사용자 가이드

AWS 텔코 네트워크 빌더

AWS 텔코 네트워크 빌더: 사용자 가이드

Table of Contents

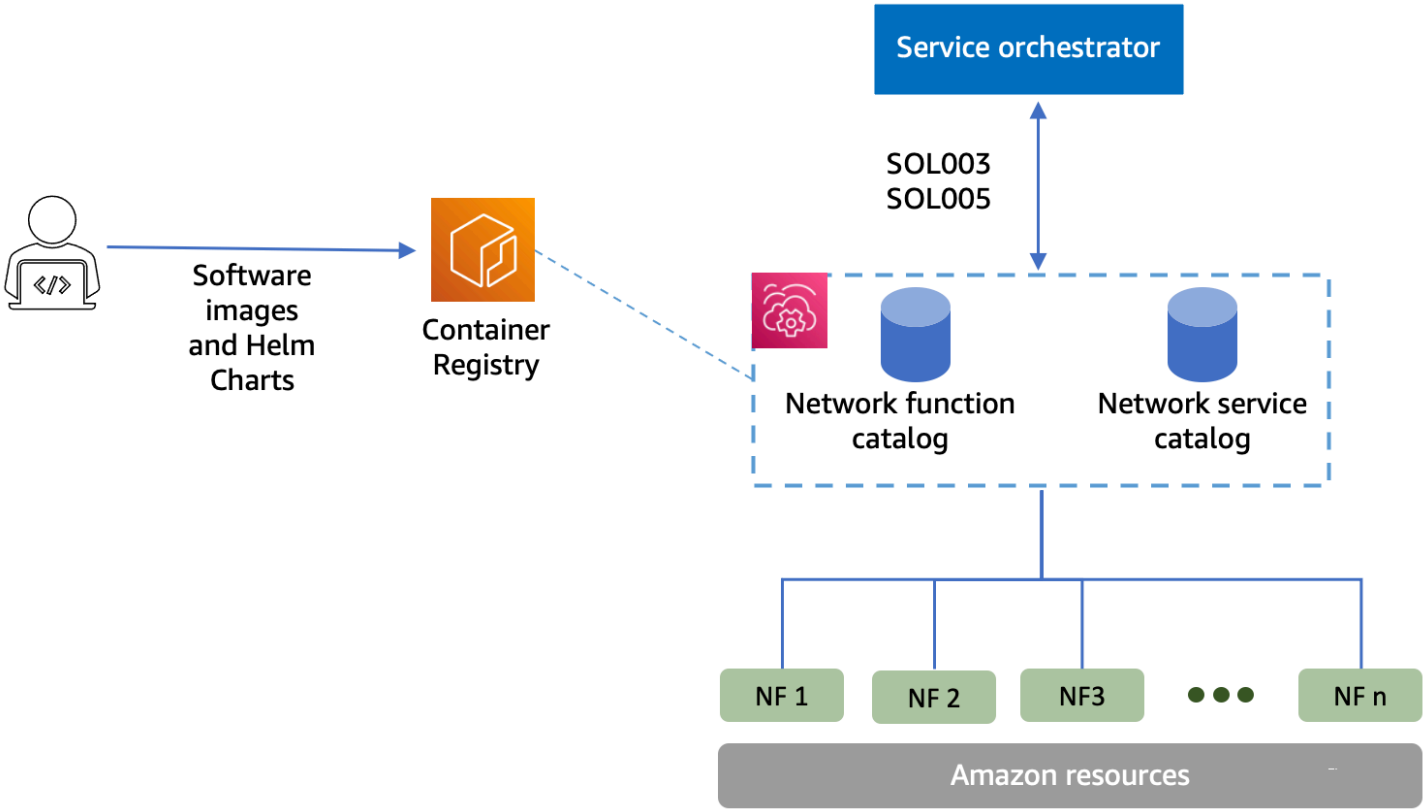
AWS Telco Network Builder(AWS TNB)란 무엇인가요?

AWS Telco Network Builder(AWS TNB)는 통신 서비스 제공업체(CSP)에게 AWS 인프라에서 5G 네트워크를 배포, 관리 및 조정할 수 있는 효율적인 방법을 제공하는 AWS 서비스입니다.

AWS TNB를 사용하면 컨테이너화된 소프트웨어 이미지를 사용하여 AWS 클라우드에 규모 조정이 가능하고 안전한 5G 네트워크를 자동화된 방식으로 배포할 수 있습니다. 새로운 기술을 배우거나, 사용할 컴퓨팅 서비스를 결정하거나, AWS 리소스를 프로비저닝하고 구성하는 방법을 익힐 필요가 없습니다.

대신 네트워크 인프라를 설명하고 독립 소프트웨어 개발 판매 회사(ISV) 파트너의 네트워크 함수에 대한 소프트웨어 이미지를 제공합니다. AWS TNB는 서드 파티 서비스 오케스트레이터 및 AWS 서비스와 통합되어 필요한 AWS 인프라를 자동으로 프로비저닝하고, 컨테이너화된 네트워크 함수를 배포하고, 네트워킹 및 액세스 관리를 구성하여 완벽하게 운영되는 네트워크 서비스를 생성합니다.

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



주제

- AWS가 처음이십니까?

AWS가 처음이십니까?

AWS 제품 및 서비스를 처음 사용하는 경우 자세한 내용은 다음 자료를 참조하세요.

- AWS 소개
- AWS 시작하기

AWS TNB는 누구를 대상으로 하나요?

AWS TNB는 네트워크 서비스를 설계, 배포 및 관리하기 위한 사용자 지정 스크립트와 구성을 작성하고 유지 관리하지 않고도 AWS 클라우드에서 제공하는 비용 효율성, 민첩성, 탄력성을 활용하려는 CSP를 위한 서비스입니다. TNB는 필요한 AWS 인프라를 자동으로 프로비저닝하고, 컨테이너화된 네트워크 함수를 배포하고, 네트워킹 및 액세스 관리를 구성하여 CSP가 정의한 네트워크 서비스 설명자와 CSP가 배포하려는 네트워크 함수를 기반으로 완벽하게 운영되는 네트워크 서비스를 생성합니다.

AWS TNB를 사용해야 하는 이유는 무엇인가요?

CSP가 AWS TNB를 사용하려는 몇 가지 이유는 다음과 같습니다.

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 서드 파티 서비스 오케스트레이터와 통합됩니다.

규모 조정

AWS TNB를 구성하여 기본 AWS 리소스를 확장하면 트래픽 수요를 충족하고, 네트워크 함수 업데이트를 보다 효율적으로 수행하고, 네트워크 인프라 토폴로지 변경을 적용하고, 새로운 5G 서비스의 배포 시간을 며칠에서 몇 시간으로 줄일 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 TNB가 자동으로 필요한 변경을 올바른 순서로 수행합니다.

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 TNB에 액세스

다음 인터페이스 중 하나를 사용하여 AWS TNB 리소스를 생성하고, 액세스하고, 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 포함하여 다양한 AWS 서비스 세트에 대한 명령을 제공합니다. 이는 Windows, macOS, Linux에서 지원됩니다. 자세한 내용은 AWS Command Line Interface 섹션을 참조하세요.
- AWS SDK - 언어별 API를 제공하고 많은 연결 세부 정보를 처리합니다. 서명 계산, 요청 재시도 처리 및 오류 처리가 여기에 포함됩니다. 자세한 정보는 AWS SDK를 참조하세요.

AWS TNB 요금

AWS TNB는 CSP가 AWS에 대한 통신 네트워크의 배포 및 관리를 자동화할 수 있도록 지원합니다. AWS TNB를 사용하면 다음 두 가지 차원에 대한 비용이 청구됩니다.

- 관리형 네트워크 함수 항목(MNFI) 시간
- API 요청 수

AWS TNB와 함께 다른 AWS 서비스를 사용할 때도 추가 요금이 부과됩니다. 자세한 내용은 AWS TNB 요금을 참조하세요.

청구 요금은 AWS Billing and Cost Management 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 결제 섹션을 참조하세요.

AWS 결제, 계정 및 이벤트에 관련된 질문은 AWS Support에 문의하세요.

AWS Trusted Advisor는 AWS 환경의 비용, 보안 및 성능을 최적화하는 데 도움이 될 수 있는 서비스입니다. 자세한 내용은 AWS Trusted Advisor를 참조하세요.

다음 단계

AWS TNB를 시작하는 방법에 대한 자세한 내용은 다음 주제를 참조하세요.

- AWS TNB 설정하기 – 사전 조건 단계를 완료합니다.
- AWS TNB와 함께 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

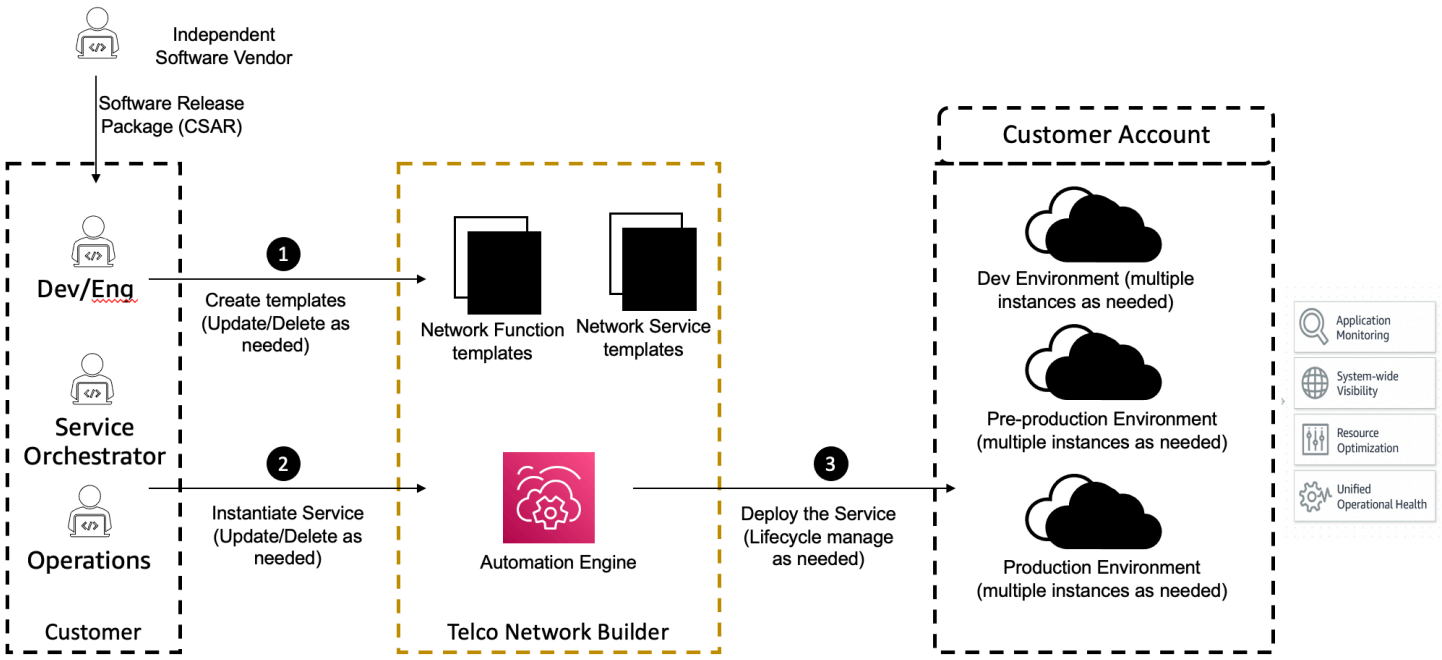
AWS TNB는 표준화된 엔드 투 엔드 오케스트레이터 및 AWS 리소스와 통합되어 완전한 5G 네트워크를 운영합니다.

AWS TNB는 네트워크 함수 패키지와 네트워크 서비스 설명자(NSD)를 수집하도록 해 주며 네트워크 운영을 위한 자동화 엔진을 제공합니다. 엔드 투 엔드 오케스트레이터를 사용하여 AWS TNB API와 통합하거나 AWS TNB SDK를 사용하여 자체 자동화 흐름을 구축할 수 있습니다. 자세한 내용은 AWS TNB 아키텍처 섹션을 참조하세요.

주제

AWS TNB 아키텍처

AWS TNB는AWS Management Console, AWS CLI, AWS TNB REST API 및 SDK를 통해 수명 주기 관리 작업을 수행할 수 있는 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 배포의 가시성을 제공합니다.



AWS 서비스와의 통합

5G 네트워크는 수천 개의 Kubernetes 클러스터에 배포된 상호 연결된 컨테이너식 네트워크 함수 세트로 구성됩니다. AWS TNB는 다음의 AWS 서비스와 같은 통신 전용 API와 통합되어 완벽하게 작동하는 네트워크 서비스를 생성합니다.

- 독립 소프트웨어 개발 판매 회사(ISV) 네트워크 함수 아티팩트를 저장하기 위한 Amazon Elastic Container Registry(Amazon ECR)
- 클러스터를 설정하기 위한 Amazon Elastic Kubernetes Service(Amazon EKS)
- 네트워킹 구성을 위한 Amazon VPC
- AWS CloudFormation을 사용하는 보안 그룹
- AWS 리전, AWS 로컬 영역, AWS Outposts의 배포 대상을 위한 AWS CodePipeline
- 역할을 정의하기 위한 IAM
- AWS TNB API에 대한 액세스를 제어하는 AWS Organizations
- 상태를 모니터링하고 지표를 게시하기 위한 AWS Health Dashboard 및 AWS CloudTrail

AWS TNB 리소스 할당량

AWS 계정에는 각 AWS 서비스 서비스에 대한 기본 할당량(이전에는 제한이라고 함)이 있습니다. 별도의 언급이 없다면 AWS 리전에 리전별로 각 할당량이 적용됩니다. 일부 할당량에 대한 증가를 요청할 수 있지만 모든 할당량에 대해 요청할 수 있는 것은 아닙니다.

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

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

AWS 계정에는 AWS TNB와 관련하여 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 컨셉

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

내용

네트워크 함수의 수명 주기

AWS TNB는 네트워크 기능의 수명 주기 전반에 걸쳐 도움을 줍니다. 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 기능을 정의하는 CSAR 패키지 (예: CU AMF, UPF) 를 업로드하고, 지속적 통합 및 지속적 전달 (CI/CD) 파이프라인과 통합하면 새 네트워크 기능 소프트웨어 이미지 또는 고객 스크립트가 제공될 때 CSAR 패키지의 새 버전을 만들 수 있습니다.

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).
- 네트워크 함수를 교체하거나 폐기합니다.

표준화된 인터페이스 사용

AWS TNB는 유럽 통신 표준 협회 (ETSI) 준수 서비스 오케스트레이터와 통합되어 네트워크 서비스 배포를 단순화할 수 있습니다. 서비스 오케스트레이터는 AWS TNB SDK, CLI 또는 API를 사용하여 네트워크 기능을 새 버전으로 인스턴스화하거나 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.

TNB용 AWS 네트워크 기능 패키지

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

- 네트워크 함수 디스크립터 - 패키지 온보딩 및 네트워크 함수 관리를 위한 메타데이터를 정의합니다.
- 소프트웨어 이미지 – 네트워크 함수 컨테이너 이미지를 참조합니다. Amazon Elastic Container Registry(Amazon ECR)는 네트워크 함수 이미지 리포지토리 역할을 할 수 있습니다.
- 추가 파일 – 네트워크 함수를 관리하는 데 사용합니다(예: 스크립트 및 차트 Helm).

CSAR은 OASIS TOSCA 표준에 정의된 패키지이며 OASIS TOSCA YAML 사양을 준수하는 네트워크/서비스 디스크립터를 포함합니다. 필수 YAML 사양에 대한 자세한 내용은 을 참조하십시오. AWS TNB에 대한 TOSCA 참조

다음은 네트워크 함수 디스크립터의 예제입니다.

```
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"
```

TNB용 네트워크 함수 서비스 디스크립터 AWS

AWS TNB는 배포하려는 네트워크 기능과 이를 카탈로그에 배포하려는 방법에 대한 네트워크 서비스 설명자 (NSD) 를 저장합니다. ETSI SOL007 설명에 따라 다음을 포함하는 YAML NSD 파일을 업로드할 수 있습니다.

- 배포하려는 네트워크 함수

- 네트워킹 지침
- 컴퓨팅 지침
- 수명 주기 후크(사용자 지정 스크립트)

AWS TNB는 TOSCA 언어로 네트워크, 서비스 및 기능과 같은 리소스를 모델링하기 위한 ETSI 표준을 지원합니다. AWS TNB를 사용하면 ETSI 호환 서비스 오케스트레이터가 이해할 수 있는 방식으로 AWS 서비스 모델링하여 더 효율적으로 사용할 수 있습니다.

다음은 모델링 방법을 보여주는 NSD 스니펫입니다. AWS 서비스네트워크 함수는 Kubernetes 버전 1.27이 설치된 Amazon EKS 클러스터에 배포됩니다. 애플리케이션용 서브넷은 Subnet01과 Subnet02입니다. 그런 다음 Amazon 머신 이미지 (AMI), 인스턴스 유형 및 자동 확장 구성을 사용하여 NodeGroups 애플리케이션에 맞게 를 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
```

TNB의 관리 및 AWS 운영

AWS TNB를 사용하면 ETSI SOL003 및 SOL005 규정에 따라 표준화된 관리 작업을 사용하여 네트워크를 관리할 수 있습니다. AWS TNB API를 사용하여 다음과 같은 라이프사이클 작업을 수행할 수 있습니다.

- 네트워크 함수 인스턴스화
- 네트워크 함수 종료
- Helm 배포를 재정의하도록 네트워크 함수를 업데이트
- 네트워크 함수 패키지 버전 관리
- NSD의 버전 관리
- 배포된 네트워크 함수에 대한 정보 검색

TNB용 네트워크 서비스 디스크립터 AWS

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 라이프사이클 작업을 구성하려면 TNB에서 지원하는 NSD TOSCA 스키마를 이해해야 합니다. AWS

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 이 름을 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
```

AWS TNB 설정하기

이 항목에 설명된 작업을 완료하여 AWS TNB를 설정합니다.

Tasks

- 가입하기 AWS
- AWS 지역 선택
- 서비스 엔드포인트를 기록해 둡니다.
- (선택 사항) 설치 AWS CLI
- IAM 사용자를 생성합니다.
- AWS TNB 역할 설정

가입하기 AWS

Amazon Web Services에 가입하면 AWS TNB를 포함하여 에 있는 AWS모든 서비스에 자동으로 AWS 계정 가입됩니다. 사용자에게는 사용한 서비스에 대해서만 요금이 청구됩니다.

AWS 계정 이미 등록한 경우 다음 작업으로 건너뛰십시오. AWS 계정이 없는 경우에는 다음 절차에 따라 계정을 만드세요.

생성하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 여세요.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중에는 전화를 받고 키패드로 인증 코드를 입력하는 과정이 있습니다.

   에 AWS 계정가입하면 AWS 계정 루트 사용자a가 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스 액세스 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 지역 선택

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

서비스 엔드포인트를 기록해 둡니다.

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

리전 이름	지역	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	지역	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 CLI시작하려면 AWS Command Line Interface 사용 설명서를 참조하십시오. AWS TNB 명령에 대한 자세한 내용은 명령 참조의 AWS CLI tnb를 참조하십시오.

IAM 사용자를 생성합니다.

AWS Identity and Access Management (IAM) 은 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 되는 웹 서비스입니다. AWS 단기 보안 인증 정보를 사용하여 AWS에 액세스하려면 IAM 사용자 역할을 생성합니다.

역할을 생성하려면 AWS IAM Identity Center 사용 설명서의 시작하기에 나온 지침을 따릅니다.

사용 AWS IAM Identity CenterAWS Command Line Interface 설명서에서 사용하도록 구성하여 프로그래밍 방식 액세스를 구성할 수도 있습니다. AWS CLI

AWS TNB 역할 설정

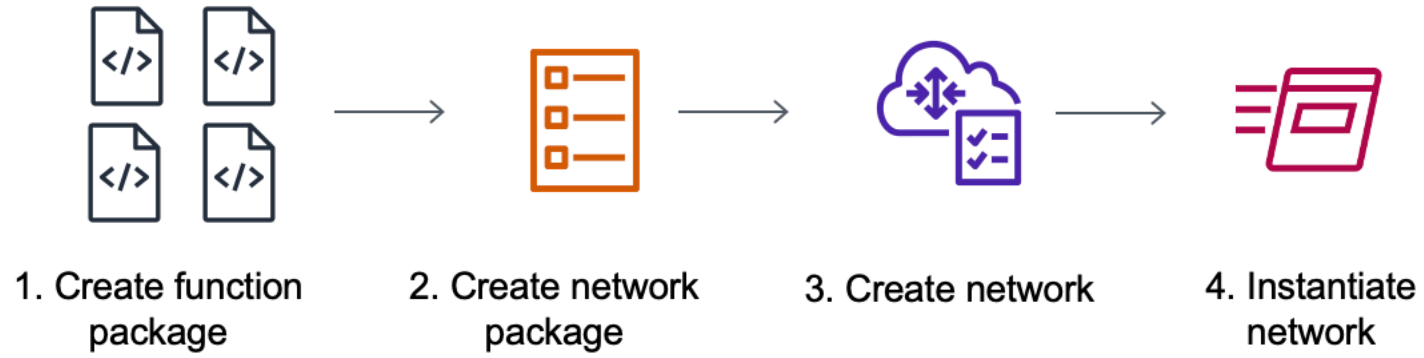
AWS TNB 솔루션의 여러 부분을 관리하려면 IAM 서비스 역할을 생성해야 합니다. AWS TNB 서비스 역할은 사용자를 대신하여 AWS CloudFormation AWS CodeBuild, 등의 다른 AWS 서비스와 다양한 컴퓨팅 및 스토리지 서비스에 API를 호출하여 배포에 필요한 리소스를 인스턴스화하고 관리할 수 있습니다.

AWS TNB 서비스 역할에 대한 자세한 내용은 을 참조하십시오. TNB의 ID 및 액세스 관리 AWS

AWS TNB와 함께 시작하기

이 자습서에서는 AWS TNB를 사용하여 중앙 장치 (CU), 액세스 및 이동성 관리 기능 (AMF) 또는 5G 사용자 플레인 기능 (UPF) 과 같은 네트워크 기능을 배포하는 방법을 보여줍니다.

다음 다이어그램은 배포 프로세스를 보여줍니다.



Tasks

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

필수 조건

배포를 성공적으로 수행하려면 먼저 다음 사항을 갖추어야 합니다.

- AWS 비즈니스 지원 플랜.
- IAM 역할을 통한 권한.
- ETSI SOL001/SOL004를 준수하는 네트워크 함수 (NF) 패키지입니다.
- ETSI SOL007 규정을 준수하는 네트워크 서비스 디스크립터 (NSD) 템플릿.

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

함수 패키지 생성

함수 패키지를 생성하려면

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

네트워크 패키지 생성

네트워크 패키지를 생성하려면

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

네트워크 인스턴스 생성 및 인스턴스화

네트워크 인스턴스를 생성 및 인스턴스화하려면

1. 탐색 창에서 네트워크를 선택합니다.
2. 네트워크 인스턴스 생성을 선택합니다.
3. 네트워크의 이름과 설명을 입력하고 다음을 선택합니다.
4. 해당 NSD를 선택합니다. 세부 정보를 확인한 후 다음을 선택합니다.

5. 네트워크 인스턴스 생성을 선택합니다. 초기 상태는 `Created`입니다.
6. 네트워크 인스턴스의 ID를 선택한 다음 인스턴스화를 선택합니다.
7. 네트워크 네트워크 인스턴스화를 선택합니다.
8. Refresh 아이콘을 사용하여 네트워크 인스턴스의 상태를 추적할 수 있습니다.

정리

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. Refresh 아이콘을 사용하여 네트워크 인스턴스의 상태를 추적할 수 있습니다.
5. (선택 사항) 네트워크를 선택한 후 삭제를 선택합니다.

AWS TNB에 대한 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.

태스크

- AWS TNB에서 함수 패키지를 생성하세요.
- AWS TNB에서 함수 패키지 보기
- AWS TNB에서 함수 패키지를 다운로드합니다.
- AWS TNB에서 함수 패키지를 삭제합니다.

AWS TNB에서 함수 패키지를 생성하세요.

AWS TNB 네트워크 함수 카탈로그에서 함수 패키지를 만드는 방법을 알아보세요. 함수 패키지를 생성하는 것은 TNB에서 네트워크를 생성하는 첫 번째 단계입니다. 함수 패키지를 업로드한 후에는 네트워크 패키지를 생성해야 합니다.

Console

콘솔을 사용하여 함수를 생성하려면

1. https://console.aws.amazon.com/tnb/에서 AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 함수 패키지를 선택합니다.
3. 함수 패키지 생성을 선택합니다.
4. 파일 선택을 선택하고 함수 패키지의 CSAR 패키지를 업로드합니다.
5. 다음을 선택합니다.
6. 패키지 세부 정보를 검토합니다.
7. 함수 패키지 생성을 선택합니다.

AWS CLI

AWS CLI를 사용하여 함수를 생성하려면

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. 예:

```
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/에서 AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 함수 패키지를 선택합니다.
3. 검색창을 사용하여 함수 패키지를 찾습니다.

AWS CLI

AWS CLI를 사용하여 함수 패키지를 보려면

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 패키지를 다운로드합니다.

AWS TNB 네트워크 함수 카탈로그에서 함수 패키지를 다운로드하는 방법을 알아보세요.

Console

콘솔을 사용하여 함수 패키지를 다운로드하려면

1. https://console.aws.amazon.com/tnb/에서 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

AWS CLI를 사용하여 함수 패키지를 다운로드하려면

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 패키지를 삭제합니다.

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

Console

콘솔을 사용하여 함수 패키지를 삭제하려면

1. https://console.aws.amazon.com/tnb/에서 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

AWS CLI를 사용하여 함수 패키지를 삭제하려면

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 TNB에 대한 네트워크 패키지

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

태스크

- 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/에서 AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크 패키지를 선택합니다.
3. 네트워크 패키지 생성을 선택합니다.
4. 파일 선택을 선택하고 CSAR 패키지를 업로드합니다.
5. 다음을 선택합니다.
6. 패키지 세부 정보를 검토합니다.
7. 네트워크 패키지 생성을 선택합니다.

AWS CLI

CLI를 사용하여 네트워크 패키지를 생성하려면

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. 예:

```
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/에서 AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크 패키지를 선택합니다.
3. 검색창을 사용하여 네트워크 패키지를 찾습니다.

AWS CLI

AWS CLI를 사용하여 네트워크 패키지를 보려면

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/에서 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

CLI를 사용하여 네트워크 패키지를 다운로드하려면

- 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/에서 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

AWS CLI를 사용하여 네트워크 패키지를 삭제하려면

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 TNB를 위한 네트워크 인스턴스

네트워크 인스턴스는 AWS TNB에서 생성되어 배포할 수 있는 단일 네트워크입니다.

태스크

- AWS TNB를 사용하여 네트워크 인스턴스를 인스턴스화
- AWS TNB에서 네트워크 인스턴스 보기
- AWS TNB에서 네트워크 인스턴스 업데이트
- AWS TNB에서 네트워크 인스턴스를 종료하고 삭제

AWS TNB를 사용하여 네트워크 인스턴스를 인스턴스화

네트워크 패키지를 생성한 후 네트워크 인스턴스를 생성합니다. 네트워크 인스턴스를 생성한 후에는 이를 인스턴스화해야 합니다. 네트워크 인스턴스를 인스턴스화할 때 AWS TNB는 네트워크 서비스 설명자의 사양에 따라 네트워크 함수를 배포합니다.

Console

콘솔을 사용하여 네트워크 인스턴스를 생성 및 인스턴스화하려면

1. https://console.aws.amazon.com/tnb/에서 AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크를 선택합니다.
3. 네트워크 인스턴스 생성을 선택합니다.
4. 인스턴스의 이름과 설명을 입력하고 다음을 선택합니다.
5. 해당 NSD를 선택합니다. 세부 정보를 확인한 후 다음을 선택합니다.
6. 네트워크 인스턴스 생성을 선택합니다.
7. 인스턴스화를 선택합니다.
8. 네트워크 인스턴스화를 선택합니다.
9. 네트워크 인스턴스의 상태를 추적하려면 새로 고침합니다.

AWS CLI

AWS CLI를 사용하여 네트워크 인스턴스를 생성 및 인스턴스화하려면

1. 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"
```

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

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

AWS TNB에서 네트워크 인스턴스 보기

네트워크 인스턴스를 보는 방법을 알아봅니다.

Console

콘솔을 사용하여 네트워크 인스턴스를 보려면

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

AWS CLI

AWS CLI를 사용하여 네트워크 인스턴스를 보려면

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

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

2. get-sol-network-instance 명령을 사용하여 네트워크 인스턴스에 대한 세부 정보를 볼 수 있습니다.

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

AWS TNB에서 네트워크 인스턴스 업데이트

네트워크 인스턴스를 업데이트하는 방법을 알아봅니다.

Console

콘솔을 사용하여 네트워크 인스턴스를 업데이트하려면

1. https://console.aws.amazon.com/tnb/에서 AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크를 선택합니다.
3. 네트워크 인스턴스의 ID를 선택합니다.
4. 함수 탭에서 업데이트할 함수 인스턴스를 선택합니다.
5. 업데이트를 선택합니다.
6. 업데이트 재정의를 입력하여 업데이트를 확인합니다.
7. 업데이트를 선택합니다.
8. 네트워크 인스턴스의 상태를 추적하려면 새로 고침합니다.

AWS CLI

CLI를 사용하여 네트워크 인스턴스 업데이트

update-sol-network-instance 명령을 사용하여 네트워크 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 인스턴스를 종료하고 삭제

네트워크 인스턴스를 삭제하려면 인스턴스가 종료 상태여야 합니다.

Console

콘솔을 사용하여 네트워크 인스턴스를 종료 및 삭제하려면

1. https://console.aws.amazon.com/tnb/에서 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

AWS CLI를 사용하여 네트워크 인스턴스를 종료 및 삭제하려면

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.

태스크

- 네트워크 작업 보기
- 네트워크 작업 취소

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/에서 AWS TNB 콘솔을 엽니다.
2. 탐색 창에서 네트워크 인스턴스를 선택합니다.
3. 검색창을 사용하여 네트워크 인스턴스를 찾습니다.
4. 배포 탭에서 네트워크 작업을 선택합니다.

AWS CLI

AWS CLI를 사용하여 네트워크 작업을 보려면

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}$
```

네트워크 작업 취소

네트워크 작업을 취소하는 방법을 알아봅니다.

Console

콘솔을 사용하여 네트워크 작업을 취소하려면

1. https://console.aws.amazon.com/tnb/에서 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

AWS CLI를 사용하여 네트워크 작업을 취소하려면

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 CSP는 템플릿을 AWS TNB에 업로드합니다. TNB는 AWS 가용 영역 전반에 걸쳐 제대로 작동하는 5G 네트워크를 구축하는 데 필요한 리소스를 종합합니다.

목차

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: 문자열

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: 문자열

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

`descriptor_name`

설명자의 이름입니다.

필수 항목 여부: 예

유형: 문자열

`provider`

VNFD의 작성자입니다.

필수 항목 여부: 예

유형: 문자열

요구 사항

`helm`

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

필수 항목 여부: 예

유형: 문자열

예

```
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: 문자열

예

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

NSD 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.EKS를 계산하십시오. AuthRole
- AWS.Compute.EKS ManagedNode
- AWS.Compute.EKS SelfManagedNode
- AWS. 컴퓨팅. PlacementGroup
- AWS.컴퓨팅. UserData
- AWS.네트워킹. SecurityGroup
- AWS.네트워킹. SecurityGroupEgressRule
- AWS.네트워킹. SecurityGroupIngressRule
- AWS.Resource.Import
- AWS.Networking.ENI
- AWS.HookExecution
- AWS.네트워킹. InternetGateway
- AWS.네트워킹. 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

예

```
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, Kubernetes 컨트롤 플레인이 NFS에 필요한 리소스를 관리할 수 있도록 하는 역할을 제공하십시오. AWS 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 사항입니다. 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

타입: 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 활성화하면 AWS Outposts, AWS Local Zones 또는 에서 Amazon EKS 자체 관리형 노드를 사용할 수 있습니다. AWS 리전자세한 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

타입: 부울

`version`

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

필수 여부: 아니요

타입: 문자열

속성

`version`

클러스터용 쿠버네티스 버전. AWS 텔코 네트워크 빌더는 쿠버네티스 버전 1.23~1.29를 지원합니다.

필수 항목 여부: 예

유형: String

가능한 값: 1.23 | 1.24 | 1.25 | 1.26 | 1.27 | 1.28 | 1.29

`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

타입: 문자열

요구 사항

subnets

AWS.Networking.Subnet 노드입니다.

필수 여부: 예

유형: 목록

예

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

AWS.Compute.EKS. AuthRole

An을 AuthRole 사용하면 사용자가 IAM 역할을 사용하여 Amazon EKS 클러스터에 액세스할 수 `aws-auth ConfigMap` 있도록 Amazon EKS 클러스터에 IAM 역할을 추가할 수 있습니다.

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 ConfigMap`에 추가해야 하는 IAM 역할을 정의하는 매핑 목록입니다.

`arn`

IAM 역할의 ARN입니다.

필수 항목 여부: 예

유형: String

`groups`

`arn`에 정의된 역할에 할당할 Kubernetes 그룹입니다.

필수 여부: 아니요

유형: 목록

요구 사항

`clusters`

AWS.Compute.EKS 노드입니다.

필수 여부: 예

유형: 목록

예

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

AWS.Compute.EKS ManagedNode

AWS TNB는 Amazon EKS 쿠버네티스 클러스터의 노드 (Amazon EC2 인스턴스) 프로비저닝 및 수명 주기 관리를 자동화하는 EKS 관리형 노드 그룹을 지원합니다. EKS 노드 그룹을 생성하려면 AMI ID 또는 AMI 유형을 제공하여 클러스터 워커 노드의 Amazon Machine Image(AMI)를 선택해야 합니다. 또한 SSH 액세스를 위한 Amazon EC2 키 쌍과 노드 그룹의 조정 속성을 제공해야 합니다. 노드 그룹은 EKS 클러스터와 연결되어야 합니다. 워커 노드에 서브넷을 제공해야 합니다.

선택적으로 보안 그룹, 노드 레이블 및 배치 그룹을 노드 그룹에 연결할 수 있습니다.

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

`ami_type`

아마존 EKS가 지원하는 AMI 유형입니다.

필수 항목 여부: 예

유형: String

가능한 값: `AL2_x86_64` | `AL2_x86_64_GPU` | `AL2_ARM_64` | `CUSTOM` | `BOTTLEROCKET_ARM_64` | `BOTTLEROCKET_x86_64` | `BOTTLEROCKET_ARM_64_NVIDIA` | `BOTTLEROCKET_x86_64_NVIDIA`

`ami_id`

AMI의 ID입니다.

필수 여부: 아니요

타입: 문자열

> **Note**
>
> 템플릿에서 `ami_type` 와 를 `ami_id` 모두 지정한 경우 AWS TNB는 해당 `ami_id` 값만 사용하여 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 속성을 제공하지 않으면 AWS TNB는 기본적으로 Amazon EBS 루트 볼륨을 암호화합니다.

필수 항목 여부: 아니요

기본값: true

타입: 부울

`root_volume_encryption_key_arn`

키의 ARN입니다. AWS KMS AWS TNB는 일반 키 ARN, 다중 지역 키 ARN 및 별칭 ARN을 지원합니다.

필수 여부: 아니요

타입: 문자열

> **Note**
>
> - `root_volume_encryption`거짓이면 포함하지 마십시오. `root_volume_encryption_key_arn`
> - AWS TNB는 Amazon EBS 기반 AMI의 루트 볼륨 암호화를 지원합니다.
> - AMI의 루트 볼륨이 이미 암호화된 경우 루트 볼륨을 다시 암호화하려면 AWS TNB를 포함해야 합니다. `root_volume_encryption_key_arn`
> - AMI의 루트 볼륨이 암호화되지 않은 경우 AWS TNB는 `root_volume_encryption_key_arn` 를 사용하여 루트 볼륨을 암호화합니다.
>
>   포함하지 `root_volume_encryption_key_arn` 않으면 AWS TNB는 에서 제공한 AWS Key Management Service 기본 키를 사용하여 루트 볼륨을 암호화합니다.
> - AWS TNB는 암호화된 AMI를 해독하지 않습니다.

`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`

여기에 있는 인스턴스의 수입니다. NodeGroup

필수 여부: 예

유형: 정수

`min_size`

이 항목의 최소 인스턴스 수입니다 NodeGroup.

필수 여부: 예

유형: 정수

`max_size`

이 항목의 최대 인스턴스 수입니다 NodeGroup.

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

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.

설정하면 network_interfacesAWS AWS.Compute.eks 노드에 multus_role 속성을 포함한 경우 TNB는 속성으로부터 ENI와 multus 관련된 권한을 얻습니다. 그렇지 않으면 AWS TNB는 node_role 속성에서 ENI와 관련된 권한을 얻습니다.

필수 여부: 아니요

유형: 목록

`security_groups`

.네트워킹.AWS SecurityGroup노드.

필수 여부: 아니요

유형: 목록

`placement_group`

토스카. 노드.AWS. 컴퓨팅. PlacementGroup노드.

필수 여부: 아니요

타입: 문자열

`user_data`

토스카. 노드.AWS. 컴퓨팅. UserData노드 레퍼런스. 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

타입: 문자열

`labels`

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

- 이름과 값은 로 구분해야 합니다=.
- 이름과 값은 각각 최대 63자까지 입력할 수 있습니다.
- 라벨에는 문자 (A-Z, a-z,), 숫자 (0-9) 및 다음 문자를 포함할 수 있습니다. `[-, _, ., *, ?]`
- 이름과 값은 영숫자 또는 문자로 시작하고 끝나야 합니다. ? *

예제: `myLabelName1=*NodeLabelValue1`

필수 여부: 아니요

유형: 목록

예

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

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

AWS.eks를 계산합니다. SelfManagedNode

AWS TNB는 Amazon EKS 자체 관리형 노드를 지원하여 Amazon EKS 쿠버네티스 클러스터용 노드(Amazon EC2 인스턴스) 의 프로비저닝 및 수명 주기 관리를 자동화합니다. Amazon EKS 노드 그룹을 생성하려면 AMI ID를 제공하여 클러스터 워커 노드의 Amazon Machine Image(AMI)를 선택해야 합니다. 선택적으로 SSH 액세스를 위한 Amazon EC2 키 쌍도 제공할 수 있습니다. 또한 인스턴스 유형과 원하는 크기, 최소 크기, 최대 크기를 제공해야 합니다. 노드 그룹은 Amazon EKS 클러스터와 연결되어야 합니다. 워커 노드에 서브넷을 제공해야 합니다.

선택적으로 보안 그룹, 노드 레이블 및 배치 그룹을 노드 그룹에 연결할 수 있습니다.

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
    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 사용된 AMI ID. AWS TNB는 IMDSv2를 활용하는 인스턴스를 지원합니다. 자세한 정보는 IMDS 버전을 참조하세요.

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 속성을 제공하지 않으면 AWS TNB는 기본적으로 Amazon EBS 루트 볼륨을 암호화합니다.

필수 항목 여부: 아니요

기본값: true

타입: 부울

`root_volume_encryption_key_arn`

키의 ARN입니다. AWS KMS AWS TNB는 일반 키 ARN, 다중 지역 키 ARN 및 별칭 ARN을 지원합니다.

필수 여부: 아니요

타입: 문자열

> **Note**
>
> - `root_volume_encryption`거짓이면 포함하지 마십시오. `root_volume_encryption_key_arn`
> - AWS TNB는 Amazon EBS 기반 AMI의 루트 볼륨 암호화를 지원합니다.
> - AMI의 루트 볼륨이 이미 암호화된 경우 루트 볼륨을 다시 암호화하려면 AWS TNB를 포함해야 합니다. `root_volume_encryption_key_arn`
> - AMI의 루트 볼륨이 암호화되지 않은 경우 AWS TNB는 `root_volume_encryption_key_arn` 를 사용하여 루트 볼륨을 암호화합니다.
>
>   포함하지 `root_volume_encryption_key_arn` 않으면 AWS TNB는 루트 볼륨을 암호화하는 AWS Managed Services 데 사용합니다.
> - AWS TNB는 암호화된 AMI를 해독하지 않습니다.

`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`

여기에 있는 인스턴스의 수입니다. NodeGroup

필수 여부: 예

유형: 정수

`min_size`

이 항목의 최소 인스턴스 수입니다 NodeGroup.

필수 여부: 예

유형: 정수

`max_size`

이 항목의 최대 인스턴스 수입니다 NodeGroup.

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.

설정하면 `network_interfaces`AWS AWS.Compute.eks 노드에 `multus_role` 속성을 포함한 경우 TNB는 속성으로부터 ENI와 `multus` 관련된 권한을 얻습니다. 그렇지 않으면 AWS TNB는 node_role 속성에서 ENI와 관련된 권한을 얻습니다.

필수 여부: 아니요

유형: 목록

`security_groups`

.네트워킹.AWS SecurityGroup노드.

필수 여부: 아니요

유형: 목록

`placement_group`

토스카. 노드.AWS. 컴퓨팅. PlacementGroup노드.

필수 여부: 아니요

타입: 문자열

`user_data`

토스카. 노드.AWS. 컴퓨팅. UserData노드 레퍼런스. 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

타입: 문자열

`labels`

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

- 이름과 값은 로 구분해야 합니다=.

- 이름과 값은 각각 최대 63자까지 입력할 수 있습니다.
- 레이블에는 문자 (A-Z, a-z,), 숫자 (0-9) 및 다음 문자를 포함할 수 있습니다. `[-, _, ., *, ?]`
- 이름과 값은 영숫자 또는 문자로 시작하고 끝나야 합니다. `? *`

예제: `myLabelName1=*NodeLabelValue1`

필수 여부: 아니요

유형: 목록

예

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

```
      - SampleSecurityGroup02
    placement_group: SamplePlacementGroup
    user_data: CustomUserData
    labels:
      - "sampleLabelName001=sampleLabelValue001"
      - "sampleLabelName002=sampleLabelValue002"
```

AWS. 컴퓨팅. 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 전략을 통해 워크로드는 고성능 컴퓨팅 (HPC) 애플리케이션에서 흔히 볼 수 있는 밀접하게 연결된 node-to-node 통신에 필요한 저지연 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

예

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

AWS.컴퓨팅. UserData

AWS TNB는 네트워크 서비스 설명자 (NSD) UserData 의 노드를 통해 사용자 지정 사용자 데이터가 포함된 Amazon EC2 인스턴스 시작을 지원합니다. 사용자 지정 사용자 데이터에 대한 자세한 내용은 Amazon EC2 사용 설명서의 사용자 데이터 및 셸 스크립트를 참조하십시오.

네트워크 인스턴스화 중에 AWS TNB는 사용자 데이터 스크립트를 통해 클러스터에 Amazon EC2 인스턴스 등록을 제공합니다. 사용자 지정 사용자 데이터도 제공되면 AWS TNB는 두 스크립트를 병합

하여 이를 멀티마임 스크립트로 Amazon EC2에 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 `multimime` 사용자 데이터 스크립트를 처리하고 AWS CloudFormation 렌더링하는 데 사용하므로 스크립트가 모든 규칙을 준수하는지 확인하십시오. AWS CloudFormation

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`

예

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

AWS.네트워킹. SecurityGroup

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

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

예

```
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.네트워킹. SecurityGroupEgressRule

AWS TNB는 .Networking에 연결할 수 있는 Amazon EC2 보안 그룹 송신 규칙의 프로비저닝을 자동화하는 보안 그룹 송신 규칙을 지원합니다. AWS SecurityGroup. 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

타입: 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

타입: 문자열

`description`

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

필수 여부: 아니요

타입: 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

타입: 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

타입: 문자열

예

```
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. 네트워킹. SecurityGroupIngressRule

AWS TNB는 .Networking에 연결할 수 있는 Amazon EC2 보안 그룹 수신 규칙의 프로비저닝을 자동화하는 보안 그룹 수신 규칙을 지원합니다. AWS SecurityGroup. 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

타입: 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

타입: 문자열

`description`

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

필수 여부: 아니요

타입: 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

타입: 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

타입: 문자열

예

```
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 가져오는 리소스 유형입니다.

필수 여부: 아니요

유형: 목록

`resource_id`

AWS TNB로 가져온 리소스 ID.

필수 여부: 아니요

유형: 목록

예

```
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 검사를 수행하는지 여부를 나타냅니다. `true` 값은 확인이 활성화됨을 의미하며, `false` 값은 확인이 비활성화됨을 의미합니다.

허용된 값: true, false

기본값: true

필수 여부: 아니요

타입: 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 A. 네트워킹. SecurityGroup노드.

필수 여부: 아니요

타입: 문자열

예

```
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

타입: 문자열

가능한 값: `CODE_BUILD`

요구 사항

`definition`

그리고 AWS. HookDefinition.Bash 노드.

필수 항목 여부: 예

유형: String

`vpc`

AWS.Networking.VPC 노드입니다.

필수 항목 여부: 예

유형: String

예

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

AWS. 네트워킹. 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.

타입: 문자열

ipv6_dest_cidr

대상 일치에 사용되는 IPv6 CIDR 블록입니다.

필수: dest_cidr 속성을 포함한 경우에는 아니요입니다.

타입: 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

타입: 부울

요구 사항

`vpc`

AWS.Networking.VPC 노드입니다.

필수 항목 여부: 예

유형: String

`route_table`

AWS.네트워킹. RouteTable노드.

필수 항목 여부: 예

유형: String

예

```
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. 네트워킹. 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

예

```
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 지역 내 AWS 가용 영역을 지원합니다 `us-west-2` (예: 미국 서부 (오레곤)). 예를 들어 가용 영역 내의 AWS Local Zone도 지원합니다`us-west-2-lax-1a`.

필수 항목 여부: 예

유형: String

`cidr_block`

서브넷에 대한 CIDR 블록입니다.

필수 여부: 아니요

타입: 문자열

`ipv6_cidr_block`

IPv6 서브넷을 생성하는 데 사용되는 CIDR 블록입니다. 이 속성을 포함하는 경우 `ipv6_cidr_block_suffix`를 포함하지 마십시오.

필수 여부: 아니요

타입: 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_block`를 포함하지 마십시오.

필수 여부: 아니요

타입: 문자열

`outpost_arn`

서브넷이 생성될 AWS Outposts 해당 서브넷의 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

타입: 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.네트워킹. RouteTable노드.

필수 항목 여부: 예

유형: String

예

```
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

타입: 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`

그리고 AWS. HookExecution노드. 이 후크는 `VNFDeployment` 노드가 배포되기 전에 실행됩니다.

필수 여부: 아니요

타입: 문자열

`post_create`

그리고 AWS. HookExecution노드. 이 후크는 `VNFDeployment` 노드 배포 후에 실행됩니다.

필수 여부: 아니요

타입: 문자열

예

```
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

타입: 문자열

`dns_support`

VPC에서 시작된 인스턴스가 DNS 호스트 이름을 가져오는지 나타냅니다.

필수 여부: 아니요

타입: 부울

기본값: `false`

`tags`

리소스에 연결할 태그입니다.

필수 여부: 아니요

유형: 목록

예

```
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 엘라스틱 IP 할당 ID를 제공하지 않으면 AWS TNB는 사용자 계정에 엘라스틱 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.네트워킹. InternetGateway노드 레퍼런스.

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

타입: 문자열

`tags`

리소스에 연결할 태그입니다.

필수 여부: 아니요

유형: 목록

예

```
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 경로를 NAT 게이트웨이에 대상 리소스로 연결하고 해당 경로를 연결된 라우팅 테이블에 추가하는 라우트 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

속성

`nat_gateway`

AWS.Networking.NATGateway 노드 참조입니다.

필수 항목 여부: 예

유형: String

`route_table`

AWS.네트워킹. RouteTable노드 레퍼런스.

필수 항목 여부: 예

유형: String

예

```
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
```

공통 노드

NSD 및 VNFD에서 사용할 노드를 정의합니다.

- AWS.HookDefinition.Bash

AWS.HookDefinition.Bash

bash에서 AWS HookDefinition을 정의합니다.

조건

```
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: 문자열

`environment_variables`

후크 bash 스크립트의 환경 변수입니다. **envName**=**envValue** 형식과 `^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+=[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*[a-zA-Z0-9]+$` 정규식을 사용하세요.

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.

예: `A123-45xYz=Example_789`

필수 항목 여부: 아니요

유형: 목록

`execution_role`

후크를 실행하는 역할입니다.

필수 항목 여부: 예

유형: 문자열

예

```
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 Telco 네트워크 빌더의 보안

클라우드 AWS 보안이 최우선 과제입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

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

- 클라우드 보안 - AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다 AWS 클라우드. AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. AWS Telco Network Builder에 적용되는 규정 준수 프로그램에 대해 자세히 알아보려면 규정 준수 프로그램별 범위 내 AWS 서비스 (규정 준수 프로그램별) 를 참조하십시오.
- 클라우드에서의 보안 - 귀하의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터의 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 항목에서는 보안 및 규정 준수 목표를 충족하도록 AWS TNB를 구성하는 방법을 보여줍니다. 또한 AWS TNB 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 알아봅니다.

내용

TNB의 데이터 보호 AWS

AWS 공동 책임 모델 공동 책임 모델 이 모델에 설명된 대로 AWS 는 모든 모델을 실행하는 글로벌 인프라를 보호하는 역할을 합니다 AWS 클라우드. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터

프라이버시에 대한 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 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center OR AWS Identity and Access Management (IAM) 을 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 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 리소스와 통신하세요. AWS TLS 1.2는 필수이며 TLS 1.3를 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해 액세스할 때 FIPS 140-2로 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용하십시오. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 FIPS(Federal Information Processing Standard) 140-2를 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔, API 또는 SDK를 AWS 서비스 사용하여 AWS TNB 또는 기타 작업을 수행하는 경우가 포함됩니다. AWS CLI AWS 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 보안 인증 정보를 URL에 포함해서는 안 됩니다.

태그 처리

AWS 계정을 폐쇄하면 AWS TNB는 데이터를 삭제 대상으로 표시하고 해당 데이터를 모든 사용에서 제거합니다. 90일 이내에 AWS 계정을 다시 활성화하면 AWS TNB가 데이터를 복원합니다. 120일 후 AWS TNB는 데이터를 영구적으로 삭제합니다. AWS 또한 TNB는 네트워크를 종료하고 함수 패키지와 네트워크 패키지를 삭제합니다.

저장 중 암호화

AWS TNB는 추가 구성 없이 서비스에 저장된 모든 데이터를 항상 암호화합니다. 이 암호화는 자동으로 실행됩니다. 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 가상 사설 클라우드 (VPC) 에 있습니다. 모든 내부 AWS TNB 트래픽은 AWS 네트워크 내에 머물며 인터넷을 통과하지 않습니다. 시뮬레이션 에이전트와 클라이언트 간의 연결은 인터넷을 통해 라우팅됩니다.

TNB의 ID 및 액세스 관리 AWS

AWS Identity and Access Management (IAM) 은 관리자가 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와줍니다. AWS IAM 관리자는 TNB 리소스를 사용할 AWS 수 있는 인증 (로그인) 및 권한 부여 (권한 보유) 를 받을 수 있는 사용자를 제어합니다. IAM은 추가 비용 AWS 서비스 없이 사용할 수 있습니다.

내용

고객

TNB에서 수행하는 작업에 따라 AWS Identity and Access Management (IAM) 사용 방식이 다릅니다. AWS

서비스 사용자 - AWS TNB 서비스를 사용하여 작업을 수행하는 경우 관리자가 필요한 자격 증명과 권한을 제공합니다. 더 많은 AWS TNB 기능을 사용하여 작업을 수행함에 따라 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. AWS TNB의 기능에 액세스할 수 없는 경우 AWS Telco Network Builder ID 및 액세스 문제 해결 섹션을 참조하세요.

서비스 관리자 — 회사에서 AWS TNB 리소스를 담당하는 경우 TNB에 AWS 대한 전체 액세스 권한이 있을 것입니다. 서비스 사용자가 액세스해야 하는 AWS TNB 기능과 리소스를 결정하는 것은 여러분의 몫입니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사에서 IAM을 AWS TNB와 함께 사용하는 방법에 대한 자세한 내용은 을 참조하십시오. AWS Telco 네트워크 빌더가 IAM과 함께 작동하는 방식

IAM 관리자 — IAM 관리자라면 TNB에 대한 액세스를 관리하기 위한 정책을 작성하는 방법에 대해 자세히 알고 싶을 것입니다. AWS IAM에서 사용할 수 있는 AWS TNB 자격 증명 기반 정책의 예를 보려면 을 참조하십시오. AWS Telco Network Builder(AWS TNB)에 대한 ID 기반 정책 예제

ID를 통한 인증

인증은 ID 자격 증명을 AWS 사용하여 로그인하는 방법입니다. IAM 사용자로 인증 (로그인 AWS) 하거나 IAM 역할을 맡아 인증 (로그인) 해야 합니다. AWS 계정 루트 사용자

ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 로그인할 수 있습니다. AWS AWS IAM Identity Center (IAM ID 센터) 사용자, 회사의 싱글 사인온 인증, Google 또는 Facebook 자격 증명이 페더레이션 ID의 예입니다. 연동 자격 증명으로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 액세스하는 경우 AWS 간접적으로 역할을 맡게 됩니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. 로그인에 대한 자세한 내용은 AWS 로그인 사용 설명서의 내 로그인 방법을 참조하십시오. AWS AWS 계정

AWS 프로그래밍 방식으로 액세스하는 경우 자격 증명을 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트 (SDK) 와 명령줄 인터페이스 (CLI) 를 AWS 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청 서명을 참조하십시오.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS 계정의 보안을 강화하기 위해 다단계 인증 (MFA) 을 사용할 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center 사용 설명서의 다중 인증 및 IAM 사용 설명서의 AWS에서 다중 인증(MFA) 사용을 참조하세요.

AWS 계정 루트 사용자

계정을 AWS 계정만들 때는 먼저 계정의 모든 AWS 서비스 리소스에 대한 완전한 액세스 권한을 가진 하나의 로그인 ID로 시작합니다. 이 ID를 AWS 계정 루트 사용자라고 하며, 계정을 만들 때 사용한 이메일 주소와 비밀번호로 로그인하여 액세스할 수 있습니다. 일상적인 태스크에 루트 사용자를 사용하지

않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 태스크를 수행하는 데 사용하세요. 루트 사용자로 로그인해야 하는 태스크의 전체 목록은 IAM 사용자 안내서의 루트 사용자 보안 인증이 필요한 태스크를 참조하세요.

연동 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 비롯한 수동 AWS 서비스 사용자가 ID 공급자와의 페더레이션을 사용하여 임시 자격 증명을 사용하여 액세스하도록 하는 것입니다.

페더레이션 ID는 기업 사용자 디렉토리, 웹 ID 공급자, Identity Center 디렉터리의 사용자 또는 ID 소스를 통해 제공된 자격 증명을 사용하여 액세스하는 AWS 서비스 모든 사용자를 말합니다. AWS Directory Service페더레이션 ID에 AWS 계정액세스하면 이들이 역할을 맡고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center(을)를 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 자체 ID 소스의 사용자 및 그룹 집합에 연결하고 동기화하여 모든 사용자 및 애플리케이션에서 사용할 수 있습니다. AWS 계정 IAM Identity Center에 대한 자세한 내용은AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇입니까?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 사용자 또는 애플리케이션에 대한 특정 권한을 AWS 계정 가진 사용자 내 자격 증명입니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명이 있는 IAM 사용자를 생성하는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 자격 증명이 필요한 특정 사용 사례가 있는 경우 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체를 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 자격 증명입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 정보는 IAM 사용 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님)를 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가진 사용자 AWS 계정 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 AWS Management Console 전환하여 에서 일시적으로 IAM 역

할을 맡을 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM 역할 사용을 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 연동 자격 증명에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 연동 자격 증명이 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 타사 자격 증명 공급자의 역할 만들기를 참조하세요. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 아이덴티티가 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관 짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 태스크에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스경우에는 역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 크로스 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.
- 서비스 간 액세스 — 일부는 다른 AWS 서비스서비스의 기능을 AWS 서비스 사용합니다. 예컨대, 어떤 서비스에서 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 순방향 액세스 세션 (FAS) — IAM 사용자 또는 역할을 사용하여 작업을 수행하는 경우 보안 AWS 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 AWS 서비스서비스에 AWS 서비스 요청하기 위한 요청과 결합하여 사용합니다. FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.
  - 서비스 연결 역할 — 서비스 연결 역할은 에 연결된 서비스 역할의 한 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은

사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

- Amazon EC2에서 실행되는 애플리케이션 — IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 API 요청을 AWS CLI 하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. AWS 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로필을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하세요.

정책을 사용한 액세스 관리

정책을 생성하고 이를 AWS ID 또는 리소스에 AWS 연결하여 액세스를 제어할 수 있습니다. 정책은 ID 또는 리소스와 연결될 때 AWS 해당 권한을 정의하는 객체입니다. AWS 주도자 (사용자, 루트 사용자 또는 역할 세션) 가 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 JSON 문서로 AWS 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

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.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole`태스크를 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 AWS Management Console, AWS CLI, 또는 AWS API에서 역할 정보를 가져올 수 있습니다.

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

자격 증명 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 내 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립형 정책입니다. AWS 계정관리형 정책에는 AWS 관리형 정책과 고객 관리형 정책이 포함됩니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 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 있는 지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

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

액세스 제어 목록(ACLs)

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

ACL을 지원하는 서비스의 예로는 아마존 S3와 아마존 VPC가 있습니다. AWS WAF ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 타입에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 보안 인증 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 엔터티의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔터티에 대한 권한 경계를 참조하세요.

- 서비스 제어 정책 (SCP) - SCP는 조직 또는 조직 단위 (OU) 에 대한 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations AWS Organizations 사업체가 소유한 여러 AWS 계정 개를 그룹화하고 중앙에서 관리하는 서비스입니다. 조직에서 모든 기능을 활성화할 경우 서비스 제어 정책 (SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 구성원 계정의 엔티티 (각 엔티티 포함) 에 대한 권한을 제한합니다. AWS 계정 루트 사용자조직 및 SCP에 대한 자세한 정보는 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할 자격 증명 기반 정책의 교차 및 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 타입

여러 정책 타입이 요청에 적용되는 경우 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련되어 있을 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하십시오.

AWS Telco 네트워크 빌더가 IAM과 함께 작동하는 방식

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

Telco 네트워크 빌더와 함께 사용할 수 있는 IAM 기능 AWS

IAM 특성	AWS TNB 지원
ID 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키	예
ACLs	아니요

IAM 특성	AWS TNB 지원
ABAC(정책의 태그)	예
임시 보안 인증	예
보안 주체 권한	예
서비스 역할	아니요
서비스 연결 역할	아니요

AWS TNB 및 기타 AWS 서비스가 대부분의 IAM 기능과 어떻게 작동하는지 자세히 알아보려면 IAM 사용 설명서의 IAM과 함께 작동하는AWS 서비스를 참조하십시오.

TNB에 대한 ID 기반 정책 AWS

ID 기반 정책 지원	예

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는 지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업와 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 자격 증명 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

TNB의 ID 기반 정책 예제 AWS

AWS TNB ID 기반 정책의 예를 보려면 을 참조하십시오. AWS Telco Network Builder(AWS TNB)에 대한 ID 기반 정책 예제

TNB 내의 리소스 기반 정책 AWS

리소스 기반 정책 지원	아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예는 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 있는 지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 등이 포함될 수 있습니다. AWS 서비스

계정 간 액세스를 활성화하려는 경우 전체 계정이나 다른 계정의 IAM 엔터티를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 보안 주체와 리소스가 다른 AWS 계정경우 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 개체 (사용자 또는 역할) 에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 개체에 자격 증명 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

TNB에 대한 AWS 정책 조치

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 태스크를 설명합니다. 정책 작업은 일반적으로 관련 AWS API 작업과 이름이 같습니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

AWS TNB 작업 목록을 보려면 서비스 권한 부여 참조의 AWS Telco Network Builder에서 정의한 작업을 참조하십시오.

AWS TNB의 정책 조치는 조치 앞에 다음 접두사를 사용합니다.

```
tnb
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

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

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

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

AWS TNB ID 기반 정책의 예를 보려면 을 참조하십시오. AWS Telco Network Builder(AWS TNB)에 대한 ID 기반 정책 예제

TNB를 위한 정책 리소스 AWS

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 있는지 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 개체를 지정합니다. 문장에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 타입을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

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

AWS TNB 리소스 유형 및 해당 ARN 목록을 보려면 서비스 권한 부여 참조의 AWS Telco Network Builder에서 정의한 리소스를 참조하십시오. 각 리소스의 ARN을 지정할 수 있는 작업에 대해 알아보려면 AWS Telco Network Builder에서 정의한 작업을 참조하십시오.

AWS TNB ID 기반 정책의 예를 보려면 을 참조하십시오. AWS Telco Network Builder(AWS TNB)에 대한 ID 기반 정책 예제

TNB의 정책 조건 키 AWS

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 있는지 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 옵션입니다. 같거나 작음과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition`요소를 지정하거나 단일 `Condition`요소에서 여러 키를 지정하는 경우 AWS 는 논리적 `AND`태스크를 사용하여 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우 는 논리적 `OR` 연산을 사용하여 조건을 AWS 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모든 AWS 글로벌 조건 키를 보려면 IAM 사용 AWS 설명서의 글로벌 조건 컨텍스트 키를 참조하십시오.

AWS TNB 조건 키 목록을 보려면 서비스 권한 부여 참조의 AWS Telco Network Builder의 조건 키를 참조하십시오. 조건 키를 사용할 수 있는 작업 및 리소스를 알아보려면 AWS Telco Network Builder에서 정의한 작업을 참조하십시오.

AWS TNB ID 기반 정책의 예를 보려면 을 참조하십시오. AWS Telco Network Builder(AWS TNB)에 대한 ID 기반 정책 예제

TNB의 ACL AWS

ACL 지원	아니요

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

ABAC (TNB 포함) AWS

ABAC 지원(정책의 태그)	예

ABAC(속성 기반 액세스 제어)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 개체 (사용자 또는 역할) 및 여러 AWS 리소스에 태그를 첨부할 수 있습니다. ABAC의 첫 번째 단계로 개체 및 리소스에 태그를 지정합니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그를 기반으로 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 타입에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 타입에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 정보는 IAM 사용 설명서의 ABAC란 무엇인가요?를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

TNB에서 임시 자격 증명 사용 AWS

임시 보안 인증 지원	예

임시 자격 증명을 사용하여 로그인하면 일부 자격 증명이 AWS 서비스 작동하지 않습니다. 임시 자격 증명을 사용하는 방법을 AWS 서비스 비롯한 추가 정보는 IAM 사용 설명서의 IAM과AWS 서비스 연동되는 내용을 참조하십시오.

사용자 이름과 암호를 제외한 다른 방법을 AWS Management Console 사용하여 로그인하면 임시 자격 증명을 사용하는 것입니다. 예를 들어 회사의 SSO (Single Sign-On) 링크를 AWS 사용하여 액세스하는 경우 이 프로세스에서 자동으로 임시 자격 증명을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 정보는 IAM 사용 설명서의 역할로 전환(콘솔)을 참조하세요.

또는 API를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다 AWS CLI . AWS 그런 다음 해당 임시 자격 증명을 사용하여 액세스할 수 AWS있습니다. AWS 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성할 것을 권장합니다. 자세한 정보는 IAM의 임시 보안 인증 섹션을 참조하세요.

TNB에 대한 서비스 간 보안 주체 권한 AWS

전달 액세스 세션(FAS) 지원	예

IAM 사용자 또는 역할을 사용하여 작업을 수행하는 AWS경우 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 전화를 거는 주체의 권한을 다운스트림 서비스에 AWS 서비스 요청하라는 요청과 결합하여 사용합니다. AWS 서비스 FAS 요청은 다른 서비스 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 서비스가 수신한 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. 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 수행하기 위해 수임하는 IAM role(IAM 역할)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

TNB의 서비스 연결 역할 AWS

서비스 연결 역할 지원	아니요

서비스 연결 역할은 에 연결된 서비스 역할 유형입니다. AWS 서비스서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은 사용자에게 AWS 계정 표시되며 해당 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

AWS Telco Network Builder(AWS TNB)에 대한 ID 기반 정책 예제

기본적으로 사용자 및 역할에는 AWS TNB 리소스를 만들거나 수정할 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

각 리소스 유형의 ARN 형식을 포함하여 AWS TNB에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 권한 부여 참조의 AWS Telco Network Builder의 작업, 리소스 및 조건 키를 참조하십시오.

내용

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

정책 모범 사례

ID 기반 정책은 누군가가 사용자 계정에서 AWS TNB 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부를 결정합니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. 자격 증명 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르십시오.

- AWS 관리형 정책으로 시작하여 최소 권한 권한으로 이동 — 사용자와 워크로드에 권한을 부여하려면 여러 일반적인 사용 사례에 권한을 부여하는 AWS 관리형 정책을 사용하세요. 해당 내용은 에서 사용할 수 있습니다. AWS 계정사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 더 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우 태스크를 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. 예를 AWS 서비스들어 특정 작업을 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 AWS CloudFormation있습니다. 자세한 정보는 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여

안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하세요.

- 멀티 팩터 인증 (MFA) 필요 - IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 AWS 계정 MFA를 활성화하십시오. API 작업을 직접 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 MFA 보호 API 액세스 구성을 참조하세요.

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

TNB 콘솔 사용 AWS

AWS Telco Network Builder 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한을 통해 내 AWS TNB 리소스에 대한 세부 정보를 나열하고 볼 수 있어야 합니다. AWS 계정최소 필수 권한보다 더 제한적인 자격 증명 기반 정책을 만들면 콘솔이 해당 정책에 연결된 엔터티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

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

서비스 역할 정책 예제

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

IAM 서비스 역할을 사용하면 AWS TNB가 사용자 대신 리소스를 호출하여 네트워크를 인스턴스화하고 관리할 수 있습니다. 서비스 역할을 지정하는 경우 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 위임하기 위한 역할 생성을 참조하십시오.

AWS TNB 서비스 역할

플랫폼 팀의 일원은 관리자로서 AWS TNB 서비스 역할을 생성하여 TNB에 AWS 제공할 수 있습니다. 이 역할을 통해 AWS TNB는 Amazon Elastic Kubernetes AWS CloudFormation Service와 같은 다른 서비스를 호출하고 네트워크에 필요한 인프라를 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 권한 범위를 축소할 때는 정책에서 범위가 해제된 리소스에 대한 액세스 거부 오류가 발생하여 AWS TNB가 실패할 수 있다는 점에 유의하십시오.

다음 코드는 AWS TNB 서비스 역할 정책을 보여줍니다.

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

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

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

```
                "eks:ListClusters",
                "eks:RegisterCluster",
                "eks:TagResource",
                "eks:DescribeAddonVersions",
                "events:DescribeRule",
                "iam:GetRole",
                "iam:ListAttachedRolePolicies",
                "iam:PassRole"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "TNBAccessComputePerms"
        },
        {
            "Action": [
                "codebuild:BatchDeleteBuilds",
                "codebuild:BatchGetBuilds",
                "codebuild:CreateProject",
                "codebuild:DeleteProject",
                "codebuild:ListBuildsForProject",
                "codebuild:StartBuild",
                "codebuild:StopBuild",
                "events:DeleteRule",
                "events:PutRule",
                "events:PutTargets",
                "events:RemoveTargets",
                "s3:CreateBucket",
                "s3:GetBucketAcl",
                "s3:GetObject",
                "eks:DescribeNodegroup",
                "eks:DeleteNodegroup",
                "eks:AssociateIdentityProviderConfig",
                "eks:CreateNodegroup",
                "eks:DeleteCluster",
                "eks:DeregisterCluster",
                "eks:UntagResource",
                "eks:DescribeCluster",
                "eks:ListNodegroups",
                "eks:CreateAddon",
                "eks:DeleteAddon",
                "eks:DescribeAddon",
                "eks:DescribeAddonVersions",
                "s3:PutObject",
                "cloudformation:CreateStack",
```

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

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

다음 코드는 AWS TNB 서비스 신뢰 정책을 보여줍니다.

```
{
  "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 예는 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 사용 설명서의 다음 섹션을 참조하십시오.AWS CloudFormation

- AWS::IAM::Role

- 스택 템플릿 선택

AWS Amazon EKS 노드 그룹을 위한 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 예는 Amazon EKS 노드 그룹 정책에 대한 AWS TNB 서비스 역할을 생성하는 AWS 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 사용 설명서의 다음 섹션을 참조하십시오.AWS CloudFormation

- AWS::IAM::Role
- 스택 템플릿 선택

AWS Multus의 TNB 서비스 역할

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

다음 예제는 Multus 정책에 대한 AWS TNB 서비스 역할을 생성하는 AWS 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 사용 설명서의 다음 섹션을 참조하십시오.AWS CloudFormation

- AWS::IAM::Role
- 스택 템플릿 선택

AWS 라이프 사이클 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 예제는 라이프 사이클 후크 AWS CloudFormation 정책에 대한 AWS TNB 서비스 역할을 생성하는 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 사용 설명서의 다음 섹션을 참조하십시오.AWS CloudFormation

- AWS::IAM::Role
- 스택 템플릿 선택

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

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

```
{
```

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

AWS Telco Network Builder ID 및 액세스 문제 해결

다음 정보를 사용하면 AWS TNB 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 해결하는 데 도움이 됩니다.

문제

- 저는 TNB에서 작업을 수행할 권한이 없습니다. AWS
- 저는 IAM을 수행할 권한이 없습니다. PassRole
- 제 외부 사람들이 제 AWS TNB 리소스에 액세스할 AWS 계정 수 있도록 허용하고 싶습니다.

저는 TNB에서 작업을 수행할 권한이 없습니다. AWS

작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 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 필요하면 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

저는 IAM을 수행할 권한이 없습니다. PassRole

작업을 수행할 권한이 없다는 오류가 발생하는 경우 AWS TNB에 iam:PassRole 역할을 넘길 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 만드는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 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 수 있는 권한을 가지고 있지 않습니다.

```
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 필요하면 관리자에게 문의하세요. AWS 관리자는 로그인 자격 증명을 제공한 사람입니다.

제 외부 사람들이 제 AWS TNB 리소스에 액세스할 AWS 계정 수 있도록 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 제

어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- AWS TNB가 이러한 기능을 지원하는지 알아보려면 을 참조하십시오. AWS Telco 네트워크 빌더가 IAM과 함께 작동하는 방식
- 소유한 리소스에 대한 액세스를 제공하는 방법을 알아보려면 IAM 사용 설명서에서 소유하고 AWS 계정 있는 다른 IAM 사용자에게 액세스 권한 제공을 참조하십시오. AWS 계정
- 제3자에게 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정AWS 계정 소유에 대한 액세스 제공을 참조하십시오.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

TNB에 대한 AWS 규정 준수 검증

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면AWS 서비스 규정 준수 프로그램의AWS 서비스 범위별, 규정 참조하여 관심 있는 규정 준수 프로그램을 선택하십시오. 일반 정보는 AWS 규정 준수 프로그램AWS 보증 프로그램 규정AWS 참조하십시오.

를 사용하여 AWS Artifact타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- 보안 및 규정 준수 퀵 스타트 가이드 - 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- Amazon Web Services의 HIPAA 보안 및 규정 준수를 위한 설계 — 이 백서에서는 기업이 HIPAA 적격 애플리케이션을 만드는 AWS 데 사용할 수 있는 방법을 설명합니다.

> **Note**
>
> 모든 AWS 서비스 사람이 HIPAA 자격을 갖춘 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조를 참조하십시오.

- AWS 규정 준수 리소스AWS — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.
- AWS 고객 규정 준수 가이드 — 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (미국 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (PCI), 국제 표준화기구 (ISO) 등) 에서 보안 제어에 대한 지침을 매핑합니다.
- AWS Config 개발자 안내서의 규칙을 통한 리소스 평가 — 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub— 이를 AWS 서비스 통해 내부 AWS보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하십시오.
- Amazon GuardDuty — 환경에 의심스럽고 악의적인 활동이 있는지 AWS 계정모니터링하여 워크로드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하여 PCI DSS와 같은 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 수 있습니다.
- AWS Audit Manager— 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

TNB의 AWS 레질리언스

AWS 글로벌 인프라는 가용 영역을 중심으로 AWS 리전 구축됩니다. AWS 리전 물리적으로 분리되고 격리된 여러 가용 영역을 제공합니다. 이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹으로 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.

가용 영역에 대한 AWS 리전 자세한 내용은 글로벌 인프라를 참조하십시오AWS .

AWS TNB는 선택한 AWS 지역의 가상 사설 클라우드 (VPC) 에 있는 EKS 클러스터에서 네트워크 서비스를 실행합니다.

TNB의 인프라 보안 AWS

관리형 서비스인 AWS Telco Network Builder는 AWS 글로벌 네트워크 보안의 보호를 받습니다. AWS 보안 서비스 및 인프라 AWS 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하십시오. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 Security Pillar AWS Well-Architected Framework의 인프라 보호를 참조하십시오.

AWS 게시된 API 호출을 사용하여 네트워크를 통해 AWS TNB에 액세스할 수 있습니다. 고객은 다음을 지원해야 합니다.

- 전송 계층 보안(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.

또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 인증을 생성하여 요청에 서명할 수 있습니다.

다음은 공동 책임의 몇 가지 예입니다.

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

> **Important**
>
> 사용자는 TNB를 통해 프로비저닝한 네트워크 서비스를 효과적으로 복구할 수 있는 재해 복구 계획을 구현할 책임이 있습니다. AWS

네트워크 연결 보안 모델

AWS TNB를 통해 프로비저닝하는 네트워크 서비스는 선택한 지역에 위치한 가상 사설 클라우드(VPC) 내의 컴퓨팅 인스턴스에서 실행됩니다. AWS VPC는 AWS 클라우드의 가상 네트워크로, 워크로드 또는 조직 엔티티별로 인프라를 분리합니다. VPC 내 컴퓨팅 인스턴스 간 통신은 AWS 네트워크 내에서 유지되며 인터넷을 통해 전달되지 않습니다. 일부 내부 서비스 통신은 인터넷을 통과하며 암호화됩니다. 동일한 지역에서 실행되는 모든 고객을 위해 AWS TNB를 통해 프로비저닝된 네트워크 서비스는 동일한 VPC를 공유합니다. AWS TNB를 통해 여러 고객을 위해 프로비저닝된 네트워크 서비스는 동일한 VPC 내에서 별도의 컴퓨팅 인스턴스를 사용합니다.

네트워크 서비스 클라이언트와 AWS TNB 네트워크 서비스 간의 통신은 인터넷을 통해 이루어집니다. AWS TNB는 이러한 연결을 관리하지 않습니다. 클라이언트 연결을 보호하는 것은 사용자의 책임입니다.

AWS Management Console, AWS Command Line Interface (AWS CLI) 및 AWS SDK를 통한 AWS TNB 연결은 암호화됩니다.

IMDS 버전

AWS TNB는 세션 지향 방식인 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 Amazon EC2 사용 설명서의 IMDSv2 사용을 참조하십시오.

AWS TNB 모니터링

모니터링은 AWS TNB 및 사용자의 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 있어서 중요한 부분입니다. AWS는 AWS TNB를 모니터링하고, 이상이 있을 때 이를 보고하고, 적절할 경우 자동 조치를 취할 수 있도록 AWS CloudTrail을 제공합니다.

CloudTrail을 사용하여 AWS API 호출에 대한 자세한 정보를 캡처할 수 있습니다. 이러한 호출을 로그 파일로 Amazon S3에 저장할 수 있습니다. 이러한 CloudTrail 로그를 사용하여 어떤 요청이 이루어졌는지, 어떤 소스 IP 주소에서 요청을 했는지, 누가 언제 요청했는지와 같은 정보를 확인할 수 있습니다.

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

AWS CloudTrail을 사용하여 AWS Telco Network Builder(AWS TNB) API 호출 로깅

AWS Telco Network Builder(AWS TNB)는 AWS TNB에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 AWS TNB에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 AWS TNB 콘솔로부터의 호출과 AWS TNB API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 AWS TNB 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 AWS TNB에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 AWS TNB 정보

CloudTrail은 계정 생성 시 AWS 계정에서 사용되도록 설정됩니다. AWS TNB에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

AWS TNB에 대한 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또는 CloudTrail 로그

에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 지역에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 AWS TNB 작업은 CloudTrail에서 로깅되고 AWS Telco Network Builder API 참조에 기록됩니다. 예를 들어 `CreateSolFunctionPackage`, `CreateSolNetworkInstance`, `CreateSolNetworkPackage` 작업을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 AWS Identity and Access Management(IAM) 사용자 자격 증명으로 했는지.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

AWS TNB 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음은 `CreateSolFunctionPackage` 작업을 보여주는 CloudTrail 로그 항목이 나타낸 예제입니다.

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

```
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111222333444:role/example",
                "accountId": "111222333444",
                "userName": "example"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2023-02-02T01:42:39Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-02-02T01:43:17Z",
    "eventSource": "tnb.amazonaws.com",
    "eventName": "CreateSolFunctionPackage",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "XXX.XXX.XXX.XXX",
    "userAgent": "userAgent",
    "requestParameters": null,
    "responseElements": {
        "vnfPkgArn": "arn:aws:tnb:us-east-1:111222333444:function-package/
fp-12345678abcEXAMPLE",
        "id": "fp-12345678abcEXAMPLE",
        "operationalState": "DISABLED",
        "usageState": "NOT_IN_USE",
        "onboardingState": "CREATED"
    },
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111222333444",
    "eventCategory": "Management"
}
```


AWS TNB 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 기능을 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.
InfrastructureCancellation	이름 변경 없음	네트워크 서비스를 취소합니다.
InfrastructureInstantation	이름 변경 없음	사용자를 대신하여 AWS 리소스를 제공합니다.
InfrastructureTermination	이름 변경 없음	TNB를 통해 AWS 호출된 AWS 리소스를 디프로비저닝합니다.
InventoryDeregistration	이름 변경 없음	TNB에서 AWS AWS 리소스 등록을 취소합니다.

2024년 3월 7일 이전에 시작된 배포의 작업 이름	2024년 3월 7일 이후에 시작된 배포의 작업 이름	[Task description]
KubernetesClusterConfiguration	ClusterConfiguration	쿠버네티스 클러스터를 구성하고 NSD에 정의된 대로 Amazon AuthMap EKS에 IAM 역할을 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.

AWS Telco Network Builder(AWS TNB)의 서비스 할당량

서비스 할당량(제한이라고도 함)은 AWS 계정의 최대 서비스 리소스 또는 작업 수입니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS 서비스 할당량을 참조하세요.

AWS TNB에 대한 서비스 할당량은 다음과 같습니다.

이름	기본값	조정 가능	설명
동시 진행 중인 네트워크 서비스 작업	지원되는 각 지역: 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 지역: 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 표에는 TNB의 설명서 릴리스가 설명되어 있습니다 AWS .

변경 사항	설명	날짜
기존 작업의 새 작업 및 새 작업 이름	새 작업을 사용할 수 있습니다. 명확성을 위해 2024년 3월 7일부터 일부 기존 작업의 이름이 변경되었습니다.	2024년 5월 7일
클러스터용 Kubernetes 버전	AWS TNB는 이제 Amazon EKS 클러스터를 생성할 수 있는 쿠버네티스 버전 1.29를 지원합니다.	2024년 4월 10일
네트워크 인터페이스 지원 `security_groups`	AWS.networking.eni 노드에 보안 그룹을 연결할 수 있습니다.	2024년 4월 2일
Amazon EBS 루트 볼륨 암호화 지원	Amazon EBS 루트 볼륨에 대해 Amazon EBS 암호화를 활성화할 수 있습니다. 활성화하려면 AWS.Compute.eks 또는 AWS.Compute.eksManagedNode 노드에 속성을 추가하십시오. SelfManagedNode	2024년 4월 2일
노드 지원 `labels`	AWS.Compute.eks 또는 ManagedNodeAWS.Compute.eks 노드의 노드 그룹에 노드 레이블을 연결할 수 있습니다. SelfManagedNode	2024년 3월 19일
네트워크 인터페이스 지원 `source_dest_check`	AWS.networking.eni 노드를 통해 네트워크 인터페이스 원본/대상 검사를 활성화할지 비활	2024년 1월 25일

	성화할지를 지정할 수 있습니다.	
사용자 지정 사용자 데이터가 있는 Amazon EC2 인스턴스 지원	.Compute를 통해 사용자 지정 사용자 데이터가 포함된 Amazon EC2 인스턴스를 시작할 수 있습니다. AWS UserData 노드.	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.eks 또는 AWS.Compute.eks SelfManagedNode 노드에 포함된 경우 AWS TNB는 가능한 경우 해당 속성 ManagedNode 또는 속성에서 ENI와 관련된 권한을 얻습니다. `multus_role` `node_role`	2023년 12월 18일
프라이빗 클러스터 지원	AWS TNB는 이제 프라이빗 클러스터를 지원합니다. 프라이빗 클러스터를 나타내려면 `access` 속성을 `PRIVATE`으로 설정합니다.	2023년 12월 11일
클러스터용 Kubernetes 버전	AWS TNB는 이제 Amazon EKS 클러스터를 생성할 수 있는 쿠버네티스 버전 1.28을 지원합니다.	2023년 12월 11일

AWS TNB는 배치 그룹을 지원합니다.	`AWS.Compute.EKSManagedNode` 및 `AWS.Compute.EKSSelfManagedNode` 노드 정의를 위한 배치 그룹을 추가했습니다.	2023년 12월 11일
AWS TNB는 IPv6에 대한 지원을 추가합니다.	AWS TNB는 이제 IPv6 인프라를 사용한 네트워크 인스턴스 생성을 지원합니다. .네트워킹.vpc, .네트워킹,AWS.서브넷, .네트워킹 노드를 확인하십시오.AWSAWS https://docs.aws.amazon.com/tnb/latest/ug/node-internet-gateway.htmlInternetGateway,. 네트워킹.AWS SecurityGroupIngressRule, AWS. 네트워킹. SecurityGroupEgressRule, IPv6 구성을 위한 AWS.Compute.EKS 또한 NAT64 구성을 위한 노드 AWS.Networking.NATGateway and AWS.Networking.Route를 추가했습니다. IPv6 권한에 대한 Amazon EKS 노드 그룹의 AWS AWS TNB 서비스 역할 및 TNB 서비스 역할을 업데이트했습니다. 서비스 역할 정책 예시를 참조하십시오.	2023년 11월 16일
TNB 서비스 역할 정책에 권한을 추가했습니다. AWS	Amazon S3의 AWS TNB 서비스 역할 정책에 권한을 추가하고 인프라 AWS CloudFormation 인스턴스화를 활성화했습니다.	2023년 10월 23일

AWS TNB는 더 많은 지역에서 출시되었습니다.	AWS TNB는 이제 아시아 태평양 (서울), 캐나다 (중부), 유럽 (스페인), 유럽 (스톡홀름), 남미 (상파울루) 지역에서 사용할 수 있습니다.	2023년 9월 27일
.Compute.eks용 태그 AWS SelfManagedNode	AWS TNB는 이제 노드 정의를 위한 태그를 지원합니다. `AWS.Compute.EKSSelfManagedNode`	2023년 8월 22일
AWS TNB는 IMDSv2를 활용하는 인스턴스를 지원합니다.	인스턴스를 시작할 때 IMDSv2를 사용해야 합니다.	2023년 8월 14일
에 대한 권한이 업데이트되었습니다. MultusRoleInlinePolicy	MultusRoleInlinePolicy 이제 `ec2:DeleteNetworkInterface` 권한이 포함됩니다.	2023년 8월 7일
클러스터용 Kubernetes 버전	AWS TNB는 이제 Amazon EKS 클러스터를 생성할 수 있는 쿠버네티스 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 있도록 지원합니다 AuthRole .	2023년 7월 19일
AWS TNB는 보안 그룹을 지원합니다.	AWS.Networking이 추가되었습니다. SecurityGroup, AWS.네트워킹. SecurityGroupEgressRule, 그리고 AWS.네트워킹. SecurityGroupIngressRuleNSD 템플릿으로.	2023년 7월 18일

클러스터용 Kubernetes 버전	AWS TNB는 쿠버네티스 버전 1.22~1.26을 지원하여 Amazon EKS 클러스터를 생성합니다. AWS TNB는 더 이상 쿠버네티스 버전 1.21을 지원하지 않습니다.	2023년 5월 11일
AWS.compute.eks SelfManagedNode	지역 내, AWS Local Zones 및 에서 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.