aws

사용 설명서

AWS 통신 네트워크 빌더

AWS 통신 네트워크 빌더: 사용 설명서

Table of Contents

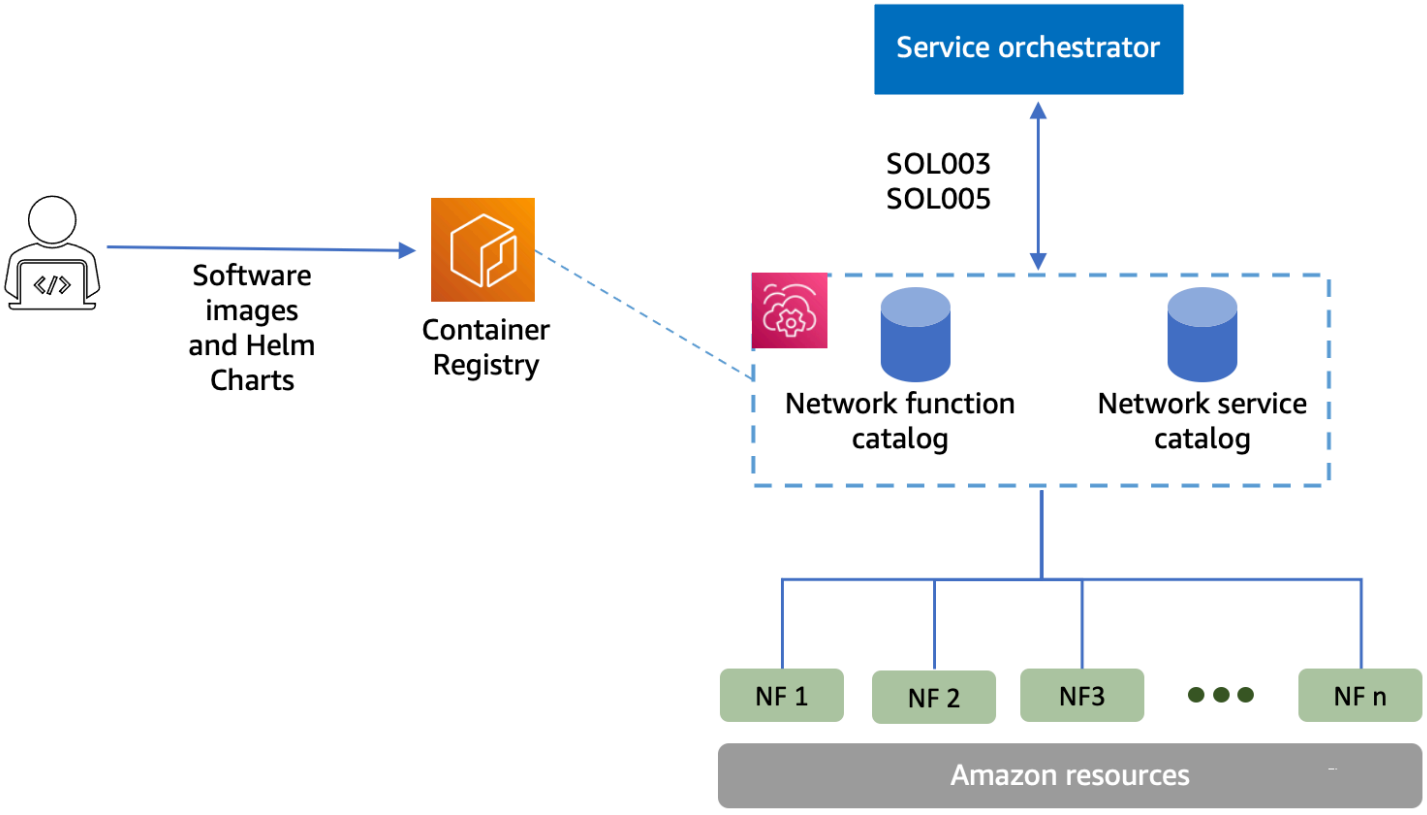
AWS Telco Network Builder란 무엇입니까?

AWS 통신 네트워크 빌더(AWS TNB)는 통신 서비스 제공업체(CSPs)에 인프라에서 AWS 5G 네트워크를 배포, 관리 및 확장할 수 있는 효율적인 방법을 제공하는 AWS 서비스입니다.

AWS TNB를 사용하면 자동화된 방식으로 네트워크 이미지를 AWS 클라우드 사용하여에 확장 가능하고 안전한 5G 네트워크를 배포할 수 있습니다. 새로운 기술을 배우거나, 사용할 컴퓨팅 서비스를 결정하거나, AWS 리소스를 프로비저닝하고 구성하는 방법을 알 필요가 없습니다.

대신 네트워크의 인프라를 설명하고 독립 소프트웨어 공급업체(ISV) 파트너의 네트워크 함수의 소프트웨어 이미지를 제공합니다. AWS TNB는 타사 서비스 오케스트레이터 및 AWS 서비스와 통합되어 필요한 AWS 인프라를 자동으로 프로비저닝하고, 컨테이너화된 네트워크 함수를 배포하고, 네트워킹 및 액세스 관리를 구성하여 완전히 작동하는 네트워크 서비스를 생성합니다.

다음 다이어그램은 유럽통신표준연구소(ETSI) 기반 표준 인터페이스를 사용하여 네트워크 함수를 배포하기 위한 AWS TNB와 서비스 오케스트레이터 간의 논리적 통합을 보여줍니다.



주제

- 를 처음 사용하시 AWS나요?
- AWS TNB는 누구를 위한 것입니까?

를 처음 사용하시 AWS나요?

AWS 제품 및 서비스를 처음 사용하는 경우 다음 리소스로 자세히 알아보세요.

- 소개 AWS
- 시작하기 AWS

AWS TNB는 누구를 위한 것입니까?

AWS TNB는 비용 효율성을 활용하려는 CSPs를 위한 것으로, 민첩성, 및 설계할 사용자 지정 스크립트 및 구성을 작성하고 유지 관리하지 않고도에서 AWS 클라우드 제공하는 탄력성, 배포, 및 네트워크 서비스 관리. AWS TNB는 필요한 AWS 인프라를 자동으로 프로비저닝합니다. 는 컨테이너화된 네트워크 함수를 배포합니다. 및는 네트워킹 및 액세스 관리를 구성하여 CSP 정의 네트워크 서비스 설명자를 기반으로 완전 운영 네트워크 서비스를 생성합니다. 및 CSP가 배포하려는 네트워크 함수.

AWS TNB 기능

다음은 CSP가 AWS TNB를 사용하려는 몇 가지 이유입니다.

태스크 단순화

새 서비스 배포, 네트워크 함수 업데이트 및 업그레이드, 네트워크 인프라 토폴로지 변경 등 네트워크 운영의 효율성을 높입니다.

오케스트레이터와 통합

AWS TNB는 ETSI를 준수하는 인기 있는 타사 서비스 오케스트레이터와 통합됩니다.

규모 조정

트래픽 수요에 맞게 기본 AWS 리소스를 확장하고, 네트워크 함수 업데이트를 더 효율적으로 수행하고, 네트워크 인프라 토폴로지 변경을 롤아웃하고, 새로운 5G 서비스의 배포 시간을 며칠에서 몇 시간으로 줄이도록 AWS TNB를 구성할 수 있습니다.

AWS 리소스 검사 및 모니터링

AWS TNB를 사용하면 Amazon VPC, Amazon EC2, Amazon EKS와 같은 단일 대시보드에서 네트워크를 지원하는 AWS 리소스를 검사하고 모니터링할 수 있습니다.

서비스 템플릿 지원

AWS TNB를 사용하면 모든 통신 워크로드(RAN, Core, IMS)에 대한 서비스 템플릿을 생성할 수 있습니다. 새 서비스 정의를 생성하거나, 기존 템플릿을 재사용하거나, 지속적 통합 및 지속적 전달(CI/CD) 파이프라인과 통합하여 새 정의를 게시할 수 있습니다.

네트워크 배포의 변경 사항 추적

네트워크 함수 배포의 기본 구성을 변경할 때(예: Amazon EC2 인스턴스 유형의 인스턴스 유형 변경), 반복 가능하고 조정 가능한 방식으로 변경 사항을 추적할 수 있습니다. 이를 수동으로 수행하려면 네트워크 상태를 관리하고, 리소스를 생성 및 삭제하고, 필요한 변경 순서에 주의를 기울여야 합니다. AWS TNB를 사용하여 네트워크 함수의 수명 주기를 관리하는 경우 네트워크 함수를 설명하는 네트워크 서비스 설명자만 변경합니다. 그러면 AWS TNB는 필요한 사항을 올바른 순서로 자동으로 변경합니다.

네트워크 함수 수명 주기 간소화

네트워크 함수의 첫 번째 버전과 모든 후속 버전을 관리하고 업그레이드 시기를 지정할 수 있습니다. 또한 RAN, Core, IMS 및 네트워크 애플리케이션을 동일한 방식으로 관리할 수 있습니다.

AWS TNB 액세스

다음 인터페이스 중 하나를 사용하여 AWS TNB 리소스를 생성, 액세스 및 관리할 수 있습니다.

- AWS TNB 콘솔 - 네트워크 관리를 위한 웹 인터페이스를 제공합니다.
- AWS TNB API - AWS TNB 작업을 수행하기 위한 RESTful API를 제공합니다. 자세한 내용은 AWS TNB API 참조를 참조하세요.
- AWS Command Line Interface (AWS CLI) - AWS TNB를 포함한 광범위한 AWS 서비스에 대한 명령을 제공합니다. 이는 Windows, macOS, Linux에서 지원됩니다. 자세한 내용은 AWS Command Line Interface 사용 설명서를 참조하십시오.
- AWS SDKs- 언어별 APIs 제공하고 많은 연결 세부 정보를 작성합니다. 서명 계산, 요청 재시도 처리 및 오류 처리가 여기에 포함됩니다. 자세한 내용은 AWS SDK를 참조하십시오.

AWS TNB 요금

AWS TNB는 CSPs에서 통신 네트워크의 배포 및 관리를 자동화하는 데 도움이 됩니다 AWS. AWS TNB를 사용할 때 다음 두 가지 차원에 대한 비용을 지불합니다.

- 관리형 네트워크 함수 항목(MNFI) 시간
- API 요청 수

또한 AWS TNB와 함께 다른 AWS 서비스를 사용할 때 추가 요금이 발생합니다. 자세한 내용은 AWS TNB 요금을 참조하세요.

청구 요금은 AWS 결제 및 비용 관리 콘솔의 결제 및 비용 관리 대시보드에서 확인할 수 있습니다. 청구서에는 요금 내역을 더 자세하게 확인할 수 있는 사용 보고서 링크가 포함됩니다. AWS 계정 결제에 대한 자세한 내용은 AWS 계정 결제를 참조하세요.

AWS 결제, 계정 및 이벤트에 대해 궁금한 점이 있으면 AWS Support에 문의하세요.

AWS Trusted Advisor 는 AWS 환경의 비용, 보안 및 성능을 최적화하는 데 사용할 수 있는 서비스입니다. 자세한 내용은 AWS Trusted Advisor를 참조하세요.

다음 단계

AWS TNB를 시작하는 방법에 대한 자세한 내용은 다음 주제를 참조하세요.

- AWS TNB 설정 – 사전 조건 단계를 완료합니다.
- AWS TNB 시작하기 – CU(Centralized Unit), AMF(Access and Mobility Management Function), UPF(User Plane Function) 또는 완전한 5G 코어와 같은 첫 번째 네트워크 함수를 배포합니다.

AWS TNB 작동 방식

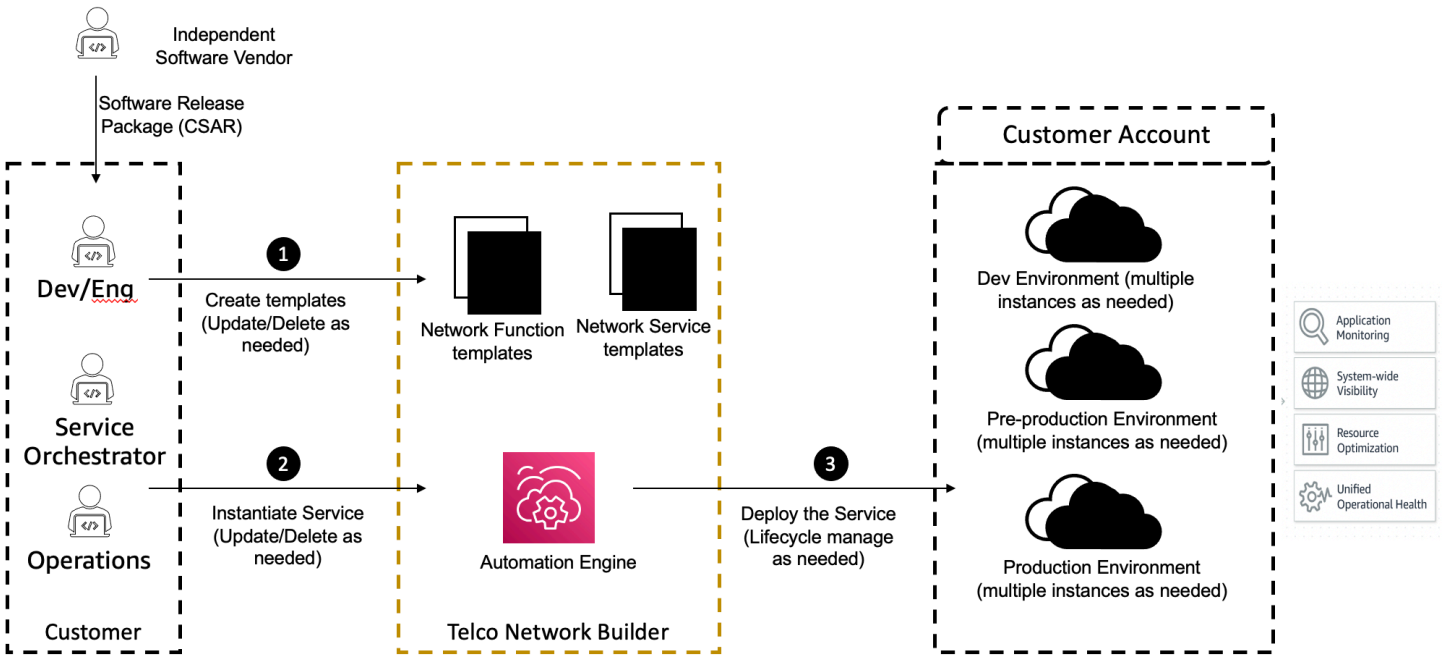
AWS TNB는 표준화된 end-to-end 오케스트레이터 및 AWS 리소스와 통합되어 전체 5G 네트워크를 운영합니다.

AWS TNB를 사용하면 네트워크 함수 패키지와 네트워크 서비스 설명자(NSDs)를 수집할 수 있으며 네트워크를 운영할 수 있는 자동화 엔진을 제공합니다. end-to-end 오케스트레이터를 사용하고 AWS TNB APIs와 통합하거나 AWS TNB SDKs 사용하여 자체 자동화 흐름을 구축할 수 있습니다. 자세한 내용은 AWS TNB 아키텍처 단원을 참조하십시오.

주제

AWS TNB 아키텍처

AWS TNB는 AWS Management Console, AWS CLI, AWS TNB REST API 및 SDKs 기능을 제공합니다. 이를 통해 엔지니어링 팀, 운영 팀, 프로그래밍 시스템 팀의 구성원 등 다양한 CSP 담당자가 AWS TNB를 활용할 수 있습니다. 네트워크 함수 패키지를 생성하여 CSAR(Cloud Service Archive) 파일로 업로드합니다. CSAR 파일에는 차트 Helm, 소프트웨어 이미지, 네트워크 함수 설명자(NFD)가 포함되어 있습니다. 템플릿을 사용하여 해당 패키지의 여러 구성을 반복적으로 배포할 수 있으며 배포하려는 인프라 및 네트워크 함수를 정의하는 네트워크 서비스 템플릿을 생성할 수 있습니다. 파라미터 재정의를 사용하여 여러 위치에 다양한 구성을 배포할 수 있습니다. 그런 다음 템플릿을 사용하여 네트워크를 인스턴스화하고 AWS 인프라에 네트워크 함수를 배포할 수 있습니다. AWS TNB는 배포에 대한 가시성을 제공합니다.



와 통합 AWS 서비스

5G 네트워크는 수천 개의 Kubernetes 클러스터에 배포된 상호 연결된 컨테이너화된 네트워크 함수 세트로 구성됩니다. AWS TNB는 다음을 AWS 서비스 통신별 APIs로 통합하여 완벽하게 작동하는 네트워크 서비스를 생성합니다.

- 독립 소프트웨어 개발 판매 회사(ISV) 네트워크 함수 아티팩트를 저장하기 위한 Amazon Elastic Container Registry(Amazon ECR)
- Amazon Elastic Kubernetes Service (Amazon EKS)를 사용하여 클러스터를 설정합니다.
- 네트워킹 구성을 위한 Amazon VPC
- 를 사용하는 보안 그룹 CloudFormation.
- AWS CodePipeline - AWS 리전, AWS 로컬 영역 및의 배포 대상. AWS Outposts
- 역할을 정의하기 위한 IAM
- AWS Organizations AWS TNB APIs.
- Health Dashboard 및 AWS CloudTrail 를 사용하여 상태를 모니터링하고 지표를 게시합니다.

AWS TNB 리소스 할당량

AWS 계정 에는 각에 대해 이전에 제한이라고 하는 기본 할당량이 있습니다 AWS 서비스. 달리 명시되지 않는 한 각 할당량은에 고유합니다 AWS 리전. 일부 할당량에 대한 증가를 요청할 수 있으며 모든 할당량에 대한 증가를 요청할 수 없습니다.

AWS TNB에 대한 할당량을 보려면 Service Quotas 콘솔을 엽니다. 탐색 창에서 AWS 서비스를 선택하고 AWS TNB를 선택합니다.

할당량 증가를 요청하려면 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요.

AWS 계정 에는 AWS TNB와 관련된 다음과 같은 할당량이 있습니다.

리소스 할당량	설명	기본값	조정 가능?
네트워크 서비스 인스턴스	한 리전의 최대 네트워크 서비스 인스턴스 수입니다.	800	예
동시 진행 중인 네트워크 서비스 작업	한 리전에서 네트워크 서비스를 동시에 실행할 수 있는 최대 개수	40	예
네트워크 패키지	한 리전의 최대 네트워크 패키지 수입니다.	40	예
함수 패키지	한 리전의 최대 함수 패키지 수입니다.	200	예

AWS TNB 개념

이 주제에서는 AWS TNB 사용을 시작하는 데 도움이 되는 필수 개념을 설명합니다.

내용

네트워크 함수의 수명 주기

AWS TNB는 네트워크 함수의 수명 주기 전반에 걸쳐 도움이 됩니다. 네트워크 함수 수명 주기에는 다음과 같은 단계와 활동이 포함됩니다.

계획

1. 배포할 네트워크 함수를 식별하여 네트워크를 계획합니다.
2. 네트워크 함수 소프트웨어 이미지를 컨테이너 이미지 리포지토리에 저장합니다.
3. 배포 또는 업그레이드할 CSAR 패키지를 생성합니다.
4. AWS TNB를 사용하여 네트워크 함수(예: CU AMF 및 UPF)를 정의하는 CSAR 패키지를 업로드하고, 새로운 네트워크 함수 소프트웨어 이미지 또는 고객 스크립트를 사용할 수 있으므로 CSAR 패키지의 새 버전을 생성하는 데 도움이 되는 지속적 통합 및 지속적 전달(CI/CD) 파이프라인과 통합합니다.

구성

1. 컴퓨팅 유형, 네트워크 함수 버전, IP 정보, 리소스 이름 등 배포에 필요한 정보를 파악합니다.
2. 이 정보를 사용하여 네트워크 서비스 설명자(NSD)를 생성합니다.
3. 네트워크 함수를 정의하는 NSD와 네트워크 함수의 인스턴스화에 필요한 리소스를 수집합니다.

인스턴스화

1. 네트워크 함수에 필요한 인프라를 생성합니다.
2. NSD에 정의된 대로 네트워크 함수를 인스턴스화(또는 프로비저닝)하고 트래픽 전달을 시작합니다.

3. 자산을 검증합니다.

프로덕션

네트워크 함수의 수명 주기 동안 다음과 같은 프로덕션 작업을 완료하게 됩니다.

- 네트워크 함수 구성을 업데이트하십시오(예: 배포된 네트워크 함수의 값 업데이트).
- 새 네트워크 패키지와 파라미터 값으로 네트워크 인스턴스를 업데이트합니다. 예를 들어 네트워크 패키지에서 Amazon EKS `version` 파라미터를 업데이트합니다.

표준화된 인터페이스 사용

AWS TNB는 유럽 통신 표준 연구소(ETSI) 준수 서비스 오케스트레이터와 통합되어 네트워크 서비스 배포를 간소화할 수 있습니다. 서비스 오케스트레이터는 AWS TNB SDKs, CLI 또는 APIs를 사용하여 네트워크 함수를 인스턴스화하거나 새 버전으로 업그레이드하는 등의 작업을 시작할 수 있습니다.

AWS TNB는 다음 사양을 지원합니다.

사양	릴리스	설명
ETSI SOL001	v3.6.1	TOSCA 기반 네트워크 함수 설명자를 허용하기 위한 표준을 정의합니다.
ETSI SOL002	v3.6.1	네트워크 함수 관리 관련 모델을 정의합니다.
ETSI SOL003	v3.6.1	네트워크 함수 수명 주기 관리에 대한 표준을 정의합니다.
ETSI SOL004	v3.6.1	네트워크 함수 패키지의 CSAR 표준을 정의합니다.
ETSI SOL005	v3.6.1	네트워크 서비스 패키지 및 네트워크 서비스 수명 주기 관리에 대한 표준을 정의합니다.
ETSI SOL007	v3.5.1	TOSCA 기반 네트워크 서비스 설명자를 허용하기 위한 표준을 정의합니다.

함수 패키지

AWS TNB를 사용하면 ETSI SOL001/SOL004를 준수하는 함수 패키지를 함수 카탈로그에 저장할 수 있습니다. 그런 다음 가상 네트워크 함수를 설명하는 아티팩트가 포함된 Cloud Service Archive(CSAR) 패키지를 업로드할 수 있습니다.

- 가상 네트워크 함수 설명자 - 패키지 온보딩 및 가상 네트워크 함수 관리를 위한 메타데이터를 정의합니다. 이 파일 `vnfd.yaml`의 이름을 지정해야 합니다.
- 소프트웨어 이미지 - 가상 네트워크 함수 컨테이너 이미지를 참조합니다. Amazon Elastic Container Registry(Amazon ECR)는 가상 네트워크 함수 이미지 리포지토리 역할을 할 수 있습니다.
- 추가 파일 - 스크립트 및 차트 Helm과 같은 가상 네트워크 함수를 관리하는 데 사용합니다.

CSAR은 OASIS TOSCA 표준에서 정의한 패키지이며 OASIS TOSCA YAML 사양을 준수하는 네트워크/서비스 설명자를 포함합니다. 필수 YAML 사양에 대한 자세한 내용은 섹션을 참조하세요AWS TNB에 대한 TOSCA 참조.

다음은 가상 네트워크 함수 설명자의 예입니다.

```
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"
```

네트워크 패키지

네트워크 패키지는 CSAR(Cloud Service Archive) 형식의 `.zip` 파일입니다. 배포하려는 함수 패키지와 배포하려는 AWS 인프라를 정의합니다.

네트워크 패키지에는 다음 파일이 포함되어 있습니다.

- ETSI SOL007에 설명된 TOSCA 형식의 네트워크 설명자 파일(`nsd.yaml`)입니다.

  `nsd.yaml` 파일에는 설명자 ID가 있는 업로드된 함수 패키지에 대한 참조가 포함되어 있습니다. IDs
- 사용자 데이터 스크립트가 있는 경우
- 수명 주기 후크 스크립트가 있는 경우
- 플러그인의 `values.yaml` 구성 파일이 있는 경우

AWS TNB는 네트워크, 서비스 및 함수와 같은 리소스를 TOSCA 언어로 모델링하기 위한 ETSI 표준을 지원합니다. AWS TNB를 사용하면 ETSI 호환 서비스 오케스트레이터가 이해할 수 있는 방식으로 모델링 AWS 서비스 하여 보다 효율적으로 사용할 수 있습니다.

AWS TNB에 대한 네트워크 서비스 설명자

네트워크 서비스 설명자(NSD)는 TOSCA 표준을 사용하여 배포하려는 네트워크 함수와 네트워크 함수를 배포하려는 AWS 인프라를 설명하는 네트워크 패키지의 `.yaml` 파일입니다. NSD를 정의하고 기본 리소스 및 네트워크 수명 주기 작업을 구성하려면 AWS TNB에서 지원하는 NSD TOSCA 스키마를 이해해야 합니다.

NSD 파일은 다음과 같은 부분으로 나뉩니다.

1. TOSCA 정의 버전 – NSD YAML 파일의 첫 번째 줄이며 다음 예제와 같은 버전 정보를 포함합니다.

   ```
   tosca_definitions_version: tnb_simple_yaml_1_0
   ```
2. VNFD – NSD에는 수명 주기 작업을 수행할 네트워크 함수의 정의가 포함되어 있습니다. 각 네트워크 함수는 다음과 같은 값으로 식별되어야 합니다.
   - `descriptor_id`의 고유한 ID. ID는 네트워크 함수 CSAR 패키지의 ID와 일치해야 합니다.
   - `namespace`의 고유한 이름. 다음 예제와 같이 NSD YAML 파일 전체에서 더 쉽게 참조하려면 이름을 고유한 ID와 연결해야 합니다.

   ```
   vnfds:
   ```

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

3. 토폴로지 템플릿 - 배포할 리소스, 네트워크 함수 배포 및 수명 주기 후크와 같은 모든 사용자 지정 스크립트를 정의합니다. 방법은 다음 예제와 같습니다.

```
topology_template:

  node_templates:

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

4. 추가 노드 - 모델링된 각 리소스에는 속성 및 요구 사항에 대한 섹션이 있습니다. 속성은 버전과 같은 리소스의 선택적 또는 필수 속성을 설명합니다. 요구 사항은 인수로 제공되어야 하는 종속성을 설명합니다. 예를 들어, Amazon EKS 노드 그룹 리소스를 생성하려면 Amazon EKS 클러스터 내에 생성해야 합니다. 방법은 다음 예제와 같습니다.

```
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
```

NSD 예

다음은 모델링 방법을 보여주는 NSD 조각입니다 AWS 서비스. 네트워크 함수는 Kubernetes 버전 1.27이 설치된 Amazon EKS 클러스터에 배포됩니다. 애플리케이션용 서브넷은 Subnet01과 Subnet02입니다. 그런 다음 Amazon Machine Image(AMI), 인스턴스 유형 및 자동 크기 조정 구성을 사용하여 애플리케이션의 노드 그룹을 정의할 수 있습니다.

```
tosca_definitions_version: tnb_simple_yaml_1_0

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

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

```
        max_size: 6
  requirements:
    cluster: SampleNFEKS
    subnets:
      - Subnet01
    network_interfaces:
      - ENI01
      - ENI02
```

AWS TNB에 대한 관리 및 작업

AWS TNB를 사용하면 ETSI SOL003 및 SOL005에 따라 표준화된 관리 작업을 사용하여 네트워크를 관리할 수 있습니다. AWS TNB APIs를 사용하여 다음과 같은 수명 주기 작업을 수행할 수 있습니다.

- 네트워크 함수 인스턴스화
- 네트워크 함수 종료
- Helm 배포를 재정의하도록 네트워크 함수를 업데이트
- 인스턴스화되거나 업데이트된 네트워크 인스턴스를 새 네트워크 패키지 및 파라미터 값으로 업데이트합니다.
- 네트워크 함수 패키지 버전 관리
- NSD의 버전 관리
- 배포된 네트워크 함수에 대한 정보 검색

AWS TNB 설정

이 주제에 설명된 작업을 완료하여 AWS TNB를 설정합니다.

업무

- 에 가입 AWS 계정
- 관리자 액세스 권한이 있는 사용자 생성
- AWS 리전 선택
- 서비스 엔드포인트를 기록해 둡니다.
- (선택 사항) 설치 AWS CLI
- AWS TNB 역할 설정

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

AWS 리전 선택

AWS TNB에 사용할 수 있는 리전 목록을 보려면 AWS 리전 서비스 목록을 참조하세요. 프로그래밍 방식 액세스를 위한 엔드포인트 목록을 보려면 AWS 일반 참조의 AWS TNB 엔드드포인트를 참조하세요.

서비스 엔드포인트를 기록해 둡니다.

AWS 서비스에 프로그래밍 방식으로 연결하려면 엔드포인트를 사용합니다. 일부 AWS 서비스는 표준 AWS 엔드포인트 외에도 선택한 리전에서 FIPS 엔드포인트를 제공합니다. 자세한 내용은 AWS 서비스 엔드포인트를 참조하세요.

리전 이름	리전	엔드포인트	프로토콜	
미국 동부(버지니아 북부)	us-east-1	tnb.us-east-1.amazonaws.com	HTTPS	
미국 서부(오레곤)	us-west-2	tnb.us-west-2.amazonaws.com	HTTPS	
아시아 태평양(서울)	ap-northeast-2	tnb.ap-northeast-2.amazonaws.com	HTTPS	
아시아 태평양(시드니)	ap-southeast-2	tnb.ap-southeast-2.amazonaws.com	HTTPS	
캐나다(중부)	ca-central-1	tnb.ca-central-1.amazonaws.com	HTTPS	
유럽(프랑크푸르트)	eu-central-1	tnb.eu-central-1.amazonaws.com	HTTPS	
유럽(파리)	eu-west-3	tnb.eu-west-3.amazonaws.com	HTTPS	

리전 이름	리전	엔드포인트	프로토콜	
유럽(스페인)	eu-south-2	tnb.eu-south-2.amazonaws.com	HTTPS	
유럽(스톡홀름)	eu-north-1	tnb.eu-north-1.amazonaws.com	HTTPS	
남아메리카(상파울루)	sa-east-1	tnb.sa-east-1.amazonaws.com	HTTPS	

(선택 사항) 설치 AWS CLI

AWS Command Line Interface (AWS CLI)는 다양한 AWS 제품에 대한 명령을 제공하며 Windows, macOS 및 Linux에서 지원됩니다. 를 사용하여 AWS TNB에 액세스할 수 있습니다 AWS CLI. 시작하려면 AWS Command Line Interface 사용 설명서를 참조하십시오. AWS TNB 명령에 대한 자세한 내용은 AWS CLI 명령 참조의 tnb를 참조하세요.

AWS TNB 역할 설정

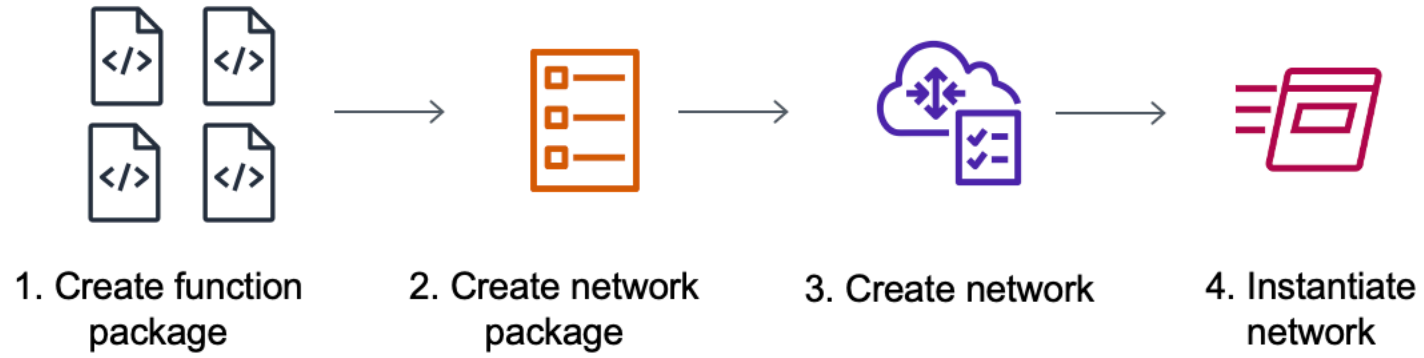
AWS TNB 솔루션의 여러 부분을 관리하려면 IAM 서비스 역할을 생성해야 합니다. AWS TNB 서비스 역할은 사용자를 대신하여 및 다양한 컴퓨팅 AWS CloudFormation AWS CodeBuild및 스토리지 AWS 서비스와 같은 다른 서비스에 API를 호출하여 배포를 위한 리소스를 인스턴스화하고 관리할 수 있습니다.

AWS TNB 서비스 역할에 대한 자세한 내용은 섹션을 참조하세요AWS TNB의 ID 및 액세스 관리.

AWS TNB 시작하기

이 자습서에서는 AWS TNB를 사용하여 중앙 집중식 단위(CU), 액세스 및 이동 관리 기능(AMF) 또는 5G 사용자 평면 기능(UPF)과 같은 네트워크 함수를 배포하는 방법을 보여줍니다.

다음 다이어그램은 배포 프로세스를 보여줍니다.



업무

- 사전 조건
- 함수 패키지 생성
- 네트워크 패키지 생성
- 네트워크 인스턴스 생성 및 인스턴스화
- 정리

사전 조건

성공적인 배포를 수행하려면 먼저 다음이 있어야 합니다.

- AWS Business Support 플랜.
- IAM 역할을 통한 권한.
- ETSI SOL001/SOL004를 준수하는 NF(Network Function) 패키지입니다.
- ETSI SOL007을 준수하는 Network Service Descriptor(NSD) 템플릿입니다.

AWS TNB GitHub용 샘플 패키지 사이트에서 샘플 함수 패키지 또는 네트워크 패키지를 사용할 수 있습니다. GitHub

함수 패키지 생성

네트워크 함수 패키지는 클라우드 서비스 아카이브(CSAR) 파일입니다. CSAR 파일에는 차트 Helm, 소프트웨어 이미지, 네트워크 함수 설명자(NFD)가 포함되어 있습니다.

함수 패키지를 생성하려면

1. https://console.aws.amazon.com/tnb/:///https://https://https:///:///https AWS ///:///https:///:///https:///:// https:///://
2. 탐색 창에서 함수 패키지를 선택합니다.
3. 함수 패키지 생성을 선택합니다.
4. 함수 패키지 업로드에서 파일 선택을 선택하고 각 CSAR 패키지를 `.zip` 파일로 업로드합니다. 최대 10개의 파일을 업로드할 수 있습니다.
5. (선택 사항) 태그에서 새 태그 추가를 선택하고 키와 값을 입력합니다. 태그를 사용하여 리소스를 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.
6. Next(다음)를 선택합니다.
7. 패키지 세부 정보를 검토한 다음 함수 패키지 생성을 선택합니다.

네트워크 패키지 생성

네트워크 패키지는 배포하려는 네트워크 함수와 이를 카탈로그에 배포하는 방법을 지정합니다.

네트워크 패키지를 생성하려면

1. 탐색 창에서 네트워크 패키지를 선택합니다.
2. 네트워크 패키지 생성을 선택합니다.
3. 네트워크 패키지 업로드에서 파일 선택을 선택하고 각 NSD를 `.zip` 파일로 업로드합니다. 최대 10개의 파일을 업로드할 수 있습니다.
4. (선택 사항) 태그에서 새 태그 추가를 선택하고 키와 값을 입력합니다. 태그를 사용하여 리소스를 검색 및 필터링하거나 AWS 비용을 추적할 수 있습니다.
5. Next(다음)를 선택합니다.
6. 네트워크 패키지 생성을 선택합니다.

네트워크 인스턴스 생성 및 인스턴스화

네트워크 인스턴스는 배포할 수 있는 AWS TNB에서 생성된 단일 네트워크입니다. 네트워크 인스턴스를 생성하고 인스턴스화해야 합니다. 네트워크 인스턴스를 인스턴스화하면 AWS TNB는 필요한 AWS 인프라를 프로비저닝하고, 컨테이너화된 네트워크 함수를 배포하고, 네트워킹 및 액세스 관리를 구성하여 완전히 작동하는 네트워크 서비스를 생성합니다.

네트워크 인스턴스를 생성 및 인스턴스화하려면

1. 탐색 창에서 네트워크를 선택합니다.
2. 네트워크 인스턴스 생성을 선택합니다.
3. 네트워크의 이름과 설명을 입력하고 다음을 선택합니다.
4. 네트워크 패키지를 선택합니다. 세부 정보를 확인하고 다음을 선택합니다.
5. 네트워크 인스턴스 생성을 선택합니다. 초기 상태는 `Created`입니다.

   상태의 새 네트워크 인스턴스를 보여주는 네트워크 페이지가 나타납니다`Not instantiated`.
6. 네트워크 인스턴스를 선택하고 작업 및 인스턴스화를 선택합니다.

   네트워크 인스턴스화 페이지가 나타납니다.
7. 세부 정보를 검토하고 파라미터 값을 업데이트합니다. 파라미터 값 업데이트는이 네트워크 인스턴스에만 적용됩니다. NSD 및 VNFD 패키지의 파라미터는 변경되지 않습니다.
8. 네트워크 인스턴스화를 선택합니다.

   배포 상태 페이지가 나타납니다.
9. 새로 고침 아이콘을 사용하여 네트워크 인스턴스의 배포 상태를 추적합니다. 배포 작업 섹션에서 자동 새로 고침을 활성화하여 각 작업의 진행 상황을 추적할 수도 있습니다.

정리

이제이 자습서에서 생성한 리소스를 삭제할 수 있습니다.

리소스를 정리하려면

1. 탐색 창에서 네트워크를 선택합니다.
2. 네트워크 ID를 선택한 다음 종료를 선택합니다.
3. 확인 메시지가 나타나면 네트워크 ID를 입력한 다음 종료를 선택합니다.

4. 새로 고침 아이콘을 사용하여 네트워크 인스턴스의 상태를 추적합니다.
5. (선택 사항) 네트워크를 선택한 후 삭제를 선택합니다.

AWS TNB용 함수 패키지

함수 패키지는 TOSCA 표준을 사용하여 네트워크에서 네트워크 함수가 실행되는 방식을 설명하는 함수 패키지 설명자 및 네트워크 함수(ETSI 표준 통신 애플리케이션)를 포함하는 CSAR(Cloud Service Archive) 형식의.zip 파일입니다.

작업

- AWS TNB에서 함수 패키지 생성
- AWS TNB에서 함수 패키지 보기
- AWS TNB에서 함수 패키지 다운로드
- AWS TNB에서 함수 패키지 삭제

AWS TNB에서 함수 패키지 생성

AWS TNB 네트워크 함수 카탈로그에서 함수 패키지를 생성하는 방법을 알아봅니다. 함수 패키지 생성은 AWS TNB에서 네트워크를 생성하는 첫 번째 단계입니다. 함수 패키지를 업로드한 후 네트워크 패키지를 생성할 수 있습니다.

Console

콘솔을 사용하여 함수를 생성하려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 함수 패키지를 선택합니다.
3. 함수 패키지 생성을 선택합니다.
4. 파일 선택을 선택하고 각 CSAR 패키지를 `.zip` 파일로 업로드합니다. 최대 10개의 파일을 업로드할 수 있습니다.
5. 다음을 선택합니다.
6. 패키지 세부 정보를 검토합니다.
7. 함수 패키지 생성을 선택합니다.

AWS CLI

를 사용하여 함수 패키지를 생성하려면 AWS CLI

1. create-sol-function-package 명령을 사용하여 새 함수 패키지를 생성합니다.

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

2. put-sol-function-package-content 명령을 사용하여 함수 패키지 콘텐츠를 업로드합니다. 예제:

```
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
```

AWS TNB에서 함수 패키지 보기

함수 패키지의 내용을 보는 방법을 알아보세요.

Console

콘솔을 사용하여 함수 패키지를 보려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 함수 패키지를 선택합니다.
3. 검색창을 사용하여 함수 패키지를 찾습니다.

AWS CLI

를 사용하여 함수 패키지를 보려면 AWS CLI

1. list-sol-function-packages 명령을 사용하여 함수 패키지를 나열합니다.

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

2. get-sol-function-package 명령을 사용하여 함수 패키지에 대한 세부 정보를 확인합니다.

```
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
```

AWS TNB에서 함수 패키지 다운로드

AWS TNB 네트워크 함수 카탈로그에서 함수 패키지를 다운로드하는 방법을 알아봅니다.

Console

콘솔을 사용하여 함수 패키지를 다운로드하려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 콘솔 왼쪽의 탐색 창에서 함수 패키지를 선택합니다.
3. 검색창을 사용하여 함수 패키지를 찾습니다.
4. 함수 패키지를 선택합니다.
5. 작업에서 다운로드를 선택합니다.

AWS CLI

를 사용하여 함수 패키지를 다운로드하려면 AWS CLI

get-sol-function-package-content 명령을 사용하여 함수 패키지를 다운로드합니다.

```
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
```

AWS TNB에서 함수 패키지 삭제

AWS TNB 네트워크 함수 카탈로그에서 함수 패키지를 삭제하는 방법을 알아봅니다. 함수 패키지를 삭제하려면 패키지가 비활성화 상태여야 합니다.

Console

콘솔을 사용하여 함수 패키지를 삭제하려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 함수 패키지를 선택합니다.
3. 검색창을 사용하여 함수 패키지를 찾습니다.
4. 함수 패키지를 선택합니다.
5. 작업, 비활성화를 선택합니다.
6. 작업, 삭제를 선택합니다.

AWS CLI

를 사용하여 함수 패키지를 삭제하려면 AWS CLI

1. update-sol-function-package 명령을 사용하여 함수 패키지를 비활성화합니다.

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

2. delete-sol-function-package 명령을 사용하여 함수 패키지를 삭제합니다.

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

AWS TNB용 네트워크 패키지

네트워크 패키지는 CSAR(Cloud Service Archive) 형식의 .zip 파일입니다. 배포하려는 함수 패키지와 배포하려는 AWS 인프라를 정의합니다.

네트워크 패키지에는 다음 파일이 포함되어 있습니다.

- ETSI SOL007에 설명된 TOSCA 형식의 네트워크 설명자 파일(`nsd.yaml`)입니다.

  `nsd.yaml` 파일에는 설명자 ID가 있는 업로드된 함수 패키지에 대한 참조가 포함되어 있습니다. IDs
- 사용자 데이터 스크립트가 있는 경우
- 수명 주기 후크 스크립트가 있는 경우
- 플러그인의 `values.yaml` 구성 파일이 있는 경우

작업

- AWS TNB에서 네트워크 패키지 생성
- AWS TNB에서 네트워크 패키지 보기
- AWS TNB에서 네트워크 패키지 다운로드
- AWS TNB에서 네트워크 패키지 삭제

AWS TNB에서 네트워크 패키지 생성

네트워크 패키지는 네트워크 서비스 설명자(NSD) 파일(필수)과 사용자의 필요에 맞는 스크립트 등의 추가 파일(선택 사항)로 구성됩니다. 예를 들어 네트워크 패키지에 함수 패키지가 여러 개 있는 경우 NSD를 사용하여 특정 VPC, 서브넷 또는 Amazon EKS 클러스터에서 실행해야 하는 네트워크 함수를 정의할 수 있습니다.

함수 패키지를 생성한 후 네트워크 패키지를 생성하세요. 네트워크 패키지를 생성한 후에는 네트워크 인스턴스를 생성해야 합니다.

Console

콘솔을 사용하여 네트워크 패키지를 생성하려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.

2. 탐색 창에서 네트워크 패키지를 선택합니다.
3. 네트워크 패키지 생성을 선택합니다.
4. 파일 선택을 선택하고 각 NSD를 `.zip` 파일로 업로드합니다. 최대 10개의 파일을 업로드할 수 있습니다.
5. 다음을 선택합니다.
6. 패키지 세부 정보를 검토합니다.
7. 네트워크 패키지 생성을 선택합니다.

AWS CLI

를 사용하여 네트워크 패키지를 생성하려면 AWS CLI

1. create-sol-network-package 명령을 사용하여 네트워크 패키지를 생성합니다.

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

2. put-sol-network-package-content 명령을 사용하여 네트워크 패키지 콘텐츠를 업로드합니다. 예제:

```
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
```

AWS TNB에서 네트워크 패키지 보기

네트워크 패키지의 콘텐츠를 보는 방법을 알아보세요.

Console

콘솔을 사용하여 네트워크 패키지를 보려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크 패키지를 선택합니다.
3. 검색창을 사용하여 네트워크 패키지를 찾습니다.

AWS CLI

를 사용하여 네트워크 패키지를 보려면 AWS CLI

1. list-sol-network-packages 명령을 사용하여 네트워크 패키지를 나열합니다.

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

2. get-sol-network-package 명령을 사용하여 네트워크 패키지에 대한 세부 정보를 볼 수 있습니다.

```
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
```

AWS TNB에서 네트워크 패키지 다운로드

AWS TNB 네트워크 서비스 카탈로그에서 네트워크 패키지를 다운로드하는 방법을 알아봅니다.

Console

콘솔을 사용하여 네트워크 패키지를 다운로드하려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크 패키지를 선택합니다.
3. 검색창을 사용하여 네트워크 패키지를 찾습니다.
4. 네트워크 패키지를 선택합니다.
5. 작업, 다운로드를 선택합니다.

AWS CLI

를 사용하여 네트워크 패키지를 다운로드하려면 AWS CLI

- get-sol-network-package-content 명령을 사용하여 네트워크 패키지를 다운로드합니다.

```
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
```

AWS TNB에서 네트워크 패키지 삭제

AWS TNB 네트워크 서비스 카탈로그에서 네트워크 패키지를 삭제하는 방법을 알아봅니다. 네트워크 패키지를 삭제하려면 패키지가 비활성화 상태여야 합니다.

Console

콘솔을 사용하여 네트워크 패키지를 삭제하려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크 패키지를 선택합니다.
3. 검색창을 사용하여 네트워크 패키지를 찾습니다.
4. 네트워크 패키지를 선택합니다.
5. 작업, 비활성화를 선택합니다.
6. 작업, 삭제를 선택합니다.

AWS CLI

를 사용하여 네트워크 패키지를 삭제하려면 AWS CLI

1. update-sol-network-package 명령을 사용하여 네트워크 패키지를 비활성화합니다.

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

2. delete-sol-network-package 명령을 사용하여 네트워크 패키지를 삭제합니다.

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

AWS TNB용 네트워크 인스턴스

네트워크 인스턴스는 AWS TNB에서 생성된 단일 네트워크로, 배포할 수 있습니다.

작업

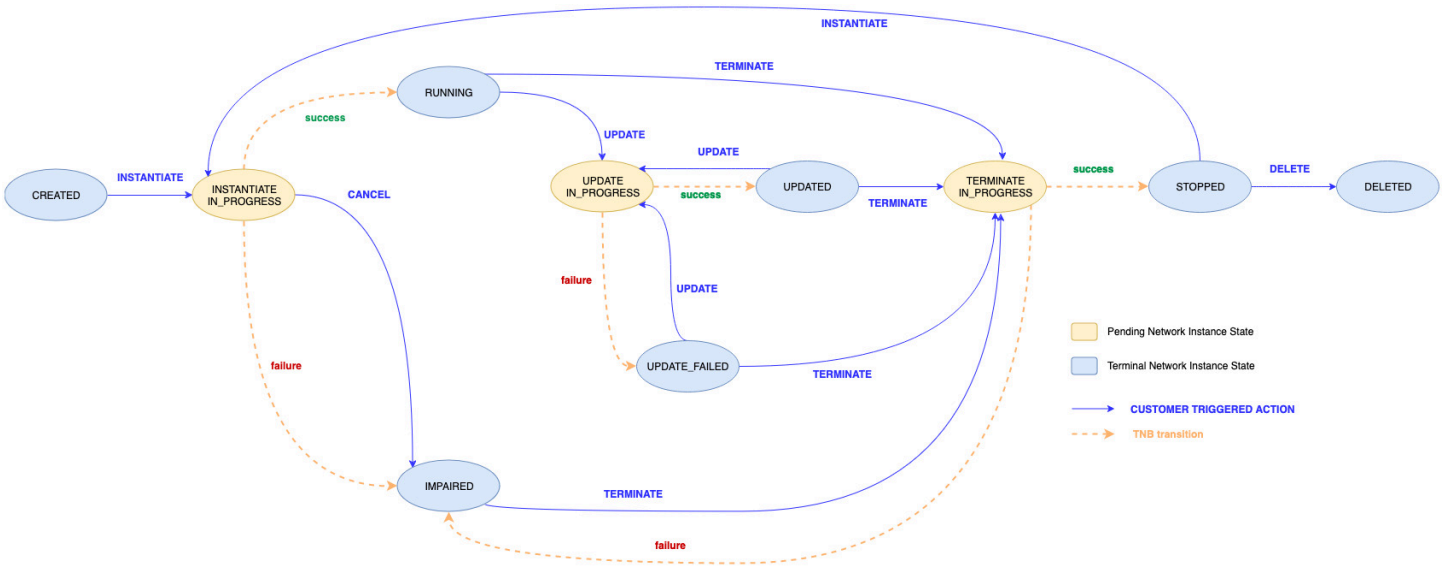
- 네트워크 인스턴스의 수명 주기 작업
- AWS TNB를 사용하여 네트워크 인스턴스 생성
- AWS TNB를 사용하여 네트워크 인스턴스 인스턴스 인스턴스 인스턴스화
- AWS TNB에서 함수 인스턴스 업데이트
- AWS TNB에서 네트워크 인스턴스 업데이트
- AWS TNB에서 네트워크 인스턴스 보기
- AWS TNB에서 네트워크 인스턴스 종료 및 삭제

네트워크 인스턴스의 수명 주기 작업

AWS TNB를 사용하면 ETSI SOL003 및 SOL005에 따라 표준화된 관리 작업을 사용하여 네트워크를 쉽게 관리할 수 있습니다. 다음과 같은 수명 주기 작업을 수행할 수 있습니다.

- 네트워크 생성
- 네트워크 인스턴스화
- 네트워크 함수 업데이트
- 네트워크 인스턴스 업데이트
- 네트워크 세부 정보 및 상태 보기
- 네트워크 종료

다음 이미지는 네트워크 관리 작업을 보여줍니다.



AWS TNB를 사용하여 네트워크 인스턴스 생성

네트워크 패키지를 생성한 후 네트워크 인스턴스를 생성합니다. 네트워크 인스턴스를 생성한 후 인스턴스를 인스턴스화합니다.

Console

콘솔을 사용하여 네트워크 인스턴스를 생성하려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크를 선택합니다.
3. 네트워크 인스턴스 생성을 선택합니다.
4. 인스턴스의 이름과 설명을 입력하고 다음을 선택합니다.
5. 네트워크 패키지를 선택하고 세부 정보를 확인한 후 다음을 선택합니다.
6. 네트워크 인스턴스 생성을 선택합니다.

   새 네트워크 인스턴스가 네트워크 페이지에 나타납니다. 다음으로이 네트워크 인스턴스를 인스턴스화합니다.

AWS CLI

를 사용하여 네트워크 인스턴스를 생성하려면 AWS CLI

- create-sol-network-instance 명령을 사용하여 네트워크 인스턴스를 생성합니다.

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

다음으로이 네트워크 인스턴스를 인스턴스화합니다.

AWS TNB를 사용하여 네트워크 인스턴스 인스턴스 인스턴스 인스턴스화

네트워크 인스턴스를 생성한 후에는 인스턴스화해야 합니다. 네트워크 인스턴스를 인스턴스화하면 AWS TNB는 필요한 AWS 인프라를 프로비저닝하고, 컨테이너화된 네트워크 함수를 배포하고, 네트워킹 및 액세스 관리를 구성하여 완전히 작동하는 네트워크 서비스를 생성합니다.

Console

콘솔을 사용하여 네트워크 인스턴스를 인스턴스화하려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크를 선택합니다.
3. 인스턴스화하려는 네트워크 인스턴스를 선택합니다.
4. 작업을 선택한 다음 인스턴스화를 선택합니다.
5. 네트워크 인스턴스화 페이지에서 세부 정보를 검토하고 선택적으로 파라미터 값을 업데이트합니다.

   파라미터 값 업데이트는이 네트워크 인스턴스에만 적용됩니다. NSD 및 VNFD 패키지의 파라미터는 변경되지 않습니다.
6. 네트워크 인스턴스화를 선택합니다.

   배포 상태 페이지가 나타납니다.
7. 새로 고침 아이콘을 사용하여 네트워크 인스턴스의 배포 상태를 추적합니다. 배포 작업 섹션에서 자동 새로 고침을 활성화하여 각 작업의 진행 상황을 추적할 수도 있습니다.

   배포 상태가 로 변경되면 `Completed`네트워크 인스턴스가 인스턴스화됩니다.

AWS CLI

를 사용하여 네트워크 인스턴스를 인스턴스화하려면 AWS CLI

1. instantiate-sol-network-instance 명령을 사용하여 네트워크 인스턴스를 인스턴스화합니다.

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

2. 그런 다음 네트워크 작업 상태를 확인합니다.

AWS TNB에서 함수 인스턴스 업데이트

네트워크 인스턴스가 인스턴스화되면 네트워크 인스턴스에서 함수 패키지를 업데이트할 수 있습니다.

Console

콘솔을 사용하여 함수 인스턴스를 업데이트하려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크를 선택합니다.
3. 네트워크 인스턴스를 선택합니다. 네트워크 인스턴스의 상태가 인 경우에만 네트워크 인스턴스를 업데이트할 수 있습니다`Instantiated`.

   네트워크 인스턴스 페이지가 나타납니다.
4. 함수 탭에서 업데이트할 함수 인스턴스를 선택합니다.
5. 업데이트를 선택합니다.
6. 업데이트 재정의를 입력합니다.
7. 업데이트를 선택합니다.

AWS CLI

CLI를 사용하여 함수 인스턴스 업데이트

update-sol-network-instance 명령을 `MODIFY_VNF_INFORMATION` 업데이트 유형과 함께 사용하여 네트워크 인스턴스의 함수 인스턴스를 업데이트합니다.

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

AWS TNB에서 네트워크 인스턴스 업데이트

네트워크 인스턴스가 인스턴스화되면 인프라 또는 애플리케이션을 업데이트해야 할 수 있습니다. 이렇게 하려면 네트워크 인스턴스의 네트워크 패키지 및 파라미터 값을 업데이트하고 업데이트 작업을 배포하여 변경 사항을 적용합니다.

고려 사항

- `Instantiated` 또는 `Updated` 상태의 네트워크 인스턴스를 업데이트할 수 있습니다.
- 네트워크 인스턴스를 업데이트할 때 `UpdateSolNetworkService` API는 새 네트워크 패키지와 파라미터 값을 사용하여 네트워크 인스턴스의 토폴로지를 업데이트합니다.
- AWS TNB는 네트워크 인스턴스의 NSD 및 VNFD 파라미터 수가 200개를 초과하지 않는지 확인합니다. 이 제한은 서비스에 영향을 미치는 잘못된 페이로드를 전달하는 악의적인 행위자로부터 보호하기 위해 적용됩니다.

업데이트할 수 있는 파라미터

인스턴스화된 네트워크 인스턴스를 업데이트할 때 다음 파라미터를 업데이트할 수 있습니다.

파라미터	설명	예: 이전	예: 이후

Amazon EKS 클러스터 버전	Amazon EKS 클러스터 컨트롤 플레인 `version` 파라미터의 값을 다음 마이너 버전으로 업데이트할 수 있습니다. 버전을 다운그레이드할 수 없습니다.	`EKSCluster:`
		`  type: tosca.nodes.AWS.Compute.EKS`
		`  properties:`
		`  version: "1.28"`
			`EKSC`
			`r:`
			
			`typ`
			`tos`
			`es.A`
			`mput`

파라미터	설명	예: 이전	예: 이후
			pro s: ver "1.

파라미터	설명	예: 이전	예: 이후
Amazon EKS 워커 노드	EKSManagedNode kubernetes_version 파라미터 값을 업데이트하여 노드 그룹을 최신 Amazon EKS 버전으로 업그레이드하거나 ami_id 파라미터를 업데이트하여 노드 그룹을 최신 EKS 최적화 AMI로 업그레이드할 수 있습니다. 의 AMI ID를 업데이트할 수 있습니다EKSSelfManagedNode . AMI의 Amazon EKS 버전은 Amazon EKS 클러스터 버전과 동일하거나 최대 2개 버전 미만이어야 합니다. 예를 들어 Amazon EKS 클러스터 버전이 1.31인 경우 Amazon EKS AMI 버전은 1.31, 1.30 또는 1.29여야 합니다.	EKSManagedNodeGroup01: ... properties: kubernetes_version: "1.28" EKSSelfManagedNode01: compute: compute: properties: ami_id: "ami-1231230LD "	EKSM dNod p01: ... pro s: kub s_ve : "1. EKS nage 01: com

파라미터	설명	예: 이전	예: 이후
			com pro s: ami "*am* *3NEW*

파라미터	설명	예: 이전	예: 이후

Amazon EKS 노드 그룹	컴퓨팅 요구 사항에 따라 노드 그룹을 추가하거나 제거할 수 있습니다.
	
	기존 노드 그룹을 삭제하고 새 노드 그룹을 추가할 때 새 노드 그룹의 IDs가 삭제된 노드 그룹과 다른지 확인합니다. 그렇지 않으면 작업이 삭제 및 추가 대신 노드 그룹 수정으로 처리됩니다. 기존 노드 그룹의 경우 제한된 파라미터 집합만 업데이트할 수 있습니다. 이 테이블을 스크롤하여 업데이트할 수 있는 파라미터를 확인합니다.
		<pre>Free5GCEKSNode01:
		  type: tosca.nodes.AWS.Compute.EKSManagedNode
		  ...
		    scaling:
		      properties:
		        desired_size: 1
		        min_size: 1
		        max_size: 1
		  ...
		Free5GCEKSNode02 : # Deleted Nodegroup
		  type: tosca.nodes.AWS.Compute.EKSManagedNode
		  ...
		    scaling:
		      properties:
		        desired_size: 1
		        min_size: 1
		        max_size: 1
		  ...
		Free5GCEKSNode03 : # Deleted Nodegroup
		  type: tosca.nodes.AWS.Compute.EKSSelfManagedNode
		  ...
		    scaling:
		      properties:
		        desired_size: 1
		        min_size: 1
		        max_size: 1
		  ...</pre>
			<pre>Free
			SNod
			
			typ
			tos
			es.A
			mput
			Mana
			de
			
			...
			
			sca
			
			pro
			s:
			
			des
			ize:
			1</pre>

파라미터	설명	예: 이전	예: 이후
			mir 1 max 1 ... Free SNod # New Nod typ tos es.A mput Self edNo ... sca pro s:

파라미터	설명	예: 이전	예: 이후
			des ize: 1 mir 1 max 1 ... Free SNod # New Nod typ tos

파라미터	설명	예: 이전	예: 이후
			es.A mput Mana de ... sca pro s: des ize: 1 mir 1

파라미터	설명	예: 이전	예: 이후
			max 1 ...

파라미터	설명	예: 이전	예: 이후

조정 속성	EKSManagedNode 및 EKSSelfManagedNode TOSCA 노드의 조정 속성을 업데이트할 수 있습니다.	`EKSNodeGroup01:`
		`  ...`
		`  scaling:`
		`  properties:`
		`  desired_size: 1`
		`  min_size: 1`
		`  max_size: 1`
			`EKSNodeGroup0`
			`  ...`
			`  scaling:`
			`  properties:`
			` desired_size:

파라미터	설명	예: 이전	예: 이후
			mir max

파라미터	설명	예: 이전	예: 이후

Amazon EBS CSI 플러그인 속성	Amazon EKS 클러스터에서 Amazon EBS CSI 플러그인을 활성화하거나 비활성화할 수 있습니다. 플러그인 버전을 변경할 수도 있습니다.	`EKSCluster:`
		`  capabilities:`
		`  ...`
		`  ebs_csi:`
		`  properties:`
		`  enabled: false`
			`EKSC`
			`r:`
			`cap`
			`ies:`
			`...`
			`ebs`
			`pro`
			`s:`
			`ena`
			`ver`
			`"v1`
			`e`

파라미터	설명	예: 이전	예: 이후
			`ksbu` "

파라미터	설명	예: 이전	예: 이후

루트 볼륨 크기	EKSManagedNode 및 EKSSelfManagedNode TOSCA 노드의 루트 볼륨 크기 속성을 추가, 제거 또는 업데이트할 수 있습니다.	`Free5GCEKSNode01:`
		`  ...`
		`  capabilities:`
		`  compute:`
		`  properties:`
		`  root_volume_size: 50`
			`Free`
			`SNod`
			`...`
			`cap`
			`ies:`
			`com`
			`pro`
			`s:`

파라미터	설명	예: 이전	예: 이후
			roo me_s

파라미터	설명	예: 이전	예: 이후
VNF	NSD에서 VNFs를 참조하고 VNFDeployment TOSCA 노드를 사용하여 NSD에서 생성된 클러스터에 배포할 수 있습니다. 업데이트의 일환으로 VNFs 있습니다.	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 ....

파라미터	설명	예: 이전	예: 이후
			Sample elmD : typ tos es.A ploy VNFD ment req nts: clu EKS r

파라미터	설명	예: 이전	예: 이후
			vnf - *v* *leVN* - *v* *leVN*

파라미터	설명	예: 이전	예: 이후
후크	네트워크 함수를 생성하기 전과 후에 수명 주기 작업을 실행하려면 VNFDeployment 노드에 pre_create 및 post_create 후크를 추가합니다. 이 예제에서는이 인스턴스화되기 전에 PreCreateHook 후크vnf3.SampleVNF3 가 실행되고이 인스턴스화되면 PostCreateHook 후크vnf3.SampleVNF3 가 실행됩니다.	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:

파라미터	설명	예: 이전	예: 이후
			typ tos es.A ploy VNFD ment rec nts: clu EKS r vnf - v leVN No cha to thi fur as the nam and uui rem

파라미터	설명	예: 이전	예: 이후
			the sam - v leVN New VNF as the nam , vnf was not pre y pre int s: Hoo pos te: eHoo

파라미터	설명	예: 이전	예: 이후
			pre e: *Hook*

파라미터	설명	예: 이전	예: 이후

후크	네트워크 함수를 업데이트하기 전과 후에 수명 주기 작업을 실행하려면 pre_update 후크와 post_update 후크를 VNFDeployment 노드에 추가할 수 있습니다.
	
	이 예제에서는 PreUpdate Hook 가 업데이트되기 전에 vnf1.SampleVNF1 가 실행되고 vnf1.SampleVNF1 가 네임스페이스 vnf1에 uuid 대해 업데이트된 로 표시된 vnf 패키지로 업데이트된 후에가 실행PostUpdateHook 됩니다.
		vnfds:
		  - descriptor_id:
		 "*43c012fa-2616-41a8-a833-0dfd4c5a049e* "
		  namespace: " *vnf1*"
		  - descriptor_id:
		 "*64222f98-ecd6-4871-bf94-7354b53f3ee5* "
		  namespace: " *vnf2*"
		...
		
		SampleVNF1HelmDeploy:
		  type: tosca.nodes.AWS.Deployment.VNFDeployment
		  requirements:
		  cluster: EKSCluster
		  vnfs:
		  - *vnf1.SampleVNF1*
		  - *vnf2.SampleVNF2*
			vnfd
			
			 -
			 des
			r_id
			 "*0e*
			bd87
			-
			b8a1
			4666
			 "
			
			 nam
			:
			 "*vn*
			
			 -
			 des
			r_id
			 "*64*
			ecd6
			-
			bf94
			4b53
			 "
			
			 nam
			:
			 "*vn*
			...
			S
			ampl
			Helm
			y:

파라미터	설명	예: 이전	예: 이후
			typ tos es.A ploy VNFD ment rec nts: clu EKS r vnf - v leVN A VNF upd as the uui cha for nam "vr

파라미터	설명	예: 이전	예: 이후

```
 - v
leVN
 No
 cha
 to
 thi
 fur
 as
 nam
 and
 uui
 rem
 the
 sam

 int
s:


 Hoo


 pre
e:
Hook


 pos
```

파라미터	설명	예: 이전	예: 이후
			te: eHoc

파라미터	설명	예: 이전	예: 이후
서브넷	네트워크에서 서브넷을 추가하고 삭제할 수 있습니다. 서브넷을 삭제하기 전에 네트워크의 리소스에서 서브넷을 사용하지 않는지 확인합니다.	*Free5GCSubnet01* : #Deleted Subnet type: tosca.nodes.AWS.Networking.Subnet properties: type: "PUBLIC" availability_zone: { get_input: subnet_01_az } cidr_block: { get_input: subnet_01_cidr_block } requirements: route_table: Free5GCRouteTable vpc: Free5GCVPC	*Free* *bnet* #Ne Sub typ tos es.A two Subn pro s: typ "PU ava ity_ { g : sub _az cid k: { g : sub _cid ck }

파라미터	설명	예: 이전	예: 이후
			rec nts: rou le: Fre uteT vpc Fre C

파라미터	설명	예: 이전	예: 이후
보안 그룹	네트워크에서 보안 그룹을 추가하고 삭제할 수 있습니다. 보안 그룹을 삭제하기 전에 네트워크의 리소스에서 보안 그룹을 사용하지 않는지 확인합니다.	*Free5GCSecurityGroup01* : #Deleted Security Group type: tosca.nodes.AWS.Networking.SecurityGroup properties: description: "SecurityGroup for Free5GC cluster" name: "Free5GCSecurityGroup01" tags: - "Name=Free5GCEKSAdditionalSecurityGroup" requirements: vpc: Free5GCVPC *Free5GCSecurityGroupEgressRule01* : #Deleted Security Group Egress Node type: tosca.nodes.AWS.Networking.SecurityGroupEgressRule properties: ip_protocol: "tcp" from_port: 8000 to_port: 9000 description: "Egress Rule for free5GC cluster" cidr_ip : "172.10.10.1/24" requirements:	*Free* *curi* *up02* #Ne Sec Gro typ tos es.A twor Secu roup pro s: des on: "Se Grou for Fre clu nam "Fr ecur oup0 tag

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

파라미터	설명	예: 이전	예: 이후
			pro s: ip_ ol: "to fro : 800 to_ 900 des on: "Eg Rul for fre clu cid "17 0.1/ rec nts:

파라미터	설명	예: 이전	예: 이후
			sec grou Fre curi up02 *Free* *curi* *upIr* *Rule* #Ne Sec Gro Ing Noc typ tos es.A twor Secu roup ssRu pro s: ip_ ol: "to

파라미터	설명	예: 이전	예: 이후
			fro : 800 to_ 900 des on: "Ir Rul for fre clu cid "17 0.1/ req nts: sec grou Fre curi up02

파라미터	설명	예: 이전	예: 이후
네트워크 인터페이스	네트워크에서 ENIs 추가, 수정 및 삭제할 수 있습니다.	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

파라미터	설명	예: 이전	예: 이후
			- Fre curi up01 Fre e5GC : #Mo ENI typ tos es.A twor ENI pro s: dev dex: 3 sou st_c tru rec nts: sub Fre ISub

파라미터	설명	예: 이전	예: 이후
			se grou - Fre curi up01 Free I03 #Ne ENI typ tos es.A twor ENI pro s: dev dex: 3 req nts: sub

파라미터	설명	예: 이전	예: 이후
			Fre bnet sec grou - Fre curi up01

네트워크 인스턴스 업데이트

Console

콘솔을 사용하여 네트워크 인스턴스를 업데이트하려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크를 선택합니다.
3. 네트워크 인스턴스를 선택합니다. 네트워크 인스턴스의 상태가 `Instantiated` 또는 인 경우에만 네트워크 인스턴스를 업데이트할 수 있습니다`Updated`.
4. 작업 및 업데이트를 선택합니다.

   현재 인프라의 네트워크 세부 정보 및 파라미터 목록과 함께 인스턴스 업데이트 페이지가 나타납니다.
5. 새 네트워크 패키지를 선택합니다.

   새 네트워크 패키지의 파라미터는 업데이트된 파라미터 섹션에 표시됩니다.

6. 선택적으로 업데이트된 파라미터 섹션에서 파라미터 값을 업데이트합니다. 업데이트할 수 있는 파라미터 값 목록은 섹션을 참조하세요업데이트할 수 있는 파라미터.
7. 네트워크 업데이트를 선택합니다.

   AWS TNB는 요청을 검증하고 배포를 시작합니다. 배포 상태 페이지가 나타납니다.
8. 새로 고침 아이콘을 사용하여 네트워크 인스턴스의 배포 상태를 추적합니다. 배포 작업 섹션에서 자동 새로 고침을 활성화하여 각 작업의 진행 상황을 추적할 수도 있습니다.

   배포 상태가 로 변경되면 Completed네트워크 인스턴스가 업데이트됩니다.
9.
   - 검증에 실패하면 네트워크 인스턴스는 업데이트를 요청하기 전과 동일한 상태인 Instantiated 또는 로 유지됩니다Updated.
   - 업데이트에 실패하면 네트워크 인스턴스 상태가 로 표시됩니다Update failed. 실패한 각 작업에 대한 링크를 선택하여 이유를 확인합니다.
   - 업데이트에 성공하면 네트워크 인스턴스 상태에가 표시됩니다Updated.

AWS CLI

CLI를 사용하여 네트워크 인스턴스 업데이트

update-sol-network-instance 명령을 UPDATE_NS 업데이트 유형과 함께 사용하여 네트워크 인스턴스를 업데이트합니다.

```
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\"}}"
```

AWS TNB에서 네트워크 인스턴스 보기

네트워크 인스턴스를 보는 방법을 알아봅니다.

Console

콘솔을 사용하여 네트워크 인스턴스를 보려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크 인스턴스를 선택합니다.
3. 검색창을 사용하여 네트워크 인스턴스를 찾습니다.

AWS CLI

를 사용하여 네트워크 인스턴스를 보려면 AWS CLI

1. list-sol-network-instances 명령을 사용하여 네트워크 인스턴스를 나열합니다.

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

2. get-sol-network-instance 명령을 사용하여 특정 네트워크 인스턴스에 대한 세부 정보를 봅니다.

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

AWS TNB에서 네트워크 인스턴스 종료 및 삭제

네트워크 인스턴스를 삭제하려면 인스턴스가 종료 상태여야 합니다.

Console

콘솔을 사용하여 네트워크 인스턴스를 종료 및 삭제하려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크를 선택합니다.
3. 네트워크 인스턴스의 ID를 선택합니다.
4. 종료를 선택합니다.
5. 확인 메시지가 나타나면 ID를 입력한 다음 종료를 선택합니다.
6. 네트워크 인스턴스의 상태를 추적하려면 새로 고침합니다.
7. (선택 사항) 네트워크 인스턴스를 선택한 후 삭제를 선택합니다.

AWS CLI

를 사용하여 네트워크 인스턴스를 종료하고 삭제하려면 AWS CLI

1. terminate-sol-network-instance 명령을 사용하여 네트워크 인스턴스를 종료합니다.

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

2. (선택 사항) delete-sol-network-instance 명령을 사용하여 네트워크 인스턴스를 삭제합니다.

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

AWS TNB에 대한 네트워크 작업

네트워크 작업은 네트워크 인스턴스 인스턴스화 또는 종료와 같이 네트워크에 수행되는 모든 작업입니다.

업무

- AWS TNB 네트워크 작업 보기
- AWS TNB 네트워크 작업 취소

AWS TNB 네트워크 작업 보기

네트워크 작업과 관련된 태스크, 해당 태스크의 상태를 비롯한 네트워크 작업 세부 정보를 볼 수 있습니다.

Console

콘솔을 사용하여 네트워크 작업을 보려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크 인스턴스를 선택합니다.
3. 검색창을 사용하여 네트워크 인스턴스를 찾습니다.
4. 배포 탭에서 네트워크 작업을 선택합니다.

AWS CLI

를 사용하여 네트워크 작업을 보려면 AWS CLI

1. list-sol-network-operations 명령을 사용하여 모든 네트워크 작업을 나열할 수 있습니다.

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

2. get-sol-network-operation 명령을 사용하여 네트워크 작업에 대한 세부 정보를 볼 수 있습니다.

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

AWS TNB 네트워크 작업 취소

네트워크 작업을 취소하는 방법을 알아봅니다.

Console

콘솔을 사용하여 네트워크 작업을 취소하려면

1. https://console.aws.amazon.com/tnb/ AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크를 선택합니다.
3. 네트워크의 ID를 선택하여 세부 정보 페이지를 엽니다.
4. 배포 탭에서 네트워크 작업을 선택합니다.
5. 작업 취소를 선택합니다.

AWS CLI

를 사용하여 네트워크 작업을 취소하려면 AWS CLI

cancel-sol-network-operation 명령을 사용하여 모든 네트워크 작업을 취소할 수 있습니다.

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

AWS TNB에 대한 TOSCA 참조

TOSCA(Topology and Orchestration Specification for Cloud Applications)는 CSP가 클라우드 기반 웹 서비스의 토폴로지, 해당 구성 요소, 관계 및 이를 관리하는 프로세스를 설명하는 데 사용하는 선언적 구문입니다. CSP는 TOSCA 템플릿에서 연결 지점, 연결 지점 간의 논리 링크, 그리고 친화도 및 보안과 같은 정책을 설명합니다. 그런 다음 CSPs 템플릿을 AWS TNB에 업로드하여 AWS 가용 영역에서 작동하는 5G 네트워크를 설정하는 데 필요한 리소스를 합성합니다.

내용

VNFD 템플릿

가상 네트워크 함수 설명자(VNFD) 템플릿을 정의합니다.

구문

```
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
```

토폴로지 템플릿

`node_templates`

TOSCA AWS 노드입니다. 가능한 노드는 다음과 같습니다.

- AWS.VNF
- AWS.Artifacts.Helm

AWS.VNF

AWS 가상 네트워크 함수(VNF) 노드를 정의합니다.

구문

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

속성

`descriptor_id`

설명자의 UUID입니다.

필수 항목 여부: 예

유형: String

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

`descriptor_version`

VNFD의 버전입니다.

필수 항목 여부: 예

유형: String

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

`descriptor_name`

설명자의 이름입니다.

필수 항목 여부: 예

유형: String

`provider`

VNFD의 작성자입니다.

필수 항목 여부: 예

유형: String

요구 사항

`helm`

컨테이너 아티팩트를 정의하는 Helm 디렉터리입니다. AWS.Artifacts.Helm에 대한 참조입니다.

필수 항목 여부: 예

유형: String

예제

```
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

AWS Helm 노드를 정의합니다.

구문

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

```
  properties:
    implementation: String
```

속성

`implementation`

CSAR 패키지 내에 차트 Helm이 포함된 로컬 디렉터리입니다.

필수 항목 여부: 예

유형: String

예제

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

네트워크 서비스 설명자 템플릿

네트워크 서비스 설명자(NSD) 템플릿을 정의합니다.

구문

```
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
```

정의된 파라미터 사용

VPC 노드의 CIDR 블록과 같은 파라미터를 동적으로 전달하려는 경우 NSD 템플릿에서 `{ get_input: input-parameter-name }` 구문을 사용하고 파라미터를 정의할 수 있습니다. 그런 다음 동일한 NSD 템플릿에서 파라미터를 재사용하세요.

다음 예제에서는 파라미터를 정의하고 사용하는 방법을 보여줍니다.

```
tosca_definitions_version: tnb_simple_yaml_1_0

topology_template:

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

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

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

VNFD 가져오기

`descriptor_id`

설명자의 UUID입니다.

필수 항목 여부: 예

유형: String

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

`namespace`

고유한 이름입니다.

필수 항목 여부: 예

유형: String

토폴로지 템플릿

`node_templates`

가능한 TOSCA AWS 노드는 다음과 같습니다.

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

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

AWS.NS

AWS 네트워크 서비스(NS) 노드를 정의합니다.

구문

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

속성

`descriptor_id`

설명자의 UUID입니다.

필수 항목 여부: 예

유형: String

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

`descriptor_version`

NSD의 버전입니다.

필수 항목 여부: 예

유형: String

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

`descriptor_name`

설명자의 이름입니다.

필수 항목 여부: 예

유형: String

예제

```
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

클러스터 이름, 원하는 Kubernetes 버전 및 Kubernetes 컨트롤 플레인이 NFs에 필요한 AWS 리소스를 관리할 수 있는 역할을 제공합니다. Multus 컨테이너 네트워크 인터페이스(CNI) 플러그인이 활성화되어 있습니다. 여러 네트워크 인터페이스를 연결하고 고급 네트워크 구성을 Kubernetes 기반 네트워크 함수에 적용할 수 있습니다. 클러스터 엔드포인트 액세스와 클러스터의 서브넷도 지정합니다.

구문

```
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
```

기능

`multus`

선택 사항. Multus 컨테이너 네트워크 인터페이스(CNI) 사용을 정의하는 속성입니다.

`multus`를 포함시킬 경우 `enabled` 및 `multus_role` 속성을 지정합니다.

`enabled`

기본 Multus 기능이 활성화되어 있는지 여부를 나타냅니다.

필수 여부: 예

유형: 부울

`multus_role`

Multus 네트워크 인터페이스 관리 역할입니다.

필수 항목 여부: 예

유형: String

`ebs_csi`

Amazon EKS 클러스터에 설치된 Amazon EBS CSI(Container Storage Interface) 드라이버를 정의하는 속성입니다.

이 플러그인을 활성화하여 AWS Outposts, AWS 로컬 영역 또는에서 Amazon EKS 자체 관리형 노드를 사용합니다 AWS 리전. 자세한 내용은 Amazon EKS 사용 설명서에서 Amazon Elastic Block Store CSI 드라이버를 참조하세요.

`enabled`

기본 Amazon EBS CSI 드라이버가 설치되어 있는지 여부를 나타냅니다.

필수 여부: 아니요

유형: 부울

`version`

Amazon EBS CSI 드라이버 추가 기능의 버전입니다. 버전은 DescribeAddonVersions 작업에서 반환된 버전 중 하나와 일치해야 합니다. 자세한 내용은 Amazon EKS API 참조의 DescribeAddonVersions를 참조하세요.

필수 여부: 아니요

유형: 문자열

속성

`version`

클러스터용 Kubernetes 버전. AWS Telco Network Builder는 Kubernetes 버전 1.25~1.32를 지원합니다.

필수 항목 여부: 예

유형: String

가능한 값: 1.25 | 1.26 | 1.27 | 1.28 | 1.29 | 1.30 | 1.31 | 1.32

`access`

클러스터 엔드포인트 액세스입니다.

필수 항목 여부: 예

유형: String

가능한 값: `PRIVATE` | `PUBLIC` | `ALL`

`cluster_role`

클러스터 관리 역할입니다.

필수 항목 여부: 예

유형: String

`tags`

리소스에 연결할 태그입니다.

필수 여부: 아니요

유형: 목록

`ip_family`

클러스터의 서비스 및 포드 주소에 대한 IP 패밀리를 나타냅니다.

허용되는 값: `IPv6`, `IPv4`

기본 값: `IPv4`

필수 여부: 아니요

유형: 문자열

요구 사항

`subnets`

AWS.Networking.Subnet 노드입니다.

필수 여부: 예

유형: 목록

예제

```
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

AuthRole은 사용자가 IAM 역할을 사용하여 Amazon EKS 클러스터 aws-auth ConfigMap에 액세스할 수 있도록 Amazon EKS 클러스터에 IAM 역할을 추가할 수 있게 해 줍니다.

구문

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

속성

`role_mappings`

Amazon EKS 클러스터 aws-auth ConfigMap에 추가해야 하는 IAM 역할을 정의하는 매핑 목록입니다.

`arn`

IAM 역할의 ARN입니다.

필수 항목 여부: 예

유형: String

`groups`

arn에 정의된 역할에 할당할 Kubernetes 그룹입니다.

필수 여부: 아니요

유형: 목록

요구 사항

`clusters`

AWS.Compute.EKS 노드입니다.

필수 여부: 예

유형: 목록

예제

```
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 TNB는 EKS 관리형 노드 그룹을 지원하여 Amazon EKS Kubernetes 클러스터용 노드(Amazon EC2 인스턴스)의 프로비저닝 및 수명 주기 관리를 자동화합니다. EKS 노드 그룹을 생성하려면 다음을 수행합니다.

- AMI의 ID 또는 AMI 유형을 제공하여 클러스터 작업자 노드의 Amazon Machine Image(AMI)를 선택합니다.
- SSH 액세스를 위한 Amazon EC2 키 페어와 노드 그룹의 조정 속성을 제공합니다.

- 노드 그룹이 Amazon EKS 클러스터와 연결되어 있는지 확인합니다.
- 작업자 노드의 서브넷을 제공합니다.
- 선택적으로 보안 그룹, 노드 레이블 및 배치 그룹을 노드 그룹에 연결합니다.

구문

```
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
```

기능

`compute`

Amazon EKS 관리형 노드 그룹의 컴퓨팅 파라미터를 정의하는 속성(예: Amazon EC2 인스턴스 유형 및 Amazon EC2 인스턴스 AMI)입니다.

`ami_type`

Amazon EKS 지원 AMI 유형입니다.

필수 항목 여부: 예

유형: String

가능한 값: `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`

AMI의 ID입니다.

필수 여부: 아니요

유형: 문자열

> **Note**
>
> 템플릿에 `ami_type` 및 `ami_id`가 모두 지정된 경우 AWS TNB는 `ami_id` 값만 사용하여 를 생성합니다`EKSManagedNode`.

`instance_types`

인스턴스의 크기입니다.

필수 여부: 예

유형: 목록

`key_pair`

SSH 액세스를 활성화하기 위한 EC2 키 쌍입니다.

필수 항목 여부: 예

유형: String

`root_volume_encryption`

Amazon EBS 루트 볼륨에 대해 Amazon EBS 암호화를 활성화합니다. 이 속성이 제공되지 않으면 AWS TNB는 기본적으로 Amazon EBS 루트 볼륨을 암호화합니다.

필수 항목 여부: 아니요

기본값: true

유형: 부울

`root_volume_encryption_key_arn`

AWS KMS key. AWS TNB의 ARN은 일반 키 ARN, 다중 리전 키 ARN 및 별칭 ARN을 지원합니다.

필수 여부: 아니요

유형: 문자열

> **Note**
>
> - `root_volume_encryption`가 false인 경우를 포함하지 마십시오`root_volume_encryption_key_arn`.
> - AWS TNB는 Amazon EBS 지원 AMI의 루트 볼륨 암호화를 지원합니다.
> - AMI의 루트 볼륨이 이미 암호화된 경우 루트 볼륨을 다시 암호화하려면 AWS TNB`root_volume_encryption_key_arn`용를 포함해야 합니다.
> - AMI의 루트 볼륨이 암호화되지 않은 경우 AWS TNB는 `root_volume_encryption_key_arn`를 사용하여 루트 볼륨을 암호화합니다.
>
>   를 포함하지 않으면 `root_volume_encryption_key_arn` AWS TNB는에서 제공하는 기본 키를 사용하여 루트 볼륨을 암호화 AWS Key Management Service 합니다.
> - AWS TNB는 암호화된 AMI를 복호화하지 않습니다.

`root_volume_size`

GiBs.

필수 여부: 아니요

기본값: 20

유형: 정수

가능한 값: 1~16,384

scaling

Amazon EKS 관리형 노드 그룹의 조정 파라미터를 정의하는 속성(예: 원하는 Amazon EC2 인스턴스 수, 노드 그룹 내 최소 및 최대 Amazon EC2 인스턴스 수)입니다.

desired_size

이 노드 그룹의 인스턴스 수입니다.

필수 여부: 예

유형: 정수

min_size

이 노드 그룹의 최소 인스턴스 수입니다.

필수 여부: 예

유형: 정수

max_size

이 노드 그룹의 최대 인스턴스 수입니다.

필수 여부: 예

유형: 정수

속성

node_role

Amazon EC2 인스턴스에 연결된 IAM 역할의 ARN입니다.

필수 항목 여부: 예

유형: String

`tags`

리소스에 연결할 태그입니다.

필수 여부: 아니요

유형: 목록

`kubernetes_version`

Managed Node 그룹의 Kubernetes 버전. AWS TNB는 Kubernetes 버전 1.25~1.32를 지원합니다. 다음을 고려하세요.

- `kubernetes_version` 또는를 지정합니다`ami_id`. 둘 다 지정하지 마세요.
- 는 AWS.Compute.EKSManagedNode 버전보다 작거나 같아야 `kubernetes_version` 합니다.
- AWS.Compute.EKSManagedNode 버전과 버전 간에는 세 가지 버전이 다를 수 있습니다`kubernetes_version`.
- `kubernetes_version` 또는 `ami_id`를 지정하지 않으면 AWS TNB는 `AWS.Compute.EKSManagedNode` 버전의 최신 AMI를 사용하여를 생성합니다. `EKSManagedNode`

필수 여부: 아니요

유형: 문자열

가능한 값: 1.25 | 1.26 | 1.27 | 1.28 | 1.29 | 1.30 | 1.31 | 1.32

요구 사항

`cluster`

AWS.Compute.EKS 노드입니다.

필수 항목 여부: 예

유형: String

`subnets`

AWS.Networking.Subnet 노드입니다.

필수 여부: 예

유형: 목록

`network_interfaces`

AWS.Networking.ENI 노드입니다. 네트워크 인터페이스와 서브넷이 동일한 가용 영역으로 설정되어 있는지 확인하세요. 그렇지 않으면 인스턴스화가 실패합니다.

를 설정하면 속성을 AWS.Compute.EKS 노드에 포함`multus_role`하면 `network_interfaces` AWS TNB는 `multus` 속성에서 ENIs와 관련된 권한을 얻습니다. 그렇지 않으면 AWS TNB는 node_role 속성에서 ENI와 관련된 권한을 얻습니다.

필수 여부: 아니요

유형: 목록

`security_groups`

AWS.Networking.SecurityGroup 노드입니다.

필수 여부: 아니요

유형: 목록

`placement_group`

tosca.nodes.AWS.Compute.PlacementGroup 노드입니다.

필수 여부: 아니요

유형: 문자열

`user_data`

tosca.nodes.AWS.Compute.UserData 노드 참조입니다. 사용자 데이터 스크립트는 관리형 노드 그룹에서 시작된 Amazon EC2 인스턴스에 전달됩니다. 사용자 지정 사용자 데이터를 실행하는 데 필요한 권한을 노드 그룹에 전달된 node_role에 추가합니다.

필수 여부: 아니요

유형: 문자열

`labels`

노드 레이블 목록입니다. 노드 레이블에는 이름과 값이 있어야 합니다. 다음 기준을 사용하여 레이블을 생성합니다.

- 이름과 값은 로 구분해야 합니다=.
- 이름과 값의 길이는 각각 최대 63자입니다.
- 레이블에는 문자(A~Z, a~z), 숫자(0~9) 및 다음 문자가 포함될 수 있습니다. `[-, _, ., *, ?]`
- 이름과 값은 영숫자, ?또는 * 문자로 시작하고 끝나야 합니다.

예: `myLabelName1=*NodeLabelValue1`

필수 여부: 아니요

유형: 목록

예제

```
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 TNB는 Amazon EKS 자체 관리형 노드를 지원하여 Amazon EKS Kubernetes 클러스터에 대한 노드(Amazon EC2 인스턴스)의 프로비저닝 및 수명 주기 관리를 자동화합니다. Amazon EKS 노드 그룹을 생성하려면 다음을 수행합니다.

- AMI의 ID 중 하나를 제공하여 클러스터 작업자 노드의 Amazon Machine Image(AMI)를 선택합니다.
- SSH 액세스를 위한 Amazon EC2 키 페어를 제공합니다.
- 노드 그룹이 Amazon EKS 클러스터와 연결되어 있는지 확인합니다.
- 인스턴스 유형과 원하는 크기, 최소 크기 및 최대 크기를 제공합니다.
- 작업자 노드의 서브넷을 제공합니다.
- 선택적으로 보안 그룹, 노드 레이블 및 배치 그룹을 노드 그룹에 연결합니다.

구문

```
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
```

기능

compute

Amazon EKS 자체 관리형 노드의 컴퓨팅 파라미터를 정의하는 속성(예: Amazon EC2 인스턴스 유형 및 Amazon EC2 인스턴스 AMI)입니다.

`ami_id`

인스턴스를 시작하는 데 사용되는 AMI ID입니다. AWS TNB는 IMDSv2를 활용하는 인스턴스를 지원합니다. 자세한 내용은 IMDS 버전 단원을 참조하십시오.

> **Note**
>
> 의 AMI ID를 업데이트할 수 있습니다`EKSSelfManagedNode`. AMI의 Amazon EKS 버전은 Amazon EKS 클러스터 버전과 동일하거나 최대 2개 버전 미만이어야 합니다. 예를 들어 Amazon EKS 클러스터 버전이 1.31인 경우 Amazon EKS AMI 버전은 1.31, 1.30 또는 1.29여야 합니다.

필수 항목 여부: 예

유형: String

`instance_type`

인스턴스의 크기입니다.

필수 항목 여부: 예

유형: String

`key_pair`

SSH 액세스를 활성화하기 위한 Amazon EC2 키 쌍입니다.

필수 항목 여부: 예

유형: String

`root_volume_encryption`

Amazon EBS 루트 볼륨에 대해 Amazon EBS 암호화를 활성화합니다. 이 속성이 제공되지 않으면 AWS TNB는 기본적으로 Amazon EBS 루트 볼륨을 암호화합니다.

필수 항목 여부: 아니요

기본값: true

유형: 부울

`root_volume_encryption_key_arn`

AWS KMS key. AWS TNB의 ARN은 일반 키 ARN, 다중 리전 키 ARN 및 별칭 ARN을 지원합니다.

필수 여부: 아니요

유형: 문자열

> **Note**
>
> - `root_volume_encryption`가 false인 경우를 포함하지 마십시오`root_volume_encryption_key_arn`.
> - AWS TNB는 Amazon EBS 지원 AMI의 루트 볼륨 암호화를 지원합니다.
> - AMI의 루트 볼륨이 이미 암호화된 경우 루트 볼륨을 다시 암호화하려면 AWS TNB`root_volume_encryption_key_arn`용를 포함해야 합니다.
> - AMI의 루트 볼륨이 암호화되지 않은 경우 AWS TNB는 `root_volume_encryption_key_arn`를 사용하여 루트 볼륨을 암호화합니다.
>
> 를 포함하지 않으면 `root_volume_encryption_key_arn` AWS TNB는 AWS Managed Services 를 사용하여 루트 볼륨을 암호화합니다.

- AWS TNB는 암호화된 AMI를 복호화하지 않습니다.

`root_volume_size`

GiBs.

필수 여부: 아니요

기본값: 20

유형: 정수

가능한 값: 1~16,384

`scaling`

Amazon EKS 자체 관리형 노드의 조정 매개 변수를 정의하는 속성(예: 원하는 Amazon EC2 인스턴스 수, 노드 그룹 내 최소 및 최대 Amazon EC2 인스턴스 수)입니다.

`desired_size`

이 노드 그룹의 인스턴스 수입니다.

필수 여부: 예

유형: 정수

`min_size`

이 노드 그룹의 최소 인스턴스 수입니다.

필수 여부: 예

유형: 정수

`max_size`

이 노드 그룹의 최대 인스턴스 수입니다.

필수 여부: 예

유형: 정수

속성

`node_role`

Amazon EC2 인스턴스에 연결된 IAM 역할의 ARN입니다.

필수 항목 여부: 예

유형: String

`tags`

리소스에 연결할 태그입니다. 태그는 리소스에서 생성한 인스턴스에 전파됩니다.

필수 여부: 아니요

유형: 목록

요구 사항

`cluster`

AWS.Compute.EKS 노드입니다.

필수 항목 여부: 예

유형: String

`subnets`

AWS.Networking.Subnet 노드입니다.

필수 여부: 예

유형: 목록

`network_interfaces`

AWS.Networking.ENI 노드입니다. 네트워크 인터페이스와 서브넷이 동일한 가용 영역으로 설정되어 있는지 확인하세요. 그렇지 않으면 인스턴스화가 실패합니다.

를 설정하면 속성을 AWS.Compute.EKS 노드에 포함`multus_role`하면 `network_interfaces` AWS TNB는 `multus` 속성에서 ENIs와 관련된 권한을 얻습니다. 그렇지 않으면 AWS TNB는 node_role 속성에서 ENI와 관련된 권한을 얻습니다.

필수 여부: 아니요

유형: 목록

`security_groups`

AWS.Networking.SecurityGroup 노드입니다.

필수 여부: 아니요

유형: 목록

`placement_group`

tosca.nodes.AWS.Compute.PlacementGroup 노드입니다.

필수 여부: 아니요

유형: 문자열

`user_data`

tosca.nodes.AWS.Compute.UserData 노드 참조입니다. 사용자 데이터 스크립트는 자체 관리형 노드 그룹에서 시작된 Amazon EC2 인스턴스로 전달됩니다. 사용자 지정 사용자 데이터를 실행하는 데 필요한 권한을 노드 그룹에 전달된 node_role에 추가합니다.

필수 여부: 아니요

유형: 문자열

`labels`

노드 레이블 목록입니다. 노드 레이블에는 이름과 값이 있어야 합니다. 다음 기준을 사용하여 레이블을 생성합니다.

- 이름과 값은 로 구분해야 합니다=.
- 이름과 값의 길이는 각각 최대 63자입니다.
- 레이블에는 문자(A~Z, a~z), 숫자(0~9) 및 다음 문자가 포함될 수 있습니다. `[-, _, ., *, ?]`
- 이름과 값은 영숫자, ?또는 * 문자로 시작하고 끝나야 합니다.

예: myLabelName1=*NodeLabelValue1

필수 여부: 아니요

유형: 목록

예제

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

```
      - "sampleLabelName001=sampleLabelValue001"
      - "sampleLabelName002=sampleLabelValue002"
```

AWS.Compute.PlacementGroup

PlacementGroup 노드는 Amazon EC2 인스턴스를 배치하기 위한 다양한 전략을 지원합니다.

새 Amazon EC2 인스턴스를 시작하면 Amazon EC2 서비스는 모든 인스턴스가 기본 하드웨어 전반에 분산되도록 하여 상호 관련 오류의 위험을 최소화합니다. 워크로드 요구 사항을 충족하기 위해 배치 그룹을 사용하여 상호 의존적 인스턴스 그룹의 배치에 영향을 줄 수 있습니다.

구문

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

속성

`strategy`

Amazon EC2 인스턴스를 배치하는 데 사용할 전략입니다.

필수 항목 여부: 예

유형: String

가능한 값: CLUSTER | PARTITION | SPREAD_HOST | SPREAD_RACK

- CLUSTER – 인스턴스를 가용 영역 안에 서로 근접하게 패킹합니다. 이 전략은 워크로드가 고성능 컴퓨팅(HPC) 애플리케이션에서 일반적인 긴밀히 결합된 노드 간 통신에 필요한 낮은 지연 시간의 네트워크 성능을 달성할 수 있습니다.
- PARTITION – 인스턴스를 논리적 파티션에 분산해, 한 파티션에 있는 인스턴스 그룹이 다른 파티션의 인스턴스 그룹과 기본 하드웨어를 공유하지 않게 합니다. 이 전략은 일반적으로 Hadoop, Cassandra, Kafka 등 대규모의 분산 및 복제된 워크로드에 필요합니다.
- SPREAD_RAC – 소규모의 인스턴스 그룹을 다른 기본 하드웨어로 분산하여 상호 관련 오류를 줄입니다.
- SPREAD_HOST – Outpost 배치 그룹에만 사용할 수 있습니다. 소규모의 인스턴스 그룹을 다른 기본 하드웨어로 분산하여 상호 관련 오류를 줄입니다.

`partition_count`

파티션 수입니다.

필수: `strategy`가 `PARTITION`으로 설정된 경우에만 필요합니다.

유형: 정수

가능한 값: 1 | 2 |3 | 4 | 5 | 6 | 7

`tags`

배치 그룹 리소스에 연결할 수 있는 태그입니다.

필수 여부: 아니요

유형: 목록

예제

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

AWS.Compute.UserData

AWS TNB는 Network Service Descriptor(NSD)의 UserData 노드를 통해 사용자 지정 사용자 데이터로 Amazon EC2 인스턴스 시작을 지원합니다. 사용자 지정 사용자 데이터에 대한 자세한 내용은 Amazon EC2 사용 설명서의 사용자 데이터 및 셸 스크립트를 참조하세요.

네트워크 인스턴스화 중에 AWS TNB는 사용자 데이터 스크립트를 통해 Amazon EC2 인스턴스 등록을 클러스터에 제공합니다. 사용자 지정 사용자 데이터도 제공되면 AWS TNB는 두 스크립트를 모두 병합하여 Amazon EC2에 멀티메 스크립트로 전달합니다. 사용자 지정 사용자 데이터 스크립트는 Amazon EKS 등록 스크립트보다 먼저 실행됩니다.

사용자 데이터 스크립트에서 사용자 지정 변수를 사용하려면 열린 중괄호 `{` 뒤에 느낌표 `!`를 추가하십시오. 예를 들어, 스크립트에서 `MyVariable`를 사용하려면 `{!MyVariable}`을 입력합니다.

> **Note**
>
> - AWS TNB는 최대 7KB 크기의 사용자 데이터 스크립트를 지원합니다.
> - AWS TNB는 CloudFormation 를 사용하여 `multimime` 사용자 데이터 스크립트를 처리하고 렌더링하므로 스크립트가 모든 CloudFormation 규칙을 준수하는지 확인합니다.

구문

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

속성

`implementation`

사용자 데이터 스크립트 정의의 상대 경로입니다. 형식은 `./scripts/script_name.sh`여야 합니다.

필수 항목 여부: 예

유형: String

`content_type`

사용자 데이터 스크립트의 콘텐츠 유형입니다.

필수 항목 여부: 예

유형: String

가능한 값: `x-shellscript`

예제

```
ExampleUserData:
  type: tosca.nodes.AWS.Compute.UserData
```

```
  properties:
    content_type: "text/x-shellscript"
    implementation: "./scripts/customUserData.sh"
```

AWS.Networking.SecurityGroup

AWS TNB는 보안 그룹을 지원하여 Amazon EKS Kubernetes 클러스터 노드 그룹에 연결할 수 있는 Amazon EC2 보안 그룹의 프로비저닝을 자동화합니다.

구문

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

속성

`description`

보안 그룹에 대한 설명입니다. 최대 255자의 문자를 사용하여 그룹을 설명할 수 있습니다. 여기에는 문자(A~Z 및 a~z), 숫자(0~9), 공백 및 특정 기호(._-:/()#,@[]+=&;{}!$*)만 사용할 수 있습니다.

필수 항목 여부: 예

유형: String

`name`

보안 그룹의 이름입니다. 이름에는 최대 255자까지 사용할 수 있습니다. 여기에는 문자(A~Z 및 a~z), 숫자(0~9), 공백 및 특정 기호(._-:/()#,@[]+=&;{}!$*)만 사용할 수 있습니다.

필수 항목 여부: 예

유형: String

`tags`

보안 그룹 리소스에 연결할 수 있는 태그입니다.

필수 여부: 아니요

유형: 목록

요구 사항

vpc

AWS.Networking.VPC 노드입니다.

필수 항목 여부: 예

유형: String

예제

```
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 TNB는 보안 그룹 송신 규칙을 지원하여 .Networking AWS.SecurityGroup에 연결할 수 있는 Amazon EC2 보안 그룹 송신 규칙의 프로비저닝을 자동화합니다. 단, 송신 트래픽의 대상으로 cidr_ip/destination_security_group/destination_prefix_list를 제공해야 합니다.

구문

```
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
```

속성

`cidr_ip`

IPv4 주소 범위(CIDR 형식)입니다. 송신 트래픽을 허용하는 CIDR 범위를 지정해야 합니다.

필수 여부: 아니요

유형: 문자열

`cidr_ipv6`

송신 트래픽의 경우 IPv6 주소 범위(CIDR 형식)입니다. 대상 보안 그룹(`destination_security_group` 또는 `destination_prefix_list`) 또는 CIDR 범위(`cidr_ip` 또는 `cidr_ipv6`)를 지정해야 합니다.

필수 여부: 아니요

유형: 문자열

`description`

송신(아웃바운드) 보안 그룹 규칙에 대한 설명입니다. 최대 255자의 문자를 사용하여 규칙을 설명할 수 있습니다.

필수 여부: 아니요

유형: 문자열

`destination_prefix_list`

기존 Amazon VPC 관리형 접두사 목록의 접두사 목록 ID입니다. 보안 그룹과 연결된 노드 그룹 인스턴스의 대상입니다. 관리형 접두사 목록에 대한 자세한 내용은 Amazon VPC 사용 설명서의 관리형 접두사 목록을 참조하세요.

필수 여부: 아니요

유형: 문자열

`from_port`

프로토콜이 TCP 또는 UDP인 경우 포트 범위의 시작입니다. 프로토콜이 ICMP 또는 ICMPv6인 경우 유형 번호입니다. 값 -1은 모든 ICMP/ICMPv6 형식을 나타냅니다. 모든 ICMP/ICMPv6 형식을 지정하는 경우 모든 ICMP/ICMPv6 코드를 지정해야 합니다.

필수 여부: 아니요

유형: 정수

`ip_protocol`

IP 프로토콜 이름(tcp, udp, icmp, icmpv6) 또는 프로토콜 번호입니다. -1을 사용하여 모든 프로토콜을 지정합니다. 보안 그룹 규칙의 승인 시 -1을 지정하거나 tcp, udp, icmp, 또는 icmpv6 이외의 프로토콜 번호를 지정하면 지정하는 포트 범위와 관계없이 모든 포트의 트래픽이 허용됩니다. tcp, udp, icmp의 경우 포트 범위를 지정해야 합니다. icmpv6의 경우 포트 범위는 선택 사항입니다. 포트 범위를 누락하는 경우 모든 형식 및 코드에 대한 트래픽이 허용됩니다.

필수 항목 여부: 예

유형: String

`to_port`

프로토콜이 TCP 또는 UDP인 경우 포트 범위의 끝입니다. 프로토콜이 ICMP 또는 ICMPv6인 경우 코드입니다. 값 -1은 모든 ICMP/ICMPv6 코드를 나타냅니다. 모든 ICMP/ICMPv6 형식을 지정하는 경우 모든 ICMP/ICMPv6 코드를 지정해야 합니다.

필수 여부: 아니요

유형: 정수

요구 사항

`security_group`

이 규칙이 추가되는 보안 그룹의 ID입니다.

필수 항목 여부: 예

유형: String

`destination_security_group`

송신 트래픽이 허용되는 대상 보안 그룹의 ID 또는 TOSCA 참조입니다.

필수 여부: 아니요

유형: 문자열

예제

```
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 TNB는 보안 그룹 수신 규칙을 지원하여 .Networking AWS.SecurityGroup에 연결할 수 있는 Amazon EC2 보안 그룹 수신 규칙의 프로비저닝을 자동화합니다. 단, 수신 트래픽의 소스로 cidr_ip/source_security_group/source_prefix_list를 제공해야 합니다.

구문

```
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
```

속성

`cidr_ip`

IPv4 주소 범위(CIDR 형식)입니다. 수신 트래픽을 허용하는 CIDR 범위를 지정해야 합니다.

필수 여부: 아니요

유형: 문자열

`cidr_ipv6`

수신 트래픽의 경우 IPv6 주소 범위(CIDR 형식)입니다. 소스 보안 그룹(`source_security_group` 또는 `source_prefix_list`)이나 CIDR 범위(`cidr_ip` 또는 `cidr_ipv6`)를 지정해야 합니다.

필수 여부: 아니요

유형: 문자열

`description`

수신(인바운드) 보안 그룹 규칙의 설명입니다. 최대 255자의 문자를 사용하여 규칙을 설명할 수 있습니다.

필수 여부: 아니요

유형: 문자열

`source_prefix_list`

기존 Amazon VPC 관리형 접두사 목록의 접두사 목록 ID입니다. 보안 그룹과 연결된 노드 그룹 인스턴스가 트래픽을 수신할 수 있는 소스입니다. 관리형 접두사 목록에 대한 자세한 내용은 Amazon VPC 사용 설명서의 관리형 접두사 목록을 참조하세요.

필수 여부: 아니요

유형: 문자열

`from_port`

프로토콜이 TCP 또는 UDP인 경우 포트 범위의 시작입니다. 프로토콜이 ICMP 또는 ICMPv6인 경우 유형 번호입니다. 값 -1은 모든 ICMP/ICMPv6 형식을 나타냅니다. 모든 ICMP/ICMPv6 형식을 지정하는 경우 모든 ICMP/ICMPv6 코드를 지정해야 합니다.

필수 여부: 아니요

유형: 정수

`ip_protocol`

IP 프로토콜 이름(tcp, udp, icmp, icmpv6) 또는 프로토콜 번호입니다. -1을 사용하여 모든 프로토콜을 지정합니다. 보안 그룹 규칙의 승인 시 -1을 지정하거나 tcp, udp, icmp, 또는 icmpv6 이외의 프로토콜 번호를 지정하면 지정하는 포트 범위와 관계없이 모든 포트의 트래픽이 허용됩니다. tcp, udp, icmp의 경우 포트 범위를 지정해야 합니다. icmpv6의 경우 포트 범위는 선택 사항입니다. 포트 범위를 누락하는 경우 모든 형식 및 코드에 대한 트래픽이 허용됩니다.

필수 항목 여부: 예

유형: String

`to_port`

프로토콜이 TCP 또는 UDP인 경우 포트 범위의 끝입니다. 프로토콜이 ICMP 또는 ICMPv6인 경우 코드입니다. 값 -1은 모든 ICMP/ICMPv6 코드를 나타냅니다. 모든 ICMP/ICMPv6 형식을 지정하는 경우 모든 ICMP/ICMPv6 코드를 지정해야 합니다.

필수 여부: 아니요

유형: 정수

요구 사항

`security_group`

이 규칙이 추가되는 보안 그룹의 ID입니다.

필수 항목 여부: 예

유형: String

`source_security_group`

수신 트래픽이 허용될 소스 보안 그룹의 ID 또는 TOSCA 참조입니다.

필수 여부: 아니요

유형: 문자열

예제

```
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

다음 AWS 리소스를 AWS TNB로 가져올 수 있습니다.

- VPC
- 서브넷
- 라우팅 테이블
- 인터넷 게이트웨이
- 보안 그룹

구문

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

속성

`resource_type`

AWS TNB로 가져온 리소스 유형입니다.

필수 여부: 아니요

유형: 목록

`resource_id`

AWS TNB로 가져온 리소스 ID입니다.

필수 여부: 아니요

유형: 목록

예제

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

AWS.Networking.ENI

네트워크 인터페이스는 VPC에서 가상 네트워크 카드를 나타내는 논리적 네트워킹 구성 요소입니다. 네트워크 인터페이스에는 서브넷을 기반으로 자동 또는 수동으로 IP 주소가 할당됩니다. Amazon EC2 인스턴스를 서브넷에 배포한 후 네트워크 인터페이스를 연결하거나, Amazon EC2 인스턴스에서 네트워크 인터페이스를 분리하고 해당 서브넷의 다른 Amazon EC2 인스턴스에 다시 연결할 수 있습니다. 디바이스 인덱스는 연결 순서에 따른 위치를 나타냅니다.

구문

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

속성

`device_index`

디바이스 인덱스는 0보다 커야 합니다.

필수 여부: 예

유형: 정수

`source_dest_check`

네트워크 인터페이스가 소스/대상 확인을 수행할지 여부를 나타냅니다. `true` 값은 확인이 활성화됨을 의미하며, `false` 값은 확인이 비활성화됨을 의미합니다.

허용된 값: true, false

기본값: true

필수 여부: 아니요

유형: 부울

`tags`

리소스에 연결할 태그입니다.

필수 여부: 아니요

유형: 목록

요구 사항

`subnet`

AWS.Networking.Subnet 노드입니다.

필수 항목 여부: 예

유형: String

`security_groups`

AWS.Networking.SecurityGroup 노드입니다.

필수 여부: 아니요

유형: 문자열

예제

```
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

수명 주기 후크를 사용하면 인프라 및 네트워크 인스턴스화의 일환으로 자체 스크립트를 실행할 수 있습니다.

구문

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

기능

execution

후크 스크립트를 실행하는 후크 실행 엔진의 속성입니다.

`type`

후크 실행 엔진 유형입니다.

필수 여부: 아니요

유형: 문자열

가능한 값: `CODE_BUILD`

요구 사항

`definition`

AWS.HookDefinition.Bash 노드입니다.

필수 항목 여부: 예

유형: String

`vpc`

AWS.Networking.VPC 노드입니다.

필수 항목 여부: 예

유형: String

예제

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

AWS.Networking.InternetGateway

AWS 인터넷 게이트웨이 노드를 정의합니다.

구문

```
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
```

기능

routing

VPC 내의 라우팅 연결을 정의하는 속성입니다. dest_cidr 또는 ipv6_dest_cidr 속성 중 하나를 포함해야 합니다.

dest_cidr

대상 일치에 사용되는 IPv4 CIDR 블록입니다. 이 속성은 RouteTable에 라우팅을 생성하는 데 사용되며 해당 값은 DestinationCidrBlock으로 사용됩니다.

필수: ipv6_dest_cidr 속성을 포함한 경우에는 아니요입니다.

유형: 문자열

ipv6_dest_cidr

대상 일치에 사용되는 IPv6 CIDR 블록입니다.

필수: dest_cidr 속성을 포함한 경우에는 아니요입니다.

유형: 문자열

속성

tags

리소스에 연결할 태그입니다.

필수 여부: 아니요

유형: 목록

`egress_only`

IPv6 전용 속성입니다. 인터넷 게이트웨이가 송신 전용인지 여부를 나타냅니다. `egress_only`가 true인 경우 `ipv6_dest_cidr` 속성을 정의해야 합니다.

필수 여부: 아니요

유형: 부울

요구 사항

`vpc`

AWS.Networking.VPC 노드입니다.

필수 항목 여부: 예

유형: String

`route_table`

AWS.Networking.RouteTable 노드입니다.

필수 항목 여부: 예

유형: String

예제

```
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

라우팅 테이블에는 VPC 또는 게이트웨이 내의 서브넷에서 네트워크 트래픽이 전송되는 위치를 결정하는 라우팅이라는 규칙 세트가 포함되어 있습니다. 라우팅 테이블을 VPC와 연결해야 합니다.

구문

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

속성

`tags`

리소스에 연결할 태그입니다.

필수 여부: 아니요

유형: 목록

요구 사항

`vpc`

AWS.Networking.VPC 노드입니다.

필수 항목 여부: 예

유형: String

예제

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

AWS.Networking.Subnet

서브넷은 VPC의 IP 주소 범위이며, 전적으로 하나의 가용 영역 내에 상주해야 합니다. 서브넷의 VPC, CIDR 블록, 가용 영역, 라우팅 테이블을 지정해야 합니다. 또한 서브넷이 프라이빗인지 퍼블릭인지도 정의해야 합니다.

구문

```
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
```

속성

`type`

이 서브넷에서 시작한 인스턴스가 퍼블릭 IPv4 주소를 수신할지 여부를 지정합니다.

필수 항목 여부: 예

유형: String

가능한 값: PUBLIC | PRIVATE

availability_zone

서브넷의 가용 영역입니다. 이 필드는 AWS 리전 내의 AWS 가용 영역(예: 미국 서부us-west-2(오레곤))을 지원합니다. 와 같이 가용 영역 내의 AWS 로컬 영역도 지원합니다us-west-2-lax-1a.

필수 항목 여부: 예

유형: String

cidr_block

서브넷에 대한 CIDR 블록입니다.

필수 여부: 아니요

유형: 문자열

ipv6_cidr_block

IPv6 서브넷을 생성하는 데 사용되는 CIDR 블록입니다. 이 속성을 포함하는 경우 ipv6_cidr_block_suffix를 포함하지 마십시오.

필수 여부: 아니요

유형: 문자열

ipv6_cidr_block_suffix

Amazon VPC를 통해 생성된 서브넷에 대한 IPv6 CIDR 블록의 2자리 16진수 접미사입니다. 다음 형식을 사용합니다. *2-digit hexadecimal*::/*subnetMask*

이 속성을 포함하는 경우 ipv6_cidr_block를 포함하지 마십시오.

필수 여부: 아니요

유형: 문자열

outpost_arn

서브넷 AWS Outposts 이 생성될의 ARN입니다. AWS Outposts에서 Amazon EKS 자체 관리형 노드를 시작하려는 경우 이 속성을 NSD 템플릿에 추가하세요. 자세한 내용은 Amazon EKS 사용 설명서의 AWS Outposts에 대한 Amazon EKS를 참조하세요.

이 속성을 NSD 템플릿에 추가하는 경우 availability_zone 속성 값을 AWS Outposts의 가용 영역으로 설정해야 합니다.

필수 여부: 아니요

유형: 문자열

`tags`

리소스에 연결할 태그입니다.

필수 여부: 아니요

유형: 목록

요구 사항

`vpc`

AWS.Networking.VPC 노드입니다.

필수 항목 여부: 예

유형: String

`route_table`

AWS.Networking.RouteTable 노드입니다.

필수 항목 여부: 예

유형: String

예제

```
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

네트워크 함수 배포는 관련 인프라 및 애플리케이션을 제공하여 모델링됩니다. 클러스터 속성은 네트워크 함수를 호스팅할 EKS 클러스터를 지정합니다. vnfs 속성은 배포를 위한 네트워크 함수를 지정합니다. 또한 pre_create 및 post_create 유형의 선택적 수명 주기 후크 작업을 제공하여 인벤토리 관리 시스템 API 호출과 같은 배포 관련 지침을 실행할 수 있습니다.

구문

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

요구 사항

`deployment`

AWS.Deployment.VNFDeployment 노드입니다.

필수 여부: 아니요

유형: 문자열

`cluster`

AWS.Compute.EKS 노드입니다.

필수 항목 여부: 예

유형: String

`vnfs`

AWS.VNF 노드입니다.

필수 항목 여부: 예

유형: String

인터페이스

후크

수명 주기 후크가 실행되는 단계를 정의합니다.

`pre_create`

AWS.HookExecution 노드입니다. 이 후크는 `VNFDeployment` 노드가 배포되기 전에 실행됩니다.

필수 여부: 아니요

유형: 문자열

`post_create`

AWS.HookExecution 노드입니다. 이 후크는 `VNFDeployment` 노드 배포 후에 실행됩니다.

필수 여부: 아니요

유형: 문자열

예제

```
SampleHelmDeploy:
  type: tosca.nodes.AWS.Deployment.VNFDeployment
  requirements:
```

```
    deployment: SampleHelmDeploy2
    cluster: SampleEKS
    vnfs:
      - vnf.SampleVNF
  interfaces:
    Hook:
      pre_create: SampleHook
```

AWS.Networking.VPC

Virtual Private Cloud(VPC)에 대한 CIDR 블록을 지정해야 합니다.

구문

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

속성

`cidr_block`

VPC에 대한 IPv4 네트워크 범위입니다(CIDR 표기).

필수 항목 여부: 예

유형: String

`ipv6_cidr_block`

VPC를 생성하는 데 사용된 IPv6 CIDR 블록입니다.

허용되는 값: `AMAZON_PROVIDED`

필수 여부: 아니요

유형: 문자열

`dns_support`

VPC에서 시작된 인스턴스가 DNS 호스트 이름을 가져오는지 나타냅니다.

필수 여부: 아니요

유형: 부울

기본값: `false`

`tags`

리소스에 연결할 태그입니다.

필수 여부: 아니요

유형: 목록

예제

```
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

서브넷을 통해 퍼블릭 또는 프라이빗 NAT 게이트웨이 노드를 정의할 수 있습니다. 퍼블릭 게이트웨이의 경우 탄력적 IP 할당 ID를 제공하지 않으면 AWS TNB는 계정에 탄력적 IP를 할당하고 이를 게이트웨이에 연결합니다.

구문

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

```
    tags: List
```

속성

`subnet`

AWS.Networking.Subnet 노드 참조입니다.

필수 항목 여부: 예

유형: String

`internet_gateway`

AWS.Networking.InternetGateway 노드 참조입니다.

필수 항목 여부: 예

유형: String

속성

`type`

게이트웨이가 퍼블릭인지 아니면 프라이빗인지를 나타냅니다.

허용되는 값: `PRIVATE`, `PUBLIC`

필수 항목 여부: 예

유형: String

`eip_allocation_id`

탄력적 IP 주소의 할당을 나타내는 ID입니다.

필수 여부: 아니요

유형: 문자열

`tags`

리소스에 연결할 태그입니다.

필수 여부: 아니요

유형: 목록

예제

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

AWS.Networking.Route

대상 라우팅을 NAT 게이트웨이에 대상 리소스로 연결하고 해당 라우팅을 연결된 라우팅 테이블에 추가하는 라우팅 노드를 정의할 수 있습니다.

구문

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

속성

`dest_cidr_blocks`

대상 리소스에 대한 대상 IPv4 경로 목록입니다.

필수 여부: 예

유형: 목록

멤버 유형: 문자열

요구 사항

`nat_gateway`

AWS.Networking.NATGateway 노드 참조입니다.

필수 항목 여부: 예

유형: String

`route_table`

AWS.Networking.RouteTable 노드 참조입니다.

필수 항목 여부: 예

유형: String

예제

```
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

AWS TNB를 통해 SSM 파라미터를 생성할 수 있습니다. 생성하는 SSM 파라미터는 SSM에서 생성되며 AWS TNB 네트워크 인스턴스 ID 접두사가 붙습니다. 이렇게 하면 동일한 NSD 템플릿을 사용하여 여러 인스턴스를 인스턴스화하고 업그레이드할 때 파라미터 값이 재정의되지 않습니다.

구문

```
tosca.nodes.AWS.Store.SSMParameters
  properties:
    parameters:
```

```
      name: String
      value: String
    tags: List
```

속성

파라미터

`name`

ssm 속성의 이름입니다. 다음 형식을 사용합니다. `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`

각 파라미터의 이름은 256자 미만이어야 합니다.

필수 항목 여부: 예

유형: String

`value`

ssm 속성의 값입니다. 다음 형식 중 하나를 사용합니다.

- 참조가 없는 값의 경우: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`
- 정적 참조의 경우: `^\$\{[a-zA-Z0-9]+\.(properties|capabilities|requirements)(\.([a-zA-Z0-9\-_]+))+\}$`
- 동적 참조의 경우: `^\$\{[a-zA-Z0-9]+\.(name|id|arn)\}$`

각 파라미터의 값은 4KB 미만이어야 합니다.

필수 항목 여부: 예

유형: String

`tags`

SSM 속성에 연결할 수 있는 태그입니다.

필수 여부: 아니요

유형: 목록

예제

```
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"
```

공통 노드

NSD 및 VNFD에 대한 노드를 정의합니다.

- AWS.HookDefinition.Bash

AWS.HookDefinition.Bash

에서 an AWS HookDefinition을 정의합니다bash.

구문

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

속성

`implementation`

후크 정의의 상대 경로입니다. 형식은 `./hooks/script_name.sh`여야 합니다.

필수 항목 여부: 예

유형: String

`environment_variables`

후크 bash 스크립트의 환경 변수입니다. 형식:를 다음 정규식 패턴과 **envName**=**envValue** 함께 사용합니다.

- 참조가 없는 값의 경우: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$`
- 정적 참조의 경우: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=\$\{[a-zA-Z0-9]+\.(properties|capabilities|requirements)(\.([a-zA-Z0-9\-_]+))+\}$`
- 동적 참조의 경우: `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=\$\{[a-zA-Z0-9]+\.(name|id|arn)\}$`

envName=**envValue** 값이 다음 기준을 충족해야 합니다.

- 공백은 사용하지 않습니다.
- **envName**은 문자(A-Z 또는 a-z) 또는 숫자(0-9)로 시작합니다.
- 환경 변수 이름을 다음과 같은 AWS TNB 예약어로 시작하지 않습니다(대/소문자를 구분하지 않음).
  - CODEBUILD
  - TNB
  - HOME
  - AWS
- **envName**과 **envValue**에는 원하는 수의 문자(A~Z 또는 a~z), 숫자 (0~9) 및 특수 문자(-, _)를 사용할 수 있습니다.
- 각 환경 변수(각각 **envName=envValue**)는 128자 미만이어야 합니다.

예시: `A123-45xYz=Example_789`

필수 여부: 아니요

유형: 목록

`execution_role`

후크를 실행하는 역할입니다.

필수 항목 여부: 예

유형: String

예제

```
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"
```

AWS TNB의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다 AWS 클라우드. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. AWS Telco Network Builder에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스 규정 준수 프로그램 .
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 AWS TNB를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표에 맞게 AWS TNB를 구성하는 방법을 보여줍니다. 또한 AWS TNB 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

내용

AWS TNB의 데이터 보호

AWS 공동 책임 모델 AWS Telco Network Builder의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프

라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 관한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 AWS TNB 또는 기타 AWS 서비스 에서 콘솔, API AWS CLI또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

태그 처리

AWS 계정을 닫으면 AWS TNB는 삭제를 위해 데이터를 표시하고 사용에서 제거합니다. 90일 이내에 AWS 계정을 다시 활성화하면 AWS TNB가 데이터를 복원합니다. 120일 후 AWS TNB는 데이터를 영구적으로 삭제합니다. AWS TNB는 또한 네트워크를 종료하고 함수 패키지와 네트워크 패키지를 삭제합니다.

저장 시 암호화

AWS TNB는 항상 추가 구성 없이 서비스에 저장된 모든 데이터를 암호화합니다. 이 암호화는 자동으로 이루어집니다 AWS Key Management Service.

전송 중 암호화

AWS TNB는 전송 계층 보안(TLS) 1.2를 사용하여 전송 중인 모든 데이터를 보호합니다.

시뮬레이션 에이전트와 클라이언트 간의 데이터를 암호화하는 것은 사용자의 책임입니다.

인터네트워크 트래픽 개인 정보 보호

AWS TNB 컴퓨팅 리소스는 모든 고객이 공유하는 Virtual Private Cloud(VPC)에 있습니다. 모든 내부 AWS TNB 트래픽은 AWS 네트워크 내에 유지되었으며 인터넷을 통과하지 않습니다. 시뮬레이션 에이전트와 클라이언트 간의 연결은 인터넷을 통해 라우팅됩니다.

AWS TNB의 ID 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 AWS TNB 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는 사용자를 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

내용

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 역할에 따라 다릅니다.

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한 요청(참조AWS Telco Network Builder 자격 증명 및 액세스 문제 해결)

- 서비스 관리자 - 사용자 액세스 결정 및 권한 요청 제출(AWS TNB가 IAM과 작동하는 방식 참조)
- IAM 관리자 - 액세스를 관리하기 위한 정책 작성(AWS Telco Network Builder의 자격 증명 기반 정책 예제 참조)

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증

명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

기타 정책 유형

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

AWS TNB가 IAM과 작동하는 방식

IAM을 사용하여 AWS TNB에 대한 액세스를 관리하기 전에 AWS TNB에서 사용할 수 있는 IAM 기능에 대해 알아봅니다.

AWS Telco Network Builder와 함께 사용할 수 있는 IAM 기능

IAM 특성	AWS TNB 지원
자격 증명 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키	예
ACL	아니요
ABAC(정책의 태그)	예
임시 보안 인증	예
엔터티 권한	예
서비스 역할	아니요
서비스 연결 역할	아니요

AWS TNB 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방식을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

AWS TNB에 대한 자격 증명 기반 정책

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

AWS TNB에 대한 자격 증명 기반 정책 예제

AWS TNB 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요AWS Telco Network Builder의 자격 증명 기반 정책 예제.

AWS TNB 내의 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는이 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM에서 교차 계정 리소스 액세스를 참조하세요.

AWS TNB에 대한 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

AWS TNB 작업 목록을 보려면 서비스 승인 참조의 AWS Telco Network Builder에서 정의한 작업을 참조하세요.

AWS TNB의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
tnb
```

단일 문에서 여러 작업을 지정하려면 쉼표로 구분합니다.

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

```
]
```

와일드카드(*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, `List`라는 단어로 시작하는 모든 작업을 지정하려면 다음 작업을 포함합니다.

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

AWS TNB 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요AWS Telco Network Builder의 자격 증명 기반 정책 예제.

AWS TNB에 대한 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

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

AWS TNB 리소스 유형 및 해당 ARNs 목록을 보려면 서비스 승인 참조의 AWS Telco Network Builder에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 AWS Telco Network Builder에서 정의한 작업을 참조하세요.

AWS TNB 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요AWS Telco Network Builder의 자격 증명 기반 정책 예제.

AWS TNB에 사용되는 정책 조건 키

서비스별 정책 조건 키 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소는 정의된 기준에 따라 문이 실행되는 시기를 지정합니다. 같음(equals) 또는 미만(less than)과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

AWS TNB 조건 키 목록을 보려면 서비스 승인 참조의 AWS Telco Network Builder에 사용되는 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 AWS Telco Network Builder에서 정의한 작업을 참조하세요.

AWS TNB 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요AWS Telco Network Builder의 자격 증명 기반 정책 예제.

AWS TNBACLs

ACL 지원: 아니요

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

AWS TNB를 사용한 ABAC

ABAC 지원(정책의 태그): 예

속성 기반 액세스 제어(ABAC)는 태그라고 불리는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. IAM 엔터티 및 AWS 리소스에 태그를 연결한 다음 보안 주체의 태그가 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계할 수 있습니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`또는 `aws:TagKeys`조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

AWS TNB에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

임시 자격 증명은 AWS 리소스에 대한 단기 액세스를 제공하며 페더레이션을 사용하거나 역할을 전환할 때 자동으로 생성됩니다. 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 것이 AWS 좋습니다. 자세한 내용은 IAM 사용 설명서의 IAM의 임시 보안 자격 증명 및 IAM으로 작업하는AWS 서비스 섹션을 참조하세요.

AWS TNB에 대한 교차 서비스 보안 주체 권한

전달 액세스 세션(FAS) 지원: 예

전달 액세스 세션(FAS)은를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

AWS TNB에 대한 서비스 역할

서비스 역할 지원: 아니요

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스 AWS에 권한을 위임할 역할 생성을 참조하세요.

AWS TNB에 대한 서비스 연결 역할

서비스 연결 역할 지원: 아니요

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 나타나 AWS 계정 며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

AWS Telco Network Builder의 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할에는 AWS TNB 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARNs 형식을 포함하여 AWS TNB에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 AWS Telco Network Builder에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

내용

- 정책 모범 사례
- AWS TNB 콘솔 사용
- 서비스 역할 정책 예제
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 AWS TNB 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

AWS TNB 콘솔 사용

AWS Telco Network Builder 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한은에서 AWS TNB 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다. 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

서비스 역할 정책 예제

관리자는 AWS TNB가 환경 및 서비스 템플릿에 정의된 대로 생성하는 리소스를 소유하고 관리합니다. AWS TNB가 네트워크 수명 주기 관리를 위한 리소스를 생성할 수 있도록 IAM 서비스 역할을 계정에 연결해야 합니다.

IAM 서비스 역할을 사용하면 AWS TNB가 사용자를 대신하여 리소스를 호출하여 네트워크를 인스턴스화하고 관리할 수 있습니다. 서비스 역할을 지정하면 AWS TNB는 해당 역할의 자격 증명을 사용합니다.

IAM 서비스를 사용하여 서비스 역할과 해당 권한 정책을 생성합니다. 서비스 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 권한을 위임할 역할 생성을 참조하세요.

AWS TNB 서비스 역할

플랫폼 팀의 구성원은 관리자로서 AWS TNB 서비스 역할을 생성하고 AWS TNB에 제공할 수 있습니다. 이 역할을 통해 AWS TNB는 Amazon Elastic Kubernetes Service와 같은 다른 서비스를 호출하고 네트워크에 필요한 인프라를 CloudFormation 프로비저닝하고 NSD에 정의된 대로 네트워크 기능을 프로비저닝할 수 있습니다.

AWS TNB 서비스 역할에는 다음 IAM 역할 및 신뢰 정책을 사용하는 것이 좋습니다. 이 정책에 대한 권한을 축소할 때는 AWS TNB가 정책에서 벗어나는 리소스에 대한 액세스 거부 오류와 함께 실패할 수 있다는 점에 유의하세요.

다음 코드는 AWS TNB 서비스 역할 정책을 보여줍니다.

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"
        }
    ]
}
```

다음 코드는 AWS TNB 서비스 신뢰 정책을 보여줍니다.

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 Amazon EKS 클러스터의 TNB 서비스 역할

NSD에서 Amazon EKS 리소스를 생성할 때는 `cluster_role` 속성을 제공하여 Amazon EKS 클러스터를 생성하는 데 사용할 역할을 지정합니다.

다음 예제에서는 Amazon EKS 클러스터 정책에 대한 AWS TNB 서비스 역할을 생성하는 AWS CloudFormation 템플릿을 보여줍니다.

```
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"
```

AWS CloudFormation 템플릿을 사용하는 IAM 역할에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 다음 섹션을 참조하세요.

- AWS::IAM::Role
- 스택 템플릿 선택

AWS Amazon EKS 노드 그룹에 대한 TNB 서비스 역할

NSD에서 Amazon EKS 노드 그룹 리소스를 생성할 때는 `node_role` 속성을 제공하여 Amazon EKS 노드 그룹을 생성하는 데 사용할 역할을 지정합니다.

다음 예제에서는 Amazon EKS 노드 그룹 정책에 대한 AWS TNB 서비스 역할을 생성하는 CloudFormation 템플릿을 보여줍니다.

```
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/*"
```

AWS CloudFormation 템플릿을 사용하는 IAM 역할에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 다음 섹션을 참조하세요.

- AWS::IAM::Role
- 스택 템플릿 선택

AWS Multus에 대한 TNB 서비스 역할

NSD에서 Amazon EKS 리소스를 생성하고 배포 템플릿의 일부로 Multus를 관리하려면 `multus_role` 속성을 제공하여 Multus 관리에 사용할 역할을 지정해야 합니다.

다음 예제에서는 Multus 정책에 대한 AWS TNB 서비스 역할을 생성하는 CloudFormation 템플릿을 보여줍니다.

```
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: "*"
```

AWS CloudFormation 템플릿을 사용하는 IAM 역할에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 다음 섹션을 참조하세요.

- AWS::IAM::Role
- 스택 템플릿 선택

AWS 수명 주기 후크 정책을 위한 TNB 서비스 역할

NSD 또는 네트워크 함수 패키지에서 수명 주기 후크를 사용하는 경우, 수명 주기 후크를 실행하기 위한 환경을 만들 수 있는 서비스 역할이 필요합니다.

> **Note**
>
> 수명 주기 후크 정책은 수명 주기 후크가 수행하려는 작업을 기반으로 해야 합니다.

다음 예제에서는 수명 주기 후크 정책에 대한 AWS TNB 서비스 역할을 생성하는 CloudFormation 템플릿을 보여줍니다.

```
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"
```

AWS CloudFormation 템플릿을 사용하는 IAM 역할에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 다음 섹션을 참조하세요.

- AWS::IAM::Role
- 스택 템플릿 선택

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

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

AWS Telco Network Builder 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 AWS TNB 및 IAM 작업 시 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

문제

- AWS TNB에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내 외부의 사람이 내 AWS TNB 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

AWS TNB에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 표시되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 tnb:*GetWidget* 권한이 없을 때 발생합니다.

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

이 경우 Mateo의 정책은 tnb:*GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스하도록 허용하도록 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 권한이 없다는 오류가 수신되면 AWS TNB에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예제 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 AWS TNB에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

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

이 경우, Mary가 iam:PassRole작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 AWS TNB 리소스 AWS 계정 에 액세스하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- AWS TNB가 이러한 기능을 지원하는지 여부를 알아보려면 섹션을 참조하세요AWS TNB가 IAM과 작동하는 방식.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

AWS TNB에 대한 규정 준수 검증

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 제공 범위 내를 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 AWS 서비스 결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 AWS 서비스참조하세요.

AWS TNB의 복원력

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.는 물리적으로 분리되고 격리된 여러 가용 영역을 AWS 리전 제공하며,이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

AWS TNB는 선택한 AWS 리전의 Virtual Private Cloud(VPC)에서 Network Service on EKS 클러스터를 실행합니다.

AWS TNB의 인프라 보안

관리형 서비스인 AWS Telco Network Builder는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안 모범 사례를 사용하여 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요 AWS .

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해 AWS TNB에 액세스합니다. 클라이언트는 다음을 지원해야 합니다.

- Transport Layer Security(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

다음은 공동 책임의 몇 가지 예입니다.

- AWS 는 다음을 포함하여 AWS TNB를 지원하는 구성 요소를 보호할 책임이 있습니다.

  - 컴퓨팅 인스턴스(작업자라고도 함)
  - 내부 데이터베이스
  - 내부 구성 요소 간 네트워크 통신
  - AWS TNB 애플리케이션 프로그래밍 인터페이스(API)
  - AWS 소프트웨어 개발 키트(SDK)
- 다음을 포함하여(이에 국한되지 않음) AWS 리소스 및 워크로드 구성 요소에 대한 액세스를 보호할 책임은 사용자에게 있습니다.
  - IAM 사용자, 그룹, 역할 및 정책
  - AWS TNB용 데이터를 저장하는 데 사용하는 S3 버킷
  - AWS TNB를 통해 프로비저닝한 네트워크 서비스를 지원하는 데 사용하는 기타 AWS 서비스 및 리소스
  - 애플리케이션 코드
  - AWS TNB를 통해 프로비저닝한 네트워크 서비스와 클라이언트 간의 연결

> **Important**
>
> AWS TNB를 통해 프로비저닝한 네트워크 서비스를 효과적으로 복구할 수 있는 재해 복구 계획을 구현하는 것은 사용자의 책임입니다.

네트워크 연결 보안 모델

AWS TNB를 통해 프로비저닝하는 네트워크 서비스는 선택한 AWS 리전에 위치한 Virtual Private Cloud(VPC) 내의 컴퓨팅 인스턴스에서 실행됩니다. VPC는 AWS 클라우드의 가상 네트워크로, 워크로드 또는 조직 엔터티별로 인프라를 격리합니다. VPC 내 컴퓨팅 인스턴스 간 통신은 AWS 네트워크 내에서 유지되며 인터넷을 통해 전달되지 않습니다. 일부 내부 서비스 통신은 인터넷을 통과하며 암호화됩니다. 동일한 리전에서 실행되는 모든 고객에 대해 AWS TNB를 통해 프로비저닝된 네트워크 서비스는 동일한 VPC를 공유합니다. 서로 다른 고객을 위해 AWS TNB를 통해 프로비저닝된 네트워크 서비스는 동일한 VPC 내에서 별도의 컴퓨팅 인스턴스를 사용합니다.

네트워크 서비스 클라이언트와 AWS TNB의 네트워크 서비스 간의 통신은 인터넷을 통과합니다. AWS TNB는 이러한 연결을 관리하지 않습니다. 클라이언트 연결을 보호하는 것은 사용자의 책임입니다.

AWS Management Console, AWS Command Line Interface (AWS CLI) 및SDK를 통해 AWS TNB에 대한 연결이 암호화됩니다. AWS SDKs

IMDS 버전

AWS TNB는 세션 지향 방법인 인스턴스 메타데이터 서비스 버전 2(IMDSv2)를 활용하는 인스턴스를 지원합니다. IMDSv2는 IMDSV1 보다 더 높은 보안을 제공합니다. 자세한 내용은 Amazon EC2 인스턴스 메타데이터 서비스의 향상된 기능을 통해 개방형 방화벽, 역방향 프록시 및 SSRF 취약성에 대한 심층적인 방어 기능 추가를 참조하세요.

인스턴스를 시작할 때는 IMDSv2를 사용해야 합니다. IMDSv2에 대한 자세한 내용은 https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/configuring-instance-metadata-service.htmlAmazon EC2 사용 설명서의 IMDSv2 사용을 참조하세요.

AWS TNB 모니터링

모니터링은 AWS TNB 및 기타 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다.는 AWS TNB를 모니터링하고, 이상이 있을 때 보고하고, 적절한 경우 자동 조치를 취할 수 AWS CloudTrail 있도록를 AWS 제공합니다.

CloudTrail을 사용하여 AWS APIs. Amazon S3에 이러한 호출을 로그 파일로 저장할 수 있습니다. 이러한 CloudTrail 로그를 사용하면 어떤 호출이 이루어졌는지, 호출한 소스 IP 주소, 호출한 사람, 호출한 시기 등의 정보를 확인할 수 있습니다.

CloudTrail 로그에는 AWS TNB에 대한 API 작업 호출에 대한 정보가 포함되어 있습니다. 또한 Amazon EC2 및 Amazon EBS와 같은 서비스에서 API 작업을 호출하기 위한 정보도 포함되어 있습니다.

를 사용하여 AWS Telco Network Builder API 호출 로깅 AWS CloudTrail

AWS Telco Network Builder는 사용자AWS CloudTrail, 역할 또는가 수행한 작업에 대한 레코드를 제공하는 서비스인와 통합됩니다 AWS 서비스. CloudTrail은 AWS TNB에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 AWS TNB 콘솔의 호출과 AWS TNB API 작업에 대한 코드 호출이 포함됩니다. CloudTrail에서 수집한 정보를 사용하여 AWS TNB에 수행된 요청, 요청이 수행된 IP 주소, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. 자격 증명을 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트 사용자로 했는지 사용자 보안 인증으로 했는지 여부.
- IAM Identity Center 사용자를 대신하여 요청이 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자의 임시 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부

CloudTrail은 계정을 생성할 AWS 계정 때에서 활성화되며 CloudTrail 이벤트 기록에 자동으로 액세스할 수 있습니다. CloudTrail 이벤트 기록은 지난 90일 간 AWS 리전의 관리 이벤트에 대해 보기, 검색 및 다운로드가 가능하고, 수정이 불가능한 레코드를 제공합니다. 자세한 설명은 AWS CloudTrail 사용 설명서의 CloudTrail 이벤트 기록 작업을 참조하세요. 이벤트 기록 보기는 CloudTrail 요금이 부과되지 않습니다.

AWS 계정 지난 90일 동안의 이벤트를 지속적으로 기록하려면 추적 또는 CloudTrail Lake 이벤트 데이터 스토어를 생성합니다.

CloudTrail 추적

CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 를 사용하여 생성된 모든 추적 AWS Management Console 은 다중 리전입니다. AWS CLI를 사용하여 단일 리전 또는 다중 리전 추적을 생성할 수 있습니다. 계정 AWS 리전 의 모든에서 활동을 캡처하므로 다중 리전 추적을 생성하는 것이 좋습니다. 단일 리전 추적을 생성하는 경우 추적의 AWS 리전에 로깅된 이벤트만 볼 수 있습니다. 추적에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS 계정에 대한 추적 생성 및 조직에 대한 추적 생성을 참조하세요.

CloudTrail에서 추적을 생성하여 진행 중인 관리 이벤트의 사본 하나를 Amazon S3 버킷으로 무료로 전송할 수는 있지만, Amazon S3 스토리지 요금이 부과됩니다. CloudTrail 요금에 관한 자세한 내용은 AWS CloudTrail 요금을 참조하세요. Amazon S3 요금에 관한 자세한 내용은 Amazon S3 요금을 참조하세요.

CloudTrail Lake 이벤트 데이터 스토어

CloudTrail Lake를 사용하면 이벤트에 대해 SQL 기반 쿼리를 실행할 수 있습니다. CloudTrail Lake는 행 기반 JSON 형식의 기존 이벤트를 Apache ORC 형식으로 변환합니다. ORC는 빠른 데이터 검색에 최적화된 열 기반 스토리지 형식입니다. 이벤트는 이벤트 데이터 스토어로 집계되며, 이벤트 데이터 스토어는 고급 이벤트 선택기를 적용하여 선택한 기준을 기반으로 하는 변경 불가능한 이벤트 컬렉션입니다. 이벤트 데이터 스토어에 적용하는 선택기는 어떤 이벤트가 지속되고 쿼리에 사용 가능한지를 제어합니다. CloudTrail Lake에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS CloudTrail Lake 작업을 참조하세요.

CloudTrail Lake 이벤트 데이터 스토어 및 쿼리에는 비용이 발생합니다. 이벤트 데이터 스토어를 생성할 때 이벤트 데이터 스토어에 사용할 요금 옵션을 선택합니다. 요금 옵션에 따라 이벤트 모으기 및 저장 비용과 이벤트 데이터 스토어의 기본 및 최대 보존 기간이 결정됩니다. CloudTrail 요금에 관한 자세한 내용은 AWS CloudTrail 요금을 참조하세요.

AWS TNB 이벤트 예제

이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청된 API 작업, 작업 날짜와 시간, 요청 파라미터 등에 관한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 추적이 아니므로 이벤트가 특정 순서로 표시되지 않습니다.

다음 예제는 `CreateSolFunctionPackage` 작업을 시연하는 CloudTrail 이벤트를 보여줍니다.

```
{
    "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"
}
```

CloudTrail 레코드 콘텐츠에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail record contents를 참조하세요.

AWS TNB 배포 작업

배포 태스크를 이해하여 배포를 효과적으로 모니터링하고 더 빠르게 조치를 취하세요.

다음 표에는 AWS TNB 배포 작업이 나열되어 있습니다.

2024년 3월 7일 이전에 시작된 배포의 작업 이름	2024년 3월 7일 이후에 시작된 배포의 작업 이름	[Task description]
AppInstallation	ClusterPluginInstall	Amazon EKS 클러스터에 Multus 플러그인을 설치합니다.
AppUpdate	이름 변경 없음	네트워크 인스턴스에 이미 설치된 네트워크 함수를 업데이트합니다.
-	ClusterPluginUninstall	Amazon EKS 클러스터에서 플러그인을 제거합니다.
ClusterStorageClassConfiguration	이름 변경 없음	Amazon EKS 클러스터에서 스토리지 클래스(CSI 드라이버)를 구성합니다.
FunctionDeletion	이름 변경 없음	AWS TNB 리소스에서 네트워크 함수를 삭제합니다.
FunctionInstantiation	FunctionInstall	HELM을 사용하여 네트워크 함수를 배포합니다.
FunctionUninstallation	FunctionUninstall	Amazon EKS 클러스터에서 네트워크 함수를 제거합니다.
HookExecution	이름 변경 없음	NSD에 정의된 대로 수명 주기 후크를 실행합니다.

2024년 3월 7일 이전에 시작된 배포의 작업 이름	2024년 3월 7일 이후에 시작된 배포의 작업 이름	[Task description]
InfrastructureCancellation	이름 변경 없음	네트워크 서비스를 취소합니다.
InfrastructureInstantation	이름 변경 없음	사용자를 대신하여 AWS 리소스를 프로비저닝합니다.
InfrastructureTermination	이름 변경 없음	AWS TNB를 통해 호출된 AWS 리소스를 프로비저닝 해제합니다.
-	InfrastructureUpdate	사용자를 대신하여 프로비저닝된 AWS 리소스를 업데이트합니다.
InventoryDeregistration	이름 변경 없음	AWS TNB에서 AWS 리소스를 등록 취소합니다.
-	InventoryRegistration	AWS TNB에 AWS 리소스를 등록합니다.
KubernetesClusterConfiguration	ClusterConfiguration	Kubernetes 클러스터를 구성하고 NSD에 정의된 대로 Amazon EKS AuthMap에 IAM 역할을 더 추가합니다.
NetworkServiceFinalization	이름 변경 없음	네트워크 서비스를 마무리하고 성공 또는 실패 상태 업데이트를 제공합니다.
NetworkServiceInstantiation	이름 변경 없음	네트워크 서비스를 초기화합니다.
SelfManagedNodesConfiguration	이름 변경 없음	Amazon EKS 및 Kubernetes 컨트롤 플레인을 사용하여 자체 관리형 노드를 부트스트랩합니다.
-	ValidateNetworkServiceUpdate	네트워크 인스턴스를 업데이트하기 전에 검증을 실행합니다.

AWS TNB에 대한 서비스 할당량

한도라고도 하는 서비스 할당량은 AWS 계정의 최대 서비스 리소스 또는 작업 수입니다. 자세한 내용은Amazon Web Services 일반 참조의 AWS Service Quotas를 참조하세요.

다음은 AWS TNB에 대한 서비스 할당량입니다.

명칭	기본값	조정 가능	설명
동시 진행 중인 네트워크 서비스 작업	지원되는 각 리전: 40	예	한 리전에서 네트워크 서비스를 동시에 실행할 수 있는 최대 개수입니다.
함수 패키지	지원되는 각 리전: 200	예	한 리전의 최대 함수 패키지 수입니다.
네트워크 패키지	지원되는 각 리전: 40	예	한 리전의 최대 네트워크 패키지 수입니다.
네트워크 서비스 인스턴스	지원되는 각 리전: 800	예	한 리전의 최대 네트워크 서비스 인스턴스 수입니다.

AWS TNB 사용 설명서의 문서 기록

다음 표에서는 AWS TNB에 대한 설명서 릴리스를 설명합니다.

변경 사항	설명	날짜
Amazon EKS 노드 그룹 네트워크 구성 업데이트	서브넷 및 보안 그룹을 추가 및 삭제합니다. 네트워크에서 ENIs를 추가, 수정 및 삭제합니다. 자세한 내용은 업데이트할 수 있는 파라미터를 참조하세요.	2025년 9월 10일
기존 클러스터에서 Amazon EKS 노드 그룹 추가 및 삭제	AWS TNB는 이제 Amazon EKS 클러스터에서 새 노드 그룹 추가 및 기존 노드 그룹 제거를 지원합니다. 자세한 내용은 업데이트할 수 있는 파라미터를 참조하세요.	2025년 6월 4일
루트 볼륨 크기	AWS.Compute.EKSManagedNode 및 .Compute.EKSAWS EKSSelfManagedNode 노드의 `root_volume_size` 필드를 통해 Amazon EKS 작업자 노드의 기본 Amazon EBS 루트 볼륨 크기를 지정할 수 있습니다.	2025년 5월 19일
스크립트의 참조 리소스	AWS TNB에서 생성한 리소스를 참조하여 수명 주기 후크 스크립트 및 사용자 데이터 스크립트에서 구성할 수 있습니다.	2025년 5월 2일
이제 Amazon EKS 노드 및 관리형 노드 그룹에 대해	AWS TNB는 .AWS Compute.EKS 및 .AWS Compute.E	2025년 4월 24일

Kubernetes 버전 1.32가 지원됩니다.	KSManagedNode용 Kubernetes 버전 1.32를 지원합니다.	
Amazon EKS 노드 및 관리형 노드 그룹에서는 Kubernetes 버전 1.24가 더 이상 지원되지 않습니다.	AWS TNB는 더 이상 .AWS Compute.EKS 및 .AWS Compute.EKSManagedNode용 Kubernetes 버전 1.24를 지원하지 않습니다.	2025년 4월 17일
Amazon EKS 관리형 노드에 대한 AL2023 AMI 지원	AWS TNB는 AWS.Compute.EKSManagedNode에 대한 AL2023 AMI 유형을 지원합니다.	2025년 4월 17일
Amazon EKS 노드 및 관리형 노드 그룹에 대해서는 Kubernetes 버전 1.23이 더 이상 지원되지 않습니다.	AWS TNB는 .AWS Compute.EKS 및 .AWS Compute.EKSManagedNode용 Kubernetes 버전 1.23을 더 이상 지원하지 않습니다.	2025년 4월 4일
AMI ID 업데이트 가능	이제 `UpdateSolNetworkService` API 호출 중에 ami_id 필드를 업데이트할 수 있습니다.	2025년 3월 31일
이제 Amazon EKS 노드 및 관리형 노드 그룹에 대해 Kubernetes 버전 1.31이 지원됩니다.	AWS TNB는 .AWS Compute.EKS 및 .AWS Compute.EKSManagedNode용 Kubernetes 버전 1.31을 지원합니다.	2025년 2월 18일
AWS용 Kubernetes 버전.Compute.EKSManagedNode	AWS TNB는 Kubernetes 버전 1.23~1.30을 지원하여 Amazon EKS 관리형 노드 그룹을 생성합니다.	2025년 1월 28일

클러스터용 Kubernetes 버전	AWS TNB는 이제 Amazon EKS 클러스터를 생성하기 위해 Kubernetes 버전 1.30을 지원합니다.	2024년 8월 19일

AWS TNB는 네트워크 수명 주기를 관리하기 위한 추가 작업을 지원합니다.	인스턴스화되거나 이전에 업데이트된 네트워크 인스턴스를 새 네트워크 패키지 및 파라미터 값으로 업데이트할 수 있습니다. 다음을 참조하세요.
	• 수명 주기 작업
	• 네트워크 인스턴스 업데이트
	• AWS TNB 서비스 역할 예제:
	  • Amazon EKS 작업 추가: `eks:UpdateAddon` , `eks:UpdateClusterVersion` , `eks:UpdateNodegroupConfig` , `eks:UpdateNodegroupVersion` , `eks:DescribeUpdate`
	  • 다음 CloudFormation 작업을 추가합니다. `cloudformation:UpdateStack`
	• 새 배포 작업: `InfrastructureUpdate` , `InventoryRegistration` , `ValidateNetworkServiceUpdate`
	• API 업데이트: GetSolNetworkOperation, ListSolNetworkOperations 및 UpdateSolNetworkInstance
		2024년 7월 30일

기존 작업에 대한 새 작업 및 새 작업 이름	새 작업을 사용할 수 있습니다. 2024년 3월 7일부터 일부 기존 작업에는 명확성을 위해 새 이름이 지정됩니다.	2024년 5월 7일
클러스터용 Kubernetes 버전	AWS TNB는 이제 Amazon EKS 클러스터를 생성하기 위해 Kubernetes 버전 1.29를 지원합니다.	2024년 4월 10일
네트워크 인터페이스 지원 `security_groups`	보안 그룹을 AWS.Networking.ENI 노드에 연결할 수 있습니다.	2024년 4월 2일
Amazon EBS 루트 볼륨 암호화 지원	Amazon EBS 루트 볼륨에 대해 Amazon EBS 암호화를 활성화할 수 있습니다. 를 활성화하려면 AWS.Compute.EKSManagedNode 또는 AWS.Compute.EKSSelfManagedNode 노드에 속성을 추가합니다.	2024년 4월 2일
노드 지원 `labels`	AWS.Compute.EKSManagedNode 또는 AWS.Compute.EKSSelfManagedNode 노드의 노드 그룹에 노드 레이블을 연결할 수 있습니다.	2024년 3월 19일
네트워크 인터페이스 지원 `source_dest_check`	.Networking AWS.ENI 노드를 통해 네트워크 인터페이스 소스/대상 확인을 활성화할지 여부를 지정할 수 있습니다.	2024년 1월 25일
사용자 지정 사용자 데이터가 있는 Amazon EC2 인스턴스 지원	AWS.Compute.UserData 노드를 통해 사용자 지정 사용자 데이터가 포함된 Amazon EC2 인스턴스를 시작할 수 있습니다.	2024년 1월 16일

보안 그룹에 대한 지원	AWS TNB를 사용하면 보안 그룹 AWS 리소스를 가져올 수 있습니다.	2024년 1월 8일
network_interfaces 의 설명을 업데이트함	network_interfaces 속성이 AWS.Compute.EKSManagedNode 또는 AWS.Compute.EKSSelfManagedNode 노드에 포함된 경우 AWS TNB는 사용 가능한 경우 multus_role 속성 또는 node_role 속성에서 ENIs와 관련된 권한을 가져옵니다.	2023년 12월 18일
프라이빗 클러스터 지원	AWS TNB는 이제 프라이빗 클러스터를 지원합니다. 프라이빗 클러스터를 나타내려면 access 속성을 PRIVATE으로 설정합니다.	2023년 12월 11일
클러스터용 Kubernetes 버전	AWS TNB는 이제 Amazon EKS 클러스터를 생성하기 위해 Kubernetes 버전 1.28을 지원합니다.	2023년 12월 11일
AWS TNB에서 배치 그룹 지원	AWS.Compute.EKSManagedNode 및 AWS.Compute.EKSSelfManagedNode 노드 정의를 위한 배치 그룹을 추가했습니다.	2023년 12월 11일

AWS TNB에 IPv6에 대한 지원 추가	AWS TNB는 이제 IPv6 인프라를 사용하여 네트워크 인스턴스 생성을 지원합니다. IPv6 구성용 노드 AWS.Networking.VPC, AWS.Networking.Subnet, AWS.Networking.InternetGateway, AWS.Networking.SecurityGroupIngressRule, AWS.Networking.SecurityGroupEgressRule, AWS.Compute.EKS를 확인합니다. 또한 NAT64 구성을 위한 노드 AWS.Networking.NATGateway and AWS.Networking.Route를 추가했습니다. IPv6 권한을 위해 Amazon EKS 노드 그룹에 대한 AWS TNB 서비스 역할과 AWS TNB 서비스 역할을 업데이트했습니다. 서비스 역할 정책 예시를 참조하십시오.	2023년 11월 16일
AWS TNB 서비스 역할 정책에 권한 추가	Amazon S3에 대한 AWS TNB 서비스 역할 정책 및 인프라 인스턴스화를 활성화 CloudFormation 할 수 있는 권한을 추가했습니다.	2023년 10월 23일
AWS 더 많은 리전에서 TNB 출시	AWS 이제 아시아 태평양(서울), 캐나다(중부), 유럽(스페인), 유럽(스톡홀름) 및 남아메리카(상파울루) 리전에서 TNB를 사용할 수 있습니다.	2023년 9월 27일

AWS.Compute.EKSSelfManagedNode에 대한 태그	AWS TNB는 이제 `AWS.Compute.EKSSelfManagedNode` 노드 정의에 대한 태그를 지원합니다.	2023년 8월 22일
AWS TNB는 IMDSv2를 활용하는 인스턴스를 지원합니다.	인스턴스를 시작할 때 IMDSv2를 사용해야 합니다.	2023년 8월 14일
MultusRoleInlinePolicy의 권한 업데이트	이제 MultusRoleInlinePolicy에 `ec2:DeleteNetworkInterface` 권한이 포함됩니다.	2023년 8월 7일
클러스터용 Kubernetes 버전	AWS TNB는 이제 Amazon EKS 클러스터를 생성하기 위해 Kubernetes 버전 1.27을 지원합니다.	2023년 7월 25일
AWS.Compute.EKS.AuthRole	AWS TNB는 사용자가 IAM 역할을 사용하여 Amazon EKS 클러스터에 액세스할 수 `aws-auth ConfigMap` 있도록 Amazon EKS 클러스터에 IAM 역할을 추가할 수 있는 AuthRole을 지원합니다.	2023년 7월 19일
AWS TNB는 보안 그룹을 지원합니다.	AWS.Networking.SecurityGroup, AWS.Networking.SecurityGroupEgressRule, AWS.Networking.SecurityGroupIngressRule을 NSD 템플릿에 추가했습니다.	2023년 7월 18일

클러스터용 Kubernetes 버전	AWS TNB는 Amazon EKS 클러스터를 생성하기 위해 Kubernetes 버전 1.22~1.26을 지원합니다. AWS TNB는 더 이상 Kubernetes 버전 1.21을 지원하지 않습니다.	2023년 5월 11일
AWS.Compute.EKSSelfManagedNode	리전 내, AWS 로컬 영역 및에서 자체 관리형 작업자 노드를 생성할 수 있습니다 AWS Outposts.	2023년 3월 29일
최초 릴리스	AWS TNB 사용 설명서의 첫 번째 릴리스입니다.	2023년 2월 21일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.