Guide de l'utilisateur

AWS Générateur de réseaux de télécommunications

AWS Générateur de réseaux de télécommunications: Guide de l’utilisateur

Table of Contents

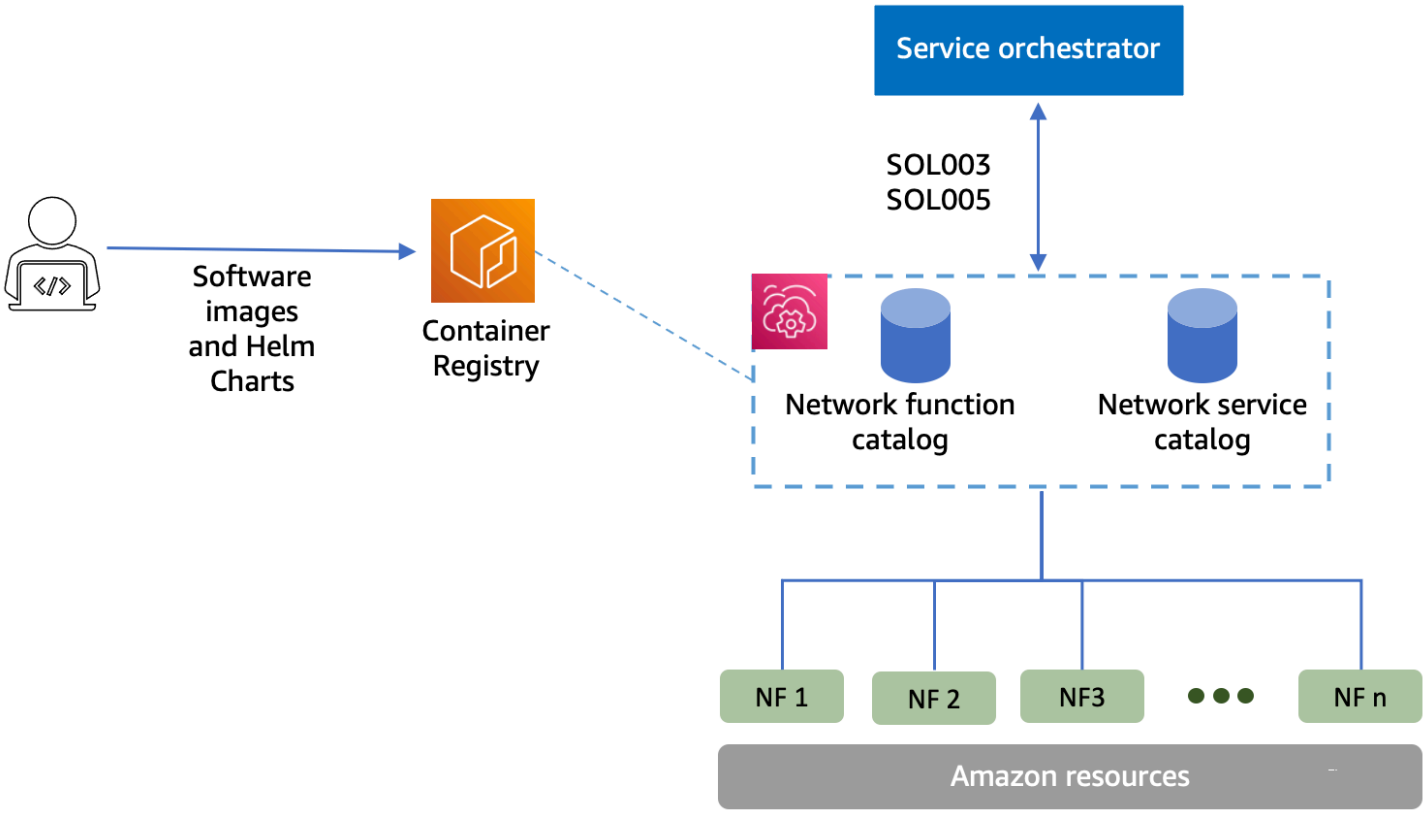
Qu'est-ce que AWS Telco Network Builder ?

AWS Telco Network Builder (AWS TNB) est un AWS service qui fournit aux fournisseurs de services de communication (CSPs) un moyen efficace de déployer, de gérer et de faire évoluer les réseaux 5G sur AWS l'infrastructure.

Avec AWS TNB, vous déployez des réseaux 5G évolutifs et sécurisés en AWS Cloud utilisant une image de votre réseau de manière automatisée. Vous n'avez pas besoin d'apprendre de nouvelles technologies, de choisir le service informatique à utiliser ou de savoir comment approvisionner et configurer les AWS ressources.

Vous décrivez plutôt l'infrastructure de votre réseau et fournissez les images logicielles des fonctions du réseau fournies par vos partenaires fournisseurs de logiciels indépendants (ISV). AWS TNB s'intègre à des orchestrateurs de AWS services et à des services tiers pour fournir automatiquement l' AWS infrastructure nécessaire, déployer des fonctions réseau conteneurisées et configurer le réseau et la gestion des accès afin de créer un service réseau entièrement opérationnel.

Le schéma suivant illustre les intégrations logiques entre le AWS TNB et les orchestrateurs de services pour déployer des fonctions réseau à l'aide d'interfaces standard basées sur l'ETSI (European Telecommunications Standards Institute).



Rubriques

Nouveau AWS ?

Si vous débutez dans le domaine des AWS produits et services, commencez à en apprendre davantage à l'aide des ressources suivantes :

- Présentation de AWS
- Commencer avec AWS

À qui s'adresse le AWS TNB ?

AWS Le TNB vise à CSPs tirer parti de la rentabilité, de l'agilité et de l'élasticité qu'il AWS Cloud offre sans écrire ni gérer de scripts et de configurations personnalisés pour concevoir, déployer et gérer des services réseau. AWS TNB fournit automatiquement l' AWS infrastructure nécessaire, déploie les fonctions réseau conteneurisées et configure le réseau et la gestion des accès afin de créer des services réseau entièrement opérationnels basés sur les descripteurs de services réseau définis par le CSP et les fonctions réseau que le CSP souhaite déployer.

AWS Caractéristiques du TNB

Voici quelques-unes des raisons pour lesquelles un CSP souhaiterait utiliser le AWS TNB :

Aide à simplifier les tâches

> Améliorez l'efficacité des opérations de votre réseau, telles que le déploiement de nouveaux services, la mise à jour et la mise à niveau des fonctions réseau et la modification des topologies de l'infrastructure réseau.

S'intègre aux orchestrateurs

AWS TNB s'intègre aux orchestrateurs de services tiers les plus populaires conformes à l'ETSI.

Balances

Vous pouvez configurer le AWS TNB pour adapter les AWS ressources sous-jacentes afin de répondre à la demande de trafic, de mettre à jour plus efficacement les fonctions du réseau, de déployer les modifications de topologie de l'infrastructure réseau et de réduire le temps de déploiement des nouveaux services 5G de plusieurs jours à quelques heures.

Inspecte et surveille les ressources AWS

AWS TNB vous permet d'inspecter et de surveiller les AWS ressources qui soutiennent votre réseau sur un tableau de bord unique, comme Amazon VPC, Amazon et EC2 Amazon EKS.

Supporte les modèles de services

AWS TNB vous permet de créer des modèles de services pour toutes les charges de travail des télécommunications (RAN, Core, IMS). Vous pouvez créer une nouvelle définition de service, réutiliser un modèle existant ou intégrer un pipeline d'intégration et de livraison continues (CI/CD) pour publier une nouvelle définition.

Suit les modifications apportées aux déploiements réseau

Lorsque vous modifiez la configuration sous-jacente d'un déploiement de fonctions réseau, par exemple en modifiant le type d'instance d'un type d' EC2 instance Amazon, vous pouvez suivre les modifications de manière reproductible et évolutive. Pour ce faire manuellement, il faudrait gérer l'état du réseau, créer et supprimer des ressources, et faire attention à l'ordre des modifications nécessaires. Lorsque vous utilisez AWS TNB pour gérer le cycle de vie de votre fonction réseau, vous modifiez uniquement les descripteurs de service réseau décrivant la fonction réseau. AWS TNB effectuera alors automatiquement les modifications requises dans le bon ordre.

Simplifie le cycle de vie des fonctions du réseau

Vous pouvez gérer la première version et toutes les versions suivantes d'une fonction réseau et spécifier le moment de la mise à niveau. Vous pouvez également gérer vos applications RAN, Core, IMS et réseau de la même manière.

Accès à AWS TNB

Vous pouvez créer, accéder et gérer vos ressources AWS TNB à l'aide de l'une des interfaces suivantes :

- AWS Console TNB : fournit une interface Web pour gérer votre réseau.
- AWS API TNB — Fournit une RESTful API pour effectuer des actions AWS TNB. Pour plus d'informations, consultez la référence de l'API AWS TNB.
- AWS Command Line Interface (AWS CLI) — Fournit des commandes pour un large éventail de AWS services, y compris AWS TNB. Il est compatible avec Windows, macOS et Linux. Pour plus d'informations, consultez le AWS Command Line Interface Guide de l'utilisateur .
- AWS SDKs— Fournit des informations spécifiques à la langue APIs et complète de nombreux détails de connexion. Ces outils incluent le calcul des signatures, la gestion des nouvelles tentatives de demande et la gestion des erreurs. Pour de plus amples informations, veuillez consulter AWS SDKs.

Tarification du AWS TNB

AWS TNB aide à CSPs automatiser le déploiement et la gestion de ses réseaux de télécommunications sur AWS. Vous payez pour les deux dimensions suivantes lorsque vous utilisez le AWS TNB :

- Par heures d'unité de fonction réseau gérée (MNFI).
- Par nombre de demandes d'API.

Vous devez également payer des frais supplémentaires lorsque vous utilisez d'autres AWS services en association avec AWS TNB. Pour plus d'informations, consultez la section Tarification AWS TNB.

Pour consulter votre facture, accédez au Tableau de bord de gestion des coûts et de la facturation dans la console AWS Billing and Cost Management. Votre facture contient des liens vers les rapports d'utilisation qui fournissent des détails supplémentaires sur votre facture. Pour plus d'informations sur la facturation AWS du compte, consultez la section Facturation AWS du compte.

Si vous avez des questions concernant la AWS facturation, les comptes et les événements, contactez le AWS Support.

AWS Trusted Advisor est un service que vous pouvez utiliser pour optimiser les coûts, la sécurité et les performances de votre AWS environnement. Pour plus d'informations, consultez AWS Trusted Advisor.

Quelle est la prochaine étape

Pour plus d'informations sur la façon de démarrer avec AWS TNB, consultez les rubriques suivantes :

- Configuration AWS TNB— Effectuez les étapes préalables.
- Débuter avec AWS TNB— Déployez votre première fonction réseau, telle qu'une unité centralisée (CU), une fonction de gestion de l'accès et de la mobilité (AMF), une fonction de plan utilisateur (UPF) ou un cœur 5G complet.

Comment AWS fonctionne TNB

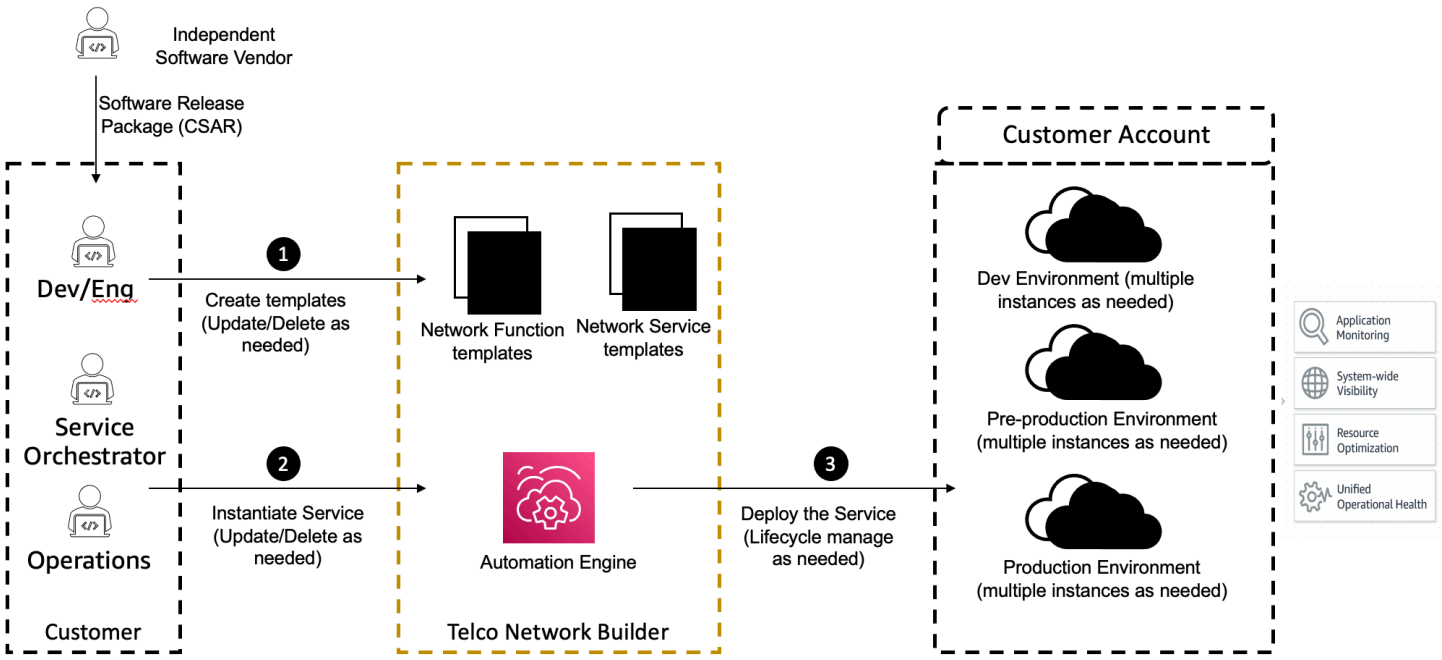
AWS TNB s'intègre à des end-to-end orchestrateurs et à des AWS ressources standardisés pour exploiter des réseaux 5G complets.

AWS TNB vous permet d'ingérer des packages de fonctions réseau et des descripteurs de services réseau (NSDs) et vous fournit le moteur d'automatisation nécessaire au fonctionnement de vos réseaux. Vous pouvez utiliser votre end-to-end orchestrateur et l'intégrer à AWS TNB APIs, ou utiliser AWS TNB SDKs pour créer votre propre flux d'automatisation. Pour de plus amples informations, veuillez consulter AWS Architecture TNB.

Rubriques

AWS Architecture TNB

AWS TNB vous permet d'effectuer des opérations de gestion du cycle de vie via l' AWS Management Console API REST AWS TNB et. AWS CLI SDKs Cela permet aux différents acteurs du CSP, tels que les membres des équipes d'ingénierie, d'exploitation et de systèmes programmatiques, de tirer parti du TNB. AWS Vous créez et téléchargez un package de fonctions réseau sous forme de fichier Cloud Service Archive (CSAR). Le fichier CSAR contient des diagrammes Helm, des images logicielles et un descripteur de fonction réseau (NFD). Vous pouvez utiliser des modèles pour déployer à plusieurs reprises plusieurs configurations de ce package. Vous créez des modèles de services réseau qui définissent l'infrastructure et les fonctions réseau que vous souhaitez déployer. Vous pouvez utiliser des remplacements de paramètres pour déployer différentes configurations à différents emplacements. Vous pouvez ensuite instancier un réseau à l'aide des modèles et déployer vos fonctions réseau sur AWS l'infrastructure. AWS TNB vous fournit la visibilité de vos déploiements.



Intégration avec Services AWS

Un réseau 5G est constitué d'un ensemble de fonctions réseau conteneurisées interconnectées déployées sur des milliers de clusters Kubernetes. AWS TNB intègre les éléments suivants Services AWS , spécifiques aux télécoms, APIs afin de créer un service réseau entièrement opérationnel :

- Amazon Elastic Container Registry (Amazon ECR) pour stocker les artefacts des fonctions réseau des fournisseurs de logiciels indépendants ISVs ().
- Amazon Elastic Kubernetes Service (Amazon EKS) pour configurer des clusters.
- Amazon VPC pour les constructions réseau.
- Groupes de sécurité utilisant CloudFormation.
- AWS CodePipeline pour les cibles de déploiement dans Régions AWS AWS les Zones Locales et AWS Outposts.
- IAM pour définir les rôles.
- AWS Organizations pour contrôler l'accès au AWS TNB APIs.
- Tableau de bord Health et AWS CloudTrail pour surveiller l'état de santé et publier des indicateurs.

AWS Quotas de ressources TNB

Vous Compte AWS disposez de quotas par défaut, anciennement appelés limites, pour chacun d'entre eux Service AWS. Sauf indication contraire, chaque quota est spécifique à un Région AWS. Vous pouvez demander l'augmentation de certains quotas, mais pas de tous les quotas.

Pour consulter les quotas de AWS TNB, ouvrez la console Service Quotas. Dans le volet de navigation, choisissez Services AWS, puis sélectionnez AWS TNB.

Pour demander une augmentation de quota, consultez Demande d'augmentation de quota dans le Guide de l'utilisateur Service Quotas.

Vous Compte AWS avez les quotas suivants liés au AWS TNB.

Quota de ressources	Description	Valeur par défaut	Ajustable?
Instances de service réseau	Le nombre maximum d'instances de service réseau dans une région.	800	Oui
Opérations de service réseau continues simultanées	Le nombre maximum d'opérations de service réseau en cours simultanées dans une région.	40	Oui
Packages réseau	Le nombre maximum de packages réseau dans une région.	40	Oui
Packages de fonctions	Le nombre maximum de packages de fonctions dans une région.	200	Oui

AWS Concepts du TNB

Cette rubrique décrit les concepts essentiels pour vous aider à commencer à utiliser le AWS TNB.

Table des matières

Cycle de vie d'une fonction réseau

AWS TNB vous aide tout au long du cycle de vie des fonctions de votre réseau. Le cycle de vie des fonctions réseau comprend les étapes et activités suivantes :

Planification

1. Planifiez votre réseau en identifiant les fonctions réseau à déployer.
2. Placez les images du logiciel de fonction réseau dans un référentiel d'images de conteneur.
3. Créez les packages CSAR à déployer ou à mettre à niveau.
4. Utilisez AWS TNB pour télécharger le package CSAR qui définit votre fonction réseau (par exemple, CU AMF et UPF) et intégrez-le à un pipeline d'intégration et de livraison continues (CI/CD) qui peut vous aider à créer de nouvelles versions de votre package CSAR à mesure que de nouvelles images du logiciel de fonction réseau ou des scripts client sont disponibles.

Configuration

1. Identifiez les informations requises pour le déploiement, telles que le type de calcul, la version de la fonction réseau, les informations IP et les noms des ressources.
2. Utilisez ces informations pour créer votre descripteur de service réseau (NSD).
3. Ingérez NSDs qui définissent les fonctions de votre réseau et les ressources nécessaires à l'instanciation de la fonction réseau.

Instanciation

1. Créez l'infrastructure requise par les fonctions du réseau.

2. Instanciez (ou provisionnez) la fonction réseau telle que définie dans son NSD et commencez à transporter le trafic.
3. Validez les actifs.

Production

Au cours du cycle de vie de la fonction réseau, vous effectuerez des opérations de production, telles que :

- Mettez à jour la configuration de la fonction réseau, par exemple, mettez à jour une valeur dans la fonction réseau déployée.
- Mettez à jour l'instance réseau avec un nouveau package réseau et de nouvelles valeurs de paramètres. Par exemple, mettez à jour le `version` paramètre Amazon EKS dans le package réseau.

Utiliser des interfaces standardisées

AWS TNB s'intègre aux orchestrateurs de services conformes à la norme ETSI (European Telecommunications Standards Institute), ce qui vous permet de simplifier le déploiement de vos services réseau. Les orchestrateurs de services peuvent utiliser le AWS TNB SDKs, la CLI ou le APIs pour lancer des opérations, telles que l'instanciation ou la mise à niveau d'une fonction réseau vers une nouvelle version.

AWS TNB prend en charge les spécifications suivantes.

Spécification de	Version	Description
ETSI SOL001	v3.6.1	Définit les normes permettant d'autoriser les descripteurs de fonctions réseau basés sur TOSCA.
ETSI SOL002	v3.6.1	Définit les modèles relatifs à la gestion des fonctions réseau.
ETSI SOL003	v3.6.1	Définit les normes pour la gestion du cycle de vie des fonctions réseau.
ETSI SOL004	v3.6.1	Définit les normes CSAR pour les packages de fonctions réseau.

Spécification de	Version	Description
ETSI SOL005	v3.6.1	Définit les normes relatives aux packages de services réseau et à la gestion du cycle de vie des services réseau.
ETSI SOL007	v3.5.1	Définit les normes permettant d'autoriser les descripteurs de service réseau basés sur TOSCA.

Package de fonctions

Avec AWS TNB, vous pouvez stocker des packages de fonctions conformes aux normes ETSI SOL001/SOL004 dans un catalogue de fonctions. Vous pouvez ensuite télécharger des packages Cloud Service Archive (CSAR) contenant des artefacts décrivant le fonctionnement de votre réseau virtuel.

- Descripteur de fonction réseau virtuel : définit les métadonnées pour l'intégration des packages et la gestion des fonctions du réseau virtuel. Vous devez nommer ce fichier`vnfd.yaml`.
- Images logicielles — Fait référence à la fonction réseau virtuel Container Images. Amazon Elastic Container Registry (Amazon ECR) peut servir de référentiel d'images de fonctions de réseau virtuel.
- Fichiers supplémentaires : à utiliser pour gérer la fonction du réseau virtuel, par exemple, les scripts et les diagrammes Helm.

Le CSAR est un package défini par la norme OASIS TOSCA et inclut un descripteur de réseau/service conforme à la spécification OASIS TOSCA YAML. Pour plus d'informations sur la spécification YAML requise, consultezRéférence TOSCA pour TNB AWS.

Voici un exemple de descripteur de fonction de réseau virtuel.

```
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"
```

Package réseau

Un package réseau est un `.zip` fichier au format CSAR (Cloud Service Archive). Il définit les packages de fonctions que vous souhaitez déployer et l' AWS infrastructure sur laquelle vous souhaitez les déployer.

Le package réseau contient les fichiers suivants :

- Un fichier descripteur réseau (`nsd.yaml`) au format TOSCA tel que décrit par l'ETSI SOL007.

  Le `nsd.yaml` fichier contient des références aux packages de fonctions téléchargés avec leur descripteur. IDs
- Scripts de données utilisateur, le cas échéant.
- Scripts Lifecycle Hook, le cas échéant.
- Les fichiers de `values.yaml` configuration des plugins, le cas échéant.

AWS TNB prend en charge les normes ETSI pour la modélisation des ressources, telles que le réseau, le service et la fonction, dans le langage TOSCA. AWS TNB vous permet de les utiliser plus efficacement en les Services AWS modélisant de manière à ce que votre orchestrateur de services conforme à la norme ETSI puisse les comprendre.

Descripteurs de service réseau pour TNB AWS

Un descripteur de service réseau (NSD) est un `.yaml` fichier d'un package réseau qui utilise la norme TOSCA pour décrire les fonctions réseau que vous souhaitez déployer et l' AWS infrastructure sur laquelle vous souhaitez déployer les fonctions réseau. Pour définir votre NSD et configurer vos ressources sous-jacentes et les opérations du cycle de vie du réseau, vous devez comprendre le schéma NSD TOSCA pris en charge par TNB. AWS

Votre fichier NSD est divisé en plusieurs parties :

1. Version de définition TOSCA — Il s'agit de la première ligne de votre fichier NSD YAML et contient les informations de version, illustrées dans l'exemple suivant.

```
tosca_definitions_version: tnb_simple_yaml_1_0
```

2. VNFD — Le NSD contient la définition de la fonction réseau sur laquelle effectuer les opérations du cycle de vie. Chaque fonction réseau doit être identifiée par les valeurs suivantes :
   - Un identifiant unique pour`descriptor_id`. L'identifiant doit correspondre à celui du package CSAR de la fonction réseau.
   - Un nom unique pour`namespace`. Le nom doit être associé à un identifiant unique afin de pouvoir le référencer plus facilement dans l'ensemble de votre fichier NSD YAML, comme illustré dans l'exemple suivant.

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

3. Modèle de topologie : définit les ressources à déployer, le déploiement des fonctions réseau et tous les scripts personnalisés, tels que les hooks du cycle de vie. Voici un exemple :

```
topology_template:

  node_templates:

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

```
        descriptor_name: "<Sample name>"
```

4. Nœuds supplémentaires : chaque ressource modélisée comporte des sections pour les propriétés et les exigences. Les propriétés décrivent les attributs facultatifs ou obligatoires d'une ressource, tels que la version. Les exigences décrivent les dépendances qui doivent être fournies en tant qu'arguments. Par exemple, pour créer une ressource de groupe de nœuds Amazon EKS, celle-ci doit être créée au sein d'un cluster Amazon EKS. Voici un exemple :

```
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
```

Exemple NSD

Ce qui suit est un extrait d'un NSD montrant comment modéliser. Services AWS La fonction réseau sera déployée sur un cluster Amazon EKS avec Kubernetes version 1.27. Les sous-réseaux des applications sont Subnet01 et Subnet02. Vous pouvez ensuite définir le NodeGroups pour vos applications à l'aide d'une Amazon Machine Image (AMI), d'un type d'instance et d'une configuration de mise à l'échelle automatique.

```
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
```

Gestion et opérations pour AWS TNB

Avec AWS TNB, vous pouvez gérer votre réseau à l'aide d'opérations de gestion standardisées conformément aux normes ETSI SOL003 et SOL005. Vous pouvez utiliser le AWS TNB APIs pour effectuer des opérations de cycle de vie telles que :

- Instanciation des fonctions de votre réseau.
- Mettre fin aux fonctions de votre réseau.
- Mettre à jour les fonctions de votre réseau pour annuler les déploiements Helm.
- Mettre à jour une instance réseau instanciée ou mise à jour avec un nouveau package réseau et de nouvelles valeurs de paramètres.
- Gestion des versions de vos packages de fonctions réseau.
- Gestion des versions de votre NSDs.
- Récupération d'informations sur les fonctions de votre réseau déployé.

Configuration AWS TNB

Configurez AWS TNB en effectuant les tâches décrites dans cette rubrique.

Tâches

- Inscrivez-vous pour un Compte AWS
- Choisissez un AWS Région
- Notez le point de terminaison du service
- (Facultatif) Installez AWS CLI
- Configuration AWS Rôles TNB

Inscrivez-vous pour un Compte AWS

Pour commencer AWS, vous avez besoin d'un Compte AWS. Pour plus d'informations sur la création d'un Compte AWS, voir Getting started with an Compte AWS dans le Guide de Gestion de compte AWS référence.

Choisissez un AWS Région

Pour consulter la liste des régions disponibles pour AWS TNB, consultez la liste des services AWS régionaux. Pour consulter la liste des points de terminaison pour un accès programmatique, voir les points de terminaison AWS TNB dans le. Références générales AWS

Notez le point de terminaison du service

Pour vous connecter par programmation à un AWS service, vous utilisez un point de terminaison. Outre les points de AWS terminaison standard, certains AWS services proposent des points de terminaison FIPS dans certaines régions. Pour plus d'informations, consultez Points de terminaison du service AWS.

Nom de la région	Région	Point de terminaison	Protocole	
USA Est (Virginie du Nord)	us-east-1	tnb.us-east-1.amazonaws.com	HTTPS	
USA Ouest (Oregon)	us-west-2	tnb.us-west-2.amazonaws.com	HTTPS	
Asia Pacific (Seoul)	ap-northeast-2	tnb.ap-northeast-2.amazonaws.com	HTTPS	
Asie-Pacifique (Sydney)	ap-southeast-2	tnb.ap-southeast-2.amazonaws.com	HTTPS	
Canada (Centre)	ca-central-1	tnb.ca-central-1.amazonaws.com	HTTPS	
Europe (Francfort)	eu-central-1	tnb.eu-central-1.amazonaws.com	HTTPS	
Europe (Paris)	eu-west-3	tnb.eu-west-3.amazonaws.com	HTTPS	
Europe (Espagne)	eu-south-2	tnb.eu-south-2.amazonaws.com	HTTPS	
Europe (Stockholm)	eu-north-1	tnb.eu-north-1.amazonaws.com	HTTPS	
Amérique du Sud	sa-east-1	tnb.sa-east-1.amazonaws.com	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole	
(São Paulo)				

(Facultatif) Installez AWS CLI

Le AWS Command Line Interface (AWS CLI) fournit des commandes pour un large éventail de AWS produits et est pris en charge sous Windows, macOS et Linux. Vous pouvez accéder à AWS TNB à l'aide du AWS CLI. Consultez le AWS Command Line Interface Guide de l'utilisateur pour démarrer. Pour plus d'informations sur les commandes pour AWS TNB, voir tnb dans le manuel de référence des AWS CLI commandes.

Configuration AWS Rôles TNB

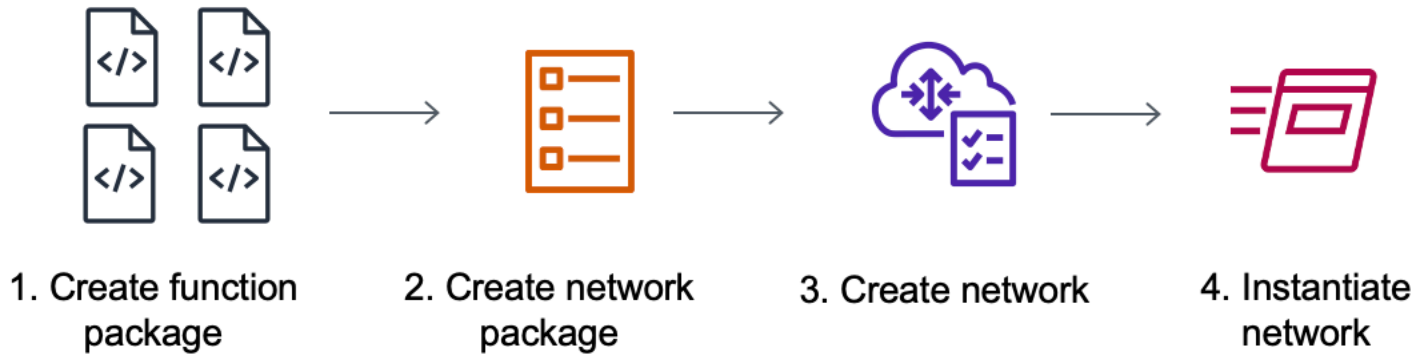
Vous devez créer un rôle de service IAM pour gérer les différentes parties de votre solution AWS TNB. AWS Les rôles de service TNB peuvent effectuer des appels d'API vers d'autres AWS services, tels que AWS CloudFormation AWS CodeBuild, et divers services de calcul et de stockage, en votre nom, afin d'instancier et de gérer les ressources pour votre déploiement.

Pour plus d'informations sur le rôle de service AWS TNB, consultezGestion des identités et des accès pour AWS TNB.

Débuter avec AWS TNB

Ce didacticiel explique comment utiliser le AWS TNB pour déployer une fonction réseau, par exemple l'unité centralisée (CU), la fonction de gestion de l'accès et de la mobilité (AMF) ou la fonction de plan utilisateur 5G (UPF).

Le schéma suivant illustre le processus de déploiement :



Tâches

- Prérequis
- Création d'un package de fonctions
- Création d'un package réseau
- Création et instanciation d'une instance réseau
- Nettoyage

Prérequis

Avant de pouvoir effectuer un déploiement réussi, vous devez disposer des éléments suivants :

- Un plan de Support aux AWS entreprises.
- Autorisations via les rôles IAM.
- Un package de fonctions réseau (NF) conforme à la norme ETSI SOL001/SOL004.
- Modèles de descripteur de service réseau (NSD) conformes à la norme ETSI SOL007.

Vous pouvez utiliser un exemple de package de fonctions ou de package réseau sur le GitHub site Sample packages for AWS TNB.

Création d'un package de fonctions

Un package de fonctions réseau est un fichier Cloud Service Archive (CSAR). Le fichier CSAR contient des diagrammes Helm, des images logicielles et un descripteur de fonction réseau (NFD).

Pour créer un package de fonctions

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, choisissez Function packages.
3. Choisissez Créer un package de fonctions.
4. Sous Upload function package, choisissez Choose files, puis téléchargez chaque package CSAR sous forme de `.zip` fichier. Vous pouvez télécharger un maximum de 10 fichiers.
5. (Facultatif) Sous Balises, choisissez Ajouter une nouvelle balise et entrez une clé et une valeur. Vous pouvez utiliser des balises pour rechercher et filtrer vos ressources ou suivre vos AWS coûts.
6. Choisissez Suivant.
7. Vérifiez les détails du package, puis choisissez Create function package.

Création d'un package réseau

Un package réseau indique les fonctions réseau que vous souhaitez déployer et la manière dont vous souhaitez les déployer dans le catalogue.

Pour créer un package réseau

1. Dans le volet de navigation, sélectionnez Network packages.
2. Choisissez Créer un package réseau.
3. Sous Télécharger le package réseau, choisissez Choisir les fichiers et chargez chaque NSD sous forme de `.zip` fichier. Vous pouvez télécharger un maximum de 10 fichiers.
4. (Facultatif) Sous Balises, choisissez Ajouter une nouvelle balise et entrez une clé et une valeur. Vous pouvez utiliser des balises pour rechercher et filtrer vos ressources ou suivre vos AWS coûts.

5. Choisissez Suivant.
6. Choisissez Créer un package réseau.

Création et instanciation d'une instance réseau

Une instance réseau est un réseau unique créé dans AWS TNB qui peut être déployé. Vous devez créer une instance réseau et l'instancier. Lorsque vous instanciez une instance réseau, AWS TNB fournit l' AWS infrastructure nécessaire, déploie des fonctions réseau conteneurisées et configure le réseau et la gestion des accès pour créer un service réseau entièrement opérationnel.

Pour créer et instancier une instance réseau

1. Dans le volet de navigation, sélectionnez Networks.
2. Choisissez Create network instance.
3. Entrez un nom et une description pour le réseau, puis choisissez Next.
4. Choisissez un package réseau. Vérifiez les informations et choisissez Next.
5. Choisissez Create network instance. L'état initial est`Created`.

   La page Réseaux apparaît et indique la nouvelle instance de réseau dans son `Not instantiated` état actuel.
6. Sélectionnez l'instance réseau, choisissez Actions et Instanciation.

   La page Network instanciate apparaît.
7. Vérifiez les détails et mettez à jour les valeurs des paramètres. Les mises à jour des valeurs des paramètres s'appliquent uniquement à cette instance réseau. Les paramètres des packages NSD et VNFD ne changent pas.
8. Choisissez Instancier le réseau.

   La page État du déploiement apparaît.
9. Utilisez l'icône Actualiser pour suivre l'état de déploiement de votre instance réseau. Vous pouvez également activer l'actualisation automatique dans la section Tâches de déploiement pour suivre la progression de chaque tâche.

Nettoyage

Vous pouvez désormais supprimer les ressources que vous avez créées pour ce didacticiel.

Pour nettoyer vos ressources

1. Dans le volet de navigation, sélectionnez Networks.
2. Choisissez l'ID du réseau, puis sélectionnez Terminate.
3. Lorsque vous êtes invité à confirmer, entrez l'ID réseau, puis choisissez Terminate.
4. Utilisez l'icône Actualiser pour suivre l'état de votre instance réseau.
5. (Facultatif) Sélectionnez le réseau, puis choisissez Supprimer.

Packages de fonctions pour AWS TNB

Un package de fonctions est un fichier .zip au format CSAR (Cloud Service Archive) qui contient une fonction réseau (une application de télécommunication standard ETSI) et un descripteur de package de fonctions qui utilise la norme TOSCA pour décrire comment les fonctions réseau doivent s'exécuter sur votre réseau.

Tâches

- Créez un package de fonctions dans AWS TNB
- Afficher un package de fonctions dans AWS TNB
- Téléchargez un package de fonctions depuis AWS TNB
- Supprimer un package de fonctions de AWS TNB

Créez un package de fonctions dans AWS TNB

Découvrez comment créer un package de fonctions dans le catalogue des fonctions réseau AWS TNB. La création d'un package de fonctions est la première étape pour créer un réseau dans AWS TNB. Après avoir chargé un package de fonctions, vous pouvez créer un package réseau.

Console

Pour créer un package de fonctions à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, choisissez Function packages.
3. Choisissez Créer un package de fonctions.
4. Choisissez Choisir des fichiers et téléchargez chaque package CSAR sous forme de `.zip` fichier. Vous pouvez télécharger un maximum de 10 fichiers.
5. Choisissez Suivant.
6. Vérifiez les détails du package.
7. Choisissez Créer un package de fonctions.

AWS CLI

Pour créer un package de fonctions à l'aide du AWS CLI

1. Utilisez la create-sol-function-packagecommande pour créer un nouveau package de fonctions :

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

2. Utilisez la commande put-sol-function-package-content pour télécharger le contenu du package de fonctions. Par exemple :

```
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
```

Afficher un package de fonctions dans AWS TNB

Découvrez comment afficher le contenu d'un package de fonctions.

Console

Pour afficher un package de fonctions à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, choisissez Function packages.
3. Utilisez le champ de recherche pour trouver le package de fonctions

AWS CLI

Pour afficher un package de fonctions à l'aide du AWS CLI

1. Utilisez la list-sol-function-packagescommande pour répertorier vos packages de fonctions.

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

2. Utilisez la get-sol-function-packagecommande pour afficher les détails d'un package de fonctions.

```
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
```

Téléchargez un package de fonctions depuis AWS TNB

Découvrez comment télécharger un package de fonctions à partir du catalogue de fonctions réseau AWS TNB.

Console

Pour télécharger un package de fonctions à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation sur le côté gauche de la console, choisissez Function packages.
3. Utilisez le champ de recherche pour trouver le package de fonctions
4. Choisissez le package de fonctions
5. Choisissez Actions, puis Télécharger.

AWS CLI

Pour télécharger un package de fonctions à l'aide du AWS CLI

Utilisez la commande get-sol-function-package-content pour télécharger un package de fonctions.

```
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
```

Supprimer un package de fonctions de AWS TNB

Découvrez comment supprimer un package de fonctions du catalogue de fonctions réseau AWS TNB. Pour supprimer un package de fonctions, celui-ci doit être désactivé.

Console

Pour supprimer un package de fonctions à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, choisissez Function packages.
3. Utilisez le champ de recherche pour trouver le package de fonctions.
4. Choisissez un pack de fonctions.
5. Choisissez Actions, Désactiver .
6. Sélectionnez Actions, Supprimer.

AWS CLI

Pour supprimer un package de fonctions à l'aide du AWS CLI

1. Utilisez la update-sol-function-packagecommande pour désactiver un package de fonctions.

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

2. Utilisez la delete-sol-function-packagecommande pour supprimer un package de fonctions.

```
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
```

Packages réseau pour AWS TNB

Un package réseau est un fichier .zip au format CSAR (Cloud Service Archive). Il définit les packages de fonctions que vous souhaitez déployer et l' AWS infrastructure sur laquelle vous souhaitez les déployer.

Le package réseau contient les fichiers suivants :

- Un fichier descripteur réseau (`nsd.yaml`) au format TOSCA tel que décrit par l'ETSI. SOL007

  Le `nsd.yaml` fichier contient des références aux packages de fonctions téléchargés avec leur descripteur. IDs
- Scripts de données utilisateur, le cas échéant.
- Scripts Lifecycle Hook, le cas échéant.
- Les fichiers de `values.yaml` configuration des plugins, le cas échéant.

Tâches

- Créez un package réseau dans AWS TNB
- Afficher un package réseau dans AWS TNB
- Téléchargez un package réseau auprès de AWS TNB
- Supprimer un package réseau de AWS TNB

Créez un package réseau dans AWS TNB

Un package réseau se compose d'un fichier descripteur de service réseau (NSD) (obligatoire) et de tout fichier supplémentaire (facultatif), tel que des scripts spécifiques à vos besoins. Par exemple, si votre package réseau contient plusieurs packages de fonctions, vous pouvez utiliser le NSD pour définir les fonctions réseau qui doivent être exécutées dans certains VPCs sous-réseaux ou clusters Amazon EKS.

Créez un package réseau après avoir créé des packages de fonctions. Une fois que vous avez créé un package réseau, vous devez créer une instance réseau.

Console

Pour créer un package réseau à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, sélectionnez Network packages.
3. Choisissez Créer un package réseau.
4. Choisissez Choisir des fichiers et chargez chaque NSD sous forme de `.zip` fichier. Vous pouvez télécharger un maximum de 10 fichiers.
5. Choisissez Suivant.
6. Vérifiez les détails du package.
7. Choisissez Créer un package réseau.

AWS CLI

Pour créer un package réseau à l'aide du AWS CLI

1. Utilisez la create-sol-network-packagecommande pour créer un package réseau.

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

2. Utilisez la commande put-sol-network-package-content pour télécharger le contenu du package réseau. Par exemple :

```
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
```

Afficher un package réseau dans AWS TNB

Découvrez comment afficher le contenu d'un package réseau.

Console

Pour afficher un package réseau à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, sélectionnez Network packages.
3. Utilisez le champ de recherche pour trouver le package réseau.

AWS CLI

Pour consulter un package réseau à l'aide du AWS CLI

1. Utilisez la list-sol-network-packagescommande pour répertorier vos packages réseau.

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

2. Utilisez la get-sol-network-packagecommande pour afficher les détails d'un package réseau.

```
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
```

Téléchargez un package réseau auprès de AWS TNB

Découvrez comment télécharger un package réseau à partir du catalogue de services réseau AWS TNB.

Console

Pour télécharger un package réseau à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, sélectionnez Network packages.
3. Utilisez le champ de recherche pour trouver le package réseau
4. Choisissez le package réseau.
5. Choisissez Actions, puis Télécharger.

AWS CLI

Pour télécharger un package réseau à l'aide du AWS CLI

- Utilisez la commande get-sol-network-package-content pour télécharger un package réseau.

```
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
```

Supprimer un package réseau de AWS TNB

Découvrez comment supprimer un package réseau du catalogue de services réseau AWS TNB. Pour supprimer un package réseau, celui-ci doit être dans un état désactivé.

Console

Pour supprimer un package réseau à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, sélectionnez Network packages.
3. Utilisez le champ de recherche pour trouver le package réseau
4. Choisissez un package réseau
5. Choisissez Actions, Désactiver .
6. Sélectionnez Actions, Supprimer.

AWS CLI

Pour supprimer un package réseau à l'aide du AWS CLI

1. Utilisez la update-sol-network-packagecommande pour désactiver un package réseau.

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

2. Utilisez la delete-sol-network-packagecommande pour supprimer un package réseau.

```
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
```

Instances réseau pour AWS TNB

Une instance réseau est un réseau unique créé dans AWS TNB qui peut être déployé.

Tâches

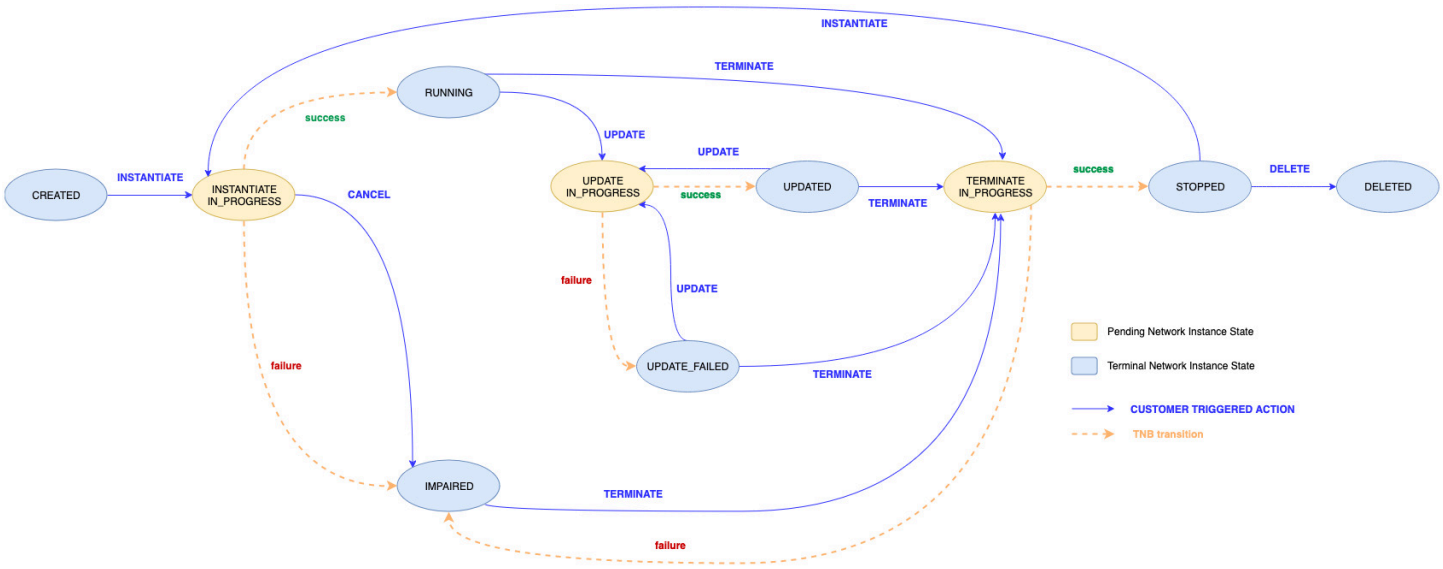
- Opérations du cycle de vie d'une instance réseau
- Création d'une instance réseau à l'aide de AWS TNB
- Instancier une instance réseau à l'aide de TNB AWS
- Mettre à jour une instance de fonction dans AWS TNB
- Mettre à jour une instance réseau dans AWS TNB
- Afficher une instance réseau dans AWS TNB
- Mettre fin à une instance réseau et la supprimer de AWS TNB

Opérations du cycle de vie d'une instance réseau

AWS TNB vous permet de gérer facilement votre réseau à l'aide d'opérations de gestion standardisées conformément à SOL003 ETSI et. SOL005 Vous pouvez effectuer les opérations de cycle de vie suivantes :

- Créez le réseau
- Instancier le réseau
- Mettre à jour la fonction réseau
- Mettre à jour l'instance réseau
- Afficher les détails et l'état du réseau
- Mettre fin au réseau

L'image suivante montre les opérations de gestion du réseau :



Création d'une instance réseau à l'aide de AWS TNB

Vous créez une instance réseau après avoir créé un package réseau. Après avoir créé une instance réseau, instanciez-la.

Console

Pour créer une instance réseau à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, sélectionnez Networks.
3. Choisissez Create network instance.
4. Entrez un nom et une description pour l'instance, puis choisissez Next.
5. Sélectionnez le package réseau, vérifiez les informations, puis choisissez Next.
6. Choisissez Create network instance.

   La nouvelle instance réseau apparaît sur la page Réseaux. Ensuite, instanciez cette instance réseau.

AWS CLI

Pour créer une instance réseau à l'aide du AWS CLI

- Utilisez la create-sol-network-instancecommande pour créer une instance réseau.

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

Ensuite, instanciez cette instance réseau.

Instancier une instance réseau à l'aide de TNB AWS

Après avoir créé une instance réseau, vous devez l'instancier. Lorsque vous instanciez une instance réseau, AWS TNB fournit l' AWS infrastructure nécessaire, déploie des fonctions réseau conteneurisées et configure le réseau et la gestion des accès pour créer un service réseau entièrement opérationnel.

Console

Pour instancier une instance réseau à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, sélectionnez Networks.
3. Sélectionnez l'instance réseau que vous souhaitez instancier.
4. Choisissez Actions, puis Instancier.
5. Sur la page Instancier le réseau, passez en revue les détails et, éventuellement, mettez à jour les valeurs des paramètres.

   Les mises à jour des valeurs des paramètres s'appliquent uniquement à cette instance réseau. Les paramètres des packages NSD et VNFD ne changent pas.
6. Choisissez Instancier le réseau.

   La page État du déploiement apparaît.
7. Utilisez l'icône Actualiser pour suivre l'état de déploiement de votre instance réseau. Vous pouvez également activer l'actualisation automatique dans la section Tâches de déploiement pour suivre la progression de chaque tâche.

   Lorsque l'état du déploiement passe à`Completed`, l'instance réseau est instanciée.

AWS CLI

Pour instancier une instance réseau à l'aide du AWS CLI

1. Utilisez la instantiate-sol-network-instancecommande pour instancier l'instance réseau.

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

2. Ensuite, consultez l'état de fonctionnement du réseau.

Mettre à jour une instance de fonction dans AWS TNB

Après l'instanciation d'une instance réseau, vous pouvez mettre à jour un package de fonctions dans l'instance réseau.

Console

Pour mettre à jour une instance de fonction à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, sélectionnez Networks.
3. Sélectionnez l'instance réseau. Vous ne pouvez mettre à jour une instance réseau que si son état est`Instantiated`.

   La page de l'instance réseau apparaît.
4. Dans l'onglet Fonctions, sélectionnez l'instance de fonction à mettre à jour.
5. Choisissez Mettre à jour.
6. Entrez vos annulations de mise à jour.
7. Choisissez Mettre à jour.

AWS CLI

Utiliser la CLI pour mettre à jour une instance de fonction

Utilisez la update-sol-network-instancecommande avec le type de `MODIFY_VNF_INFORMATION` mise à jour pour mettre à jour une instance de fonction dans une instance réseau.

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

Mettre à jour une instance réseau dans AWS TNB

Après l'instanciation d'une instance réseau, vous devrez peut-être mettre à jour l'infrastructure ou l'application. Pour ce faire, vous devez mettre à jour le package réseau et les valeurs des paramètres de l'instance réseau et déployer l'opération de mise à jour pour appliquer les modifications.

Considérations

- Vous pouvez mettre à jour une instance réseau à l'`Updated`état `Instantiated` ou.
- Lorsque vous mettez à jour une instance réseau, l'`UpdateSolNetworkService`API utilise le nouveau package réseau et les nouvelles valeurs de paramètres pour mettre à jour la topologie de l'instance réseau.
- AWS TNB vérifie que le nombre de paramètres NSD et VNFD dans l'instance réseau ne dépasse pas 200. Cette limite est appliquée pour empêcher les acteurs malveillants de transmettre des charges utiles erronées ou énormes qui affectent le service.

Paramètres que vous pouvez mettre à jour

Vous pouvez mettre à jour les paramètres suivants lorsque vous mettez à jour une instance réseau instanciée :

Paramètre	Description	Exemple : Avant	Exem Après

Version du cluster Amazon EKS	Vous pouvez mettre à jour la valeur du `version` paramètre du plan de contrôle du cluster Amazon EKS vers la version mineure suivante. Vous ne pouvez pas rétrograder la version.	`EKSCluster:`
		`  type: tosca.nod`
		`es.AWS.Compute.EKS`
		`  properties:`
		`  version: "1.28"`
			`EKSC`
			`r:`
			
			`  typ`
			`  tos`
			`es.A`
			`mput`

Paramètre	Description	Exemple : Avant	Exem Après
			pro s: ver "1.

Paramètre	Description	Exemple : Avant	Exem Après

Nœuds de travail Amazon EKS	Vous pouvez mettre à jour la valeur du `EKSManagedNode` `kubernetes_version` paramètre pour mettre à niveau votre groupe de nœuds vers une version plus récente d'Amazon EKS, ou vous pouvez mettre à jour le `ami_id` paramètre pour mettre à niveau votre groupe de nœuds vers la dernière AMI optimisée pour EKS.
	
	Vous pouvez mettre à jour l'ID AMI pour`EKSSelfMa` `nagedNode` . La version Amazon EKS de l'AMI doit être identique ou inférieure à 2 versions au maximum à la version du cluster Amazon EKS. Par exemple, si la version du cluster Amazon EKS est 1.31, la version de l'AMI Amazon EKS doit être 1.31, 1.30 ou 1.29.
		`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`

Paramètre	Description	Exemple : Avant	Exem Après
			com pro s: ami "am 3NEW

Paramètre	Description	Exemple : Avant	Exem Après

Groupes de nœuds Amazon EKS	Vous pouvez ajouter ou supprimer des groupes de nœuds en fonction de vos besoins informatiques.
	
	Lorsque vous supprimez des groupes de nœuds existants et que vous en ajoutez de nouveaux, assurez-vous que les nouveaux groupes de nœuds sont différents IDs des groupes de nœuds supprimés , sinon l'opération sera traitée comme une modification de groupe de nœuds plutôt que comme une suppression et un ajout. Notez que pour les groupes de nœuds existants , seul un ensemble limité de paramètres peut être mis à jour. Parcourez ce tableau pour voir quels paramètres vous pouvez mettre à jour.
		`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`

Paramètre	Description	Exemple : Avant	Exem Après
			mir 1 max 1 ... Free SNod # New Nod typ tos es.A mput Self edNo ... sca pro s:

Paramètre	Description	Exemple : Avant	Exem Après
			des ize: 1 mir 1 max 1 ... *Free* *SNod* # New Nod typ tos es.A mput Mana de

Paramètre	Description	Exemple : Avant	Exem Après
			... sca pro s: des ize: 1 mir 1 max 1

Paramètre	Description	Exemple : Avant	Exem Après
			...

Paramètre	Description	Exemple : Avant	Exem Après
Propriétés de dimension nement	Vous pouvez mettre à jour les propriétés de mise à l'échelle des EKSManagedNode nœuds et EKSSelfMa nagedNode TOSCA.	EKSNodeGroup01: ... scaling: properties: desired_s ize: 1 min_size: 1 max_size: 1	EKSN oup0 ... sca pro s: des ize:

Paramètre	Description	Exemple : Avant	Exem Après
			mir max

Paramètre	Description	Exemple : Avant	Exem Après

Propriétés du plugin Amazon EBS CSI	Vous pouvez activer ou désactiver le plug-in Amazon EBS CSI sur vos clusters Amazon EKS. Vous pouvez également modifier la version du plugin.	`EKSCluster:`
		`  capabilities:`
		`  ...`
		`  ebs_csi:`
		`  properties:`
		`  enabled: false`
			`EKSC`
			`r:`
			`cap`
			`ies:`
			`...`
			`ebs`
			`pro`
			`s:`
			`ena`
			`ver`
			`"v1`
			`e`
			`ksbu`
			`"`

Paramètre	Description	Exemple : Avant	Exem Après

Taille du volume racine	Vous pouvez ajouter, supprimer ou mettre à jour la propriété de taille du volume racine des EKSManaged nœuds Node et EKSSelf ManagedNode TOSCA.	`Free5GCEKSNode01:`
		`  ...`
		`  capabilities:`
		
		`  compute:`
		`  properties:`
		`  root_volu`
		`me_size: 50`
			Free
			SNod
			
			...
			
			cap
			ies:
			
			com
			
			pro
			s:

Paramètre	Description	Exemple : Avant	Exem Après
			roo me_s

Paramètre	Description	Exemple : Avant	Exem Après
VNF	Vous pouvez les référence r VNFs dans le NSD et les déployer sur le cluster créé dans le NSD à l'aide du nœud VNFDeployment TOSCA. Dans le cadre de la mise à jour, vous serez en mesure d'ajouter, de mettre à jour et de VNFs supprimer des informations sur le réseau.	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

Paramètre	Description	Exemple : Avant	Exem Après
			elmD : typ tos es.A ploy VNFD ment rec nts: clu EKS r vnf

Paramètre	Description	Exemple : Avant	Exem Après
			- v leVN - v leVN

Paramètre	Description	Exemple : Avant	Exem Après

Crochets	Pour exécuter des opérations de cycle de vie avant et après la création d'une fonction réseau, ajoutez les `post_create` crochets `pre_create` et au `VNFDeployment` nœud.
	
	Dans cet exemple, le `PreCreateHook` hook s'exécutera avant d'être instancié et le `PostCreateHook` hook `vnf3.SampleVNF3` s'exécutera après `vnf3.SampleVNF3` l'instanciation.
		`vnfds:`
		`  - descriptor_id: "43c012fa-2616-41a8-a833-0dfd4c5a049e"`
		`    namespace: "vnf1"`
		`  - descriptor_id: "64222f98-ecd6-4871-bf94-7354b53f3ee5"`
		`    namespace: "vnf2"`
		`...`
		`SampleVNF1HelmDeploy:`
		`  type: tosca.nodes.AWS.Deployment.VNFDeployment`
		`  requirements:`
		`    cluster: EKSCluster`
		`    vnfs:`
		`      - vnf1.SampleVNF1`
		`      - vnf2.SampleVNF2  // Removed during update`
			`vnfd`
			` -`
			` des`
			`r_id`
			` "43`
			`2616`
			`-`
			`a833`
			`d4c5`
			` "`
			` nam`
			`:`
			` "vn`
			` -`
			` des`
			`r_id`
			` "b7`
			`839d`
			`-916`
			`a166`
			` "`
			` nam`
			`:`
			` "vn`
			`....`
			`S`
			`ampl`
			`Helm`
			`y:`
			` typ`
			` tos`

Paramètre	Description	Exemple : Avant	Exem Après
			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

Paramètre	Description	Exemple : Avant	Exem Après
			- v leVN New VNF as the nam , vnf was not pre y pre int s: Hoo pos te: eHoo pre e: Hook

Paramètre	Description	Exemple : Avant	Exem Après

Crochets	Pour exécuter des opération s de cycle de vie avant et après la mise à jour d'une fonction réseau, vous pouvez ajouter le `pre_update` `post_update` hook et le hook au `VNFDeployment` nœud.
	
	Dans cet exemple, `PreUpdateHook` sera exécuté avant `vnf1.Samp leVNF1` la mise à jour et s'`PostUpdat eHook` exécutera après `vnf1.SampleVNF1` la mise à jour vers le `vnf` package indiqué par la mise à jour `uuid` pour l'espace de noms vnf1.
		`vnfds:`
		`  - descriptor_id:`
		`"43c012fa-2616-41a8-a833-0dfd4c5a049e"`
		`    namespace: "vnf1"`
		`  - descriptor_id:`
		`"64222f98-ecd6-4871-bf94-7354b53f3ee5"`
		`    namespace: "vnf2"`
		`...`
		
		`SampleVNF1HelmDeploy:`
		`  type: tosca.nod es.AWS.Deployment. VNFDeployment`
		`  requirements:`
		`    cluster: EKSCluster`
		`    vnfs:`
		`      - vnf1.SampleVNF1`
		`      - vnf2.Samp leVNF2`
			`vnfd`
			
			`-`
			`des`
			`r_id`
			`"0e`
			`bd87`
			`-`
			`b8a1`
			`4666`
			`"`
			
			`nam`
			`:`
			`"vn`
			
			`-`
			`des`
			`r_id`
			`"64`
			`ecd6`
			`-`
			`bf94`
			`4b53`
			`"`
			
			`nam`
			`:`
			`"vn`
			`...`
			`S`
			`ampl`
			`Helm`
			`y:`
			
			`typ`

Paramètre	Description	Exemple : Avant	Exem Après
			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

Paramètre	Description	Exemple : Avant	Exem Après
			leVN No cha to thi fur as nam and uui rem the sam int s: Hoo pre e: Hook pos te: eHoo

Paramètre	Description	Exemple : Avant	Exem Après

Subnets	Vous pouvez ajouter et supprimer des sous-rése aux sur le réseau. Avant de supprimer un sous-réseau, vérifiez qu'il n'est utilisé par aucune ressource du réseau.	`Free5GCSubnet01 :`
		`#Deleted Subnet`
		`type: tosca.nod es.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:`

Paramètre	Description	Exemple : Avant	Exem Après
			rou le: Fre uteT vpc Fre C

Paramètre	Description	Exemple : Avant	Exem Après
Groupes de sécurité	Vous pouvez ajouter et supprimer des groupes de sécurité sur le réseau. Avant de supprimer un groupe de sécurité, vérifiez qu'il n'est utilisé par aucune ressource du réseau.	*Free5GCSecurityGroup01* : #Deleted Security Group type: tosca.nodes.AWS.Networking.SecurityGroup properties: description: "SecurityGroup for Free5GC cluster" name: "Free5GCSecurityGroup01" tags: - "Name=Free5GCEKSAdditionalSecurityGroup" requirements: vpc: Free5GCVPC *Free5GCSecurityGroupEgressRule01* : #Deleted Security Group Egress Node type: tosca.nodes.AWS.Networking.SecurityGroupEgressRule properties: ip_protocol: "tcp" from_port: 8000 to_port: 9000 description: "Egress Rule for free5GC cluster" cidr_ip : "172.10.10.1/24" requirements: security_group: Free5GCSecurityGroup01	*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

Paramètre	Description	Exemple : Avant	Exem Après
		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	- "Name5GCditiecuroup" req nts: vpc Free5GC *Free curiupEgule0* #Ne Sec Gro Egr Noc typ tos es.A twor Secu roup sRul pro s:

Paramètre	Description	Exemple : Avant	Exem Après
			ip_ ol: "to fro : 800 to_ 900 des on: "Eg Rul for fre clu cid "17 0.1/ rec nts: sec grou Fre

Paramètre	Description	Exemple : Avant	Exem Après

```
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
```

Paramètre	Description	Exemple : Avant	Exem Après
			des on: "Ir Rul for fre clu cid "17 0.1/ rec nts: sec grou Fre curi up02

Paramètre	Description	Exemple : Avant	Exem Après

Interfaces réseau	Vous pouvez ajouter, modifier et supprimer ENIs du réseau.	`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`

Paramètre	Description	Exemple : Avant	Exem Après
			curi up01 Fre e5GC : #Mc ENI typ tos es.A twor ENI pro s: dev dex: 3 sou st_c tru rec nts: sub Fre ISub se grou

Paramètre	Description	Exemple : Avant	Exem Après
			- 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

Paramètre	Description	Exemple : Avant	Exem Après
			- Fre curi up01

Mettre à jour une instance réseau

Console

Pour mettre à jour une instance réseau à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, sélectionnez Networks.
3. Sélectionnez l'instance réseau. Vous ne pouvez mettre à jour une instance réseau que si son état est `Instantiated` ou`Updated`.
4. Choisissez Actions et Mettre à jour.

   La page Mettre à jour l'instance apparaît avec les détails du réseau et une liste des paramètres de l'infrastructure actuelle.
5. Choisissez un nouveau package réseau.

   Les paramètres du nouveau package réseau apparaissent dans la section Paramètres mis à jour.
6. Vous pouvez éventuellement mettre à jour les valeurs des paramètres dans la section Paramètres mis à jour. Pour la liste des valeurs de paramètres que vous pouvez mettre à jour, consultezParamètres que vous pouvez mettre à jour.
7. Choisissez Mettre à jour le réseau.

   AWS TNB valide la demande et lance le déploiement. La page État du déploiement apparaît.

8. Utilisez l'icône Actualiser pour suivre l'état de déploiement de votre instance réseau. Vous pouvez également activer l'actualisation automatique dans la section Tâches de déploiement pour suivre la progression de chaque tâche.

   Lorsque l'état du déploiement passe à`Completed`, l'instance réseau est mise à jour.

9.
   - Si la validation échoue, l'instance réseau reste dans le même état qu'avant que vous ne demandiez la mise à jour, `Instantiated` soit`Updated`.
   - Si la mise à jour échoue, l'état de l'instance réseau s'affiche`Update failed`. Choisissez le lien correspondant à chaque tâche ayant échoué pour en déterminer la raison.
   - Si la mise à jour réussit, l'état de l'instance réseau s'affiche`Updated`.

AWS CLI

Utiliser la CLI pour mettre à jour une instance réseau

Utilisez la update-sol-network-instancecommande avec le type de `UPDATE_NS` mise à jour pour mettre à jour une instance réseau.

```
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\"}}"
```

Afficher une instance réseau dans AWS TNB

Découvrez comment afficher une instance réseau.

Console

Pour afficher une instance réseau à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, sélectionnez Network instances.
3. Utilisez le champ de recherche pour trouver l'instance réseau.

AWS CLI

Pour afficher une instance réseau à l'aide du AWS CLI

1. Utilisez la list-sol-network-instancescommande pour répertorier vos instances réseau.

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

2. Utilisez la get-sol-network-instancecommande pour afficher les détails d'une instance réseau spécifique.

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

Mettre fin à une instance réseau et la supprimer de AWS TNB

Pour supprimer une instance réseau, celle-ci doit être dans un état terminé.

Console

Pour mettre fin à une instance réseau et la supprimer à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, sélectionnez Networks.
3. Sélectionnez l'ID de l'instance réseau.
4. Sélectionnez Résilier.
5. Lorsque vous êtes invité à confirmer, entrez l'ID et choisissez Terminate.
6. Actualisez pour suivre l'état de votre instance réseau.
7. (Facultatif) Sélectionnez l'instance réseau et choisissez Supprimer.

AWS CLI

Pour mettre fin à une instance réseau et la supprimer à l'aide du AWS CLI

1. Utilisez la terminate-sol-network-instancecommande pour mettre fin à une instance réseau.

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

2. (Facultatif) Utilisez la delete-sol-network-instancecommande pour supprimer une instance réseau.

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

Opérations réseau pour AWS TNB

Une opération réseau est toute opération effectuée sur votre réseau, telle que l'instanciation ou la terminaison d'une instance réseau.

Tâches

- Afficher le fonctionnement d'un réseau AWS TNB
- Annuler une opération AWS du réseau TNB

Afficher le fonctionnement d'un réseau AWS TNB

Affichez les détails d'une opération réseau, y compris les tâches impliquées dans le fonctionnement du réseau et l'état des tâches.

Console

Pour afficher une opération réseau à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, sélectionnez Network instances.
3. Utilisez le champ de recherche pour trouver l'instance réseau.
4. Dans l'onglet Déploiements, choisissez le fonctionnement du réseau.

AWS CLI

Pour visualiser une opération réseau à l'aide du AWS CLI

1. Utilisez la list-sol-network-operationscommande pour répertorier toutes les opérations réseau.

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

2. Utilisez la get-sol-network-operationcommande pour afficher les détails d'une opération réseau.

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

Annuler une opération AWS du réseau TNB

Découvrez comment annuler une opération réseau.

Console

Pour annuler une opération réseau à l'aide de la console

1. Ouvrez la console AWS TNB à https://console.aws.amazon.com/tnb/l'adresse.
2. Dans le volet de navigation, sélectionnez Networks.
3. Sélectionnez l'ID du réseau pour ouvrir sa page de détails.
4. Dans l'onglet Déploiements, choisissez le fonctionnement du réseau.
5. Choisissez Annuler l'opération.

AWS CLI

Pour annuler une opération réseau à l'aide du AWS CLI

Utilisez la cancel-sol-network-operationcommande pour annuler une opération réseau.

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

Référence TOSCA pour TNB AWS

La spécification de topologie et d'orchestration pour les applications cloud (TOSCA) est une syntaxe déclarative CSPs utilisée pour décrire une topologie de services Web basés sur le cloud, leurs composants, leurs relations et les processus qui les gèrent. CSPs décrire les points de connexion, les liens logiques entre les points de connexion et les politiques telles que l'affinité et la sécurité dans un modèle TOSCA. CSPs puis téléchargez le modèle sur AWS TNB qui synthétise les ressources nécessaires pour établir un réseau 5G fonctionnel dans les zones de AWS disponibilité.

Table des matières

Modèle VNFD

Définit un modèle de descripteur de fonction réseau virtuel (VNFD).

Syntaxe

```
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
```

Modèle de topologie

`node_templates`

Les nœuds TOSCA AWS . Les nœuds possibles sont les suivants :

- AWS.VNF
- AWS.Artefacts. Casque

AWS.VNF

Définit un AWS nœud de fonction réseau virtuelle (VNF).

Syntaxe

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

Propriétés

`descriptor_id`

L'UUID du descripteur.

Obligatoire : oui

Type : String

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

`descriptor_version`

La version du VNFD.

Obligatoire : oui

Type : String

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

`descriptor_name`

Le nom du descripteur.

Obligatoire : oui

Type : String

`provider`

L'auteur du VNFD.

Obligatoire : oui

Type : String

Prérequis

`helm`

Le répertoire Helm définissant les artefacts du conteneur. Il s'agit d'une référence à AWS.Artifacts.Helm.

Obligatoire : oui

Type : String

exemple

```
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

Définit un nœud AWS Helm.

Syntaxe

```
tosca.nodes.AWS.Artifacts.Helm:
```

```
  properties:
    implementation: String
```

Propriétés

`implementation`

Le répertoire local qui contient le graphique Helm dans le package CSAR.

Obligatoire : oui

Type : String

exemple

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

Modèle de descripteur de service réseau

Définit un modèle de descripteur de service réseau (NSD).

Syntaxe

```
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
```

Utilisation de paramètres définis

Lorsque vous souhaitez transmettre dynamiquement un paramètre, tel que le bloc CIDR pour le nœud VPC, vous pouvez utiliser `{ get_input: input-parameter-name }` la syntaxe et définir les paramètres dans le modèle NSD. Réutilisez ensuite le paramètre dans le même modèle NSD.

L'exemple suivant montre comment définir et utiliser des paramètres :

```
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 }
```

Importation VNFD

`descriptor_id`

L'UUID du descripteur.

Obligatoire : oui

Type : String

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

`namespace`

Le nom unique.

Obligatoire : oui

Type : String

Modèle de topologie

`node_templates`

Les AWS nœuds TOSCA possibles sont les suivants :

- AWS N.S.
- AWS.Compute.EKS
- AWS.Compute.EKS.AuthRole
- AWS.Compute.EKSManagedNode
- AWS.Compute.EKSSelfManagedNode
- AWS.Compute.PlacementGroup
- AWS.Compute.UserData
- AWS.Networking.SecurityGroup
- AWS.Networking.SecurityGroupEgressRule
- AWS.Networking.SecurityGroupIngressRule
- AWS.Resource.Import
- AWS.Networking.ENI
- AWS.HookExecution
- AWS.Networking.InternetGateway
- AWS.Networking.RouteTable
- AWS.Networking.Subnet
- AWS.Deployment.VNFDeployment

- AWS.Networking.VPC
- AWS.Networking.NATGateway
- AWS.Networking.Route

AWS.NS

Définit un nœud de service AWS réseau (NS).

Syntaxe

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

Propriétés

`descriptor_id`

L'UUID du descripteur.

Obligatoire : oui

Type : String

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

`descriptor_version`

Version du NSD.

Obligatoire : oui

Type : String

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

`descriptor_name`

Nom du descripteur.

Obligatoire : oui

Type : String

Exemple

```
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

Indiquez le nom du cluster, la version de Kubernetes souhaitée et un rôle permettant au plan de contrôle Kubernetes de gérer les ressources requises pour vos NF. AWS Les plugins CNI (Multus Container Network Interface) sont activés. Vous pouvez connecter plusieurs interfaces réseau et appliquer une configuration réseau avancée aux fonctions du Kubernetes-based réseau. Vous spécifiez également l'accès au point de terminaison du cluster et les sous-réseaux de votre cluster.

Syntaxe

```
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
```

Fonctionnalités

`multus`

Facultatif. Propriétés qui définissent l'utilisation de l'interface réseau de conteneurs (CNI) Multus.

Si vous incluez`multus`, spécifiez les `multus_role` propriétés `enabled` et.

`enabled`

Indique si la fonctionnalité Multus par défaut est activée.

Obligatoire : oui

Type : Boolean

`multus_role`

Le rôle de la gestion de l'interface réseau Multus.

Obligatoire : oui

Type : String

`ebs_csi`

Propriétés qui définissent le pilote Amazon EBS Container Storage Interface (CSI) installé dans le cluster Amazon EKS.

Activez ce plugin pour utiliser les nœuds autogérés Amazon EKS sur AWS Outposts les Zones AWS Locales ou Régions AWS. Pour plus d'informations, consultez le pilote Amazon Elastic Block Store CSI dans le guide de l'utilisateur Amazon EKS.

`enabled`

Indique si le pilote Amazon EBS CSI par défaut est installé.

Obligatoire : non

Type : Boolean

`version`

Version du module complémentaire de pilote Amazon EBS CSI. La version doit correspondre à l'une des versions renvoyées par l'DescribeAddonVersionsaction. Pour plus d'informations, consultez DescribeAddonVersionsle manuel Amazon EKS API Reference

Obligatoire : non

Type : Chaîne

Propriétés

`version`

Version de Kubernetes pour le cluster. AWS Telco Network Builder prend en charge les versions 1.27 à 1.34 de Kubernetes.

Obligatoire : oui

Type : String

Valeurs possibles : 1,27 | 1,28 | 1,29 | 1,30 | 1,31 | 1,32 | 1,33 | 1,34

`access`

L'accès au point de terminaison du cluster.

Obligatoire : oui

Type : String

Valeurs possibles : `PRIVATE` | `PUBLIC` | `ALL`

`cluster_role`

Le rôle de la gestion des clusters.

Obligatoire : oui

Type : String

`tags`

Balises à associer à la ressource.

Obligatoire : non

Type : liste

`ip_family`

Indique la famille d'adresses IP pour les adresses de service et de pod dans le cluster.

Valeur autorisée :`IPv4`, `IPv6`

Valeur par défaut : `IPv4`

Obligatoire : non

Type : Chaîne

Exigences

`subnets`

Un AWS. Networking.Subnetnœud.

Obligatoire : oui

Type : liste

Exemple

```
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 vous AuthRole permet d'ajouter des rôles IAM au cluster Amazon EKS `aws-auth` `ConfigMap` afin que les utilisateurs puissent accéder au cluster Amazon EKS à l'aide d'un rôle IAM.

Syntaxe

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

Propriétés

`role_mappings`

Liste des mappages qui définissent les rôles IAM qui doivent être ajoutés au cluster Amazon EKS. `aws-auth` `ConfigMap`

`arn`

ARN du rôle IAM.

Obligatoire : oui

Type : String

`groups`

Groupes Kubernetes à attribuer au rôle défini dans. `arn`

Obligatoire : non

Type : liste

Exigences

`clusters`

Un AWS. Compute.EKSnœud.

Obligatoire : oui

Type : liste

Exemple

```
EKSAuthMapRoles:
    type: tosca.nodes.AWS.Compute.EKS.AuthRole
    properties:
        role_mappings:
        - arn: arn:aws:iam::${AWS::TNB::AccountId}:role/TNBHookRole1
          groups:
          - system:nodes
          - system:bootstrappers
        - arn: arn:aws:iam::${AWS::TNB::AccountId}:role/TNBHookRole2
          groups:
          - system:nodes
          - system:bootstrappers
    requirements:
         clusters:
         - Free5GCEKS1
         - Free5GCEKS2
```

AWS.Compute.EKSManagedNode

AWS TNB prend en charge les groupes de nœuds gérés par EKS pour automatiser le provisionnement et la gestion du cycle de vie des nœuds (instances Amazon EC2) pour les clusters Amazon EKS Kubernetes. Pour créer un groupe de nœuds EKS, procédez comme suit :

- Choisissez les Amazon Machine Images (AMI) pour les nœuds de travail de votre cluster en fournissant soit l'ID de l'AMI, soit le type d'AMI.
- Fournissez une paire de clés Amazon EC2 pour l'accès SSH et les propriétés de dimensionnement de votre groupe de nœuds.
- Assurez-vous que votre groupe de nœuds est associé à un cluster Amazon EKS.

- Fournissez les sous-réseaux pour les nœuds de travail.
- Vous pouvez éventuellement associer des groupes de sécurité, des étiquettes de nœuds et un groupe de placement à votre groupe de nœuds.

Syntaxe

```
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
```

Fonctionnalités

`Calcul`

Propriétés qui définissent les paramètres informatiques du groupe de nœuds gérés par Amazon EKS, tels que les types d'instances Amazon EC2 et les AMI d'instance Amazon EC2.

`ami_type`

Type d' EKS-supported AMI Amazon.

Obligatoire : oui

Type : String

Valeurs possibles : `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`

ID de l'AMI.

Obligatoire : non

Type : Chaîne

> **Note**
>
> Si `ami_type` les deux `ami_id` sont spécifiés dans le modèle, AWS TNB n'utilisera que la `ami_id` valeur pour créer`EKSManagedNode`.

`instance_types`

Taille de l'instance.

Obligatoire : oui

Type : liste

`key_pair`

La paire de clés EC2 pour activer l'accès SSH.

Obligatoire : oui

Type : String

`root_volume_encryption`

Active le chiffrement Amazon EBS pour le volume racine Amazon EBS. Si cette propriété n'est pas fournie, AWS TNB chiffre les volumes racine Amazon EBS par défaut.

Obligatoire : non

Valeur par défaut : true

Type : Boolean

`root_volume_encryption_key_arn`

L'ARN de la AWS KMS clé. AWS TNB prend en charge l'ARN clé standard, l'ARN clé multirégional et l'ARN alias.

Obligatoire : non

Type : Chaîne

> **Note**
>
> - Si `root_volume_encryption` c'est faux, ne l'incluez pas`root_volume_encryption_key_arn`.
> - AWS TNB prend en charge le chiffrement du volume racine des EBS-backed AMI Amazon.
> - Si le volume racine de l'AMI est déjà chiffré, vous devez inclure le AWS TNB `root_volume_encryption_key_arn` pour le rechiffrer.
> - Si le volume racine de l'AMI n'est pas chiffré, AWS TNB utilise le `root_volume_encryption_key_arn` pour chiffrer le volume racine.
>
> Si vous ne l'incluez pas`root_volume_encryption_key_arn`, AWS TNB utilise la clé par défaut fournie par AWS Key Management Service pour chiffrer le volume racine.
> - AWS TNB ne déchiffre pas une AMI chiffrée.

`root_volume_size`

Taille du volume racine d'Amazon Elastic Block Store en GiBs.

Obligatoire : non

Valeur par défaut : 20

Type : Integer

Valeurs possibles : 1 à 16 384

`mise à l’échelle`

Propriétés qui définissent les paramètres de dimensionnement pour le groupe de nœuds géré par Amazon EKS, tels que le nombre souhaité d'instances Amazon EC2 et le nombre minimum et maximum d'instances Amazon EC2 dans le groupe de nœuds.

`desired_size`

Le nombre d'instances qu'il contient NodeGroup.

Obligatoire : oui

Type : Integer

`min_size`

Le nombre minimum d'instances dans ce cas NodeGroup.

Obligatoire : oui

Type : Integer

`max_size`

Le nombre maximum d'instances dans ce cas NodeGroup.

Obligatoire : oui

Type : Integer

Propriétés

`node_role`

L'ARN du rôle IAM attaché à l'instance Amazon EC2.

Obligatoire : oui

Type : String

`tags`

Les balises à associer à la ressource.

Obligatoire : non

Type : liste

`kubernetes_version`

Version Kubernetes pour le groupe Managed Node. AWS TNB prend en charge les versions 1.27 à 1.34 de Kubernetes. Éléments à prendre en compte :

- Spécifiez soit le `kubernetes_version` soit`ami_id`. Ne spécifiez pas les deux.
- Le `kubernetes_version` doit être inférieur ou égal au AWS. Compute.EKSManagedNode version.
- Il peut y avoir une différence de 3 versions entre les AWS. Compute.EKSManagedNode version et`kubernetes_version`.
- Si aucun des `kubernetes_version` ou `ami_id` n'est spécifié, AWS TNB utilisera la dernière AMI de la `AWS.Compute.EKSManagedNode` version pour créer `EKSManagedNode`

Obligatoire : non

Type : Chaîne

Valeurs possibles : 1,27 | 1,28 | 1,29 | 1,30 | 1,31 | 1,32 | 1,33 | 1,34

Exigences

`cluster`

Un AWS. Compute.EKSnœud.

Obligatoire : oui

Type : String

`subnets`

Un AWS. Networking.Subnetnœud.

Obligatoire : oui

Type : liste

`network_interfaces`

Un AWS. Networking.ENInœud. Assurez-vous que les interfaces réseau et les sous-réseaux sont définis sur la même zone de disponibilité, sinon l'instanciation échouera.

Lorsque vous définissez`network_interfaces`, AWS TNB obtient l'autorisation relative aux ENI auprès de la `multus_role` propriété si vous avez inclus la `multus` propriété dans le nœud. AWS.Compute.EKS Sinon, AWS TNB obtient l'autorisation relative aux ENI à partir de la propriété node_role.

Obligatoire : non

Type : liste

`security_groups`

Un AWS. Networking.SecurityGroupnœud.

Obligatoire : non

Type : liste

`placement_group`

Un tosca.nodes.AWS. Compute.PlacementGroupnœud.

Obligatoire : non

Type : Chaîne

`user_data`

Un tosca.nodes.AWS. Compute.UserDataréférence de nœud. Un script de données utilisateur est transmis aux instances Amazon EC2 lancées par le groupe de nœuds gérés. Ajoutez les autorisations requises pour exécuter des données utilisateur personnalisées au node_role transmis au groupe de nœuds.

Obligatoire : non

Type : Chaîne

`labels`

Liste des étiquettes de nœuds. L'étiquette d'un nœud doit avoir un nom et une valeur. Créez une étiquette en utilisant les critères suivants :

- Le nom et la valeur doivent être séparés par=.
- Le nom et la valeur peuvent chacun comporter jusqu'à 63 caractères.
- L'étiquette peut inclure des lettres (A-Z, a-z,), des chiffres (0-9) et les caractères suivants : `[-, _, ., *, ?]`
- Le nom et la valeur doivent commencer et se terminer par un * caractère alphanumérique ou. ?

Par exemple, `myLabelName1=*NodeLabelValue1`

Obligatoire : non

Type : liste

Exemple

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

```
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
    kubernetes_version:
      - "1.30"
  requirements:
    cluster: SampleEKS
    subnets:
      - SampleSubnet
    network_interfaces:
      - SampleENI01
      - SampleENI02
    security_groups:
      - SampleSecurityGroup01
      - SampleSecurityGroup02
    placement_group: SamplePlacementGroup
    user_data: CustomUserData
    labels:
      - "sampleLabelName001=sampleLabelValue001"
      - "sampleLabelName002=sampleLabelValue002"
```

AWS.Compute.EKSSelfManagedNode

AWS TNB prend en charge les nœuds autogérés Amazon EKS pour automatiser le provisionnement et la gestion du cycle de vie des nœuds (instances Amazon EC2) pour les clusters Amazon EKS Kubernetes. Pour créer un groupe de nœuds Amazon EKS, procédez comme suit :

- Choisissez les Amazon Machine Images (AMI) pour les nœuds de travail de votre cluster en fournissant soit l'ID de l'AMI.
- Fournissez une paire de clés Amazon EC2 pour l'accès SSH.
- Assurez-vous que votre groupe de nœuds est associé à un cluster Amazon EKS.
- Indiquez le type d'instance et les tailles souhaitées, minimales et maximales.
- Fournissez les sous-réseaux pour les nœuds de travail.
- Vous pouvez éventuellement associer des groupes de sécurité, des étiquettes de nœuds et un groupe de placement à votre groupe de nœuds.

Syntaxe

```
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
```

Fonctionnalités

Calcul

Propriétés qui définissent les paramètres de calcul pour les nœuds autogérés Amazon EKS, tels que les types d'instances Amazon EC2 et les AMI d'instance Amazon EC2.

`ami_id`

ID d'AMI utilisé pour lancer l'instance. AWS TNB prend en charge les instances qui exploitent IMDSv2. Pour de plus amples informations, veuillez consulter Version IMDS.

> **Note**
>
> Vous pouvez mettre à jour l'ID AMI pour`EKSSelfManagedNode`. La version Amazon EKS de l'AMI doit être identique ou inférieure à 2 versions au maximum à la version du cluster

Amazon EKS. Par exemple, si la version du cluster Amazon EKS est 1.31, la version de l'AMI Amazon EKS doit être 1.31, 1.30 ou 1.29.

Obligatoire : oui

Type : String

`instance_type`

Taille de l'instance.

Obligatoire : oui

Type : String

`key_pair`

La paire de clés Amazon EC2 pour activer l'accès SSH.

Obligatoire : oui

Type : String

`root_volume_encryption`

Active le chiffrement Amazon EBS pour le volume racine Amazon EBS. Si cette propriété n'est pas fournie, AWS TNB chiffre les volumes racine Amazon EBS par défaut.

Obligatoire : non

Valeur par défaut : true

Type : Boolean

`root_volume_encryption_key_arn`

L'ARN de la AWS KMS clé. AWS TNB prend en charge l'ARN clé standard, l'ARN clé multirégional et l'ARN alias.

Obligatoire : non

Type : Chaîne

> **Note**
>
> - Si `root_volume_encryption` c'est faux, ne l'incluez pas`root_volume_encryption_key_arn`.
> - AWS TNB prend en charge le chiffrement du volume racine des EBS-backed AMI Amazon.
> - Si le volume racine de l'AMI est déjà chiffré, vous devez inclure le AWS TNB `root_volume_encryption_key_arn` pour le rechiffrer.
> - Si le volume racine de l'AMI n'est pas chiffré, AWS TNB utilise le `root_volume_encryption_key_arn` pour chiffrer le volume racine.
>
> Si vous ne l'incluez pas`root_volume_encryption_key_arn`, AWS TNB l'utilise AWS Managed Services pour chiffrer le volume racine.
> - AWS TNB ne déchiffre pas une AMI chiffrée.

`root_volume_size`

Taille du volume racine d'Amazon Elastic Block Store en GiBs.

Obligatoire : non

Valeur par défaut : 20

Type : Integer

Valeurs possibles : 1 à 16 384

`mise à l'échelle`

Propriétés qui définissent les paramètres de dimensionnement pour les nœuds autogérés Amazon EKS, tels que le nombre souhaité d'instances Amazon EC2 et le nombre minimum et maximum d'instances Amazon EC2 dans le groupe de nœuds.

`desired_size`

Le nombre d'instances qu'il contient NodeGroup.

Obligatoire : oui

Type : Integer

`min_size`

Le nombre minimum d'instances dans ce cas NodeGroup.

Obligatoire : oui

Type : Integer

`max_size`

Le nombre maximum d'instances dans ce cas NodeGroup.

Obligatoire : oui

Type : Integer

Propriétés

`node_role`

L'ARN du rôle IAM attaché à l'instance Amazon EC2.

Obligatoire : oui

Type : String

`tags`

Les balises à associer à la ressource. Les balises seront propagées aux instances créées par la ressource.

Obligatoire : non

Type : liste

Exigences

`cluster`

Un AWS. Compute.EKSnœud.

Obligatoire : oui

Type : String

`subnets`

Un AWS. Networking.Subnetnœud.

Obligatoire : oui

Type : liste

`network_interfaces`

Un AWS. Networking.ENInœud. Assurez-vous que les interfaces réseau et les sous-réseaux sont définis sur la même zone de disponibilité, sinon l'instanciation échouera.

Lorsque vous définissez`network_interfaces`, AWS TNB obtient l'autorisation relative aux ENI auprès de la `multus_role` propriété si vous avez inclus la `multus` propriété dans le nœud. AWS.Compute.EKS Sinon, AWS TNB obtient l'autorisation relative aux ENI à partir de la propriété node_role.

Obligatoire : non

Type : liste

`security_groups`

Un AWS. Networking.SecurityGroupnœud.

Obligatoire : non

Type : liste

`placement_group`

Un tosca.nodes.AWS. Compute.PlacementGroupnœud.

Obligatoire : non

Type : Chaîne

`user_data`

Un tosca.nodes.AWS. Compute.UserDataréférence de nœud. Un script de données utilisateur est transmis aux instances Amazon EC2 lancées par le groupe de nœuds autogéré. Ajoutez

les autorisations requises pour exécuter des données utilisateur personnalisées au node_role transmis au groupe de nœuds.

Obligatoire : non

Type : Chaîne

`labels`

Liste des étiquettes de nœuds. L'étiquette d'un nœud doit avoir un nom et une valeur. Créez une étiquette en utilisant les critères suivants :

- Le nom et la valeur doivent être séparés par=.
- Le nom et la valeur peuvent chacun comporter jusqu'à 63 caractères.
- L'étiquette peut inclure des lettres (A-Z, a-z,), des chiffres (0-9) et les caractères suivants : `[-, _, ., *, ?]`
- Le nom et la valeur doivent commencer et se terminer par un `*` caractère alphanumérique ou. `?`

Par exemple, `myLabelName1=*NodeLabelValue1`

Obligatoire : non

Type : liste

Exemple

```
SampleEKSSelfManagedNode:
  type: tosca.nodes.AWS.Compute.EKSSelfManagedNode
  capabilities:
    compute:
      properties:
        ami_id: "ami-123123EXAMPLE"
        instance_type: "c5.large"
        key_pair: "SampleKeyPair"
        root_volume_encryption: true
        root_volume_encryption_key_arn: "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        root_volume_size: 1500
    scaling:
      properties:
        desired_size: 1
```

```
        min_size: 1
        max_size: 1
  properties:
    node_role: "arn:aws:iam::${AWS::TNB::AccountId}:role/SampleNodeRole"
    tags:
      - "Name=SampleVPC"
      - "Environment=Testing"
  requirements:
    cluster: SampleEKSCluster
    subnets:
      - SampleSubnet
    network_interfaces:
      - SampleNetworkInterface01
      - SampleNetworkInterface02
    security_groups:
      - SampleSecurityGroup01
      - SampleSecurityGroup02
    placement_group: SamplePlacementGroup
    user_data: CustomUserData
    labels:
      - "sampleLabelName001=sampleLabelValue001"
      - "sampleLabelName002=sampleLabelValue002"
```

AWS.Compute.PlacementGroup

Un PlacementGroup nœud prend en charge différentes stratégies pour placer des instances Amazon EC2.

Lorsque vous lancez une nouvelle instance Amazon EC2, le service Amazon EC2 tente de placer l'instance de telle sorte que toutes vos instances soient réparties sur le matériel sous-jacent afin de minimiser les défaillances corrélées. Vous pouvez utiliser des groupes de placement pour influencer le placement d'un groupe d'instances interdépendantes afin de répondre aux besoins de votre charge de travail.

Syntaxe

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

Propriétés

`strategy`

La stratégie à utiliser pour placer les instances Amazon EC2.

Obligatoire : oui

Type : String

Valeurs possibles : CLUSTER | PARTITION | SPREAD_HOST | SPREAD_RACK

- CLUSTER : regroupe les instances à proximité les unes des autres au sein d'une zone de disponibilité. Cette stratégies permet aux charges de travail d’atteindre les performances réseau à faible latence nécessaires à une communication de nœud à nœud étroitement couplée, typique des applications ce calcul haute performance (HPC).
- PARTITION : répartit vos instances sur des partitions logiques de telle sorte que les groupes d'instances d'une partition ne partagent pas le matériel sous-jacent avec des groupes d'instances situés dans différentes partitions. Cette stratégie est généralement utilisée par les grandes charges de travail distribuées et répliquées telles que Hadoop, Cassandra, et Kafka.
- SPREAD_RACK — place un petit groupe d'instances sur un matériel sous-jacent distinct afin de réduire les défaillances corrélées.
- SPREAD_HOST : utilisé uniquement avec les groupes de placement Outpost. Place un petit groupe d'instances sur un matériel sous-jacent distinct afin de réduire les défaillances corrélées.

`partition_count`

Nombre de partitions.

Obligatoire : obligatoire uniquement lorsque `strategy` ce paramètre est défini sur`PARTITION`.

Type : Integer

Valeurs possibles : 1 | 2 | 3 | 4 | 5 | 6 | 7

`tags`

Les balises que vous pouvez associer à la ressource du groupe de placement.

Obligatoire : non

Type : liste

Exemple

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

AWS.Compute.UserData

AWS TNB prend en charge le lancement d'instances Amazon EC2 avec des données utilisateur personnalisées, via UserData le nœud du Network Service Descriptor (NSD). Pour plus d'informations sur les données utilisateur personnalisées, consultez la section Données utilisateur et scripts shell dans le guide de l'utilisateur Amazon EC2.

Lors de l'instanciation du réseau, AWS TNB fournit l'enregistrement de l'instance Amazon EC2 au cluster via un script de données utilisateur. Lorsque des données utilisateur personnalisées sont également fournies, AWS TNB fusionne les deux scripts et les transmet en tant que script multimime à Amazon EC2. Le script de données utilisateur personnalisé est exécuté avant le script d'enregistrement Amazon EKS.

Pour utiliser des variables personnalisées dans le script de données utilisateur, ajoutez un point d'exclamation `!` après l'accolade ouverte. `{` Par exemple, pour l'utiliser `MyVariable` dans le script, entrez : `{!MyVariable}`

> **Note**
>
> - AWS TNB prend en charge les scripts de données utilisateur d'une taille maximale de 7 Ko.
> - AWS TNB étant utilisé CloudFormation pour traiter et afficher le script de `multimime` données utilisateur, assurez-vous que le script respecte toutes les règles. CloudFormation

Syntaxe

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

```
    content_type: String
```

Propriétés

`implementation`

Le chemin relatif vers la définition du script de données utilisateur. Le format doit être le suivant : `./scripts/script_name.sh`

Obligatoire : oui

Type : String

`content_type`

Type de contenu du script de données utilisateur.

Obligatoire : oui

Type : String

Valeurs possibles : `x-shellscript`

Exemple

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

AWS.Networking.SecurityGroup

AWS TNB prend en charge les groupes de sécurité pour automatiser le provisionnement des groupes de sécurité Amazon EC2 que vous pouvez associer aux groupes de nœuds du cluster Amazon EKS Kubernetes.

Syntaxe

```
tosca.nodes.AWS.Networking.SecurityGroup
  properties:
    description: String
```

```
    name: String
    tags: List
  requirements:
    vpc: String
```

Propriétés

`description`

Description du groupe de sécurité. Vous pouvez utiliser jusqu'à 255 caractères pour décrire le groupe. Vous ne pouvez inclure que des lettres (A-Z et a-z), des chiffres (0-9), des espaces et les caractères spéciaux suivants : ._- :/() #, @ [] +=& ; {} ! $*

Obligatoire : oui

Type : String

`name`

Nom du groupe de sécurité. Vous pouvez utiliser jusqu'à 255 caractères pour le nom. Vous ne pouvez inclure que des lettres (A-Z et a-z), des chiffres (0-9), des espaces et les caractères spéciaux suivants : ._- :/() #, @ [] +=& ; {} ! $*

Obligatoire : oui

Type : String

`tags`

Les balises que vous pouvez associer à la ressource du groupe de sécurité.

Obligatoire : non

Type : liste

Exigences

`vpc`

Un AWS. Networking.VPCnœud.

Obligatoire : oui

Type : String

Exemple

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

AWS.Networking.SecurityGroupEgressRule

AWS TNB prend en charge les règles de sortie des groupes de sécurité afin d'automatiser le provisionnement des règles de sortie des groupes de sécurité Amazon EC2 auxquelles il est possible de les associer. AWS Networking.SecurityGroup. Notez que vous devez fournir un cidr_ip/ destination_security_group /destination_prefix_list comme destination pour le trafic de sortie.

Syntaxe

```
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
```

Propriétés

`cidr_ip`

La plage d'adresses IPv4 au format CIDR. Vous devez spécifier une plage CIDR qui autorise le trafic sortant.

Obligatoire : non

Type : Chaîne

`cidr_ipv6`

La plage d'adresses IPv6 au format CIDR, pour le trafic sortant. Vous devez spécifier un groupe de sécurité de destination (`destination_security_group` ou `destination_prefix_list`) ou une plage d'adresses CIDR (`cidr_ip` ou `cidr_ipv6`).

Obligatoire : non

Type : Chaîne

`description`

Description d'une règle de groupe de sécurité pour le trafic entrant (sortant). Vous pouvez utiliser jusqu'à 255 caractères pour décrire la règle.

Obligatoire : non

Type : Chaîne

`destination_prefix_list`

L'ID de liste de préfixes d'une liste de préfixes gérée par Amazon VPC existante. Il s'agit de la destination à partir des instances de groupes de nœuds associées au groupe de sécurité. Pour plus d'informations sur les listes de préfixes gérées, consultez la section Listes de préfixes gérées dans le guide de l'utilisateur Amazon VPC.

Obligatoire : non

Type : Chaîne

`from_port`

Si le protocole est TCP ou UDP, il s'agit du début de la plage de ports. Si le protocole est ICMP ou ICMPv6, il s'agit du numéro de type. La valeur -1 indique tous les ICMP/ICMPv6 types. Si vous spécifiez tous les ICMP/ICMPv6 types, vous devez spécifier tous les ICMP/ICMPv6 codes.

Obligatoire : non

Type : Integer

`ip_protocol`

Nom du protocole IP (tcp, udp, icmp, icmpv6) ou numéro de protocole. Utilisez -1 pour spécifier tous les protocoles. Lorsque vous autorisez les règles du groupe de sécurité, la spécification de -1 ou d'un numéro de protocole autre que TCP, UDP, ICMP ou ICMPv6 autorise le trafic sur tous les ports, quelle que soit la plage de ports que vous spécifiez. Pour TCP, UDP et ICMP, vous devez spécifier une plage de ports. Pour icmpv6, la plage de ports est facultative ; si vous omettez la plage de ports, le trafic est autorisé pour tous les types et codes.

Obligatoire : oui

Type : String

`to_port`

Si le protocole est TCP ou UDP, il s'agit de la fin de la plage de ports. Si le protocole est ICMP ou ICMPv6, il s'agit du code. La valeur -1 indique tous les ICMP/ICMPv6 codes. Si vous spécifiez tous les ICMP/ICMPv6 types, vous devez spécifier tous les ICMP/ICMPv6 codes.

Obligatoire : non

Type : Integer

Exigences

`security_group`

ID du groupe de sécurité auquel cette règle doit être ajoutée.

Obligatoire : oui

Type : String

`destination_security_group`

ID ou référence TOSCA du groupe de sécurité de destination vers lequel le trafic de sortie est autorisé.

Obligatoire : non

Type : Chaîne

Exemple

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

AWS.Networking.SecurityGroupIngressRule

AWS TNB prend en charge les règles d'entrée des groupes de sécurité afin d'automatiser le provisionnement des règles d'entrée des groupes de sécurité Amazon EC2 auxquelles il est possible de s'associer. AWS Networking.SecurityGroup. Notez que vous devez fournir un cidr_ip/ source_security_group /source_prefix_list comme source pour le trafic entrant.

Syntaxe

```
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
```

Propriétés

`cidr_ip`

La plage d'adresses IPv4 au format CIDR. Vous devez spécifier une plage CIDR qui autorise le trafic entrant.

Obligatoire : non

Type : Chaîne

`cidr_ipv6`

La plage d'adresses IPv6 au format CIDR, pour le trafic entrant. Vous devez spécifier un groupe de sécurité source (`source_security_group` ou `source_prefix_list`) ou une plage d'adresses CIDR (`cidr_ip` ou `cidr_ipv6`).

Obligatoire : non

Type : Chaîne

`description`

Description d'une règle de groupe de sécurité d'entrée (entrante). Vous pouvez utiliser jusqu'à 255 caractères pour décrire la règle.

Obligatoire : non

Type : Chaîne

`source_prefix_list`

L'ID de liste de préfixes d'une liste de préfixes gérée par Amazon VPC existante. Il s'agit de la source à partir de laquelle les instances du groupe de nœuds associées au groupe de sécurité seront autorisées à recevoir du trafic. Pour plus d'informations sur les listes de préfixes gérées, consultez la section Listes de préfixes gérées dans le guide de l'utilisateur Amazon VPC.

Obligatoire : non

Type : Chaîne

`from_port`

Si le protocole est TCP ou UDP, il s'agit du début de la plage de ports. Si le protocole est ICMP ou ICMPv6, il s'agit du numéro de type. La valeur -1 indique tous les ICMP/ICMPv6 types. Si vous spécifiez tous les ICMP/ICMPv6 types, vous devez spécifier tous les ICMP/ICMPv6 codes.

Obligatoire : non

Type : Integer

`ip_protocol`

Nom du protocole IP (tcp, udp, icmp, icmpv6) ou numéro de protocole. Utilisez -1 pour spécifier tous les protocoles. Lorsque vous autorisez les règles du groupe de sécurité, la spécification de -1 ou d'un numéro de protocole autre que TCP, UDP, ICMP ou ICMPv6 autorise le trafic sur tous les ports, quelle que soit la plage de ports que vous spécifiez. Pour TCP, UDP et ICMP, vous devez spécifier une plage de ports. Pour icmpv6, la plage de ports est facultative ; si vous omettez la plage de ports, le trafic est autorisé pour tous les types et codes.

Obligatoire : oui

Type : String

`to_port`

Si le protocole est TCP ou UDP, il s'agit de la fin de la plage de ports. Si le protocole est ICMP ou ICMPv6, il s'agit du code. La valeur -1 indique tous les ICMP/ICMPv6 codes. Si vous spécifiez tous les ICMP/ICMPv6 types, vous devez spécifier tous les ICMP/ICMPv6 codes.

Obligatoire : non

Type : Integer

Exigences

`security_group`

ID du groupe de sécurité auquel cette règle doit être ajoutée.

Obligatoire : oui

Type : String

`source_security_group`

ID ou référence TOSCA du groupe de sécurité source à partir duquel le trafic entrant doit être autorisé.

Obligatoire : non

Type : Chaîne

Exemple

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

AWS.Resource.Import

Vous pouvez importer les AWS ressources suivantes dans AWS TNB :

- VPC
- Sous-réseau
- Table de routage
- Internet Gateway
- Security Group

Syntaxe

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

Propriétés

`resource_type`

Type de ressource importé dans AWS TNB.

Obligatoire : non

Type : liste

`resource_id`

ID de ressource importé dans AWS TNB.

Obligatoire : non

Type : liste

Exemple

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

AWS.Networking.ENI

Une interface réseau est un composant réseau logique d'un VPC qui représente une carte réseau virtuelle. Une adresse IP est attribuée à une interface réseau automatiquement ou manuellement en fonction de son sous-réseau. Après avoir déployé une instance Amazon EC2 dans un sous-réseau, vous pouvez y associer une interface réseau ou détacher une interface réseau de cette instance Amazon EC2 et la rattacher à une autre instance Amazon EC2 de ce sous-réseau. L'index de l'appareil identifie la position dans l'ordre de fixation.

Syntaxe

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

Propriétés

`device_index`

L'indice de l'appareil doit être supérieur à zéro.

Obligatoire : oui

Type : Integer

`source_dest_check`

Indique si l'interface réseau effectue source/destination la vérification. La valeur `true` signifie que la vérification est activée, tandis que la valeur `false` signifie qu'elle est désactivée.

Valeur autorisée : vrai, faux

Valeur par défaut : true

Obligatoire : non

Type : Boolean

`tags`

Les balises à associer à la ressource.

Obligatoire : non

Type : liste

Exigences

`subnet`

Un AWS. Networking.Subnetnœud.

Obligatoire : oui

Type : String

`security_groups`

Un AWS. Networking.SecurityGroupnœud.

Obligatoire : non

Type : Chaîne

Exemple

```
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

Un hook de cycle de vie vous permet d'exécuter vos propres scripts dans le cadre de votre infrastructure et de l'instanciation de votre réseau.

Syntaxe

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

Fonctionnalités

`execution`

Propriétés du moteur d'exécution du hook qui exécute les scripts du hook.

`type`

Type de moteur d'exécution du hook.

Obligatoire : non

Type : Chaîne

Valeurs possibles : `CODE_BUILD`

Exigences

`definition`

Un AWS. HookDefinition.Bashnœud.

Obligatoire : oui

Type : String

`vpc`

Un AWS. Networking.VPCnœud.

Obligatoire : oui

Type : String

Exemple

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

AWS.Networking.InternetGateway

Définit un nœud AWS Internet Gateway.

Syntaxe

```
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
```

Fonctionnalités

`routage`

Propriétés qui définissent la connexion de routage au sein du VPC. Vous devez inclure la `ipv6_dest_cidr` propriété `dest_cidr` ou.

`dest_cidr`

Bloc d'adresse CIDR IPv4 utilisé pour la correspondance de destination. Cette propriété est utilisée pour créer un itinéraire dans `RouteTable` et sa valeur est utilisée comme`DestinationCidrBlock`.

Obligatoire : Non si vous avez inclus la `ipv6_dest_cidr` propriété.

Type : Chaîne

`ipv6_dest_cidr`

Bloc d'adresse CIDR IPv6 utilisé pour la correspondance de destination.

Obligatoire : Non si vous avez inclus la `dest_cidr` propriété.

Type : Chaîne

Propriétés

`tags`

Les balises à associer à la ressource.

Obligatoire : non

Type : liste

`egress_only`

Une IPv6-specific propriété. Indique si la passerelle Internet est uniquement destinée à la communication de sortie ou non. Lorsque `egress_only` c'est vrai, vous devez définir la `ipv6_dest_cidr` propriété.

Obligatoire : non

Type : Boolean

Exigences

`vpc`

Un AWS. Networking.VPCnœud.

Obligatoire : oui

Type : String

`route_table`

Un AWS. Networking.RouteTablenœud.

Obligatoire : oui

Type : String

Exemple

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

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

AWS.Networking.RouteTable

Une table de routage contient un ensemble de règles, appelées routes, qui déterminent la direction du trafic réseau provenant des sous-réseaux de votre VPC ou de votre passerelle. Vous devez associer une table de routage à un VPC.

Syntaxe

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

Propriétés

`tags`

Balises à associer à la ressource.

Obligatoire : non

Type : liste

Exigences

`vpc`

Un AWS. Networking.VPCnœud.

Obligatoire : oui

Type : String

Exemple

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

AWS.Networking.Subnet

Un sous-réseau est une plage d'adresses IP de votre VPC, qui doit résider entièrement dans une seule zone de disponibilité. Vous devez spécifier un VPC, un bloc CIDR, une zone de disponibilité et une table de routage pour votre sous-réseau. Vous devez également définir si votre sous-réseau est privé ou public.

Syntaxe

```
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
```

Propriétés

`type`

Indique si les instances qui sont lancées dans ce sous-réseau reçoivent une adresse IPv4 publique.

Obligatoire : oui

Type : String

Valeurs possibles : `PUBLIC` | `PRIVATE`

`availability_zone`

Zone de disponibilité du sous-réseau. Ce champ prend en charge les zones de AWS disponibilité au sein d'une AWS région, par exemple `us-west-2` (USA Ouest (Oregon)). Il prend également en charge les zones AWS locales au sein de la zone de disponibilité, par exemple`us-west-2-lax-1a`.

Obligatoire : oui

Type : String

`cidr_block`

Le bloc CIDR pour le sous-réseau.

Obligatoire : non

Type : Chaîne

`ipv6_cidr_block`

Le bloc CIDR utilisé pour créer le sous-réseau IPv6. Si vous incluez cette propriété, ne l'incluez pas`ipv6_cidr_block_suffix`.

Obligatoire : non

Type : Chaîne

`ipv6_cidr_block_suffix`

Suffixe hexadécimal à 2 chiffres du bloc d'adresse CIDR IPv6 pour le sous-réseau créé via Amazon VPC. Utilisez le format suivant : *`2-digit hexadecimal`*`::/`*`subnetMask`*

Si vous incluez cette propriété, ne l'incluez pas`ipv6_cidr_block`.

Obligatoire : non

Type : Chaîne

`outpost_arn`

L'ARN dans AWS Outposts lequel le sous-réseau sera créé. Ajoutez cette propriété au modèle NSD si vous souhaitez lancer des nœuds autogérés Amazon EKS sur. AWS Outposts Pour plus d'informations, consultez Amazon EKS AWS Outposts dans le guide de l'utilisateur Amazon EKS.

Si vous ajoutez cette propriété au modèle NSD, vous devez définir la valeur de la `availability_zone` propriété sur la zone de disponibilité du AWS Outposts.

Obligatoire : non

Type : Chaîne

`tags`

Les balises à associer à la ressource.

Obligatoire : non

Type : liste

Exigences

`vpc`

Un AWS. Networking.VPCnœud.

Obligatoire : oui

Type : String

`route_table`

Un AWS. Networking.RouteTablenœud.

Obligatoire : oui

Type : String

Exemple

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

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

AWS.Deployment.VNFDeployment

Les déploiements NF sont modélisés en fournissant l'infrastructure et l'application qui y sont associées. L'attribut cluster indique le cluster EKS qui hébergera vos NF. L'attribut vnfs spécifie les fonctions réseau pour votre déploiement. Vous pouvez également fournir des opérations d'accroche du cycle de vie facultatives de type pre_create et post_create pour exécuter des instructions spécifiques à votre déploiement, telles que l'appel d'une API du système de gestion des stocks.

Syntaxe

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

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

Exigences

`deployment`

Un AWS. Deployment.VNFDeploymentnœud.

Obligatoire : non

Type : Chaîne

`cluster`

Un AWS. Compute.EKSnœud.

Obligatoire : oui

Type : String

`vnfs`

Un nœud AWS.VNF.

Obligatoire : oui

Type : String

Interfaces

Crochets

Définit l'étape au cours de laquelle les hooks du cycle de vie sont exécutés.

`pre_create`

Un AWS. HookExecutionnœud. Ce hook est exécuté avant le déploiement du `VNFDeployment` nœud.

Obligatoire : non

Type : Chaîne

`post_create`

Un AWS. HookExecutionnœud. Ce hook est exécuté après le déploiement du `VNFDeployment` nœud.

Obligatoire : non

Type : Chaîne

Exemple

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

AWS.Networking.VPC

Vous devez spécifier un bloc CIDR pour votre cloud privé virtuel (VPC).

Syntaxe

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

Propriétés

`cidr_block`

La gamme de réseau IPv4 pour le VPC, en notation CIDR.

Obligatoire : oui

Type : String

`ipv6_cidr_block`

Le bloc d'adresse CIDR IPv6 utilisé pour créer le VPC.

Valeur autorisée : `AMAZON_PROVIDED`

Obligatoire : non

Type : Chaîne

`dns_support`

Indique si les instances lancées dans le VPC ont obtenu des noms d'hôte DNS.

Obligatoire : non

Type : Boolean

Valeur par défaut : `false`

`tags`

Balises à associer à la ressource.

Obligatoire : non

Type : liste

Exemple

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

AWS.Networking.NATGateway

Vous pouvez définir un nœud de passerelle NAT public ou privé sur un sous-réseau. Pour une passerelle publique, si vous ne fournissez pas d'identifiant d'allocation d'adresse IP élastique, AWS TNB attribuera une adresse IP élastique à votre compte et l'associera à la passerelle.

Syntaxe

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

Propriétés

`subnet`

Le AWS. Networking.Subnetréférence de nœud.

Obligatoire : oui

Type : String

`internet_gateway`

Le AWS. Networking.InternetGatewayréférence de nœud.

Obligatoire : oui

Type : String

Propriétés

`type`

Indique si la passerelle est publique ou privée.

Valeur autorisée :`PUBLIC`, `PRIVATE`

Obligatoire : oui

Type : String

`eip_allocation_id`

L'ID qui représente l'allocation de l'adresse IP élastique.

Obligatoire : non

Type : Chaîne

`tags`

Balises à associer à la ressource.

Obligatoire : non

Type : liste

Exemple

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

AWS.Networking.Route

Vous pouvez définir un nœud de route qui associe la route de destination à la passerelle NAT en tant que ressource cible et ajoute la route à la table de routage associée.

Syntaxe

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

```
    route_table: String
```

Propriétés

`dest_cidr_blocks`

La liste des routes IPv4 de destination vers la ressource cible.

Obligatoire : oui

Type : liste

Type de membre : Chaîne

Exigences

`nat_gateway`

Le AWS. Networking.NATGatewayréférence de nœud.

Obligatoire : oui

Type : String

`route_table`

Le AWS. Networking.RouteTableréférence de nœud.

Obligatoire : oui

Type : String

Exemple

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

```
    route_table: Free5GCRouteTable
```

AWS.Store.SSMParameters

Vous pouvez créer des paramètres SSM via AWS TNB. Les paramètres SSM que vous créez sont créés dans SSM et préfixés par l'ID de l'instance réseau AWS TNB. Cela empêche le remplacement des valeurs des paramètres lorsque plusieurs instances sont instanciées et mises à niveau à l'aide du même modèle NSD.

Syntaxe

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

Propriétés

Parameters

`name`

Nom de la propriété SSM. Utilisez le format suivant : `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`

Le nom de chaque paramètre doit comporter moins de 256 caractères.

Obligatoire : oui

Type : String

`value`

Valeur de la propriété ssm. Utilisez l'un des formats suivants :

- Pour les valeurs sans références : `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`
- Pour les références statiques : `^\$\{[a-zA-Z0-9]+\.(properties|capabilities|requirements)(\.([a-zA-Z0-9\-_]+))+\}$`
- Pour les références dynamiques : `^\$\{[a-zA-Z0-9]+\.(name|id|arn)\}$`

La valeur de chaque paramètre doit être inférieure à 4 Ko.

Obligatoire : oui

Type : String

`tags`

Les balises que vous pouvez associer à une propriété SSM.

Obligatoire : non

Type : liste

Exemple

```
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"
```

Nœuds communs

Définissez des nœuds pour le NSD et le VNFD.

- AWS. HookDefinition.Bash

AWS.HookDefinition.Bash

Définit une AWS HookDefinition entréebash.

Syntaxe

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

Propriétés

`implementation`

Le chemin relatif vers la définition du crochet. Le format doit être le suivant : `./hooks/`*`script_name`*`.sh`

Obligatoire : oui

Type : String

`environment_variables`

Les variables d'environnement pour le script hook bash. Utilisez le format suivant : **`envName`**`=`**`envValue`** avec les modèles de regex suivants :

- Pour les valeurs sans références : `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`
- Pour les références statiques : `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=\$\{[a-zA-Z0-9]+\.(properties|capabilities|requirements)(\.([a-zA-Z0-9\-_]+))+\}$`
- Pour les références dynamiques : `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=\$\{[a-zA-Z0-9]+\.(name|id|arn)\}$`

Assurez-vous que la **`envName`**`=`**`envValue`** valeur répond aux critères suivants :

- N'utilisez pas d'espaces.
- **`envName`**Commencez par une lettre (A-Z ou a-z) ou un chiffre (0-9).
- Ne commencez pas le nom de la variable d'environnement par les mots clés réservés AWS TNB suivants (sans distinction majuscules/minuscules) :
  - CONSTRUCTION DE CODE
  - TNB

  - MAISON
  - AWS
- Vous pouvez utiliser n'importe quel nombre de lettres (A-Z ou a-z), de chiffres (0-9), de caractères spéciaux et pour - et_. **envName envValue**
- Chaque variable d'environnement (each **envName** =**envValue**) doit comporter moins de 128 caractères.

Exemple : `A123-45xYz=Example_789`

Obligatoire : non

Type : liste

`execution_role`

Le rôle de l'exécution du hook.

Obligatoire : oui

Type : String

exemple

```
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"
```

Sécurité dans AWS TNB

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez de centres de données et d'architectures réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit ceci comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS Cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Third-party les auditeurs testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité Programmes de de conformité. Pour en savoir plus sur les programmes de conformité qui s'appliquent à AWS Telco Network Builder, voir AWS Services concernés par programme de conformitéAWS .
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation du AWS TNB. Les rubriques suivantes expliquent comment configurer le AWS TNB pour répondre à vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui vous aident à surveiller et à sécuriser vos ressources AWS TNB.

Table des matières

Protection des données dans AWS TNB

Le modèle de responsabilité AWS partagée (modèle de) s'applique à la protection des données dans AWS Telco Network Builder. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la FAQ sur la confidentialité des données et les . Pour plus d'informations sur la protection des données en Europe, consultez le Centre du règlement général sur la protection des données (RGPD).

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou Gestion des identités et des accès AWS (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section Utilisation des CloudTrail sentiers dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez Norme FIPS (Federal Information Processing Standard) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec AWS TNB ou d'autres utilisateurs à Services AWS l'aide de la console, de l'API ou des AWS SDK. AWS CLI Toutes les données que vous

entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d’informations d’identification dans l’adresse URL permettant de valider votre demande adressée à ce serveur.

Manipulation des données

Lorsque vous fermez votre AWS compte, AWS TNB marque vos données pour suppression et les supprime de toute utilisation. Si vous réactivez votre AWS compte dans les 90 jours, AWS TNB restaure vos données. Après 120 jours, AWS TNB supprime définitivement vos données. AWS TNB met également fin à vos réseaux et supprime vos packages de fonctions et vos packages réseau.

Chiffrement au repos

AWS TNB chiffre toujours toutes les données stockées dans le service au repos sans nécessiter de configuration supplémentaire. Ce cryptage est automatique via AWS Key Management Service.

Chiffrement en transit

AWS TNB sécurise toutes les données en transit à l'aide du protocole TLS (Transport Layer Security) 1.2.

Il est de votre responsabilité de chiffrer les données entre vos agents de simulation et leurs clients.

Inter-network confidentialité du trafic

AWS Les ressources informatiques du TNB résident dans un cloud privé virtuel (VPC) partagé par tous les clients. Tout le trafic interne du AWS TNB est resté sur le AWS réseau et n'a pas transité par Internet. Les connexions entre vos agents de simulation et leurs clients sont acheminées via Internet.

Gestion des identités et des accès pour AWS TNB

Gestion des identités et des accès AWS (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les ressources AWS TNB. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Table des matières

Public ciblé

La façon dont vous utilisez Gestion des identités et des accès AWS (IAM) varie en fonction de votre rôle :

- Utilisateur du service : demandez des autorisations à votre administrateur si vous ne pouvez pas accéder aux fonctionnalités (voir Résolution des problèmes AWS Identité et accès à Telco Network Builder)
- Administrateur du service : déterminez l'accès des utilisateurs et soumettez les demandes d'autorisation (voir Comment ? AWS TNB travaille avec IAM)
- Administrateur IAM : rédigez des politiques pour gérer l'accès (voir Identity-based exemples de politiques pour AWS Générateur de réseaux de télécommunications)

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une authentification unique ou d'informations d'identification. Google/Facebook Pour plus d'informations sur la connexion, consultez Connexion à votre Compte AWS dans le Guide de l'utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez Signature AWS Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une seule identité de connexion appelée utilisateur Compte AWS root qui dispose d'un accès complet à toutes Services AWS les ressources. Il est vivement déconseillé d'utiliser l'utilisateur racine pour vos tâches quotidiennes. Pour les tâches qui requièrent des informations d'identification de l'utilisateur racine, consultez Tâches qui requièrent les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Identité fédérée

Il est recommandé d'obliger les utilisateurs humains à utiliser la fédération avec un fournisseur d'identité pour accéder à Services AWS l'aide d'informations d'identification temporaires.

Une identité fédérée est un utilisateur provenant de l'annuaire de votre entreprise, de votre fournisseur d'identité Web ou Directory Service qui y accède à Services AWS l'aide d'informations d'identification provenant d'une source d'identité. Les identités fédérées assument des rôles qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Pour plus d'informations, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.

Les groupes IAM spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une identité dotée d'autorisations spécifiques qui fournit des informations d'identification temporaires. Vous pouvez assumer un rôle en passant d'un rôle d'utilisateur à un rôle IAM (console) ou en appelant une opération d' AWS API AWS CLI ou d'API. Pour plus d'informations, consultez Méthodes pour endosser un rôle dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès intercompte, les accès entre services et les applications exécutées sur Amazon EC2. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Gestion de l'accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique définit les autorisations lorsqu'elles sont associées à une identité ou à une ressource. AWS évalue ces politiques lorsqu'un directeur fait une demande. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations les documents de politique JSON, consultez Vue d'ensemble des politiques JSON dans le Guide de l'utilisateur IAM.

À l'aide de politiques, les administrateurs précisent qui a accès à quoi en définissant quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Un administrateur IAM crée des politiques IAM et les ajoute aux rôles, que les utilisateurs peuvent ensuite assumer. Les politiques IAM définissent les autorisations quelle que soit la méthode que vous utilisez pour exécuter l'opération.

Identity-based politiques

Identity-based les politiques sont des documents de politique d'autorisation JSON que vous attachez à une identité (utilisateur, groupe ou rôle). Ces politiques contrôlent les actions que peuvent exécuter ces identités, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Identity-based les politiques peuvent être des politiques intégrées (intégrées directement dans une seule identité) ou des politiques gérées (politiques autonomes associées à plusieurs identités). Pour découvrir comment choisir entre des politiques gérées et en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

Resource-based politiques

Resource-based les politiques sont des documents de politique JSON que vous attachez à une ressource. Les exemples incluent les politiques de confiance de rôle IAM et les stratégies de

compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Vous devez spécifier un principal dans une politique basée sur les ressources.

Resource-based les politiques sont des politiques intégrées qui se trouvent dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Autres types de politique

AWS prend en charge des types de politiques supplémentaires qui peuvent définir les autorisations maximales accordées par les types de politiques les plus courants :

- Limites d'autorisations : une limite des autorisations définit le nombre maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM. Pour plus d'informations, consultez Limites d'autorisations pour des entités IAM dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCP) : spécifient les autorisations maximales pour une organisation ou une unité organisationnelle dans AWS Organizations. Pour plus d'informations, consultez Politiques de contrôle de service dans le Guide de l'utilisateur AWS Organizations .
- Politiques de contrôle des ressources (RCP) : définissent les autorisations maximales disponibles pour les ressources de votre organisation. Pour plus d'informations, consultez Politiques de contrôle des ressources (RCP) dans le Guide de l'utilisateur AWS Organizations .
- Politiques de session : politiques avancées que vous passez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Comment ? AWS TNB travaille avec IAM

Avant d'utiliser IAM pour gérer l'accès à AWS TNB, découvrez quelles fonctionnalités IAM peuvent être utilisées avec TNB. AWS

Fonctionnalités IAM que vous pouvez utiliser avec AWS Générateur de réseaux de télécommunications

Fonctionnalité IAM	AWS Assistance TNB
Identity-based politiques	Oui
Resource-based politiques	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition de politique	Oui
ACL	Non
ABAC (étiquettes dans les politiques)	Oui
Informations d'identification temporaires	Oui
Autorisations de principal	Oui
Rôles du service	Non
Service-linked rôles	Non

Pour obtenir une vue d'ensemble du fonctionnement du AWS TNB et des autres AWS services avec la plupart des fonctionnalités IAM, consultez la section AWS Services compatibles avec IAM dans le guide de l'utilisateur IAM.

Identity-based politiques pour AWS TNB

Prend en charge les politiques basées sur l'identité : oui

Identity-based les politiques sont des documents de politique d'autorisation JSON que vous pouvez associer à une identité, telle qu'un utilisateur IAM, un groupe d'utilisateurs ou un rôle. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Pour découvrir tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

Identity-based exemples de politiques pour AWS TNB

Pour consulter des exemples de politiques basées sur l'identité du AWS TNB, voir. Identity-based exemples de politiques pour AWS Générateur de réseaux de télécommunications

Resource-based politiques au sein de AWS TNB

Prend en charge les politiques basées sur les ressources : non

Resource-based les politiques sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Actions politiques pour AWS TNB

Prend en charge les actions de politique : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour consulter la liste des actions AWS TNB, voir Actions définies par AWS Telco Network Builder dans la référence d'autorisation de service.

Les actions politiques dans AWS TNB utilisent le préfixe suivant avant l'action :

```
tnb
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

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

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `List`, incluez l'action suivante :

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

Pour consulter des exemples de politiques basées sur l'identité du AWS TNB, voir. Identity-based exemples de politiques pour AWS Générateur de réseaux de télécommunications

Ressources politiques pour AWS TNB

Prend en charge les ressources de politique : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

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

Pour consulter la liste des types de ressources AWS TNB et de leurs ARN, consultez la section Ressources définies par AWS Telco Network Builder dans la référence d'autorisation de service.

Pour savoir avec quelles actions vous pouvez spécifier l'ARN de chaque ressource, voir Actions définies par AWS Telco Network Builder.

Pour consulter des exemples de politiques basées sur l'identité du AWS TNB, voir. Identity-based exemples de politiques pour AWS Générateur de réseaux de télécommunications

Clés de conditions de politique pour AWS TNB

Prend en charge les clés de condition de politique spécifiques au service : oui

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` indique à quel moment les instructions s'exécutent en fonction de critères définis. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

Pour consulter la liste des clés de condition AWS TNB, consultez la section Clés de condition pour AWS Telco Network Builder dans la référence d'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, voir Actions définies par AWS Telco Network Builder.

Pour consulter des exemples de politiques basées sur l'identité du AWS TNB, voir. Identity-based exemples de politiques pour AWS Générateur de réseaux de télécommunications

ACL dans AWS TNB

Prend en charge les ACL : non

Les listes de contrôle d'accès (ACL) vérifie quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

ABAC avec AWS TNB

Prise en charge d'ABAC (balises dans les politiques) : Oui

Attribute-based le contrôle d'accès (ABAC) est une stratégie d'autorisation qui définit les autorisations en fonction d'attributs appelés balises. Vous pouvez associer des balises aux entités et aux AWS ressources IAM, puis concevoir des politiques ABAC pour autoriser les opérations lorsque la balise du principal correspond à la balise de la ressource.

Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'élément de condition d'une politique utilisant les clés de condition `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informations sur ABAC, consultez Définition d'autorisations avec l'autorisation ABAC dans le Guide de l'utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez Utilisation du contrôle d'accès par attributs (ABAC) dans le Guide de l'utilisateur IAM.

Utilisation d'informations d'identification temporaires avec AWS TNB

Prend en charge les informations d'identification temporaires : oui

Les informations d'identification temporaires fournissent un accès à court terme aux AWS ressources et sont automatiquement créées lorsque vous utilisez la fédération ou que vous changez de rôle. AWS recommande de générer dynamiquement des informations d'identification temporaires au lieu d'utiliser des clés d'accès à long terme. Pour plus d'informations, consultez Informations d'identification de sécurité temporaires dans IAM et Services AWS compatibles avec IAM dans le Guide de l'utilisateur IAM.

Cross-service autorisations principales pour AWS TNB

Prend en charge les sessions d'accès direct (FAS) : oui

Les sessions d'accès direct (FAS) utilisent les autorisations du principal appelant et Service AWS, combinées Service AWS à la demande d'envoi de demandes aux services en aval. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez la section Sessions de transmission d'accès.

Rôles de service pour AWS TNB

Prend en charge les rôles de service : Non

Un rôle de service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

Service-linked rôles pour AWS TNB

Prend en charge les rôles liés à un service : non

Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut assumer le rôle d'effectuer une action en votre nom. Service-linked les rôles apparaissent dans votre fichier Compte AWS et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Identity-based exemples de politiques pour AWS Générateur de réseaux de télécommunications

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou à modifier des ressources AWS TNB. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM peut créer des politiques IAM.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez Création de politiques IAM (console) dans le Guide de l'utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par AWS TNB, y compris le format des ARN pour chacun des types de ressources, voir Actions, ressources et clés de condition pour AWS Telco Network Builder dans la référence d'autorisation de service.

Table des matières

Bonnes pratiques en matière de politiques

Identity-based les politiques déterminent si quelqu'un peut créer, accéder ou supprimer les ressources AWS TNB de votre compte. Ces actions peuvent entraîner des frais pour votre

Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l'utilisateur IAM.
- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez politiques et autorisations dans IAM dans le Guide de l'utilisateur IAM.
- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service AWS, tel que CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.
- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politiques avec IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Sécurisation de l'accès aux API avec MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de AWS Console TNB

Pour accéder à la console AWS Telco Network Builder, vous devez disposer d'un ensemble minimal d'autorisations. Ces autorisations doivent vous permettre de répertorier et d'afficher les détails des ressources AWS TNB de votre Compte AWS. Si vous créez une politique basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette politique.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API qu'ils tentent d'effectuer.

Exemples de politiques relatives aux rôles de service

En tant qu'administrateur, vous possédez et gérez les ressources créées par AWS TNB, telles que définies par les modèles d'environnement et de service. Vous devez associer des rôles de service IAM à votre compte pour permettre à AWS TNB de créer des ressources pour la gestion du cycle de vie de votre réseau.

Un rôle de service IAM permet à AWS TNB de passer des appels aux ressources en votre nom afin d'instancier et de gérer vos réseaux. Si vous spécifiez un rôle de service, AWS TNB utilise les informations d'identification de ce rôle.

Le rôle de service et sa politique d'autorisation sont créés à partir du service IAM. Pour plus d'informations sur la création d'un rôle de service, consultez la section Création d'un rôle pour déléguer des autorisations à un AWS service dans le Guide de l'utilisateur IAM.

AWS Rôle de service TNB

En tant que membre de l'équipe de la plateforme, vous pouvez, en tant qu'administrateur, créer un rôle de service AWS TNB et le fournir à AWS TNB. Ce rôle permet à AWS TNB de passer des appels vers d'autres services tels qu'Amazon Elastic Kubernetes CloudFormation Service, de fournir l'infrastructure requise pour votre réseau et de fournir des fonctions réseau telles que définies dans votre NSD.

Nous vous recommandons d'utiliser le rôle IAM et la politique de confiance suivants pour votre rôle de service AWS TNB. Lorsque vous délimitez les autorisations relatives à cette politique, gardez à

l'esprit que AWS TNB peut échouer en cas d'erreurs d'accès refusé vers des ressources exclues de votre politique.

Le code suivant illustre une politique de rôle de service AWS TNB :

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"
        }
    ]
```

```
}
```

Le code suivant illustre la politique de confiance du service AWS TNB :

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 Rôle de service TNB pour le cluster Amazon EKS

Lorsque vous créez des ressources Amazon EKS dans votre NSD, vous fournissez l'`cluster_role`attribut pour spécifier le rôle qui sera utilisé pour créer votre cluster Amazon EKS.

L'exemple suivant montre un AWS CloudFormation modèle qui crée un rôle de service AWS TNB pour la politique de cluster Amazon EKS.

```
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"
```

Pour plus d'informations sur les rôles IAM utilisant AWS CloudFormation un modèle, consultez les sections suivantes du guide de l'AWS CloudFormation utilisateur :

- AWS::IAM::Role
- Sélection d'un modèle de pile

AWS Rôle de service TNB pour le groupe de nœuds Amazon EKS

Lorsque vous créez des ressources de groupe de nœuds Amazon EKS dans votre NSD, vous fournissez l'`node_role`attribut pour spécifier le rôle qui sera utilisé pour créer votre groupe de nœuds Amazon EKS.

L'exemple suivant montre un CloudFormation modèle qui crée un rôle de service AWS TNB pour la politique de groupe de nœuds Amazon EKS.

```
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/*"
```

Pour plus d'informations sur les rôles IAM utilisant AWS CloudFormation un modèle, consultez les sections suivantes du guide de l'AWS CloudFormation utilisateur :

- AWS::IAM::Role
- Sélection d'un modèle de pile

AWS Rôle de service TNB pour Multus

Lorsque vous créez une ressource Amazon EKS dans votre NSD et que vous souhaitez gérer Multus dans le cadre de votre modèle de déploiement, vous devez fournir l'`multus_role`attribut pour spécifier le rôle qui sera utilisé pour gérer Multus.

L'exemple suivant montre un CloudFormation modèle qui crée un rôle de service AWS TNB pour une politique Multus.

```
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: "*"
```

Pour plus d'informations sur les rôles IAM utilisant AWS CloudFormation un modèle, consultez les sections suivantes du guide de l'AWS CloudFormation utilisateur :

- AWS::IAM::Role
- Sélection d'un modèle de pile

AWS Rôle du service TNB dans le cadre d'une politique d'accrochage du cycle de vie

Lorsque votre NSD ou votre package de fonctions réseau utilise un hook de cycle de vie, vous avez besoin d'un rôle de service vous permettant de créer un environnement pour l'exécution de vos hooks de cycle de vie.

> **Note**
>
> Votre politique d'accrochage du cycle de vie doit être basée sur ce que tente de faire votre crochet du cycle de vie.

L'exemple suivant montre un CloudFormation modèle qui crée un rôle de service AWS TNB pour une politique d'accrochage du cycle de vie.

```
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"
```

Pour plus d'informations sur les rôles IAM utilisant AWS CloudFormation un modèle, consultez les sections suivantes du guide de l'AWS CloudFormation utilisateur :

- AWS::IAM::Role
- Sélection d'un modèle de pile

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or 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": "*"
        }
    ]
}
```

Résolution des problèmes AWS Identité et accès à Telco Network Builder

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec AWS TNB et IAM.

Problèmes

- Je ne suis pas autorisé à effectuer une action dans AWS TNB
- Je ne suis pas autorisé à effectuer iam : PassRole

- Je souhaite autoriser des personnes extérieures à mon Compte AWS pour accéder à mon AWS Ressources du TNB

Je ne suis pas autorisé à effectuer une action dans AWS TNB

Si vous recevez une erreur qui indique que vous n'êtes pas autorisé à effectuer une action, vos politiques doivent être mises à jour afin de vous permettre d'effectuer l'action.

L'exemple d'erreur suivant se produit quand l'utilisateur IAM `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource *`my-example-widget`* fictive, mais ne dispose pas des autorisations `tnb:`*`GetWidget`* fictives.

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

Dans ce cas, la stratégie de Mateo doit être mise à jour pour l'autoriser à accéder à la ressource *`my-example-widget`* à l'aide de l'action `tnb:`*`GetWidget`*.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez un message d'erreur indiquant que vous n'êtes pas autorisé à effectuer l'`iam:PassRole`action, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à AWS TNB.

Certains vous Services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, vous devez disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans AWS TNB. Toutefois, l'action nécessite que le service ait des autorisations accordées par un rôle de service. Mary n'est pas autorisée à transmettre le rôle au service.

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

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je souhaite autoriser des personnes extérieures à mon Compte AWS pour accéder à mon AWS Ressources du TNB

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si AWS TNB prend en charge ces fonctionnalités, consultezComment ? AWS TNB travaille avec IAM.
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section Fournir un accès à des ressources Comptes AWS détenues par des tiers dans le guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Validation de conformité pour AWS TNB

Pour savoir si un programme Services AWS de conformité Service AWS s'inscrit dans le champ d'application de programmes de conformité spécifiques, consultez Services AWS la section de

conformité et sélectionnez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de AWS conformité Programmes AWS de .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir Téléchargement de rapports dans AWS Artifact .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise et les lois et réglementations applicables. Pour plus d'informations sur votre responsabilité en matière de conformité lors de l'utilisation Services AWS, consultez AWS la documentation de sécurité.

Résilience dans AWS TNB

L'infrastructure AWS mondiale est construite autour Régions AWS de zones de disponibilité. Régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur les zones de disponibilité Régions AWS et les zones de disponibilité, consultez la section Infrastructure AWS globale.

AWS TNB exécute votre service réseau sur des clusters EKS dans un cloud privé virtuel (VPC) dans AWS la région de votre choix.

Sécurité de l'infrastructure dans AWS TNB

En tant que service géré, AWS Telco Network Builder est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à AWS TNB via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Des suites de chiffrement dotées d'un secret de transmission parfait (PFS), telles que DHE (Ephemeral) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

Voici quelques exemples de responsabilités partagées :

- AWS est chargé de sécuriser les composants compatibles avec le AWS TNB, notamment :
  - Instances de calcul (également appelées « travailleurs »)
  - Bases de données internes
  - Communications réseau entre les composants internes
  - L'interface de programmation d'applications (API) AWS TNB
  - AWS Kits de développement logiciel (SDK)
- Vous êtes responsable de la sécurisation de votre accès à vos AWS ressources et aux composants de votre charge de travail, notamment (mais sans s'y limiter) :
  - Utilisateurs, groupes, rôles et politiques IAM
  - Buckets S3 que vous utilisez pour stocker vos données pour TNB AWS
  - Autres ressources Services AWS et ressources que vous utilisez pour prendre en charge le service réseau que vous avez fourni via TNB AWS
  - Le code de votre application
  - Connexions entre le service réseau que vous avez fourni via AWS TNB et ses clients

> **⚠ Important**
>
> Vous êtes responsable de la mise en œuvre d'un plan de reprise après sinistre capable de restaurer efficacement un service réseau que vous avez fourni par le biais de AWS TNB.

Modèle de sécurité de connectivité réseau

Les services réseau que vous fournissez via AWS TNB s'exécutent sur des instances de calcul au sein d'un cloud privé virtuel (VPC) situé dans AWS une région que vous sélectionnez. Un VPC est un réseau virtuel dans le AWS cloud qui isole l'infrastructure par charge de travail ou entité organisationnelle. Les communications entre les instances de calcul au sein des VPC restent au

sein du AWS réseau et ne transitent pas par Internet. Certaines communications internes du service transitent par Internet et sont cryptées. Les services réseau fournis via AWS TNB pour tous les clients opérant dans la même région partagent le même VPC. Les services réseau fournis via AWS TNB pour différents clients utilisent des instances de calcul distinctes au sein du même VPC.

Les communications entre les clients de votre service réseau et votre service réseau dans AWS TNB passent par Internet. AWS TNB ne gère pas ces connexions. Il est de votre responsabilité de sécuriser les connexions avec vos clients.

Vos connexions à AWS TNB via le AWS Management Console, AWS Command Line Interface (AWS CLI) et les AWS SDK sont cryptées.

Version IMDS

AWS TNB prend en charge les instances qui exploitent le service de métadonnées d'instance version 2 (IMDSv2), une méthode axée sur les sessions. IMDSv2 inclut un niveau de sécurité supérieur à celui de l'IMDSV1. Pour plus d'informations, consultez Renforcer la défense contre les pare-feux ouverts, les proxys inverses et les vulnérabilités SSRF grâce aux améliorations apportées au service de métadonnées d'instance Amazon EC2.

Lorsque vous lancez votre instance, vous devez utiliser IMDSv2. Pour plus d'informations sur IMDSv2, consultez la section Utiliser IMDSv2 dans le guide de l'utilisateur Amazon EC2.

Surveillance du AWS TNB

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances de AWS TNB et de vos autres AWS solutions. AWS permet AWS CloudTrail de surveiller le AWS TNB, de signaler tout problème et de prendre des mesures automatiques le cas échéant.

CloudTrail À utiliser pour capturer des informations détaillées sur les appels passés à AWS APIs. Vous pouvez stocker ces appels sous forme de fichiers journaux dans Amazon S3. Vous pouvez utiliser ces CloudTrail journaux pour déterminer des informations telles que l'appel a été effectué, l'adresse IP source d'où provient l'appel, l'auteur de l'appel et la date de l'appel.

Les CloudTrail journaux contiennent des informations sur les appels aux actions d'API pour AWS TNB. Ils contiennent également des informations relatives aux appels à des actions d'API provenant de services tels qu'Amazon EC2 et Amazon EBS.

Enregistrement des appels d'API AWS Telco Network Builder à l'aide de AWS CloudTrail

AWS Telco Network Builder est intégré à AWS CloudTrailun service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un Service AWS. CloudTrail capture tous les appels d'API pour AWS TNB sous forme d'événements. Les appels capturés incluent des appels provenant de la console AWS TNB et des appels de code vers les opérations de l'API AWS TNB. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite à AWS TNB, l'adresse IP à partir de laquelle la demande a été faite, la date à laquelle elle a été faite et des détails supplémentaires.

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer :

- Si la demande a été effectuée avec des informations d'identification d'utilisateur root ou d'utilisateur root.
- Si la demande a été faite au nom d'un utilisateur du centre d'identité IAM.
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la requête a été effectuée par un autre Service AWS.

CloudTrail est actif dans votre compte Compte AWS lorsque vous créez le compte et vous avez automatiquement accès à l'historique des CloudTrail événements. L'historique des CloudTrail événements fournit un enregistrement consultable, consultable, téléchargeable et immuable des 90 derniers jours des événements de gestion enregistrés dans un. Région AWS Pour plus d'informations, consultez la section Utilisation de l'historique des CloudTrail événements dans le guide de AWS CloudTrail l'utilisateur. La consultation de CloudTrail l'historique des événements est gratuite.

Pour un enregistrement continu des événements de vos 90 Compte AWS derniers jours, créez un magasin de données sur les événements de Trail ou CloudTrailLake.

CloudTrail sentiers

Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Tous les sentiers créés à l'aide du AWS Management Console sont multirégionaux. Vous ne pouvez créer un journal de suivi en une ou plusieurs régions à l'aide de l' AWS CLI. Il est recommandé de créer un parcours multirégional, car vous capturez l'activité dans l'ensemble Régions AWS de votre compte. Si vous créez un journal de suivi pour une seule région, il convient de n'afficher que les événements enregistrés dans le journal de suivi pour une seule région Région AWS. Pour plus d'informations sur les journaux de suivi, consultez Créez un journal de suivi dans vos Compte AWS et Création d'un journal de suivi pour une organisation dans le AWS CloudTrail Guide de l'utilisateur.

Vous pouvez envoyer une copie de vos événements de gestion en cours dans votre compartiment Amazon S3 gratuitement CloudTrail en créant un journal. Toutefois, des frais de stockage Amazon S3 sont facturés. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification. Pour obtenir des informations sur la tarification Amazon S3, consultez Tarification Amazon S3.

CloudTrail Stockages de données sur les événements du lac

CloudTrail Lake vous permet d'exécuter des requêtes SQL sur vos événements. CloudTrail Lake convertit les événements existants au format JSON basé sur les lignes au format Apache ORC. ORC est un format de stockage en colonnes qui est optimisé pour une récupération rapide des données. Les événements sont agrégés dans des magasins de données d'événement. Ceux-ci constituent des collections immuables d'événements basées sur des critères que vous sélectionnez en appliquant des sélecteurs d'événements avancés. Les sélecteurs que vous appliquez à un magasin de données d'événement contrôlent les événements qui persistent et

que vous pouvez interroger. Pour plus d'informations sur CloudTrail Lake, consultez la section Travailler avec AWS CloudTrail Lake dans le guide de AWS CloudTrail l'utilisateur.

CloudTrail Les stockages et requêtes de données sur les événements de Lake entraînent des coûts. Lorsque vous créez un magasin de données d'événement, vous choisissez l'option de tarification que vous voulez utiliser pour le magasin de données d'événement. L'option de tarification détermine le coût d'ingestion et de stockage des événements, ainsi que les périodes de conservation par défaut et maximale pour le magasin de données d'événement. Pour plus d'informations sur la CloudTrail tarification, consultez la section AWS CloudTrail Tarification.

AWS Exemples d'événements TNB

Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'opération d'API demandée, la date et l'heure de l'opération, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics. Les événements n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre un CloudTrail événement illustrant l'`CreateSolFunctionPackage`opération.


```
{
    "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"
}
```


Pour plus d'informations sur le contenu des CloudTrail enregistrements, voir le contenu des CloudTrail enregistrements dans le Guide de AWS CloudTrail l'utilisateur.

AWS Tâches de déploiement TNB

Comprenez les tâches de déploiement pour surveiller efficacement les déploiements et agir plus rapidement.

Le tableau suivant répertorie les tâches de déploiement du AWS TNB :

Nom de la tâche pour les déploiements commencés avant le 7 mars 2024	Nom de la tâche pour les déploiements commencés le 7 mars 2024 ou après	Description de la tâche
AppInstallation	ClusterPluginInstall	Installe le plug-in Multus sur le cluster Amazon EKS.
AppUpdate	aucun changement de nom	Met à jour les fonctions réseau déjà installées dans une instance réseau.
-	ClusterPluginUninstall	Désinstalle les plug-ins sur le cluster Amazon EKS.
ClusterStorageClassConfiguration	aucun changement de nom	Configure la classe de stockage (pilote CSI) sur un cluster Amazon EKS.
FunctionDeletion	aucun changement de nom	Supprime les fonctions réseau des ressources AWS TNB.
FunctionInstantiation	FunctionInstall	Déploie les fonctions réseau à l'aide de HELM.
FunctionUninstallation	FunctionUninstall	Désinstalle la fonction réseau d'un cluster Amazon EKS.
HookExecution	aucun changement de nom	Exécute les hooks du cycle de vie tels que définis dans le NSD.
InfrastructureCancellation	aucun changement de nom	Annule un service réseau.
InfrastructureInstantation	aucun changement de nom	Fournit AWS des ressources pour le compte de l'utilisateur.
InfrastructureTermination	aucun changement de nom	Déprovisionne les AWS ressources invoquées via AWS TNB.
-	InfrastructureUpdate	Met à jour les AWS ressources mises en service pour le compte de l'utilisateur.

Nom de la tâche pour les déploiements commencés avant le 7 mars 2024	Nom de la tâche pour les déploiements commencés le 7 mars 2024 ou après	Description de la tâche
InventoryDeregistration	aucun changement de nom	Désenregistre les AWS ressources du TNB. AWS
-	InventoryRegistration	Enregistre les AWS ressources dans AWS TNB.
KubernetesClusterConfiguration	ClusterConfiguration	Configure le cluster Kubernetes et ajoute des rôles IAM supplémentaires à Amazon EKS, AuthMap comme défini dans le NSD.
NetworkServiceFinalization	aucun changement de nom	Finalise le service réseau et fournit une mise à jour de l'état de réussite ou d'échec.
NetworkServiceInstantiation	aucun changement de nom	Initialise le service réseau.
SelfManagedNodesConfiguration	aucun changement de nom	Démarre les nœuds autogérés avec le plan de contrôle Amazon EKS et Kubernetes.
-	ValidateNetworkServiceUpdate	Exécute les validations avant de mettre à jour une instance réseau.

Quotas de service pour AWS TNB

Les quotas de service, également appelés limites, correspondent au nombre maximal de ressources ou d'opérations de service pour votre AWS compte. Pour plus d'informations, consultez la section Quotas du service AWS dans le Référence générale d'Amazon Web Services.

Les quotas de service pour AWS TNB sont les suivants.

Nom	Par défaut	Ajusta	Description
Opérations de service réseau continues simultanées	Chaque Région prise en charge : 40	Oui	Le nombre maximum d'opérations de service réseau en cours simultanées dans une région.
Packages de fonctions	Chaque région prise en charge : 200	Oui	Le nombre maximum de packages de fonctions dans une région.
Packages réseau	Chaque Région prise en charge : 40	Oui	Le nombre maximum de packages réseau dans une région.
Instances de service réseau	Chaque région prise en charge : 800	Oui	Le nombre maximum d'instances de service réseau dans une région.

Historique du document pour le guide de l'utilisateur du AWS TNB

Le tableau suivant décrit les versions de documentation pour AWS TNB.

Modification	Description	Date
Mises à jour de la configuration réseau du groupe de nœuds Amazon EKS	Ajoutez et supprimez des sous-réseaux et des groupes de sécurité. Ajoutez, modifiez et supprimez ENIs du réseau. Pour plus d'informations, consultez la section Paramètres que vous pouvez mettre à jour.	10 septembre 2025
Ajout et suppression de groupes de nœuds Amazon EKS dans des clusters existants	AWS TNB prend désormais en charge l'ajout de nouveaux groupes de nœuds et la suppression de groupes de nœuds existants des clusters Amazon EKS. Pour plus d'informations, consultez la section Paramètres que vous pouvez mettre à jour.	4 juin 2025
Taille du volume racine	Vous pouvez spécifier la taille du volume racine Amazon EBS sous-jacent de vos nœuds de travail Amazon EKS via le `root_volume_size` champ du AWS fichier .Compute. EKSManagedNode et AWS.Compute. EKSSelfManagedNode Nœuds TOSCA.	19 mai 2025

Ressources de référence dans les scripts	Vous pouvez référencer les ressources créées par AWS TNB pour les configurer dans vos scripts Lifecycle Hook et vos scripts de données utilisateur.	2 mai 2025
La version 1.32 de Kubernetes est désormais prise en charge pour les nœuds Amazon EKS et les groupes de nœuds gérés.	AWS TNB prend en charge la version 1.32 de Kubernetes pour .Compute.EKS et .Compute.AWSAWS EKSManagedNœud.	24 avril 2025
La version 1.24 de Kubernetes n'est plus prise en charge pour les nœuds Amazon EKS et les groupes de nœuds gérés	AWS TNB ne prend plus en charge la version 1.24 de Kubernetes pour .Compute. EKS et .Compute.AWSAWS EKSManagedNœud.	17 avril 2025
AL2023 Support des AMI pour les nœuds gérés par Amazon EKS	AWS TNB prend en charge les types d' AL2023 AMI pour AWS.Compute. EKSManagedNœud.	17 avril 2025
La version 1.23 de Kubernetes n'est plus prise en charge pour les nœuds Amazon EKS et les groupes de nœuds gérés	AWS TNB ne prend plus en charge la version 1.23 de Kubernetes pour .Compute. EKS et .Compute.AWSAWS EKSManagedNœud.	4 avril 2025
L'ID AMI peut être mis à jour	Vous pouvez désormais mettre à jour le champ ami_id lors d'un appel d'`UpdateSolNetworkService` API.	31 mars 2025

La version 1.31 de Kubernetes est désormais prise en charge pour les nœuds Amazon EKS et les groupes de nœuds gérés.	AWS TNB prend en charge la version 1.31 de Kubernetes pour .Compute.EKS et .Compute.AWSAWS EKSManagedNœud.	18 février 2025
Version Kubernetes pour .Compute. AWS EKSManagedNœud	AWS TNB prend en charge les versions 1.23 à 1.30 de Kubernetes pour créer un groupe de nœuds géré par Amazon EKS.	28 janvier 2025
Version Kubernetes pour cluster	AWS TNB prend désormais en charge la version 1.30 de Kubernetes pour créer des clusters Amazon EKS.	19 août 2024

AWS TNB prend en charge une opération supplémentaire pour gérer le cycle de vie du réseau.	Vous pouvez mettre à jour une instance réseau instanciée ou précédemment mise à jour avec un nouveau package réseau et de nouvelles valeurs de paramètres. Consultez :
	• Opérations liées au cycle
	• Mettre à jour une instance réseau
	• AWS Exemple de rôle de service TNB :
	  • Ajoutez les actions Amazon EKS suivantes : `eks:UpdateAddon` `eks:UpdateClusterVersion` ,`eks:UpdateNodegroupConfig` ,`eks:UpdateNodegroupVersion` , `eks:DescribeUpdate`
	  • Ajoutez cette CloudFormation action : `cloudformation:UpdateStack`
	• Nouvelles tâches de déploiement :`InfrastructureUpdate` ,`InventoryRegistration` , `ValidateNetworkServiceUpdate`
	• Mises à jour de l'API : GetSolNetworkOpera
		30 juillet 2024

	tionListSolNetworkOperations , et UpdateSolNetworkInstance	
Nouvelle tâche et nouveaux noms de tâches pour les tâches existantes	Une nouvelle tâche est disponible. Depuis le 7 mars 2024, certaines tâches existantes portent de nouveaux noms pour des raisons de clarté.	7 mai 2024
Version Kubernetes pour cluster	AWS TNB prend désormais en charge la version 1.29 de Kubernetes pour créer des clusters Amazon EKS.	10 avril 2024
Support pour l'interface réseau `security_groups`	Vous pouvez associer des groupes de sécurité au nœud AWS.Networking.ENI.	2 avril 2024
Support pour le chiffrement du volume racine Amazon EBS	Vous pouvez activer le chiffrement Amazon EBS pour le volume racine Amazon EBS. Pour l'activer, ajoutez les propriétés dans le fichier AWS.Compute. EKSManagedNode ou AWS.Compute. EKSSelfManagedNodenœud.	2 avril 2024
Support pour le nœud `labels`	Vous pouvez associer des étiquettes de nœud à votre groupe de nœuds dans le fichier AWS.Compute. EKSManagedNode ou AWS.Compute. EKSSelfManagedNodenœud.	19 mars 2024

Support pour l'interface réseau `source_dest_check`	Vous pouvez indiquer si vous souhaitez activer ou désactiver la source/destination vérification de l'interface réseau via le nœud AWS.Networking.ENI.	25 janvier 2024
Support pour les instances Amazon EC2 avec données utilisateur personnalisées	Vous pouvez lancer des instances Amazon EC2 avec des données utilisateur personnalisées via le AWS fichier .Compute. UserData nœud.	16 janvier 2024
Support pour le groupe de sécurité	AWS TNB vous permet d'importer la AWS ressource Security Group.	8 janvier 2024
Description mise à jour de `network_interfaces`	Lorsque la `network_interfaces` propriété est incluse dans le fichier AWS.Compute. EKSManagedNode ou AWS.Compute. EKSSelfManagedNodenœud, AWS TNB obtient l'autorisation associée ENIs depuis la `multus_role` propriété si elle est disponible, ou depuis la `node_role` propriété.	18 décembre 2023
Support pour les clusters privés	AWS TNB prend désormais en charge les clusters privés. Pour indiquer un cluster privé, définissez la `access` propriété sur`PRIVATE`.	11 décembre 2023

Version Kubernetes pour cluster	AWS TNB prend désormais en charge la version 1.28 de Kubernetes pour créer des clusters Amazon EKS.	11 décembre 2023
AWS TNB soutient un groupe de placement	Ajout d'un groupe de placement pour les définitions `AWS.Compute.EKSManagedNode` des `AWS.Compute.EKSSelfManagedNode` nœuds et.	11 décembre 2023

AWS TNB ajoute le support pour IPv6	AWS TNB prend désormais en charge la création d'instances réseau avec IPv6 infrastructure. Vérifiez les nœuds AWS.Networking.VPC, .Networking.Subnet ,AWS.Networking.AWS InternetGateway, AWS.Réseautage. SecurityGroupIngressRule, AWS.Réseautage. SecurityGroupEgressRule, et AWS.compute.EKS pour les configurations. IPv6 Nous avons également ajouté les nœuds AWS.Networking. NATGatewayet AWS.Networking.Route pour la configuration. NAT64 Nous avons mis à jour le rôle de service AWS TNB et le rôle de service AWS TNB pour le groupe de nœuds Amazon EKS pour les IPv6 autorisations. Consultez les exemples de politiques relatives aux rôles de service.	16 novembre 2023
Autorisations ajoutées à la politique des rôles de service AWS TNB	Nous avons ajouté des autorisations à la politique des rôles de service AWS TNB pour Amazon S3 et pour CloudFormation permettre l'instanciation de l'infrastructure.	23 octobre 2023

AWS TNB est lancé dans d'autres régions	AWS TNB est désormais disponible dans les régions Asie-Pacifique (Séoul), Canada (centre), Europe (Espagne), Europe (Stockholm) et Amérique du Sud (São Paulo).	27 septembre 2023
Balises pour AWS.Compute.EKSSelfManagedNode	AWS TNB prend désormais en charge les balises pour la définition du `AWS.Compute.EKSSelfManagedNode` nœud.	22 août 2023
AWS TNB prend en charge les instances qui tirent parti IMDSv2	Lorsque vous lancez votre instance, vous devez utiliser IMDSv2.	14 août 2023
Autorisations mises à jour pour MultusRoleInlinePolicy	Cela inclut MultusRolInlinePolicy désormais l'`ec2:DeleteNetworkInterface` autorisation.	7 août 2023
Version Kubernetes pour cluster	AWS TNB prend désormais en charge les versions 1.27 de Kubernetes pour créer des clusters Amazon EKS.	25 juillet 2023
AWS.Computer.EKS.AuthRole	AWS TNB vous permet d'ajouter des rôles IAM au cluster Amazon EKS `aws-auth ConfigMap` afin que les utilisateurs puissent accéder au cluster Amazon EKS à l'aide d'un rôle IAM. AuthRole	19 juillet 2023

AWS TNB prend en charge les groupes de sécurité.	Ajout du AWS.Networking. SecurityGroup, AWS.Réseautage. SecurityGroupEgressRule, et AWS.Networking. SecurityGroupIngressRuleau modèle NSD.	18 juillet 2023
Version Kubernetes pour cluster	AWS TNB prend en charge les versions 1.22 à 1.26 de Kubernetes pour créer des clusters Amazon EKS. AWS TNB ne prend plus en charge les versions 1.21 de Kubernetes.	11 mai 2023
AWS.Calculer. EKSSelfManagedNode	Vous pouvez créer des nœuds de travail autogérés dans la région, dans les Zones AWS Locales et. AWS Outposts	29 mars 2023
Première version	Il s'agit de la première version du guide de l'utilisateur du AWS TNB.	21 février 2023

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.