefehl, um ein Netzwerkpaket zu deaktivieren.

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

2. Verwenden Sie den delete-sol-network-packageBefehl, um ein Netzwerkpaket zu löschen.

```
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
```

Netzwerkinstanzen für AWS TNB

Eine Netzwerkinstanz ist ein einzelnes Netzwerk, das in AWS TNB erstellt wurde und bereitgestellt werden kann.

Aufgaben

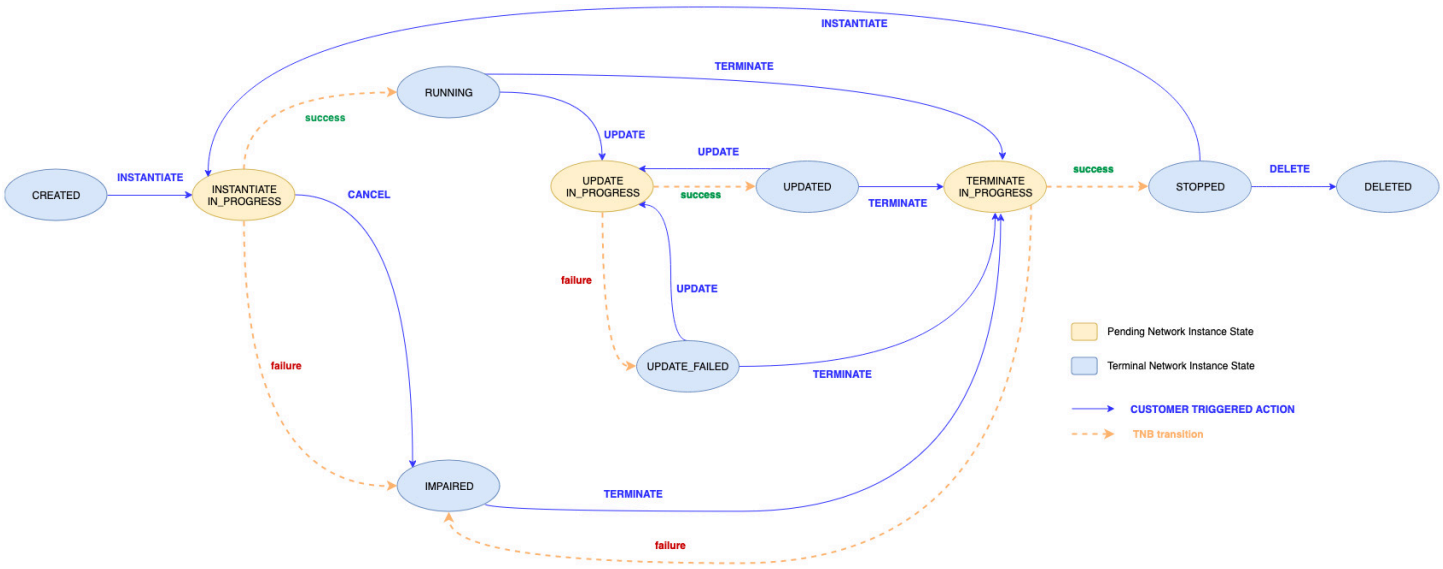
- Lebenszyklusvorgänge einer Netzwerkinstanz
- Erstellen Sie eine Netzwerkinstanz mit TNB AWS
- Instanziieren Sie eine Netzwerkinstanz mithilfe von TNB AWS
- Aktualisieren Sie eine Funktionsinstanz in AWS TNB
- Aktualisieren Sie eine Netzwerkinstanz in AWS TNB
- Eine Netzwerkinstanz in AWS TNB anzeigen
- Beenden und löschen Sie eine Netzwerkinstanz aus AWS TNB

Lebenszyklusvorgänge einer Netzwerkinstanz

AWS TNB ermöglicht Ihnen die einfache Verwaltung Ihres Netzwerks mithilfe standardisierter Verwaltungsvorgänge gemäß SOL003 ETSI und. SOL005 Sie können die folgenden Lebenszyklusoperationen ausführen:

- Erstellen Sie das Netzwerk
- Instanziieren Sie das Netzwerk
- Aktualisieren Sie die Netzwerkfunktion
- Aktualisieren Sie die Netzwerkinstanz
- Netzwerkdetails und Status anzeigen
- Beenden Sie das Netzwerk

Die folgende Abbildung zeigt die Netzwerkverwaltungsvorgänge:



Erstellen Sie eine Netzwerkinstanz mit TNB AWS

Sie erstellen eine Netzwerkinstanz, nachdem Sie ein Netzwerkpaket erstellt haben. Nachdem Sie eine Netzwerkinstanz erstellt haben, instanziieren Sie sie.

Console

Um eine Netzwerkinstanz mit der Konsole zu erstellen

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich Netzwerke aus.
3. Wählen Sie Netzwerkinstanz erstellen aus.
4. Geben Sie einen Namen und eine Beschreibung für die Instanz ein und wählen Sie dann Weiter.
5. Wählen Sie das Netzwerkpaket aus, überprüfen Sie die Details und klicken Sie auf Weiter.
6. Wählen Sie Netzwerkinstanz erstellen.

   Die neue Netzwerkinstanz wird auf der Seite Netzwerke angezeigt. Als Nächstes instanziieren Sie diese Netzwerkinstanz.

AWS CLI

Um eine Netzwerkinstanz mit dem zu erstellen AWS CLI

- Verwenden Sie den create-sol-network-instanceBefehl, um eine Netzwerkinstanz zu erstellen.

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

Als Nächstes instanziieren Sie diese Netzwerkinstanz.

Instanziieren Sie eine Netzwerkinstanz mithilfe von TNB AWS

Nachdem Sie eine Netzwerkinstanz erstellt haben, müssen Sie sie instanziieren. Wenn Sie eine Netzwerkinstanz instanziieren, stellt AWS TNB die erforderliche AWS Infrastruktur bereit, stellt containerisierte Netzwerkfunktionen bereit und konfiguriert die Netzwerk- und Zugriffsverwaltung, um einen voll funktionsfähigen Netzwerkdienst zu schaffen.

Console

Um eine Netzwerkinstanz mithilfe der Konsole zu instanziieren

1. Öffnen Sie die AWS TNB-Konsole unter. https://console.aws.amazon.com/tnb/
2. Wählen Sie im Navigationsbereich Netzwerke aus.
3. Wählen Sie die Netzwerkinstanz aus, die Sie instanziieren möchten.
4. Wählen Sie Aktionen und dann Instanziieren.
5. Überprüfen Sie auf der Seite Netzwerk instanziieren die Details und aktualisieren Sie optional die Parameterwerte.

   Aktualisierungen der Parameterwerte gelten nur für diese Netzwerkinstanz. Die Parameter in den NSD- und VNFD-Paketen ändern sich nicht.
6. Wählen Sie Netzwerk instantiieren.

   Die Seite mit dem Bereitstellungsstatus wird angezeigt.
7. Verwenden Sie das Aktualisierungssymbol, um den Bereitstellungsstatus Ihrer Netzwerkinstanz zu verfolgen. Sie können die automatische Aktualisierung auch im Bereich Bereitstellungsaufgaben aktivieren, um den Fortschritt der einzelnen Aufgaben zu verfolgen.

Wenn sich der Bereitstellungsstatus auf ändert`Completed`, wird die Netzwerkinstanz instanziiert.

AWS CLI

Um eine Netzwerkinstanz mit dem zu instanziieren AWS CLI

1. Verwenden Sie den instantiate-sol-network-instanceBefehl, um die Netzwerkinstanz zu instanziieren.

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

2. Sehen Sie sich als Nächstes den Status des Netzwerkbetriebs an.

Aktualisieren Sie eine Funktionsinstanz in AWS TNB

Nachdem eine Netzwerkinstanz instanziiert wurde, können Sie ein Funktionspaket in der Netzwerkinstanz aktualisieren.

Console

Um eine Funktionsinstanz mithilfe der Konsole zu aktualisieren

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich Netzwerke aus.
3. Wählen Sie die Netzwerkinstanz aus. Sie können eine Netzwerkinstanz nur aktualisieren, wenn ihr Status `Instantiated`

   Die Seite mit der Netzwerkinstanz wird angezeigt.
4. Wählen Sie auf der Registerkarte Funktionen die zu aktualisierende Funktionsinstanz aus.
5. Wählen Sie Aktualisieren aus.
6. Geben Sie Ihre Aktualisierungsüberschreibungen ein.
7. Wählen Sie Aktualisieren aus.

AWS CLI

Verwenden Sie die CLI, um eine Funktionsinstanz zu aktualisieren

Verwenden Sie den update-sol-network-instanceBefehl mit dem `MODIFY_VNF_INFORMATION` Aktualisierungstyp, um eine Funktionsinstanz in einer Netzwerkinstanz zu aktualisieren.

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

Aktualisieren Sie eine Netzwerkinstanz in AWS TNB

Nachdem eine Netzwerkinstanz instanziiert wurde, müssen Sie möglicherweise die Infrastruktur oder Anwendung aktualisieren. Dazu aktualisieren Sie das Netzwerkpaket und die Parameterwerte für die Netzwerkinstanz und führen den Aktualisierungsvorgang durch, um die Änderungen zu übernehmen.

Überlegungen

- Sie können eine Netzwerkinstanz aktualisieren, die sich im `Updated` Status `Instantiated` oder befindet.
- Wenn Sie eine Netzwerkinstanz aktualisieren, verwendet die `UpdateSolNetworkService` API das neue Netzwerkpaket und die Parameterwerte, um die Topologie der Netzwerkinstanz zu aktualisieren.
- AWS TNB überprüft, ob die Anzahl der NSD- und VNFD-Parameter in der Netzwerkinstanz 200 nicht überschreitet. Dieses Limit wird durchgesetzt, um zu verhindern, dass böswillige Akteure fehlerhafte oder riesige Payloads weitergeben, die den Service beeinträchtigen.

Parameter, die Sie aktualisieren können

Sie können die folgenden Parameter aktualisieren, wenn Sie eine instanziierte Netzwerkinstanz aktualisieren:

Parameter	Description	Beispiel: Vorher	Beisp Nachl
Amazon EKS-Clusterversion	Sie können den Wert für den `version` Parameter	`EKSCluster:`	`EKSC` `r:`

Parameter	Description	Beispiel: Vorher	Beisp Nachl
	der Amazon EKS-Cluster-Steuerebene auf die nächste Nebenversion aktualisieren. Sie können die Version nicht herabstufen.	<pre>type: tosca.nod es.AWS.Compute.EKS properties: version: "1.28"</pre>	<pre>typ tos es.A mput pro s: ver "1.</pre>

Parameter	Description	Beispiel: Vorher	Beisp Nachl
Amazon EKS-Worker-Knoten	Sie können den Wert für den `EKSManagedNode` `kubernetes_version` Parameter aktualisieren, um Ihre Knotengruppe auf eine neuere Amazon EKS-Version zu aktualisieren, oder Sie können den `ami_id` Parameter aktualisieren, um Ihre Knotengruppe auf das neueste EKS-optimierte AMI zu aktualisieren. Sie können die AMI-ID für aktualisieren`EKSSelfMa nagedNode` . Die Amazon EKS-Version des AMI muss mit der Amazon EKS-Cluster-Version identisch sein oder bis zu 2 Versionen niedriger sein. Wenn die Amazon EKS-Clust er-Version beispielsweise 1.31 ist, muss die Amazon EKS AMI-Version 1.31, 1.30 oder 1.29 sein.	`EKSManagedNodeGroup01: ... properties: kubernete s_version: " 1.28" EKSSelfManagedNode 01: compute: compute: properties: ami_id: "ami-1231230LD "`	`EKSM dNoc p01: ... pro s: kub s_ve : "1. EKS nage 01: com`

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			com pro s: ami "am 3NEW

Parameter	Description	Beispiel: Vorher	Beisp Nachl

Amazon EKS-Knotengruppen	Sie können je nach Ihren Rechenanforderungen Knotengruppen hinzufügen oder entfernen.
	
	Stellen Sie beim Löschen vorhandener Knotengruppen und beim Hinzufügen neuer Knotengruppen sicher, dass die neuen Knotengruppen andere IDs als die gelöschte n Knotengruppen haben. Andernfalls wird der Vorgang als Änderung einer Knotengru ppe behandelt und nicht als Löschen und Hinzufüge n. Beachten Sie, dass für bestehende Knotengruppen nur ein begrenzter Satz von Parametern aktualisiert werden kann. Scrollen Sie durch diese Tabelle, um zu sehen, welche Parameter Sie aktualisieren können.
		`Free5GCEKSNode01:`
		`  type: tosca.nod`
		`es.AWS.Compute.EKS`
		`ManagedNode`
		`  ...`
		`    scaling:`
		`      properties:`
		`        desired_size: 1`
		`        min_size: 1`
		`        max_size: 1`
		`  ...`
		`Free5GCEKSNode02 : #`
		` Deleted Nodegroup`
		`  type: tosca.nod`
		`es.AWS.Compute.EKS`
		`ManagedNode`
		`  ...`
		`    scaling:`
		`      properties:`
		`        desired_size: 1`
		`        min_size: 1`
		`        max_size: 1`
		`  ...`
		`Free5GCEKSNode03 : #`
		` Deleted Nodegroup`
		`  type: tosca.nod`
		`es.AWS.Compute.EKS`
		`SelfManagedNode`
		`  ...`
		`    scaling:`
		`      properties:`
		`        desired_size: 1`
		`        min_size: 1`
		`        max_size: 1`
		`  ...`
			`Free`
			`SNod`
			
			` typ`
			` tos`
			`es.A`
			`mput`
			`Mana`
			`de`
			
			` ...`
			
			` sca`
			
			` pro`
			`s:`
			
			` des`
			`ize:`
			` 1`

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			mir 1 max 1 ... Free SNod # New Nod typ tos es.A mput Self edNo ... sca pro s:

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			des ize: 1 mir 1 max 1 ... Free SNod # New Nod typ tos es.A mput Mana de

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			... sca pro s: des ize: 1 mir 1 max 1

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			...

Parameter	Description	Beispiel: Vorher	Beisp Nachl

Eigenschaften skalieren	Sie können die Skalierun gseigenschaften der Knoten `EKSManagedNode` und `EKSSelfManagedNode` TOSCA aktualisieren.	`EKSNodeGroup01:`
		`  ...`
		`  scaling:`
		`  properties:`
		`  desired_s ize: 1`
		`  min_size: 1`
		`  max_size: 1`
			`EKSN oup0`
			`...`
			`sca`
			`pro s:`
			`des ize:`

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			mir max

Parameter	Description	Beispiel: Vorher	Beisp Nachl

Eigenschaften des Amazon EBS CSI-Plug-ins	Sie können das Amazon EBS CSI-Plugin auf Ihren Amazon EKS-Clustern aktivieren oder deaktivieren. Sie können auch die Plugin-Version ändern.	`EKSCluster:`
		`  capabilities:`
		`  ...`
		`  ebs_csi:`
		`  properties:`
		`  enabled: false`
			`EKSC`
			`r:`
			`cap`
			`ies:`
			`...`
			`ebs`
			`pro`
			`s:`
			`ena`
			`ver`
			`"v1`
			`e`
			`ksbu`
			`"`

Parameter	Description	Beispiel: Vorher	Beisp Nachl

Root-Volume-Größe	Sie können die Eigenscha ft „Größe des Root-Volu mes“ der EKSManaged Knoten Node und EKSSelf ManagedNode TOSCA hinzufügen, entfernen oder aktualisieren.	`Free5GCEKSNode01:`
		`  ...`
		`  capabilities:`
		`  compute:`
		`  properties:`
		`  root_volu me_size: 50`
			`Free SNod`
			`...`
			`cap ies:`
			`com`
			`pro s:`

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			roo me_s

Parameter	Description	Beispiel: Vorher	Beisp Nachl
VNF	Sie können VNFs im NSD auf sie verweisen und sie mithilfe des TOSCA-Knotens auf dem in NSD erstellten Cluster bereitstellen. VNFDeployment Im Rahmen des Updates können Sie Inhalte zum Netzwerk hinzufügen, aktualisieren und löschen VNFs .	vnfds: - descriptor_id: "43c012fa-2616-41a8-a833-0dfd4c5a049e " namespace: " vnf1" - descriptor_id: "64222f98-ecd6-4871-bf94-7354b53f3ee5 " namespace: "vnf2" // Deleted VNF ... SampleVNF1HelmDeploy: type: tosca.nodes.AWS.Deployment.VNFDeployment requirements: cluster: EKSCluster vnfs: - vnf1.SampleVNF1 - vnf2.SampleVNF2	vnfd - des r_id "59 79e9 - be53 2ad0 " nam : "vr Upd VNF - des r_id "b7 839d -916 a166 " nam : "vr Add VNF .... Sa mple

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			elmD :
			typ tos es.A ploy VNFD ment
			rec nts:
			clu EKS r
			vnf

Parameter	Description	Beispiel: Vorher	Beisp Nachh
			- v leVN - v leVN

Parameter	Description	Beispiel: Vorher	Beisp Nachl

Hook	Um Lebenszyklusoperationen vor und nach dem Erstellen einer Netzwerkfunktion auszuführen, fügen Sie dem `VNFDeployment` Knoten die `post_create` Hooks `pre_create` und hinzu.
	
	In diesem Beispiel wird der `PreCreateHook` Hook ausgeführt, bevor er `vnf3.SampleVNF3` instanziiert wurde, und der `PostCreateHook` Hook wird ausgeführt, nachdem `vnf3.SampleVNF3` er instanziiert wurde.
		`vnfds:`
		`  - descriptor_id:`
		`  "43c012fa-2616-41a8-a833-0dfd4c5a049e  "`
		`  namespace: " vnf1"`
		`  - descriptor_id:`
		`  "64222f98-ecd6-4871-bf94-7354b53f3ee5  "`
		`  namespace: " vnf2"`
		`...`
		`SampleVNF1HelmDeploy:`
		`  type: tosca.nodes.AWS.Deployment.VNFDeployment`
		`  requirements:`
		`  cluster: EKSCluster`
		`  vnfs:`
		`  - vnf1.SampleVNF1`
		`  - vnf2.SampleVNF2  // Removed during update`
			vnfc
			
			-
			des
			r_ic
			"43
			2616
			-
			a833
			d4c5
			"
			
			nam
			:
			"vr
			
			-
			des
			r_ic
			"b7
			839c
			-916
			a166
			"
			
			nam
			:
			"vr
			....
			S
			ampl
			Helm
			y:
			
			typ
			tos

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			es.A ploy VNFD ment rec nts: clu EKS r vnf - v leVN No cha to thi fur as the nam and uui rem the sam

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			- v leVN New VNF as the nam , vnf was not pre y pre int s: Hoo pos te: eHoo pre e: Hook

Parameter	Description	Beispiel: Vorher	Beisp Nachl

Hook	Um Lebenszyklusoperationen vor und nach dem Update einer Netzwerkfunktion auszuführen, können Sie den `pre_update` Hook und den `post_update` Hook dem Knoten hinzufügen. `VNFDeployment`
	
	In diesem Beispiel ist `PreUpdateHook` will run before `vnf1.SampleVNF1` is updated und `PostUpdateHook` will run after `vnf1.SampleVNF1` ist das Paket, das durch das aktualisierte `vnf` Paket `uuid` für den Namespace vnf1 gekennzeichnet ist.
		`vnfds:`
		`  - descriptor_id: "43c012fa-2616-41a8-a833-0dfd4c5a049e "`
		`    namespace: " vnf1"`
		`  - descriptor_id: "64222f98-ecd6-4871-bf94-7354b53f3ee5 "`
		`    namespace: " vnf2"`
		`...`
		
		`SampleVNF1HelmDeploy:`
		`  type: tosca.nodes.AWS.Deployment.VNFDeployment`
		`  requirements:`
		`    cluster: EKSCluster`
		`    vnfs:`
		`      - vnf1.SampleVNF1`
		`      - vnf2.SampleVNF2`
			`vnfc`
			
			` -`
			` des`
			`r_ic`
			` "0e`
			`bd87`
			`-`
			`b8a1`
			`4666`
			` "`
			
			` nam`
			`:`
			` "vr`
			
			` -`
			` des`
			`r_ic`
			` "64`
			`ecd6`
			`-`
			`bf94`
			`4b53`
			` "`
			
			` nam`
			`:`
			` "vr`
			`...`
			`S`
			`ampl`
			`Helm`
			`y:`
			
			` typ`

Parameter	Description	Beispiel: Vorher	Beisp Nachh
			tos es.A ploy VNFD ment rec nts: clu EKS r vnf - v leVN A VNF upd as the uui cha for nam "vr - v

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			leVN No cha to thi fur as nam and uui rem the sam int s: Hoo pre e: Hook pos te: eHoo

Parameter	Description	Beispiel: Vorher	Beisp Nachl
Subnets	Sie können Subnetze zum Netzwerk hinzufügen und daraus löschen. Stellen Sie vor dem Löschen eines Subnetzes sicher, dass das Subnetz von keiner Ressource im Netzwerk verwendet wird.	*Free5GCSubnet01* : #Deleted Subnet type: tosca.nodes.AWS.Networking.Subnet properties: type: "PUBLIC" availability_zone: { get_input: subnet_01_az } cidr_block: { get_input: subnet_01_cidr_block } requirements: route_table: Free5GCRouteTable vpc: Free5GCVPC	*Free* *bnet* #Ne Sub typ tos es.A twor Subn pro s: typ "PU ava ity_ { g : sub _az cid k: { g : sub _cid ck } req nts:

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			rou le: Fre uteT vpc Fre C

Parameter	Description	Beispiel: Vorher	Beisp Nachl
Sicherheitsgruppen	Sie können Sicherhei tsgruppen zum Netzwerk hinzufügen und daraus löschen. Stellen Sie vor dem Löschen einer Sicherhei tsgruppe sicher, dass die Sicherheitsgruppe von keiner Ressource im Netzwerk verwendet wird.	*Free5GCSecurityGro up01* : #Deleted Security Group type: tosca.nod es.AWS.Networking. SecurityGroup properties: description: "SecurityGroup for Free5GC cluster" name: "Free5GCS ecurityGroup01" tags: - "Name=Fre e5GCEKSAdditionalS ecurityGroup" requirements: vpc: Free5GCVPC *Free5GCSecurityGro upEgressRule01* : #Deleted Security Group Egress Node type: tosca.nod es.AWS.Networking. SecurityGroupEgres sRule properties: ip_protocol: "tcp" from_port: 8000 to_port: 9000 description: "Egress Rule for free5GC cluster" cidr_ip : "172.10.1 0.1/24" requirements: security_group: Free5GCSecurityGro up01	*Free* *curi* *up02* #Ne Sec Gro typ tos es.A twor Secu roup pro s: des on: "Se Grou for Fre clu nam "Fr ecur oup0 tag

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

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			ip_ ol: "to fro : 800 to_ 900 des on: "Eg Rul for fre clu cid "17 0.1/ req nts: sec grou Fre

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			curi up02 *Free* *curi* *upIr* *Rule* #Ne Sec Gro Ing Noc typ tos es.A twor Secu roup ssRu pro s: ip_ ol: "to fro : 800 to_ 900

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			des on: "Ir Rul for fre clu cid "17 0.1/ rec nts: sec grou Fre curi up02

Parameter	Description	Beispiel: Vorher	Beisp Nachl

Netzwerkschnittstellen	Sie können dem Netzwerk hinzufügen, es ändern und ENIs aus dem Netzwerk löschen.	`Free5GCENI01: #Modified ENI`
		`  type: tosca.nodes.AWS.Networking.ENI`
		`  properties:`
		`  device_index: 2`
		`  requirements:`
		`  subnet: Free5GCENISubnet01`
		`  security_groups:`
		`  - Free5GCSecurityGroup01`
		
		`Free5GCENI02: #Modified ENI`
		`  type: tosca.nodes.AWS.Networking.ENI`
		`  properties:`
		`  device_index: 3`
		`  source_dest_check: true`
		`  requirements:`
		`  subnet: Free5GCENISubnet01`
		
		`Free5GCENI04 : #Deleted ENI`
		`  type: tosca.nodes.AWS.Networking.ENI`
		`  properties:`
		`  device_index: 4`
		`  source_dest_check: true`
		`  requirements:`
		`  subnet: Free5GCENISubnet01`
			`Free`
			`I01:`
			`  #Mo`
			`  ENI`
			
			`  typ`
			`  tos`
			`es.A`
			`twor`
			`ENI`
			
			`  pro`
			`s:`
			
			`  dev`
			`dex:`
			`  2`
			
			`  req`
			`nts:`
			
			`  sub`
			`ISub`
			
			`  sec`
			`grou`
			
			`  -`
			`  Fre`

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			curi up01 Fre e5GC : #Mc ENI typ tos es.A twor ENI pro s: dev dex: 3 sou st_c tru req nts: sub Fre ISub se grou

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			- Fre curi up01 Free I03 #Ne ENI typ tos es.A twor ENI pro s: dev dex: 3 rec nts: sub Fre bnet sec grou

Parameter	Description	Beispiel: Vorher	Beisp Nachl
			- Fre curi up01

Eine Netzwerkinstanz aktualisieren

Console

Um eine Netzwerkinstanz mithilfe der Konsole zu aktualisieren

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich Netzwerke aus.
3. Wählen Sie die Netzwerkinstanz aus. Sie können eine Netzwerkinstanz nur aktualisieren, wenn ihr Status `Instantiated` oder ist`Updated`.
4. Wählen Sie Aktionen und Update aus.

   Die Seite „Instanz aktualisieren“ wird mit den Netzwerkdetails und einer Liste von Parametern in der aktuellen Infrastruktur angezeigt.
5. Wählen Sie ein neues Netzwerkpaket aus.

   Die Parameter des neuen Netzwerkpakets werden im Abschnitt Aktualisierte Parameter angezeigt.
6. Aktualisieren Sie optional die Parameterwerte im Abschnitt Aktualisierte Parameter. Eine Liste der Parameterwerte, die Sie aktualisieren können, finden Sie unterParameter, die Sie aktualisieren können.
7. Wählen Sie Netzwerk aktualisieren.

   AWS TNB validiert die Anfrage und startet die Bereitstellung. Die Seite mit dem Bereitstellungsstatus wird angezeigt.

8. Verwenden Sie das Aktualisierungssymbol, um den Bereitstellungsstatus Ihrer Netzwerkinstanz zu verfolgen. Sie können die automatische Aktualisierung auch im Bereich Bereitstellungsaufgaben aktivieren, um den Fortschritt der einzelnen Aufgaben zu verfolgen.

   Wenn sich der Bereitstellungsstatus auf ändert`Completed`, wird die Netzwerkinstanz aktualisiert.

9.
   - Schlägt die Überprüfung fehl, verbleibt die Netzwerkinstanz im gleichen Zustand wie vor der Anforderung des Updates — entweder `Instantiated` oder`Updated`.
   - Schlägt das Update fehl, wird der Status der Netzwerkinstanz angezeigt`Update failed`. Wählen Sie den Link für jede fehlgeschlagene Aufgabe, um den Grund zu ermitteln.
   - Wenn das Update erfolgreich ist, wird der Status der Netzwerkinstanz angezeigt`Updated`.

AWS CLI

Verwenden Sie die CLI, um eine Netzwerkinstanz zu aktualisieren

Verwenden Sie den update-sol-network-instanceBefehl mit dem `UPDATE_NS` Aktualisierungstyp, um eine Netzwerkinstanz zu aktualisieren.

```
aws tnb update-sol-network-instance --ns-instance-id ^ni-[a-f0-9]{17}$ --
update-type UPDATE_NS --update-ns "{\"nsdInfoId\":\"^np-[a-f0-9]{17}$\",
 \"additionalParamsForNs\": {\"param1\": \"value1\"}}"
```

Eine Netzwerkinstanz in AWS TNB anzeigen

Erfahren Sie, wie Sie eine Netzwerkinstanz anzeigen.

Console

So zeigen Sie eine Netzwerkinstanz mithilfe der Konsole an

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich Netzwerkinstanzen aus.
3. Verwenden Sie das Suchfeld, um die Netzwerkinstanz zu finden.

AWS CLI

Um eine Netzwerkinstanz mit dem anzuzeigen AWS CLI

1. Verwenden Sie den list-sol-network-instancesBefehl, um Ihre Netzwerkinstanzen aufzulisten.

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

2. Verwenden Sie den get-sol-network-instanceBefehl, um Details zu einer bestimmten Netzwerkinstanz anzuzeigen.

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

Beenden und löschen Sie eine Netzwerkinstanz aus AWS TNB

Um eine Netzwerkinstanz zu löschen, muss sich die Instance in einem beendeten Zustand befinden.

Console

Um eine Netzwerkinstanz mithilfe der Konsole zu beenden und zu löschen

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich Netzwerke aus.
3. Wählen Sie die ID der Netzwerkinstanz aus.
4. Wähen Sie Beenden.
5. Wenn Sie zur Bestätigung aufgefordert werden, geben Sie die ID ein und wählen Sie Terminate.
6. Aktualisieren Sie, um den Status Ihrer Netzwerkinstanz zu verfolgen.
7. (Optional) Wählen Sie die Netzwerkinstanz aus und klicken Sie auf Löschen.

AWS CLI

Um eine Netzwerkinstanz zu beenden und zu löschen, verwenden Sie AWS CLI

1. Verwenden Sie den terminate-sol-network-instanceBefehl, um eine Netzwerkinstanz zu beenden.

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

2. (Optional) Verwenden Sie den delete-sol-network-instanceBefehl, um eine Netzwerkinstanz zu löschen.

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

Netzwerkbetrieb für AWS TNB

Ein Netzwerkvorgang ist jeder Vorgang, der in Ihrem Netzwerk ausgeführt wird, z. B. die Instanziierung oder Beendigung einer Netzwerkinstanz.

Aufgaben

- Sehen Sie sich einen AWS TNB-Netzwerkvorgang an
- Brechen Sie einen AWS TNB-Netzwerkvorgang ab

Sehen Sie sich einen AWS TNB-Netzwerkvorgang an

Sehen Sie sich die Details eines Netzwerkvorgangs an, einschließlich der mit dem Netzwerkbetrieb verbundenen Aufgaben und des Status der Aufgaben.

Console

So zeigen Sie einen Netzwerkvorgang mit der Konsole an

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich Netzwerkinstanzen aus.
3. Verwenden Sie das Suchfeld, um die Netzwerkinstanz zu finden.
4. Wählen Sie auf der Registerkarte Bereitstellungen den Netzwerkvorgang aus.

AWS CLI

Um einen Netzwerkvorgang mit dem anzuzeigen AWS CLI

1. Verwenden Sie den list-sol-network-operationsBefehl, um alle Netzwerkoperationen aufzulisten.

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

2. Verwenden Sie den get-sol-network-operationBefehl, um Details zu einem Netzwerkvorgang anzuzeigen.

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

Brechen Sie einen AWS TNB-Netzwerkvorgang ab

Erfahren Sie, wie Sie einen Netzwerkvorgang abbrechen.

Console

So brechen Sie einen Netzwerkvorgang mithilfe der Konsole ab

1. Öffnen Sie die AWS TNB-Konsole unter https://console.aws.amazon.com/tnb/.
2. Wählen Sie im Navigationsbereich Netzwerke aus.
3. Wählen Sie die ID des Netzwerks aus, um die zugehörige Detailseite zu öffnen.
4. Wählen Sie auf der Registerkarte Bereitstellungen den Netzwerkbetrieb aus.
5. Wählen Sie Vorgang abbrechen aus.

AWS CLI

Um einen Netzwerkvorgang abzubrechen, verwenden Sie AWS CLI

Verwenden Sie den cancel-sol-network-operationBefehl, um einen Netzwerkvorgang abzubrechen.

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

TOSCA-Referenz für TNB AWS

Die Topologie- und Orchestrierungsspezifikation für Cloud-Anwendungen (TOSCA) ist eine deklarative Syntax, die CSPs verwendet wird, um eine Topologie von Cloud-basierten Webdiensten, ihren Komponenten, Beziehungen und den Prozessen, mit denen sie verwaltet werden, zu beschreiben. CSPs beschreiben die Verbindungspunkte, die logischen Verknüpfungen zwischen den Verbindungspunkten und die Richtlinien wie Affinität und Sicherheit in einer TOSCA-Vorlage. CSPs laden Sie dann die Vorlage auf AWS TNB hoch, wo die Ressourcen zusammengefasst werden, die für den Aufbau eines funktionierenden 5G-Netzwerks in allen Availability Zones erforderlich sind. AWS

Inhalt

VNFD-Vorlage

Definiert eine Vorlage für einen virtuellen Netzwerkfunktionsdeskriptor (VNFD).

Syntax

```
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
```

Topologie-Vorlage

`node_templates`

Die TOSCA-Knoten. AWS Die möglichen Knoten sind:

- AWS.VNF
- AWS.Artefakte.Helm

AWS.VNF

Definiert einen VNF-Knoten (AWS Virtual Network Function).

Syntax

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

Eigenschaften

`descriptor_id`

Die UUID des Deskriptors.

Erforderlich: Ja

Typ: Zeichenfolge

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

`descriptor_version`

Die Version des VNFD.

Erforderlich: Ja

Typ: Zeichenfolge

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

`descriptor_name`

Der Name des Deskriptors.

Erforderlich: Ja

Typ: Zeichenfolge

`provider`

Der Autor des VNFD.

Erforderlich: Ja

Typ: Zeichenfolge

Voraussetzungen

`helm`

Das Helm-Verzeichnis, das Container-Artefakte definiert. Dies ist ein Verweis auf AWS.Artifacts.Helm.

Erforderlich: Ja

Typ: Zeichenfolge

Beispiel

```
SampleVNF:
  type: tosca.nodes.AWS.VNF
  properties:
    descriptor_id: "6a792e0c-be2a-45fa-989e-5f89d94ca898"
    descriptor_version: "1.0.0"
    descriptor_name: "Test VNF Template"
    provider: "Operator"
  requirements:
    helm: SampleHelm
```

AWS.Artifacts.Helm

Definiert einen AWS Helm-Knoten.

Syntax

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

Eigenschaften

`implementation`

Das lokale Verzeichnis, das das Helm-Diagramm im CSAR-Paket enthält.

Erforderlich: Ja

Typ: Zeichenfolge

Beispiel

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

Vorlage für Netzwerkdienst-Deskriptoren

Definiert eine NSD-Vorlage (Network Service Descriptor).

Syntax

```
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
```

Verwendung definierter Parameter

Wenn Sie einen Parameter dynamisch übergeben möchten, z. B. den CIDR-Block für den VPC-Knoten, können Sie die `{ get_input: input-parameter-name }` Syntax verwenden und die Parameter in der NSD-Vorlage definieren. Verwenden Sie den Parameter dann in derselben NSD-Vorlage erneut.

Das folgende Beispiel zeigt, wie Parameter definiert und verwendet werden:

```
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-Import

`descriptor_id`

Die UUID des Deskriptors.

Erforderlich: Ja

Typ: Zeichenfolge

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

`namespace`

Der eindeutige Name.

Erforderlich: Ja

Typ: Zeichenfolge

Topologie-Vorlage

`node_templates`

Die möglichen AWS TOSCA-Knoten sind:

- AWS.NS
- AWS.compute.eks
- AWS.Compute.EKS. AuthRole
- AWS.Rechnen. EKSManagedKnoten
- AWS. Berechnen. EKSSelfManagedNode
- AWS.Berechnen. PlacementGroup
- AWS.Rechnen. UserData
- AWS.Netzwerke. SecurityGroup
- AWS.Netzwerke. SecurityGroupEgressRule
- AWS.Netzwerke. SecurityGroupIngressRule
- AWS.Ressource.Import

- AWS.Netzwerking.ENI
- AWS.HookExecution
- AWS.Netzwerke. InternetGateway
- AWS.Netzwerke. RouteTable
- AWS.Netzwerk.Subnetz
- AWS.Bereitstellung. VNFDeployment
- AWS.Netzwerk.VPC
- AWS.Netzwerke. NATGateway
- AWS.Netzwerken.Route

AWS.NS

Definiert einen AWS Netzwerkdienstknoten (NS).

Syntax

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

Eigenschaften

`descriptor_id`

Die UUID des Deskriptors.

Erforderlich: Ja

Typ: Zeichenfolge

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

`descriptor_version`

Die Version des NSD.

Erforderlich: Ja

Typ: Zeichenfolge

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

`descriptor_name`

Der Name des Deskriptors.

Erforderlich: Ja

Typ: Zeichenfolge

Beispiel

```
SampleNS:
  type: tosca.nodes.AWS.NS
  properties:
    descriptor_id: "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
    descriptor_version: "1.0.0"
    descriptor_name: "Test NS Template"
```

AWS.compute.EKS

Geben Sie den Namen des Clusters, die gewünschte Kubernetes-Version und eine Rolle an, die es der Kubernetes-Steuerebene ermöglicht, die für Sie erforderlichen AWS Ressourcen zu verwalten. NFs Multus Container Network Interface (CNI) -Plugins sind aktiviert. Sie können mehrere Netzwerkschnittstellen anhängen und eine erweiterte Netzwerkkonfiguration auf die Kubernetes-basierten Netzwerkfunktionen anwenden. Sie geben auch den Cluster-Endpunktzugriff und die Subnetze für Ihren Cluster an.

Syntax

```
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
```

Funktionen

`multus`

Optional. Eigenschaften, die die Verwendung des Multus Container Network Interface (CNI) definieren.

Wenn Sie einschließen`multus`, geben Sie die Eigenschaften `enabled` und `multus_role` an.

`enabled`

Gibt an, ob die standardmäßige Multus-Funktion aktiviert ist.

Erforderlich: Ja

Typ: Boolesch

`multus_role`

Die Rolle für die Multus-Netzwerkschnittstellenverwaltung.

Erforderlich: Ja

Typ: Zeichenfolge

`ebs_csi`

Eigenschaften, die den im Amazon EKS-Cluster installierten Amazon EBS Container Storage Interface (CSI) -Treiber definieren.

Aktivieren Sie dieses Plugin, um selbstverwaltete Amazon EKS-Nodes auf AWS Outposts, AWS Local Zones oder AWS-Regionen zu verwenden. Weitere Informationen finden Sie unter Amazon Elastic Block Store CSI-Treiber im Amazon EKS-Benutzerhandbuch.

`enabled`

Zeigt an, ob der standardmäßige Amazon EBS CSI-Treiber installiert ist.

Erforderlich: Nein

Typ: Boolesch

`version`

Die Version des Amazon EBS CSI-Treiber-Add-ons. Die Version muss mit einer der von der DescribeAddonVersionsAktion zurückgegebenen Versionen übereinstimmen. Weitere Informationen finden Sie DescribeAddonVersionsin der Amazon EKS API-Referenz

Erforderlich: Nein

Typ: Zeichenfolge

Eigenschaften

`version`

Die Kubernetes-Version für den Cluster. AWS Telco Network Builder unterstützt die Kubernetes-Versionen 1.25 bis 1.32.

Erforderlich: Ja

Typ: Zeichenfolge

Mögliche Werte: 1,25 | 1,26 | 1,27 | 1,28 | 1,29 | 1,30 | 1,31 | 1,32

`access`

Der Zugriff auf den Cluster-Endpunkt.

Erforderlich: Ja

Typ: Zeichenfolge

Mögliche Werte: `PRIVATE` | `PUBLIC` | `ALL`

`cluster_role`

Die Rolle für die Clusterverwaltung.

Erforderlich: Ja

Typ: Zeichenfolge

`tags`

Tags, die an die Ressource angehängt werden sollen.

Erforderlich: Nein

Typ: Liste

`ip_family`

Gibt die IP-Familie für Service- und Pod-Adressen im Cluster an.

Zulässiger Wert:`IPv4`, `IPv6`

Standardwert: `IPv4`

Erforderlich: Nein

Typ: Zeichenfolge

Voraussetzungen

`subnets`

Ein AWS.Networking.Subnet-Knoten.

Erforderlich: Ja

Typ: Liste

Beispiel

```
SampleEKS:
  type: tosca.nodes.AWS.Compute.EKS
  properties:
    version: "1.26"
```

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

AWS.compute.EKS. AuthRole

An AuthRole ermöglicht es Ihnen, dem Amazon EKS-Cluster IAM-Rollen hinzuzufügen, `aws-auth ConfigMap` sodass Benutzer mithilfe einer IAM-Rolle auf den Amazon EKS-Cluster zugreifen können.

Syntax

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

Eigenschaften

`role_mappings`

Liste der Zuordnungen, die IAM-Rollen definieren, die dem Amazon EKS-Cluster hinzugefügt werden müssen. `aws-auth ConfigMap`

`arn`

Der ARN der IAM-Rolle.

Erforderlich: Ja

Typ: Zeichenfolge

`groups`

Kubernetes-Gruppen, die der in definierten Rolle zugewiesen werden sollen. `arn`

Erforderlich: Nein

Typ: Liste

Voraussetzungen

`clusters`

Ein AWS.compute.EKS-Knoten.

Erforderlich: Ja

Typ: Liste

Beispiel

```
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.Berechnen. EKSManagedKnoten

AWS TNB unterstützt EKS Managed Node-Gruppen, um die Bereitstellung und das Lebenszyklusmanagement von Knoten (EC2 Amazon-Instances) für Amazon EKS Kubernetes-Cluster zu automatisieren. Gehen Sie wie folgt vor, um eine EKS-Knotengruppe zu erstellen:

- Wählen Sie die Amazon Machine Images (AMI) für Ihre Cluster-Worker-Knoten aus, indem Sie entweder die AMI-ID oder den AMI-Typ angeben.
- Geben Sie ein EC2 Amazon-Schlüsselpaar für den SSH-Zugriff und die Skalierungseigenschaften für Ihre Knotengruppe an.
- Stellen Sie sicher, dass Ihre Knotengruppe mit einem Amazon EKS-Cluster verknüpft ist.
- Stellen Sie die Subnetze für die Worker-Knoten bereit.
- Fügen Sie Ihrer Knotengruppe optional Sicherheitsgruppen, Knotenbezeichnungen und eine Platzierungsgruppe hinzu.

Syntax

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

```
    kubernetes_version: String
  requirements:
    cluster: String
    subnets: List
    network_interfaces: List
    security_groups: List
    placement_group: String
    user_data: String
    labels: List
```

Funktionen

`compute`

Eigenschaften, die die Rechenparameter für die von Amazon EKS verwaltete Knotengruppe definieren, z. B. EC2 Amazon-Instance-Typen und EC2 Amazon-Instances AMIs.

`ami_type`

Der von Amazon EKS unterstützte AMI-Typ.

Erforderlich: Ja

Typ: Zeichenfolge

Mögliche Werte: `AL2_x86_64` | | `AL2_x86_64_GPU` | `AL2_ARM_64` | | `AL2023_x86_64` | `AL2023_ARM_64` | `AL2023_x86_64_NVIDIA` | `AL2023_x86_64_NEURON` | | `CUSTOM` | `BOTTLEROCKET_ARM_64` | `BOTTLEROCKET_x86_64` `BOTTLEROCKET_ARM_64_NVIDIA` `BOTTLEROCKET_x86_64_NVIDIA`

`ami_id`

Die ID des AMI.

Erforderlich: Nein

Typ: Zeichenfolge

> **Note**
>
> Wenn beide `ami_type` und in der Vorlage angegeben `ami_id` sind, verwendet AWS TNB nur den `ami_id` Wert für die Erstellung`EKSManagedNode`.

`instance_types`

Die Instanzgröße.

Erforderlich: Ja

Typ: Liste

`key_pair`

Das EC2 Schlüsselpaar zur Aktivierung des SSH-Zugriffs.

Erforderlich: Ja

Typ: Zeichenfolge

`root_volume_encryption`

Aktiviert die Amazon EBS-Verschlüsselung für das Amazon EBS-Root-Volume. Wenn diese Eigenschaft nicht angegeben wird, verschlüsselt AWS TNB standardmäßig Amazon EBS-Root-Volumes.

Erforderlich: Nein

Standard: true

Typ: Boolesch

`root_volume_encryption_key_arn`

Der ARN des AWS KMS Schlüssels. AWS TNB unterstützt reguläre Schlüssel-ARN, Multi-Region-Schlüssel-ARN und Alias-ARN.

Erforderlich: Nein

Typ: Zeichenfolge

> **Note**
>
> - Wenn der Wert falsch `root_volume_encryption` ist, schließen Sie ihn nicht ein. `root_volume_encryption_key_arn`
> - AWS TNB unterstützt die Root-Volume-Verschlüsselung von Amazon EBS-gestützten AMIs.

- Wenn das Root-Volume des AMI bereits verschlüsselt ist, müssen Sie das `root_volume_encryption_key_arn` für AWS TNB hinzufügen, um das Root-Volume erneut zu verschlüsseln.
- Wenn das Root-Volume des AMI nicht verschlüsselt ist, verwendet AWS TNB das, `root_volume_encryption_key_arn` um das Root-Volume zu verschlüsseln.

  Wenn Sie dies nicht angeben`root_volume_encryption_key_arn`, verwendet AWS TNB den von bereitgestellten Standardschlüssel, um das Root-Volume AWS Key Management Service zu verschlüsseln.
- AWS TNB entschlüsselt kein verschlüsseltes AMI.

`root_volume_size`

Die Größe des Amazon Elastic Block Store-Root-Volumes in GiBs.

Erforderlich: Nein

Standard: 20

Typ: Ganzzahl

Mögliche Werte: 1 bis 16.384

`scaling`

Eigenschaften, die die Skalierungsparameter für die von Amazon EKS verwaltete Knotengruppe definieren, z. B. die gewünschte Anzahl von EC2 Amazon-Instances und die Mindest- und Höchstanzahl von EC2 Amazon-Instances in der Knotengruppe.

`desired_size`

Die Anzahl der Instances in dieser NodeGroup.

Erforderlich: Ja

Typ: Ganzzahl

`min_size`

Die Mindestanzahl von Instanzen in diesem Bereich NodeGroup.

Erforderlich: Ja

Typ: Ganzzahl

`max_size`

Die maximale Anzahl von Instanzen in diesem Bereich NodeGroup.

Erforderlich: Ja

Typ: Ganzzahl

Eigenschaften

`node_role`

Der ARN der IAM-Rolle, die der EC2 Amazon-Instance zugeordnet ist.

Erforderlich: Ja

Typ: Zeichenfolge

`tags`

Die Tags, die an die Ressource angehängt werden sollen.

Erforderlich: Nein

Typ: Liste

`kubernetes_version`

Die Kubernetes-Version für die Managed Node-Gruppe. AWS TNB unterstützt die Kubernetes-Versionen 1.25 bis 1.32. Berücksichtigen Sie dabei Folgendes:

- Geben Sie entweder oder an. `kubernetes_version` `ami_id` Geben Sie nicht beides an.
- Der `kubernetes_version` muss kleiner oder gleich dem AWS.Compute sein. EKSManagedKnotenversion.
- Es kann einen Unterschied von 3 Versionen zwischen den Versionen von AWS.Compute geben. EKSManagedKnotenversion und. `kubernetes_version`
- Wenn weder das `kubernetes_version` oder `ami_id` angegeben ist, verwendet AWS TNB das neueste AMI der `AWS.Compute.EKSManagedNode` Version zum Erstellen `EKSManagedNode`

Erforderlich: Nein

Typ: Zeichenfolge

Mögliche Werte: 1,25 | 1,26 | 1,27 | 1,28 | 1,29 | 1,30 | 1,31 | 1,32

Voraussetzungen

`cluster`

Ein AWS.compute.EKS-Knoten.

Erforderlich: Ja

Typ: Zeichenfolge

`subnets`

Ein .Networking.Subnet-Knoten.AWS

Erforderlich: Ja

Typ: Liste

`network_interfaces`

Ein .Networking.ENI-Knoten.AWS Stellen Sie sicher, dass die Netzwerkschnittstellen und Subnetze auf dieselbe Availability Zone eingestellt sind. Andernfalls schlägt die Instanziierung fehl.

Wenn Sie diese Eigenschaft festlegen`network_interfaces`, erhält AWS TNB die entsprechende Berechtigung für die Eigenschaft, sofern Sie die `multus_role` Eigenschaft in ENIs den AWS.Compute.eks-Knoten aufgenommen haben. `multus` Andernfalls erhält AWS TNB die entsprechende Berechtigung von der Eigenschaft node_role. ENIs

Erforderlich: Nein

Typ: Liste

`security_groups`

Ein .Networking.AWS SecurityGroupKnoten.

Erforderlich: Nein

Typ: Liste

`placement_group`

Ein tosca.nodes.AWS. Rechnen. PlacementGroupKnoten.

Erforderlich: Nein

Typ: Zeichenfolge

`user_data`

Ein tosca.nodes.AWS. Rechnen. UserDataKnotenreferenz. Ein Benutzerdatenskript wird an die EC2 Amazon-Instances übergeben, die von der verwalteten Knotengruppe gestartet wurden. Fügen Sie der an die Knotengruppe übergebenen node_role die Berechtigungen hinzu, die für die Ausführung benutzerdefinierter Benutzerdaten erforderlich sind.

Erforderlich: Nein

Typ: Zeichenfolge

`labels`

Eine Liste von Knotenbezeichnungen. Eine Knotenbezeichnung muss einen Namen und einen Wert haben. Erstellen Sie ein Label anhand der folgenden Kriterien:

- Der Name und der Wert müssen durch getrennt werden=.
- Der Name und der Wert können jeweils bis zu 63 Zeichen lang sein.
- Das Etikett kann Buchstaben (A-Z, a-z,), Zahlen (0-9) und die folgenden Zeichen enthalten: `[-, _, ., *, ?]`
- Der Name und der Wert müssen mit einem alphanumerischen Zeichen oder beginnen und enden. `? *`

Beispiel: `myLabelName1=*NodeLabelValue1`

Erforderlich: Nein

Typ: Liste

Beispiel

```
SampleEKSManagedNode:
  type: tosca.nodes.AWS.Compute.EKSManagedNode
  capabilities:
```

```
    compute:
      properties:
        ami_type: "AL2_x86_64"
        instance_types:
          - "t3.xlarge"
        key_pair: "SampleKeyPair"
        root_volume_encryption: true
        root_volume_encryption_key_arn: "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        root_volume_size: 1500
    scaling:
      properties:
        desired_size: 1
        min_size: 1
        max_size: 1
  properties:
    node_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleRole"
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
    kubernetes_version:
      - "1.30"
  requirements:
    cluster: SampleEKS
    subnets:
      - SampleSubnet
    network_interfaces:
      - SampleENI01
      - SampleENI02
    security_groups:
      - SampleSecurityGroup01
      - SampleSecurityGroup02
    placement_group: SamplePlacementGroup
    user_data: CustomUserData
    labels:
      - "sampleLabelName001=sampleLabelValue001"
      - "sampleLabelName002=sampleLabelValue002"
```

AWS. Berechne. EKSSelfManagedNode

AWS TNB unterstützt selbstverwaltete Amazon EKS-Knoten, um die Bereitstellung und das Lebenszyklusmanagement von Knoten (EC2 Amazon-Instances) für Amazon EKS Kubernetes-Cluster zu automatisieren. Gehen Sie wie folgt vor, um eine Amazon EKS-Knotengruppe zu erstellen:

- Wählen Sie die Amazon Machine Images (AMI) für Ihre Cluster-Worker-Knoten aus, indem Sie entweder die AMI-ID angeben.
- Stellen Sie ein EC2 Amazon-Schlüsselpaar für den SSH-Zugriff bereit.
- Stellen Sie sicher, dass Ihre Knotengruppe mit einem Amazon EKS-Cluster verknüpft ist.
- Geben Sie den Instance-Typ und die gewünschten Mindest- und Maximalgrößen an.
- Stellen Sie die Subnetze für die Worker-Knoten bereit.
- Fügen Sie Ihrer Knotengruppe optional Sicherheitsgruppen, Knotenbezeichnungen und eine Platzierungsgruppe hinzu.

Syntax

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

Funktionen

compute

Eigenschaften, die die Rechenparameter für die selbstverwalteten Amazon EKS-Knoten definieren, z. B. EC2 Amazon-Instance-Typen und EC2 Amazon-Instances AMIs.

`ami_id`

Die AMI-ID, die zum Starten der Instance verwendet wurde. AWS TNB unterstützt Instances, die diese Funktion nutzen IMDSv2. Weitere Informationen finden Sie unter IMDS-Version.

> **Note**
>
> Sie können die AMI-ID für aktualisieren`EKSSelfManagedNode`. Die Amazon EKS-Version des AMI muss mit der Amazon EKS-Cluster-Version identisch sein oder bis zu 2 Versionen niedriger sein. Wenn die Amazon EKS-Cluster-Version beispielsweise 1.31 ist, muss die Amazon EKS AMI-Version 1.31, 1.30 oder 1.29 sein.

Erforderlich: Ja

Typ: Zeichenfolge

`instance_type`

Die Instance-Größe.

Erforderlich: Ja

Typ: Zeichenfolge

`key_pair`

Das EC2 Amazon-Schlüsselpaar zur Aktivierung des SSH-Zugriffs.

Erforderlich: Ja

Typ: Zeichenfolge

`root_volume_encryption`

Aktiviert die Amazon EBS-Verschlüsselung für das Amazon EBS-Root-Volume. Wenn diese Eigenschaft nicht angegeben wird, verschlüsselt AWS TNB standardmäßig Amazon EBS-Root-Volumes.

Erforderlich: Nein

Standard: true

Typ: Boolesch

`root_volume_encryption_key_arn`

Der ARN des AWS KMS Schlüssels. AWS TNB unterstützt reguläre Schlüssel-ARN, Multi-Region-Schlüssel-ARN und Alias-ARN.

Erforderlich: Nein

Typ: Zeichenfolge

> **Note**
>
> - Wenn der Wert falsch `root_volume_encryption` ist, schließen Sie ihn nicht ein. `root_volume_encryption_key_arn`
> - AWS TNB unterstützt die Root-Volume-Verschlüsselung von Amazon EBS-gestützten AMIs.
> - Wenn das Root-Volume des AMI bereits verschlüsselt ist, müssen Sie das `root_volume_encryption_key_arn` für AWS TNB hinzufügen, um das Root-Volume erneut zu verschlüsseln.
> - Wenn das Root-Volume des AMI nicht verschlüsselt ist, verwendet AWS TNB das, `root_volume_encryption_key_arn` um das Root-Volume zu verschlüsseln.
>
> Wenn Sie dies nicht angeben`root_volume_encryption_key_arn`, verwendet AWS TNB das AWS Managed Services Root-Volume zur Verschlüsselung.
> - AWS TNB entschlüsselt kein verschlüsseltes AMI.

`root_volume_size`

Die Größe des Amazon Elastic Block Store-Root-Volumes in GiBs.

Erforderlich: Nein

Standard: 20

Typ: Ganzzahl

Mögliche Werte: 1 bis 16.384

scaling

Eigenschaften, die die Skalierungsparameter für die selbstverwalteten Amazon EKS-Knoten definieren, z. B. die gewünschte Anzahl von EC2 Amazon-Instances und die Mindest- und Höchstanzahl von EC2 Amazon-Instances in der Knotengruppe.

`desired_size`

Die Anzahl der Instances in dieser NodeGroup.

Erforderlich: Ja

Typ: Ganzzahl

`min_size`

Die Mindestanzahl von Instanzen in diesem Bereich NodeGroup.

Erforderlich: Ja

Typ: Ganzzahl

`max_size`

Die maximale Anzahl von Instanzen in diesem Bereich NodeGroup.

Erforderlich: Ja

Typ: Ganzzahl

Eigenschaften

`node_role`

Der ARN der IAM-Rolle, die der EC2 Amazon-Instance zugeordnet ist.

Erforderlich: Ja

Typ: Zeichenfolge

`tags`

Die Tags, die an die Ressource angehängt werden sollen. Die Tags werden an die von der Ressource erstellten Instanzen weitergegeben.

Erforderlich: Nein

Typ: Liste

Voraussetzungen

`cluster`

Ein AWS.compute.EKS-Knoten.

Erforderlich: Ja

Typ: Zeichenfolge

`subnets`

Ein .Networking.Subnet-Knoten.AWS

Erforderlich: Ja

Typ: Liste

`network_interfaces`

Ein .Networking.ENI-Knoten.AWS Stellen Sie sicher, dass die Netzwerkschnittstellen und Subnetze auf dieselbe Availability Zone eingestellt sind. Andernfalls schlägt die Instanziierung fehl.

Wenn Sie diese Eigenschaft festlegen`network_interfaces`, erhält AWS TNB die entsprechende Berechtigung für die Eigenschaft, sofern Sie die `multus_role` Eigenschaft in ENIs den AWS.Compute.eks-Knoten aufgenommen haben. `multus` Andernfalls erhält AWS TNB die entsprechende Berechtigung von der Eigenschaft node_role. ENIs

Erforderlich: Nein

Typ: Liste

`security_groups`

Ein .Networking.AWS SecurityGroupKnoten.

Erforderlich: Nein

Typ: Liste

`placement_group`

Ein tosca.nodes.AWS. Rechnen. PlacementGroupKnoten.

Erforderlich: Nein

Typ: Zeichenfolge

`user_data`

Ein tosca.nodes.AWS. Rechnen. UserDataKnotenreferenz. Ein Benutzerdatenskript wird an die EC2 Amazon-Instances übergeben, die von der selbstverwalteten Knotengruppe gestartet wurden. Fügen Sie der an die Knotengruppe übergebenen node_role die Berechtigungen hinzu, die für die Ausführung benutzerdefinierter Benutzerdaten erforderlich sind.

Erforderlich: Nein

Typ: Zeichenfolge

`labels`

Eine Liste von Knotenbezeichnungen. Eine Knotenbezeichnung muss einen Namen und einen Wert haben. Erstellen Sie ein Label anhand der folgenden Kriterien:

- Der Name und der Wert müssen durch getrennt werden=.
- Der Name und der Wert können jeweils bis zu 63 Zeichen lang sein.
- Die Bezeichnung kann Buchstaben (A-Z, a-z,), Zahlen (0-9) und die folgenden Zeichen enthalten: `[-, _, ., *, ?]`
- Der Name und der Wert müssen mit einem alphanumerischen Zeichen oder beginnen und enden. `? *`

Beispiel: `myLabelName1=*NodeLabelValue1`

Erforderlich: Nein

Typ: Liste

Beispiel

```
SampleEKSSelfManagedNode:
  type: tosca.nodes.AWS.Compute.EKSSelfManagedNode
```

```
  capabilities:
    compute:
      properties:
        ami_id: "ami-123123EXAMPLE"
        instance_type: "c5.large"
        key_pair: "SampleKeyPair"
        root_volume_encryption: true
        root_volume_encryption_key_arn: "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        root_volume_size: 1500
    scaling:
      properties:
        desired_size: 1
        min_size: 1
        max_size: 1
  properties:
    node_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleNodeRole"
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
  requirements:
    cluster: SampleEKSCluster
    subnets:
      - SampleSubnet
    network_interfaces:
      - SampleNetworkInterface01
      - SampleNetworkInterface02
    security_groups:
      - SampleSecurityGroup01
      - SampleSecurityGroup02
    placement_group: SamplePlacementGroup
    user_data: CustomUserData
    labels:
      - "sampleLabelName001=sampleLabelValue001"
      - "sampleLabelName002=sampleLabelValue002"
```

AWS. Berechne. PlacementGroup

Ein PlacementGroup Knoten unterstützt verschiedene Strategien zum Platzieren von EC2 Amazon-Instances.

Wenn Sie ein neues Amazon starten EC2instance, versucht der EC2 Amazon-Service, die Instance so zu platzieren, dass alle Ihre Instances auf die zugrunde liegende Hardware verteilt sind, um

korrelierte Ausfälle zu minimieren. Mithilfe von Placement-Gruppen können Sie die Platzierung einer Gruppe von untereinander abhängigen Instances beeinflussen, um den Anforderungen Ihres Workloads nachzukommen.

Syntax

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

Eigenschaften

`strategy`

Die Strategie, mit der EC2 Amazon-Instances platziert werden sollen.

Erforderlich: Ja

Typ: Zeichenfolge

Mögliche Werte: CLUSTER | PARTITION | SPREAD_HOST | SPREAD_RACK

- CLUSTER — fasst Instanzen nahe beieinander in einer Availability Zone zusammen. Diese Strategie ermöglicht es Workloads, die Netzwerkleistung mit niedriger Latenz zu erreichen, die für eng gekoppelte node-to-node Kommunikation erforderlich ist, wie sie für HPC-Anwendungen (High Performance Computing) typisch ist.
- PARTITION — verteilt Ihre Instances auf logische Partitionen, sodass Gruppen von Instanzen in einer Partition die zugrunde liegende Hardware nicht gemeinsam mit Gruppen von Instanzen in verschiedenen Partitionen nutzen. Diese Strategie wird in der Regel für große verteilte und replizierte Workloads wie Hadoop, Cassandra und Kafka verwendet.
- SPREAD_RACK — platziert eine kleine Gruppe von Instanzen auf unterschiedlicher zugrunde liegender Hardware, um korrelierte Ausfälle zu reduzieren.
- SPREAD_HOST — wird nur mit Outpost-Platzierungsgruppen verwendet. Platziert eine kleine Gruppe von Instances auf unterschiedlicher zugrundeliegender Hardware, um korrelierte Ausfälle zu reduzieren.

`partition_count`

Die Anzahl an Partitionen.

Erforderlich: Nur erforderlich, wenn auf `PARTITION` gesetzt `strategy` ist.

Typ: Ganzzahl

Mögliche Werte: 1 | 2 | 3 | 4 | 5 | 6 | 7

`tags`

Die Tags, die Sie der Platzierungsgruppenressource zuordnen können.

Erforderlich: Nein

Typ: Liste

Beispiel

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

AWS. Berechnen. UserData

AWS TNB unterstützt das Starten von EC2 Amazon-Instances mit benutzerdefinierten Benutzerdaten über den UserData Knoten im Network Service Descriptor (NSD). Weitere Informationen zu benutzerdefinierten Benutzerdaten finden Sie unter Benutzerdaten und Shell-Skripts im EC2 Amazon-Benutzerhandbuch.

Während der Netzwerkinstanziierung stellt AWS TNB die EC2 Amazon-Instance-Registrierung für den Cluster über ein Benutzerdatenskript bereit. Wenn auch benutzerdefinierte Benutzerdaten bereitgestellt werden, führt AWS TNB beide Skripte zusammen und gibt sie als Multimime-Skript an Amazon weiter. EC2 Das benutzerdefinierte Benutzerdatenskript wird vor dem Amazon EKS-Registrierungsskript ausgeführt.

Um benutzerdefinierte Variablen im Benutzerdatenskript zu verwenden, fügen Sie `!` nach der geöffneten geschweiften Klammer ein Ausrufezeichen hinzu. `{` Um es beispielsweise `MyVariable` im Skript zu verwenden, geben Sie Folgendes ein: `{!MyVariable}`

> **Note**
>
> - AWS TNB unterstützt Benutzerdatenskripten mit einer Größe von bis zu 7 KB.
> - Da AWS TNB das `multimime` Benutzerdatenskript verarbeitet und rendert, sollten Sie sicherstellen, dass das Skript alle Regeln einhält. CloudFormation CloudFormation

Syntax

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

Eigenschaften

`implementation`

Der relative Pfad zur Benutzerdatenskriptdefinition. Das Format muss wie folgt sein: `./scripts/script_name.sh`

Erforderlich: Ja

Typ: Zeichenfolge

`content_type`

Inhaltstyp des Benutzerdatenskripts.

Erforderlich: Ja

Typ: Zeichenfolge

Mögliche Werte: `x-shellscript`

Beispiel

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

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

AWS.Netzwerke. SecurityGroup

AWS TNB unterstützt Sicherheitsgruppen, um die Bereitstellung von EC2Amazon-Sicherheitsgruppen zu automatisieren, die Sie Amazon EKS Kubernetes-Cluster-Knotengruppen zuordnen können.

Syntax

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

Eigenschaften

`description`

Die Beschreibung der Sicherheitsgruppe. Sie können bis zu 255 Zeichen verwenden, um die Gruppe zu beschreiben. Sie können nur Buchstaben (A-Z und a-z), Zahlen (0-9), Leerzeichen und die folgenden Sonderzeichen verwenden: ._-:/() #, @ [] +=&; {}! $*

Erforderlich: Ja

Typ: Zeichenfolge

`name`

Ein Name für die Sicherheitsgruppe. Sie können bis zu 255 Zeichen für den Namen verwenden. Sie können nur Buchstaben (A-Z und a-z), Zahlen (0-9), Leerzeichen und die folgenden Sonderzeichen verwenden: ._-:/() #, @ [] +=&; {}! $*

Erforderlich: Ja

Typ: Zeichenfolge

`tags`

Die Tags, die Sie an die Sicherheitsgruppenressource anhängen können.

Erforderlich: Nein

Typ: Liste

Voraussetzungen

`vpc`

Ein AWS.networking.VPC-Knoten.

Erforderlich: Ja

Typ: Zeichenfolge

Beispiel

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

AWS.Netzwerke. SecurityGroupEgressRule

AWS TNB unterstützt Ausgangsregeln für Sicherheitsgruppen, um die Bereitstellung von Amazon EC2 Security Group Egress Rules zu automatisieren, die an .Networking angehängt werden können. AWS SecurityGroup. Beachten Sie, dass Sie eine cidr_ ip/destination_security_group/destination _prefix_list als Ziel für ausgehenden Datenverkehr angeben müssen.

Syntax

```
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
```

Eigenschaften

`cidr_ip`

Der Adressbereich im CIDR-Format. IPv4 Sie müssen einen CIDR-Bereich angeben, der ausgehenden Datenverkehr zulässt.

Erforderlich: Nein

Typ: Zeichenfolge

`cidr_ipv6`

Der IPv6 Adressbereich im CIDR-Format für ausgehenden Verkehr. Sie müssen eine Zielsicherheitsgruppe (`destination_security_group` oder `destination_prefix_list`) oder einen CIDR-Bereich (`cidr_ip` oder `cidr_ipv6`) angeben.

Erforderlich: Nein

Typ: Zeichenfolge

`description`

Die Beschreibung einer Sicherheitsgruppenregel für ausgehenden Datenverkehr. Sie können bis zu 255 Zeichen verwenden, um die Regel zu beschreiben.

Erforderlich: Nein

Typ: Zeichenfolge

`destination_prefix_list`

Die Präfixlisten-ID einer bestehenden von Amazon VPC verwalteten Präfixliste. Dies ist das Ziel von Knotengruppen-Instances, die der Sicherheitsgruppe zugeordnet sind. Weitere Informationen zu verwalteten Präfixlisten finden Sie unter Verwaltete Präfixlisten im Amazon VPC-Benutzerhandbuch.

Erforderlich: Nein

Typ: Zeichenfolge

`from_port`

Wenn das Protokoll TCP oder UDP ist, ist dies der Anfang des Portbereichs. Wenn das Protokoll ICMP oder ist ICMPv6, ist dies die Typnummer. Ein Wert von -1 steht für alle ICMP/ICMPv6 types. If you specify all ICMP/ICMPv6 types, you must specify all ICMP/ICMPv 6 Codes.

Erforderlich: Nein

Typ: Ganzzahl

`ip_protocol`

Der IP-Protokollname (tcp, udp, icmp, icmpv6) oder die Protokollnummer. Verwenden Sie -1, um alle Protokolle anzugeben. Bei der Autorisierung von Sicherheitsgruppenregeln ermöglicht die Angabe von -1 oder einer anderen Protokollnummer als tcp, udp, icmp oder icmpv6 den Verkehr auf allen Ports, unabhängig vom angegebenen Portbereich. Für tcp, udp und icmp müssen Sie einen Portbereich angeben. Für icmpv6 ist der Portbereich optional. Wenn Sie den Portbereich weglassen, ist Verkehr für alle Typen und Codes zulässig.

Erforderlich: Ja

Typ: Zeichenfolge

`to_port`

Wenn das Protokoll TCP oder UDP ist, ist dies das Ende des Portbereichs. Wenn das Protokoll ICMP oder ist ICMPv6, ist dies der Code. Ein Wert von -1 steht für alle ICMP/ICMPv6 codes. If you specify all ICMP/ICMPv6 types, you must specify all ICMP/ICMPv 6 Codes.

Erforderlich: Nein

Typ: Ganzzahl

Voraussetzungen

`security_group`

Die ID der Sicherheitsgruppe, zu der diese Regel hinzugefügt werden soll.

Erforderlich: Ja

Typ: Zeichenfolge

`destination_security_group`

Die ID oder TOSCA-Referenz der Zielsicherheitsgruppe, zu der ausgehender Datenverkehr zugelassen ist.

Erforderlich: Nein

Typ: Zeichenfolge

Beispiel

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

AWS.Netzwerke. SecurityGroupIngressRule

AWS TNB unterstützt Ingress-Regeln für Sicherheitsgruppen, um die Bereitstellung von Amazon EC2 Security Group Ingress Rules zu automatisieren, die an .Networking angehängt werden können. AWS SecurityGroup. Beachten Sie, dass Sie eine cidr_ ip/source_security_group/source _prefix_list als Quelle für eingehenden Datenverkehr angeben müssen.

Syntax

```
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
```

Eigenschaften

`cidr_ip`

Der Adressbereich im CIDR-Format. IPv4 Sie müssen einen CIDR-Bereich angeben, der eingehenden Datenverkehr zulässt.

Erforderlich: Nein

Typ: Zeichenfolge

`cidr_ipv6`

Der IPv6 Adressbereich im CIDR-Format für eingehenden Verkehr. Sie müssen eine Quellsicherheitsgruppe (`source_security_group` oder `source_prefix_list`) oder einen CIDR-Bereich (`cidr_ip` oder `cidr_ipv6`) angeben.

Erforderlich: Nein

Typ: Zeichenfolge

`description`

Die Beschreibung einer Sicherheitsgruppenregel für eingehenden (eingehenden) Datenverkehr. Sie können bis zu 255 Zeichen verwenden, um die Regel zu beschreiben.

Erforderlich: Nein

Typ: Zeichenfolge

`source_prefix_list`

Die Präfixlisten-ID einer bestehenden von Amazon VPC verwalteten Präfixliste. Dies ist die Quelle, von der Knotengruppen-Instances, die der Sicherheitsgruppe zugeordnet sind, Datenverkehr empfangen dürfen. Weitere Informationen zu verwalteten Präfixlisten finden Sie unter Verwaltete Präfixlisten im Amazon VPC-Benutzerhandbuch.

Erforderlich: Nein

Typ: Zeichenfolge

`from_port`

Wenn das Protokoll TCP oder UDP ist, ist dies der Anfang des Portbereichs. Wenn das Protokoll ICMP oder ist ICMPv6, ist dies die Typnummer. Ein Wert von -1 steht für alle ICMP/ICMPv6 types. If you specify all ICMP/ICMPv6 types, you must specify all ICMP/ICMPv 6 Codes.

Erforderlich: Nein

Typ: Ganzzahl

`ip_protocol`

Der IP-Protokollname (tcp, udp, icmp, icmpv6) oder die Protokollnummer. Verwenden Sie -1, um alle Protokolle anzugeben. Bei der Autorisierung von Sicherheitsgruppenregeln ermöglicht die Angabe von -1 oder einer anderen Protokollnummer als tcp, udp, icmp oder icmpv6 den Verkehr auf allen Ports, unabhängig vom angegebenen Portbereich. Für tcp, udp und icmp müssen Sie einen Portbereich angeben. Für icmpv6 ist der Portbereich optional. Wenn Sie den Portbereich weglassen, ist Verkehr für alle Typen und Codes zulässig.

Erforderlich: Ja

Typ: Zeichenfolge

`to_port`

Wenn das Protokoll TCP oder UDP ist, ist dies das Ende des Portbereichs. Wenn das Protokoll ICMP oder ist ICMPv6, ist dies der Code. Ein Wert von -1 steht für alle ICMP/ICMPv6 codes. If you specify all ICMP/ICMPv6 types, you must specify all ICMP/ICMPv 6 Codes.

Erforderlich: Nein

Typ: Ganzzahl

Voraussetzungen

`security_group`

Die ID der Sicherheitsgruppe, zu der diese Regel hinzugefügt werden soll.

Erforderlich: Ja

Typ: Zeichenfolge

`source_security_group`

Die ID oder TOSCA-Referenz der Quellsicherheitsgruppe, von der eingehender Datenverkehr zugelassen werden soll.

Erforderlich: Nein

Typ: Zeichenfolge

Beispiel

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

AWS.Ressource.Import

Sie können die folgenden AWS Ressourcen in AWS TNB importieren:

- VPC
- Subnetz
- Routing-Tabelle
- Internet Gateway
- Sicherheitsgruppe

Syntax

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

Eigenschaften

`resource_type`

Der Ressourcentyp, der in AWS TNB importiert wird.

Erforderlich: Nein

Typ: Liste

`resource_id`

Die Ressourcen-ID, die in AWS TNB importiert wird.

Erforderlich: Nein

Typ: Liste

Beispiel

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

AWS.networking.ENI

Eine Netzwerkschnittstelle ist eine logische Netzwerkkomponente in einer VPC, die eine virtuelle Netzwerkkarte darstellt. Einer Netzwerkschnittstelle wird anhand ihres Subnetzes entweder automatisch oder manuell eine IP-Adresse zugewiesen. Nachdem Sie eine EC2 Amazon-Instance in einem Subnetz bereitgestellt haben, können Sie ihr eine Netzwerkschnittstelle hinzufügen oder eine Netzwerkschnittstelle von dieser EC2 Amazon-Instance trennen und erneut eine Verbindung zu einer anderen EC2 Amazon-Instance in diesem Subnetz herstellen. Der Geräteindex identifiziert die Position in der Reihenfolge der Dateianhänge.

Syntax

```
tosca.nodes.AWS.Networking.ENI:
  properties:
    device_index: Integer
```

```
    source_dest_check: Boolean
    tags: List
  requirements:
    subnet: String
    security_groups: List
```

Eigenschaften

`device_index`

Der Geräteindex muss größer als Null sein.

Erforderlich: Ja

Typ: Ganzzahl

`source_dest_check`

Gibt an, ob die Netzwerkschnittstelle eine Quell-/Zielüberprüfung durchführt. Der Wert `true` bedeutet, dass die Prüfung aktiviert ist, und der Wert `false` bedeutet, dass die Prüfung deaktiviert ist.

Zulässiger Wert: wahr, falsch

Standard: true

Erforderlich: Nein

Typ: Boolesch

`tags`

Die Tags, die an die Ressource angehängt werden sollen.

Erforderlich: Nein

Typ: Liste

Voraussetzungen

`subnet`

Ein AWS.Networking.Subnet-Knoten.

Erforderlich: Ja

Typ: Zeichenfolge

`security_groups`

Ein AWS.Networking. SecurityGroupKnoten.

Erforderlich: Nein

Typ: Zeichenfolge

Beispiel

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

AWS.HookExecution

Ein Lifecycle-Hook bietet Ihnen die Möglichkeit, Ihre eigenen Skripts als Teil Ihrer Infrastruktur und Netzwerkinstanziierung auszuführen.

Syntax

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

```
    vpc: String
```

Funktionen

`execution`

Eigenschaften für die Hook-Ausführungs-Engine, die die Hook-Skripte ausführt.

`type`

Der Typ der Hook-Ausführungs-Engine.

Erforderlich: Nein

Typ: Zeichenfolge

Mögliche Werte: `CODE_BUILD`

Voraussetzungen

`definition`

Ein AWS. HookDefinition.Bash-Knoten.

Erforderlich: Ja

Typ: Zeichenfolge

`vpc`

Ein AWS.networking.VPC-Knoten.

Erforderlich: Ja

Typ: Zeichenfolge

Beispiel

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

AWS.Netzwerke. InternetGateway

Definiert einen AWS Internet-Gateway-Knoten.

Syntax

```
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
```

Funktionen

routing

Eigenschaften, die die Routing-Verbindung innerhalb der VPC definieren. Sie müssen entweder die `ipv6_dest_cidr` Eigenschaft `dest_cidr` oder angeben.

`dest_cidr`

Der IPv4 CIDR-Block, der für die Zielübereinstimmung verwendet wurde. Diese Eigenschaft wird verwendet, um eine Route in zu erstellen, `RouteTable` und ihr Wert wird als. `DestinationCidrBlock`

Erforderlich: Nein, wenn Sie die `ipv6_dest_cidr` Eigenschaft angegeben haben.

Typ: Zeichenfolge

`ipv6_dest_cidr`

Der IPv6 CIDR-Block, der für die Zielübereinstimmung verwendet wurde.

Erforderlich: Nein, wenn Sie die `dest_cidr` Eigenschaft angegeben haben.

Typ: Zeichenfolge

Eigenschaften

`tags`

Die Tags, die an die Ressource angehängt werden sollen.

Erforderlich: Nein

Typ: Liste

`egress_only`

Eine IPv6 -spezifische Eigenschaft. Gibt an, ob das Internet-Gateway nur für die ausgehende Kommunikation vorgesehen ist oder nicht. Wenn wahr `egress_only` ist, müssen Sie die `ipv6_dest_cidr` Eigenschaft definieren.

Erforderlich: Nein

Typ: Boolesch

Voraussetzungen

`vpc`

Ein AWS.networking.VPC-Knoten.

Erforderlich: Ja

Typ: Zeichenfolge

`route_table`

Ein AWS.Networking. RouteTableKnoten.

Erforderlich: Ja

Typ: Zeichenfolge

Beispiel

```
Free5GCIGW:
  type: tosca.nodes.AWS.Networking.InternetGateway
  properties:
```

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

AWS. Netzwerke. RouteTable

Eine Routentabelle enthält eine Reihe von Regeln, die als Routen bezeichnet werden und bestimmen, wohin der Netzwerkverkehr von Subnetzen innerhalb Ihrer VPC oder Ihres Gateways geleitet wird. Sie müssen einer VPC eine Routing-Tabelle zuordnen.

Syntax

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

Eigenschaften

`tags`

Tags, die an die Ressource angehängt werden sollen.

Erforderlich: Nein

Typ: Liste

Voraussetzungen

`vpc`

Ein AWS.networking.VPC-Knoten.

Erforderlich: Ja

Typ: Zeichenfolge

Beispiel

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

AWS.Networking.Subnet

Ein Subnetz ist ein Bereich von IP-Adressen in Ihrer VPC, der sich vollständig innerhalb einer Availability Zone befinden muss. Sie müssen eine VPC, einen CIDR-Block, eine Availability Zone und eine Routing-Tabelle für Ihr Subnetz angeben. Sie müssen auch definieren, ob Ihr Subnetz privat oder öffentlich ist.

Syntax

```
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
```

Eigenschaften

`type`

Gibt an, ob in diesem Subnetz gestartete Instances eine öffentliche IPv4 Adresse erhalten.

Erforderlich: Ja

Typ: Zeichenfolge

Mögliche Werte: `PUBLIC` | `PRIVATE`

`availability_zone`

Die Availability Zone für das Subnetz. Dieses Feld unterstützt AWS Availability Zones innerhalb einer AWS Region, zum Beispiel `us-west-2` (USA West (Oregon)). Es unterstützt beispielsweise auch AWS Local Zones innerhalb der Availability Zone`us-west-2-lax-1a`.

Erforderlich: Ja

Typ: Zeichenfolge

`cidr_block`

Der CIDR-Block für das Subnetz.

Erforderlich: Nein

Typ: Zeichenfolge

`ipv6_cidr_block`

Der CIDR-Block, der zur Erstellung des Subnetzes verwendet wurde. IPv6 Wenn Sie diese Eigenschaft angeben, schließen Sie sie nicht ein. `ipv6_cidr_block_suffix`

Erforderlich: Nein

Typ: Zeichenfolge

`ipv6_cidr_block_suffix`

Das zweistellige hexadezimale Suffix des IPv6 CIDR-Blocks für das über Amazon VPC erstellte Subnetz. Verwenden Sie das folgende Format: *`2-digit hexadecimal`*`::/`*`subnetMask`*

Wenn Sie diese Eigenschaft angeben, schließen Sie sie nicht ein. `ipv6_cidr_block`

Erforderlich: Nein

Typ: Zeichenfolge

`outpost_arn`

Der ARN, in dem AWS Outposts das Subnetz erstellt wird. Fügen Sie diese Eigenschaft zur NSD-Vorlage hinzu, wenn Sie selbstverwaltete Amazon EKS-Knoten auf starten möchten. AWS Outposts Weitere Informationen finden Sie unter Amazon EKS on AWS Outposts im Amazon EKS-Benutzerhandbuch.

Wenn Sie diese Eigenschaft zur NSD-Vorlage hinzufügen, müssen Sie den Wert für die `availability_zone` Eigenschaft auf die Availability Zone von festlegen. AWS Outposts

Erforderlich: Nein

Typ: Zeichenfolge

`tags`

Die Tags, die an die Ressource angehängt werden sollen.

Erforderlich: Nein

Typ: Liste

Voraussetzungen

`vpc`

Ein AWS.networking.VPC-Knoten.

Erforderlich: Ja

Typ: Zeichenfolge

`route_table`

Ein AWS.Networking. RouteTableKnoten.

Erforderlich: Ja

Typ: Zeichenfolge

Beispiel

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

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

AWS. Einsatz. VNFDeployment

NF-Bereitstellungen werden modelliert, indem die Infrastruktur und die damit verbundene Anwendung bereitgestellt werden. Das Clusterattribut gibt den EKS-Cluster an, der Ihren hosten soll. NFs Das vnfs-Attribut gibt die Netzwerkfunktionen für Ihre Bereitstellung an. Sie können auch optionale Lifecycle-Hook-Operationen vom Typ pre_create und post_create bereitstellen, um Anweisungen auszuführen, die für Ihre Bereitstellung spezifisch sind, z. B. das Aufrufen einer API für das Inventarverwaltungssystem.

Syntax

```
tosca.nodes.AWS.Deployment.VNFDeployment:
  requirements:
    deployment: String
```

```
    cluster: String
    vnfs: List
  interfaces:
    Hook:
      pre_create: String
      post_create: String
```

Voraussetzungen

`deployment`

Ein .Deployment.AWS VNFDeploymentKnoten.

Erforderlich: Nein

Typ: Zeichenfolge

`cluster`

Ein AWS.compute.EKS-Knoten.

Erforderlich: Ja

Typ: Zeichenfolge

`vnfs`

Ein .VNF-Knoten.AWS

Erforderlich: Ja

Typ: Zeichenfolge

Schnittstellen

Haken

Definiert die Phase, in der Lifecycle-Hooks ausgeführt werden.

`pre_create`

Ein AWS. HookExecutionKnoten. Dieser Hook wird ausgeführt, bevor der `VNFDeployment` Knoten bereitgestellt wird.

Erforderlich: Nein

Typ: Zeichenfolge

`post_create`

Ein AWS. HookExecutionKnoten. Dieser Hook wird ausgeführt, nachdem der `VNFDeployment` Knoten bereitgestellt wurde.

Erforderlich: Nein

Typ: Zeichenfolge

Beispiel

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

AWS.networking.VPC

Sie müssen einen CIDR-Block für Ihre Virtual Private Cloud (VPC) angeben.

Syntax

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

Eigenschaften

`cidr_block`

Der IPv4 Netzwerkbereich für die VPC in CIDR-Notation.

Erforderlich: Ja

Typ: Zeichenfolge

`ipv6_cidr_block`

Der IPv6 CIDR-Block, der zur Erstellung der VPC verwendet wurde.

Zulässiger Wert: `AMAZON_PROVIDED`

Erforderlich: Nein

Typ: Zeichenfolge

`dns_support`

Gibt an, ob die in VPC gestarteten Instances DNS-Hostnamen erhalten.

Erforderlich: Nein

Typ: Boolesch

Standard: `false`

`tags`

Tags, die an die Ressource angehängt werden sollen.

Erforderlich: Nein

Typ: Liste

Beispiel

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

AWS. Netzwerke. NATGateway

Sie können einen öffentlichen oder privaten NAT-Gateway-Knoten über ein Subnetz definieren. Wenn Sie bei einem öffentlichen Gateway keine Elastic IP-Zuweisungs-ID angeben, weist AWS TNB Ihrem Konto eine Elastic IP zu und ordnet diese dem Gateway zu.

Syntax

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

Eigenschaften

`subnet`

Die Node-Referenz AWS.Networking.Subnet.

Erforderlich: Ja

Typ: Zeichenfolge

`internet_gateway`

Das AWS.Networking. InternetGatewayKnotenreferenz.

Erforderlich: Ja

Typ: Zeichenfolge

Eigenschaften

`type`

Gibt an, ob das Gateway öffentlich oder privat ist.

Zulässiger Wert:`PUBLIC`, `PRIVATE`

Erforderlich: Ja

Typ: Zeichenfolge

`eip_allocation_id`

Die ID, die die Zuweisung der Elastic IP-Adresse darstellt.

Erforderlich: Nein

Typ: Zeichenfolge

`tags`

Tags, die an die Ressource angehängt werden sollen.

Erforderlich: Nein

Typ: Liste

Beispiel

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

AWS.Netzwerking.Route

Sie können einen Routenknoten definieren, der die Zielroute dem NAT-Gateway als Zielressource zuordnet und die Route der zugehörigen Routentabelle hinzufügt.

Syntax

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

```
    route_table: String
```

Eigenschaften

`dest_cidr_blocks`

Die Liste der IPv4 Zielrouten zur Zielressource.

Erforderlich: Ja

Typ: Liste

Elementtyp: Zeichenfolge

Voraussetzungen

`nat_gateway`

Das AWS.Networking. NATGatewayKnotenreferenz.

Erforderlich: Ja

Typ: Zeichenfolge

`route_table`

Das AWS.Networking. RouteTableKnotenreferenz.

Erforderlich: Ja

Typ: Zeichenfolge

Beispiel

```
Free5GCRoute:
  type: tosca.nodes.AWS.Networking.Route
  properties:
    dest_cidr_blocks:
      - 0.0.0.0/0
      - 10.0.0.0/28
  requirements:
    nat_gateway: Free5GCNatGateway01
```

```
    route_table: Free5GCRouteTable
```

AWS. Speichern. SSMParameters

Sie können SSM-Parameter über TNB erstellen. AWS Die SSM-Parameter, die Sie erstellen, werden in SSM erstellt und haben als Präfix die AWS TNB-Netzwerkinstanz-ID. Dadurch wird verhindert, dass Parameterwerte überschrieben werden, wenn mehrere Instanzen mithilfe derselben NSD-Vorlage instanziiert und aktualisiert werden.

Syntax

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

Eigenschaften

Parameter

`name`

Der Name der SSM-Eigenschaft. Verwenden Sie das folgende Format: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`

Der Name jedes Parameters muss weniger als 256 Zeichen lang sein.

Erforderlich: Ja

Typ: Zeichenfolge

`value`

Der Wert der Eigenschaft ssm. Verwenden Sie eines der folgenden Formate:

- Für Werte ohne Referenzen: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`
- Für statische Referenzen: `^\$\{[a-zA-Z0-9]+\.(properties|capabilities|requirements)(\.([a-zA-Z0-9\-_]+))+\}$`
- Für dynamische Referenzen: `^\$\{[a-zA-Z0-9]+\.(name|id|arn)\}$`

Der Wert jedes Parameters muss kleiner als 4 KB sein.

Erforderlich: Ja

Typ: Zeichenfolge

`tags`

Die Tags, die Sie an eine SSM-Eigenschaft anhängen können.

Erforderlich: Nein

Typ: Liste

Beispiel

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

Gemeinsame Knoten

Definieren Sie Knoten für NSD und VNFD.

- AWS. HookDefinition. Bash

AWS.HookDefinition.Bash

Definiert einen AWS HookDefinition in`bash`.

Syntax

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

Eigenschaften

`implementation`

Der relative Pfad zur Hook-Definition. Das Format muss wie folgt sein: `./hooks/`*`script_name`*`.sh`

Erforderlich: Ja

Typ: Zeichenfolge

`environment_variables`

Die Umgebungsvariablen für das Hook-Bash-Skript. Verwenden Sie das folgende Format: **`envName`**`=`**`envValue`** mit den folgenden Regex-Mustern:

- Für Werte ohne Referenzen: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`
- Für statische Referenzen: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=\$\{[a-zA-Z0-9]+\.(properties|capabilities|requirements)(\.([a-zA-Z0-9\-\_]+))+\}$`
- Für dynamische Referenzen: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=\$\{[a-zA-Z0-9]+\.(name|id|arn)\}$`

Stellen Sie sicher, dass der **`envName`**`=`**`envValue`** Wert die folgenden Kriterien erfüllt:

- Verwenden Sie keine Leerzeichen.
- Beginne **`envName`** mit einem Buchstaben (A-Z oder a-z) oder einer Zahl (0-9).
- Beginnen Sie den Namen der Umgebungsvariablen nicht mit den folgenden reservierten AWS TNB-Schlüsselwörtern (Groß- und Kleinschreibung wird nicht beachtet):
  - CODEBUILD
  - TNB

- ZUHAUSE
- AWS

- Sie können eine beliebige Anzahl von Buchstaben (A-Z oder a-z), Zahlen (0-9) und Sonderzeichen - sowie für und verwenden. _ **envName envValue**
- Jede Umgebungsvariable (jedes **envName** =**envValue**) muss weniger als 128 Zeichen lang sein.

Beispiel: `A123-45xYz=Example_789`

Erforderlich: Nein

Typ: Liste

`execution_role`

Die Rolle für die Hook-Ausführung.

Erforderlich: Ja

Typ: Zeichenfolge

Beispiel

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

Sicherheit in TNB AWS

Cloud-Sicherheit AWS hat höchste Priorität. Als AWS Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die darauf ausgelegt sind, die Anforderungen der sicherheitssensibelsten Unternehmen zu erfüllen.

Sicherheit ist eine gemeinsame AWS Verantwortung von Ihnen und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, auf der AWS Dienste in der ausgeführt AWS Cloud werden. AWS bietet Ihnen auch Dienste, die Sie sicher nutzen können. Externe Prüfer testen und verifizieren regelmäßig die Wirksamkeit unserer Sicherheitsmaßnahmen im Rahmen der AWS . Weitere Informationen zu den Compliance-Programmen, die für AWS Telco Network Builder gelten, finden Sie unter AWS Services im Bereich nach Compliance-Programm AWS .
- Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem AWS Dienst, den Sie nutzen. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der gemeinsamen Verantwortung bei der Nutzung von AWS TNB anwenden können. In den folgenden Themen erfahren Sie, wie Sie AWS TNB konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Sie erfahren auch, wie Sie andere AWS Dienste nutzen können, die Sie bei der Überwachung und Sicherung Ihrer AWS TNB-Ressourcen unterstützen.

Inhalt

Datenschutz in TNB AWS

Das Modell der AWS gemeinsamen Verantwortung und gilt für den Datenschutz in AWS Telco Network Builder. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der AWS Cloud alle Systeme laufen. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit AWS TNB oder anderen AWS-Services über die Konsole, API oder arbeiten. AWS CLI AWS SDKs Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen

Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Umgang mit Daten

Wenn Sie Ihr AWS Konto schließen, markiert AWS TNB Ihre Daten für die Löschung und entfernt sie für jegliche Verwendung. Wenn Sie Ihr AWS Konto innerhalb von 90 Tagen reaktivieren, stellt AWS TNB Ihre Daten wieder her. Nach 120 Tagen löscht AWS TNB Ihre Daten dauerhaft. AWS TNB terminiert auch Ihre Netzwerke und löscht Ihre Funktionspakete und Ihre Netzwerkpakete.

Verschlüsselung im Ruhezustand

AWS TNB verschlüsselt immer alle im Dienst gespeicherten Daten im Ruhezustand, ohne dass eine zusätzliche Konfiguration erforderlich ist. Diese Verschlüsselung erfolgt automatisch. AWS Key Management Service

Verschlüsselung während der Übertragung

AWS TNB schützt alle Daten während der Übertragung mit Transport Layer Security (TLS) 1.2.

Es liegt in Ihrer Verantwortung, Daten zwischen Ihren Simulationsagenten und deren Clients zu verschlüsseln.

Datenschutz für den Datenverkehr zwischen Netzwerken

AWS TNB-Rechenressourcen befinden sich in einer Virtual Private Cloud (VPC), die von allen Kunden gemeinsam genutzt wird. Der gesamte interne AWS TNB-Verkehr blieb im AWS Netzwerk und durchquert nicht das Internet. Verbindungen zwischen Ihren Simulationsagenten und ihren Clients werden über das Internet geleitet.

Identitäts- und Zugriffsmanagement für TNB AWS

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um TNB-Ressourcen zu verwenden AWS . IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Inhalt

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von Ihrer Rolle ab:

- Servicebenutzer – Fordern Sie von Ihrem Administrator Berechtigungen an, wenn Sie nicht auf Features zugreifen können (siehe Fehlerbehebung bei Identität und Zugriff bei AWS Telco Network Builder).
- Serviceadministrator – Bestimmen Sie den Benutzerzugriff und stellen Sie Berechtigungsanfragen (siehe Wie funktioniert AWS TNB mit IAM).
- IAM-Administrator – Schreiben Sie Richtlinien zur Zugriffsverwaltung (siehe Beispiele für identitätsbasierte Richtlinien für AWS Telco Network Builder).

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

Verbundidentität

Es hat sich bewährt, dass menschliche Benutzer für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensverzeichnis, Ihrem Directory Service Web-Identitätsanbieter oder der AWS-Services mithilfe von Anmeldeinformationen aus einer Identitätsquelle zugreift. Verbundene Identitäten übernehmen Rollen, die temporäre Anmeldeinformationen bereitstellen.

Für die zentrale Zugriffsverwaltung empfehlen wir AWS IAM Identity Center. Weitere Informationen finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer den Verbund mit einem Identitätsanbieter verwenden müssen, um AWS mithilfe temporärer Anmeldeinformationen darauf zugreifen zu können.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln oder indem Sie eine AWS Oder-API-Operation AWS CLI

aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2 laufen. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität (Benutzer, Gruppe oder Rolle) anfügen können. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mit mehreren Identitäten verbunden sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- Berechtigungsgrenzen – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im -IAM-Benutzerhandbuch.
- Richtlinien zur Dienstkontrolle (SCPs) — Geben Sie die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in an AWS Organizations. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Richtlinien zur Ressourcenkontrolle (RCPs) — Legen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten fest. Weitere Informationen finden Sie im AWS Organizations Benutzerhandbuch unter Richtlinien zur Ressourcenkontrolle (RCPs).
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn für eine Anfrage mehrere Arten von Richtlinien gelten, sind die sich daraus ergebenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter Bewertungslogik für Richtlinien im IAM-Benutzerhandbuch.

Wie funktioniert AWS TNB mit IAM

Bevor Sie IAM zur Verwaltung des Zugriffs auf AWS TNB verwenden, sollten Sie sich darüber informieren, welche IAM-Funktionen mit TNB verwendet werden können. AWS

IAM-Funktionen, die Sie mit Telco Network Builder verwenden können AWS

IAM-Feature	AWS TNB-Unterstützung
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Bedingungsschlüssel für die Richtlinie	Ja
ACLs	Nein
ABAC (Tags in Richtlinien)	Ja
Temporäre Anmeldeinformationen	Ja
Prinzipalberechtigungen	Ja
Servicerollen	Nein
Serviceverknüpfte Rollen	Nein

Einen allgemeinen Überblick darüber, wie AWS TNB und andere AWS Dienste mit den meisten IAM-Funktionen funktionieren, finden Sie im AWS IAM-Benutzerhandbuch unter Dienste, die mit IAM funktionieren.

Identitätsbasierte Richtlinien für TNB AWS

Unterstützt Richtlinien auf Identitätsbasis: Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern,

welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für TNB AWS

Beispiele für identitätsbasierte AWS TNB-Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS Telco Network Builder

Ressourcenbasierte Richtlinien innerhalb von TNB AWS

Unterstützt ressourcenbasierte Richtlinien: Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Politische Maßnahmen für TNB AWS

Unterstützt Richtlinienaktionen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Nehmen Sie Aktionen in eine Richtlinie auf, um Berechtigungen zur Ausführung des zugehörigen Vorgangs zu erteilen.

Eine Liste der AWS TNB-Aktionen finden Sie unter Von AWS Telco Network Builder definierte Aktionen in der Service Authorization Reference.

Bei Richtlinienaktionen in AWS TNB wird vor der Aktion das folgende Präfix verwendet:

```
tnb
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

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

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `List` beginnen, einschließlich der folgenden Aktion:

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

Beispiele für identitätsbasierte AWS TNB-Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS Telco Network Builder

Politische Ressourcen für TNB AWS

Unterstützt Richtlinienressourcen: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer Zugriff auf was hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Als Best Practice geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

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

Eine Liste der AWS TNB-Ressourcentypen und ihrer ARNs Eigenschaften finden Sie unter Von AWS Telco Network Builder definierte Ressourcen in der Service Authorization Reference. Informationen darüber, mit welchen Aktionen Sie den ARN jeder Ressource angeben können, finden Sie unter Von AWS Telco Network Builder definierte Aktionen.

Beispiele für identitätsbasierte AWS TNB-Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS Telco Network Builder

Bedingungsschlüssel für Richtlinien für TNB AWS

Unterstützt servicespezifische Richtlinienbedingungsschlüssel: Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen kann.

Das Element `Condition` gibt an, wann Anweisungen auf der Grundlage definierter Kriterien ausgeführt werden. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontextschlüssel für AWS globale Bedingungen im IAM-Benutzerhandbuch.

Eine Liste der AWS TNB-Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für AWS Telco Network Builder in der Service Authorization Reference. Informationen zu den Aktionen und Ressourcen, mit denen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Von AWS Telco Network Builder definierte Aktionen.

Beispiele für identitätsbasierte AWS TNB-Richtlinien finden Sie unter. Beispiele für identitätsbasierte Richtlinien für AWS Telco Network Builder

ACLs AWS in TNB

Unterstützt ACLs: Nein

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit TNB AWS

Unterstützt ABAC (Tags in Richtlinien): Ja

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen, auch als Tags bezeichnet, definiert werden. Sie können Tags an IAM-Entitäten und AWS -Ressourcen anhängen und dann ABAC-Richtlinien entwerfen, um Operationen zu ermöglichen, wenn das Tag des Prinzipals mit dem Tag auf der Ressource übereinstimmt.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Definieren von Berechtigungen mit ABAC-Autorisierung im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Verwenden temporärer Anmeldeinformationen mit TNB AWS

Unterstützt temporäre Anmeldeinformationen: Ja

Temporäre Anmeldeinformationen ermöglichen kurzfristigen Zugriff auf AWS Ressourcen und werden automatisch erstellt, wenn Sie einen Verbund verwenden oder die Rollen wechseln. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Anmeldeinformationen in IAM und AWS-Services , die mit IAM funktionieren im IAM-Benutzerhandbuch.

Serviceübergreifende Prinzipalberechtigungen für TNB AWS

Unterstützt Forward Access Sessions (FAS): Ja

Forward Access Sessions (FAS) verwenden die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, in Kombination mit der Anforderung, Anfragen an AWS-Service nachgelagerte Dienste zu stellen. Einzelheiten zu den Richtlinien für FAS-Anforderungen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für TNB AWS

Unterstützt Servicerollen: Nein

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern

und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

Mit Diensten verknüpfte Rollen für TNB AWS

Unterstützt serviceverknüpfte Rollen: Ja

Eine dienstgebundene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Beispiele für identitätsbasierte Richtlinien für AWS Telco Network Builder

Standardmäßig sind Benutzer und Rollen nicht berechtigt, AWS TNB-Ressourcen zu erstellen oder zu ändern. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien (Konsole) im IAM-Benutzerhandbuch.

Einzelheiten zu den von AWS TNB definierten Aktionen und Ressourcentypen, einschließlich des Formats ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für AWS Telco Network Builder in der Service Authorization Reference.

Inhalt

Best Practices für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand AWS TNB-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Wenn Sie identitätsbasierte Richtlinien erstellen oder bearbeiten, befolgen Sie diese Richtlinien und Empfehlungen:

- Erste Schritte mit AWS verwalteten Richtlinien und Umstellung auf Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um damit zu beginnen, Ihren Benutzern und Workloads Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter Von AWS verwaltete Richtlinien oder Von AWS verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten – Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs – Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese für einen bestimmten Zweck verwendet werden AWS-Service, z. CloudFormation B. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten – IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung mit IAM Access Analyzer im IAM-Benutzerhandbuch.
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Sicherer API-Zugriff mit MFA im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Best Practices für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Verwenden der TNB-Konsole AWS

Um auf die AWS Telco Network Builder-Konsole zugreifen zu können, benötigen Sie ein Mindestmaß an Berechtigungen. Diese Berechtigungen müssen es Ihnen ermöglichen, Details zu den AWS TNB-Ressourcen in Ihrem aufzulisten und anzuzeigen. AWS-Konto Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Beispiele für Richtlinien für Servicerollen

Als Administrator besitzen und verwalten Sie die Ressourcen, die AWS TNB gemäß den Umgebungs- und Dienstvorlagen erstellt. Sie müssen Ihrem Konto IAM-Servicerollen zuordnen, damit AWS TNB Ressourcen für Ihr Netzwerk-Lebenszyklusmanagement erstellen kann.

Eine IAM-Servicerolle ermöglicht es AWS TNB, in Ihrem Namen Ressourcen aufzurufen, um Ihre Netzwerke zu instanziieren und zu verwalten. Wenn Sie eine Servicerolle angeben, verwendet AWS TNB die Anmeldeinformationen dieser Rolle.

Sie erstellen die Service-Rolle und die Berechtigungsrichtlinie mit dem IAM-Service. Weitere Informationen zum Erstellen einer Servicerolle finden Sie unter Creating a role to delegate permissions to an AWS service im IAM-Benutzerhandbuch.

AWS TNB-Servicerolle

Als Mitglied des Plattformteams können Sie als Administrator eine AWS TNB-Servicerolle erstellen und sie TNB zur Verfügung stellen. AWS Diese Rolle ermöglicht es AWS TNB, andere Dienste wie Amazon Elastic Kubernetes Service aufzurufen und CloudFormation die erforderliche Infrastruktur für Ihr Netzwerk bereitzustellen und Netzwerkfunktionen bereitzustellen, wie in Ihrer NSD definiert.

Wir empfehlen Ihnen, die folgende IAM-Rollen- und Vertrauensrichtlinie für Ihre TNB-Servicerolle zu verwenden. AWS Denken Sie bei der Einschränkung der Zugriffsrechte für diese Richtlinie daran, dass AWS TNB bei Ressourcen, die nicht in Ihrer Richtlinie enthalten sind, möglicherweise mit dem Fehler „Zugriff verweigert“ fehlschlägt.

Der folgende Code zeigt eine AWS TNB-Servicerollenrichtlinie:

JSON

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

```
                ]
            }
        },
        "Action": [
            "iam:CreateServiceLinkedRole"
        ],
        "Resource": "*",
        "Effect": "Allow",
        "Sid": "TNBAccessSLRPermissions"
    },
    {
        "Action": [
            "autoscaling:CreateAutoScalingGroup",
            "autoscaling:CreateOrUpdateTags",
            "autoscaling:DeleteAutoScalingGroup",
            "autoscaling:DeleteTags",
            "autoscaling:DescribeAutoScalingGroups",
            "autoscaling:DescribeAutoScalingInstances",
            "autoscaling:DescribeScalingActivities",
            "autoscaling:DescribeTags",
            "autoscaling:UpdateAutoScalingGroup",
            "ec2:AuthorizeSecurityGroupEgress",
            "ec2:AuthorizeSecurityGroupIngress",
            "ec2:CreateLaunchTemplate",
            "ec2:CreateLaunchTemplateVersion",
            "ec2:CreateSecurityGroup",
            "ec2:DeleteLaunchTemplateVersions",
            "ec2:DescribeLaunchTemplates",
            "ec2:DescribeLaunchTemplateVersions",
            "ec2:DeleteLaunchTemplate",
            "ec2:DeleteSecurityGroup",
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeTags",
            "ec2:GetLaunchTemplateData",
            "ec2:RevokeSecurityGroupEgress",
            "ec2:RevokeSecurityGroupIngress",
            "ec2:RunInstances",
            "ec2:AssociateRouteTable",
            "ec2:AttachInternetGateway",
            "ec2:CreateInternetGateway",
            "ec2:CreateNetworkInterface",
            "ec2:CreateRoute",
            "ec2:CreateRouteTable",
            "ec2:CreateSubnet",
```

```
                "ec2:CreateTags",
                "ec2:CreateVpc",
                "ec2:DeleteInternetGateway",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteRoute",
                "ec2:DeleteRouteTable",
                "ec2:DeleteSubnet",
                "ec2:DeleteTags",
                "ec2:DeleteVpc",
                "ec2:DetachNetworkInterface",
                "ec2:DescribeInstances",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroupRules",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DetachInternetGateway",
                "ec2:DisassociateRouteTable",
                "ec2:ModifySecurityGroupRules",
                "ec2:ModifySubnetAttribute",
                "ec2:ModifyVpcAttribute",
                "ec2:AllocateAddress",
                "ec2:AssignIpv6Addresses",
                "ec2:AssociateAddress",
                "ec2:AssociateNatGatewayAddress",
                "ec2:AssociateVpcCidrBlock",
                "ec2:CreateEgressOnlyInternetGateway",
                "ec2:CreateNatGateway",
                "ec2:DeleteEgressOnlyInternetGateway",
                "ec2:DeleteNatGateway",
                "ec2:DescribeAddresses",
                "ec2:DescribeEgressOnlyInternetGateways",
                "ec2:DescribeNatGateways",
                "ec2:DisassociateAddress",
                "ec2:DisassociateNatGatewayAddress",
                "ec2:DisassociateVpcCidrBlock",
                "ec2:ReleaseAddress",
                "ec2:UnassignIpv6Addresses",
                "ec2:DescribeImages",
                "eks:CreateCluster",
                "eks:ListClusters",
                "eks:RegisterCluster",
```

```
                "eks:TagResource",
                "eks:DescribeAddonVersions",
                "events:DescribeRule",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TNBAccessComputePerms"
        },
        {
            "Resource": "*",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Condition": {
                "StringEquals": {
                   "iam:PassedToService": [
                       "ec2.amazonaws.com",
                       "eks.amazonaws.com",
                       "eks-nodegroup.amazonaws.com",
                       "events.amazonaws.com",
                       "autoscaling.amazonaws.com",
                       "codebuild.amazonaws.com"
                   ]
                }
            }
        },
        {
            "Action": [
                "codebuild:BatchDeleteBuilds",
                "codebuild:BatchGetBuilds",
                "codebuild:CreateProject",
                "codebuild:DeleteProject",
                "codebuild:ListBuildsForProject",
                "codebuild:StartBuild",
                "codebuild:StopBuild",
                "events:DeleteRule",
                "events:PutRule",
                "events:PutTargets",
                "events:RemoveTargets",
                "s3:CreateBucket",
                "s3:GetBucketAcl",
```

```
                "s3:GetObject",
                "eks:DescribeNodegroup",
                "eks:DeleteNodegroup",
                "eks:AssociateIdentityProviderConfig",
                "eks:CreateNodegroup",
                "eks:DeleteCluster",
                "eks:DeregisterCluster",
                "eks:UpdateAddon",
                "eks:UpdateClusterVersion",
                "eks:UpdateNodegroupConfig",
                "eks:UpdateNodegroupVersion",
                "eks:DescribeUpdate",
                "eks:UntagResource",
                "eks:DescribeCluster",
                "eks:ListNodegroups",
                "eks:CreateAddon",
                "eks:DeleteAddon",
                "eks:DescribeAddon",
                "eks:DescribeAddonVersions",
                "s3:PutObject",
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:DescribeStackResources",
                "cloudformation:DescribeStacks",
                "cloudformation:ListStackResources",
                "cloudformation:UpdateStack",
                "cloudformation:UpdateTerminationProtection",
                "ssm:PutParameter",
                "ssm:GetParameters",
                "ssm:GetParameter",
                "ssm:DeleteParameter",
                "ssm:AddTagsToResource",
                "ssm:ListTagsForResource",
                "ssm:RemoveTagsFromResource"
            ],
            "Resource": [
                "arn:aws:events:*:*:rule/tnb*",
                "arn:aws:codebuild:*:*:project/tnb*",
                "arn:aws:logs:*:*:log-group:/aws/tnb*",
                "arn:aws:s3:::tnb*",
                "arn:aws:eks:*:*:addon/tnb*/*/*",
                "arn:aws:eks:*:*:cluster/tnb*",
                "arn:aws:eks:*:*:nodegroup/tnb*/tnb*/*",
                "arn:aws:cloudformation:*:*:stack/tnb*",
```

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

Der folgende Code zeigt die Vertrauensrichtlinie für den AWS TNB-Dienst:

JSON

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

```
}
```

AWS TNB-Servicerolle für Amazon EKS-Cluster

Wenn Sie eine Amazon EKS-Ressource in Ihrem NSD erstellen, geben Sie das `cluster_role` Attribut an, um anzugeben, welche Rolle zur Erstellung Ihres Amazon EKS-Clusters verwendet wird.

Das folgende Beispiel zeigt eine AWS CloudFormation Vorlage, die eine AWS TNB-Servicerolle für die Amazon EKS-Cluster-Richtlinie erstellt.

```
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"
```

Weitere Informationen zu IAM-Rollen mithilfe von AWS CloudFormation Vorlagen finden Sie in den folgenden Abschnitten des AWS CloudFormation Benutzerhandbuchs:

- AWS::IAM::Role
- Auswählen einer Stack-Vorlage

AWS TNB-Servicerolle für die Amazon EKS-Knotengruppe

Wenn Sie eine Amazon EKS-Knotengruppenressource in Ihrer NSD erstellen, geben Sie das `node_role` Attribut an, um anzugeben, welche Rolle zur Erstellung Ihrer Amazon EKS-Knotengruppe verwendet wird.

Das folgende Beispiel zeigt eine CloudFormation Vorlage, die eine AWS TNB-Servicerolle für die Amazon EKS-Knotengruppenrichtlinie erstellt.

```
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/*"
```

Weitere Informationen zu IAM-Rollen mithilfe von AWS CloudFormation Vorlagen finden Sie in den folgenden Abschnitten des AWS CloudFormation Benutzerhandbuchs:

- AWS::IAM::Role
- Auswählen einer Stack-Vorlage

AWS TNB-Servicerolle für Multus

Wenn Sie eine Amazon EKS-Ressource in Ihrem NSD erstellen und Multus als Teil Ihrer Bereitstellungsvorlage verwalten möchten, müssen Sie das `multus_role` Attribut angeben, um anzugeben, welche Rolle für die Verwaltung von Multus verwendet werden soll.

Das folgende Beispiel zeigt eine CloudFormation Vorlage, die eine AWS TNB-Servicerolle für eine Multus-Richtlinie erstellt.

```
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: "*"
```

Weitere Informationen zu IAM-Rollen mithilfe von AWS CloudFormation Vorlagen finden Sie in den folgenden Abschnitten des AWS CloudFormation Benutzerhandbuchs:

- AWS::IAM::Role
- Auswählen einer Stack-Vorlage

AWS TNB-Servicerolle für eine Life-Cycle-Hook-Richtlinie

Wenn Ihr NSD- oder Netzwerkfunktionspaket einen Life-Cycle-Hook verwendet, benötigen Sie eine Service-Rolle, damit Sie eine Umgebung für die Ausführung Ihrer Life-Cycle-Hooks einrichten können.

> **Note**
>
> Ihre Lifecycle-Hook-Richtlinie sollte auf dem basieren, was Ihr Life-Cycle-Hook zu tun versucht.

Das folgende Beispiel zeigt eine CloudFormation Vorlage, die eine AWS TNB-Servicerolle für eine Life-Cycle-Hook-Richtlinie erstellt.

```
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"
```

Weitere Informationen zu IAM-Rollen mithilfe von AWS CloudFormation Vorlagen finden Sie in den folgenden Abschnitten des AWS CloudFormation Benutzerhandbuchs:

- AWS::IAM::Role
- Auswählen einer Stack-Vorlage

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der AWS CLI OR-API. AWS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
```

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

Fehlerbehebung bei Identität und Zugriff bei AWS Telco Network Builder

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit AWS TNB und IAM auftreten können.

Problembereiche

- Ich bin nicht berechtigt, eine Aktion in TNB durchzuführen AWS
- Ich bin nicht berechtigt, iam durchzuführen: PassRole
- Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine AWS TNB-Ressourcen ermöglichen

Ich bin nicht berechtigt, eine Aktion in TNB durchzuführen AWS

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson` IAM-Benutzer versucht, die Konsole zum Anzeigen von Details zu einer fiktiven *`my-example-widget`*-Ressource zu verwenden, jedoch nicht über `tnb:`*`GetWidget`*-Berechtigungen verfügt.

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

In diesem Fall muss die Mateo-Richtlinie aktualisiert werden, damit er mit der `tnb:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam durchzuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zur Durchführung der `iam:PassRole` Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie eine Rolle an AWS TNB übergeben können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in AWS TNB auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

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

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine AWS TNB-Ressourcen ermöglichen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob AWS TNB diese Funktionen unterstützt, finden Sie unter. Wie funktioniert AWS TNB mit IAM
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Konformitätsvalidierung für TNB AWS

Informationen darüber, ob AWS-Service ein in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter AWS-Services Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der AWS Sicherheitsdokumentation.

Resilienz in AWS TNB

Die AWS globale Infrastruktur basiert auf Availability AWS-Regionen Zones. AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu Availability Zones AWS-Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

AWS TNB führt Ihren Netzwerkdienst auf EKS-Clustern in einer Virtual Private Cloud (VPC) in der von Ihnen AWS ausgewählten Region aus.

Sicherheit der Infrastruktur in TNB AWS

Als verwalteter Dienst ist AWS Telco Network Builder durch AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf AWS TNB zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Hier sind einige Beispiele für gemeinsame Verantwortlichkeiten:

- AWS ist verantwortlich für die Sicherung von Komponenten, die AWS TNB unterstützen, darunter:
  - Recheninstanzen (auch bekannt als Worker)
  - Interne Datenbanken
  - Netzwerkkommunikation zwischen internen Komponenten
  - Die AWS TNB-Anwendungsprogrammierschnittstelle (API)
  - AWS Softwareentwicklungskits (SDK)
- Sie sind dafür verantwortlich, Ihren Zugriff auf Ihre AWS Ressourcen und Ihre Workload-Komponenten zu sichern, einschließlich (aber nicht beschränkt auf):
  - IAM-Benutzer, -Gruppen, -Rollen und -Richtlinien
  - S3-Buckets, die Sie zum Speichern Ihrer Daten für TNB verwenden AWS
  - Andere Ressourcen AWS-Services und Ressourcen, die Sie zur Unterstützung des Netzwerkdienstes verwenden, den Sie über TNB bereitgestellt haben AWS
  - Ihr Anwendungscode
  - Verbindungen zwischen dem Netzwerkdienst, den Sie über AWS TNB bereitgestellt haben, und seinen Clients

> **Important**
>
> Sie sind für die Implementierung eines Notfallwiederherstellungsplans verantwortlich, mit dem ein Netzwerkdienst, den Sie über TNB bereitgestellt haben, effektiv wiederhergestellt werden kann. AWS

Sicherheitsmodell für Netzwerkkonnektivität

Die Netzwerkdienste, die Sie über AWS TNB bereitstellen, werden auf Recheninstanzen innerhalb einer Virtual Private Cloud (VPC) ausgeführt, die sich in einer von Ihnen AWS ausgewählten Region befindet. Eine VPC ist ein virtuelles Netzwerk in der AWS Cloud, das die Infrastruktur nach Arbeitslast oder organisatorischer Einheit isoliert. Die Kommunikation zwischen Recheninstanzen innerhalb des Netzwerks VPCs bleibt innerhalb des AWS Netzwerks und wird nicht über das Internet übertragen. Ein Teil der internen Dienstkommunikation erfolgt über das Internet und ist verschlüsselt. Netzwerkdienste, die über AWS TNB für alle Kunden bereitgestellt werden, die in derselben Region laufen, teilen sich dieselbe VPC. Netzwerkdienste, die über AWS TNB für verschiedene Kunden bereitgestellt werden, verwenden separate Recheninstanzen innerhalb derselben VPC.

Die Kommunikation zwischen Ihren Netzwerkdienstclients und Ihrem Netzwerkdienst in AWS TNB erfolgt über das Internet. AWS TNB verwaltet diese Verbindungen nicht. Es liegt in Ihrer Verantwortung, Ihre Kundenverbindungen zu sichern.

Ihre Verbindungen zu AWS TNB über AWS-Managementkonsole, AWS Command Line Interface (AWS CLI) und AWS SDKs sind verschlüsselt.

IMDS-Version

AWS TNB unterstützt Instances, die Instance Metadata Service Version 2 (IMDSv2), eine sitzungsorientierte Methode, nutzen. IMDSv2 beinhaltet eine höhere Sicherheit als. IMDSV1 Weitere Informationen finden Sie unter Umfassender Schutz vor offenen Firewalls, Reverse-Proxys und SSRF-Schwachstellen mit Verbesserungen am Amazon EC2 Instance Metadata Service.

Wenn Sie Ihre Instance starten, müssen Sie Folgendes verwenden. IMDSv2 Weitere Informationen zu finden Sie unter Verwendung IMDSv2 im Amazon EC2 EC2-Benutzerhandbuch. IMDSv2

AWS TNB überwachen

Die Überwachung ist ein wichtiger Bestandteil der Aufrechterhaltung der Zuverlässigkeit, Verfügbarkeit und Leistung von AWS TNB und Ihren anderen AWS Lösungen. AWS bietet AWS CloudTrail die Möglichkeit, AWS TNB zu beobachten, zu melden, wenn etwas nicht stimmt, und gegebenenfalls automatische Maßnahmen zu ergreifen.

Wird verwendet CloudTrail , um detaillierte Informationen über die getätigten Anrufe zu AWS APIs erfassen. Sie können diese Aufrufe als Protokolldateien in Amazon S3 speichern. Anhand dieser CloudTrail Protokolle können Sie beispielsweise ermitteln, welcher Anruf getätigt wurde, von welcher Quell-IP-Adresse der Anruf kam, wer den Anruf getätigt hat und wann der Anruf getätigt wurde.

Die CloudTrail Protokolle enthalten Informationen über die Aufrufe von API-Aktionen für AWS TNB. Sie enthalten auch Informationen für Aufrufe von API-Aktionen von Diensten wie Amazon EC2 und Amazon EBS.

Protokollieren von AWS Telco Network Builder API-Aufrufen mit AWS CloudTrail

AWS Telco Network Builder ist in einen Dienst integriert AWS CloudTrail, der eine Aufzeichnung der von einem Benutzer, einer Rolle oder einem ausgeführten Aktionen bereitstellt. AWS-Service CloudTrail erfasst alle API-Aufrufe für AWS TNB als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der AWS TNB-Konsole und Codeaufrufen für die AWS TNB-API-Operationen. Anhand der von CloudTrail gesammelten Informationen können Sie die Anfrage an AWS TNB, die IP-Adresse, von der aus die Anfrage gestellt wurde, den Zeitpunkt der Anfrage und weitere Details ermitteln.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anforderung mit Anmeldeinformationen des Root-Benutzers oder des Benutzers gestellt wurde.
- Die Anforderung wurde im Namen eines IAM-Identity-Center-Benutzers erstellt.
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde.

CloudTrail ist in Ihrem aktiv AWS-Konto , wenn Sie das Konto erstellen, und Sie haben automatisch Zugriff auf den CloudTrail Eventverlauf. Der CloudTrail Ereignisverlauf bietet eine einsehbare, durchsuchbare, herunterladbare und unveränderliche Aufzeichnung der aufgezeichneten Verwaltungsereignisse der letzten 90 Tage in einem. AWS-Region Weitere Informationen finden Sie im AWS CloudTrail Benutzerhandbuch unter Arbeiten mit dem CloudTrail Ereignisverlauf. Für die Anzeige des Eventverlaufs CloudTrail fallen keine Gebühren an.

Für eine fortlaufende Aufzeichnung der Ereignisse in AWS-Konto den letzten 90 Tagen erstellen Sie einen Trail- oder CloudTrailLake-Event-Datenspeicher.

CloudTrail Pfade

Ein Trail ermöglicht CloudTrail die Übermittlung von Protokolldateien an einen Amazon S3 S3-Bucket. Alle mit dem erstellten Pfade AWS-Managementkonsole sind regionsübergreifend. Sie können mithilfe von AWS CLI einen Einzel-Region- oder einen Multi-Region-Trail erstellen. Es wird empfohlen, einen Trail mit mehreren Regionen zu erstellen, da Sie alle Aktivitäten AWS-Regionen in Ihrem Konto erfassen. Wenn Sie einen Einzel-Region-Trail erstellen, können Sie nur die Ereignisse anzeigen, die im AWS-Region des Trails protokolliert wurden. Weitere Informationen zu Trails finden Sie unter Erstellen eines Trails für Ihr AWS-Konto und Erstellen eines Trails für eine Organisation im AWS CloudTrail -Benutzerhandbuch.

Sie können eine Kopie Ihrer laufenden Verwaltungsereignisse kostenlos an Ihren Amazon S3 S3-Bucket senden, CloudTrail indem Sie einen Trail erstellen. Es fallen jedoch Amazon S3 S3-Speichergebühren an. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preise. Informationen zu Amazon-S3-Preisen finden Sie unter Amazon S3 – Preise.

CloudTrail Datenspeicher für Ereignisse in Lake

CloudTrail Mit Lake können Sie SQL-basierte Abfragen für Ihre Ereignisse ausführen. CloudTrail Lake konvertiert bestehende Ereignisse im zeilenbasierten JSON-Format in das Apache ORC-Format. ORC ist ein spaltenförmiges Speicherformat, das für den schnellen Abruf von Daten optimiert ist. Die Ereignisse werden in Ereignisdatenspeichern zusammengefasst, bei denen es sich um unveränderliche Sammlungen von Ereignissen handelt, die auf Kriterien basieren, die Sie mit Hilfe von erweiterten Ereignisselektoren auswählen. Die Selektoren, die Sie auf einen Ereignisdatenspeicher anwenden, steuern, welche Ereignisse bestehen bleiben und für Sie zur Abfrage verfügbar sind. Weitere Informationen zu CloudTrail Lake finden Sie unter Arbeiten mit AWS CloudTrail Lake im AWS CloudTrail Benutzerhandbuch.

CloudTrail Für das Speichern und Abfragen von Ereignisdaten in Lake fallen Kosten an. Beim Erstellen eines Ereignisdatenspeichers wählen Sie die Preisoption aus, die für den

Ereignisdatenspeicher genutzt werden soll. Die Preisoption bestimmt die Kosten für die Erfassung und Speicherung von Ereignissen sowie die standardmäßige und maximale Aufbewahrungsdauer für den Ereignisdatenspeicher. Weitere Informationen zur CloudTrail Preisgestaltung finden Sie unter AWS CloudTrail Preise.

AWS Beispiele für TNB-Veranstaltungen

Ein Ereignis stellt eine einzelne Anfrage aus einer beliebigen Quelle dar und enthält Informationen über den angeforderten API-Vorgang, Datum und Uhrzeit des Vorgangs, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordneter Stack-Trace der öffentlichen API-Aufrufe, sodass Ereignisse nicht in einer bestimmten Reihenfolge angezeigt werden.

Das folgende Beispiel zeigt ein CloudTrail Ereignis, das den `CreateSolFunctionPackage` Vorgang demonstriert.


```
{
    "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"
}
```

Informationen zu CloudTrail Datensatzinhalten finden Sie im AWS CloudTrail Benutzerhandbuch unter CloudTrailDatensatzinhalt.

AWS TNB-Bereitstellungsaufgaben

Machen Sie sich mit den Bereitstellungsaufgaben vertraut, um Bereitstellungen effektiv zu überwachen und schneller Maßnahmen zu ergreifen.

In der folgenden Tabelle sind die AWS TNB-Bereitstellungsaufgaben aufgeführt:

Aufgabenname für Bereitstellungen, die vor dem 7. März 2024 gestartet wurden	Aufgabenname für Bereitstellungen, die am und nach dem 7. März 2024 gestartet wurden	Task description (Aufgabenbeschreibung)
AppInstallation	ClusterPluginInstall	Installiert das Multus-Plugin auf dem Amazon EKS-Cluster.

Aufgabenname für Bereitstellungen, die vor dem 7. März 2024 gestartet wurden	Aufgabenname für Bereitstellungen, die am und nach dem 7. März 2024 gestartet wurden	Task description (Aufgabenbeschreibung)
AppUpdate	keine Änderung des Namens	Aktualisiert die Netzwerkfunktionen, die bereits in einer Netzwerkinstanz installiert sind.
-	ClusterPluginUninstall	Deinstalliert die Plugins auf dem Amazon EKS-Cluster.
ClusterStorageClassConfiguration	keine Änderung des Namens	Konfiguriert die Speicherklasse (CSI-Treiber) auf einem Amazon EKS-Cluster.
FunctionDeletion	keine Änderung des Namens	Löscht Netzwerkfunktionen aus AWS TNB-Ressourcen.
FunctionInstantiation	FunctionInstall	Stellt Netzwerkfunktionen mithilfe von HELM bereit.
FunctionUninstallation	FunctionUninstall	Deinstalliert die Netzwerkfunktion von einem Amazon EKS-Cluster.
HookExecution	keine Änderung des Namens	Führt Lifecycle-Hooks aus, wie in der NSD definiert.
InfrastructureCancellation	keine Änderung des Namens	Bricht einen Netzwerkdienst ab.
InfrastructureInstantation	keine Änderung des Namens	Stellt AWS Ressourcen im Namen des Benutzers bereit.
InfrastructureTermination	keine Änderung des Namens	Stellt über TNB aufgerufene AWS Ressourcen ab. AWS
-	InfrastructureUpdate	Aktualisiert die im Namen des Benutzers bereitgestellten AWS Ressourcen.

Aufgabenname für Bereitstellungen, die vor dem 7. März 2024 gestartet wurden	Aufgabenname für Bereitstellungen, die am und nach dem 7. März 2024 gestartet wurden	Task description (Aufgabenbeschreibung)
InventoryDeregistration	keine Änderung des Namens	Meldet AWS Ressourcen von TNB ab AWS .
-	InventoryRegistration	Registriert die AWS Ressourcen in TNB. AWS
KubernetesClusterConfiguration	ClusterConfiguration	Konfiguriert den Kubernetes-Cluster und fügt dem Amazon EKS zusätzliche IAM-Rollen hinzu, AuthMap wie im NSD definiert.
NetworkServiceFinalization	keine Änderung des Namens	Schließt den Netzwerkdienst ab und informiert über den Status eines Erfolgs oder Fehlers.
NetworkServiceInstantiation	keine Änderung des Namens	Initialisiert den Netzwerkdienst.
SelfManagedNodesConfiguration	keine Änderung des Namens	Bootstrappt selbstverwaltete Knoten mit Amazon EKS- und Kubernetes-Steuerebene.
-	ValidateNetworkServiceUpdate	Führt die Validierungen aus, bevor eine Netzwerkinstanz aktualisiert wird.

Servicekontingenten für AWS TNB

Servicekontingenten, auch Limits genannt, sind die maximale Anzahl von Serviceressourcen oder Vorgängen für Ihr AWS Konto. Weitere Informationen finden Sie unter AWS Servicekontingente im Allgemeine Amazon Web Services-Referenz.

Im Folgenden sind die Servicekontingenten für AWS TNB aufgeführt.

Name	Standard	Anpas	Description
Gleichzeitiger laufender Netzwerkdienstbetrieb	Jede unterstützte Region: 40	Ja	Die maximale Anzahl gleichzeitiger laufender Netzwerkdienstoperationen in einer Region.
Funktionspakete	Jede unterstützte Region: 200	Ja	Die maximale Anzahl von Funktionspaketen in einer Region.
Netzwerk-Pakete	Jede unterstützte Region: 40	Ja	Die maximale Anzahl von Netzwerkpaketen in einer Region.
Netzwerkdienstinstanzen	Jede unterstützte Region: 800	Ja	Die maximale Anzahl von Netzwerkdienstinstanzen in einer Region.

Dokumentenverlauf für das AWS TNB-Benutzerhandbuch

In der folgenden Tabelle werden die Dokumentationsversionen für AWS TNB beschrieben.

Änderung	Beschreibung	Datum
Aktualisierungen der Netzwerkkonfiguration der Amazon EKS-Knotengruppe	Fügen Sie Subnetze und Sicherheitsgruppen hinzu und löschen Sie sie. Fügen Sie dem Netzwerk hinzu, ändern Sie es und löschen Sie es ENIs aus dem Netzwerk. Weitere Informationen finden Sie unter Parameter, die Sie aktualisieren können.	10. September 2025
Hinzufügen und Löschen von Amazon EKS-Knotengruppen in vorhandenen Clustern	AWS TNB unterstützt jetzt das Hinzufügen neuer Knotengruppen und das Entfernen vorhandener Knotengruppen aus Amazon EKS-Clustern. Weitere Informationen finden Sie unter Parameter, die Sie aktualisieren können.	4. Juni 2025
Größe des Root-Volumes	Sie können die Größe des zugrunde liegenden Amazon EBS-Root-Volumes Ihrer Amazon EKS-Worker-Knoten über das `root_volume_size` Feld im AWS.Compute angeben. EKSManagedNode und .Compute.AWS EKSSelfManagedNode TOSCA-Knoten.	19. Mai 2025

Referenzressourcen in Skripten	Sie können auf von AWS TNB erstellte Ressourcen verweisen, um sie in Ihren Lifecycle-Hook-Skripten und Benutzerdatenskripten zu konfigurieren.	2. Mai 2025
Kubernetes Version 1.32 wird jetzt für Amazon EKS-Knoten und verwaltete Knotengruppen unterstützt.	AWS TNB unterstützt Kubernetes Version 1.32 für .compute.EKS und .Compute.AWSAWS EKSManagedKnoten.	24. April 2025
Kubernetes Version 1.24 wird für Amazon EKS-Knoten und verwaltete Knotengruppen nicht mehr unterstützt	AWS TNB unterstützt Kubernetes Version 1.24 für .compute.EKS und .Compute nicht mehr.AWSAWS EKSManagedKnoten.	17. April 2025
AL2023 AMI-Unterstützung für von Amazon EKS verwaltete Knoten	AWS TNB unterstützt AL2023 AMI-Typen für AWS.Compute.EKSManagedKnoten.	17. April 2025
Kubernetes Version 1.23 wird für Amazon EKS-Knoten und verwaltete Knotengruppen nicht mehr unterstützt	AWS TNB unterstützt Kubernetes Version 1.23 für .compute.EKS und .Compute nicht mehr.AWSAWS EKSManagedKnoten.	4. April 2025
AMI-ID kann aktualisiert werden	Sie können das Feld ami_id jetzt während eines `UpdateSolNetworkService` API-Aufrufs aktualisieren.	31. März 2025

Kubernetes Version 1.31 wird jetzt für Amazon EKS-Knoten und verwaltete Knotengruppen unterstützt.	AWS TNB unterstützt Kubernetes Version 1.31 für .compute.EKS und .Compute.AWSAWS EKSManagedKnoten.	18. Februar 2025
Kubernetes-Version für AWS.Compute. EKSManagedKnoten	AWS TNB unterstützt die Kubernetes-Versionen 1.23 bis 1.30, um eine von Amazon EKS verwaltete Knotengruppe zu erstellen.	28. Januar 2025
Kubernetes-Version für Cluster	AWS TNB unterstützt jetzt Kubernetes Version 1.30 zur Erstellung von Amazon EKS-Clustern.	19. August 2024

AWS TNB unterstützt einen zusätzlichen Vorgang zur Verwaltung des Netzwerklebenszyklus.	Sie können eine instanziierte oder zuvor aktualisierte Netzwerkinstanz mit einem neuen Netzwerkpaket und neuen Parameterwerten aktualisieren. Siehe:
	• Lebenszyklus-Operationen
	• Aktualisieren Sie eine Netzwerkinstanz
	• AWS Beispiel für eine TNB-Servicerolle:
	  • Fügen Sie diese Amazon EKS-Aktionen hinzu: `eks:UpdateAddon` `eks:UpdateClusterVersion` ,`eks:UpdateNodegroupConfig` ,`eks:UpdateNodegroupVersion` , `eks:DescribeUpdate`
	  • Fügen Sie diese CloudFormation Aktion hinzu: `cloudformation:UpdateStack`
	• Neue Bereitstellungsaufgaben:`InfrastructureUpdate` ,`InventoryRegistration` , `ValidateNetworkServiceUpdate`
		30. Juli 2024

	• API-Updates: GetSolNetworkOperationListSolNetworkOperations , und UpdateSolNetworkInstance	
Neue Aufgabe und neue Aufgabennamen für bestehende Aufgaben	Eine neue Aufgabe ist verfügbar. Seit dem 7. März 2024 haben einige bestehende Aufgaben aus Gründen der Übersichtlichkeit neue Namen.	7. Mai 2024
Kubernetes-Version für Cluster	AWS TNB unterstützt jetzt Kubernetes Version 1.29 zur Erstellung von Amazon EKS-Clustern.	10. April 2024
Support für Netzwerkschnittstellen `security_groups`	Sie können Sicherheitsgruppen an den Knoten AWS.networking.ENI anhängen.	2. April 2024
Support für Amazon EBS-Root-Volume-Verschlüsselung	Sie können die Amazon EBS-Verschlüsselung für das Amazon EBS-Root-Volume aktivieren. Um dies zu aktivieren, fügen Sie die Eigenschaften in AWS.Compute hinzu. EKSManagedNode oder AWS.Compute.EKSSelfManagedNodeKnoten.	2. April 2024
Support für Node `labels`	Sie können Ihrer Knotengruppe in AWS.Compute Knotenbezeichnungen hinzufügen. EKSManagedNode oder AWS.Compute.EKSSelfManagedNodeKnoten.	19. März 2024

Support für Netzwerkschnittstellen `source_dest_check`	Sie können über den Knoten AWS.networking.ENI angeben, ob Sie die source/destination Überprüfung der Netzwerkschnittstelle aktivieren oder deaktivieren möchten.	25. Januar 2024
Support für Amazon EC2 EC2-Instances mit benutzerdefinierten Benutzerdaten	Sie können Amazon EC2 EC2-Instances mit benutzerdefinierten Benutzerdaten über AWS.Compute starten. UserData Knoten.	16. Januar 2024
Support für Security Group	AWS TNB ermöglicht es Ihnen, die Security AWS Group-Ressource zu importieren.	8. Januar 2024
Die Beschreibung von wurde aktualisiert `network_interfaces`	Wenn die `network_interfaces` Eigenschaft in AWS.Compute enthalten ist. EKSManagedNode oder AWS.Compute. EKSSelfManagedNodeKnoten, für den AWS TNB die entsprechende Berechtigung ENIs von der `multus_role` Eigenschaft erhält, falls verfügbar, oder von der Eigenschaft. `node_role`	18. Dezember 2023
Support für private Cluster	AWS TNB unterstützt jetzt private Cluster. Um einen privaten Cluster anzugeben, setzen Sie die `access` Eigenschaft auf`PRIVATE`.	11. Dezember 2023

Kubernetes-Version für Cluster	AWS TNB unterstützt jetzt Kubernetes Version 1.28 zur Erstellung von Amazon EKS-Clustern.	11. Dezember 2023
AWS TNB unterstützt Platzierungsgruppen	Platzierungsgruppe für die `AWS.Compute.EKSSelfManagedNode` Knotendefinitionen `AWS.Compute.EKSManagedNode` und hinzugefügt.	11. Dezember 2023

AWS TNB fügt Unterstützung hinzu für IPv6	AWS TNB unterstützt jetzt die Erstellung von Netzwerkinstanzen mit IPv6 Infrastruktur. Überprüfen Sie die Knoten AWS.Networking.VPC, .Networking.Subnet, .Networking.AWSAWS https://docs.aws.amazon.com/tnb/latest/ug/node-internet-gateway.htmlInternetGatewayAWS, .Netzwerke. SecurityGroupIngressRule, AWS.Netzwerke. SecurityGroupEgressRule, und AWS.compute.EKS für Konfigurationen. IPv6 Wir haben auch die Knoten .Networking hinzugefügt.AWS NATGatewayund AWS.Networking.Route für die Konfiguration. NAT64 Wir haben die AWS TNB-Servicerolle und die AWS TNB-Servicerolle für die Amazon EKS-Knotengruppe für IPv6 Berechtigungen aktualisiert. Siehe Beispiele für Richtlinien für Servicerollen.	16. November 2023
Der AWS TNB-Servicerollenrichtlinie wurden Berechtigungen hinzugefügt	Wir haben der AWS TNB-Servicerollenrichtlinie für Amazon S3 und CloudFormation zur Aktivierung der Infrastrukturinstanziierung Berechtigungen hinzugefügt.	23. Oktober 2023

AWS TNB wurde in mehr Regionen eingeführt	AWS TNB ist jetzt in den Regionen Asien-Pazifik (Seoul), Kanada (Zentral), Europa (Spanien), Europa (Stockholm) und Südamerika (São Paulo) verfügbar.	27. September 2023
Tags für AWS.Compute. EKSSelfManagedNode	AWS TNB unterstützt jetzt Tags für die `AWS.Compute.EKSSelfManagedNode` Knotendefinition.	22. August 2023
AWS TNB unterstützt Instances, die IMDSv2	Wenn Sie Ihre Instance starten, müssen Sie Folgendes verwenden IMDSv2.	14. August 2023
Aktualisierte Berechtigungen für MultusRoleInlinePolicy	Das beinhaltet MultusRoleInlinePolicy jetzt die `ec2:DeleteNetworkInterface` Erlaubnis.	7. August 2023
Kubernetes-Version für Cluster	AWS TNB unterstützt jetzt Kubernetes-Versionen 1.27 zur Erstellung von Amazon EKS-Clustern.	25. Juli 2023
AWS.compute.EKS. AuthRole	AWS TNB unterstützt AuthRole das, sodass Sie dem Amazon EKS-Cluster IAM-Rollen hinzufügen können, `aws-auth ConfigMap` sodass Benutzer über eine IAM-Rolle auf den Amazon EKS-Cluster zugreifen können.	19. Juli 2023

AWS TNB unterstützt Sicherheitsgruppen.	Das AWS.Networking wurde hinzugefügt. SecurityGroup, AWS.Netzwerke. SecurityGroupEgressRule, und AWS.Networking. SecurityGroupIngressRulezur NSD-Vorlage.	18. Juli 2023
Kubernetes-Version für Cluster	AWS TNB unterstützt die Kubernetes-Versionen 1.22 bis 1.26 zur Erstellung von Amazon EKS-Clustern. AWS TNB unterstützt die Kubernetes-Versionen 1.21 nicht mehr.	11. Mai 2023
AWS. Rechnen. EKSSelfManagedNode	Sie können selbstverwaltete Worker-Knoten in regionalen, AWS Local Zones und erstellen. AWS Outposts	29. März 2023
Erstversion	Dies ist die erste Version des AWS TNB-Benutzerhandbuchs.	21. Februar 2023

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.