aws

사용자 가이드

Amazon EKS

Amazon EKS: 사용자 가이드

Table of Contents

Amazon EKS란 무엇입니까?

Amazon Elastic Kubernetes Service(Amazon EKS)는 Amazon Web Services(AWS)에 Kubernetes 컨트롤 플레인을 설치, 운영 및 유지 관리할 필요가 없는 관리형 서비스입니다. Kubernetes는 컨테이너화된 애플리케이션의 관리, 규모 조정 및 배포를 자동화하는 오픈 소스 시스템입니다.

Amazon EKS의 기능

Amazon EKS의 주요 기능은 다음과 같습니다.

보안 네트워킹 및 인증

Amazon EKS는 Kubernetes 워크로드를 AWS 네트워킹 및 보안 서비스와 통합합니다. 또한 AWS Identity and Access Management(IAM)와의 통합으로 Kubernetes 클러스터에 대한 인증을 제공합니다.

간편한 클러스터 규모 조정

Amazon EKS를 사용하면 워크로드 수요에 따라 Kubernetes 클러스터 규모를 쉽게 조정할 수 있습니다. Amazon EKS는 CPU 또는 사용자 지정 지표를 기반으로 수평 Pod 자동 규모 조정, 그리고 전체 워크로드 수요를 기반으로 클러스터 자동 규모 조정을 지원합니다.

관리형 Kubernetes 경험

`eksctl`, AWS Management Console, AWS Command Line Interface(AWS CLI), API, `kubectl` 및 Terraform을 사용하여 Kubernetes 클러스터를 변경할 수 있습니다.

높은 가용성

Amazon EKS는 여러 가용 영역의 컨트롤 플레인에 대한 고가용성을 제공합니다.

AWS 서비스와 통합

Amazon EKS는 다른 AWS 서비스와 통합되어, 컨테이너화된 애플리케이션을 배포하고 관리하기 위한 포괄적인 플랫폼을 제공합니다. 또한 다양한 관찰성 도구를 통해 Kubernetes 워크크로드 문제를 손쉽게 해결할 수 있습니다.

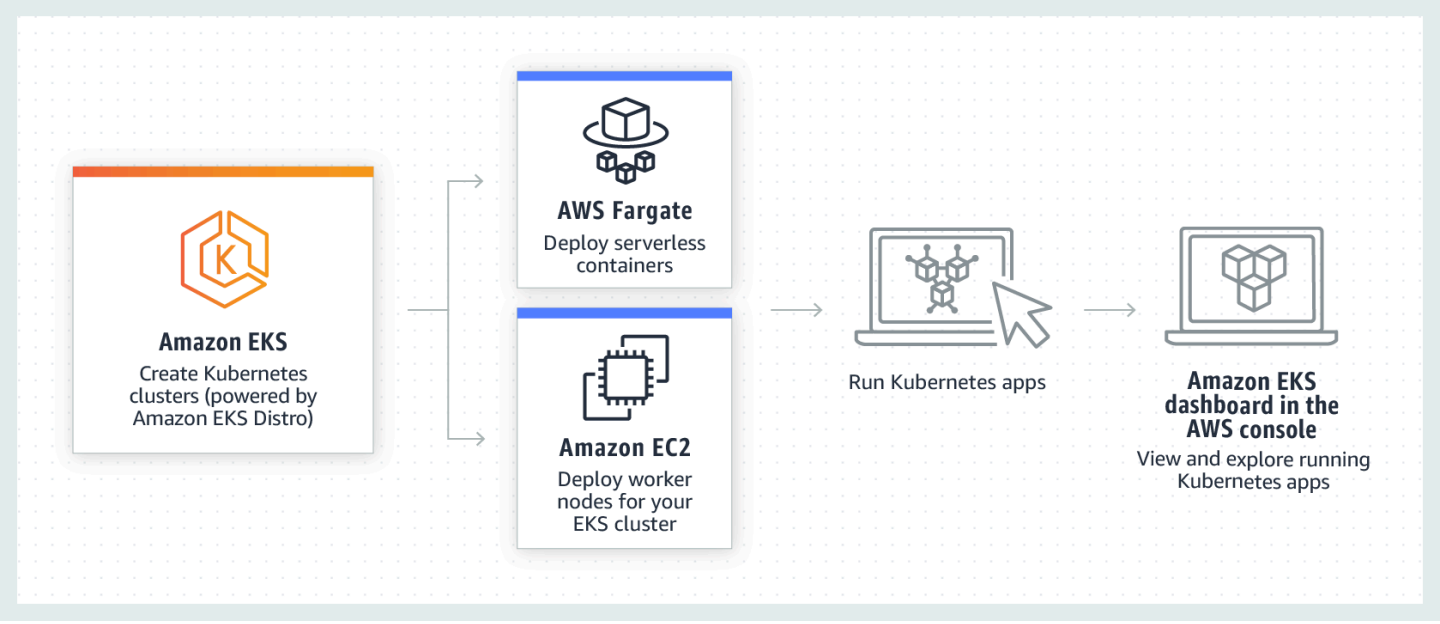
Amazon EKS의 다른 기능에 대한 자세한 내용은 Amazon EKS 기능을 참조하세요.

Amazon EKS 시작하기

첫 번째 클러스터와 연결된 리소스를 생성하려면 Amazon EKS 시작하기 부분을 참조하세요. 일반적으로 Amazon EKS를 시작하려면 다음 단계를 거쳐야 합니다.

1. 클러스터 생성 - 먼저 `eksctl`, AWS Management Console, AWS CLI, 또는 AWS SDK 중 하나를 사용하여 클러스터 생성을 시작합니다.
2. 컴퓨팅 리소스에 대한 접근 방식 선택 - AWS Fargate, Karpenter, 관리형 노드 그룹 및 자체 관리형 노드 중에서 결정합니다.
3. 설정 - 필요한 컨트롤러, 드라이버 및 서비스를 설정합니다.
4. 워크로드 배포 - 선택한 노드 유형의 리소스와 기능을 가장 잘 활용하도록 Kubernetes 워크로드를 조정합니다.
5. 관리 - 워크로드 감독하여 AWS 서비스를 통합하고 운영을 간소화하고 워크로드 성능을 향상시킵니다. AWS Management Console을 사용하여 워크로드에 대한 정보를 볼 수 있습니다.

다음 다이어그램은 클라우드에서 Amazon EKS를 실행하는 기본 흐름을 보여줍니다. 기타 Kubernetes 배포 옵션에 대한 자세한 내용은 배포 옵션 부분을 참조하세요.



Amazon EKS 요금

Amazon EKS 클러스터는 컨트롤 플레인과 Amazon Elastic Compute Cloud(Amazon EC2) 또는 Pods를 실행하는 Fargate 컴퓨팅으로 구성됩니다. 컨트롤 플레인의 요금에 대한 자세한 내용은 Amazon EKS 요금을 참조하세요. Amazon EC2와 Fargate 모두 다음을 제공합니다.

온디맨드 인스턴스

장기 약정이나 선결제 금액 없이 초 단위로 사용한 인스턴스에 대한 요금을 지불하는 방식입니다. 자세한 내용은 Amazon EC2 온디맨드 요금 및 AWS Fargate 요금을 참조하세요.

, 절감형 플랜

1년 또는 3년 기간 동안 시간당 USD로 일관된 사용량을 약정하여 비용을 절감할 수 있습니다. 자세한 내용은 Pricing with Savings Plans를 참조하세요.

Amazon EKS의 일반 사용 사례

Amazon EKS는 컨테이너화된 애플리케이션을 최적화하도록 설계된 AWS에 강력한 관리형 Kubernetes 서비스를 제공합니다. 다음은 Amazon EKS의 가장 일반적인 몇 가지 사용 사례로, 특정 요구 사항에 맞게 제품의 강점을 활용하는 데 도움이 됩니다.

고가용성 애플리케이션 배포

Elastic Load Balancing을 활용하여 애플리케이션이 여러 가용 영역에서 높은 가용성으로 제공되도록 할 수 있습니다.

마이크로서비스 아키텍처 구축

AWS Cloud Map 또는 Amazon VPC Lattice에 Kubernetes 서비스 검색 기능을 사용하여 복원력이 뛰어난 시스템을 구축합니다.

소프트웨어 릴리스 프로세스 자동화

애플리케이션의 자동화된 구축, 테스트, 배포 프로세스를 간소화하는 지속적인 통합 및 지속적인 배포(CICD) 파이프라인을 관리합니다.

서버리스 애플리케이션 실행

Amazon EKS에 AWS Fargate를 사용하여 서버리스 애플리케이션을 실행합니다. 즉, Amazon EKS와 Fargate가 기본 인프라를 처리하는 동안 애플리케이션 개발에만 집중할 수 있습니다.

기계 학습 워크로드 실행

Amazon EKS는 TensorFlow, MXNet 및 PyTorch 등과 같은 인기 기계 학습 프레임워크와 호환됩니다. GPU 지원을 통해 복잡한 기계 학습 작업도 효과적으로 처리할 수 있습니다.

온프레미스와 클라우드에서 일관적인 배포

Amazon EKS Anywhere를 사용하여 클라우드의 Amazon EKS와 동일한 도구를 통해 Kubernetes 클러스터를 자체 인프라에서 운영합니다.

비용 효율적인 배치 처리 및 빅 데이터 워크로드 실행

스팟 인스턴스를 활용하여 배치 처리와 Apache Hadoop 및 Spark와 같은 빅 데이터 워크로드를 저렴한 비용으로 실행합니다. 이를 통해 미사용 Amazon EC2 용량을 할인된 가격으로 활용할 수 있습니다.

애플리케이션 보안 및 규정 준수 보장

강력한 보안 관행을 구현하고 Amazon EKS 규정 준수를 유지하여 AWS Identity and Access Management(IAM), Amazon Virtual Private Cloud(Amazon VPC) 및 AWS Key Management Service(AWS KMS) 등 AWS 보안 서비스와 통합합니다. 이를 통해 업계 표준에 따른 데이터 프라이버시 및 보호가 보장됩니다.

Amazon EKS 아키텍처

Amazon EKS는 Kubernetes의 일반 클러스터 아키텍처와 일치합니다. 자세한 내용을 알아보려면 Kubernetes 설명서의 Kubernetes 컴포넌트를 참조하세요. 다음 섹션에는 Amazon EKS의 몇 가지 추가 아키텍처 세부 정보가 요약되어 있습니다.

컨트롤 플레인

Amazon EKS는 모든 클러스터가 고유한 Kubernetes 컨트롤 플레인을 보유하도록 보장합니다. 이 설계는 각 클러스터의 인프라를 분리하여 클러스터 또는 AWS 계정 간에 중복이 발생하지 않도록 합니다. 설정에는 다음이 포함됩니다.

분산 구성 요소

컨트롤 플레인은 AWS 리전 내 3개의 AWS 가용 영역 전반에 2개 이상의 API 서버 인스턴스와 3개의 `etcd` 인스턴스를 배치합니다.

최적의 성능

Amazon EKS는 컨트롤 플레인 인스턴스를 능동적으로 모니터링 및 조정하여 최고의 성능을 유지합니다.

복원성

컨트롤 플레인 인스턴스가 불안정해지면 Amazon EKS는 필요한 경우 다른 가용 영역을 사용하여 신속하게 이를 대체합니다.

일관적인 가동 시간

여러 가용 영역에서 클러스터를 실행함으로써 안정적인 API 서버 엔드포인트 가용성 서비스 수준에 관한 계약(SLA)을 달성합니다.

Amazon EKS는 Amazon Virtual Private Cloud(Amazon VPC)를 사용하여 단일 클러스터 내의 컨트롤 플레인 구성 요소 간 트래픽을 제한합니다. 클러스터 구성 요소는 Kubernetes 역할 기반 액세스 제어(RBAC) 정책에 따라 권한을 부여받지 않은 경우를 제외하면 다른 클러스터 또는 AWS 계정의 통신을 보거나 수신할 수 없습니다.

컴퓨팅

컨트롤 플레인 외에 Amazon EKS 클러스터에는 노드라고 하는 작업자 컴퓨터 세트가 있습니다. 특정 요구 사항을 충족하고 리소스 사용률을 최적화하려면 적절한 Amazon EKS 클러스터 노드 유형을 선택하는 것이 중요합니다. Amazon EKS는 다음 프라이머리 노드 유형을 제공합니다.

AWS Fargate

Fargate는 기본 인스턴스를 관리할 필요가 없는 컨테이너용 서버리스 컴퓨팅 엔진입니다. Fargate를 사용하면 애플리케이션의 리소스 요구 사항을 지정하고, AWS에서 인프라를 자동으로 프로비저닝, 규모 조정 및 유지 관리합니다. 이 옵션은 사용 편의성을 우선시하고 인프라 관리보다는 애플리케이션 개발 및 배포에 집중하고자 하는 사용자에게 적합합니다.

Karpenter

Karpenter는 유연한 고성능 Kubernetes 클러스터 자동 규모 조정기로 애플리케이션 가용성과 클러스터 효율성 개선에 도움이 됩니다. Karpenter는 변화하는 애플리케이션 로드에 대응하여 적절한 크기의 컴퓨팅 리소스를 시작합니다. 이 옵션은 워크로드의 요구 사항을 충족하는 적시 컴퓨팅 리소스를 프로비저닝할 수 있습니다.

관리형 노드 그룹

관리형 노드 그룹은 Amazon EKS 클러스터 내 Amazon EC2 인스턴스 컬렉션 관리에 대한 자동화와 사용자 지정을 결합합니다. AWS는 노드 패치, 업데이트 및 규모 조정과 같은 작업을 처리하여 운영상의 부담을 덜어줍니다. 동시에 사용자 지정 `kubelet` 인수가 지원되므로 고급 CPU 및 메모리 관리 정책을 사용할 수 있습니다. 또한 클러스터별 별도 권한의 필요성을 우회하면서 서비스 계정에 대한 AWS Identity and Access Management(IAM) 역할을 통해 보안을 강화합니다.

자체 관리형 노드

자체 관리형 노드를 통해 Amazon EKS 클러스터 내에서 Amazon EC2 인스턴스를 완벽하게 제어할 수 있습니다. 사용자는 노드 관리, 규모 조정 및 유지 관리를 담당하므로, 기본 인프라를 완전히 제어할 수 있습니다. 이 옵션은 노드를 세밀하게 제어하고 사용자 지정해야 하며 인프라 관리 및 유지 관리에 시간을 투자할 준비가 된 사용자에게 적합합니다.

Kubernetes 개념

Amazon Elastic Kubernetes Service(Amazon EKS)는 오픈 소스 Kubernetes 프로젝트를 기반으로 하는 AWS 관리형 서비스입니다. Amazon EKS 서비스가 AWS 클라우드와 통합되는 방식에 대해 알아야 할 사항이 있지만(특히 Amazon EKS 클러스터를 처음 생성할 때), 일단 가동되어 실행되면 다른 Kubernetes 클러스터와 거의 동일한 방식으로 Amazon EKS 클러스터를 사용하게 됩니다. 따라서 Kubernetes 클러스터를 관리하고 워크로드를 배포하려면 최소한 Kubernetes 개념에 대한 기본적인 이해가 필요합니다.

이 페이지에서는 Kubernetes 개념을 Kubernetes를 사용해야 하는 이유, 클러스터 및 워크로드의 세 섹션에 걸쳐 설명합니다. 첫 번째 섹션에서는 Kubernetes 서비스, 특히 Amazon EKS와 같은 관리형 서비스를 실행하는 것의 가치에 대해 설명합니다. 워크로드 섹션에서는 Kubernetes 애플리케이션의 구축, 저장, 실행 및 관리 방법에 대해 다룹니다. 클러스터 섹션에서는 Kubernetes 클러스터를 구성하는 다양한 구성 요소와 Kubernetes 클러스터를 생성하고 유지 관리하는 사용자의 책임에 대해 설명합니다.

주제

내용을 살펴보면서 여기에서 다루는 주제에 대해 더 자세히 알아보고 싶을 경우 링크를 통해 Amazon EKS 및 Kubernetes 설명서에 있는 Kubernetes 개념에 대한 추가 설명으로 이동할 수 있습니다. Amazon EKS가 Kubernetes 컨트롤 플레인 및 컴퓨팅 기능을 구현하는 방법에 대한 자세한 내용은 Amazon EKS 아키텍처를 참조하세요.

왜 Kubernetes를 사용해야 합니까?

Kubernetes는 미션 크리티컬한 프로덕션 품질의 컨테이너화된 애플리케이션을 실행할 때 가용성과 확장성을 개선하도록 설계되었습니다. Kubernetes는 이러한 목표를 달성하기 위해 단일 머신에서 Kubernetes를 실행하는 것이 아니라(물론 가능하지만), 수요에 따라 확장되거나 축소될 수 있는 컴퓨터 세트에서 애플리케이션을 실행합니다. Kubernetes에는 다음과 같은 작업을 쉽게 만드는 기능이 포함되어 있습니다.

- 여러 시스템에 애플리케이션 배포(포드에 배포된 컨테이너 사용)
- 컨테이너 상태 모니터링 및 장애가 발생한 컨테이너 다시 시작
- 부하에 따라 컨테이너 확장 및 축소
- 새 버전으로 컨테이너 업데이트
- 컨테이너 간 리소스 할당
- 시스템 간 트래픽 균형 조정

Kubernetes가 이러한 유형의 복잡한 작업을 자동화해주므로 애플리케이션 개발자는 인프라에 대한 걱정 없이 애플리케이션 워크로드를 구축하고 개선하는 데 집중할 수 있습니다. 개발자는 일반적으로 원하는 애플리케이션 상태를 설명하는 구성 파일을 YAML 파일 형식으로 생성합니다. 여기에는 실행할 컨테이너, 리소스 제한, 포드 복제본 수, CPU/메모리 할당, 선호도 규칙 등이 포함될 수 있습니다.

Kubernetes의 속성

Kubernetes에는 목표를 달성하기 위한 다음과 같은 특징이 있습니다.

- 컨테이너화 - Kubernetes는 컨테이너 오케스트레이션 도구입니다. Kubernetes를 사용하려면 먼저 애플리케이션을 컨테이너화해야 합니다. 애플리케이션 유형에 따라, 이러한 컨테이너화는 마이크로서비스 세트, 배치 작업 또는 다른 형태가 될 수 있습니다. 이제 애플리케이션에서 Kubernetes 워크플로우를 활용할 수 있으며, 이 워크플로우에는 컨테이너를 컨테이너 레지스트리의 이미지로 저장하고, Kubernetes 클러스터에 배포하고, 사용 가능한 노드에서 실행할 수 있는 거대한 도구 에코시스템이 포함됩니다. Kubernetes 클러스터에 컨테이너를 배포하기 전에 로컬 컴퓨터에서 Docker 또는 다른 컨테이너 런타임을 사용하여 개별 컨테이너를 빌드하고 테스트할 수 있습니다.
- 확장 가능 - 애플리케이션에 대한 수요가 해당 애플리케이션의 실행 중인 인스턴스 용량을 초과하는 경우 Kubernetes를 확장할 수 있습니다. 필요에 따라 Kubernetes는 애플리케이션에 더 많은 CPU 또

는 메모리가 필요한지 여부를 확인하고 사용 가능한 용량을 자동으로 확장하거나 기존 용량을 더 많이 사용하여 이에 대응할 수 있습니다. 애플리케이션의 더 많은 인스턴스를 실행하기에 충분한 컴퓨팅 용량을 사용할 수 있는 경우 포드 수준에서 확장을 수행할 수 있고(수평적 포드 자동 확장), 늘어난 용량을 처리하기 위해 더 많은 노드를 가져와야 하는 경우 노드 수준에서 확장을 수행할 수 있습니다(Cluster Autoscaler 또는 Karpenter). 용량이 더 이상 필요하지 않게 되면 이러한 서비스는 필요하지 않은 포드를 삭제하고 불필요한 노드를 종료할 수 있습니다.

- 사용 가능 - 애플리케이션 또는 노드가 비정상이거나 사용할 수 없게 되면 Kubernetes는 실행 중인 워크로드를 사용 가능한 다른 노드로 이동할 수 있습니다. 워크로드의 실행 중인 인스턴스 또는 워크로드를 실행 중인 노드를 삭제하기만 하면 문제를 강제로 해결할 수 있습니다. 여기서 중요한 것은 워크로드를 현재 위치에서 더 이상 실행할 수 없는 경우 다른 위치로 가져올 수 있다는 것입니다.
- 선언적 - Kubernetes는 활성 조정을 사용하여 클러스터에 대해 선언한 상태가 실제 상태와 일치하는지 지속적으로 확인합니다. 예를 들어 일반적으로 YAML 형식의 구성 파일을 통해 클러스터에 Kubernetes 객체를 적용하면 클러스터에서 실행할 워크로드를 시작하도록 요청할 수 있습니다. 나중에 구성을 변경하여 최신 버전의 컨테이너를 사용하거나 추가 메모리를 할당하는 등의 작업을 수행할 수 있습니다. Kubernetes는 원하는 상태를 설정하는 데 필요한 작업을 수행합니다. 이러한 작업에는 노드 가동 또는 중단, 워크로드 중지 및 재시작, 업데이트된 컨테이너 가져오기 등이 포함될 수 있습니다.
- 구성 가능 - 애플리케이션은 일반적으로 여러 구성 요소로 구성되므로 이러한 구성 요소 세트(대개 여러 컨테이너로 표시됨)를 함께 관리할 수 있어야 합니다. Docker Compose는 Docker를 사용하여 직접 이 작업을 수행하는 방법을 제공하지만 Kubernetes Kompose 명령을 사용하면 Kubernetes에서 이 작업을 수행할 수 있습니다. 이 작업을 수행하는 방법에 대한 예제는 Translate a Docker Compose File to Kubernetes Resources를 참조하세요.
- 확장 가능 - 독점 소프트웨어와 달리 오픈 소스 Kubernetes 프로젝트는 필요에 따라 원하는 방식으로 Kubernetes를 확장할 수 있도록 설계되었습니다. API 및 구성 파일은 직접 수정할 수 있습니다. 타사에서 자체 컨트롤러를 작성하여 인프라 및 최종 사용자 Kubernetes 기능을 모두 확장하는 것을 권장합니다. 웹후크를 사용하면 클러스터 규칙을 설정하여 정책을 적용하고 변화하는 조건에 맞게 조정할 수 있습니다. Kubernetes 클러스터를 확장하는 방법에 대한 자세한 내용은 Kubernetes 확장을 참조하세요.
- 이식 가능 - Kubernetes를 사용하면 모든 애플리케이션 요구 사항을 동일한 방식으로 관리할 수 있기 때문에 많은 조직에서 이를 기반으로 운영을 표준화하고 있습니다. 개발자는 동일한 파이프라인을 사용하여 컨테이너화된 애플리케이션을 빌드하고 저장할 수 있습니다. 그런 다음 이러한 애플리케이션을 온프레미스, 클라우드, 레스토랑의 POS 터미널 또는 회사 원격 사이트에 분산된 IOT 디바이스에서 실행되는 Kubernetes 클러스터에 배포할 수 있습니다. 오픈 소스라는 특성 덕분에 사용자가 이러한 특수 Kubernetes 배포판과 이를 관리하는 데 필요한 도구를 함께 개발할 수 있습니다.

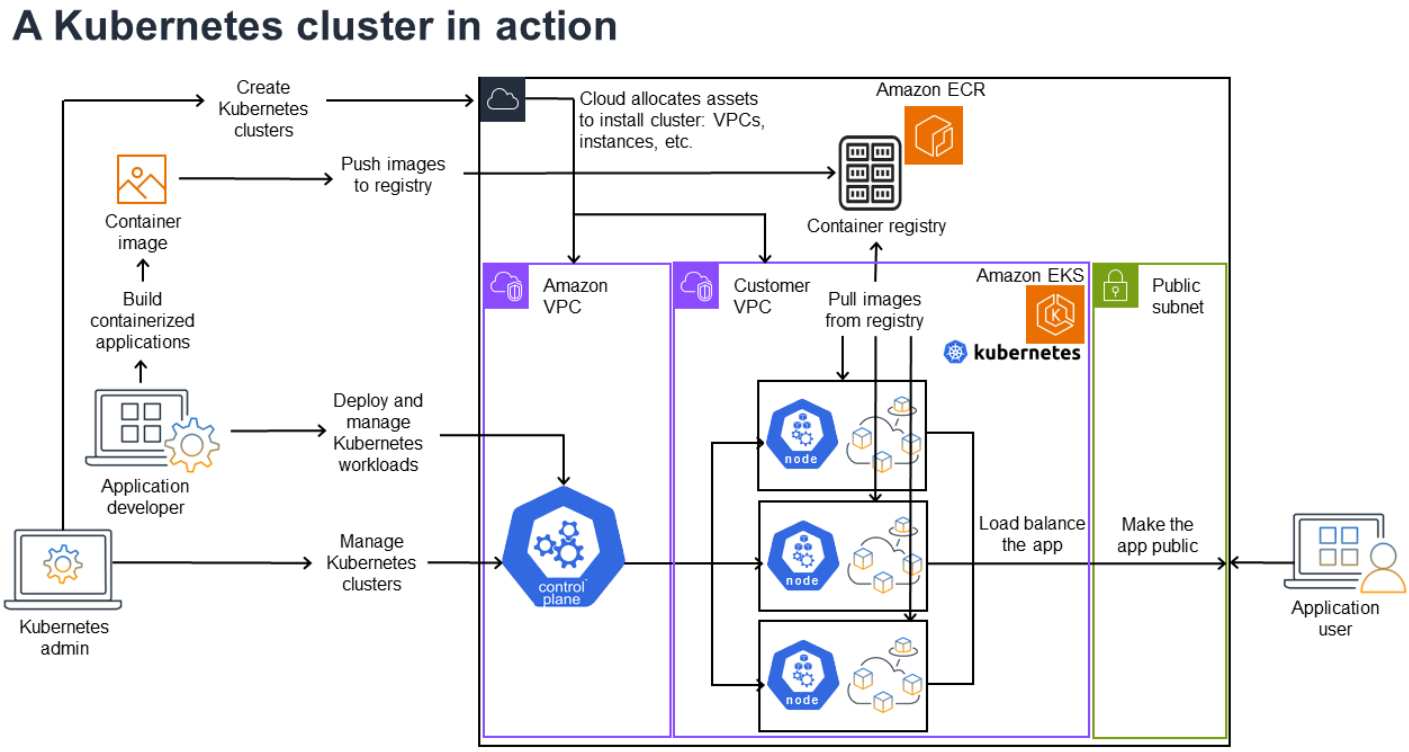
Kubernetes 관리

Kubernetes 소스 코드는 무료로 제공되므로 자체 장비만 있으면 직접 Kubernetes를 설치하고 관리할 수 있습니다. 그러나 Kubernetes를 자체 관리하려면 심층적인 운영 전문 지식이 필요하며 시간과 노력을 들여 유지 관리해야 합니다. 이러한 이유 때문에 프로덕션 워크로드를 배포하는 대부분의 사용자는 자체에서 테스트한 Kubernetes 배포판과 Kubernetes 전문가 지원을 제공하는 클라우드 제공업체(예: Amazon EKS)나 온프레미스 제공업체(예: Amazon EKS Anywhere)를 선택합니다. 이렇게 하면 클러스터를 유지 관리하는 데 필요한 다음과 같은 힘든 작업을 대부분 제거할 수 있습니다.

- 하드웨어 - 요구 사항에 따라 Kubernetes를 실행할 수 있는 하드웨어가 없는 경우 AWS Amazon EKS와 같은 클라우드 공급자를 통해 초기 비용을 절약할 수 있습니다. Amazon EKS를 사용하면 컴퓨터 인스턴스(Amazon Elastic Compute Cloud), 자체 프라이빗 환경(Amazon VPC), 중앙 집중식 자격 증명 및 권한 관리(IAM), 스토리지(Amazon EBS) 등을 비롯한 AWS에서 제공하는 최상의 클라우드 리소스를 사용할 수 있습니다. AWS가 컴퓨터, 네트워크, 데이터 센터 및 Kubernetes 실행에 필요한 기타 모든 물리적 구성 요소를 관리합니다. 마찬가지로, 수요가 가장 많은 기간 동안의 최대 용량을 처리하도록 데이터 센터를 계획할 필요도 없습니다. Amazon EKS Anywhere 또는 기타 온프레미스 Kubernetes 클러스터의 경우 Kubernetes 배포에 사용된 인프라를 관리할 책임은 사용자에게 있지만 여전히 AWS를 사용하여 Kubernetes를 최신 상태를 유지할 수 있습니다.
- 컨트롤 플레인 관리 - Amazon EKS가 컨테이너 예약, 애플리케이션 가용성 관리 및 기타 주요 작업을 담당하는 AWS 호스팅 Kubernetes 컨트롤 플레인의 보안 및 가용성을 관리하므로 사용자는 애플리케이션 워크로드에 집중할 수 있습니다. 사용자의 클러스터가 중단되는 경우 AWS에 클러스터를 실행 상태로 복원할 수 있는 수단이 있어야 합니다. Amazon EKS Anywhere의 경우 컨트롤 플레인을 직접 관리할 수 있습니다.
- 테스트된 업그레이드 - 클러스터를 업그레이드할 때 Amazon EKS 또는 Amazon EKS Anywhere를 사용하여 Kubernetes 배포의 테스트된 버전을 제공할 수 있습니다.
- 추가 기능 - Kubernetes를 확장하고 통합하도록 구축된 수백 가지의 프로젝트가 있으며, 이를 클러스터의 인프라에 추가하거나 워크로드 실행을 지원하는 데 사용할 수 있습니다. 이러한 추가 기능을 직접 구축하고 관리할 필요가 없도록, AWS는 클러스터와 함께 사용할 수 있는 Amazon EKS 추가 기능을 제공합니다. Amazon EKS Anywhere는 여러 인기 있는 오픈 소스 프로젝트의 빌드가 포함된 선별 패키지를 제공합니다. 따라서 소프트웨어를 직접 빌드하거나 주요 보안 패치, 버그 수정 또는 업그레이드를 관리할 필요가 없습니다. 마찬가지로 기본 구성이 요구 사항을 충족하는 경우 일반적으로 이러한 추가 기능을 거의 구성할 필요가 없습니다. 추가 기능으로 클러스터를 확장하는 방법에 대한 자세한 내용은 클러스터 확장을 참조하세요.

Kubernetes 작업

다음 다이어그램에서는 Kubernetes 관리자 또는 애플리케이션 개발자가 Kubernetes 클러스터를 생성하고 사용하기 위해 수행하는 주요 활동을 보여줍니다. 이 프로세스에서는 AWS 클라우드를 기반 클라우드 공급자의 예로 사용하여 Kubernetes 구성 요소가 서로 상호 작용하는 방식을 보여줍니다.



Kubernetes 관리자는 클러스터가 구축될 제공자의 유형에 맞는 도구를 사용하여 Kubernetes 클러스터를 생성합니다. 이 예에서는 Amazon EKS라는 관리형 Kubernetes 서비스를 제공하는 AWS 클라우드를 공급자로 사용합니다. 관리형 서비스는 클러스터를 위한 새로운 두 가상 프라이빗 클라우드(Amazon VPC)를 생성하고, 네트워킹을 설정하고, 클라우드의 자산을 관리할 클러스터에 Kubernetes 권한을 매핑하고, 컨트롤 플레인 서비스가 실행되도록 배치되었는지 확인하고, 0개 이상의 Amazon EC2 인스턴스를 워크로드 실행을 위한 Kubernetes 노드로 할당하는 등 클러스터를 생성하는 데 필요한 리소스를 자동으로 할당합니다. AWS는 컨트롤 플레인을 위한 Amazon VPC 하나를 관리하는 반면, 다른 Amazon VPC는 워크로드를 실행하는 고객 노드를 포함합니다.

앞으로 Kubernetes 관리자의 많은 작업은 kubectl과 같은 Kubernetes 도구를 사용하여 수행됩니다. 이 도구는 클러스터의 컨트롤 플레인에 직접 서비스를 요청합니다. 클러스터를 쿼리하고 변경하는 방식은 다른 Kubernetes 클러스터에서 수행하는 방식과 매우 비슷합니다.

이 클러스터에 워크로드를 배포하려는 애플리케이션 개발자는 여러 작업을 수행할 수 있습니다. 개발자는 애플리케이션을 하나 이상의 컨테이너 이미지로 빌드한 다음 Kubernetes 클러스터에서 액세스

할 수 있는 컨테이너 레지스트리로 해당 이미지를 푸시해야 합니다. AWS는 이 용도로 Amazon Elastic Container Registry(Amazon ECR)를 제공합니다.

애플리케이션을 실행하기 위해 개발자는 레지스트리에서 가져올 컨테이너와 해당 컨테이너를 포드에 래핑하는 방법을 포함하여 클러스터에 애플리케이션 실행 방법을 알려주는 YAML 형식의 구성 파일을 생성할 수 있습니다. 컨트롤 플레인(스케줄러)은 컨테이너를 하나 이상의 노드에 예약하고 각 노드의 컨테이너 런타임은 실제로 필요한 컨테이너를 가져와 실행합니다. 또한 개발자는 애플리케이션 로드 밸런서를 설정하여 각 노드에서 실행 중인 사용 가능한 컨테이너에 대한 트래픽의 균형을 조정하고 공용 네트워크를 통해 외부에서 사용할 수 있도록 애플리케이션을 노출할 수 있습니다. 이 모든 작업이 완료되면 애플리케이션을 사용하려는 사용자가 애플리케이션 엔드포인트에 연결하여 액세스할 수 있습니다.

다음 섹션에서는 Kubernetes 클러스터 및 워크로드의 관점에서 이러한 각 기능에 대해 자세히 살펴봅니다.

클러스터

작업이 Kubernetes 클러스터를 시작하고 관리하는 것이라면 Kubernetes 클러스터가 생성, 강화, 관리 및 삭제되는 방법을 알아야 합니다. 또한 클러스터를 구성하는 구성 요소가 무엇인지, 이러한 구성 요소를 유지 관리하기 위해 어떤 작업이 필요한지도 알아야 합니다.

클러스터를 관리하는 도구는 Kubernetes 서비스와 기반 하드웨어 공급자 간의 중복을 처리합니다. 이러한 이유 때문에 Kubernetes 공급자(예: Amazon EKS 또는 Amazon EKS Anywhere)는 공급자 전용 도구를 사용하여 이러한 작업을 자동화하는 경향이 있습니다. 예를 들어, Amazon EKS 클러스터를 시작하려면 `eksctl create cluster`를 사용할 수 있고 Amazon EKS Anywhere에서는 `eksctl anywhere create cluster`를 사용할 수 있습니다. 이러한 명령은 Kubernetes 클러스터를 생성하지만 공급자별로 다르며 Kubernetes 프로젝트 자체에 포함되지는 않습니다.

클러스터 생성 및 관리 도구

Kubernetes 프로젝트는 Kubernetes 클러스터를 수동으로 생성할 수 있는 도구를 제공합니다. 따라서 단일 시스템에 Kubernetes를 설치하거나 시스템에서 컨트롤 플레인을 실행하고 노드를 수동으로 추가하려는 경우 Kubernetes 도구 설치 아래에 나열된 kind, minikube, kubeadm 등과 같은 CLI 도구를 사용할 수 있습니다. 클러스터 생성 및 관리의 전체 수명 주기를 간소화하고 자동화하려면 Amazon EKS 또는 Amazon EKS Anywhere와 같이 확립된 Kubernetes 공급자가 지원하는 도구를 사용하는 것이 훨씬 쉽습니다.

AWS 클라우드에서는 eksctl과 같은 CLI 도구나 Terraform(Amazon EKS Blueprints for Terraform 참조) 같은 보다 선언적인 도구를 사용하여 Amazon EKS 클러스터를 생성할 수 있습니다. AWS 관리 콘

솔에서 클러스터를 생성할 수도 있습니다. Amazon EKS에서 제공되는 기능 목록은 Amazon EKS 기능을 참조하세요. Amazon EKS가 사용자를 대신하여 수행하는 Kubernetes 책임은 다음과 같습니다.

- 관리형 컨트롤 플레인 - AWS는 자동으로 컨트롤 플레인을 관리하고 AWS 가용 영역 전체에서 사용할 수 있도록 만들기 때문에 Amazon EKS 클러스터의 가용성과 확장성이 보장됩니다.
- 노드 관리 - 노드를 수동으로 추가하는 대신 Amazon EKS에서 관리형 노드 그룹 또는 Karpenter를 사용하여 필요에 따라 노드를 자동으로 생성하도록 할 수 있습니다. 관리형 노드 그룹은 Kubernetes 클러스터 오토 스케일링과 통합됩니다. 노드 관리 도구를 사용하면 워크로드를 배포하고 노드를 선택하는 방법을 설정하는 일정 예약 기능을 사용하여 스팟 인스턴스, 노드 통합, 가용성 등과 같은 작업에서 비용 절감 효과를 누릴 수 있습니다.
- 클러스터 네트워킹 - CloudFormation 템플릿을 사용하여 `eksctl`은 Kubernetes 클러스터의 컨트롤 플레인과 데이터 플레인(노드) 구성 요소 간의 네트워킹을 설정합니다. 또한 내부 및 외부 통신을 수행할 수 있는 엔드포인트를 설정합니다. 자세한 내용은 Amazon EKS 작업자 노드에 대한 클러스터 네트워킹 설명을 참조하세요. Amazon EKS의 포드 간 통신은 포드가 자격 증명 및 권한을 관리하는 AWS 클라우드 메서드를 활용할 수 있는 수단을 제공하는 Amazon EKS Pod Identity를 사용하여 이루어집니다.
- 추가 기능 - Amazon EKS를 사용하면 Kubernetes 클러스터를 지원하는 데 일반적으로 사용되는 소프트웨어 구성 요소를 빌드하고 추가할 필요가 없습니다. 예를 들어, AWS 관리 콘솔에서 Amazon EKS 클러스터를 생성하면 Amazon EKS kube-proxy, Kubernetes용 Amazon VPC CNI 플러그인 및 CoreDNS 추가 기능이 자동으로 추가됩니다. 사용 가능한 추가 기능의 목록을 포함하여 이러한 추가 기능에 대한 자세한 내용은 Amazon EKS 추가 기능을 참조하세요.

사용자가 자체 온프레미스 컴퓨터 및 네트워크에서 클러스터를 실행할 수 있도록 Amazon은 Amazon EKS Anywhere를 제공합니다. AWS 클라우드가 공급자가 되는 대신 자체 장비를 사용하여 VMWare vSphere, 베어 메탈(Tinkerbell provider), Snow, CloudStack 또는 Nutanix 플랫폼에서 Amazon EKS Anywhere를 실행할 수 있습니다.

Amazon EKS Anywhere는 Amazon EKS에서 사용하는 것과 동일한 Amazon EKS Distro 소프트웨어를 기반으로 합니다. 하지만 Amazon EKS Anywhere는 다양한 Kubernetes Cluster API(CAPI) 인터페이스 구현을 사용하여 Amazon EKS Anywhere 클러스터(예: vSphere의 경우 CAPV, CloudStack의 경우 CAPC)에 있는 시스템의 전체 수명 주기를 관리합니다. 전체 클러스터가 사용자 장비에서 실행되므로 컨트롤 플레인을 관리하고 해당 데이터를 백업해야 하는 추가적인 책임이 사용자에게 있습니다(이 문서 뒷부분의 etcd 참조).

클러스터 구성 요소

Kubernetes 클러스터 구성 요소는 컨트롤 플레인과 워커 노드라는 두 가지 주요 영역으로 구분됩니다. 컨트롤 플레인 구성 요소는 클러스터를 관리하고 해당 API에 대한 액세스를 제공합니다. 워커 노드(간

단히 노드라고도 함)는 실제 워크로드가 실행되는 위치를 제공합니다. 노드 구성 요소는 각 노드에서 실행되어 컨트롤 플레인과 통신하고 컨테이너를 실행하는 서비스로 구성됩니다. 클러스터의 워커 노드 세트를 데이터 플레인이라고 합니다.

컨트롤 플레인

컨트롤 플레인은 클러스터를 관리하는 서비스 세트로 구성됩니다. 이러한 서비스는 모두 단일 컴퓨터에서 실행되거나 여러 컴퓨터에 분산되어 있을 수 있습니다. 내부적으로는 이를 컨트롤 플레인 인스턴스(CPI)라고 합니다. CPI 실행 방법은 클러스터 규모와 고가용성 요구 사항에 따라 다릅니다. 클러스터의 수요가 증가하면 컨트롤 플레인 서비스가 확장되어 해당 서비스의 인스턴스를 더 많이 제공할 수 있으며, 이때 요청은 인스턴스 간에서 로드 밸런싱됩니다.

Kubernetes 컨트롤 플레인의 구성 요소가 수행하는 작업에는 다음이 포함됩니다.

- 클러스터 구성 요소(API 서버)와의 통신 - API 서버(kube-apiserver)는 Kubernetes API를 노출하므로 클러스터 내부와 외부 모두에서 클러스터에 요청을 보낼 수 있습니다. 즉, 클러스터의 객체(포드, 서비스, 노드 등)를 추가하거나 변경하는 요청은 포드를 실행하는 `kubectl`의 요청과 같은 외부 명령에서 발생할 수 있습니다. 마찬가지로, API 서버에서 클러스터 내 구성 요소로 요청을 보낼 수 있습니다(예: 포드 상태에 대한 `kubelet` 서비스의 쿼리).
- 클러스터에 대한 데이터 저장(etcd 키 값 저장소) - etcd 서비스는 클러스터의 현재 상태를 추적하는 중요한 역할을 합니다. etcd 서비스에 액세스할 수 없게 되면 얼마 동안 워크로드가 계속 실행되더라도 클러스터의 상태를 업데이트하거나 쿼리할 수 없게 됩니다. 이러한 이유로 중요 클러스터는 일반적으로 로드 밸런싱된 여러 개의 etcd 서비스 인스턴스를 동시에 실행하고 데이터 손실 또는 손상에 대비하여 etcd 키 값 저장소를 주기적으로 백업합니다. Amazon EKS에서는 기본적으로 이 모든 작업이 자동으로 처리된다는 점에 유의하십시오. Amazon EKS Anywhere는 etcd 백업 및 복원에 대한 지침을 제공합니다. etcd가 데이터를 관리하는 방법을 알아보려면 etcd 데이터 모델을 참조하세요.
- 포드를 노드에 예약(스케줄러) - Kubernetes에서 포드를 시작 또는 중지하는 요청은 Kubernetes 스케줄러(kube-scheduler)로 전달됩니다. 클러스터에는 포드를 실행할 수 있는 노드가 여러 개 있을 수 있으므로 포드를 실행할 노드(복제본의 경우 여러 노드)를 선택하는 것은 스케줄러입니다. 요청된 포드를 기존 노드에서 실행할 수 있는 가용 용량이 충분하지 않은 경우 다른 프로비저닝이 수행되지 않는 한 요청이 실패합니다. 이러한 프로비저닝에는 새 노드를 자동으로 시작하여 워크로드를 처리할 수 있는 관리형 노드 그룹이나 Karpenter와 같은 서비스를 활성화하는 것이 포함될 수 있습니다.
- 구성 요소를 원하는 상태로 유지(컨트롤러 관리자) - Kubernetes 컨트롤러 관리자는 데몬 프로세스(kube-controller-manager)로 실행되어 클러스터의 상태를 감시하고 클러스터를 변경하여 예상 상태를 재설정합니다. 특히 node-lifecycle-controller, statefulset-controller, endpoint-controller, cronjob-controller 등을 포함하는 다양한 Kubernetes 객체를 감시하는 여러 컨트롤러가 있습니다.
- 클라우드 리소스 관리(Cloud Controller Manager) - Kubernetes와 기본 데이터 센터 리소스에 대한 요청을 수행하는 클라우드 공급자 간의 상호 작용은 Cloud Controller Manager(cloud-controller-

manager)에서 처리합니다. Cloud Controller Manager에서 관리하는 컨트롤러에는 경로 컨트롤러(클라우드 네트워크 경로 설정에 사용), 서비스 컨트롤러(클라우드 로드 밸런싱 서비스에 사용), 노드 컨트롤러(클라우드 API를 사용하여 Kubernetes 노드를 클라우드 노드와 동기화하는 데 사용)가 포함될 수 있습니다.

워커 노드(데이터 플레인)

단일 노드 Kubernetes 클러스터의 경우 워크로드는 컨트롤 플레인과 동일한 시스템에서 실행됩니다. 하지만 좀 더 일반적인 구성은 Kubernetes 워크로드 실행 전용으로 사용되는 별도의 컴퓨터 시스템(노드)을 하나 이상 두는 것입니다.

Kubernetes 클러스터를 처음 생성할 때 일부 클러스터 생성 도구를 사용하면 기존 컴퓨터 시스템을 식별하거나 공급자가 새 컴퓨터 시스템을 생성하도록 하여 클러스터에 추가할 특정 수의 노드를 구성할 수 있습니다. 이러한 시스템에 워크로드를 추가하기 전에 각 노드에 서비스를 추가하여 다음 기능을 구현합니다.

- 각 노드(kubelet) 관리 - API 서버는 각 노드에서 실행되는 kubelet 서비스와 통신하여 노드가 제대로 등록되어 있고 스케줄러가 요청한 포드가 실행되고 있는지 확인합니다. kubelet은 포드 매니페스트를 읽고 로컬 시스템의 포드에 필요한 스토리지 볼륨 또는 기타 기능을 설정할 수 있습니다. 또한 로컬에서 실행 중인 컨테이너의 상태를 확인할 수 있습니다.
- 노드에서 컨테이너 실행(컨테이너 런타임) - 각 노드의 컨테이너 런타임은 노드에 할당된 각 포드에 요청된 컨테이너를 관리합니다. 즉, 적절한 레지스트리에서 컨테이너 이미지를 가져와 컨테이너를 실행하고 중지하며 컨테이너에 대한 쿼리에 응답할 수 있습니다. 기본 컨테이너 런타임은 containerd입니다. Kubernetes 1.24부터 컨테이너 런타임으로 사용할 수 있는 Docker(Dockershim)의 특수 통합이 Kubernetes에서 삭제되었습니다. 로컬 시스템에서 컨테이너를 테스트하고 실행하는 데 여전히 Docker를 사용할 수 있지만 Docker와 Kubernetes를 함께 사용하려면 이제 각 노드에 Docker 엔진을 설치하여 Kubernetes와 함께 사용해야 합니다.
- 컨테이너 간 네트워킹 관리(kube-proxy) - 서비스를 사용하여 포드 간 통신을 지원할 수 있게 하려면 Kubernetes에 포드 네트워크를 설정하여 해당 포드와 관련된 IP 주소 및 포트를 추적할 방법이 필요했습니다. kube-proxy 서비스는 모든 노드에서 실행되어 포드 간 통신이 이루어질 수 있도록 합니다.

확장 클러스터

클러스터를 지원하기 위해 Kubernetes에 추가할 수 있는 일부 서비스가 있지만 이러한 서비스는 컨트롤 플레인에서는 실행되지 않습니다. 이러한 서비스는 종종 kube-system 네임스페이스의 노드나 자체 네임스페이스에서 직접 실행됩니다(타사 서비스 제공업체에서 자주 실행됨). 일반적인 예로는 클러스

터에 DNS 서비스를 제공하는 CoreDNS 서비스를 들 수 있습니다. 클러스터의 kube-system에서 실행 중인 클러스터 서비스를 확인하는 방법에 대한 자세한 내용은 기본 제공 서비스 검색을 참조하세요.

클러스터에 추가하는 것을 고려할 수 있는 다양한 유형의 추가 기능이 있습니다. 클러스터를 정상 상태로 유지하기 위해 로깅, 감사, 지표 등의 작업을 수행할 수 있는 관찰성 기능을 추가할 수 있습니다. 이 정보를 사용하여 종종 동일한 관찰성 인터페이스를 통해 발생하는 문제를 해결할 수 있습니다. 예를 들어, 이러한 유형의 서비스로는 Amazon GuardDuty, CloudWatch, AWS Distro for OpenTelemetry, Amazon VPC CNI plugin for Kubernetes 및 Grafana Kubernetes Monitoring 등이 있습니다. 스토리지의 경우 Amazon EKS의 추가 기능에는 Amazon Elastic Block Store CSI 드라이버(블록 스토리지 디바이스 추가용), Amazon Elastic File System CSI 드라이버(파일 시스템 스토리지 추가용) 및 여러 타사 스토리지 추가 기능(예: Amazon FSx for NetApp ONTAP CSI 드라이버)이 포함됩니다.

Amazon EKS 추가 기능의 전체 목록은 Amazon EKS 추가 기능을 참조하세요.

워크로드

Kubernetes는 워크로드를 "Kubernetes에서 실행되는 애플리케이션"으로 정의합니다. 해당 애플리케이션은 포드에서 컨테이너로 실행되는 마이크로서비스 세트로 구성되거나 배치 작업 또는 다른 유형의 애플리케이션으로 실행될 수 있습니다. Kubernetes의 역할은 해당 객체의 설정 또는 배포에 대한 요청이 수행되도록 하는 것입니다. 애플리케이션을 배포하는 사용자는 컨테이너가 빌드되는 방식, 포드가 정의되는 방식 및 컨테이너를 배포하는 데 사용할 수 있는 방법에 대해 알아야 합니다.

컨테이너

Kubernetes에서 배포하고 관리하는 애플리케이션 워크로드의 가장 기본적인 요소는 포드입니다. 포드는 애플리케이션의 구성 요소를 유지하는 방법과 포드의 속성을 설명하는 사양을 정의하는 방법을 나타냅니다. 이것은 Linux 시스템용 소프트웨어를 함께 패키징하지만 그 자체가 하나의 엔터티로 실행되지는 않는 RPM 또는 Deb 패키지와 대조를 이룹니다.

포드는 배포 가능한 가장 작은 단위이기 때문에 보통 단일 컨테이너를 유지합니다. 하지만 컨테이너가 긴밀하게 연계된 경우 한 포드에 여러 컨테이너가 있을 수 있다. 예를 들어, 웹 서버 컨테이너는 로깅, 모니터링 또는 웹 서버 컨테이너와 밀접하게 연결된 기타 서비스를 제공할 수 있는 사이드카 유형의 컨테이너와 함께 포드에 패키징될 수 있습니다. 이 경우 동일한 포드에 있으면 실행 중인 각 포드 인스턴스에 대해 두 컨테이너가 항상 같은 노드에서 실행됩니다. 마찬가지로, 포드의 모든 컨테이너는 동일한 환경을 공유하며, 포드의 컨테이너는 동일한 격리된 호스트에 있는 것처럼 실행됩니다. 결과적으로, 컨테이너는 포드에 대한 액세스를 제공하는 단일 IP 주소를 공유하고 컨테이너는 자체 로컬 호스트에서 실행되는 것처럼 서로 통신할 수 있습니다.

포드 사양(PodSpec)은 원하는 포드 상태를 정의합니다. 워크로드 리소스를 사용하여 포드 템플릿을 관리하는 방식으로 개별 포드 또는 여러 포드를 배포할 수 있습니다. 워크로드 리소스에는

Deployments(여러 포드 복제본 관리), StatefulSets(데이터베이스 포드와 같이 고유해야 하는 포드를 배포) 및 DaemonSets(포드가 모든 노드에서 지속적으로 실행되어야 하는 경우)가 포함됩니다. 이에 대한 자세한 내용은 나중에 다루겠습니다.

포드는 배포하는 가장 작은 단위인 반면, 컨테이너는 빌드하고 관리하는 가장 작은 단위입니다.

컨테이너 빌드

포드는 실제로 하나 이상의 컨테이너를 포함하는 구조이며, 각 컨테이너 자체는 파일 시스템, 실행 파일, 구성 파일, 라이브러리 및 애플리케이션을 실제로 실행하기 위한 기타 구성 요소를 유지합니다. Docker Inc.라는 회사가 처음으로 컨테이너를 대중화했기 때문에 일부 사람들은 컨테이너를 Docker 컨테이너라고 부릅니다. 그러나 이후 Open Container Initiative가 업계를 위한 컨테이너 런타임, 이미지 및 배포 방법을 정의했습니다. 컨테이너가 기존의 많은 Linux 기능을 기반으로 생성되었다는 사실에 추가로, 컨테이너를 OCI 컨테이너, Linux 컨테이너 또는 그냥 컨테이너라고 부르는 경우가 많습니다.

컨테이너를 빌드할 때는 일반적으로 Dockerfile(문자 그대로 이름이 지정됨)로 시작합니다. 해당 Dockerfile 내에서 다음을 식별합니다.

- 기본 이미지 - 기본 컨테이너 이미지는 일반적으로 최소 버전의 운영 체제 파일 시스템(예: Red Hat Enterprise Linux 또는 Ubuntu) 또는 특정 유형의 애플리케이션을 실행하기 위한 소프트웨어를 제공하도록 개선된 최소 시스템(예: nodejs 또는 python 앱)에서 빌드되는 컨테이너입니다.
- 애플리케이션 소프트웨어 - Linux 시스템에 추가하는 것과 거의 같은 방식으로 애플리케이션 소프트웨어를 컨테이너에 추가할 수 있습니다. 예를 들어, Dockerfile에서 `npm` 및 `yarn`을 실행하여 Java 애플리케이션을 설치하거나 `yum` 및 `dnf`를 실행하여 RPM 패키지를 설치할 수 있습니다. 즉, Dockerfile에서 RUN 명령을 사용하면 기본 이미지의 파일 시스템에서 사용 가능한 모든 명령을 실행하여 결과 컨테이너 이미지 내에서 소프트웨어를 설치하거나 소프트웨어를 구성할 수 있습니다.
- 지침 - Dockerfile 참조는 구성 시 Dockerfile에 추가할 수 있는 지침을 기술합니다. 여기에는 컨테이너 자체의 내용을 빌드하는 데 사용되고(로컬 시스템의 파일 `ADD` 또는 `COPY`), 컨테이너가 실행될 때 실행할 명령을 식별하고(`CMD` 또는 `ENTRYPOINT`), 컨테이너를 실행되는 시스템에 연결하는 데 사용되는(실행할 `USER`, 마운트할 로컬 `VOLUME` 또는 `EXPOSE`할 포트) 지침이 포함됩니다.

`docker` 명령 및 서비스는 전통적으로 컨테이너를 빌드하는 데 사용되었지만(`docker build`), 컨테이너 이미지를 빌드하는 데 사용할 수 있는 다른 도구로는 podman 및 nerdctl이 있습니다. 컨테이너 빌드에 대해 자세히 알아보려면 Building Better Container Images 또는 Build with Docker를 참조하세요.

컨테이너 저장

컨테이너 이미지를 빌드한 후에는 워크스테이션의 컨테이너 배포 레지스트리 또는 퍼블릭 컨테이너 레지스트리에 이미지를 저장할 수 있습니다. 워크스테이션에 있는 프라이빗 컨테이너 레지스트리를 실행하면 컨테이너 이미지를 로컬에 저장하여 필요할 때 즉시 사용할 수 있습니다.

컨테이너 이미지를 보다 공개적인 방식으로 저장하려면 퍼블릭 컨테이너 레지스트리로 푸시할 수 있습니다. 퍼블릭 컨테이너 레지스트리는 컨테이너 이미지를 저장하고 배포할 수 있는 중앙 위치를 제공합니다. 퍼블릭 컨테이너 레지스트리의 예로는 Amazon Elastic Container Registry, Red Hat Quay 레지스트리 및 Docker Hub 레지스트리가 있습니다.

Amazon Elastic Kubernetes Service(Amazon EKS)에서 컨테이너화된 워크로드를 실행할 경우 Amazon Elastic Container Registry에 저장된 Docker 공식 이미지의 사본을 가져오는 것이 좋습니다. AWS Amazon ECR은 2021년부터 이러한 이미지를 저장하고 있습니다. Amazon ECR 퍼블릭 갤러리에서 인기 있는 컨테이너 이미지를 검색할 수 있으며, 특히 Docker Hub 이미지의 경우 Amazon ECR Docker 갤러리를 검색할 수 있습니다.

실행 중인 컨테이너

컨테이너는 표준 형식으로 빌드되므로 컨테이너 런타임을 실행할 수 있고(예: Docker) 콘텐츠가 로컬 시스템의 아키텍처와 일치하는(예: `x86_64` 또는 `arm`) 모든 시스템에서 컨테이너를 실행할 수 있습니다. 컨테이너를 테스트하거나 로컬 데스크톱에서 실행하려면 `docker run` 또는 `podman run` 명령을 사용하여 로컬 호스트에서 컨테이너를 시작할 수 있습니다. 하지만 Kubernetes의 경우 각 워커 노드에는 컨테이너 런타임이 배포되어 있으며 Kubernetes가 노드에서 컨테이너를 실행하도록 요청하기만 하면 됩니다.

컨테이너가 노드에서 실행되도록 할당되면 노드는 요청된 버전의 컨테이너 이미지가 노드에 이미 존재하는지 확인합니다. 존재하지 않는 경우 Kubernetes는 컨테이너 런타임에 적절한 컨테이너 레지스트리에서 해당 컨테이너를 가져온 다음 해당 컨테이너를 로컬에서 실행하도록 지시합니다. 컨테이너 이미지는 랩톱, 컨테이너 레지스트리 및 Kubernetes 노드 사이에서 이동하는 소프트웨어 패키지를 나타낸다는 것을 기억하십시오. 컨테이너는 해당 이미지의 실행 중인 인스턴스를 나타냅니다.

포드

컨테이너가 준비되면 포드를 사용하는 작업에 포드를 구성하고, 배포하고, 액세스 가능하게 만드는 작업이 포함됩니다.

포드 구성

포드를 정의할 때 포드에 속성 세트를 할당합니다. 이러한 속성에는 최소한 포드 이름과 실행할 컨테이너 이미지가 포함되어야 합니다. 하지만 포드 정의를 사용하여 구성할 수 있는 다른 사항도 많습니다(포드에 들어갈 수 있는 항목에 대한 자세한 내용은 PodSpec 페이지 참조). 다음이 포함됩니다.

- 스토리지 - 실행 중인 컨테이너를 중지하고 삭제하면 더 많은 영구 스토리지를 설정하지 않는 한 해당 컨테이너의 데이터 스토리지가 사라집니다. Kubernetes는 매우 다양한 스토리지 유형을 지원하며 볼륨이라는 범주로 추상화됩니다. 스토리지 유형에는 CephFS, NFS, iSCSI 등이 포함됩니다. 로컬 컴퓨터의 로컬 블록 디바이스를 사용할 수도 있습니다. 클러스터에서 이러한 스토리지 유형 중 하나를 사용할 경우 컨테이너 파일 시스템에서 선택한 탑재 지점에 스토리지 볼륨을 마운트할 수 있습니다. 영구 볼륨은 포드가 삭제된 후에도 계속 존재하는 볼륨이고, 임시 볼륨은 포드가 삭제될 때 삭제됩니다. 클러스터 관리자가 클러스터에 대해 서로 다른 StorageClasses를 생성한 경우, 사용 후 볼륨을 삭제하거나 회수할지 여부, 추가 공간이 필요할 경우 볼륨을 확장할지 여부, 특정 성능 요구 사항을 충족하는지 여부 등, 사용하는 스토리지의 속성을 선택할 수 있는 옵션이 제공됩니다.
- 보안 암호 - 포드 사양의 컨테이너에 보안 암호를 제공할 때 해당 컨테이너에 파일 시스템, 데이터베이스 또는 기타 보호된 자산에 액세스하는 데 필요한 권한을 제공할 수 있습니다. 키, 암호 및 토큰은 비밀로 저장할 수 있는 항목입니다. 보안 암호를 사용하면 이 정보를 컨테이너 이미지에 저장할 필요가 없어지고 실행 중인 컨테이너에 암호만 제공하면 됩니다. 보안 암호와 비슷한 것은 ConfigMap입니다. ConfigMap에는 서비스 구성을 위한 키-값 쌍과 같은 덜 중요한 정보가 들어 있는 경향이 있습니다.
- 컨테이너 리소스 - 컨테이너를 추가로 구성하기 위한 객체는 리소스 구성의 형태를 취할 수 있습니다. 각 컨테이너에 사용할 수 있는 메모리 및 CPU 양을 요청하고 컨테이너가 사용할 수 있는 리소스의 총량을 제한할 수 있습니다. 예제는 포드 및 컨테이너 리소스 관리를 참조하세요.
- 중단 - 포드는 모르는 사이에 중단되거나(노드가 다운됨) 자발적으로 중단될 수 있습니다(업그레이드가 필요함). 포드 중단 예산을 구성하면 중단이 발생했을 때 애플리케이션의 가용성을 어느 정도 제어할 수 있습니다. 예제는 애플리케이션에 대한 중단 예산 지정을 참조하세요.
- 네임스페이스 - Kubernetes는 Kubernetes 구성 요소와 워크로드를 서로 분리하는 다양한 방법을 제공합니다. 동일한 네임스페이스에서 특정 애플리케이션의 모든 포드를 실행하는 것은 해당 포드를 함께 보호하고 관리하는 일반적인 방법입니다. 사용할 네임스페이스를 직접 생성하거나 네임스페이스를 지정하지 않도록 선택할 수 있습니다(이 경우 Kubernetes가 `default` 네임스페이스 사용). Kubernetes 컨트롤 플레인 구성 요소는 일반적으로 kube-system 네임스페이스에서 실행됩니다.

앞서 설명한 구성은 일반적으로 Kubernetes 클러스터에 적용할 YAML 파일에 함께 수집됩니다. 개인용 Kubernetes 클러스터의 경우 이러한 YAML 파일을 로컬 시스템에 저장하기만 하면 됩니다. 그러나 더 중요한 클러스터와 워크로드의 경우 GitOps가 워크로드 및 Kubernetes 인프라 리소스 모두에 대한 스토리지 및 업데이트를 자동화하는 데 널리 사용되는 방법입니다.

포드 정보를 함께 수집하고 배포하는 데 사용되는 객체는 다음 배포 방법 중 하나로 정의됩니다.

포드 배포

포드를 배포하기 위해 선택하는 방법은 해당 포드로 실행하려는 애플리케이션의 유형에 따라 달라집니다. 다음은 몇 가지 옵션입니다.

- 상태 비저장 애플리케이션 - 상태 비저장 애플리케이션은 클라이언트의 세션 데이터를 저장하지 않으므로 다른 세션에서 이전 세션에서 발생한 일을 다시 참조할 필요가 없습니다. 따라서 포드가 비정상 상태가 된 경우 새 포드로 대체하거나 상태를 저장하지 않고 다른 포드로 이동하는 것이 더 쉬워집니다. 상태 비저장 애플리케이션(예: 웹 서버)을 실행 중인 경우 배포를 사용하여 포드와 ReplicaSet을 배포할 수 있습니다. ReplicaSet은 동시에 실행하려는 포드의 인스턴스 수를 정의합니다. ReplicaSet을 직접 실행할 수도 있지만 한 번에 실행해야 하는 포드의 복제본 수를 정의하기 위해 배포 내에서 직접 복제본을 실행하는 것이 일반적입니다.
- 상태 저장 애플리케이션 - 상태 저장 애플리케이션은 포드의 자격 증명과 포드가 시작되는 순서가 중요한 애플리케이션입니다. 이러한 애플리케이션에는 안정적인 영구 스토리지가 필요하며 일관된 방식으로 배포하고 확장해야 합니다. Kubernetes에 상태 저장 애플리케이션을 배포하려면 StatefulSet을 사용하면 됩니다. 일반적으로 StatefulSet으로 실행되는 애플리케이션의 예로는 데이터베이스가 있습니다. StatefulSet 내에서 복제본, 포드 및 해당 컨테이너, 마운트할 스토리지 볼륨, 컨테이너에서 데이터가 저장되는 위치를 정의할 수 있습니다. ReplicaSet으로 배포되는 데이터베이스의 예는 Run a Replicated Stateful Application을 참조하세요.
- 노드별 애플리케이션 - Kubernetes 클러스터의 모든 노드에서 애플리케이션을 실행하려는 경우가 있습니다. 예를 들어, 데이터 센터에서 모든 컴퓨터가 모니터링 애플리케이션 또는 특정 원격 액세스 서비스를 실행하도록 요구할 수 있습니다. Kubernetes의 경우 DaemonSet을 사용하여 선택한 애플리케이션이 클러스터의 모든 노드에서 실행되도록 할 수 있습니다.
- 애플리케이션 실행 완료 - 일부 애플리케이션은 특정 작업을 완료하기 위해 실행되어야 합니다. 여기에는 월별 상태 보고서를 실행하거나 오래된 데이터를 정리하는 애플리케이션이 포함될 수 있습니다. Job 객체를 사용하여 애플리케이션을 시작 및 실행한 다음 작업이 완료되면 종료하도록 설정할 수 있습니다. CronJob 객체를 사용하면 Linux crontab 형식으로 정의된 구조를 사용하여 특정 시간, 분, 날짜 또는 요일에 애플리케이션이 실행되도록 설정할 수 있습니다.

네트워크에서 애플리케이션에 액세스할 수 있게 만들기

애플리케이션이 여러 위치로 이동하는 마이크로서비스 세트로 배포되는 경우가 많기 때문에 Kubernetes에는 이러한 마이크로서비스가 서로를 찾을 수 있는 방법이 필요했습니다. 또한 다른 사용자가 Kubernetes 클러스터 외부에서 애플리케이션에 액세스하려면 Kubernetes에는 해당 애플리케이션을 외부 주소 및 포트에 노출할 방법이 필요했습니다. 이러한 네트워킹 관련 기능은 각각 서비스 객체와 수신 객체를 통해 수행됩니다.

- 서비스 - 포드는 서로 다른 노드와 주소로 이동할 수 있기 때문에 첫 번째 포드와 통신해야 하는 다른 포드가 첫 번째 포드의 위치를 찾기가 어려울 수 있습니다. 이 문제를 해결하기 위해 Kubernetes에서는 애플리케이션을 서비스로 표현할 수 있습니다. 서비스를 사용하면 특정 이름으로 포드 또는 포드 세트를 식별한 다음, 포드에서 해당 애플리케이션의 서비스를 노출하는 포트와 다른 애플리케이션이 해당 서비스에 연결하기 위해 사용할 수 있는 포트를 지정할 수 있다. 클러스터 내의 다른 포드는 단순히 이름을 사용하여 서비스를 요청할 수 있으며 Kubernetes는 해당 요청을 해당 서비스를 실행하는 포드의 인스턴스에 대한 적절한 포트로 전달합니다.
- 수신 - 수신은 Kubernetes 서비스로 표시되는 애플리케이션을 클러스터 외부의 클라이언트도 사용할 수 있게 해주는 역할을 합니다. 수신의 기본 기능에는 로드 밸런서(수신에서 관리), 수신 컨트롤러, 컨트롤러에서 서비스로 요청을 라우팅하는 규칙이 포함됩니다. Kubernetes에서 선택할 수 있는 수신 컨트롤러는 여러 개가 있습니다.

다음 단계

기본적인 Kubernetes 개념 및 Amazon EKS와의 관계를 이해하면 Amazon EKS 설명서와 Kubernetes 설명서에서 Amazon EKS 클러스터를 관리하고 해당 클러스터에 워크로드를 배포하는 데 필요한 정보를 쉽게 찾을 수 있습니다. Amazon EKS 사용을 시작하려면 다음 중에서 선택하십시오.

- 간단한 클러스터 만들기
- 더 복잡한 클러스터 만들기
- 샘플 애플리케이션 배포
- 클러스터 관리 방법 살펴보기

배포 옵션

다음 옵션을 사용하여 Amazon EKS를 배포할 수 있습니다.

클라우드의 Amazon EKS

Kubernetes 컨트롤 플레인 또는 노드를 설치, 운영 및 유지 관리할 필요 없이 AWS 클라우드에서 Kubernetes를 실행할 수 있습니다. 이 가이드에서는 이 옵션을 다룹니다.

Outposts의 Amazon EKS

AWS Outposts는 온프레미스 시설의 기본 AWS 서비스, 인프라 및 운영 모델을 지원합니다. Outposts의 Amazon EKS를 사용하면 확장 또는 로컬 클러스터를 실행하도록 선택할 수 있습니다. 확장 클러스터를 사용하면 Kubernetes 컨트롤 플레인이 AWS 리전에서 실행되고 노드가 Outposts에서 실행됩니다. 로컬 클러스터를 사용하면 Kubernetes 컨트롤 플레인과 노드를 모두 포함하여 전

체 Kubernetes 클러스터가 Outposts에서 로컬로 실행됩니다. 자세한 내용은 AWS Outposts에 대한 Amazon EKS 섹션을 참조하세요.

Amazon EKS Anywhere

Amazon EKS Anywhere는 온프레미스에서 Kubernetes 클러스터를 쉽게 생성하고 운영할 수 있도록 하는 Amazon EKS용 배포 옵션입니다. Amazon EKS 및 Amazon EKS Anywhere 모두 Amazon EKS Distro를 기반으로 구축됩니다. Amazon EKS Anywhere 및 Amazon EKS와의 차이점에 대해 자세히 알아보려면 Amazon EKS Anywhere 설명서의 개요 및 Amazon EKS Anywhere와 Amazon EKS의 비교를 참조하세요. 몇 가지 일반적인 질문에 대한 답변은 Amazon EKS Anywhere FAQ를 참조하세요.

Amazon EKS Distro

Amazon EKS Distro는 Amazon EKS에서 클라우드에 배포한 Kubernetes의 오픈소스 소프트웨어 및 종속성과 동일한 배포입니다. Amazon EKS Distro는 Amazon EKS와 동일한 Kubernetes 버전 릴리스 주기를 따르며 오픈 소스 프로젝트로 제공됩니다. 자세한 내용은 Amazon EKS Distro를 참조하세요. GitHub에서 Amazon EKS Distro의 소스 코드를 보고 다운로드할 수도 있습니다.

Kubernetes 클러스터에 어떤 배포 옵션을 사용할지 선택할 때 다음 사항을 고려하세요.

기능	Amazon EKS	Outposts의 Amazon EKS	Amazon EKS Anywhere	Amazon EKS Distro
Hardware(하드웨어)	AWS 제공	AWS 제공	사용자 제공	사용자 제공
배포 위치	AWS 클라우드	사용자 데이터 센터	사용자 데이터 센터	사용자 데이터 센터
Kubernetes 컨트롤 플레인 위치	AWS 클라우드	AWS 클라우드 또는 데이터 센터	사용자 데이터 센터	사용자 데이터 센터
Kubernetes 데이터 영역 위치	AWS 클라우드	사용자 데이터 센터	사용자 데이터 센터	사용자 데이터 센터
지원	AWS Support	AWS Support	AWS Support	OSS 커뮤니티 지원

Amazon EKS 사용 설정

일반적으로 AWS 리소스에는 리소스를 생성한 AWS 엔터티에 대한 액세스를 제한하는 액세스 제한이 있습니다. 따라서 처음부터 AWS Command Line Interface에서 적절한 사용자 구성을 설정하는 것이 중요합니다. 또한 Amazon EKS 클러스터의 효율적인 명령줄 관리를 위한 필수 도구를 로컬 시스템에 갖추어야 합니다. 이 주제는 클러스터의 명령줄 관리를 준비하는 데 도움이 됩니다.

1단계: AWS CLI 설정

AWS CLI는 Amazon EKS를 비롯한 AWS 서비스를 사용한 작업을 위한 명령줄 도구입니다. 로컬 시스템에서 Amazon EKS 클러스터 및 기타 AWS 리소스에 액세스할 수 있도록 IAM 사용자 또는 역할을 인증하는 데에도 사용됩니다. 명령줄에서 AWS의 리소스를 프로비저닝하려면 명령줄에서 사용할 AWS 액세스 키 ID와 비밀 키를 확보해야 합니다. 그런 다음, AWS CLI에서 이러한 보안 인증 정보를 구성해야 합니다. 아직 AWS CLI를 설치하지 않은 경우 AWS Command Line Interface 사용 설명서에서 최신 버전의 AWS CLI 설치 또는 업데이트를 참조하세요.

액세스 키 생성

1. AWS Management Console에 로그인합니다.
2. 오른쪽 상단에서 AWS 사용자 이름을 선택하여 탐색 메뉴를 엽니다. 예를 들어 **webadmin**를 선택합니다. 그런 다음, 보안 인증 정보를 선택합니다.
3. 액세스 키에서 액세스 키 생성을 선택합니다.
4. 명령줄 인터페이스(CLI)를 선택하고 다음을 선택합니다.
5. 액세스 키 생성을 선택합니다.
6. .csv 파일 다운로드를 선택합니다.

AWS CLI를 구성하려면

AWS CLI 다운로드 후 다음 단계에 따라 구성합니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI 구성을 참조하세요.

1. 터미널 창에 다음 명령을 입력합니다.

```
aws configure
```

선택적으로 명명된 프로파일(예: **--profile cluster-admin**)을 구성할 수 있습니다. AWS CLI에서 명명된 프로파일을 구성하는 경우 후속 명령에서 항상 이 플래그를 전달해야 합니다.

2. AWS 보안 인증을 입력합니다. 예:

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: region-code
Default output format [None]: json
```

보안 토큰을 받는 방법

필요한 경우 다음 명령을 실행하여 AWS CLI에 대한 새 보안 토큰을 가져옵니다. 자세한 내용은 AWS CLI 명령 레퍼런스의 get-session-token 섹션을 참조하십시오.

기본적으로 토큰은 15분 동안 유효합니다. 기본 세션 제한 시간을 변경하려면 **--duration-seconds** 플래그를 전달합니다. 예:

```
aws sts get-session-token --duration-seconds 3600
```

이 명령은 AWS CLI 세션의 임시 보안 인증 정보를 반환합니다. 다음과 같은 응답 출력이 표시되어야 합니다.

```
{
    "Credentials": {
        "AccessKeyId": "ASIA5FTRU3LOEXAMPLE",
        "SecretAccessKey": "JnKgvwfqUD9mNsPoi9IbxAYEXAMPLE",
        "SessionToken": "VERYLONGSESSIONTOKENSTRING",
        "Expiration": "2023-02-17T03:14:24+00:00"
    }
}
```

사용자 ID를 확인하는 방법

필요한 경우 다음 명령을 실행하여 터미널 세션에 대한 IAM 사용자 ID(예:*ClusterAdmin*)의 AWS 보안 인증 정보를 확인합니다.

```
aws sts get-caller-identity
```

이 명령은 AWS CLI에 대해 구성된 IAM 엔터티의 Amazon 리소스 이름(ARN)을 반환합니다. 다음 예제 응답 출력이 표시됩니다.

```
{
    "UserId": "AKIAIOSFODNN7EXAMPLE",
    "Account": "01234567890",
    "Arn": "arn:aws:iam::01234567890:user/ClusterAdmin"
}
```

2단계: Kubernetes 도구 설치

Kubernetes 클러스터와 통신하려면 Kubernetes API와 상호 작용하는 도구가 필요합니다. 또한 몇 가지 다른 도구(예: 로컬 컴퓨터의 Kubernetes 환경을 관리하는 도구)가 필요합니다.

AWS 리소스를 생성하려면

- Amazon EKS 클러스터 리소스 - AWS를 처음 사용하는 경우 `eksctl` 설치를 권장합니다. `eksctl`은 AWS CloudFormation을 사용하여 Amazon EKS 클러스터를 쉽게 생성하는 코드형 인프라(IaC) 유틸리티입니다. 또한 서비스 계정과 같은 추가 Kubernetes 리소스를 생성합니다. `eksctl` 설치에 대한 지침은 `eksctl` 설명서에서 Installation을 참조하세요.
- AWS 리소스 - AWS 인프라 프로비저닝 및 배포를 자동화하는 데 익숙하다면 Terraform 설치를 권장합니다. Terraform은 HashiCorp에서 개발한 오픈 소스 코드형 인프라(IaC) 도구입니다. HashiCorp Configuration Language(HCL) 또는 JSON과 같은 고급 구성 언어를 사용하여 인프라를 정의하고 프로비저닝할 수 있습니다. Terraform 설치에 대한 지침은 Terraform 설명서에서 Install Terraform을 참조하세요.

`kubectl`을(를) 설치하려면

`kubectl`은 Amazon EKS 클러스터의 Kubernetes API 서버와 통신하는 데 사용되는 오픈 소스 명령줄 도구입니다. 로컬 컴퓨터에 아직 설치하지 않은 경우 다음 옵션 중에서 선택합니다.

- AWS 버전 - Amazon EKS 지원 `kubectl` 버전을 설치하려면 `kubectl` 설치 또는 업데이트 섹션을 참조하세요.
- 커뮤니티 버전 - `kubectl`의 최신 커뮤니티 버전을 설치하려면 Kubernetes 설명서에서 Install tools 페이지를 참조하세요.

개발 환경을 설정하는 방법

- 로컬 배포 도구 - Kubernetes를 처음 사용하는 경우 `minikube` 또는 `kind` 같은 로컬 배포 도구를 설치해 보세요. 이러한 도구를 통해 로컬 시스템에서 Amazon EKS 클러스터를 관리할 수 있습니다.
- 패키지 관리자 - Helm은 복잡한 패키지의 설치 및 관리를 단순화하는 데 널리 사용되는 Kubernetes 용 패키지 관리자입니다. Helm을 사용하면 Amazon EKS 클러스터에서 AWS 로드 밸런서 컨트롤러와 같은 패키지를 더 쉽게 설치하고 관리할 수 있습니다.

다음 단계

- Amazon EKS 시작하기

`kubectl` 설치 또는 업데이트

`Kubectl`는 Kubernetes API 서버와 통신하기 위해 사용하는 명령줄 도구입니다. 이 `kubectl` 바이너리는 많은 운영 체제 패키지 관리자에서 사용할 수 있습니다. 설치에 패키지 관리자를 사용하는 것이 수동 다운로드 및 설치 프로세스보다 쉬운 경우가 많습니다.

이 주제는 장치에 `kubectl` 바이너리를 다운로드하고 설치하거나 업데이트하는 데 도움이 됩니다. 바이너리는 업스트림 커뮤니티 버전과 동일합니다. 바이너리가 Amazon EKS 또는 AWS에 고유하지 않은 경우.

> **Note**
>
> Amazon EKS 클러스터 제어 영역과 마이너 버전이 하나 다른 `kubectl` 버전을 사용해야 합니다. 예를 들어 `1.28` `kubectl` 클라이언트는 Kubernetes `1.27`, `1.28`, `1.29` 클러스터와 함께 작동합니다.

`kubectl`를 설치하거나 업데이트하려면 다음을 수행합니다.

1. 디바이스에 이미 `kubectl`이 설치되어 있는지 확인합니다.

```
kubectl version --client
```

디바이스의 경로에 kubectl이(가) 설치되어 있는 경우 출력 예제에는 다음과 유사한 정보가 포함돼 있습니다. 현재 설치한 버전을 최신 버전으로 업데이트하려면 다음 단계를 완료하고 현재 버전이 있는 동일한 위치에 새 버전을 설치해야 합니다.

```
Client Version: v1.29.X-eks-1234567
```

출력이 되지 않는다면 kubectl이 설치되지 않았거나 디바이스의 경로에 있는 위치에 설치되지 않은 것입니다.

2. macOS, Linux, Windows 운영 체제에 kubectl을 설치하거나 업데이트하세요.

macOS

macOS에 kubectl을 설치하거나 업데이트하려면 다음을 수행합니다.

1. Amazon S3에서 클러스터의 Kubernetes 버전에 대한 바이너리를 다운로드합니다.
   - Kubernetes 1.29

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.0/2024-01-04/bin/darwin/amd64/kubectl
     ```

   - Kubernetes 1.28

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.5/2024-01-04/bin/darwin/amd64/kubectl
     ```

   - Kubernetes 1.27

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.9/2024-01-04/bin/darwin/amd64/kubectl
     ```

   - Kubernetes 1.26

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.12/2024-01-04/bin/darwin/amd64/kubectl
     ```

   - Kubernetes 1.25

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-01-04/bin/darwin/amd64/kubectl
     ```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-01-04/bin/darwin/amd64/kubectl
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-01-04/bin/darwin/amd64/kubectl
```

2. (선택 사항) 해당 바이너리의 SHA-256 체크섬을 사용하여 다운로드한 바이너리를 확인합니다.

   a. 클러스터의 Kubernetes 버전에 대한 SHA-256 체크섬을 다운로드합니다.

   - Kubernetes 1.29

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.0/2024-01-04/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.28

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.5/2024-01-04/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.27

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.9/2024-01-04/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.26

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.12/2024-01-04/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.25

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-01-04/bin/darwin/amd64/kubectl.sha256
   ```

   - Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-01-04/
bin/darwin/amd64/kubectl.sha256
```

- Kubernetes `1.23`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-01-04/
bin/darwin/amd64/kubectl.sha256
```

b. 다운로드한 바이너리에 대해 SHA-256 체크섬을 확인합니다.

```
openssl sha1 -sha256 kubectl
```

c. 출력에 생성된 체크섬이 다운로드한 `kubectl.sha256` 파일의 체크섬과 일치하는지 확인합니다.

3. 바이너리에 실행 권한을 적용합니다.

```
chmod +x ./kubectl
```

4. 바이너리를 PATH의 폴더에 복사합니다. `kubectl` 버전이 이미 설치된 경우 `$HOME/bin/kubectl`을 생성하고 `$HOME/bin`이 `$PATH`로 시작하도록 해야 합니다.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:
$PATH
```

5. (선택 사항) 셸 초기화 파일에 `$HOME/bin` 경로를 추가하면 셸을 열 때 구성됩니다.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bash_profile
```

Linux (amd64)

Linux(**amd64**)에 **kubectl**을(를) 설치하거나 업데이트하는 방법

1. Amazon S3에서 클러스터의 Kubernetes 버전에 대한 `kubectl` 바이너리를 다운로드합니다.

- Kubernetes `1.29`

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.0/2024-01-04/bin/
linux/amd64/kubectl
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.5/2024-01-04/bin/linux/amd64/kubectl
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.9/2024-01-04/bin/linux/amd64/kubectl
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.12/2024-01-04/bin/linux/amd64/kubectl
```

- Kubernetes 1.25

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-01-04/bin/linux/amd64/kubectl
```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-01-04/bin/linux/amd64/kubectl
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-01-04/bin/linux/amd64/kubectl
```

2. (선택 사항) 해당 바이너리의 SHA-256 체크섬을 사용하여 다운로드한 바이너리를 확인합니다.

   a. 디바이스의 하드웨어 플랫폼용 명령을 사용하여 Amazon S3에서 클러스터의 Kubernetes 버전에 대한 SHA-256 체크섬을 다운로드합니다. 각 버전의 첫 번째 링크는 amd64, 두 번째 링크는 arm64을 위한 것입니다.

   - Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.0/2024-01-04/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.5/2024-01-04/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.9/2024-01-04/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.12/2024-01-04/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.25

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-01-04/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-01-04/bin/linux/amd64/kubectl.sha256
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-01-04/bin/linux/amd64/kubectl.sha256
```

b. 다음 명령 중 하나를 사용하여 다운로드한 바이너리의 SHA-256 체크섬을 확인합니다.

- ```
  sha256sum -c kubectl.sha256
  ```

  이 명령을 사용할 때는 다음 출력이 표시되는지 확인하세요.

  ```
  kubectl: OK
  ```

- ```
  openssl sha1 -sha256 kubectl
  ```

이 명령을 사용할 때 출력에 생성된 체크섬이 다운로드한 kubectl.sha256 파일의 체크섬과 일치하는지 확인합니다.

3. 바이너리에 실행 권한을 적용합니다.

```
chmod +x ./kubectl
```

4. 바이너리를 PATH의 폴더에 복사합니다. kubectl 버전이 이미 설치된 경우 $HOME/bin/kubectl을 생성하고 $HOME/bin이 $PATH로 시작하도록 해야 합니다.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (선택 사항) 셸 초기화 파일에 $HOME/bin 경로를 추가하면 셸을 열 때 구성됩니다.

> **Note**
>
> 이 단계에는 Bash 셸을 사용한다고 가정합니다. 다른 셸을 사용하는 경우, 특정 셸 초기화 파일을 사용하도록 명령을 변경하십시오.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Linux(arm64)

Linux(**arm64**)에 **kubectl**을(를) 설치하거나 업데이트하는 방법

1. Amazon S3에서 클러스터의 Kubernetes 버전에 대한 kubectl 바이너리를 다운로드합니다.
   - Kubernetes 1.29

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.0/2024-01-04/bin/linux/arm64/kubectl
```

   - Kubernetes 1.28

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.5/2024-01-04/bin/linux/arm64/kubectl
```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.9/2024-01-04/bin/linux/arm64/kubectl
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.12/2024-01-04/bin/linux/arm64/kubectl
```

- Kubernetes 1.25

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-01-04/bin/linux/arm64/kubectl
```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-01-04/bin/linux/arm64/kubectl
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-01-04/bin/linux/arm64/kubectl
```

2. (선택 사항) 해당 바이너리의 SHA-256 체크섬을 사용하여 다운로드한 바이너리를 확인합니다.

   a. 디바이스의 하드웨어 플랫폼용 명령을 사용하여 Amazon S3에서 클러스터의 Kubernetes 버전에 대한 SHA-256 체크섬을 다운로드합니다. 각 버전의 첫 번째 링크는 amd64, 두 번째 링크는 arm64을 위한 것입니다.

   - Kubernetes 1.29

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.0/2024-01-04/bin/linux/arm64/kubectl.sha256
     ```

   - Kubernetes 1.28

     ```
     curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.5/2024-01-04/bin/linux/arm64/kubectl.sha256
     ```

- Kubernetes 1.27

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.9/2024-01-04/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.26

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.12/2024-01-04/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.25

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.25.16/2024-01-04/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.24

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.24.17/2024-01-04/bin/linux/arm64/kubectl.sha256
```

- Kubernetes 1.23

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.23.17/2024-01-04/bin/linux/arm64/kubectl.sha256
```

b. 다음 명령 중 하나를 사용하여 다운로드한 바이너리의 SHA-256 체크섬을 확인합니다.

- ```
  sha256sum -c kubectl.sha256
  ```

  이 명령을 사용할 때는 다음 출력이 표시되는지 확인하세요.

  ```
  kubectl: OK
  ```

- ```
  openssl sha1 -sha256 kubectl
  ```

  이 명령을 사용할 때 출력에 생성된 체크섬이 다운로드한 `kubectl.sha256` 파일의 체크섬과 일치하는지 확인합니다.

3. 바이너리에 실행 권한을 적용합니다.

```
chmod +x ./kubectl
```

4. 바이너리를 PATH의 폴더에 복사합니다. kubectl 버전이 이미 설치된 경우 $HOME/bin/kubectl을 생성하고 $HOME/bin이 $PATH로 시작하도록 해야 합니다.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
```

5. (선택 사항) 셸 초기화 파일에 $HOME/bin 경로를 추가하면 셸을 열 때 구성됩니다.

> **Note**
>
> 이 단계에는 Bash 셸을 사용한다고 가정합니다. 다른 셸을 사용하는 경우, 특정 셸 초기화 파일을 사용하도록 명령을 변경하십시오.

```
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
```

Windows

Windows에 **kubectl**을 설치하거나 업데이트하려면 다음을 수행합니다.

1. PowerShell 터미널을 엽니다.
2. Amazon S3에서 클러스터의 Kubernetes 버전에 대한 kubectl 바이너리를 다운로드합니다.
   - Kubernetes 1.29

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.29.0/2024-01-04/bin/windows/amd64/kubectl.exe
```

   - Kubernetes 1.28

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.28.5/2024-01-04/bin/windows/amd64/kubectl.exe
```

   - Kubernetes 1.27

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.9/2024-01-04/
bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.26

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.26.12/2024-01-04/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.25

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.25.16/2024-01-04/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.24

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.24.17/2024-01-04/bin/windows/amd64/kubectl.exe
```

- Kubernetes 1.23

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.23.17/2024-01-04/bin/windows/amd64/kubectl.exe
```

3. (선택 사항) 해당 바이너리의 SHA-256 체크섬을 사용하여 다운로드한 바이너리를 확인합니다.

   a. Windows용 클러스터의 Kubernetes 버전에 대한 SHA-256 체크섬을 다운로드합니다.

   - Kubernetes 1.29

   ```
   curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
   eks/1.29.0/2024-01-04/bin/windows/amd64/kubectl.exe.sha256
   ```

   - Kubernetes 1.28

   ```
   curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
   eks/1.28.5/2024-01-04/bin/windows/amd64/kubectl.exe.sha256
   ```

   - Kubernetes 1.27

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.27.9/2024-01-04/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.26`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.26.12/2024-01-04/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.25`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.25.16/2024-01-04/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.24`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.24.17/2024-01-04/bin/windows/amd64/kubectl.exe.sha256
```

- Kubernetes `1.23`

```
curl.exe -O https://s3.us-west-2.amazonaws.com/amazon-
eks/1.23.17/2024-01-04/bin/windows/amd64/kubectl.exe.sha256
```

b. 다운로드한 바이너리에 대해 SHA-256 체크섬을 확인합니다.

```
Get-FileHash kubectl.exe
```

c. 출력에 생성된 체크섬이 다운로드한 `kubectl.sha256` 파일의 체크섬과 일치하는지 확인합니다. PowerShell 출력은 해당하는 대문자 문자열이어야 합니다.

4. 바이너리를 PATH의 폴더에 복사합니다. PATH에 명령줄 유틸리티에 사용하는 기존 디렉터리가 있으면 해당 디렉터리로 바이너리를 복사하세요. 그렇지 않은 경우 다음 단계를 완료합니다.

   a. `C:\bin`과 같이, 명령줄 이진 파일용 새 디렉터리를 생성합니다.

   b. `kubectl.exe` 이진 파일을 새 디렉터리로 복사합니다.

   c. 사용자 또는 시스템 PATH 환경 변수를 편집하여 PATH에 새 디렉터리를 추가합니다.

   d. PowerShell 터미널을 닫고 새 PATH 변수를 가져오기 위해 새 터미널을 엽니다.

3. `kubectl`을 설치한 이후 버전을 확인할 수 있습니다.

```
kubectl version --client
```

kubectl을 처음 설치할 때 아직 어떤 서버와도 통신이 구성되지 않았습니다. 필요에 따라 다른 절차에서 이 구성을 설명합니다. 특정 클러스터와의 통신을 위해 구성을 업데이트해야 하는 경우 다음 명령을 실행할 수 있습니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다. *my-cluster*를 클러스터 이름으로 바꿉니다.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

Amazon EKS 시작하기

시작 안내서를 검토하기 전에 Amazon EKS를 사용하도록 설정되었는지 확인합니다. 자세한 내용은 Amazon EKS 사용 설정 단원을 참조하십시오.

Amazon EKS에서 노드로 새 Kubernetes 클러스터를 생성할 때 사용할 수 있는 시작 가이드는 다음 두 가지입니다.

- Amazon EKS 시작하기 - `eksctl` - 이 시작 가이드는 Amazon EKS에서 Kubernetes 클러스터를 생성 및 관리하기 위한 간단한 명령줄 유틸리티인 `eksctl`을 사용하여 Amazon EKS를 시작할 때 필요한 모든 리소스를 설치하는 데 도움이 됩니다. 자습서의 마지막에서는 애플리케이션을 배포할 수 있는 Amazon EKS에 클러스터를 실행하게 됩니다. 이것이 Amazon EKS를 시작하는 가장 빠르고 쉬운 방법입니다.
- Amazon EKS 시작하기 - AWS Management Console 및 AWS CLI - 이 시작 가이드는 AWS Management Console 및 AWS CLI를 사용하여 Amazon EKS를 시작할 때 필요한 모든 리소스를 만드는 데 도움이 됩니다. 자습서의 마지막에서는 애플리케이션을 배포할 수 있는 Amazon EKS에 클러스터를 실행하게 됩니다. 이 가이드에서는 Amazon EKS 클러스터에 필요한 리소스를 수동으로 생성합니다. 이 절차를 통해 각 리소스가 만들어지는 방식과 리소스가 상호 작용하는 방식을 파악할 수 있습니다.

엄선된 실습 자습서 모음도 제공합니다. 자세한 내용은 AWS 커뮤니티의 Amazon EKS 탐색을 참조하세요.

Amazon EKS 시작하기 - `eksctl`

이 가이드는 Amazon EKS에서 Kubernetes 클러스터를 생성 및 관리하기 위한 간단한 명령줄 유틸리티인 `eksctl`을 사용하여 Amazon Elastic Kubernetes Service(Amazon EKS)를 시작할 때 필요한 모든 리소스를 설치하는 데 도움이 됩니다. 이 자습서의 마지막에서는 애플리케이션을 배포할 수 있는 Amazon EKS에 클러스터를 실행하게 됩니다.

이 가이드의 절차에서는 AWS Management Console을 사용하여 클러스터를 생성할 경우 수동으로 생성해야 하는 여러 리소스를 자동으로 생성합니다. 대부분의 리소스를 수동으로 만들고 리소스가 상호 작용하는 방식을 더 잘 이해하려면 AWS Management Console을 사용하여 클러스터 및 컴퓨팅을 생성합니다. 자세한 내용은 Amazon EKS 시작하기 - AWS Management Console 및 AWS CLI 단원을 참조하십시오.

사전 조건

이 튜토리얼에서는 Amazon EKS 클러스터를 생성하고 관리할 때 필요한 다음 도구 및 리소스를 설치하고 구성해야 합니다.

- **`kubectl`** - Kubernetes 클러스터 작업을 위한 명령줄 도구입니다. 자세한 내용은 kubectl 설치 또는 업데이트 단원을 참조하십시오.
- **`eksctl`** - 많은 개별 태스크를 자동화하는 EKS 클러스터를 사용하기 위한 명령줄 도구입니다. 자세한 내용은 `eksctl` 설명서에서 Installation을 참조하세요.
- 필요한 IAM 권한 - 사용하는 IAM 보안 주체에 Amazon EKS IAM 역할, 서비스 연결 역할, AWS CloudFormation, VPC 및 관련 리소스를 사용할 수 있는 권한이 있어야 합니다. 자세한 내용은 IAM 사용 설명서에서 Amazon Elastic Container Service for Kubernetes에 사용되는 작업, 리소스 및 조건 키Kubernetes 및 서비스 연결 역할 사용 섹션을 참조하세요. 이 가이드의 모든 단계를 동일한 사용자로 완료해야 합니다. 현재 사용자를 확인하려면 다음 명령을 실행합니다.

```
aws sts get-caller-identity
```

1단계: Amazon EKS 클러스터 및 노드 생성

> **Important**
>
> 이 주제에는 가능한 한 간단하고 빠르게 시작하기 위해 기본 설정으로 클러스터와 노드를 생성하는 단계가 포함되어 있습니다. 프로덕션 용도로 클러스터 및 노드를 생성하기 전에 모든 설정을 숙지하고 요구 사항을 충족하는 설정으로 클러스터와 노드를 배포하는 것이 좋습니다. 자세한 내용은 Amazon EKS 클러스터 생성 및 Amazon EKS 노드 섹션을 참조하세요. 일부 설정은 클러스터와 노드를 생성할 때만 사용 설정할 수 있습니다.

다음 노드 유형 중 하나를 사용하여 클러스터를 생성할 수 있습니다. 각 유형에 대한 자세한 내용은 Amazon EKS 노드 부분을 참조하세요. 클러스터를 배포한 후에 다른 노드 유형을 추가할 수 있습니다.

- Fargate – Linux – AWS Fargate에서 Linux 애플리케이션을 실행하려는 경우 이 노드 유형을 선택합니다. Fargate는 Amazon EC2 인스턴스를 관리하지 않고도 Kubernetes Pods를 배포할 수 있는 서버리스 컴퓨팅 엔진입니다.

- 관리형 노드 — Linux - Amazon EC2 인스턴스에서 Amazon Linux 애플리케이션을 실행하려면 이 노드 유형을 선택합니다. 이 가이드에서는 다루지 않지만 Windows 자체 관리형 및 Bottlerocket 노드를 클러스터에 추가할 수도 있습니다.

다음 명령을 사용하여 Amazon EKS 클러스터를 생성합니다. *my-cluster*를 고유한 값으로 바꿀 수 있습니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다. *region-code*를 Amazon EKS에서 지원하는 AWS 리전으로 바꿉니다. AWS 리전 목록은 AWS 일반 참조 가이드의 Amazon EKS 엔드포인트 및 할당량을 참조하세요.

Fargate – Linux

```
eksctl create cluster --name my-cluster --region region-code --fargate
```

Managed nodes – Linux

```
eksctl create cluster --name my-cluster --region region-code
```

클러스터 생성에 몇 분 정도 걸립니다. 생성하는 동안 여러 줄의 출력이 표시됩니다. 출력의 마지막 줄은 다음 예제와 유사합니다.

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

`eksctl`이 `~/.kube`에 `kubectl config` 파일을 생성했거나 컴퓨터터의 `~/.kube`에 있는 기존 `config` 파일에 새 클러스터의 구성을 추가했습니다.

클러스터 생성이 완료되면 AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)에서 `eksctl-`*my-cluster*`-cluster`라는 AWS CloudFormation 스택을 살펴보세요.

2단계: Kubernetes 리소스 보기

1. 클러스터 노드를 확인합니다.

   ```
   kubectl get nodes -o wide
   ```

   예제 출력은 다음과 같습니다.

Fargate – Linux

```
NAME                                                STATUS   ROLES    AGE
   VERSION              INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-
VERSION                   CONTAINER-RUNTIME
fargate-ip-192-0-2-0.region-code.compute.internal   Ready    <none>
  8m3s     v1.2.3-eks-1234567   192.0.2.0     <none>        Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
fargate-ip-192-0-2-1.region-code.compute.internal   Ready    <none>
  7m30s    v1.2.3-eks-1234567   192-0-2-1     <none>        Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

Managed nodes – Linux

```
NAME                                          STATUS   ROLES    AGE    VERSION
          INTERNAL-IP   EXTERNAL-IP   OS-IMAGE         KERNEL-VERSION
       CONTAINER-RUNTIME
ip-192-0-2-0.region-code.compute.internal   Ready    <none>   6m7s
   v1.2.3-eks-1234567   192.0.2.0     192.0.2.2     Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
ip-192-0-2-1.region-code.compute.internal   Ready    <none>   6m4s
   v1.2.3-eks-1234567   192.0.2.1     192.0.2.3     Amazon Linux 2
  1.23.456-789.012.amzn2.x86_64   containerd://1.2.3
```

출력에 표시되는 항목에 대한 자세한 내용은 Kubernetes 리소스 보기 섹션을 참조하세요.

2. 클러스터에서 실행 중인 워크로드를 확인합니다.

```
kubectl get pods -A -o wide
```

예제 출력은 다음과 같습니다.

Fargate – Linux

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE   IP
          NODE                                                NOMINATED NODE
 READINESS GATES
kube-system   coredns-1234567890-abcde   1/1     Running   0          18m
  192.0.2.0   fargate-ip-192-0-2-0.region-code.compute.internal   <none>
   <none>
```

```
kube-system   coredns-1234567890-12345   1/1     Running   0          18m
  192.0.2.1   fargate-ip-192-0-2-1.region-code.compute.internal   <none>
   <none>
```

Managed nodes – Linux

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE     IP
          NODE                                          NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-12345             1/1     Running   0          7m43s
   192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
kube-system   aws-node-67890             1/1     Running   0          7m46s
   192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-abcde   1/1     Running   0          14m
   192.0.2.3   ip-192-0-2-3.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          14m
   192.0.2.4   ip-192-0-2-4.region-code.compute.internal   <none>
 <none>
kube-system   kube-proxy-12345           1/1     Running   0          7m46s
   192.0.2.0   ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   kube-proxy-67890           1/1     Running   0          7m43s
   192.0.2.1   ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
```

출력에 표시되는 항목에 대한 자세한 내용은 Kubernetes 리소스 보기 섹션을 참조하세요.

3단계: 클러스터 및 노드 삭제

이 자습서용으로 생성한 클러스터와 노드의 사용을 끝낸 후에는 다음 명령으로 클러스터와 노드를 삭제하여 정리해야 합니다. 정리하기 전에 이 클러스터로 추가 작업을 수행하려면 다음 단계 섹션을 참조하세요.

```
eksctl delete cluster --name my-cluster --region region-code
```

다음 단계

다음은 클러스터의 기능을 확장하는 데 도움이 되는 설명서 주제입니다.

- 클러스터에 샘플 애플리케이션을 배포합니다.
- 클러스터를 생성한 IAM 보안 주체는 `kubectl` 또는 AWS Management Console를 사용하여 Kubernetes API 서버를 호출할 수 있는 유일한 보안 주체입니다. 다른 IAM 보안 주체가 클러스터에 액세스할 수 있게 하려면 해당 보안 주체를 추가해야 합니다. 자세한 내용은 Kubernetes API에 대한 액세스 권한 부여 및 필요한 권한 단원을 참조하세요.
- 프로덕션 용도로 클러스터를 배포하기 전에 클러스터와 노드에 대한 모든 설정을 숙지하는 것이 좋습니다. Amazon EC2 노드에 대한 SSH 액세스 사용 설정과 같은 일부 설정은 클러스터를 생성할 때 지정해야 합니다.
- 클러스터의 보안을 강화하려면 서비스 계정에 IAM 역할을 사용하도록 Amazon VPC 컨테이너 네트워킹 인터페이스 플러그 인을 구성합니다.

Amazon EKS 시작하기 - AWS Management Console 및 AWS CLI

이 가이드는 AWS Management Console 및 AWS CLI를 사용하여 Amazon Elastic Kubernetes Service(Amazon EKS)를 시작할 때 필요한 모든 리소스를 생성하는 데 도움이 됩니다. 이 가이드에서는 수동으로 각 리소스를 만듭니다. 이 튜토리얼의 마지막에서는 애플리케이션을 배포할 수 있는 Amazon EKS에 클러스터를 실행하게 됩니다.

이 안내서의 절차를 통해 각 리소스가 만들어지는 방식과 리소스가 상호 작용하는 방식을 완벽하게 파악할 수 있습니다. 대부분의 리소스를 자동으로 생성하려는 경우 `eksctl` CLI를 사용하여 클러스터 및 노드를 생성합니다. 자세한 내용은 Amazon EKS 시작하기 - `eksctl` 단원을 참조하십시오.

사전 조건

이 튜토리얼에서는 Amazon EKS 클러스터를 생성하고 관리할 때 필요한 다음 도구 및 리소스를 설치하고 구성해야 합니다.

- AWS CLI - Amazon EKS를 비롯한 AWS 서비스를 사용한 작업을 위한 명령줄 도구입니다. 자세한 내용은 AWS Command Line Interface 사용 설명서에서 AWS CLI 설치, 업데이트, 제거를 참조하세요. AWS CLI 설치 후, 구성 작업도 수행하는 것이 좋습니다. 자세한 내용은 AWS Command Line Interface 사용 설명서에서 `aws configure`를 사용한 빠른 구성 부분을 참조하세요.
- **`kubectl`** - Kubernetes 클러스터 작업을 위한 명령줄 도구입니다. 자세한 내용은 `kubectl` 설치 또는 업데이트 단원을 참조하십시오.

- 필요한 IAM 권한 - 사용하는 IAM 보안 주체에 Amazon EKS IAM 역할, 서비스 연결 역할, AWS CloudFormation, VPC 및 관련 리소스를 사용할 수 있는 권한이 있어야 합니다. 자세한 내용은 IAM 사용 설명서의 Amazon Elastic Kubernetes Service에 사용되는 작업, 리소스 및 조건 키와 서비스 연결 역할 사용을 참조하세요. 이 가이드의 모든 단계를 동일한 사용자로 완료해야 합니다. 현재 사용자를 확인하려면 다음 명령을 실행합니다.

```
aws sts get-caller-identity
```

- Bash 셸에서 이 주제의 단계를 완료하는 것이 좋습니다. Bash 셸을 사용하지 않는 경우 줄 연속 문자 및 변수 설정 및 사용 방식과 같은 일부 스크립트 명령을 통해 셸이 조정되어야 합니다. 또한 쉘의 인용 및 이스케이프 규칙이 다를 수 있습니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI에서 문자열에 따옴표 사용을 참조하세요.

1단계: Amazon EKS 클러스터 생성

> **Important**
>
> 이 주제에는 가능한 한 간단하고 빠르게 시작하기 위해 기본 설정으로 클러스터를 생성하는 단계가 포함되어 있습니다. 프로덕션 용도로 클러스터를 생성하기 전에 모든 설정을 숙지하고 요구 사항을 충족하는 설정으로 클러스터를 배포하는 것이 좋습니다. 자세한 내용은 Amazon EKS 클러스터 생성 단원을 참조하십시오. 일부 설정은 클러스터를 생성할 때만 사용 설정할 수 있습니다.

클러스터를 생성하려면

1. Amazon EKS 요구 사항을 충족하는 퍼블릭 및 프라이빗 서브넷이 있는 Amazon VPC PC를 생성합니다. *region-code*를 Amazon EKS에서 지원하는 AWS 리전으로 바꿉니다. AWS 리전 목록은 AWS 일반 참조 가이드의 Amazon EKS 엔드포인트 및 할당량을 참조하세요. *my-eks-vpc-stack*을 선택한 모든 이름으로 바꿀 수 있습니다.

```
aws cloudformation create-stack \
  --region region-code \
  --stack-name my-eks-vpc-stack \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
```

> **Tip**
>
> 이전 명령이 생성하는 모든 리소스 목록을 보려면 AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다. *my-eks-vpc-stack* 스택을 선택한 다음 리소스(Resources) 탭을 선택합니다.

2. 클러스터 IAM 역할을 생성하고 필요한 Amazon EKS IAM 관리형 정책을 여기에 연결합니다. Amazon EKS에서 관리하는 Kubernetes 클러스터는 사용자를 대신하여 다른 AWS 서비스를 호출하여 서비스와 함께 사용하는 리소스를 관리합니다.

   a. 다음 콘텐츠를 *eks-cluster-role-trust-policy.json*라는 파일에 복사합니다.

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

   b. 역할을 생성합니다.

   ```
   aws iam create-role \
     --role-name myAmazonEKSClusterRole \
     --assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
   ```

   c. 필요한 Amazon EKS 관리형 IAM 정책을 역할에 연결합니다.

   ```
   aws iam attach-role-policy \
     --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
     --role-name myAmazonEKSClusterRole
   ```

3. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.

   콘솔의 오른쪽 상단에 표시된 AWS 리전이 클러스터를 생성하려는 AWS 리전인지 확인합니다. 그렇지 않은 경우 AWS 리전 이름 옆에 있는 드롭다운을 선택하고 사용할 AWS 리전을 선택합니다.

4. 클러스터 추가(Add cluster)를 선택하고 생성(Create)을 선택합니다. 이 옵션이 표시되지 않으면 먼저 왼쪽 검색 창에서 클러스터(Clusters)를 선택합니다.
5. 클러스터 구성(Configure cluster) 페이지에서 다음을 수행합니다.
   a. 클러스터의 이름(Name)(예: **`my-cluster`**)을 입력합니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다.
   b. 클러스터 서비스 역할(Cluster Service Role)에서 *`myAmazonEKSClusterRole`*을 선택합니다.
   c. 나머지 설정을 기본값으로 두고 다음(Next)을 선택합니다.
6. 네트워킹 지정(Specify networking) 페이지에서 다음을 수행합니다.
   a. VPC 드롭다운 목록에서 이전 단계에서 생성한 VPC의 ID를 선택합니다. *`vpc-00x0000x000x0x000`* | *`my-eks-vpc-stack-VPC`*를 예로 들 수 있습니다.
   b. 나머지 설정을 기본값으로 두고 다음(Next)을 선택합니다.
7. 관찰성 구성 페이지에서 다음을 선택합니다.
8. 추가 기능 선택 페이지에서 다음을 선택합니다.

   추가 기능에 대한 자세한 내용은 Amazon EKS 추가 기능 섹션을 참조하세요.
9. 선택한 추가 기능 설정 구성 페이지에서 다음을 선택합니다.
10. 검토 및 생성(Review and create) 페이지에서 생성(Create)을 선택합니다.

    클러스터 이름 오른쪽에 있는 클러스터 상태는 클러스터 프로비저닝 프로세스가 완료될 때까지 몇 분 동안 [생성 중(Creating)]으로 표시됩니다. 상태가 [활성(Active)]이 되면 다음 단계를 진행합니다.

> **Note**
>
> 요청의 가용 영역 중 하나에 Amazon EKS 클러스터를 생성하는 데 충분한 용량이 없다는 오류가 표시될 수 있습니다. 이 경우 오류 출력에는 새 클러스터를 지원할 수 있는 가용 영역이 포함됩니다. 사용자 계정의 지원 가용 영역에 있는 2개 이상의 서브넷을 사용하여 클러스터를 다시 생성합니다. 자세한 내용은 용량 부족 단원을 참조하십시오.

2단계: 클러스터와 통신하도록 컴퓨터 구성

이 부분에서는 클러스터에 대해 kubeconfig 파일을 생성합니다. 이 파일의 설정을 사용하면 kubectl CLI를 사용하여 클러스터와 통신할 수 있습니다.

클러스터와 통신하도록 컴퓨터를 구성하려면

1. 클러스터에 대해 kubeconfig 파일을 생성 또는 업데이트합니다. *region-code*를 클러스터를 생성할 AWS 리전으로 바꿉니다. *my-cluster*를 클러스터 이름으로 바꿉니다.

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   기본적으로 config 파일이 ~/.kube에 생성되거나 새 클러스터의 구성이 ~/.kube의 기존 config 파일에 추가됩니다.
2. 구성을 테스트합니다.

   ```
   kubectl get svc
   ```

   > **Note**
   >
   > 권한 부여 또는 리소스 유형 오류가 표시되는 경우 문제 해결 주제의 권한이 없거나 액세스가 거부됨(kubectl) 부분을 참조하세요.

   예제 출력은 다음과 같습니다.

   ```
   NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
   svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
   ```

3단계: 노드 생성

> **Important**
>
> 이 주제에는 가능한 한 간단하고 빠르게 시작하기 위해 기본 설정으로 노드를 생성하는 단계가 포함되어 있습니다. 프로덕션 용도로 노드를 생성하기 전에 모든 설정을 숙지하고 요구 사항을

충족하는 설정으로 노드를 배포하는 것이 좋습니다. 자세한 내용은 Amazon EKS 노드 섹션을 참조하세요. 일부 설정은 노드를 생성할 때만 사용 설정할 수 있습니다.

다음 노드 유형 중 하나를 사용하여 클러스터를 생성할 수 있습니다. 각 유형에 대한 자세한 내용은 Amazon EKS 노드 부분을 참조하세요. 클러스터를 배포한 후에 다른 노드 유형을 추가할 수 있습니다.

- Fargate – Linux – AWS Fargate에서 Linux 애플리케이션을 실행하려는 경우 이 노드 유형을 선택합니다. Fargate는 Amazon EC2 인스턴스를 관리하지 않고도 Kubernetes Pods를 배포할 수 있는 서버리스 컴퓨팅 엔진입니다.
- 관리형 노드 – Linux – 실행하려는 경우 이 노드 유형을 선택합니다. Amazon EC2 인스턴스에서 Amazon Linux 애플리케이션을 실행할 수 있습니다. 이 가이드에서는 다루지 않지만 Windows 자체 관리형 및 Bottlerocket 노드를 클러스터에 추가할 수도 있습니다.

Fargate – Linux

Fargate 프로파일을 생성합니다. 프로파일에 정의된 기준과 일치하는 기준을 사용하여 Kubernetes Pods를 배포하면 Pods가 Fargate 에 배포됩니다.

Fargate 프로파일을 생성하려면

1. IAM 역할을 생성하고 필요한 Amazon EKS IAM 관리형 정책을 연결합니다. 클러스터가 Fargate 인프라에서 Pods를 생성하는 경우, Fargate 인프라에서 실행되는 구성 요소는 사용자를 대신하여 AWS API를 호출해야 합니다. 이를 통해 Amazon ECR에서 컨테이너 이미지를 가져오거나 로그를 다른 AWS 서버로 라우팅하는 등의 작업을 수행할 수 있습니다. Amazon EKS Pod 실행 역할은 이 작업을 수행할 수 있는 IAM 권한을 제공합니다.

   a. 다음 콘텐츠를 *pod-execution-role-trust-policy*.json라는 파일에 복사합니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다. 계정에서 모든 AWS 리전의 동일한 역할을 사용하려는 경우 *region-code*를 *로 바꾸세요. *111122223333*를 계정 ID로, *my-cluster*를 클러스터의 이름으로 바꿉니다. 계정의 모든 클러스터에 대해 동일한 역할을 사용하려는 경우 *를 *my-cluster*로 바꾸세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
```

```
            "ArnLike": {
               "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
            }
        },
        "Principal": {
          "Service": "eks-fargate-pods.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
      }
    ]
}
```

b. Pod 실행 IAM 역할을 생성합니다.

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
  --assume-role-policy-document file://"pod-execution-role-trust-
policy.json"
```

c. 필요한 Amazon EKS 관리형 IAM 정책을 역할에 연결합니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/
AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

2. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
3. 클러스터(Clusters) 페이지에서 *my-cluster* 클러스터를 선택합니다.
4. ***my-cluster*** 페이지에서 다음을 수행합니다.

   a. 컴퓨팅(Compute) 탭을 선택합니다.

   b. Fargate 프로파일(Fargate Profiles)에서 Fargate 프로파일 추가(Add Fargate Profile)를 선택합니다.

5. Fargate 프로파일 구성(Configure Fargate Profile) 페이지에서 다음을 수행합니다.

   a. 이름(Name)에 Fargate 프로파일(예: ***my-profile***)의 고유한 이름을 입력합니다.

   b. 포드 실행 역할(Pod execution role)의 경우 이전 단계에서 생성한 AmazonEKSFargatePodExecutionRole을 선택합니다.

c. 서브넷(Subnets) 드롭다운을 선택하고 이름에 `Public`이 포함된 서브넷을 모두 선택 취소합니다. Fargate에서 실행되는 Pods에는 프라이빗 서브넷만 지원됩니다.

d. 다음을 선택합니다.

6. Pod 선택 구성(Configure pod selection) 페이지에서 다음을 수행합니다.

a. 네임스페이스(Namespace)에 **`default`**를 입력합니다.

b. 다음을 선택합니다.

7. 검토 및 생성(Review and create) 페이지에서 Fargate 프로필에 대한 정보를 검토하고 생성(Create)을 선택합니다.

8. 몇 분 후 Fargate 프로파일 구성(Fargate Profile configuration) 부분의 상태(Status)가 생성 중(Creating)에서 활성(Active)으로 바뀝니다. 상태가 활성(Active)이 되면 다음 단계를 진행합니다.

9. 모든 Pods를 Fargate에 배포하려는 경우(Amazon EC2 노드에는 없음) 다음을 수행하여 다른 Fargate 프로파일을 생성하고 Fargate에서 기본 이름 해석기(CoreDNS)를 실행합니다.

> **Note**
>
> 이 작업을 수행하지 않으면 현재 아무 노드도 없습니다.

a. Fargate 프로파일(Fargate Profile) 페이지에서 ***`my-profile`***을 선택합니다.

b. Fargate 프로파일(Fargate profiles)에서 Fargate 프로파일 추가(Add Fargate Profile)를 선택합니다.

c. 이름에 ***`CoreDNS`***를 입력합니다.

d. 포드 실행 역할(Pod execution role)의 경우 이전 단계에서 생성한 AmazonEKSFargatePodExecutionRole을 선택합니다.

e. 서브넷(Subnets) 드롭다운을 선택하고 이름에 `Public`이 포함된 서브넷을 모두 선택 취소합니다. Fargate에서 실행되는 Pods에는 프라이빗 서브넷만 지원됩니다.

f. 다음을 선택합니다.

g. 네임스페이스(Namespace)에 **`kube-system`**를 입력합니다.

h. 레이블 일치(Match labels)를 선택하고 레이블 추가(Add label)를 선택합니다.

i. 키(Key)에 **`k8s-app`**을 입력하고 값에 **`kube-dns`**를 입력합니다. 이는 기본 이름 해석기(CoreDNS)를 Fargate에 배포하는 데 필요합니다.

j. 다음을 선택합니다.

k. 검토 및 생성(Review and create) 페이지에서 Fargate 프로파일에 대한 정보를 검토하고 생성(Create)을 선택합니다.

l. 다음 명령을 실행하여 CoreDNS Pods에서 기본 `eks.amazonaws.com/compute-type : ec2` 주석을 제거합니다.

```
kubectl patch deployment coredns \
    -n kube-system \
    --type json \
    -p='[{"op": "remove", "path": "/spec/template/metadata/annotations/
eks.amazonaws.com~1compute-type"}]'
```

> **Note**
>
> 시스템은 추가한 Fargate 프로파일 레이블을 기반으로 2개의 노드를 생성하고 배포합니다. Fargate 노드에 적용할 수 없기 때문에 Node groups(노드 그룹)에 아무것도 표시되지 않지만 Overview(개요) 탭에는 새 노드가 나열됩니다.

Managed nodes – Linux

이전 단계에서 생성한 서브넷과 노드 IAM 역할을 지정하여 관리형 노드 그룹을 생성합니다.

Amazon EC2 Linux 관리형 노드 그룹을 생성하려면 다음을 수행합니다.

1. 노드 IAM 역할을 생성하고 필요한 Amazon EKS IAM 관리형 정책을 연결합니다. Amazon EKS 노드 `kubelet` 데몬은 사용자를 대신하여 AWS API를 호출합니다. 노드는 IAM 인스턴스 프로파일 및 연결 정책을 통해 이 API 호출에 대한 권한을 수신합니다.

   a. 다음 콘텐츠를 *`node-role-trust-policy.json`*라는 파일에 복사합니다.

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

```
    }
  ]
}
```

b. 노드 IAM 역할을 생성합니다.

```
aws iam create-role \
  --role-name myAmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-policy.json"
```

c. 필요한 관리형 IAM 정책을 역할에 연결합니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name myAmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name myAmazonEKSNodeRole
```

2. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
3. 1단계: Amazon EKS 클러스터 생성에서 생성한 클러스터의 이름을 선택합니다(예: *my-cluster*).
4. ***my-cluster*** 페이지에서 다음을 수행합니다.

   a. 컴퓨팅(Compute) 탭을 선택합니다.

   b. 노드 그룹 추가(Add Node Group)를 선택합니다.

5. 노드 그룹 구성(Configure Node Group) 페이지에서 다음을 수행합니다.

   a. 이름(Name)에 관리형 노드 그룹(예: ***my-nodegroup***)의 고유한 이름을 입력합니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다.

   b. 노드 IAM 역할 이름(Node IAM role name)에서 이전 단계에서 생성한 *myAmazonEKSNodeRole* 역할을 선택합니다. 각 노드 그룹은 고유한 IAM 역할을 사용하는 것이 좋습니다.

   c. 다음을 선택합니다.

6. 컴퓨팅 및 크기 조정 구성 설정(Set compute and scaling configuration) 페이지에서 기본값으로 두고 다음(Next)을 선택합니다.
7. 네트워킹 지정(Specify networking) 페이지에서 기본값을 수락하고 다음(Next)을 선택합니다.
8. 검토 및 생성(Review and create) 페이지에서 관리형 노드 그룹 구성을 검토하고 생성을 선택합니다.
9. 몇 분 후 [노드 그룹 구성(Node Group configuration)] 부분의 [상태(Status)]가 [생성 중(Creating)]에서 [활성(Active)]으로 바뀝니다. 상태가 [활성(Active)]이 되면 다음 단계를 진행합니다.

4단계: 리소스 보기

노드 및 Kubernetes 워크로드를 볼 수 있습니다.

노드 및 워크로드를 보려면

1. 좌측 탐색 창에서 클러스터를 선택합니다. 클러스터(Clusters) 목록에서 생성한 클러스터의 이름을 선택합니다(예: *my-cluster*).
2. ***my-cluster*** 페이지에서 다음을 선택합니다.
   a. 컴퓨팅 탭 - 클러스터에 대해 배포된 노드(Nodes) 목록이 표시됩니다. 노드 이름을 선택하면 노드에 대한 자세한 정보를 볼 수 있습니다.
   b. 리소스(Resources) 탭 - Amazon EKS 클러스터에 기본적으로 배포되는 Kubernetes 리소스가 모두 표시됩니다. 자세한 내용을 알아보려면 콘솔에서 모든 리소스 유형을 선택하세요.

5단계: 리소스 삭제

이 튜토리얼용으로 생성한 클러스터와 노드의 사용을 끝낸 후에는 생성한 리소스를 삭제해야 합니다. 리소스를 삭제하기 전에 이 클러스터로 추가 작업을 수행하려면 다음 단계 부분을 참조하세요.

이 가이드에서 생성한 리소스를 삭제하려면

1. 생성한 노드 그룹 또는 Fargate 프로파일을 삭제합니다.
   a. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
   b. 좌측 탐색 창에서 클러스터를 선택합니다. 클러스터 목록에서 *my-cluster*를 선택합니다.
   c. 컴퓨팅(Compute) 탭을 선택합니다.

d. 노드 그룹을 생성한 경우 *my-nodegroup* 노드 그룹을 한 다음 삭제(Delete)를 선택합니다. ***my-nodegroup***를 입력한 다음 삭제를 선택합니다.

e. 생성한 각 Fargate 프로파일에 대해 선택하고 삭제(Delete)를 선택합니다. 프로파일의 이름을 입력한 다음 삭제(Delete)를 선택합니다.

> **Note**
>
> 두 번째 Fargate 프로파일을 삭제할 때 첫 번째 프로파일이 삭제를 마칠 때까지 기다려야 할 수 있습니다.

f. 노드 그룹 또는 Fargate 프로파일이 삭제 때까지 계속하지 마세요.

2. 클러스터를 삭제합니다.

a. 좌측 탐색 창에서 클러스터를 선택합니다. 클러스터 목록에서 *my-cluster*를 선택합니다.

b. 클러스터 삭제(Delete cluster)를 선택합니다.

c. ***my-cluster***을 입력한 다음 삭제(Delete)를 선택합니다. 클러스터가 삭제될 때까지 계속하지 마세요.

3. 생성한 VPC AWS CloudFormation 스택을 삭제합니다.

a. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.

b. *my-eks-vpc-stack* 스택을 선택한 다음 삭제(Delete)를 선택합니다.

c. ***my-eks-vpc-stack*** 삭제(Delete my-eks-vpc-stack) 확인 대화 상자에서 스택 삭제(Delete stack)를 선택합니다.

4. 생성한 IAM 역할을 삭제합니다.

a. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.

b. 왼쪽 탐색 창에서 역할을 선택합니다.

c. 목록에서 생성한 각 역할(***myAmazonEKSClusterRole*** 및 AmazonEKSFargatePodExecutionRole 또는 *myAmazonEKSNodeRole*)을 선택합니다. 삭제(Delete)를 선택하고 요청한 확인 텍스트를 입력한 다음 삭제(Delete)를 선택합니다.

다음 단계

다음은 클러스터의 기능을 확장하는 데 도움이 되는 설명서 주제입니다.

- 클러스터를 생성한 IAM 보안 주체는 `kubectl` 또는 AWS Management Console를 사용하여 Kubernetes API 서버를 호출할 수 있는 유일한 보안 주체입니다. 다른 IAM 보안 주체가 클러스터에 액세스할 수 있게 하려면 해당 보안 주체를 추가해야 합니다. 자세한 내용은 Kubernetes API에 대한 액세스 권한 부여 및 필요한 권한 단원을 참조하세요.
- 클러스터에 샘플 애플리케이션을 배포합니다.
- 프로덕션 용도로 클러스터를 배포하기 전에 클러스터와 노드에 대한 모든 설정을 숙지하는 것이 좋습니다. Amazon EC2 노드에 대한 SSH 액세스 사용 설정과 같은 일부 설정은 클러스터를 생성할 때 지정해야 합니다.
- 클러스터의 보안을 강화하려면 서비스 계정에 IAM 역할을 사용하도록 Amazon VPC 컨테이너 네트워킹 인터페이스 플러그 인을 구성합니다.

Amazon EKS 클러스터

Amazon EKS 클러스터에는 두 가지 기본 구성 요소가 있습니다.

- Amazon EKS 제어 영역
- 제어 영역에 등록된 Amazon EKS 노드

Amazon EKS 컨트롤 플레인에는 `etcd` 및 Kubernetes API 서버와 같은 Kubernetes 소프트웨어를 실행하는 컨트롤 플레인 노드가 포함되어 있습니다. 컨트롤 플레인은 AWS에서 관리하는 계정에서 실행되며, Kubernetes API는 클러스터와 연결된 Amazon EKS 엔드포인트를 통해 노출됩니다. 각 Amazon EKS 클러스터 제어 영역은 단일 테넌트이고 고유하며 자체 Amazon EC2 인스턴스 집합에서 실행합니다.

`etcd` 노드에 의해 저장된 모든 데이터 및 연결된 Amazon EBS 볼륨은 AWS KMS를 사용하여 암호화됩니다. 클러스터 제어 플레인은 여러 가용 영역에 프로비저닝되고 Elastic Load Balancing Network Load Balancer와 마주합니다. Amazon EKS는 또한 VPC 서브넷의 탄력적 네트워크 인터페이스를 프로비저닝하여 제어 영역 인스턴스에서 노드(예: `kubectl exec logs proxy` 데이터 흐름)로의 연결을 제공합니다.

Important

Amazon EKS 환경에서 `etcd` 스토리지는 업스트림 지침에 따라 8GiB로 제한됩니다. 다음 명령을 실행하여 현재 데이터베이스 크기에 대한 지표를 모니터링할 수 있습니다. 클러스터에서 Kubernetes 버전이 `1.28` 아래인 경우 *`apiserver_storage_size_bytes`*를 다음으로 바꿉니다.

- Kubernetes 버전 `1.27` 및 `1.26` - **`apiserver_storage_db_total_size_in_bytes`**
- Kubernetes 버전 `1.25` 이하 - **`etcd_db_total_size_in_bytes`**

```
kubectl get --raw=/metrics | grep "apiserver_storage_size_bytes"
```

Amazon EKS 노드는 AWS 계정에서 실행되고, 클러스터에 대해 생성된 인증서 파일 및 API 서버 엔드포인트를 통해 클러스터의 제어 영역에 연결됩니다.

> **Note**
>
> - Amazon EKS의 다양한 구성 요소가 어떻게 작동하는지는 Amazon EKS 네트워킹에서 살펴볼 수 있습니다.
> - 연결된 클러스터는 Amazon EKS Connector 섹션을 참조하세요.

주제

Amazon EKS 클러스터 생성

이 주제에서는 사용 가능한 옵션에 대한 개요와 Amazon EKS 클러스터를 생성할 때 고려해야 할 사항을 설명합니다. AWS Outpost에서 클러스터를 생성해야 하는 경우 AWS Outposts의 Amazon EKS용 로컬 클러스터 부분을 참조하세요. Amazon EKS 클러스터를 처음 생성하는 경우 Amazon EKS 시작하기 가이드 중 한 가지를 따르는 것이 좋습니다. 이 가이드를 사용하면 사용 가능한 모든 옵션으로 확장하지 않고 간단한 기본 클러스터를 생성할 수 있습니다.

필수 조건

- Amazon EKS 요구 사항을 충족하는 기존 VPC 및 서브넷 보유. 프로덕션 용도로 클러스터를 배포하기 전에 VPC 및 서브넷 요구 사항을 충분히 이해하는 것이 좋습니다. VPC 및 서브넷이 없는 경우 Amazon EKS 제공 AWS CloudFormation 템플릿을 사용하여 생성할 수 있습니다.

- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 kubectl 설치 또는 업데이트 부분을 참조하세요.
- AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.
- Amazon EKS 클러스터를 `create` 및 `describe` 할 수 있는 권한이 있는 IAM 보안 주체. 자세한 내용은 Outpost에서 로컬 Kubernetes 클러스터 생성 및 모든 클러스터 나열 또는 설명 단원을 참조하세요.

Amazon EKS 클러스터 생성하는 방법

1. 이미 클러스터 IAM 역할이 있거나 `eksctl`을 사용하여 클러스터를 생성하려는 경우 이 단계를 건너뛸 수 있습니다. 기본적으로 `eksctl`은 사용자를 위한 역할을 생성합니다.

   Amazon EKS 클러스터 IAM 역할 생성

   1. 다음 명령을 실행하여 IAM 신뢰 정책 JSON 파일을 생성합니다.

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

2. Amazon EKS 클러스터 IAM 역할을 생성합니다. 필요한 경우 *eks-cluster-role-trust-policy.json* 앞에 이전 단계에서 파일을 작성한 컴퓨터의 경로를 붙입니다. 명령은 사용자가 이전 단계에서 생성한 신뢰 정책을 역할에 연결합니다. IAM 보안 주체를 생성하려면 역할을 생성하는 IAM 보안 주체에 `iam:CreateRole` 작업(권한)을 할당해야 합니다.

```
aws iam create-role --role-name myAmazonEKSClusterRole --assume-role-policy-
document file://"eks-cluster-role-trust-policy.json"
```

3. Amazon EKS 관리형 정책을 할당하거나 사용자 지정 정책을 생성할 수 있습니다. 사용자 지정 정책에서 사용해야 하는 최소 권한은 Amazon EKS 클러스터 IAM 역할을(를) 참조하십시오.

   Amazon EKS 관리형 `AmazonEKSClusterPolicy`를 역할에 연결합니다. IAM 정책을 IAM 보안 주체에 연결하려면 정책을 연결하는 보안 주체에 `iam:AttachUserPolicy` 또는 `iam:AttachRolePolicy`의 IAM 작업(권한) 중 하나를 할당해야 합니다.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name myAmazonEKSClusterRole
```

2. Amazon EKS 클러스터를 생성합니다.

`eksctl`, AWS Management Console 또는 AWS CLI를 사용하여 클러스터를 생성할 수 있습니다.

eksctl

전제 조건

디바이스 또는 `0.175.0`에 설치된 버전 AWS CloudShell 이상의 `eksctl` 명령줄 도구. `eksctl`을 설치 또는 업그레이드하려면 `eksctl` 설명서에서 Installation을 참조하세요.

클러스터 생성

기본 AWS 리전에서 Amazon EKS 기본 Kubernetes 버전으로 Amazon EKS `IPv4` 클러스터를 생성합니다. 명령을 실행하기 전에 다음과 같은 바꾸기를 합니다.

- *region-code*를 클러스터를 생성할 AWS 리전으로 바꿉니다.
- *my-cluster*를 클러스터 이름으로 바꿉니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다. 이름은 클러스터를 생성하는 AWS 리전과 AWS 계정 내에서 고유해야 합니다.
- *1.28*를 모든 Amazon EKS 지원 버전으로 변경합니다.

- 요구 사항을 충족하는 vpc-private-subnets의 값을 변경합니다. 또한 추가 ID를 추가할 수 있습니다. 2개 이상의 서브넷 ID를 지정해야 합니다. 퍼블릭 서브넷을 지정하려는 경우 --vpc-private-subnets을 --vpc-public-subnets으로 변경할 수 있습니다. 퍼블릭 서브넷에는 인터넷 게이트웨이에 대한 경로가 포함된 연결된 라우팅 테이블이 있지만 프라이빗 서브넷에는 연결된 라우팅 테이블이 없습니다. 가능하면 프라이빗 서브넷을 사용하는 것이 좋습니다.

  선택한 서브넷은 Amazon EKS 서브넷 요구 사항을 충족해야 합니다. 서브넷을 선택하기 전에 Amazon EKS VPC 및 서브넷 요구 사항 및 고려 사항 모두를 숙지하는 것이 좋습니다.

```
eksctl create cluster --name my-cluster --region region-code --version 1.28 --
vpc-private-subnets subnet-ExampleID1,subnet-ExampleID2 --without-nodegroup
```

클러스터 프로비저닝에는 몇 분 정도 걸립니다. 클러스터가 생성되는 동안 여러 줄의 출력이 나타납니다. 출력의 마지막 줄은 다음 예제와 유사합니다.

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

> **Tip**
>
> eksctl을 사용하여 클러스터를 생성할 때 지정할 수 있는 대부분의 옵션을 보려면 **eksctl create cluster --help** 명령을 사용합니다. 사용 가능한 옵션을 모두 확인하려면 config 파일을 사용할 수 있습니다. 자세한 내용은 eksctl 설명서에서 config 파일 사용 및 config 파일 스키마 부분을 참조하세요. GitHub에서 구성 파일 예제를 찾을 수 있습니다.

설정(선택 사항)

다음은 필요한 경우 이전 명령에 추가해야 하는 선택 가능한 설정입니다. 클러스터 생성 후가 아닌 생성할 때만 이러한 옵션을 사용 설정할 수 없습니다. 이러한 옵션을 지정해야 하는 경우 eksctl 구성 파일을 사용하여 클러스터를 생성한 다음 이전 명령을 사용하지 않고 설정을 지정해야 합니다.

- 생성하는 네트워크 인터페이스에 Amazon EKS가 할당하는 보안 그룹을 하나 이상 지정하는 경우 securityGroup 옵션을 지정합니다.

보안 그룹 선택 여부에 관계없이 Amazon EKS는 클러스터와 VPC 간에 통신을 사용 설정할 수 있는 보안 그룹을 생성합니다. Amazon EKS는 이 보안 그룹과 사용자가 선택한 보안 그룹을 생성하는 네트워크 인터페이스에 연결합니다. Amazon EKS에서 생성하는 클러스터 보안 그룹에 대한 자세한 내용을 알아보려면 the section called "보안 그룹 요구 사항" 부분을 참조하세요. Amazon EKS가 생성하는 클러스터 보안 그룹에서 규칙을 수정할 수 있습니다.

- Kubernetes가 서비스 IP 주소를 할당하는 IPv4 Classless Inter-Domain Routing(CIDR) 블록 Kubernetes를 지정하려는 경우 `serviceIPv4CIDR` 옵션을 지정합니다.

  자체 범위를 지정하면 VPC에 피어링되거나 연결된 Kubernetes 서비스 및 기타 네트워크 사이의 충돌을 방지할 수 있습니다. CIDR 표기법으로 범위를 입력합니다. 예: `10.2.0.0/16`.

  CIDR 블록은 다음 요구 사항을 충족해야 합니다.

  - 다음 범위 `10.0.0.0/8`, `172.16.0.0/12` 또는 `192.168.0.0/16` 중 하나 내에 있어야 합니다..
  - 최소 크기는 `/24`고 최대 크기는 `/12`입니다.
  - Amazon EKS 리소스의 VPC 범위와 겹치지 않습니다.

  이 옵션은 IPv4 주소 패밀리를 사용할 때 및 클러스터 생성 시에만 지정할 수 있습니다. 이 옵션을 지정하지 않은 경우 Kubernetes는 `10.100.0.0/16` 또는 `172.20.0.0/16` CIDR 블록 중 하나에서 서비스 IP 주소를 할당합니다.
- 클러스터를 생성하고 클러스터가 IPv4 주소 대신 Pods와 서비스에 IPv6 주소를 할당하도록 하려면 `ipFamily` 옵션을 지정합니다.

  Kubernetes는 기본적으로 Pods와 서비스에 IPv4 주소를 할당합니다. IPv6 패밀리를 사용하기로 결정하기 전에 the section called "VPC 요구 사항 및 고려 사항", the section called "서브넷 요구 사항 및 고려 사항", the section called "보안 그룹 요구 사항" 및 the section called "IPv6" 주제의 모든 고려 사항 및 요구 사항을 숙지해야 합니다. IPv6 패밀리를 선택하는 경우 IPv4 패밀리와 마찬가지로 Kubernetes의 주소 범위를 지정하여 IPv6 서비스 주소를 할당할 수 없습니다. Kubernetes는 고유 로컬 주소 범위(`fc00::/7`)에서 서비스 주소를 할당합니다.

AWS Management Console

클러스터 생성

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.

2. 클러스터 추가(Add cluster)를 선택하고 생성(Create)을 선택합니다.
3. 클러스터 구성(Configure cluster) 페이지에서 다음 필드를 입력합니다.
   - 이름(Name) - 클러스터의 이름 AWS 계정에서 고유해야 합니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다. 이름은 클러스터를 생성하는 AWS 리전과 AWS 계정 내에서 고유해야 합니다.
   - Kubernetes 버전 - 클러스터에 대해 사용할 Kubernetes 버전. 이전 버전이 필요한 경우가 아니면 최신 버전을 선택하는 것이 좋습니다.
   - 클러스터 서비스 역할 - 생성한 Amazon EKS 클러스터 IAM 역할을 선택하여 Kubernetes 컨트롤 플레인이 사용자를 대신하여 AWS 리소스를 관리하게 할 수 있습니다.
   - 비밀 암호화 - (선택 사항) KMS 키를 사용하여 Kubernetes 비밀의 비밀 암호화를 사용하도록 선택합니다. 클러스터를 생성한 후에도 이 기능을 사용 설정할 수 있습니다. 이 기능을 사용 설정하기 전에 기존 클러스터에서 비밀 암호화 활성화의 정보를 숙지해야 합니다.
   - 태그(Tags) - (선택 사항) 클러스터에 태그를 추가합니다. 자세한 내용은 Amazon EKS 리소스 태깅 단원을 참조하십시오.

     이 페이지를 모두 완료하면 다음을 선택합니다.
4. 네트워킹 지정(Specify networking) 페이지에서 다음 필드의 값을 선택합니다.
   - VPC - 클러스터를 생성하려면 Amazon EKS VPC 요구 사항을 충족하는 기존 VPC를 선택합니다. VPC를 선택하기 전에 Amazon EKS VPC 및 서브넷 요구 사항과 고려 사항의 요구 사항 및 고려 사항 모두를 숙지하는 것이 좋습니다. 클러스터 생성 후에는 사용하려는 VPC를 변경할 수 없습니다. 목록에 기존 VPC가 없는 경우 먼저 VPC를 생성해야 합니다. 자세한 내용은 Amazon EKS 클러스터에 대해 VPC 생성 단원을 참조하십시오.
   - 서브넷(Subnets) - 기본적으로 이전 필드에서 지정한 VPC에서 사용 가능한 모든 서브넷이 사전 선택됩니다. 두 개 이상을 선택해야 합니다.

     선택한 서브넷은 Amazon EKS 서브넷 요구 사항을 충족해야 합니다. 서브넷을 선택하기 전에 Amazon EKS VPC 및 서브넷 요구 사항 및 고려 사항 모두를 숙지하는 것이 좋습니다.

     보안 그룹(Security groups) – (선택 사항) 생성하는 네트워크 인터페이스에 Amazon EKS가 연결하려는 보안 그룹을 하나 이상 지정합니다.

     보안 그룹 선택 여부에 관계없이 Amazon EKS는 클러스터와 VPC 간에 통신을 사용 설정할 수 있는 보안 그룹을 생성합니다. Amazon EKS는 이 보안 그룹과 사용자가 선택한 보안 그룹을 생성하는 네트워크 인터페이스에 연결합니다. Amazon EKS에서 생성하는 클

러스터 보안 그룹에 대한 자세한 내용을 알아보려면 the section called “보안 그룹 요구 사항” 부분을 참조하세요. Amazon EKS가 생성하는 클러스터 보안 그룹에서 규칙을 수정할 수 있습니다.

- 클러스터 IP 주소 패밀리 선택 - IPv4와 IPv6 중 하나를 선택할 수 있습니다.

  Kubernetes는 기본적으로 Pods와 서비스에 IPv4 주소를 할당합니다. IPv6 패밀리를 사용하기로 결정하기 전에 the section called “VPC 요구 사항 및 고려 사항”, the section called “서브넷 요구 사항 및 고려 사항”, the section called “보안 그룹 요구 사항” 및 the section called “IPv6” 주제의 모든 고려 사항 및 요구 사항을 숙지해야 합니다. IPv6 패밀리를 선택하는 경우 IPv4 패밀리와 마찬가지로 Kubernetes의 주소 범위를 지정하여 IPv6 서비스 주소를 할당할 수 없습니다. Kubernetes는 고유 로컬 주소 범위(fc00::/7)에서 서비스 주소를 할당합니다.

- (선택 사항) Kubernetes 서비스 IP 주소 범위 구성을 선택하고 서비스 **IPv4** 범위를 지정합니다.

  자체 범위를 지정하면 VPC에 피어링되거나 연결된 Kubernetes 서비스 및 기타 네트워크 사이의 충돌을 방지할 수 있습니다. CIDR 표기법으로 범위를 입력합니다. 예: 10.2.0.0/16.

  CIDR 블록은 다음 요구 사항을 충족해야 합니다.

  - 다음 범위 10.0.0.0/8, 172.16.0.0/12 또는 192.168.0.0/16 중 하나 내에 있어야 합니다..
  - 최소 크기는 /24고 최대 크기는 /12입니다.
  - Amazon EKS 리소스의 VPC 범위와 겹치지 않습니다.

  이 옵션은 IPv4 주소 패밀리를 사용할 때 및 클러스터 생성 시에만 지정할 수 있습니다. 이 옵션을 지정하지 않은 경우 Kubernetes는 10.100.0.0/16 또는 172.20.0.0/16 CIDR 블록 중 하나에서 서비스 IP 주소를 할당합니다.

- 클러스터 엔드포인트 액세스(Cluster endpoint access)에서 옵션을 선택합니다. 클러스터를 생성한 후에는 이 옵션을 변경할 수 있습니다. 기본값이 아닌 옵션을 선택하기 전에 옵션과 그 의미를 숙지해야 합니다. 자세한 내용은 Amazon EKS 클러스터 엔드포인트 액세스 제어 단원을 참조하십시오.

  이 페이지를 모두 완료하면 다음을 선택합니다.

5. (선택 사항) 관찰성 구성 페이지에서 설정할 지표 및 컨트롤 플레인 로깅 옵션을 선택합니다. 기본적으로 각 로그 유형은 해제되어 있습니다.

- Prometheus 지표 옵션에 자세한 내용은 클러스터를 생성할 때 Prometheus 지표 켜기 섹션을 참조하세요.
- 컨트롤 플레인 로깅 옵션에 대한 자세한 내용은 Amazon EKS 컨트롤 플레인 로깅 섹션을 참조하세요.

이 페이지를 모두 완료하면 다음을 선택합니다.

6. Select add-ons(추가 기능 선택) 페이지에서 클러스터에 추가할 추가 기능을 선택합니다. Amazon EKS add-ons(Amazon EKS 추가 기능)와 AWS Marketplace add-ons(추가 기능)는 필요한 만큼 선택할 수 있습니다. 설치하려는 AWS Marketplace add-ons(추가 기능)가 목록에 없는 경우 검색 상자에 텍스트를 입력하여 사용 가능한 AWS Marketplace add-ons(추가 기능)를 검색할 수 있습니다. category(카테고리), vendor(공급업체) 또는 pricing model(요금 모델)로 검색한 다음에 검색 결과 중에서 추가 기능을 선택할 수도 있습니다. 이 페이지를 모두 완료하면 다음을 선택합니다.
7. 선택한 추가 기능 설정 구성 페이지에서 설치할 버전을 선택합니다. 클러스터 생성 후 언제든지 최신 버전으로 업데이트할 수 있습니다. 클러스터 생성 후 각 추가 기능의 구성을 업데이트할 수 있습니다. 추가 기능 구성에 대한 자세한 내용은 추가 기능 업데이트 섹션을 참조하세요. 이 페이지를 모두 완료하면 다음을 선택합니다.
8. 검토 및 생성 페이지에서 이전 페이지에서 입력하거나 선택한 정보를 검토합니다. 변경해야 하는 경우 편집(Edit)을 선택합니다 만족하는 경우 생성(Create)을 선택합니다. 클러스터가 프로비저닝되는 동안 상태(Status) 필드에는 생성 중(CREATING)이 표시됩니다.

> **Note**
>
> 요청의 가용 영역 중 하나에 Amazon EKS 클러스터를 생성하는 데 충분한 용량이 없다는 오류가 표시될 수 있습니다. 이 경우 오류 출력에는 새 클러스터를 지원할 수 있는 가용 영역이 포함됩니다. 사용자 계정의 지원 가용 영역에 있는 2개 이상의 서브넷을 사용하여 클러스터를 다시 생성합니다. 자세한 내용은 용량 부족 단원을 참조하십시오.

클러스터 프로비저닝에는 몇 분 정도 걸립니다.

AWS CLI

클러스터 생성

1. 다음 명령을 사용하여 클러스터를 생성합니다. 명령을 실행하기 전에 다음과 같은 바꾸기를 합니다.
   - *region-code*를 클러스터를 생성할 AWS 리전으로 바꿉니다.
   - *my-cluster*를 클러스터 이름으로 바꿉니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다. 이름은 클러스터를 생성하는 AWS 리전과 AWS 계정 내에서 고유해야 합니다.
   - *1.29*를 모든 Amazon EKS 지원 버전으로 변경합니다.
   - *111122223333*를 계정 ID로, *myAmazonEKSClusterRole*를 클러스터 IAM 역할의 이름으로 변경합니다.
   - subnetIds의 값을 고유한 값으로 바꿉니다. 또한 추가 ID를 추가할 수 있습니다. 2개 이상의 서브넷 ID를 지정해야 합니다.

     선택한 서브넷은 Amazon EKS 서브넷 요구 사항을 충족해야 합니다. 서브넷을 선택하기 전에 Amazon EKS VPC 및 서브넷 요구 사항 및 고려 사항 모두를 숙지하는 것이 좋습니다.
   - 보안 그룹 ID를 지정하지 않으려면 명령에서 ,securityGroupIds=sg-*ExampleID1*을 제거합니다. 하나 이상의 보안 그룹 ID를 지정하려는 경우 securityGroupIds의 값을 고유한 값으로 바꿉니다. 또한 추가 ID를 추가할 수 있습니다.

     보안 그룹 선택 여부에 관계없이 Amazon EKS는 클러스터와 VPC 간에 통신을 사용 설정할 수 있는 보안 그룹을 생성합니다. Amazon EKS는 이 보안 그룹과 사용자가 선택한 보안 그룹을 생성하는 네트워크 인터페이스에 연결합니다. Amazon EKS에서 생성하는 클러스터 보안 그룹에 대한 자세한 내용을 알아보려면 the section called “보안 그룹 요구 사항” 부분을 참조하세요. Amazon EKS가 생성하는 클러스터 보안 그룹에서 규칙을 수정할 수 있습니다.

```
aws eks create-cluster --region region-code --name my-cluster --kubernetes-
version 1.29 \
   --role-arn arn:aws:iam::111122223333:role/myAmazonEKSClusterRole \
   --resources-vpc-config
 subnetIds=subnet-ExampleID1,subnet-ExampleID2,securityGroupIds=sg-ExampleID1
```

> **Note**
>
> 요청의 가용 영역 중 하나에 Amazon EKS 클러스터를 생성하는 데 충분한 용량이 없다는 오류가 표시될 수 있습니다. 이 경우 오류 출력에는 새 클러스터를 지원할 수 있는 가용 영역이 포함됩니다. 사용자 계정의 지원 가용 영역에 있는 2개 이상의 서브넷을 사용하여 클러스터를 다시 생성합니다. 자세한 내용은 용량 부족 단원을 참조하십시오.

설정(선택 사항)

다음은 필요한 경우 이전 명령에 추가해야 하는 선택 가능한 설정입니다. 클러스터 생성 후가 아닌 생성할 때만 이러한 옵션을 사용 설정할 수 없습니다.

- Kubernetes가 서비스 IP 주소를 할당하는 IPv4 Classless Inter-Domain Routing(CIDR) 블록을 지정하려는 경우 다음 명령에 **`--kubernetes-network-config serviceIpv4Cidr=`*`CIDR block`***을 추가하여 해당 블록을 지정해야 합니다.

  자체 범위를 지정하면 VPC에 피어링되거나 연결된 Kubernetes 서비스 및 기타 네트워크 사이의 충돌을 방지할 수 있습니다. CIDR 표기법으로 범위를 입력합니다. 예: `10.2.0.0/16`.

  CIDR 블록은 다음 요구 사항을 충족해야 합니다.

  - 다음 범위 `10.0.0.0/8`, `172.16.0.0/12` 또는 `192.168.0.0/16` 중 하나 내에 있어야 합니다..
  - 최소 크기는 `/24`고 최대 크기는 `/12`입니다.
  - Amazon EKS 리소스의 VPC 범위와 겹치지 않습니다.

  이 옵션은 IPv4 주소 패밀리를 사용할 때 및 클러스터 생성 시에만 지정할 수 있습니다. 이 옵션을 지정하지 않은 경우 Kubernetes는 `10.100.0.0/16` 또는 `172.20.0.0/16` CIDR 블록 중 하나에서 서비스 IP 주소를 할당합니다.
- 클러스터를 생성하고 클러스터가 IPv4 주소 대신 Pods와 서비스에 IPv6 주소를 할당하도록 하려면 다음 명령에 **`--kubernetes-network-config ipFamily=ipv6`**을 추가합니다.

  Kubernetes는 기본적으로 Pods와 서비스에 IPv4 주소를 할당합니다. IPv6 패밀리를 사용하기로 결정하기 전에 the section called “VPC 요구 사항 및 고려 사항”, the section

called “서브넷 요구 사항 및 고려 사항”, the section called “보안 그룹 요구 사항” 및 the section called “IPv6” 주제의 모든 고려 사항 및 요구 사항을 숙지해야 합니다. IPv6 패밀리를 선택하는 경우 IPv4 패밀리와 마찬가지로 Kubernetes의 주소 범위를 지정하여 IPv6 서비스 주소를 할당할 수 없습니다. Kubernetes는 고유 로컬 주소 범위(fc00::/7)에서 서비스 주소를 할당합니다.

2. 클러스터를 프로비저닝하는 데 몇 분 정도 걸립니다. 다음 명령을 사용하여 클러스터의 상태를 쿼리할 수 있습니다.

```
aws eks describe-cluster --region region-code --name my-cluster --query
 "cluster.status"
```

반환되는 출력이 ACTIVE가 되면 다음 단계로 진행하세요.

3. eksctl을 사용하여 클러스터를 생성한 경우 이 단계를 건너뛸 수 있습니다. 이는 eksctl이 사용자 대신 이 단계를 이미 완료했기 때문입니다. kubectl config 파일에 새 컨텍스트를 추가하여 kubectl이 클러스터와 통신하도록 사용 설정합니다. 파일 생성 및 업데이트 방법에 대한 자세한 내용을 알아보려면 Amazon EKS 클러스터용 kubeconfig 파일 생성 또는 업데이트 부분을 참조하세요.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

예제 출력은 다음과 같습니다.

```
Added new context arn:aws:eks:region-code:111122223333:cluster/my-cluster to /home/
username/.kube/config
```

4. 다음 명령을 실행하여 클러스터와의 통신을 확인합니다.

```
kubectl get svc
```

예제 출력은 다음과 같습니다.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

5. (권장 사항) 일부 Amazon EKS 추가 기능을 사용하거나 개별 Kubernetes 워크로드에 특정 AWS Identity and Access Management(IAM) 권한을 부여하려면 클러스터에 알맞은 IAM OpenID Connect(OIDC) 제공업체를 생성합니다. 클러스터에 대해 IAM OIDC 제공업체를 한 번만 생성하

면 됩니다. Amazon EKS 추가 기능에 대한 자세한 내용은 Amazon EKS 추가 기능 부분을 참조하세요. 워크로드에 특정 IAM 권한을 할당하는 방법에 대한 자세한 내용은 서비스 계정에 대한 IAM 역할 부분을 참조하세요.

6. (권장 사항) Amazon EC2 노드를 클러스터에 배포하기 전에 Amazon VPC CNI plugin for Kubernetes 플러그 인에 대한 클러스터를 구성합니다. 기본적으로 플러그 인은 클러스터와 함께 설치되어 있습니다. 클러스터에 Amazon EC2 노드를 추가하면 추가하는 각 Amazon EC2 노드에 플러그 인이 자동으로 배포됩니다. 플러그 인을 사용하려면 다음 IAM 정책 중 하나를 IAM 역할에 연결해야 합니다.

   AmazonEKS_CNI_Policy 관리형 IAM 정책

   클러스터에서 IPv4 패밀리를 사용한 경우

   생성한 IAM 정책

   클러스터에서 IPv6 패밀리를 사용한 경우

   정책을 연결하는 IAM 역할은 노드 IAM 역할이거나 플러그 인에만 사용되는 전용 역할일 수 있습니다. 이 역할에 정책을 연결하는 것이 좋습니다. 역할 생성에 대한 자세한 내용을 알아보려면 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 또는 Amazon EKS 노드 IAM 역할 부분을 참조하세요.

7. AWS Management Console을 사용하여 클러스터를 배포한 경우 이 단계를 건너뛸 수 있습니다. AWS Management Console은 기본적으로 Amazon VPC CNI plugin for Kubernetes, CoreDNS 및 kube-proxy Amazon EKS 추가 기능을 배포합니다.

   eksctl 또는 AWS CLI 중 하나를 사용하여 클러스터를 배포하는 경우 Amazon VPC CNI plugin for Kubernetes, CoreDNS 및 kube-proxy 자체 관리형 추가 기능이 배포됩니다. 클러스터와 함께 배포되는 Amazon VPC CNI plugin for Kubernetes, CoreDNS 및 kube-proxy 자체 관리형 추가 기능을 Amazon EKS 추가 기능으로 마이그레이션할 수 있습니다. 자세한 내용은 Amazon EKS 추가 기능 단원을 참조하십시오.

8. (선택 사항) 아직 수행하지 않은 경우 클러스터의 Prometheus 지표를 활성화할 수 있습니다. 자세한 내용은 Amazon Managed Service for Prometheus 사용 설명서의 스크레이퍼 생성을 참조하세요.

9. Prometheus 지표를 활성화한 경우 스크레이퍼에 클러스터 내 권한을 부여하려면 aws-auth ConfigMap을 설정해야 합니다. 자세한 내용은 Amazon Managed Service for Prometheus 사용 설명서의 Amazon EKS 클러스터 구성을 참조하세요.

10. Amazon EBS 볼륨을 사용하는 클러스터에 워크로드를 배포할 계획이고 1.23 이상의 클러스터를 생성한 경우 워크로드를 배포하기 전에 클러스터에 Amazon EBS CSI 드라이버를 설치해야 합니다.

권장되는 다음 단계:

- 클러스터를 생성한 IAM 보안 주체는 클러스터에 대한 액세스 권한이 있는 유일한 보안 주체입니다. 클러스터에 액세스할 수 있도록 다른 IAM 보안 주체에 권한을 부여합니다.
- 클러스터를 생성한 IAM 보안 주체에 사전 조건에서 참조하는 최소 IAM 권한만 있는 경우 해당 보안 주체에 대한 Amazon EKS 권한을 추가할 수 있습니다. IAM 보안 주체에 Amazon EKS 권한 부여에 대한 자세한 내용을 알아보려면 Amazon EKS용 자격 증명 및 액세스 관리 부분을 참조하세요.
- 클러스터를 생성한 IAM 보안 주체 또는 다른 보안 주체가 Amazon EKS 콘솔에서 Kubernetes 리소스를 보도록 하려면 엔터티에 필요한 권한을 부여합니다.
- 노드와 IAM 보안 주체가 VPC 내에서 클러스터에 액세스하도록 하려면 클러스터에 대한 프라이빗 엔드포인트를 활성화합니다. 퍼블릭 엔드포인트는 기본적으로 활성화되어 있습니다. 원하는 경우 프라이빗 엔드포인트를 활성화한 후 퍼블릭 엔드포인트를 비활성화할 수 있습니다. 자세한 내용은 Amazon EKS 클러스터 엔드포인트 액세스 제어 단원을 참조하십시오.
- 클러스터에서 보안 암호 암호화 활성화
- 클러스터에서 로깅 구성
- 클러스터에 노드 추가

클러스터 인사이트

Amazon EKS 클러스터 인사이트는 Amazon EKS 및 Kubernetes 모범 사례를 따르는 데 도움이 되는 권장 사항을 제공합니다. 모든 Amazon EKS 클러스터는 Amazon EKS에서 선별한 인사이트 목록을 대상으로 자동 반복 검사를 거칩니다. 이러한 인사이트 검사는 Amazon EKS에서 모두 관리되며, 결과를 처리하는 방법에 대한 권장 사항을 제공합니다.

클러스터 인사이트 권장 사용법:

- 클러스터 Kubernetes 버전을 업데이트하기 전에 EKS 콘솔에서 클러스터 인사이트를 확인합니다.
- 클러스터에서 문제가 식별된 경우 문제를 검토하고 적절하게 수정하십시오. 문제에는 Amazon EKS 및 Kubernetes에 대한 링크가 포함됩니다.
- 문제를 해결한 후에는 클러스터 인사이트가 새로 고쳐질 때까지 기다리십시오. 모든 문제가 해결되었으면 클러스터를 업데이트하십시오.

현재 Amazon EKS는 Kubernetes 버전 업그레이드 준비와 관련된 인사이트만 반환합니다.

업그레이드 인사이트는 Kubernetes 클러스터 업그레이드에 영향을 미칠 수 있는 잠재적 문제를 식별합니다. 이를 통해 관리자가 업그레이드를 준비하는 데 드는 노력을 최소화하고 최신 Kubernetes 버전에서 애플리케이션의 신뢰성을 높일 수 있습니다. Amazon EKS는 문제에 영향을 줄 수 있는 가능한 Kubernetes 버전 업그레이드 목록을 대상으로 클러스터를 자동으로 스캔합니다. Amazon EKS는 각 Kubernetes 버전 릴리스의 변경 사항 검토를 기반으로 인사이트 검사 목록을 자주 업데이트합니다.

Amazon EKS 업그레이드 인사이트는 새 버전에 대한 테스트 및 확인 프로세스를 가속화합니다. 또한 클러스터 관리자 및 애플리케이션 개발자는 우려 사항을 강조하고 문제 해결 조언을 제공하여 최신 Kubernetes 기능을 활용할 수 있습니다. 수행된 인사이트 검사 목록과 Amazon EKS에서 식별한 관련 문제를 확인하려면 Amazon EKS `ListInsights` API 작업을 직접 호출하거나 Amazon EKS 콘솔을 확인합니다.

클러스터 인사이트는 주기적으로 업데이트됩니다. 수동으로 클러스터 인사이트를 새로 고칠 수 없습니다. 클러스터 문제를 해결한 경우 클러스터 인사이트가 업데이트되는 데 다소 시간이 걸립니다. 문제가 해결되었는지 확인하려면 변경 사항이 배포된 시간을 클러스터 인사이트의 "마지막 새로 고침 시간"과 비교하십시오.

클러스터 인사이트 보기(콘솔)

Amazon EKS 클러스터의 인사이트를 확인하는 방법:

a. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.

b. 클러스터 목록에서 인사이트를 확인하려는 Amazon EKS 클러스터 이름을 선택합니다.

c. 업그레이드 인사이트 탭을 선택합니다.

d. 업그레이드 인사이트 페이지에는 다음 필드가 표시됩니다.

- 이름 - 클러스터를 대상으로 Amazon EKS에서 수행한 검사입니다.
- 인사이트 상태 - '오류' 상태의 인사이트는 일반적으로 영향을 받는 Kubernetes 버전이 현재 클러스터 버전의 N+1임을 의미하고, '경고' 상태는 인사이트가 향후 Kubernetes 버전 N+2 이상에 적용됨을 의미합니다. '통과' 상태의 인사이트는 Amazon EKS가 클러스터에서 이 인사이트 검사와 관련된 문제를 발견하지 못했음을 의미합니다. '알 수 없음' 상태의 인사이트는 Amazon EKS가 클러스터가 이 인사이트 검사의 영향을 받는지 여부를 판단할 수 없음을 의미합니다.
- 버전 - 인사이트가 잠재적 문제가 있는지 검사한 Kubernetes 버전입니다.
- 마지막 새로 고침 시간(UTC-5:00) - 이 클러스터에서 인사이트 상태를 마지막으로 새로 고친 시간입니다.
- 마지막 전환 시간(UTC-5:00) - 이 인사이트의 상태가 마지막으로 전환된 시간입니다.

- 설명 - 알림 및 문제 해결을 위한 권장 작업을 포함하는 인사이트 검사 정보입니다.

클러스터 인사이트 보기(AWS CLI)

Amazon EKS 클러스터의 인사이트를 확인하는 방법:

a. 인사이트를 확인하려는 클러스터를 결정합니다. 다음 명령은 지정된 클러스터에 대한 인사이트 정보를 나열합니다. 필요에 따라 명령을 다음과 같이 수정한 다음에 수정한 명령을 실행합니다.

- *region-code*를 AWS 리전의 코드로 바꿉니다.
- *my-cluster*를 클러스터 이름으로 바꿉니다.

```
aws eks list-insights --region region-code --cluster-name my-cluster
```

예제 출력은 다음과 같습니다.

```
{
    "insights": [
        {
            "category": "UPGRADE_READINESS",
            "name": "Deprecated APIs removed in Kubernetes v1.29",
            "insightStatus": {
                "status": "PASSING",
                "reason": "No deprecated API usage detected within the last 30 days."
            },
            "kubernetesVersion": "1.29",
            "lastTransitionTime": 1698774710.0,
            "lastRefreshTime": 1700157422.0,
            "id": "123e4567-e89b-42d3-a456-579642341238",
            "description": "Checks for usage of deprecated APIs that are scheduled
 for removal in Kubernetes v1.29. Upgrading your cluster before migrating to the
 updated APIs supported by v1.29 could cause application impact."
        }
    ]
}
```

b. 인사이트에 대한 설명 정보를 보려면 다음 명령을 실행합니다. 필요에 따라 명령을 다음과 같이 수정한 다음에 수정한 명령을 실행합니다.

- *region-code*를 AWS 리전의 코드로 바꿉니다.

- *123e4567-e89b-42d3-a456-579642341238*을 클러스터 인사이트 목록에서 검색한 인사이트 ID로 바꿉니다.
- *my-cluster*를 클러스터 이름으로 바꿉니다.

```
aws eks describe-insight --region region-code --id 123e4567-e89b-42d3-
a456-579642341238 --cluster-name my-cluster
```

예제 출력은 다음과 같습니다.

```
{
    "insight": {
        "category": "UPGRADE_READINESS",
        "additionalInfo": {
            "EKS update cluster documentation": "https://docs.aws.amazon.com/eks/
latest/userguide/update-cluster.html",
            "Kubernetes v1.29 deprecation guide": "https://kubernetes.io/docs/
reference/using-api/deprecation-guide/#v1-29"
        },
        "name": "Deprecated APIs removed in Kubernetes v1.29",
        "insightStatus": {
            "status": "PASSING",
            "reason": "No deprecated API usage detected within the last 30 days."
        },
        "kubernetesVersion": "1.29",
        "recommendation": "Update manifests and API clients to use newer Kubernetes
 APIs if applicable before upgrading to Kubernetes v1.29.",
        "lastTransitionTime": 1698774710.0,
        "lastRefreshTime": 1700157422.0,
        "categorySpecificSummary": {
            "deprecationDetails": [
                {
                    "usage": "/apis/flowcontrol.apiserver.k8s.io/v1beta2/
flowschemas",
                    "replacedWith": "/apis/flowcontrol.apiserver.k8s.io/v1beta3/
flowschemas",
                    "stopServingVersion": "1.29",
                    "clientStats": [],
                    "startServingReplacementVersion": "1.26"
                },
                {
                    "usage": "/apis/flowcontrol.apiserver.k8s.io/v1beta2/
prioritylevelconfigurations",
```

```
                    "replacedWith": "/apis/flowcontrol.apiserver.k8s.io/v1beta3/
prioritylevelconfigurations",
                    "stopServingVersion": "1.29",
                    "clientStats": [],
                    "startServingReplacementVersion": "1.26"
                }
            ]
        },
        "id": "f6a11fe4-77f7-48c6-8326-9a13f022ecb3",
        "resources": [],
        "description": "Checks for usage of deprecated APIs that are scheduled for
 removal in Kubernetes v1.29. Upgrading your cluster before migrating to the updated
 APIs supported by v1.29 could cause application impact."
    }
}
```


Amazon EKS 클러스터 Kubernetes 버전 업데이트

Amazon EKS에서 새 Kubernetes 버전을 사용할 수 있는 경우 Amazon EKS 클러스터를 최신 버전으로 업데이트할 수 있습니다.

> **Important**
>
> 클러스터를 업그레이드한 후에는 이전 버전으로 다운그레이드할 수 없습니다. 새 Kubernetes 버전으로 업데이트하기 전에 Amazon EKS Kubernetes 버전의 정보 및 이 주제의 업데이트 단계를 검토하는 것이 좋습니다.

새로운 Kubernetes 버전에는 때로는 큰 변화가 도입되기도 합니다. 따라서 프로덕션 클러스터를 업데이트하기 전에 새 Kubernetes 버전에 대해 애플리케이션의 동작을 테스트하는 것이 좋습니다. 새 Kubernetes 버전으로 이동하기 전에 애플리케이션 동작을 테스트하기 위한 지속적인 통합 워크플로를 구축하여 이 작업을 수행할 수 있습니다.

업데이트 프로세스는 Amazon EKS에서 업데이트된 Kubernetes 버전으로 새 API 서버 노드를 시작해 기존 노드를 대체합니다. Amazon EKS는 이러한 새 노드에서 네트워크 트래픽의 표준 인프라 및 준비 상태 검사를 수행하여 예상대로 작동하고 있는지 확인합니다. 하지만 클러스터 업그레이드를 시작한 후에는 일시 중지하거나 중지할 수 없습니다. 이러한 검사 중 하나라도 실패하면 Amazon EKS는 인프라 배포를 되돌리고, 클러스터는 이전 Kubernetes 버전으로 남아 있게 됩니다. 실행 중인 애플리케이션은 영향을 받지 않으며, 클러스터가 불확정적 또는 복구 불가능 상태로 남아 있는 일은 없습니다.

Amazon EKS는 모든 관리형 클러스터를 정기적으로 백업하고, 필요할 경우 클러스터를 복구하는 메커니즘이 있습니다. Kubernetes 인프라 관리 프로세스를 지속적으로 평가 및 개선하고 있습니다.

클러스터를 업데이트하려면 Amazon EKS에서 클러스터를 생성할 때 지정된 서브넷의 사용 가능한 IP 주소 최대 5개가 필요합니다. Amazon EKS에서는 지정한 서브넷에 새 클러스터 탄력적 네트워크 인터페이스(네트워크 인터페이스)를 생성합니다. 네트워크 인터페이스는 기존 네트워크 인터페이스가 있는 다른 서브넷에 생성될 수 있으므로 보안 그룹 규칙에서 클러스터를 생성할 때 지정한 모든 서브넷에 대해 필수 클러스터 통신을 허용해야 합니다. 클러스터를 생성할 때 지정한 서브넷이 없거나, 해당 서브넷에 사용 가능한 IP 주소가 충분하지 않거나 필요한 클러스터 통신을 허용하는 보안 그룹 규칙이 없는 경우 업데이트가 실패할 수 있습니다.

> **Note**
>
> Amazon EKS는 클러스터의 API 서버 엔드포인트에 항상 액세스할 수 있도록 가용성 높은 Kubernetes 컨트롤 플레인을 제공하고 업데이트 작업 중에 API 서버 인스턴스의 롤링 업데이트를 수행합니다. Kubernetes API 서버 엔드포인트를 지원하는 API 서버 인스턴스의 IP 주소 변경을 고려하려면 API 서버 클라이언트가 재연결을 효과적으로 관리하도록 해야 합니다. `kubectl`의 최신 버전과 공식적으로 지원되는 Kubernetes 고객 라이브러리, 재연결 프로세스는 이 재연결 프로세스를 투명하게 수행합니다.

Amazon EKS 클러스터의 Kubernetes 버전 업데이트

클러스터의 Kubernetes 버전을 업데이트하려면 다음을 수행합니다.

1. 클러스터 컨트롤 플레인의 Kubernetes 버전과 사용자 노드의 Kubernetes 버전을 비교합니다.
   - 클러스터 컨트롤 플레인의 Kubernetes 버전을 가져옵니다.

     ```
     kubectl version
     ```

   - 노드의 Kubernetes 버전을 가져옵니다. 이 명령은 자체 관리형 및 관리형 Amazon EC2 및 Fargate 노드를 모두 반환합니다. 각 Fargate Pod는 자체 노드로 나열됩니다.

     ```
     kubectl get nodes
     ```

   컨트롤 플레인을 새 Kubernetes 버전으로 업데이트하기 전에 클러스터에 있는 관리형 노드 및 Fargate 노드 모두의 Kubernetes 마이너 버전이 컨트롤 플레인의 현재 버전과 동일한지 확인합니

다. 예를 들어 컨트롤 플레인에서 버전 1.28을 실행 중이고 노드 중 하나에서 버전 1.27을 실행 중인 경우에는 컨트롤 플레인을 1.29로 업데이트하기 전에 노드를 버전 1.28로 업데이트해야 합니다. 또한 제어 플레인을 업데이트하기 전에 자체 관리형 노드를 컨트롤 평면과 동일한 버전으로 업데이트하는 것이 좋습니다. 자세한 내용은 관리형 노드 그룹 업데이트 및 자체 관리형 노드 업데이트 단원을 참조하세요. 컨트롤 플레인 버전보다 낮은 마이너 버전이 적용된 Fargate 노드가 있는 경우 먼저 이 노드로 대표되는 Pod을(를) 삭제합니다. 그런 다음 컨트롤 플레인을 업데이트합니다. 나머지 Pods는 모두 다시 배포한 후 새 버전으로 업데이트됩니다.

2. 원래 클러스터를 배포한 Kubernetes 버전이 Kubernetes 1.25 이상인 경우 이 단계를 건너뜁니다.

   기본적으로 Pod 보안 정책 승인 컨트롤러는 Amazon EKS 클러스터에 사용 설정되어 있습니다. 클러스터를 업데이트하기 전에 적절한 Pod 보안 정책이 있는지 확인하세요. 이는 잠재적인 보안 문제를 방지하기 위함입니다. **kubectl get psp eks.privileged** 명령을 사용하여 기본 정책을 확인할 수 있습니다.

   ```
   kubectl get psp eks.privileged
   ```

   다음 오류가 표시되는 경우 계속하기 전에 Amazon EKS 기본 Pod 보안 정책 단원을 참조하십시오.

   ```
   Error from server (NotFound): podsecuritypolicies.extensions "eks.privileged" not
    found
   ```

3. 원래 클러스터를 배포한 Kubernetes 버전이 Kubernetes 1.18 이상인 경우 이 단계를 건너뜁니다.

   중단된 용어를 CoreDNS 매니페스트에서 제거해야 할 수 있습니다.

   a. CoreDNS 매니페스트에 upstream이라는 단어만 있는 줄이 있는지 확인합니다.

      ```
      kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |
       grep upstream
      ```

      결과가 반환되지 않으면 매니페스트에 해당 줄이 없는 것입니다. 이 경우 다음 단계로 건너뜁니다. upstream이라는 단어가 반환되면 줄을 제거합니다.

   b. configmap 파일에서 upstream이라는 단어만 있는 파일의 맨 위 근처의 줄을 제거합니다. 파일에서 다른 것을 변경하지 마세요. 줄을 제거한 후 변경 사항을 저장하세요.

      ```
      kubectl edit configmap coredns -n kube-system -o yaml
      ```

4. `eksctl`, AWS Management Console 또는 AWS CLI를 사용하여 클러스터를 업데이트합니다.

> **Important**
>
> - 버전 `1.23`으로 업데이트하고 클러스터에서 Amazon EBS 볼륨을 사용하는 경우 워크로드 중단을 방지하기 위해 클러스터를 버전 `1.23`으로 업데이트하기 전에 클러스터에 Amazon EBS CSI 드라이버를 설치해야 합니다. 자세한 내용은 Kubernetes 1.23 및 Amazon EBS CSI 드라이버 단원을 참조하세요.
> - Kubernetes `1.24` 이상에서는 `containerd`를 기본 컨테이너 런타임으로 사용한다. `containerd` 런타임으로 전환하고 Container Insights에 대해 Fluentd를 이미 구성한 경우 클러스터를 업데이트하기 전에 Fluentd를 Fluent Bit로 마이그레이션해야 합니다. Fluentd 구문 분석기는 JSON 형식의 로그 메시지만 구문 분석하도록 구성됩니다. `dockerd`와 달리 `containerd` 컨테이너 런타임에는 JSON 형식이 아닌 로그 메시지가 있습니다. Fluent Bit로 마이그레이션하지 않으면 구성된 Fluentd's 구문 분석기 중 일부가 Fluentd 컨테이너 내에서 엄청난 양의 오류를 생성합니다. 마이그레이션에 대한 자세한 내용은 Set up Fluent Bit as a DaemonSet to send logs to CloudWatch Logs를 참조하세요.
> - Amazon EKS는 가용성 높은 제어 영역을 실행하므로 한 번에 하나의 마이너 버전만 업데이트할 수 있습니다. 이 요구 사항에 대한 자세한 내용을 알아보려면 Kubernetes 버전 및 버전 차이 지원 정책을 참조하세요. 현재 클러스터 버전이 `1.27` 버전이고 `1.29` 버전으로 업데이트하려는 경우를 예로 들어보겠습니다. 먼저 `1.27` 버전 클러스터를 `1.28` 버전으로 업데이트한 다음에 `1.28` 버전 클러스터를 `1.29` 버전으로 업데이트해야 합니다.
> - 노드의 `kubelet` 및 Kubernetes `kube-apiserver` 사이에서 버전 차이를 검토합니다.
>   - Kubernetes 버전 `1.28`부터 `kubelet` 버전은 `kube-apiserver`보다 오래된 최대 3개의 마이너 버전에 해당할 수 있습니다. Kubernetes upstream version skew policy를 참조하세요.
>   - Fargate 노드 및 관리형 노드의 `kubelet`가 Kubernetes 버전 `1.25` 이상인 경우 `kubelet` 버전으로 업데이트하지 않고 최대 3개 버전까지 클러스터를 업데이트할 수 있습니다. 예를 들어 `kubelet` 버전이 `1.25`인 경우 `kubelet` 버전은 `1.25`로 유지하면서 Amazon EKS 클러스터 버전을 `1.25`에서 `1.26`, `1.27`, `1.28`로 업데이트할 수 있습니다.
>   - Fargate 노드 및 관리형 노드의 Kubernetes가 `kubelet` 버전 `1.24` 이상인 경우 `kube-apiserver` 이상의 최대 2개 마이너 버전만 가능합니다. 즉, `kubelet`이 버전 `1.24` 이상인 경우 클러스터를 향후 최대 2개 버전만 업데이트할 수 있습니다. 예를

들어 kubelet 버전이 1.21인 경우 kubelet 버전은 1.21로 유지하면서 Amazon EKS 클러스터 버전을 1.21에서 1.22, 1.23으로 업데이트할 수 있지만, 1.24로는 업데이트할 수 없습니다.

- 업데이트를 시작하기 전에 모범 사례는 노드의 kubelet 버전이 컨트롤 플레인과 동일한지 Kubernetes 버전인지 확인하는 것입니다.
- 클러스터가 1.8.0 이전 버전의 Amazon VPC CNI plugin for Kubernetes로 구성된 경우에는 클러스터를 업데이트하기 전에 플러그인을 최신 버전으로 업데이트하는 것이 좋습니다. 플러그인을 업데이트하려면 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업을 참조하십시오.
- 클러스터를 버전 1.25 이상으로 업데이트하고 클러스터에 AWS Load Balancer Controller를 배포한 경우, 클러스터 버전을 1.25로 업데이트하기 전에 컨트롤러를 버전 2.4.7 이상으로 업데이트하십시오. 자세한 내용은 Kubernetes1.25 릴리스 정보를 참조하세요.

eksctl

이 절차에는 eksctl 버전 0.175.0 이상이 필요합니다. 버전은 다음 명령을 통해 확인할 수 있습니다.

```
eksctl version
```

eksctl 설치 또는 업데이트에 대한 지침은 eksctl 설명서에서 Installation을 참조하세요.

Amazon EKS 컨트롤 플레인의 Kubernetes 버전을 업데이트합니다. *my-cluster*을 클러스터 이름으로 교체합니다. *1.29*를 클러스터를 업데이트하려는 Amazon EKS에서 지원하는 버전 번호로 바꿉니다. 지원되는 버전 번호 목록은 Amazon EKS Kubernetes 버전 섹션을 참조하세요.

```
eksctl upgrade cluster --name my-cluster --version 1.29 --approve
```

업데이트를 완료하는 데 몇 분 정도 걸립니다.

AWS Management Console

a. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.

b. 업데이트할 Amazon EKS 클러스터의 이름을 선택한 다음 클러스터 버전 업데이트(Update cluster version)를 선택합니다.

c. Kubernetes 버전에서 클러스터를 업데이트할 버전을 선택하고 업데이트(Update)를 선택합니다.

d. 클러스터 이름(Cluster name)에 클러스터 이름을 입력하고 확인(Confirm)을 선택합니다.

업데이트를 완료하는 데 몇 분 정도 걸립니다.

AWS CLI

a. 다음 AWS CLI 명령을 사용하여 Amazon EKS 클러스터를 업데이트합니다. *example values*을 사용자의 값으로 교체합니다. *1.29*를 클러스터를 업데이트하려는 Amazon EKS에서 지원하는 버전 번호로 바꿉니다. 지원되는 버전 번호 목록은 Amazon EKS Kubernetes 버전 부분을 참조하세요.

```
aws eks update-cluster-version --region region-code --name my-cluster --
kubernetes-version 1.29
```

예제 출력은 다음과 같습니다.

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "InProgress",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.29"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
[...]
        "errors": []
    }
}
```

b. 다음 명령을 사용하여 클러스터의 업데이트 상태를 모니터링할 수 있습니다. 이전 명령에서 반환된 클러스터 이름과 업데이트 ID를 사용합니다. `Successful` 상태가 표시되면 업데이트가 완료됩니다. 업데이트를 완료하는 데 몇 분 정도 걸립니다.

```
aws eks describe-update --region region-code --name my-cluster --update-
id b5f0ba18-9a87-4450-b5a0-825e6e84496f
```

예제 출력은 다음과 같습니다.

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "Successful",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.29"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
[...]
        "errors": []
    }
}
```

5. 클러스터 업데이트가 완료된 후 업데이트한 클러스터와 동일한 Kubernetes 마이너 버전으로 노드를 업데이트합니다. 자세한 내용은 자체 관리형 노드 업데이트 및 관리형 노드 그룹 업데이트 섹션을 참조하세요. Fargate에서 시작된 새 Pods에는 클러스터 버전과 일치하는 `kubelet` 버전이 있습니다. 기존 Fargate Pods는 변경되지 않습니다.
6. (선택 사항) 클러스터를 업데이트하기 전에 Kubernetes Cluster Autoscaler를 클러스터에 배포한 경우 업그레이드한 Kubernetes 메이저 및 마이너 버전과 일치하는 최신 버전으로 Cluster Autoscaler를 업데이트합니다.

   a. 웹 브라우저에서 Cluster Autoscaler 릴리스 페이지를 열고 클러스터의 Kubernetes 메이저 및 마이너 버전과 일치하는 최신 Cluster Autoscaler 버전을 검색합니다. 예를 들어 클러스터의

Kubernetes 버전이 `1.29`이라면 `1.29`으로 시작하는 Cluster Autoscaler 릴리스를 검색합니다. 다음 단계에서 사용할 해당 릴리스의 의미 체계 버전 번호(예: `1.29.n`)를 기록합니다.

b. 다음 명령을 사용하여 Cluster Autoscaler 이미지 태그를 이전 단계에서 적어 둔 버전으로 설정합니다. 필요한 경우 *`1.29.n`*을 고유 값으로 바꿉니다.

```
kubectl -n kube-system set image deployment.apps/cluster-autoscaler cluster-autoscaler=registry.k8s.io/autoscaling/cluster-autoscaler:v1.29.n
```

7. (GPU 노드가 있는 클러스터만 해당) 클러스터에 GPU 지원이 포함된 노드 그룹이 있는 경우(예: `p3.2xlarge`), 클러스터에서 Kubernetes용 NVIDIA 디바이스 플러그인 DaemonSet을(를) 업데이트해야 합니다. 다음 명령을 실행하기 전에 *`vX.X.X`*을(를) 원하는 NVIDIA/k8s-device-plugin 버전으로 교체합니다.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/nvidia-device-plugin.yml
```

8. Amazon VPC CNI plugin for Kubernetes, CoreDNS 및 `kube-proxy` 추가 기능을 업데이트합니다. 서비스 계정 토큰에 나열된 최소 버전으로 추가 기능을 업데이트하는 것이 좋습니다.

   - Amazon EKS 추가 기능을 사용하는 경우 Amazon EKS 콘솔에서 Clusters(클러스터)를 선택한 다음 왼쪽 검색 창에서 업데이트한 클러스터의 이름을 선택합니다. 콘솔에 알림이 표시됩니다. 사용 가능한 업데이트가 있는 추가 기능마다 새 버전을 사용할 수 있다는 알림이 표시됩니다. 추가 기능을 업데이트하려면 추가 기능(Add-ons) 탭을 선택합니다. 사용 가능한 업데이트가 있는 추가 기능의 상자 중 하나에서 [지금 업데이트(Update now)]를 선택하고 사용 가능한 버전을 선택한 다음 [업데이트(Update)]를 선택합니다.
   - AWS CLI 또는 `eksctl`을 사용하여 추가 기능을 업데이트할 수도 있습니다. 자세한 내용은 추가 기능 업데이트 단원을 참조하십시오.

9. 필요하면 `kubectl`의 버전을 업데이트합니다. Amazon EKS 클러스터 제어 영역과 마이너 버전이 하나 다른 `kubectl` 버전을 사용해야 합니다. 예를 들어 `1.28` `kubectl` 클라이언트는 Kubernetes `1.27`, `1.28`, `1.29` 클러스터와 함께 작동합니다. 현재 설치된 버전은 다음과 같은 명령으로 확인할 수 있습니다.

```
kubectl version --client
```

Amazon EKS 클러스터 삭제

Amazon EKS 클러스터 사용을 완료한 경우 불필요한 비용이 발생하지 않도록 클러스터와 연결된 리소스를 삭제해야 합니다.

연결된 클러스터를 제거하려면 클러스터 등록 취소 섹션을 참조하세요.

Important

- 클러스터의 활성 서비스가 로드 밸런서와 연결된 경우 로드 밸런서가 적절하게 삭제되도록 클러스터 삭제 전에 해당 서비스를 삭제해야 합니다. 그렇지 않으면 VPC에 분리된 리소스가 발생하여 VPC를 삭제할 수 없게 됩니다.
- 클러스터 생성자가 제거되어 오류가 발생하는 경우 문제를 해결하려면 이 문서를 참조하세요.
- Amazon Managed Service for Prometheus 리소스는 클러스터 수명 주기를 벗어나며 클러스터에 독립적으로 유지 관리되어야 합니다. 클러스터를 삭제할 때 해당 스크레이퍼도 모두 삭제하여 관련 비용을 줄여야 합니다. 자세한 내용은 Amazon Managed Service for Prometheus 사용자 가이드의 스크레이퍼 찾기 및 삭제를 참조하세요.

`eksctl`, AWS Management Console 또는 AWS CLI를 사용하여 클러스터를 삭제할 수 있습니다.

eksctl

`eksctl`로 Amazon EKS 클러스터 및 노드를 삭제하려면

이 절차에는 `eksctl` 버전 `0.175.0` 이상이 필요합니다. 버전은 다음 명령을 통해 확인할 수 있습니다.

```
eksctl version
```

`eksctl` 설치 또는 업데이트에 대한 지침은 `eksctl` 설명서의 Installation을 참조하세요.

1. 클러스터에서 실행 중인 모든 서비스를 나열합니다.

```
kubectl get svc --all-namespaces
```

2. EXTERNAL-IP 값과 연결된 모든 서비스를 삭제합니다. 이러한 서비스는 Elastic Load Balancing 로드 밸런서에 의해 설정되고 Kubernetes에서 이를 삭제하여 로드 밸런서 및 연결된 리소스가 적절하게 릴리스되어야 합니다.

```
kubectl delete svc service-name
```

3. 다음 명령으로 클러스터 및 연결된 노드를 삭제하되, *prod*를 클러스터 이름으로 변경합니다.

```
eksctl delete cluster --name prod
```

출력:

```
[#]  using region region-code
[#]  deleting EKS cluster "prod"
[#]  will delete stack "eksctl-prod-nodegroup-standard-nodes"
[#]  waiting for stack "eksctl-prod-nodegroup-standard-nodes" to get deleted
[#]  will delete stack "eksctl-prod-cluster"
[#]  the following EKS cluster resource(s) for "prod" will be deleted: cluster.
 If in doubt, check CloudFormation console
```

AWS Management Console

AWS Management Console로 Amazon EKS 클러스터를 삭제하려면

1. 클러스터에서 실행 중인 모든 서비스를 나열합니다.

```
kubectl get svc --all-namespaces
```

2. EXTERNAL-IP 값과 연결된 모든 서비스를 삭제합니다. 이러한 서비스는 Elastic Load Balancing 로드 밸런서에 의해 설정되고 Kubernetes에서 이를 삭제하여 로드 밸런서 및 연결된 리소스가 적절하게 릴리스되어야 합니다.

```
kubectl delete svc service-name
```

3. 모든 노드 그룹 및 Fargate 프로필을 삭제합니다.

   a. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.

   b. 왼쪽 탐색 창에서 Amazon EKS 클러스터(Clusters)를 선택한 다음 탭으로 구분된 클러스터 목록에서 삭제할 클러스터의 이름을 선택합니다.

c. 컴퓨팅(Compute)탭을 선택하고 삭제할 노드 그룹을 선택합니다. 삭제(Delete)를 선택하고 노드 그룹의 이름을 입력한 다음 삭제(Delete)를 선택합니다. 클러스터의 모든 노드 그룹을 삭제합니다.

> **Note**
>
> 관리형 노드 그룹만 나열됩니다.

d. 삭제할 Fargate 프로필(Fargate Profile)에 이어 삭제(Delete)를 선택하고 프로필의 이름을 입력한 다음 삭제(Delete)를 선택합니다. 클러스터의 모든 Fargate 프로필을 삭제합니다.

4. 모든 자체 관리형 노드 AWS CloudFormation 스택을 삭제합니다.

a. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.

b. 삭제할 노드 스택을 선택하고 삭제(Delete)를 선택합니다.

c. 스택 삭제(Delete stack) 확인 대화 상자에서 스택 삭제(Delete stack)를 선택합니다. 클러스터의 모든 자체 관리형 노드 스택을 삭제합니다.

5. 클러스터를 삭제합니다.

a. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.

b. 삭제할 클러스터를 선택하고 삭제(Delete)를 선택합니다.

c. 클러스터 삭제 확인 화면에서 삭제(Delete)를 선택합니다.

6. (선택 사항) VPC AWS CloudFormation 스택을 삭제합니다.

a. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.

b. 삭제할 VPC 스택을 선택하고 삭제(Delete)를 선택합니다.

c. 스택 삭제(Delete stack) 확인 대화 상자에서 스택 삭제(Delete stack)를 선택합니다.

AWS CLI

AWS CLI로 Amazon EKS 클러스터를 삭제하려면

1. 클러스터에서 실행 중인 모든 서비스를 나열합니다.

```
kubectl get svc --all-namespaces
```

2. EXTERNAL-IP 값과 연결된 모든 서비스를 삭제합니다. 이러한 서비스는 Elastic Load Balancing 로드 밸런서에 의해 설정되고 Kubernetes에서 이를 삭제하여 로드 밸런서 및 연결된 리소스가 적절하게 릴리스되어야 합니다.

```
kubectl delete svc service-name
```

3. 모든 노드 그룹 및 Fargate 프로필을 삭제합니다.

   a. 다음 명령을 사용하여 클러스터의 노드 그룹을 나열합니다.

```
aws eks list-nodegroups --cluster-name my-cluster
```

> **Note**
>
> 관리형 노드 그룹만 나열됩니다.

   b. 다음 명령을 사용하여 각 노드 그룹을 삭제합니다. 클러스터의 모든 노드 그룹을 삭제합니다.

```
aws eks delete-nodegroup --nodegroup-name my-nodegroup --cluster-name my-cluster
```

   c. 다음 명령을 사용하여 클러스터에 Fargate 프로필을 나열합니다.

```
aws eks list-fargate-profiles --cluster-name my-cluster
```

   d. 다음 명령을 사용하여 각 Fargate 프로필을 삭제합니다. 클러스터의 모든 Fargate 프로필을 삭제합니다.

```
aws eks delete-fargate-profile --fargate-profile-name my-fargate-profile --cluster-name my-cluster
```

4. 모든 자체 관리형 노드 AWS CloudFormation 스택을 삭제합니다.

   a. 다음 명령으로 사용 가능한 AWS CloudFormation 스택을 나열합니다. 결과 출력에서 노드 템플릿의 이름을 찾습니다.

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

b. 다음 명령으로 각 노드 스택을 삭제하되, *node-stack*을 노드 스택 이름으로 변경합니다. 클러스터의 모든 자체 관리형 노드 스택을 삭제합니다.

```
aws cloudformation delete-stack --stack-name node-stack
```

5. 다음 명령으로 클러스터를 삭제하되, *my-cluster*를 사용할 클러스터 이름으로 변경합니다.

```
aws eks delete-cluster --name my-cluster
```

6. (선택 사항) VPC AWS CloudFormation 스택을 삭제합니다.

a. 다음 명령으로 사용 가능한 AWS CloudFormation 스택을 나열합니다. 결과 출력에서 VPC 템플릿의 이름을 찾습니다.

```
aws cloudformation list-stacks --query "StackSummaries[].StackName"
```

b. 다음 명령으로 VPC 스택을 삭제하되, *my-vpc-stack*을 사용할 VPC 스택의 이름으로 변경합니다.

```
aws cloudformation delete-stack --stack-name my-vpc-stack
```

Amazon EKS 클러스터 엔드포인트 액세스 제어

이 주제에서는 Amazon EKS 클러스터의 Kubernetes API 서버 엔드포인트에 대한 프라이빗 액세스를 활성화하거나 인터넷에서 퍼블릭 액세스를 완전히 비활성화하는 방법을 설명합니다.

새 클러스터를 생성할 때 Amazon EKS에서는 클러스터와 통신하는 데 사용하는 관리형 Kubernetes API 서버에 대한 엔드포인트를 생성합니다(`kubectl`과 같은 Kubernetes 관리 도구 사용). 기본적으로 이 API 서버 엔드포인트는 인터넷에 공개되어 있으며, API 서버에 대한 액세스는 AWS Identity and Access Management(IAM) 및 기본 Kubernetes 역할 기반 액세스 제어(RBAC)의 조합을 통해 보호됩니다.

노드와 API 서버 간의 모든 통신이 VPC 내에 유지되도록 Kubernetes API 서버에 대한 프라이빗 액세스를 활성화할 수 있습니다. 인터넷에서 API 서버로 액세스하는 IP 주소를 제한하거나 API 서버로의 인터넷 액세스를 완전히 비활성화할 수 있습니다.

> **Note**
>
> 이 엔드포인트는 Kubernetes API 서버이며 AWS API와 통신하기 위한 기존 AWS PrivateLink 엔드포인트가 아니므로 Amazon VPC 콘솔에서 엔드포인트로 표시되지 않습니다.

클러스터에 대해 엔드포인트 프라이빗 액세스를 활성화하면 Amazon EKS에서는 사용자 대신 Route 53 프라이빗 호스팅 영역을 생성하고 클러스터의 VPC에 연결합니다. 이 프라이빗 호스팅 영역은 Amazon EKS에서 관리하며, 사용자 계정의 Route 53 리소스에는 표시되지 않습니다. 프라이빗 호스팅 영역이 API 서버로 트래픽을 올바로 라우팅하려면 VPC에서 `enableDnsHostnames` 및 `enableDnsSupport`가 `true`로 설정되어야 하고, VPC에 설정된 DHCP 옵션이 도메인 이름 서버 목록에 `AmazonProvidedDNS`를 포함해야 합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC에 대한 DNS 지원 업데이트를 참조하세요.

새 클러스터를 생성할 때 API 서버 엔드포인트 액세스 요구 사항을 정의할 수 있으며, 언제든지 클러스터에 대한 API 서버 엔드포인트 액세스를 업데이트할 수 있습니다.

클러스터 엔드포인트 액세스 수정

이 단원의 절차를 통해 기존 클러스터에 대한 엔드포인트 액세스를 수정합니다. 다음 표에는 지원되는 API 서버 엔드포인트 액세스 조합과 연결된 동작이 나와 있습니다.

API 서버 엔드포인트 액세스 옵션

엔드포인트 퍼블릭 액세스	엔드포인트 프라이빗 액세스	동작
활성화됨	Disabled(비활성)	• 새 Amazon EKS 클러스터의 기본 동작입니다. • 클러스터의 VPC 내에서 비롯된 Kubernetes API 요청(예: 컨트롤 플레인 통신에 대한 노드)은 VPC에서 벗어나지만 Amazon의 네트워크에서는 벗어나지 않습니다. • 인터넷에서 클러스터 API 서버에 액세스할 수 있습니다. 선택적으로 퍼블릭 엔드포인트에 액세스할 수 있는 CIDR

엔드포인트 퍼블릭 액세스	엔드포인트 프라이빗 액세스	동작
		블록을 제한할 수 있습니다. 특정 CIDR 블록에 대한 액세스를 제한하는 경우, 프라이빗 엔드포인트를 활성화하거나 지정된 CIDR 블록에 노드와 Fargate Pods(사용하는 경우)가 퍼블릭 엔드포인트에 액세스하는 주소를 포함합니다
활성화됨	활성화됨	• 클러스터의 VPC 내 Kubernetes API 요청(예: 컨트롤 플레인 통신에 대한 노드)은 프라이빗 VPC 엔드포인트를 사용합니다. • 인터넷에서 클러스터 API 서버에 액세스할 수 있습니다. 선택적으로 퍼블릭 엔드포인트에 액세스할 수 있는 CIDR 블록을 제한할 수 있습니다.

엔드포인트 퍼블릭 액세스	엔드포인트 프라이빗 액세스	동작

Disabled(비활성)	활성화됨	• 클러스터 API 서버에 대한 모든 트래픽은 클러스터의 VPC 또는 연결된 네트워크 내에서 비롯되어야 합니다.
		• 인터넷에서 API 서버로 퍼블릭 액세스할 수 없습니다. 모든 `kubectl` 명령은 VPC 또는 연결된 네트워크 내에서 가져와야 합니다. 연결 옵션은 프라이빗 전용 API 서버 액세스 섹션을 참조하세요.
		• 클러스터의 API 서버 엔드포인트는 퍼블릭 DNS 서버에 의해 VPC의 프라이빗 IP 주소로 확인됩니다. 이전에는 VPC 내에서만 엔드포인트를 확인할 수 있었습니다.
		
		엔드포인트가 VPC 내부에서 기존 클러스터에 대한 프라이빗 IP 주소로 확인되지 않는 경우 다음을 수행할 수 있습니다.
		  • 퍼블릭 액세스를 활성화한 다음 다시 비활성화합니다. 클러스터에 대해 한 번만 수행하면 엔드포인트가 해당 시점부터 프라이빗 IP 주소로 확인됩니다.
		  • 클러스터를 업데이트합니다.

AWS Management Console 또는 AWS CLI를 사용하여 클러스터 API 서버 엔드포인트 액세스를 수정할 수 있습니다.

AWS Management Console

AWS Management Console을 사용하여 클러스터 API 서버 엔드포인트 액세스를 수정하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 클러스터 이름을 선택하여 클러스터 정보를 표시합니다.
3. 네트워킹(Networking) 탭을 선택하고 업데이트(Update)를 선택합니다.
4. 프라이빗 액세스에서 클러스터의 Kubernetes API 서버 엔드포인트에 대해 프라이빗 액세스를 활성화할지 아니면 비활성화할지를 선택합니다. 프라이빗 액세스를 활성화하면 클러스터의 VPC 내에서 비롯된 Kubernetes API 요청이 프라이빗 VPC 엔드포인트를 사용합니다. 퍼블릭 액세스를 비활성화하려면 프라이빗 액세스를 활성화해야 합니다.
5. 퍼블릭 액세스에서 클러스터의 Kubernetes API 서버 엔드포인트에 대해 퍼블릭 액세스를 활성화할지 아니면 비활성화할지를 선택합니다. 퍼블릭 액세스를 비활성화하면 클러스터의 Kubernetes API 서버가 클러스터의 VPC 내에서 비롯된 요청만 수신할 수 있습니다.
6. (선택 사항) 퍼블릭 액세스를 활성화한 경우, 인터넷에서 퍼블릭 엔드포인트와 통신할 수 있는 주소를 지정할 수 있습니다. Advanced Settings(고급 설정)를 선택합니다. *203.0.113.5/32*와 같은 CIDR 블록을 입력합니다. 블록에는 예약된 주소가 포함될 수 없습니다. Add Source(소스 추가)를 선택하여 추가 블록을 입력할 수 있습니다. 지정할 수 있는 최대 CIDR 블록 수가 있습니다. 자세한 내용은 Amazon EKS 서비스 할당량 단원을 참조하십시오. 블록을 지정하지 않은 경우 퍼블릭 API 서버 엔드포인트는 모든 (`0.0.0.0/0`) IP 주소에서 요청을 수신합니다. CIDR 블록을 사용하여 퍼블릭 엔드포인트에 대한 액세스를 제한하는 경우, 노드와 Fargate Pods(사용하는 경우)가 클러스터와 통신할 수 있도록 프라이빗 엔드포인트 액세스도 활성화하는 것이 좋습니다. 프라이빗 엔드포인트를 활성화하지 않은 경우, 퍼블릭 액세스 엔드포인트 CIDR 소스에는 VPC의 송신 소스가 포함되어야 합니다. 예를 들어 NAT 게이트웨이를 통해 인터넷과 통신하는 프라이빗 서브넷에 노드가 있는 경우, 퍼블릭 엔드포인트에서 허용하는 CIDR 블록의 일부로 NAT 게이트웨이의 아웃바운드 IP 주소를 추가해야 합니다.
7. Update(업데이트)를 선택하여 완료합니다.

AWS CLI

AWS CLI을 사용하여 클러스터 API 서버 엔드포인트 액세스를 수정하려면

AWS CLI 버전 `1.27.160` 이상을 사용하여 다음 단계를 완료합니다. `aws --version`을 사용하여 현재 버전을 확인할 수 있습니다. AWS CLI를 설치하거나 업그레이드하려면 AWS CLI 설치를 참조하세요.

1. 다음 AWS CLI 명령을 통해 클러스터 API 서버 엔드포인트 액세스를 업데이트합니다. 클러스터 이름과 원하는 엔드포인트 액세스 값을 대체합니다. `endpointPublicAccess=true`를 설정한 경우 선택적으로 단일 CIDR 블록 또는 `publicAccessCidrs`에 대한 쉼표로 구분된 CIDR 블록 목록을 입력할 수 있습니다. 블록에는 예약된 주소가 포함될 수 없습니다. CIDR 블록을 지정한 경우 퍼블릭 API 서버 엔드포인트는 나열된 블록에서만 요청을 수신합니다. 지정할 수 있는 최대 CIDR 블록 수가 있습니다. 자세한 내용은 Amazon EKS 서비스 할당량 단원을 참조하십시오. CIDR 블록을 사용하여 퍼블릭 엔드포인트에 대한 액세스를 제한하는 경우, 노드와 Fargate Pods(사용하는 경우)가 클러스터와 통신할 수 있도록 프라이빗 엔드포인트 액세스도 활성화하는 것이 좋습니다. 프라이빗 엔드포인트를 활성화하지 않은 경우, 퍼블릭 액세스 엔드포인트 CIDR 소스에는 VPC의 송신 소스가 포함되어야 합니다. 예를 들어 NAT 게이트웨이를 통해 인터넷과 통신하는 프라이빗 서브넷에 노드가 있는 경우, 퍼블릭 엔드포인트에서 허용하는 CIDR 블록의 일부로 NAT 게이트웨이의 아웃바운드 IP 주소를 추가해야 합니다. CIDR 블록을 지정하지 않은 경우 퍼블릭 API 서버 엔드포인트는 모든 (0.0.0.0/0) IP 주소에서 요청을 수신합니다.

   > **Note**
   >
   > 다음 명령은 API 서버 엔드포인트에 대한 단일 IP 주소에서 프라이빗 액세스 및 퍼블릭 액세스를 활성화합니다. *`203.0.113.5/32`*를 단일 CIDR 블록 또는 네트워크 액세스를 제한할 쉼표로 구분된 CIDR 블록 목록으로 바꿉니다.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --resources-vpc-config
 endpointPublicAccess=true,publicAccessCidrs="203.0.113.5/32",endpointPrivateAccess=true
```

   예제 출력은 다음과 같습니다.

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "InProgress",
        "type": "EndpointAccessUpdate",
        "params": [
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

2. 이전 명령에서 반환된 클러스터 이름과 업데이트 ID를 사용하여 다음 명령으로 엔드포인트 액세스 업데이트의 상태를 모니터링합니다. 상태가 Successful로 표시되면 업데이트가 완료된 것입니다.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000
```

예제 출력은 다음과 같습니다.

```
{
    "update": {
        "id": "e6f0905f-a5d4-4a2a-8c49-EXAMPLE00000",
        "status": "Successful",
        "type": "EndpointAccessUpdate",
        "params": [
```

```
            {
                "type": "EndpointPublicAccess",
                "value": "true"
            },
            {
                "type": "EndpointPrivateAccess",
                "value": "true"
            },
            {
                "type": "publicAccessCidrs",
                "value": "[\203.0.113.5/32\"]"
            }
        ],
        "createdAt": 1576874258.137,
        "errors": []
    }
}
```

프라이빗 전용 API 서버 액세스

클러스터의 Kubernetes API 서버 엔드포인트에 대해 퍼블릭 액세스를 비활성화하면 VPC 또는 연결된 네트워크 내에서만 API 서버에 액세스할 수 있습니다. Kubernetes API 서버 엔드포인트에 액세스하는 방법은 다음과 같습니다.

연결된 네트워크

AWS 전송 게이트웨이 또는 기타 연결 옵션을 사용하여 네트워크를 VPC에 연결한 다음 연결된 네트워크의 컴퓨터를 사용합니다. Amazon EKS 제어 영역 보안 그룹에 연결된 네트워크의 포트 443에서 수신 트래픽을 허용하는 규칙이 포함되어 있는지 확인해야 합니다.

Amazon EC2 Bastion Host

Amazon EC2 인스턴스를 클러스터의 VPC에 있는 퍼블릭 서브넷으로 시작한 다음 SSH를 통해 해당 인스턴스에 로그인하여 `kubectl` 명령을 실행할 수 있습니다. 자세한 내용은 AWS 기반 Linux Bastion Host를 참조하세요. Amazon EKS 제어 영역 보안 그룹에 Bastion Host의 포트 443에서 수신 트래픽을 허용하는 규칙이 포함되어 있는지 확인해야 합니다. 자세한 내용은 Amazon EKS 보안 그룹 요구 사항 및 고려 사항 단원을 참조하십시오.

Bastion Host에 대해 `kubectl`을 구성할 때 클러스터의 RBAC 구성에 이미 매핑된 AWS 자격 증명을 사용하거나, 엔드포인트 퍼블릭 액세스를 제거하기 전에 Bastion에서 사용할 IAM 보안 주체를

RBAC 구성에 추가해야 합니다. 자세한 내용은 the section called “Kubernetes API에 대한 액세스 권한 부여” 및 권한이 없거나 액세스가 거부됨(kubectl) 단원을 참조하세요.

AWS Cloud9 IDE

AWS Cloud9은 브라우저만으로 코드를 작성, 실행 및 디버깅할 수 있는 클라우드 기반 통합 개발 환경(IDE)입니다. 클러스터의 VPC에 AWS Cloud9 IDE를 생성할 수 있으며, IDE를 사용하여 클러스터와 통신할 수 있습니다. 자세한 내용은 AWS Cloud9에서 환경 생성 섹션을 참조하세요. Amazon EKS 제어 영역 보안 그룹에 IDE 보안 그룹의 포트 443에서 수신 트래픽을 허용하는 규칙이 포함되어 있는지 확인해야 합니다. 자세한 내용은 Amazon EKS 보안 그룹 요구 사항 및 고려 사항 단원을 참조하십시오.

AWS Cloud9 IDE에 대해 `kubectl`을 구성할 때 클러스터의 RBAC 구성에 이미 매핑된 AWS 자격 증명을 사용하거나, 엔드포인트 퍼블릭 액세스를 제거하기 전에 IDE에서 사용할 IAM 보안 주체를 RBAC 구성에 추가해야 합니다. 자세한 내용은 Kubernetes API에 대한 액세스 권한 부여 및 권한이 없거나 액세스가 거부됨(kubectl) 단원을 참조하세요.

기존 클러스터에서 비밀 암호화 활성화

비밀 암호화를 사용 설정하는 경우 Kubernetes 비밀은 선택한 AWS KMS key 키를 사용하여 암호화됩니다. KMS 키는 다음 조건을 충족해야 합니다.

- 대칭
- 데이터 암호화 및 해독 가능
- 클러스터와 같은 AWS 리전에 생성됨
- KMS 키가 다른 계정에서 생성된 경우 IAM 보안 주체는 KMS 키에 액세스할 수 있어야 합니다.

자세한 내용은 AWS Key Management Service 개발자 안내서의 다른 계정의 IAM 보안 주체가 KMS 키를 사용하도록 허용을 참조하세요.

> **Warning**
>
> 비밀 암호화를 사용 설정한 후에는 사용 중지할 수 없습니다. 이 작업은 되돌릴 수 없습니다.

eksctl

다음 두 가지 방법으로 암호화를 활성화할 수 있습니다.

- 단일 명령으로 클러스터에 암호화를 추가합니다.

  비밀을 자동으로 다시 암호화하려면 다음 명령을 실행합니다.

  ```
  eksctl utils enable-secrets-encryption \
      --cluster my-cluster \
      --key-arn arn:aws:kms:region-code:account:key/key
  ```

  비밀을 자동으로 다시 암호화하는 것을 옵트아웃하려면 다음 명령을 실행합니다.

  ```
  eksctl utils enable-secrets-encryption
      --cluster my-cluster \
      --key-arn arn:aws:kms:region-code:account:key/key \
      --encrypt-existing-secrets=false
  ```

- `kms-cluster.yaml` 파일을 사용하여 클러스터에 암호화를 추가합니다.

  ```
  apiVersion: eksctl.io/v1alpha5
  kind: ClusterConfig

  metadata:
    name: my-cluster
    region: region-code

  secretsEncryption:
    keyARN: arn:aws:kms:region-code:account:key/key
  ```

  비밀을 자동으로 다시 암호화하려면 다음 명령을 실행합니다.

  ```
  eksctl utils enable-secrets-encryption -f kms-cluster.yaml
  ```

  비밀을 자동으로 다시 암호화하는 것을 옵트아웃하려면 다음 명령을 실행합니다.

  ```
  eksctl utils enable-secrets-encryption -f kms-cluster.yaml --encrypt-existing-
  secrets=false
  ```

AWS Management Console

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.

2. KMS 암호화를 추가하려는 클러스터를 선택합니다.
3. 개요(Overview) 탭을 선택합니다(기본적으로 선택되어 있습니다).
4. 아래로 스크롤하여 암호 암호화(Secrets encryption) 단원으로 이동하고 활성화(Enable)를 선택합니다.
5. 드롭다운 목록에서 키를 선택하고 활성화(Enable) 버튼을 선택합니다. 나열된 키가 없으면 먼저 키를 생성해야 합니다. 자세한 내용은 키 생성을 참조하세요.
6. 확인(Confirm) 버튼을 선택하고 선택한 키를 사용합니다.

AWS CLI

1. 다음 AWS CLI 명령을 사용하여 비밀 암호화(secrets encryption) 구성을 클러스터와 연결합니다. *example values*을 사용자의 값으로 교체합니다.

```
aws eks associate-encryption-config \
    --cluster-name my-cluster \
    --encryption-config '[{"resources":["secrets"],"provider":
{"keyArn":"arn:aws:kms:region-code:account:key/key"}}]'
```

예제 출력은 다음과 같습니다.

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "InProgress",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws:kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734,
    "errors": []
  }
}
```

2. 다음 명령을 사용하여 암호화 업데이트의 상태를 모니터링할 수 있습니다. 이전 출력에서 반환된 특정 `cluster name` 및 `update ID`를 사용합니다. `Successful` 상태가 표시되면 업데이트가 완료됩니다.

```
aws eks describe-update \
    --region region-code \
    --name my-cluster \
    --update-id 3141b835-8103-423a-8e68-12c2521ffa4d
```

예제 출력은 다음과 같습니다.

```
{
  "update": {
    "id": "3141b835-8103-423a-8e68-12c2521ffa4d",
    "status": "Successful",
    "type": "AssociateEncryptionConfig",
    "params": [
      {
        "type": "EncryptionConfig",
        "value": "[{\"resources\":[\"secrets\"],\"provider\":{\"keyArn\":
\"arn:aws:kms:region-code:account:key/key\"}}]"
      }
    ],
    "createdAt": 1613754188.734>,
    "errors": []
  }
}
```

3. 클러스터에서 암호화가 활성화되었는지 확인하려면 `describe-cluster` 명령를 실행합니다. 응답에는 `EncryptionConfig` 문자열이 포함됩니다.

```
aws eks describe-cluster --region region-code --name my-cluster
```

클러스터에서 암호화를 사용 설정한 후에는 기존 비밀을 모두 새 키로 암호화해야 합니다.

> **Note**
>
> `eksctl`을 사용하면 비밀을 자동으로 다시 암호화하는 것을 옵트아웃하는 경우에만 다음 명령을 실행해야 합니다.

```
kubectl get secrets --all-namespaces -o json | kubectl annotate --overwrite -f - kms-encryption-timestamp="time value"
```

Warning

기존 클러스터에 대해 비밀 암호화를 사용 설정하고 사용하는 KMS 키가 삭제된 경우에는 클러스터 복구 경로가 없습니다. KMS 키를 삭제하면 클러스터가 영구적으로 성능 저하 상태가 됩니다. 자세한 내용은 AWS KMS 키 삭제를 참조하세요.

Note

기본적으로 `create-key` 명령은 AWS KMS 작업 및 리소스에 대한 액세스 권한을 계정 루트 관리자에게 부여하는 키 정책으로 대칭 암호화 KMS 키를 생성합니다. 권한을 축소하려면 `create-cluster` API를 호출할 보안 주체의 정책에서 `kms:DescribeKey` 및 `kms:CreateGrant` 작업이 허용되는지 확인합니다.
KMS 봉투 암호화를 사용하는 클러스터의 경우 `kms:CreateGrant` 권한이 필요합니다.
조건 `kms:GrantIsForAWSResource`는 CreateCluster 작업에 대해 지원되지 않으며, CreateCluster를 수행하는 사용자에 대한 `kms:CreateGrant` 권한을 제어하기 위해 KMS 정책을 사용해서는 안 됩니다.

Amazon EKS 클러스터에 대해 Windows 지원 사용 설정

Windows 노드를 배포하기 전에 다음 사항을 고려해야 합니다.

고려 사항

- `HostProcess` 포드로 Windows 노드에서 호스트 네트워킹을 사용할 수 있습니다. 자세한 내용은 Kubernetes 설명서의 Create a Windows `HostProcess`Pod를 참조하세요.
- Amazon EKS 클러스터에는 CoreDNS와 같이 Linux에서만 실행되는 코어 시스템 Pods를 실행하기 위한 Linux 또는 Fargate 노드가 하나 이상 있어야 합니다.
- `kubelet` 및 `kube-proxy` 이벤트 로그는 EKS Windows 이벤트 로그로 리디렉션되며 200MB 제한으로 설정됩니다.
- Windows 노드에서 실행되는 Pods에 Pods의 보안 그룹를 사용할 수 없습니다.
- Windows 노드에는 사용자 지정 네트워킹을 사용할 수 없습니다.

- Windows 노드에서는 IPv6를 사용할 수 없습니다.
- Windows 노드는 노드당 하나의 Elastic 네트워크 인터페이스를 지원합니다. 기본적으로 Windows 노드당 실행할 수 있는 Pods 수는 노드의 인스턴스 유형에 대해 탄력적 네트워크 인터페이스당 사용 가능한 IP 주소 수에서 1을 뺀 값과 같습니다. 자세한 내용은 Windows 인스턴스용 Amazon EC2 사용 설명서의 인스턴스 유형별 네트워크 인터페이스당 IP 주소를 참조하세요.
- Amazon EKS 클러스터에서 로드 밸런서가 있는 단일 서비스는 최대 1,024개의 백엔드 Pods를 지원할 수 있습니다. 각 Pod에는 고유한 IP 주소가 있습니다. OS 빌드 17763.2746부터 Windows Server 업데이트 후 이전의 64개 Pods 제한이 더 이상 적용되지 않습니다.
- Windows 컨테이너는 Fargate의 Amazon EKS Pods에 대해 지원되지 않습니다.
- vpc-resource-controller 포드에서 로그를 검색할 수 없습니다. 이전에는 컨트롤러를 데이터 영역에 배포할 때 가능했습니다.
- IPv4 주소가 새 포드에 할당되기 전에 휴지 기간이 있습니다. 이렇게 하면 오래된 kube-proxy 규칙으로 인해 동일한 IPv4 주소를 사용하는 이전 포드로 트래픽이 흐르는 것을 방지할 수 있습니다.
- 컨트롤러의 소스는 GitHub에서 관리됩니다. 컨트롤러에 기여하거나 컨트롤러에 대한 문제를 제기하려면 GitHub의 프로젝트를 방문하세요.
- Windows 관리형 노드 그룹에 사용자 지정 AMI ID를 지정할 때는 AWS IAM Authenticator 구성 맵에 eks:kube-proxy-windows를 추가합니다. 자세한 내용은 AMI ID 지정 시 제한과 조건 단원을 참조하십시오.

사전 조건

- 기존 클러스터가 있어야 합니다. 클러스터는 다음 표에 나열된 Kubernetes 버전 및 플랫폼 버전 중 하나를 실행해야 합니다. 나열된 것보다 이후의 모든 Kubernetes 및 플랫폼 버전이 지원됩니다. 클러스터 또는 플랫폼 버전이 다음 버전 중 하나보다 이전인 경우 클러스터의 데이터 영역에서 레거시 Windows 지원을 사용 설정해야 합니다. 클러스터가 다음 Kubernetes 및 플랫폼 버전 이상 중 하나에 있으면 컨트롤 플레인에서 레거시 Windows 지원을 제거하고 Windows 지원을 사용 설정할 수 있습니다.

Kubernetes 버전	플랫폼 버전
1.29	eks.1
1.28	eks.1
1.27	eks.1

Kubernetes 버전	플랫폼 버전
1.26	eks.1
1.25	eks.1
1.24	eks.2

- CoreDNS를 실행하려면 클러스터에 Linux 노드 또는 Fargate Pod가 1개 이상(최소 2개 권장) 있어야 합니다. 레거시 Windows 지원을 사용 설정하는 경우 Linux 노드(Fargate Pod 사용 불가)를 사용하여 CoreDNS를 실행해야 합니다.
- 기존 Amazon EKS 클러스터 IAM 역할 보유.

Windows 지원 활성화

클러스터가 전제 조건에 나열된 Kubernetes 및 플랫폼 버전 중 하나 이상이 아니면 대신에 레거시 Windows 지원을 활성화해야 합니다. 자세한 내용은 레거시 Windows 지원 사용 단원을 참조하십시오.

클러스터에서 Windows 지원을 사용 설정한 적이 없다면 다음 단계로 건너뜁니다.

전제 조건에 나열된 Kubernetes 또는 플랫폼 버전보다 이전인 클러스터에서 Windows 지원을 사용 설정한 경우 먼저 데이터 영역에서 `vpc-resource-controller` 및 `vpc-admission-webhook`를 제거해야 합니다. 낙후되었으며 더 이상 필요하지 않습니다.

클러스터에 Windows 지원을 사용하려면 다음을 수행합니다.

1. 클러스터에 Amazon Linux 노드가 없고 Pods에 보안 그룹을 사용하는 경우 다음 단계로 건너뜁니다. 그렇지 않으면 `AmazonEKSVPCResourceController` 관리형 정책이 클러스터 역할에 연결되어 있는지 확인합니다. *`eksClusterRole`*를 클러스터 역할 이름으로 교체합니다.

   ```
   aws iam list-attached-role-policies --role-name eksClusterRole
   ```

   예제 출력은 다음과 같습니다.

   ```
   {
       "AttachedPolicies": [
           {
               "PolicyName": "AmazonEKSClusterPolicy",
               "PolicyArn": "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
           },
   ```

```
        {
            "PolicyName": "AmazonEKSVPCResourceController",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonEKSVPCResourceController"
        }
    ]
}
```

이전 출력에서와 같이 정책이 연결된 경우 다음 단계를 건너뜁니다.

2. AmazonEKSVPCResourceController 관리형 정책을 Amazon EKS 클러스터 IAM 역할에 연결합니다. *eksClusterRole*를 클러스터 역할 이름으로 교체합니다.

```
aws iam attach-role-policy \
  --role-name eksClusterRole \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
```

3. 다음 콘텐츠를 가진 *vpc-resource-controller-configmap.yaml*이라는 파일을 생성합니다:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: amazon-vpc-cni
  namespace: kube-system
data:
  enable-windows-ipam: "true"
```

4. 클러스터에 ConfigMap 적용.

```
kubectl apply -f vpc-resource-controller-configmap.yaml
```

5. aws-auth ConfigMap에 Windows 노드의 인스턴스 역할에 대한 매핑이 포함되어 eks:kube-proxy-windows RBAC 권한 그룹을 포함하는지 확인합니다. 다음 명령을 실행하여 확인할 수 있습니다.

```
kubectl get configmap aws-auth -n kube-system -o yaml
```

예제 출력은 다음과 같습니다.

```
apiVersion: v1
kind: ConfigMap
```

```
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      - eks:kube-proxy-windows # This group is required for Windows DNS resolution
 to work
      rolearn: arn:aws:iam::111122223333:role/eksNodeRole
      username: system:node:{{EC2PrivateDNSName}}
[...]
```

그룹 아래에 eks:kube-proxy-windows가 표시될 것입니다. 그룹이 지정되지 않은 경우 ConfigMap을 업데이트하거나 생성하여 필수 그룹을 포함해야 합니다. aws-auth ConfigMap에 대한 자세한 내용은 클러스터에 aws-authConfigMap 적용 섹션을 참조하세요.

데이터 영역에서 레거시 Windows 지원 제거

전제 조건에 나열된 Windows 또는 플랫폼 버전보다 이전인 클러스터에서 Kubernetes 지원을 사용 설정한 경우 먼저 데이터 영역에서 vpc-resource-controller 및 vpc-admission-webhook를 제거해야 합니다. 제공한 기능이 이제 제어 영역에서 사용 설정되기 때문에 낙후되어 더 이상 필요하지 않습니다.

1. 다음 명령을 사용하여 vpc-resource-controller를 제거합니다. 원래 설치 시 사용한 도구에 관계없이 이 명령을 사용합니다. *region-code*(/manifests/ 뒤의 해당 텍스트 인스턴스만)를 클러스터가 있는 AWS 리전으로 바꿉니다.

   ```
   kubectl delete -f https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-resource-controller/latest/vpc-resource-controller.yaml
   ```

2. 설치 시 사용한 도구에 대한 지침을 사용하여 vpc-admission-webhook를 제거합니다.

   eksctl

   다음 명령을 실행합니다.

   ```
   kubectl delete deployment -n kube-system vpc-admission-webhook
   kubectl delete service -n kube-system vpc-admission-webhook
   ```

```
kubectl delete mutatingwebhookconfigurations.admissionregistration.k8s.io vpc-
admission-webhook-cfg
```

kubectl on macOS or Windows

다음 명령을 실행합니다. *region-code*(/manifests/ 뒤의 해당 텍스트 인스턴스만)를 클러스터가 있는 AWS 리전으로 바꿉니다.

```
kubectl delete -f https://s3.us-west-2.amazonaws.com/amazon-
eks/manifests/region-code/vpc-admission-webhook/latest/vpc-admission-webhook-
deployment.yaml
```

3. 컨트롤 플레인에서 클러스터에 대한 Windows 지원을 사용 설정합니다.

Windows 지원 사용 중지

클러스터에서 Windows 지원을 사용 중지하려면 다음을 수행합니다.

1. 클러스터에 Amazon Linux 노드가 포함되어 있고 Pods에 보안 그룹을 함께 사용하는 경우 이 단계를 건너뜁니다.

   클러스터 역할에서 AmazonVPCResourceController 관리형 IAM 정책을 제거합니다. *eksClusterRole*을 클러스터 역할의 이름으로 바꾸고 *111122223333*를 계정 ID로 바꿉니다.

   ```
   aws iam detach-role-policy \
       --role-name eksClusterRole \
       --policy-arn arn:aws:iam::aws:policy/AmazonEKSVPCResourceController
   ```

2. amazon-vpc-cni ConfigMap에서 Windows IPAM을 사용 중지합니다.

   ```
   kubectl patch configmap/amazon-vpc-cni \
                       -n kube-system \
                       --type merge \
                       -p '{"data":{"enable-windows-ipam":"false"}}'
   ```

포드 배포

클러스터에 포드를 배포할 때 노드 유형을 혼합하여 실행하는 경우 사용되는 운영 체제를 지정해야 합니다.

Linux Pods의 경우 매니페스트에서 다음 노드 선택기 텍스트를 사용합니다.

```
nodeSelector:
        kubernetes.io/os: linux
        kubernetes.io/arch: amd64
```

Windows Pods의 경우 매니페스트에서 다음 노드 셀렉터 텍스트를 사용합니다.

```
nodeSelector:
        kubernetes.io/os: windows
        kubernetes.io/arch: amd64
```

샘플 애플리케이션을 배포하여 사용 중인 노드 셀렉터를 볼 수 있습니다.

레거시 Windows 지원 사용

클러스터가 전제 조건에 나열된 Kubernetes 및 플랫폼 버전 중 하나 이상인 경우 대신 컨트롤 플레인에서 Windows 지원을 사용 설정하는 것이 좋습니다. 자세한 내용은 Windows 지원 활성화 단원을 참조하십시오.

클러스터 또는 플랫폼 버전이 전제 조건에 나열된 버전보다 이전인 경우 다음 단계에 따라 Amazon EKS 클러스터의 데이터 영역에 대한 레거시 Windows 지원을 사용 설정할 수 있습니다. 클러스터 및 플랫폼 버전이 전제 조건에 나열된 버전 이상인 경우 레거시 Windows 지원을 제거하고 컨트롤 플레인에 대해 사용 설정하는 것이 좋습니다.

`eksctl`, Windows 클라이언트, macOS 또는 Linux 클라이언트를 사용하여 클러스터에 대해 레거시 Windows 지원을 사용할 수 있습니다.

eksctl

`eksctl`로 클러스터에 대한 레거시 Windows 지원을 사용하려면 다음을 수행합니다.

전제 조건

이 절차에는 `eksctl` 버전 `0.175.0` 이상이 필요합니다. 버전은 다음 명령을 통해 확인할 수 있습니다.

```
eksctl version
```

`eksctl` 업그레이드 또는 설치에 관한 자세한 내용은 `eksctl` 설명서에서 Installation를 참조하세요.

1. 다음 `eksctl` 명령을 사용하여 Amazon EKS 클러스터에 대한 Windows 지원을 사용 설정합니다. *my-cluster*를 클러스터 이름으로 바꿉니다. 이 명령은 Amazon EKS 클러스터에서 Windows 워크로드를 실행하는 데 필요한 VPC 리소스 컨트롤러 및 VPC 승인 컨트롤러 웹후크를 배포합니다.

```
eksctl utils install-vpc-controllers --cluster my-cluster --approve
```

> **Important**
>
> VPC 승인 컨트롤러 Webhook는 발급일로부터 1년 후에 만료되는 인증서로 서명됩니다. 가동 중지 시간을 방지하려면 인증서가 만료되기 전에 인증서를 갱신해야 합니다. 자세한 내용은 VPC 승인 Webhook 인증서 갱신 단원을 참조하십시오.

2. Windows 지원을 활성화한 후 클러스터에서 Windows 노드 그룹을 시작할 수 있습니다. 자세한 내용은 자체 관리형 Windows 노드 시작 단원을 참조하십시오.

Windows

Windows 클라이언트로 클러스터에 대한 레거시 Windows 지원을 사용하려면 다음을 수행합니다.

다음 단계에서 *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다.

1. VPC 리소스 컨트롤러를 클러스터에 배포합니다.

```
kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-resource-controller/latest/vpc-resource-controller.yaml
```

2. VPC 승인 컨트롤러 Webhook을 클러스터에 배포합니다.

   a. 필요한 스크립트 및 배포 파일을 다운로드합니다.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/vpc-admission-webhook-deployment.yaml;
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/Setup-VPCAdmissionWebhook.ps1;
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.ps1;
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-patch-ca-bundle.ps1;
```

b. OpenSSL 및 jq를 설치합니다.

c. VPC 승인 Webhook을 설정하고 배포합니다.

```
./Setup-VPCAdmissionWebhook.ps1 -DeploymentTemplate ".\vpc-admission-webhook-deployment.yaml"
```

> **Important**
>
> VPC 승인 컨트롤러 Webhook는 발급일로부터 1년 후에 만료되는 인증서로 서명됩니다. 가동 중지 시간을 방지하려면 인증서가 만료되기 전에 인증서를 갱신해야 합니다. 자세한 내용은 VPC 승인 Webhook 인증서 갱신 단원을 참조하십시오.

3. 클러스터에 필요한 클러스터 역할 바인딩이 있는지 확인합니다.

```
kubectl get clusterrolebinding eks:kube-proxy-windows
```

다음 예제 출력과 유사한 출력이 반환되면 클러스터에 필요한 역할 바인딩이 있습니다.

```
NAME                       AGE
eks:kube-proxy-windows     10d
```

출력에 `Error from server (NotFound)`가 포함되어 있으면 클러스터에 필요한 클러스터 역할 바인딩이 없습니다. 다음 내용으로 *eks-kube-proxy-windows-crb.yaml* 파일을 생성하여 바인딩을 추가합니다.

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: eks:kube-proxy-windows
  labels:
    k8s-app: kube-proxy
    eks.amazonaws.com/component: kube-proxy
subjects:
  - kind: Group
    name: "eks:kube-proxy-windows"
roleRef:
  kind: ClusterRole
  name: system:node-proxier
```

```
  apiGroup: rbac.authorization.k8s.io
```

클러스터에 구성을 적용합니다.

```
kubectl apply -f eks-kube-proxy-windows-crb.yaml
```

4. Windows 지원을 사용 설정한 후 클러스터에서 Windows 노드 그룹을 시작할 수 있습니다. 자세한 내용은 자체 관리형 Windows 노드 시작 단원을 참조하십시오.

macOS and Linux

macOS 또는 Linux 클라이언트로 클러스터에 대한 레거시 Windows 지원을 사용하려면 다음을 수행합니다.

이 절차를 수행하려면 클라이언트 시스템에 `openssl` 라이브러리 및 `jq` JSON 프로세서가 설치되어 있어야 합니다.

다음 단계에서 *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다.

1. VPC 리소스 컨트롤러를 클러스터에 배포합니다.

```
kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-resource-controller/latest/vpc-resource-controller.yaml
```

2. 클러스터에 대한 VPC 승인 컨트롤러 Webhook 매니페스트를 생성합니다.

   a. 필요한 스크립트 및 배포 파일을 다운로드합니다.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.sh
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-patch-ca-bundle.sh
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/vpc-admission-webhook-deployment.yaml
```

   b. 셸 스크립트를 실행할 수 있도록 셸 스크립트에 권한을 추가합니다.

```
chmod +x webhook-create-signed-cert.sh webhook-patch-ca-bundle.sh
```

   c. 보안 통신을 위한 보안 암호를 생성합니다.

```
./webhook-create-signed-cert.sh
```

d. 보안 암호를 확인합니다.

```
kubectl get secret -n kube-system vpc-admission-webhook-certs
```

e. Webhook을 구성하고 배치 파일을 생성합니다.

```
cat ./vpc-admission-webhook-deployment.yaml | ./webhook-patch-ca-bundle.sh >
 vpc-admission-webhook.yaml
```

3. VPC 승인 Webhook을 배포합니다.

```
kubectl apply -f vpc-admission-webhook.yaml
```

> **Important**
>
> VPC 승인 컨트롤러 Webhook는 발급일로부터 1년 후에 만료되는 인증서로 서명됩니다. 가동 중지 시간을 방지하려면 인증서가 만료되기 전에 인증서를 갱신해야 합니다. 자세한 내용은 VPC 승인 Webhook 인증서 갱신 단원을 참조하십시오.

4. 클러스터에 필요한 클러스터 역할 바인딩이 있는지 확인합니다.

```
kubectl get clusterrolebinding eks:kube-proxy-windows
```

다음 예제 출력과 유사한 출력이 반환되면 클러스터에 필요한 역할 바인딩이 있습니다.

```
NAME                     ROLE                                AGE
eks:kube-proxy-windows   ClusterRole/system:node-proxier   19h
```

출력에 `Error from server (NotFound)`가 포함되어 있으면 클러스터에 필요한 클러스터 역할 바인딩이 없습니다. 다음 내용으로 *eks-kube-proxy-windows-crb.yaml* 파일을 생성하여 바인딩을 추가합니다.

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: eks:kube-proxy-windows
```

```
  labels:
    k8s-app: kube-proxy
    eks.amazonaws.com/component: kube-proxy
subjects:
  - kind: Group
    name: "eks:kube-proxy-windows"
roleRef:
  kind: ClusterRole
  name: system:node-proxier
  apiGroup: rbac.authorization.k8s.io
```

클러스터에 구성을 적용합니다.

```
kubectl apply -f eks-kube-proxy-windows-crb.yaml
```

5. Windows 지원을 사용 설정한 후 클러스터에서 Windows 노드 그룹을 시작할 수 있습니다. 자세한 내용은 자체 관리형 Windows 노드 시작 단원을 참조하십시오.

VPC 승인 Webhook 인증서 갱신

VPC 승인 Webhook에 사용되는 인증서는 발행 후 1년이 지나면 만료됩니다. 가동 중지 시간을 방지하려면 만료되기 전에 인증서를 갱신하는 것이 중요합니다. 다음 명령을 사용하여 현재 인증서의 만료 날짜를 확인할 수 있습니다.

```
kubectl get secret \
    -n kube-system \
    vpc-admission-webhook-certs -o json | \
    jq -r '.data."cert.pem"' | \
    base64 -decode | \
    openssl x509 \
    -noout \
    -enddate | \
    cut -d= -f2
```

예제 출력은 다음과 같습니다.

```
May 28 14:23:00 2022 GMT
```

`eksctl`, Windows 또는 Linux/macOS 컴퓨터를 사용하여 인증서를 갱신할 수 있습니다. VPC 승인 Webhook를 설치하는 데 처음 사용했던 도구에 대한 지침을 따릅니다. 예를 들어 `eksctl`을 사용하여 VPC 승인 Webhook를 처음 설치한 경우 `eksctl` 탭의 지침에 따라 인증서를 갱신해야 합니다.

eksctl

1. 인증서 재설치 *`my-cluster`*를 클러스터 이름으로 바꿉니다.

```
eksctl utils install-vpc-controllers -cluster my-cluster -approve
```

2. 다음 출력이 표시되는지 확인합니다.

```
2021/05/28 05:24:59 [INFO] generate received request
2021/05/28 05:24:59 [INFO] received CSR
2021/05/28 05:24:59 [INFO] generating key: rsa-2048
2021/05/28 05:24:59 [INFO] encoded CSR
```

3. Webhook 배포를 다시 시작합니다.

```
kubectl rollout restart deployment -n kube-system vpc-admission-webhook
```

4. 갱신한 인증서가 만료되어 Windows Pods 포드가 `Container creating` 상태인 경우 해당 Pods를 삭제하고 다시 배포해야 합니다.

Windows

1. 새 인증서를 생성하는 스크립트를 가져옵니다.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.ps1;
```

2. 스크립트에 대한 파라미터를 준비합니다.

```
./webhook-create-signed-cert.ps1 -ServiceName vpc-admission-webhook-svc -SecretName vpc-admission-webhook-certs -Namespace kube-system
```

3. Webhook 배포를 다시 시작합니다.

```
kubectl rollout restart deployment -n kube-system vpc-admission-webhook-deployment
```

4. 갱신한 인증서가 만료되어 Windows Pods 포드가 `Container creating` 상태인 경우 해당 Pods를 삭제하고 다시 배포해야 합니다.

Linux and macOS

전제 조건

컴퓨터에 OpenSSL 및 jq가 `jq`설치되어 있어야 합니다.

1. 새 인증서를 생성하는 스크립트를 가져옵니다.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/manifests/region-code/vpc-admission-webhook/latest/webhook-create-signed-cert.sh
```

2. 권한을 변경합니다.

```
chmod +x webhook-create-signed-cert.sh
```

3. 스크립트를 실행합니다.

```
./webhook-create-signed-cert.sh
```

4. Webhook를 다시 시작합니다.

```
kubectl rollout restart deployment -n kube-system vpc-admission-webhook-deployment
```

5. 갱신한 인증서가 만료되어 Windows Pods 포드가 `Container creating` 상태인 경우 해당 Pods를 삭제하고 다시 배포해야 합니다.

Windows 노드에서 더 높은 Pod 밀도 지원

Amazon EKS에서 각 Pod에는 VPC의 `IPv4` 주소가 할당됩니다. 이로 인해 노드에서 더 많은 Pods를 실행하기에 충분한 리소스가 있더라도 노드에 배포할 수 있는 Pods 수는 사용 가능한 IP 주소에 의해 제한됩니다. Windows 노드에서는 탄력적 네트워크 인터페이스를 하나만 지원하므로 기본적으로 Windows 노드에서 사용 가능한 최대 IP 주소 수는 다음과 같습니다.

```
Number of private IPv4 addresses for each interface on the node - 1
```

하나의 IP 주소는 네트워크 인터페이스의 기본 IP 주소로 사용되므로 Pods에 할당할 수 없습니다.

IP 접두사 위임을 활성화하여 Windows 노드에서 더 높은 Pod 밀도를 활성화할 수 있습니다. 이 기능을 사용하면 보조 IPv4 주소를 할당하는 대신 기본 네트워크 인터페이스에 /28 IPv4 접두사를 할당할 수 있습니다. IP 접두사를 할당하면 노드에서 사용 가능한 최대 IPv4 주소가 다음으로 증가합니다.

```
(Number of private IPv4 addresses assigned to the interface attached to the node - 1) *
 16
```

이렇게 사용 가능한 IP 주소의 수가 상당히 많기 때문에 사용 가능한 IP 주소가 노드의 Pods 수를 확장하는 기능을 제한하지는 않습니다. 자세한 내용은 Amazon EC2 노드에 사용 가능한 IP 주소 증량 단원을 참조하십시오.

프라이빗 클러스터 요구 사항

이 주제에서는 아웃바운드 인터넷 액세스 권한이 없는 AWS 클라우드에 대해, Amazon EKS 클러스터를 배포하는 방법을 설명합니다. AWS Outposts에 로컬 클러스터가 있는 경우 이 주제 대신 Outpost에서 자체 관리형 Amazon Linux 노드 시작하기(Amazon EKS 로컬 클러스터 플랫폼 버전)를 참조하세요.

Amazon EKS 네트워킹에 익숙하지 않은 경우 Amazon EKS 작업자 노드에 대한 클러스터 네트워킹 설명을 참조하세요. 클러스터에 아웃바운드 인터넷 액세스 권한이 없는 경우에는 다음과 같은 요구 사항을 충족해야 합니다.

- 클러스터에서는 VPC에 있는 컨테이너 레지스트리에서 이미지를 가져와야 합니다. VPC에서 Amazon Elastic Container Registry를 생성하고 노드에서 가져올 컨테이너 이미지를 복사할 수 있습니다. 자세한 내용은 한 리포지토리에서 다른 리포지토리로 컨테이너 이미지 복사 단원을 참조하십시오.
- 클러스터에 활성화된 엔드포인트 프라이빗 액세스 권한이 있어야 합니다. 노드에서 클러스터 엔드포인트에 등록하려면 이 권한이 필요합니다. 엔드포인트 퍼블릭 액세스는 선택 사항입니다. 자세한 내용은 Amazon EKS 클러스터 엔드포인트 액세스 제어 단원을 참조하십시오.
- 자체 관리형 Linux 노드와 Windows 노드에서는 시작하기 전에 다음과 같은 부트스트랩 인수가 포함되어야 합니다. 이러한 인수에서는 Amazon EKS 내부 검사를 우회하며 VPC 내에서 Amazon EKS API에 액세스하는 권한이 필요하지 않습니다.

  1. 다음과 같은 명령으로 클러스터의 엔드포인트 값을 결정합니다. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다.

     ```
     aws eks describe-cluster --name my-cluster --query cluster.endpoint --output text
     ```

     예제 출력은 다음과 같습니다.

```
https://EXAMPLE108C897D9B2F1B21D5EXAMPLE.sk1.region-code.eks.amazonaws.com
```

2. 다음과 같은 명령으로 클러스터의 인증 기관 값을 결정합니다. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다.

```
aws eks describe-cluster --name my-cluster --query cluster.certificateAuthority --output text
```

반환된 출력은 긴 문자열입니다.

3. 다음과 같은 명령의 *cluster-endpoint*와 *certificate-authority*를 이전 명령의 출력에서 반환된 값으로 바꿉니다. 자체 관리형 노드 시작 시 부트스트랩 인수 지정에 대한 자세한 내용은 자체 관리형 Amazon Linux 노드 시작하기 및 자체 관리형 Windows 노드 시작 부분을 참조하세요.

- Linux 노드용:

```
--apiserver-endpoint cluster-endpoint --b64-cluster-ca certificate-authority
```

추가 인수는 GitHub에서 부트스트랩 스크립트를 참조하세요.

- Windows 노드용:

> **Note**
>
> 사용자 지정 서비스 CIDR을 사용하는 경우-ServiceCIDR 파라미터를 사용하여 CIDR을 지정해야 합니다. 그렇지 않으면 클러스터의 Pods에 대한 DNS 확인이 실패합니다.

```
-APIServerEndpoint cluster-endpoint -Base64ClusterCA certificate-authority
```

추가 인수는 부트스트랩 스크립트 구성 파라미터 부분을 참조하세요.

- 클러스터의 aws-auth ConfigMap은 VPC 내에서 생성해야 합니다. 항목을 생성하고 aws-auth ConfigMap에 추가하는 방법에 대한 자세한 내용을 보려면 터미널에 **eksctl create iamidentitymapping --help**를 입력합니다. ConfigMap이 서버에 없는 경우 eksctl에서는 명령을 사용해 ID 매핑을 추가할 때 해당 항목을 생성합니다.
- 서비스 계정에 대한 IAM 역할로 구성된 Pods에서는 AWS Security Token Service(AWS STS) API 호출에서 보안 인증 정보를 취득합니다. 아웃바운드 인터넷 액세스 권한이 없는 경우 VPC에서 AWS

STS VPC 엔드포인트를 생성하고 사용해야 합니다. 대부분의 AWS v1 SDK는 기본적으로 AWS STS VPC 엔드포인트를 사용하지 않는 글로벌 AWS STS 엔드포인트(`sts.amazonaws.com`)를 사용합니다. AWS STS VPC 엔드포인트를 사용하려면 리전별 AWS STS 엔드포인트(`sts.region-code.amazonaws.com`)를 사용하도록 SDK를 구성해야 할 수도 있습니다. 자세한 내용은 서비스 계정의 AWS Security Token Service 엔드포인트 구성 단원을 참조하십시오.

- 클러스터의 VPC 서브넷에는 Pods에서 액세스해야 하는 모든 AWS 서비스에 대한 VPC 인터페이스 엔드포인트가 있어야 합니다. 자세한 내용은 인터페이스 VPC 엔드포인트를 사용하여 AWS 서비스에 액세스를 참조하세요. 일반적으로 사용하는 몇 가지 서비스와 엔드포인트가 아래 표에 나열되어 있습니다. 전체 엔드포인트 목록은 AWS PrivateLink 안내서의 AWS PrivateLink와 통합되는 AWS 서비스를 참조하세요.

Service	엔드포인트
Amazon EC2	com.amazonaws.*region-code* .ec2
Amazon Elastic Container Registry(컨테이너 이미지 가져오기용)	com.amazonaws.*region-code* .ecr.api, com.amazonaws.*region-code* .ecr.dkr, and com.amazonaws.*region-code* .s3
Application Load Balancer 및 Network Load Balancer	com.amazonaws.*region-code* .elasticloadbalancing
AWS X-Ray	com.amazonaws.*region-code* .xray
Amazon CloudWatch Logs	com.amazonaws.*region-code* .logs
AWS Security Token Service(서비스 계정에 IAM 역할 사용 시 필요)	com.amazonaws.*region-code* .sts

고려 사항

- 자체 관리형 노드는 필요한 VPC 인터페이스 엔드포인트가 있는 서브넷에 배포해야 합니다. 관리형 노드 그룹을 생성하는 경우 VPC 인터페이스 엔드포인트 보안 그룹에서 서브넷에 대한 CIDR을 허용하거나 본인이 생성된 노드 보안 그룹을 VPC 인터페이스 엔드포인트 보안 그룹에 추가해야 합니다.

- Pods에서 Amazon EFS 볼륨을 사용하는 경우 Amazon EFS CSI 드라이버를 배포하기 전에 컨테이너 이미지에서 Amazon EKS 클러스터와 동일한 AWS 리전을 사용하도록 설정하려면 드라이버의 kustomization.yaml 파일을 변경해야 합니다.
- AWS Load Balancer Controller를 사용하여 AWS ALB(Application Load Balancer) 및 Network Load Balancer를 프라이빗 클러스터에 배포할 수 있습니다. 배포할 때 명령줄 플래그를 사용하여 `enable-shield`, `enable-waf` 및 `enable-wafv2`를 false로 설정해야 합니다. 수신 객체의 호스트 이름을 사용한 인증서 검색은 지원되지 않습니다. 이는 VPC 인터페이스 엔드포인트가 없는 AWS Certificate Manager에 컨트롤러가 도달해야 하기 때문입니다.

  컨트롤러는 Fargate와 함께 사용하는 데 필요한 IP 대상이 있는 Network Load Balancer를 지원합니다. 자세한 내용은 Amazon EKS 애플리케이션 로드 밸런싱 및 Network Load Balancer 생성 섹션을 참조하세요.
- Cluster Autoscaler가 지원됩니다. Cluster Autoscaler Pods를 배포할 때 명령줄이 `--aws-use-static-instance-list=true`를 포함하는지 확인하세요. 자세한 내용은 GitHub에서 고정적 인스턴스 목록 사용을 참조하세요. 워커 노드 VPC에는 AWS STS VPC 엔드포인트 및 자동 크기 조정 VPC 엔드포인트도 포함되어야 합니다.
- 일부 컨테이너 소프트웨어 제품에서는 사용량을 모니터링하는 AWS Marketplace Metering Service에 액세스하는 API 호출을 사용합니다. 프라이빗 클러스터에서는 이러한 호출이 허용되지 않으므로 이러한 컨테이너 유형은 프라이빗 클러스터에서 사용할 수 없습니다.

Amazon EKS Kubernetes 버전

Kubernetes는 새로운 기능, 디자인 업데이트, 버그 수정으로 빠르게 진화하고 있습니다. 커뮤니티에서는 평균 4개월에 한 번씩 새로운 Kubernetes 마이너 버전(예:`1.29`)을 릴리스합니다. Amazon EKS는 마이너 버전의 업스트림 릴리스 및 지원 중단 주기를 따릅니다. Amazon EKS에서 새로운 Kubernetes 버전을 사용할 수 있게 되면 미리 클러스터를 업데이트하여 최신 버전을 사용하는 것이 좋습니다.

마이너 버전은 릴리스 후 처음 14개월 동안 Amazon EKS에서 표준 지원을 받습니다. 버전 표준 지원 종료일이 지나면 자동으로 다음 12개월 동안 추가 지원이 시작됩니다. 추가 지원을 통해 클러스터 시간당 추가 비용을 지불하고 특정 Kubernetes 버전을 더 오래 사용할 수 있습니다. 추가 지원 기간이 종료되기 전에 클러스터를 업데이트하지 않은 경우 클러스터는 현재 지원되는 가장 오래된 추가 버전으로 자동 업그레이드됩니다.

Amazon EKS에서 지원하는 사용 가능한 최신 Kubernetes 버전으로 클러스터를 생성하는 것이 좋습니다. 애플리케이션에 특정 버전의 Kubernetes 버전이 필요한 경우 이전 버전을 선택할 수 있습니다. 표준 또는 추가 지원으로 제공되는 모든 버전에서 새 Amazon EKS 클러스터를 생성할 수 있습니다.

추가 지원으로 사용 가능한 버전

현재 Amazon EKS 표준 지원으로 사용할 수 있는 Kubernetes 버전은 다음과 같습니다.

- 1.29
- 1.28
- 1.27
- 1.26
- 1.25

표준 지원의 각 버전에 대해 알아야 할 중요한 변경 사항은 표준 지원 버전에 대한 릴리스 정보을(를) 참조하세요.

확장 지원으로 사용 가능한 버전

현재 Amazon EKS 추가 지원으로 사용할 수 있는 Kubernetes 버전은 다음과 같습니다.

- 1.24
- 1.23

추가 지원의 각 버전에 대해 알아야 할 중요한 변경 사항은 추가 지원 버전에 대한 릴리스 정보을(를) 참조하세요.

다음과 같은 Kubernetes 버전은 현재 Amazon EKS 추가 지원으로 사용할 수 있지만, 이러한 버전에서는 새 클러스터를 생성할 수 없다는 추가 요구 사항이 있습니다.

- 1.22
- 1.21

  이러한 버전에 대한 자세한 내용은 버전 1.21 및 1.22에 대한 릴리스 정보 섹션을 참조하세요.

Amazon EKS Kubernetes 릴리스 일정

다음 표에는 각 Kubernetes 버전에서 고려해야 할 중요한 릴리스 및 지원 날짜가 나와 있습니다.

> **Note**
>
> 월과 연도만 있는 날짜는 대략적인 날짜이며 알 수 있는 정확한 날짜로 업데이트됩니다.

Kubernetes 버전	업스트림 릴리스	Amazon EKS 릴리스	표준 지원 종료일	추가 지원 종료일
1.29	2023년 12월 13일	2024년 1월 23일	2025년 3월 23일	2026년 3월 23일
1.28	2023년 8월 15일	2023년 9월 26일	2024년 11월 26일	2025년 11월 26일
1.27	2023년 4월 11일	2023년 5월 24일	2024년 7월 24일	2025년 7월 24일
1.26	2022년 12월 9일	2023년 4월 11일	2024년 6월 11일	2025년 6월 11일
1.25	2022년 8월 23일	2023년 2월 22일	2024년 5월 1일	2025년 5월 1일
1.24	2022년 5월 3일	2022년 11월 15일	2024년 1월 31일	2025년 1월 31일
1.23	2021년 12월 7일	2022년 8월 11일	2023년 10월 11일	2024년 10월 11일
1.22	2021년 8월 4일	2022년 4월 4일	2023년 6월 4일	2024년 9월 1일
1.21	2021년 4월 8일	2021년 7월 19일	2023년 2월 16일	2024년 7월 15일

Amazon EKS 버전 FAQ

표준 지원으로 사용할 수 있는 Kubernetes 버전은 몇 개인가요?

Kubernetes 커뮤니티가 지원하는 Kubernetes 버전에 따라 Amazon EKS는 해당하는 시간에 최소 4개의 프로덕션 지원 Kubernetes 버전에 대한 표준 지원을 제공하는 데 전념하고 있습니다. 지정된 Kubernetes 마이너 버전의 표준 지원 종료일은 최소 60일 전에 공지할 예정입니다. 새로운 Kubernetes 버전에 대한 Amazon EKS 인증 및 릴리스 프로세스로 인해, Amazon EKS의

Kubernetes 버전 표준 지원 종료일은 Kubernetes 프로젝트가 버전 업스트림 지원을 중지하는 날짜 또는 이후가 됩니다.

Kubernetes는 Amazon EKS의 표준 지원을 얼마나 오래 받나요?

Kubernetes 버전은 Amazon EKS에서 처음 제공된 후 14개월 동안 표준 지원을 받습니다. 업스트림 Kubernetes가 Amazon EKS에서 제공되는 버전을 더 이상 지원하지 않는 경우에도 마찬가지입니다. Amazon EKS에서 지원되는 Kubernetes 버전에 적용할 수 있는 보안 패치를 백포트하고 있습니다.

Amazon EKS에서 Kubernetes 버전에 대한 표준 지원이 종료되면 알림을 받을 수 있나요?

예. 계정의 클러스터에서 지원 종료 시점이 거의 도래한 버전을 실행 중인 경우 Amazon EKS는 Kubernetes 버전이 Amazon EKS에서 릴리스된 후 약 12개월되는 시점에 AWS Health Dashboard을(를) 통해 알림을 보냅니다. 알림에는 지원 종료 날짜가 포함됩니다. 공지일로부터 최소 60일이 소요됩니다.

Amazon EKS에서 지원하는 Kubernetes 기능은 무엇인가요?

Amazon EKS는 Kubernetes API의 모든 상용(GA) 기능을 지원합니다. Kubernetes 버전 `1.24`부터 새 베타 API가 기본적으로 클러스터에서 활성화되어 있지 않습니다. 하지만 이전 기존 베타 API와 기존 베타 API의 새 버전이 기본적으로 계속 활성화되어 있습니다. 알파 기능은 지원되지 않습니다.

Amazon EKS 관리형 노드 그룹은 클러스터 컨트롤 플레인 버전과 함께 자동으로 업데이트되나요?

아니요. 관리형 노드 그룹은 사용자 계정에 Amazon EC2 인스턴스를 생성합니다. 이러한 인스턴스는 사용자 또는 Amazon EKS가 컨트롤 플레인을 업데이트할 때 자동으로 업그레이드되지 않습니다. 자세한 내용은 관리형 노드 그룹 업데이트 단원을 참조하십시오. 컨트롤 플레인과 노드에 동일한 Kubernetes 버전을 유지하는 것이 좋습니다.

자체 관리형 노드 그룹은 클러스터 컨트롤 플레인 버전과 함께 자동으로 업데이트되나요?

아니요. 자체 관리형 노드 그룹은 사용자 계정에 Amazon EC2 인스턴스를 포함합니다. 이러한 인스턴스는 사용자 또는 Amazon EKS가 컨트롤 플레인 버전을 업데이트할 때 자동으로 업그레이드되지 않습니다. 자체 관리형 노드 그룹에는 콘솔에 업데이트가 필요하다는 표시가 없습니다. 클러스터의 개요(Overview) 탭에 있는 노드(Nodes) 목록에서 노드를 선택하여 노드에 설치된 `kubelet` 버전을 확인하면 업데이트가 필요한 노드를 확인할 수 있습니다. 노드를 수동으로 업데이트해야 합니다. 자세한 내용은 자체 관리형 노드 업데이트 단원을 참조하십시오.

Kubernetes 프로젝트는 최대 세 개의 마이너 버전에 대한 컨트롤 플레인과 노드 간의 호환성을 테스트합니다. 예를 들어 `1.26` 노드는 `1.29` 컨트롤 플레인에 의해 오케스트레이션될 경우 계속 작동합니다. 그러나 컨트롤 플레인보다 지속적으로 세 버전이 낮은 마이너 버전이 있는 노드가 있는 클러스터를 실행하는 것은 권장되지 않습니다. 자세한 내용을 알아보려면 Kubernetes 설

명서의 Kubernetes 버전 및 버전 차이 지원 정책을 참조하세요. 컨트롤 플레인과 노드에 동일한 Kubernetes 버전을 유지하는 것이 좋습니다.

Fargate에서 실행되는 Pods는 자동 클러스터 컨트롤 플레인 버전 업그레이드로 자동 업그레이드되나요?

아니요. Kubernetes 배포와 같은 복제 컨트롤러의 일부로 Fargate Pods를 실행하는 것이 좋습니다. 그런 다음 모든 Fargate Pods의 롤링을 재시작세요. Fargate Pod의 새 버전은 업데이트된 클러스터 컨트롤 플레인 버전과 동일한 `kubelet` 버전을 사용하여 배포됩니다. 자세한 내용을 알아보려면 Kubernetes 문서의 배포를 참조하세요.

> **Important**
>
> 컨트롤 플레인을 업데이트하는 경우 Fargate 노드를 직접 업데이트해야 합니다. Fargate 노드를 업데이트하려면 노드가 나타내는 Fargate Pod를 삭제하고 Pod를 다시 배포합니다. 새 Pod는 클러스터와 동일한 `kubelet` 버전을 사용하여 배포됩니다.

Amazon 추가 지원 FAQ

표준 지원 및 추가 지원이라는 용어가 생소합니다. 이 용어들은 무엇을 의미하나요?

Amazon EKS Kubernetes 버전에 대한 표준 지원은 Amazon EKS에서 Kubernetes 버전이 릴리스될 때 시작되며 릴리스 날짜로부터 14개월 후에 종료됩니다. Kubernetes 버전에 대한 추가 지원은 표준 지원이 종료된 직후에 시작되며 이후 12개월 후에 종료됩니다. 예를 들어, Amazon EKS 버전에 `1.23` 대한 표준 지원은 2023년 10월 11일에 종료됩니다. 버전 `1.23`에 대한 추가 지원은 2023년 10월 12일에 시작되었으며 2024년 10월 11일에 종료될 예정입니다.

Amazon EKS 클러스터에 대한 추가 지원을 받으려면 어떻게 해야 하나요?

Amazon EKS 클러스터에 대한 추가 지원을 받는 데에는 별도의 조치를 취하지 않아도 됩니다. 표준 지원은 Amazon EKS에 Kubernetes 버전이 릴리스될 때 시작되며 릴리스 날짜로부터 14개월 후에 종료됩니다. Kubernetes 버전에 대한 추가 지원은 표준 지원이 종료된 직후에 시작되며 이후 12개월 후에 종료됩니다. 표준 지원이 종료된 이후 Kubernetes 버전에서 실행 중인 클러스터는 추가 지원에 자동으로 등록됩니다.

어떤 Kubernetes 버전에 대해 추가 지원을 받을 수 있나요?

추가 지원은 Kubernetes `1.23` 버전 이상에서 사용할 수 있습니다. 해당 버전에 대한 표준 지원이 종료된 후 최대 12개월 동안 모든 버전에서 클러스터를 실행할 수 있습니다. 즉, Amazon EKS에서 각 버전이 26개월(표준 지원 14개월 + 추가 지원 12개월) 동안 지원됩니다.

추가 지원을 사용하고 싶지 않으면 어떻게 하나요?

추가 지원에 자동으로 등록되는 것을 원하지 않는 경우 클러스터를 표준 Amazon EKS 지원의 Kubernetes 버전으로 업그레이드할 수 있습니다. 표준 지원의 Kubernetes 버전으로 업그레이드되지 않은 클러스터는 자동으로 추가 지원으로 들어갑니다.

12개월의 추가 지원이 종료되면 어떻게 되나요?

26개월의 수명 주기(14개월의 표준 지원 + 12개월의 추가 지원)를 마친 Kubernetes 버전에서 실행되는 클러스터는 다음 버전으로 자동 업그레이드됩니다.

추가 지원 종료 날짜에는 지원되지 않는 버전으로 새 Amazon EKS 클러스터를 더 이상 생성할 수 없게 됩니다. 지원 종료 날짜 이후 점진적인 배포 프로세스를 통해 Amazon EKS에 의해 기존 컨트롤 플레인이 지원되는 최신 버전으로 자동 업데이트됩니다. 자동 컨트롤 플레인 업데이트 후에는 클러스터 추가 기능 및 Amazon EC2 노드를 수동으로 업데이트해야 합니다. 자세한 내용은 Amazon EKS 클러스터의 Kubernetes 버전 업데이트 단원을 참조하십시오.

추가 지원 종료 날짜 이후에 컨트롤 플레인이 정확히 언제 자동으로 업데이트되나요?

Amazon EKS는 특정 기간을 제공할 수 없습니다. 자동 업데이트는 추가 지원 종료 날짜 이후 언제든지 발생할 수 있습니다. 업데이트 전에 알림이 제공되지 않습니다. Amazon EKS 자동 업데이트 프로세스에 의존하지 않고 선제적으로 컨트롤 플레인을 업데이트하는 것이 좋습니다. 자세한 내용은 Amazon EKS 클러스터 Kubernetes 버전 업데이트 단원을 참조하십시오.

컨트롤 플레인을 Kubernetes 버전에 무한정 적용할 수 있나요?

아니요. AWS에서는 클라우드 보안을 가장 중요하게 생각합니다. 특정 시점(보통 1년)이 지난 후 Kubernetes 커뮤니티는 일반 취약성 및 노출(CVE) 패치 릴리스를 중단하고 지원되지 않는 버전의 CVE 제출을 금지합니다. 즉, Kubernetes의 이전 버전과 관련된 취약성은 보고되지 않을 수도 있습니다. 이렇게 하면 취약성의 경우 통지 및 수정 옵션 없이 클러스터를 노출합니다. 이를 고려하여 Amazon EKS는 컨트롤 플레인이 추가 지원 종료에 도달한 버전을 유지하도록 허용하지 않습니다.

추가 지원을 받으려면 추가 비용이 드나요?

예. 추가 지원으로 실행되는 Amazon EKS 클러스터에는 추가 비용이 부과됩니다. 요금 세부 정보는 AWS 블로그의 Amazon EKS extended support for Kubernetes version pricing을 참조하세요.

추가 지원에는 무엇이 포함되나요?

추가 지원의 Amazon EKS 클러스터는 Kubernetes 컨트롤 플레인에 대한 지속적인 보안 패치를 받습니다. 또한 Amazon EKS는 추가 지원 버전의 Amazon VPC CNI, `kube-proxy` 및 CoreDNS 추가 기능에 대한 패치를 릴리스할 예정입니다. Amazon EKS는 Amazon Linux, Bottlerocket 및 Windows용 AWS 게시 Amazon EKS 최적화 AMI용 패치와 해당 버전용 Amazon EKS Fargate 노드

도 출시할 예정입니다. 추가 지원의 모든 클러스터는 계속해서 AWS의 기술 지원을 받을 수 있습니다.

> **Note**
>
> AWS에서 게시한 Amazon EKS 최적화 Windows AMI에 대한 추가 지원은 Kubernetes 버전 `1.23`에서는 사용할 수 없지만 Kubernetes 버전 `1.24` 이상에서는 사용할 수 있습니다.

추가 지원에서 Kubernetes 구성 요소가 아닌 패치에 적용되는 제한 사항이 있나요?

추가 지원은 AWS의 모든 Kubernetes 특정 구성 요소를 포함하지만 항상 Amazon Linux, Bottlerocket 및 Windows용 AWS 게시 Amazon EKS 최적화 AMI에 대한 지원만 제공합니다. 즉, 추가 지원을 사용하는 동안 Amazon EKS 최적화 AMI에 새로운 구성 요소(예: OS 또는 커널)가 있을 수 있습니다. 예를 들어, Amazon Linux 2의 수명 주기가 2025년에 종료되면 Amazon EKS 최적화 Amazon Linux AMI는 최신 Amazon Linux OS를 사용하여 구축됩니다. Amazon EKS는 각 Kubernetes 버전에 대해 이와 같은 중요한 지원 수명 주기 불일치 사항을 발표하고 문서화합니다.

추가 지원 버전을 사용하여 새 클러스터를 생성할 수 있습니까?

예. `1.22` 및 `1.21` 버전은 제외됩니다. 예를 들어, `1.23` 클러스터는 생성할 수는 있지만 `1.22` 클러스터는 생성할 수 없습니다.

표준 지원 버전에 대한 릴리스 정보

이 주제에서는 표준 지원의 각 Kubernetes 버전에 대해 알아야 할 중요한 변경 사항을 제공합니다. 업그레이드할 때는 클러스터의 이전 버전과 새 버전 간에 발생한 변경 사항을 주의 깊게 검토하세요.

> **Note**
>
> `1.24` 및 이후 클러스터에서는 공식적으로 발표된 Amazon EKS AMI에 유일한 런타임으로 `containerd`가 포함됩니다. `1.24` 이전 버전의 Kubernetes에서는 Docker를 기본 런타임으로 사용합니다. 이러한 버전에는 `containerd`로 지원되는 클러스터에서 워크로드를 테스트하기 위해 사용할 수 있는 부트스트랩 플래그 옵션이 있습니다. 자세한 내용은 Amazon EKS에서는 `Dockershim`에 대한 지원을 종료했습니다. 단원을 참조하십시오.

Kubernetes1.29

이제 Kubernetes 1.29는 Amazon EKS에서 사용 가능합니다. Kubernetes 1.29에 대한 자세한 내용을 알아보려면 공식 릴리스 발표를 참조하세요.

> **Important**
>
> - FlowSchema 및 PriorityLevelConfiguration의 사용 중단된 flowcontrol.apiserver.k8s.io/v1beta2 API 버전은 Kubernetes v1.29에서 더 이상 제공되지 않습니다. 사용 중단된 베타 API 그룹을 사용하는 매니페스트 또는 클라이언트 소프트웨어가 있는 경우 v1.29로 업그레이드하기 전에 이를 변경해야 합니다.

- Node 객체의 .status.kubeProxyVersion 필드는 이제 사용 중단되었으며 Kubernetes 프로젝트는 향후 릴리스에서 해당 필드를 제거할 것을 제안하고 있습니다. 사용 중단된 필드는 정확하지 않으며 kube-proxy 버전을 실제로 알지 못하거나 kube-proxy가 실행 중인지 여부조차 모르는 kubelet에 의해 관리되었습니다. 클라이언트 소프트웨어에서 이 필드를 사용했다면 중지하세요. 정보를 신뢰할 수 없으며 해당 필드는 이제 사용 중단되었습니다.
- 잠재적인 공격 표면을 줄이기 위해 Kubernetes 1.29에서 LegacyServiceAccountTokenCleanUp 기능은 레거시 자동 생성 비밀 기반 토큰이 오랫동안(기본값으로 1년) 사용되지 않으면 유효하지 않은 것으로 레이블을 지정하고, 유효하지 않은 것으로 표시된 후 오랫동안(기본값으로 추가 1년) 사용을 시도하지 않으면 자동으로 제거합니다. 이러한 토큰을 식별하려면 다음을 실행합니다.

```
kubectl get cm kube-apiserver-legacy-service-account-token-tracking -nkube-system
```

전체 Kubernetes 1.29 변경 로그는 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.29.md#changelog-since-v1280를 참조하세요.

Kubernetes1.28

이제 Kubernetes 1.28는 Amazon EKS에서 사용 가능합니다. Kubernetes 1.28에 대한 자세한 내용을 알아보려면 공식 릴리스 발표를 참조하세요.

- Kubernetes v1.28은 코어 노드와 컨트롤 플레인 구성 요소 간에 지원되는 스큐를 n-2 에서 n-3로 하나의 마이너 버전으로 확장하여 지원되는 가장 오래된 마이너 버전의 노드 구성 요소(kubelet 및 kube-proxy)가 지원되는 최신 마이너 버전의 컨트롤 플레인 구성 요소(kube-apiserver, kube-

scheduler, kube-controller-manager, cloud-controller-manager)와 함께 작동할 수 있도록 했습니다.

- Pod GC Controller의 지표 force_delete_pods_total 및 force_delete_pod_errors_total은 모든 강제 포드 삭제를 처리하도록 향상되었습니다. 포드가 종료됐거나, 끊겼거나, 서비스 중단 테인트로 종료되었거나, 종료 및 예정되지 않았기 때문에 포드가 강제 삭제됐는지 여부를 나타내기 위해 지표에 이유가 추가됩니다.
- storageClassName이 설정되지 않은 바인딩되지 않은 PersistentVolumeClaim에 기본 StorageClass를 자동으로 할당하도록 PersistentVolume (PV) 컨트롤러가 수정되었습니다. 또한 API 서버 내의 PersistentVolumeClaim 승인 검증 메커니즘이 설정되지 않은 상태에서 실제 StorageClass 이름으로 값을 변경할 수 있도록 조정되었습니다.

전체 Kubernetes 1.28 변경 로그는 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.28.md#changelog-since-v1270를 참조하세요.

Kubernetes1.27

이제 Kubernetes 1.27는 Amazon EKS에서 사용 가능합니다. Kubernetes 1.27에 대한 자세한 내용을 알아보려면 공식 릴리스 발표를 참조하세요.

Important

- 알파 seccomp 주석 seccomp.security.alpha.kubernetes.io/pod 및 container.seccomp.security.alpha.kubernetes.io 주석에 대한 지원이 제거되었습니다. 알파 seccomp 주석은 1.19에서 더 이상 사용되지 않으며 1.27에서 제거됨에 따라 seccomp 주석이 있는 Pods에 대해 seccomp 필드가 더 이상 자동으로 채워지지 않습니다. 대신 Pods 또는 컨테이너에 securityContext.seccompProfile 필드를 사용하여 seccomp 프로파일을 구성합니다. 클러스터에서 더 이상 사용되지 않는 알파 seccomp 주석을 사용하고 있는지 확인하려면 다음 명령을 실행합니다.

```
kubectl get pods --all-namespaces -o json | grep
 -E 'seccomp.security.alpha.kubernetes.io/pod|
container.seccomp.security.alpha.kubernetes.io'
```

- kubelet에 대한 --container-runtime 명령줄 인수가 제거되었습니다. Amazon EKS의 기본 컨테이너 런타임은 1.24부터 containerd가 되어 컨테이너 런타임을 지정할 필요가 없습니다. 1.27부터 Amazon EKS는 부트스트랩 스크립트에 전달된 --container-runtime 인수를 무시합니다. 노드 부트스트랩 프로세스 중 오류를 방지하기 위해 이 인수

를 --kubelet-extra-args에 전달하지 않는 것이 중요합니다. 모든 노드 생성 워크플로 및 빌드 스크립트에서 --container-runtime 인수를 제거해야 합니다.

- Kubernetes 1.27의 kubelet은 기본 kubeAPIQPS를 50으로, kubeAPIBurst를 100으로 늘렸다. 이러한 향상된 기능을 통해 kubelet은 더 많은 양의 API 쿼리를 처리하여 응답 시간과 성능을 개선할 수 있습니다. 조정 요구 사항으로 인해 Pods에 대한 수요가 증가하면 수정된 기본값을 통해 kubelet이 증가한 워크로드를 효율적으로 관리할 수 있습니다. 결과적으로 Pod 실행이 더 빨라지고 클러스터 작업이 더 효과적입니다.
- 보다 세분화된 Pod 토폴로지를 사용하여 minDomain과 같은 정책을 분산할 수 있습니다. 이 파라미터는 Pods가 분산되어야 하는 최소 도메인 수를 지정할 수 있는 기능을 제공합니다. nodeAffinityPolicy 및 nodeTaintPolicy를 사용하면 더 세부적으로 Pod 배포를 관리할 수 있습니다. 이는 노드 친화도, 테인트 및 Pod's 사양의 topologySpreadConstraints에 있는 matchLabelKeys 필드에 따른 것입니다. 이를 통해 롤링 업그레이드 후 계산을 분산하기 위해 Pods를 선택할 수 있습니다.
- Kubernetes1.27은 PersistentVolumeClaims(PVCs)의 수명을 제어하는 StatefulSets에 대한 새로운 정책 메커니즘을 베타 버전으로 승격했습니다. 새로운 PVC 보존 정책을 사용하면 StatefulSet가 삭제되거나 StatefulSet의 복제본이 스케일 다운될 때 StatefulSet 사양 템플릿에서 생성된 PVCs를 자동으로 삭제할지 아니면 유지할지 지정할 수 있습니다.
- Kubernetes API 서버의 goaway-chance 옵션은 연결을 임의로 닫아 HTTP/2 클라이언트 연결이 단일 API 서버 인스턴스에서 멈추지 않도록 합니다. 연결이 닫히면 클라이언트가 다시 연결을 시도하고 로드 밸런싱의 결과로 다른 API 서버에 도달할 가능성이 높습니다. Amazon EKS 버전 1.27은 goaway-chance 플래그를 활성화했습니다. Amazon EKS 클러스터에서 실행 중인 워크로드가 HTTP GOAWAY와 호환되지 않는 클라이언트를 사용하는 경우 연결 종료 시 다시 연결하여 GOAWAY를 처리하도록 클라이언트를 업데이트하는 것이 좋습니다.

전체 Kubernetes 1.27 변경 로그는 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.27.md#changelog-since-v1260를 참조하세요.

Kubernetes1.26

이제 Kubernetes 1.26는 Amazon EKS에서 사용 가능합니다. Kubernetes 1.26에 대한 자세한 내용을 알아보려면 공식 릴리스 발표를 참조하세요.

> **Important**
>
> Kubernetes 1.26는 더 이상 CRI v1alpha2를 지원하지 않습니다. 이로 인해 컨테이너 런타임이 CRI v1을 지원하지 않는 경우 kubelet이(가) 더 이상 노드를 등록하지 않습니다. 또한 이는 Kubernetes 1.26이 containerd 마이너 버전 1.5 이하를 지원하지 않음을 의미합니다. containerd를 사용하는 경우 노드를 Kubernetes 1.26으로 업그레이드하기 전에 containerd 버전 1.6.0 이상으로 업그레이드해야 합니다. v1alpha2만 지원하는 다른 컨테이너 런타임도 업그레이드해야 합니다. 자세한 내용은 컨테이너 런타임 공급업체에 문의하세요. 기본적으로 Amazon Linux 및 Bottlerocket AMI에는 containerd 버전 1.6.6이 포함됩니다.

- Kubernetes 1.26으로 업그레이드하기 전에 Amazon VPC CNI plugin for Kubernetes를 버전 1.12 이상으로 업그레이드하세요. Amazon VPC CNI plugin for Kubernetes버전 1.12 이상으로 업그레이드하지 않는 경우 Amazon VPC CNI plugin for Kubernetes이(가) 충돌합니다. 자세한 내용은 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 단원을 참조하십시오.
- Kubernetes API 서버의 goaway-chance 옵션은 연결을 임의로 닫아 HTTP/2 클라이언트 연결이 단일 API 서버 인스턴스에서 멈추지 않도록 합니다. 연결이 닫히면 클라이언트가 다시 연결을 시도하고 로드 밸런싱의 결과로 다른 API 서버에 도달할 가능성이 높습니다. Amazon EKS 버전 1.26은 goaway-chance 플래그를 활성화했습니다. Amazon EKS 클러스터에서 실행 중인 워크로드가 HTTP GOAWAY와 호환되지 않는 클라이언트를 사용하는 경우 연결 종료 시 다시 연결하여 GOAWAY를 처리하도록 클라이언트를 업데이트하는 것이 좋습니다.

전체 Kubernetes 1.26 변경 로그는 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.26.md#changelog-since-v1250를 참조하세요.

Kubernetes1.25

이제 Kubernetes 1.25는 Amazon EKS에서 사용 가능합니다. Kubernetes 1.25에 대한 자세한 내용을 알아보려면 공식 릴리스 발표를 참조하세요.

> **Important**
>
> - Kubernetes 버전 1.25부터 Amazon EKS 최적화 가속 Amazon Linux AMI와 함께 즉시 사용 가능한 Amazon EC2 P2 인스턴스를 더 이상 사용할 수 없습니다. Kubernetes 버전 1.25 이상의 이러한 AMI는 P2 인스턴스와 호환되지 않는 NVIDIA 525 이후 시리즈의 드라이버를 지원합니다. 하지만 NVIDIA 525 이후 시리즈의 드라이버는 P3, P4, 및 P5 인스턴스와 호환

되므로 이러한 인스턴스를 Kubernetes 버전 1.25 이상의 AMI와 함께 사용할 수 있습니다. Amazon EKS 클러스터를 버전 1.25로 업그레이드하기 전에 P2 인스턴스를 P3, P4 및 P5 인스턴스로 마이그레이션하세요. 또한, NVIDIA 525 이후 시리즈에서 작동하려면 애플리케이션을 사전에 업그레이드해야 합니다. 2024년 1월 말에 최신 NVIDIA 525 시리즈 이상의 드라이버를 Kubernetes 버전 1.23 및 1.24로 백포팅할 계획입니다.

- PodSecurityPolicy (PSP)가 Kubernetes 1.25에서 제거됩니다. PSPs는 포드 보안 승인 (PSA) 및 보드 보안 표준 (PSS)으로 대체됩니다. PSA은 PSS에서 간략히 설명된 보안 컨트롤을 구현하는 기본 제공 승인 컨트롤러입니다. PSA 및 PSS은 Kubernetes 1.25에서 안정화 상태로 전환되며 기본적으로 Amazon EKS에서 활성화됩니다. 클러스터에 PSPs가 있는 경우 클러스터를 버전 1.25으로 업그레이드하기 전에 PSP를 빌트인 Kubernetes PSS 또는 policy-as-code 솔루션으로 마이그레이션해야 합니다. PSP에서 마이그레이션하지 않으면 워크로드가 중단될 수 있습니다. 자세한 내용은 포드 보안 정책 (PSP) 제거 FAQ 부분을 참조하세요.
- Kubernetes 버전 1.25에는 API 우선순위 및 공정성(APF) 이라는 기존 기능의 동작을 변경하는 변경 사항이 포함되어 있습니다. APF는 요청량이 급증하는 기간에 잠재적인 과부하로부터 API 서버를 보호하는 역할을 합니다. 이렇게 하려면 특정 시점에 처리할 수 있는 동시 요청 수를 제한해야 합니다. 이는 다양한 워크로드 또는 사용자로부터 발생하는 요청에 고유한 우선 순위 수준과 제한을 적용하여 달성됩니다. 이 접근 방식은 중요한 애플리케이션 또는 우선 순위가 높은 요청이 우선적으로 처리되도록 하는 동시에 우선 순위가 낮은 요청이 API 서버에 부담을 주지 않도록 합니다. 자세한 내용은 Kubernetes 문서의 API 우선순위 및 공정성 또는 EKS 모범 사례 가이드의 API 우선순위 및 공정성을 참조하세요.

  이러한 업데이트는 PR #10352 및 PR #118601에 도입되었습니다. 이전에는 APF가 모든 유형의 요청을 일률적으로 처리했으며, 각 요청은 단일 단위의 동시 요청 한도를 소비했습니다. APF 동작 변경은 이러한 요청에 따라 API 서버에 부과되는 예외적으로 무거운 부하로 인해 LIST 요청에 더 높은 동시성 단위를 할당합니다. API 서버는 LIST 요청에 따라 반환할 객체 수를 추정합니다. 반환되는 객체 수에 비례하는 동시성 단위를 할당합니다.

  Amazon EKS 버전 1.25 이상으로 업그레이드할 때 이 업데이트된 동작으로 인해 LIST 요청이 많은 워크로드(이전에는 문제 없이 작동했지만)에서 속도 제한이 발생할 수 있습니다. 이는 HTTP 429 응답 코드로 표시됩니다. LIST요청 속도가 제한되어 있으므로 잠재적인 워크로드 중단을 방지하기 위해 이러한 요청의 비율을 줄이려면 워크로드를 재구성하는 것이 아주 좋습니다. 아니면 APF 설정을 조정하여 필수 요청에는 더 많은 용량을 할당하고 필수가 아닌 요청에는 할당되는 용량을 줄임으로써 이 문제를 해결할 수 있습니다. 이러한 완화 기술에 대한 자세한 내용은 EKS 모범 사례 가이드의 요청 삭제 방지을 참조하세요.

- Amazon EKS 1.25에는 업데이트된 YAML 라이브러리가 포함된 향상된 클러스터 인증 기능이 포함되어 있습니다. kube-system 네임스페이스에서 검색된 aws-auth ConfigMap의 YAML 값이 첫 번째 문자가 중괄호인 매크로로 시작하는 경우 중괄호({ }) 앞뒤에 따옴표(" ")를 추가해야 합니다. 이는 Amazon EKS 1.25의 aws-authConfigMap에서 aws-iam-authenticator 버전 v0.6.3이 정확하게 파싱되도록하는 데 필요합니다.
- EndpointSlice의 베타 API 버전(discovery.k8s.io/v1beta1)은 Kubernetes 1.21에서 더 이상 사용되지 않으며 Kubernetes 1.25 일자로 더 이상 제공되지 않습니다. 이 API는 discovery.k8s.io/v1로 업데이트되었습니다. 자세한 내용은 Kubernetes 설명서의 EndpointSlice을(를) 참조하세요. AWS Load Balancer Controller v2.4.6 및 이전 버전에서는 v1beta1 엔드포인트를 사용하여 EndpointSlices와 통신했습니다. AWS Load Balancer Controller에 대한 EndpointSlices 구성을 사용하는 경우 Amazon EKS 클러스터를 1.25로 업그레이드하기 전에 AWS Load Balancer Controller v2.4.7 로 업그레이드해야 합니다. AWS Load Balancer Controller에 대한 EndpointSlices 구성을 사용하는 동안 1.25로 업그레이드하면 컨트롤러가 충돌하여 워크로드가 중단됩니다. 컨트롤러를 업그레이드하려면 AWS Load Balancer Controller란 무엇인가요? 부분을 참조하세요.

- SeccompDefault이 Kubernetes 1.25의 베타로 승격되었습니다. kubelet를 구성할 때 --seccomp-default 플래그를 설정하면 컨테이너 런타임은 무제한(seccomp disabled) 모드가 아닌 해당 RuntimeDefaultseccomp 프로필을 사용합니다. 기본 프로필은 워크로드의 기능을 유지하면서 강력한 보안 기본값 세트를 제공합니다. 이 플래그를 사용할 수 있지만 Amazon EKS는 기본적으로 이 플래그를 활성화하지 않으므로 Amazon EKS 동작은 사실상 변경되지 않습니다. 원하는 경우 노드에서 이 기능을 활성화할 수 있습니다. 자세한 내용은 Kubernetes 설명서의 seccomp를 사용한 컨테이너의 Syscalls 제한 튜토리얼을 참조하세요.
- Docker(Dockershim이라고도 함)의 CRI(컨테이너 런타임 인터페이스) 지원은 Kubernetes 1.24 이후 버전에서 제거되었습니다. Kubernetes 1.24 이후 클러스터에 대한 Amazon EKS 공식 AMIs의 유일한 컨테이너 런타임은 containerd입니다. Amazon EKS 1.24 또는 이후 버전으로 업그레이드하기 전에 더는 지원되지 않는 부트스트랩 스크립트 플래그에 대한 참조를 제거하세요. 자세한 내용은 Amazon EKS에서는 Dockershim에 대한 지원을 종료했습니다. 단원을 참조하십시오.
- 와일드카드 쿼리에 대한 지원은 CoreDNS 1.8.7에서 더 이상 사용되지 않으며 CoreDNS 1.9에서 제거되었습니다. 이는 보안 조치로 수행되었습니다. 와일드카드 쿼리는 더 이상 작동하지 않으며 IP 주소 대신 NXDOMAIN을 반환합니다.
- Kubernetes API 서버의 goaway-chance 옵션은 연결을 임의로 닫아 HTTP/2 클라이언트 연결이 단일 API 서버 인스턴스에서 멈추지 않도록 합니다. 연결이 닫히면 클라이언트가 다시 연결

을 시도하고 로드 밸런싱의 결과로 다른 API 서버에 도달할 가능성이 높습니다. Amazon EKS 버전 1.25은 goaway-chance 플래그를 활성화했습니다. Amazon EKS 클러스터에서 실행 중인 워크로드가 HTTP GOAWAY와 호환되지 않는 클라이언트를 사용하는 경우 연결 종료 시 다시 연결하여 GOAWAY를 처리하도록 클라이언트를 업데이트하는 것이 좋습니다.

전체 Kubernetes 1.25 변경 로그는 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.25.md#changelog-since-v1240를 참조하세요.

추가 지원 버전에 대한 릴리스 정보

이 주제에서는 추가 지원의 각 Kubernetes 버전에 대해 알아야 할 중요한 변경 사항을 제공합니다. 업그레이드할 때는 클러스터의 이전 버전과 새 버전 간에 발생한 변경 사항을 주의 깊게 검토하세요.

Kubernetes 1.24

이제 Kubernetes 1.24는 Amazon EKS에서 사용 가능합니다. Kubernetes 1.24에 대한 자세한 내용을 알아보려면 공식 릴리스 발표를 참조하세요.

Important

- Kubernetes 1.24부터 새 베타 API가 기본적으로 클러스터에서 활성화되어 있지 않습니다. 기본적으로 기존 베타 API와 기존 베타 API의 새 버전이 계속 활성화되어 있습니다. Amazon EKS에서는 업스트림 Kubernetes 1.24와 동일한 동작을 따릅니다. 새 기능을 제어하는 기능 게이트는 새 API 작업과 기존 API 작업에서 모두 기본적으로 활성화되어 있습니다. 이는 업스트림 Kubernetes와 일치합니다. 자세한 내용은 GitHub의 KEP-3136: Beta APIs Are Off by Default(KEP-3136: 베타 API가 기본적으로 꺼져 있음)를 참조하세요.
- Docker(Dockershim이라고도 함)의 CRI(컨테이너 런타임 인터페이스) 지원은 Kubernetes 1.24에서 제거되었습니다. Amazon EKS 공식 AMI에 유일한 런타임인 containerd가 있습니다. Amazon EKS 1.24 또는 이후 버전으로 이동하기 전에 더는 지원되지 않는 부트스트랩 스크립트 플래그에 대한 참조를 제거해야 합니다. 또한 워커 노드에 대해 IP 전달이 활성화되어 있는지 확인해야 합니다. 자세한 내용은 Amazon EKS에서는 Dockershim에 대한 지원을 종료했습니다. 단원을 참조하십시오.
- Container Insights에 대해 Fluentd를 이미 구성한 경우 클러스터를 업데이트하기 전에 Fluentd를 Fluent Bit로 마이그레이션해야 합니다. Fluentd 구문 분석기는 JSON 형식의 로그 메시지만 구문 분석하도록 구성됩니다. dockerd와 달리 containerd 컨테이너 런타임에는 JSON 형식이 아닌 로그 메시지가 있습니다. Fluent Bit로 마이그레이션하지 않으면 구성된 Fluentd's 구문 분석기 중 일부가 Fluentd 컨테이너 내에서 엄청난 양의 오류를 생성

합니다. 마이그레이션에 대한 자세한 내용은 Set up Fluent Bit as a DaemonSet to send logs to CloudWatch Logs를 참조하세요.

- Kubernetes `1.23` 및 이전 버전에서는 `kubelet`에서 제공되는 확인할 수 없는 IP와 DNS SAN(주체 대체 이름)이 있는 인증서가 확인할 수 없는 SAN과 함께 자동으로 발급됩니다. 이러한 확인할 수 없는 SAN은 프로비저닝된 인증서에서 생략됩니다. `1.24` 및 이후 버전의 클러스터에서는 SAN을 확인할 수 없는 경우 `kubelet`에서 제공되는 인증서가 발급되지 않습니다. 이렇게 하면 `kubectl exec` 및`kubectl log` 명령이 작동하지 않습니다. 자세한 내용은 클러스터를 Kubernetes 1.24로 업그레이드하기 전의 인증서 서명 고려 사항 단원을 참조하십시오.
- Fluent Bit를 사용하는 Amazon EKS `1.23` 클러스터를 업그레이드할 때는 `k8s/1.3.12` 이상을 실행하고 있는지 확인해야 합니다. GitHub에서 적용 가능한 최신 Fluent Bit YAML 파일을 다시 적용하여 이 작업을 수행할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Fluent Bit 설정을 참조하세요.

- 토폴로지 인식 힌트를 사용하여 클러스터 워커 노드가 여러 가용 영역에 배포될 때 영역에서 트래픽을 유지하기 위한 기본 설정을 나타낼 수 있습니다. 영역 내에서 트래픽을 라우팅하면 비용을 절감하고 네트워크 성능을 개선하는 데 도움이 될 수 있습니다. 기본적으로 Amazon EKS `1.24`에서는 토폴로지 인식 힌트가 활성화되어 있습니다. 자세한 내용은 Kubernetes 설명서의 토폴로지 인식 힌트를 참조하세요.
- `PodSecurityPolicy`(PSP)는 Kubernetes `1.25`에서 제거될 예정입니다. PSPs는 PSA(포드 보안 승인)로 대체되고 있습니다. PSA는 PSS(포드 보안 표준)에 간략히 설명된 보안 컨트롤을 사용하는 기본 제공 승인 컨트롤러입니다. PSA와 PSS는 모두 베타 기능이며 기본적으로 Amazon EKS에서 활성화되어 있습니다. 버전 `1.25`의 PSP 제거를 해결하려면 Amazon EKS에서 PSS를 구현하는 것이 좋습니다. 자세한 내용은 AWS 블로그의 Implementing Pod Security Standards in Amazon EKS(Amazon EKS에서 포드 보안 표준 구현)를 참조하세요.
- `client.authentication.k8s.io/v1alpha1` ExecCredential이 Kubernetes `1.24`에서 제거되었습니다. ExecCredential API는 Kubernetes `1.22`에서 일반적으로 사용할 수 있었습니다. `v1alpha1` API를 이용하는 client-go 보안 인증 플러그인을 사용하는 경우 플러그인 배포자에게 `v1` API로 마이그레이션하는 방법을 문의하세요.
- Kubernetes `1.24`의 경우 Amazon EKS 관리형 노드 그룹과 0 노드 사이의 크기 조정을 간소화하는 기능을 업스트림 Cluster Autoscaler 프로젝트에 제공했습니다. 이전에는 Cluster Autoscaler에서 0 노드로 크기가 조정된 관리형 노드 그룹의 리소스, 레이블 및 테인트를 이해하려면 담당하는 노드의 세부 정보로 기본 Amazon EC2 Auto Scaling 그룹에 태그를 지정해야 했습니다. 이제는 관리형 노드 그룹에 실행 중인 노드가 없을 때 Cluster Autoscaler에서 Amazon EKS `DescribeNodegroup`

API 작업을 호출합니다. 이 API 작업에서는 Cluster Autoscaler에 필요한 관리 노드 그룹의 리소스, 레이블 및 테인트에 대한 정보가 제공됩니다. 이 기능을 사용하려면 Cluster Autoscaler 서비스 계정 IAM 정책에 `eks:DescribeNodegroup` 권한을 추가해야 합니다. Amazon EKS 관리형 노드 그룹을 지원하는 Auto Scaling의 Cluster Autoscaler 태그 값이 노드 그룹 자체와 충돌하는 경우 Cluster Autoscaler에서는 Auto Scaling 태그의 값을 선호합니다. 이는 필요에 따라 값을 재정의할 수 있도록 하기 위한 것입니다. 자세한 내용은 AutoScaling 단원을 참조하십시오.

- Amazon EKS `1.24`와 함께 Inferentia 또는Trainium 인스턴스 유형을 사용하려는 경우 AWS Neuron 디바이스 플러그인 버전 1.9.3.0 또는 이후 버전으로 업그레이드해야 합니다. 자세한 내용은 AWS Neuron 설명서의 Neuron K8 릴리스[1.9.3.0]를 참조하세요.
- `Containerd`은 기본적으로 Pods에 대해 활성화된 `IPv6`을 가지고 있습니다. 노드 커널 설정을 Pod 네트워크 네임스페이스에 적용합니다. 이 때문에 Pod의 컨테이너는 `IPv4(127.0.0.1)` 및 `IPv6(::1)` 루프백 주소 모두에 바인딩됩니다. `IPv6`은 통신을 위한 기본 프로토콜입니다. 클러스터를 버전 `1.24`로 업데이트하기 전에 다중 컨테이너 Pods를 테스트하는 것이 좋습니다. 루프백 인터페이스의 모든 IP 주소에 바인딩할 수 있도록 앱을 수정합니다. 대부분의 라이브러리는 `IPv4`를 통해 이전 버전과 호환되는 `IPv6` 바인딩을 활성화합니다. 애플리케이션 코드를 수정할 수 없는 경우 두 가지 옵션이 있습니다.
  - `init` 컨테이너를 실행하고 `disable ipv6`를 `true(sysctl -w net.ipv6.conf.all.disable_ipv6=1)`로 설정합니다.
  - 애플리케이션 Pods와 함께 `init` 컨테이너를 삽입하도록 변형 승인 웹훅을 구성합니다.

  모든 노드의 모든 Pods에 대한 `IPv6`을 차단해야 하는 경우 인스턴스의 `IPv6`을 비활성화해야 할 수도 있습니다.
- Kubernetes API 서버의 `goaway-chance` 옵션은 연결을 임의로 닫아 `HTTP/2` 클라이언트 연결이 단일 API 서버 인스턴스에서 멈추지 않도록 합니다. 연결이 닫히면 클라이언트가 다시 연결을 시도하고 로드 밸런싱의 결과로 다른 API 서버에 도달할 가능성이 높습니다. Amazon EKS 버전 `1.24`은 `goaway-chance` 플래그를 활성화했습니다. Amazon EKS 클러스터에서 실행 중인 워크로드가 `HTTP GOAWAY`와 호환되지 않는 클라이언트를 사용하는 경우 연결 종료 시 다시 연결하여 `GOAWAY`를 처리하도록 클라이언트를 업데이트하는 것이 좋습니다.

전체 Kubernetes `1.24` 변경 로그는 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.24.md#changelog-since-v1230를 참조하세요.

Kubernetes 1.23

이제 Kubernetes `1.23`는 Amazon EKS에서 사용 가능합니다. Kubernetes `1.23`에 대한 자세한 내용을 알아보려면 공식 릴리스 발표를 참조하세요.

Important

- CSI(컨테이너 스토리지 인터페이스로 Kubernetes in-tree 볼륨 마이그레이션 기능이 활성화되었습니다. 이 기능을 사용하면 Amazon EBS의 기존 Kubernetes in-tree 스토리지 플러그인을 해당 Amazon EBS CSI 드라이버로 대체할 수 있습니다. 자세한 내용은 Kubernetes 블로그의 Kubernetes 1.17 Feature: Kubernetes In-Tree to CSI Volume Migration Moves to Beta(Kubernetes 1.17 기능: Kubernetes In-Tree에서 CSI로의 볼륨 마이그레이션이 베타 버전으로 전환)를 참조하세요.

  이 기능은 in-tree API를 동등한 CSI API로 변환하고 작업을 대체 CSI 드라이버에 위임합니다. 이 기능을 사용하면 이러한 워크로드에 속하는 기존 `StorageClass`, `PersistentVolume`, `PersistentVolumeClaim` 객체를 사용하는 경우 눈에 띄는 변화가 없을 것입니다. 이 기능을 사용하면 Kubernetes에서 모든 스토리지 관리 작업을 in-tree 플러그인에서 CSI 드라이버로 위임합니다. 기존 클러스터에서 Amazon EBS 볼륨을 사용하는 경우 클러스터를 버전 `1.23`으로 업데이트하기 전에 클러스터에 Amazon EBS CSI 드라이버를 설치합니다. 기존 클러스터를 업데이트하기 전에 드라이버를 설치하지 않은 경우 워크로드가 중단될 수 있습니다. 새 `1.23` 클러스터에서 Amazon EBS 볼륨을 사용하는 워크로드를 배포하려는 경우 클러스터에 워크로드를 배포하기 전에 클러스터에 Amazon EBS CSI 드라이버를 설치합니다. 클러스터에 Amazon EBS CSI 드라이버를 설치하는 방법에 대한 지침은 Amazon EBS CSI 드라이버 부분을 참조하세요. 마이그레이션 기능에 대한 자주 묻는 질문은 Amazon EBS CSI 마이그레이션 관련 자주 묻는 질문 부분을 참조하세요.
- AWS에서 게시한 Amazon EKS 최적화 Windows AMI에 대한 추가 지원은 Kubernetes 버전 `1.23`에서는 사용할 수 없지만 Kubernetes 버전 `1.24` 이상에서는 사용할 수 있습니다.

- Kubernetes는 버전 `1.20`에서의 `dockershim` 지원을 중단하고 버전 `1.24`에서 `dockershim`을 제거했습니다. 자세한 내용은 Kubernetes 블로그의 Kubernetes is Moving on From Dockershim: Commitments and Next Steps(Kubernetes는 에서 이동 중: 약속 및 다음 단계)를 참조하세요. Amazon EKS에서는 Amazon EKS 버전 `1.24`부터 `dockershim`에 대한 지원이 종료됩니다. Amazon EKS 버전 `1.24`부터 Amazon EKS 공식 AMI에는 `containerd`가 유일한 런타임으로 포함됩니다.

  Amazon EKS 버전 `1.23`에서 `dockershim`을 계속 지원하더라도 지금 애플리케이션 테스트를 시작하여 Docker 종속성을 식별하고 제거하는 것이 좋습니다. 이렇게 하면 클러스터를 버전 `1.24`로 업데이트할 준비가 됩니다. `dockershim` 제거에 대한 자세한 내용은 Amazon EKS에서는 `Dockershim`에 대한 지원을 종료했습니다. 부분을 참조하세요.

- Kubernetes에서 Pods, 서비스, 노드에 대한 IPv4/IPv6 듀얼 스택 네트워킹이 정식 출시 단계로 전환되었습니다. 그러나 Amazon EKS와 Amazon VPC CNI plugin for Kubernetes는 듀얼 스택 네트워킹을 지원하지 않습니다. 클러스터가 Pods 및 서비스에 IPv4 또는 IPv6 주소를 할당할 수 있지만 두 주소 유형을 모두 할당할 수는 없습니다.
- Kubernetes에서 PSA(포드 보안 승인) 기능이 베타 상태로 전환되었습니다. 이 기능은 기본적으로 활성화되어 있습니다. 자세한 내용은 Kubernetes 설명서의 포드 보안 승인을 참조하세요. PSA는 포드 보안 정책(PSP) 승인 컨트롤러를 대체합니다. PSP 승인 컨트롤러는 지원되지 않으며 Kubernetes 버전 1.25에서 제거될 예정입니다.

  PSP 승인 컨트롤러는 적용 수준을 설정하는 특정 네임스페이스 레이블을 기반으로 하는 네임스페이스에 Pods에 대한 Pod 보안 표준을 적용합니다. 자세한 내용은 Amazon EKS 모범 사례 안내서의 포드 보안 표준(PSS) 및 포드 보안 승인(PSA)을 참조하세요.
- 클러스터와 함께 배포된 kube-proxy 이미지는 이제 Amazon EKS Distro(EKS-D)에서 관리하는 최소 기본 이미지입니다. 이미지에는 최소 패키지가 포함되어 있으며 셸 또는 패키지 관리자가 없습니다.
- Kubernetes에서 임시 컨테이너가 베타 상태로 전환되었습니다. 임시 컨테이너는 기존 Pod와 동일한 네임스페이스에서 실행되는 임시 컨테이너입니다. 임시 컨테이너를 사용하여 문제 해결 및 디버깅을 위해 Pods 및 컨테이너 상태를 관찰할 수 있습니다. 이 방법은 컨테이너가 충돌했거나 컨테이너 이미지에 디버깅 유틸리티가 포함되어 있지 않아 kubectl exec가 충분하지 않은 경우에 대화형 문제 해결에 특히 유용합니다. 디버깅 유틸리티를 포함하는 컨테이너의 예는 Distroless 이미지입니다. 자세한 내용은 Kubernetes 설명서에서 임시 디버그 컨테이너를 사용한 디버깅을 참조하세요.
- Kubernetes에서 HorizontalPodAutoscaler autoscaling/v2 안정적 API가 정식 출시 단계로 전환되었습니다. HorizontalPodAutoscaler autoscaling/v2beta2 API는 사용 중단되었습니다. 1.26에서 사용할 수 없습니다.
- Kubernetes API 서버의 goaway-chance 옵션은 연결을 임의로 닫아 HTTP/2 클라이언트 연결이 단일 API 서버 인스턴스에서 멈추지 않도록 합니다. 연결이 닫히면 클라이언트가 다시 연결을 시도하고 로드 밸런싱의 결과로 다른 API 서버에 도달할 가능성이 높습니다. Amazon EKS 버전 1.23은 goaway-chance 플래그를 활성화했습니다. Amazon EKS 클러스터에서 실행 중인 워크로드가 HTTP GOAWAY와 호환되지 않는 클라이언트를 사용하는 경우 연결 종료 시 다시 연결하여 GOAWAY를 처리하도록 클라이언트를 업데이트하는 것이 좋습니다.

전체 Kubernetes 1.23 변경 로그는 https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG/CHANGELOG-1.23.md#changelog-since-v1220를 참조하세요.

버전 1.21 및 1.22에 대한 릴리스 정보

> **Important**
>
> 이러한 버전에서는 새 클러스터를 생성할 수 없습니다.

이 주제에서는 버전 `1.22` 및 `1.21`에 대해 알아야 할 중요한 변경 사항을 제공합니다. 업그레이드할 때는 클러스터의 이전 버전과 새 버전 간에 발생한 변경 사항을 주의 깊게 검토하세요.

Kubernetes 버전 **`1.22`**

다음 승인 컨트롤러는 모든 `1.22` 플랫폼 버전(`DefaultStorageClass`, `DefaultTolerationSeconds`, `LimitRanger`, `MutatingAdmissionWebhook`, `NamespaceLifecycle`, `NodeRestriction`, `ResourceQuota`, `ServiceAccount`, `ValidatingAdmissionWebhook`, `PodSecurityPolicy`, `TaintNodesByCondition`, `StorageObjectInUseProtection`, `PersistentVolumeClaimResize`, `ExtendedResourceToleration`, `CertificateApproval`, `PodPriority`, `CertificateSigning`, `CertificateSubjectRestriction`, `RuntimeClass`, `DefaultIngressClass`)에 대해 활성화됩니다.

Kubernetes 버전	Amazon EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
`1.22.17`	`eks.26`	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 4월 1일
`1.22.17`	`eks.14`	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 30일
`1.22.17`	`eks.13`	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 9일
`1.22.17`	`eks.12`	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 5월 5일
`1.22.17`	`eks.11`	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 3월 24일

Kubernetes 버전	Amazon EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.22.16	eks.10	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 1월 27일
1.22.15	eks.9	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 12월 5일
1.22.15	eks.8	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 11월 18일
1.22.15	eks.7	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 11월 7일
1.22.13	eks.6	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 9월 21일
1.22.10	eks.5	향상된 etcd 복원력을 갖춘 새로운 플랫폼 버전.	2022년 8월 15일
1.22.10	eks.4	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전. 이 플랫폼 버전에는 모든 작업자 노드에 aws:eks:cluster-name 태그를 지정하여 이러한 작업자 노드에 대한 비용을 쉽게 할당할 수 있는 새로운 태그 지정 컨트롤러도 도입되었습니다. 자세한 내용은 리소스에 결제용 태그 지정 단원을 참조하십시오.	2022년 7월 21일
1.22.10	eks.3	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 7월 7일
1.22.9	eks.2	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 5월 31일
1.22.6	eks.1	Amazon EKS에 대한 Kubernetes 버전 1.22의 최초 릴리스.	2022년 4월 4일

Kubernetes 버전 **1.21**

다음 승인 컨트롤러는 모든 1.21 플랫폼 버전(DefaultStorageClass, DefaultTolerationSeconds, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, ResourceQuota, ServiceAccount, ValidatingAdmissionWebhook, PodSecurityPolicy, TaintNodesByCondition, StorageObjectInUseProtection, PersistentVolumeClaimResize, ExtendedResourceToleration, CertificateApproval, PodPriority, CertificateSigning, CertificateSubjectRestriction, RuntimeClass, DefaultIngressClass)에 대해 활성화됩니다.

Kubernetes 버전	Amazon EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.21.14	eks.31	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 4월 1일
1.21.14	eks.18	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 9일
1.21.14	eks.17	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 5월 5일
1.21.14	eks.16	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 3월 24일
1.21.14	eks.15	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 1월 27일
1.21.14	eks.14	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 12월 5일
1.21.14	eks.13	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 11월 18일
1.21.14	eks.12	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 11월 7일

Kubernetes 버전	Amazon EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.21.13	eks.11	향상된 etcd 복원력을 갖춘 새로운 플랫폼 버전.	2022년 10월 10일
1.21.13	eks.10	향상된 etcd 복원력을 갖춘 새로운 플랫폼 버전.	2022년 8월 15일
1.21.13	eks.9	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전. 이 플랫폼 버전에는 모든 작업자 노드에 aws:eks:cluster-name 태그를 지정하여 이러한 작업자 노드에 대한 비용을 쉽게 할당할 수 있는 새로운 태그 지정 컨트롤러도 도입되었습니다. 자세한 내용은 리소스에 결제용 태그 지정 단원을 참조하십시오.	2022년 7월 21일
1.21.13	eks.8	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 7월 7일
1.21.12	eks.7	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 5월 31일
1.21.9	eks.6	AWS Security Token Service 엔드포인트는 이전 플랫폼 버전에서 글로벌 엔드포인트로 되돌아갑니다. 서비스 계정에 IAM 역할을 사용할 때 리전 엔드포인트를 사용하려면 사용 설정해야 합니다. 리전 엔드포인트를 사용 설정하는 방법에 대한 지침은 서비스 계정의 AWS Security Token Service 엔드포인트 구성 섹션을 참조하세요.	2022년 4월 8일

Kubernetes 버전	Amazon EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.21.5	eks.5	서비스 계정에 대한 IAM 역할를 사용하는 경우 기본적으로 글로벌 엔드포인트가 아닌 AWS Security Token Service 리전 엔드포인트가 사용됩니다. 그러나 이 변경은 eks.6의 글로벌 엔드포인트로 되돌아갑니다. 업데이트된 Fargate 스케줄러는 대규모 배포 중에 상당히 빠른 속도로 노드를 프로비저닝합니다.	2022년 3월 10일
1.21.5	eks.4	Amazon VPC CNI 자체 관리형 및 Amazon EKS 추가 기능의 버전 1.10.1-eksbuild.1 이 이제 배포된 기본 버전입니다.	2021년 12월 13일
1.21.2	eks.3	Kubernetes 컨트롤 플레인에서 실행되는 VPC 리소스 컨트롤러에서 Windows IPv4 주소 관리를 지원하는 새 플랫폼 버전. Fargate Fluent Bit 로깅을 위한 Kubernetes 필터 지시문 추가됨.	2021년 11월 8일
1.21.2	eks.2	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2021년 9월 17일
1.21.2	eks.1	Amazon EKS에 대한 Kubernetes 버전 1.21의 최초 릴리스.	2021년 7월 19일

Amazon EKS 플랫폼 버전

Amazon EKS 플랫폼 버전은 현재 Kubernetes 패치 버전뿐만 아니라 활성화된 Kubernetes API 서버 플래그 등 Amazon EKS 클러스터 컨트롤 플레인의 기능을 나타냅니다. 각 Kubernetes 마이너 버전에는 하나 이상의 연결된 Amazon EKS 플랫폼 버전이 있습니다. 서로 다른 Kubernetes 마이너 버전에 대한 플랫폼 버전은 독립적입니다. AWS CLI 또는 AWS Management Console을 사용하여 클러스터의 현재 플랫폼 버전을 검색할 수 있습니다. AWS Outposts에 로컬 클러스터가 있는 경우 이 주제 대신 Amazon EKS 로컬 클러스터 플랫폼 버전(Amazon EKS 로컬 클러스터 플랫폼 버전)를 참조하세요.

Amazon EKS에서 새 Kubernetes 마이너 버전(예: 1.29)을 사용할 수 있는 경우, 해당 Kubernetes 마이너 버전에 대한 초기 Amazon EKS 플랫폼 버전은 `eks.1`에서 시작합니다. 그러나 Amazon EKS에서는 새 Kubernetes 컨트롤 플레인 설정을 활성화하고 보안 수정 사항을 제공하기 위해 새 플랫폼 버전을 주기적으로 릴리스합니다.

마이너 버전에 대해 새 Amazon EKS 플랫폼 버전을 사용할 수 있게 된 경우 다음과 같은 사항이 발생합니다.

- Amazon EKS 플랫폼 버전 번호가 증가합니다(`eks.`*n+1*).
- Amazon EKS에서 기존의 모든 클러스터를 해당하는 Kubernetes 마이너 버전에 대한 최신 Amazon EKS 플랫폼 버전으로 자동으로 업그레이드합니다. 기존 Amazon EKS 플랫폼 버전의 자동 업그레이드는 증분 방식으로 롤아웃됩니다. 이 롤아웃 프로세스는 다소 시간이 걸릴 수 있습니다. 최신 Amazon EKS 플랫폼 버전 기능이 즉시 필요한 경우 새 Amazon EKS 클러스터를 생성해야 합니다.

  클러스터가 현재 플랫폼 버전보다 3 이상의 플랫폼 버전인 경우 Amazon EKS가 클러스터를 자동으로 업데이트하지 못했을 수 있습니다. 이 문제의 원인에 대한 자세한 내용은 Amazon EKS 플랫폼 버전이 현재 플랫폼 버전보다 두 버전 이상 뒤떨어져 있음을 참조하세요.
- Amazon EKS에서는 해당 패치 버전으로 새 AMI를 게시할 수 있습니다. 그러나 모든 패치 버전은 지정된 Kubernetes 마이너 버전에 대한 노드 AMI와 EKS 컨트롤 플레인 간에 호환됩니다.

새 Amazon EKS 플랫폼 버전은 주요 변경 사항을 도입하거나 서비스 중단을 초래하지 않습니다.

클러스터는 항상 지정된 Kubernetes 버전에 사용 가능한 최신 Amazon EKS 플랫폼 버전(`eks.`*n*)으로 생성됩니다. 클러스터를 새 Kubernetes 마이너 버전으로 업데이트하면 클러스터가 업데이트한 Kubernetes 마이너 버전에 대한 현재 Amazon EKS 플랫폼 버전을 수신합니다.

현재 및 최근 Amazon EKS 플랫폼 버전은 다음 표에 설명되어 있습니다.

Kubernetes 버전 **1.29**

다음 승인 컨트롤러는 모든 1.29 플랫폼 버전(NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota)에 대해 활성화됩니다.

Kubernetes 버전	EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.29.1	eks.5	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 29일
1.29.1	eks.4	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 20일
1.29.1	eks.3	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 12일
1.29.0	eks.1	EKS에 대한 Kubernetes 버전 1.29의 최초 릴리스. 자세한 내용은 Kubernetes1.29 단원을 참조하십시오.	2024년 1월 23일

Kubernetes 버전 **1.28**

다음 승인 컨트롤러는 모든 1.28 플랫폼 버전(NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota)에 대해 활성화됩니다.

Kubernetes 버전	EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.28.7	eks.11	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 29일
1.28.7	eks.10	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 20일
1.28.6	eks.9	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 12일
1.28.5	eks.7	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 1월 17일
1.28.4	eks.6	액세스 항목, 보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 12월 14일
1.28.4	eks.5	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 12월 12일
1.28.3	eks.4	EKS Pod Identity, 보안 수정 및 개선 사항이 포함된 새 플랫폼 버전.	2023년 11월 10일
1.28.3	eks.3	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 11월 3일
1.28.2	eks.2	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 10월 16일
1.28.1	eks.1	EKS에 대한 Kubernetes 버전 1.28의 최초 릴리스. 자세한 내용은 Kubernetes1.28 단원을 참조하십시오.	2023년 9월 26일

Kubernetes 버전 **1.27**

다음 승인 컨트롤러는 모든 1.27 플랫폼 버전(NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger,

ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota)에 대해 활성화됩니다.

Kubernetes 버전	EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.27.11	eks.15	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 29일
1.27.11	eks.14	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 20일
1.27.10	eks.13	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 12일
1.27.9	eks.11	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 1월 17일
1.27.8	eks.10	액세스 항목, 보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 12월 14일
1.27.8	eks.9	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 12월 12일
1.27.7	eks.8	EKS Pod Identity, 보안 수정 및 개선 사항이 포함된 새 플랫폼 버전.	2023년 11월 10일
1.27.7	eks.7	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 11월 3일
1.27.6	eks.6	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 10월 16일
1.27.4	eks.5	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 8월 30일

Kubernetes 버전	EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.27.4	eks.4	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 7월 30일
1.27.3	eks.3	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 30일
1.27.2	eks.2	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 9일
1.27.1	eks.1	EKS에 대한 Kubernetes 버전 1.27의 최초 릴리스. 자세한 내용은 Kubernetes1.27 단원을 참조하십시오.	2023년 5월 24일

Kubernetes 버전 **1.26**

다음 승인 컨트롤러는 모든 1.26 플랫폼 버전(NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota)에 대해 활성화됩니다.

Kubernetes 버전	EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.26.14	eks.16	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 29일
1.26.14	eks.15	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 20일
1.26.13	eks.14	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 12일

Kubernetes 버전	EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.26.12	eks.12	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 1월 17일
1.26.11	eks.11	액세스 항목, 보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 12월 14일
1.26.11	eks.10	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 12월 12일
1.26.10	eks.9	EKS Pod Identity, 보안 수정 및 개선 사항이 포함된 새 플랫폼 버전.	2023년 11월 10일
1.26.10	eks.8	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 11월 3일
1.26.9	eks.7	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 10월 16일
1.26.7	eks.6	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 8월 30일
1.26.7	eks.5	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 7월 30일
1.26.6	eks.4	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 30일
1.26.5	eks.3	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 9일
1.26.4	eks.2	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 5월 5일
1.26.2	eks.1	EKS에 대한 Kubernetes 버전 1.26의 최초 릴리스. 자세한 내용은 Kubernetes1.26 단원을 참조하십시오.	2023년 4월 11일

Kubernetes 버전 **1.25**

다음 승인 컨트롤러는 모든 1.25 플랫폼 버전(NodeRestriction, ExtendedResourceToleration, NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionWebhook, ResourceQuota)에 대해 활성화됩니다.

Kubernetes 버전	EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.25.16	eks.17	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 29일
1.25.16	eks.16	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 20일
1.25.16	eks.15	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 12일
1.25.16	eks.13	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 1월 17일
1.25.16	eks.12	액세스 항목, 보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 12월 14일
1.25.16	eks.11	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 12월 12일
1.25.15	eks.10	EKS Pod Identity, 보안 수정 및 개선 사항이 포함된 새 플랫폼 버전.	2023년 11월 10일
1.25.15	eks.9	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 11월 3일
1.25.14	eks.8	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 10월 16일

Kubernetes 버전	EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.25.12	eks.7	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 8월 30일
1.25.12	eks.6	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 7월 30일
1.25.11	eks.5	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 30일
1.25.10	eks.4	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 9일
1.25.9	eks.3	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 5월 5일
1.25.8	eks.2	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 3월 24일
1.25.6	eks.1	EKS에 대한 Kubernetes 버전 1.25의 최초 릴리스. 자세한 내용은 Kubernetes1.25 단원을 참조하십시오.	2023년 2월 21일

Kubernetes 버전 **1.24**

다음 승인 컨트롤러는 모든 1.24 플랫폼 버전(CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, PersistentVolumeClaimResize, Priority, PodSecurityPolicy, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionWebhook)에 대해 활성화됩니다.

Kubernetes 버전	EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.24.17	eks.20	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 29일
1.24.17	eks.19	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 20일
1.24.17	eks.18	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 12일
1.24.17	eks.16	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 1월 17일
1.24.17	eks.15	액세스 항목, 보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 12월 14일
1.24.17	eks.14	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 12월 12일
1.24.17	eks.13	EKS Pod Identity, 보안 수정 및 개선 사항이 포함된 새 플랫폼 버전.	2023년 11월 10일
1.24.17	eks.12	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 11월 3일
1.24.17	eks.11	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 10월 16일
1.24.16	eks.10	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 8월 30일
1.24.16	eks.9	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 7월 30일
1.24.15	eks.8	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 30일

Kubernetes 버전	EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.24.14	eks.7	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 9일
1.24.13	eks.6	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 5월 5일
1.24.12	eks.5	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 3월 24일
1.24.8	eks.4	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 1월 27일
1.24.7	eks.3	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 12월 5일
1.24.7	eks.2	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 11월 18일
1.24.7	eks.1	EKS에 대한 Kubernetes 버전 1.24의 최초 릴리스. 자세한 내용은 Kubernetes 1.24 단원을 참조하십시오.	2022년 11월 15일

Kubernetes 버전 **1.23**

다음 승인 컨트롤러는 모든 1.23 플랫폼 버전(CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, PersistentVolumeClaimResize, Priority, PodSecurityPolicy, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionWebhook)에 대해 활성화됩니다.

Kubernetes 버전	EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.23.17	eks.22	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 29일
1.23.17	eks.21	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 20일
1.23.17	eks.20	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 3월 12일
1.23.17	eks.18	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2024년 1월 17일
1.23.17	eks.17	액세스 항목, 보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 12월 14일
1.23.17	eks.16	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 12월 12일
1.23.17	eks.15	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 11월 10일
1.23.17	eks.14	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 11월 3일
1.23.17	eks.13	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 10월 16일
1.23.17	eks.12	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 8월 30일
1.23.17	eks.11	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 7월 30일
1.23.17	eks.10	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 30일

Kubernetes 버전	EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.23.17	eks.9	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 6월 9일
1.23.17	eks.8	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 5월 5일
1.23.17	eks.7	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 3월 24일
1.23.14	eks.6	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 1월 27일
1.23.13	eks.5	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 12월 5일
1.23.13	eks.4	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 11월 18일
1.23.12	eks.3	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 11월 7일
1.23.10	eks.2	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2022년 9월 21일
1.23.7	eks.1	EKS에 대한 Kubernetes 버전 1.23의 최초 릴리스. 자세한 내용은 Kubernetes 1.23 단원을 참조하십시오.	2022년 8월 11일

현재 플랫폼 버전 가져오기

클러스터의 현재 플랫폼 버전 가져오기(콘솔)

1. Amazon EKS 콘솔을 엽니다.
2. 탐색 창에서 클러스터를 선택합니다.
3. 클러스터 목록에서 플랫폼 버전을 확인할 클러스터 이름을 선택합니다.

4. Overview(개요) 탭을 선택합니다.
5. 플랫폼 버전은 세부 정보 섹션에서 확인할 수 있습니다.

클러스터의 현재 플랫폼 버전 가져오기(AWS CLI)

1. 플랫폼 버전을 확인하려는 클러스터의 이름을 결정합니다.
2. 다음 명령을 실행합니다:

```
aws eks describe-cluster --name my-cluster --query cluster.platformVersion
```

예제 출력은 다음과 같습니다.

```
"eks.10"
```

AutoScaling

자동 크기 조정은 변화하는 요구 사항에 맞게 리소스 규모를 자동으로 조정하는 기능입니다. 이는 Kubernetes의 주요 기능이며 그렇지 않으면 수동으로 수행하기 위해 많은 인적 자원을 필요로 할 것입니다.

Amazon EKS는 두 가지 자동 크기 조정 제품을 지원합니다.

Karpenter

Karpenter는 유연한 고성능 Kubernetes 클러스터 자동 규모 조정기로 애플리케이션 가용성과 클러스터 효율성 개선에 도움이 됩니다. Karpenter는 1분 이내에 변화하는 애플리케이션 로드에 대응하여 적절한 크기의 컴퓨팅 리소스(예: Amazon EC2 인스턴스)를 시작합니다. Kubernetes와 AWS의 통합을 통해 Karpenter는 워크로드의 요구 사항을 정확히 충족하는 적시 컴퓨팅 리소스를 프로비저닝할 수 있습니다. Karpenter는 클러스터 워크로드의 특정 요구 사항에 따라 새 컴퓨팅 리소스를 자동으로 프로비저닝합니다. 여기에는 컴퓨팅, 스토리지, 가속화 및 예약 요구 사항이 포함됩니다. Amazon EKS는 Karpenter를 사용하는 클러스터를 지원하지만 Karpenter는 호환되는 모든 Kubernetes 클러스터에서 작동합니다. 자세한 내용은 Karpenter 설명서를 참조하세요.

Cluster Autoscaler

Kubernetes Cluster Autoscaler는 포드가 실패하거나 다른 노드로 다시 예약될 때 클러스터의 노드 수를 자동으로 조정합니다. Cluster Autoscaler는 오토 스케일링을 사용합니다. 자세한 내용은 AWS의 Cluster Autoscaler를 참조하십시오.

액세스 관리

Amazon EKS 클러스터에 대한 액세스를 관리하는 방법을 알아봅니다. Amazon EKS를 사용하려면 Kubernetes 및 AWS Identity and Access Management(AWS IAM) 액세스 처리 방법을 모두 알고 있어야 합니다.

이 섹션에는 다음 내용이 포함되어 있습니다.

the section called “Kubernetes API에 대한 액세스 권한 부여” - 애플리케이션 또는 사용자가 Kubernetes API에 인증할 수 있도록 설정하는 방법을 알아봅니다. 액세스 항목, aws-auth ConfigMap 또는 외부 OIDC 공급자를 사용할 수 있습니다.

the section called “kubectl을 통해 내 클러스터 액세스” - Amazon EKS 클러스터와 통신하도록 kubectl을 구성하는 방법을 알아봅니다. AWS CLI를 사용하여 kubeconfig 파일을 생성할 수 있습니다.

the section called “워크로드에 AWS에 대한 액세스 권한 부여” - Kubernetes 서비스 계정과 AWS IAM 역할을 연결하는 방법을 알아봅니다. Pod Identity 또는 서비스 계정에 대한 IAM 역할(IRSA)을 사용할 수 있습니다.

일반적인 작업:

- 개발자에게 Kubernetes API에 대한 액세스 권한을 부여합니다. AWS Management Console에서 Kubernetes 리소스를 봅니다.
  - 해결 방법: 액세스 항목을 사용하여 Kubernetes RBAC 권한을 AWS IAM 사용자 또는 역할과 연결합니다.
- AWS 자격 증명을 사용하여 Amazon EKS 클러스터와 통신하도록 kubectl을 구성합니다.
  - 해결 방법: AWS CLI를 사용하여 kubeconfig 파일을 생성합니다.
- Ping Identity와 같은 외부 ID 제공업체를 사용하여 Kubernetes API에 대해 사용자를 인증합니다.
  - 해결 방법: 외부 OIDC 공급자를 연결합니다.
- Kubernetes 클러스터의 워크로드에 AWS API를 호출하는 기능을 부여합니다.
  - 해결 방법: Pod Identity를 사용하여 AWS IAM 역할을 Kubernetes 서비스 계정에 연결합니다.

배경:

- Kubernetes 서비스 계정의 작동 방식을 알아봅니다.
- Kubernetes 역할 기반 액세스 제어(RBAC) 모델 검토

- AWS 리소스에 대한 액세스를 관리하는 방법에 자세한 내용은 AWS IAM 사용 설명서를 참조하세요. 또는 AWS IAM 사용에 대한 무료 입문 교육을 수강할 수도 있습니다.

Kubernetes API에 대한 액세스 권한 부여

클러스터에는 Kubernetes API 엔드포인트가 있습니다. Kubectl은 이 API를 사용합니다. 다음 두 유형의 ID를 사용하여 이 API에 인증할 수 있습니다.

- AWS Identity and Access Management(IAM) 보안 주체(역할 또는 사용자) - 이 유형에는 IAM에 대한 인증이 필요합니다. 자격 증명 소스를 통해 제공된 보안 인증 정보를 사용하여 IAM 사용자 또는 페더레이션 ID로 AWS에 로그인할 수 있습니다. 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정한 경우 페더레이션형 ID로만 로그인할 수 있습니다. 페더레이션을 사용하여 AWS에 액세스하면 간접적으로 역할을 수임합니다. 이 유형의 ID를 사용하는 경우 다음을 수행할 수 있습니다.
  - 클러스터의 Kubernetes 객체에서 작업할 수 있도록 Kubernetes 권한을 할당할 수 있습니다. 클러스터의 Kubernetes 객체에 액세스할 수 있도록 IAM 보안 주체에 권한을 할당하는 방법에 대한 자세한 내용은 액세스 항목 관리을 참조하세요.
  - Amazon EKS API, AWS CLI, AWS CloudFormation, AWS Management Console 또는 `eksctl`을 사용하여 Amazon EKS 클러스터 및 해당 리소스에서 작업할 수 있도록 IAM 권한을 할당할 수 있습니다. 자세한 내용은 서비스 권한 부여 참조에서 Amazon Elastic Kubernetes Service에서 정의한 작업을 참조하세요.
  - 노드는 IAM 역할을 수임하여 클러스터에 조인합니다. IAM 보안 주체를 사용하는 클러스터에 액세스하는 기능은 Kubernetes용 AWS IAM Authenticator에서 제공되며, Amazon EKS 컨트롤 플레인에서 실행됩니다.
- 자체 OpenID Connect(OIDC) 제공업체의 사용자 - 이 유형을 사용하려면 OIDC 제공업체에 대한 인증이 필요합니다. Amazon EKS 클러스터에서 자체 OIDC 제공업체를 설정하는 방법에 대한 자세한 내용은 OpenID Connect 자격 증명 공급자에서 클러스터에 대해 사용자 인증 섹션을 참조하세요. 이 유형의 ID를 사용하는 경우 다음을 수행할 수 있습니다.
  - 클러스터의 Kubernetes 객체에서 작업할 수 있도록 Kubernetes 권한을 할당할 수 있습니다.
  - Amazon EKS API, AWS CLI, AWS CloudFormation, AWS Management Console 또는 `eksctl`을 사용하여 Amazon EKS 클러스터 및 해당 리소스에서 작업할 수 있도록 IAM 권한을 할당할 수 있습니다.

클러스터에서는 두 가지 유형의 ID를 모두 사용할 수 있습니다. IAM 인증 방법은 비활성화할 수 없습니다. OIDC 인증 방법은 선택 사항입니다.

IAM 자격 증명을 Kubernetes 권한과 연결

Kubernetes용 AWS IAM Authenticator는 클러스터의 컨트롤 플레인에 설치됩니다. 이를 통해 AWS Identity and Access Management(IAM) 보안 주체(역할 및 사용자)가 클러스터의 Kubernetes 리소스에 액세스할 수 있습니다. 다음 방법 중 하나를 사용하여 IAM 보안 주체가 클러스터의 Kubernetes 객체에 액세스하도록 허용할 수 있습니다.

- 액세스 항목 생성 - 클러스터가 클러스터 Kubernetes 버전의 사전 조건 섹션에 나열된 플랫폼 버전 이상인 경우 이 옵션을 사용하는 것이 좋습니다.

  액세스 항목을 사용하여 클러스터 외부에서 IAM 보안 주체의 Kubernetes 권한을 관리합니다. EKS API, AWS Command Line Interface, AWS SDK, AWS CloudFormation 및 AWS Management Console을 사용하여 클러스터에 대한 액세스를 추가하고 관리할 수 있습니다. 즉, 클러스터를 생성할 때 사용한 것과 동일한 도구로 사용자를 관리할 수 있습니다.

  시작하려면 액세스 항목 설정, 기존 `aws-auth` `ConfigMap` 항목을 액세스 항목으로 마이그레이션 섹션을 차례로 따르세요.

- **`aws-auth` `ConfigMap`**에 항목 추가 - 클러스터의 플랫폼 버전이 사전 조건 섹션에 나열된 버전 이하인 경우 이 옵션을 사용해야 합니다. 클러스터의 플랫폼 버전이 클러스터 Kubernetes 버전의 사전 조건 섹션에 나열된 플랫폼 버전 이상이고 `ConfigMap`에 항목을 추가한 경우 해당 항목을 액세스 항목으로 마이그레이션하는 것이 좋습니다. 그러나 관리형 노드 그룹 또는 Fargate 프로파일과 함께 사용되는 IAM 역할 항목과 같이 Amazon EKS가 `ConfigMap`에 추가한 항목은 마이그레이션할 수 없습니다. 자세한 내용은 the section called “Kubernetes API에 대한 액세스 권한 부여” 단원을 참조하십시오.

  - `aws-auth` `ConfigMap` 옵션을 사용해야 하는 경우 **`eksctl create iamidentitymapping`** 명령을 사용하여 `ConfigMap`에 항목을 추가할 수 있습니다. 자세한 내용은 `eksctl` 설명서의 IAM 사용자 및 역할 관리를 참조하세요.

클러스터 인증 모드 설정

각 클러스터에는 인증 모드가 있습니다. 인증 모드는 IAM 보안 주체가 클러스터의 Kubernetes 객체에 액세스하도록 허용하는 데 사용할 수 있는 방법을 결정합니다. 세 가지 인증 모드가 있습니다.

> **Important**
>
> 액세스 항목 방법을 활성화한 후에는 비활성화할 수 없습니다.

클러스터 생성 중에 ConfigMap 방법을 활성화하지 않으면 나중에 활성화할 수 없습니다. 액세스 항목이 도입되기 전에 생성된 모든 클러스터에는 ConfigMap 메서드가 활성화되어 있습니다.

클러스터 내부 aws-auth ConfigMap

Amazon EKS 클러스터의 원래 인증 모드입니다. 클러스터를 생성한 IAM 보안 주체는 kubectl을 사용하여 클러스터에 액세스할 수 있는 초기 사용자입니다. 초기 사용자는 aws-auth ConfigMap에서 목록에 다른 사용자를 추가하고 클러스터 내 다른 사용자에게 영향을 주는 권한을 할당해야 합니다. ConfigMap에 관리할 항목이 없기 때문에 이러한 다른 사용자는 초기 사용자를 관리하거나 제거할 수 없습니다.

ConfigMap 및 액세스 항목 모두

이 인증 모드에서는 두 가지 방법을 모두 사용하여 클러스터에 IAM 보안 주체를 추가할 수 있습니다. 각 방법에서는 별도의 항목을 저장합니다. 예를 들어 AWS CLI에서 액세스 항목을 추가하는 경우 aws-auth ConfigMap이 업데이트되지 않습니다.

액세스 항목만

이 인증 모드에서는 EKS API, AWS Command Line Interface, AWS SDK, AWS CloudFormation 및 AWS Management Console을 사용하여 IAM 보안 주체의 클러스터에 대한 액세스를 관리할 수 있습니다.

각 액세스 항목에는 유형이 있으며, 보안 주체를 특정 네임스페이스로 제한하는 액세스 범위와 사전 구성된 재사용 가능한 권한 정책을 설정하는 액세스 정책을 조합하여 사용할 수 있습니다. 또는 STANDARD 유형 및 Kubernetes RBAC 그룹을 사용하여 사용자 지정 권한을 할당할 수 있습니다.

인증 모드	메서드
ConfigMap 만 해당(CONFIG_MAP)	aws-auth ConfigMap
EKS API 및 ConfigMap (API_AND_CONFIG_MAP)	EKS API, AWS Command Line Interface, AWS SDK, AWS CloudFormation, AWS Management Console 및 aws-auth ConfigMap 에 있는 액세스 항목

인증 모드	메서드
EKS API만 해당(API)	EKS API, AWS Command Line Interface, AWS SDK, AWS CloudFormation 및 AWS Management Console에 있는 액세스 항목

액세스 항목 관리

필수 조건

- Amazon EKS 클러스터의 클러스터 액세스 옵션을 숙지합니다. 자세한 내용은 Kubernetes API에 대한 액세스 권한 부여 단원을 참조하십시오.
- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요. 액세스 항목을 사용하고 클러스터의 인증 모드를 변경하려면 클러스터의 플랫폼 버전이 다음 테이블에 나열된 버전 이상이거나 Kubernetes 버전이 테이블에 나열된 버전보다 높아야 합니다.

Kubernetes 버전	플랫폼 버전
1.28	eks.6
1.27	eks.10
1.26	eks.11
1.25	eks.12
1.24	eks.15
1.23	eks.17

다음 명령에서 *my-cluster*를 클러스터 이름으로 바꾸고 수정된 명령(**aws eks describe-cluster --name *my-cluster* --query 'cluster.{"Kubernetes Version": version, "Platform Version": platformVersion}'**)을 실행하여 현재 Kubernetes 및 플랫폼 버전을 확인할 수 있습니다.

> **Important**
>
> Amazon EKS가 테이블에 나열된 플랫폼 버전으로 클러스터를 업데이트한 후 Amazon EKS는 클러스터를 처음 생성한 IAM 보안 주체에서 클러스터에 대한 관리자 권한을 가진 액세스 항목을 생성합니다. IAM 보안 주체가 클러스터에 대한 관리자 권한을 갖지 않도록 하려면 Amazon EKS에서 생성한 액세스 항목을 제거합니다.
> 플랫폼 버전이 이전 테이블에 나열된 버전보다 이전 버전인 클러스터의 경우 클러스터 생성자가 항상 클러스터 관리자입니다. 클러스터를 생성한 IAM 사용자 또는 역할에서는 클러스터 관리자 권한을 제거할 수 없습니다.

- 클러스터에 대해 `CreateAccessEntry`, `ListAccessEntries`, `DescribeAccessEntry`, `DeleteAccessEntry`, `UpdateAccessEntry` 권한을 보유한 IAM 보안 주체. Amazon EKS 권한에 대한 자세한 내용은 서비스 권한 부여 참조에서 Actions defined by Amazon Elastic Kubernetes Service를 참조하세요.
- 액세스 항목을 생성하기 위한 기존 IAM 보안 주체 또는 업데이트나 삭제할 기존 액세스 항목.

액세스 항목 설정

액세스 항목 사용을 시작하려면 클러스터의 인증 모드를 `API_AND_CONFIG_MAP` 또는 `API` 모드로 변경해야 합니다. 그러면 액세스 항목에 대한 API가 추가됩니다.

AWS Management Console

액세스 항목을 생성하는 방법

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 관리형 노드 그룹을 생성하려는 클러스터의 이름을 선택합니다.
3. 액세스 탭을 선택합니다.
4. 인증 모드는 클러스터의 현재 인증 모드를 표시합니다. EKS API 모드로 표시되면 이미 액세스 항목을 추가할 수 있으며 나머지 단계는 건너뛰어도 됩니다.
5. 액세스 관리를 선택합니다.
6. 클러스터 인증 모드에서 EKS API로 모드를 선택합니다. EKS API 및 액세스 항목을 제거한 모드로 인증 모드를 다시 변경할 수 없습니다.
7. Save changes(변경 사항 저장)를 선택합니다. Amazon EKS가 클러스터 업데이트를 시작하고 클러스터가 Updating 상태로 변경되며 변경 사항은 업데이트 기록 탭에 기록됩니다.

8. 클러스터가 Active 상태로 돌아갈 때까지 기다립니다. 클러스터가 Active 상태이면 액세스 항목 생성의 단계에 따라 IAM 보안 주체에서 클러스터에 대한 액세스를 추가할 수 있습니다.

AWS CLI

전제 조건

디바이스 또는 AWS CloudShell에 설치 및 구성된 AWS CLI v1의 최신 버전. AWS CLI v2는 한동안 새 기능을 지원하지 않습니다. `aws --version | cut -d / -f2 | cut -d ' ' -f1`을 사용하여 현재 버전을 확인할 수 있습니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 `aws configure`를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.

1.
2. 다음 명령을 실행합니다. *`my-cluster`*를 해당 클러스터의 이름으로 바꿉니다. 이 `ConfigMap` 방법을 영구적으로 비활성화하려면 `API_AND_CONFIG_MAP`을 `API`으로 바꿉니다.

   Amazon EKS가 클러스터 업데이트를 시작하고 클러스터가 UPDATING 상태로 변경되며 변경 사항은 aws eks list-updates에 기록됩니다.

   ```
   aws eks update-cluster-config --name my-cluster --access-config
    authenticationMode=API_AND_CONFIG_MAP
   ```
3. 클러스터가 Active 상태로 돌아갈 때까지 기다립니다. 클러스터가 Active 상태이면 액세스 항목 생성의 단계에 따라 IAM 보안 주체에서 클러스터에 대한 액세스를 추가할 수 있습니다.

액세스 항목 생성

고려 사항

액세스 항목을 생성하기 전에 다음을 고려합니다.

- 액세스 항목에는 기존 IAM 보안 주체 하나의 Amazon 리소스 이름(ARN)만 포함됩니다. IAM 보안 주체는 둘 이상의 액세스 항목에 포함될 수 없습니다. 지정하는 ARN에 대한 추가 고려 사항:
  - IAM 모범 사례에서는 장기 보안 인증 정보를 보유한 IAM 사용자 대신, 단기 보안 인증 정보를 보유한 IAM 역할을 사용하여 클러스터에 액세스하도록 권장합니다. 자세한 내용은 IAM 사용 설명서에

서 임시 보안 인증을 사용하여 AWS에 액세스하려면 인간 사용자가 ID 공급자와의 페더레이션을 사용하도록 요구합니다.를 참조하세요.

- IAM 역할에 대한 ARN인 경우 경로를 포함할 수 있습니다. `aws-auth ConfigMap` 항목의 ARN은 경로를 포함할 수 없습니다. 예를 들어 ARN은 `arn:aws:iam::111122223333:role/development/apps/my-role` 또는 `arn:aws:iam::111122223333:role/my-role`일 수 있습니다.
- 액세스 항목 유형이 `STANDARD` 이외의 유형인 경우(유형에 대한 다음 고려 사항 참조) ARN은 클러스터가 속한 같은 AWS 계정에 있어야 합니다. 유형이 `STANDARD`인 경우 ARN은 클러스터가 있는 계정과 같거나 다른 AWS 계정에 있을 수 있습니다.
- 액세스 항목이 생성된 후에는 IAM 보안 주체를 변경할 수 없습니다.
- 이 ARN으로 IAM 보안 주체를 삭제해도 액세스 항목은 자동으로 삭제되지 않습니다. 삭제하는 IAM 보안 주체에 대한 ARN을 사용하여 액세스 항목을 삭제하는 것이 좋습니다. 액세스 항목을 삭제하지 않고 IAM 보안 주체를 다시 생성하면 ARN이 같아도 액세스 항목이 작동하지 않습니다. 다시 생성된 IAM 보안 주체에서 ARN이 동일하더라도 `roleID` 또는 `userID`(`aws sts get-caller-identity` AWS CLI 명령으로 확인할 수 있음)는 다시 생성된 IAM 보안 주체의 해당 항목과 다릅니다(원래 IAM 보안 주체의 항목임). 액세스 항목의 IAM 보안 주체 `roleID` 또는 `userID`가 표시되지 않더라도 Amazon EKS는 액세스 항목과 함께 이를 저장합니다.

- 각 액세스 항목에는 유형이 있습니다. `EC2 Linux`(Linux 또는 Bottlerocket 자체 관리형 노드에 사용되는 IAM 역할의 경우), `EC2 Windows`(Windows 자체 관리형 노드에 사용되는 IAM 역할의 경우), `FARGATE_LINUX`(AWS Fargate (Fargate)에서 사용되는 IAM 역할의 경우) 또는 `STANDARD`를 유형으로 지정할 수 있습니다. 유형을 지정하지 않으면 Amazon EKS에서 자동으로 유형을 `STANDARD`로 설정합니다. Amazon EKS는 클러스터의 플랫폼 버전에 상관없이 `aws-auth ConfigMap`에 해당 역할의 항목을 추가하기 때문에 관리형 노드 그룹 또는 Fargate 프로파일에 사용되는 IAM 역할에 대한 액세스 항목을 생성하지 않아도 됩니다.

  액세스 항목을 생성한 후에는 유형을 변경할 수 없습니다.
- 액세스 항목 유형이 `STANDARD`인 경우 액세스 항목의 사용자 이름을 지정할 수 있습니다. 사용자 이름 값을 지정하지 않는 경우 Amazon EKS는 지정한 IAM 보안 주체(IAM 역할 또는 IAM 사용자) 및 액세스 항목의 유형에 따라 다음 값 중 하나를 자동으로 설정합니다. 고유한 사용자 이름을 지정해야 하는 특별한 이유가 없는 한, 사용자 이름을 지정하지 말고 Amazon EKS에서 자동으로 생성하게 두는 것이 좋습니다. 고유한 사용자 이름을 지정하는 경우:
  - `system:`, `eks:`, `aws:`, `amazon:` 또는 `iam:`으로 시작할 수 없습니다.
  - IAM 역할에 대한 사용자 이름인 경우 사용자 이름 끝에 `{{SessionName}}`을 추가하는 것이 좋습니다. 사용자 이름에 `{{SessionName}}`을 추가하는 경우 사용자 이름에서 {{SessionName}} 앞에 콜론을 포함해야 합니다. 이 역할을 수임하면 역할을 수임할 때 지정한 세션 이름이 클러스

터에 자동으로 전달되고 CloudTrail 로그에 표시됩니다. 예를 들어 john{{SessionName}}의 사용자 이름을 포함할 수 없습니다. 사용자 이름은 :john{{SessionName}} 또는 jo:hn{{SessionName}}이어야 합니다. 콜론은 {{SessionName}} 앞에 와야 합니다. 다음 테이블에서 Amazon EKS가 생성한 사용자 이름에는 ARN이 포함됩니다. ARN에는 콜론이 포함되므로 이 요구 사항을 충족합니다. 사용자 이름에 {{SessionName}}을 포함하지 않는 경우 콜론은 필요하지 않습니다.

IAM 보안 주체 유형	유형	Amazon EKS가 자동으로 설정하는 사용자 이름 값
User	STANDARD	사용자의 ARN. 예제: arn:aws:iam:: *111122223333* :user/*my-user*
역할	STANDARD	역할을 수임한 경우 STS ARN. Amazon EKS가 역할에 {{SessionName}} 을 추가합니다. 예제: arn:aws:sts:: *111122223333* :assumed-role/ *my-role*/{{SessionName}} 지정한 역할의 ARN에 경로가 포함된 경우 Amazon EKS는 생성된 사용자 이름에서 해당 경로를 제거합니다.
역할	EC2 Linux 또는 EC2 Windows	system:node:{{EC2PrivateDNSName}}
역할	FARGATE_LINUX	system:node:{{SessionName}}

액세스 항목을 생성한 후 사용자 이름을 변경할 수 있습니다.

- 액세스 항목 유형이 `STANDARD`이고 Kubernetes RBAC 권한 부여를 사용하려는 경우 액세스 항목에 하나 이상의 그룹 이름을 추가할 수 있습니다. 액세스 항목을 생성한 후 그룹 이름을 추가 및 제거할 수 있습니다. IAM 보안 주체가 클러스터의 Kubernetes 객체에 액세스할 수 있으려면 Kubernetes 역할 기반 권한 부여(RBAC) 객체를 생성하고 관리해야 합니다. 클러스터에서 `kind: Group`에 대해 그룹 이름을 `subject`로 지정하는 Kubernetes `RoleBinding` 또는 `ClusterRoleBinding` 객체를 생성합니다. Kubernetes에서는 Kubernetes `Role` 또는 `ClusterRole` 객체에 지정한 클러스터 객체(바인딩의 `roleRef`에서도 지정함)에 대한 액세스 권한을 IAM 보안 주체에 부여합니다. 그룹 이름을 지정하는 경우 Kubernetes 역할 기반 인증(RBAC) 객체에 익숙해지는 것이 좋습니다. 자세한 내용은 Kubernetes 설명서의 RBAC 승인 사용을 참조하세요.

  > **Important**
  >
  > Amazon EKS는 클러스터에 있는 Kubernetes RBAC 객체에 지정한 그룹 이름이 포함되었는지 확인하지 않습니다.

  Kubernetes에서 클러스터의 Kubernetes 객체에 대한 액세스 권한을 IAM 보안 주체에 부여하는 대신 또는 이 방법 외에도 Amazon EKS 액세스 정책을 액세스 항목에 연결할 수 있습니다. Amazon EKS는 IAM 보안 주체에 액세스 정책의 권한과 함께 클러스터의 Kubernetes 객체에 액세스할 수 있는 권한을 부여합니다. 지정한 Kubernetes 네임스페이스로 액세스 정책의 권한 범위를 지정할 수 있습니다. 액세스 정책을 사용할 경우 Kubernetes RBAC 객체를 관리할 필요가 없습니다. 자세한 내용은 액세스 항목에 대한 액세스 정책 연결 및 연결 해제 단원을 참조하십시오.
- 유형이 `EC2 Linux` 또는 `EC2 Windows`인 액세스 항목을 생성하는 경우 액세스 항목을 생성하는 IAM 보안 주체에 `iam:PassRole` 권한이 있어야 합니다. 자세한 내용은 IAM 사용 설명서에서 사용자에게 AWS 서비스에 역할을 전달할 권한 부여를 참조하세요.
- 표준 IAM 동작과 마찬가지로, 액세스 항목 생성 및 업데이트는 실질적으로 일관되며 초기 API 직접 호출이 성공적으로 반환된 후 효력이 발생하는 데 몇 초가 걸릴 수 있습니다. 이러한 잠재적 지연을 고려하도록 애플리케이션을 설계해야 합니다. 액세스 항목 생성 또는 업데이트는 애플리케이션의 중요한 고가용성 코드 경로에 포함하지 않는 것이 좋습니다. 대신 자주 실행하지 않는 별도의 초기화 루틴이나 설정 루틴에서 을 변경하십시오. 또한 프로덕션 워크플로우에서 변경 사항을 적용하기 전에 변경 사항이 전파되었는지 확인하세요.
- 액세스 항목은 서비스 연결 역할을 지원하지 않습니다. 보안 주체 ARN이 서비스 연결 역할인 경우 액세스 항목을 생성할 수 없습니다. `arn:aws:iam::*:role/aws-service-role/*` 형식의 ARN으로 서비스 연결 역할을 식별할 수 있습니다.

AWS Management Console 또는 AWS CLI를 사용하여 액세스 항목을 생성할 수 있습니다.

AWS Management Console

액세스 항목을 생성하는 방법

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 관리형 노드 그룹을 생성하려는 클러스터의 이름을 선택합니다.
3. 액세스 탭을 선택합니다.
4. 액세스 항목 생성을 선택합니다.
5. IAM 보안 주체의 경우 기존 IAM 역할 또는 사용자를 선택합니다. IAM 모범 사례에서는 장기 보안 인증 정보를 보유한 IAM 사용자 대신, 단기 보안 인증 정보를 보유한 IAM 역할을 사용하여 클러스터에 액세스하도록 권장합니다. 자세한 내용은 IAM 사용 설명서에서 임시 보안 인증을 사용하여 AWS에 액세스하려면 인간 사용자가 ID 공급자와의 페더레이션을 사용하도록 요구합니다.를 참조하세요.
6. 자체 관리형 Amazon EC2 노드에 사용되는 노드 역할에 대한 액세스 항목인 경우 유형에서 EC2 Linux 또는 EC2 Windows를 선택합니다. 그렇지 않으면 기본값(표준)을 그대로 수락합니다.
7. 선택한 유형이 표준이고 사용자 이름을 지정하려면 사용자 이름을 입력합니다.
8. 선택한 유형이 표준이고 IAM 보안 주체에 대해 Kubernetes RBAC 권한 부여를 사용하려는 경우 그룹에 대한 하나 이상의 이름을 지정합니다. 그룹 이름을 지정하지 않고 Amazon EKS 권한 부여를 사용하려는 경우 이후 단계에서 또는 액세스 항목을 생성한 후에 액세스 정책을 연결할 수 있습니다.
9. (선택 사항) 태그에서 액세스 항목에 레이블을 할당합니다. 예를 들어 태그가 동일한 모든 리소스를 더 쉽게 찾을 수 있습니다.
10. 다음을 선택합니다.
11. 액세스 정책 추가 페이지에서 선택한 유형이 표준이고 Amazon EKS가 IAM 보안 주체에 클러스터의 Kubernetes 객체에 대한 권한을 부여하게 하려면 다음 단계를 완료합니다. 그렇지 않은 경우 [Next]를 선택합니다.

    a. 정책 이름에서 액세스 정책을 선택합니다. 액세스 정책의 권한은 볼 수 없지만 Kubernetes 사용자 대면 `ClusterRole` 객체에서 이와 유사한 권한을 포함합니다. 자세한 내용은 Kubernetes 설명서에서 User-facing roles를 참조하세요.

    b. 다음 옵션 중 하나를 선택합니다:

    - 클러스터 - Amazon EKS에서 IAM 보안 주체가 클러스터의 모든 Kubernetes 객체에 대한 액세스 정책의 권한을 갖도록 권한을 부여하려면 이 옵션을 선택합니다.

- Kubernetes 네임스페이스 - Amazon EKS에서 IAM 보안 주체가 클러스터 내 특정 Kubernetes 네임스페이스의 모든 Kubernetes 객체에 대한 액세스 정책의 권한을 갖도록 권한을 부여하려면 이 옵션을 선택합니다. 네임스페이스의 경우 클러스터의 Kubernetes 네임스페이스 이름을 입력합니다. 네임스페이스를 더 추가하려면 새 네임스페이스 추가를 선택하고 네임스페이스 이름을 입력합니다.

c. 정책을 더 추가하려면 정책 추가를 선택합니다. 각 정책의 범위를 다르게 지정할 수 있지만 각 정책은 한 번만 추가할 수 있습니다.

d. 다음을 선택합니다.

12. 액세스 항목의 구성을 검토합니다. 문제가 있는 것 같으면 이전을 선택하여 이전 단계로 돌아가서 오류를 수정합니다. 구성이 올바르면 생성을 선택합니다.

AWS CLI

전제 조건

디바이스 또는 AWS CloudShell에 설치 및 구성된 AWS CLI v1의 최신 버전. AWS CLI v2는 한동안 새 기능을 지원하지 않습니다. `aws --version | cut -d / -f2 | cut -d ' ' -f1`을 사용하여 현재 버전을 확인할 수 있습니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 `aws configure`를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.

액세스 항목을 생성하는 방법

다음 예제를 사용하여 액세스 항목을 생성할 수 있습니다.

- 자체 관리형 Amazon EC2 Linux 노드 그룹에 대한 액세스 항목을 생성합니다. *my-cluster*를 클러스터 이름으로, *111122223333*을 AWS 계정 ID로, *EKS-my-cluster-self-managed-ng-1*을 노드 IAM 역할로 바꿉니다. 노드 그룹이 Windows 노드 그룹인 경우 *EC2_Linux*를 **EC2_Windows**로 바꿉니다.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/EKS-my-cluster-self-managed-ng-1 --type EC2_Linux
```

STANDARD 이외의 유형을 지정할 때는 이 `--kubernetes-groups` 옵션을 사용할 수 없습니다. 유형이 STANDARD 이외의 값이므로 액세스 정책을 이 액세스 항목에 연결할 수 없습니다.

- Kubernetes에서 클러스터에 대한 액세스를 부여하려는 Amazon EC2 자체 관리형 노드 그룹에 사용되지 않는 IAM 역할을 허용하는 액세스 항목을 생성합니다. *my-cluster*를 클러스터 이름으로, *111122223333*을 AWS 계정 ID로, *my-role*을 IAM 역할 이름으로 바꿉니다. *Viewers*를 클러스터의 Kubernetes `RoleBinding` 또는 `ClusterRoleBinding` 객체에 지정한 그룹 이름으로 바꿉니다.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role --type STANDARD --user Viewers --
kubernetes-groups Viewers
```

- 클러스터에서 IAM 사용자를 인증할 수 있는 액세스 항목을 생성합니다. IAM 모범 사례에서는 장기 보안 인증 정보를 보유한 IAM 사용자 대신, 단기 보안 인증 정보를 보유한 IAM 역할을 사용하여 클러스터에 액세스하도록 권장하지만, 이 예제는 가능한 상황을 보여줍니다. 자세한 내용은 IAM 사용 설명서에서 임시 보안 인증을 사용하여 AWS에 액세스하려면 인간 사용자가 ID 공급자와의 페더레이션을 사용하도록 요구합니다.를 참조하세요.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:user/my-user --type STANDARD --username my-user
```

  이 사용자에게 Kubernetes API 검색 역할의 권한보다 더 많은 액세스 권한을 부여하려는 경우 `--kubernetes-groups` 옵션이 사용되지 않으므로 액세스 정책을 액세스 항목에 연결해야 합니다. 자세한 내용은 Kubernetes 설명서에서 액세스 항목에 대한 액세스 정책 연결 및 연결 해제 및 API discovery roles를 참조하세요.

액세스 항목 업데이트

AWS Management Console 또는 AWS CLI를 사용하여 액세스 항목을 업데이트할 수 있습니다.

AWS Management Console

액세스 항목을 업데이트하는 방법

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 관리형 노드 그룹을 생성하려는 클러스터의 이름을 선택합니다.
3. 액세스 탭을 선택합니다.
4. 업데이트할 액세스 항목을 선택합니다.
5. 편집을 선택합니다.

6. 사용자 이름에서 기존 값을 변경할 수 있습니다.
7. 그룹에서 기존 그룹 이름을 제거하거나 새 그룹 이름을 추가할 수 있습니다. system:nodes 또는 system:bootstrappers와 같은 그룹 이름이 있는 경우에는 제거하지 않습니다. 이러한 그룹을 제거하면 클러스터가 제대로 작동하지 않을 수 있습니다. 그룹 이름을 지정하지 않고 Amazon EKS 권한 부여를 사용하려면 이후 단계에서 액세스 정책을 연결합니다.
8. 태그에서 액세스 항목에 레이블을 할당할 수 있습니다. 예를 들어 태그가 동일한 모든 리소스를 더 쉽게 찾을 수 있습니다. 기존 태그를 제거할 수도 있습니다.
9. Save changes(변경 사항 저장)를 선택합니다.
10. 액세스 정책을 항목에 연결하려면 액세스 항목에 대한 액세스 정책 연결 및 연결 해제 섹션을 참조하세요.

AWS CLI

전제 조건

AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.

액세스 항목을 업데이트하는 방법

*`my-cluster`*를 클러스터 이름으로, *`111122223333`*을 AWS 계정 ID로, *`EKS-my-cluster-my-namespace-Viewers`*를 IAM 역할 이름으로 바꿉니다.

```
aws eks update-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers --kubernetes-
groups Viewers
```

액세스 항목이 `STANDARD` 이외의 값인 경우 `--kubernetes-groups` 옵션을 사용할 수 없습니다. 또한 `STANDARD` 이외의 유형인 액세스 항목에 액세스 정책을 연결할 수 없습니다.

액세스 항목 삭제

액세스 항목을 잘못 삭제했다면 언제든지 다시 생성할 수 있습니다. 삭제하려는 액세스 항목이 액세스 정책과 연결된 경우 연결은 자동으로 삭제됩니다. 액세스 항목을 삭제하기 전에 액세스 항목에서 액세스 정책을 연결 해제하지 않아도 됩니다.

AWS Management Console 또는 AWS CLI를 사용하여 액세스 항목을 삭제할 수 있습니다.

AWS Management Console

액세스 항목을 삭제하는 방법

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 액세스 항목을 삭제하려는 클러스터의 이름을 선택합니다.
3. 액세스 탭을 선택합니다.
4. 액세스 항목 목록에서 삭제할 액세스 항목을 선택합니다.
5. 삭제를 선택합니다.
6. 확인 대화 상자에서 Delete(삭제)를 선택합니다.

AWS CLI

전제 조건

AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.

액세스 항목을 삭제하는 방법

*`my-cluster`*를 클러스터 이름으로, *`111122223333`*을 AWS 계정 ID로, *`my-role`*을 클러스터에 대한 액세스 권한을 제거할 IAM 역할 이름으로 바꿉니다.

```
aws eks delete-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role
```

액세스 항목에 대한 액세스 정책 연결 및 연결 해제

STANDARD 유형의 액세스 항목에 하나 이상의 액세스 정책을 할당할 수 있습니다. Amazon EKS는 클러스터에서 제대로 작동하는 데 필요한 권한을 다른 유형의 액세스 항목에 자동으로 부여합니다. Amazon EKS 액세스 정책에는 IAM 권한이 아니라 Kubernetes 권한이 포함됩니다. 액세스 정책을 액세스 항목에 연결하기 전에 각 액세스 정책에 포함된 Kubernetes 권한을 친숙해야 합니다. 자세한 내용은 액세스 정책 권한 단원을 참조하십시오. 요구 사항을 충족하는 액세스 정책이 없는 경우 액세스 정책을 액세스 항목에 연결하지 않습니다. 대신 액세스 항목에 대해 하나 이상의 그룹 이름을 지정하고 Kubernetes 역할 기반 액세스 제어 객체를 생성하고 관리합니다. 자세한 내용은 액세스 항목 생성 단원을 참조하십시오.

사전 조건

- 기존 액세스 항목. 파일을 만들려면 액세스 항목 생성 섹션을 참조하세요.
- `ListAccessEntries`, `DescribeAccessEntry`, `UpdateAccessEntry`, `ListAccessPolicies`, `AssociateAccessPolicy`, `DisassociateAccessPolicy` 권한을 보유한 AWS Identity and Access Management 역할 또는 사용자. 자세한 내용은 서비스 권한 부여 참조에서 Amazon Elastic Kubernetes Service에서 정의한 작업을 참조하세요.

액세스 정책을 액세스 항목에 연결하기 전에 다음 요구 사항을 고려하세요.

- 각 액세스 항목에 여러 액세스 정책을 연결할 수 있지만 각 정책은 액세스 항목에 한 번만 연결할 수 있습니다. 여러 액세스 정책을 연결하는 경우 액세스 항목의 IAM 보안 주체는 연결된 모든 액세스 정책에 모든 권한을 포함합니다.
- 클러스터의 모든 리소스에 대한 액세스 정책 범위를 지정하거나 하나 이상의 Kubernetes 네임스페이스 이름을 지정하여 범위를 지정할 수 있습니다. 네임스페이스 이름에 와일드카드 문자를 사용할 수 있습니다. 예를 들어 `dev-`로 시작하는 모든 네임스페이스로 액세스 정책 범위를 지정하려는 경우 네임스페이스 이름으로 `dev-*`를 지정할 수 있습니다. 클러스터에 네임스페이스가 존재하고 철자가 클러스터의 실제 네임스페이스 이름과 일치하는지 확인합니다. Amazon EKS는 클러스터에 있는 네임스페이스의 철자나 존재 여부를 확인하지 않습니다.
- 액세스 항목에 연결한 후에 액세스 정책의 액세스 범위를 변경할 수 있습니다. 액세스 정책의 범위를 Kubernetes 네임스페이스로 지정한 경우 필요에 따라 연결을 위해 네임스페이스를 추가 및 제거할 수 있습니다.
- 그룹 이름도 지정된 액세스 항목에 액세스 정책을 연결하는 경우 IAM 보안 주체는 연결된 모든 액세스 정책에 모든 권한을 보유합니다. 또한 그룹 이름을 지정하는 Kubernetes `Role` 및 `RoleBinding` 객체에 지정된 Kubernetes `Role` 또는 `ClusterRole` 객체에 모든 권한을 보유합니다.

- `kubectl auth can-i --list` 명령을 실행하면 명령을 실행할 때 사용한 IAM 보안 주체의 액세스 항목에 연결된 액세스 정책에서 할당한 Kubernetes 권한이 표시되지 않습니다. 이 명령은 액세스 항목에 지정한 그룹 이름 또는 사용자 이름에 바인딩하는 Kubernetes `Role` 또는 `ClusterRole` 객체에 Kubernetes 권한을 부여하는 경우에만 권한을 표시합니다.
- `kubectl` 명령을 **`--as`** ***`username`*** 또는 **`--as-group`** ***`group-name`***과 함께 사용하는 등 클러스터에서 Kubernetes 객체와 상호 작용할 때 Kubernetes 사용자 또는 그룹을 가장하면 Kubernetes RBAC 인증을 강제로 사용합니다. 따라서 IAM 보안 주체는 액세스 항목에 연결된 액세스 정책에서 할당한 권한을 보유하지 않습니다. IAM 보안 주체가 가장하는 사용자 또는 그룹에 있는 유일한 Kubernetes 권한은 해당 그룹 이름 또는 사용자 이름에 바인딩하는 Kubernetes `Role` 또는 `ClusterRole` 객체에 부여한 Kubernetes 권한입니다. IAM 보안 주체가 연결된 액세스 정책의 권한을 보유하려면 Kubernetes 사용자 또는 그룹을 가장하지 않습니다. 또한 IAM 보안 주체는 액세스 항목에 지정한 그룹 이름 또는 사용자 이름에 바인딩하는 Kubernetes `Role` 또는 `ClusterRole` 객체에 부여한 권한도 계속 보유합니다. 자세한 내용은 Kubernetes 설명서에서 User impersonation을 참조하세요.

AWS Management Console 또는 AWS CLI를 사용하여 액세스 정책을 액세스 항목에 연결할 수 있습니다.

AWS Management Console

AWS Management Console을 사용하여 액세스 정책을 액세스 항목에 연결하는 방법

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 액세스 정책을 연결할 액세스 항목이 있는 클러스터 이름을 선택합니다.
3. 액세스 탭을 선택합니다.
4. 액세스 항목 유형이 표준인 경우 Amazon EKS 액세스 정책을 연결하거나 연결을 해제할 수 있습니다. 액세스 항목 유형이 표준 이외의 항목인 경우 이 옵션을 사용할 수 없습니다.
5. 액세스 정책 연결을 선택합니다.
6. 정책 이름에서 IAM 보안 주체에 부여하려는 권한이 있는 정책을 선택합니다. 이러한 권한이 포함된 정책을 보려면 액세스 정책 권한 섹션을 참조하세요.
7. 액세스 범위에서 액세스 범위를 선택합니다. 클러스터를 선택하면 액세스 정책의 권한이 모든 Kubernetes 네임스페이스의 리소스에 대한 IAM 보안 주체에 부여됩니다. Kubernetes 네임스페이스를 선택한 경우 새 네임스페이스 추가를 선택할 수 있습니다. 나타나는 네임스페이스 필드에 클러스터의 Kubernetes 네임스페이스 이름을 입력할 수 있습니다. IAM 보안 주체가 여러 네임스페이스에 대한 권한을 보유하게 하려면 네임스페이스를 여러 개 입력하면 됩니다.

8. 액세스 정책 추가를 선택합니다.

AWS CLI

전제 조건

AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.

액세스 정책을 액세스 항목에 연결하는 방법

1. 사용 가능한 액세스 정책을 봅니다.

```
aws eks list-access-policies --output table
```

예제 출력은 다음과 같습니다.

```
-------------------------------------------------------------------------------------
|                                        ListAccessPolicies                         |
|                                                                                   |
+-----------------------------------------------------------------------------------+
||                                      accessPolicies                             ||
|+---------------------------------------------------------------------------------+|
||                                  arn                                            ||
||  name                                                                           ||
|+---------------------------------------------------------------------------------+|
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy                    ||
||  AmazonEKSAdminPolicy                                                          ||
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy             ||
||  AmazonEKSClusterAdminPolicy                                                    ||
```

```
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy         |
 AmazonEKSEditPolicy          ||
||  arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy         |
 AmazonEKSViewPolicy          ||
|+-------------------------------------------------------------------
+------------------------------+|
```

이러한 권한이 포함된 정책을 보려면 액세스 정책 권한 섹션을 참조하세요.

2. 기존 액세스 항목을 확인합니다. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다.

```
aws eks list-access-entries --cluster-name my-cluster
```

예제 출력은 다음과 같습니다.

```
{
    "accessEntries": [
        "arn:aws:iam::111122223333:role/my-role",
        "arn:aws:iam::111122223333:user/my-user"
    ]
}
```

3. 액세스 정책을 액세스 항목에 연결합니다. 다음 예제에서는 AmazonEKSViewPolicy 액세스 정책을 액세스 항목에 연결합니다. *my-role* IAM 역할이 클러스터의 Kubernetes 객체에 액세스를 시도할 때마다 Amazon EKS는 정책에 있는 권한을 사용하여 *my-namespace1* 및 *my-namespace2* Kubernetes 네임스페이스의 Kubernetes 객체에만 액세스할 수 있는 권한을 역할에 부여합니다. *my-cluster*를 클러스터 이름으로, *111122223333*을 AWS 계정 ID로, *my-role*을 Amazon EKS가 Kubernetes 클러스터 객체에 대한 액세스를 승인하려는 IAM 역할의 이름으로 바꿉니다.

```
aws eks associate-access-policy --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role \
    --access-scope type=namespace,namespaces=my-namespace1,my-namespace2 --
policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy
```

IAM 보안 주체가 클러스터 전체 권한을 갖게 하려면 **type=namespace,namespaces=*my-namespace1*,*my-namespace2***를 **type=cluster**로 바꿉니다. 여러 액세스 정책을 액세스 항목에 연결하려면 각각 고유한 액세스 정책을 사용하여 명령을 여러 번 실행합니다. 연결된 각 액세스 정책의 범위는 고유합니다.

> **Note**
>
> 나중에 연결된 액세스 정책의 범위를 변경하려면 새 범위를 사용하여 이전 명령을 다시 실행합니다. 예를 들어 *my-namespace2*를 제거하려는 경우 **type=namespace,namespaces=*my-namespace1***만 사용하여 명령을 다시 실행합니다. 범위를 **namespace**에서 **cluster**로 변경하려면 **type=cluster**를 사용하여 다시 명령을 실행해 **type=namespace,namespaces=*my-namespace1*,*my-namespace2***를 제거합니다.

액세스 항목에서 액세스 정책 연결을 해제하는 방법

1. 액세스 항목에 연결된 액세스 정책을 결정합니다.

```
aws eks list-associated-access-policies --cluster-name my-cluster --principal-
arn arn:aws:iam::111122223333:role/my-role
```

예제 출력은 다음과 같습니다.

```
{
    "clusterName": "my-cluster",
    "principalArn": "arn:aws:iam::111122223333",
    "associatedAccessPolicies": [
        {
            "policyArn": "arn:aws:eks::aws:cluster-access-
policy/AmazonEKSViewPolicy",
            "accessScope": {
                "type": "cluster",
                "namespaces": []
            },
            "associatedAt": "2023-04-17T15:25:21.675000-04:00",
            "modifiedAt": "2023-04-17T15:25:21.675000-04:00"
        },
        {
            "policyArn": "arn:aws:eks::aws:cluster-access-
policy/AmazonEKSAdminPolicy",
            "accessScope": {
                "type": "namespace",
                "namespaces": [
                    "my-namespace1",
```

```
                    "my-namespace2"
                ]
            },
            "associatedAt": "2023-04-17T15:02:06.511000-04:00",
            "modifiedAt": "2023-04-17T15:02:06.511000-04:00"
        }
    ]
}
```

이전 예제에서 이 액세스 항목의 IAM 보안 주체는 클러스터의 모든 네임스페이스에 대한 보기 권한과 두 Kubernetes 네임스페이스에 대한 관리자 권한을 보유합니다.

2. 액세스 항목에서 액세스 정책 연결을 해제합니다. 이 예제에서 AmazonEKSAdminPolicy 정책은 액세스 항목과 연결이 해제되었습니다. 하지만 IAM 보안 주체는 *my-namespace1* 및 *my-namespace2* 네임스페이스의 객체에 대한 AmazonEKSViewPolicy 액세스 정책에 있는 권한을 보유합니다. 해당 액세스 정책이 액세스 항목과 연결 해제되지 않았기 때문입니다.

```
aws eks disassociate-access-policy --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/my-role \
    --policy-arn arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy
```

액세스 정책 권한

액세스 정책에는 Kubernetes verbs(권한) 및 resources가 포함된 rules가 있습니다. 액세스 정책에는 IAM 권한 또는 리소스가 없습니다. Kubernetes Role 및 ClusterRole 객체와 마찬가지로 액세스 정책에는 allow rules만 포함됩니다. 액세스 정책의 콘텐츠는 수정할 수 없습니다. 자체 액세스 정책을 생성할 수 없습니다. 액세스 정책의 권한이 요구 사항에 맞지 않는 경우 Kubernetes RBAC 객체를 생성하고 액세스 항목의 그룹 이름을 지정합니다. 자세한 내용은 액세스 항목 생성 단원을 참조하십시오. 액세스 정책에 포함된 권한은 Kubernetes 사용자 대면 클러스터 역할의 권한과 유사합니다. 자세한 내용은 Kubernetes 설명서에서 User-facing roles를 참조하세요.

해당 콘텐츠를 보려는 액세스 정책을 선택합니다. 각 액세스 정책에 있는 각 테이블의 각 행은 별도의 규칙입니다.

AmazonEKSAdminPolicy

이 액세스 정책에는 리소스에 대한 대부분의 권한을 IAM 보안 주체에 부여하는 권한이 포함됩니다. 액세스 항목에 연결된 경우 액세스 범위는 일반적으로 하나 이상의 Kubernetes 네임스페이스입니다. IAM 보안 주체가 클러스터의 모든 리소스에 대한 관리자 액세스 권한을 보유하려면 대신 AmazonEKSClusterAdminPolicy 액세스 정책을 액세스 항목에 연결합니다.

ARN - arn:aws:eks::aws:cluster-access-policy/AmazonEKSAdminPolicy

Kubernetes API 그룹	Kubernetes 리소스	Kubernetes 동사(권한)
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
authorization.k8s.io	localsubjectaccessreviews	create
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
batch	cronjobs, jobs	create, delete, deletecollection , patch, update

Kubernetes API 그룹	Kubernetes 리소스	Kubernetes 동사(권한)
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update

Kubernetes API 그룹	Kubernetes 리소스	Kubernetes 동사(권한)
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
rbac.authorization.k8s.io	rolebindings , roles	create, delete, deletecollection , get, list, patch, update, watch
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get,list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update

Kubernetes API 그룹	Kubernetes 리소스	Kubernetes 동사(권한)
	pods, pods/attach , pods/exec , pods/port forward , pods/proxy	create, delete, deletecollection , patch, update
	serviceaccounts	impersonate
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch
	namespaces	get,list, watch

AmazonEKSClusterAdminPolicy

이 액세스 정책에는 클러스터에 대한 액세스 권한을 IAM 보안 주체 관리자에게 부여하는 권한이 포함됩니다. 액세스 항목에 연결된 경우 액세스 범위는 일반적으로 Kubernetes 네임스페이스가 아닌 클러스터입니다. IAM 보안 주체의 관리 범위를 더 제한하려면 대신 AmazonEKSAdminPolicy 액세스 정책을 액세스 항목에 연결하는 방법을 고려합니다.

ARN - arn:aws:eks::aws:cluster-access-policy/AmazonEKSClusterAdminPolicy

Kubernetes API 그룹	Kubernetes nonResourceURLs	Kubernetes 리소스	Kubernetes 동사(권한)
*		*	*
	*		*

AmazonEKSEditPolicy

이 액세스 정책에는 IAM 보안 주체가 대부분의 Kubernetes 리소스를 편집할 수 있는 권한이 포함되어 있습니다.

ARN - `arn:aws:eks::aws:cluster-access-policy/AmazonEKSEditPolicy`

Kubernetes API 그룹	Kubernetes 리소스	Kubernetes 동사(권한)
apps	daemonsets , deployments , deployments/rollback , deployments/scale , replicasets , replicasets/scale, statefulsets , statefulsets/scale	create, delete, deletecollection , patch, update
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicasets/scale , replicasets/status , statefulsets , statefulsets/scale , statefulsets/status	get, list, watch
autoscaling	horizontalpodautoscalers , horizontalpodautoscalers/status	get, list, watch
autoscaling	horizontalpodautoscalers	create, delete, deletecollection , patch, update

Kubernetes API 그룹	Kubernetes 리소스	Kubernetes 동사(권한)
batch	cronjobs, jobs	create, delete, deletecollection , patch, update
batch	cronjobs, cronjobs/status , jobs, jobs/status	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , deployments , deployments/rollback , deployments/scale , ingresses , networkpolicies , replicasets , replicasets/scale , replicationcontrollers/scale	create, delete, deletecollection , patch, update
extensions	daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , ingresses , ingresses/status , networkpolicies , replicasets , replicasets/scale , replicasets/status , replicationcontrollers/scale	get, list, watch
networking.k8s.io	ingresses , networkpolicies	create, delete, deletecollection , patch, update
networking.k8s.io	ingresses , ingresses/status , networkpolicies	get, list, watch

Kubernetes API 그룹	Kubernetes 리소스	Kubernetes 동사(권한)
policy	poddisruptionbudgets	create, delete, deletecollection , patch, update
policy	poddisruptionbudgets , poddisruptionbudgets/status	get, list, watch
	namespaces	get, list, watch
	pods/attach , pods/exec , pods/portforward , pods/proxy , secrets, services/proxy	get, list, watch
	serviceaccounts	impersonate
	pods, pods/attach , pods/exec , pods/portforward , pods/proxy	create, delete, deletecollection , patch, update
	configmaps , events, persistentvolumeclaims , replicationcontrollers , replicationcontrollers/scale , secrets, serviceaccounts , services, services/proxy	create, delete, deletecollection , patch, update

Kubernetes API 그룹	Kubernetes 리소스	Kubernetes 동사(권한)
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch

AmazonEKSViewPolicy

이 액세스 정책에는 IAM 보안 주체가 대부분의 Kubernetes 리소스를 볼 수 있는 권한이 포함되어 있습니다.

ARN - arn:aws:eks::aws:cluster-access-policy/AmazonEKSViewPolicy

Kubernetes API 그룹	Kubernetes 리소스	Kubernetes 동사(권한)
apps	controllerrevisions , daemonsets , daemonsets/status , deployments , deployments/scale , deployments/status , replicasets , replicase	get, list, watch

Kubernetes API 그룹	Kubernetes 리소스	Kubernetes 동사(권한)
	ts/scale , replicaset ts/status , statefuls ets , statefulsets/ scale , statefulsets/ status	
autoscaling	horizontalpodautos calers , horizonta lpodautoscalers/st atus	get, list, watch
batch	cronjobs, cronjobs/ status , jobs, jobs/stat us	get, list, watch
discovery.k8s.io	endpointslices	get, list, watch
extensions	daemonsets , daemonset s/status , deploymen ts , deployments/scale , deployments/status , ingresses , ingresses /status , networkpo licies , replicasets , replicasets/scale , replicasets/status , replicationcontrol lers/scale	get, list, watch
networking.k8s.io	ingresses , ingresses /status , networkpo licies	get, list, watch
policy	poddisruptionbudgets , poddisruptionbudge ts/status	get, list, watch

Kubernetes API 그룹	Kubernetes 리소스	Kubernetes 동사(권한)
	configmaps , endpoints , persistentvolumeclaims , persistentvolumeclaims/status , pods, replicationcontrollers , replicationcontrollers/scale , serviceaccounts , services, services/status	get, list, watch
	bindings, events, limitranges , namespaces/status , pods/log, pods/status , replicationcontrollers/status , resourcequotas , resourcequotas/status	get, list, watch
	namespaces	get, list, watch

액세스 정책 업데이트

액세스 정책이 도입된 이후 액세스 정책에 대한 업데이트 세부 정보를 확인합니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 Amazon EKS 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
액세스 정책이 도입되었습니다.	Amazon EKS에서 액세스 정책을 도입했습니다.	2023년 5월 29일

기존 aws-auth ConfigMap 항목을 액세스 항목으로 마이그레이션

클러스터의 aws-auth ConfigMap에 항목을 추가한 경우 aws-auth ConfigMap의 기존 항목에 대한 액세스 항목을 생성하는 것이 좋습니다. 액세스 항목을 생성한 후 해당 항목을 ConfigMap에서 제거할 수 있습니다. aws-auth ConfigMap의 항목에 액세스 정책을 연결할 수 없습니다. 액세스 정책을 IAM 보안 주체에 연결하려면 액세스 항목을 생성합니다.

> **Important**
>
> 클러스터에 관리형 노드 그룹 또는 Fargate 프로파일을 추가할 때 Amazon EKS에서 생성한 기존 aws-auth ConfigMap 항목을 제거하지 않습니다. Amazon EKS가 ConfigMap에서 생성한 항목을 제거하면 클러스터가 제대로 작동하지 않습니다. 하지만 자체 관리형 노드 그룹에 대한 액세스 항목을 생성한 후에는 해당 항목을 제거할 수 있습니다.

필수 조건

- 액세스 항목 및 액세스 정책에 익숙해야 합니다. 자세한 내용은 액세스 항목 관리 및 액세스 항목에 대한 액세스 정책 연결 및 연결 해제 단원을 참조하세요.
- 플랫폼 버전이 Amazon EKS 클러스터의 Kubernetes 객체에 대한 IAM 역할 또는 사용자 액세스 허용 주제의 사전 조건에 나열된 버전 이상인 기존 클러스터.
- 디바이스 또는 0.175.0에 설치된 버전 AWS CloudShell 이상의 eksctl 명령줄 도구. eksctl을 설치 또는 업그레이드하려면 eksctl 설명서에서 Installation을 참조하세요.
- kube-system 네임스페이스에서 aws-auth ConfigMap을 수정할 Kubernetes 권한.
- CreateAccessEntry 및 ListAccessEntries 권한을 보유한 AWS Identity and Access Management 역할 또는 사용자. 자세한 내용은 서비스 권한 부여 참조에서 Amazon Elastic Kubernetes Service에서 정의한 작업을 참조하세요.

항목을 **aws-auth ConfigMap**에서 액세스 항목으로 마이그레이션하는 방법

1. aws-auth ConfigMap에서 기존 항목을 봅니다. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다.

   ```
   eksctl get iamidentitymapping --cluster my-cluster
   ```

   예제 출력은 다음과 같습니다.

```
ARN
              USERNAME                                      GROUPS
                            ACCOUNT
arn:aws:iam::111122223333:role/EKS-my-cluster-Admins
              Admins                                        system:masters
arn:aws:iam::111122223333:role/EKS-my-cluster-my-namespace-Viewers
              my-namespace-Viewers                          Viewers
arn:aws:iam::111122223333:role/EKS-my-cluster-self-managed-ng-1
                              system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws:iam::111122223333:user/my-user
              my-user
arn:aws:iam::111122223333:role/EKS-my-cluster-fargateprofile1
                              system:node:{{SessionName}}
 system:bootstrappers,system:nodes,system:node-proxier
arn:aws:iam::111122223333:role/EKS-my-cluster-managed-ng
                              system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

2. 생성 후 이전 출력에서 반환된 모든 ConfigMap 항목에 대해 액세스 항목을 생성합니다. 액세스 항목을 생성하는 경우 출력에서 반환된 ARN, USERNAME, GROUPS 및 ACCOUNT에 대해 동일한 값을 지정해야 합니다. 예제 출력에서는 Amazon EKS가 Fargate 프로파일 및 관리형 노드 그룹에 대한 액세스 항목을 생성했기 때문에 마지막 두 항목을 제외한 모든 항목에 대한 액세스 항목을 생성합니다.
3. 생성한 모든 액세스 항목에 대한 항목을 ConfigMap에서 삭제합니다. ConfigMap에서 항목을 삭제하지 않으면 IAM 보안 주체 ARN의 액세스 항목 설정이 ConfigMap 항목보다 우선합니다. *111122223333*을 사용자 AWS 계정 ID로, *EKS-my-cluster-my-namespace-Viewers*를 ConfigMap 항목의 역할 이름으로 바꿉니다. IAM 역할이 아닌 IAM 사용자에 대한 항목을 제거하는 경우 **role**을 **user**로, *EKS-my-cluster-my-namespace-Viewers*를 사용자 이름으로 바꿉니다.

```
eksctl delete iamidentitymapping --arn arn:aws:iam::111122223333:role/EKS-my-
cluster-my-namespace-Viewers --cluster my-cluster
```

클러스터에 대한 IAM 보안 주체 액세스 사용

> **Important**
>
> aws-auth ConfigMap은 더 이상 사용되지 않습니다. Kubernetes API에 대한 액세스를 관리하는 권장 방법은 액세스 항목입니다.

IAM 보안 주체를 사용하는 클러스터에 대한 액세스는 Amazon EKS 컨트롤 플레인에서 실행되는 Kubernetes용 AWS IAM Authenticator에 의해 사용 설정됩니다. 인증자는 aws-auth ConfigMap에서 구성 정보를 가져옵니다. 모든 aws-auth ConfigMap 설정은 GitHub의 Full Configuration Format을 참조하세요.

Amazon EKS 클러스터에 IAM 보안 주체 추가

Amazon EKS 클러스터를 생성할 경우 클러스터를 생성하는 IAM 보안 주체에게는 Amazon EKS 제어 영역의 클러스터 역할 기반 액세스 제어(RBAC) 구성에 system:masters 권한이 자동으로 부여됩니다. 이 보안 주체는 표시되는 구성에 나타나지 않으므로 클러스터를 원래 생성한 보안 주체를 추적해야 합니다. 추가 IAM 보안 주체에 클러스터와 상호 작용할 수 있는 기능을 부여하려면 Kubernetes 내에서 aws-auth ConfigMap을 편집하고 aws-auth ConfigMap에 지정하는 group의 이름으로 Kubernetes rolebinding 또는 clusterrolebinding을 생성해야 합니다.

> **Note**
>
> Kubernetes 역할 기반 액세스 제어(RBAC) 구성에 대한 자세한 내용은 Kubernetes 설명서의 RBAC 승인 섹션을 참조하세요.

Amazon EKS 클러스터에 IAM 보안 주체를 추가하려면

1. kubectl이 클러스터에 액세스하는 데 사용하는 자격 증명을 확인합니다. 컴퓨터에서 다음 명령으로 kubectl이 사용하는 자격 증명을 볼 수 있습니다. 기본 경로를 사용하지 않는 경우 ***~/.kube/config***를 kubeconfig 파일의 경로로 바꿉니다.

   ```
   cat ~/.kube/config
   ```

   예제 출력은 다음과 같습니다.

   ```
   [...]
   ```

```
contexts:
- context:
    cluster: my-cluster.region-code.eksctl.io
    user: admin@my-cluster.region-code.eksctl.io
  name: admin@my-cluster.region-code.eksctl.io
current-context: admin@my-cluster.region-code.eksctl.io
[...]
```

이전 예제 출력에서는 *admin*이라는 사용자의 자격 증명이 *my-cluster*라는 클러스터에 대해 구성되었습니다. 클러스터를 생성한 사용자인 경우 클러스터에 대한 액세스 권한이 이미 있습니다. 클러스터를 생성한 사용자가 아닌 경우 다른 IAM 보안 주체에 대한 클러스터 액세스를 사용 설정하기 위해 나머지 단계를 완료해야 합니다. IAM 모범 사례에서는 되도록 사용자 대신 역할에 권한을 부여하세요. 다음 명령을 사용하여 현재 클러스터에 액세스할 수 있는 다른 보안 주체를 확인할 수 있습니다.

```
kubectl describe -n kube-system configmap/aws-auth
```

예제 출력은 다음과 같습니다.

```
Name:         aws-auth
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Data
====
mapRoles:
----
- groups:
  - system:bootstrappers
  - system:nodes
  rolearn: arn:aws:iam::111122223333:role/my-node-role
  username: system:node:{{EC2PrivateDNSName}}


BinaryData
====

Events:  <none>
```

이전 예는 기본 aws-auth ConfigMap입니다. 노드 인스턴스 역할만 클러스터에 액세스할 수 있습니다.

2. IAM 보안 주체를 매핑할 수 있는 기존 Kubernetes roles 및 rolebindings 또는 clusterroles 및 clusterrolebindings가 있어야 합니다. 이런 리소스에 대한 자세한 내용을 알아보려면 Kubernetes 설명서의 RBAC 승인 사용을 참조하세요.

   1. 기존 Kubernetes roles 또는 clusterroles를 확인합니다. Roles는 namespace로 범위가 지정되지만 clusterroles는 클러스터로 범위가 지정됩니다.

      ```
      kubectl get roles -A
      ```

      ```
      kubectl get clusterroles
      ```

   2. 이전 출력에서 반환된 모든 role 또는 clusterrole의 세부 정보를 보고 IAM 보안 주체가 클러스터에서 보유하려는 권한(rules)이 있는지 확인합니다.

      *role-name*을 이전 명령의 출력에서 반환된 role 이름으로 바꿉니다. *kube-system*을 role의 네임스페이스로 바꿉니다.

      ```
      kubectl describe role role-name -n kube-system
      ```

      *cluster-role-name*을 이전 명령의 출력에서 반환된 clusterrole 이름으로 바꿉니다.

      ```
      kubectl describe clusterrole cluster-role-name
      ```

   3. 기존 Kubernetes rolebindings 또는 clusterrolebindings를 확인합니다. Rolebindings는 namespace로 범위가 지정되지만 clusterrolebindings는 클러스터로 범위가 지정됩니다.

      ```
      kubectl get rolebindings -A
      ```

      ```
      kubectl get clusterrolebindings
      ```

   4. rolebinding 또는 clusterrolebinding의 모든 세부 정보를 보고 roleRef로 나열된 이전 단계의 role 또는 clusterrole 그리고 subjects에 나열된 그룹 이름이 있는지 확인합니다.

*role-binding-name*을 이전 명령의 출력에서 반환된 rolebinding 이름으로 바꿉니다. *kube-system*을 rolebinding의 namespace로 바꿉니다.

```
kubectl describe rolebinding role-binding-name -n kube-system
```

예제 출력은 다음과 같습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eks-console-dashboard-restricted-access-role-binding
  namespace: default
subjects:
- kind: Group
  name: eks-console-dashboard-restricted-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: eks-console-dashboard-restricted-access-role
  apiGroup: rbac.authorization.k8s.io
```

*cluster-role-binding-name*을 이전 명령의 출력에서 반환된 clusterrolebinding 이름으로 바꿉니다.

```
kubectl describe clusterrolebinding cluster-role-binding-name
```

예제 출력은 다음과 같습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks-console-dashboard-full-access-binding
subjects:
- kind: Group
  name: eks-console-dashboard-full-access-group
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: eks-console-dashboard-full-access-clusterrole
```

```
  apiGroup: rbac.authorization.k8s.io
```

3. aws-auth ConfigMap을 편집합니다. eksctl과 같은 도구를 사용하여 ConfigMap을 업데이트하거나 이를 편집하여 수동으로 업데이트할 수 있습니다.

> **Important**
>
> eksctl 또는 다른 도구를 사용하여 ConfigMap을 편집하는 것이 좋습니다. 사용할 수 있는 다른 도구에 대한 자세한 내용을 알아보려면 Amazon EKS 모범 사례 가이드의 도구를 사용하여 aws-authConfigMap을 변경을 참조하세요. 부적절하게 형식이 지정된 aws-auth ConfigMap으로 인해 클러스터에 대한 액세스 권한을 상실할 수 있습니다.

eksctl

전제 조건

디바이스 또는 0.175.0에 설치된 버전 AWS CloudShell 이상의 eksctl 명령줄 도구. eksctl을 설치 또는 업그레이드하려면 eksctl 설명서에서 Installation을 참조하세요.

1. ConfigMap에서 현재 매핑을 확인합니다. *my-cluster*를 클러스터 이름으로 바꿉니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

예제 출력은 다음과 같습니다.

```
ARN
                    USERNAME                                GROUPS
              ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA     system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

2. 역할에 대한 매핑을 추가합니다. *my-role*을 역할 이름으로 바꿉니다. *eks-console-dashboard-full-access-group*을 Kubernetes RoleBinding 또는 ClusterRoleBinding 객체에 지정된 그룹 이름으로 바꿉니다. *111122223333*을 계정 ID로 바꿉니다. *admin*을 선택한 모든 이름으로 바꿀 수 있습니다.

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
```

```
    --arn arn:aws:iam::111122223333:role/my-role --username admin --group eks-
console-dashboard-full-access-group \
    --no-duplicate-arns
```

> **Important**
>
> 역할 ARN에는 role/my-team/developers/my-role과 같은 경로가 포함될 수 없습니다. ARN 형식은 arn:aws:iam::*111122223333*:role/*my-role*이어야 합니다. 이 예에서는 my-team/developers/를 제거해야 합니다.

예제 출력은 다음과 같습니다.

```
[...]
2022-05-09 14:51:20 [#]  adding identity "arn:aws:iam::111122223333:role/my-
role" to auth ConfigMap
```

3. 사용자에 대한 매핑을 추가합니다. IAM 모범 사례에서는 되도록 사용자 대신 역할에 권한을 부여하세요. *my-user*를 사용자 이름으로 바꿉니다. *eks-console-dashboard-restricted-access-group*을 Kubernetes RoleBinding 또는 ClusterRoleBinding 객체에 지정된 그룹 이름으로 바꿉니다. *111122223333*을 계정 ID로 바꿉니다. *my-user*를 선택한 모든 이름으로 바꿀 수 있습니다.

```
eksctl create iamidentitymapping --cluster my-cluster --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user --username my-user --
group eks-console-dashboard-restricted-access-group \
    --no-duplicate-arns
```

예제 출력은 다음과 같습니다.

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws:iam::111122223333:user/my-
user" to auth ConfigMap
```

4. ConfigMap에서 매핑을 다시 확인합니다.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

예제 출력은 다음과 같습니다.

```
ARN
                    USERNAME                                      GROUPS
                      ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws:iam::111122223333:role/admin
                        my-role                                   eks-console-
dashboard-full-access-group
arn:aws:iam::111122223333:user/my-user
                        my-user                                   eks-console-
dashboard-restricted-access-group
```

Edit ConfigMap manually

1. 편집을 위해 ConfigMap을 엽니다.

```
kubectl edit -n kube-system configmap/aws-auth
```

> **Note**
>
> "Error from server (NotFound): configmaps "aws-auth" not found"와 같은 오류가 발생할 경우 클러스터에 aws-authConfigMap 적용의 절차를 통해 스톡 ConfigMap을 적용합니다.

2. IAM 보안 주체를 ConfigMap에 추가하세요. IAM 그룹은 IAM 보안 주체가 아니므로 ConfigMap에 추가할 수 없습니다..
   - IAM 역할(예: 페더레이션 사용자)을 추가하려면: 역할 세부 정보를 data 아래 ConfigMap의 mapRoles 섹션에 추가합니다. 파일에 이미 존재하지 않는 경우 이 섹션을 추가합니다. 각 항목은 다음 파라미터를 지원합니다.
     - rolearn: 추가할 IAM 역할의 ARN. 이 값은 경로를 포함할 수 없습니다. 예를 들어 arn:aws:iam::*111122223333*:role/my-team/developers/*role-name*과 같은 ARN을 지정할 수 없습니다. 대신 ARN은 arn:aws:iam::*111122223333*:role/*role-name*이어야 합니다.
     - username: Kubernetes 내에서 IAM 역할에 매핑할 사용자 이름.

  - groups: 역할을 매핑할 그룹 또는 Kubernetes 그룹 목록. 그룹은 기본 그룹이나 `clusterrolebinding` 또는 `rolebinding`에 지정된 그룹일 수 있습니다. 자세한 내용은 Kubernetes 설명서의 Default roles and role bindings를 참조하세요.
- IAM 사용자를 추가하는 방법: IAM 모범 사례에서는 되도록 사용자 대신 역할에 권한을 부여하세요. 사용자 세부 정보를 `data` 아래 `ConfigMap`의 `mapUsers` 섹션에 추가합니다. 파일에 이미 존재하지 않는 경우 이 섹션을 추가합니다. 각 항목은 다음 파라미터를 지원합니다.
  - userarn: 추가할 IAM 사용자의 ARN.
  - username: Kubernetes 내에서 IAM 사용자에 매핑할 사용자 이름.
  - groups: 사용자를 매핑할 그룹 또는 Kubernetes 그룹 목록. 그룹은 기본 그룹이나 `clusterrolebinding` 또는 `rolebinding`에 지정된 그룹일 수 있습니다. 자세한 내용은 Kubernetes 설명서의 Default roles and role bindings를 참조하세요.

예를 들어 아래 YAML 블록에는 다음이 포함됩니다.

- 노드가 클러스터에 직접 등록할 수 있도록 IAM 노드 인스턴스를 Kubernetes 그룹에 매핑하는 `mapRoles` 섹션 및 모든 클러스터에 대한 모든 Kubernetes 리소스를 볼 수 있는 Kubernetes 그룹에 매핑되는 `my-console-viewer-role` IAM 역할. `my-console-viewer-role` IAM 역할에 필요한 IAM 및 Kubernetes 그룹 권한 목록은 필요한 권한 섹션을 참조하세요.
- 기본 AWS 계정의 `admin` IAM 사용자를 `system:masters` Kubernetes 그룹에 매핑하는 `mapUsers` 섹션 및 특정 네임스페이스에 대한 Kubernetes 리소스를 볼 수 있는 Kubernetes 그룹에 매핑되는 다른 AWS 계정의 `my-user` 사용자. `my-user` IAM 사용자에 필요한 IAM 및 Kubernetes 그룹 권한 목록은 필요한 권한 섹션을 참조하세요.

필요에 따라 줄을 추가하거나 제거하고 모든 *`example values`*를 자신의 값으로 바꿉니다.

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this
 file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::111122223333:role/my-role
```

```
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - eks-console-dashboard-full-access-group
      rolearn: arn:aws:iam::111122223333:role/my-console-viewer-role
      username: my-console-viewer-role
  mapUsers: |
    - groups:
      - system:masters
      userarn: arn:aws:iam::111122223333:user/admin
      username: admin
    - groups:
      - eks-console-dashboard-restricted-access-group
      userarn: arn:aws:iam::444455556666:user/my-user
      username: my-user
```

3. 파일을 저장하고 텍스트 편집기를 종료합니다.

클러스터에 **aws-authConfigMap** 적용

aws-auth ConfigMap은 관리형 노드 그룹을 생성하거나 eksctl을 사용하여 노드 그룹을 생성할 때 자동으로 생성되어 클러스터에 적용됩니다. 이 ConfigMap은 처음에는 노드를 클러스터에 조인하기 만들어졌으나 이 ConfigMap을 사용하여 IAM 보안 주체에 역할 기반 액세스 제어(RBAC) 액세스를 추가할 수도 있습니다. 자체 관리형 노드를 시작했고 클러스터에 aws-auth ConfigMap을(를) 적용하지 않았다면 다음 절차를 수행하면 됩니다.

클러스터에 **aws-authConfigMap** 적용

1. aws-auth ConfigMap을(를) 이미 적용했는지 확인합니다.

   ```
   kubectl describe configmap -n kube-system aws-auth
   ```

   "Error from server (NotFound): configmaps "aws-auth" not found"와 같은 오류가 발생할 경우 다음 단계를 수행하여 스톡 ConfigMap을 적용합니다.

2. AWS Authenticator 구성 맵을 다운로드, 편집 및 적용합니다.

   a. 구성 맵을 다운로드합니다.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-
   eks/cloudformation/2020-10-29/aws-auth-cm.yaml
   ```

b. `aws-auth-cm.yaml` 파일에서 `rolearn`을 노드와 연결된 IAM 역할의 Amazon 리소스 이름(ARN)으로 설정합니다. 이 작업은 텍스트 편집기를 사용하거나 *`my-node-instance-role`*을 대체하고 다음 명령을 실행하여 수행할 수 있습니다.

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-instance-role|' aws-auth-cm.yaml
```

이 파일에서 어떠한 행도 수정하지 마세요.

> **Important**
>
> 역할 ARN에는 `role/my-team/developers/my-role`과 같은 경로가 포함될 수 없습니다. ARN 형식은 `arn:aws:iam::`*`111122223333`*`:role/`*`my-role`*이어야 합니다. 이 예에서는 `my-team/developers/`를 제거해야 합니다.

노드 그룹에 대해 AWS CloudFormation 스택 출력을 점검하고 다음 값을 찾습니다.

- InstanceRoleARN - `eksctl`로 생성된 노드 그룹용
- NodeInstanceRole - AWS Management Console에서 Amazon EKS 판매 AWS CloudFormation 템플릿으로 생성된 노드 그룹용

c. 구성을 적용합니다. 이 명령을 완료하는 데 몇 분이 걸릴 수 있습니다.

```
kubectl apply -f aws-auth-cm.yaml
```

> **Note**
>
> 권한 부여 또는 리소스 유형 오류가 표시되는 경우 문제 해결 주제의 권한이 없거나 액세스가 거부됨(kubectl) 섹션을 참조하세요.

3. 노드의 상태를 확인하고 `Ready` 상태가 될 때까지 대기합니다.

```
kubectl get nodes --watch
```

`Ctrl+C`를 입력하여 쉘 프롬프트로 돌아갑니다.

OpenID Connect 자격 증명 공급자에서 클러스터에 대해 사용자 인증

Amazon EKS는 클러스터에서 사용자를 인증하는 방법으로 OpenID Connect(OIDC) ID 제공업체를 지원합니다. OIDC ID 제공업체는 AWS Identity and Access Management(IAM)와 함께 사용하거나 대안으로 사용할 수 있습니다. IAM 사용에 대한 자세한 내용은 the section called “Kubernetes API에 대한 액세스 권한 부여”를 참조하십시오. 클러스터에 인증을 구성한 후, Kubernetes `roles` 및 `clusterroles`를 생성하여 역할에 권한을 할당한 다음 Kubernetes `rolebindings` 및 `clusterrolebindings`를 사용하여 역할을 ID에 바인딩할 수 있습니다. 자세한 내용은 Kubernetes 설명서의 RBAC 승인 사용을 참조하세요.

고려 사항

- 하나의 OIDC ID 제공업체를 클러스터에 연결할 수 있습니다.
- Kubernetes에서는 OIDC ID 제공업체를 제공하지 않습니다. 기존 퍼블릭 OIDC ID 제공업체를 사용하거나 자체 ID 제공업체를 실행할 수 있습니다. 인증된 공급자 목록은 OpenID 사이트에서 OpenID 인증을 참조하세요.
- Amazon EKS가 서명 키를 검색할 수 있도록 OIDC ID 제공업체의 발급자 URL에 공개적으로 액세스할 수 있어야 합니다. Amazon EKS는 자체 서명된 인증서를 사용하는 OIDC ID 제공업체를 지원하지 않습니다.
- 클러스터에 노드를 조인하는 데 여전히 필요하므로 클러스터에 IAM 인증을 비활성화할 수 없습니다.
- 하지만 여전히 OIDC ID 제공업체 사용자가 아니라 AWS IAM 보안 주체가 Amazon EKS 클러스터를 생성해야 합니다. 이는 클러스터 생성자가 Kubernetes API가 아닌 Amazon EKS API와 상호 작용하기 때문입니다.
- 컨트롤 플레인에 대해 CloudWatch Logs가 설정되어 있는 경우 OIDC ID 제공업체 인증 사용자는 클러스터의 감사 로그에 표시됩니다. 자세한 내용은 컨트롤 플레인 로그 활성화 및 비활성화 단원을 참조하십시오.
- OIDC 공급자가 제공한 계정으로 AWS Management Console에 로그인할 수 없습니다. AWS Management Console에 AWS Identity and Access Management 계정으로 로그인한 경우에만 콘솔에서 Kubernetes 리소스 보기를 할 수 있습니다.

OIDC ID 제공업체 연결

OIDC ID 제공업체를 클러스터에 연결하려면 먼저 다음과 같은 제공업체 정보가 필요합니다.

발급자 URL

API 서버가 토큰을 확인하기 위한 퍼블릭 서명 키를 검색할 수 있도록 허용하는 OIDC ID 제공자의 URL입니다. URL은 `https://`로 시작해야 하며 공급자의 OIDC ID 토큰에 있는 `iss` 클레임에 해당해야 합니다. OIDC 표준에 따라 경로 구성 요소는 허용되지만 쿼리 파라미터는 허용되지 않습니다. 일반적으로 URL은 `https://server.example.org` 또는 `https://example.com` 같은 하나의 호스트 이름으로만 구성됩니다. 이 URL은 `.well-known/openid-configuration` 아래 레벨을 가리켜야 하며 인터넷을 통해 공개적으로 액세스할 수 있어야 합니다.

클라이언트 ID(대상이라고도 함)

OIDC ID 제공자에게 인증을 요청하는 클라이언트 애플리케이션의 ID입니다.

`eksctl` 또는 AWS Management Console을 사용하여 자격 증명 공급자를 연결할 수 있습니다.

eksctl

`eksctl`을 사용하여 OIDC ID 제공업체를 클러스터에 연결하는 방법

1. 다음 콘텐츠를 가진 *`associate-identity-provider.yaml`*이라는 파일을 생성합니다: *`example values`*을 사용자의 값으로 교체합니다. `identityProviders` 섹션의 값은 OIDC ID 제공업체로부터 가져옵니다. 값은 `identityProviders`의 `name`, `type`, `issuerUrl`, `clientId` 설정에만 필요합니다.

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: your-region-code

identityProviders:
  - name: my-provider
    type: oidc
    issuerUrl: https://example.com
    clientId: kubernetes
    usernameClaim: email
    usernamePrefix: my-username-prefix
    groupsClaim: my-claim
    groupsPrefix: my-groups-prefix
    requiredClaims:
```

```
    string: string
  tags:
    env: dev
```

> **Important**
>
> groupsPrefix 또는 usernamePrefix에 대해 system: 또는 해당 문자열의 어떤 부분도 지정하지 마세요.

2. 공급자를 생성합니다.

```
eksctl associate identityprovider -f associate-identity-provider.yaml
```

3. 클러스터와 OIDC ID 제공업체에서 kubectl을 사용하려면 Kubernetes 설명서에서 Using kubectl을 참조하세요.

AWS Management Console

AWS Management Console을 사용하여 OIDC ID 제공업체를 연결하는 방법

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 클러스터를 선택한 다음 액세스 탭을 선택합니다.
3. OIDC ID 제공업체 섹션에서 ID 제공업체 연결을 선택합니다.
4. OIDC ID 제공업체 연결 페이지에서 다음 옵션을 입력하거나 선택한 후 연결을 선택합니다.
   - [이름(Name)]에 공급자의 고유한 이름을 입력합니다.
   - [발급자 URL(Issuer URL)]에 공급자의 URL을 입력합니다. 이는 인터넷을 통해 액세스할 수 있는 URL이어야 합니다.
   - 클라이언트 ID에 OIDC ID 제공업체의 클라이언트 ID(대상이라고도 함)를 입력합니다.
   - [사용자 이름 클레임(Username claim)]에 사용자 이름으로 사용할 클레임을 입력합니다.
   - [그룹 클레임(Groups claim)]에 사용자의 그룹으로 사용할 클레임을 입력합니다.
   - (선택 사항) [고급 옵션(Advanced options)]을 선택하고 다음 정보를 입력하거나 선택합니다.
     - 사용자 이름 접두사 — 사용자 이름 클레임 앞에 추가할 접두사를 입력합니다. 기존 이름과의 충돌을 방지하기 위해 접두사가 사용자 이름 클레임 앞에 추가됩니다. 값을 제공하지 않은 경우 사용자 이름이 email 이외의 값이면 접두사는 발급자 URL에 대한 기본값으로

설정됩니다. 값 `-`를 사용하여 모든 접두사를 사용 중지할 수 있습니다. `system:` 또는 해당 문자열의 어떤 부분도 지정하지 마세요.

- 그룹 접두사 — 그룹 클레임 앞에 추가할 접두사를 입력합니다. 기존 이름(예: `system:groups`)과의 충돌을 방지하기 위해 접두사가 그룹 클레임 앞에 추가됩니다. 예를 들어 값 `oidc:`는 `oidc:engineering` 및 `oidc:infra`와 같은 그룹 이름을 생성합니다. `system:` 또는 해당 문자열의 어떤 부분도 지정하지 마세요.
- 필수 클레임 — [클레임 추가(Add claim)]를 선택하고 클라이언트 ID 토큰에 필요한 클레임을 설명하는 하나 이상의 키 값 페어를 입력합니다. 키 값 페어는 ID 토큰에 필요한 클레임을 설명합니다. 설정된 경우 각 클레임은 일치하는 값이 ID 토큰에 존재하는 것으로 확인됩니다.

5. 클러스터와 OIDC ID 제공업체에서 `kubectl`을 사용하려면 Kubernetes 설명서에서 Using `kubectl`을 참조하세요.

클러스터에서 OIDC ID 제공업체 연결 해제

클러스터에서 OIDC ID 제공업체의 연결을 해제하면 공급자에 포함된 사용자가 더 이상 클러스터에 액세스할 수 없습니다. 하지만 IAM 보안 주체로 계속 클러스터에 액세스할 수 있습니다.

AWS Management Console을 사용하여 클러스터에서 OIDC ID 제공업체의 연결 해제하는 방법

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. OIDC ID 제공업체 섹션에서 연결 해제를 선택하고 ID 제공업체 이름을 입력한 후 `Disassociate`를 선택합니다.

IAM 정책 예제

OIDC ID 제공업체가 클러스터에 연결되지 않도록 하려면 다음 IAM 정책을 생성하여 Amazon EKS 관리자의 IAM 계정에 연결합니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 생성 및 IAM 자격 증명 권한 추가와 서비스 권한 부여 참조의 Amazon Elastic Kubernetes Service에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "denyOIDC",
            "Effect": "Deny",
```

```
            "Action": [
                "eks:AssociateIdentityProviderConfig"
            ],
            "Resource": "arn:aws:eks:us-west-2.amazonaws.com:111122223333:cluster/*"

        },
        {
            "Sid": "eksAdmin",
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        }
    ]
}
```

다음 예제 정책은 clientID가 kubernetes이고 issuerUrl이 https://cognito-idp.us-west-2amazonaws.com/*인 경우 OIDC ID 제공업체 연결을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCognitoOnly",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotLikeIfExists": {
                    "eks:issuerUrl": "https://cognito-idp.us-west-2.amazonaws.com/*"
                }
            }
        },
        {
            "Sid": "DenyOtherClients",
            "Effect": "Deny",
            "Action": "eks:AssociateIdentityProviderConfig",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-instance",
            "Condition": {
                "StringNotEquals": {
                    "eks:clientId": "kubernetes"
                }
```

```
            }
        },
        {
            "Sid": "AllowOthers",
            "Effect": "Allow",
            "Action": "eks:*",
            "Resource": "*"
        }
    ]
}
```

파트너가 검증한OIDC 자격 증명 제공업체

Amazon EKS는 호환 가능한 OIDC 자격 증명 공급자에 대한 지원을 제공하는 파트너 네트워크와 관계를 유지합니다. 자격 증명 공급자와 Amazon EKS를 통합하는 자세한 방법은 다음 파트너 설명서를 참조하세요.

파트너	제품	설명서
PingIdentity	엔터프라이즈용 PingOne	설치 지침

Amazon EKS는 모든 사용 사례를 포괄하는 다양한 옵션을 제공하는 것을 목표로 합니다. 여기에 나열되지 않은 상업적으로 지원되는 OIDC 호환 가능한 자격 증명 공급자를 개발하는 경우 자세한 내용은 파트너 팀(aws-container-partners@amazon.com)으로 문의하세요.

Amazon EKS 클러스터용 **kubeconfig** 파일 생성 또는 업데이트

이 주제에서는 클러스터에 대한 `kubeconfig` 파일을 생성하거나 기존 파일을 업데이트합니다.

`kubectl` 명령줄 도구는 `kubeconfig` 파일의 구성 정보를 사용하여 클러스터의 API 서버와 통신합니다. 자세한 내용을 알아보려면 Kubernetes 문서의 kubeconfig 파일을 사용하여 클러스터 접근 구성하기를 참조하세요.

Amazon EKS는 클러스터 인증을 위해 `kubectl`과 함께 `aws eks get-token` 명령을 사용합니다. 기본적으로 AWS CLI에서는 다음 명령을 사용하여 반환되는 것과 동일한 보안 인증 정보를 사용합니다.

```
aws sts get-caller-identity
```

필수 조건

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요.
- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 `kubectl` 설치 또는 업데이트 부분을 참조하세요.
- AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.
- 지정한 클러스터에 대해 `eks:DescribeCluster` API 작업을 사용할 수 있는 권한을 보유한 IAM 사용자 또는 역할. 자세한 내용은 Amazon EKS 자격 증명 기반 정책 예제 단원을 참조하십시오. 자체 OpenID Connect 제공업체 ID를 사용하여 클러스터에 액세스하는 경우 Kubernetes 설명서에서 Using `kubectl`을 참조하거나 `kube config` 파일을 생성 또는 업데이트합니다.

자동으로 **`kubeconfig`** 파일 생성

필수 조건

- AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.
- 지정한 클러스터에 대해 `eks:DescribeCluster` API 작업을 사용할 수 있는 권한. 자세한 내용은 Amazon EKS 자격 증명 기반 정책 예제 단원을 참조하십시오.

AWS CLI로 **kubeconfig**를 생성하려면

1. 클러스터에 대해 kubeconfig 파일을 생성 또는 업데이트합니다. *region-code*을 해당 클러스터가 있는 AWS 리전으로 바꾸고, *my-cluster*를 클러스터의 이름으로 바꿉니다.

   ```
   aws eks update-kubeconfig --region region-code --name my-cluster
   ```

   기본적으로 구성 파일은 홈 디렉터리의 기본 kubeconfig 경로(.kube)에 생성되거나 해당 위치의 기존 config 파일과 병합됩니다. **--kubeconfig** 옵션을 사용하여 다른 경로를 지정할 수 있습니다.

   --role-arn 옵션으로 IAM 역할 ARN을 지정하면 kubectl 명령을 실행할 때 인증에 사용할 수 있습니다. 그렇지 않으면 기본 AWS CLI 또는 SDK 자격 증명 체인의 IAM 보안 주체가 사용됩니다. aws sts get-caller-identity 명령을 실행하여 기본 AWS CLI 또는 SDK ID를 확인할 수 있습니다.

   사용 가능한 모든 옵션을 알아보려면 aws eks update-kubeconfig help 명령을 실행하거나 AWS CLI 명령 참조에서 update-kubeconfig을 참조하세요.
2. 구성을 테스트합니다.

   ```
   kubectl get svc
   ```

   예제 출력은 다음과 같습니다.

   ```
   NAME             TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
   svc/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   1m
   ```

   권한 부여 또는 리소스 유형 오류가 표시되는 경우 문제 해결 주제의 권한이 없거나 액세스가 거부됨(kubectl) 부분을 참조하세요.

Kubernetes 서비스 계정을 사용하여 Kubernetes 워크로드에 AWS에 대한 액세스 권한 부여

Kubernetes 서비스 계정은 Pod에서 실행되는 프로세스의 ID를 제공합니다. 자세한 내용을 알아보려면 Kubernetes 설명서의 서비스 계정 관리를 참조하세요. Pod에 AWS 서비스에 대한 액세스 권한이 필요한 경우 서비스 계정을 AWS Identity and Access Management ID에 매핑하여 해당 액세스 권한을 부여할 수 있습니다. 자세한 내용은 서비스 계정에 대한 IAM 역할 단원을 참조하십시오.

서비스 계정 토큰

BoundServiceAccountTokenVolume 기능은 기본적으로 Kubernetes 버전에서 활성화됩니다. 이 기능은 Kubernetes에서 실행되는 워크로드가 대상, 시간 및 키 바인딩인 JSON 웹 토큰을 요청하도록 허용하여 서비스 계정 토큰의 보안을 향상시킵니다. 서비스 계정 토큰에는 1시간 만료 기간이 있습니다. 이전 Kubernetes 버전에는 토큰에 만료 기간이 없었습니다. 즉, 이러한 토큰에 의존하는 클라이언트는 1시간 이내에 토큰을 새로 고쳐야 합니다. 다음 Kubernetes 클라이언트 SDK는 필요한 시간 내에 토큰을 자동으로 새로 고칩니다.

- Go 버전 `0.15.7` 이상
- Python 버전 `12.0.0` 이상
- Java 버전 `9.0.0` 이상
- JavaScript 버전 `0.10.3` 이상
- Ruby `master` 브랜치
- Haskell 버전 `0.3.0.0`
- C# 버전 `7.0.5` 이상

워크로드가 이전 클라이언트 버전을 사용하는 경우 업데이트해야 합니다. 클라이언트가 최신 시간 바운드 서비스 계정 토큰으로 원활하게 마이그레이션되도록 Kubernetes는 기본 1시간 서비스 계정 토큰의 만료 기간을 추가로 연장합니다. Amazon EKS 클러스터의 경우 연장 만료 기간은 90일입니다. Amazon EKS 클러스터 Kubernetes API 서버는 90일이 지난 토큰을 사용한 요청을 거부합니다. 애플리케이션 및 해당 종속성을 확인하여 Kubernetes 클라이언트 SDK가 이전에 나열된 버전과 동일하거나 이후 버전인지 확인하는 것이 좋습니다.

API 서버가 1시간 이상 된 토큰으로 요청을 수신하면 API 감사 로그 이벤트에 `annotations.authentication.k8s.io/stale-token`으로 주석을 추가합니다. 주석의 값은 다음 예와 유사합니다.

```
subject: system:serviceaccount:common:fluent-bit, seconds after warning threshold:
 4185802.
```

클러스터에 컨트롤 플레인 로깅이 사용 설정되어 있는 경우 주석은 감사 로그에 있습니다. 다음과 같은 CloudWatch Logs Insights 쿼리를 사용하여 Amazon EKS 클러스터에서 오래된 토큰을 사용하는 모든 Pods를 식별할 수 있습니다.

```
fields @timestamp
```

```
| filter @logStream like /kube-apiserver-audit/
| filter @message like /seconds after warning threshold/
| parse @message "subject: *, seconds after warning threshold:*\"" as subject,
 elapsedtime
```

`subject`는 Pod가 사용한 서비스 계정을 참조합니다. `elapsedtime`은 최신 토큰을 읽은 후 경과 시간(초)을 표시합니다. API 서버에 대한 요청은 `elapsedtime`이 90일(7,776,000 초)을 초과하는 경우 거부됩니다. 이전에 나열된 버전 중 하나를 사용하여 토큰을 자동으로 새로 고치도록 하려면 애플리케이션의 Kubernetes 클라이언트 SDK를 적극적으로 업데이트해야 합니다. 사용된 서비스 계정 토큰이 90일에 가까우며 토큰 만료 전에 클라이언트 SDK 버전을 업데이트할 시간이 충분하지 않은 경우 기존 Pods를 종료하고 새 포드를 생성할 수 있습니다. 이로 인해 서비스 계정 토큰을 다시 가져와서 클라이언트 버전 SDK를 업데이트하는 데 추가로 90일이 제공됩니다.

Pod가 배포의 일부인 경우 고가용성을 유지하면서 Pods를 종료하라고 제안된 방법은 다음 명령으로 롤아웃을 수행하는 것입니다. *`my-deployment`*를 배포 이름으로 바꿉니다.

```
kubectl rollout restart deployment/my-deployment
```

클러스터 추가 기능

다음 클러스터 추가 기능은 Kubernetes 클라이언트 SDK를 사용하여 서비스 계정 토큰을 자동으로 다시 가져오도록 업데이트되었습니다. 나열된 버전 또는 이후 버전이 클러스터에 설치되었는지 확인하는 것이 좋습니다.

- Amazon VPC CNI plugin for Kubernetes 및 CNI 지표 헬퍼 플러그인 버전`1.8.0` 이상. 현재 버전을 확인하거나 업데이트하려면 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 및 cni-metrics-helper 부분을 참조하세요.
- CoreDNS 버전 `1.8.4` 이상 현재 버전을 확인하거나 업데이트하려면 CoreDNS Amazon EKS 추가 기능을 사용한 작업 섹션을 참조하세요.
- AWS Load Balancer Controller 버전 `2.0.0` 이상 현재 버전을 확인하거나 업데이트하려면 AWS Load Balancer Controller란 무엇인가요? 부분을 참조하세요.
- 현재 `kube-proxy` 버전 현재 버전을 확인하거나 업데이트하려면 Kubernetes `kube-proxy` 추가 기능으로 작업하기 부분을 참조하세요.
- AWS for Fluent Bit 버전 `2.25.0` 이상. 현재 버전을 업데이트하려면 GitHub의 릴리스를 참조하세요.
- Fluentd 이미지 버전 1.14.6-1.2 이상 및 Kubernetes 메타데이터 버전 2.11.1 이상용 Fluentd 필터 플러그인.

Amazon Elastic Kubernetes Service 클러스터의 워크로드에 AWS Identity and Access Management 권한 부여

Amazon EKS는 서비스 계정에 대한 IAM 역할 및 EKS Pod Identity를 통해 Amazon EKS 클러스터에서 실행되는 워크로드에 AWS Identity and Access Management 권한을 부여합니다.

서비스 계정에 대한 IAM 역할

서비스 계정에 대한 IAM 역할(IRSA)은 세분화된 IAM 권한으로 AWS에서 실행되는 Kubernetes 애플리케이션을 구성하여 Amazon S3 버킷, Amazon DynamoDB 테이블 등과 같은 다양한 기타 AWS 리소스에 액세스할 수 있도록 합니다. 동일한 Amazon EKS 클러스터에서 여러 애플리케이션을 함께 실행하고 각 애플리케이션이 필요한 최소 권한 집합만 갖도록 할 수 있습니다. IRSA는 Amazon EKS, Amazon EKS Anywhere Red Hat OpenShift Service on AWS 및 Amazon EC2 인스턴스의 자체 관리형 Kubernetes 클러스터 등 AWS에서 지원하는 다양한 Kubernetes 배포 옵션을 지원하도록 빌드되었습니다. 따라서 IRSA는 IAM과 같은 기본 AWS 서비스를 사용하여 빌드되었으며 Amazon EKS 서비스 및 EKS API에 대한 직접적인 종속성을 갖지 않았습니다. 자세한 내용은 서비스 계정에 대한 IAM 역할 단원을 참조하십시오.

EKS Pod Identity

EKS Pod Identity는 클러스터 관리자에게 Amazon S3 버킷, Amazon DynamoDB 테이블 등과 같은 다양한 기타 AWS 리소스에 액세스할 수 있도록 간소화된 애플리케이션 인증 워크플로를 제공합니다. EKS Pod Identity는 EKS 전용이므로 클러스터 관리자가 IAM 권한을 획득하도록 Kubernetes 애플리케이션을 구성하는 방법이 간소화됩니다. 이제 AWS Management Console, EKS API, AWS CLI를 통해 직접 몇 단계만 거치면 이러한 권한을 쉽게 구성할 수 있으며, 임의의 Kubernetes 객체에 있는 클러스터 내에서 별도의 조치를 취하지 않아도 됩니다. 클러스터 관리자는 EKS와 IAM 서비스 간에 전환하거나 권한이 있는 IAM 작업을 사용하여 애플리케이션에 필요한 권한을 구성할 필요가 없습니다. 이제 새 클러스터를 생성할 때 역할 신뢰 정책을 업데이트할 필요 없이 여러 클러스터에서 IAM 역할을 사용할 수 있습니다. EKS Pod Identity에서 제공하는 IAM 보안 인증 정보에는 클러스터 이름, 네임스페이스, 서비스 계정 이름 등의 속성이 포함된 역할 세션 태그가 포함됩니다. 관리자는 역할 세션 태그를 사용하여 일치하는 태그를 기반으로 AWS 리소스에 대한 액세스를 허용하여 여러 서비스 계정에서 작업할 수 있는 단일 역할을 작성할 수 있습니다. 자세한 내용은 EKS Pod Identity 단원을 참조하십시오.

EKS Pod Identity와 IRSA 비교

대략적으로 EKS Pod Identity와 IRSA 모두 Kubernetes 클러스터에서 실행되는 애플리케이션에 IAM 권한을 부여할 수 있도록 합니다. 하지만 구성 방법, 지원되는 제한 사항, 활성화된 기능 등이 근본적으로 다릅니다. 아래에서는 두 솔루션의 몇 가지 주요 측면을 비교해 보겠습니다.

	EKS Pod Identity	IRSA
역할 확장성	새로 도입된 Amazon EKS 서비스 보안 주체 `pods.eks.amazonaws.com` 과의 신뢰를 구축하려면 각 역할을 한 번 설정해야 합니다. 이 한 번의 단계를 거친 후에는 새 클러스터에서 역할이 사용될 때마다 역할의 신뢰 정책을 업데이트할 필요가 없습니다.	새 클러스터에서 역할을 사용할 때마다 새 EKS 클러스터 OIDC 공급자 엔드포인트로 IAM 역할의 신뢰 정책을 업데이트해야 합니다.
클러스터 확장성	EKS Pod Identity의 경우 사용자가 IAM OIDC 공급자를 설정할 필요가 없으므로 이 제한은 적용되지 않습니다.	각 EKS 클러스터에는 OpenID Connect(OIDC) 발급자 URL이 연결되어 있습니다. IRSA를 사용하려면 IAM의 각 EKS 클러스터에 대해 고유한 OpenID Connect 공급자를 생성해야 합니다. IAM의 기본 글로벌 제한은 각 AWS 계정에 대해 100개의 OIDC 공급자입니다. IRSA가 있는 각 AWS 계정에 대해 100개를 초과하는 EKS 클러스터를 보유하려는 계획인 경우 IAM OIDC 공급자 한도에 도달합니다.
역할 확장성	EKS Pod Identity의 경우 사용자가 신뢰 정책에서 IAM 역할과 서비스 계정 간의 신뢰 관계를 정의할 필요가 없으므로 이 제한은 적용되지 않습니다.	IRSA에서는 역할의 신뢰 정책에서 IAM 역할과 서비스 계정 간의 신뢰 관계를 정의합니다. 기본적으로 신뢰 정책 크기의 길이는 2048입니다. 즉, 일반

	EKS Pod Identity	IRSA
		적으로 단일 신뢰 정책에서 4개의 신뢰 관계를 정의할 수 있습니다. 신뢰 정책 길이 제한을 늘릴 수 있지만 일반적으로 단일 신뢰 정책 내에서 최대 8개의 신뢰 관계로 제한됩니다.
역할 재사용 가능성	EKS Pod Identity에서 제공하는 AWS STS 보안 인증 정보에는 클러스터 이름, 네임스페이스, 서비스 계정 이름 등과 같은 역할 세션 태그가 포함됩니다. 역할 세션 태그를 사용하면 관리자는 연결된 태그를 기반으로 AWS 리소스에 대한 액세스를 허용함으로써 유효 권한이 다른 여러 서비스 계정에서 사용할 수 있는 단일 IAM 역할을 작성할 수 있습니다. 이를 속성 기반 액세스 제어(ABAC)라고 합니다. 자세한 내용은 EKS Pod Identity가 태그를 기반으로 역할을 맡을 수 있는 권한을 정의합니다. 단원을 참조하십시오.	AWS STS 세션 태그는 지원되지 않습니다. 클러스터 간에 역할을 재사용할 수 있지만 모든 포드는 해당 역할의 모든 권한을 받습니다.
지원되는 환경	EKS Pod Identity는 Amazon EKS에서만 사용할 수 있습니다.	IRSA는 Amazon EKS, Amazon EKS Anywhere, Red Hat OpenShift Service on AWS, Amazon EC2 인스턴스의 자체 관리형 Kubernetes 클러스터 등에서 사용할 수 있습니다.

	EKS Pod Identity	IRSA
지원되는 EKS 버전	EKS Kubernetes 버전 `1.24` 이상. 특정 플랫폼 버전은 EKS Pod Identity 클러스터 버전 섹션을 참조하세요.	지원되는 모든 EKS 클러스터 버전.

EKS Pod Identity

Pod의 컨테이너에 있는 애플리케이션에서는 AWS SDK 또는 AWS CLI를 통해 AWS Identity and Access Management(IAM) 권한을 사용하여 AWS 서비스에 API를 요청할 수 있습니다. 애플리케이션은 AWS 자격 증명으로 AWS API 요청에 서명해야 합니다.

EKS Pod Identity는 Amazon EC2 인스턴스 프로파일이 Amazon EC2 인스턴스에 자격 증명을 제공하는 것과 비슷한 방식으로 애플리케이션에 대한 자격 증명을 관리하는 기능을 제공합니다. AWS 자격 증명을 생성하여 컨테이너에 배포하거나 Amazon EC2 인스턴스의 역할을 사용하는 대신에 IAM 역할을 Kubernetes 서비스 계정과 연결하고 서비스 계정을 사용하도록 Pods를 구성합니다.

각 EKS Pod Identity 연결은 역할을 지정된 클러스터의 네임스페이스에 있는 서비스 계정에 매핑합니다. 여러 클러스터에 동일한 애플리케이션이 있는 경우 역할의 신뢰 정책을 수정하지 않고 각 클러스터에서 동일한 연결을 생성할 수 있습니다.

포드가 연결이 있는 서비스 계정을 사용하는 경우 Amazon EKS는 포드의 컨테이너에 환경 변수를 설정합니다. 환경 변수는 AWS CLI를 포함한 AWS SDK를 구성하여 EKS Pod Identity 보안 인증 정보를 사용합니다.

EKS Pod Identity의 이점

EKS Pod Identity는 다음 이점을 제공합니다.

- 최소 권한 – IAM 권한의 범위를 서비스 계정으로 지정할 수 있습니다. 그러면 해당 서비스 계정을 사용하는 Pods만 이 권한에 액세스할 수 있습니다. 또한 이 기능을 사용하면 `kiam`, `kube2iam` 같은 타사 솔루션이 필요 없습니다.
- 자격 증명 격리 - Pod's 컨테이너는 컨테이너가 사용하는 서비스 계정과 연결된 IAM 역할에 대한 자격 증명만 검색할 수 있습니다. 컨테이너는 다른 Pods의 다른 컨테이너에서 사용하는 자격 증명에 액세스할 수 없습니다. Pod Identity를 사용할 때 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 대한 Pod 액세스를 차단하지 않는 한 Pod's 컨테이너는 Amazon EKS 노드 IAM 역할에 할

당된 권한도 갖습니다. 자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 부분을 참조하세요.

- 감사 - 소급적 감사를 위해 AWS CloudTrail을 통해 액세스 및 이벤트 로깅이 가능합니다.

EKS Pod Identity는 OIDC 자격 증명 공급자를 사용하지 않으므로 서비스 계정에 대한 IAM 역할에 비해 간단한 방법입니다. EKS Pod Identity의 개선 사항:

- 독립적 운영 - 많은 조직에서 OIDC 자격 증명 공급자를 만드는 것은 Kubernetes 클러스터를 관리하는 팀이 아닌 다른 팀의 책임입니다. EKS Pod Identity는 업무가 분명히 분리되어 있어 EKS Pod Identity 연결의 모든 구성은 Amazon EKS에서, IAM 권한의 모든 구성은 IAM에서 이루어집니다.
- 재사용 가능성 - EKS Pod Identity는 서비스 계정에 대한 IAM 역할이 사용하는 각 클러스터에 대해 별도의 보안 주체 대신 단일 IAM 보안 주체를 사용합니다. IAM 관리자는 모든 역할의 신뢰 정책에 다음 보안 주체를 추가하여 EKS Pod Identity에서 사용할 수 있도록 합니다.

```
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            }
```

- 확장성 - 각 임시 보안 인증 정보 세트는 각 포드에서 실행하는 각 AWS SDK 대신 EKS Pod Identity의 EKS Auth 서비스에서 수임합니다. 그러면 각 노드에서 실행되는 Amazon EKS Pod Identity Agent가 SDK에 보안 인증 정보를 발급합니다. 따라서 부하가 각 노드당 한 번으로 줄어들고 각 포드에 중복되지 않습니다. 프로세스의 자세한 내용은 EKS Pod Identity 작동 방식 섹션을 참조하세요.

두 대안의 비교에 대한 자세한 내용은 Kubernetes 서비스 계정을 사용하여 Kubernetes 워크로드에 AWS에 대한 액세스 권한 부여 섹션을 참조하세요.

EKS Pod Identity 설정 개요

다음 절차를 완료하여 EKS Pod Identity를 활성화합니다.

1. Amazon EKS Pod Identity Agent 설정 - 각 클러스터에 대해 한 번만 이 절차를 완료합니다.
2. EKS Pod Identity 관련 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 - 애플리케이션에 부여하려는 고유한 권한 세트에 대해 이 절차를 완료합니다.
3. Kubernetes 서비스 계정을 사용하도록 Pods 구성 - AWS 서비스에 액세스해야 하는 각 Pod에 대해 이 절차를 완료합니다.

4. 지원되는 AWS SDK 사용 - 워크로드가 지원되는 버전의 AWS SDK를 사용하고 워크로드가 기본 보안 인증 정보 체인을 사용하는지 확인합니다.

EKS Pod Identity 고려 사항

- 각 클러스터의 각 Kubernetes 서비스 계정에 IAM 역할 하나를 연결할 수 있습니다. EKS Pod Identity 연결을 편집하여 서비스 계정에 매핑되는 역할을 변경할 수 있습니다.
- 클러스터와 동일한 AWS 계정에 있는 역할만 연결할 수 있습니다. EKS Pod Identity가 사용하도록 구성한 이 계정의 역할에 다른 계정의 액세스 권한을 위임할 수 있습니다. 액세스 권한 위임 및 `AssumeRole`에 대한 자습서는 IAM 사용 설명서의 IAM 역할을 사용한 AWS 계정 간 액세스 권한 위임을 참조하세요.
- EKS Pod Identity Agent가 필요합니다. 노드에서 Kubernetes `DaemonSet`로 실행되고 실행되는 노드의 포드에만 보안 인증 정보를 제공합니다. EKS Pod Identity Agent 호환성에 대한 자세한 내용은 EKS Pod Identity 제한 사항 섹션을 참조하세요.
- EKS Pod Identity Agent는 노드의 `hostNetwork`를 사용하고 노드의 링크-로컬 주소에 있는 포트 `80` 및 포트 `2703`을 사용합니다. 이 주소는 IPv4의 경우 `169.254.170.23`, IPv6 클러스터의 경우 `[fd00:ec2::23]`입니다.

EKS Pod Identity 클러스터 버전

EKS Pod Identity를 사용하려면 클러스터의 플랫폼 버전이 다음 표에 나열된 버전과 동일한 버전 또는 이후 버전이거나 Kubernetes 버전이 표에 나열된 버전보다 이후 버전이어야 합니다.

Kubernetes 버전	플랫폼 버전
`1.28`	`eks.4`
`1.27`	`eks.8`
`1.26`	`eks.9`
`1.25`	`eks.10`
`1.24`	`eks.13`

EKS Pod Identity 제한 사항

EKS Pod Identity는 다음에서 사용 가능합니다.

- 이전 주제 EKS Pod Identity 클러스터 버전에 나열된 Amazon EKS 클러스터 버전.
- Linux Amazon EC2 인스턴스인 클러스터의 워커 노드.

EKS Pod Identity는 다음에서 사용할 수 없습니다.

- 중국 리전.
- AWS GovCloud (US).
- AWS Outposts.
- Amazon EKS Anywhere.
- Amazon EC2에서 생성하고 실행하는 Kubernetes 클러스터. EKS Pod Identity 구성 요소는 Amazon EKS에서만 사용할 수 있습니다.

다음에서는 EKS Pod Identity를 사용할 수 없습니다.

- Linux Amazon EC2 인스턴스를 제외한 모든 위치에서 실행되는 포드. AWS Fargate (Fargate)에서 실행되는 Linux 및 Windows 포드는 지원되지 않습니다. Windows Amazon EC2 인스턴스에서 실행되는 포드는 지원되지 않습니다.
- IAM 보안 인증 정보가 필요한 Amazon EKS 추가 기능. EKS 추가 기능은 서비스 계정에 대한 IAM 역할 대신으로만 사용할 수 있습니다. IAM 보안 인증 정보를 사용하는 EKS 추가 기능 목록:
  - Amazon VPC CNI plugin for Kubernetes
  - AWS Load Balancer Controller
  - CSI 스토리지 드라이버: EBS CSI, EFS CSI, Amazon FSx for Lustre CSI 드라이버, Amazon FSx for NetApp ONTAP CSI 드라이버, Amazon FSx for OpenZFS CSI 드라이버, Amazon File Cache CSI 드라이버

> **Note**
>
> 이러한 컨트롤러, 드라이버 및 플러그인을 EKS 추가 기능 대신 자체 관리형 추가 기능으로 설치한 경우 최신 AWS SDK를 사용하도록 업데이트되면 EKS Pod Identity가 지원됩니다.

EKS Pod Identity 작동 방식

Amazon EKS Pod Identity는 Amazon EC2 인스턴스 프로파일이 Amazon EC2 인스턴스에 자격 증명을 제공하는 것과 비슷한 방식으로 애플리케이션에 대한 자격 증명을 관리하는 기능을 제공합니다.

Amazon EKS Pod Identity는 추가 EKS Auth API와 각 노드에서 실행되는 에이전트 포드를 통해 워크로드에 보안 인증 정보를 제공합니다.

Amazon EKS 추가 기능 및 자체 관리형 컨트롤러, 운영자 및 기타 추가 기능과 같은 추가 기능에서 작성자는 최신 AWS SDK를 사용하도록 소프트웨어를 업데이트해야 합니다. EKS Pod Identity와 Amazon EKS에서 만든 추가 기능 간의 호환성 목록은 이전 섹션(EKS Pod Identity 제한 사항)을 참조하세요.

코드에서 EKS Pod Identity 사용

코드에서 AWS SDK를 사용하여 AWS 서비스에 액세스할 수 있습니다. 코드를 작성하여 SDK가 포함된 AWS 서비스에 대한 클라이언트를 만들고, 기본적으로 SDK는 일련의 위치에서 사용할 AWS Identity and Access Management 보안 인증 정보를 검색합니다. 유효한 보안 인증 정보를 찾은 후에는 검색이 중지됩니다. 사용되는 기본 위치에 대한 자세한 내용은 AWS SDK 및 도구 참조 가이드의 보안 인증 정보 공급자 체인을 참조하세요.

기본 보안 인증 정보 체인의 한 단계에서 검색되는 컨테이너 보안 인증 정보 공급자에 EKS Pod Identity가 추가되었습니다. 워크로드가 현재 보안 인증 정보 체인의 이전 단계에 있는 보안 인증 정보를 사용하는 경우 동일한 워크로드에 대해 EKS Pod Identity 연결을 구성하더라도 해당 보안 인증 정보는 계속 사용됩니다. 이렇게 하면 이전 보안 인증 정보를 제거하기 전에 먼저 연결을 생성하여 다른 유형의 보안 인증 정보에서 안전하게 마이그레이션할 수 있습니다.

컨테이너 보안 인증 정보 공급자는 각 노드에서 실행되는 에이전트의 임시 보안 인증 정보를 제공합니다. Amazon EKS에서 에이전트는 Amazon EKS Pod Identity Agent이고 Amazon Elastic Container Service에서 에이전트는 `amazon-ecs-agent`입니다. SDK는 환경 변수를 사용하여 연결할 에이전트를 찾습니다.

반대로 서비스 계정에 대한 IAM 역할은 AWS SDK가 `AssumeRoleWithWebIdentity` 사용을 통해 AWS Security Token Service와 교환해야 하는 웹 자격 증명 토큰을 제공합니다.

EKS Pod Identity Agent와 Pod의 작동 방식

1. Amazon EKS가 EKS Pod Identity 연결이 있는 서비스 계정을 사용하는 새 포드를 시작하면 클러스터는 Pod 매니페스트에 다음 콘텐츠를 추가합니다.

```
    env:
```

```
    - name: AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE
      value: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/eks-pod-
identity-token"
    - name: AWS_CONTAINER_CREDENTIALS_FULL_URI
      value: "http://169.254.170.23/v1/credentials"
    volumeMounts:
    - mountPath: "/var/run/secrets/pods.eks.amazonaws.com/serviceaccount/"
      name: eks-pod-identity-token
  volumes:
  - name: eks-pod-identity-token
    projected:
      defaultMode: 420
      sources:
      - serviceAccountToken:
          audience: pods.eks.amazonaws.com
          expirationSeconds: 86400 # 24 hours
          path: eks-pod-identity-token
```

2. Kubernetes는 포드를 실행할 노드를 선택합니다. 그러면 노드의 Amazon EKS Pod Identity Agent는 AssumeRoleForPodIdentity 작업을 사용하여 EKS Auth API에서 임시 보안 인증 정보를 검색합니다.
3. EKS Pod Identity Agent는 컨테이너 내에서 실행하는 AWS SDK에 이러한 보안 인증 정보를 사용할 수 있도록 합니다.
4. 기본 보안 인증 정보 체인을 사용하도록 보안 인증 정보 공급자를 지정하지 않고도 애플리케이션에서 SDK를 사용할 수 있습니다. 또는 컨테이너 보안 인증 정보 공급자를 지정합니다. 사용되는 기본 위치에 대한 자세한 내용은 AWS SDK 및 도구 참조 가이드의 보안 인증 정보 공급자 체인을 참조하세요.
5. SDK는 환경 변수를 사용하여 EKS Pod Identity Agent에 연결하고 보안 인증 정보를 검색합니다.

> **Note**
>
> 워크로드가 현재 보안 인증 정보 체인의 이전 단계에 있는 보안 인증 정보를 사용하는 경우 동일한 워크로드에 대해 EKS Pod Identity 연결을 구성하더라도 해당 보안 인증 정보는 계속 사용됩니다.

Amazon EKS Pod Identity Agent 설정

Amazon EKS Pod Identity는 Amazon EC2 인스턴스 프로파일이 Amazon EC2 인스턴스에 자격 증명을 제공하는 것과 비슷한 방식으로 애플리케이션에 대한 자격 증명을 관리하는 기능을 제공합니다.

Amazon EKS Pod Identity는 추가 EKS Auth API와 각 노드에서 실행되는 에이전트 포드를 통해 워크로드에 보안 인증 정보를 제공합니다.

Amazon EKS Pod Identity Agent 생성

에이전트 필수 조건

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요. 클러스터 버전 및 플랫폼 버전은 EKS Pod Identity 클러스터 버전에 나열된 버전과 동일한 버전 또는 이후 버전이어야 합니다.
- 노드 역할에는 에이전트가 EKS Auth API에서 `AssumeRoleForPodIdentity` 작업을 수행할 수 있는 권한이 있습니다. AWS 관리형 정책: AmazonEKSWorkerNodePolicy 사용 또는 다음과 유사한 사용자 지정 정책 추가가 가능합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks-auth:AssumeRoleForPodIdentity",
            ],
            "Resource": "*"
        }
    ]
}
```

  이 작업은 태그로 제한하여 에이전트를 사용하는 포드가 맡을 수 있는 역할을 제한할 수 있습니다.
- 노드는 Amazon ECR에 연결하여 이미지를 다운로드할 수 있습니다. 추가 기능의 컨테이너 이미지는 Amazon 컨테이너 이미지 레지스트리에 나열된 레지스트리에 있습니다.

  AWS Management Console의 선택적 구성 설정, 그리고 AWS CLI의 `--configuration-values`에서 이미지 위치를 변경하고 EKS 추가 기능에 대해 `imagePullSecrets`를 제공할 수 있습니다.
- 노드는 Amazon EKS Auth API에 연결할 수 있습니다. 프라이빗 클러스터의 경우 AWS PrivateLink의 `eks-auth` 엔드포인트가 필요합니다.

AWS Management Console

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 클러스터를 선택한 다음 EKS Pod Identity Agent 추가 기능을 구성할 클러스터의 이름을 선택합니다.
3. 추가 기능(Add-ons) 탭을 선택합니다.
4. 추가 기능 더 가져오기를 선택합니다.
5. EKS Pod Identity Agent 추가 기능 상자의 오른쪽 상단에 있는 상자를 선택하고 다음을 선택합니다.
6. 선택한 추가 기능 설정 구성 페이지의 버전 드롭다운 목록에서 임의의 버전을 선택합니다.
7. (선택 사항) 선택적 구성 설정을 확장하여 추가 구성을 입력합니다. 예를 들어 대체 컨테이너 이미지 위치 및 `ImagePullSecrets`를 제공할 수 있습니다. 허용된 키가 포함된 JSON Schema는 추가 기능 구성 스키마에 표시됩니다.

   구성 값에 구성 키와 값을 입력합니다.
8. 다음을 선택합니다.
9. EKS Pod Identity Agent 포드가 클러스터에서 실행 중인지 확인합니다.

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

예제 출력은 다음과 같습니다.

```
eks-pod-identity-agent-gmqp7                                          1/1
 Running   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1
 Running   1 (24h ago)   24h
```

이제 클러스터에서 EKS Pod Identity 연결을 사용할 수 있습니다. 자세한 내용은 EKS Pod Identity 관련 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 단원을 참조하십시오.

AWS CLI

1. 다음 AWS CLI 명령을 실행합니다. `my-cluster`를 클러스터 이름으로 바꿉니다.

```
aws eks create-addon --cluster-name my-cluster --addon-name eks-pod-identity-
agent --addon-version v1.0.0-eksbuild.1
```

> **Note**
>
> EKS Pod Identity Agent는 서비스 계정에 대한 IAM 역할에 `service-account-role-arn`을 사용하지 않습니다. EKS Pod Identity Agent에 노드 역할의 권한을 제공해야 합니다.

2. EKS Pod Identity Agent 포드가 클러스터에서 실행 중인지 확인합니다.

```
kubectl get pods -n kube-system | grep 'eks-pod-identity-agent'
```

예제 출력은 다음과 같습니다.

```
eks-pod-identity-agent-gmqp7                                          1/1
 Running   1 (24h ago)   24h
eks-pod-identity-agent-prnsh                                          1/1
 Running   1 (24h ago)   24h
```

이제 클러스터에서 EKS Pod Identity 연결을 사용할 수 있습니다. 자세한 내용은 EKS Pod Identity 관련 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 단원을 참조하십시오.

Amazon EKS Pod Identity Agent 업데이트

추가 기능의 Amazon EKS 유형 업데이트. 클러스터에 Amazon EKS 유형의 추가 기능을 추가하지 않은 경우 Amazon EKS Pod Identity Agent 생성 섹션을 참조하세요.

AWS Management Console

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 클러스터를 선택한 다음 EKS Pod Identity Agent 추가 기능을 구성할 클러스터의 이름을 선택합니다.
3. 추가 기능(Add-ons) 탭을 선택합니다.
4. 새 버전의 추가 기능을 사용할 수 있는 경우 EKS Pod Identity Agent에 버전 업데이트 버튼이 있습니다. 버전 업데이트를 선택합니다.
5. Amazon EKS Pod Identity Agent 구성 페이지의 버전 드롭다운 목록에서 새 버전을 선택합니다.
6. 변경 사항 저장(Save changes)을 선택합니다.

업데이트가 완료되는 데 몇 초 정도 걸릴 수 있습니다. 그런 다음 상태를 확인하여 추가 기능 버전이 업데이트되었는지 확인합니다.

AWS CLI

1. 클러스터에 설치된 추가 기능의 버전을 확인하세요. *my-cluster*을 클러스터 이름으로 교체합니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --query "addon.addonVersion" --output text
```

예제 출력은 다음과 같습니다.

```
v1.0.0-eksbuild.1
```

이 절차로 업데이트하려면 먼저 추가 기능을 생성해야 합니다.

2. AWS CLI를 사용하여 추가 기능 업데이트. AWS Management Console 또는 `eksctl`를 사용하여 추가 기능을 업데이트하려면 추가 기능 업데이트 부분을 참조하세요. 다음 명령을 디바이스에 복사합니다. 필요에 따라 명령을 다음과 같이 수정한 다음에 수정한 명령을 실행합니다.

- *my-cluster*를 클러스터 이름으로 바꿉니다.
- *v1.0.0-eksbuild.1*을 원하는 버전으로 바꿉니다.
- *111122223333*을 계정 ID로 바꿉니다.
- 다음 명령을 실행합니다:

```
aws eks update-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent --addon-version v1.0.0-eksbuild.1'
```

업데이트가 완료되는 데 몇 초 정도 걸릴 수 있습니다.

3. 추가 기능 버전이 업데이트되었는지 확인합니다. *my-cluster*를 클러스터 이름으로 바꿉니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name eks-pod-identity-agent
```

업데이트가 완료되는 데 몇 초 정도 걸릴 수 있습니다.

예제 출력은 다음과 같습니다.

```
{
    "addon": {
        "addonName": "eks-pod-identity-agent",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.0.0-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/eks-pod-
identity-agent/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
        "createdAt": "2023-04-12T18:25:19.319000+00:00",
        "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
        "tags": {}
    }
}
```

EKS Pod Identity 관련 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성

이 주제에서는 EKS Pod Identity 관련 AWS Identity and Access Management(IAM) 역할을 수임하도록 Kubernetes 서비스 계정을 구성하는 방법을 다룹니다. 서비스 계정을 사용하도록 구성된 모든 Pods는 해당 역할에 액세스 권한이 있는 모든 AWS 서비스에 액세스할 수 있습니다.

EKS Pod Identity 연결을 생성하려면 한 단계만 거치면 됩니다. AWS Management Console, AWS CLI, AWS SDK AWS CloudFormation 및 기타 도구를 통해 EKS에서 연결을 생성하면 됩니다. 임의 Kubernetes 개체의 클러스터 내부에 연결에 대한 데이터나 메타데이터가 없으므로 서비스 계정에 주석을 추가하지 않습니다.

필수 조건

- 기존 클러스터가 있어야 합니다. 아직 없는 경우 Amazon EKS 시작하기 안내서 중 하나에 따라 생성할 수 있습니다.
- 연결을 생성하는 IAM 보안 주체에 `iam:PassRole`이 있어야 합니다.
- AWS CLI의 최신 버전이 디바이스나 AWS CloudShell에 설치 및 구성되어 있어야 합니다. `aws --version | cut -d / -f2 | cut -d ' ' -f1`을 사용하여 현재 버전을 확인할 수 있습니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명

서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.

- 디바이스 또는 AWS CloudShell에 설치된 kubectl 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 1.28인 경우 kubectl 버전 1.27, 1.28 또는 1.29를 함께 사용할 수 있습니다. kubectl을 설치하거나 업그레이드하려면 kubectl 설치 또는 업데이트 부분을 참조하세요.
- 클러스터 구성이 포함된 기존 kubectl config 파일입니다. kubectl config 파일을 생성하려면 Amazon EKS 클러스터용 kubeconfig 파일 생성 또는 업데이트 섹션을 참조하세요.

EKS Pod Identity 연결 생성

AWS Management Console

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 클러스터를 선택한 다음 EKS Pod Identity Agent 추가 기능을 구성할 클러스터의 이름을 선택합니다.
3. 액세스 탭을 선택합니다.
4. Pod Identity 연결에서 생성을 선택합니다.
5. IAM 역할의 경우 워크로드에 부여하려는 권한이 있는 IAM 역할을 선택합니다.

> **Note**
>
> 이 목록에는 EKS Pod Identity에서의 사용을 허용하는 다음 신뢰 정책이 있는 역할만 포함되어 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
```

```
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`

EKS Pod Identity는 `TagSession`을 사용하여 임시 보안 인증 정보를 포드에 전달하기 전에 IAM 역할을 맡습니다.

`sts:TagSession`

EKS Pod Identity는 `TagSession`을 사용하여 세션 태그를 AWS STS 요청에 포함합니다.

신뢰 정책의 condition keys에서 이러한 태그를 사용하여 이 역할을 사용할 수 있는 서비스 계정, 네임스페이스 및 클러스터를 제한할 수 있습니다.

Amazon EKS 조건 키 목록을 보려면 서비스 인증 참조의 Amazon Elastic Kubernetes Service의 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon Elastic Kubernetes Service에서 정의한 작업을 참조하세요.

6. Kubernetes 네임스페이스의 경우 서비스 계정 및 워크로드가 포함된 Kubernetes 네임스페이스를 선택합니다. 클러스터에 없는 네임스페이스를 이름으로 지정할 수 있습니다.
7. Kubernetes 서비스 계정의 경우 사용할 Kubernetes 서비스 계정을 선택합니다. Kubernetes 워크로드의 매니페스트에서 이 서비스 계정을 지정해야 합니다. 클러스터에 없는 서비스 계정을 이름으로 지정할 수 있습니다.
8. (선택 사항) 태그의 경우 태그 추가를 선택하여 키-값 페어에 메타데이터를 추가합니다. 이러한 태그는 연결에 적용되며 IAM 정책에서 사용할 수 있습니다.

   이 단계를 반복하여 여러 개의 태그를 추가할 수 있습니다.
9. 생성(Create)을 선택합니다.

AWS CLI

1. IAM 역할에 기존 IAM 정책을 연결하려면 다음 단계로 건너뜁니다.

IAM 정책을 생성합니다. 자체 정책을 생성하거나 필요한 권한 중 일부를 이미 부여하는 AWS 관리형 정책을 복사하여 특정 요구 사항에 맞게 사용자 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서에서 IAM 정책 생성을 참조하세요.

a. Pods가 액세스할 AWS 서비스에 대한 권한이 포함된 파일을 생성합니다. 모든 AWS 서비스에 대한 모든 작업 목록은 서비스 승인 참조에서 확인하세요.

   다음 명령을 실행하여 Amazon S3 버킷에 대한 읽기 전용 액세스를 허용하는 예제 정책 파일을 생성할 수 있습니다. 이 버킷에 구성 정보 또는 부트스트랩 스크립트를 저장할 수도 있고, Pod의 컨테이너는 이 버킷에서 파일을 읽어 애플리케이션으로 로드할 수 있습니다. 이 예제 정책을 생성하려면 다음 내용을 디바이스에 복사합니다. *my-pod-secrets-bucket*을 버킷 이름으로 바꾸고 명령을 실행합니다.

```
cat >my-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
EOF
```

b. IAM 정책을 생성합니다.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

2. IAM 역할을 생성하고 Kubernetes 서비스 계정과 연결합니다.

1. IAM 역할을 수임하려는 기존 Kubernetes 서비스 계정이 있는 경우 이 단계를 건너뛸 수 있습니다.

   Kubernetes 서비스 계정을 생성합니다. 다음 콘텐츠를 디바이스에 복사합니다. *my-service-account*를 원하는 이름으로 바꾸고 필요한 경우 *default*를 다른 네임스페이스로 바꿉니다. *default*를 변경하는 경우 네임스페이스가 이미 존재해야 합니다.

```
cat >my-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-service-account
  namespace: default
EOF
kubectl apply -f my-service-account.yaml
```

다음 명령을 실행합니다.

```
kubectl apply -f my-service-account.yaml
```

2. 다음 명령을 실행하여 IAM 역할에 대한 신뢰 정책 파일을 생성합니다.

```
cat >trust-relationship.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
EOF
```

3. 역할을 생성합니다. *my-role*을 IAM 역할의 이름으로, *my-role-description*를 역할에 대한 간략한 설명으로 바꿉니다.

```
aws iam create-role --role-name my-role --assume-role-policy-document
 file://trust-relationship.json --description "my-role-description"
```

4. 역할에 IAM 정책을 연결합니다. *my-role*을 IAM 역할의 이름으로 바꾸고 *my-policy*를 생성한 기존 정책의 이름으로 바꿉니다.

```
aws iam attach-role-policy --role-name my-role --policy-
arn=arn:aws:iam::111122223333:policy/my-policy
```

> **Note**
>
> 서비스 계정에 대한 IAM 역할과 달리 EKS Pod Identity는 서비스 계정의 주석을 사용하지 않습니다.

5. 다음 명령을 실행하여 연결을 생성합니다. `my-cluster`를 클러스터 이름으로 바꾸고, 필요한 경우 *my-service-account*를 원하는 이름으로 바꾸고 *default*를 다른 네임스페이스로 바꿉니다.

```
aws eks create-pod-identity-association --cluster-name my-cluster --role-
arn arn:aws:iam::111122223333:role/my-role --namespace default --service-
account my-service-account
```

예제 출력은 다음과 같습니다.

```
{
    "association": {
        "clusterName": "my-cluster",
        "namespace": "default",
        "serviceAccount": "my-service-account",
        "roleArn": "arn:aws:iam::111122223333:role/my-role",
        "associationArn": "arn:aws::111122223333:podidentityassociation/my-
cluster/a-abcdefghijklmnop1",
        "associationId": "a-abcdefghijklmnop1",
        "tags": {},
        "createdAt": 1700862734.922,
        "modifiedAt": 1700862734.922
    }
}
```

> **Note**
>
> 클러스터에 없는 네임스페이스와 서비스 계정을 이름으로 지정할 수 있습니다. EKS Pod Identity 연결이 작동하려면 네임스페이스, 서비스 계정 및 서비스 계정을 사용하는 워크로드를 생성해야 합니다.

3. 역할 및 서비스 계정이 올바르게 구성되었는지 확인합니다.

   a. IAM 역할의 신뢰 정책이 올바르게 구성되었는지 확인합니다.

   ```
   aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
   ```

   예제 출력은 다음과 같습니다.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Sid": "Allow EKS Auth service to assume this role for Pod
    Identities",
               "Effect": "Allow",
               "Principal": {
                   "Service": "pods.eks.amazonaws.com"
               },
               "Action": [
                   "sts:AssumeRole",
                   "sts:TagSession"
               ]
           }
       ]
   }
   ```

   b. 이전 단계에서 역할에 연결한 정책이 해당 역할에 연결되었는지 확인합니다.

   ```
   aws iam list-attached-role-policies --role-name my-role --query
    AttachedPolicies[].PolicyArn --output text
   ```

   예제 출력은 다음과 같습니다.

```
arn:aws:iam::111122223333:policy/my-policy
```

c. 사용하려는 정책의 Amazon 리소스 이름(ARN)을 저장할 변수를 설정합니다. *my-policy*를 권한을 확인하려는 정책의 이름으로 바꿉니다.

```
export policy_arn=arn:aws:iam::111122223333:policy/my-policy
```

d. 정책의 기본 버전을 봅니다.

```
aws iam get-policy --policy-arn $policy_arn
```

예제 출력은 다음과 같습니다.

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws:iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

e. 정책 내용을 보고 Pod에 필요한 모든 권한이 정책에 포함되어 있는지 확인합니다. 필요한 경우 다음 명령에서 *1*을 이전 출력에서 반환된 버전으로 바꿉니다.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

예제 출력은 다음과 같습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
```

```
}
```

이전 단계에서 예제 정책을 생성한 경우 출력은 동일합니다. 다른 정책을 생성한 경우 ## 내용이 다릅니다.

다음 단계

Kubernetes 서비스 계정을 사용하도록 Pods 구성

Kubernetes 서비스 계정을 사용하도록 Pods 구성

Pod가 AWS 서비스에 액세스해야 하는 경우 Kubernetes 서비스 계정을 사용하도록 구성해야 합니다. 서비스 계정은 AWS 서비스에 액세스할 수 있는 권한이 있는 AWS Identity and Access Management(IAM) 역할에 연결되어야 합니다.

필수 조건

- 기존 클러스터가 있어야 합니다. 아직 없는 경우 Amazon EKS 시작하기 가이드 중 하나를 사용하여 생성할 수 있습니다.
- 기존 Kubernetes 서비스 계정 및 서비스 계정을 IAM 역할에 연결하는 EKS Pod Identity 연결 Pods가 AWS 서비스를 사용하는 데 필요한 권한을 포함하는 연결된 IAM 정책이 역할에 있어야 합니다. 서비스 계정 및 역할을 만들고 구성하는 방법에 대한 자세한 내용을 알아보려면 EKS Pod Identity 관련 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 섹션을 참조하세요.
- AWS CLI의 최신 버전이 디바이스나 AWS CloudShell에 설치 및 구성되어 있어야 합니다. `aws --version | cut -d / -f2 | cut -d ' ' -f1`을 사용하여 현재 버전을 확인할 수 있습니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 `aws configure`를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.
- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 `kubectl` 설치 또는 업데이트 부분을 참조하세요.
- 클러스터 구성이 포함된 기존 `kubectl config` 파일입니다. `kubectl config` 파일을 생성하려면 Amazon EKS 클러스터용 `kubeconfig` 파일 생성 또는 업데이트 섹션을 참조하세요.

서비스 계정을 사용하도록 Pod 구성

1. 다음 명령을 사용하여 구성을 확인할 Pod를 배포할 수 있는 배포 매니페스트를 생성합니다. *example values*를 고유한 값으로 바꿉니다.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. 클러스터에 매니페스트를 배포합니다.

```
kubectl apply -f my-deployment.yaml
```

3. Pod에 필요한 환경 변수가 있는지 확인합니다.

   a. 이전 단계에서 배포와 함께 배포된 Pods를 봅니다.

   ```
   kubectl get pods | grep my-app
   ```

   예제 출력은 다음과 같습니다.

   ```
   my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
   ```

   b. Pod에 서비스 계정 토큰 파일 탑재가 있는지 확인합니다.

```
kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
 AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:
```

예제 출력은 다음과 같습니다.

```
AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE:  /var/run/secrets/
pods.eks.amazonaws.com/serviceaccount/eks-pod-identity-token
```

4. 역할에 연결된 IAM 정책에서 할당한 권한을 사용하여 Pods가 AWS 서비스와 상호 작용할 수 있는지 확인합니다.

> **Note**
>
> Pod가 서비스 계정과 연결된 IAM 역할의 AWS 자격 증명을 사용하는 경우 해당 AWS CLI의 컨테이너에 있는 Pod 또는 기타 SDK에서는 이 역할이 제공하는 자격 증명을 사용합니다. Amazon EKS 노드 IAM 역할에 제공된 자격 증명에 대한 액세스를 제한하지 않으면 Pod는 계속해서 해당 자격 증명에 액세스할 수 있습니다. 자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 부분을 참조하세요.

Pods가 예상대로 서비스와 상호 작용할 수 없으면 다음 단계를 완료하여 모두 제대로 구성되었는지 확인합니다.

a. Pods에서 EKS Pod Identity 연결을 통해 IAM 역할을 수임할 수 있도록 지원하는 AWS SDK 버전을 사용하는지 확인합니다. 자세한 내용은 지원되는 AWS SDK 사용 단원을 참조하십시오.

b. 배포에서 서비스 계정을 사용하고 있는지 확인합니다.

```
kubectl describe deployment my-app | grep "Service Account"
```

예제 출력은 다음과 같습니다.

```
Service Account:  my-service-account
```

EKS Pod Identity가 태그를 기반으로 역할을 맡을 수 있는 권한을 정의합니다.

EKS Pod Identity는 클러스터 이름, 네임스페이스, 서비스 계정 이름 등의 속성을 사용하여 각 포드의 임시 보안 인증 정보에 태그를 연결합니다. 관리자는 이러한 역할 세션 태그를 사용하여 일치하는 태그를 기반으로 AWS 리소스에 대한 액세스를 허용하여 여러 서비스 계정에서 작업할 수 있는 단일 역할을 작성할 수 있습니다. 역할 세션 태그에 대한 지원을 추가함으로써 고객은 동일한 IAM 역할과 IAM 정책을 재사용하면서 클러스터 간 및 클러스터 내 워크로드 간에 더 엄격한 보안 경계를 적용할 수 있습니다.

예를 들어 다음 정책에서는 객체에 EKS 클러스터 이름이 태그가 지정된 경우 `s3:GetObject` 작업을 허용합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/eks-cluster-name": "${aws:PrincipalTag/eks-
cluster-name}"
                }
            }
        }
    ]
}
```

EKS Pod Identity에서 추가한 세션 태그 목록

다음 목록에는 Amazon EKS의 `AssumeRole` 요청에 추가된 태그의 모든 키가 포함되어 있습니다. 정책에서 이러한 태그를 사용하려면 `${aws:PrincipalTag/` 다음에 키를 사용합니다(예: `${aws:PrincipalTag/kubernetes-namespace}`).

- `eks-cluster-arn`
- `eks-cluster-name`
- `kubernetes-namespace`
- `kubernetes-service-account`
- `kubernetes-pod-name`
- `kubernetes-pod-uid`

교차 계정 태그

EKS Pod Identity에서 추가한 모든 세션 태그는 전이적이며, 워크로드가 역할을 다른 계정으로 전환하는 데 사용하는 모든 `AssumeRole` 작업에 태그 키와 값이 전달됩니다. 다른 계정의 정책에 이러한 태그를 사용하여 교차 계정 시나리오에서 액세스를 제한할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 세션 태그를 사용하는 역할 체인을 참조하세요.

사용자 지정 태그

EKS Pod Identity는 수행하는 `AssumeRole` 작업에 사용자 지정 태그를 추가할 수 없습니다. 하지만 IAM 역할에 적용하는 태그는 항상 동일한 형식(`${aws:PrincipalTag/` 다음에 키 사용)을 통해 사용 가능합니다(예: `${aws:PrincipalTag/MyCustomTag}`).

> **Note**
>
> `sts:AssumeRole` 요청을 통해 세션에 추가된 태그가 충돌하는 경우 우선시됩니다. 예를 들어 EKS가 고객 역할을 수임할 때 Amazon EKS가 세션에 `eks-cluster-name` 키와 `my-cluster` 값을 추가한다고 가정해 보겠습니다. 또한 IAM 역할에 `my-own-cluster` 값이 있는 `eks-cluster-name` 태그를 추가했습니다. 이 경우 전자가 우선하여 `eks-cluster-name` 태그의 값은 `my-cluster`입니다.

지원되는 AWS SDK 사용

> **Important**
>
> 이전 버전의 설명서가 잘못되었습니다. AWS SDK for Java v1은 EKS Pod Identity를 지원하지 않습니다.

EKS Pod Identity 사용 시 Pods에 있는 컨테이너는 EKS Pod Identity Agent에서 IAM 역할 수임을 지원하는 AWS SDK 버전을 사용해야 합니다. AWS SDK에 대해 다음과 같은 버전 또는 이후 버전을 사용해야 합니다.

- Java(버전 2) – 2.21.30
- Go v1 – v1.47.11
- Go v2 – release-2023-11-14
- Python(Boto3) – 1.34.41
- Python(botocore) – 1.32.0
- AWS CLI – 1.30.0

  AWS CLI – 2.15.0
- JavaScript v2 – 2.1550.0
- JavaScript v3 – 3.458.0
- Ruby – 3.188.0
- C++ - 1.11.263
- .NET - 3.7.734.0 -
- PHP – 3.287.1

지원되는 SDK를 사용 중인지 확인하려면 컨테이너를 빌드할 때 AWS에서의 구축을 위한 도구에서 선호하는 SDK에 대한 설치 지침을 따르세요.

EKS Pod Identity 보안 인증 정보 사용

EKS Pod Identity 연결의 보안 인증 정보를 사용하려면 코드에서 임의의 AWS SDK를 사용하여 SDK가 포함된 AWS 서비스에 대한 클라이언트를 만들 수 있고, 기본적으로 SDK는 일련의 위치에서 사용할 AWS Identity and Access Management 보안 인증 정보를 검색합니다. 클라이언트를 생성하거나 SDK

를 초기화했을 때 보안 인증 정보 공급자를 지정하지 않으면 EKS Pod Identity 보안 인증 정보가 사용됩니다.

이는 기본 보안 인증 정보 체인의 한 단계에서 검색되는 컨테이너 보안 인증 정보 공급자에 EKS Pod Identity가 추가되기 때문에 가능합니다. 워크로드가 현재 보안 인증 정보 체인의 이전 단계에 있는 보안 인증 정보를 사용하는 경우 동일한 워크로드에 대해 EKS Pod Identity 연결을 구성하더라도 해당 보안 인증 정보는 계속 사용됩니다.

EKS Pod Identity 작동 방식에 대한 자세한 내용은 EKS Pod Identity 작동 방식 섹션을 참조하세요.

EKS Pod Identity 역할

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowEksAuthToAssumeRoleForPodIdentity",
            "Effect": "Allow",
            "Principal": {
                "Service": "pods.eks.amazonaws.com"
            },
            "Action": [
                "sts:AssumeRole",
                "sts:TagSession"
            ]
        }
    ]
}
```

`sts:AssumeRole`

EKS Pod Identity는 `TagSession`을 사용하여 임시 보안 인증 정보를 포드에 전달하기 전에 IAM 역할을 맡습니다.

`sts:TagSession`

EKS Pod Identity는 `TagSession`을 사용하여 세션 태그를 AWS STS 요청에 포함합니다.

신뢰 정책의 condition keys에서 이러한 태그를 사용하여 이 역할을 사용할 수 있는 서비스 계정, 네임스페이스 및 클러스터를 제한할 수 있습니다.

Amazon EKS 조건 키 목록을 보려면 서비스 인증 참조의 Amazon Elastic Kubernetes Service의 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon Elastic Kubernetes Service에서 정의한 작업을 참조하세요.

서비스 계정에 대한 IAM 역할

Pod의 컨테이너에 있는 애플리케이션에서는 AWS SDK 또는 AWS CLI를 통해 AWS Identity and Access Management(IAM) 권한을 사용하여 AWS 서비스에 API를 요청할 수 있습니다. 애플리케이션은 AWS 자격 증명으로 AWS API 요청에 서명해야 합니다. 서비스 계정에 대한 IAM 역할은 Amazon EC2 인스턴스 프로파일이 Amazon EC2 인스턴스에 자격 증명을 제공하는 것과 비슷한 방식으로 애플리케이션에 대한 자격 증명을 관리하는 기능을 제공합니다. AWS 자격 증명을 생성하여 컨테이너에 배포하거나 Amazon EC2 인스턴스의 역할을 사용하는 대신에 IAM 역할을 Kubernetes 서비스 계정과 연결하고 서비스 계정을 사용하도록 Pods를 구성합니다. AWS Outposts의 Amazon EKS에 대한 로컬 클러스터가 있는 서비스 계정에는 IAM 역할을 사용할 수 없습니다.

서비스 계정의 IAM 역할을 통해 다음과 같은 이점을 누릴 수 있습니다.

- 최소 권한 – IAM 권한의 범위를 서비스 계정으로 지정할 수 있습니다. 그러면 해당 서비스 계정을 사용하는 Pods만 이 권한에 액세스할 수 있습니다. 또한 이 기능을 사용하면 `kiam`, `kube2iam` 같은 타사 솔루션이 필요 없습니다.
- 자격 증명 격리 - Pod's 컨테이너는 컨테이너가 사용하는 서비스 계정과 연결된 IAM 역할에 대한 자격 증명만 검색할 수 있습니다. 컨테이너는 다른 Pods의 다른 컨테이너에서 사용하는 자격 증명에 액세스할 수 없습니다. 서비스 계정에 대해 IAM 역할을 사용할 때 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 대한 Pod 액세스를 차단하지 않는 한 Pod's 컨테이너는 Amazon EKS 노드 IAM 역할에 할당된 권한도 갖습니다. 자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 부분을 참조하세요.
- 감사 - 소급적 감사를 위해 AWS CloudTrail을 통해 액세스 및 이벤트 로깅이 가능합니다.

다음 절차를 완료하여 서비스 계정에 대한 IAM 역할을 활성화합니다.

1. 클러스터에 대한 IAM OIDC 공급자 생성 - 각 클러스터에 대해 한 번만 이 절차를 완료합니다.

> **Note**
>
> EKS VPC 엔드포인트를 활성화할 경우 해당 VPC 내에서 EKS OIDC 서비스 엔드포인트에 액세스할 수 없습니다. 따라서 VPC에서 `eksctl`을 통해 OIDC 공급자를 생성하는 등의 작

업은 작동하지 않으며 `https://oidc.eks.region.amazonaws.com` 요청을 시도할 때 시간 초과가 발생합니다. 오류 메시지의 예는 다음과 같습니다.

```
** server can't find oidc.eks.region.amazonaws.com: NXDOMAIN
```

이 단계를 완료하려면 VPC 외부(예: AWS CloudShell 또는 인터넷에 연결된 컴퓨터)에서 명령을 실행할 수 있습니다.

2. IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 - 애플리케이션에 부여하려는 고유한 권한 세트에 대해 이 절차를 완료합니다.
3. Kubernetes 서비스 계정을 사용하도록 Pods 구성 - AWS 서비스에 액세스해야 하는 각 Pod에 대해 이 절차를 완료합니다.
4. 지원되는 AWS SDK 사용 - 워크로드가 지원되는 버전의 AWS SDK를 사용하고 워크로드가 기본 보안 인증 정보 체인을 사용하는지 확인합니다.

IAM, Kubernetes 및 OpenID Connect(OIDC) 배경 정보

2014년 AWS Identity and Access Management에서는 OpenID Connect(OIDC)를 사용하여 페더레이션형 ID를 지원하는 기능을 추가했습니다. 이 기능을 사용하면 지원되는 ID 제공업체를 이용해 AWS API 호출을 인증하고 유효한 OIDC JSON 웹 토큰(JWT)을 수신할 수 있습니다. 이 토큰을 AWS STS `AssumeRoleWithWebIdentity` API 작업에 전달하고 IAM 임시 역할 자격 증명을 수신할 수 있습니다. 이 자격 증명을 사용하여 Amazon S3 및 DynamoDB와 같은 AWS 서비스와 상호 작용할 수 있습니다.

각 JWT 토큰은 서명 키 페어로 서명됩니다. 키는 Amazon EKS에서 관리하는 OIDC 공급자가 제공하며 프라이빗 키는 7일마다 교체됩니다. Amazon EKS는 퍼블릭 키를 만료될 때까지 보관합니다. 외부 OIDC 클라이언트를 연결하는 경우 퍼블릭 키가 만료되기 전에 서명 키를 갱신해야 합니다. the section called “서명 키 가져오기” 방법에 대해 알아봅니다.

Kubernetes는 오랫동안 서비스 계정을 자체 내부 ID 시스템으로 사용해 왔습니다. Pods는 Kubernetes API 서버만 검증할 수 있는 자동 탑재 토큰(OIDC가 아닌 JWT)을 사용하여 Kubernetes API 서버에 인증할 수 있습니다. 이러한 레거시 서비스 계정 토큰은 만료되지 않으며, 서명 키를 교체하려면 까다로운 프로세스를 거쳐야 합니다. Kubernetes 버전 `1.12`에서 새로운 `ProjectedServiceAccountToken` 기능에 대한 지원이 추가되었습니다. 이 기능은 서비스 계정 ID도 포함된 OIDC JSON 웹 토큰이며 구성 가능한 대상을 지원합니다.

Amazon EKS는 `ProjectedServiceAccountToken` JSON 웹 토큰에 대한 서명 키가 포함된 각 클러스터에 대한 퍼블릭 OIDC 검색 엔드포인트를 호스팅하므로 IAM과 같은 외부 시스템이 Kubernetes에서 발급한 OIDC 토큰을 확인하고 수락할 수 있습니다.

클러스터에 대한 IAM OIDC 공급자 생성

클러스터에는 OpenID Connect(OIDC) 발급자 URL이 연결되어 있습니다. 서비스 계정에 AWS Identity and Access Management(IAM) 역할을 사용하려면 클러스터의 OIDC 발급자 URL에 IAM OIDC 제공업체가 있어야 합니다.

필수 조건

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요.
- AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.
- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 `kubectl` 설치 또는 업데이트 부분을 참조하세요.
- 클러스터 구성이 포함된 기존 `kubectl config` 파일입니다. `kubectl config` 파일을 생성하려면 Amazon EKS 클러스터용 `kubeconfig` 파일 생성 또는 업데이트 섹션을 참조하세요.

`eksctl` 또는 AWS Management Console을 사용하여 클러스터에 대한 IAM OIDC 제공업체를 생성할 수 있습니다.

eksctl

전제 조건

디바이스 또는 `0.175.0`에 설치된 버전 AWS CloudShell 이상의 `eksctl` 명령줄 도구. `eksctl`을 설치 또는 업그레이드하려면 `eksctl` 설명서에서 Installation을 참조하세요.

eksctl로 클러스터의 IAM OIDC ID 제공업체 생성

1. 클러스터의 OIDC 발행자 ID를 판단합니다.

   클러스터의 OIDC 발행자 ID를 검색하고 변수에 저장합니다. *my-cluster*를 사용자의 고유한 값으로 교체합니다.

   ```
   cluster_name=my-cluster
   ```

   ```
   oidc_id=$(aws eks describe-cluster --name $cluster_name --query
    "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
   ```

   ```
   echo $oidc_id
   ```

2. 클러스터의 발행자 ID를 가진 IAM OIDC 제공업체가 이미 계정에 있는지 판단합니다.

   ```
   aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
   ```

   출력이 반환되면 클러스터에 대한 IAM OIDC 제공업체가 이미 있는 것이므로 다음 단계를 건너뛸 수 있습니다. 출력이 반환되지 않은 경우 해당 클러스터에 대한 IAM OIDC 제공업체를 생성해야 합니다.

3. 다음 명령을 사용하여 클러스터의 IAM OIDC ID 제공업체를 생성합니다.

   ```
   eksctl utils associate-iam-oidc-provider --cluster $cluster_name --approve
   ```

   > **Note**
   >
   > EKS VPC 엔드포인트를 활성화할 경우 해당 VPC 내에서 EKS OIDC 서비스 엔드포인트에 액세스할 수 없습니다. 따라서 VPC에서 `eksctl`을 통해 OIDC 공급자를 생성하는 등의 작업은 작동하지 않으며 `https://oidc.eks.region.amazonaws.com` 요청을 시도할 때 시간 초과가 발생합니다. 오류 메시지의 예는 다음과 같습니다.
   >
   > ```
   > ** server can't find oidc.eks.region.amazonaws.com: NXDOMAIN
   > ```
   >
   > 이 단계를 완료하려면 VPC 외부(예: AWS CloudShell 또는 인터넷에 연결된 컴퓨터)에서 명령을 실행할 수 있습니다.

AWS Management Console

AWS Management Console로 클러스터의 IAM OIDC ID 제공업체 생성

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 왼쪽 창에서 클러스터(Clusters)를 선택한 다음 클러스터(Clusters) 페이지에서 클러스터 이름을 선택합니다.
3. 개요(Overview) 탭의 세부 정보(Details) 섹션에서 OpenID Connect 공급자 URL(OpenID Connect provider URL)의 값을 적어 둡니다.
4. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
5. 왼쪽 탐색 창의 액세스 관리(Access management)에서 자격 증명 공급자(Identity Providers)를 선택합니다. 클러스터의 URL과 일치하는 공급자가 목록에 있으면 클러스터에 대한 공급자가 이미 있는 것입니다. 클러스터의 URL과 일치하는 공급자가 나열되지 않는 경우 공급자를 생성해야 합니다.
6. 제공업체를 생성하려면 Add Provider(공급자 추가)를 선택합니다.
7. Provider type(제공업체 유형)에서 OpenID Connect를 선택합니다.
8. Provider URL(제공업체 URL)에 클러스터의 OIDC 제공업체 URL을 입력하고 Get thumbprint(지문 가져오기)를 선택합니다.
9. [대상(Audience)에서 **`sts.amazonaws.com`**을 입력하고 [공급자 추가(Add provider)]를 선택합니다.

다음 단계

IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성

IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성

이 주제에서는 AWS Identity and Access Management(IAM) 역할을 수임하도록 Kubernetes 서비스 계정을 구성하는 방법을 다룹니다. 서비스 계정을 사용하도록 구성된 모든 Pods는 해당 역할에 액세스 권한이 있는 모든 AWS 서비스에 액세스할 수 있습니다.

필수 조건

- 기존 클러스터가 있어야 합니다. 아직 없는 경우 Amazon EKS 시작하기 안내서 중 하나에 따라 생성할 수 있습니다.
- 클러스터에 대한 기존 IAM OpenID Connect(OIDC) 제공업체입니다. 이미 있는지 확인하거나 생성하는 방법을 알아보려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요.

- AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.
- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 `kubectl` 설치 또는 업데이트 부분을 참조하세요.
- 클러스터 구성이 포함된 기존 `kubectl config` 파일입니다. `kubectl config` 파일을 생성하려면 Amazon EKS 클러스터용 `kubeconfig` 파일 생성 또는 업데이트 섹션을 참조하세요.

Kubernetes 서비스 계정과 IAM 역할 연결

1. IAM 역할에 기존 IAM 정책을 연결하려면 다음 단계로 건너뜁니다.

   IAM 정책을 생성합니다. 자체 정책을 생성하거나 필요한 권한 중 일부를 이미 부여하는 AWS 관리형 정책을 복사하여 특정 요구 사항에 맞게 사용자 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서에서 IAM 정책 생성을 참조하세요.

   a. Pods가 액세스할 AWS 서비스에 대한 권한이 포함된 파일을 생성합니다. 모든 AWS 서비스에 대한 모든 작업 목록은 서비스 승인 참조에서 확인하세요.

      다음 명령을 실행하여 Amazon S3 버킷에 대한 읽기 전용 액세스를 허용하는 예제 정책 파일을 생성할 수 있습니다. 이 버킷에 구성 정보 또는 부트스트랩 스크립트를 저장할 수도 있고, Pod의 컨테이너는 이 버킷에서 파일을 읽어 애플리케이션으로 로드할 수 있습니다. 이 예제 정책을 생성하려면 다음 내용을 디바이스에 복사합니다. ***`my-pod-secrets-bucket`***을 버킷 이름으로 바꾸고 명령을 실행합니다.

```
cat >my-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
        }
    ]
}
EOF
```

b. IAM 정책을 생성합니다.

```
aws iam create-policy --policy-name my-policy --policy-document file://my-policy.json
```

2. IAM 역할을 생성하고 Kubernetes 서비스 계정과 연결합니다. eksctl 또는 AWS CLI를 사용할 수 있습니다.

eksctl

전제 조건

디바이스 또는 0.175.0에 설치된 버전 AWS CloudShell 이상의 eksctl 명령줄 도구. eksctl을 설치 또는 업그레이드하려면 eksctl 설명서에서 Installation을 참조하세요.

*my-service-account*를 eksctl이 생성하고 IAM 역할과 연결할 Kubernetes 서비스 계정의 이름으로 바꿉니다. *default*를 eksctl이 서비스 계정을 생성할 네임스페이스로 바꿉니다. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다. *my-role*을 서비스 계정을 연결할 역할의 이름으로 바꿉니다. 역할이 아직 없는 경우 eksctl이 자동으로 생성합니다. *111122223333*을 계정 ID로 바꾸고 *my-policy*를 기존 정책의 이름으로 바꿉니다.

```
eksctl create iamserviceaccount --name my-service-account --namespace default --cluster my-cluster --role-name my-role \
    --attach-policy-arn arn:aws:iam::111122223333:policy/my-policy --approve
```

> **Important**
>
> 역할 또는 서비스 계정이 이미 있는 경우 이전 명령이 실패할 수 있습니다. eksctl에는 이러한 상황에서 제공할 수 있는 다양한 옵션이 있습니다. 자세한 내용을 알아보려면 **eksctl create iamserviceaccount --help**를 실행하세요.

AWS CLI

1. IAM 역할을 수임하려는 기존 Kubernetes 서비스 계정이 있는 경우 이 단계를 건너뛸 수 있습니다.

   Kubernetes 서비스 계정을 생성합니다. 다음 콘텐츠를 디바이스에 복사합니다. *my-service-account*를 원하는 이름으로 바꾸고 필요한 경우 *default*를 다른 네임스페이스로 바꿉니다. *default*를 변경하는 경우 네임스페이스가 이미 존재해야 합니다.

   ```
   cat >my-service-account.yaml <<EOF
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: my-service-account
     namespace: default
   EOF
   kubectl apply -f my-service-account.yaml
   ```

2. 다음 명령을 사용해 AWS 계정 ID를 환경 변수로 설정합니다.

   ```
   account_id=$(aws sts get-caller-identity --query "Account" --output text)
   ```

3. 다음 명령을 사용해 클러스터의 OIDC ID 제공업체를 환경 변수로 설정합니다. *my-cluster*를 클러스터 이름으로 바꿉니다.

   ```
   oidc_provider=$(aws eks describe-cluster --name my-cluster --region
    $AWS_REGION --query "cluster.identity.oidc.issuer" --output text | sed -e "s/
   ^https:\/\///")
   ```

4. 서비스 계정의 네임스페이스 및 이름에 대한 변수를 설정합니다. *my-service-account*를 역할을 수임할 Kubernetes 서비스 계정으로 바꿉니다. *default*를 서비스 계정의 네임스페이스로 바꿉니다.

   ```
   export namespace=default
   export service_account=my-service-account
   ```

5. 다음 명령을 실행하여 IAM 역할에 대한 신뢰 정책 파일을 생성합니다. 네임스페이스 내의 모든 서비스 계정이 역할을 사용하도록 허용하려면 다음 내용을 디바이스에 복사합니다. *StringEquals*를 **StringLike**로 바꾸고 *$service_account*를 *로 바꿉니다.

여러 서비스 계정 또는 네임스페이스가 역할을 수임할 수 있도록 StringEquals 또는 StringLike 조건에 여러 항목을 추가할 수 있습니다. 클러스터가 속한 계정과 다른 AWS 계정의 역할이 역할을 수임하도록 허용하려면 교차 계정 IAM 권한 섹션에서 자세한 내용을 참조하세요.

```
cat >trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::$account_id:oidc-provider/$oidc_provider"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "$oidc_provider:aud": "sts.amazonaws.com",
          "$oidc_provider:sub": "system:serviceaccount:
$namespace:$service_account"
        }
      }
    }
  ]
}
EOF
```

6. 역할을 생성합니다. *my-role*을 IAM 역할의 이름으로, *my-role-description*를 역할에 대한 간략한 설명으로 바꿉니다.

```
aws iam create-role --role-name my-role --assume-role-policy-document
 file://trust-relationship.json --description "my-role-description"
```

7. 역할에 IAM 정책을 연결합니다. *my-role*을 IAM 역할의 이름으로 바꾸고 *my-policy*를 생성한 기존 정책의 이름으로 바꿉니다.

```
aws iam attach-role-policy --role-name my-role --policy-arn=arn:aws:iam::
$account_id:policy/my-policy
```

8. 서비스 계정이 수임할 IAM 역할의 Amazon 리소스 이름(ARN)을 서비스 계정에 주석으로 답니다. *my-role*을 기존 IAM 역할의 이름으로 바꿉니다. 클러스터가 속한 계정과 다른 AWS 계정의 역할이 이전 단계에서 역할을 수임하도록 허용했다고 가정합니다. 그런 다음 AWS

계정과 다른 계정의 역할을 지정해야 합니다. 자세한 내용은 교차 계정 IAM 권한 단원을 참조하십시오.

```
kubectl annotate serviceaccount -n $namespace $service_account
 eks.amazonaws.com/role-arn=arn:aws:iam::$account_id:role/my-role
```

3. 역할 및 서비스 계정이 올바르게 구성되었는지 확인합니다.

   a. IAM 역할의 신뢰 정책이 올바르게 구성되었는지 확인합니다.

```
aws iam get-role --role-name my-role --query Role.AssumeRolePolicyDocument
```

   예제 출력은 다음과 같습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:default:my-
service-account",
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
                }
            }
        }
    ]
}
```

   b. 이전 단계에서 역할에 연결한 정책이 해당 역할에 연결되었는지 확인합니다.

```
aws iam list-attached-role-policies --role-name my-role --query
 AttachedPolicies[].PolicyArn --output text
```

예제 출력은 다음과 같습니다.

```
arn:aws:iam::111122223333:policy/my-policy
```

c. 사용하려는 정책의 Amazon 리소스 이름(ARN)을 저장할 변수를 설정합니다. *my-policy*를 권한을 확인하려는 정책의 이름으로 바꿉니다.

```
export policy_arn=arn:aws:iam::111122223333:policy/my-policy
```

d. 정책의 기본 버전을 봅니다.

```
aws iam get-policy --policy-arn $policy_arn
```

예제 출력은 다음과 같습니다.

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "EXAMPLEBIOWGLDEXAMPLE",
        "Arn": "arn:aws:iam::111122223333:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        [...]
    }
}
```

e. 정책 내용을 보고 Pod에 필요한 모든 권한이 정책에 포함되어 있는지 확인합니다. 필요한 경우 다음 명령에서 *1*을 이전 출력에서 반환된 버전으로 바꿉니다.

```
aws iam get-policy-version --policy-arn $policy_arn --version-id v1
```

예제 출력은 다음과 같습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::my-pod-secrets-bucket"
```

```
        }
    ]
}
```

이전 단계에서 예제 정책을 생성한 경우 출력은 동일합니다. 다른 정책을 생성한 경우 ## 내용이 다릅니다.

f. Kubernetes 서비스 계정에 역할이 주석으로 달렸는지 확인합니다.

```
kubectl describe serviceaccount my-service-account -n default
```

예제 출력은 다음과 같습니다.

```
Name:                my-service-account
Namespace:           default
Annotations:         eks.amazonaws.com/role-arn:
 arn:aws:iam::111122223333:role/my-role
Image pull secrets:  <none>
Mountable secrets:   my-service-account-token-qqjfl
Tokens:              my-service-account-token-qqjfl
[...]
```

4. (선택 사항) 서비스 계정의 AWS Security Token Service 엔드포인트 구성. AWS는 글로벌 엔드포인트 대신 리전별 AWS STS 엔드포인트를 사용할 것을 권장합니다. 이렇게 하면 지연 시간이 줄고, 중복성이 기본 제공되고, 세션 토큰 유효성이 향상됩니다.

다음 단계

Kubernetes 서비스 계정을 사용하도록 Pods 구성

Kubernetes 서비스 계정을 사용하도록 Pods 구성

Pod가 AWS 서비스에 액세스해야 하는 경우 Kubernetes 서비스 계정을 사용하도록 구성해야 합니다. 서비스 계정은 AWS 서비스에 액세스할 수 있는 권한이 있는 AWS Identity and Access Management(IAM) 역할에 연결되어야 합니다.

필수 조건

- 기존 클러스터가 있어야 합니다. 아직 없는 경우 Amazon EKS 시작하기 가이드 중 하나를 사용하여 생성할 수 있습니다.

- 클러스터에 대한 기존 IAM OpenID Connect(OIDC) 제공업체입니다. 이미 있는지 확인하거나 생성하는 방법을 알아보려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요.
- IAM 역할과 연결된 기존 Kubernetes 서비스 계정. 서비스 계정에 IAM 역할의 Amazon 리소스 이름(ARN)을 주석으로 달아야 합니다. Pods가 AWS 서비스를 사용하는 데 필요한 권한을 포함하는 연결된 IAM 정책이 역할에 있어야 합니다. 서비스 계정 및 역할을 만들고 구성하는 방법에 대한 자세한 내용을 알아보려면 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 섹션을 참조하세요.
- AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.
- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 `kubectl` 설치 또는 업데이트 부분을 참조하세요.
- 클러스터 구성이 포함된 기존 `kubectl config` 파일입니다. `kubectl config` 파일을 생성하려면 Amazon EKS 클러스터용 `kubeconfig` 파일 생성 또는 업데이트 섹션을 참조하세요.

서비스 계정을 사용하도록 Pod 구성

1. 다음 명령을 사용하여 구성을 확인할 Pod를 배포할 수 있는 배포 매니페스트를 생성합니다. *`example values`*를 고유한 값으로 바꿉니다.

```
cat >my-deployment.yaml <<EOF
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
```

```
        app: my-app
    spec:
      serviceAccountName: my-service-account
      containers:
      - name: my-app
        image: public.ecr.aws/nginx/nginx:X.XX
EOF
```

2. 클러스터에 매니페스트를 배포합니다.

```
kubectl apply -f my-deployment.yaml
```

3. Pod에 필요한 환경 변수가 있는지 확인합니다.

   a. 이전 단계에서 배포와 함께 배포된 Pods를 봅니다.

   ```
   kubectl get pods | grep my-app
   ```

   예제 출력은 다음과 같습니다.

   ```
   my-app-6f4dfff6cb-76cv9   1/1     Running   0          3m28s
   ```

   b. Pod가 사용 중인 IAM 역할의 ARN을 봅니다.

   ```
   kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep AWS_ROLE_ARN:
   ```

   예제 출력은 다음과 같습니다.

   ```
   AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/my-role
   ```

   역할 ARN이 기존 서비스 계정에 주석으로 단 역할 ARN과 일치해야 합니다. 서비스 계정에 주석 달기에 대한 자세한 내용을 알아보려면 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 섹션을 참조하세요.

   c. Pod에 웹 ID 토큰 파일 탑재가 있는지 확인합니다.

   ```
   kubectl describe pod my-app-6f4dfff6cb-76cv9 | grep
    AWS_WEB_IDENTITY_TOKEN_FILE:
   ```

   예제 출력은 다음과 같습니다.

```
AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

kubelet은 Pod를 대신하여 토큰을 요청하고 저장합니다. 기본적으로 kubelet은 총 TTL(Time To Live)의 80% 또는 24시간보다 오래된 토큰을 새로 고칩니다. Pod 사양의 설정을 사용하여 기본 서비스 계정을 제외한 모든 계정의 만료 기간을 수정할 수 있습니다. 자세한 내용은 Kubernetes 설명서의 서비스 계정 토큰 볼륨 예측(Service Account Token Volume Projection)을 참조하세요.

클러스터의 Amazon EKS Pod Identity 웹후크는 다음 주석을 이용해 서비스 계정을 사용하는 Pods를 관찰합니다.

```
eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/my-role
```

웹후크는 이전 환경 변수를 해당 Pods에 적용합니다. 클러스터는 환경 변수 및 토큰 파일 탑재를 구성하기 위해 웹후크를 사용할 필요가 없습니다. 이러한 환경 변수를 갖도록 Pods를 수동으로 구성할 수 있습니다. 지원되는 AWS SDK 버전은 자격 증명 체인 제공자에서 이 환경 변수를 먼저 찾습니다. 역할 자격 증명은 이 기준을 충족하는 Pods에 사용됩니다.

4. 역할에 연결된 IAM 정책에서 할당한 권한을 사용하여 Pods가 AWS 서비스와 상호 작용할 수 있는지 확인합니다.

> **Note**
>
> Pod가 서비스 계정과 연결된 IAM 역할의 AWS 자격 증명을 사용하는 경우 해당 AWS CLI의 컨테이너에 있는 Pod 또는 기타 SDK에서는 이 역할이 제공하는 자격 증명을 사용합니다. Amazon EKS 노드 IAM 역할에 제공된 자격 증명에 대한 액세스를 제한하지 않으면 Pod는 계속해서 해당 자격 증명에 액세스할 수 있습니다. 자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 부분을 참조하세요.

Pods가 예상대로 서비스와 상호 작용할 수 없으면 다음 단계를 완료하여 모두 제대로 구성되었는지 확인합니다.

a. Pods에서 OpenID Connect 웹 ID 토큰 파일을 통해 IAM 역할을 수임할 수 있도록 지원하는 AWS SDK 버전을 사용하는지 확인합니다. 자세한 내용은 지원되는 AWS SDK 사용 단원을 참조하십시오.

b. 배포에서 서비스 계정을 사용하고 있는지 확인합니다.

```
kubectl describe deployment my-app | grep "Service Account"
```

예제 출력은 다음과 같습니다.

```
Service Account:  my-service-account
```

c. Pods가 여전히 서비스에 액세스할 수 없는 경우 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성에 설명된 단계를 검토하여 역할 및 서비스 계정이 제대로 구성되었는지 확인합니다.

서비스 계정의 AWS Security Token Service 엔드포인트 구성

서비스 계정에 대한 IAM 역할이 포함된 Kubernetes 서비스 계정을 사용하는 경우 클러스터 및 플랫폼 버전이 동일하거나 다음 표에 나열된 버전보다 이후라면 서비스 계정에서 사용하는 AWS Security Token Service 엔드포인트 유형을 구성할 수 있습니다. Kubernetes 또는 플랫폼 버전이 테이블에 나열된 버전보다 이전 버전인 경우 서비스 계정은 글로벌 엔드포인트만 사용할 수 있습니다.

Kubernetes 버전	플랫폼 버전	기본 엔드포인트 유형
1.29	eks.1	리전
1.28	eks.1	리전
1.27	eks.1	리전
1.26	eks.1	리전
1.25	eks.1	리전
1.24	eks.2	리전
1.23	eks.1	리전

AWS는 글로벌 엔드포인트 대신 리전별 AWS STS 엔드포인트를 사용할 것을 권장합니다. 이렇게 하면 지연 시간이 줄고, 중복성이 기본 제공되고, 세션 토큰 유효성이 향상됩니다. AWS Security Token Service는 Pod가 실행 중인 AWS 리전에서 활성 상태여야 합니다. 또한 AWS 리전에서 서비스 장애가

발생할 경우 다른 AWS 리전에 대한 기본 제공 중복성이 애플리케이션에 있어야 합니다. 자세한 내용은 IAM 사용 설명서의 AWS 리전에서 AWS STS 관리를 참조하세요.

필수 조건

- 기존 클러스터가 있어야 합니다. 아직 없는 경우 Amazon EKS 시작하기 가이드 중 하나를 사용하여 생성할 수 있습니다.
- 클러스터에 대한 기존 IAM OIDC 공급자입니다. 자세한 내용은 클러스터에 대한 IAM OIDC 공급자 생성 단원을 참조하십시오.
- 서비스 계정에 대한 Amazon EKS IAM 기능에서 사용하도록 구성된 기존 Kubernetes 서비스 계정.

Kubernetes 서비스 계정에서 사용하는 엔드포인트 유형 구성

다음 예에서는 모두 Amazon VPC CNI 플러그인에서 사용하는 aws-node Kubernetes 서비스 계정을 사용합니다. *example values*을 고유 서비스 계정, Pods, 네임스페이스, 기타 리소스로 바꿀 수 있습니다.

1. 엔드포인트를 변경하려는 서비스 계정을 사용하는 Pod를 선택합니다. Pod가 실행되는 AWS 리전을 결정합니다. *aws-node-6mfgv*를 Pod 이름으로, *kube-system*을 Pod의 네임스페이스로 바꿉니다.

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep Node:
   ```

   예제 출력은 다음과 같습니다.

   ```
   ip-192-168-79-166.us-west-2/192.168.79.166
   ```

   이전 출력에서 Pod는 us-west-2 AWS 리전의 노드에서 실행 중입니다.
2. Pod's 서비스 계정이 사용 중인 엔드포인트 유형을 결정합니다.

   ```
   kubectl describe pod aws-node-6mfgv -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
   ```

   예제 출력은 다음과 같습니다.

   ```
   AWS_STS_REGIONAL_ENDPOINTS: regional
   ```

현재 엔드포인트가 전역인 경우 `global`은 출력에서 반환됩니다. 출력이 반환되지 않는 경우 기본 엔드포인트 유형이 재정의되지 않고 사용 중에 있습니다.

3. 클러스터 또는 플랫폼 버전이 표에 나열된 버전과 동일하거나 이후인 경우 다음 명령 중 하나를 사용하여 서비스 계정에서 사용하는 엔드포인트 유형을 기본 유형에서 다른 유형으로 변경할 수 있습니다. *aws-node*를 서비스 계정의 이름으로, *kube-system*을 서비스 계정의 네임스페이스로 바꿉니다.

   - 기본 또는 현재 엔드포인트 유형이 전역이고 다음과 같은 리전으로 변경하려는 경우:

     ```
     kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=true
     ```

     서비스 계정에 대한 IAM 역할을 사용하여 Pods의 컨테이너에서 실행되는 애플리케이션에서 미리 서명된 S3 URL을 생성하는 경우 리전 엔드포인트에 대한 URL 형식은 다음 예와 비슷합니다.

     ```
     https://bucket.s3.us-west-2.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
     ```

   - 기본 또는 현재 엔드포인트 유형이 리전이고 다음과 같은 전역으로 변경하려는 경우:

     ```
     kubectl annotate serviceaccount -n kube-system aws-node eks.amazonaws.com/sts-regional-endpoints=false
     ```

     애플리케이션이 AWS STS 글로벌 엔드포인트에 명시적으로 요청하고 Amazon EKS 클러스터에서 리전 엔드포인트를 사용하는 기본 동작을 재정의하지 않는 경우 오류로 인해 요청이 실패합니다. 자세한 내용은 포드 컨테이너는 다음과 같은 오류가 발생합니다. An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region 단원을 참조하십시오.

     서비스 계정에 대한 IAM 역할을 사용하여 Pods의 컨테이너에서 실행되는 애플리케이션에서 미리 서명된 S3 URL을 생성하는 경우 글로벌 엔드포인트에 대한 URL 형식은 다음 예와 비슷합니다.

     ```
     https://bucket.s3.amazonaws.com/path?...&X-Amz-Credential=your-access-key-id/date/us-west-2/s3/aws4_request&...
     ```

사전 서명된 URL을 특정 형식으로 예상하는 자동화가 있거나 사전 서명된 URL을 사용하는 애플리케이션 또는 다운스트림 종속성이 대상 지정된 AWS 리전을 예상하는 경우, 필요한 변경을 수행하여 적절한 AWS STS 엔드포인트를 사용합니다.

4. 서비스 계정에 연결된 기존 Pods를 삭제하고 다시 생성하여 보안 인증 정보 환경 변수를 적용합니다. 변형 웹후크는 이미 실행 중인 Pods에 이 변수를 적용하지 않습니다. *Pods*, *kube-system*, *-l k8s-app=aws-node*를 주석을 설정한 Pods에 대한 정보로 바꿀 수 있습니다.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

5. 모든 Pods가 다시 시작되었는지 확인합니다.

```
kubectl get Pods -n kube-system -l k8s-app=aws-node
```

6. Pods 중 하나에 대한 환경 변수를 봅니다. AWS_STS_REGIONAL_ENDPOINTS 값이 이전 단계에서 설정한 값인지 확인합니다.

```
kubectl describe pod aws-node-kzbtr -n kube-system |grep AWS_STS_REGIONAL_ENDPOINTS
```

예제 출력은 다음과 같습니다.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

교차 계정 IAM 권한

다른 계정의 클러스터에서 ID 제공업체를 생성하거나 연결된 AssumeRole 작업을 사용하여 크로스 계정 IAM 권한을 구성할 수 있습니다. 다음 예제에서 계정 A는 서비스 계정에 대한 IAM 역할을 지원하는 Amazon EKS 클러스터를 소유합니다. 해당 클러스터에서 실행 중인 Pods는 계정 B의 IAM 권한을 수임해야 합니다.

Example 다른 계정의 클러스터에서 ID 제공업체 생성

Example

이 예제에서 계정 A는 계정 B에 클러스터의 OpenID Connect(OIDC) 발행자 URL을 제공합니다. 계정 B는 계정 A의 클러스터에서 OIDC 발행자 URL을 사용하여 클러스터에 대한 IAM OIDC 공급자 생성 및 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성의 지침을 따릅니다. 그런 다음 클러스터 관리자는 계정 B(*444455556666*)의 역할을 사용할 계정 A의 클러스터에 있는 서비스 계정에 주석을 답니다.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    eks.amazonaws.com/role-arn: arn:aws:iam::444455556666:role/account-b-role
```

Example 연결된 **`AssumeRole`** 작업 사용

Example

이 예에서 계정 B는 계정 A의 클러스터에 있는 Pods에 부여할 권한이 포함된 IAM 정책을 생성합니다. 계정 B(*444455556666*)는 계정 A(*111122223333*)에 AssumeRole 권한을 허용하는 신뢰 관계를 사용하여 이 정책을 IAM 역할에 연결합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

계정 A는 클러스터의 OIDC 발행자 주소로 생성된 ID 제공업체의 자격 증명을 가져오는 신뢰 정책을 이용해 역할을 생성합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.region-
code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity"
    }
```

```
  ]
}
```

계정 A는 이 역할을 계정 B가 생성한 역할을 맡을 수 있는 다음과 같은 권한이 포함된 정책에 연결합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::444455556666:role/account-b-role"
        }
    ]
}
```

Pods가 계정 B의 역할을 맡을 수 있게 해주는 애플리케이션 코드에서는 두 가지 프로필, 즉 `account_b_role` 및 `account_a_role`을 사용합니다. `account_b_role` 프로필에서는 `account_a_role` 프로필을 소스로 사용합니다. AWS CLI의 경우 `~/.aws/config` 파일은 다음과 유사합니다.

```
[profile account_b_role]
source_profile = account_a_role
role_arn=arn:aws:iam::444455556666:role/account-b-role

[profile account_a_role]
web_identity_token_file = /var/run/secrets/eks.amazonaws.com/serviceaccount/token
role_arn=arn:aws:iam::111122223333:role/account-a-role
```

연결된 프로파일을 다른 AWS SDK에 지정하려면 사용 중인 SDK에 대한 설명서를 참조하세요. 자세한 내용은 AWS 기반의 도구를 참조하십시오.

지원되는 AWS SDK 사용

서비스 계정에 대한 IAM 역할를 사용할 때 Pods에 있는 컨테이너에서는 OpenID Connect 웹 ID 토큰 파일을 통해 IAM 역할을 수임할 수 있도록 지원하는 AWS SDK 버전을 사용해야 합니다. AWS SDK에 대해 다음과 같은 버전 또는 이후 버전을 사용해야 합니다.

- Java(버전 2) - 2.10.11
- Java - 1.11.704

- Go - 1.23.13
- Python(Boto3) - 1.9.220
- Python(botocore) - 1.12.200
- AWS CLI - 1.16.232
- 노드 – 2.525.0 및 3.27.0
- Ruby - 3.58.0
- C++ - 1.7.174
- .NET – 3.3.659.1 – `AWSSDK.SecurityToken`도 포함해야 합니다.
- PHP - 3.110.7

Cluster Autoscaler,AWS Load Balancer Controller란 무엇인가요? 및 Amazon VPC CNI plugin for Kubernetes 등 널리 사용되는 Kubernetes 추가 기능에서 서비스 계정의 IAM 역할을 지원합니다.

지원되는 SDK를 사용 중인지 확인하려면 컨테이너를 빌드할 때 AWS에서의 구축을 위한 도구에서 선호하는 SDK에 대한 설치 지침을 따르세요.

보안 인증 정보 사용

서비스 계정에 대한 IAM 역할의 보안 인증 정보를 사용하려면 코드에서 임의의 AWS SDK를 사용하여 SDK가 포함된 AWS 서비스에 대한 클라이언트를 만들 수 있고, 기본적으로 SDK는 일련의 위치에서 사용할 AWS Identity and Access Management 보안 인증 정보를 검색합니다. 클라이언트를 생성하거나 SDK를 초기화했을 때 보안 인증 정보 공급자를 지정하지 않으면 서비스 계정에 대한 IAM 역할 보안 인증 정보가 사용됩니다.

이는 서비스 계정에 대한 IAM 역할이 기본 보안 인증 정보 체인의 한 단계로 추가되었기 때문에 작동합니다. 워크로드가 현재 보안 인증 정보 체인의 이전 단계에 있는 보안 인증 정보를 사용하는 경우 동일한 워크로드에 대해 서비스 계정에 대한 IAM 역할을 구성하더라도 해당 보안 인증 정보는 계속 사용됩니다.

SDK는 `AssumeRoleWithWebIdentity` 작업을 사용하여 AWS Security Token Service에서 임시 보안 인증 정보에 대한 서비스 계정 OIDC 토큰을 자동 교환합니다. Amazon EKS와 이 SDK 작업은 만료되기 전에 갱신하여 임시 보안 인증 정보를 계속 교체합니다.

서명 키 가져오기

Kubernetes는 각 Kubernetes Service Account에 `ProjectedServiceAccountToken`을 발급합니다. 이 토큰은 JSON web token (JWT)의 한 유형인 OIDC 토큰입니다. Amazon EKS는 외부 시스템에서 토

큰을 검증할 수 있도록 토큰에 대한 서명 키가 포함된 각 클러스터에 대해 퍼블릭 OIDC 엔드포인트를 호스트합니다.

`ProjectedServiceAccountToken`을 검증하려면 JSON Web Key Set (JWKS)라고도 하는 OIDC 공개 서명 키를 가져와야 합니다. 애플리케이션에서 이러한 키를 사용하여 토큰을 검증합니다. 예를 들어 PyJWT Python 라이브러리를 사용하여 이러한 키로 토큰을 검증할 수 있습니다. `ProjectedServiceAccountToken`에 대한 자세한 내용은 the section called “IAM, Kubernetes 및 OpenID Connect(OIDC) 배경 정보” 섹션을 참조하세요.

필수 조건

- 클러스터에 대한 기존 AWS Identity and Access Management(IAM) OpenID Connect(OIDC) 제공업체입니다. 이미 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 부분을 참조하세요.
- AWS CLI - Amazon EKS를 비롯한 AWS 서비스를 사용한 작업을 위한 명령줄 도구입니다. 자세한 내용은 AWS Command Line Interface 사용 설명서에서 AWS CLI 설치, 업데이트, 제거를 참조하세요. AWS CLI 설치 후, 구성 작업도 수행하는 것이 좋습니다. 자세한 내용은 AWS Command Line Interface 사용 설명서에서 `aws configure`를 사용한 빠른 구성 부분을 참조하세요.

OIDC 공개 서명 키 가져오기(AWS CLI)

1. AWS CLI를 사용하여 Amazon EKS 클러스터의 OIDC URL을 검색합니다.

```
$ aws eks describe-cluster --name my-cluster --query 'cluster.identity.oidc.issuer'
"https://oidc.eks.us-east-1.amazonaws.com/id/8EBDXXXX00BAE"
```

2. curl 또는 유사한 도구를 사용하여 공개 서명 키를 검색합니다. 결과는 JSON Web Key Set (JWKS)입니다.

> **Important**
>
> Amazon EKS는 OIDC 엔드포인트에 대한 호출을 제한합니다. 공개 서명 키를 캐시해야 합니다. 응답에 포함된 `cache-control` 헤더를 준수합니다.

> **Important**
>
> Amazon EKS는 7일마다 OIDC 서명 키를 교체합니다.

```
$ curl https://oidc.eks.us-east-1.amazonaws.com/id/8EBDXXXX00BAE/keys
{"keys":
[{"kty":"RSA","kid":"2284XXXX4a40","use":"sig","alg":"RS256","n":"wklbXXXXMVfQ","e":"AQAB"}
```

Amazon EKS 노드

Kubernetes 노드는 컨테이너화된 애플리케이션을 실행하는 시스템입니다. 각 노드에는 다음과 같은 구성 요소가 있습니다.

- 컨테이너 런타임 – 컨테이너 실행을 담당하는 소프트웨어입니다.
- **`kubelet`** – 컨테이너가 정상 상태이고 연결된 Pod 내에서 실행되고 있는지 확인합니다.
- **`kube-proxy`** – Pods에 대한 통신을 허용하는 네트워크 규칙을 유지 관리합니다.

자세한 내용은 Kubernetes 설명서의 노드를 참조하세요.

Amazon EKS 클러스터는 모든 자체 관리형 노드, Amazon EKS 및 관리형 노드 그룹 및 AWS Fargate의 조합에서 Pods를 예약할 수 있습니다. 클러스터에 배포된 노드에 대한 자세한 내용은 Kubernetes 리소스 보기 부분을 참조하세요.

> **Important**
>
> Amazon EKS가 있는 AWS Fargate은 AWS GovCloud(미국 동부) 및 AWSGovCloud(미국 서부)를 제외한 모든 Amazon EKS 지역에서 사용할 수 있습니다.

> **Note**
>
> 노드는 클러스터를 생성할 때 선택한 서브넷과 동일한 VPC에 있어야 합니다. 하지만 노드가 동일한 서브넷에 있을 필요는 없습니다.

다음 표에는 요구 사항에 가장 적합한 옵션을 결정할 때 평가할 몇 가지 기준이 나와 있습니다. 이 테이블에는 Amazon EKS 외부에서 생성되었으며 볼 수만 있는 연결된 노드가 포함되지 않습니다.

> **Note**
>
> Bottlerocket은 이 표의 일반 정보와 몇 가지 구체적인 차이점이 있습니다. 자세한 내용은 GitHub의 Bottlerocket 설명서를 참조하세요.

기준	EKS 관리형 노드 그룹	자체 관리형 노드	AWS Fargate
AWS Outposts에 배포할 수 있습니다.	아니요	예	아니요
AWS Local Zones에 배포할 수 있습니다.	아니요	예 — 자세한 내용은 Amazon EKS 및 AWS Local Zones 부분을 참조하세요.	아니요
Windows가 필요한 컨테이너를 실행할 수 있습니다.	예	예 — 클러스터에는 최소한 하나 이상의 Linux 노드(가용성을 위해 두 개 권장)가 필요합니다.	아니요
Linux가 필요한 컨테이너를 실행할 수 있습니다.	예	예	예
Inferentia 칩이 필요한 워크로드를 실행할 수 있습니다.	예 — Amazon Linux 노드만 해당	예 — Amazon Linux만 해당	아니요
GPU가 필요한 워크로드를 실행할 수 있습니다.	예 — Amazon Linux 노드만 해당	예 — Amazon Linux만 해당	아니요
Arm 프로세서가 필요한 워크로드를 실행할 수 있습니다.	예	예	아니요
AWS Bottlerocket을 실행할 수 있습니다.	예	예	아니요
포드가 커널 런타임 환경을 다른 Pods와 공유합니다.	예 — 각 노드의 모든 Pods	예 — 각 노드의 모든 Pods	아니요 — 각 Pod에는 전용 커널이 있습니다.

기준	EKS 관리형 노드 그룹	자체 관리형 노드	AWS Fargate
포드가 CPU, 메모리, 스토리지 및 네트워크 리소스를 다른 Pods와 공유합니다.	예 — 각 노드에서 사용되지 않는 리소스가 발생할 수 있습니다.	예 — 각 노드에서 사용되지 않는 리소스가 발생할 수 있습니다.	아니요 — 각 Pod에는 전용 리소스가 있으며 독립적으로 크기를 조정하여 리소스 활용도를 극대화할 수 있습니다.
포드가 Pod 사양에서 요청한 것보다 더 많은 하드웨어 및 메모리를 사용할 수 있습니다.	예 — Pod에 요청된 것보다 많은 리소스가 필요하고 노드에서 리소스를 사용할 수 있는 경우 Pod는 추가 리소스를 사용할 수 있습니다.	예 — Pod에 요청된 것보다 많은 리소스가 필요하고 노드에서 리소스를 사용할 수 있는 경우 Pod는 추가 리소스를 사용할 수 있습니다.	아니요 — 더 큰 vCPU 및 메모리 구성을 사용하여 Pod를 다시 배포할 수 있습니다.
Amazon EC2 인스턴스를 배포하고 관리해야 합니다.	예 — Amazon EKS 최적화 AMI를 배포한 경우 Amazon EKS를 통해 자동화됩니다. 사용자 지정 AMI 배포한 경우 인스턴스를 수동으로 업데이트해야 합니다.	예 - 수동 구성을 통해 또는 Amazon EKS에서 제공되는 AWS CloudFormation 템플릿을 사용하여 Linux(x86), Linux(Arm) 또는 Windows 노드를 배포합니다.	아니요
Amazon EC2 인스턴스의 운영 체제를 보호, 유지 관리 및 패치해야 합니다.	예	예	아니요

기준	EKS 관리형 노드 그룹	자체 관리형 노드	AWS Fargate
노드 배포 시 추가 `kubelet` 인수와 같은 부트스트랩 인수를 제공할 수 있습니다.	예 — 사용자 지정 AMI와 함께 시작 템플릿 `eksctl`사용	예 - 자세한 내용은 GitHub의 부트스트랩 스크립트 사용 정보를 참조하세요.	아니요
노드에 할당된 IP 주소와 다른 CIDR 블록의 Pods에 IP 주소를 할당할 수 있습니다.	예 - 사용자 지정 AMI와 함께 시작 템플릿 사용. 자세한 내용은 사용자 지정 시작 템플릿이 있는 관리형 노드 섹션을 참조하세요.	예 — 자세한 내용은 포드에 대한 사용자 지정 네트워킹 부분을 참조하세요.	아니요
노드에 SSH를 연결할 수 있습니다.	예	예	아니요 - SSH에 사용할 노드 호스트 운영 체제가 없습니다.
사용자 지정 AMI 노드에 배포할 수 있습니다.	예 - 시작 템플릿 사용	예	아니요
사용자 지정 CNI 노드에 배포할 수 있습니다.	예 — 사용자 지정 AMI와 함께 시작 템플릿 사용	예	아니요

기준	EKS 관리형 노드 그룹	자체 관리형 노드	AWS Fargate
노드 AMI를 직접 업데이트해야 합니다.	예 - Amazon EKS 최적화 AMI를 배포한 경우 업데이트가 제공되면 Amazon EKS 콘솔에 알림이 표시됩니다. 콘솔에서 클릭 한 번으로 업데이트를 수행할 수 있습니다. 사용자 정의 AMI를 배포한 경우 업데이트가 제공되면 Amazon EKS 콘솔에 알림이 표시되지 않습니다. 업데이트는 직접 수행해야 합니다.	예 – Amazon EKS 콘솔 이외의 도구 사용. 자체 관리형 노드는 Amazon EKS 콘솔로 관리할 수 없기 때문입니다.	아니요

기준	EKS 관리형 노드 그룹	자체 관리형 노드	AWS Fargate
노드 Kubernetes 버전을 직접 업데이트해야 합니다.	예 - Amazon EKS 최적화 AMI를 배포한 경우 업데이트가 제공되면 Amazon EKS 콘솔에 알림이 표시됩니다. 콘솔에서 클릭 한 번으로 업데이트를 수행할 수 있습니다. 사용자 정의 AMI를 배포한 경우 업데이트가 제공되면 Amazon EKS 콘솔에 알림이 표시되지 않습니다. 업데이트는 직접 수행해야 합니다.	예 – Amazon EKS 콘솔 이외의 도구 사용. 자체 관리형 노드는 Amazon EKS 콘솔로 관리할 수 없기 때문입니다.	아니요 — 노드를 관리하지 않습니다.
Pods에 Amazon EBS 스토리지를 사용할 수 있습니다.	예	예	아니요
Pods에 Amazon EFS 스토리지를 사용할 수 있습니다.	예	예	예
Pods에 Amazon FSx for Lustre 스토리지를 사용할 수 있습니다.	예	예	아니요
서비스에 Network Load Balancer를 사용할 수 있습니다.	예	예	예, Network Load Balancer 생성 사용 시
포드가 퍼블릭 서브넷에서 실행될 수 있습니다.	예	예	아니요

기준	EKS 관리형 노드 그룹	자체 관리형 노드	AWS Fargate
개별 Pods에 서로 다른 VPC 보안 그룹을 할당할 수 있습니다.	예 – Linux 노드 전용	예 – Linux 노드 전용	예
Kubernetes DaemonSets을 실행할 수 있습니다.	예	예	아니요
Pod 매니페스트에서 `HostPort` 및 `HostNetwork` 지원	예	예	아니요
AWS 리전 가용성	모든 Amazon EKS 지원 리전	모든 Amazon EKS 지원 리전	일부 Amazon EKS 지원 리전
Amazon EC2 전용 호스트에서 컨테이너를 실행할 수 있습니다.	예	예	아니요
요금	여러 Pods를 실행하는 Amazon EC2 인스턴스의 비용입니다. 자세한 정보는 Amazon EC2 요금을 참조하세요.	여러 Pods를 실행하는 Amazon EC2 인스턴스의 비용입니다. 자세한 정보는 Amazon EC2 요금을 참조하세요.	개별 Fargate 메모리 및 CPU 구성 비용입니다. 각 Pod에는 자체 비용이 있습니다. 자세한 내용은 AWS Fargate 요금을 참조하세요.

관리형 노드 그룹

Amazon EKS 관리형 노드 그룹은 Amazon EKS Kubernetes 클러스터의 노드(Amazon EC2 인스턴스) 프로비저닝 및 수명 주기 관리를 자동화합니다.

Amazon EKS 관리형 노드 그룹을 사용하면 Kubernetes 애플리케이션을 실행하기 위해 컴퓨팅 용량을 제공하는 Amazon EC2 인스턴스를 별도로 프로비저닝하거나 등록할 필요가 없습니다. 한 번의 조작으로 클러스터에 대한 노드를 자동으로 생성, 업데이트 또는 종료할 수 있습니다. 노드 업데이트 및 종료는 자동으로 노드를 드레이닝하여 애플리케이션을 계속 사용할 수 있도록 합니다.

모든 관리형 노드는 Amazon EKS에서 관리하는 Amazon EC2 Auto Scaling 그룹의 일부로 프로비저닝됩니다. 인스턴스 및 Auto Scaling 그룹을 포함한 모든 리소스는 AWS 계정 내에서 실행됩니다. 각 노드 그룹은 정의한 여러 가용 영역에서 실행됩니다.

Amazon EKS 콘솔, `eksctl`, AWS CLI, AWS API 또는 AWS CloudFormation을 포함한 코드형 인프라를 사용하여 관리형 노드 그룹을 신규 또는 기존 클러스터에 추가할 수 있습니다. 관리형 노드 그룹의 일부로 시작된 노드는 Kubernetes Cluster Autoscaler에서 자동 검색할 수 있도록 태그가 자동으로 지정됩니다. 노드 그룹을 사용하여 Kubernetes 레이블을 노드에 적용하고 언제든지 업데이트할 수 있습니다.

Amazon EKS 관리형 노드 그룹을 사용하기 위한 추가 비용은 없으며 프로비저닝한 AWS 리소스에 대해서만 비용을 지불합니다. 여기에는 Amazon EC2 인스턴스, Amazon EBS 볼륨, Amazon EKS 클러스터 시간 및 기타 AWS 인프라가 포함됩니다. 최소 요금 및 선수금은 없습니다.

새 Amazon EKS 클러스터 및 관리형 노드 그룹을 시작하려면 Amazon EKS 시작하기 - AWS Management Console 및 AWS CLI 부분을 참조하세요.

기존 클러스터에 관리형 노드 그룹을 추가하려면 관리형 노드 그룹 생성을 참조하십시오.

관리형 노드 그룹 개념

- Amazon EKS 관리형 노드 그룹은 사용자를 위해 Amazon EC2 인스턴스를 생성하고 관리합니다.
- 모든 관리형 노드는 Amazon EKS에서 관리하는 Amazon EC2 Auto Scaling 그룹의 일부로 프로비저닝됩니다. 게다가 Amazon EC2 인스턴스 및 Auto Scaling 그룹을 포함한 모든 리소스는 AWS 계정 내에서 실행됩니다.
- 관리형 노드 그룹의 Auto Scaling 그룹은 그룹을 생성할 때 지정하는 모든 서브넷에 걸쳐 있습니다.
- Amazon EKS는 관리형 노드 그룹 리소스를 태깅하여 Kubernetes Cluster Autoscaler를 사용하도록 구성합니다.

  > **Important**
  >
  > Amazon EBS 볼륨에 의해 백업되고 Kubernetes AutoScaling를 사용하는 상태 기반 애플리케이션을 여러 가용 영역에서 실행하는 경우 각 단일 가용 영역으로 범위가 지정된 여러 노드 그룹을 구성해야 합니다. 또한 `--balance-similar-node-groups` 기능을 활성화해야 합니다.

- 관리형 노드를 배포할 때 더 높은 수준의 유연성과 사용자 지정을 위해 사용자 정의 시작 템플릿을 사용할 수 있습니다. 예를 들어 추가`kubelet` 인수를 지정하고 사용자 지정 AMI를 사용할 수 있습니다. 자세한 내용은 사용자 지정 시작 템플릿이 있는 관리형 노드 단원을 참조하십시오. 관리형 노드

그룹을 처음 생성할 때 사용자 정의 시작 템플릿을 사용하지 않으면 자동 생성된 시작 템플릿이 표시됩니다. 이 자동 생성된 템플릿을 수동으로 수정하지 마세요. 수정하면 오류가 발생합니다.

- Amazon EKS는 관리형 노드 그룹의 CVE 및 보안 패치에 대한 공동 책임 모델을 따릅니다. 관리형 노드가 Amazon EKS 최적화 AMI를 실행하는 경우 버그나 문제가 보고될 때 패치 버전의 AMI를 빌드해야 합니다. 수정 사항을 게시할 수 있습니다. 그러나 이렇게 패치된 AMI 버전은 사용자가 관리형 노드 그룹에 배포해야 합니다. 관리형 노드가 사용자 지정 AMI를 실행하는 경우 버그나 문제가 보고되고 AMI를 배포할 때 패치 버전의 AMI를 빌드해야 합니다. 자세한 내용은 관리형 노드 그룹 업데이트 단원을 참조하십시오.
- Amazon EKS 관리형 노드 그룹은 퍼블릭 서브넷과 프라이빗 서브넷 모두에서 시작할 수 있습니다. 2020년 4월 22일 또는 이후에 퍼블릭 서브넷에서 관리형 노드 그룹을 시작하는 경우 인스턴스가 클러스터에 성공적으로 조인하려면 서브넷에서 `MapPublicIpOnLaunch`를 true로 설정해야 합니다. 2020년 3월 26일 이후에 `eksctl` 또는 Amazon EKS 벤딩 AWS CloudFormation 템플릿을 사용하여 퍼블릭 서브넷을 생성한 경우 이 설정이 이미 true로 설정되어 있습니다. 2020년 3월 26일 이전에 퍼블릭 서브넷을 생성한 경우 해당 설정을 수동으로 변경해야 합니다. 자세한 내용을 알아보려면 서브넷의 퍼블릭 IPv4 주소 지정 속성 수정을 참조하세요.
- 프라이빗 서브넷에 관리형 노드 그룹을 배포할 때 컨테이너 이미지를 가져오기 위해 Amazon ECR에 액세스할 수 있는지 확인해야 합니다. NAT 게이트웨이를 서브넷의 라우팅 테이블에 연결하거나 다음 AWS PrivateLink VPC 엔드포인트를 추가하여 이 작업을 수행할 수 있습니다.
  - Amazon ECR API 엔드포인트 인터페이스 - `com.amazonaws.region-code.ecr.api`
  - Amazon ECR Docker 레지스트리 API 엔드포인트 인터페이스 – `com.amazonaws.region-code.ecr.dkr`
  - Amazon S3 게이트웨이 엔드포인트 - `com.amazonaws.region-code.s3`

  일반적으로 사용되는 다른 서비스와 엔드포인트는 프라이빗 클러스터 요구 사항 섹션을 참조하세요.
- 관리형 노드 그룹을 AWS Outposts이나 AWS Wavelength 또는 AWSLocal Zones에 배포할 수 없습니다.
- 단일 클러스터 내에서 여러 관리형 노드 그룹을 생성할 수 있습니다. 예를 들어, 일부 워크로드에는 표준 Amazon EKS 최적화 Amazon Linux AMI를 사용하는 노드 그룹을 생성하고 GPU 지원이 필요한 워크로드에는 GPU 변형을 사용하는 다른 노드 그룹을 생성할 수 있습니다.
- 관리형 노드 그룹에 Amazon EC2 인스턴스 상태 확인 오류가 발생하면 Amazon EKS는 문제 진단에 도움이 되는 오류 메시지를 반환합니다. 자세한 내용은 관리형 노드 그룹 오류 단원을 참조하십시오.
- Amazon EKS는 관리형 노드 그룹 인스턴스에 Kubernetes 레이블을 추가합니다. 이러한 Amazon EKS 제공 레이블에는 `eks.amazonaws.com` 접두사가 붙습니다.
- Amazon EKS는 종료 또는 업데이트 중에 Kubernetes API를 사용하여 노드를 자동으로 비웁니다.

- AZRebalance로 노드를 종료하거나 원하는 노드 수를 줄이는 경우 포드 중단 예산은 반영되지 않습니다. 이러한 작업은 노드에서 Pods 제거를 시도합니다. 그러나 15분 넘게 걸리면 노드의 모든 Pods가 종료되었는지 여부에 관계없이 노드가 종료됩니다. 노드가 종료될 때까지 기간을 연장하려면 Auto Scaling 그룹에 수명 주기 후크를 추가하세요. 자세한 내용을 알아보려면 Amazon EC2 Auto Scaling 사용 설명서의 수명 주기 후크 추가를 참조하세요.
- 스팟 중단 알림 또는 용량 재조정 알림을 수신한 후 드레인 프로세스를 올바르게 실행하려면 CapacityRebalance를 true로 설정해야 합니다.
- 관리 노드 그룹을 업데이트하면 Pods에 대해 설정한 Pod 중단 예산을 준수합니다. 자세한 내용은 관리형 노드 업데이트 동작 단원을 참조하십시오.
- Amazon EKS 관리형 노드 그룹을 사용하는 데 따른 추가 비용은 없습니다. 프로비저닝하는 AWS 리소스에 대해서만 비용을 지불합니다.
- 노드에 대해 Amazon EBS 볼륨을 암호화하려는 경우 시작 템플릿을 사용하여 노드를 배포할 수 있습니다. 시작 템플릿을 사용하지 않고 암호화된 Amazon EBS 볼륨이 있는 관리형 노드를 배포하려면 계정에 생성된 모든 새 Amazon EBS 볼륨을 암호화합니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서에서 기본적으로 암호화를 참조하십시오..

관리형 노드 그룹 용량

관리형 노드 그룹을 생성할 때 온디맨드 또는 스팟 용량 유형을 선택할 수 있습니다. Amazon EKS는 온디맨드 또는 Amazon EC2 스팟 인스턴스만 포함하는 Amazon EC2 Auto Scaling 그룹과 함께 관리형 노드 그룹을 배포합니다. 내결함성 애플리케이션이 스팟 관리 노드 그룹에 대해 Pods를 예약하고, 단일 Kubernetes 클러스터 내의 온디맨드 노드 그룹에 내결함성 애플리케이션을 예약할 수 있습니다. 기본적으로 관리형 노드 그룹은 온디맨드 Amazon EC2 인스턴스를 배포합니다.

온디맨드

온디맨드 인스턴스를 사용하면 장기 약정 없이 초 단위로 컴퓨팅 용량을 구입할 수 있습니다.

작동 방식

기본적으로, 용량 유형을 지정하지 않은 경우 관리형 노드 그룹이 온디맨드 인스턴스를 사용하여 프로비저닝됩니다. 관리형 노드 그룹은 다음 설정을 적용하여 사용자를 대신하여 Amazon EC2 Auto Scaling 그룹을 구성합니다.

- 온디맨드 용량을 프로비저닝하기 위한 할당 전략이 prioritized로 설정됩니다. 관리형 노드 그룹은 API에 전달된 인스턴스 유형의 순서를 사용하여 온디맨드 용량을 채울 때 먼저 사용할 인스턴스 유형을 결정합니다. 예를 들어 세 가지 인스턴스 유형을 c5.large,c4.large, c3.large의 순서

로 지정할 수 있습니다. 온디맨드 인스턴스가 시작되면 관리형 노드 그룹은 `c5.large`, `c4.large`, `c3.large` 순으로 시작하여 온디맨드 용량을 채웁니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon EC2 Auto Scaling 그룹을 참조하세요.

- Amazon EKS는 `eks.amazonaws.com/capacityType: ON_DEMAND` 용량 유형을 지정하는 관리형 노드 그룹의 모든 노드에 다음 Kubernetes 레이블을 추가합니다. 이 레이블을 사용하여 온디맨드 노드에서 상태 저장 또는 내결함성이 없는 애플리케이션을 예약할 수 있습니다.

스팟

Amazon EC2 스팟 인스턴스는 온디맨드 가격에서 큰 폭의 할인을 제공하는 여분의 Amazon EC2 용량입니다. EC2에서 용량을 복구하려면 2분 중단 알림으로 Amazon EC2 스팟 인스턴스를 중단시킬 수 있습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 스팟 인스턴스를 참조하세요. Amazon EC2 스팟 인스턴스로 관리형 노드 그룹을 구성하여 Amazon EKS 클러스터에서 실행되는 컴퓨팅 노드의 비용을 최적화할 수 있습니다.

작동 방식

관리되는 노드 그룹 내에서 스팟 인스턴스를 사용하려면 용량 유형을 `spot`으로 설정하여 관리형 노드 그룹을 생성합니다. 관리형 노드 그룹은 다음 스팟 모범 사례를 적용하여 사용자를 대신하여 Amazon EC2 Auto Scaling 그룹을 구성합니다.

- 스팟 노드가 최적의 스팟 용량 풀에서 프로비저닝되도록 다음 중 하나로 할당 전략을 설정합니다.
  - `price-capacity-optimized`(PCO) - Kubernetes 버전 `1.28` 이상의 클러스터에서 새 노드 그룹을 생성할 때 할당 전략은 `price-capacity-optimized`로 설정됩니다. 하지만 Amazon EKS 관리형 노드 그룹이 PCO 지원을 시작하기 전에 `capacity-optimized`로 이미 생성된 노드 그룹의 할당 전략은 변경되지 않습니다.
  - `capacity-optimized`(CO) - Kubernetes 버전 `1.27` 이하의 클러스터에서 새 노드 그룹을 생성할 때 할당 전략은 `capacity-optimized`로 설정됩니다.

  용량 할당에 사용할 수 있는 스팟 용량 풀 수를 늘리려면 여러 인스턴스 유형을 사용하도록 관리형 노드 그룹을 구성합니다.
- Amazon EC2 스팟 용량 리밸런싱이 활성화되어 있으므로 Amazon EKS가 스팟 노드를 정상적으로 비우고 리밸런싱하여 스팟 노드가 중단 위험이 높을 때 애플리케이션 중단을 최소화할 수 있습니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Amazon EC2 Auto Scaling 용량 재분배를 참조하세요.
  - 스팟 노드가 리밸런싱 권고를 수신하면 Amazon EKS는 자동으로 새로운 대체 스팟 노드를 시작하려고 시도합니다.

  - 교체 스팟 노드가 Ready 상태가 되면 Amazon EKS가 재조정 권장 사항을 받은 스팟 노드를 비우기 시작합니다. Amazon EKS는 최선을 다해 노드를 드레이닝합니다. 따라서 Amazon EKS가 기존 노드를 드레이닝하기 전에 대체 노드가 클러스터에 조인할 때까지 기다린다는 보장이 없습니다.
  - 대체 스팟 노드가 부트스트랩되고 Kubernetes의 Ready 상태에서 Amazon EKS 코드를 실행하고 리밸런싱 권장 사항을 받은 스팟 노드를 비웁니다. 스팟 노드를 코드화하면 서비스 컨트롤러가 이 스팟 노드로 새 요청을 보내지 않습니다. 또한 정상 활성 스팟 노드 목록에서 제거합니다. 스팟 노드를 비우면 실행 중인 Pods가 정상적으로 제거됩니다.
- Amazon EKS는 `eks.amazonaws.com/capacityType: SPOT` 용량 유형을 지정하는 관리형 노드 그룹의 모든 노드에 다음 Kubernetes 레이블을 추가합니다. 이 레이블을 사용하여 스팟 노드에서 내결함성 애플리케이션을 예약할 수 있습니다.

용량 유형 선택 시 고려 사항

온디맨드 또는 스팟 용량을 사용하여 노드 그룹을 배포할지 여부를 결정할 때 다음 조건을 고려해야 합니다.

- 스팟 인스턴스는 유연한 무상태 내결함성 애플리케이션에 적합합니다. 여기에는 배치 및 기계 학습 교육 워크로드, Apache Spark와 같은 빅 데이터 ETL, 대기열 처리 애플리케이션 및 무상태 API 엔드포인트가 포함됩니다. 스팟은 시간이 지남에 따라 변경될 수 있는 예비 Amazon EC2 용량이므로 중단 방지 워크로드에 스팟 용량을 사용하는 것이 좋습니다. 보다 구체적으로 말하면 스팟 용량은 필요한 용량을 사용할 수 없는 기간을 허용할 수 있는 워크로드에 적합합니다.
- 내결함성이 없는 애플리케이션의 경우 온디맨드를 사용하는 것이 좋습니다. 모니터링 및 운영 도구와 같은 클러스터 관리 도구, `StatefulSets`가 필요한 배포, 상태 유지 애플리케이션(예: 데이터베이스)이 여기에 포함됩니다.
- 스팟 인스턴스를 사용하는 동안 애플리케이션의 가용성을 극대화하려면 여러 인스턴스 유형을 사용하도록 스팟 관리 노드 그룹을 구성하는 것이 좋습니다. 여러 인스턴스 유형을 사용할 때는 다음 규칙을 적용하는 것이 좋습니다.
  - 관리형 노드 그룹 내에서 Cluster Autoscaler를 사용하는 경우 vCPU 및 메모리 리소스의 양이 같은 유연한 인스턴스 유형 집합을 사용하는 것이 좋습니다. 이는 클러스터의 노드가 예상대로 확장되도록 하기 위한 것입니다. 예를 들어 4개의 vCPU와 8개의 GiB 메모리가 필요한 경우 `c3.xlarge`, `c4.xlarge`, `c5.xlarge`, `c5d.xlarge`, `c5a.xlarge`, `c5n.xlarge` 또는 기타 유사한 인스턴스 유형을 사용합니다.
  - 애플리케이션 가용성을 높이려면 여러 스팟 관리형 노드 그룹을 배포하는 것이 좋습니다. 이를 위해 각 그룹은 동일한 vCPU 및 메모리 리소스를 가진 유연한 인스턴스 유형 집합을 사용해야 합니다. 예를 들어 4개의 vCPU와 8개의 GiB 메모리가 필요한 경우 `c3.xlarge`, `c4.xlarge`,

`c5.xlarge`, `c5d.xlarge`, `c5a.xlarge`, `c5n.xlarge` 또는 기타 유사한 인스턴스 유형을 사용하여 관리형 노드 그룹을 하나 생성하고 `m3.xlarge`, `m4.xlarge`, `m5.xlarge`, `m5d.xlarge`, `m5a.xlarge`, `m5n.xlarge` 또는 기타 유사한 인스턴스 유형을 사용하여 두 번째 관리형 노드 그룹을 생성하는 것이 좋습니다.

- 사용자 정의 시작 템플릿을 사용하는 스팟 용량 유형으로 노드 그룹을 배포하는 경우 API를 사용하여 여러 인스턴스 유형을 전달합니다. 시작 템플릿을 통해 단일 인스턴스 유형을 전달하지 마세요. 시작 템플릿을 사용하여 노드 그룹을 배포하는 방법에 대한 자세한 내용은 사용자 지정 시작 템플릿이 있는 관리형 노드 부분을 참조하세요.

관리형 노드 그룹 생성

이 주제는 Amazon EKS 클러스터에 등록하는 노드의 Amazon EKS 관리형 노드 그룹을 시작하는 데 도움이 됩니다. 노드가 클러스터에 조인한 이후 Kubernetes 애플리케이션을 배포할 수 있습니다.

Amazon EKS 관리형 노드 그룹을 처음 시작하는 경우, 대신 Amazon EKS 시작하기 지침 중 하나를 따르는 것이 좋습니다. 이 가이드에서는 노드가 있는 Amazon EKS 클러스터를 생성하기 위한 시연을 제공합니다.

> **Important**
>
> - Amazon EKS 노드는 표준 Amazon EC2 인스턴스입니다. 일반 Amazon EC2 가격을 기준으로 요금이 청구됩니다. 자세한 내용은 Amazon EC2 요금을 참조하세요.
> - AWS Outposts, AWS Wavelength 또는 AWS Local Zones이 사용 설정되어 있는 AWS 리전에는 관리형 노드를 생성할 수 없습니다. AWS Outposts, AWS Wavelength, 또는 AWSLocal Zones가 사용 설정되어 있는 AWS 리전에서는 자체 관리형 노드를 생성할 수 있습니다. 자세한 내용은 자체 관리형 Amazon Linux 노드 시작하기,자체 관리형 Windows 노드 시작 및 자체 관리형 Bottlerocket 노드 시작 부분을 참조하세요. Outpost에서 자체 관리형 Amazon Linux 노드 그룹을 생성할 수도 있습니다. 자세한 내용은 Outpost에서 자체 관리형 Amazon Linux 노드 시작하기 단원을 참조하십시오.
> - Amazon EKS 최적화 Linux 또는 Bottlerocket에 포함된`bootstrap.sh` 파일에 AMI ID를 지정하지 않는 경우 관리형 노드 그룹은 최대 값 수를 적용합니다`maxPods`. vCPU가 30개 이하인 인스턴스의 경우 최대 수는 `110` 입니다. vCPU가 30개 이상인 인스턴스의 경우 최대 개수가 `250` 로 바뀝니다. 이 수치는 내부 Amazon EKS Kubernetes확장성 팀 테스트의 확장성 임계값 및 권장 설정을 기반으로 합니다. 자세한 내용은 Amazon VPC CNI 플러그인을 통한 노드당 포드 제한 증가 블로그 게시물을 참조하세요.

필수 조건

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 클러스터 생성 섹션을 참조하세요.
- 노드가 사용할 기존 IAM 역할. 파일을 만들려면 Amazon EKS 노드 IAM 역할 섹션을 참조하세요. 이 역할에 VPC CNI에 대한 정책 중 하나도 없는 경우 VPC CNI 포드에 대해 다음과 같은 별도의 역할이 필요합니다.
- (선택 사항이지만 권장됨) 필요한 IAM 정책이 연결된 자체 IAM 역할로 구성된 Amazon VPC CNI plugin for Kubernetes 추가 기능. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오.
- Amazon EC2 인스턴스 유형 선택에 나열된 고려 사항 숙지. 선택하는 인스턴스 유형에 따라 클러스터 및 VPC에 대한 추가 전제 조건이 있을 수 있습니다.
- Windows관리형 노드 그룹을 추가하려면 먼저 클러스터에 대한Windows 지원을 활성화해야 합니다. 자세한 내용은 Amazon EKS 클러스터에 대해 Windows 지원 사용 설정 단원을 참조하십시오.

`eksctl` 또는 AWS Management Console을 사용하여 관리형 노드 그룹을 생성할 수 있습니다.

eksctl

`eksctl`을 사용하여 관리형 노드 그룹을 생성하려면

이 절차에는 `eksctl` 버전 `0.175.0` 이상이 필요합니다. 버전은 다음 명령을 통해 확인할 수 있습니다.

```
eksctl version
```

`eksctl` 설치 또는 업데이트에 대한 지침은 `eksctl` 설명서에서 Installation을 참조하세요.

1. (선택 사항) AmazonEKS_CNI_Policy 관리형 IAM 정책이 Amazon EKS 노드 IAM 역할에 연결되어 있는 경우 Kubernetes `aws-node` 서비스 계정에 연결하는 IAM 역할에 대신 할당하는 것이 좋습니다. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오.
2. 사용자 정의 시작 템플릿을 사용하거나 사용하지 않고 관리형 노드 그룹을 생성합니다. 시작 템플릿을 수동으로 지정하면 노드 그룹을 더욱 수월하게 사용자 지정할 수 있습니다. 예를 들어, Amazon EKS 최적화 AMI의 `boostrap.sh` 스크립트에 사용자 지정 AMI를 배포하거나 인수를 제공할 수 있습니다. 사용 가능한 모든 옵션 및 기본값의 전체 목록을 보려면 다음 명령을 입력합니다.

```
eksctl create nodegroup --help
```

다음 명령에서 *my-cluster*를 클러스터 이름으로 바꾸고 *my-mng*를 노드 그룹 이름으로 바꿉니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다.

> **Important**
>
> 관리형 노드 그룹을 처음 생성할 때 사용자 정의 시작 템플릿을 사용하지 않는 경우 나중에 노드 그룹에 대해 시작 템플릿을 사용하지 마세요. 사용자 정의 시작 템플릿을 지정하지 않은 경우 시스템에서 시작 템플릿을 자동으로 생성하므로 수동으로 수정하지 않는 것이 좋습니다. 이 자동 생성된 시작 템플릿을 수동으로 수정하면 오류가 발생할 수 있습니다.

시작 템플릿 제외

`eksctl`은 계정에 기본 Amazon EC2 시작 템플릿을 생성하고 사용자가 지정한 옵션에 따라 생성되는 시작 템플릿을 사용하여 노드 그룹을 배포합니다. `--node-type`의 값을 지정하려면 Amazon EC2 인스턴스 유형 선택 부분을 참조하세요.

허용된 키워드로 *ami-family*를 바꿉니다. 자세한 내용은 `eksctl` 설명서의 노드 AMI 패밀리 설정을 참조하세요. *my-key*를 Amazon EC2 키 페어 또는 퍼블릭 키 이름으로 바꿉니다. 이 키는 노드를 시작한 후 SSH로 연결하는 데 사용됩니다.

> **Note**
>
> Windows의 경우 이 명령에서 SSH를 활성화하지 않습니다. 대신 Amazon EC2 키 페어를 인스턴스와 연결하고 인스턴스에 RDP할 수 있습니다.

Amazon EC2 키 페어가 아직 없는 경우 AWS Management Console에서 새로 생성할 수 있습니다. Linux 정보는 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어 및 Linux 인스턴스를 참조하세요. Windows 정보는 Windows 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어 및 Windows 인스턴스를 참조하세요.

다음과 같은 조건에 해당하면 IMDS에 대한 Pod 액세스를 차단하는 것이 좋습니다.

- Pods에 필요한 최소 권한만 있도록 모든 Kubernetes 서비스 계정에 IAM 역할을 할당할 계획입니다.
- 클러스터의 어떤 Pods도 현재 AWS 리전 검색과 같은 다른 이유로 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 액세스할 필요가 없습니다.

자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 부분을 참조하세요.

Pod가 IMDS에 액세스하지 못하게 차단하고 싶다면 **--disable-pod-imds** 옵션을 다음 명령에 추가합니다.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --region region-code \
  --name my-mng \
  --node-ami-family ami-family \
  --node-type m5.large \
  --nodes 3 \
  --nodes-min 2 \
  --nodes-max 4 \
  --ssh-access \
  --ssh-public-key my-key
```

인스턴스는 필요에 따라 Pods에 훨씬 더 많은 수의 IP 주소를 할당하고, 인스턴스와 다른 CIDR 블록의 Pods에 IP 주소를 할당하며, 인터넷 액세스 없이 클러스터에 배포할 수 있습니다. 자세한 내용은 Amazon EC2 노드에 사용 가능한 IP 주소 증량, 포드에 대한 사용자 지정 네트워킹 및 프라이빗 클러스터 요구 사항에서 이전 명령에 추가할 추가 옵션을 참조하세요.

관리형 노드 그룹은 인스턴스 유형에 따라 노드 그룹의 각 노드에서 실행될 수 있는 최대 Pods 수에 대해 단일 값을 계산하고 적용합니다. 다른 인스턴스 유형으로 노드 그룹을 생성하는 경우 모든 인스턴스 유형에 대해 계산된 가장 작은 값이 노드 그룹의 모든 인스턴스 유형에서 실행될 수 있는 최대 Pods 수로 적용됩니다. 관리형 노드 그룹은 각 Amazon EC2 인스턴스 유형의 Amazon EKS 권장 최대 Pods 수에서 참조된 스크립트를 사용하여 값을 계산합니다.

시작 템플릿 사용

시작 템플릿이 이미 존재해야 하며 시작 템플릿 기본 사항에 명시된 요구 사항을 충족해야 합니다.

다음과 같은 조건에 해당하면 IMDS에 대한 Pod 액세스를 차단하는 것이 좋습니다.

- Pods에 필요한 최소 권한만 있도록 모든 Kubernetes 서비스 계정에 IAM 역할을 할당할 계획입니다.
- 클러스터의 어떤 Pods도 현재 AWS 리전 검색과 같은 다른 이유로 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 액세스할 필요가 없습니다.

자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 부분을 참조하세요.

IMDS에 대한 Pod 액세스를 차단하려면 시작 템플릿에서 필요한 설정을 지정합니다.

a. 다음 콘텐츠를 디바이스에 복사합니다. *example values*예제 값을 바꾼 다음 수정된 명령을 실행하여 `eks-nodegroup.yaml` 파일을 생성합니다. 시작 템플릿 없이 배포할 때 지정하는 몇 가지 설정이 시작 템플릿으로 이동됩니다. `version`을 지정하지 않으면 템플릿의 기본 버전이 사용됩니다.

```
cat >eks-nodegroup.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
managedNodeGroups:
- name: my-mng
  launchTemplate:
    id: lt-id
    version: "1"
EOF
```

`eksctl` config 파일 설정의 전체 목록은 `eksctl` 설명서에서 Config 파일 스키마를 참조하세요. 인스턴스는 필요에 따라 Pods에 훨씬 더 많은 수의 IP 주소를 할당하고, 인스턴스와 다른 CIDR 블록의 Pods에 IP 주소를 할당하며, `containerd` 런타임을 사용하고, 아웃바운드 인터넷 액세스 없이 클러스터에 배포할 수 있습니다. 자세한 내용은 Amazon EC2 노드에 사용 가능한 IP 주소 증량, 포드에 대한 사용자 지정 네트워킹, Docker에서 `containerd`로 마이그레이션 테스트 및 프라이빗 클러스터 요구 사항에서 config 파일에 추가할 추가 옵션을 참조하세요.

시작 템플릿에서 AMI ID를 지정하지 않은 경우 관리형 노드 그룹은 인스턴스 유형에 따라 노드 그룹의 각 노드에서 실행될 수 있는 최대 Pods 수에 대해 단일 값을 계산하고

적용합니다. 다른 인스턴스 유형으로 노드 그룹을 생성하는 경우 모든 인스턴스 유형에 대해 계산된 가장 작은 값이 노드 그룹의 모든 인스턴스 유형에서 실행될 수 있는 최대 Pods 수로 적용됩니다. 관리형 노드 그룹은 각 Amazon EC2 인스턴스 유형의 Amazon EKS 권장 최대 Pods 수에서 참조된 스크립트를 사용하여 값을 계산합니다.

작 템플릿에서 AMI ID를 지정한 경우 사용자 지정 네트워킹을 사용하거나 인스턴스에 할당된 IP 주소의 수를 늘리고 싶으면 노드 그룹의 각 노드에서 실행될 수 있는 최대 Pods 수를 지정합니다. 자세한 내용은 각 Amazon EC2 인스턴스 유형의 Amazon EKS 권장 최대 Pods 수 단원을 참조하십시오.

b. 다음 명령을 사용하여 노드 그룹을 배포합니다.

```
eksctl create nodegroup --config-file eks-nodegroup.yaml
```

AWS Management Console

AWS Management Console을 사용하여 관리형 노드 그룹을 생성하려면

1. 클러스터 상태가 ACTIVE가 되기를 기다립니다. 이미 ACTIVE 상태가 아닌 클러스터에 대한 관리형 노드 그룹을 생성할 수 없습니다.
2. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
3. 관리형 노드 그룹을 만들려는 클러스터의 이름을 선택합니다.
4. 컴퓨팅(Compute)을 선택합니다.
5. 노드 그룹 추가(Add Node Group)를 선택합니다.
6. 노드 그룹 구성(Configure node group) 페이지에서 적절히 파라미터를 입력하고 다음(Next)을 선택합니다.

   - 이름 - 관리형 노드 그룹의 고유한 이름을 입력합니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다.
   - 노드 IAM 역할 - 노드 그룹에 사용할 노드 인스턴스 역할을 선택합니다. 자세한 내용은 Amazon EKS 노드 IAM 역할 단원을 참조하십시오.

     ⚠ Important
     - 클러스터를 생성하는 데 사용된 것과 동일한 역할을 사용할 수 없습니다.

- 자체 관리형 노드 그룹에서 현재 사용하지 않는 역할을 사용하는 것이 좋습니다. 그렇지 않으면 새로운 자체 관리형 노드 그룹과 함께 사용할 계획입니다. 자세한 내용은 관리형 노드 그룹 삭제 단원을 참조하십시오.

- 시작 템플릿 사용 - (선택 사항) 기존 시작 템플릿을 사용할지 여부를 선택합니다. 시작 템플릿 이름(Launch Template Name)을 선택합니다. 그런 다음 시작 템플릿 버전(Launch template version)을 선택합니다. 버전을 선택하지 않으면 Amazon EKS가 템플릿의 기본 버전을 사용합니다. 시작 템플릿을 사용하면 사용자 지정 AMI 배포를 허용하고, Pods에 훨씬 더 많은 수의 IP 주소를 할당하고, 인스턴스와 다른 CIDR 블록의 Pods에 IP 주소를 할당하며, 인스턴스에 대해 `containerd` 런타임을 활성화하고, 아웃바운드 인터넷 액세스 없이 클러스터에 노드를 배포하는 등 더욱 다양한 노드 그룹 사용자 지정이 허용됩니다. 자세한 내용은 Amazon EC2 노드에 사용 가능한 IP 주소 증량, 포드에 대한 사용자 지정 네트워킹, Docker에서 `containerd`로 마이그레이션 테스트 및 프라이빗 클러스터 요구 사항 부분을 참조하세요.

  시작 템플릿이 사용자 지정 시작 템플릿이 있는 관리형 노드의 요구 사항을 충족해야 합니다. 자체 시작 템플릿을 사용하지 않는 경우 Amazon EKS API는 계정에 기본 Amazon EC2 시작 템플릿을 생성하고 기본 시작 템플릿을 사용하여 노드 그룹을 배포합니다.

  서비스 계정에 대한 IAM 역할을 구현하고 AWS 서비스에 대한 액세스 권한이 필요한 모든 Pod에 직접 필요한 권한을 할당하며 클러스터의 Pods가 현재 AWS 리전를 검색하는 등 다른 이유로 IMDS에 액세스할 필요가 없는 경우 시작 템플릿에서 호스트 네트워킹을 사용하지 않는 Pods에 대해 IMDS에 대한 액세스를 사용 중지할 수도 있습니다. 자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 부분을 참조하세요.

- Kubernetes 레이블 - (선택 사항) 관리형 노드 그룹의 노드에 Kubernetes 레이블을 적용하도록 선택할 수 있습니다.

- Kubernetes 테인트 - (선택 사항) 관리형 노드 그룹의 노드에 Kubernetes 테인트를 적용하도록 선택할 수 있습니다. 효과(Effect) 메뉴에서 사용할 수 있는 옵션은 **`NoSchedule`**, **`NoExecute`** 및 **`PreferNoSchedule`**입니다. 자세한 내용은 관리형 노드 그룹의 노드 테인트 단원을 참조하십시오.

- 태그 - (선택 사항) Amazon EKS 관리형 노드 그룹에 태그를 지정하도록 선택할 수 있습니다. 이러한 태그는 노드 그룹의 다른 리소스(예: Auto Scaling 그룹이나 인스턴스)에는 전파되지 않습니다. 자세한 내용은 Amazon EKS 리소스 태깅 단원을 참조하십시오.

7. [컴퓨팅 및 크기 조정 구성 설정(Set compute and scaling configuration)] 페이지에서 파라미터를 입력하고 [다음(Next)]을 선택합니다.

- AMI 유형 – AMI 유형을 선택합니다. Arm 인스턴스를 배포하는 경우 배포하기 전에 Amazon EKS 최적화 Arm Amazon Linux AMI의 고려 사항을 검토하세요.

  이전 페이지에서 시작 템플릿을 지정하고 시작 템플릿에 AMI를 지정한 경우 값을 선택할 수 없습니다. 템플릿의 값이 표시됩니다. 템플릿에 지정된 AMI에서 AMI 지정의 요구 사항을 충족해야 합니다.

- 용량 유형 - 용량 유형을 선택합니다. 용량 유형을 선택하는 방법에 대한 자세한 내용은 관리형 노드 그룹 용량 부분을 참조하세요. 동일한 노드 그룹 내에 서로 다른 용량 유형을 혼합할 수 없습니다. 두 용량 유형을 모두 사용하려면 각각 고유한 용량 및 인스턴스 유형을 가진 별도의 노드 그룹을 생성합니다.

- 인스턴스 유형 - 기본적으로 하나 이상의 인스턴스 유형이 지정됩니다. 기본 인스턴스 유형을 제거하려면 인스턴스 유형의 오른쪽에 있는 X를 선택합니다. 관리형 노드 그룹에 사용할 인스턴스 유형을 선택합니다. 자세한 내용은 Amazon EC2 인스턴스 유형 선택 단원을 참조하십시오.

  콘솔에는 일반적으로 사용되는 인스턴스 유형 세트가 표시됩니다. 표시되지 않은 인스턴스 유형을 사용하여 관리형 노드 그룹을 생성해야 하는 경우 `eksctl`, AWS CLI, AWS CloudFormation 또는 SDK를 사용하여 노드 그룹을 생성합니다. 이전 페이지에서 시작 템플릿을 지정한 경우 인스턴스 유형이 시작 템플릿에 지정되어야 하므로 값을 선택할 수 없습니다. 시작 템플릿의 값이 표시됩니다. 용량 유형(Capacity type)에 대한 스팟(Spot)을 선택한 경우 가용성을 높이기 위해 여러 인스턴스 유형을 지정하는 것이 좋습니다.

- 디스크 크기 - 노드 루트 볼륨에 사용할 디스크 크기(GiB)를 입력합니다.

  이전 페이지에서 시작 템플릿을 지정한 경우 시작 템플릿에 값을 지정해야 하므로 값을 선택할 수 없습니다.

- 원하는 크기 - 시작할 때 관리형 노드 그룹에서 유지해야 하는 현재 노드 수를 지정합니다.

  > **Note**
  >
  > Amazon EKS는 노드 그룹을 자동으로 확장 또는 축소하지 않습니다. 그러나 Kubernetes Cluster Autoscaler가 이 작업을 수행하도록 구성할 수 있습니다.

- 최소 크기 - 관리형 노드 그룹이 확장될 수 있는 최소 노드 수를 지정합니다.

- 최대 크기 - 관리형 노드 그룹이 확장될 수 있는 최대 노드 수를 지정합니다.

- 노드 그룹 업데이트 구성 - (선택 사항) 병렬로 업데이트할 노드의 수 또는 백분율을 선택할 수 있습니다. 이러한 노드는 업데이트 중에 사용할 수 없습니다. [최대 사용 불가(Maximum unavailable)]에서 다음 옵션 중 하나를 선택하고 [값(Value)]을 지정합니다.
  - 숫자(Number) - 노드 그룹에서 병렬로 업데이트할 수 있는 노드 수를 선택하고 지정합니다.
  - 비율(Percentage) - 노드 그룹에서 병렬로 업데이트할 수 있는 노드의 비율을 선택하고 지정합니다. 이 기능은 노드 그룹에 많은 수의 노드가 있는 경우에 유용합니다.

8. [네트워킹 지정(Specify networking)] 페이지에서 파라미터를 입력하고 [다음(Next)]을 선택합니다.

- 서브넷 - 관리형 노드를 시작할 서브넷을 선택합니다.

> **Important**
>
> Amazon EBS 볼륨에 의해 백업되고 Kubernetes AutoScaling를 사용하는 상태 기반 애플리케이션을 여러 가용 영역에서 실행하는 경우 각 단일 가용 영역으로 범위가 지정된 여러 노드 그룹을 구성해야 합니다. 또한 `--balance-similar-node-groups` 기능을 활성화해야 합니다.

> **Important**
>
> - 퍼블릭 서브넷을 선택한 경우 클러스터에 퍼블릭 API 서버 엔드포인트만 활성화되어 있으면, 인스턴스의 서브넷에 `MapPublicIPOnLaunch`가 `true`로 설정되어 있어야 클러스터에 성공적으로 조인할 수 있습니다. 2020년 3월 26일 이후에 `eksctl` 또는 Amazon EKS 벤딩 AWS CloudFormation 템플릿을 사용하여 퍼블릭 서브넷을 생성한 경우 이 설정이 이미 `true`로 설정되어 있습니다. 2020년 3월 26일 이전에 `eksctl` 또는 AWS CloudFormation 템플릿으로 서브넷을 생성한 경우 해당 설정을 수동으로 변경해야 합니다. 자세한 내용을 알아보려면 서브넷의 퍼블릭 IPv4 주소 지정 속성 수정을 참조하세요.
> - 시작 템플릿을 사용하고 여러 네트워크 인터페이스를 지정하는 경우 `MapPublicIpOnLaunch`가 `true`로 설정되어 있더라도 Amazon EC2가 퍼블릭 IPv4 주소를 자동으로 할당하지 않습니다. 이 시나리오에서 노드가 클러스터에 조인하려면 클러스터의 프라이빗 API 서버 엔드포인트를 활성화하거나 NAT 게이트웨이와 같은 대체 방법을 통해 제공되는 아웃바운드 인터넷 액세스를 사용하여

프라이빗 서브넷에서 노드를 시작해야 합니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 인스턴스 IP 주소 지정을 참조하세요.

- 노드에 대한 SSH 액세스 구성(Configure SSH access to nodes)(선택 사항). SSH를 활성화하면 문제가 있는 경우 인스턴스에 연결하여 진단 정보를 수집할 수 있습니다. 노드 그룹을 생성할 때 원격 액세스를 사용 설정하는 것이 좋습니다. 노드 그룹을 생성한 후에는 원격 액세스를 사용 설정할 수 없습니다.

  시작 템플릿을 사용하도록 선택한 경우 이 옵션은 표시되지 않습니다. 노드에 대한 원격 액세스를 활성화하려면 시작 템플릿에 키 페어를 지정하고 시작 템플릿에서 지정한 보안 그룹의 노드에 적절한 포트가 열려 있는지 확인합니다. 자세한 내용은 사용자 지정 보안 그룹 사용 단원을 참조하십시오.

  > **Note**
  >
  > Windows의 경우 이 명령에서 SSH를 활성화하지 않습니다. 대신 Amazon EC2 키 페어를 인스턴스와 연결하고 인스턴스에 RDP할 수 있습니다.

- [SSH 키 페어(SSH key pair)]에서 사용할 Amazon EC2 SSH 키를 선택합니다. Linux 정보는 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어 및 Linux 인스턴스를 참조하세요. Windows 정보는 Windows 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어 및 Windows 인스턴스를 참조하세요. 시작 템플릿을 사용하도록 선택한 경우 시작 템플릿을 선택할 수 없습니다. Bottlerocket AMI를 사용하여 노드 그룹에 Amazon EC2 SSH 키가 제공되면 관리 컨테이너도 사용됩니다. 자세한 내용은 GitHub의 관리자 컨테이너를 참조하세요.
- 특정 인스턴스에 대한 액세스를 제한하려면 다음에서 SSH 원격 액세스 허용(Allow SSH remote access from)에서 해당 인스턴스에 연결된 보안 그룹을 선택합니다. 특정 보안 그룹을 선택하지 않는 경우 인터넷의 어느 곳에서나(`0.0.0.0/0`) SSH 액세스가 허용됩니다.

9. 검토 및 생성(Review and create) 페이지에서 관리형 노드 그룹 구성을 검토하고 생성을 선택합니다.

   노드가 클러스터에 조인하지 못한 경우 문제 해결 가이드의 노드가 클러스터 조인에 실패 섹션을 참조하세요.

10. 노드의 상태를 확인하고 `Ready` 상태가 될 때까지 대기합니다.

```
kubectl get nodes --watch
```

11. (GPU 노드만 해당) GPU 인스턴스 유형과 Amazon EKS 최적화 가속 AMI를 선택한 경우 클러스터에 Kubernetes용 NVIDIA 디바이스 플러그인을 DaemonSet(으)로 적용해야 합니다. 다음 명령을 실행하기 전에 *vX.X.X*을(를) 원하는 NVIDIA/k8s-device-plugin 버전으로 교체합니다.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/nvidia-device-plugin.yml
```

정상 작동하는 Amazon EKS 클러스터와 노드가 있으므로, Kubernetes 추가 기능을 설치하고 클러스터에 애플리케이션을 배포하기 시작할 준비가 되었습니다. 다음은 클러스터의 기능을 확장하는 데 도움이 되는 설명서 주제입니다.

- 클러스터를 생성한 IAM 보안 주체는 `kubectl` 또는 AWS Management Console를 사용하여 Kubernetes API 서버를 호출할 수 있는 유일한 보안 주체입니다. 다른 IAM 보안 주체가 클러스터에 액세스할 수 있게 하려면 해당 보안 주체를 추가해야 합니다. 자세한 내용은 Kubernetes API에 대한 액세스 권한 부여 및 필요한 권한 단원을 참조하세요.
- 다음과 같은 조건에 해당하면 IMDS에 대한 Pod 액세스를 차단하는 것이 좋습니다.
  - Pods에 필요한 최소 권한만 있도록 모든 Kubernetes 서비스 계정에 IAM 역할을 할당할 계획입니다.
  - 클러스터의 어떤 Pods도 현재 AWS 리전 검색과 같은 다른 이유로 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 액세스할 필요가 없습니다.

  자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 부분을 참조하세요.
- AutoScaling - 노드 그룹의 노드 수를 자동으로 조정하도록 Kubernetes Cluster Autoscaler를 구성합니다.
- 클러스터에 샘플 애플리케이션을 배포합니다.
- 클러스터 관리 - 클러스터를 관리하는 데 중요한 도구를 사용하는 방법에 대해 알아봅니다.

관리형 노드 그룹 업데이트

관리형 노드 그룹 업데이트를 시작하면 Amazon EKS가 관리형 노드 업데이트 동작에 나열된 단계를 완료하여 자동으로 노드를 업데이트합니다. Amazon EKS 최적화 AMI를 사용하는 경우 Amazon EKS는 최신 AMI 릴리스 버전의 일부로 최신 보안 패치 및 운영 체제 업데이트를 노드에 자동으로 적용합니다.

Amazon EKS 관리형 노드 그룹의 버전 또는 구성을 업데이트하는 것이 유용한 몇 가지 시나리오가 있습니다.

- Amazon EKS 클러스터의 Kubernetes 버전을 업데이트했으며 동일한 Kubernetes 버전을 사용하도록 노드를 업데이트하려고 합니다.
- 관리형 노드 그룹에 새 AMI 릴리스 버전을 사용할 수 있습니다. AMI 버전에 대한 자세한 내용은 다음 섹션을 참조하세요.
  - Amazon EKS 최적화 Amazon Linux AMI 버전
  - Amazon EKS 최적화 Bottlerocket AMI
  - Amazon EKS 최적화 Windows AMI 버전
- 관리형 노드 그룹에 있는 인스턴스의 최소, 최대 또는 원하는 수를 조정하려고 합니다.
- 관리형 노드 그룹의 인스턴스에서 Kubernetes 레이블을 추가하거나 제거하려고 합니다.
- 관리형 노드 그룹에서 AWS 태그를 추가하거나 제거하려고 합니다.
- 업데이트된 사용자 지정 AMI와 같은 구성 변경 사항이 있는 시작 템플릿의 새 버전을 배포해야 합니다.
- Amazon VPC CNI 추가 기능의 버전 `1.9.0` 이상을 배포하고, 접두사 위임에 대해 추가 기능을 사용하도록 설정했으며, 노드 그룹의 새로운 AWS Nitro System 인스턴스가 크게 증가된 Pods 수를 지원하도록 했습니다. 자세한 내용은 Amazon EC2 노드에 사용 가능한 IP 주소 증량 섹션을 참조하세요.
- Windows 노드에 대해 IP 접두사 위임을 활성화했으며 노드 그룹의 새 AWS Nitro System 인스턴스가 크게 늘어난 수의 Pods를 지원하도록 하려고 합니다. 자세한 내용은 Amazon EC2 노드에 사용 가능한 IP 주소 증량 섹션을 참조하세요.

관리형 노드 그룹의 Kubernetes 버전에 대한 최신 AMI 릴리스 버전이 있는 경우 노드 그룹의 버전을 업데이트하여 최신 AMI 버전을 사용할 수 있습니다. 마찬가지로 클러스터에서 노드 그룹보다 최신 버전의 Kubernetes 버전을 실행 중인 경우 최신 AMI 릴리스 버전을 사용하여 클러스터의 Kubernetes 버전과 일치하도록 노드 그룹을 업데이트할 수 있습니다.

크기 조정 작업 또는 업데이트로 인해 관리형 노드 그룹의 노드가 종료되면 해당 노드의 Pods가 먼저 드레이닝됩니다. 자세한 내용은 관리형 노드 업데이트 동작 섹션을 참조하세요.

노드 그룹 버전 업데이트

`eksctl` 또는 AWS Management Console을 사용하여 노드 그룹 버전을 업데이트할 수 있습니다. 업데이트하는 버전은 컨트롤 플레인 버전보다 이후일 수 없습니다.

eksctl

`eksctl`이 있는 노드 그룹 버전을 업데이트하려면

- 다음 명령을 사용하여 노드에 현재 배포된 동일한 Kubernetes 버전의 최신 AMI 릴리스로 관리형 노드 그룹을 업데이트합니다. *`example value`*를 고유한 값으로 바꿉니다.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code
```

> **Note**
>
> 시작 템플릿과 함께 배포된 노드 그룹을 새 시작 템플릿 버전으로 업그레이드하는 경우 이전 명령에 `--launch-template-version` *`version-number`*을 추가합니다. 시작 템플릿이 사용자 지정 시작 템플릿이 있는 관리형 노드에서 설명하는 요구 사항을 충족해야 합니다. 시작 템플릿에 사용자 지정 AMI 가 포함되어 있는 경우 해당 AMI 가 AMI 지정의 요구 사항을 충족해야 합니다. 노드 그룹을 최신 버전의 시작 템플릿으로 업그레이드하면 지정된 시작 템플릿 버전의 새 구성과 일치하도록 모든 노드가 재활용됩니다.
>
> 시작 템플릿 없이 배포된 노드 그룹은 새 시작 템플릿 버전으로 직접 업그레이드할 수 없습니다. 대신 시작 템플릿을 사용하여 새 노드 그룹을 배포하여 노드 그룹을 새 시작 템플릿 버전으로 업데이트해야 합니다.

노드 그룹을 컨트롤 플레인의 Kubernetes 버전과 동일한 버전으로 업그레이드할 수 있습니다. 예를 들어 Kubernetes `1.28`을 실행하는 클러스터가 있는 경우 다음 명령을 사용하여 현재 Kubernetes `1.27`을 실행 중인 작업자를 버전 `1.28`로 업그레이드할 수 있습니다.

```
eksctl upgrade nodegroup \
  --name=node-group-name \
  --cluster=my-cluster \
  --region=region-code \
  --kubernetes-version=1.28
```

AWS Management Console

AWS Management Console이 있는 노드 그룹 버전을 업데이트하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 업데이트할 노드 그룹이 포함된 클러스터를 선택합니다.
3. 하나 이상의 노드 그룹에 사용 가능한 업데이트가 있는 경우 페이지 상단에 사용 가능한 업데이트를 알리는 상자가 나타납니다. Compute(컴퓨팅) 탭을 선택하면 사용 가능한 업데이트가 있는 노드 그룹에 대한 Node groups(노드 그룹) 테이블의 AMI release version(AMI 릴리스 버전) 열에 Update now(지금 업데이트)가 표시됩니다. 노드 그룹을 업데이트하려면 Update now(지금 업데이트)를 선택합니다.

   사용자 지정 AMI와 함께 배포된 노드 그룹에 대한 알림이 표시되지 않습니다. 노드가 사용자 지정 AMI와 함께 배포된 경우 다음 단계를 완료하여 새로운 업데이트된 사용자 지정 AMI를 배포합니다.

   a. AMI의 새 버전을 생성합니다.

   b. 새 AMI ID를 사용하여 새 시작 템플릿 버전을 생성합니다.

   c. 노드를 새 버전의 시작 템플릿으로 업그레이드합니다.
4. Update node group version(노드 그룹 버전 업데이트) 대화 상자에서 다음과 같은 옵션을 활성화하거나 비활성화합니다.

   - Update node group version(노드 그룹 버전 업데이트) - 사용자 지정 AMI를 배포했거나 Amazon EKS에 최적화된 AMI가 현재 클러스터의 최신 버전에 있는 경우에는 이 옵션을 사용할 수 없습니다.
   - Change launch template version(시작 템플릿 버전 변경) - 노드 그룹이 사용자 지정 시작 템플릿 없이 배포된 경우에는 이 옵션을 사용할 수 없습니다. 사용자 지정 시작 템플릿을 사용하여 배포된 노드 그룹의 시작 템플릿 버전만 업데이트할 수 있습니다. 노드 그룹을 업데이트할 Launch template version(시작 템플릿 버전)을 선택합니다. 노드 그룹이 사용자 지정 AMI로 구성된 경우 선택한 버전에서도 AMI를 지정해야 합니다. 최신 버전의 시작 템플릿으로 업그레이드하면 지정된 시작 템플릿 버전의 새 구성과 일치하도록 모든 노드가 재활용됩니다.
5. Update strategy(전략 업데이트)의 경우 다음과 같은 옵션 중 하나를 선택합니다.

   - 롤링 업데이트(Rolling update) - 이 옵션은 클러스터에 대한 Pod 중단 예산을 고려합니다. Pod 중단 예산 문제로 인해 Amazon EKS에서 이 노드 그룹에서 실행 중인 Pods를 적절하게 드레이닝할 수 없는 경우 업데이트에 실패합니다.

- 강제 업데이트(Force update) - 이 옵션은 Pod 중단 예산을 따르지 않습니다. 노드 재시작을 강제로 적용하여 Pod 중단 예산 문제와 관계없이 업데이트가 수행됩니다.

6. 업데이트를 선택합니다.

노드 그룹 구성 편집

관리형 노드 그룹의 일부 구성을 수정할 수 있습니다.

노드 그룹 구성을 편집하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 편집할 노드 그룹이 포함된 클러스터를 선택합니다.
3. 컴퓨팅(Compute) 탭을 선택합니다.
4. 편집할 노드 그룹을 선택한 다음에 Edit(편집)를 선택합니다.
5. (선택 사항) Edit node group(노드 그룹 편집) 페이지에서 다음을 수행합니다.

   a. Node group scaling configuration(노드 그룹 조정 구성)을 편집합니다.

      - 원하는 크기 - 관리형 노드 그룹에서 유지해야 하는 현재 노드 수를 지정합니다.
      - 최소 크기 - 관리형 노드 그룹이 확장될 수 있는 최소 노드 수를 지정합니다.
      - 최대 크기 - 관리형 노드 그룹이 확장될 수 있는 최대 노드 수를 지정합니다. 노드 그룹에서 지원되는 최대 노드 수는 Amazon EKS 서비스 할당량 섹션을 참조하세요.

   b. (선택 사항) Kubernetes 레이블을 노드 그룹의 노드에 추가하거나 제거합니다. 여기에 표시된 레이블은 Amazon EKS에 적용한 레이블일 뿐입니다. 여기에 표시되지 않는 노드에 다른 레이블이 존재할 수 있습니다.

   c. (선택 사항) Kubernetes테인트를 노드 그룹의 노드에 추가하거나 제거합니다. 추가된 테인트는 **`NoSchedule`**,**`NoExecute`** 또는 **`PreferNoSchedule`** 중 하나의 효과를 가질 수 있습니다. 자세한 내용은 관리형 노드 그룹의 노드 테인트 섹션을 참조하세요.

   d. (선택 사항) 태그를 추가하거나 노드 그룹 리소스에서 제거합니다. 이러한 태그는 Amazon EKS 노드 그룹에만 적용됩니다. 노드 그룹의 서브넷 또는 Amazon EC2 인스턴스 등의 다른 리소스로 전파되지 않습니다.

   e. (선택 사항) 노드 그룹 업데이트 구성(Node Group update configuration)을 편집합니다. [숫자(Number)] 또는 [비율(Percentage)]을 선택합니다.

      - 숫자(Number) - 노드 그룹에서 병렬로 업데이트할 수 있는 노드 수를 선택하고 지정합니다. 이러한 노드는 업데이트 중에 사용할 수 없습니다.

- 비율(Percentage) - 노드 그룹에서 병렬로 업데이트할 수 있는 노드의 비율을 선택하고 지정합니다. 이러한 노드는 업데이트 중에 사용할 수 없습니다. 이 기능은 노드 그룹에 많은 노드가 있는 경우에 유용합니다.

f. 편집을 마쳤으면 [변경 사항 저장(Save changes)]을 선택합니다.

관리형 노드 업데이트 동작

Amazon EKS 관리형 작업자 노드 업그레이드 전략에는 다음 섹션에서 설명하는 4가지 단계가 있습니다.

설정 단계

설정 단계는 다음과 같습니다.

1. 노드 그룹과 연결된 Auto Scaling 그룹의 새 Amazon EC2 시작 템플릿 버전을 생성합니다. 새 시작 템플릿 버전은 업데이트를 위해 대상 AMI 또는 사용자 지정 시작 템플릿 버전을 사용합니다.
2. 최신 시작 템플릿 버전을 사용하도록 Auto Scaling 그룹을 업데이트합니다.
3. 노드 그룹의 `updateConfig` 속성을 사용하여 병렬로 업그레이드할 최대 노드 수를 결정합니다. 사용할 수 없는 최대 노드의 할당량은 100개입니다. 기본값은 노드 1개입니다. 자세한 내용은 Amazon EKS API 참조의 `updateConfig` 속성을 참조하세요.

확장 단계

관리형 노드 그룹의 노드를 업그레이드할 때 업그레이드된 노드는 업그레이드 중인 노드와 동일한 가용 영역에서 시작됩니다. 이 배치를 보장하기 위해 Amazon EC2의 가용 영역 재분배를 사용합니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 가용 영역 재분배를 참조하세요. 이 요구 사항이 충족되도록 관리형 노드 그룹의 가용 영역당 최대 2개의 인스턴스를 시작할 수 있습니다.

확장 단계는 다음과 같습니다.

1. Auto Scaling 그룹의 최대 크기와 원하는 크기를 다음 중 더 큰 값으로 늘립니다.
   - Auto Scaling 그룹이 배포된 가용 영역 수의 최대 2배
   - 업그레이드할 수 없는 최대값입니다.

     예를 들어 노드 그룹에 5개의 가용 영역이 있고 `maxUnavailable`이 하나인 경우 업그레이드 프로세스에서 최대 10개의 노드를 시작할 수 있습니다. 하지만 `maxUnavailable`이 20이거나 10보다 크면 프로세스에서 20개의 새 노드를 시작합니다.

2. Auto Scaling 그룹 규모를 조정한 후 노드 그룹에 최신 구성을 사용하는 노드가 있는지 확인합니다. 이 단계는 다음 기준을 충족하는 경우에만 성공합니다.
   - 노드가 있는 모든 가용 영역에서 하나 이상의 새 노드가 시작됩니다.
   - 모든 새 노드는 Ready 상태여야 합니다.
   - 새 노드에 Amazon EKS 적용 레이블이 있어야 합니다.

     다음은 일반 노드 그룹에 있는 작업자 노드의 Amazon EKS 적용 레이블입니다.

     - `eks.amazonaws.com/nodegroup-image=`*`$amiName`*
     - `eks.amazonaws.com/nodegroup=`*`$nodeGroupName`*

     다음은 사용자 정의 시작 템플릿 또는 AMI 노드 그룹에 있는 작업자 노드의 Amazon EKS 적용 레이블입니다.

     - `eks.amazonaws.com/nodegroup-image=`*`$amiName`*
     - `eks.amazonaws.com/nodegroup=`*`$nodeGroupName`*
     - `eks.amazonaws.com/sourceLaunchTemplateId=`*`$launchTemplateId`*
     - `eks.amazonaws.com/sourceLaunchTemplateVersion=`*`$launchTemplateVersion`*
3. 새 Pods의 스케줄링을 피하기 위해 노드를 스케줄 불가능으로 표시합니다. 또한 노드를 종료하기 전에 로드 밸런서에서 노드를 제거하도록 `node.kubernetes.io/exclude-from-external-load-balancers=true`로 노드에 레이블을 지정합니다.

이 단계에서 `NodeCreationFailure` 오류가 발생하는 알려진 이유는 다음과 같습니다.

가용 영역의 용량 부족

요청된 인스턴스 유형의 용량이 가용 영역에 없을 가능성이 있습니다. 관리형 노드 그룹을 생성하는 동안 여러 인스턴스 유형을 구성하는 것이 좋습니다.

계정의 EC2 인스턴스 제한

Service Quotas를 사용하여 계정이 동시에 실행할 수 있는 Amazon EC2 인스턴스 수를 늘려야 할 수 있습니다. 자세한 내용은 Linux 인스턴스용 Amazon Elastic Compute Cloud 사용 설명서의 EC2 Service Quotas를 참조하세요.

사용자 정의 사용자 데이터

사용자 정의 사용자 데이터로 인해 부트스트랩 프로세스가 중단될 수 있습니다. 이 시나리오에서는 노드에서 `kubelet`이 시작되지 않거나 예상되는 Amazon EKS 레이블을 가져오지 못할 수 있습니다. 자세한 내용은 AMI 지정 단원을 참조하십시오.

노드를 비정상적이거나 준비되지 않은 상태로 만드는 모든 변경 사항

노드 디스크 압력, 메모리 압력 및 이와 유사한 조건으로 인해 노드가 Ready 상태가 되지 않을 수 있습니다.

업그레이드 단계

업그레이드 단계는 다음과 같습니다.

1. 노드 그룹에 대해 구성된 사용 불가능한 최대값까지 업그레이드해야 하는 노드를 무작위로 선택합니다.
2. 노드에서 Pods를 드레이닝합니다. Pods가 15분 이내에 노드를 떠나지 않고 force 플래그가 없으면 PodEvictionFailure 오류와 함께 업그레이드 단계가 실패합니다. 이 시나리오에서는 update-nodegroup-version 요청과 함께 force 플래그를 적용하여 Pods를 삭제할 수 있습니다.
3. 모든 Pod가 제거된 후 노드를 차단하고 60초 동안 기다립니다. 이 작업은 서비스 컨트롤러가 이 노드에 새 요청을 보내지 않고 활성 노드 목록에서 이 노드를 제거하도록 하기 위해 수행됩니다.
4. 코드화된 노드에 대한 Auto Scaling 그룹에 종료 요청을 보냅니다.
5. 이전 버전의 시작 템플릿으로 배포된 노드 그룹에 노드가 없을 때까지 이전 업그레이드 단계를 반복합니다.

이 단계에서 PodEvictionFailure 오류가 발생하는 알려진 이유는 다음과 같습니다.

적극적인 PDB

적극적인 PDB가 Pod에 정의되어 있거나 동일한 Pod를 가리키는 여러 PDB가 있습니다.

모든 테인트를 허용하는 배포

모든 Pod가 제거되면 이전 단계에서 노드가 테인트되었기 때문에 노드가 비어 있어야 합니다. 그러나 배포가 모든 테인트를 허용하는 경우 노드가 비어 있지 않을 가능성이 높아져 Pod 제거 실패로 이어집니다.

축소 단계

축소 단계는 Auto Scaling 그룹의 최대 크기와 원하는 크기를 하나씩 줄여 업데이트가 시작되기 전의 값으로 돌아갑니다.

업그레이드 워크플로에서 Cluster Autoscaler가 워크플로의 축소 단계에서 노드 그룹을 확장하고 있다고 판단하면 노드 그룹을 원래 크기로 되돌리지 않고 즉시 종료됩니다.

관리형 노드 그룹의 노드 테인트

Amazon EKS는 관리형 노드 그룹을 통해 Kubernetes 테인트 구성을 지원합니다. 테인트 및 허용 오차는 함께 작동하여 Pods가 부적절한 노드에 예약되지 않도록 합니다. 하나 이상의 테인트를 노드에 적용할 수 있습니다. 이것은 노드가 테인트를 용납하지 않는 Pods를 허용해서는 안 된다는 것을 나타냅니다. 허용 오차는 Pods에 적용되며 일치하는 테인트가 있는 노드에 Pods를 예약할 수는 있지만 필요하지 않습니다. 테인트와 허용 오차에 대한 자세한 내용을 알아보려면 Kubernetes설명서를 참조하세요.

AWS Management Console을 사용하거나 Amazon EKS API를 통해 Kubernetes 노드 테인트를 신규 및 기존 관리형 노드 그룹에 적용할 수 있습니다.

- AWS Management Console를 사용하여 테인트가 있는 노드 그룹을 생성하는 방법에 대한 자세한 내용은 관리형 노드 그룹 생성를 참조하십시오.
- 다음은 AWS CLI를 사용하여 테인트가 있는 노드 그룹을 생성하는 예제입니다.

```
aws eks create-nodegroup \
 --cli-input-json '
{
  "clusterName": "my-cluster",
  "nodegroupName": "node-taints-example",
  "subnets": [
     "subnet-1234567890abcdef0",
     "subnet-abcdef01234567890",
     "subnet-021345abcdef67890"
   ],
  "nodeRole": "arn:aws:iam::111122223333:role/AmazonEKSNodeRole",
  "taints": [
     {
         "key": "dedicated",
         "value": "gpuGroup",
         "effect": "NO_SCHEDULE"
     }
   ]
}'
```

테인트 의 자세한 내용 및 사용 예제는 Kubernetes참조 설명서를 참조하세요.

> **Note**
>
> - `UpdateNodegroupConfig` API를 사용하여 노드 그룹을 생성한 후 테인트를 업데이트할 수 있습니다.
> - 테인트 키는 문자 또는 숫자로 시작해야 합니다. 문자, 숫자, 하이픈(`-`), 마침표(`.`) 또는 밑줄(`_`)을 포함할 수 있습니다. 최대 63자까지 가능합니다.
> - 경우에 따라, 테인트 키는 DNS 하위 도메인 접두사와 단일 `/`로 시작할 수 있습니다. DNS 하위 도메인 접두사로 시작하는 경우 253자가 될 수 있습니다.
> - 이 값은 선택 사항이며 문자 또는 숫자로 시작해야 합니다. 문자, 숫자, 하이픈(`-`), 마침표(`.`) 또는 밑줄(`_`)을 포함할 수 있습니다. 최대 63자까지 가능합니다.
> - Kubernetes을 직접 사용하거나 AWS Management Console를 사용하는 경우 오염 효과는**`NoSchedule`**, **`PreferNoSchedule`**, 또는 **`NoExecute`**이어야 합니다. AWS CLI나 API를 직접 사용하는 경우 오염 효과는**`NO_SCHEDULEPREFER_NO_SCHEDULE`**, 또는 **`NO_EXECUTE`** 이어야 합니다.
> - 한 노드 그룹에 대해 최대 50개의 테인트가 허용됩니다.
> - 관리형 노드 그룹을 사용하여 생성한 테인트를 노드에서 수동으로 제거하는 경우 Amazon EKS는 해당 테인트를 노드에 다시 추가하지 않습니다. 이는 관리형 대상 노드 그룹 구성에 테인트가 지정된 경우에도 마찬가지입니다.

`aws eks update-nodegroup-config` AWS CLI 명령을 사용하여 관리형 노드 그룹의 테인트를 추가, 제거 또는 교체할 수 있습니다.

사용자 지정 시작 템플릿이 있는 관리형 노드

최고 수준의 사용자 지정을 위해 자체 시작 템플릿과 사용자 지정 AMI를 사용하여 관리형 노드를 배포할 수 있습니다. 시작 템플릿을 사용하면 다음과 같은 기능을 사용할 수 있습니다.

- 노드 배포 시 추가 `kubelet` 인수와 같은 부트스트랩 인수를 제공할 수 있습니다.
- 노드에 할당된 IP 주소와 다른 CIDR 블록의 Pods에 IP 주소를 할당할 수 있습니다.
- 사용자 지정 AMI 노드에 배포할 수 있습니다.
- 사용자 지정 CNI 노드에 배포할 수 있습니다.

관리형 노드 그룹을 처음 만들 때 자체 시작 템플릿을 제공하면 나중에 더 유연하게 사용할 수 있습니다. 자체 시작 템플릿을 사용하여 관리형 노드 그룹을 배포한 후 동일한 시작 템플릿의 다른 버전으로

업데이트할 수 있습니다. 노드 그룹을 다른 버전의 시작 템플릿으로 업데이트하면 그룹의 모든 노드가 지정된 시작 템플릿 버전의 새 구성과 일치하도록 재활용됩니다.

관리형 노드 그룹은 항상 Amazon EC2 Auto Scaling 그룹에서 사용할 시작 템플릿과 함께 배포됩니다. Amazon EKS API는 사용자가 제공한 템플릿을 복사하거나 계정의 기본값으로 자동 생성하여 이 시작 템플릿을 생성합니다. 자동 생성된 시작 템플릿을 수정하지 않는 것이 좋습니다. 사용자 정의 시작 템플릿을 사용하지 않는 기존 노드 그룹은 직접 업데이트할 수 없습니다. 대신 사용자 정의 시작 템플릿을 사용하여 새 노드 그룹을 생성해야 합니다.

시작 템플릿 기본 사항

AWS Management Console, AWS CLI 또는 AWS SDK를 사용하여 Amazon EC2 Auto Scaling 시작 템플릿을 생성할 수 있습니다. 자세한 내용을 알아보려면 Amazon EC2 Auto Scaling 사용 설명서의 Auto Scaling 그룹을 위한 시작 템플릿 생성을 참조하세요. 시작 템플릿의 일부 설정은 관리형 노드 구성에 사용되는 설정과 유사합니다. 시작 템플릿을 사용하여 노드 그룹을 배포하거나 업데이트할 때 노드 그룹 구성 또는 시작 템플릿에 일부 설정을 지정해야 합니다. 두 위치 모두에 설정을 지정하지 않습니다. 잘못된 위치에 설정이 있으면 노드 그룹 생성 또는 업데이트와 같은 작업이 실패합니다.

다음 표에는 시작 템플릿에서 금지된 설정이 나열되어 있습니다. 사용 가능한 경우 관리형 노드 그룹 구성에 필요한 유사한 설정도 나열되어 있습니다. 나열된 설정은 콘솔에 표시되는 설정입니다. AWS CLI 및 SDK에서 비슷하지만 다른 이름을 가질 수 있습니다.

시작 템플릿 - 금지됨	Amazon EKS 노드 그룹 구성
[네트워크 인터페이스(Network interfaces)]의 [서브넷(Subnet)]([네트워크 인터페이스 추가(Add network interface)])	네트워킹 지정(Specify networking) 페이지의 노드 그룹 네트워크 구성(Node Group network configuration)에 있는 서브넷(Subnets)
[고급 세부 정보(Advanced details)]의 [IAM 인스턴스 프로필(IAM instance profile)]	노드 그룹 구성(Configure Node Group) 페이지의 노드 그룹 구성(Node Group configuration)에 있는 노드 IAM 역할(Node IAM Role)
[고급 세부 정보(Advanced details)]의 [종료 동작(Shutdown behavior)] 및 [중지 - 최대 절전 모드 동작(Stop - Hibernate behavior)]. 두 설정 모두에 대해 기본값인 시작 템플릿에 [시작 템플릿 설정에 포함 안 함(Don't include in launch template setting)]을 그대로 둡니다.	동일한 사항 없음 Amazon EKS는 Auto Scaling 그룹이 아니라 인스턴스 수명 주기를 제어해야 합니다.

다음 표에는 관리형 노드 그룹 구성에서 금지된 설정이 나열되어 있습니다. 사용 가능한 경우 시작 템플릿에 필요한 유사한 설정도 나열되어 있습니다. 나열된 설정은 콘솔에 표시되는 설정입니다. AWS CLI 및 SDK에서 비슷한 이름을 가질 수 있습니다.

Amazon EKS 노드 그룹 구성 — 금지됨	시작 템플릿

(시작 템플릿에서 사용자 정의 AMI를 지정한 경우에만 해당) 컴퓨팅 및 크기 조정 구성 설정(Set compute and scaling configuration) 페이지의 노드 그룹 컴퓨팅 구성(Node Group compute configuration)에 있는 AMI 유형(AMI type) - 콘솔에 시작 템플릿에서 지정됨(Specified in launch template)이라는 메시지와 지정된 AMI ID가 표시됩니다.
 	
Application and OS Images (Amazon Machine Image)(애플리케이션 및 OS 이미지(Amazon Machine Image))가 시작 템플릿에 지정되지 않은 경우 노드 그룹 구성에서 AMI를 선택할 수 있습니다.
	Launch template contents(시작 템플릿 내용)의 Application and OS Images (Amazon Machine Image)(애플리케이션 및 OS 이미지(Amazon Machine Image)) - 다음 요구 사항 중 하나가 있는 경우 ID를 지정해야 합니다.
	
	• 사용자 지정 AMI 사용 AMI 지정에 나열된 요구 사항을 충족하지 않는 AMI 지정하면 노드 그룹 배포가 실패합니다.
	• Amazon EKS 최적화 AMI에 포함된 `bootstrap.sh` 파일에 인수를 제공하기 위해 사용자 데이터를 제공하려고 함 인스턴스가 Pods에 훨씬 더 많은 수의 IP 주소를 할당하도록 하거나, 인스턴스와 다른 CIDR 블록의 Pods에 IP 주소를 할당하거나, 아웃바운드 인터넷 액세스 없이 프라이빗 클러스터를 배포하도록 할 수 있습니다. 자세한 정보는 다음 주제를 참조하십시오.
	  • Amazon EC2 노드에 사용 가능한 IP 주소 증량
	  • 포드에 대한 사용자 지정 네트워킹
	  • 프라이빗 클러스터 요구 사항
	  • AMI 지정

컴퓨팅 및 크기 조정 구성 설정(Set compute and scaling configuration) 페이지의 노드 그룹 컴퓨팅 구성(Node Group compute configuration)에 있는 디스크 크기(Disk size) - 콘솔에 시작 템플릿에 지정됨(Specified in launch template)이라는 메시지가 표시됩니다.	[스토리지(볼륨)(Storage (Volumes))]의 [크기(Size)]([새 볼륨 추가(Add new volume)]). 시작 템플릿에서 이 옵션을 지정해야 합니다.

Amazon EKS 노드 그룹 구성 — 금지됨	시작 템플릿
네트워킹 지정(Specify Networking) 페이지의 노드 그룹 구성(Node Group configuration)에 있는 SSH 키 페어(SSH key pair) - 콘솔에 시작 템플릿에 지정되어 있는 키가 표시되거나 시작 템플릿에 지정되지 않음(Not specified in launch template)이라는 메시지가 표시됩니다.	[키 페어(로그인)(Key pair (login))]의 [키 페어 이름(Key pair name)].
시작 템플릿을 사용할 때 원격 액세스가 허용되는 소스 보안 그룹을 지정할 수 없습니다.	인스턴스의 [네트워크 설정(Network settings)]에 있는]보안 그룹(Security groups)] 또는 [네트워크 인터페이스(Network interfaces)]의 [보안 그룹(Security groups)]([네트워크 인터페이스 추가(Add network interface)]), 두 가지 모두 설정할 수는 없음. 자세한 내용은 사용자 지정 보안 그룹 사용 단원을 참조하십시오.

Note

- 시작 템플릿을 사용하여 노드 그룹을 배포하는 경우 시작 템플릿의 시작 템플릿 콘텐츠(Launch template contents)에서 인스턴스 유형(Instance type)을 0개 또는 1개 지정합니다. 또는 콘솔의 컴퓨팅 및 크기 조정 구성 설정(Set compute and scaling configuration) 페이지에서 인스턴스 유형(Instance types)에 대해 0~20개의 인스턴스 유형을 지정할 수 있습니다. 또는 Amazon EKS API를 사용하는 다른 도구로 인스턴스 유형을 지정할 수 있습니다. 시작 템플릿에서 인스턴스 유형을 지정하고 해당 시작 템플릿을 사용하여 노드 그룹을 배포하는 경우 콘솔에서 인스턴스 유형을 지정하거나 Amazon EKS API를 사용하는 다른 도구를 사용하여 인스턴스 유형을 지정할 수 없습니다. 시작 템플릿, 콘솔 또는 Amazon EKS API를 사용하는 다른 도구를 사용하여 인스턴스 유형을 지정하지 않으면 `t3.medium` 인스턴스 유형이 사용됩니다. 노드 그룹에서 스팟 용량 유형을 사용하는 경우 콘솔을 사용하여 여러 인스턴스 유형을 지정하는 것이 좋습니다. 자세한 내용은 관리형 노드 그룹 용량 단원을 참조하십시오.
- 노드 그룹에 배포하는 컨테이너가 인스턴스 메타데이터 서비스 버전 2를 사용하는 경우 시작 템플릿에 메타데이터 응답 홉 제한(Metadata response hop limit)이 2로 설정되어 있어야 합니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스 메타데이터

및 사용자 데이터를 참조하세요. 사용자 지정 시작 템플릿을 사용하지 않고 관리형 노드 그룹을 배포하는 경우 이 값은 기본 시작 템플릿의 노드 그룹에 대해 자동으로 설정됩니다.

Amazon EC2 인스턴스 태깅

시작 템플릿의 `TagSpecification` 파라미터를 사용하여 노드 그룹의 Amazon EC2 인스턴스에 적용할 태그를 지정할 수 있습니다. `CreateNodegroup` 또는 `UpdateNodegroupVersion`API를 호출하는 IAM 엔터티에는 `ec2:RunInstances` 및 `ec2:CreateTags`에 대한 권한이 있어야 하며, 시작 템플릿에 태그가 추가되어 있어야 합니다.

사용자 지정 보안 그룹 사용

시작 템플릿을 사용하여 사용자 지정 Amazon EC2 보안 그룹 지정하고 노드 그룹의 인스턴스에 적용할 수 있습니다. 이는 인스턴스 수준 보안 그룹 파라라미터 또는 네트워크 인터페이스 구성 파라미터의 일부로 사용할 수 있습니다. 하지만 인스턴스 수준 및 네트워크 인터페이스 보안 그룹을 모두 지정하는 시작 템플릿을 생성할 수 없습니다. 관리형 노드 그룹과 함께 사용자 지정 보안 그룹을 사용할 때 적용되는 다음 조건을 고려하세요.

- Amazon EKS는 단일 네트워크 인터페이스 사양이 있는 시작 템플릿만 허용합니다.
- 기본적으로 Amazon EKS는 클러스터 보안 그룹을 노드 그룹의 인스턴스에 적용하여 노드와 제어 플레인 간의 통신을 용이하게 합니다. 앞에서 언급한 옵션 중 하나를 사용하여 시작 템플릿에 사용자 지정 보안 그룹을 지정하는 경우 Amazon EKS는 클러스터 보안 그룹을 추가하지 않습니다. 따라서 보안 그룹의 인바운드 및 아웃바운드 규칙이 클러스터 엔드포인트와의 통신을 사용 설정하는지 확인해야 합니다. 보안 그룹 규칙이 올바르지 않으면 작업자 노드가 클러스터에 참여할 수 없습니다. 보안 그룹 규칙에 대한 자세한 내용은 Amazon EKS 보안 그룹 요구 사항 및 고려 사항 부분을 참조하세요.
- 노드 그룹의 인스턴스에 대한 SSH 액세스가 필요한 경우 해당 액세스를 허용하는 보안 그룹을 포함합니다.

Amazon EC2 사용자 데이터

시작 템플릿에는 사용자 정의 사용자 데이터에 대한 부분이 포함되어 있습니다. 개별 사용자 정의 AMI를 수동으로 생성하지 않고 이 부분에서 노드 그룹에 대한 구성 설정을 지정할 수 있습니다. Bottlerocket에 사용할 수 있는 설정에 대한 자세한 내용을 알아보려면 GitHub의 사용자 데이터 사용을 참조하세요.

인스턴스를 시작할 때 `cloud-init`를 사용하여 시작 템플릿에 Amazon EC2 사용자 데이터를 제공할 수 있습니다. 자세한 내용은 cloud-init 설명서를 참조하세요. 사용자 데이터를 사용하여 일반적인 구성 작업을 수행할 수 있습니다. 여기에는 다음 옵션이 포함됩니다.

- 사용자 또는 그룹 포함
- 패키지 설치

관리형 노드 그룹과 함께 사용되는 시작 템플릿의 Amazon EC2 사용자 데이터는 Amazon Linux AMI의 경우 MIME 멀티파트 아카이브 형식이어야 하고 Bottlerocket AMI의 경우 TOML 형식이어야 합니다. 이는 사용자 데이터가 노드가 클러스터에 조인하는 데 필요한 Amazon EKS 사용자 데이터와 병합되기 때문입니다. `kubelet`을 시작하거나 수정하는 명령을 사용자 데이터에 지정하지 마세요. 이는 Amazon EKS에 의해 병합된 사용자 데이터의 일부로 수행됩니다. 노드의 레이블 설정과 같은 특정 `kubelet` 파라미터는 관리형 노드 그룹 API를 통해 직접 구성될 수 있습니다.

> **Note**
>
> 수동으로 시작하거나 사용자 정의 구성 파라미터를 전달하는 등의 고급 `kubelet` 사용자 지정에 대한 자세한 내용은 AMI 지정 부분을 참조하세요. 시작 템플릿에 사용자 정의 AMI ID가 지정된 경우 Amazon EKS는 사용자 데이터를 병합하지 않습니다.

다음 세부 정보는 사용자 데이터 섹션에 대한 자세한 정보를 제공합니다.

Amazon Linux 2 user data

여러 사용자 데이터 블록을 단일 MIME 멀티파트 파일로 결합할 수 있습니다. 예를 들어 Docker 데몬을 구성하는 클라우드 boothook를 사용자 지정 패키지를 설치하는 사용자 데이터 셸 스크립트와 결합할 수 있습니다. MIME 멀티파트 파일은 다음과 같은 구성 요소로 이루어집니다.

- 콘텐츠 유형 및 요소 경계 선언 - `Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="`
- MIME 버전 선언 - `MIME-Version: 1.0`
- 다음 구성 요소를 포함하는 하나 이상의 사용자 데이터 블록:
  - 사용자 데이터 블록의 시작을 나타내는 열린 경계 - `--==MYBOUNDARY==`
  - 블록에 대한 콘텐츠 유형 선언: `Content-Type: text/cloud-config; charset="us-ascii"`. 콘텐츠 유형에 대한 자세한 내용은 cloud-init 문서를 참조하세요.
  - 사용자 데이터 콘텐츠(예: 셸 명령 또는 `cloud-init` 지시어 목록)

- MIME 멀티파트 파일의 끝을 나타내는 폐쇄 경계: --==MYBOUNDARY==--

다음은 직접 파일을 작성하는 데 사용할 수 있는 MIME 멀티파트 파일의 예입니다.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
echo "Running custom user data script"

--==MYBOUNDARY==--
```

Amazon Linux 2023 user data

Amazon Linux 2023(AL2023)에 YAML 구성 스키마를 사용하는 새로운 노드 초기화 프로세스 nodeadm이 도입됩니다. 자체 관리형 노드 그룹 또는 시작 템플릿이 있는 AMI를 사용하는 경우 이제 새 노드 그룹을 생성할 때 추가 클러스터 메타데이터를 명시적으로 제공해야 합니다. 최소 필수 파라미터의 예시는 다음과 같으며, 여기에서 apiServerEndpoint, certificateAuthority 및 서비스 cidr가 필수입니다.

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

대체로 이 구성은 사용자 데이터에 있는 그대로 또는 MIME 멀티파트 문서에 포함되도록 설정합니다.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="BOUNDARY"

--BOUNDARY
```

```
Content-Type: application/node.eks.aws

---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig spec: [...]

--BOUNDARY--
```

AL2에서 이러한 파라미터의 메타데이터는 Amazon EKS `DescribeCluster` API 직접 호출에서 확인되었습니다. AL2023 버전에서는 대형 노드 스케일 업 도중 추가 API 직접 호출로 인해 제한이 발생할 위험이 있기 때문에 이러한 동작이 변경되었습니다. 시작 템플릿이 없는 관리형 노드 그룹을 사용하거나 Karpenter를 사용하는 경우 이 변경 사항이 영향을 미치지 않습니다. `certificateAuthority` 및 서비스 `cidr`에 대한 자세한 내용은 Amazon EKS API 참조의 `DescribeCluster`를 참조하세요.

Bottlerocket user data

Bottlerocket은 사용자 데이터를 TOML 형식으로 구성합니다. Amazon EKS에서 제공하는 사용자 데이터와 병합할 사용자 데이터를 제공할 수 있습니다. 예를 들어 추가 `kubelet` 설정을 제공할 수 있습니다.

```
[settings.kubernetes.system-reserved]
cpu = "10m"
memory = "100Mi"
ephemeral-storage= "1Gi"
```

지원되는 설정에 대한 자세한 내용을 알아보려면 Bottlerocket 설명서를 참조하세요. 사용자 데이터에서 노드 레이블과 테인트를 구성할 수 있습니다. 그러나 대신 노드 그룹 내에서 구성하는 것이 좋습니다. 그러면 Amazon EKS에서 이러한 구성을 적용합니다.

사용자 데이터가 병합되면 서식이 유지되지 않지만 내용은 동일하게 유지됩니다. 사용자 데이터에 제공하는 구성은 Amazon EKS에서 구성하는 모든 설정을 재정의합니다. 따라서 `settings.kubernetes.max-pods` 또는 `settings.kubernetes.cluster-dns-ip`를 설정하면 사용자 데이터의 값이 노드에 적용됩니다.

Amazon EKS는 모든 유효한 TOML을 지원하지 않습니다. 다음은 지원되지 않는 알려진 형식 목록입니다.

- 따옴표 붙은 키 안의 따옴표: `'quoted "value"' = "value"`
- 값의 이스케이프된 따옴표: `str = "I'm a string. \"You can quote me\""`

- 혼합 부동 소수점 및 정수: `numbers = [ 0.1, 0.2, 0.5, 1, 2, 5 ]`
- 배열의 혼합 유형: `contributors = ["foo@example.com", { name = "Baz", email = "baz@example.com" }]`
- 따옴표 붙은 키가 있는 대괄호로 묶은 헤더: `[foo."bar.baz"]`

Windows user data

윈도우 사용자 데이터에서는 PowerShell 명령을 사용합니다. 관리형 노드 그룹을 생성할 때 사용자 지정 사용자 데이터는 Amazon EKS 관리형 사용자 데이터와 결합됩니다. PowerShell 명령과 관리형 사용자 데이터 명령이 모두 하나의 **<powershell></powershell>** 태그 내에서 차례로 나타납니다.

> **Note**
>
> 시작 템플릿에 AMI ID가 지정되어 있지 않으면 사용자 데이터의 Windows Amazon EKS Bootstrap 스크립트를 사용하여 Amazon EKS를 구성하지 마세요.

예시 사용자 데이터는 다음과 같습니다.

```
<powershell>
Write-Host "Running custom user data script"
</powershell>
```

AMI 지정

다음 요구 사항 중 하나가 있는 경우 시작 템플릿의 `ImageId` 필드에 AMI ID를 지정합니다. 추가 정보를 보려면 요구 사항을 선택하세요.

Amazon EKS 최적화 Linux/Bottlerocket AMI에 포함된 **bootstrap.sh** 파일에 인수를 전달하기 위해 사용자 데이터를 제공함

부트스트래핑은 인스턴스가 시작될 때 실행할 수 있는 명령의 추가에 대해 설명하는 데 사용되는 용어입니다. 예를 들어, 부트스트래핑을 통해 추가 kubelet인수를 사용할 수 있습니다. 시작 템플릿을 지정하지 않고 `eksctl`을 사용하여 `bootstrap.sh` 스크립트에 인수를 전달할 수 있습니다. 이를 위해 시작 템플릿의 사용자 데이터 부분에 정보를 지정할 수도 있습니다.

eksctl without specifying a launch template

다음 콘텐츠를 가진 *my-nodegroup.yaml*이라는 파일을 생성합니다: *example value*를 고유한 값으로 바꿉니다. `--apiserver-endpoint`, `--b64-cluster-ca` 및 `--dns-cluster-ip` 인수는 선택 사항입니다. 그러나 이를 정의하면 `bootstrap.sh` 스크립트가 `describeCluster`를 호출하는 것을 방지할 수 있습니다. 이는 노드를 자주 확장하고 축소하는 프라이빗 클러스터 설정이나 클러스터에서 유용합니다. `bootstrap.sh` 스크립트에 대한 자세한 내용을 알아보려면 GitHub의 bootstrap.sh 파일을 참조하세요.

- 유일한 필수 인수는 클러스터 이름(*my-cluster*)입니다.
- ami-*1234567890abcdef0*에 대해 최적화된 AMI ID를 검색하려면 다음 부분의 표를 사용할 수 있습니다.
  - Amazon EKS 최적화 Amazon Linux AMI ID 검색
  - Amazon EKS 최적화 Bottlerocket AMI ID 검색
  - Amazon EKS 최적화 Windows AMI ID 검색
- 클러스터의 *certificate-authority*를 검색하려면 다음 명령을 실행합니다.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- 클러스터의 *api-server-endpoint*를 검색하려면 다음 명령을 실행합니다.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- `--dns-cluster-ip`의 값은 `.10`으로 끝나는 서비스 CIDR입니다. 클러스터의 *service-cidr*를 검색하려면 다음 명령을 실행합니다. 예를 들어 반환 값이 `ipv4 10.100.0.0/16`이면 값은 *10.100.0.10*입니다.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- 이 예에서는 추가 `kubelet` 인수를 제공하여 Amazon EKS 최적화 AMI에 포함된 `bootstrap.sh` 스크립트로 사용자 지정 `max-pods` 값을 설정합니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다. *my-max-pods-value* 선택에 대한 도움말은 각 Amazon EC2 인스턴스 유형의 Amazon EKS 권장 최대 Pods 수 부분을 참조하세요.

```
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code

managedNodeGroups:
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    instanceType: m5.large
    privateNetworking: true
    disableIMDSv1: true
    labels: { x86-al2-specified-mng }
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh my-cluster \
        --b64-cluster-ca certificate-authority \
        --apiserver-endpoint api-server-endpoint \
        --dns-cluster-ip service-cidr.10 \
        --kubelet-extra-args '--max-pods=my-max-pods-value' \
        --use-max-pods false
```

사용 가능한 모든 eksctl config 파일 옵션은 eksctl 설명서의 구성 파일 스키마를 참조하세요. eksctl 유틸리티는 여전히 사용자를 위해 시작 템플릿을 생성하고 해당 사용자 데이터를 config 파일에서 제공되는 데이터로 채웁니다.

다음 명령을 사용하여 노드 그룹을 생성합니다.

```
eksctl create nodegroup --config-file=my-nodegroup.yaml
```

User data in a launch template

시작 템플릿의 사용자 데이터 부분에서 다음 정보를 지정합니다. 모든 *example value*를 고유한 값으로 바꿉니다. --apiserver-endpoint, --b64-cluster-ca 및 --dns-cluster-ip 인수는 선택 사항입니다. 그러나 이를 정의하면 bootstrap.sh 스크립트가 describeCluster를 호출하는 것을 방지할 수 있습니다. 이는 노드를 자주 확장하고 축소하는 프라이빗 클러스터 설정이나 클러스터에서 유용합니다. bootstrap.sh 스크립트에 대한 자세한 내용을 알아보려면 GitHub의 bootstrap.sh 파일을 참조하세요.

- 유일한 필수 인수는 클러스터 이름(*my-cluster*)입니다.
- 클러스터의 *certificate-authority*를 검색하려면 다음 명령을 실행합니다.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text
 --name my-cluster --region region-code
```

- 클러스터의 *api-server-endpoint*를 검색하려면 다음 명령을 실행합니다.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-
cluster --region region-code
```

- `--dns-cluster-ip`의 값은 `.10`으로 끝나는 서비스 CIDR입니다. 클러스터의 *service-cidr*를 검색하려면 다음 명령을 실행합니다. 예를 들어 반환 값이 `ipv4 10.100.0.0/16`이면 값은 *10.100.0.10*입니다.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr"
 --output text --name my-cluster --region region-code
```

- 이 예에서는 추가 `kubelet` 인수를 제공하여 Amazon EKS 최적화 AMI에 포함된 `bootstrap.sh` 스크립트로 사용자 지정 `max-pods` 값을 설정합니다. *my-max-pods-value* 선택에 대한 도움말은 각 Amazon EC2 인스턴스 유형의 Amazon EKS 권장 최대 Pods 수 부분을 참조하세요.

```
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="==MYBOUNDARY=="

--==MYBOUNDARY==
Content-Type: text/x-shellscript; charset="us-ascii"

#!/bin/bash
set -ex
/etc/eks/bootstrap.sh my-cluster \
  --b64-cluster-ca certificate-authority \
  --apiserver-endpoint api-server-endpoint \
  --dns-cluster-ip service-cidr.10 \
  --kubelet-extra-args '--max-pods=my-max-pods-value' \
  --use-max-pods false

--==MYBOUNDARY==--
```

Amazon EKS 최적화 Windows AMI에 포함된 **Start-EKSBootstrap.ps1** 파일에 인수를 전달하기 위해 사용자 데이터를 제공함

부트스트래핑은 인스턴스가 시작될 때 실행할 수 있는 명령의 추가에 대해 설명하는 데 사용되는 용어입니다. 시작 템플릿을 지정하지 않고 eksctl을 사용하여 Start-EKSBootstrap.ps1 스크립트에 인수를 전달할 수 있습니다. 이를 위해 시작 템플릿의 사용자 데이터 부분에 정보를 지정할 수도 있습니다.

사용자 지정 Windows AMI ID를 지정하려면 다음과 같은 사항을 고려하세요.

- 시작 템플릿을 사용하고 사용자 데이터 부분의 필수 부트스트랩 명령을 제공해야 합니다. 원하는 Windows ID를 검색하려면 Amazon EKS 최적화 Windows AMI의 표를 사용할 수 있습니다.
- 몇 가지 제한과 조건이 있습니다. 예를 들어 AWS IAM Authenticator 구성 맵에 eks:kube-proxy-windows를 추가해야 합니다. 자세한 내용은 AMI ID 지정 시 제한과 조건 단원을 참조하십시오.

시작 템플릿의 사용자 데이터 부분에서 다음 정보를 지정합니다. 모든 *example value*를 고유한 값으로 바꿉니다. -APIServerEndpoint, -Base64ClusterCA 및 -DNSClusterIP 인수는 선택 사항입니다. 그러나 이를 정의하면 Start-EKSBootstrap.ps1 스크립트가 describeCluster를 호출하는 것을 방지할 수 있습니다.

- 유일한 필수 인수는 클러스터 이름(*my-cluster*)입니다.
- 클러스터의 *certificate-authority*를 검색하려면 다음 명령을 실행합니다.

```
aws eks describe-cluster --query "cluster.certificateAuthority.data" --output text --
name my-cluster --region region-code
```

- 클러스터의 *api-server-endpoint*를 검색하려면 다음 명령을 실행합니다.

```
aws eks describe-cluster --query "cluster.endpoint" --output text --name my-cluster
 --region region-code
```

- --dns-cluster-ip의 값은 .10으로 끝나는 서비스 CIDR입니다. 클러스터의 *service-cidr*를 검색하려면 다음 명령을 실행합니다. 예를 들어 반환 값이 ipv4 10.100.0.0/16이면 값은 *10.100.0.10*입니다.

```
aws eks describe-cluster --query "cluster.kubernetesNetworkConfig.serviceIpv4Cidr" --
output text --name my-cluster --region region-code
```

- 추가 인수는 부트스트랩 스크립트 구성 파라미터 부분을 참조하세요.

> **Note**
>
> 사용자 지정 서비스 CIDR을 사용하는 경우-ServiceCIDR 파라미터를 사용하여 CIDR을 지정해야 합니다. 그렇지 않으면 클러스터의 Pods에 대한 DNS 확인이 실패합니다.

```
<powershell>
[string]$EKSBootstrapScriptFile = "$env:ProgramFiles\Amazon\EKS\Start-EKSBootstrap.ps1"
& $EKSBootstrapScriptFile -EKSClusterName my-cluster `
  -Base64ClusterCA certificate-authority `
  -APIServerEndpoint api-server-endpoint `
  -DNSClusterIP service-cidr.10
</powershell>
```

특정 보안, 규정 준수 또는 내부 정책 요구 사항으로 인해 사용자 정의 AMI 실행함

자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon Machine Image(AMI)를 참조하세요. Amazon EKS AMI 빌드 사양에는 Amazon Linux를 기반으로 사용자 지정 Amazon EKS AMI를 구축하기 위한 리소스 및 구성 스크립트가 포함되어 있습니다. 자세한 내용은 GitHub에서 Amazon EKS AMI 빌드 사양을 참조하세요. 다른 운영 체제와 함께 설치된 사용자 지정 AMI를 생성하려면 GitHub에서 Amazon EKS 샘플 사용자 지정 AMI를 참조하세요.

> **Important**
>
> AMI를 지정할 때 Amazon EKS는 사용자 데이터를 병합하지 않습니다. 사용자가 필요한 bootstrap 명령을 사용하여 노드를 클러스터에 조인해야 합니다. 노드가 클러스터에 조인되지 않은 경우 Amazon EKS CreateNodegroup 및 UpdateNodegroupVersion 작업도 실패합니다.

AMI ID 지정 시 제한과 조건

다음은 관리형 노드 그룹으로 AMI ID를 지정하는 것과 관련된 제한 및 조건입니다.

- 시작 템플릿에서 AMI ID를 지정하는 것과 AMI ID를 지정하지 않는 것 사이를 전환하려면 새 노드 그룹을 생성해야 합니다.

- 최신 AMI 버전을 사용할 수 있는 경우에도 콘솔에 알림이 표시되지 않습니다. 노드 그룹을 최신 AMI 버전으로 업데이트하려면 업데이트된 AMI ID로 시작 템플릿의 새 버전을 생성해야 합니다. 그런 다음 새 시작 템플릿 버전으로 노드 그룹을 업데이트해야 합니다.
- AMI ID를 지정하는 경우 API에서 다음 필드를 설정할 수 없습니다.
  - `amiType`
  - `releaseVersion`
  - `version`
- AMI ID를 지정하는 경우 API에 설정된 모든 `taints`가 비동기적으로 적용됩니다. 노드가 클러스터에 합류하기 전에 테인트를 적용하려면 `kubelet` 명령줄 플래그를 사용하여 사용자 데이터의 `--register-with-taints`에 테인트를 전달해야 합니다. 자세한 내용은 Kubernetes 설명서의 kubelet을(를) 참조하세요.
- Windows 관리형 노드 그룹에 사용자 지정 AMI ID를 지정할 때는 AWS IAM Authenticator 구성 맵에 `eks:kube-proxy-windows`를 추가합니다. DNS가 제대로 작동하려면 필요합니다.

  1. 편집을 위해 AWS IAM Authenticator 구성 맵을 엽니다.

     ```
     kubectl edit -n kube-system cm aws-auth
     ```

  2. Windows 노드와 연결된 각 `rolearn` 아래의 `groups` 목록에 이 항목을 추가합니다. 구성 맵은 aws-auth-cm-windows.yaml와 비슷해야 합니다.

     ```
     - eks:kube-proxy-windows
     ```

  3. 파일을 저장하고 텍스트 편집기를 종료합니다.

관리형 노드 그룹 삭제

이 주제에서는 Amazon EKS 관리형 노드 그룹을 삭제하는 방법에 대해 설명합니다. 관리형 노드 그룹을 삭제하면 Amazon EKS는 먼저 Auto Scaling 그룹의 최소, 최대 및 원하는 크기를 0으로 설정합니다. 그러면 노드 그룹이 축소됩니다.

각 인스턴스가 종료되기 전에 Amazon EKS는 신호를 보내 해당 노드에서 Pods를 드레이닝합니다. 몇 분 후에 Pods가 드레이닝되지 않을 경우 Amazon EKS는 Auto Scaling이 인스턴스 종료를 계속 진행할 수 있도록 합니다. 모든 인스턴스가 종료되면 Auto Scaling 그룹이 삭제됩니다.

> **Important**
>
> 클러스터의 다른 관리형 노드 그룹에서 사용하지 않는 노드 IAM 역할을 사용하는 관리형 노드 그룹을 삭제하면 역할이 `aws-auth ConfigMap`에서 제거됩니다. 클러스터의 자체 관리형 노드 그룹이 동일한 노드 IAM 역할을 사용하는 경우 자체 관리형 노드는 `NotReady` 상태로 전환됩니다. 또한 클러스터 작업도 중단됩니다. 자체 관리형 노드 그룹에만 사용하는 역할에 매핑을 추가하려는 경우 클러스터의 플랫폼 버전이 액세스 항목 관리의 사전 조건 섹션에 나열된 최소 버전이면 액세스 항목 생성 섹션을 참조하세요. 플랫폼 버전이 액세스 항목에 필요한 최소 버전보다 이전인 경우 항목을 `aws-auth ConfigMap`에 다시 추가할 수 있습니다. 자세한 내용을 보려면 터미널에 `eksctl create iamidentitymapping --help`를 입력합니다.

`eksctl` 또는 AWS Management Console을 사용하여 관리형 노드 그룹을 삭제할 수 있습니다.

eksctl

`eksctl`을 사용하여 관리형 노드 그룹을 삭제하려면

다음 명령을 입력합니다. *`example value`*를 고유한 값으로 바꿉니다.

```
eksctl delete nodegroup \
  --cluster my-cluster \
  --name my-mng \
  --region region-code
```

추가 옵션은 `eksctl` 설명서의 Deleting and draining nodegroups를 참조하세요.

AWS Management Console

AWS Management Console을 사용하여 관리형 노드 그룹 삭제

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 클러스터 페이지에서 삭제할 노드 그룹이 포함된 클러스터를 선택합니다.
3. 선택한 클러스터 페이지에서 컴퓨팅 탭을 선택합니다.
4. 노드 그룹(Node Groups) 섹션에서 삭제할 노드 그룹을 선택합니다. 그런 다음 삭제를 선택합니다.
5. 노드 그룹 삭제 확인 대화 상자에서 노드 그룹의 이름을 입력합니다. 그런 다음 삭제를 선택합니다.

AWS CLI

AWS CLI을 사용하여 관리형 노드 그룹 삭제

1. 다음 명령을 입력합니다. *example value*를 고유한 값으로 바꿉니다.

```
aws eks delete-nodegroup \
  --cluster-name my-cluster \
  --nodegroup-name my-mng \
  --region region-code
```

2. 키보드의 화살표 키를 사용하여 응답 출력을 스크롤합니다. 작업을 마치면 **q** 키를 누릅니다.

추가 옵션은 AWS CLI Command Reference의 `delete-nodegroup` 명령을 참조하세요.

자체 관리형 노드

클러스터에는 Pods가 예약된 하나 이상의 Amazon EC2 노드가 포함되어 있습니다. Amazon EKS 노드는 AWS 계정에서 실행되고 클러스터 API 서버 엔드포인트를 통해 클러스터의 제어 영역에 연결됩니다. Amazon EC2 가격을 기준으로 요금이 청구됩니다. 자세한 정보는 Amazon EC2 요금을 참조하세요.

클러스터에는 여러 노드 그룹이 포함될 수 있습니다. 각 노드 그룹에는 Amazon EC2 Auto Scaling 그룹에 배포된 하나 이상의 노드가 포함되어 있습니다. 그룹 내 노드의 인스턴스 유형은 Karpenter를 사용한 속성 기반 인스턴스 유형 선택을 사용하는 경우와 같이 달라질 수 있습니다. 노드 그룹의 모든 인스턴스는 Amazon EKS 노드 IAM 역할을 사용해야 합니다.

Amazon EKS에서는 Amazon EKS 최적화 AMI라는 특별한 Amazon Machine Image(AMI)를 제공합니다. AMI는 Amazon EKS와 함께 작동하도록 구성되어 있습니다. 구성 요소에는 `containerd`, `kubelet`, AWS IAM 인증자가 포함됩니다. 또한 AMI에는 특별한 부트스트랩 스크립트도 포함되어 있어 클러스터의 제어 영역을 자동으로 찾고 연결할 수 있습니다.

CIDR 블록을 사용하여 클러스터의 퍼블릭 엔드포인트에 대한 액세스를 제한하는 경우 프라이빗 엔드포인트 액세스도 사용 설정하는 것이 좋습니다. 이는 노드가 클러스터와 통신할 수 있도록 하기 위한 것입니다. 프라이빗 엔드포인트를 활성화하지 않은 경우, 퍼블릭 액세스에 대해 지정하는 CIDR 블록에는 VPC의 송신 소스가 포함되어야 합니다. 자세한 내용은 Amazon EKS 클러스터 엔드포인트 액세스 제어 섹션을 참조하세요.

Amazon EKS 클러스터에 자체 관리형 노드를 추가하려면 다음 항목을 참조하세요. 자체 관리형 노드를 수동으로 시작하는 경우 각 노드에 다음 태그를 추가합니다. 자세한 내용은 개별 리소스에 대한 태그 추가 및 삭제를 참조하십시오. 가이드의 단계를 수행하면 필수 태그가 자동으로 노드에 추가됩니다.

키	값
`kubernetes.io/cluster/ my-cluster`	`owned`

일반 Kubernetes 관점에서 노드에 대한 자세한 내용을 알아보려면 Kubernetes 설명서의 노드를 참조하세요.

주제

- 자체 관리형 Amazon Linux 노드 시작하기
- 자체 관리형 Bottlerocket 노드 시작
- 자체 관리형 Windows 노드 시작
- 자체 관리형 노드 업데이트

자체 관리형 Amazon Linux 노드 시작하기

이 주제에서는 Amazon EKS 클러스터에 등록하는 Linux 노드의 Auto Scaling 그룹을 시작하는 방법을 설명합니다. 노드가 클러스터에 조인한 이후 Kubernetes 애플리케이션을 배포할 수 있습니다. `eksctl` 또는 AWS Management Console을 사용하여 자체 관리형 Amazon Linux 노드를 시작할 수도 있습니다. AWS Outposts에서 노드를 시작해야 하는 경우 Outpost에서 자체 관리형 Amazon Linux 노드 시작하기 부분을 참조하세요.

필수 조건

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 클러스터 생성 섹션을 참조하세요. AWS Outposts, AWS Wavelength 또는 AWS Local Zones가 사용된 AWS 리전에 서브넷이 있는 경우 클러스터를 생성할 때 해당 서브넷이 전달되지 않은 상태여야 합니다.
- 노드가 사용할 기존 IAM 역할. 파일을 만들려면 Amazon EKS 노드 IAM 역할 섹션을 참조하세요. 이 역할에 VPC CNI에 대한 정책 중 하나도 없는 경우 VPC CNI 포드에 대해 다음과 같은 별도의 역할이 필요합니다.
- (선택 사항이지만 권장됨) 필요한 IAM 정책이 연결된 자체 IAM 역할로 구성된 Amazon VPC CNI plugin for Kubernetes 추가 기능. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오.

- Amazon EC2 인스턴스 유형 선택에 나열된 고려 사항 숙지. 선택하는 인스턴스 유형에 따라 클러스터 및 VPC에 대한 추가 전제 조건이 있을 수 있습니다.

eksctl

> **Note**
>
> `eksctl`에서는 현재 Amazon Linux 2023을 지원하지 않습니다.

전제 조건

디바이스 또는 `0.175.0`에 설치된 버전 AWS CloudShell 이상의 `eksctl` 명령줄 도구. `eksctl`을 설치 또는 업그레이드하려면 `eksctl` 설명서에서 Installation을 참조하세요.

`eksctl`을 사용하여 자체 관리형 Linux 노드를 시작하려면 다음을 수행합니다.

1. (선택 사항) AmazonEKS_CNI_Policy 관리형 IAM 정책이 Amazon EKS 노드 IAM 역할에 연결되어 있는 경우 Kubernetes `aws-node` 서비스 계정에 연결하는 IAM 역할에 대신 할당하는 것이 좋습니다. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오.
2. 다음 명령은 기존 클러스터의 노드 그룹을 생성합니다. *`al-nodes`*을 노드 그룹의 이름으로 바꿉니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다. *`my-cluster`*를 클러스터 이름으로 바꿉니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다. 나머지 *`example value`*를 고유한 값으로 바꿉니다. 노드는 기본적으로 제어 영역과 동일한 Kubernetes 버전으로 작성됩니다.

   `--node-type`의 값을 선택하기 전에 Amazon EC2 인스턴스 유형 선택 부분을 검토하세요.

   *`my-key`*를 Amazon EC2 키 페어 또는 퍼블릭 키 이름으로 바꿉니다. 이 키는 노드를 시작한 후 SSH로 연결하는 데 사용됩니다. Amazon EC2 키 페어가 아직 없는 경우 AWS Management Console에서 새로 생성할 수 있습니다. 자세한 내용을 알아보려면 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하세요.

   다음 명령을 사용하여 노드 그룹을 생성합니다.

> **Important**
>
> 노드 그룹을 AWS Outposts, Wavelength 또는 Local Zones 서브넷에 배포하려는 경우 추가 고려 사항이 있습니다.
>
> - 클러스터를 생성할 때 서브넷이 전달되지 않았어야 합니다.
> - 서브넷과 `volumeType: gp2`을 지정하는 구성 파일로 노드 그룹을 생성해야 합니다. 자세한 내용은 `eksctl` 문서의 구성 파일을 사용하여 nodegroup 생성 및 구성 파일 스키마 부분을 참조하세요.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --name al-nodes \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --ssh-access \
  --managed=false \
  --ssh-public-key my-key
```

다음과 같은 노드 그룹을 배포할 수 있습니다.

- Pods에 기본 구성보다 훨씬 더 많은 수의 IP 주소를 할당할 수 있는 노드 그룹을 배포하려면 Amazon EC2 노드에 사용 가능한 IP 주소 증량 부분을 참조하세요.
- 인스턴스와 다른 CIDR 블록의 Pods에 IPv4 주소를 할당할 수 있는 노드 그룹을 배포하려면 포드에 대한 사용자 지정 네트워킹 부분을 참조하세요.
- Pods 및 서비스에 IPv6 주소를 할당할 수 있는 노드 그룹을 배포하려면 클러스터, Pods 및 services용 IPv6 주소 섹션을 참조하세요.
- `containerd` 런타임을 사용하는 노드 그룹을 배포하려면 `config` 파일을 사용하여 노드 그룹을 배포해야 합니다. 자세한 내용은 Docker에서 `containerd`로 마이그레이션 테스트 단원을 참조하십시오.
- 아웃바운드 인터넷 액세스가 없는 노드 그룹을 배포하려면 프라이빗 클러스터 요구 사항 섹션을 참조하세요.

사용 가능한 모든 옵션 및 기본값의 전체 목록을 보려면 다음 명령을 입력합니다.

```
eksctl create nodegroup --help
```

노드가 클러스터에 조인하지 못한 경우 문제 해결 가이드의 노드가 클러스터 조인에 실패 섹션을 참조하세요.

예제 출력은 다음과 같습니다. 노드가 생성되는 동안 여러 줄이 출력됩니다. 출력의 마지막 줄 중 하나는 다음 예제 줄과 유사합니다.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (선택 사항) 샘플 애플리케이션을 배포하여 클러스터 및 Linux 노드를 테스트합니다.
4. 다음과 같은 조건에 해당하면 IMDS에 대한 Pod 액세스를 차단하는 것이 좋습니다.
   - Pods에 필요한 최소 권한만 있도록 모든 Kubernetes 서비스 계정에 IAM 역할을 할당할 계획입니다.
   - 클러스터의 어떤 Pods도 현재 AWS 리전 검색과 같은 다른 이유로 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 액세스할 필요가 없습니다.

   자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 부분을 참조하세요.

AWS Management Console

1단계: Linux을 사용하여 자체 관리형 AWS Management Console 노드를 시작하기

1. AWS CloudFormation 템플릿의 최신 버전 다운로드

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/
amazon-eks-nodegroup.yaml
```

2. 클러스터 상태가 ACTIVE가 되기를 기다립니다. 클러스터가 활성화되기 전에 노드를 시작하면 노드가 클러스터에 등록되지 않고 노드를 다시 시작해야 합니다.
3. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.

4. 스택 생성(Create stack)을 선택한 다음 새 리소스 사용(표준)(With new resources(standard))을 선택합니다.
5. 템플릿 지정(Specify template)에서 템플릿 파일 업로드(Upload a template file)를 선택한 다음 파일 선택(Choose file)을 선택합니다.
6. 다운로드한 `amazon-eks-nodegroup.yaml` 파일을 선택합니다.
7. 다음을 선택합니다.
8. 스택 세부 정보 지정(Specify stack details) 페이지에서 이에 따라 다음 파라미터를 입력한 후 다음(Next)을 선택합니다.

   - 스택 이름: AWS CloudFormation 스택에 대한 스택 이름을 선택합니다. 예를 들어 ***my-cluster-nodes***라고 할 수 있습니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다.
   - ClusterName: Amazon EKS 클러스터 생성 시 사용할 이름을 입력합니다. 이 이름은 클러스터 이름과 같아야 합니다. 그렇지 않으면 노드가 클러스터에 조인할 수 없습니다.
   - ClusterControlPlaneSecurityGroup: VPC를 생성할 때 생성한 AWS CloudFormation 출력에서 SecurityGroups값을 선택합니다.

     다음 단계에서는 해당 그룹을 검색하는 한 가지 작업을 보여줍니다.

     1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
     2. 클러스터의 이름을 선택합니다.
     3. 네트워킹 탭을 선택합니다.
     4. ClusterControlPlaneSecurityGroup 드롭다운 목록에서 선택할 때 추가 보안 그룹(Additional Security Group) 값을 참조로 사용하세요.
   - NodeGroupName: 노드 그룹의 이름을 입력합니다. 이 이름은 나중에 노드에 대해 생성된 Auto Scaling 노드 그룹을 식별하는 데 사용할 수 있습니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다.
   - NodeAutoScalingGroupMinSize: Auto Scaling 그룹이 축소할 수 있는 노드의 최소 노드 수를 입력합니다.
   - NodeAutoScalingGroupDesiredCapacity: 스택을 생성할 때 조정할 원하는 노드 수를 입력합니다.
   - NodeAutoScalingGroupMaxSize: Auto Scaling 그룹이 확장할 수 있는 노드의 최대 노드 수를 입력합니다.

- NodeInstanceType: 노드에 대한 인스턴스 유형을 선택합니다. 자세한 내용은 Amazon EC2 인스턴스 유형 선택 단원을 참조하십시오.
- NodeImageIdSSMParam: 가변 Kubernetes 버전용 최신 Amazon EKS 최적화 AMI의 Amazon EC2 Systems Manager 파라미터로 미리 채워집니다. Amazon EKS에서 지원되는 다른 Kubernetes 마이너 버전을 사용하려면 *1.XX*를 다른 지원되는 버전으로 바꿉니다. 클러스터와 동일한 Kubernetes 버전을 지정하는 것이 좋습니다.

  *amazon-linux-2*를 다른 AMI 유형으로 바꿀 수도 있습니다. 자세한 내용은 Amazon EKS 최적화 Amazon Linux AMI ID 검색 단원을 참조하십시오.

  > **Note**
  >
  > Amazon EKS 노드 AMI는 Amazon Linux를 기반으로 합니다. Amazon Linux 보안 센터에서 Amazon Linux 2의 보안 또는 프라이버시 이벤트를 추적하거나 관련 RSS 피드를 구독하세요. 보안 및 프라이버시 이벤트에는 문제의 개요, 영향을 받는 패키지 및 인스턴스를 업데이트하여 문제를 해결하는 방법이 포함됩니다.

- NodeImageId: (선택 사항) Amazon EKS 최적화 AMI 대신 사용자 정의 AMI를 사용 중인 경우 해당 AWS 리전에 노드 AMI ID를 입력합니다. 여기에서 값을 지정하면 NodeImageIdSSMParam 필드에 있는 모든 값이 재정의됩니다.
- NodeVolumeSize: 노드에 대해 루트 볼륨 크기를 GiB 단위로 지정합니다.
- NodeVolumeType: 노드의 루트 볼륨 유형을 지정합니다.
- KeyName: 시작 이후 SSH를 사용하여 노드에 연결하는 데 사용할 수 있는 Amazon EC2 SSH 키 페어 이름을 입력합니다. Amazon EC2 키 페어가 아직 없는 경우 AWS Management Console에서 새로 생성할 수 있습니다. 자세한 내용을 알아보려면 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하세요.

  > **Note**
  >
  > 여기에 키 페어를 입력하지 않으면 AWS CloudFormation 스택이 생성되지 않습니다.

- BootstrapArguments: 별도의 `kubelet` 인수와 같이 노드 부트스트랩 스크립트에 전달할 선택적 인수를 지정합니다. 자세한 내용은 GitHub의 부트스트랩 스크립트 사용 정보를 참조하세요.

  다음과 같은 노드 그룹을 배포할 수 있습니다.

  - Pods에 기본 구성보다 훨씬 더 많은 수의 IP 주소를 할당할 수 있는 노드 그룹을 배포하려면 Amazon EC2 노드에 사용 가능한 IP 주소 증량 부분을 참조하세요.
  - 인스턴스와 다른 CIDR 블록의 Pods에 IPv4 주소를 할당할 수 있는 노드 그룹을 배포하려면 포드에 대한 사용자 지정 네트워킹 부분을 참조하세요.
  - Pods 및 서비스에 IPv6 주소를 할당할 수 있는 노드 그룹을 배포하려면 클러스터, Pods 및 services용 IPv6 주소 섹션을 참조하세요.
  - containerd 런타임을 사용하는 노드 그룹을 배포하려면 config 파일을 사용하여 노드 그룹을 배포해야 합니다. 자세한 내용은 Docker에서 containerd로 마이그레이션 테스트 단원을 참조하십시오.
  - 아웃바운드 인터넷 액세스가 없는 노드 그룹을 배포하려면 프라이빗 클러스터 요구 사항 부분을 참조하세요.
- DisableIMDSv1: 기본적으로 각 노드는 인스턴스 메타데이터 서비스 버전 1(IMDSv1) 및 IMDSv2를 지원합니다. IMDSv1을 사용 중지할 수 있습니다. 향후 노드 그룹의 노드와 Pods가 MDSv1을 사용하지 못하게 하려면 DisableIMDSv1을 true로 설정합니다. IMDS에 대한 자세한 내용은 인스턴스 메타데이터 서비스 구성을 참조하세요. 노드의 IMDS 액세스 제한에 대한 자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한을 참조하세요.
- VpcId: 생성한 VPC에 대한 ID를 입력합니다.
- 서브넷: VPC용으로 생성한 서브넷을 선택합니다. Amazon EKS 클러스터에 대해 VPC 생성에 설명된 단계를 사용하여 VPC를 생성한 경우, VPC 내에서 노드가 시작될 프라이빗 서브넷만 지정합니다. 클러스터의 네트워킹 탭에서 각 서브넷 링크를 열면 어떤 서브넷이 프라이빗인지 확인할 수 있습니다.

> **Important**
>
> - 서브넷 중 하나가 퍼블릭 서브넷인 경우 자동 퍼블릭 IP 주소 할당 설정을 사용하도록 설정해야 합니다. 퍼블릭 서브넷에 대해 이 설정을 사용하지 않으면 해당 퍼블릭 서브넷에 배포하는 노드에는 퍼블릭 IP 주소가 할당되지 않으며 클러스터 또는 기타 AWS 서비스와 통신할 수 없습니다. 서브넷이 2020년 3월 26일 이전에 Amazon EKS AWS CloudFormation VPC 템플릿 또는 eksctl을 사용하여 배포된 경우 퍼블릭 서브넷에 대해 자동 퍼블릭 IP 주소 할당이 사용 중지됩니다. 서브넷에 퍼블릭 IP 주소 할당을 활성화하는 방법에 대한 자세한 내용을 알아보려면 서브넷에 대한 퍼블릭 IPv4 주소 지정 속성 수정을 참조하세요. 노드가 프라이빗 서브넷에 배포되면 NAT 게이트웨이를 통해 클러스터 및 다른 AWS 서비스와 통신할 수 있습니다.

- 서브넷에 인터넷 액세스가 없는 경우 프라이빗 클러스터 요구 사항의 고려 사항 및 추가 단계를 알고 있는지 확인합니다.
- AWS Outposts, Wavelength 또는 Local Zones 서브넷을 선택하는 경우 클러스터를 생성할 때 해당 서브넷이 전달되지 않은 상태여야 합니다.

9. 스택 옵션 구성(Configure stack options) 페이지에서 원하는 선택 항목을 선택한 후 다음(Next)을 선택합니다.
10. AWS CloudFormation에서 IAM 리소스를 생성할 수 있음을 인정합니다.(I acknowledge that might create IAM resources.)의 왼쪽에 있는 확인란을 선택한 다음 스택 생성(Create stack)을 선택합니다.
11. 스택이 생성된 후 콘솔에서 이를 선택하고 출력(Outputs)을 선택합니다.
12. 생성된 노드 그룹에 대해 NodeInstanceRole을 기록합니다. Amazon EKS 노드를 구성할 때 필요합니다.

2단계: 노드가 클러스터에 조인하도록 하려면

> **Note**
>
> 아웃바운드 인터넷 액세스 없이 프라이빗 VPC 내에서 노드를 시작한 경우 VPC 내에서 노드가 클러스터에 조인하도록 설정해야 합니다.

1. `aws-auth ConfigMap`이 이미 있는지 확인합니다.

```
kubectl describe configmap -n kube-system aws-auth
```

2. `aws-auth ConfigMap`이 표시되면 필요에 따라 이를 업데이트합니다.

   a. 편집을 위해 `ConfigMap`을 엽니다.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. 필요에 따라 새 `mapRoles` 항목을 추가합니다. `rolearn` 값을 이전 절차에서 기록한 NodeInstanceRole 값으로 설정합니다.

```
[...]
data:
```

```
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

c. 파일을 저장하고 텍스트 편집기를 종료합니다.

3. "Error from server (NotFound): configmaps "aws-auth" not found라는 오류 메시지가 표시되면 스톡 ConfigMap을 적용합니다.

a. 구성 맵을 다운로드합니다.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/aws-auth-cm.yaml
```

b. aws-auth-cm.yaml 파일에서 rolearn 값을 이전 절차에서 기록한 NodeInstanceRole 값으로 설정합니다. 이 작업은 텍스트 편집기를 사용하거나 *my-node-instance-role*을 대체하고 다음 명령을 실행하여 수행할 수 있습니다.

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-
instance-role|' aws-auth-cm.yaml
```

c. 구성을 적용합니다. 이 명령을 완료하는 데 몇 분이 걸릴 수 있습니다.

```
kubectl apply -f aws-auth-cm.yaml
```

4. 노드의 상태를 확인하고 Ready 상태가 될 때까지 대기합니다.

```
kubectl get nodes --watch
```

Ctrl+C를 입력하여 쉘 프롬프트로 돌아갑니다.

> **Note**
>
> 권한 부여 또는 리소스 유형 오류가 표시되는 경우 문제 해결 주제의 권한이 없거나 액세스가 거부됨(kubectl) 섹션을 참조하세요.

노드가 클러스터에 조인하지 못한 경우 문제 해결 가이드의 노드가 클러스터 조인에 실패 섹션을 참조하세요.

5. (GPU 노드만 해당) GPU 인스턴스 유형과 Amazon EKS 최적화 가속 AMI를 선택한 경우 클러스터에 Kubernetes용 NVIDIA 디바이스 플러그인을 DaemonSet(으)로 적용해야 합니다. 다음 명령을 실행하기 전에 *vX.X.X*을(를) 원하는 NVIDIA/k8s-device-plugin 버전으로 교체합니다.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/nvidia-device-plugin.yml
```

3단계: 추가 작업

1. (선택 사항) 샘플 애플리케이션을 배포하여 클러스터 및 Linux 노드를 테스트합니다.
2. (선택 사항) AmazonEKS_CNI_Policy 관리형 IAM 정책(IPv4 클러스터가 있는 경우) 또는 *AmazonEKS_CNI_IPv6_Policy*(IPv6 클러스터가 있는 경우 직접 생성)가 the section called “노드 IAM 역할”에 연결된 경우 대신 Kubernetes aws-node 서비스 계정에 연결하는 IAM 역할에 할당하는 것이 좋습니다. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오.
3. 다음과 같은 조건에 해당하면 IMDS에 대한 Pod 액세스를 차단하는 것이 좋습니다.
   - Pods에 필요한 최소 권한만 있도록 모든 Kubernetes 서비스 계정에 IAM 역할을 할당할 계획입니다.
   - 클러스터의 어떤 Pods도 현재 AWS 리전 검색과 같은 다른 이유로 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 액세스할 필요가 없습니다.

   자세한 내용은 워커 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 섹션을 참조하세요.

ML용 용량 블록

> **Important**
>
> 이 기능은 현재 미국 동부(오하이오) 및 미국 동부(버지니아 북부) AWS 리전의 P5 인스턴스와 미국 동부(오하이오) 및 미국 서부(오레곤) AWS 리전의 P4d에만 사용할 수 있습니다.

기계 학습용 용량 블록(ML)을 사용하면 향후 날짜에 GPU 인스턴스를 예약하여 단기간의 ML 워크로드를 지원할 수 있습니다. 용량 블록 내에서 실행되는 인스턴스는 Amazon EC2 UltraClusters 내에 자동으로 서로 가깝게 배치되므로 클러스터 배치 그룹을 사용할 필요가 없습니다. 자세한 내용을 알아보려면 Linux 인스턴스용 Amazon EC2 사용 설명서의 ML 용량 블록을 참조하세요.

Amazon EKS와 함께 용량 블록을 사용하여 자체 관리형 노드를 프로비저닝하고 규모를 조정할 수 있습니다. 다음 단계는 일반적인 예제 개요를 제공합니다.

1. AWS Management Console에 시작 템플릿을 생성합니다. 자세한 내용은 Amazon EC2 Auto Scaling 사용 설명서의 Use Capacity Blocks for machine learning workloads를 참조하세요.

   인스턴스 유형 구성과 Amazon Machine Image(AMI)를 포함해야 합니다.
2. 용량 예약 ID를 사용하여 용량 블록을 시작 템플릿에 연결합니다.

   용량 블록을 대상으로 하는 시작 템플릿을 생성하는 예제 AWS CloudFormation 템플릿은 다음과 같습니다.

   ```
   NodeLaunchTemplate:
       Type: "AWS::EC2::LaunchTemplate"
       Properties:
         LaunchTemplateData:
           InstanceMarketOptions:
             MarketType: "capacity-block"
           CapacityReservationSpecification:
             CapacityReservationTarget:
               CapacityReservationId: "cr-02168da1478b509e0"
           IamInstanceProfile:
             Arn: iam-instance-profile-arn
           ImageId: image-id
           InstanceType: p5.48xlarge
           KeyName: key-name
           SecurityGroupIds:
           - sg-05b1d815d1EXAMPLE
           UserData: user-data
   ```

   용량 블록은 영역 단위이므로 예약이 이루어진 가용 영역의 서브넷을 전달해야 합니다.
3. 용량 예약이 활성화되기 전에 자체 관리형 노드 그룹을 생성하는 경우 원하는 용량을 0(으)로 설정합니다. 노드 그룹을 생성할 때는 용량이 예약된 가용 영역에 해당하는 서브넷만 지정하도록 해야 합니다.

사용할 수 있는 샘플 CloudFormation 템플릿은 다음과 같습니다. 이 예제에서는 이전 예제에 표시된 `AWS::Amazon EC2::LaunchTemplate` 리소스의 `LaunchTemplateId`와(과) `Version`을(를) 가져옵니다. 또한 동일한 템플릿의 다른 곳에 선언된 `DesiredCapacity` `MaxSize`, `MinSize` 및 `VPCZoneIdentifier`의 값도 가져옵니다.

```
NodeGroup:
  Type: "AWS::AutoScaling::AutoScalingGroup"
  Properties:
    DesiredCapacity: !Ref NodeAutoScalingGroupDesiredCapacity
    LaunchTemplate:
      LaunchTemplateId: !Ref NodeLaunchTemplate
      Version: !GetAtt NodeLaunchTemplate.LatestVersionNumber
    MaxSize: !Ref NodeAutoScalingGroupMaxSize
    MinSize: !Ref NodeAutoScalingGroupMinSize
    VPCZoneIdentifier: !Ref Subnets
    Tags:
      - Key: Name
        PropagateAtLaunch: true
        Value: !Sub ${ClusterName}-${NodeGroupName}-Node
      - Key: !Sub kubernetes.io/cluster/${ClusterName}
        PropagateAtLaunch: true
        Value: owned
```

4. 노드 그룹이 성공적으로 생성되면 생성된 노드 그룹의 `NodeInstanceRole`을(를) 기록해야 합니다. 노드 그룹이 규모를 조정할 때 새 노드가 클러스터에 가입하고 Kubernetes이(가) 노드를 인식할 수 있도록 하려면 이 정보가 필요합니다. 자세한 내용은 자체 관리형 Amazon Linux 노드 시작하기의 AWS Management Console 지침을 참조하세요.
5. 용량 블록 예약 시간에 맞춰 Auto Scaling 그룹에 대해 예약된 규모 조정 정책을 생성하는 것이 좋습니다. 자세한 정보는 Amazon EC2 Auto Scaling 사용 설명서의 Amazon EC2 Auto Scaling 예정 스케일링을 참조하세요.

   용량 블록 종료 시간 30분 전까지 예약한 모든 인스턴스를 사용할 수 있습니다. 해당 시점에 계속 실행 중인 인스턴스는 종료되기 시작합니다. 노드를 부드럽게 드레인하는 데 충분한 시간을 할애하려면 용량 블록 예약 종료 시각 30분 전까지 0으로 규모를 조정하도록 스케일링을 예약하는 것이 좋습니다.

   용량 예약이 `Active`이(가)될 때마다 수동으로 규모를 조정하려면 용량 블록 예약을 시작할 때 Auto Scaling 그룹의 원하는 용량을 업데이트해야 합니다. 또한 용량 블록 예약 종료 시각 30분 전까지 수동으로 스케일 다운해야 합니다.

6. 노드 그룹은 이제 워크로드와 Pods을(를) 처리할 준비가 되었으며 일정을 계획할 수 있습니다.
7. Pods을(를) 부드럽게 드레이닝하려면 AWS 노드 종료 핸들러를 설정하는 것이 좋습니다. 이 핸들러는 EventBridge를 사용하여 Amazon EC2 Auto Scaling에서 "ASG 스케일 인" 수명 주기 이벤트를 감시하고 인스턴스가 사용할 수 없게 되기 전에 Kubernetes 컨트롤 플레인이 필요한 조치를 취하도록 할 수 있습니다. 그렇지 않으면 Pods 및 Kubernetes 객체가 보류 상태에서 멈추게 됩니다. 자세한 내용은 GitHub의 AWS 노드 종료 핸들러를 참조하세요.

   노드 종료 핸들러를 설정하지 않은 경우, 30분 전에 수동으로 Pods의 드레인을 시작하여 부드럽게 드레이닝할 수 있는 충분한 시간을 확보하는 것이 좋습니다.

자체 관리형 Bottlerocket 노드 시작

> **Note**
>
> 관리형 노드 그룹은 사용 사례에 대한 몇 가지 이점을 제공할 수 있습니다. 자세한 내용은 관리형 노드 그룹 단원을 참조하십시오.

이 주제에서는 Amazon EKS 클러스터에 등록하는 Bottlerocket 노드의 Auto Scaling을 시작하는 방법을 설명합니다. Bottlerocket은 AWS의 Linux 기반 오픈 소스 운영 체제로서 가상 머신 또는 베어 메탈 호스트에서 컨테이너를 실행하는 데 사용할 수 있습니다. 노드가 클러스터에 조인한 이후 Kubernetes 애플리케이션을 배포할 수 있습니다. Bottlerocket에 대한 자세한 내용을 알아보려면 GitHub의 Amazon EKS를 통해 Bottlerocket AMI 사용 및 `eksctl` 설명서의 사용자 지정 AMI 지원을 참조하세요.

현재 위치 업그레이드에 대한 자세한 내용을 알아보려면 Bottlerocket의 GitHub 업데이트 오퍼레이터를 참조하세요.

> **Important**
>
> - Amazon EKS 노드는 표준 Amazon EC2 인스턴스이고, 일반 Amazon EC2 인스턴스 가격을 기반으로 비용이 청구됩니다. 자세한 설명은 Amazon EC2 요금을 참조하세요.
> - Bottlerocket 노드를 AWS Outposts의 Amazon EKS 확장 클러스터에서 시작할 수 있지만 AWS Outposts의 로컬 클러스터에서는 시작할 수 없습니다. 자세한 내용은 AWS Outposts에 대한 Amazon EKS 단원을 참조하십시오.
> - x86 또는 Arm 프로세서가 있는 Amazon EC2 인스턴스에 배포할 수 있습니다. 그러나 Inferentia 칩이 있는 인스턴스에는 배포할 수 없습니다.

- Bottlerocket은 AWS CloudFormation과 호환됩니다. 그러나 Amazon EKS용 Bottlerocket 노드를 배포하기 위해 복사할 수 있는 공식 CloudFormation 템플릿은 없습니다.
- Bottlerocket 이미지는 SSH 서버나 셸과 함께 제공되지 않습니다. 대역 외 액세스 방법을 사용하여 SSH가 관리 컨테이너를 사용 설정하고 사용자 데이터와 함께 일부 부트스트래핑 구성 단계를 전달하도록 할 수 있습니다. 자세한 내용은 GitHub의 bottlerocket README.md에 있는 관련 부분을 참조하세요.
  - 탐색
  - 관리자 컨테이너
  - Kubernetes 설정

eksctl을 사용하여 Bottlerocket 노드 시작

이 절차에는 `eksctl` 버전 `0.175.0` 이상이 필요합니다. 버전은 다음 명령을 통해 확인할 수 있습니다.

```
eksctl version
```

`eksctl` 설치 또는 업데이트에 대한 지침은 `eksctl` 설명서에서 Installation을 참조하세요.

> **Note**
>
> 이 방법은 `eksctl`로 생성된 클러스터에만 사용할 수 있습니다.

1. 다음 콘텐츠를 디바이스에 복사합니다. *`my-cluster`*를 클러스터 이름으로 바꿉니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다. *`ng-bottlerocket`*을 노드 그룹의 이름으로 바꿉니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다. Arm 인스턴스에 배포하려면 *`m5.large`*을 Arm 인스턴스 유형으로 바꿉니다. *`my-ec2-keypair-name`*을 시작 이후 SSH를 사용하여 노드에 연결하는 데 사용할 수 있는 Amazon EC2 SSH 키 페어 이름으로 변경합니다. Amazon EC2 키 페어가 아직 없는 경우 AWS Management Console에서 새로 생성할 수 있습니다. 자세한 내용을 알아보려면 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하세요. 나머지 *`example values`*를 고유한 값으로 모두 바꿉니다. 다 바꾼 후 수정된 명령을 실행하여 `bottlerocket.yaml` 파일을 생성합니다.

   Arm Amazon EC2 인스턴스 유형을 지정하는 경우 배포하기 전에 Amazon EKS 최적화 Arm Amazon Linux AMI의 고려 사항을 검토하세요. 사용자 지정 AMI를 사용하여 배포하는 방법에 대

한 지침은 Bottlerocket의 GitHub 빌드 및 `eksctl` 설명서의 사용자 지정 AMI 지원을 참조하세요. 관리형 노드 그룹을 배포하려면 시작 템플릿을 사용하여 사용자 정의 AMI를 배포합니다. 자세한 내용은 사용자 지정 시작 템플릿이 있는 관리형 노드 단원을 참조하십시오.

> **Important**
>
> 노드 그룹을 AWS Outposts, AWS Wavelength 또는 AWS Local Zones 서브넷에 배포하려면 클러스터를 생성할 때 AWS Outposts, AWS Wavelength 또는 AWS Local Zones 서브넷을 전달하지 마세요. 다음 예에서는 서브넷을 지정해야 합니다. 자세한 내용은 `eksctl` 문서에서 구성 파일을 사용하여 nodegroup 생성 및 구성 파일 스키마 부분을 참조하세요. AWS 리전를 클러스터가 있는 *region-code*으로 바꿉니다.

```
cat >bottlerocket.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: '1.29'

iam:
  withOIDC: true

nodeGroups:
  - name: ng-bottlerocket
    instanceType: m5.large
    desiredCapacity: 3
    amiFamily: Bottlerocket
    ami: auto-ssm
    iam:
       attachPolicyARNs:
          - arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
          - arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
          - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
          - arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
    ssh:
        allow: true
        publicKeyName: my-ec2-keypair-name
```

```
EOF
```

2. 다음 명령을 사용하여 노드를 배포합니다.

```
eksctl create nodegroup --config-file=bottlerocket.yaml
```

예제 출력은 다음과 같습니다.

노드가 생성되는 동안 여러 줄이 출력됩니다. 출력의 마지막 줄 중 하나는 다음 예제 줄과 유사합니다.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (선택 사항) Amazon EBS CSI 플러그 인을 사용하여 Bottlerocket 노드에 Kubernetes 영구 볼륨을 생성합니다. 기본 Amazon EBS 드라이버는 Bottlerocket에 포함되지 않은 파일 시스템 도구에 의존합니다. 드라이버를 사용하여 스토리지 클래스를 생성하는 방법에 대한 자세한 내용은 Amazon EBS CSI 드라이버 부분을 참조하세요.
4. (선택 사항) 기본적으로 kube-proxy는 Bottlerocket이 부팅 시에 원래 설정한 것과 다를 수 있는 기본값으로 nf_conntrack_max 커널 파라미터를 설정합니다. Bottlerocket의 기본 설정을 유지하려면 다음 명령을 사용하여 kube-proxy 구성을 편집합니다.

```
kubectl edit -n kube-system daemonset kube-proxy
```

--conntrack-max-per-core 및 --conntrack-min을 다음 예에 있는 kube-proxy 인수에 추가합니다. 0으로 설정할 경우 변경하지 않음을 나타냅니다.

```
      containers:
      - command:
        - kube-proxy
        - --v=2
        - --config=/var/lib/kube-proxy-config/config
        - --conntrack-max-per-core=0
        - --conntrack-min=0
```

5. (선택 사항) 샘플 애플리케이션을 배포하여 Bottlerocket 노드를 테스트합니다.
6. 다음과 같은 조건에 해당하면 IMDS에 대한 Pod 액세스를 차단하는 것이 좋습니다.

   - Pods에 필요한 최소 권한만 있도록 모든 Kubernetes 서비스 계정에 IAM 역할을 할당할 계획입니다.

- 클러스터의 어떤 Pods도 현재 AWS 리전 검색과 같은 다른 이유로 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 액세스할 필요가 없습니다.

자세한 내용은 워커 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 섹션을 참조하세요.

자체 관리형 Windows 노드 시작

이 주제에서는 Amazon EKS 클러스터에 등록하는 Windows 노드의 Auto Scaling을 시작하는 방법을 설명합니다. 노드가 클러스터에 조인한 이후 Kubernetes 애플리케이션을 배포할 수 있습니다.

> **Important**
>
> - Amazon EKS 노드는 표준 Amazon EC2 인스턴스이고, 일반 Amazon EC2 인스턴스 가격을 기반으로 비용이 청구됩니다. 자세한 설명은 Amazon EC2 요금을 참조하세요.
> - Windows 노드를 AWS Outposts의 Amazon EKS 확장 클러스터에서 시작할 수 있지만 AWS Outposts의 로컬 클러스터에서는 시작할 수 없습니다. 자세한 내용은 AWS Outposts에 대한 Amazon EKS 단원을 참조하십시오.

클러스터에 Windows 지원을 사용합니다. Windows 노드 그룹을 시작하기 전에 중요한 고려 사항을 검토하는 것이 좋습니다. 자세한 내용은 Windows 지원 활성화 단원을 참조하십시오.

`eksctl` 또는 AWS Management Console을 사용하여 자체 관리형 Windows 노드를 시작할 수 있습니다.

eksctl

`eksctl`을 사용하여 자체 관리형 Windows 노드를 시작하려면 다음을 수행합니다.

이 절차에서는 `eksctl`을 설치하고 `eksctl` 버전 `0.175.0` 이상을 요구합니다. 버전은 다음 명령을 통해 확인할 수 있습니다.

```
eksctl version
```

`eksctl` 설치 또는 업데이트에 대한 지침은 `eksctl` 설명서에서 Installation을 참조하세요.

> **Note**
>
> 이 방법은 `eksctl`로 생성된 클러스터에만 사용할 수 있습니다.

1. (선택 사항) AmazonEKS_CNI_Policy 관리형 IAM 정책(IPv4 클러스터가 있는 경우) 또는 *AmazonEKS_CNI_IPv6_Policy*(IPv6 클러스터가 있는 경우 직접 생성)가 the section called "노드 IAM 역할"에 연결된 경우 대신 Kubernetes `aws-node` 서비스 계정에 연결하는 IAM 역할에 할당하는 것이 좋습니다. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오.
2. 이 절차에서는 기존 클러스터가 있다고 가정합니다. Windows 노드 그룹을 추가할 Amazon EKS 클러스터 및 Amazon Linux 노드 그룹이 아직 없는 경우에는 Amazon EKS 시작하기 - `eksctl` 가이드를 따르는 것이 좋습니다. 이 설명서에서는 Amazon Linux 노드로 Amazon EKS 클러스터를 생성하는 방법에 대한 전체 시연을 제공합니다.

   다음 명령을 사용하여 노드 그룹을 생성합니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다. *my-cluster*을 클러스터 이름으로 교체합니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다. *ng-windows*을 노드 그룹의 이름으로 바꿉니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다. Kubernetes 버전 `1.24` 이상의 경우 *2019*를 2022로 바꾸어 Windows Server 2022를 사용할 수 있습니다. 나머지 *example values*를 고유한 값으로 바꿉니다.

   > **Important**
   >
   > 노드 그룹을 AWS Outposts, AWS Wavelength 또는 AWS Local Zones 서브넷에 배포하려면 클러스터를 생성할 때 AWS Outposts, Wavelength 또는 로컬 영역 서브넷을 전달하지 마세요. 구성 파일을 사용하여 노드 그룹을 생성합니다. 이때 AWS Outposts, Wavelength 또는 Local Zones 서브넷을 지정합니다. 자세한 내용은 `eksctl` 문서의 구성 파일을 사용하여 nodegroup 생성 및 구성 파일 스키마 섹션을 참조하세요.

```
eksctl create nodegroup \
    --region region-code \
    --cluster my-cluster \
    --name ng-windows \
    --node-type t2.large \
    --nodes 3 \
```

```
    --nodes-min 1 \
    --nodes-max 4 \
    --managed=false \
    --node-ami-family WindowsServer2019FullContainer
```

> **Note**
>
> - 노드가 클러스터에 조인하지 못한 경우 문제 해결 안내서의 노드가 클러스터 조인에 실패 부분을 참조하세요.
> - `eksctl` 명령에 사용 가능한 옵션을 보려면 다음 명령을 입력합니다.
>
>   ```
>   eksctl command -help
>   ```

예제 출력은 다음과 같습니다. 노드가 생성되는 동안 여러 줄이 출력됩니다. 출력의 마지막 줄 중 하나는 다음 예제 줄과 유사합니다.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (선택 사항) 샘플 애플리케이션을 배포하여 클러스터 및 Windows 노드를 테스트합니다.
4. 다음과 같은 조건에 해당하면 IMDS에 대한 Pod 액세스를 차단하는 것이 좋습니다.
   - Pods에 필요한 최소 권한만 있도록 모든 Kubernetes 서비스 계정에 IAM 역할을 할당할 계획입니다.
   - 클러스터의 어떤 Pods도 현재 AWS 리전 검색과 같은 다른 이유로 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 액세스할 필요가 없습니다.

   자세한 내용은 워커 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 섹션을 참조하세요.

AWS Management Console

필수 조건

- 기존 Amazon EKS 클러스터 및 Linux 노드 그룹. 이러한 리소스가 없는 경우 Amazon EKS 시작하기 설명서 중 하나에 따라 리소스를 생성하는 것이 좋습니다. 이 가이드에서는 Linux 노드로 Amazon EKS 클러스터를 생성하는 방법에 대해 설명합니다.

- Amazon EKS 클러스터의 요구 사항을 충족하는 기존 VPC 및 전용 보안 그룹. 자세한 내용은 Amazon EKS VPC 및 서브넷 요구 사항과 고려 사항 및 Amazon EKS 보안 그룹 요구 사항 및 고려 사항 섹션을 참조하세요. Amazon EKS 시작하기 설명서에서는 요구 사항을 충족하는 VPC를 생성합니다. 또는 Amazon EKS 클러스터에 대해 VPC 생성에 따라 수동으로 만들 수도 있습니다.
- Amazon EKS 클러스터의 요구 사항을 충족하는 VPC 및 보안 그룹을 사용하는 기존 Amazon EKS 클러스터. 자세한 내용은 Amazon EKS 클러스터 생성 단원을 참조하십시오. AWS Outposts, AWS Wavelength 또는 AWS Local Zones이 사용된 AWS 리전에 서브넷이 있는 경우 클러스터를 생성할 때 해당 서브넷이 전달되지 않은 상태여야 합니다.

1단계: AWS Management Console을 사용하여 자체 관리형 Windows 노드를 시작하기

1. 클러스터 상태가 ACTIVE가 되기를 기다립니다. 클러스터가 활성화되기 전에 노드를 시작하면 노드가 클러스터에 등록되지 않고 노드를 다시 시작해야 합니다.
2. https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
3. 스택 생성을 선택합니다.
4. [템플릿 지정(Specify template)]에서 [Amazon S3 URL]을 선택합니다.
5. 다음과 같은 URL을 복사하여 Amazon S3 URL에 붙여 넣습니다.

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2023-02-09/amazon-eks-windows-nodegroup.yaml
```

6. Next(다음)를 두 번 선택합니다
7. 빠른 스택 생성 페이지에서 다음 파라미터를 적절하게 입력합니다.
   - 스택 이름: AWS CloudFormation 스택에 대한 스택 이름을 선택합니다. 예를 들어 ***my-cluster*-nodes**라고 할 수 있습니다.
   - ClusterName: Amazon EKS 클러스터 생성 시 사용할 이름을 입력합니다.

     > **Important**
     >
     > 이 이름은 1단계: Amazon EKS 클러스터 생성에 사용한 이름과 정확히 일치해야 합니다. 그렇지 않으면 노드가 클러스터에 가입할 수 없습니다.

   - ClusterControlPlaneSecurityGroup: VPC를 만들 때 생성한 AWS CloudFormation 출력에서 보안 그룹을 선택합니다.

다음 단계에서는 해당 그룹을 검색하는 한 가지 방법을 보여줍니다.

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 클러스터의 이름을 선택합니다.
3. 네트워킹 탭을 선택합니다.
4. ClusterControlPlaneSecurityGroup 드롭다운 목록에서 선택할 때 추가 보안 그룹(Additional Security Group) 값을 참조로 사용하세요.

- NodeGroupName: 노드 그룹의 이름을 입력합니다. 이 이름은 나중에 노드에 대해 생성된 Auto Scaling 노드 그룹을 식별하는 데 사용할 수 있습니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다.
- NodeAutoScalingGroupMinSize: Auto Scaling 그룹이 축소할 수 있는 노드의 최소 노드 수를 입력합니다.
- NodeAutoScalingGroupDesiredCapacity: 스택을 생성할 때 조정할 원하는 노드 수를 입력합니다.
- NodeAutoScalingGroupMaxSize: Auto Scaling 그룹이 확장할 수 있는 노드의 최대 노드 수를 입력합니다.
- NodeInstanceType: 노드에 대한 인스턴스 유형을 선택합니다. 자세한 내용은 Amazon EC2 인스턴스 유형 선택 단원을 참조하십시오.

  > **Note**
  >
  > 최신 버전의 Amazon VPC CNI plugin for Kubernetes에 대해 지원되는 인스턴스 유형은 GitHub의 vpc_ip_resource_limit.go에 나열되어 있습니다. 지원되는 최신 인스턴스 유형을 사용하려면 CNI 버전을 업데이트해야 할 수 있습니다. 자세한 내용은 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 단원을 참조하십시오.

- NodeImageIdSSMParam: 현재 권장되는 Amazon EKS 최적화 Windows Core AMI ID의 Amazon EC2 Systems Manager 파라미터로 미리 채워집니다. 전체 버전의 Windows를 사용하려면 *Core*를 Full로 바꿉니다.
- NodeImageId: (선택 사항) Amazon EKS 최적화 AMI 대신 사용자 정의 AMI를 사용 중인 경우 해당 AWS 리전에 노드 AMI ID를 입력합니다. 이 필드의 값을 지정하면 NodeImageIdSSMParam 필드에 있는 모든 값이 재정의됩니다.
- NodeVolumeSize: 노드에 대해 루트 볼륨 크기를 GiB 단위로 지정합니다.

- KeyName: 시작 이후 SSH를 사용하여 노드에 연결하는 데 사용할 수 있는 Amazon EC2 SSH 키 페어 이름을 입력합니다. Amazon EC2 키 페어가 아직 없는 경우 AWS Management Console에서 새로 생성할 수 있습니다. 자세한 내용은 Windows 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하세요.

  > **Note**
  >
  > 여기에 키 페어를 입력하지 않으면 AWS CloudFormation 스택이 생성되지 않습니다.

- BootstrapArguments: `-KubeletExtraArgs`를 사용하여 별도의 `kubelet` 인수와 같이 노드 부트스트랩 스크립트에 전달할 선택적 인수를 지정합니다.
- DisableIMDSv1: 기본적으로 각 노드는 인스턴스 메타데이터 서비스 버전 1(IMDSv1) 및 IMDSv2를 지원합니다. IMDSv1을 사용 중지할 수 있습니다. 향후 노드 그룹의 노드와 Pods가 MDSv1을 사용하지 못하게 하려면 DisableIMDSv1을 true로 설정합니다. IMDS에 대한 자세한 내용은 인스턴스 메타데이터 서비스 구성을 참조하세요.
- VpcId: 생성한 VPC에 대한 ID를 선택합니다.
- NodeSecurityGroups: VPC를 생성할 때 Linux 노드 그룹에 대해 생성된 보안 그룹을 선택합니다. Linux 노드에 둘 이상의 보안 그룹이 연결되어 있는 경우 모든 보안 그룹을 지정합니다. 예를 들어 Linux 노드 그룹이 `eksctl`로 생성된 경우입니다.
- Subnets: 생성한 서브넷을 선택합니다. Amazon EKS 클러스터에 대해 VPC 생성의 단계를 사용하여 VPC를 생성한 경우, VPC 내에서 노드가 시작될 프라이빗 서브넷만 지정합니다.

  > **Important**
  >
  > - 서브넷 중 하나가 퍼블릭 서브넷인 경우 자동 퍼블릭 IP 주소 할당 설정을 사용하도록 설정해야 합니다. 퍼블릭 서브넷에 대해 이 설정을 사용하지 않으면 해당 퍼블릭 서브넷에 배포하는 노드에는 퍼블릭 IP 주소가 할당되지 않으며 클러스터 또는 기타 AWS 서비스와 통신할 수 없습니다. 서브넷이 2020년 3월 26일 이전에 Amazon EKS AWS CloudFormation VPC 템플릿 또는 `eksctl`을 사용하여 배포된 경우 퍼블릭 서브넷에 대해 자동 퍼블릭 IP 주소 할당이 사용 중지됩니다. 서브넷에 퍼블릭 IP 주소 할당을 활성화하는 방법에 대한 자세한 내용을 알아보려면 서브넷에 대한 퍼블릭 IPv4 주소 지정 속성 수정을 참조하세요. 노드가 프라이빗 서브넷에 배포되면 NAT 게이트웨이를 통해 클러스터 및 다른 AWS 서비스와 통신할 수 있습니다.

- 서브넷에 인터넷 액세스가 없는 경우 프라이빗 클러스터 요구 사항의 고려 사항 및 추가 단계를 알고 있는지 확인합니다.
- AWS Outposts, Wavelength 또는 Local Zones 서브넷을 선택하는 경우 클러스터를 생성할 때 해당 서브넷이 전달되지 않은 상태여야 합니다.

8. 스택에서 IAM 리소스를 생성할 수 있는지 확인한 다음 스택 생성(Create stack)을 선택합니다.
9. 스택이 생성된 후 콘솔에서 이를 선택하고 출력(Outputs)을 선택합니다.
10. 생성된 노드 그룹에 대해 NodeInstanceRole을 기록합니다. Amazon EKS Windows 노드를 구성할 때 필요합니다.

2단계: 노드가 클러스터에 조인하도록 하려면

1. `aws-auth ConfigMap`이 이미 있는지 확인합니다.

```
kubectl describe configmap -n kube-system aws-auth
```

2. `aws-auth ConfigMap`이 표시되면 필요에 따라 이를 업데이트합니다.

   a. 편집을 위해 `ConfigMap`을 엽니다.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. 필요에 따라 새 `mapRoles` 항목을 추가합니다. `rolearn` 값을 이전 절차에서 기록한 NodeInstanceRole 값으로 설정합니다.

```
[...]
data:
  mapRoles: |
- rolearn: <ARN of linux instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: <ARN of windows instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
        - eks:kube-proxy-windows
```

```
[...]
```

c. 파일을 저장하고 텍스트 편집기를 종료합니다.

3. "Error from server (NotFound): configmaps "aws-auth" not found라는 오류 메시지가 표시되면 스톡 ConfigMap을 적용합니다.

a. 구성 맵을 다운로드합니다.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/aws-auth-cm-windows.yaml
```

b. aws-auth-cm-windows.yaml 파일에서 rolearn 값을 이전 절차에서 기록한 해당 NodeInstanceRole 값으로 설정합니다. 이 작업은 텍스트 편집기를 사용하거나 *example values*를 바꾸고 다음 명령을 실행하여 수행할 수 있습니다.

```
sed -i.bak -e 's|<ARN of linux instance role (not instance profile)>|my-
node-linux-instance-role|' \
    -e 's|<ARN of windows instance role (not instance profile)>|my-node-
windows-instance-role|' aws-auth-cm-windows.yaml
```

> **Important**
>
> - 이 파일에서 어떠한 행도 수정하지 마세요.
> - Windows 및 Linux 노드 모두에 동일한 IAM 역할을 사용하지 마세요.

c. 구성을 적용합니다. 이 명령을 완료하는 데 몇 분이 걸릴 수 있습니다.

```
kubectl apply -f aws-auth-cm-windows.yaml
```

4. 노드의 상태를 확인하고 Ready 상태가 될 때까지 대기합니다.

```
kubectl get nodes --watch
```

Ctrl+C를 입력하여 쉘 프롬프트로 돌아갑니다.

> **Note**
>
> 권한 부여 또는 리소스 유형 오류가 표시되는 경우 문제 해결 주제의 권한이 없거나 액세스가 거부됨(kubectl) 섹션을 참조하세요.

노드가 클러스터에 조인하지 못한 경우 문제 해결 가이드의 노드가 클러스터 조인에 실패 섹션을 참조하세요.

3단계: 추가 작업

1. (선택 사항) 샘플 애플리케이션을 배포하여 클러스터 및 Windows 노드를 테스트합니다.
2. (선택 사항) AmazonEKS_CNI_Policy 관리형 IAM 정책(IPv4 클러스터가 있는 경우) 또는 *AmazonEKS_CNI_IPv6_Policy*(IPv6 클러스터가 있는 경우 직접 생성)가 the section called “노드 IAM 역할”에 연결된 경우 대신 Kubernetes aws-node 서비스 계정에 연결하는 IAM 역할에 할당하는 것이 좋습니다. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오.
3. 다음과 같은 조건에 해당하면 IMDS에 대한 Pod 액세스를 차단하는 것이 좋습니다.
   - Pods에 필요한 최소 권한만 있도록 모든 Kubernetes 서비스 계정에 IAM 역할을 할당할 계획입니다.
   - 클러스터의 어떤 Pods도 현재 AWS 리전 검색과 같은 다른 이유로 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 액세스할 필요가 없습니다.

   자세한 내용은 워커 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 섹션을 참조하세요.

자체 관리형 노드 업데이트

새로운 Amazon EKS 최적화 AMI가 릴리스되면 자체 관리형 노드 그룹의 노드를 새 AMI로 교체하는 것을 고려합니다. 마찬가지로, Amazon EKS 클러스터용 Kubernetes 버전을 업데이트한 경우 동일한 Kubernetes 버전이 있는 노드를 사용하도록 노드를 업데이트합니다.

> **Important**
>
> 이 주제에서는 자체 관리형 노드 그룹에 대한 노드 업데이트에 대해 설명합니다. 관리형 노드 그룹을 사용 중인 경우 관리형 노드 그룹 업데이트을 참조하십시오.

새 AMI를 사용하도록 클러스터의 자체 관리형 노드 그룹을 업데이트하는 기본 방법에는 두 가지가 있습니다.

새 노드 그룹으로 마이그레이션

새 노드 그룹을 생성하고 Pods를 해당 그룹으로 마이그레이션합니다. 기존 AWS CloudFormation 스택에서 단순히 AMI ID를 업데이트하는 것보다 새 노드 그룹으로 마이그레이션하는 것이 낫습니다. 이는 마이그레이션 프로세스가 이전 노드 그룹을 NoSchedule로 테인트하고 새 스택이 기존 Pod워크로드를 수락할 준비가 된 후 노드를 드레이닝하기 때문입니다.

기존 자체 관리형 노드 그룹 업데이트

새 AMI를 사용하도록 기존 노드 그룹의 AWS CloudFormation 스택을 업데이트합니다. 이 방법은 `eksctl`로 생성한 노드 그룹에는 사용할 수 없습니다.

새 노드 그룹으로 마이그레이션

이 주제에서는 새로운 노드 그룹을 생성하고, 기존 애플리케이션을 새 그룹으로 안정적으로 마이그레이션한 다음, 이전 노드 그룹을 클러스터에서 제거하는 방법을 설명합니다. `eksctl` 또는 AWS Management Console을 사용하여 새 노드 그룹으로 마이그레이션할 수 있습니다.

eksctl

`eksctl`을 사용하여 애플리케이션을 새 노드 그룹으로 마이그레이션하려면

마이그레이션에 eksctl을 사용하는 방법에 대한 자세한 내용은 `eksctl` 설명서의 비관리형 노드 그룹을 참조하세요.

이 절차에는 `eksctl` 버전 `0.175.0` 이상이 필요합니다. 버전은 다음 명령을 통해 확인할 수 있습니다.

```
eksctl version
```

`eksctl` 설치 또는 업데이트에 대한 지침은 `eksctl` 설명서에서 Installation을 참조하세요.

> **Note**
>
> 이 절차는 `eksctl`을 사용하여 생성한 클러스터 및 노드 그룹에 대해서만 사용할 수 있습니다.

1. *`my-cluster`*를 클러스터 이름으로 바꿔서 기존 노드 그룹의 이름을 검색합니다.

   ```
   eksctl get nodegroups --cluster=my-cluster
   ```

   예제 출력은 다음과 같습니다.

   ```
   CLUSTER      NODEGROUP          CREATED               MIN SIZE      MAX SIZE
    DESIRED CAPACITY     INSTANCE TYPE     IMAGE ID
   default      standard-nodes   2019-05-01T22:26:58Z  1             4             3
                       t3.medium         ami-05a71d034119ffc12
   ```

2. 다음 명령어로 `eksctl`을 사용하여 새 노드 그룹을 시작합니다. 명령에서 모든 *`example value`*를 고유한 값으로 바꿉니다. 버전 번호는 컨트롤 플레인의 Kubernetes 버전보다 이후일 수 없습니다. 또한 컨트롤 플레인의 Kubernetes 버전보다 이전인 마이너 버전이 2개를 초과할 수 없습니다. 컨트롤 플레인과 동일한 버전을 사용하는 것이 좋습니다.

   다음과 같은 조건에 해당하면 IMDS에 대한 Pod 액세스를 차단하는 것이 좋습니다.

   - Pods에 필요한 최소 권한만 있도록 모든 Kubernetes 서비스 계정에 IAM 역할을 할당할 계획입니다.
   - 클러스터의 어떤 Pods도 현재 AWS 리전 검색과 같은 다른 이유로 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 액세스할 필요가 없습니다.

   자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 부분을 참조하세요.

   Pod가 IMDS에 액세스하지 못하게 차단하려면 `--disable-pod-imds` 옵션을 다음 명령에 추가합니다.

   > **Note**
   >
   > 사용 가능한 플래그 및 설명에 대한 자세한 내용은 https://eksctl.io/를 참조하세요.

```
eksctl create nodegroup \
  --cluster my-cluster \
  --version 1.29 \
  --name standard-nodes-new \
  --node-type t3.medium \
  --nodes 3 \
  --nodes-min 1 \
  --nodes-max 4 \
  --managed=false
```

3. 이전 명령이 완료되면 다음 명령으로 모든 노드가 Ready 상태가 되었는지 확인합니다.

```
kubectl get nodes
```

4. 다음 명령을 사용하여 원래 노드 그룹을 삭제합니다. 명령에서 모든 *example value*를 클러스터 이름과 노드 그룹 이름으로 바꿉니다.

```
eksctl delete nodegroup --cluster my-cluster --name standard-nodes-old
```

AWS Management Console and AWS CLI

AWS Management Console 및 AWS CLI에서 애플리케이션을 새 노드 그룹으로 마이그레이션하려면

1. 자체 관리형 Amazon Linux 노드 시작하기에 설명된 단계에 따라 새 노드 그룹을 시작합니다.
2. 스택이 생성된 후 콘솔에서 이를 선택하고 출력(Outputs)을 선택합니다.
3. 생성된 노드 그룹에 대해 NodeInstanceRole을 기록합니다. 새 Amazon EKS 노드를 클러스터에 추가하려면 이 정보가 필요합니다.

> **Note**
>
> 추가 IAM 정책을 이전 노드 그룹의 IAM 역할에 연결한 경우 새 그룹에서 해당 기능을 유지하려면 동일한 정책을 새 노드 그룹의 IAM 역할에 연결해야 합니다. 이는 예를 들어 Kubernetes Cluster Autoscaler에 대한 권한을 추가한 경우에 적용됩니다.

4. 두 노드 그룹이 서로 통신할 수 있도록 두 노드 그룹의 보안 그룹을 업데이트합니다. 자세한 내용은 Amazon EKS 보안 그룹 요구 사항 및 고려 사항을 참조하세요.

a. 두 노드 그룹에 대한 보안 그룹 ID를 모두 기록합니다. 이는 AWS CloudFormation 스택 출력에 NodeSecurityGroup 값으로 표시됩니다.

   다음 AWS CLI 명령을 사용하여 스택 이름에서 보안 그룹 ID를 가져옵니다. 이러한 명령에서 `oldNodes`는 이전 노드 스택의 AWS CloudFormation 스택 이름이고 `newNodes`는 마이그레이션하는 대상 스택의 이름입니다. 모든 *`example value`*를 고유한 값으로 바꿉니다.

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name
 $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name
 $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
```

b. 각 노드 보안 그룹이 서로의 트래픽을 수락하도록 노드 보안 그룹에 인그레스 규칙을 추가합니다.

   다음 AWS CLI 명령은 상대 보안 그룹의 모든 프로토콜에서 모든 트래픽을 허용하는 인바운드 규칙을 각 보안 그룹에 추가합니다. 이렇게 구성하면 워크로드를 새 그룹으로 마이그레이션하는 동안 각 노드 그룹의 Pods가 서로 통신할 수 있습니다.

```
aws ec2 authorize-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 authorize-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

5. `aws-auth` configmap을 편집하여 RBAC에서 새 노드 인스턴스 역할을 매핑합니다.

```
kubectl edit configmap -n kube-system aws-auth
```

새 노드 그룹에 대한 새 mapRoles 항목을 추가합니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 arn:aws:를 arn:aws-us-gov:로 바꿉니다.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: ARN of instance role (not instance profile)
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-16-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

ARN of instance role (not instance profile) 코드 조각을 이전 단계에서 기록한 NodeInstanceRole 값으로 바꿉니다. 그런 다음 파일을 저장하고 닫아서 업데이트된 configmap을 적용합니다.

6. 노드의 상태를 보면서 새 노드가 클러스터에 조인하고 Ready 상태에 도달할 때까지 기다립니다.

```
kubectl get nodes --watch
```

7. (선택 사항) Kubernetes Cluster Autoscaler를 사용하는 경우 배포를 복제본 0개로 축소하여 조정 작업의 충돌을 방지합니다.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

8. 다음 명령을 사용하여 NoSchedule로 제거하려는 각 노드를 테인트합니다. 이는 바꾸고 있는 노드에서 새 Pods가 예약되거나 다시 예약되지 않도록 하기 위한 것입니다. 테인트와 허용 오차에 대한 자세한 내용을 알아보려면 Kubernetes설명서를 참조하세요.

```
kubectl taint nodes node_name key=value:NoSchedule
```

노드를 새 Kubernetes 버전으로 업그레이드하는 경우 다음 코드 조각을 사용하여 특정 Kubernetes 버전(이 경우 1.27)의 모든 노드를 식별하고 테인트할 수 있습니다. 버전 번호는 컨트롤 플레인의 Kubernetes 버전보다 이후일 수 없습니다. 또한 컨트롤 플레인의 Kubernetes 버전보다 이전인 마이너 버전이 2개를 초과할 수 없습니다. 컨트롤 플레인과 동일한 버전을 사용하는 것이 좋습니다.

```
K8S_VERSION=1.27
nodes=$(kubectl get nodes -o jsonpath="{.items[?
(@.status.nodeInfo.kubeletVersion==\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Tainting $node"
    kubectl taint nodes $node key=value:NoSchedule
done
```

9. 클러스터의 DNS 공급자를 결정합니다.

```
kubectl get deployments -l k8s-app=kube-dns -n kube-system
```

예제 출력은 다음과 같습니다. 이 클러스터는 DNS 확인에 CoreDNS를 사용하고 있지만, 클러스터는 그 대신 kube-dns를 반환할 수 있습니다.

```
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
coredns   1         1         1            1           31m
```

10. 현재 배포가 두 개 미만의 복제본을 실행하고 있는 경우, 배포를 복제본 두 개로 확장합니다. 이전 명령 출력에서 해당 값이 대신 반환된 경우 *coredns*을 **kubedns**로 교체합니다.

```
kubectl scale deployments/coredns --replicas=2 -n kube-system
```

11. 다음 명령을 사용하여 클러스터에서 제거할 각 노드를 드레이닝합니다.

```
kubectl drain node_name --ignore-daemonsets --delete-local-data
```

노드를 새 Kubernetes 버전으로 업그레이드하는 경우 다음 코드 조각을 사용하여 특정 Kubernetes 버전(이 경우 *1.27*)의 모든 노드를 식별하고 테인트할 수 있습니다.

```
K8S_VERSION=1.27
```

```
nodes=$(kubectl get nodes -o jsonpath="{.items[?
(@.status.nodeInfo.kubeletVersion==\"v$K8S_VERSION\")].metadata.name}")
for node in ${nodes[@]}
do
    echo "Draining $node"
    kubectl drain $node --ignore-daemonsets --delete-local-data
done
```

12. 이전 노드가 드레이닝을 완료한 후 앞서 권한을 부여한 보안 그룹 인바운드 규칙을 취소합니다. 그런 다음 AWS CloudFormation 스택을 삭제하여 인스턴스를 종료합니다.

> **Note**
>
> 추가 IAM 정책을 기존 노드 그룹 IAM 역할에 연결한 경우(예: Kubernetes Cluster Autoscaler에 대한 권한 추가) AWS CloudFormation 스택을 삭제하려면 먼저 해당 추가 정책을 역할에서 분리합니다.

a. 위에서 노드 보안 그룹에 대해 생성한 인바운드 규칙을 취소합니다. 이러한 명령에서 `oldNodes`는 이전 노드 스택의 AWS CloudFormation 스택 이름이고 `newNodes`는 마이그레이션하는 대상 스택의 이름입니다.

```
oldNodes="old_node_CFN_stack_name"
newNodes="new_node_CFN_stack_name"

oldSecGroup=$(aws cloudformation describe-stack-resources --stack-name
 $oldNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
newSecGroup=$(aws cloudformation describe-stack-resources --stack-name
 $newNodes \
--query 'StackResources[?
ResourceType==`AWS::EC2::SecurityGroup`].PhysicalResourceId' \
--output text)
aws ec2 revoke-security-group-ingress --group-id $oldSecGroup \
--source-group $newSecGroup --protocol -1
aws ec2 revoke-security-group-ingress --group-id $newSecGroup \
--source-group $oldSecGroup --protocol -1
```

b. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.

c. 이전 노드 스택을 선택합니다.

d. 삭제를 선택합니다.

e. 스택 삭제(Delete stack) 확인 대화 상자에서 스택 삭제(Delete stack)를 선택합니다.

13. aws-auth configmap을 편집하여 RBAC에서 이전 노드 인스턴스 역할을 제거합니다.

```
kubectl edit configmap -n kube-system aws-auth
```

이전 노드 그룹에 대한 mapRoles 항목을 삭제합니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 arn:aws:를 arn:aws-us-gov:로 바꿉니다.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-16-NodeInstanceRole-
W70725MZQFF8
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws:iam::111122223333:role/nodes-1-15-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes>
```

파일을 저장하고 닫아서 업데이트된 configmap을 적용합니다.

14. (선택 사항) Kubernetes Cluster Autoscaler를 사용하는 경우 배포를 다시 복제본 한 개로 축소합니다.

> **Note**
>
> 또한 새 Auto Scaling 그룹에 적절한 태그를 지정하고(예: k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/*my-cluster*) 태그가 새로 지정된 Auto Scaling 그룹을 가리키도록 Cluster Autoscaler 배포에 대한 명령을 업데이트해야 합니다. 자세한 내용은 AWS의 Cluster Autoscaler를 참조하십시오.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

15. (선택 사항) 최신 버전의 Kubernetes용 Amazon VPC CNI 플러그 인을 사용하고 있는지 확인합니다. 지원되는 최신 인스턴스 유형을 사용하려면 CNI 버전을 업데이트해야 할 수 있습니다. 자세한 내용은 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 단원을 참조하십시오.
16. 클러스터가 DNS 확인에 kube-dns를 사용하는 경우(이전 단계 참조) kube-dns 배포를 복제본 한 개로 축소합니다.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

기존 자체 관리형 노드 그룹 업데이트

이 주제에서는 기존 AWS CloudFormation 자체 관리형 노드 스택을 새 AMI로 업데이트하는 방법을 설명합니다. 이 절차를 사용하여 클러스터 업데이트 후 새 버전의 Kubernetes로 노드를 업데이트할 수 있습니다. 그렇지 않으면 기존 Kubernetes 버전에 대해 최신 Amazon EKS 최적화 AMI로 업데이트할 수 있습니다.

> **Important**
>
> 이 주제에서는 자체 관리형 노드 그룹에 대한 노드 업데이트에 대해 설명합니다. 관리형 노드 그룹 사용에 대한 자세한 내용은 관리형 노드 그룹 업데이트을 참조하십시오.

최신 기본 Amazon EKS 노드 AWS CloudFormation 템플릿은 예전 인스턴스를 제거하기 전에 한 번에 하나씩 새 AMI로 인스턴스를 시작하도록 구성되어 있습니다. 이렇게 구성하면 업데이트를 적용하는 동안 클러스터의 Auto Scaling 그룹에 항상 활성 인스턴스 수를 원하는 대로 항상 유지할 수 있습니다.

> **Note**
>
> 이 방법은 `eksctl`로 생성한 노드 그룹에는 사용할 수 없습니다. `eksctl`을 사용하여 클러스터 또는 노드 그룹을 생성한 경우, 새 노드 그룹으로 마이그레이션 부분을 참조하세요.

기존 노드 그룹을 업데이트하려면

1. 클러스터의 DNS 공급자를 결정합니다.

   ```
   kubectl get deployments -l k8s-app=kube-dns -n kube-system
   ```

   예제 출력은 다음과 같습니다. 이 클러스터는 DNS 확인에 CoreDNS를 사용하고 있지만, 클러스터는 그 대신 kube-dns를 반환할 수 있습니다. 출력은 사용 중인 버전에 따라 다르게 보일 수 있습니다.kubectl

   ```
   NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
   coredns   1         1         1            1           31m
   ```

2. 현재 배포가 두 개 미만의 복제본을 실행하고 있는 경우, 배포를 복제본 두 개로 확장합니다. 이전 명령 출력에서 해당 값이 대신 반환된 경우 *coredns*을 **kube-dns**로 교체합니다.

   ```
   kubectl scale deployments/coredns --replicas=2 -n kube-system
   ```

3. (선택 사항) Kubernetes Cluster Autoscaler를 사용하는 경우 배포를 복제본 0개로 축소하여 조정 작업의 충돌을 방지합니다.

   ```
   kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
   ```

4. 현재 노드 그룹의 인스턴스 유형과 원하는 인스턴스 수를 결정합니다. 나중에 그룹에 대한 AWS CloudFormation 템플릿을 업데이트할 때 이러한 값을 입력합니다.

   a. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.

   b. 왼쪽 탐색 창에서 시작 구성(Launch Configurations)을 선택하고 기존 노드 시작 구성에 대한 인스턴스 유형을 기록해 둡니다.

   c. 왼쪽 탐색 창에서 Auto Scaling 그룹(Auto Scaling Groups)을 선택하고 기존 노드 Auto Scaling 그룹에 대해 원하는(Desired) 인스턴스 수를 적어둡니다.

5. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.

6. 노드 그룹 스택을 선택한 다음 [업데이트(Update)]를 선택합니다.

7. Replace current template(현재 템플릿 대치)을 선택하고 Amazon S3 URL을 선택합니다.

8. Amazon S3 URL에서 다음 URL을 텍스트 영역에 붙여 넣어 최신 버전의 노드 AWS CloudFormation 템플릿을 사용하고 있는지 확인합니다. 그리고 다음(Next)을 선택합니다.

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

9. Specify stack details(스택 세부 정보 지정) 페이지에서 다음 파라미터를 입력하고 다음을 선택합니다.

   - NodeAutoScalingGroupDesiredCapacity - 이전 단계에서 기록한 원하는 인스턴스 수를 입력합니다. 또는 스택을 업데이트할 때 조정할 원하는 새 노드 수를 입력합니다.
   - NodeAutoScalingGroupMaxSize - Auto Scaling 그룹이 확장할 수 있는 노드의 최대 노드 수를 입력합니다. 이 값은 원하는 용량보다 하나 이상의 노드여야 합니다. 이는 업데이트 중에 노드 수를 줄이지 않고 노드의 롤링 업데이트를 수행할 수 있도록 하기 위한 것입니다.
   - NodeInstanceType - 이전 단계에서 기록한 인스턴스 유형을 선택합니다. 또는 노드에 대해 다른 인스턴스 유형을 선택합니다. 다른 인스턴스 유형을 선택하려면 Amazon EC2 인스턴스 유형 선택 부분을 검토하세요. 각 Amazon EC2 인스턴스 유형은 탄력적 네트워크 인터페이스(네트워크 인터페이스)의 최대 수를 지원하며 각 네트워크 인터페이스는 최대 IP 주소 수를 지원합니다. 각 워커 노드와 Pod에는 고유한 IP 주소가 할당되므로 각 Amazon EC2 노드에서 실행할 최대 Pods 수를 지원하는 인스턴스 유형을 선택하는 것이 중요합니다. 인스턴스 유형이 지원하는 네트워크 인터페이스 및 IP 주소의 수 목록은 인스턴스 유형별 네트워크 인터페이스당 IP 주소를 참조하세요. 예를 들어, `m5.large` 인스턴스 유형은 작업자 노드 및 Pods에 대해 최대 30개의 IP 주소를 지원합니다.

     > **Note**
     >
     > 최신 Amazon VPC CNI plugin for Kubernetes 버전에 대해 지원되는 인스턴스 유형은 GitHub의 vpc_ip_resource_limit.go에 표시되어 있습니다. 지원되는 최신 인스턴스 유형을 사용하려면 Amazon VPC CNI plugin for Kubernetes 버전을 업데이트하는 것이 좋습니다. 자세한 내용은 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 단원을 참조하십시오.

     > **Important**
     >
     > AWS 리전에 따라 일부 인스턴스 유형은 사용할 수 없습니다.

   - NodeImageIdSSMParam – 업데이트하려는 AMI ID의 Amazon EC2 Systems Manager 파라미터입니다. 다음 값은 Kubernetes 버전 `1.29`에 대해 최신 Amazon EKS 최적화 AMI를 사용합니다.

```
/aws/service/eks/optimized-ami/1.29/amazon-linux-2/recommended/image_id
```

*1.29*을 지원되는 동일한 Kubernetes 버전으로 바꿀 수 있습니다. 또는 제어 영역에서 실행 중인 Kubernetes 버전보다 최대 한 버전 이전이어야 합니다. 노드를 제어 영역과 동일한 버전으로 유지하는 것이 좋습니다. *amazon-linux-2*를 다른 AMI 유형으로 바꿀 수도 있습니다. 자세한 내용은 Amazon EKS 최적화 Amazon Linux AMI ID 검색 단원을 참조하십시오.

> **Note**
>
> Amazon EC2 Systems Manager 파라미터를 사용하면 AMI ID를 조회하고 지정하지 않아도 나중에 노드를 업데이트할 수 있습니다. AWS CloudFormation 스택에서 이 값을 사용 중이라면 모든 스택 업데이트에서는 지정된 Kubernetes 버전에 권장되는 최신 Amazon EKS 최적화 AMI를 항상 시작합니다. 템플릿에서 값을 변경하지 않은 경우에도 마찬가지입니다.

- NodeImageId – 고유한 사용자 지정 AMI를 사용하려면 사용할 AMI의 ID를 입력하십시오.

> **Important**
>
> 이 값은 NodeImageIdSSMParam에 지정된 값을 재정의합니다. NodeImageIdSSMParam 값을 사용하려면 NodeImageId의 값이 비어 있는지 확인하십시오.

- DisableIMDSv1 - 기본적으로 각 노드는 인스턴스 메타데이터 서비스 버전 1(IMDSv1) 및 IMDSv2를 지원합니다. 그러나 IMDSv1을 사용 중지할 수 있습니다. 노드 그룹에서 예약된 노드 또는 Pods가 IMDSv1을 사용하지 않도록 하려면 true를 선택합니다. IMDS에 대한 자세한 내용은 인스턴스 메타데이터 서비스 구성을 참조하세요. 서비스 계정에 대한 IAM 역할을 구현한 경우 AWS 서비스에 대한 액세스 권한이 필요한 모든 Pods에 직접 필요한 권한을 할당합니다. 이렇게 하면 현재 AWS 리전 검색 등의 다른 이유로 클러스터의 Pods가 IMDS에 액세스할 필요가 없습니다. 그런 다음 호스트 네트워킹을 사용하지 않는 Pods에 대해 IMDSv2 액세스를 사용 중지할 수도 있습니다. 자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 부분을 참조하세요.

10. (선택 사항) 옵션(Options) 페이지에서 스택 리소스에 태그를 지정합니다. 다음을 선택합니다.
11. [검토(Review)] 페이지에서 정보를 검토하고, 스택이 IAM 리소스를 생성할 수 있는지 확인한 다음, [스택 업데이트(Update stack)]를 선택합니다.

> **Note**
>
> 클러스터의 각 노드를 업데이트하는 데 몇 분 정도 걸립니다. 모든 노드의 업데이트가 완료될 때까지 기다린 후 다음 단계를 수행합니다.

12. 클러스터의 DNS 공급자가 kube-dns인 경우 kube-dns 배포를 복제본 한 개로 축소합니다.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

13. (선택 사항) Kubernetes Cluster Autoscaler를 사용하는 경우 배포를 다시 원하는 양의 복제본으로 조정합니다.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

14. (선택 사항) 최신 Amazon VPC CNI plugin for Kubernetes 버전을 사용하고 있는지 확인합니다. 지원되는 최신 인스턴스 유형을 사용하려면 Amazon VPC CNI plugin for Kubernetes 버전을 업데이트하는 것이 좋습니다. 자세한 내용은 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 단원을 참조하십시오.

AWS Fargate

> **Important**
>
> Amazon EKS가 있는 AWS Fargate은 AWS GovCloud(미국 동부) 및 AWSGovCloud(미국 서부)를 제외한 모든 Amazon EKS 지역에서 사용할 수 있습니다.

이 주제에서는 AWS Fargate에서 Amazon EKS를 사용하여 Kubernetes Pods를 실행하는 방법을 설명합니다. Fargate는 컨테이너에 대한 적정 규모의 온디맨드 컴퓨팅 용량을 제공하는 기술입니다. Fargate를 사용하면 컨테이너를 실행하려면 직접 가상 머신 그룹을 프로비저닝, 구성 또는 크기를 조정할 필요가 없습니다. 따라서 서버 유형을 선택하거나, 노드 그룹을 조정할 시점을 결정하거나, 클러스터 패킹을 최적화할 필요가 없습니다.

Fargate에서 시작하는 Pods와 Fargate 프로필에서 실행되는 방법을 제어할 수 있습니다. Fargate 프로필은 Amazon EKS 클러스터의 일부로 정의됩니다. Amazon EKS에서는 Kubernetes에서 제공한 확장 가능한 업스트림 모델을 사용하여 AWS에서 빌드한 컨트롤러를 사용하여 Kubernetes를 Fargate와 통합합니다. 이러한 컨트롤러는 Amazon EKS 관리형 Kubernetes 컨트롤 플레인의 일부로 실행되며

Fargate에 기본 Kubernetes Pods를 예약하는 것을 담당합니다. Fargate 컨트롤러에는 여러 개의 변형 및 검증 승인 컨트롤러 외에도 기본 Kubernetes 스케줄러와 함께 실행되는 새 스케줄러가 포함되어 있습니다. Fargate에서 실행하기 위한 조건을 충족하는 Pod를 시작하면 클러스터에서 실행되는 Fargate 컨트롤러가 해당 Pod를 인식하고 업데이트하고 Fargate에 예약합니다.

이 주제에서는 Fargate에서 실행되는 Pods의 여러 구성 요소에 대해 설명하고 Amazon EKS에서 Fargate를 사용하기 위한 특별한 고려 사항을 살펴봅니다.

AWS Fargate 고려 사항

Amazon EKS에서 Fargate를 사용할 때 고려해야 할 몇 가지 사항이 있습니다.

- Fargate에서 실행되는 각 Pod에는 자체 격리 경계가 있습니다. 기본 커널, CPU 리소스, 메모리 리소스 또는 Elastic 네트워크 인터페이스를 다른 Pod와 공유하지 않습니다.
- Network Load Balancer 및 Application Load Balancer(ALB)는 IP 대상을 통해서만 Fargate에서 사용할 수 있습니다. 자세한 내용을 알아보려면 Network Load Balancer 생성 및 Amazon EKS 애플리케이션 로드 밸런싱 섹션을 참조하세요.
- Fargate 노출 서비스는 노드 IP 모드가 아닌 대상 유형 IP 모드에서만 실행됩니다. 관리형 노드에서 실행 중인 서비스와 Fargate에서 실행 중인 서비스의 연결을 확인하는 권장 방법은 서비스 이름을 통해 연결하는 것입니다.
- 포드는 Fargate에서 실행되도록 예약된 시간에 Fargate 프로필과 일치해야 합니다. Fargate 프로필과 일치하지 않는 포드는 `Pending`과 같이 중단될 수 있습니다. 일치하는 Fargate 프로필이 있는 경우 사용자가 생성한 보류 중인 Pods를 삭제하여 Fargate에 다시 예약할 수 있습니다.
- DaemonSets는 Fargate에서 지원되지 않습니다. 애플리케이션에 대몬이 필요한 경우 Pods에서 사이드카 컨테이너로 실행되도록 해당 대몬을 다시 구성합니다.
- 권한 있는 컨테이너(Privileged container)는 Fargate에서 지원되지 않습니다.
- Fargate에서 실행되는 포드는 Pod 매니페스트에서 `HostPort` 또는 `HostNetwork`를 지정할 수 없습니다.
- Fargate Pods의 경우 기본 `nofile` 및 `nproc` 소프트 제한은 1,024이고 하드 제한은 65,535입니다.
- GPU는 현재 Fargate에서 사용할 수 없습니다.
- Fargate에서 실행되는 포드는 인터넷 게이트웨이에 대한 직접 경로가 없고 AWS 서비스에 대한 NAT 게이트웨이 액세스 권한이 있는 프라이빗 서브넷에서만 지원되므로 클러스터의 VPC에 프라이빗 서브넷이 있어야 합니다. 아웃바운드 인터넷 액세스가 없는 클러스터의 경우 프라이빗 클러스터 요구 사항을 참조하십시오.
- Vertical Pod Autoscaler를 사용하여 Fargate Pods의 올바른 초기 CPU 및 메모리 크기를 설정한 다음 Horizontal Pod Autoscaler를 사용하여 해당 Pods를 조정할 수 있습니다. Vertical Pod Autoscaler

가 더 큰 CPU 및 메모리 조합으로 Pods를 Fargate에 자동으로 다시 배포하도록 하려면 Vertical Pod Autoscaler의 모드를 `Auto` 또는 `Recreate`로 설정하여 올바로 작동하게 해야 합니다. 자세한 내용은 GitHub에서 Vertical Pod Autoscaler 설명서를 참조하세요.

- VPC에 대해 DNS 확인 및 DNS 호스트 이름을 활성화해야 합니다. 자세한 내용은 VPC에 대한 DNS 지원 보기 및 업데이트를 참조하십시오.
- Amazon EKS Fargate는 가상 머신(VM) 내에서 각 포드를 격리하여 Kubernetes 애플리케이션에 대한 심층 방어 기능을 추가합니다. 이 VM 경계를 통해 컨테이너 이스케이프 시 다른 포드에서 사용하는 호스트 기반 리소스에 대한 액세스를 방지하는데, 이 방법은 컨테이너화된 애플리케이션을 공격하고 컨테이너 외부의 리소스에 대한 액세스 권한을 얻는 일반적인 방법입니다.

  Amazon EKS를 사용해도 공동 책임 모델에서 책임은 변경되지 않습니다. 클러스터 보안 및 거버넌스 제어의 구성을 신중하게 고려해야 합니다. 애플리케이션을 격리하는 가장 안전한 방법은 항상 별도의 클러스터에서 실행하는 것입니다.
- Fargate 프로필은 VPC 보조 CIDR 블록의 서브넷 지정을 지원합니다. 보조 CIDR 블록을 지정할 수도 있습니다. 서브넷에서 사용 가능한 IP 주소의 수는 제한되어 있기 때문입니다. 따라서 클러스터에서 생성할 수 있는 Pods의 수가 제한되어 있습니다. Pods에 다른 서브넷을 사용하여 사용 가능한 IP 주소의 수를 늘릴 수 있습니다. 자세한 내용을 알아보려면 VPC에 `IPv4` CIDR 블록 추가를 참조하세요.
- Amazon EC2 인스턴스 메타데이터 서비스(IMDS)는 Fargate 노드에 배포된 Pods에는 사용할 수 없습니다. IAM 자격 증명이 필요한 Pods를 Fargate에 배포한 경우 서비스 계정에 대한 IAM 역할을 사용하여 Pods에 할당합니다. Pods가 IMDS를 통해 제공되는 다른 정보에 액세스해야 할 경우 이 정보를 Pod 사양에 하드 코딩해야 합니다. 여기에는 Pod가 배포된 AWS 리전 또는 가용 영역이 포함됩니다.
- Fargate Pods은(는) AWS Outposts, AWS Wavelength 또는 AWS 로컬 영역에 배포할 수 없습니다.
- Amazon EKS는 정기적으로 Fargate Pods를 패치하여 보안을 유지해야 합니다. 영향을 줄이는 방식으로 업데이트를 시도하지만 Pods가 성공적으로 제거되지 않으면 삭제해야 하는 경우가 있습니다. 중단을 최소화하기 위해 취할 수 있는 몇 가지 조치가 있습니다. 자세한 내용은 Fargate OS 패치 단원을 참조하십시오.
- Amazon EKS용 Amazon VPC CNI 플러그인은 Fargate 노드에 설치됩니다. Fargate 노드에서는 호환 가능한 대체 CNI 플러그 인을 사용할 수 없습니다.
- Fargate에서 실행되는 Pod은(는) Amazon EFS 파일 시스템을 자동으로 탑재합니다. Fargate 노드에는 동적 영구 볼륨 프로비저닝을 사용할 수 없지만 고정적인 프로비저닝은 사용할 수 있습니다.
- Amazon EBS 볼륨을 Fargate Pods에 탑재할 수 없습니다.
- Amazon EBS CSI 컨트롤러는 Fargate 노드에서 실행할 수 있지만 Amazon EBS CSI 노드 DaemonSet은(는) Amazon EC2 인스턴스에서만 실행할 수 있습니다.

- Kubernetes Job이 `Completed` 또는 `Failed`로 표시된 후에도 Job이 생성하는 Pods는 정상적으로 계속 존재합니다. 이 동작을 통해 로그와 결과를 볼 수 있지만 Fargate를 사용하는 경우 나중에 Job을 정리하지 않으면 비용이 발생합니다.

  Job이 완료되거나 실패한 후 관련 Pods를 자동으로 삭제하려면 TTL(Time-to-Live) 컨트롤러를 사용하여 기간을 지정할 수 있습니다. 다음 예제에서는 Job 매니페스트에 `.spec.ttlSecondsAfterFinished`를 지정하는 방법을 보여줍니다.

```
apiVersion: batch/v1
kind: Job
metadata:
  name: busybox
spec:
  template:
    spec:
      containers:
      - name: busybox
        image: busybox
        command: ["/bin/sh", "-c", "sleep 10"]
      restartPolicy: Never
  ttlSecondsAfterFinished: 60 # <-- TTL controller
```

Amazon EKS를 사용하여 AWS Fargate 시작

> **Important**
>
> Amazon EKS가 있는 AWS Fargate은 AWS GovCloud(미국 동부) 및 AWSGovCloud(미국 서부)를 제외한 모든 Amazon EKS 지역에서 사용할 수 있습니다.

이 주제에서는 Amazon EKS 클러스터를 사용하여 Pods에서 AWS Fargate 실행을 시작하는 방법을 설명합니다.

CIDR 블록을 사용하여 클러스터의 퍼블릭 엔드포인트에 대한 액세스를 제한하는 경우 프라이빗 엔드포인트 액세스도 사용 설정하는 것이 좋습니다. 이렇게 하면 Fargate Pods가 클러스터와 통신할 수 있습니다. 프라이빗 엔드포인트를 활성화하지 않은 경우, 퍼블릭 액세스에 대해 지정하는 CIDR 블록에는 VPC의 아웃바운드 소스가 포함되어야 합니다. 자세한 내용은 Amazon EKS 클러스터 엔드포인트 액세스 제어 단원을 참조하십시오.

전제 조건

기존 클러스터가 있어야 합니다. Amazon EKS 클러스터가 아직 없는 경우 Amazon EKS 시작하기 부분을 참조하세요.

기존 노드가 Fargate Pods와 통신할 수 있는지 확인

노드가 없는 새 클러스터 또는 관리형 노드 그룹만 있는 클러스터로 작업하는 경우 Fargate Pod 실행 역할 생성로 건너뛸 수 있습니다.

이미 연결된 노드가 있는 기존 클러스터로 작업하고 있다고 가정합니다. 이러한 노드의 Pods가 Fargate에서 실행 중인 Pods와 자유롭게 통신할 수 있는지 확인합니다. Fargate에서 실행되는 Pods는 연결된 클러스터에 대해 클러스터 보안 그룹을 사용하도록 자동으로 구성됩니다. 클러스터의 기존 노드가 클러스터 보안 그룹과 트래픽을 송수신할 수 있는지 확인해야 합니다. 관리형 노드 그룹은 클러스터 보안 그룹도 사용하도록 자동으로 구성되므로 이 호환성에 대해 수정하거나 확인할 필요가 없습니다.

`eksctl` 또는 Amazon EKS 관리형 AWS CloudFormation 템플릿을 사용하여 생성한 기존 노드 그룹의 경우 클러스터 보안 그룹을 노드에 수동으로 추가할 수 있습니다. 또는 노드 그룹에 대한 Auto Scaling 그룹 시작 템플릿을 수정하여 클러스터 보안 그룹을 인스턴스에 연결할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 인스턴스의 보안 그룹 변경 부분을 참조하세요.

클러스터에 대한 네트워킹(Networking) 섹션의 AWS Management Console에서 해당 클러스터에 대한 보안 그룹을 확인할 수 있습니다. 또는 다음 AWS CLI 명령을 사용하여 확인할 수 있습니다. 이 명령을 사용하는 경우 *my-cluster*를 클러스터의 이름으로 바꿉니다.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

Fargate Pod 실행 역할 생성

클러스터가 AWS Fargate에서 Pods를 생성하는 경우, Fargate 인프라에서 실행되는 구성 요소는 사용자를 대신하여 AWS API를 호출해야 합니다. Amazon EKS Pod 실행 역할은 이 작업을 수행할 수 있는 IAM 권한을 제공합니다. AWS Fargate Pod 실행 역할을 만들려면 Amazon EKS Pod 실행 IAM 역할 부분을 참조하세요.

> **Note**
>
> `--fargate` 옵션을 사용하여 `eksctl`로 클러스터를 생성한 경우 클러스터에는 이미 패턴 `eksctl-`*my-cluster*`-FargatePodExecutionRole-`*ABCDEFGHIJKL*에서 IAM 콘솔을 찾

을 수 있는 Pod 실행 역할이 있습니다. 마찬가지로 eksctl을 사용하여 Fargate 프로필을 생성하는 경우 Pod 실행 역할이 생성되지 않았으면 eksctl을 사용하여 포드 실행 역할을 생성합니다.

클러스터에 대한 Fargate 프로필 생성

클러스터의 Fargate에서 실행되는 Pods를 예약하려면 먼저 포드가 시작될 때 Fargate를 사용할 Pods를 지정하는 Fargate 프로파일을 정의해야 합니다. 자세한 내용은 AWS Fargate 프로파일 단원을 참조하십시오.

> **Note**
>
> `--fargate` 옵션을 사용하여 eksctl로 클러스터를 생성한 경우 `kube-system` 및 `default` 네임스페이스의 모든 Pods에 대한 선택기를 사용하여 클러스터에 대한 Fargate 프로파일이 이미 생성되어 있습니다. Fargate에서 사용할 다른 네임스페이스에 대한 Fargate 프로필을 생성하려면 다음 절차를 따르세요.

eksctl 또는 AWS Management Console을 사용하여 Fargate 프로필을 생성할 수 있습니다.

eksctl

이 절차에는 eksctl 버전 `0.175.0` 이상이 필요합니다. 버전은 다음 명령을 통해 확인할 수 있습니다.

```
eksctl version
```

eksctl 설치 또는 업데이트에 대한 지침은 eksctl 설명서의 Installation을 참조하세요.

eksctl를 사용하여 Fargate 프로파일을 생성하려면

다음 eksctl 명령으로 Fargate 프로파일을 생성하고 모든 *example value*를 고유한 값으로 바꿉니다. 네임스페이스를 지정해야 합니다. 그러나 `--labels` 옵션은 필요하지 않습니다.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

my-kubernetes-namespace 및 *key=value* 레이블에 특정 와일드카드를 사용할 수 있습니다. 자세한 내용은 Fargate 프로파일 와일드카드 단원을 참조하십시오.

AWS Management Console

AWS Management Console에서 클러스터용 Fargate 프로파일을 생성하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. Fargate 프로파일을 생성할 클러스터를 선택합니다.
3. 컴퓨팅(Compute) 탭을 선택합니다.
4. [Fargate 프로파일(Fargate profiles)]에서 [Fargate 프로파일 추가(Add Fargate profile)]를 선택합니다.
5. Fargate 프로파일 구성(Configure Fargate profile) 페이지에서 다음을 수행합니다.
   a. Name(이름)에 Fargate 프로파일의 이름을 입력합니다. 이름은 고유해야 합니다.
   b. 포드 실행 역할(Pod execution role)에서 Fargate 프로파일과 함께 사용할 Pod 실행 역할을 선택합니다. `eks-fargate-pods.amazonaws.com` 서비스 보안 주체가 있는 IAM 역할만 표시됩니다. 나열된 역할이 표시되지 않으면 역할을 만들어야 합니다. 자세한 내용은 Amazon EKS Pod 실행 IAM 역할 단원을 참조하십시오.
   c. 선택한 서브넷을 필요에 따라 수정합니다.

      > **Note**
      > Fargate에서 실행되는 Pods에는 프라이빗 서브넷만 지원됩니다.

   d. [태그(Tags)]에 선택적으로 Fargate 프로파일에 태그를 지정할 수 있습니다. 이러한 태그는 프로파일과 연결된 다른 리소스(예: Pods)에 전파되지 않습니다.
   e. 다음을 선택합니다.
6. Pod 선택 구성(Configure pod selection) 페이지에서 다음을 수행합니다.
   a. Namespace(네임스페이스)에 Pods에 대해 일치시킬 네임스페이스를 입력합니다.
      - **`kube-system`** 또는 **`default`**와 같은 특정 네임스페이스를 사용하여 일치시킬 수 있습니다.
      - 특정 와일드카드(예: **`prod-*`**)를 사용하여 여러 네임스페이스(예: `prod-deployment` 및 `prod-test`)를 일치시킬 수 있습니다. 자세한 내용은 Fargate 프로파일 와일드카드 단원을 참조하십시오.

b. (선택 사항) 셀렉터에 Kubernetes 레이블을 추가합니다. 특히 지정된 네임스페이스의 Pods가 일치해야 하는 포드에 추가합니다.

- `infrastructure: fargate` Kubernetes 레이블도 있는 지정된 네임스페이스의 Pods만 선택기와 일치하도록 **`infrastructure: fargate`** 레이블을 선택기에 추가할 수 있습니다.
- 특정 와일드카드(예: **`key?: value?`**)를 사용하여 여러 네임스페이스(예: `keya: valuea` 및 `keyb: valueb`)를 일치시킬 수 있습니다. 자세한 내용은 Fargate 프로파일 와일드카드 단원을 참조하십시오.

c. Next(다음)를 선택합니다.

7. 검토 및 생성(Review and create) 페이지에서 Fargate 프로파일에 대한 정보를 검토하고 생성(Create)을 선택합니다.

업데이트: CoreDNS

기본적으로 CoreDNS는 Amazon EKS 클러스터의 Amazon EC2 인프라에서 실행되도록 구성됩니다. 클러스터의 Fargate에서만 Pods를 실행하려면 다음 단계를 수행하세요.

> **Note**
>
> `--fargate` 옵션을 사용하여 `eksctl`로 클러스터를 생성한 경우 다음 단계로 건너뛸 수 있습니다.

1. 다음 명령을 사용하여 CoreDNS에 대한 각 Fargate 프로필을 삭제합니다. *`my-cluster`*를 클러스터 이름으로, *`111122223333`*을 계정 ID로, *`AmazonEKSFargatePodExecutionRole`*을 Pod 실행 역할 이름으로, *`0000000000000001`*, *`0000000000000002`*, *`0000000000000003`*을 프라이빗 서브넷의 ID로 바꿉니다. Pod 실행 역할이 없는 경우 먼저 하나를 생성해야 합니다.

> **Important**
>
> 역할 ARN에 / 이외의 경로를 포함할 수 없습니다. 예를 들어, 역할 이름이 `development/apps/my-role`인 경우 역할에 대한 ARN을 지정할 때 역할 이름을 `my-role`로 변경해야 합니다. 역할 ARN의 형식은 `arn:aws:iam::`*`111122223333`*`:role/`*`role-name`*이어야 합니다.

```
aws eks create-fargate-profile \
    --fargate-profile-name coredns \
    --cluster-name my-cluster \
    --pod-execution-role-arn
 arn:aws:iam::111122223333:role/AmazonEKSFargatePodExecutionRole \
    --selectors namespace=kube-system,labels={k8s-app=kube-dns} \
    --subnets subnet-0000000000000001 subnet-0000000000000002
 subnet-0000000000000003
```

2. 다음 명령을 실행하여 CoreDNS Pods에서 `eks.amazonaws.com/compute-type : ec2` 주석을 제거합니다.

```
kubectl patch deployment coredns \
    -n kube-system \
    --type json \
    -p='[{"op": "remove", "path": "/spec/template/metadata/annotations/
eks.amazonaws.com~1compute-type"}]'
```

다음 단계

- 다음 워크플로를 사용하여 Fargate에서 실행할 기존 애플리케이션 마이그레이션을 시작할 수 있습니다.
  1. 애플리케이션의 Kubernetes 네임스페이스 및 Kubernetes 레이블과 일치하는 Fargate 프로필을 생성합니다.
  2. Fargate에 예약되도록 기존 Pods를 삭제하고 다시 생성합니다. 예를 들어 다음 명령은 `coredns` 배포의 롤아웃을 트리거합니다. 네임스페이스 및 배포 유형을 수정하여 특정 Pods를 업데이트할 수 있습니다.

```
kubectl rollout restart -n kube-system deployment coredns
```

- Amazon EKS 애플리케이션 로드 밸런싱를 배포하여 Fargate에서 실행되는 Pods에 대해 인그레스 대상을 허용합니다.
- Vertical Pod Autoscaler를 사용하여 Fargate Pods의 올바른 초기 CPU 및 메모리 크기를 설정한 다음 Horizontal Pod Autoscaler를 사용하여 해당 Pods를 조정할 수 있습니다. Vertical Pod Autoscaler가 더 높은 CPU 및 메모리 조합으로 Pods를 Fargate에 자동으로 다시 배포하도록 하려면 Vertical

Pod Autoscaler의 모드를 `Auto` 또는 `Recreate`로 설정합니다. 이를 통해 기능을 올바르게 사용할 수 있습니다. 자세한 내용은 GitHub에서 Vertical Pod Autoscaler 설명서를 참조하세요.

- 이러한 지침을 따라 애플리케이션을 모니터링하도록 AWS Distro for OpenTelemetry(ADOT) Collector를 설정할 수 있습니다.

AWS Fargate 프로파일

> **Important**
>
> Amazon EKS가 있는 AWS Fargate은 AWS GovCloud(미국 동부) 및 AWSGovCloud(미국 서부)를 제외한 모든 Amazon EKS 지역에서 사용할 수 있습니다.

클러스터의 Fargate에 Pods를 예약하려면 먼저 포드가 시작될 때 Fargate를 사용할 Pods를 지정하는 Fargate 프로파일을 하나 이상 정의해야 합니다.

관리자는 Fargate 프로파일을 사용하여 Fargate에서 실행되는 Pods를 선언할 수 있습니다. 프로파일의 셀렉터를 통해 이 작업을 수행할 수 있습니다. 각 프로파일에 최대 5개의 셀렉터를 추가할 수 있습니다. 각 셀렉터에 네임스페이스를 포함해야 합니다. 레이블도 셀렉터에 포함될 수 있습니다. 레이블 필드는 여러 개의 선택적 키-값 페어로 구성됩니다. 셀렉터와 일치하는 포드는 Fargate에 예약되어 있습니다. 포드는 선택기에 지정된 네임스페이스와 레이블을 사용하여 일치됩니다. 네임스페이스 셀렉터가 레이블 없이 정의되면 Amazon EKS는 프로파일을 사용하여 해당 네임스페이스에서 실행되는 모든 Pods를 Fargate에 예약하려고 시도합니다. 예약할 Pod가 Fargate 프로파일의 셀렉터와 일치하는 경우 해당 Pod가 Fargate에 예약됩니다.

Pod가 여러 Fargate 프로파일과 일치하는 경우 Pod 사양에 다음 Kubernetes 레이블 `eks.amazonaws.com/fargate-profile: my-fargate-profile`을 추가하여 Pod가 사용하는 프로파일을 지정할 수 있습니다. Pod는 Fargate에 예약할 해당 프로파일의 셀렉터와 일치해야 합니다. Kubernetes 선호도/반선호도 규칙은 Amazon EKS Fargate Pods에 적용되지 않으며 필요하지 않습니다.

Fargate 프로파일을 생성할 때 Pod 실행 역할을 지정해야 합니다. 이 실행 역할은 프로파일을 사용하여 Fargate 인프라에서 실행되는 Amazon EKS 구성 요소를 위한 것입니다. 이 역할은 권한 부여를 위해 클러스터의 Kubernetes 역할 기반 액세스 컨트롤(RBAC)에 추가됩니다. 이렇게 하면 Fargate 인프라에서 실행되는 `kubelet`이 Amazon EKS 클러스터에 등록되어 클러스터에 노드로 표시될 수 있습니다. 또한 Pod 실행 역할은 Amazon ECR 이미지 리포지토리에 대한 읽기 액세스를 허용하는 Fargate 인프라에 대한 IAM 권한을 제공합니다. 자세한 내용은 Amazon EKS Pod 실행 IAM 역할 단원을 참조하십시오.

Fargate 프로파일은 변경할 수 없습니다. 그러나 업데이트된 프로파일을 새로 생성하여 기존 프로파일을 바꾼 다음 원본을 삭제할 수 있습니다.

> **Note**
>
> Fargate 프로파일을 사용하여 실행 중인 모든 Pods는 중지되고 프로파일이 삭제되면 보류 상태로 전환됩니다.

클러스터의 Fargate 프로파일이 `DELETING` 상태인 경우 해당 클러스터에 다른 프로파일을 생성하려면 Fargate 프로파일이 삭제될 때까지 기다려야 합니다.

Amazon EKS 및 Fargate는 Fargate 프로파일에 정의된 각 서브넷에 Pods를 분산시킵니다. 그러나 고르게 분산되지 않을 수 있습니다. 고른 분산이 필요한 경우 Fargate 프로파일을 2개 사용합니다. 복제본을 2개 배포하려고 하고 가동 중지를 원하지 않는 경우 고른 분산이 중요합니다. 각 프로파일에는 서브넷이 하나만 있는 것이 좋습니다.

Fargate 프로파일 구성 요소

다음 구성 요소는 Fargate 프로파일에 포함되어 있습니다.

포드 실행 역할

클러스터가 AWS Fargate에 Pods를 생성하는 경우 Fargate 인프라에서 실행 중인 `kubelet`이 사용자를 대신하여 AWS API를 호출해야 합니다. 예를 들어, Amazon ECR에서 컨테이너 이미지를 가져오기 위해 호출해야 합니다. Amazon EKS Pod 실행 역할은 이 작업을 수행할 수 있는 IAM 권한을 제공합니다.

Fargate 프로파일을 생성할 때 Pod와 함께 사용할 Pods 실행 역할을 지정해야 합니다. 이 역할은 권한 부여를 위해 클러스터의 Kubernetes 역할 기반 액세스 컨트롤(RBAC)에 추가됩니다. 이렇게 하면 Fargate 인프라에서 실행 중인 `kubelet`이 Amazon EKS 클러스터에 등록되어 클러스터에 노드로 표시될 수 있습니다. 자세한 내용은 Amazon EKS Pod 실행 IAM 역할 단원을 참조하십시오.

서브넷

이 프로파일을 사용하는 Pods를 시작할 서브넷의 ID입니다. 현재 Fargate에서 실행 중인 Pods에는 퍼블릭 IP 주소가 할당되지 않습니다. 따라서 이 파라미터에 대해서는 인터넷 게이트웨이로 가는 직접 경로가 없는 프라이빗 서브넷만 허용됩니다.

선택기

이 Fargate 프로파일을 사용하기 위해 Pods에 대해 일치시킬 선택기입니다. Fargate 프로파일에 선택기를 최대 5개까지 지정할 수 있습니다. 셀렉터에는 다음 구성 요소가 있습니다.

- 네임스페이스 - 셀렉터에 대한 네임스페이스를 지정해야 합니다. 셀렉터는 이 네임스페이스에서 생성된 Pods만 일치시킵니다. 그러나 여러 셀렉터를 생성하여 여러 네임스페이스를 대상으로 지정할 수 있습니다.
- 레이블 - 선택적으로 셀렉터에 대해 일치시킬 Kubernetes 레이블을 지정할 수 있습니다. 셀렉터는 셀렉터에 지정된 모든 레이블이 있는 Pods와 일치합니다.

Fargate 프로파일 와일드카드

Kubernetes에서 허용하는 문자 외에도 네임스페이스, 레이블 키 및 레이블 값에 대한 선택기 기준에 *****와 **?**를 사용할 수 있습니다.

- *****는 없음, 하나 또는 여러 문자를 나타냅니다. 예를 들어, **prod***는 `prod`와 `prod-metrics`를 나타낼 수 있습니다.
- **?**는 단일 문자를 나타냅니다. 예를 들어, **value?**는 `valuea`를 나타낼 수 있습니다. 그러나 ?는 정확히 하나의 문자만 나타낼 수 있기 때문에 `value`와 `value-a`를 나타낼 수는 없습니다.

이러한 와일드카드 문자는 모든 위치에서 조합하여 사용할 수 있습니다(예: **prod***, ***dev** 및 **frontend*?**). 다른 와일드카드와 패턴 일치 형식(정규식 등)은 지원되지 않습니다.

Pod 사양의 네임스페이스 및 레이블에 대해 일치하는 프로파일이 여러 개 있는 경우 Fargate는 프로파일 이름별 영숫자 정렬을 기준으로 프로파일을 선택합니다. 예를 들어, 프로파일 A(이름 `beta-workload`)와 프로파일 B(이름 `prod-workload`)에 시작할 Pods에 대해 일치하는 선택기가 있는 경우 Fargate는 Pods에 대해 프로파일 A(`beta-workload`)를 선택합니다. 프로파일 A로 Pods의 레이블이 지정됩니다(예: `eks.amazonaws.com/fargate-profile=beta-workload`).

와일드카드를 사용하는 새 프로파일로 기존 Fargate Pods를 마이그레이션하려는 경우 다음 두 가지 방법이 있습니다.

- 일치하는 셀렉터로 새 프로파일을 생성한 다음 이전 프로파일을 삭제합니다. 이전 프로파일로 레이블이 지정된 포드는 일치하는 새 프로파일로 다시 예약됩니다.
- 워크로드를 마이그레이션하고 싶지만 각 Fargate Pod에 어떤 Fargate 레이블이 있는지 확실하지 않은 경우 다음 방법을 사용할 수 있습니다. 동일한 클러스터의 프로파일 중에서 영숫자순으로 먼

저 정렬되는 이름으로 새 프로파일을 생성합니다. 그런 다음 새 프로파일로 마이그레이션해야 하는 Fargate Pods를 재활용합니다.

Fargate 프로파일 생성

이 주제에서는 Fargate 프로파일을 생성하는 방법을 설명합니다. 또한 Fargate 프로파일에 사용할 Pod 실행 역할을 생성해야 합니다. 자세한 내용은 Amazon EKS Pod 실행 IAM 역할 섹션을 참조하세요. Fargate에서 실행되는 Pods는 인터넷 게이트웨이에 대한 직접 경로가 없고 AWS 서비스에 대한 NAT 게이트웨이 액세스 권한이 있는 프라이빗 서브넷에서만 지원됩니다. 이를 위해서는 클러스터의 VPC에 사용 가능한 프라이빗 서브넷이 있어야 합니다. `eksctl` 또는 AWS Management Console을 사용하여 프로파일을 생성할 수 있습니다.

이 절차에는 `eksctl` 버전 `0.175.0` 이상이 필요합니다. 버전은 다음 명령을 통해 확인할 수 있습니다.

```
eksctl version
```

`eksctl` 설치 또는 업데이트에 대한 지침은 `eksctl` 설명서의 Installation을 참조하세요.

eksctl

`eksctl`를 사용하여 Fargate 프로파일을 생성하려면

다음 `eksctl` 명령으로 Fargate 프로파일을 생성하고 모든 *`example value`*를 고유한 값으로 바꿉니다. 네임스페이스를 지정해야 합니다. 그러나 `--labels` 옵션은 필요하지 않습니다.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --name my-fargate-profile \
    --namespace my-kubernetes-namespace \
    --labels key=value
```

`my-kubernetes-namespace` 및 *`key=value`* 레이블에 특정 와일드카드를 사용할 수 있습니다. 자세한 내용은 Fargate 프로파일 와일드카드 단원을 참조하십시오.

AWS Management Console

AWS Management Console에서 클러스터용 Fargate 프로파일을 생성하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. Fargate 프로파일을 생성할 클러스터를 선택합니다.

3. 컴퓨팅(Compute) 탭을 선택합니다.
4. [Fargate 프로파일(Fargate profiles)]에서 [Fargate 프로파일 추가(Add Fargate profile)]를 선택합니다.
5. Fargate 프로파일 구성(Configure Fargate profile) 페이지에서 다음을 수행합니다.

   a. 이름(Name)에 Fargate 프로파일(예: ***my-profile***)의 고유한 이름을 입력합니다.

   b. 포드 실행 역할(Pod execution role)에서 Fargate 프로파일과 함께 사용할 Pod 실행 역할을 선택합니다. `eks-fargate-pods.amazonaws.com` 서비스 보안 주체가 있는 IAM 역할만 표시됩니다. 나열된 역할이 표시되지 않으면 역할을 만들어야 합니다. 자세한 내용은 Amazon EKS Pod 실행 IAM 역할 단원을 참조하십시오.

   c. 선택한 서브넷을 필요에 따라 수정합니다.

   > **Note**
   >
   > Fargate에서 실행되는 Pods에는 프라이빗 서브넷만 지원됩니다.

   d. [태그(Tags)]에 선택적으로 Fargate 프로파일에 태그를 지정할 수 있습니다. 이러한 태그는 프로파일과 연결된 다른 리소스(예: Pods)에 전파되지 않습니다.

   e. 다음을 선택합니다.
6. Pod 선택 구성(Configure pod selection) 페이지에서 다음을 수행합니다.

   a. Namespace(네임스페이스)에 Pods에 대해 일치시킬 네임스페이스를 입력합니다.

   - **`kube-system`** 또는 **`default`**와 같은 특정 네임스페이스를 사용하여 일치시킬 수 있습니다.
   - 특정 와일드카드(예: **`prod-*`**)를 사용하여 여러 네임스페이스(예: `prod-deployment` 및 `prod-test`)를 일치시킬 수 있습니다. 자세한 내용은 Fargate 프로파일 와일드카드 단원을 참조하십시오.

   b. (선택 사항) 셀렉터에 Kubernetes 레이블을 추가합니다. 특히 지정된 네임스페이스의 Pods가 일치해야 하는 포드에 추가합니다.

   - `infrastructure: fargate` Kubernetes 레이블도 있는 지정된 네임스페이스의 Pods만 선택기와 일치하도록 **`infrastructure: fargate`** 레이블을 셀렉터에 추가할 수 있습니다.

- 특정 와일드카드(예: **`key?: value?`**)를 사용하여 여러 네임스페이스(예: `keya: valuea` 및 `keyb: valueb`)를 일치시킬 수 있습니다. 자세한 내용은 Fargate 프로파일 와일드카드 단원을 참조하십시오.

c. Next(다음)를 선택합니다.

7. 검토 및 생성(Review and create) 페이지에서 Fargate 프로파일에 대한 정보를 검토하고 생성(Create)을 선택합니다.

Fargate 프로파일 삭제

이 주제에서는 Fargate 프로파일을 삭제하는 방법을 설명합니다.

Fargate 프로파일을 삭제하면 해당 프로파일을 사용하여 Fargate에 예약된 모든 Pods가 삭제됩니다. 이러한 Pods가 다른 Fargate 프로파일과 일치하면 해당 프로파일을 사용하여 Fargate에 예약됩니다. Fargate 프로파일과 더이상 일치하지 않으면 Fargate에 예약되지 않고 보류 상태로 남아있을 수 있습니다.

클러스터의 Fargate 프로파일은 한 번에 하나만 `DELETING` 상태가 될 수 있습니다. Fargate 프로파일이 삭제될 때까지 기다려야 해당 클러스터에서 다른 프로파일을 삭제할 수 있습니다.

`eksctl`, AWS Management Console 또는 AWS CLI를 사용하여 프로파일을 삭제할 수 있습니다. 프로파일을 삭제하는 데 사용할 도구 이름이 있는 탭을 선택합니다.

eksctl

`eksctl`를 사용하여 Fargate 프로파일을 삭제하는 방법

다음 명령을 사용하여 클러스터에서 프로파일을 삭제합니다. 모든 *`example value`*를 고유한 값으로 바꿉니다.

```
eksctl delete fargateprofile  --name my-profile --cluster my-cluster
```

AWS Management Console

AWS Management Console에서 클러스터의 Fargate 프로파일을 삭제하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터를 선택합니다. 목록 클러스터에서 Fargate 프로파일을 삭제할 클러스터를 선택합니다.

3. 컴퓨팅(Compute) 탭을 선택합니다.
4. 삭제할 Fargate 프로파일을 선택한 다음 삭제(Delete)를 선택합니다.
5. Delete Fargate Profile(Fargate 프로파일 삭제) 페이지에서 프로파일 이름을 입력한 다음 Delete(삭제)를 선택합니다.

AWS CLI

AWS CLI를 사용하여 Fargate 프로파일 삭제 방법

다음 명령을 사용하여 클러스터에서 프로파일을 삭제합니다. 모든 *example value*를 고유한 값으로 바꿉니다.

```
aws eks delete-fargate-profile --fargate-profile-name my-profile --cluster-name my-cluster
```

Fargate Pod 구성

Important

Amazon EKS가 있는 AWS Fargate은 AWS GovCloud(미국 동부) 및 AWSGovCloud(미국 서부)를 제외한 모든 Amazon EKS 지역에서 사용할 수 있습니다.

이 섹션에서는 AWS Fargate에서 Kubernetes Pods를 실행하기 위한 몇 가지 고유한 Pod 구성 세부 정보에 대해 설명합니다.

Pod CPU 및 메모리

Kubernetes을(를) 사용하여 Pod에서 각 컨테이너에 할당되는 요청, 최소 vCPU 양 및 메모리 리소스를 정의할 수 있습니다. Kubernetes에서 Pods을(를) 스케줄링하여 최소한 각 Pod에 대해 요청된 리소스가 컴퓨팅 리소스에서 사용 가능하도록 합니다. 자세한 내용을 알아보려면 Kubernetes 설명서의 컨테이너의 컴퓨팅 리소스 관리를 참조하세요.

Note

Amazon EKS Fargate는 노드당 하나의 Pod만 실행하므로 리소스가 더 적은 경우 Pods를 제거하는 시나리오가 발생하지 않습니다. 모든 Amazon EKS Fargate Pods는 보장된 우선순위로

실행되므로 요청된 CPU와 메모리는 모든 컨테이너의 한도와 같아야 합니다. 자세한 내용을 알아보려면 Kubernetes 문서의 Pods에 대한 서비스 품질 구성을 참조하세요.

Pods가 Fargate에 예약되면 Pod 사양 내의 vCPU 및 메모리 예약에 따라 Pod에 프로비저닝할 CPU 및 메모리 양이 결정됩니다.

- Init 컨테이너의 최대 요청은 Init 요청 vCPU 및 메모리 요구 사항을 결정하는 데 사용됩니다.
- 장기 실행 요청 vCPU 및 메모리 요구 사항을 결정하기 위해 모든 장기 실행 컨테이너에 대한 요청이 추가됩니다.
- Pod에 사용할 vCPU 및 메모리 요청에 대해서는 이전의 두 값 중 큰 값이 선택됩니다.
- Fargate는 필요한 Kubernetes 구성 요소(`kubelet`, `kube-proxy`, `containerd`)에 대한 각 Pod의 메모리 예약에 256MB를 추가합니다.

Fargate는 Pods가 실행해야 하는 리소스를 항상 보유하도록 vCPU 및 메모리 요청의 합계와 가장 일치하는 다음의 컴퓨팅 구성으로 올림 처리합니다.

vCPU 및 메모리 조합을 지정하지 않으면 사용 가능한 가장 작은 조합(.25 vCPU 및 0.5GB 메모리)이 사용됩니다.

아래 표에는 Fargate에서 실행되는 Pods에 사용할 수 있는 vCPU 및 메모리 조합이 나와 있습니다.

vCPU 값	메모리 값
.25 vCPU	0.5GB, 1GB, 2GB
.5 vCPU	1GB, 2GB, 3GB, 4GB
1 vCPU	2GB, 3GB, 4GB, 5GB, 6GB, 7GB, 8GB
2 vCPU	4~16GB(1GB 증분)
4 vCPU	8~30GB(1GB 증분)
8 vCPU	16~60GB(4GB 단위)
16 vCPU	32~120GB(8GB 단위)

Kubernetes 구성 요소에 예약된 추가 메모리로 인해 요청된 것보다 많은 vCPU를 사용하는 Fargate 태스크가 프로비저닝될 수 있습니다. 예를 들어 1개의 vCPU 및 8GB 메모리에 대한 요청의 경우 해당 메모리 요청에 256MB가 추가되며, vCPU 1개와 9GB 메모리를 사용하는 태스크가 없으므로 2개의 vCPU 및 9GB 메모리로 Fargate 태스크를 프로비저닝합니다.

Fargate에서 실행 중인 Pod의 크기와 `kubectl get nodes`를 사용하여 Kubernetes가 보고하는 노드 크기 사이에는 상관 관계가 없습니다. 보고된 노드 크기는 대개 Pod의 용량보다 큽니다. 다음 명령을 사용하여 Pod 용량을 확인할 수 있습니다. *`default`*을(를) Pod의 네임스페이스로, *`pod-name`*을(를) Pod의 이름으로 바꿉니다.

```
kubectl describe pod --namespace default pod-name
```

예제 출력은 다음과 같습니다.

```
[...]
annotations:
    CapacityProvisioned: 0.25vCPU 0.5GB
[...]
```

`CapacityProvisioned` 주석은 적용된 Pod 용량을 나타내며 Fargate에서 실행되는 Pod의 비용을 결정합니다. 이러한 컴퓨팅 구성에 대한 요금 정보는 AWS Fargate 요금 책정을 참조하세요.

Fargate 스토리지

Fargate에서 실행되는 Pod은(는) Amazon EFS 파일 시스템을 자동으로 탑재합니다. Fargate 노드에는 동적 영구 볼륨 프로비저닝을 사용할 수 없지만 고정적인 프로비저닝은 사용할 수 있습니다. 자세한 내용은 GitHub에서 Amazon EFS CSI 드라이버를 참조하세요.

프로비저닝 시 각 Fargate에서 Pod을(를) 실행하면 기본 20GiB의 임시 스토리지가 제공됩니다. 이 유형의 스토리지는 Pod이(가) 중지된 후에 삭제됩니다. Fargate에서 시작된 새 Pods에는 임시 스토리지 볼륨의 암호화가 기본적으로 활성화되어 있습니다. 임시 Pod 스토리지는 AWS Fargate 관리형 키를 사용하여 AES-256 암호화 알고리즘으로 암호화됩니다.

> **Note**
>
> Fargate에서 Pods이(가) 실행하는 Amazon EKS의 기본 가용 스토리지는 20GiB 미만입니다. 이는 일부 공간을 `kubelet` 및 Pod 내부에 로드되는 기타 Kubernetes 모듈에서 사용하기 때문입니다.

임시 스토리지의 총량은 최대 175GiB까지 높일 수 있습니다. Kubernetes을(를) 사용하여 크기를 구성하려면 Pod의 각 컨테이너에 대한 `ephemeral-storage` 리소스 요청을 지정합니다. Kubernetes이(가) Pods을(를) 스케줄링할 때 이는 각 Pod에 대한 리소스 요청의 합계가 Fargate 작업의 용량보다 작도록 보장합니다. 자세한 내용은 Kubernetes 문서의 Pods 및 컨테이너 리소스 관리를 참조하세요.

Amazon EKS Fargate는 시스템 사용 목적으로 요청한 것보다 더 많은 임시 스토리지를 프로비저닝합니다. 예를 들어 100GiB를 요청하면 Fargate 작업에 115GiB의 임시 스토리지가 프로비저닝됩니다.

Fargate OS 패치

> **Important**
>
> Amazon EKS가 있는 AWS Fargate은 AWS GovCloud(미국 동부) 및 AWSGovCloud(미국 서부)를 제외한 모든 Amazon EKS 지역에서 사용할 수 있습니다.

Amazon EKS는 AWS Fargate 노드의 OS를 주기적으로 패치하여 보안을 유지합니다. 패치 적용 프로세스의 일부로 노드를 재활용하여 OS 패치를 설치합니다. 서비스에 미치는 영향이 가장 적은 방식으로 업데이트가 시도됩니다. 그러나 Pods가 성공적으로 제거되지 않으면 포드를 삭제해야 할 때가 있습니다. 다음 작업은 중단 가능성을 최소화하기 위해 수행할 수 있는 작업입니다.

- 적절한 Pod 중단 예산(PDB)을 설정하여 동시에 중단되는 Pods의 수를 제어한다.
- Pods가 삭제되기 전에 실패한 제거를 처리하는 Amazon EventBridge 규칙을 생성합니다.
- AWS 사용자 알림에서 알림 구성을 생성합니다.

Amazon EKS는 Kubernetes 커뮤니티와 밀접하게 협력하여 버그 수정 및 보안 패치를 최대한 빨리 사용할 수 있도록 합니다. 모든 Fargate Pods는 Kubernetes 패치의 최신 버전에서 시작됩니다. 이 버전은 클러스터의 Kubernetes 버전에 대한 Amazon EKS에서 사용할 수 있습니다. 이전 패치 버전의 Pod가 있는 경우 Amazon EKS는 해당 포드를 재활용하여 최신 버전으로 업데이트할 수 있습니다. 이렇게 하면 Pods에 최신 보안 업데이트가 설치됩니다. 그렇게 하면 중요한 CVE(일반 취약성 및 노출) 문제가 있는 경우 최신 상태로 유지되어 보안 위험을 줄여줍니다.

Pods를 재활용할 때 한 번에 중단되는 Pods 수를 제한하려면 Pod 중단 예산(PDB)을 설정할 수 있습니다. PDB를 사용하여 각 애플리케이션의 요구 사항에 따라 최소 가용성을 정의하면서 계속 업데이트도 할 수 있습니다. 자세한 내용은 Kubernetes 설명서의 애플리케이션에 대한 중단 예산 지정을 참조하세요.

Amazon EKS가 제거 API를 사용하여 애플리케이션에 대해 설정한 PDB를 준수하면서 Pod를 안전하게 드레이닝합니다. 가용 영역에서 포드를 제거하여 영향을 최소화합니다. 제거가 성공하면 새 Pod가 최신 패치를 받아 추가 작업이 필요하지 않습니다.

Pod를 제거하는 데 실패하면 Amazon EKS는 제거에 실패한 Pods에 대한 세부 정보와 함께 이벤트를 계정에 전송합니다. 예약된 종료 시간 전에는 메시지에서 작업할 수 있습니다. 특정 시간은 패치의 긴급성에 따라 다릅니다. 때가 되면 Amazon EKS는 Pods를 다시 제거하려고 시도합니다. 그러나 이때 제거에 실패하면 새 이벤트가 발송되지 않습니다. 다시 제거하는 데 실패하면 새 Pods가 최신 패치를 받을 수 있도록 기존 Pods가 정기적으로 삭제됩니다.

다음 이벤트는 Pod 제거가 실패할 때 수신되는 샘플 이벤트입니다. 여기에는 클러스터, Pod 이름, Pod 네임스페이스, Fargate 프로필 및 예약된 종료 시간에 대한 세부 정보가 포함되어 있습니다.

```
{
    "version": "0",
    "id": "12345678-90ab-cdef-0123-4567890abcde",
    "detail-type": "EKS Fargate Pod Scheduled Termination",
    "source": "aws.eks",
    "account": "111122223333",
    "time": "2021-06-27T12:52:44Z",
    "region": "region-code",
    "resources": [
        "default/my-database-deployment"
    ],
    "detail": {
        "clusterName": "my-cluster",
        "fargateProfileName": "my-fargate-profile",
        "podName": "my-pod-name",
        "podNamespace": "default",
        "evictErrorMessage": "Cannot evict pod as it would violate the pod's disruption
 budget",
        "scheduledTerminationTime": "2021-06-30T12:52:44.832Z[UTC]"
    }
}
```

또한 여러 PDB가 Pod에 연결되어 있으면 제거 실패 이벤트가 발생할 수 있습니다. 이 이벤트는 다음과 같은 오류 메시지를 반환합니다.

```
"evictErrorMessage": "This pod has multiple PodDisruptionBudget, which the eviction
 subresource does not support",
```

이 이벤트를 기반으로 원하는 작업을 생성할 수 있습니다. 예를 들어 Pod 중단 예산(PDB)을 조정하여 Pods가 제거되는 방식을 제어할 수 있습니다. 조금 더 구체적으로 사용 가능한 Pods의 대상 백분율을 지정하는 PDB로 시작한다고 가정합니다. 업그레이드 중에 Pods가 강제 종료되기 전에 PDB를 다른 백분율의 Pods로 조정할 수 있습니다. 이 이벤트를 수신하려면 클러스터가 속한 AWS 계정 및 AWS 리전에서 Amazon EventBridge 규칙을 생성해야 합니다. 규칙은 다음과 같은 사용자 지정 패턴을 사용해야 합니다. 자세한 내용을 알아보려면 Amazon EventBridge 사용 설명서의 이벤트에 응답하는 Amazon EventBridge 규칙 생성을 참조하세요.

```
{
  "source": ["aws.eks"],
  "detail-type": ["EKS Fargate Pod Scheduled Termination"]
}
```

이벤트를 캡처하기 위해 적합한 대상을 설정할 수 있습니다. 사용 가능한 대상의 전체 목록은 Amazon EventBridge 사용 설명서의 Amazon EventBridge 대상을 참조하세요. AWS 사용자 알림에서 알림 구성을 생성할 수도 있습니다. AWS Management Console을 사용하여 알림을 생성할 때 이벤트 규칙에서 AWS 서비스 이름으로 Elastic Kubernetes Service(EKS)를 선택하고 이벤트 유형으로 EKS Fargate 포드 예약된 종료를 선택합니다. 자세한 내용은 AWS User Notifications User Guide의 Getting started with AWS User Notifications를 참조하세요.

Fargate 지표

> **Important**
>
> Amazon EKS가 있는 AWS Fargate은 AWS GovCloud(미국 동부) 및 AWSGovCloud(미국 서부)를 제외한 모든 Amazon EKS 지역에서 사용할 수 있습니다.

시스템 지표 및 AWS Fargate의 CloudWatch 사용량 지표를 수집할 수 있습니다.

애플리케이션 지표

Amazon EKS 및 AWS Fargate에서 실행되는 애플리케이션의 경우 AWS Distro for OpenTelemetry(ADOT)를 사용할 수 있습니다. ADOT를 사용하면 시스템 지표를 수집하여 CloudWatch 컨테이너 인사이트 대시보드로 전송할 수 있습니다. Fargate에서 실행되는 애플리케이션용 ADOT로 시작하려면 ADOT 설명서의 AWS Distro for OpenTelemetry를 통한 CloudWatch 컨테이너 인사이트 사용을 참조하세요.

사용량 지표

CloudWatch 사용량 지표를 사용하여 계정의 리소스 사용량을 확인할 수 있습니다. 이러한 지표를 사용하여 CloudWatch 그래프 및 대시보드에서 현재 서비스 사용량을 시각화합니다.

AWS Fargate 사용량 지표는 AWS 서비스 할당량에 해당합니다. 사용량이 서비스 할당량에 가까워지면 경고하는 경보를 구성할 수 있습니다. Fargate 서비스 할당량에 대한 자세한 정보는 Amazon EKS 서비스 할당량 섹션을 참조하세요.

AWS Fargate는 `AWS/Usage` 네임스페이스에 다음 지표를 게시합니다.

지표	설명
`ResourceCount`	계정에서 실행 중인 지정된 리소스의 총 수입니다. 리소스는 지표와 연결된 차원으로 정의됩니다.

다음 차원은 AWS Fargate에 의해 게시되는 사용량 지표를 구체화하는 데 사용됩니다.

차원	설명
`Service`	리소스가 포함된 AWS 서비스의 이름 AWS Fargate 사용량 지표의 경우 이 차원 값은 `Fargate`입니다.
`Type`	보고되는 엔터티의 유형입니다. 현재 AWS Fargate 사용량 지표에 대한 유일한 유효 값은 `Resource`입니다.
`Resource`	실행 중인 리소스의 유형입니다. 현재 AWS Fargate는 Fargate 온디맨드 사용량에 대한 정보를 반환합니다. Fargate 온디맨드 사용량에 대한 리소스 값은 `OnDemand`입니다. **Note** Fargate 온디맨드 사용량은 Fargate를 사용하는 Amazon EKS Pods, Fargate 시작 유형을 사용하는 Amazon ECS 태스크 및 `FARGATE` 용량 공급자를 사용하는 Amazon ECS가 합쳐진 값입니다.

차원	설명
`Class`	추적 중인 리소스의 클래스. 현재 AWS Fargate에서는 클래스 차원을 사용하지 않습니다.

Fargate 리소스 사용량 지표 모니터링을 위한 CloudWatch 경보 생성

AWS Fargate는 Fargate 온디맨드 리소스 사용량에 대한 AWS 서비스 할당량에 해당하는 CloudWatch 사용량 지표를 제공합니다. Service Quotas 콘솔에서 사용량을 그래프로 시각화할 수 있습니다. 사용량이 서비스 할당량에 가까워지면 사용자에게 경고를 보내도록 경보(alarms)를 구성할 수 있습니다. 자세한 내용은 Fargate 지표 섹션을 참조하세요.

다음 단계를 사용하여 Fargate 리소스 사용량 지표에 기반하여 CloudWatch 경보를 생성합니다.

Fargate 사용량 할당량(AWS Management Console)을 기반으로 경보를 생성하려면

1. Service Quotas 콘솔(https://console.aws.amazon.com/servicequotas/)을 엽니다.
2. 왼쪽 탐색 창에서 AWS 서비스(services)를 선택합니다.
3. AWS 서비스(services) 목록에서 AWS Fargate을(를) 검색하여 선택합니다.
4. 서비스 할당량(Service quotas) 목록에서 경보를 만들려는 Fargate 사용량 할당량을 선택합니다.
5. Amazon CloudWatch 경보(Amazon CloudWatch alarms) 단원에서 생성(Create)을 선택합니다.
6. 경보 임계값(Alarm threshold)에서 경보 값으로 설정할 적용된 할당량 값의 백분율을 선택합니다.
7. 경보 이름(Alarm name)에서 경보 이름을 입력한 다음 생성(Create)을 선택합니다.

Fargate 로깅

> **Important**
>
> Amazon EKS가 있는 AWS Fargate은 AWS GovCloud(미국 동부) 및 AWSGovCloud(미국 서부)를 제외한 모든 Amazon EKS 지역에서 사용할 수 있습니다.

Amazon EKS on Fargate는 Fluent Bit를 기반으로 내장된 로그 라우터를 제공합니다. 즉, 사용자가 Fluent Bit 컨테이너를 사이드카로 명시적으로 실행하지는 않지만 Amazon이 자동으로 실행합니다. 로그 라우터를 구성하기만 하면 됩니다. 이 구성은 다음 기준을 충족해야 하는 전용 `ConfigMap`을 통해 이루어집니다.

- `aws-logging`으로 이름 지정
- `aws-observability`라는 전용 네임스페이스에서 생성
- 5300자를 초과할 수 없습니다.

`ConfigMap`을 생성하고 나면 Fargate의 Amazon EKS가 자동으로 감지하고 로그 라우터를 구성합니다. Fargate는 Fluent Bit에 대한 AWS의 한 버전인 AWS에서 관리하는 Fluent Bit의 업스트림 호환 배포판을 사용합니다. 자세한 내용을 알아보려면 GitHub의 Fluent Bit에 대한 AWS를 참조하세요.

로그 라우터를 사용하면 AWS에서 로그 분석 및 저장을 위한 다양한 서비스를 사용할 수 있습니다. Fargate에서 Amazon CloudWatch, Amazon OpenSearch Service로 로그를 직접 스트리밍할 수 있습니다. Amazon S3, Amazon Kinesis Data Streams, Amazon Data Firehose를 통한 파트너 도구 등의 대상으로 로그를 스트리밍할 수도 있습니다.

필수 조건

- Fargate Pods를 배포할 기존 Kubernetes 네임스페이스를 지정하는 기존 Fargate 프로파일. 자세한 내용은 클러스터에 대한 Fargate 프로필 생성 단원을 참조하십시오.
- 기존 Fargate Pod 실행 역할. 자세한 내용은 Fargate Pod 실행 역할 생성 단원을 참조하십시오.

로그 라우터 구성

로그 라우터 구성하기

다음 단계에서 모든 *example value*를 고유한 값으로 바꿉니다.

1. `aws-observability`로 이름이 지정된 전용 Kubernetes 네임스페이스를 생성합니다.

   a. 다음 콘텐츠를 컴퓨터에 *aws-observability-namespace*.yaml이라는 파일에 저장합니다. `name`의 값은 `aws-observability`여야 하며 `aws-observability: enabled` 레이블이 필요합니다.

   ```
   kind: Namespace
   apiVersion: v1
   metadata:
     name: aws-observability
     labels:
       aws-observability: enabled
   ```

   b. 네임스페이스를 생성합니다.

```
kubectl apply -f aws-observability-namespace.yaml
```

2. Fluent Conf 데이터 값을 사용하여 ConfigMap을 생성함으로써 컨테이너 로그를 대상에 전달합니다. Fluent Conf는 Fluent Bit입니다. 이는 원하는 로그 대상에 컨테이너 로그를 라우팅하는 데 사용되는 빠르고 가벼운 로그 프로세서 구성 언어입니다. 자세한 내용을 알아보려면 Fluent Bit 설명서의 구성 파일을 참조하세요.

> **Important**
>
> 일반적인 Fluent Conf에 포함된 주요 단원은 Service, Input, Filter, Output입니다. 하지만 Fargate 로그 라우터는 다음 부분만 수락합니다.
>
> - Filter 및Output 부분입니다.
> - Parser 부분.
>
> 기타 부분을 제공하는 경우 해당 부분은 거부됩니다.

Fargate 로그 라우터에서는 Service 및Input 부분을 관리합니다. 여기에는 수정할 수 없고 ConfigMap에서 필요하지 않은 다음과 같은 Input 부분이 있습니다. 그러나 메모리 버퍼 제한과 로그에 적용된 태그와 같은 인사이트를 얻을 수 있습니다.

```
[INPUT]
    Name tail
    Buffer_Max_Size 66KB
    DB /var/log/flb_kube.db
    Mem_Buf_Limit 45MB
    Path /var/log/containers/*.log
    Read_From_Head On
    Refresh_Interval 10
    Rotate_Wait 30
    Skip_Long_Lines On
    Tag kube.*
```

ConfigMap 생성 시에는 Fargate가 필드를 검증하는 데 사용하는 다음 규칙을 고려하세요.

- [FILTER], [OUTPUT], [PARSER]는 각 해당 키 아래에 지정되어야 합니다. 예를 들어 filters.conf은 [FILTER]에 있어야 합니다. filters.conf에 하나 이상의 [FILTER]가

있을 수 있습니다. [OUTPUT] 및 [PARSER] 부분도 해당 키 아래에 있어야 합니다. 여러 개의 [OUTPUT] 부분을 지정하여 로그를 여러 대상으로 동시에 라우팅할 수 있습니다.

- Fargate는 각 섹션에 필요한 키를 검증합니다 Name 및 match는 [FILTER] 및 [OUTPUT]에 각각 필요합니다. Name 및 format은 [PARSER]에 각각 필요합니다. 태그는 대/소문자를 구분합니다.
- ${ENV_VAR}과 같은 환경 변수는 ConfigMap에서 허용되지 않습니다.
- 들여쓰기는 각 filters.conf, output.conf, parsers.conf 내의 지시문 또는 키 값 페어에 대해 동일해야합니다. 키 값 페어는 지시문보다 들여 써야 합니다.
- Fargate는 grep, parser, record_modifier, rewrite_tag, throttle, nest, modify, kubernetes 등의 지원 필터에 대해 검증을 실시합니다.
- Fargate는 다음과 같은 지원되는 출력에 대해 검증합니다. es, firehose, kinesis_firehose, cloudwatch, cloudwatch_logs 및 kinesis.
- 로깅을 사용 설정하려면 ConfigMap에서 지원되는 Output 플러그 인이 1개 이상 제공되어야 합니다. Filter 및 Parser는 로깅을 사용 설정하는 데 필요하지 않습니다.

검증에서 발생하는 문제를 해결하기 위해 원하는 구성을 사용하여 Amazon EC2에서 Fluent Bit를 실행할 수 있습니다. 다음 예 중 하나를 사용하여 ConfigMap을 생성합니다.

> **Important**
>
> Amazon EKS Fargate 로깅은 ConfigMaps의 동적 구성을 지원하지 않습니다. ConfigMaps의 모든 변경 사항은 새 Pods에만 적용됩니다. 기존 Pods에는 변경 사항이 적용되지 않습니다.

원하는 로그 대상에 대한 예제를 사용하여 ConfigMap을 생성합니다.

> **Note**
>
> 로그 대상에 Amazon Kinesis Data Streams를 사용할 수도 있습니다. Kinesis Data Streams를 사용하는 경우 포드 실행 역할에 kinesis:PutRecords 권한이 부여되었는지 확인하세요. 자세한 내용은 Fluent Bit: Official Manual의 Amazon Kinesis Data Streams Permissions를 참조하세요.

CloudWatch

CloudWatch의 **ConfigMap**를 생성하는 방법

CloudWatch를 사용할 때는 다음 두 가지 출력 옵션이 있습니다.

- C로 작성된 출력 플러그 인
- Golang으로 작성된 출력 플러그 인

다음 예에서는 `cloudwatch_logs` 플러그 인을 사용하여 CloudWatch로 로그를 전송하는 방법을 보여줍니다.

1. 다음 콘텐츠를 *aws-logging-cloudwatch-configmap*.yaml이라는 파일에 저장합니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다. [OUTPUT] 아래의 파라미터는 필수입니다.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  flb_log_cw: "false"  # Set to true to ship Fluent Bit process logs to
 CloudWatch.
  filters.conf: |
    [FILTER]
        Name parser
        Match *
        Key_name log
        Parser crio
    [FILTER]
        Name kubernetes
        Match kube.*
        Merge_Log On
        Keep_Log Off
        Buffer_Size 0
        Kube_Meta_Cache_TTL 300s
  output.conf: |
    [OUTPUT]
        Name cloudwatch_logs
```

```
        Match   kube.*
        region region-code
        log_group_name my-logs
        log_stream_prefix from-fluent-bit-
        log_retention_days 60
        auto_create_group true
  parsers.conf: |
    [PARSER]
        Name crio
        Format Regex
        Regex ^(?<time>[^ ]+) (?<stream>stdout|stderr) (?<logtag>P|F) (?
<log>.*)$
        Time_Key    time
        Time_Format %Y-%m-%dT%H:%M:%S.%L%z
```

2. 매니페스트를 클러스터에 적용합니다.

```
kubectl apply -f aws-logging-cloudwatch-configmap.yaml
```

3. CloudWatch IAM 정책을 컴퓨터에 다운로드합니다. GitHub에서 정책을 볼 수도 있습니다.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/cloudwatchlogs/permissions.json
```

Amazon OpenSearch Service

Amazon OpenSearch Service에 대한 **ConfigMap** 생성

Amazon OpenSearch Service로 로그를 전송하려는 경우 C로 작성된 플러그 인인 es 출력을 사용할 수 있습니다. 다음 예에서는 플러그 인을 사용하여 OpenSearch로 로그를 전송하는 방법을 보여줍니다.

1. 다음 콘텐츠를 *aws-logging-opensearch-configmap*.yaml이라는 파일에 저장합니다. 모든 *example value*를 고유한 값으로 바꿉니다.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
```

```
  output.conf: |
    [OUTPUT]
      Name  es
      Match *
      Host  search-example-gjxdcilagiprbglqn42jsty66y.region-
code.es.amazonaws.com
      Port  443
      Index example
      Type  example_type
      AWS_Auth On
      AWS_Region region-code
      tls   On
```

2. 매니페스트를 클러스터에 적용합니다.

```
kubectl apply -f aws-logging-opensearch-configmap.yaml
```

3. OpenSearch IAM 정책을 컴퓨터에 다운로드합니다. GitHub에서 정책을 볼 수도 있습니다.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-
fluent-logging-examples/mainline/examples/fargate/amazon-elasticsearch/
permissions.json
```

OpenSearch Dashboard의 액세스 컨트롤이 제대로 구성되어 있는지 확인합니다. OpenSearch Dashboard의 `all_access role`에는 Fargate Pod 실행 역할과 IAM 역할이 매핑되어야 합니다. `security_manager` 역할에 대해서도 동일한 매핑이 수행되어야 합니다. 이전 매핑을 추가하려면 `Menu`, `Security`, `Roles`를 차례로 선택한 다음 해당 역할을 선택합니다. 자세한 내용은 CloudWatch Logs가 내 Amazon ES 도메인으로 스트리밍되도록 문제를 해결하려면 어떻게 해야 하나요? 부분을 참조하세요.

Firehose

Firehose용 **`ConfigMap`**을 생성하려면

Firehose로 로그를 전송할 때는 다음과 같은 두 가지 출력 옵션이 있습니다.

- `kinesis_firehose` - C로 작성된 출력 플러그 인
- `firehose`-Golang으로 작성된 출력 플러그 인

다음 예에서는 kinesis_firehose 플러그 인을 사용하여 Firehose로 로그를 전송하는 방법을 보여줍니다.

1. 다음 콘텐츠를 *aws-logging-firehose-configmap*.yaml이라는 파일에 저장합니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: aws-logging
  namespace: aws-observability
data:
  output.conf: |
    [OUTPUT]
     Name  kinesis_firehose
     Match *
     region region-code
     delivery_stream my-stream-firehose
```

2. 매니페스트를 클러스터에 적용합니다.

```
kubectl apply -f aws-logging-firehose-configmap.yaml
```

3. Firehose IAM 정책을 컴퓨터에 다운로드합니다. GitHub에서 정책을 볼 수도 있습니다.

```
curl -O https://raw.githubusercontent.com/aws-samples/amazon-eks-fluent-
logging-examples/mainline/examples/fargate/kinesis-firehose/permissions.json
```

3. 이전 단계에서 다운로드한 정책 파일에서 IAM 정책을 만듭니다.

```
aws iam create-policy --policy-name eks-fargate-logging-policy --policy-document
 file://permissions.json
```

4. 다음 명령을 사용하여 Fargate 프로필에 대해 지정된 포드 실행 역할에 IAM 정책을 연결합니다. *111122223333*을 계정 ID로 바꿉니다. *AmazonEKSFargatePodExecutionRole*을 Pod 실행 역할로 바꿉니다(자세한 내용은 Fargate Pod 실행 역할 생성 참조).

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/eks-fargate-logging-policy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Kubernetes 필터 지원

이 기능을 사용하려면 다음과 같은 최소 Kubernetes 버전 및 플랫폼 수준 이상이 필요합니다.

Kubernetes 버전	플랫폼 수준
1.23 이상	eks.1

Fluent Bit Kubernetes 필터를 사용하면 로그 파일에 Kubernetes 메타데이터를 추가할 수 있습니다. 필터에 대한 자세한 내용은 Fluent Bit 설명서의 Kubernetes를 참조하세요. API 서버 엔드포인트를 사용하여 필터를 적용할 수 있습니다.

```
filters.conf: |
    [FILTER]
        Name                kubernetes
        Match               kube.*
        Merge_Log           On
        Buffer_Size         0
        Kube_Meta_Cache_TTL 300s
```

Important

- `Kube_URL`, `Kube_CA_File`, `Kube_Token_Command` 및 `Kube_Token_File`은 서비스 소유 구성 파라미터이므로 지정하면 안 됩니다. Amazon EKS Fargate가 이러한 값을 채웁니다.
- `Kube_Meta_Cache_TTL`은 Fluent Bit가 최신 메타데이터에 대해 API 서버와 통신할 때까지 대기하는 시간입니다. `Kube_Meta_Cache_TTL`을 지정하지 않으면 Amazon EKS Fargate는 API 서버의 로드를 줄이기 위해 기본값 30분을 추가합니다.

계정에 Fluent Bit 프로세스 로그 전송

다음 `ConfigMap`을 사용하여 Fluent Bit 프로세스 로그를 Amazon CloudWatch로 선택적으로 전송할 수 있습니다. Fluent Bit 프로세스 로그를 CloudWatch로 전송하려면 추가 로그 수집 및 스토리지 비용이 필요합니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다.

```
kind: ConfigMap
apiVersion: v1
```

```
metadata:
  name: aws-logging
  namespace: aws-observability
  labels:
data:
  # Configuration files: server, input, filters and output
  # ======================================================
  flb_log_cw: "true"  # Ships Fluent Bit process logs to CloudWatch.

  output.conf: |
    [OUTPUT]
        Name cloudwatch
        Match kube.*
        region region-code
        log_group_name fluent-bit-cloudwatch
        log_stream_prefix from-fluent-bit-
        auto_create_group true
```

로그는 CloudWatch 아래에 클러스터가 있는 AWS 리전에 있습니다. 로그 그룹 이름은 *my-cluster*-fluent-bit-logs이고 Fluent Bit 로그스트림 이름은 fluent-bit-*podname*-*pod-namespace*입니다.

> **Note**
>
> - 프로세스 로그는 Fluent Bit 프로세스가 성공적으로 시작된 경우에만 전송됩니다. Fluent Bit를 시작하는 동안 오류가 발생하면 프로세스 로그가 누락됩니다. CloudWatch에는 프로세스 로그만 전송할 수 있습니다.
> - 계정으로 프로세스 로그 전송을 디버깅하려면 이전 ConfigMap을 적용하여 프로세스 로그를 가져올 수 있습니다. Fluent Bit가 시작하지 못하는 것은 대개 시작하는 동안 Fluent Bit에 의해 구문 분석 또는 수락되지 않은 ConfigMap 때문입니다.

Fluent Bit 프로세스 로그 전송을 중지하려면

Fluent Bit 프로세스 로그를 CloudWatch로 전송하려면 추가 로그 수집 및 스토리지 비용이 필요합니다. 기존 ConfigMap 설정에서 프로세스 로그를 제외하려면 다음 단계를 수행하세요.

1. Fargate 로깅을 활성화한 후 Amazon EKS 클러스터의 Fluent Bit 프로세스 로그에 대해 자동으로 생성된 CloudWatch 로그 그룹을 찾습니다. 형식 {cluster_name}-fluent-bit-logs을 따릅니다.

2. CloudWatch 로그 그룹에서 각 Pod's의 프로세스 로그에 대해 생성된 기존 CloudWatch 로그 스트림을 삭제합니다.
3. `ConfigMap`을 편집하고 `flb_log_cw: "false"`를 설정합니다.
4. 클러스터의 기존 Pods를 다시 시작합니다.

애플리케이션 테스트

1. 샘플 Pod를 배포합니다.
   a. 다음 콘텐츠를 컴퓨터에 *sample-app*.yaml이라는 파일에 저장합니다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-app
  namespace: same-namespace-as-your-fargate-profile
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - name: http
              containerPort: 80
```

   b. 매니페스트를 클러스터에 적용합니다.

```
kubectl apply -f sample-app.yaml
```

2. `ConfigMap`에서 구성한 대상을 사용하여 NGINX 로그를 봅니다.

크기 조정 고려 사항

로그 라우터에 대해 최대 50MB의 메모리를 계획하는 것이 좋습니다. 애플리케이션에서 매우 높은 처리량으로 로그를 생성할 것으로 예상되는 경우 최대 100MB까지 계획해야 합니다.

문제 해결

잘못된 ConfigMap과 같은 원인으로 로깅 기능이 사용 또는 사용 중지되었는지 여부와 잘못된 이유를 확인하려면 **kubectl describe pod *pod_name***을 사용하여 Pod 이벤트를 살펴봅니다. 출력에는 다음 예제 출력과 같이 로깅의 사용 여부를 명확히 하는 Pod 이벤트가 포함될 수 있습니다.

```
[...]
Annotations:          CapacityProvisioned: 0.25vCPU 0.5GB
                      Logging: LoggingDisabled: LOGGING_CONFIGMAP_NOT_FOUND
                      kubernetes.io/psp: eks.privileged
[...]
Events:
  Type     Reason           Age        From
               Message
  ----     ------           ----       ----
               -------
  Warning  LoggingDisabled  <unknown>  fargate-scheduler
                Disabled logging because aws-logging configmap was not found. configmap
 "aws-logging" not found
```

Pod 이벤트는 설정에 따라 기간이 정해진 일시적인 이벤트입니다. 또한 **kubectl describe pod *pod-name***을 사용하여 Pod's 주석을 볼 수 있습니다. Pod 주석에는 로깅 기능이 사용 또는 사용 중지되었는지 여부와 이유에 대한 정보가 있습니다.

Amazon EC2 인스턴스 유형 선택

Amazon EC2에서는 워커 노드에 대한 다양한 인스턴스 유형을 제공합니다. 인스턴스 유형마다 서로 다른 컴퓨팅, 메모리, 스토리지 및 네트워크 기능을 제공합니다. 또한 각 인스턴스는 이러한 기능에 따라 한 인스턴스 패밀리로 그룹화됩니다. 목록은 Linux 인스턴스용 Amazon EC2 사용 설명서의 사용 가능한 인스턴스 유형 및 Windows 인스턴스용 Amazon EC2 사용 설명서의 사용 가능한 인스턴스 유형을 참조하세요. Amazon EKS에서는 Amazon EC2 AMI의 여러 변형을 출시하여 지원을 사용 설정합니다. 선택한 인스턴스 유형이 Amazon EKS와 호환되는지 확인하려면 다음 기준을 고려하세요.

- 모든 Amazon EKS AMI에서는 현재 g5g 및 mac 패밀리를 지원하지 않습니다.

- Arm 및 가속화되지 않은 Amazon EKS AMI에서는 g3, g4, inf 및 p 패밀리를 지원하지 않습니다.
- 가속화된 Amazon EKS AMI에서는 a, c, hpc, m 및 t 패밀리를 지원하지 않습니다.
- ARM 기반 인스턴스에서 Amazon Linux 2023(AL2023)은 Graviton2 이상 프로세서를 사용하는 인스턴스 유형만 지원합니다. AL2023은 A1 인스턴스를 지원하지 않습니다.

Amazon EKS에서 지원하는 인스턴스 유형 중에서 선택하는 경우 각 유형의 다음 기능을 고려하세요.

노드 그룹의 인스턴스 수

특히 Daemonsets가 많은 경우, 일반적으로 수는 적고 크기는 큰 인스턴스가 더 좋습니다. 각 인스턴스에는 API 서버에 대한 API 호출이 필요하므로 인스턴스가 많을수록 API 서버에 더 많은 부하가 발생합니다.

운영 체제

Linux, Windows 및 Bottlerocket에 대해 지원되는 인스턴스 유형을 검토합니다. Windows 인스턴스를 생성하기 전에 Amazon EKS 클러스터에 대해 Windows 지원 사용 설정를 검토하세요.

하드웨어 아키텍처

x86 또는 Arm이 필요합니까? Arm에 Linux만 배포할 수 있습니다. Arm 인스턴스를 배포하기 전에 Amazon EKS 최적화 Arm Amazon Linux AMI를 검토하세요. Nitro System(Linux 또는 Windows)에 구축된 인스턴스나 가속화된 기능이 있는 인스턴스가 필요하신가요? 가속화된 기능이 필요한 경우 Amazon EKS에서 Linux만 사용할 수 있습니다.

Pods의 최대 개수

각 Pod에 고유한 IP 주소가 할당되므로 인스턴스 유형에서 지원하는 IP 주소 수는 인스턴스에서 실행할 수 있는 Pods 수를 결정하는 요소입니다. 인스턴스 유형이 지원하는 Pods의 수를 수동으로 확인하려면 각 Amazon EC2 인스턴스 유형의 Amazon EKS 권장 최대 Pods 수 섹션을 참조하세요.

> **Note**
>
> Amazon EKS에 최적화된 v20220406 이상의 Amazon Linux 2 AMI를 사용하는 경우 최신 AMI로 업그레이드하지 않고도 새로운 인스턴스 유형을 사용할 수 있습니다. 이러한 AMI의 경우 AMI는 eni-max-pods.txt 파일에 나열되지 않으면 필요한 max-pods 값을 자동으로 계산합니다. 현재 미리 보기 중인 인스턴스 유형은 기본적으로 Amazon EKS에서 지원되지 않을 수 있습니다. 이러한 유형에 대한 max-pods의 값은 여전히 AMI의 eni-max-pods.txt에 추가되어야 합니다.

AWS Nitro 시스템(Nitro System) 인스턴스 유형은 Nitro 시스템 이외의 인스턴스 유형보다 훨씬 많은 IP 주소를 선택적으로 지원합니다. 그러나 인스턴스에 할당된 일부 IP 주소만 Pods에 사용할 수 있습니다. 훨씬 더 많은 수의 IP 주소를 인스턴스에 할당하려면 클러스터에 버전 `1.9.0` 이상의 Amazon VPC CNI 추가 기능이 설치되어 있어야 하며 올바르게 구성되어 있어야 합니다. 자세한 내용은 Amazon EC2 노드에 사용 가능한 IP 주소 증량 단원을 참조하십시오. 인스턴스에 가장 많은 수의 IP 주소를 할당하려면 클러스터에 버전 `1.10.1` 이상의 Amazon VPC CNI 추가 기능이 설치되어 있어야 하며 `IPv6` 패밀리를 사용하여 클러스터를 배포해야 합니다.

IP 패밀리

클러스터가 Pods 및 서비스에 프라이빗 `IPv4` 주소를 할당하도록 허용하는 클러스터용 `IPv4` 패밀리를 사용하는 경우 지원되는 모든 인스턴스 유형을 사용할 수 있습니다. 클러스터에 대한 `IPv6` 패밀리를 사용하려는 경우 AWS Nitro 시스템(Nitro System) 인스턴스 유형 또는 베어 메탈 인스턴스 유형을 사용해야 합니다. Windows 인스턴스에는 `IPv4`만 지원됩니다. 클러스터에서 버전 `1.10.1` 이상의 Amazon VPC CNI 추가 기능을 실행하고 있어야 합니다. `IPv6` 사용에 관한 자세한 내용은 클러스터, Pods 및 services용 `IPv6` 주소 부분을 참조하세요.

실행 중인 Amazon VPC CNI 추가 기능의 버전

최신 버전의 Kubernetes용 Amazon VPC CNI 플러그인은 다음 인스턴스 유형을 지원합니다. 지원되는 최신 인스턴스 유형을 활용하려면 Amazon VPC CNI 추가 기능 버전을 업데이트해야 할 수 있습니다. 자세한 내용은 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 단원을 참조하십시오. 최신 버전은 Amazon EKS에서 사용할 최신 기능을 지원합니다. 이전 버전에서는 일부 기능을 지원하지 않습니다. GitHub의 Changelog에 있는 다양한 버전에서 지원하는 기능을 볼 수 있습니다.

노드를 생성하는 AWS 리전

AWS 리전에 따라 일부 인스턴스 유형은 사용할 수 없습니다.

Pods에 대한 보안 그룹 사용 여부

Pods에 대한 보안 그룹을 사용하는 경우 특정 인스턴스 유형만 지원됩니다. 자세한 내용은 Pods의 보안 그룹 단원을 참조하십시오.

각 Amazon EC2 인스턴스 유형의 Amazon EKS 권장 최대 Pods 수

각 Pod에 고유한 IP 주소가 할당되므로 인스턴스 유형에서 지원하는 IP 주소 수는 인스턴스에서 실행할 수 있는 Pods 수를 결정하는 요소입니다. Amazon EKS는 각 인스턴스 유형에서 실행할 Amazon EKS 권장 최대 Pods 수를 결정하기 위해 다운로드하여 실행할 수 있는 스크립트를 제공합니다. 스크립트는 각 인스턴스의 하드웨어 속성 및 구성 옵션을 사용하여 최대 Pods 수를 결정합니다. 이 단계에

서 반환된 수를 사용하여 인스턴스의 서브넷과 다른 서브넷의 Pods에 IP 주소 할당 및 인스턴스에 대한 IP 주소 수 대량 증가와 같은 기능을 활성화할 수 있습니다. 여러 인스턴스 유형을 통해 관리형 노드 그룹을 사용하는 경우 모든 인스턴스 유형에 작동하는 값을 사용합니다.

1. 각 인스턴스 유형에 대한 최대 Pods 수를 계산하는 데 사용할 수 있는 스크립트를 다운로드합니다.

```
curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/files/max-pods-calculator.sh
```

2. 컴퓨터에서 스크립트를 실행 파일로 표시합니다.

```
chmod +x max-pods-calculator.sh
```

3. 스크립트를 실행하여 *m5.large*를 배포하려는 인스턴스 유형으로 바꾸고 *1.9.0-eksbuild.1*을 Amazon VPC CNI 추가 기능 버전으로 바꿉니다. 추가 기능 버전을 확인하려면 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업의 업데이트 절차를 참조하세요.

```
./max-pods-calculator.sh --instance-type m5.large --cni-version 1.9.0-eksbuild.1
```

예제 출력은 다음과 같습니다.

```
29
```

스크립트에 다음 옵션을 추가하여 선택적 기능을 사용할 때 지원되는 최대 Pods 수를 확인할 수 있습니다.

- `--cni-custom-networking-enabled` - 인스턴스의 서브넷과 다른 서브넷의 IP 주소를 할당하려는 경우 이 옵션을 사용합니다. 자세한 내용은 포드에 대한 사용자 지정 네트워킹 단원을 참조하십시오. 동일한 예제 값을 사용하여 이전 스크립트에 이 옵션을 추가하면 `20`이 표시됩니다.
- `--cni-prefix-delegation-enabled` - 각 Elastic 네트워크 인터페이스에 훨씬 더 많은 IP 주소를 할당하려는 경우 이 옵션을 사용합니다. 이 기능을 사용하려면 Nitro 시스템과 Amazon VPC CNI 추가 기능 버전 `1.9.0` 이상에서 실행되는 Amazon Linux 인스턴스가 필요합니다. 자세한 내용은 Amazon EC2 노드에 사용 가능한 IP 주소 증량 단원을 참조하십시오. 동일한 예제 값을 사용하여 이전 스크립트에 이 옵션을 추가하면 `110`이 표시됩니다.

`--help` 옵션을 사용하여 스크립트를 실행하면 사용 가능한 모든 옵션을 볼 수 있습니다.

> **Note**
>
> 최대 Pods 계산기 스크립트는 Kubernetes 확장성 임곗값과 권장 설정을 기반으로 반환 값을 110으로 제한합니다. 인스턴스 유형의 vCPU가 30개 이상인 경우 한도는 내부 Amazon EKS 확장성 팀 테스팅을 기반으로 한 수치로 올라갑니다.250 자세한 내용은 Amazon VPC CNI 플러그인을 통한 노드당 포드 제한 증가 블로그 게시물을 참조하세요.

Amazon EKS 최적화 AMI

사전 구축된 Amazon EKS 최적화 Amazon Machine Image(AMI) 또는 사용자 지정 AMI를 사용하여 노드를 배포할 수 있습니다. Amazon EKS 최적화 AMI의 각 유형에 대한 자세한 내용은 다음 주제 중 하나를 참조하세요. 고유한 사용자 지정 AMI를 생성하는 방법에 대한 설명은 Amazon EKS 최적화 Amazon Linux AMI 빌드 스크립트 섹션을 참조하세요.

주제

- Amazon EKS에서는 Dockershim에 대한 지원을 종료했습니다.
- Amazon EKS 최적화 Amazon Linux AMI
- Amazon EKS 최적화 Bottlerocket AMI
- Amazon EKS 최적화 Ubuntu Linux AMI
- Amazon EKS 최적화 Windows AMI

Amazon EKS에서는 **Dockershim**에 대한 지원을 종료했습니다.

Kubernetes는 더 이상 `Dockershim`을 지원하지 않습니다. Kubernetes 팀이 Kubernetes 버전 `1.24`에서 런타임을 제거했습니다. 자세한 내용을 알아보려면 Kubernetes 블로그의 Kubernetes는 Dockershim에서 이동 중: 약속 및 다음 단계를 참조하세요.

Amazon EKS에서는 Kubernetes 버전 `1.24` 릴리스부터 `Dockershim`에 대한 지원을 종료합니다. 공식적으로 게시된 Amazon EKS AMI에는 버전 `1.24`부터 유일한 런타임으로 `containerd`가 있습니다. 이 항목에서는 몇 가지 세부 정보를 다루지만 자세한 내용은 Amazon EKS containerd로의 이동에 관해 알아야 할 모든 사항에서 확인할 수 있습니다.

어떤 Kubernetes 워크로드가 Docker 소켓 볼륨을 탑재하는지 확인하는 데 사용할 수 있는 `kubectl` 플러그인이 있습니다. 자세한 내용을 알아보려면 GitHub의 Detector for Docker Socket(DDS)을 참조하세요. `1.24` 이전 버전의 Kubernetes를 실행하는 Amazon EKS AMI에서는 기본 런타임으로 Docker를 사용합니다. 그러나 이러한 Amazon EKS AMI에는 `containerd`을 사용하여 지원되는 클러스터의

워크로드를 테스트하는 데 사용할 수 있는 부트스트랩 플래그 옵션이 있습니다. 자세한 내용은 Docker에서 containerd로 마이그레이션 테스트 섹션을 참조하세요.

지원 날짜가 끝날 때까지 기존 Kubernetes 버전에 대한 AMI를 계속 게시할 예정입니다. 자세한 내용은 Amazon EKS Kubernetes 릴리스 일정 섹션을 참조하세요. containerd에서 워크로드를 테스트하는 시간이 더 필요하면 1.24 이전의 지원되는 버전을 사용하세요. 그러나 공식 Amazon EKS AMI를 1.24 또는 이후 버전으로 업그레이드하려면 워크로드가 containerd에서 실행되는지 검증하세요.

containerd 런타임에서는 신뢰성과 보안이 향상됩니다.containerd는 Amazon EKS 전체에서 표준화되고 있는 런타임입니다. Fargate와 Bottlerocket에서는 이미 containerd만 사용합니다. containerd는 Dockershim CVE(일반적인 취약성 및 노출)를 해결하는 데 필요한 Amazon EKS AMI 릴리스 수 최소화에 도움이 됩니다. Dockershim에서는 이미 내부적으로 containerd를 사용하고 있으므로 변경할 필요가 없을 수도 있습니다. 그러나 변경이 필요할 수 있거나 필요해야 하는 몇 가지 상황이 있습니다.

- Docker 소켓을 탑재하는 애플리케이션을 변경해야 합니다. 예를 들어, 컨테이너로 빌드하는 컨테이너 이미지가 영향을 받습니다. 많은 모니터링 도구도 Docker 소켓을 탑재합니다. 업데이트를 기다리거나 런타임 모니터링을 위해 워크로드를 재배포해야 할 수도 있습니다.
- 특정 Docker 설정에 의존하는 애플리케이션을 변경해야 할 수도 있습니다. 예를 들어, HTTPS_PROXY 프로토콜은 이제 지원되지 않습니다. 이 프로토콜을 사용하는 애플리케이션을 업데이트해야 합니다. 자세한 내용은 Docker Docs의 dockerd 섹션을 참조하세요.
- Amazon ECR 자격 증명 헬퍼를 사용하여 이미지를 가져오는 경우 kubelet 이미지 보안 인증 제공업체로 전환해야 합니다. 자세한 내용은 Kubernetes 설명서의 kubelet 이미지 보안 인증 제공업체 구성 섹션을 참조하세요.
- Amazon EKS 1.24에서 더는 Docker를 지원하지 않으므로 Amazon EKS 부트스트랩 스크립트에서 이전에 지원되던 일부 플래그가 더는 지원되지 않습니다. Amazon EKS 1.24 또는 이후 버전으로 이동하기 전에 지금 지원되지 않는 플래그에 대한 참조를 모두 제거해야 합니다.
  - --container-runtime dockerd(containerd가 지원되는 유일한 값임)
  - --enable-docker-bridge
  - --docker-config-json
- Container Insights에 대해 Fluentd를 이미 구성한 경우 containerd로 변경하기 전에 Fluentd를 Fluent Bit로 마이그레이션해야 합니다. Fluentd 구문 분석기는 JSON 형식의 로그 메시지만 구문 분석하도록 구성됩니다. dockerd와 달리 containerd 컨테이너 런타임에는 JSON 형식이 아닌 로그 메시지가 있습니다. Fluent Bit로 마이그레이션하지 않으면 구성된 Fluentd's 구문 분석기 중 일부가 Fluentd 컨테이너 내에서 엄청난 양의 오류를 생성합니다. 마이그레이션에 대한 자세한 내용은 Set up Fluent Bit as a DaemonSet to send logs to CloudWatch Logs를 참조하세요.

- 사용자 지정 AMI를 사용하고 Amazon EKS `1.24`로 업그레이드하는 경우 워커 노드에 대해 IP 전달이 활성화되어 있는지 확인해야 합니다. 이 설정은 Docker에 필요하지 않았지만 `containerd`에 필요합니다. 이는 Pod와 Pod, Pod와 외부 또는 Pod와 apiserver 간 네트워크 연결 문제를 해결하는 데 필요합니다.

  워커 노드에서 이 설정을 확인하려면 다음 명령 중 하나를 실행합니다.

  - `sysctl net.ipv4.ip_forward`
  - `cat /proc/sys/net/ipv4/ip_forward`

  출력이 `0`이면 다음 명령 중 하나를 실행하여 `net.ipv4.ip_forward` 커널 변수를 활성화합니다.

  - `sysctl -w net.ipv4.ip_forward=1`
  - `echo 1 > /proc/sys/net/ipv4/ip_forward`

  `containerd` 런타임의 Amazon EKS AMI에 대한 설정 활성화는 GitHub의 `install-worker.sh` 섹션을 참조하세요.

Amazon EKS 최적화 Amazon Linux AMI

Amazon EKS 최적화 Amazon Linux AMI는 Amazon Linux 2(AL2) 및 Amazon Linux 2023(AL2023) 기반으로 빌드됩니다. Amazon EKS 노드의 기본 이미지 역할을 하도록 구성되었습니다. AMI는 Amazon EKS와 연동하도록 구성되며 다음과 같은 구성 요소가 포함됩니다.

- `kubelet`
- AWS IAM 인증자
- Docker(Amazon EKS 버전 `1.23` 이하)
- `containerd`

> **Note**
>
> - Amazon Linux 보안 센터에서 AL2의 보안 또는 프라이버시 이벤트를 추적하거나 관련 RSS 피드를 구독할 수 있습니다. 보안 및 프라이버시 이벤트에는 문제의 개요, 영향을 받는 패키지 및 인스턴스를 업데이트하여 문제를 해결하는 방법이 포함됩니다.
> - 가속 또는 Arm AMI를 배포하기 전에 Amazon EKS 최적화 가속 Amazon Linux AMI 및 Amazon EKS 최적화 Arm Amazon Linux AMI의 정보를 검토하세요.

- Kubernetes 버전 `1.23`의 경우 선택적 부트스트랩 플래그를 사용하여 Docker에서 `containerd`로의 마이그레이션을 테스트할 수 있습니다. 자세한 내용은 Docker에서 `containerd`로 마이그레이션 테스트 단원을 참조하십시오.
- Kubernetes 버전 `1.25`부터 Amazon EKS 최적화 가속 Amazon Linux AMI와 함께 즉시 사용 가능한 Amazon EC2 P2 인스턴스를 더 이상 사용할 수 없습니다. Kubernetes 버전 `1.25` 이상의 이러한 AMI는 P2 인스턴스와 호환되지 않는 `NVIDIA 525` 이후 시리즈의 드라이버를 지원합니다. 하지만 `NVIDIA 525` 이후 시리즈의 드라이버는 P3, P4, 및 P5 인스턴스와 호환되므로 이러한 인스턴스를 Kubernetes 버전 `1.25` 이상의 AMI와 함께 사용할 수 있습니다. Amazon EKS 클러스터를 버전 `1.25`로 업그레이드하기 전에 P2 인스턴스를 P3, P4 및 P5 인스턴스로 마이그레이션하세요. 또한, `NVIDIA 525` 이후 시리즈에서 작동하려면 애플리케이션을 사전에 업그레이드해야 합니다. 2024년 1월 말에 최신 `NVIDIA 525` 시리즈 이상의 드라이버를 Kubernetes 버전 `1.23` 및 `1.24`로 백포팅할 계획입니다.
- Amazon EKS 버전 `1.30` 이상부터 새로 생성되는 모든 관리형 노드 그룹은 자동으로 AL2023을 기본 노드 운영 체제로 사용합니다. 이전에는 새 노드 그룹이 AL2를 기본 노드 운영 체제로 사용했습니다. 새 노드 그룹을 생성할 때 AL2를 AMI 유형으로 선택하여 계속 사용할 수 있습니다.
- AL2 지원은 2025년 6월 30일에 종료됩니다. 자세한 내용은 Amazon Linux 2 FAQs를 참조하십시오.

AL2에서 AL2023으로 업그레이드

Amazon EKS 최적화 AMI는 AL2와 AL2023 기반으로 제공됩니다. AL2023은 클라우드 애플리케이션에 안전하고 안정적이면서 고성능의 환경을 제공하도록 설계된 새로운 Linux 기반 운영 체제입니다. Amazon Web Services의 차세대 Amazon Linux로, 확장 지원 버전 `1.23` 및 `1.24`를 포함하여 지원되는 모든 Amazon EKS 버전에서 사용할 수 있습니다. AL2023 기반의 Amazon EKS 가속 AMI는 향후에 제공될 예정입니다. 워크로드를 가속화하면 AL2 가속 AMI 또는 Bottlerocket을 계속 사용해야 합니다.

AL2023은 AL2에 비해 몇 가지 향상된 기능을 제공합니다. 전체 비교는 Amazon Linux 2023 사용 설명서의 AL2와 Amazon Linux 2023 비교를 참조하세요. AL2에서 여러 패키지가 추가, 업그레이드 및 제거되었습니다. 업그레이드하기 전에 AL2023 버전으로 애플리케이션을 테스트하는 것이 좋습니다. AL2023 패키지의 모든 변경 사항 목록은 Amazon Linux 2023 릴리스 정보의 Amazon Linux 2023의 패키지 변경 사항을 참조하세요.

이러한 변경 사항 외에도 다음 사항을 숙지해야 합니다.

- AL2023에 YAML 구성 스키마를 사용하는 새로운 노드 초기화 프로세스 nodeadm이 도입됩니다. 자체 관리형 노드 그룹 또는 시작 템플릿이 있는 AMI를 사용하는 경우 이제 새 노드 그룹을 생성할 때 추가 클러스터 메타데이터를 명시적으로 제공해야 합니다. 최소 필수 파라미터의 예시는 다음과 같으며, 여기에서 apiServerEndpoint, certificateAuthority 및 서비스 cidr가 필수입니다.

```
---
apiVersion: node.eks.aws/v1alpha1
kind: NodeConfig
spec:
  cluster:
    name: my-cluster
    apiServerEndpoint: https://example.com
    certificateAuthority: Y2VydGlmaWNhdGVBdXRob3JpdHk=
    cidr: 10.100.0.0/16
```

  AL2에서 이러한 파라미터의 메타데이터는 Amazon EKS DescribeCluster API 직접 호출에서 확인되었습니다. AL2023 버전에서는 대형 노드 스케일 업 도중 추가 API 직접 호출로 인해 제한이 발생할 위험이 있기 때문에 이러한 동작이 변경되었습니다. 시작 템플릿이 없는 관리형 노드 그룹을 사용하거나 Karpenter를 사용하는 경우 이 변경 사항이 영향을 미치지 않습니다. certificateAuthority 및 서비스 cidr에 대한 자세한 내용은 Amazon EKS API 참조의 DescribeCluster를 참조하세요.
- Docker는 지원되는 모든 Amazon EKS 버전에 대해 AL2023 버전에서 지원되지 않습니다. AL2의 Amazon EKS 버전 1.24 이상에서 Docker 지원이 종료 및 제거되었습니다. 지원 중단에 대한 자세한 내용은 Dockershim에 대한 Amazon EKS 지원 종료를 참조하세요.
- AL2023의 경우 Amazon VPC CNI 버전 1.16.2 이상이 필요합니다.
- AL2023의 필수 기본값은 IMDSv2입니다. IMDSv2에는 보안 태세 개선에 도움이 되는 몇 가지 이점이 있습니다. 세션을 시작하려면 간단한 HTTP PUT 요청으로 비밀 토큰을 생성해야 하는 세션 지향 인증 방법을 사용합니다. 세션의 토큰은 1초에서 6시간 사이로 유효할 수 있습니다. IMDSv1에서 IMDSv2로 전환하는 방법에 대한 자세한 내용은 인스턴스 메타데이터 서비스 버전 2 사용으로 전환 및 Get the full benefits of IMDSv2 and disable IMDSv1 across your AWS infrastructure를 참조하세요. IMDSv1을 사용하려는 경우 인스턴스 메타데이터 옵션 시작 속성을 사용하여 설정을 수동으로 재정의하면 계속 사용할 수 있습니다.

> **Note**
>
> IMDSv2의 경우 관리형 노드 그룹의 기본 홉 수는 1로 설정되어 있습니다. 따라서 컨테이너는 IMDS를 사용하여 노드의 자격 증명에 액세스할 수 없습니다. 노드의 자격 증명에 대한 컨테이너 액세스가 필요한 경우에도 사용자 지정 Amazon EC2 시작 템플릿에서

`HttpPutResponseHopLimit`를 수동으로 재정의하여 2로 늘리면 됩니다. 아니면 Amazon EKS Pod Identity를 사용하여 `IMDSv2` 대신 자격 증명을 제공할 수도 있습니다.

- AL2023에는 차세대 통합 제어 그룹 계층 구조(`cgroupv2`)가 있습니다. `cgroupv2`는 컨테이너 런타임을 구현하는 데 사용되며, `systemd`에 따라 사용됩니다. AL2023에 시스템이 `cgroupv1`을 사용하여 실행하도록 할 수 있는 코드가 포함되어 있지만 이 구성은 권장되거나 지원되지 않습니다. 이 구성은 Amazon Linux의 향후 메이저 릴리스에서 완전히 제거될 예정입니다.

기존 관리형 노드 그룹의 경우 시작 템플릿을 사용하는 방식에 따라 인플레이스 업그레이드 또는 블루/그린 업그레이드를 수행할 수 있습니다.

- 관리형 노드 그룹에서 사용자 지정 AMI를 사용하는 경우 시작 템플릿에서 AMI ID를 교체하여 인플레이스 업그레이드를 수행할 수 있습니다. 이 업그레이드 전략을 수행하기 전에 먼저 애플리케이션과 사용자 데이터가 AL2023으로 전송되는지 확인해야 합니다.
- 표준 시작 템플릿 또는 AMI ID를 지정하지 않은 사용자 지정 시작 템플릿에서 관리형 노드 그룹을 사용하는 경우 블루/그린 전략을 사용하여 업그레이드해야 합니다. 블루/그린 업그레이드는 대체로 더 복잡하고 AMI 유형으로 AL2023을 지정하는 완전히 새로운 노드 그룹을 생성해야 합니다. 이후 AL2 노드 그룹의 모든 사용자 지정 데이터가 새 OS와 호환되도록 새 노드 그룹을 신중하게 구성해야 합니다. 애플리케이션에서 새 노드 그룹을 테스트하고 검증한 후에는 이전 노드 그룹에서 새 노드 그룹으로 Pods를 마이그레이션할 수 있습니다. 마이그레이션이 완료되면 이전 노드 그룹을 삭제할 수 있습니다.

Karpenter를 사용 중이고 AL2023을 사용하려는 경우 `EC2NodeClass` `amiFamily` 필드를 AL2023으로 수정해야 합니다. 기본적으로 드리프트는 Karpenter에서 활성화됩니다. 따라서 `amiFamily` 필드가 변경되면 Karpenter에서 사용 가능할 때 워커 노드를 최신 AMI로 자동으로 업데이트합니다.

Amazon EKS 최적화 가속 Amazon Linux AMI

Note

AL2023 기반의 Amazon EKS 가속 AMI는 향후에 제공될 예정입니다. 워크로드를 가속화하면 AL2 가속 AMI 또는 Bottlerocket을 계속 사용해야 합니다.

Amazon EKS 최적화 가속 Amazon Linux AMI는 표준 Amazon EKS 최적화 Amazon Linux AMI를 기반으로 구축되었습니다. 이는 GPU, Inferentia 및 Trainium 기반 워크로드를 지원하기 위해 Amazon EKS 노드에 대한 선택적 이미지 역할을 하도록 구성되었습니다.

표준 Amazon EKS 최적화 AMI 구성 외에도 가속 AMI에는 다음이 포함됩니다.

- NVIDIA 드라이버
- `nvidia-container-runtime`(기본 실행 시간으로)
- AWS Neuron 컨테이너 런타임

가속 AMI에 포함된 최신 구성 요소 목록은 GitHub의 `amazon-eks-ami` 릴리스를 참조하세요.

> **Note**
>
> - Amazon EKS 최적화 가속 AMI는 GPU 및 Inferentia 기반 인스턴스 유형만 지원합니다. 노드 AWS CloudFormation 템플릿에서 이러한 인스턴스 유형을 지정해야 합니다. Amazon EKS 최적화 가속 AMI를 사용하게 되면 NVIDIA의 EULA(사용자 라이선스 계약)에 동의하게 됩니다.
> - 이전에는 Amazon EKS 최적화 가속 AMI를 GPU를 지원하는 Amazon EKS 최적화된 AMI라고 불렀습니다.
> - Amazon EKS 최적화 가속 AMI의 이전 버전에서는 `nvidia-docker` 리포지토리를 설치했습니다. Amazon EKS AMI 버전 `v20200529` 이상에는 더 이상 이 리포지토리가 포함되지 않습니다.

GPU 기반 워크로드를 활성화하려면

다음 절차에서는 Amazon EKS 최적화 가속 AMI를 사용하여 GPU 기반 인스턴스에서 워크로드를 실행하는 방법에 대해 설명합니다. 다른 옵션은 다음을 참조하세요.

- Inferentia 기반 워크로드 사용에 대한 자세한 내용을 알아보려면 AWS Inferentia를 사용한 기계 학습 추론 부분을 참조하세요.
- Neuron 사용에 관한 추가 정보는 AWS Neuron 설명서의 Containers - Kubernetes - Getting Started를 참조하세요.

1. GPU 노드가 클러스터에 조인하면 클러스터에서 Kubernetes용 NVIDIA 디바이스 플러그 인을 DaemonSet(으)로 적용해야 합니다. 다음 명령을 실행하기 전에 *vX.X.X*을(를) 원하는 NVIDIA/k8s-device-plugin 버전으로 교체합니다.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/vX.X.X/
nvidia-device-plugin.yml
```

2. 다음 명령으로 노드에 할당 가능한 GPU가 있는지 확인할 수 있습니다.

```
kubectl get nodes "-o=custom-
columns=NAME:.metadata.name,GPU:.status.allocatable.nvidia\.com/gpu"
```

GPU 노드의 구성이 올바른지 테스트하기 위한 Pod 배포하기

1. 다음 콘텐츠를 가진 nvidia-smi.yaml이라는 파일을 생성합니다: *tag*을(를) 원하는 nvidia/cuda용 태그와 바꿉니다. 이 매니페스트는 노드에서 nvidia-smi을(를) 실행하는 NVIDIA CUDA 컨테이너를 시작합니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: nvidia/cuda:tag
    args:
    - "nvidia-smi"
    resources:
      limits:
        nvidia.com/gpu: 1
```

2. 다음 명령으로 매니페스트를 적용합니다.

```
kubectl apply -f nvidia-smi.yaml
```

3. Pod 실행이 끝난 후, 다음 명령을 사용하여 로그를 확인합니다.

```
kubectl logs nvidia-smi
```

예제 출력은 다음과 같습니다.

```
Mon Aug  6 20:23:31 20XX
+-----------------------------------------------------------------------------+
| NVIDIA-SMI XXX.XX                 Driver Version: XXX.XX                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:1C.0 Off |                    0 |
| N/A   46C    P0    47W / 300W |      0MiB / 16160MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Amazon EKS 최적화 Arm Amazon Linux AMI

Arm 인스턴스는 웹 서버, 컨테이너식 마이크로서비스, 캐싱 플릿 및 분산 데이터 스토어와 같은 스케일 아웃 및 Arm 기반 애플리케이션에 상당한 비용 절감 효과를 제공합니다. 클러스터에 Arm 노드를 추가하는 경우 다음 고려 사항을 검토하세요.

고려 사항

- 클러스터가 2020년 8월 17일 이전에 배포된 경우 중요한 클러스터 추가 기능 매니페스트의 일회성 업그레이드를 수행해야 합니다. 이는 Kubernetes가 클러스터에서 사용 중인 각 하드웨어 아키텍처에 대해 올바른 이미지를 가져올 수 있도록 하기 위함입니다. 클러스터 추가 기능을 업데이트하는 방법에 대한 자세한 내용은 Amazon EKS 클러스터의 Kubernetes 버전 업데이트 부분을 참조하세요. 2020년 8월 17일 또는 이후에 클러스터를 배포한 경우는 이미 CoreDNS, `kube-proxy` 및 Amazon VPC CNI plugin for Kubernetes가 다중 아키텍처를 지원합니다.
- Arm 노드에 배포된 애플리케이션은 Arm용으로 컴파일되어야 합니다.
- 기존 클러스터에 배포된 DaemonSets이 있거나, Arm 노드도 배포할 새 클러스터에 DaemonSet를 배포하려는 경우 DaemonSet가 클러스터의 모든 하드웨어 아키텍처에서 실행될 수 있는지 확인합니다.
- 동일한 클러스터에서 Arm 노드 그룹과 x86 노드 그룹을 실행할 수 있습니다. 이 경우에 다중 아키텍처 컨테이너 이미지를 Amazon Elastic Container Registry와 같은 컨테이너 리포지토리에 배포한 다

음, 매니페스트에 노드 선택기를 추가하여 어떤 하드웨어에 Pod를 배포할 수 있는지 Kubernetes가 알 수 있도록 해야 합니다. 자세한 내용은 Amazon ECR 사용 설명서의 다중 아키텍처 이미지 푸시 및 Amazon ECR용 다중 아키텍처 컨테이너 이미지 소개 블로그 게시물을 참조하세요.

Docker에서 **containerd**로 마이그레이션 테스트

Amazon EKS에서는 Kubernetes 버전 `1.24` 출시부터 Docker에 대한 지원이 종료되었습니다. 자세한 내용은 Amazon EKS에서는 Dockershim에 대한 지원을 종료했습니다. 단원을 참조하십시오.

Kubernetes 버전 `1.23`의 경우 선택적 부트스트랩 플래그를 사용하여 Amazon EKS 최적화 AL2 AMI에 대해 `containerd` 런타임을 활성화할 수 있습니다. 이 기능에서는 버전 `1.24` 또는 이후 버전으로 업데이트할 때 `containerd`로 마이그레이션하는 명확한 경로를 제공합니다. Amazon EKS에서는 Kubernetes 버전 `1.24` 출시부터 Docker에 대한 지원이 종료되었습니다. `containerd` 런타임은 Kubernetes 커뮤니티에서 널리 채택되었으며 CNCF의 마무리 단계를 통과한 프로젝트입니다. 노드 그룹을 새 클러스터나 기존 클러스터에 추가하여 테스트할 수 있습니다.

다음 노드 그룹 유형 중 하나를 생성하여 부트스트랩 플래그를 사용할 수 있습니다.

자체 관리형

자체 관리형 Amazon Linux 노드 시작하기의 지침에 따라 노드 그룹을 생성합니다. Amazon EKS 최적화 AMI를 지정하고 `BootstrapArguments` 파라미터에 다음 텍스트를 지정합니다.

```
--container-runtime containerd
```

관리형

`eksctl`을 사용하는 경우 다음 내용이 포함된 *`my-nodegroup`*`.yaml`이라는 파일을 생성합니다. 모든 *`example value`*를 고유한 값으로 바꿉니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다. `ami-`*`1234567890abcdef0`*에 대해 최적화된 AMI ID를 검색하려면 Amazon EKS 최적화 Amazon Linux AMI ID 검색 섹션을 참조하세요.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code
  version: 1.23
managedNodeGroups:
```

```
  - name: my-nodegroup
    ami: ami-1234567890abcdef0
    overrideBootstrapCommand: |
      #!/bin/bash
      /etc/eks/bootstrap.sh my-cluster --container-runtime containerd
```

> **Note**
>
> 많은 노드를 동시에 시작하는 경우 `--apiserver-endpoint`, `--b64-cluster-ca` 및 `--dns-cluster-ip` 부트스트랩 인수에 대한 값을 지정하여 오류를 방지하려 할 수도 있습니다. 자세한 내용은 AMI 지정 단원을 참조하십시오.

다음 명령을 실행하여 노드 그룹을 생성합니다.

```
eksctl create nodegroup -f my-nodegroup.yaml
```

다른 도구를 사용하여 관리형 노드 그룹을 생성하려면 시작 템플릿을 사용하여 노드 그룹을 배포해야 합니다. 시작 템플릿에서 Amazon EKS 최적화 AMI ID를 지정한 다음 시작 템플릿을 사용하여 노드 그룹 배포하고 다음 사용자 데이터를 제공합니다. 이 사용자 데이터는 인수를 `bootstrap.sh` 파일에 전달합니다. 부트스트랩 파일에 대한 자세한 내용은 GitHub에서 bootstrap.sh 부분을 참조하세요.

```
/etc/eks/bootstrap.sh my-cluster --container-runtime containerd
```

추가 정보

Amazon EKS 최적화 Amazon Linux AMI 사용에 대한 자세한 내용은 다음 섹션을 참조하세요.

- 관리형 노드 그룹과 함께 Amazon Linux를 사용하려면 관리형 노드 그룹 섹션을 참조하세요.
- 자체 관리형 Amazon Linux 노드를 시작하려면 Amazon EKS 최적화 Amazon Linux AMI ID 검색 섹션을 참조하세요.
- 버전 정보는 Amazon EKS 최적화 Amazon Linux AMI 버전를 참조하십시오.
- Amazon EKS 최적화 Amazon Linux AMI의 최신 ID를 검색하려면 Amazon EKS 최적화 Amazon Linux AMI ID 검색 섹션을 참조하세요.
- Amazon EKS 최적화 AMI를 빌드하는 데 사용되는 오픈 소스 스크립트는 Amazon EKS 최적화 Amazon Linux AMI 빌드 스크립트 섹션을 참조하세요.

Amazon EKS 최적화 Amazon Linux AMI 버전

Amazon EKS 최적화 Amazon Linux AMI는 Kubernetes 버전과 AMI의 출시 날짜에 따라 다음 형식으로 버전이 지정됩니다.

```
k8s_major_version.k8s_minor_version.k8s_patch_version-release_date
```

각 AMI 릴리스에는 다양한 버전의 `kubelet`, Docker, Linux 커널 및 `containerd`가 포함되어 있습니다. 가속 AMI에는 다양한 버전의 NVIDIA 드라이버도 포함되어 있습니다. 이 버전 정보는 GitHub의 Changelog에서 확인할 수 있습니다.

Amazon EKS 최적화 Amazon Linux AMI ID 검색

AWS Systems Manager 파라미터 스토어 API를 쿼리하여 Amazon EKS 최적화 AMI의 Amazon Machine Image(AMI) ID를 프로그래밍 방식으로 검색할 수 있습니다. 이 파라미터를 사용하면 Amazon EKS 최적화 AMI ID를 수동으로 조회할 필요가 없습니다. Systems Manager 파라미터 스토어 API에 대한 자세한 내용은 GetParameter 섹션을 참조하세요.

AWS CLI를 사용하여 Amazon EKS 최적화 AMI의 AMI ID 검색

1. `us-east-1`과 같이 노드 인스턴스가 배포될 리전을 결정합니다.
2. 필요한 AMI의 유형을 결정합니다. Amazon EC2 인스턴스의 유형에 대한 자세한 내용은 인스턴스 유형을 참조하세요.
   - `amazon-linux-2`는 Amazon Linux 2(AL2) x86 기반 인스턴스용입니다.
   - `amazon-linux-2-arm64`는 AWS Graviton 기반 인스턴스와 같은 AL2 ARM 인스턴스용입니다.
   - `amazon-linux-2-gpu`는 AL2 GPU 가속 인스턴스용입니다.
   - `amazon-linux-2023/x86_64/standard`는 Amazon Linux 2023(AL2023) x86 기반 인스턴스용입니다.
   - `amazon-linux-2023/arm64/standard`는 AL2023 ARM 인스턴스용입니다.
3. 노드가 연결될 클러스터의 Kubernetes 버전(예: 1.29)을 결정합니다.
4. 다음 AWS CLI 명령을 실행하여 적절한 AMI ID를 검색합니다. AWS 리전, Kubernetes 버전과 플랫폼을 적절하게 바꿉니다. Amazon EKS 최적화 AMI 메타데이터를 검색하려면 `ssm:GetParameter` IAM 권한이 있는 IAM 보안 주체를 사용하여 AWS CLI에 로그인해야 합니다.

```
aws ssm get-parameter --name /aws/service/eks/optimized-ami/1.29/amazon-linux-2/
recommended/image_id \
                --region region-code --query "Parameter.Value" --output text
```

예제 출력은 다음과 같습니다.

```
ami-1234567890abcdef0
```

Amazon EKS 최적화 Amazon Linux AMI 빌드 스크립트

Amazon Elastic Kubernetes Service(Amazon EKS)에는 Amazon EKS 최적화 AMI를 빌드하는 데 사용되는 오픈 소스 스크립트가 있습니다. 이러한 빌드 스크립트를 이제 GitHub에서 사용할 수 있습니다.

Amazon EKS 최적화 Amazon Linux AMI는 특히 Amazon EKS 클러스터에서 노드로 사용하기 위해 Amazon Linux 2(AL2) 및 Amazon Linux 2023(AL2023) 기반으로 빌드됩니다. 이 리포지토리를 사용해 Amazon EKS 팀이 `kubelet`, Docker, AWS IAM Authenticator for Kubernetes 등을 구성하고 고유한 Amazon Linux 기반 AMI를 처음부터 빌드하는 방법을 자세히 볼 수 있습니다.

이 빌드 스크립트 리포지토리는 AMI를 생성하기 위한 HashiCorp packer 템플릿 및 빌드 스크립트를 포함합니다. 이러한 스크립트는 Amazon EKS 최적화 AMI 빌드의 단일 출처이므로 GitHub 리포지토리를 따라 AMI에 대한 변경을 모니터링할 수 있습니다. 예를 들어 사용자가 Amazon EKS 팀이 공식 AMI에 사용하는 것과 동일한 버전의 Docker를 자체 AMI에 사용하기를 원할 수 있습니다.

또한 GitHub 리포지토리는 부팅 시 실행되어 인스턴스의 인증 데이터, 컨트롤 플레인 엔드포인트, 클러스터 이름 등을 구성하는 특수 부트스트랩 스크립트와 nodeadm 스크립트도 포함합니다.

또한 GitHub 리포지토리는 Amazon EKS 노드 AWS CloudFormation 템플릿을 포함합니다. 이러한 템플릿을 사용하면 보다 간단하게 Amazon EKS 최적화 AMI를 실행하는 인스턴스를 가동하고 클러스터에 등록할 수 있습니다.

자세한 내용은 GitHub(https://github.com/awslabs/amazon-eks-ami)에서 리포지토리를 참조하세요.

Amazon EKS 최적화 AL2에는 `containerd` 런타임을 사용 설정할 수 있는 선택적 부트스트랩 플래그가 포함되어 있습니다.

사용자 지정 Amazon Linux AMI용 VT1 구성

Amazon EKS의 사용자 지정 Amazon Linux AMI는 Amazon Linux 2(AL2), Ubuntu 18 및 Ubuntu 20용 VT1 비디오 트랜스코딩 인스턴스 패밀리를 지원할 수 있습니다. VT1은 가속 H.264/AVC 및 H.265/

HEVC 코덱으로 Xilinx U30 미디어 트랜스코딩 카드를 지원합니다. 이러한 가속 인스턴스의 이점을 얻으려면 다음 단계를 따라야 합니다.

1. AL2, Ubuntu 18 또는 Ubuntu 20에서 기본 AMI를 생성하고 시작합니다.
2. 기반 AMI가 시작된 후 노드에 XRT 드라이버 및 런타임을 설치합니다.
3. Amazon EKS 클러스터 생성.
4. 클러스터에 Kubernetes FPGA 플러그인을 설치합니다.

```
kubectl apply -f fpga-device-plugin.yml
```

이제 플러그 인은 Amazon EKS 클러스터에 노드당 Xilinx U30 디바이스를 알립니다. FFMPEG Docker 이미지를 사용하여 Amazon EKS 클러스터에서 예제 비디오 트랜스코딩 워크로드를 실행할 수 있습니다.

사용자 지정 Amazon Linux 2 AMI용 DL1 구성

Amazon EKS의 사용자 지정 Amazon Linux 2(AL2) AMI는 추가 구성 및 Kubernetes 추가 기능을 통해 대규모 딥 러닝 워크로드를 지원할 수 있습니다. 이 문서에서는 온프레미스 설정을 위한 일반 Kubernetes 솔루션을 설정하거나 더 큰 클라우드 구성의 기준으로 설정하는 데 필요한 구성 요소에 대해 설명합니다. 이 기능을 지원하려면 사용자 지정 환경에서 다음 단계를 수행해야 합니다.

- 시스템에 로드된 SynapaSeAI® 소프트웨어 드라이버 - 이러한 드라이버는 Github에서 사용 가능한 AMI에 포함되어 있습니다.

  Habana 디바이스 플러그인 - Kubernetes 클러스터에서 Habana 디바이스의 등록을 자동으로 활성화하고 디바이스 상태를 추적할 수 있는 DaemonSet입니다.
- Helm 3.x
- MPI 연산자 설치를 위한 Helm 차트.
- MPI 연산자

1. AL2, Ubuntu 18 또는 Ubuntu 20에서 기본 AMI를 생성하고 시작합니다.
2. 이러한 지침을 따라 DL1용 환경을 설정합니다.

Amazon EKS 최적화 Bottlerocket AMI

Bottlerocket은 AWS에서 후원하고 지원하는 오픈 소스 Linux 배포판입니다. Bottlerocket은 컨테이너 워크로드를 호스팅하기 위해 특별히 제작되었습니다. Bottlerocket을 사용하면 컨테이너 인프라에 대한 업데이트를 자동화하여 컨테이너식 배포의 가용성을 개선하고 운영 비용을 절감할 수 있습니다. Bottlerocket에는 컨테이너를 실행하는 데 필수적인 소프트웨어만 포함되어 있어 리소스 사용량이 개선되고 보안 위협과 관리 오버헤드가 감소합니다. Bottlerocket AMI에는`containerd`, `kubelet`, AWS IAM 인증자가 포함됩니다. 관리형 노드 그룹 및 자체 관리형 노드 외에도 Karpenter는 Bottlerocket을 지원합니다.

장점

Amazon EKS 클러스터와 함께 Bottlerocket을 사용하면 다음과 같은 이점이 있습니다.

- 운영 비용 절감 및 관리 복잡성 감소로 가동 시간 증가 – Bottlerocket은 다른 Linux 배포판보다 리소스 공간이 작고 부팅 시간이 짧으며 보안 위협에 덜 취약합니다. Bottlerocket's은 공간이 작아 스토리지, 컴퓨팅 및 네트워킹 리소스를 적게 사용하여 비용을 절감할 수 있습니다.
- 자동 OS 업데이트로 보안 강화 – Bottlerocket 업데이트는 필요한 경우 롤백할 수 있는 단일 단위로 적용됩니다. 이렇게 하면 시스템을 사용 불가 상태로 둘 수 있는 손상되거나 실패한 업데이트의 위험이 제거됩니다. Bottlerocket을 사용하면 보안 업데이트를 사용할 수 있는 즉시 중단을 최소화하는 방식으로 자동 적용하고 장애 발생 시 롤백할 수 있습니다.
- 프리미엄 서포트 – AWS에서 제공하는 Amazon EC2 기반 Bottlerocket 빌드에는 Amazon EC2, Amazon EKS, Amazon ECR 등의 AWS 서비스에도 적용되는 동일한 AWS Support 플랜이 적용됩니다.

고려 사항

AMI 유형에 Bottlerocket을 사용할 때 다음 사항을 고려하세요.

- Bottlerocket은 `x86_64` 및 `arm64` 프로세서가 있는 Amazon EC2 인스턴스를 지원합니다. Bottlerocket AMI를 Inferentia 칩이 있는 Amazon EC2 인스턴스와 함께 사용하는 것은 권장되지 않습니다.
- 현재 Bottlerocket 노드를 배포하는 데 사용할 수 있는 AWS CloudFormation 템플릿은 없습니다.
- Bottlerocket 이미지에는 SSH 서버 또는 쉘이 포함되지 않습니다. 대역 외 액세스 방법을 사용하여 SSH를 허용할 수 있습니다. 이 접근 방식을 통해 관리 컨테이너를 활성화하고 사용자 데이터와 함께 일부 부트스트래핑 구성 단계를 전달할 수 있습니다. 자세한 내용은 GitHub의 Bottlerocket OS에서 다음 섹션을 참조하세요.

  - 탐색
  - 관리자 컨테이너
  - Kubernetes 설정
- Bottlerocket은 다양한 컨테이너 유형을 사용합니다.
  - 기본적으로 제어 컨테이너는 활성화되어 있습니다. 이 컨테이너는 Amazon EC2 Bottlerocket 인스턴스에서 명령을 실행하거나 셸 세션을 시작하는 데 사용할 수 있는 AWS Systems Manager 에이전트를 실행합니다. 자세한 내용은 AWS Systems Manager 사용 설명서의 세션 관리자 설정을 참조하세요.
  - 노드 그룹을 생성할 때 SSH 키가 제공되는 경우 관리 컨테이너가 활성화됩니다. 개발 및 테스트 시나리오에만 관리 컨테이너를 사용하는 것이 좋습니다. 프로덕션 환경에서는 이를 사용하지 않는 것이 좋습니다. 자세한 내용은 GitHub의 관리자 컨테이너를 참조하세요.

추가 정보

Amazon EKS 최적화 Bottlerocket AMI 사용에 대한 자세한 내용은 다음 섹션을 참조하세요.

- Bottlerocket에 대한 자세한 내용은 GitHub의 설명서와 릴리스를 참조하세요.
- 관리형 노드 그룹과 함께 Bottlerocket을 사용하려면 관리형 노드 그룹 섹션을 참조하세요.
- 자체 관리형 Bottlerocket 노드를 시작하려면 자체 관리형 Bottlerocket 노드 시작 섹션을 참조하세요.
- Amazon EKS 최적화 Bottlerocket AMI의 최신 ID를 검색하려면 Amazon EKS 최적화 Bottlerocket AMI ID 검색 섹션을 참조하세요.
- 규정 준수 지원에 대한 자세한 내용은 Bottlerocket 규정 준수 지원 섹션을 참조하세요.

Amazon EKS 최적화 Bottlerocket AMI ID 검색

AWS Systems Manager 파라미터 스토어 API를 쿼리하여 Amazon EKS 최적화 AMI의 Amazon Machine Image(AMI) ID를 검색할 수 있습니다. 이 파라미터를 사용하면 Amazon EKS 최적화 AMI ID를 수동으로 조회할 필요가 없습니다. Systems Manager 파라미터 스토어 API에 대한 자세한 내용은 GetParameter 섹션을 참조하세요. Amazon EKS 최적화 AMI 메타데이터를 검색하려면 IAM 보안 주체에 `ssm:GetParameter` IAM 권한이 있어야 합니다.

다음 AWS CLI 명령에서 하위 파라미터 `image_id`를 사용하여 권장되는 최신 Amazon EKS 최적화 Bottlerocket AMI의 이미지 ID를 검색할 수 있습니다. *`1.29`*를 지원 버전으로 바꾸고 *`region-code`*를 AMI ID가 필요한 Amazon EKS 지원 리전으로 바꿉니다.

```
aws ssm get-parameter --name /aws/service/bottlerocket/aws-k8s-1.29/x86_64/latest/
image_id --region region-code --query "Parameter.Value" --output text
```

예제 출력은 다음과 같습니다.

```
ami-1234567890abcdef0
```

Bottlerocket 규정 준수 지원

Bottlerocket은 다양한 조직에서 정의한 권장 사항을 준수합니다.

- Bottlerocket에 대해 정의된 CIS Benchmark가 있습니다. 기본 구성에서 Bottlerocket 이미지에는 CIS 레벨 1 구성 프로파일에 필요한 대부분의 컨트롤이 있습니다. CIS 레벨 2 구성 프로파일에 필요한 컨트롤을 구현할 수 있습니다. 자세한 내용은 AWS 블로그의 CIS Benchmark에 대해 Amazon EKS 최적화 Bottlerocket AMI 검증을 참조하세요.
- 최적화된 기능 세트와 감소된 공격 표면은 Bottlerocket 인스턴스가 PCI DSS 요구 사항을 충족하는 데 필요한 구성이 더 적다는 것을 의미합니다. Bottlerocket용 CIS Benchmark는 지침 강화를 위한 훌륭한 리소스이며 PCI DSS 요구 사항 2.2에 따른 보안 구성 표준에 대한 요구 사항을 지원합니다. 또한 Fluent Bit를 활용하여 PCI DSS 요구 사항 10.2에 따라 운영 체제 수준 감사 로깅에 대한 요구 사항을 지원할 수 있습니다. AWS는 PCI DSS 요구 사항 6.2(v3.2.1의 경우) 및 요구 사항 6.3.3(v4.0의 경우)을 충족할 수 있도록 새로운(패치된) Bottlerocket 인스턴스를 주기적으로 게시합니다.
- Bottlerocket은 Amazon EC2 및 Amazon EKS 모두에 대해 규제된 워크로드와 함께 사용하도록 승인된 HIPAA 적격 기능입니다. 자세한 내용은 Amazon EKS에서 HIPAA 보안 및 규정 준수를 위한 설계 백서를 참조하세요.

Amazon EKS 최적화 Ubuntu Linux AMI

Canonical은 Amazon EKS와 협력하여 클러스터에서 사용할 수 있는 노드 AMI를 만들었습니다.

Canonical은 특별히 구축된 Kubernetes Node OS 이미지를 제공합니다. 이 최소화된 Ubuntu 이미지는 Amazon EKS에 최적화되어 있으며 AWS와 공동 개발한 사용자 지정 AWS 커널을 포함합니다. 자세한 내용은 Amazon Elastic Kubernetes Service(EKS)의 Ubuntu를 참조하세요. 지원에 대한 자세한 내용은 AWS 프리미엄 서포트 FAQ의 서드 파티 소프트웨어 섹션을 참조하세요.

Amazon EKS 최적화 Windows AMI

Windows Amazon EKS 최적화 AMI는 Windows Server 2019 및 Windows Server 2022를 기반으로 빌드되었습니다. Amazon EKS 노드의 기본 이미지 역할을 하도록 구성되었습니다. 기본적으로 AMI에는 다음 구성 요소가 포함됩니다.

- kubelet
- kube-proxy
- Kubernetes용 AWS IAM Authenticator
- csi-proxy
- containerd

> **Note**
>
> Microsoft 보안 업데이트 가이드를 사용하여 Windows Server의 보안 또는 개인 정보 보호 이벤트를 추적할 수 있습니다.

Amazon EKS는 Windows 컨테이너에 최적화된 AMI를 다음 변형된 형태로 제공합니다.

- Amazon EKS 최적화 Windows Server 2019 Core AMI
- Amazon EKS 최적화 Windows Server 2019 Full AMI
- Amazon EKS 최적화 Windows Server 2022 Core AMI
- Amazon EKS 최적화 Windows Server 2022 Full AMI

> **Important**
>
> - Amazon EKS 최적화 Windows Server 20H2 Core AMI는 사용되지 않습니다. 이 AMI의 새 버전은 릴리스되지 않습니다.
> - 항상 최신 보안 업데이트를 기본으로 유지할 수 있도록 Amazon EKS는 지난 4개월 동안 최적화 Windows AMI를 유지합니다. 각각의 새 AMI는 최초 릴리스 시점부터 4개월 동안 사용할 수 있습니다. 이 기간이 지나면 오래된 AMI는 프라이빗으로 전환되어 더 이상 액세스할 수 없습니다. 보안 취약성을 방지하고 지원 수명이 다한 오래된 AMI에 대한 액세스를 상실하

지 않도록 최신 AMI를 사용하는 것이 좋습니다. 프라이빗으로 설정된 AMI에 대한 액세스 제공을 보장할 수는 없지만 AWS Support에 티켓을 제출하여 액세스를 요청할 수 있습니다.

출시 일정

다음 표에는 Amazon EKS의 Windows 버전 릴리스 및 지원 종료 날짜가 나와 있습니다. 종료 날짜가 비어 있으면 버전이 계속 지원되는 것입니다.

Windows 버전	Amazon EKS 릴리스	Amazon EKS 지원 종료
Windows Server 2022 Core	10/17/2022	
Windows Server 2022 Full	10/17/2022	
Windows Server 20H2 Core	8/12/2021	8/9/2022
Windows Server 2004 Core	8/19/2020	12/14/2021
Windows Server 2019 Core	10/7/2019	
Windows Server 2019 Full	10/7/2019	
Windows Server 1909 Core	10/7/2019	12/8/2020

부트스트랩 스크립트 구성 파라미터

Windows 노드를 생성하는 경우 노드에 다른 파라미터를 구성할 수 있는 스크립트가 있습니다. 설정에 따라 이 스크립트는 `C:\Program Files\Amazon\EKS\Start-EKSBootstrap.ps1`과 유사한 위치의 노드에서 찾을 수 있습니다. 사용자 지정 파라미터 값을 부트스트랩 스크립트의 인수로 지정하여 설정할 수 있습니다. 예를 들어, 시작 템플릿에서 사용자 데이터를 업데이트할 수 있습니다. 자세한 내용은 Amazon EC2 사용자 데이터 단원을 참조하십시오.

이 스크립트에는 다음 명령줄 파라미터가 포함되어 있습니다.

- `-EKSClusterName` - 조인할 이 작업자 노드의 Amazon EKS 클러스터 이름을 지정합니다.
- `-KubeletExtraArgs` - `kubelet`에 대한 추가 인수를 지정합니다(선택 사항).
- `-KubeProxyExtraArgs` - `kube-proxy`에 대한 추가 인수를 지정합니다(선택 사항).

- `-APIServerEndpoint` - Amazon EKS 클러스터 API 서버 엔드포인트를 지정합니다(선택 사항). `-Base64ClusterCA`와 함께 사용하는 경우에만 유효합니다. `Get-EKSCluster`.
- `-Base64ClusterCA` - base64로 인코딩된 클러스터 CA 콘텐츠를 지정합니다(선택 사항). `-APIServerEndpoint`와 함께 사용하는 경우에만 유효합니다. `Get-EKSCluster` 호출을 무시합니다.
- `-DNSClusterIP` - 클러스터 내의 DNS 쿼리에 사용할 IP 주소를 재정의합니다(선택 사항). 기본 인터페이스의 IP 주소에 따라 기본값은 `10.100.0.10` 또는 `172.20.0.10`입니다.
- `-ServiceCIDR` – 클러스터 서비스가 처리되는Kubernetes 서비스 IP 주소 범위를 재정의합니다. 기본 인터페이스의 IP 주소에 따라 기본값은 `172.20.0.0/16` 또는 `10.100.0.0/16`입니다.
- `-ExcludedSnatCIDRs` – SNAT(Source Network Address Translation)에서 제외할 `IPv4` CIDR 목록입니다. 즉, VPC 주소 지정이 가능한 포드 프라이빗 IP는 아웃바운드 트래픽에 대한 인스턴스 ENI의 기본 `IPv4` 주소의 IP 주소로 변환되지 않습니다. 기본적으로 Amazon EKS Windows 노드에 대한 VPC의 `IPv4` CIDR이 추가됩니다. 이 파라미터에 CIDR을 지정하면 지정된 CIDR도 추가로 제외됩니다. 자세한 내용은 Pods용 SNAT 단원을 참조하십시오.

명령줄 파라미터 외에도 일부 환경 변수 파라미터를 지정할 수도 있습니다. 명령줄 파라미터를 지정하는 경우 해당 파라미터가 해당 환경 변수보다 우선시됩니다. 부트스트랩 스크립트는 컴퓨터 범위 변수만 읽으므로 환경 변수는 컴퓨터(또는 시스템) 범위로 정의해야 합니다.

이 스크립트는 다음과 같은 환경 변수를 고려합니다.

- `SERVICE_IPV4_CIDR`— 정의는 `ServiceCIDR` 명령줄 파라미터를 참조하세요.
- `EXCLUDED_SNAT_CIDRS`— 쉼표로 구분된 문자열이어야 합니다. 정의는 `ExcludedSnatCIDRs` 명령줄 파라미터를 참조하세요.

`eksctl`로 자체 관리형 Windows Server 2022 시작

Windows Server 2022를 자체 관리형 노드로 실행하는 참조로 다음 **`test-windows-2022.`**`yaml`을 사용할 수 있습니다. 모든 *`example value`*를 고유한 값으로 바꿉니다.

> **Note**
>
> 자체 관리형 Windows Server 2022 노드를 실행하려면 `eksctl` 버전 `0.116.0` 이상을 사용해야 합니다.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: windows-2022-cluster
  region: region-code
  version: '1.29'

nodeGroups:
  - name: windows-ng
    instanceType: m5.2xlarge
    amiFamily: WindowsServer2022FullContainer
    volumeSize: 100
    minSize: 2
    maxSize: 3
  - name: linux-ng
    amiFamily: AmazonLinux2
    minSize: 2
    maxSize: 3
```

그러면 다음 명령을 사용하여 노드 그룹을 생성할 수 있습니다.

```
eksctl create cluster -f test-windows-2022.yaml
```

gMSA 인증 지원

Amazon EKS Windows Pods는 다양한 유형의 그룹 관리형 서비스 계정(gMSA) 인증을 허용합니다.

- Amazon EKS는 인증을 위해 Active Directory 도메인 자격 증명을 지원합니다. 도메인에 가입된 gMSA에 대한 자세한 내용은 AWS 블로그의 Amazon EKS Windowspods에서 Windows 인증을 참조하세요.
- Amazon EKS는 도메인에 가입되지 않은 Windows 노드가 이식 가능한 사용자 자격 증명으로 gMSA 자격 증명을 검색할 수 있도록 하는 플러그인을 제공합니다. 도메인 없는 gMSA에 대한 자세한 내용은 AWS 블로그의 Amazon EKS Windowspods에 대한 도메인 없는 Windows 인증을 참조하세요.

캐시된 컨테이너 이미지

Amazon EKS Windows 최적화 AMI에는 `containerd` 런타임에 대해 캐시된 특정 컨테이너 이미지가 있습니다. 컨테이너 이미지는 Amazon 관리형 빌드 구성 요소를 사용하여 사용자 지정 AMI를 빌드할 때 캐시됩니다. 자세한 내용은 Amazon 관리형 빌드 구성 요소 사용 단원을 참조하십시오.

다음 캐시된 컨테이너 이미지는 containerd 런타임용입니다.

- amazonaws.com/eks/pause-windows
- mcr.microsoft.com/windows/nanoserver
- mcr.microsoft.com/windows/servercore

추가 정보

Amazon EKS 최적화 Windows AMI 사용에 대한 자세한 내용은 다음 섹션을 참조하세요.

- 관리형 노드 그룹과 함께 Windows을 사용하려면 관리형 노드 그룹 섹션을 참조하세요.
- 자체 관리형 Windows 노드를 시작하려면 자체 관리형 Windows 노드 시작 섹션을 참조하세요.
- 버전 정보는 Amazon EKS 최적화 Windows AMI 버전를 참조하십시오.
- Amazon EKS 최적화 Windows AMI의 최신 ID를 검색하려면 Amazon EKS 최적화 Windows AMI ID 검색 섹션을 참조하세요.
- Amazon EC2 Image Builder를 사용하여 사용자 지정 Amazon EKS 최적화 Windows AMI를 생성하려면 사용자 지정 Amazon EKS 최적화 Windows AMI 생성 섹션을 참조하세요.
- 모범 사례는 EKS 모범 사례 가이드의 Amazon EKS 최적화 Windows AMI 관리를 참조하세요.

Amazon EKS 최적화 Windows AMI 버전

> **Important**
>
> AWS에서 게시한 Amazon EKS 최적화 Windows AMI에 대한 확장 지원은 Kubernetes 버전 1.23에서는 사용할 수 없지만 Kubernetes 버전 1.24 이상에서는 사용할 수 있습니다.

이 주제에서는 Amazon EKS 최적화 Windows AMI의 버전과 해당하는 kubelet, containerd, csi-proxy의 버전을 나열합니다.

AMI ID를 포함하여 각 변형에 대한 Amazon EKS 최적화 AMI 메타데이터를 프로그래밍 방식으로 가져올 수 있습니다. 자세한 내용은 Amazon EKS 최적화 Windows AMI ID 검색 단원을 참조하십시오.

AMI는 Kubernetes 버전과 AMI의 출시 날짜에 따라 다음 형식으로 버전이 지정됩니다.

```
k8s_major_version.k8s_minor_version-release_date
```

> **Note**
>
> Amazon EKS 관리형 노드 그룹에서는 Windows AMI의 2022년 11월 및 이후 릴리스를 지원합니다.

Amazon EKS 최적화 Windows Server 2022 Core AMI

다음 표에는 Amazon EKS 최적화 Windows Server 2022 Core AMI의 현재 및 이전 버전이 나와 있습니다.

Kubernetes version 1.29

Kubernetes 버전 **1.29**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	containerd 를 1.6.28로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	kubelet 가비지 수집 프로세스에서 일시 중지 이미지가 잘못 삭제되는 버그를 수정했습니다.
1.29-2024.01.11	1.29.0	1.6.18	1.1.2	Windows Server 2022 Core AMI에서 독립 실행형 Windows 업데이트 KB5034439가 제외되었습니다. KB는 Amazon EKS 최적화 Windows AMI에 포함되지 않

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
				은 별도의 WinRE 파티션이 있는 Windows 설치에만 적용됩니다.

Kubernetes version 1.28

Kubernetes 버전 **1.28**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.11	1.28.5	1.6.18	1.1.2	Windows Server 2022 Core AMI에서 독립 실행형 Windows 업데이트 KB5034439가 제외되었습니다. KB는 Amazon EKS 최적화 Windows AMI에 포함되지 않은 별도의 WinRE 파티션이 있는 Windows 설치에만 적용됩니다.
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	kubelet에서 보안 권고를 수정했습니다.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes 버전 **1.27**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.27-2024.03.12	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.27-2024.01.11	1.27.9	1.6.18	1.1.2	Windows Server 2022 Core AMI에서 독립 실행형 Windows 업데이트 KB5034439가 제외되었습니다. KB는 Amazon EKS 최적화 Windows AMI에 포함되지 않은 별도의 WinRE 파티션이 있는 Windows 설치에만 적용됩니다.
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	kubelet에서 보안 권고를 수정했습니다.
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	노드가 프라이빗 Amazon ECR 이미지를 풀링하지 못하게 하는 containers-roadmap 문제 #2042를 수정했습니다.
1.27-2023.05.17	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes 버전 **1.26**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.26-2024.03.12	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.11	1.26.12	1.6.18	1.1.2	Windows Server 2022 Core AMI에서 독립 실행형 Windows 업데이트 KB5034439가 제외되었습니다. KB는 Amazon EKS 최적화 Windows AMI에 포함되지 않은 별도의 WinRE 파티션이 있는 Windows 설치에만 적용됩니다.
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. kubelet를 1.26.9로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Kubernetes를 1.26.4로 업그레이드했습니다. CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	노드 재시작 후 포드에서 네트워크 연결 문제 #1126을 유발하는 버그를 수정했습니다. 새 부트스트랩 스크립트 구성 파라미터(ExcludedSnatCIDRs)를 도입했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	서비스 충돌 시 kubelet 및 kube-proxy 에 대한 복구 메커니즘을 추가했습니다.
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes 버전 **1.25**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.25-2024.03.12	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.11	1.25.16	1.6.18	1.1.2	Windows Server 2022 Core AMI에서 독립 실행형 Windows 업데이트 KB5034439가 제외되었습니다. KB는 Amazon EKS 최적화 Windows AMI에 포함되지 않

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
				은 별도의 WinRE 파티션이 있는 Windows 설치에만 적용됩니다.
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. kubelet를 1.25.14로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.25-2023.06.20	1.25.9	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.25-2023.06.14	1.25.9	1.6.6	1.1.1	Kubernetes를 1.25.9로 업그레이드했습니다. CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	노드 재시작 후 포드에서 네트워크 연결 문제 #1126을 유발하는 버그를 수정했습니다. 새 부트스트랩 스크립트 구성 파라미터(ExcludedSnatCIDRs)를 도입했습니다.
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	서비스 충돌 시 kubelet 및 kube-proxy 에 대한 복구 메커니즘을 추가했습니다.
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	Amazon EKS의 Windows 컨테이너에 대한 gMSA 인증을 용이하게 하기 위해 도메인 없는 gMSA 플러그인을 설치했습니다.
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes 버전 **1.24**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.24-2024.03.12	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.11	1.24.17	1.6.18	1.1.2	Windows Server 2022 Core AMI에서 독립 실행형 Windows 업데이트 KB5034439가 제외되었습니다. KB는 Amazon EKS 최적화 Windows AMI에 포함되지 않은 별도의 WinRE 파티션이 있는 Windows 설치에만 적용됩니다.
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. kubelet를 1.24.17로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
				SNAT_CIDRS)가 추가되었습니다.
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.20	1.24.13	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	Kubernetes를 1.24.13로 업그레이드했습니다. CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	노드 재시작 후 포드에서 네트워크 연결 문제 #1126을 유발하는 버그를 수정했습니다. 새 부트스트랩 스크립트 구성 파라미터(ExcludedSnatCIDRs)를 도입했습니다.
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	서비스 충돌 시 kubelet 및 kube-proxy 에 대한 복구 메커니즘을 추가했습니다.
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	Amazon EKS의 Windows 컨테이너에 대한 gMSA 인증을 용이하게 하기 위해 도메인 없는 gMSA 플러그인을 설치했습니다.
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	Kubernetes에서 보고된 문제가1.24.7 있으므로 1.24.10 버전이kube-proxy 다운그레이드되었습니다.
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.13	1.24.7	1.6.6	1.1.1	

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2022.10.11	1.24.7	1.6.6	1.1.1	

Amazon EKS 최적화 Windows Server 2022 Full AMI

다음 표에는 Amazon EKS 최적화 Windows Server 2022 Full AMI의 현재 및 이전 버전이 나와 있습니다.

Kubernetes version 1.29

Kubernetes 버전 **1.29**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	containerd 를 1.6.28로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.29-2024.03.12	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	kubelet 가비지 수집 프로세스에서 일시 중지 이미지가 잘못 삭제되는 버그를 수정했습니다.
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Kubernetes 버전 **1.28**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.28-2024.03.12	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	kubelet에서 보안 권고를 수정했습니다.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes 버전 **1.27**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.27-2024.03.12	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	kubelet에서 보안 권고를 수정했습니다.
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernete

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
				s 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	노드가 프라이빗 Amazon ECR 이미지를 풀링하지 못하게 하는 containers-roadmap 문제 #2042를 수정했습니다.
1.27-2023.05.18	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes 버전 **1.26**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.26-2024.03.12	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. kubelet를 1.26.9로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
				록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Kubernetes를 1.26.4로 업그레이드했습니다. CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	노드 재시작 후 포드에서 네트워크 연결 문제 #1126을 유발하는 버그를 수정했습니다. 새 부트스트랩 스크립트 구성 파라미터(ExcludedSnatCIDRs)를 도입했습니다.
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	서비스 충돌 시 kubelet 및 kube-proxy 에 대한 복구 메커니즘을 추가했습니다.
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes 버전 **1.25**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.25-2024.03.12	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.09	1.25.16	1.6.18	1.1.2	
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. kubelet를 1.25.14로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	
1.25-2023.06.20	1.25.9	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.25-2023.06.14	1.25.9	1.6.6	1.1.1	Kubernetes를 1.25.9로 업그레이드했습니다. CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	노드 재시작 후 포드에서 네트워크 연결 문제 #1126을 유발하는 버그를 수정했습니다. 새 부트스트랩 스크립트 구성 파라미터(ExcludedSnatCIDRs)를 도입했습니다.
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	서비스 충돌 시 kubelet 및 kube-proxy 에 대한 복구 메커니즘을 추가했습니다.
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	Amazon EKS의 Windows 컨테이너에 대한 gMSA 인증을 용이하게 하기 위해 도메인 없는 gMSA 플러그인을 설치했습니다.
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes 버전 **1.24**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2024.03.12	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.09	1.24.17	1.6.18	1.1.2	
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. kubelet를 1.24.17로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.20	1.24.13	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	Kubernetes를 1.24.13로 업그레이드했습니다. CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	노드 재시작 후 포드에서 네트워크 연결 문제 #1126을 유발하는 버그를 수정했습니다. 새 부트스트랩 스크립트 구성 파라미터(ExcludedSnatCIDRs)를 도입했습니다.
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	서비스 충돌 시 kubelet 및 kube-proxy 에 대한 복구 메커니즘을 추가했습니다.
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	Amazon EKS의 Windows 컨테이너에 대한 gMSA 인증을 용이하게 하기 위해 도메인 없는 gMSA 플러그인을 설치했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	Kubernetes에서 보고된 문제가1.24.7 있으므로 1.24.10 버전이kube-proxy 다운그레이드되었습니다.
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.14	1.24.7	1.6.6	1.1.1	
1.24-2022.10.11	1.24.7	1.6.6	1.1.1	

Amazon EKS 최적화 Windows Server 2019 Core AMI

다음 표에는 Amazon EKS 최적화 Windows Server 2019 Core AMI의 현재 및 이전 버전이 나와 있습니다.

Kubernetes version 1.29

Kubernetes 버전 **1.29**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	containerd 를 1.6.28로 업그레이드했습니다. golang

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
				1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	kubelet 가비지 수집 프로세스에서 일시 중지 이미지가 잘못 삭제되는 버그를 수정했습니다.
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Kubernetes 버전 **1.28**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	kubelet에서 보안 권고를 수정했습니다.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes 버전 **1.27**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.27-2024.03.13	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	kubelet에서 보안 권고를 수정했습니다.
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	노드가 프라이빗 Amazon ECR 이미지를 풀링하지 못하게 하는 containers-roadmap 문제 #2042를 수정했습니다.
11.27-2023.05.18	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes 버전 **1.26**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
				1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.26-2024.03.13	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. kubelet를 1.26.9로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Kubernetes를 1.26.4로 업그레이드했습니다. CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	노드 재시작 후 포드에서 네트워크 연결 문제 #1126을 유발하는 버그를 수정했습니다. 새 부트스트랩 스크립트 구성 파라미터(ExcludedSnatCIDRs)를 도입했습니다.
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	서비스 충돌 시 kubelet 및 kube-proxy 에 대한 복구 메커니즘을 추가했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes 버전 **1.25**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.25-2024.03.13	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.09	1.25.16	1.6.18	1.1.2	
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. kubelet를 1.25.14로 업그레이드했습니다. 새 부트스트랩 스

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
				크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	
1.25-2023.06.20	1.25.9	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.25-2023.06.14	1.25.9	1.6.6	1.1.1	Kubernetes를 1.25.9로 업그레이드했습니다. CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	노드 재시작 후 포드에서 네트워크 연결 문제 #1126을 유발하는 버그를 수정했습니다. 새 부트스트랩 스크립트 구성 파라미터(ExcludedSnatCIDRs)를 도입했습니다.
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	서비스 충돌 시 kubelet 및 kube-proxy 에 대한 복구 메커니즘을 추가했습니다.
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	Amazon EKS의 Windows 컨테이너에 대한 gMSA 인증을 용이하게 하기 위해 도메인 없는 gMSA 플러그인을 설치했습니다.
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes 버전 **1.24**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2024.03.13	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.09	1.24.17	1.6.18	1.1.2	
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. kubelet를 1.24.17로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.20	1.24.13	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	Kubernetes를 1.24.13로 업그레이드했습니다. CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	노드 재시작 후 포드에서 네트워크 연결 문제 #1126을 유발하는 버그를 수정했습니다. 새 부트스트랩 스크립트 구성 파라미터(ExcludedSnatCIDRs)를 도입했습니다.
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	서비스 충돌 시 kubelet 및 kube-proxy 에 대한 복구 메커니즘을 추가했습니다.
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	Amazon EKS의 Windows 컨테이너에 대한 gMSA 인증을 용이하게 하기 위해 도메인 없는 gMSA 플러그인을 설치했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	Kubernetes에서 보고된 문제가1.24.7 있으므로 1.24.10 버전이kube-proxy 다운그레이드되었습니다.
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.13	1.24.7	1.6.6	1.1.1	
1.24-2022.11.08	1.24.7	1.6.6	1.1.1	

Amazon EKS 최적화 Windows Server 2019 Full AMI

다음 표에는 Amazon EKS 최적화 Windows Server 2019 Full AMI의 현재 및 이전 버전이 나와 있습니다.

Kubernetes version 1.29

Kubernetes 버전 **1.29**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.29-2024.04.09	1.29.0	1.6.28	1.1.2	containerd 를 1.6.28로 업그레이드했습니다. golang

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
				1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.29-2024.03.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.13	1.29.0	1.6.25	1.1.2	
1.29-2024.02.06	1.29.0	1.6.25	1.1.2	kubelet 가비지 수집 프로세스에서 일시 중지 이미지가 잘못 삭제되는 버그를 수정했습니다.
1.29-2024.01.09	1.29.0	1.6.18	1.1.2	

Kubernetes version 1.28

Kubernetes 버전 **1.28**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.28-2024.04.09	1.28.5	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.28-2024.03.13	1.28.5	1.6.18	1.1.2	
1.28-2024.02.13	1.28.5	1.6.18	1.1.2	

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.28-2024.01.09	1.28.5	1.6.18	1.1.2	
1.28-2023.12.12	1.28.3	1.6.18	1.1.2	
1.28-2023.11.14	1.28.3	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.28-2023.10.19	1.28.2	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.28-2023-09.27	1.28.2	1.6.6	1.1.2	kubelet에서 보안 권고를 수정했습니다.
1.28-2023.09.12	1.28.1	1.6.6	1.1.2	

Kubernetes version 1.27

Kubernetes 버전 **1.27**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.27-2024.04.09	1.27.9	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.27-2024.03.13	1.27.9	1.6.18	1.1.2	
1.27-2024.02.13	1.27.9	1.6.18	1.1.2	
1.27-2024.01.09	1.27.9	1.6.18	1.1.2	
1.27-2023.12.12	1.27.7	1.6.18	1.1.2	
1.27-2023.11.14	1.27.7	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.27-2023.10.19	1.27.6	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.27-2023-09.27	1.27.6	1.6.6	1.1.2	kubelet에서 보안 권고를 수정했습니다.
1.27-2023.09.12	1.27.4	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.27-2023.08.17	1.27.4	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.
1.27-2023.08.08	1.27.3	1.6.6	1.1.1	
1.27-2023.07.11	1.27.3	1.6.6	1.1.1	
1.27-2023.06.20	1.27.1	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.27-2023.06.14	1.27.1	1.6.6	1.1.1	CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.
1.27-2023.06.06	1.27.1	1.6.6	1.1.1	노드가 프라이빗 Amazon ECR 이미지를 풀링하지 못하게 하는 containers-roadmap 문제 #2042를 수정했습니다.
1.27-2023.05.17	1.27.1	1.6.6	1.1.1	

Kubernetes version 1.26

Kubernetes 버전 **1.26**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.26-2024.04.09	1.26.12	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
				1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.26-2024.03.13	1.26.12	1.6.18	1.1.2	
1.26-2024.02.13	1.26.12	1.6.18	1.1.2	
1.26-2024.01.09	1.26.12	1.6.18	1.1.2	
1.26-2023.12.12	1.26.10	1.6.18	1.1.2	
1.26-2023.11.14	1.26.10	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.26-2023.10.19	1.26.9	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. kubelet를 1.26.9로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.26-2023.09.12	1.26.7	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.26-2023.08.17	1.26.7	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.
1.26-2023.08.08	1.26.6	1.6.6	1.1.1	
1.26-2023.07.11	1.26.6	1.6.6	1.1.1	
1.26-2023.06.20	1.26.4	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.26-2023.06.14	1.26.4	1.6.6	1.1.1	Kubernetes를 1.26.4로 업그레이드했습니다. CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.
1.26-2023.05.09	1.26.2	1.6.6	1.1.1	노드 재시작 후 포드에서 네트워크 연결 문제 #1126을 유발하는 버그를 수정했습니다. 새 부트스트랩 스크립트 구성 파라미터(ExcludedSnatCIDRs)를 도입했습니다.
1.26-2023.04.26	1.26.2	1.6.6	1.1.1	
1.26-2023.04.11	1.26.2	1.6.6	1.1.1	서비스 충돌 시 kubelet 및 kube-proxy 에 대한 복구 메커니즘을 추가했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.26-2023.03.24	1.26.2	1.6.6	1.1.1	

Kubernetes version 1.25

Kubernetes 버전 **1.25**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.25-2024.04.09	1.25.16	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.
1.25-2024.03.13	1.25.16	1.6.18	1.1.2	
1.25-2024.02.13	1.25.16	1.6.18	1.1.2	
1.25-2024.01.09	1.25.16	1.6.18	1.1.2	
1.25-2023.12.12	1.25.15	1.6.18	1.1.2	
1.25-2023.11.14	1.25.15	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.25-2023.10.19	1.25.14	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. kubelet를 1.25.14로 업그레이드했습니다. 새 부트스트랩 스

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
				크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.25-2023.09.12	1.25.12	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.
1.25-2023.08.17	1.25.12	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.
1.25-2023.08.08	1.25.9	1.6.6	1.1.1	
1.25-2023.07.11	1.25.9	1.6.6	1.1.1	
1.25-2023.06.20	1.25.9	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.25-2023.06.14	1.25.9	1.6.6	1.1.1	Kubernetes를 1.25.9로 업그레이드했습니다. CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.25-2023.05.09	1.25.7	1.6.6	1.1.1	노드 재시작 후 포드에서 네트워크 연결 문제 #1126을 유발하는 버그를 수정했습니다. 새 부트스트랩 스크립트 구성 파라미터(ExcludedSnatCIDRs)를 도입했습니다.
1.25-2023.04.11	1.25.7	1.6.6	1.1.1	서비스 충돌 시 kubelet 및 kube-proxy 에 대한 복구 메커니즘을 추가했습니다.
1.25-2023.03.27	1.25.6	1.6.6	1.1.1	Amazon EKS의 Windows 컨테이너에 대한 gMSA 인증을 용이하게 하기 위해 도메인 없는 gMSA 플러그인을 설치했습니다.
1.25-2023.03.20	1.25.6	1.6.6	1.1.1	
1.25-2023.02.14	1.25.6	1.6.6	1.1.1	

Kubernetes version 1.24

Kubernetes 버전 **1.24**

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2024.04.09	1.24.17	1.6.25	1.1.2	containerd 를 1.6.25로 업그레이드했습니다. golang 1.22.1을 사용하여 CNI 및 csi-proxy 를 재구축했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2024.03.13	1.24.17	1.6.18	1.1.2	
1.24-2024.02.13	1.24.17	1.6.18	1.1.2	
1.24-2024.01.09	1.24.17	1.6.18	1.1.2	
1.24-2023.12.12	1.24.17	1.6.18	1.1.2	
1.24-2023.11.14	1.24.17	1.6.18	1.1.2	CVE-2023-5528 에 대한 패치 포함.
1.24-2023.10.19	1.24.17	1.6.18	1.1.2	containerd 를 1.6.18로 업그레이드했습니다. kubelet를 1.24.17로 업그레이드했습니다. 새 부트스트랩 스크립트 환경 변수(SERVICE_IPV4_CIDR 및 EXCLUDED_SNAT_CIDRS)가 추가되었습니다.
1.24-2023.09.12	1.24.16	1.6.6	1.1.2	Kubernetes API 서버에서 Pod IP 주소를 가져오는 Kubernetes 커넥터 바이너리를 사용하도록 Amazon VPC CNI 플러그인을 업그레이드했습니다. 풀 요청 #100을 병합했습니다.
1.24-2023.08.17	1.24.16	1.6.6	1.1.2	CVE-2023-3676 , CVE-2023-3893 및 CVE-2023-3955 에 대한 패치 포함.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2023.08.08	1.24.13	1.6.6	1.1.1	
1.24-2023.07.11	1.24.13	1.6.6	1.1.1	
1.24-2023.06.21	1.24.13	1.6.6	1.1.1	DNS 접미사 검색 목록이 잘못 채워지는 문제가 해결되었습니다.
1.24-2023.06.14	1.24.13	1.6.6	1.1.1	Kubernetes를 1.24.13로 업그레이드했습니다. CNI에서 호스트 포트 매핑에 대한 지원이 추가되었습니다. 풀 요청 #93을 병합했습니다.
1.24-2023.05.09	1.24.7	1.6.6	1.1.1	노드 재시작 후 포드에서 네트워크 연결 문제 #1126을 유발하는 버그를 수정했습니다. 새 부트스트랩 스크립트 구성 파라미터(ExcludedSnatCIDRs)를 도입했습니다.
1.24-2023.04.11	1.24.7	1.6.6	1.1.1	서비스 충돌 시 kubelet 및 kube-proxy 에 대한 복구 메커니즘을 추가했습니다.
1.24-2023.03.27	1.24.7	1.6.6	1.1.1	Amazon EKS의 Windows 컨테이너에 대한 gMSA 인증을 용이하게 하기 위해 도메인 없는 gMSA 플러그인을 설치했습니다.

AMI 버전	**kubelet** 버전	**containerd** 버전	**csi-proxy** 버전	릴리스 정보
1.24-2023.03.20	1.24.7	1.6.6	1.1.1	Kubernetes에서 보고된 문제가1.24.7 있으므로 1.24.10 버전이kube-proxy 다운그레이드되었습니다.
1.24-2023.02.14	1.24.10	1.6.6	1.1.1	
1.24-2023.01.23	1.24.7	1.6.6	1.1.1	
1.24-2023.01.11	1.24.7	1.6.6	1.1.1	
1.24-2022.12.14	1.24.7	1.6.6	1.1.1	
1.24-2022.10.12	1.24.7	1.6.6	1.1.1	

Amazon EKS 최적화 Windows AMI ID 검색

AWS Systems Manager 파라미터 스토어 API를 쿼리하여 Amazon EKS 최적화 AMI의 Amazon Machine Image(AMI) ID를 프로그래밍 방식으로 검색할 수 있습니다. 이 파라미터를 사용하면 Amazon EKS 최적화 AMI ID를 수동으로 조회할 필요가 없습니다. Systems Manager 파라미터 스토어 API에 대한 자세한 내용은 GetParameter 섹션을 참조하세요. Amazon EKS 최적화 AMI 메타데이터를 검색하려면 IAM 보안 주체에 `ssm:GetParameter` IAM 권한이 있어야 합니다.

다음 명령에서 하위 파라미터 `image_id`를 사용하여 권장되는 최신 Amazon EKS 최적화 Windows AMI의 이미지 ID를 검색할 수 있습니다. *`1.29`*을 지원되는 모든 Amazon EKS 버전으로 바꿀 수 있으며, *`region-code`*를 AMI ID가 필요한 Amazon EKS 지원 리전으로 바꿀 수 있습니다. Windows Server 전체 AMI ID를 확인하려면 *`Core`*를 `Full`로 바꿉니다. Kubernetes 버전 `1.24` 이상의 경우*`2019`*를 2022로 바꾸어 Windows Server 2022 AMI ID를 확인할 수 있습니다.

```
aws ssm get-parameter --name /aws/service/ami-windows-latest/Windows_Server-2019-
English-Core-EKS_Optimized-1.29/image_id --region region-code --query "Parameter.Value"
 --output text
```

예제 출력은 다음과 같습니다.

```
ami-1234567890abcdef0
```

사용자 지정 Amazon EKS 최적화 Windows AMI 생성

EC2 Image Builder를 사용하여 다음 옵션 중 하나로 사용자 지정 Amazon EKS 최적화 Windows AMI를 생성할 수 있습니다.

- Amazon EKS 최적화 Windows AMI를 기반으로 사용
- Amazon 관리형 빌드 구성 요소 사용

두 방법 모두 고유한 Image Builder 레시피를 만들어야 합니다. 자세한 내용은 Image Builder 사용 설명서의 이미지 레시피의 새 버전 생성을 참조하세요.

> **Important**
>
> eks에 대한 다음 Amazon 관리형 구성 요소에는 CVE-2023-5528에 대한 패치가 포함됩니다.
>
> - 1.24.3 이상
> - 1.25.2 이상
> - 1.26.2 이상
> - 1.27.0 이상
> - 1.28.0 이상

Amazon EKS 최적화 Windows AMI를 기반으로 사용

이 옵션은 사용자 지정 Windows AMI를 구축하는 데 권장되는 방법입니다. AWS에서 제공하는 Amazon EKS 최적화 Windows AMI는 Amazon 관리형 빌드 구성 요소보다 더 자주 업데이트됩니다.

1. 새 Image Builder 레시피를 시작합니다.
   a. https://console.aws.amazon.com/imagebuilder에서 EC2 Image Builder 콘솔을 엽니다.

b. 왼쪽 탐색 창에서 이미지 레시피를 선택합니다.

c. 이미지 레시피 생성을 선택합니다.

2. 레시피 세부 정보 섹션에서 이름과 버전을 입력합니다.

3. 기본 이미지 섹션에서 Amazon EKS 최적화 Windows AMI의 ID를 지정합니다.

a. 사용자 지정 AMI ID 입력을 선택합니다.

b. 필요한 Windows OS 버전의 AMI ID를 검색합니다. 자세한 내용은 Amazon EKS 최적화 Windows AMI ID 검색 단원을 참조하십시오.

c. 사용자 지정 AMI ID를 입력합니다. AMI ID를 찾을 수 없는 경우 AMI ID의 AWS 리전이 콘솔의 오른쪽 상단에 표시된 AWS 리전과 일치하는지 확인합니다.

4. (선택 사항) 최신 보안 업데이트를 받으려면 빌드 구성 요소 - 섹션에서 `update-windows` 구성 요소를 추가합니다.

a. 이름으로 구성 요소 찾기 검색 상자 오른쪽의 드롭다운 목록에서 Amazon 관리형을 선택합니다.

b. 이름으로 구성 요소 찾기 상자에 **`update-windows`**를 입력합니다.

c. **`update-windows`** 검색 결과의 확인란을 선택합니다. 이 구성 요소에는 운영 체제의 최신 Windows 패치가 포함되어 있습니다.

5. 나머지 이미지 레시피 입력을 필요한 구성으로 완료합니다. 자세한 내용은 Image Builder 사용 설명서의 새 이미지 레시피 버전 생성(콘솔)을 참조하세요.

6. 레시피 생성을 선택합니다.

7. 신규 또는 기존 이미지 파이프라인에서 새 이미지 레시피를 사용합니다. 이미지 파이프라인이 성공적으로 실행되면 사용자 지정 AMI가 출력 이미지로 나열되고 사용할 준비가 됩니다. 자세한 내용은 EC2 Image Builder 콘솔 마법사를 사용하여 이미지 파이프라인 생성을 참조하세요.

Amazon 관리형 빌드 구성 요소 사용

Amazon EKS 최적화 Windows AMI를 기반으로 사용할 수 없는 경우 Amazon 관리형 빌드 구성 요소를 대신 사용할 수 있습니다. 이 옵션은 지원되는 최신 Kubernetes 버전보다 뒤처질 수 있습니다.

1. 새 Image Builder 레시피를 시작합니다.

a. https://console.aws.amazon.com/imagebuilder에서 EC2 Image Builder 콘솔을 엽니다.

b. 왼쪽 탐색 창에서 이미지 레시피를 선택합니다.

c. 이미지 레시피 생성을 선택합니다.

2. 레시피 세부 정보 섹션에서 이름과 버전을 입력합니다.
3. 기본 이미지 섹션에서 사용자 지정 AMI를 생성하는 데 사용할 옵션을 결정합니다.
   - 관리형 이미지 선택 - 이미지 운영 체제(OS)에서 Windows를 선택합니다. 그리고 나서 이미지 오리진에 대해 다음 옵션 중 하나를 선택합니다.
     - 빠른 시작(Amazon 관리형) - 이미지 이름 드롭다운에서 Amazon EKS 지원 Windows Server 버전을 선택합니다. 자세한 내용은 Amazon EKS 최적화 Windows AMI 단원을 참조하십시오.
     - 내가 소유한 이미지 - 이미지 이름으로 자체 라이선스가 있는 이미지의 ARN을 선택합니다. 제공하는 이미지에 Amazon EKS 구성 요소가 이미 설치되어 있을 수 없습니다.
   - 사용자 지정 AMI ID 입력 - AMI ID에 자체 라이선스가 있는 AMI의 ID를 입력합니다. 제공하는 이미지에 Amazon EKS 구성 요소가 이미 설치되어 있을 수 없습니다.
4. 빌드 구성 요소 - Windows 섹션에서 다음을 수행합니다.
   a. 이름으로 구성 요소 찾기 검색 상자 오른쪽의 드롭다운 목록에서 Amazon 관리형을 선택합니다.
   b. 이름으로 구성 요소 찾기 상자에 **eks**를 입력합니다.
   c. 반환되는 결과가 원하는 버전이 아닐 수도 있지만, **eks-optimized-ami-windows** 검색 결과의 확인란을 선택합니다.
   d. 이름으로 구성 요소 찾기 상자에 **update-windows**를 입력합니다.
   e. update-windows 검색 결과의 확인란을 선택합니다. 이 구성 요소에는 운영 체제의 최신 Windows 패치가 포함되어 있습니다.
5. 선택한 구성 요소 섹션에서 다음을 수행합니다.
   a. **eks-optimized-ami-windows**에 대한 버전 관리 옵션을 선택합니다.
   b. 구성 요소 버전 지정을 선택합니다.
   c. 구성 요소 버전 필드에 ***version.x*** 를 입력합니다. 이때 *version*을 지원되는 Kubernetes 버전으로 바꿉니다. 버전 번호의 일부로 *x*를 입력하면 사용자가 명시적으로 정의하는 버전 부분과도 일치하는 최신 구성 요소 버전을 사용한다는 의미입니다. 콘솔 출력에서 원하는 버전을 관리형 구성 요소로 사용할 수 있는지 여부를 알려주므로 유의하세요. 최신 Kubernetes 버전은 빌드 구성 요소에 사용하지 못할 수 있습니다. 사용 가능한 버전에 대한 자세한 내용은 eks-optimized-ami-windows 구성 요소 버전에 대한 정보 검색 섹션을 참조하세요.

> **Note**
>
> 다음 `eks-optimized-ami-windows` 빌드 구성 요소 버전에는 `eksctl` 버전 `0.129` 이하가 필요합니다.
>
> - `1.24.0`

6. 나머지 이미지 레시피 입력을 필요한 구성으로 완료합니다. 자세한 내용은 Image Builder 사용 설명서의 새 이미지 레시피 버전 생성(콘솔)을 참조하세요.
7. 레시피 생성을 선택합니다.
8. 신규 또는 기존 이미지 파이프라인에서 새 이미지 레시피를 사용합니다. 이미지 파이프라인이 성공적으로 실행되면 사용자 지정 AMI가 출력 이미지로 나열되고 사용할 준비가 됩니다. 자세한 내용은 EC2 Image Builder 콘솔 마법사를 사용하여 이미지 파이프라인 생성을 참조하세요.

eks-optimized-ami-windows 구성 요소 버전에 대한 정보 검색

각 구성 요소와 함께 설치되는 항목에 대한 특정 정보를 검색할 수 있습니다. 예를 들어 설치된 `kubelet` 버전을 확인할 수 있습니다. 구성 요소는 Amazon EKS 지원 Windows 운영 체제 버전에 대한 기능 테스트를 거칩니다. 자세한 내용은 출시 일정 단원을 참조하십시오. 지원되는 것으로 나열되지 않았거나 지원이 종료된 다른 Windows OS 버전은 구성 요소와 호환되지 않을 수 있습니다.

1. https://console.aws.amazon.com/imagebuilder에서 EC2 Image Builder 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 구성 요소를 선택합니다.
3. 이름으로 구성 요소 찾기 검색 상자 오른쪽의 드롭다운 목록에서 내 소유를 빠른 시작(Amazon 관리형)으로 변경합니다.
4. [이름으로 구성 요소 찾기(Find components by name)] 상자에 **eks**를 입력합니다.
5. (선택 사항) 최신 버전을 사용하는 경우 버전 열을 두 번 선택하여 내림차순으로 정렬합니다.
6. 원하는 버전의 **eks-optimized-ami-windows** 링크를 선택합니다.

결과 페이지의 설명에 특정 정보가 표시됩니다.

스토리지

이 장에서는 Amazon EKS 클러스터를 위한 스토리지 옵션에 관해 다룹니다.

주제

- Amazon EBS CSI 드라이버
- Amazon EFS CSI 드라이버
- Amazon FSx for Lustre CSI 드라이버
- Amazon FSx for NetApp ONTAP CSI 드라이버
- Amazon FSx for OpenZFS CSI 드라이버
- Amazon File Cache CSI 드라이버
- Mountpoint for Amazon S3 CSI 드라이버
- CSI 스냅샷 컨트롤러

Amazon EBS CSI 드라이버

Amazon Elastic Block Store(Amazon EBS)) CSI(Container Storage Interface) 드라이버에서는 Amazon EBS 볼륨의 수명 주기를 사용자가 생성하는 Kubernetes 볼륨의 스토리지로 관리합니다. Amazon EBS CSI 드라이버는 Amazon EBS 볼륨을 Kubernetes 볼륨 유형인 일반 임시 볼륨 및 영구 볼륨에 사용할 수 있도록 만듭니다.

Amazon EBS CSI 드라이버 사용 시 고려해야 할 몇 가지 사항이 있습니다.

- Amazon EBS CSI 플러그 인이 사용자를 대신하여 AWS API를 호출하려면 IAM 권한이 필요합니다. 자세한 내용은 Amazon EBS CSI 드라이버 IAM 역할 생성 단원을 참조하십시오.
- Amazon EBS 볼륨을 Fargate Pods에 탑재할 수 없습니다.
- Amazon EBS CSI 컨트롤러는 Fargate 노드에서 실행할 수 있지만 Amazon EBS CSI 노드 DaemonSet은(는) Amazon EC2 인스턴스에서만 실행할 수 있습니다.

클러스터를 처음 생성할 때 Amazon EBS CSI 드라이버가 설치되지 않습니다. 드라이버를 사용하려면 Amazon EKS 추가 기능 또는 자체 관리형 추가 기능으로 드라이버를 추가해야 합니다.

- Amazon EKS 추가 기능으로 드라이버를 추가하는 방법에 대한 지침은 Amazon EKS 추가 기능으로 Amazon EBS CSI 드라이버 관리 섹션을 참조하세요.

- 자체 관리형 설치 기능으로 드라이버를 추가하는 방법에 대한 지침은 GitHub의 Amazon EBS Container Storage Interface(CSI) 드라이버 프로젝트를 참조하세요.

두 방법 중 하나로 CSI 드라이버를 설치하면 샘플 애플리케이션으로 기능을 테스트할 수 있습니다. 자세한 내용은 샘플 애플리케이션 배포 및 CSI 드라이버 작동 여부 확인 단원을 참조하십시오.

Amazon EBS CSI 드라이버 IAM 역할 생성

Amazon EBS CSI 플러그 인이 사용자를 대신하여 AWS API를 호출하려면 IAM 권한이 필요합니다. 자세한 내용은 GitHub의 드라이버 권한 설정을 참조하세요.

> **Note**
>
> IMDS에 대한 액세스 권한을 차단하지 않는 한 Pods는 IAM 역할에 할당된 권한에 대한 액세스 권한이 있습니다. 자세한 내용은 Amazon EKS에 대한 보안 모범 사례 단원을 참조하십시오.

사전 조건

- 기존 클러스터가 있어야 합니다.
- 클러스터에 대한 기존 AWS Identity and Access Management(IAM) OpenID Connect(OIDC) 제공업체입니다. 이미 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 부분을 참조하세요.

다음 절차에서는 IAM 역할을 생성하고 AWS 관리형 정책을 연결하는 방법을 보여줍니다. `eksctl`, AWS Management Console 또는 AWS CLI를 사용할 수 있습니다.

> **Note**
>
> 이 절차의 특정 단계는 드라이버를 Amazon EKS 추가 기능으로 사용하기 위해 작성되었습니다. 드라이버를 자체 관리형 추가 기능으로 드라이버를 사용하려면 여러 단계가 필요합니다. 자세한 내용은 GitHub의 드라이버 권한 설정을 참조하세요.

eksctl

`eksctl`을 사용하여 Amazon EBS CSI 플러그 인 IAM 역할 생성

1. IAM 역할을 생성하여 정책을 연결합니다. AWS은(는) AWS 관리형 정책을 유지 관리하거나 자체 사용자 지정 정책을 생성할 수 있습니다. IAM 역할을 생성하고 다음 명령을 사용하여 AWS 관리형 정책을 연결할 수 있습니다. *`my-cluster`*를 클러스터 이름으로 바꿉니다. 이 명령은 IAM 역할을 생성하고 IAM 정책을 연결하는 AWS CloudFormation 스택을 배포합니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

```
eksctl create iamserviceaccount \
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_EBS_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEBSCSIDriverPolicy \
    --approve
```

2. Amazon EBS 볼륨의 암호화에 사용자 지정 KMS 키를 사용하는 경우 필요에 따라 IAM 역할을 사용자 지정합니다. 예를 들어, 다음을 수행합니다.

   a. 다음 코드를 복사하여 새로운 *`kms-key-for-encryption-on-ebs`*`.json` 파일에 붙여 넣습니다. *`custom-key-arn`*을 사용자 지정 KMS 키 ARN으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-arn"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
```

```
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-arn"]
    }
  ]
}
```

b. 정책을 생성합니다. *KMS_Key_For_Encryption_On_EBS_Policy*를 다른 이름으로 변경할 수 있습니다. 그러나 이렇게 하는 경우 이후 단계에서도 변경해야 합니다.

```
aws iam create-policy \
  --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
  --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. 다음 명령을 사용하여 IAM 정책을 역할에 연결합니다. *111122223333*을 계정 ID로 바꿉니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

```
aws iam attach-role-policy \
  --policy-arn
 arn:aws:iam::111122223333:policy/KMS_Key_For_Encryption_On_EBS_Policy \
  --role-name AmazonEKS_EBS_CSI_DriverRole
```

AWS Management Console

AWS Management Console을 사용하여 Amazon EBS CSI 플러그 인 IAM 역할을 생성하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 역할(Roles) 페이지에서 역할 생성(Create role)을 선택합니다.
4. 신뢰할 수 있는 엔터티 선택(Select trusted entity) 페이지에서 다음을 수행합니다.

a. 신뢰할 수 있는 엔터티 유형(Trusted entity type) 섹션에서 웹 자격 증명(Web identity)을 선택합니다.

b. 보안 인증 공급자의 경우 클러스터에 대해 OpenID Connect 공급자 URL을 선택합니다 (Amazon EKS의 개요에 표시된 대로).

c. 대상(Audience)에서 `sts.amazonaws.com`을 입력합니다.

d. 다음을 선택합니다.

5. 권한 추가(Add permissions) 페이지에서 다음을 수행합니다.

a. 필터 정책(Filter policies) 상자에 `AmazonEBSCSIDriverPolicy`를 입력합니다.

b. 검색에서 반환된 `AmazonEBSCSIDriverPolicy` 왼쪽에 있는 확인란을 선택합니다.

c. 다음을 선택합니다.

6. 이름, 검토 및 생성(Name, review, and create) 페이지에서 다음을 수행합니다.

a. 역할 이름(Role name)에 역할의 고유한 이름(예: ***AmazonEKS_EBS_CSI_DriverRole***)을 입력합니다.

b. 태그 추가(선택 사항)에서 태그를 키-값 페어로 연결하여 메타데이터를 역할에 추가합니다. IAM에서 태그 사용에 대한 자세한 내용을 알아보려면 IAM 사용 설명서의 IAM 리소스에 태그 지정을 참조하세요.

c. 역할 생성을 선택합니다.

7. 역할을 생성한 후 편집할 수 있도록 콘솔에서 이 역할을 선택하여 엽니다.

8. 신뢰 관계(Trust relationships) 탭을 선택한 후 신뢰 정책 편집(Edit trust policy)을 선택합니다.

9. 다음과 비슷한 줄을 찾습니다.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

이전 줄의 끝에 쉼표를 추가한 다음 이전 줄 뒤에 다음 줄을 추가합니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다. *EXAMPLED539D4633E53DE1B71EXAMPLE*을 클러스터의 OIDC 공급자 ID로 바꿉니다.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:ebs-csi-controller-sa"
```

10. 신뢰 정책 업데이트(Update Trust Policy)를 선택하여 종료합니다.

11. Amazon EBS 볼륨의 암호화에 사용자 지정 KMS 키를 사용하는 경우 필요에 따라 IAM 역할을 사용자 지정합니다. 예를 들어, 다음을 수행합니다.

    a. 왼쪽 탐색 창에서 정책을 선택합니다.

    b. 정책(Policies) 페이지에서 정책 생성(Create Policy)을 선택합니다.

    c. 정책 생성(Create Policy) 페이지에서 JSON 탭을 선택합니다.

    d. 다음 코드를 복사하여 편집기에 붙여 넣어 *custom-key-arn*을 사용자 지정 KMS 키 ARN으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-arn"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-arn"]
    }
  ]
}
```

    e. 다음: 태그(Next: Tags)를 선택합니다.

f. 태그 추가(선택 사항)(Add tags (Optional)) 페이지에서 다음: 검토(Next: Review)를 선택합니다.

g. 이름(Name)에 정책의 고유 이름을 입력합니다(예: ***KMS_Key_For_Encryption_On_EBS_Policy***).

h. 정책 생성을 선택합니다.

i. 왼쪽 탐색 창에서 역할을 선택합니다.

j. 콘솔에서 ***AmazonEKS_EBS_CSI_DriverRole***을 선택하여 열고 편집합니다.

k. 권한 추가 드롭다운 목록에서 정책 연결을 선택합니다.

l. 필터 정책(Filter policies) 상자에 *KMS_Key_For_Encryption_On_EBS_Policy*를 입력합니다.

m. 검색에서 반환된 *KMS_Key_For_Encryption_On_EBS_Policy* 왼쪽에 있는 확인란을 선택합니다.

n. 정책 연결을 선택합니다.

AWS CLI

AWS CLI을 사용하여 Amazon EBS CSI 플러그 인 IAM 역할을 생성하려면

1. 클러스터의 OIDC 공급자 URL을 확인합니다. *my-cluster*을 클러스터 이름으로 교체합니다. 명령의 출력이 None인 경우 사전 요구 사항을 검토합니다.

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

예제 출력은 다음과 같습니다.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. IAM 역할을 생성하여 `AssumeRoleWithWebIdentity` 작업을 부여합니다.

a. 다음 콘텐츠를 *aws-ebs-csi-driver-trust-policy*.json이라는 파일에 복사합니다. *111122223333*을 계정 ID로 바꿉니다. *EXAMPLED539D4633E53DE1B71EXAMPLE* 및 *region-code*를 이전 단계에서 반환된 값으로 바꿉니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:ebs-csi-controller-sa"
        }
      }
    }
  ]
}
```

b. 역할을 생성합니다. *AmazonEKS_EBS_CSI_DriverRole*를 다른 이름으로 변경할 수 있습니다. 변경하는 경우 이후 단계에서 변경해야 합니다.

```
aws iam create-role \
  --role-name AmazonEKS_EBS_CSI_DriverRole \
  --assume-role-policy-document file://"aws-ebs-csi-driver-trust-
policy.json"
```

3. 정책을 연결합니다. AWS은(는) AWS 관리형 정책을 유지 관리하거나 자체 사용자 지정 정책을 생성할 수 있습니다. 다음 명령으로 AWS 관리형 정책을 역할에 연결합니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy \
  --role-name AmazonEKS_EBS_CSI_DriverRole
```

4. Amazon EBS 볼륨의 암호화에 사용자 지정 KMS 키를 사용하는 경우 필요에 따라 IAM 역할을 사용자 지정합니다. 예를 들어, 다음을 수행합니다.

   a. 다음 코드를 복사하여 새로운 *kms-key-for-encryption-on-ebs*.json 파일에 붙여 넣습니다. *custom-key-arn*을 사용자 지정 KMS 키 ARN으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:ListGrants",
        "kms:RevokeGrant"
      ],
      "Resource": ["custom-key-arn"],
      "Condition": {
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
      ],
      "Resource": ["custom-key-arn"]
    }
  ]
}
```

   b. 정책을 생성합니다. *KMS_Key_For_Encryption_On_EBS_Policy*를 다른 이름으로 변경할 수 있습니다. 그러나 이렇게 하는 경우 이후 단계에서도 변경해야 합니다.

```
aws iam create-policy \
  --policy-name KMS_Key_For_Encryption_On_EBS_Policy \
```

```
  --policy-document file://kms-key-for-encryption-on-ebs.json
```

c. 다음 명령을 사용하여 IAM 정책을 역할에 연결합니다. *111122223333*을 계정 ID로 바꿉니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

```
aws iam attach-role-policy \
  --policy-arn
 arn:aws:iam::111122223333:policy/KMS_Key_For_Encryption_On_EBS_Policy \
  --role-name AmazonEKS_EBS_CSI_DriverRole
```

Amazon EBS CSI 드라이버 IAM 역할을 생성했으므로 Amazon EBS CSI 드라이버 추가 기능 추가로 진행할 수 있습니다. 해당 절차에서 플러그인이 배포되면 `ebs-csi-controller-sa`라는 서비스 계정을 생성하고 해당 계정을 사용하도록 구성됩니다. 이 서비스 계정은 필요한 Kubernetes 권한이 할당되어 있는 Kubernetes `clusterrole`에 바인딩됩니다.

Amazon EKS 추가 기능으로 Amazon EBS CSI 드라이버 관리

보안을 강화하고 작업량을 줄이려면 Amazon EKS CSI 드라이버를 Amazon EKS 추가 기능으로 관리할 수 있습니다. Amazon EKS 추가 기능에 대한 자세한 내용은 Amazon EKS 추가 기능 섹션을 참조하세요. Amazon EBS CSI 드라이버 추가 기능 추가의 단계에 따라 Amazon EBS CSI 추가 기능을 추가할 수 있습니다.

Amazon EBS CSI 추가 기능을 추가한 경우 Amazon EKS 추가 기능으로 Amazon EBS CSI 드라이버 업데이트 및 Amazon EBS CSI 추가 기능 제거 섹션의 단계에 따라 관리할 수 있습니다.

사전 조건

- 기존 클러스터가 있어야 합니다. 필요한 플랫폼 버전을 확인하려면 다음 명령을 실행합니다.

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver
```

- 클러스터에 대한 기존 AWS Identity and Access Management(IAM) OpenID Connect(OIDC) 제공업체입니다. 이미 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 부분을 참조하세요.
- Amazon EBS CSI 드라이버 IAM 역할입니다. 이 사전 조건을 충족하지 않는 경우 추가 기능 설치를 시도하고 `kubectl describe pvc`를 실행하면 `could not create volume in EC2: UnauthorizedOperation` 오류와 함께 `failed to provision volume with`

StorageClass가 표시됩니다. 자세한 내용은 Amazon EBS CSI 드라이버 IAM 역할 생성 섹션을 참조하세요.

- 클러스터 전체에서 제한된 PodSecurityPolicy를 사용하는 경우 배포할 수 있는 충분한 권한이 추가 기능에 부여되었는지 확인합니다. 각 추가 기능 Pod에 필요한 권한을 알아보려면 GitHub의 관련 추가 기능 매니페스트 정의를 참조하세요.

> **Important**
>
> Amazon EBS CSI 드라이버의 스냅샷 기능을 사용하려면 추가 기능을 설치하기 전에 외부 스냅샷을 설치해야 합니다. 외부 스냅샷 구성 요소는 다음 순서로 설치해야 합니다.
>
> - volumesnapshotclasses, volumesnapshots 및 volumesnapshotcontents에 대한 CustomResourceDefinition(CRD)
> - RBAC(ClusterRole, ClusterRoleBinding 등)
> - 컨트롤러 배포
>
> 자세한 내용은 GitHub의 CSI Snapshotter를 참조하세요.

Amazon EBS CSI 드라이버 추가 기능 추가

> **Important**
>
> Amazon EBS 드라이버를 Amazon EKS 추가 기능으로 추가하기 전에 클러스터에 자체 관리형 버전의 드라이버가 설치되어 있지 않은지 확인하세요. 설치되어 있다면 GitHub의 자체 관리형 Amazon EBS CSI 제거를 참조하세요.

eksctl, AWS Management Console 또는 AWS CLI를 사용하여 Amazon EBS CSI 추가 기능을 클러스터에 추가할 수 있습니다.

eksctl

eksctl을 사용하여 Amazon EBS CSI 추가 기능을 추가하려면

다음 명령을 실행합니다. *my-cluster*를 클러스터 이름으로, *111122223333*을 계정 ID로, *AmazonEKS_EBS_CSI_DriverRole*을 이전에 생성한 IAM 역할의 이름으로 바꿉니다. 클러

스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

```
eksctl create addon --name aws-ebs-csi-driver --cluster my-cluster --service-
account-role-arn arn:aws:iam::111122223333:role/AmazonEKS_EBS_CSI_DriverRole --force
```

기존 설정과 충돌하는 **`--force`** 옵션과 Amazon EKS 추가 기능 설정을 제거할 경우, Amazon EKS 추가 기능의 업데이트 작업이 실패하고 충돌을 해결하는 데 도움이 되는 오류 메시지가 표시됩니다. 이 옵션을 사용하여 이러한 설정을 덮어쓰기 때문에 이 옵션을 지정하기 전에 Amazon EKS 추가 기능이 관리해야 하는 설정을 관리하지 않는지 확인합니다. 이 설정의 기타 옵션에 대한 자세한 내용은 `eksctl` 설명서의 추가 기능을 참조하세요. Amazon EKS Kubernetes 필드 관리에 대한 자세한 내용은 Kubernetes 필드 관리 섹션을 참조하세요.

AWS Management Console

AWS Management Console을 사용하여 Amazon EBS CSI 추가 기능을 추가하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터를 선택합니다.
3. Amazon EBS CSI 추가 기능을 구성할 클러스터의 이름을 선택합니다.
4. 추가 기능(Add-ons) 탭을 선택합니다.
5. 추가 기능 더 가져오기를 선택합니다.
6. 추가 기능 선택 페이지에서 다음을 수행합니다.
   a. Amazon EKS 추가 기능 섹션에서 Amazon EBS CSI 드라이버 확인란을 선택합니다.
   b. 다음을 선택합니다.
7. 선택한 추가 기능 설정 구성 페이지에서 다음을 수행합니다.
   a. 사용할 버전(Version)을 선택합니다.
   b. IAM 역할 선택에서 Amazon EBS CSI 드라이버 IAM 정책을 연결한 IAM 역할의 이름을 선택합니다.
   c. (선택 사항) 선택적 구성 설정을 확장할 수 있습니다. 충돌 해결 방법에서 재정의를 선택한 경우 기존 추가 기능에 대한 하나 이상의 설정을 Amazon EKS 추가 기능의 설정으로 덮어쓸 수 있습니다. 이 옵션을 사용 설정하지 않고 기존 설정과 충돌이 있는 경우 작업이 실패합니다. 결과 오류 메시지를 사용하여 충돌을 해결할 수 있습니다. 이 옵션을 선택하기 전에 Amazon EKS 추가 기능이 사용자가 자체 관리해야 하는 설정을 관리하지 않는지 확인하세요.

d. 다음을 선택합니다.

8. 검토 및 추가 페이지에서 생성을 선택합니다. 추가 기능 설치가 완료되면 설치한 추가 기능이 표시됩니다.

AWS CLI

AWS CLI을 사용하여 Amazon EBS CSI 추가 기능을 추가하려면

다음 명령을 실행합니다. *my-cluster*를 클러스터 이름으로, *111122223333*을 계정 ID로, *AmazonEKS_EBS_CSI_DriverRole*을 이전에 생성한 역할의 이름으로 바꿉니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-ebs-csi-driver \
  --service-account-role-arn
 arn:aws:iam::111122223333:role/AmazonEKS_EBS_CSI_DriverRole
```

Amazon EKS 추가 기능으로 Amazon EBS CSI 드라이버를 추가했으므로 샘플 애플리케이션 배포 및 CSI 드라이버 작동 여부 확인으로 진행할 수 있습니다. 이 절차에는 스토리지 클래스 설정이 포함됩니다.

Amazon EKS 추가 기능으로 Amazon EBS CSI 드라이버 업데이트

Amazon EKS는 새 버전이 릴리스되거나 새 Kubernetes 마이너 버전으로 클러스터를 업데이트한 후 클러스터의 Amazon EBS CSI를 자동으로 업데이트하지 않습니다. 기존 클러스터에서 Amazon EBS CSI를 업데이트하려면 업데이트를 시작해야 합니다. 그러면 Amazon EKS가 추가 기능을 업데이트합니다.

eksctl

`eksctl`을 사용하여 Amazon EBS CSI 추가 기능을 업데이트하려면

1. Amazon EBS CSI 추가 기능의 최신 버전을 확인합니다. *my-cluster*을 클러스터 이름으로 교체합니다.

   ```
   eksctl get addon --name aws-ebs-csi-driver --cluster my-cluster
   ```

   예제 출력은 다음과 같습니다.

```
NAME                    VERSION                        STATUS  ISSUES  IAMROLE
 UPDATE AVAILABLE
aws-ebs-csi-driver      v1.11.2-eksbuild.1             ACTIVE  0
  v1.11.4-eksbuild.1
```

2. 추가 기능을 이전 단계의 출력에서 UPDATE AVAILABLE에 반환된 버전으로 업데이트합니다.

```
eksctl update addon --name aws-ebs-csi-driver --version v1.11.4-eksbuild.1 --
cluster my-cluster \
  --service-account-role-arn
 arn:aws:iam::111122223333:role/AmazonEKS_EBS_CSI_DriverRole --force
```

기존 설정과 충돌하는 **--force** 옵션과 Amazon EKS 추가 기능 설정을 제거할 경우, Amazon EKS 추가 기능의 업데이트 작업이 실패하고 충돌을 해결하는 데 도움이 되는 오류 메시지가 표시됩니다. 이 옵션을 사용하여 이러한 설정을 덮어쓰기 때문에 이 옵션을 지정하기 전에 Amazon EKS 추가 기능이 관리해야 하는 설정을 관리하지 않는지 확인합니다. 이 설정의 기타 옵션에 대한 자세한 내용은 eksctl 설명서의 추가 기능을 참조하세요. Amazon EKS Kubernetes 필드 관리에 대한 자세한 내용은 Kubernetes 필드 관리 섹션을 참조하세요.

AWS Management Console

AWS Management Console을 사용하여 Amazon EBS CSI 추가 기능을 업데이트하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터를 선택합니다.
3. Amazon EBS CSI 추가 기능을 업데이트할 클러스터의 이름을 선택합니다.
4. 추가 기능(Add-ons) 탭을 선택합니다.
5. Amazon EBS CSI 드라이버를 선택합니다.
6. 편집을 선택합니다.
7. Amazon EBS CSI 드라이버 구성 페이지에서 다음을 수행합니다.
   a. 사용할 버전(Version)을 선택합니다.
   b. IAM 역할 선택에서 Amazon EBS CSI 드라이버 IAM 정책을 연결한 IAM 역할의 이름을 선택합니다.
   c. (선택 사항) 필요한 경우 선택적 구성 설정을 확장하고 수정할 수 있습니다.
   d. 변경 사항 저장을 선택합니다.

AWS CLI

AWS CLI을 사용하여 Amazon EBS CSI 추가 기능을 업데이트하려면

1. Amazon EBS CSI 추가 기능의 최신 버전을 확인합니다. *my-cluster*을 클러스터 이름으로 교체합니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name aws-ebs-csi-driver
 --query "addon.addonVersion" --output text
```

예제 출력은 다음과 같습니다.

```
v1.11.2-eksbuild.1
```

2. 클러스터 버전에 사용할 수 있는 Amazon EBS CSI 추가 기능 버전을 확인합니다.

```
aws eks describe-addon-versions --addon-name aws-ebs-csi-driver --kubernetes-
version 1.23 \
  --query "addons[].addonVersions[].[addonVersion,
 compatibilities[].defaultVersion]" --output text
```

예제 출력은 다음과 같습니다.

```
v1.11.4-eksbuild.1
True
v1.11.2-eksbuild.1
False
```

아래에 `True`가 있는 버전이 추가 기능이 생성될 때 배포된 기본 버전입니다. 추가 기능이 생성될 때 배포된 버전이 사용 가능한 최신 버전이 아닐 수도 있습니다. 이전 출력에서는 추가 기능이 생성될 때 최신 버전이 배포됩니다.

3. 추가 기능을 이전 단계의 출력에서 `True`가 반환된 버전으로 업데이트합니다. 출력에서 반환된 경우 이후 버전으로 업데이트할 수도 있습니다.

```
aws eks update-addon --cluster-name my-cluster --addon-name aws-ebs-csi-driver
 --addon-version v1.11.4-eksbuild.1 \
  --service-account-role-arn
 arn:aws:iam::111122223333:role/AmazonEKS_EBS_CSI_DriverRole --resolve-
conflicts PRESERVE
```

PRESERVE(보존) 옵션에서는 추가 기능에 대해 설정한 사용자 정의 설정이 있으면 보존합니다. 이 설정의 기타 옵션에 대한 자세한 내용을 알아보려면 Amazon Web Services CLI Command Reference(Amazon Web Services 명령 레퍼런스)의 update-addon을 참조하세요. Amazon EKS 추가 기능에 대한 자세한 내용은 Kubernetes 필드 관리 섹션을 참조하세요.

Amazon EBS CSI 추가 기능 제거

Amazon EKS 추가 기능을 제거하는 경우 다음 두 가지 옵션이 있습니다.

- 클러스터에 추가 기능 소프트웨어 보존 – 이 옵션은 모든 설정에 대한 Amazon EKS 관리를 제거합니다. 또한 업데이트를 시작한 후 Amazon EKS가 업데이트를 알리고 Amazon EKS 추가 기능을 자동으로 업데이트하는 기능을 제거합니다. 그러나 클러스터에 추가 기능 소프트웨어는 유지됩니다. 이 옵션을 사용하면 추가 기능을 Amazon EKS 추가 기능이 아닌 자체 관리형 설치 기능으로 만들 수 있습니다. 이 옵션을 사용하면 추가 기능에 대한 가동 중지 시간이 없습니다. 이 절차의 명령은 이 옵션을 사용합니다.
- 클러스터에서 추가 기능 소프트웨어 완전히 제거(Remove the add-on software entirely from your cluster) - 클러스터에 종속된 리소스가 없는 경우에만 클러스터에서 Amazon EKS 추가 기능을 제거하는 것이 좋습니다. 이 옵션을 수행하려면 이 절차에서 사용하는 명령에서 `--preserve`를 삭제합니다.

추가 기능에 연결된 IAM 계정이 있는 경우 IAM 계정은 제거되지 않습니다.

`eksctl`, AWS Management Console 또는 AWS CLI를 사용하여 Amazon EBS CSI 추가 기능을 제거할 수 있습니다.

eksctl

`eksctl`을 사용하여 Amazon EBS CSI 추가 기능을 제거하려면

*`my-cluster`*를 클러스터 이름으로 바꾸고 다음 명령을 실행합니다.

```
eksctl delete addon --cluster my-cluster --name aws-ebs-csi-driver --preserve
```

AWS Management Console

AWS Management Console을 사용하여 Amazon EBS CSI 추가 기능을 제거하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.

2. 좌측 탐색 창에서 클러스터를 선택합니다.
3. Amazon EBS CSI 추가 기능을 제거할 클러스터의 이름을 선택합니다.
4. 추가 기능(Add-ons) 탭을 선택합니다.
5. Amazon EBS CSI 드라이버를 선택합니다.
6. 제거(Remove)를 선택합니다.
7. 제거: aws-ebs-csi-driver 확인 대화 상자에서 다음을 수행합니다.
   a. 추가 기능에 대한 Amazon EKS의 설정 관리를 중지하려면 클러스터에 보존을 선택합니다. 클러스터에 추가 소프트웨어를 유지하려면 이렇게 합니다. 이는 추가 기능의 모든 설정을 스스로 관리할 수 있도록 하기 위한 것입니다.
   b. **aws-ebs-csi-driver**을 입력합니다.
   c. 제거(Remove)를 선택합니다.

AWS CLI

AWS CLI을 사용하여 Amazon EBS CSI 추가 기능을 제거하려면

*my-cluster*를 클러스터 이름으로 바꾸고 다음 명령을 실행합니다.

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-ebs-csi-driver --
preserve
```

샘플 애플리케이션 배포 및 CSI 드라이버 작동 여부 확인

샘플 애플리케이션을 사용하여 CSI 드라이버 기능을 테스트할 수 있습니다. 이 주제에서는 한 가지 예를 보여 주는데 다음을 수행할 수도 있습니다.

- 외부 스냅샷을 사용하는 샘플 애플리케이션을 배포하여 볼륨 스냅샷을 생성합니다. 자세한 내용은 GitHub의 볼륨 스냅샷을 참조하세요.
- 볼륨 크기 조정을 사용하는 샘플 애플리케이션을 배포합니다. 자세한 내용은 GitHub의 볼륨 크기 조정을 참조하세요.

이 절차에서는 Amazon EBS Container Storage Interface(CSI) 드라이버 GitHub 리포지토리의 동적 볼륨 프로비저닝 예제를 사용하여 동적으로 프로비저닝된 Amazon EBS 볼륨을 소비합니다.

1. Amazon EBS 컨테이너 스토리지 인터페이스(CSI) 드라이버 GitHub 리포지토리를 로컬 시스템에 복제합니다.

```
git clone https://github.com/kubernetes-sigs/aws-ebs-csi-driver.git
```

2. dynamic-provisioning 예제 디렉터리로 이동합니다.

```
cd aws-ebs-csi-driver/examples/kubernetes/dynamic-provisioning/
```

3. (선택 사항)manifests/storageclass.yaml 파일은 기본적으로gp2 Amazon EBS 볼륨을 프로비저닝합니다. gp3볼륨을 대신 사용하려면 type: gp3을 manifests/storageclass.yaml에 추가하세요.

```
echo "parameters:
  type: gp3" >> manifests/storageclass.yaml
```

4. manifests 디렉터리에서 ebs-sc 스토리지 클래스, ebs-claim 영구 볼륨 클레임, app 샘플 애플리케이션을 배포합니다.

```
kubectl apply -f manifests/
```

5. ebs-sc 스토리지 클래스에 관해 설명합니다.

```
kubectl describe storageclass ebs-sc
```

예제 출력은 다음과 같습니다.

```
Name:            ebs-sc
IsDefaultClass:  No
Annotations:     kubectl.kubernetes.io/last-applied-
configuration={"apiVersion":"storage.k8s.io/v1","kind":"StorageClass","metadata":
{"annotations":{},"name":"ebs-
sc"},"provisioner":"ebs.csi.aws.com","volumeBindingMode":"WaitForFirstConsumer"}

Provisioner:           ebs.csi.aws.com
Parameters:            <none>
AllowVolumeExpansion:  <unset>
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     WaitForFirstConsumer
```

```
Events:                <none>
```

> **Note**
>
> 스토리지 클래스에서는 `WaitForFirstConsumer` 볼륨 바인딩 모드를 사용합니다. 이는 Pod에서 영구 볼륨 클레임이 있을 때까지는 볼륨이 활동적으로 프로비저닝되지 않음을 의미합니다. 자세한 내용을 알아보려면 Kubernetes 문서의 볼륨 바인딩 모드를 참조하세요.

6. 기본 네임스페이스에서 Pods를 관찰합니다. 몇 분 후 app Pod의 상태가 `Running`으로 변경됩니다.

```
kubectl get pods --watch
```

`Ctrl+C`를 입력하여 쉘 프롬프트로 돌아갑니다.

7. 기본 네임스페이스에 있는 영구 볼륨을 나열합니다. `default/ebs-claim` 클레임이 포함된 영구 볼륨을 검색합니다.

```
kubectl get pv
```

예제 출력은 다음과 같습니다.

```
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY
   STATUS   CLAIM               STORAGECLASS   REASON   AGE
pvc-37717cd6-d0dc-11e9-b17f-06fad4858a5a   4Gi        RWO            Delete
   Bound    default/ebs-claim   ebs-sc                  30s
```

8. 영구 볼륨에 관해 설명합니다. `pvc-37717cd6-d0dc-11e9-b17f-06fad4858a5a`를 이전 단계의 출력 값으로 변경합니다.

```
kubectl describe pv pvc-37717cd6-d0dc-11e9-b17f-06fad4858a5a
```

예제 출력은 다음과 같습니다.

```
Name:              pvc-37717cd6-d0dc-11e9-b17f-06fad4858a5a
Labels:            <none>
Annotations:       pv.kubernetes.io/provisioned-by: ebs.csi.aws.com
Finalizers:        [kubernetes.io/pv-protection external-attacher/ebs-csi-aws-com]
```

```
StorageClass:      ebs-sc
Status:            Bound
Claim:             default/ebs-claim
Reclaim Policy:    Delete
Access Modes:      RWO
VolumeMode:        Filesystem
Capacity:          4Gi
Node Affinity:
  Required Terms:
    Term 0:        topology.ebs.csi.aws.com/zone in [region-code]
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume source)
    Driver:            ebs.csi.aws.com
    VolumeHandle:      vol-0d651e157c6d93445
    ReadOnly:          false
    VolumeAttributes:      storage.kubernetes.io/
csiProvisionerIdentity=1567792483192-8081-ebs.csi.aws.com
Events:                <none>
```

Amazon EBS 볼륨 ID는 이전 출력에서 VolumeHandle의 값입니다.

9. Pod에서 볼륨에 데이터를 쓰고 있는지 확인합니다.

```
kubectl exec -it app -- cat /data/out.txt
```

예제 출력은 다음과 같습니다.

```
Wed May 5 16:17:03 UTC 2021
Wed May 5 16:17:08 UTC 2021
Wed May 5 16:17:13 UTC 2021
Wed May 5 16:17:18 UTC 2021
[...]
```

10. 작업을 완료하면 이 샘플 애플리케이션의 리소스를 삭제합니다.

```
kubectl delete -f manifests/
```

Amazon EBS CSI 마이그레이션 관련 자주 묻는 질문

> **Important**
>
> 버전 1.22 이전의 클러스터에서 Pods를 실행하고 있는 경우 서비스 중단이 발생하지 않도록 클러스터를 버전 1.23으로 업데이트하기 전에 Amazon EBS 드라이버를 설치해야 합니다.

Amazon EBS 컨테이너 스토리지 인터페이스(CSI) 마이그레이션 기능은 스토리지 작업 처리에 대한 책임을 Amazon EBS 인트리 EBS 스토리지 프로비저너에서 Amazon EBS CSI 드라이버로 이전합니다.

CSI 드라이버란 무엇인가요?

CSI 드라이버:

- Kubernetes 프로젝트 소스 코드에 있는 Kubernetes 'in-tree' 스토리지 드라이버 드라이버를 대체합니다.
- Amazon EBS와 같은 스토리지 공급자와 협력합니다.
- AWS와 같은 스토리지 제공업체에서 Kubernetes 릴리스 주기에 의존하지 않고 더 쉽게 기능을 릴리스하고 지원을 유지할 수 있는 간소화된 플러그인 모델을 제공합니다.

자세한 내용은 Kubernetes CSI 설명서에서 소개를 참조하세요.

CSI 마이그레이션이란 무엇인가요?

Kubernetes CSI 마이그레이션 기능은 스토리지 작업을 처리하는 책임을 `kubernetes.io/aws-ebs`와 같은 기존 in-tree 스토리지 플러그인에서 해당 CSI 드라이버로 이전합니다. 기존 `StorageClass`, `PersistentVolume`, `PersistentVolumeClaim`(PVC) 객체는 해당 CSI 드라이버가 설치되어 있는 한 계속 작동합니다. 기능이 활성화된 경우

- PVC를 활용하는 기존 워크로드는 항상 그렇듯이 계속 작동합니다.
- Kubernetes에서 모든 스토리지 관리 작업에 대한 제어를 CSI 드라이버에 전달합니다.

자세한 내용은 Kubernetes 블로그의 Kubernetes1.23: Kubernetes In-Tree to CSI Volume Migration Status Update(In-Tree에서 CSI 볼륨 마이그레이션 상태 업데이트)를 참조하세요.

in-tree 플러그인에서 CSI 드라이버로 마이그레이션하는 데 도움이 되도록 Amazon EKS 버전 1.23 이후 클러스터에서 기본적으로 CSIMigration 및 CSIMigrationAWS 플래그가 활성화됩니다. 이러한 플래그를 사용하면 클러스터가 in-tree API를 동일한 CSI API로 변환할 수 있습니다. 이 플래그는 Amazon EKS에서 관리하는 Kubernetes 컨트롤 플레인과 Amazon EKS 최적화 AMI에 구성된 kubelet 설정에서 설정합니다. Pods에서 클러스터의 Amazon EBS 볼륨을 사용하는 경우 클러스터를 버전 **1.23**으로 업데이트하기 전에 Amazon EBS CSI 드라이버를 설치해야 합니다. 그렇지 않으면 프로비저닝 및 탑재와 같은 볼륨 작업이 예상대로 작동하지 않을 수 있습니다. 자세한 내용은 Amazon EBS CSI 드라이버 섹션을 참조하세요.

> **Note**
>
> in-tree StorageClass 프로비저너는 kubernetes.io/aws-ebs라고 합니다. Amazon EBS CSI StorageClass 프로비저너는 ebs.csi.aws.com이라고 합니다.

버전 **1.23** 이후 클러스터에 **kubernetes.io/aws-ebs StorageClass** 볼륨을 탑재할 수 있나요?

예. Amazon EBS CSI 드라이버가 설치되어 있는 한 가능합니다. 새로 생성된 버전 1.23 이후 클러스터의 경우 클러스터 생성 프로세스의 일부로 Amazon EBS CSI 드라이버를 설치하는 것이 좋습니다. 또한 ebs.csi.aws.com 프로비저너를 기반으로 한 StorageClasses만을 사용하는 것이 좋습니다.

클러스터 컨트롤 플레인을 버전 1.23으로 업데이트했으며 아직 노드를 1.23으로 업데이트하지 않은 경우 CSIMigration 및 CSIMigrationAWS kubelet 플래그가 활성화되지 않습니다. 이 경우 in-tree 드라이버를 사용하여 kubernetes.io/aws-ebs 기반 볼륨을 탑재합니다. 그러나 kubernetes.io/aws-ebs 기반 볼륨을 사용하는 Pods를 예약할 수 있으려면 Amazon EBS CSI 드라이버가 여전히 설치되어 있어야 합니다. 다른 볼륨 작업이 성공하려면 드라이버도 필요합니다.

Amazon EKS **1.23** 이후 클러스터에서 **kubernetes.io/aws-ebs StorageClass** 볼륨을 프로비저닝할 수 있나요?

예. Amazon EBS CSI 드라이버가 설치되어 있는 한 가능합니다.

Amazon EKS에서 **kubernetes.io/aws-ebs StorageClass** 프로비저너가 제거되나요?

kubernetes.io/aws-ebsStorageClass프로비저닝 장치 및awsElasticBlockStore 볼륨 유형은 더 이상 지원되지 않지만 제거할 계획은 없습니다. 이러한 리소스는Kubernetes API의 일부로 취급됩니다.

Amazon EBS CSI 드라이버는 어떻게 설치하나요?

Amazon EBS CSI 드라이버 Amazon EKS 추가 기능을 설치하는 것이 좋습니다. Amazon EKS 추가 기능에 업데이트가 필요한 경우 사용자가 업데이트를 시작합니다. 그러면 Amazon EKS가 추가 기능을 업데이트합니다. 드라이버를 직접 관리하려면 오픈 소스 Helm 차트를 사용하여 설치할 수 있습니다.

> **Important**
>
> Kubernetes in-tree Amazon EBS 드라이버가 Kubernetes 컨트롤 플레인에서 실행됩니다. Amazon EKS 클러스터 IAM 역할에 할당된 IAM 권한을 사용하여 Amazon EBS 볼륨을 프로비저닝합니다. Amazon EBS CSI 드라이버가 노드에서 실행됩니다. 드라이버에 볼륨을 프로비저닝하려면 IAM 권한이 필요합니다. 자세한 내용은 Amazon EBS CSI 드라이버 IAM 역할 생성 섹션을 참조하세요.

Amazon EBS CSI 드라이버가 클러스터에 설치되어 있는지 확인하려면 어떻게 해야 하나요?

클러스터에 드라이버가 설치되어 있는지 확인하려면 다음 명령을 실행합니다.

```
kubectl get csidriver ebs.csi.aws.com
```

해당 설치가 Amazon EKS에서 관리되는지 확인하려면 다음 명령을 실행합니다.

```
aws eks list-addons --cluster-name my-cluster
```

Amazon EBS CSI 드라이버를 아직 설치하지 않은 경우 Amazon EKS에서 클러스터를 버전 **1.23**으로 업데이트할 수 없나요?

아니요.

클러스터를 버전 1.23으로 업데이트하기 전에 Amazon EBS CSI 드라이버를 설치하지 않은 경우 어떻게 해야 하나요? 클러스터를 업데이트한 후에 드라이버를 설치할 수 있나요?

예. 그러나 Amazon EBS CSI 드라이버가 필요한 볼륨 작업은 클러스터 업데이트 후 드라이버가 설치될 때까지 실패합니다.

새로 생성된 Amazon EKS 버전 **1.23** 이후 클러스터에 적용되는 기본 **StorageClass**는 무엇인가요?

기본 StorageClass 동작은 변경되지 않습니다. Amazon EKS는 각각의 새 클러스터에 gp2라는 kubernetes.io/aws-ebs 기반 StorageClass를 적용합니다. 새로 생성된 클러스터에서 이 StorageClass를 제거할 계획은 없습니다. 클러스터 기본 StorageClass와 별개로 볼륨 유형을 지정하지 않고 ebs.csi.aws.com 기반 StorageClass를 생성하는 경우 Amazon EBS CSI 드라이버는 기본적으로 gp3를 사용하도록 설정됩니다.

클러스터를 버전 **1.23**으로 업데이트할 때 Amazon EKS에서 기존 클러스터에 이미 있던 **StorageClasses**를 변경하나요?

아니요.

스냅샷을 이용하여 **kubernetes.io/aws-ebsStorageClass**에서 **ebs.csi.aws.com**으로 영구 볼륨을 어떻게 마이그레이션하나요?

영구 볼륨을 마이그레이션하려면 AWS 블로그의 Migrating Amazon EKS clusters from gp2 to gp3 EBS volumes(gp2에서 gp3 EBS 볼륨으로 Amazon EKS 클러스터 마이그레이션)를 참조하세요.

주석을 사용하여 Amazon EBS 볼륨을 수정하려면 어떻게 해야 하나요?

aws-ebs-csi-driver v1.19.0-eksbuild.2부터 PersistentVolumeClaim(PVC) 내의 주석을 사용하여 Amazon EBS 볼륨을 수정할 수 있습니다. 새로운 볼륨 수정 기능은 volumemodifier라는 추가 사이드카로 구현됩니다. 자세한 내용은 AWS 블로그의 Simplifying Amazon EBS volume migration and modification on Kubernetes using the EBS CSI Driver를 참조하세요.

Windows 워크로드에 대해 마이그레이션이 지원되나요?

예. 오픈 소스 차트 Helm을 사용하여 Amazon EBS CSI 드라이버를 설치하는 경우 node.enableWindows를 true로 설정합니다. Amazon EKS 추가 기능으로 Amazon EBS CSI 드라이버를 설치하는 경우 기본적으로 설정되어 있습니다. StorageClasses를 생성할 때 fsType을 ntfs와 같은 Windows 파일 시스템으로 설정합니다. 그런 다음 Windows 워크로드에 대한 볼륨 작업이 Linux 워크로드와 마찬가지로 Amazon EBS CSI 드라이버로 마이그레이션됩니다.

Amazon EFS CSI 드라이버

Amazon Elastic File System(Amazon EFS)은 완전히 탄력적인 서버리스 파일 스토리지를 제공하므로 스토리지 용량과 성능을 프로비저닝하거나 관리하지 않고도 파일 데이터를 공유할 수 있습니다.

Amazon EFS Container Storage Interface(CSI) 드라이버는 AWS에서 실행되는 Kubernetes 클러스터가 Amazon EFS 파일 시스템의 수명 주기를 관리할 수 있게 해주는 CSI 인터페이스를 제공합니다. 이 주제에서는 Amazon EFS CSI 드라이버를 Amazon EKS 클러스터에 배포하는 방법을 설명합니다.

고려 사항

- Amazon EFS CSI 드라이버는 Windows 기반 컨테이너 이미지와 호환되지 않습니다.
- Fargate 노드에는 영구 볼륨에 대해 동적 프로비저닝을 사용할 수 없지만 정적 프로비저닝은 사용할 수 있습니다.
- 동적 프로비저닝은 `1.2` 이상의 드라이버가 필요합니다. 모든 지원되는 Amazon EKS 클러스터 버전에서 드라이버 버전 `1.1`을 사용하여 영구 볼륨에 대해 정적 프로비저닝을 사용할 수 있습니다.
- 이 드라이버의 버전 `1.3.2` 이상은 Amazon EC2 Graviton 기반 인스턴스를 포함하여 Arm64 아키텍처를 지원합니다.
- 이 드라이버의 버전 `1.4.2` 이상은 파일 시스템 탑재에 FIPS 사용을 지원합니다.
- Amazon EFS의 리소스 할당량을 기록해 둡니다. 예를 들어, 각 Amazon EFS 파일 시스템에 대해 생성할 수 있는 액세스 포인트 할당량은 1,000개입니다. 자세한 내용은 변경할 수 없는 Amazon EFS 리소스 할당량을 참조하세요.

필수 조건

- 클러스터에 대한 기존 AWS Identity and Access Management(IAM) OpenID Connect(OIDC) 제공업체입니다. 이미 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 부분을 참조하세요.
- AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.
- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 `kubectl` 설치 또는 업데이트 부분을 참조하세요.

> **Note**
>
> AWS Fargate에서 실행되는 Pod는 Amazon EFS 파일 시스템을 자동으로 탑재합니다.

IAM 역할 생성

Amazon EFS CSI 드라이버가 파일 시스템과 상호 작용하려면 IAM 권한이 필요합니다. IAM 역할을 생성하고 필요한 AWS 관리형 정책을 연결합니다. `eksctl`, AWS Management Console 또는 AWS CLI를 사용할 수 있습니다.

> **Note**
>
> 이 절차의 특정 단계는 드라이버를 Amazon EKS 추가 기능으로 사용하기 위해 작성되었습니다. 자체 관리형 설치에 대한 자세한 내용은 GitHub에서 Set up driver permission을 참조하세요.

eksctl

`eksctl`을(를) 사용하여 Amazon EFS CSI 드라이버 IAM 역할 생성

다음 명령을 실행하여 IAM 역할을 생성합니다. *`my-cluster`*을(를) 클러스터 이름으로 바꾸고, *`AmazonEKS_EFS_CSI_DriverRole`*을(를) 역할의 이름으로 바꿉니다.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEFSCSIDriverPolicy \
    --approve
TRUST_POLICY=$(aws iam get-role --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

AWS Management Console

AWS Management Console을(를) 사용하여 Amazon EFS CSI 드라이버 IAM 역할 생성

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 역할(Roles) 페이지에서 역할 생성(Create role)을 선택합니다.
4. 신뢰할 수 있는 엔터티 선택(Select trusted entity) 페이지에서 다음을 수행합니다.
   a. 신뢰할 수 있는 엔터티 유형(Trusted entity type) 섹션에서 웹 자격 증명(Web identity)을 선택합니다.
   b. 보안 인증 공급자의 경우 클러스터에 대해 OpenID Connect 공급자 URL을 선택합니다(Amazon EKS의 개요에 표시된 대로).
   c. 대상(Audience)에서 `sts.amazonaws.com`을 입력합니다.
   d. 다음을 선택합니다.
5. 권한 추가(Add permissions) 페이지에서 다음을 수행합니다.
   a. 필터 정책(Filter policies) 상자에 *AmazonEFSCSIDriverPolicy*를 입력합니다.
   b. 검색에서 반환된 *AmazonEFSCSIDriverPolicy* 왼쪽에 있는 확인란을 선택합니다.
   c. 다음을 선택합니다.
6. 이름, 검토 및 생성(Name, review, and create) 페이지에서 다음을 수행합니다.
   a. 역할 이름(Role name)에 역할의 고유한 이름(예: ***AmazonEKS_EFS_CSI_DriverRole***)을 입력합니다.
   b. 태그 추가(선택 사항)에서 태그를 키-값 페어로 연결하여 메타데이터를 역할에 추가합니다. IAM에서 태그 사용에 대한 자세한 내용을 알아보려면 IAM 사용 설명서의 IAM 리소스에 태그 지정을 참조하세요.
   c. 역할 생성을 선택합니다.
7. 역할을 생성한 후 편집할 수 있도록 콘솔에서 이 역할을 선택하여 엽니다.
8. 신뢰 관계(Trust relationships) 탭을 선택한 후 신뢰 정책 편집(Edit trust policy)을 선택합니다.
9. 다음과 비슷한 줄을 찾습니다.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

이전 줄 위에 다음 줄을 추가합니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다. *EXAMPLED539D4633E53DE1B71EXAMPLE*을 클러스터의 OIDC 공급자 ID로 바꿉니다.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:efs-csi-*",
```

10. Condition 운영자를 "StringEquals"에서 "StringLike"(으)로 수정합니다.
11. 신뢰 정책 업데이트(Update Trust Policy)를 선택하여 종료합니다.

AWS CLI

AWS CLI을(를) 사용하여 Amazon EFS CSI 드라이버 IAM 역할 생성

1. 클러스터의 OIDC 공급자 URL을 확인합니다. *my-cluster*을 클러스터 이름으로 교체합니다. 명령의 출력이 None인 경우 사전 요구 사항을 검토합니다.

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

예제 출력은 다음과 같습니다.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. AssumeRoleWithWebIdentity 작업을 부여하는 IAM 역할을 생성합니다.

a. 다음 콘텐츠를 *aws-efs-csi-driver-trust-policy*.json라는 파일에 복사합니다. *111122223333*을 계정 ID로 바꿉니다. *EXAMPLED539D4633E53DE1B71EXAMPLE* 및 *region-code*를 이전 단계에서 반환된 값으로 바꿉니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 arn:aws:를 arn:aws-us-gov:로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
```

```
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:efs-csi-*",
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

b. 역할을 생성합니다. *AmazonEKS_EFS_CSI_DriverRole*를 다른 이름으로 변경할 수 있지만, 그렇게 할 경우 이후 단계에서도 변경해야 합니다.

```
aws iam create-role \
  --role-name AmazonEKS_EFS_CSI_DriverRole \
  --assume-role-policy-document file://"aws-efs-csi-driver-trust-
policy.json"
```

3. 다음 명령어로 필요한 AWS 관리형 정책을 역할에 연결합니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy \
  --role-name AmazonEKS_EFS_CSI_DriverRole
```

Amazon EFS CSI 드라이버 설치

Amazon EKS 추가 기능을 통해 Amazon EFS CSI 드라이버를 설치하는 것이 좋습니다. Amazon EKS 추가 기능을 클러스터에 추가하려면 추가 기능 생성 섹션을 참조하세요. Amazon EKS 추가 기능에 대한 자세한 내용은 Amazon EKS 추가 기능 섹션을 참조하세요. Amazon EKS 추가 기능을 사용할 수 없는 경우, 사용할 수 없는 이유에 대한 문제를 컨테이너 로드맵 GitHub 리포지토리에 제출하는 것이 좋습니다.

대신 Amazon EFS CSI 드라이버의 자체 관리형 설치를 원하는 경우 GitHub의 설치를 참조하세요.

Amazon Elastic 파일 시스템 생성

Amazon EFS 파일 시스템을 생성하려면 GitHub의 Create an Amazon EFS file system for Amazon EKS를 참조하세요.

샘플 애플리케이션 배포

다양한 샘플 앱을 배포하고 필요에 따라 수정할 수 있습니다. 자세한 내용은 GitHub에 있는 예제를 참조하세요.

Amazon FSx for Lustre CSI 드라이버

FSx for Lustre 컨테이너 스토리지(CSI) 드라이버에서는 Amazon EKS 클러스터가 FSx for Lustre 파일 시스템의 수명 주기를 관리할 수 있게 해주는 CSI 인터페이스를 제공합니다. 자세한 내용을 알아보려면 FSx for Lustre 사용 설명서를 참조하세요.

이 주제에서는 FSx for Lustre CSI 드라이버를 Amazon EKS 클러스터에 배포하고 제대로 작동하는지 확인하는 방법에 관해 설명합니다. 드라이버는 최신 버전을 사용하는 것이 좋습니다. 사용 가능한 버전은 GitHub의 CSI Specification Compatibility Matrix를 참조하세요.

> **Note**
>
> Fargate에서는 드라이버가 지원되지 않습니다.

사용할 수 있는 파라미터에 대한 자세한 설명과 드라이버의 기능을 보여주는 전체 예제는 GitHub의 FSx for Lustre 컨테이너 스토리지 인터페이스(CSI) 드라이버 프로젝트를 참조하세요.

필수 조건

수행해야 할 사항:

- AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.

- 디바이스 또는 `0.175.0`에 설치된 버전 AWS CloudShell 이상의 `eksctl` 명령줄 도구. `eksctl`을 설치 또는 업그레이드하려면 `eksctl` 설명서에서 Installation을 참조하세요.
- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 kubectl 설치 또는 업데이트 부분을 참조하세요.

다음 절차를 수행하면 FSx for Lustre CSI 드라이버를 사용하여 간단한 테스트 클러스터를 생성할 수 있으므로 클러스터의 작동 방식을 살펴볼 수 있습니다. 프로덕션 워크로드에는 테스트 클러스터를 사용하지 않는 것이 좋습니다. 본 자습서에서는 *example values*를 바꾸라고 언급된 경우를 제외하고는 이를 사용하는 것이 좋습니다. 프로덕션 클러스터의 단계를 완료하는 경우 모든 *example value*를 바꿀 수 있습니다. 변수가 여러 단계에서 설정 및 사용되고 서로 다른 터미널에 있지 않기 때문에 동일한 터미널에서 모든 단계를 완료하는 것이 좋습니다.

Amazon EKS 클러스터에 FSx for Lustre 드라이버 배포

1. 나머지 단계에서 사용할 몇 가지 변수를 설정합니다. *my-csi-fsx-cluster*를 생성하려는 테스트 클러스터의 이름으로 바꾸고, *region-code*를 테스트 클러스터를 생성하려는 AWS 리전으로 바꿉니다.

   ```
   export cluster_name=my-csi-fsx-cluster
   export region_code=region-code
   ```

2. 테스트 클러스터를 생성합니다.

   ```
   eksctl create cluster \
     --name $cluster_name \
     --region $region_code \
     --with-oidc \
     --ssh-access \
     --ssh-public-key my-key
   ```

   클러스터 프로비저닝에는 몇 분 정도 걸립니다. 클러스터를 생성하는 동안 여러 줄의 출력이 표시됩니다. 출력의 마지막 줄은 다음 예제와 유사합니다.

   ```
   [#]  EKS cluster "my-csi-fsx-cluster" in "region-code" region is ready
   ```

3. 다음 명령을 사용하여 드라이버에 대한 Kubernetes 서비스 계정을 생성하고 `AmazonFSxFullAccess` AWS 관리형 정책을 서비스 계정에 연결합니다. 클러스터가 AWS

GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

```
eksctl create iamserviceaccount \
  --name fsx-csi-controller-sa \
  --namespace kube-system \
  --cluster $cluster_name \
  --attach-policy-arn arn:aws:iam::aws:policy/AmazonFSxFullAccess \
  --approve \
  --role-name AmazonEKSFSxLustreCSIDriverFullAccess \
  --region $region_code
```

서비스 계정이 생성되면 여러 줄의 출력이 표시됩니다. 출력의 마지막 줄은 다음과 유사합니다.

```
[#]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/fsx-csi-controller-sa",
        create serviceaccount "kube-system/fsx-csi-controller-sa",
    } }
[#]  building iamserviceaccount stack "eksctl-my-csi-fsx-cluster-addon-
iamserviceaccount-kube-system-fsx-csi-controller-sa"
[#]  deploying stack "eksctl-my-csi-fsx-cluster-addon-iamserviceaccount-kube-
system-fsx-csi-controller-sa"
[#]  waiting for CloudFormation stack "eksctl-my-csi-fsx-cluster-addon-
iamserviceaccount-kube-system-fsx-csi-controller-sa"
[#]  created serviceaccount "kube-system/fsx-csi-controller-sa"
```

배포된 AWS CloudFormation 스택의 이름을 기록합니다. 이전 예제 출력에서 스택의 이름이 `eksctl-my-csi-fsx-cluster-addon-iamserviceaccount-kube-system-fsx-csi-controller-sa`로 지정되어 있습니다.

4. 다음 명령을 사용하여 드라이버를 배포합니다. *release-X.XX*를 원하는 브랜치로 바꿉니다. 마스터 브랜치는 현재 릴리스된 안정적 버전의 드라이버와 호환되지 않는 향후 기능을 포함할 수 있으므로 지원되지 않습니다. 릴리스된 최신 버전을 사용하는 것이 좋습니다. 활성 브랜치 목록은 GitHub의 aws-fsx-csi-driver 섹션을 참조하세요.

> **Note**
>
> GitHub의 aws-fsx-csi-driver에서 적용되는 내용을 볼 수 있습니다.

```
kubectl apply -k "github.com/kubernetes-sigs/aws-fsx-csi-driver/deploy/kubernetes/
overlays/stable/?ref=release-X.XX"
```

예제 출력은 다음과 같습니다.

```
serviceaccount/fsx-csi-controller-sa created
serviceaccount/fsx-csi-node-sa created
clusterrole.rbac.authorization.k8s.io/fsx-csi-external-provisioner-role created
clusterrole.rbac.authorization.k8s.io/fsx-external-resizer-role created
clusterrolebinding.rbac.authorization.k8s.io/fsx-csi-external-provisioner-binding
 created
clusterrolebinding.rbac.authorization.k8s.io/fsx-csi-resizer-binding created
deployment.apps/fsx-csi-controller created
daemonset.apps/fsx-csi-node created
csidriver.storage.k8s.io/fsx.csi.aws.com created
```

5. 생성된 역할에 대한 ARN을 기록합니다. 이전에 기록하지 않았고 더 이상 AWS CLI 출력에서 사용할 수 없는 경우 다음을 수행하면 AWS Management Console에서 볼 수 있습니다.

   a. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.

   b. 콘솔이 IAM 역할을 생성한 AWS 리전으로 설정되어 있는지 확인한 다음 스택(Stacks)을 선택합니다.

   c. 이름이 `eksctl-`*`my-csi-fsx-cluster`*`-addon-iamserviceaccount-kube-system-`*`fsx-csi-controller-sa`*인 스택을 선택합니다.

   d. 출력 탭을 선택합니다. 역할1 ARN이 출력(1) 페이지에 나열됩니다.

6. 다음 명령을 사용하여 이전에 생성한 서비스 계정을 추가하도록 드라이버 배포를 패치합니다. ARN을 기록해 둔 ARN으로 바꿉니다. *`111122223333`*을 계정 ID로 바꿉니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

```
kubectl annotate serviceaccount -n kube-system fsx-csi-controller-sa \
  eks.amazonaws.com/role-
arn=arn:aws:iam::111122223333:role/AmazonEKSFSxLustreCSIDriverFullAccess --
overwrite=true
```

예제 출력은 다음과 같습니다.

```
serviceaccount/fsx-csi-controller-sa annotated
```

Kubernetes 스토리지 클래스, 영구 볼륨 클레임 및 샘플 애플리케이션을 배포하여 CSI 드라이버가 작동하는지 확인

이 절차에서는 FSx for Lustre Container Storage Interface(CSI) 드라이버 GitHub 리포지토리를 사용하여 동적으로 프로비저닝된 FSx for Lustre 볼륨을 사용합니다.

1. 클러스터의 보안 그룹을 기록합니다. 이는 네트워킹 섹션 아래의 AWS Management Console에서 또는 다음을 AWS CLI 명령을 사용하여 볼 수 있습니다.

```
aws eks describe-cluster --name $cluster_name --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

2. Amazon FSx for Lustre 사용 설명서의 Amazon VPC 보안 그룹에 표시된 기준에 따라 Amazon FSx 파일 시스템에 대한 보안 그룹을 생성합니다. VPC의 경우 네트워킹 섹션 아래 표시된 대로 클러스터의 VPC를 선택합니다. “Lustre 클라이언트와 연결된 보안 그룹”의 경우 클러스터 보안 그룹을 사용합니다. 아웃바운드 규칙을 그대로 두어 모든 트래픽을 허용할 수 있습니다.
3. 다음 명령을 사용하여 스토리지 클래스 매니페스트를 다운로드합니다.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/
master/examples/kubernetes/dynamic_provisioning/specs/storageclass.yaml
```

4. `storageclass.yaml` 파일의 파라미터 섹션을 편집합니다. 모든 *`example value`*를 고유한 값으로 바꿉니다.

```
parameters:
  subnetId: subnet-0eabfaa81fb22bcaf
  securityGroupIds: sg-068000ccf82dfba88
  deploymentType: PERSISTENT_1
  automaticBackupRetentionDays: "1"
  dailyAutomaticBackupStartTime: "00:00"
  copyTagsToBackups: "true"
  perUnitStorageThroughput: "200"
  dataCompressionType: "NONE"
  weeklyMaintenanceStartTime: "7:09:00"
  fileSystemTypeVersion: "2.12"
```

- **subnetId** - Amazon FSx for Lustre 파일 시스템을 생성해야 하는 서브넷 ID입니다. Amazon FSx for Lustre는 일부 가용 영역에서 지원되지 않습니다. https://console.aws.amazon.com/fsx/에서 Amazon FSx for Lustre 콘솔을 열어 사용하려는 서브넷이 지원되는 가용 영역에 있는지 확인합니다. 서브넷은 노드를 포함할 수도 있고 다른 서브넷이나 VPC일 수도 있습니다.
  - 컴퓨팅 섹션에서 노드 그룹을 선택하여 AWS Management Console에서 노드 서브넷을 확인할 수 있습니다.
  - 지정한 서브넷이 노드가 있는 서브넷과 다른 경우 VPC가 연결되어 있어야 하며 보안 그룹에 필요한 포트가 열려 있는지 확인해야 합니다.
- **securityGroupIds** - 파일 시스템에 대해 생성된 보안 그룹의 ID.
- **deploymentType**(선택 사항) - 파일 시스템 배포 유형입니다. 유효한 값은 SCRATCH_1, SCRATCH_2, PERSISTENT_1 및 PERSISTENT_2입니다. 배포 유형에 대한 자세한 내용은 Amazon FSx for Lustre 파일 시스템 생성을 참조하세요.
- 기타 파라미터(선택 사항) - 기타 파라미터에 대한 자세한 내용을 알아보려면 GitHub의 StorageClass 편집을 참조하세요.

5. 스토리지 클래스 매니페스트를 생성합니다.

```
kubectl apply -f storageclass.yaml
```

예제 출력은 다음과 같습니다.

```
storageclass.storage.k8s.io/fsx-sc created
```

6. 영구 볼륨 클레임 매니페스트를 다운로드합니다.

```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/
master/examples/kubernetes/dynamic_provisioning/specs/claim.yaml
```

7. (선택 사항) claim.yaml 파일을 편집합니다. 스토리지 요구 사항과 이전 단계에서 선택한 deploymentType에 따라 *1200Gi*를 다음 증분 값 중 하나로 변경합니다.

```
storage: 1200Gi
```

- SCRATCH_2 및 PERSISTENT - **1.2 TiB**, **2.4 TiB** 또는 2.4TiB 대비 2.4TiB씩 증가합니다.
- SCRATCH_1 - **1.2 TiB**, **2.4 TiB**, **3.6 TiB** 또는 3.6TiB 대비 3.6TiB씩 증가합니다.

8. 영구 볼륨 클레임을 생성합니다.

```
kubectl apply -f claim.yaml
```

예제 출력은 다음과 같습니다.

```
persistentvolumeclaim/fsx-claim created
```

9. 파일 시스템이 프로비저닝되었는지 확인합니다.

```
kubectl describe pvc
```

예제 출력은 다음과 같습니다.

```
Name:          fsx-claim
Namespace:     default
StorageClass:  fsx-sc
Status:        Bound
[...]
```

> **Note**
>
> Status는 Bound로 변경되기 전에 5~10분 동안 Pending으로 표시될 수 있습니다. Status가 Bound가 될 때까지 다음 단계를 계속하지 마십시오. Status에 10분 이상 동안 Pending이 표시되는 경우 모든 문제를 해결하기 위한 참조로 Events의 경고 메시지를 사용합니다.

10. 샘플 애플리케이션을 배포합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-fsx-csi-driver/master/examples/kubernetes/dynamic_provisioning/specs/pod.yaml
```

11. 샘플 애플리케이션이 실행 중인지 확인합니다.

```
kubectl get pods
```

예제 출력은 다음과 같습니다.

```
NAME      READY   STATUS              RESTARTS   AGE
```

```
fsx-app   1/1     Running             0          8s
```

12. 파일 시스템이 애플리케이션에 의해 올바르게 탑재되었는지 확인합니다.

```
kubectl exec -ti fsx-app -- df -h
```

예제 출력은 다음과 같습니다.

```
Filesystem                  Size  Used Avail Use% Mounted on
overlay                      80G  4.0G   77G   5% /
tmpfs                        64M     0   64M   0% /dev
tmpfs                       3.8G     0  3.8G   0% /sys/fs/cgroup
192.0.2.0@tcp:/abcdef01     1.1T  7.8M  1.1T   1% /data
/dev/nvme0n1p1               80G  4.0G   77G   5% /etc/hosts
shm                          64M     0   64M   0% /dev/shm
tmpfs                       6.9G   12K  6.9G   1% /run/secrets/kubernetes.io/
serviceaccount
tmpfs                       3.8G     0  3.8G   0% /proc/acpi
tmpfs                       3.8G     0  3.8G   0% /sys/firmware
```

13. 샘플 앱에서 데이터가 FSx for Lustre 파일 시스템에 작성되었는지 확인합니다.

```
kubectl exec -it fsx-app -- ls /data
```

예제 출력은 다음과 같습니다.

```
out.txt
```

이 출력 예는 샘플 앱이 파일 시스템에 `out.txt` 파일을 성공적으로 기록했다는 것을 보여줍니다.

> **Note**
>
> 클러스터를 삭제하기 전에 FSx for Lustre 파일 시스템을 삭제해야 합니다. 자세한 내용은 FSx for Lustre 사용 설명서의 리소스 정리를 참조하세요.

Amazon FSx for NetApp ONTAP CSI 드라이버

NetApp's Astra Trident에서는 Container Storage Interface(CSI) 호환 드라이버를 사용하여 동적 스토리지 오케스트레이션을 제공합니다. 이를 사용하면 Amazon FSx for NetApp ONTAP 파일 시스템에서 지원하는 영구 볼륨(PV)의 수명 주기를 Amazon EKS 클러스터에서 관리할 수 있습니다. 시작하려면 Astra Trident 설명서의 Amazon FSx for NetApp ONTAP에서 Astra Trident 사용을 참조하세요.

Amazon FSx for NetApp ONTAP은 클라우드에서 완전관리형 ONTAP 파일 시스템을 시작하고 실행할 수 있는 스토리지 서비스입니다. ONTAP은 널리 채택된 데이터 액세스 및 데이터 관리 기능의 집합을 제공하는 NetApp's 파일 시스템 기술입니다. FSx for ONTAP은 완전관리형 AWS 서비스의 민첩성, 확장성 및 단순성을 온프레미스 NetApp 파일 시스템의 기능, 성능 및 API에 제공합니다. 자세한 내용은 FSx for ONTAP 사용 설명서를 참조하세요.

Amazon FSx for OpenZFS CSI 드라이버

Amazon FSx for OpenZFS는 온프레미스 ZFS 또는 기타 Linux 기반 파일 서버에서 AWS로 데이터를 손쉽게 이동할 수 있는 완전관리형 파일 스토리지 서비스입니다. 애플리케이션 코드나 데이터 관리 방법을 변경하지 않고도 이 작업을 수행할 수 있습니다. 또한 Amazon FSx for OpenZFS는 오픈 소스 OpenZFS 파일 시스템에 구축된 매우 안정적이고 확장 가능하며 효율적이고 기능이 풍부한 파일 스토리지를 제공합니다. 이러한 기능을 완전관리형 AWS 서비스의 민첩성, 확장성 및 단순성과 결합합니다. 자세한 내용은 Amazon FSx for OpenZFS User Guide를 참조하세요.

Amazon FSx for OpenZFS 컨테이너 스토리지(CSI) 드라이버에서는 Amazon EKS 클러스터가 Amazon FSx for OpenZFS 볼륨의 수명 주기를 관리할 수 있게 하는 CSI 인터페이스를 제공합니다. Amazon FSx for OpenZFS CSI 드라이버를 Amazon EKS 클러스터에 배포하려면 GitHub의 `aws-fsx-openzfs-csi-driver` 섹션을 참조하세요.

Amazon File Cache CSI 드라이버

Amazon File Cache는 데이터 저장 위치에 관계없이 파일 데이터를 처리하는 데 사용되는 완전 관리형 고속 캐시입니다.AWS Amazon File Cache는 처음 액세스하면 캐시에 데이터를 자동으로 로드하고 사용하지 않을 때는 데이터를 릴리스합니다. 자세한 내용은 Amazon SQS 사용 설명서를 참조하세요.

Amazon EBS Container Storage Interface(CSI) 드라이버는 클러스터가 Amazon EFS 파일 시스템의 수명 주기를 관리할 수 있게 해주는 CSI 인터페이스를 제공합니다. Amazon 파일 캐시 CSI 드라이버를 Amazon EKS `aws-file-cache-csi-driver`클러스터에 배포하려면 GitHub를 참조하십시오.

Mountpoint for Amazon S3 CSI 드라이버

Mountpoint for Amazon S3 Container Storage Interface(CSI) 드라이버를 사용하면 Kubernetes 애플리케이션이 파일 시스템 인터페이스를 통해 Amazon S3 객체에 액세스하여 애플리케이션 코드를 변경하지 않고도 총 처리량을 높일 수 있습니다. Mountpoint for Amazon S3 기반의 CSI 드라이버는 Amazon S3 버킷을 Amazon EKS 및 자체 관리형 Kubernetes 클러스터의 컨테이너가 액세스할 수 있는 볼륨으로 제공합니다. 이 주제에서는 Mountpoint for Amazon S3 CSI 드라이버를 Amazon EKS 클러스터에 배포하는 방법을 설명합니다.

고려 사항

- Mountpoint for Amazon S3 CSI 드라이버는 현재 Windows 기반 컨테이너 이미지와 호환되지 않습니다.
- Mountpoint for Amazon S3 CSI 드라이버는 AWS Fargate를 지원하지 않습니다. 하지만 Amazon EC2에서 실행되는 컨테이너(Amazon EKS 또는 사용자 지정 Kubernetes 설치)는 지원됩니다.
- Mountpoint for Amazon S3 CSI 드라이버는 정적 프로비저닝만 지원합니다. 동적 프로비저닝 또는 새 버킷의 생성은 지원되지 않습니다.

  > **Note**
  >
  > 정적 프로비저닝이란 `PersistentVolume` 객체의 `volumeHandle`에서 `bucketName`으로 지정된 기존 Amazon S3 버킷을 사용하는 것을 말합니다. 자세한 내용은 GitHub의 정적 프로비저닝을 참조하세요.

- Mountpoint for Amazon S3 CSI 드라이버가 탑재된 볼륨은 모든 POSIX 파일 시스템 기능을 지원하지는 않습니다. 파일 시스템 동작에 대한 자세한 내용은 GitHub의 Mountpoint for Amazon S3 파일 시스템 동작을 참조하세요.

필수 조건

- 클러스터에 대한 기존 AWS Identity and Access Management(IAM) OpenID Connect(OIDC) 제공업체입니다. 이미 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 부분을 참조하세요.
- 디바이스 또는 AWS CloudShell에 AWS CLI 버전 2.12.3 이후 버전이 설치 및 구성됩니다.
- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전

이 1.28인 경우 kubectl 버전 1.27, 1.28 또는 1.29를 함께 사용할 수 있습니다. kubectl을 설치하거나 업그레이드하려면 kubectl 설치 또는 업데이트 부분을 참조하세요.

IAM 정책 생성

Mountpoint for Amazon S3 CSI 드라이버가 파일 시스템과 상호 작용하려면 Amazon S3 권한이 필요합니다. 이 섹션에서는 필요한 권한을 부여하는 IAM 정책을 생성하는 방법을 소개합니다.

다음 정책 예시는 Mountpoint에 대한 IAM 권한 권장 사항을 따릅니다. 또는 AWS 관리형 정책 AmazonS3FullAccess를 사용할 수 있지만 이 관리형 정책은 Mountpoint에 필요한 것보다 더 많은 권한을 부여합니다.

Mountpoint에 대해 권장되는 권한에 대한 자세한 내용은 GitHub의 Mountpoint IAM 권한을 참조하세요.

IAM 콘솔에서 IAM 정책 생성

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 정책을 선택합니다.
3. 정책 페이지에서 정책 생성을 선택합니다.
4. 정책 편집기에서 JSON을 선택합니다.
5. 정책 편집기에서 다음을 복사하고 붙여 넣습니다.

> **Important**
>
> *DOC-EXAMPLE-BUCKET1*을 사용자의 Amazon S3 버킷 이름으로 대체합니다.

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Sid": "MountpointFullBucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET1"
```

```
            ]
        },
        {
            "Sid": "MountpointFullObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
            ]
        }
    ]
}
```

Amazon S3 Express One Zone 스토리지 클래스에 도입된 디렉터리 버킷은 범용 버킷과 다른 인증 메커니즘을 사용합니다. `s3:*` 작업을 사용하는 대신 `s3express:CreateSession` 작업을 사용해야 합니다. 디렉터리 버킷에 대한 자세한 내용은 Amazon S3 사용 설명서의 디렉터리 버킷을 참조하세요.

다음은 디렉터리 버킷에 사용할 최소 권한 정책의 예입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3express:CreateSession",
            "Resource": "arn:aws:s3express:aws-region:111122223333:bucket/DOC-
EXAMPLE-BUCKET1--az_id--x-s3"
        }
    ]
}
```

6. 다음을 선택합니다.
7. 검토 및 생성 페이지에서 정책의 이름을 지정합니다. 이 예시 연습에서는 `AmazonS3CSIDriverPolicy`라는 이름을 사용합니다.
8. 정책 생성을 선택합니다.

IAM 역할 생성

Mountpoint for Amazon S3 CSI 드라이버가 파일 시스템과 상호 작용하려면 Amazon S3 권한이 필요합니다. 이 섹션에서는 IAM 역할을 생성하여 이러한 권한을 위임하는 방법을 설명합니다. `eksctl`, IAM 콘솔 또는 AWS CLI를 사용하여 역할을 생성할 수 있습니다.

> **Note**
>
> IAM 정책 `AmazonS3CSIDriverPolicy`는 이전 섹션에서 생성되었습니다.

eksctl

`eksctl`을 사용하여 Mountpoint for Amazon S3 CSI 드라이버 IAM 역할 생성

IAM 역할 및 Kubernetes 서비스 계정을 생성하려면 다음 명령을 실행합니다. 또한 이러한 명령은 `AmazonS3CSIDriverPolicy` IAM 정책을 역할에 연결하고, Kubernetes 서비스 계정(`s3-csi-controller-sa`)에 IAM 역할 Amazon 리소스 이름(ARN)을 주석으로 추가하고, Kubernetes 서비스 계정 이름을 IAM 역할에 대한 신뢰 정책에 추가합니다.

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

IAM console

AWS Management Console을 사용하여 Mountpoint for Amazon S3 CSI 드라이버 IAM 역할 생성

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.

3. 역할(Roles) 페이지에서 역할 생성(Create role)을 선택합니다.
4. 신뢰할 수 있는 엔터티 선택(Select trusted entity) 페이지에서 다음을 수행합니다.
   a. 신뢰할 수 있는 엔터티 유형(Trusted entity type) 섹션에서 웹 자격 증명(Web identity)을 선택합니다.
   b. 보안 인증 공급자의 경우 클러스터에 대해 OpenID Connect 공급자 URL을 선택합니다(Amazon EKS의 개요에 표시된 대로).

      URL이 표시되지 않는 경우 필수 조건 섹션을 검토하세요.
   c. 대상(Audience)에서 `sts.amazonaws.com`을 입력합니다.
   d. 다음을 선택합니다.
5. 권한 추가(Add permissions) 페이지에서 다음을 수행합니다.
   a. 필터 정책(Filter policies) 상자에 **`AmazonS3CSIDriverPolicy`**를 입력합니다.

      > **Note**
      > 이 정책은 이전 섹션에서 생성되었습니다.

   b. 검색에서 반환된 `AmazonS3CSIDriverPolicy` 결과 왼쪽에 있는 확인란을 선택합니다.
   c. 다음을 선택합니다.
6. 이름, 검토 및 생성(Name, review, and create) 페이지에서 다음을 수행합니다.
   a. 역할 이름(Role name)에 역할의 고유한 이름(예: **`AmazonEKS_S3_CSI_DriverRole`**)을 입력합니다.
   b. 태그 추가(선택 사항)에서 태그를 키-값 페어로 연결하여 메타데이터를 역할에 추가합니다. IAM에서 태그 사용에 대한 자세한 내용을 알아보려면 IAM 사용 설명서의 IAM 리소스에 태그 지정을 참조하세요.
   c. 역할 생성을 선택합니다.
7. 역할을 생성한 후 편집할 수 있도록 콘솔에서 이 역할을 선택하여 엽니다.
8. 신뢰 관계(Trust relationships) 탭을 선택한 후 신뢰 정책 편집(Edit trust policy)을 선택합니다.
9. 다음과 비슷한 줄을 찾습니다.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud":
 "sts.amazonaws.com"
```

이전 줄의 끝에 쉼표를 추가한 다음 이전 줄 뒤에 다음 줄을 추가합니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다. *EXAMPLED539D4633E53DE1B71EXAMPLE*을 클러스터의 OIDC 공급자 ID로 바꿉니다.

```
"oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub":
 "system:serviceaccount:kube-system:s3-csi-*"
```

10. Condition 운영자를 "StringEquals"에서 "StringLike"로 변경합니다.
11. 신뢰 정책 업데이트(Update Trust Policy)를 선택하여 종료합니다.

AWS CLI

AWS CLI을 사용하여 Mountpoint for Amazon S3 CSI 드라이버 IAM 역할 생성

1. 클러스터의 OIDC 공급자 URL을 확인합니다. *my-cluster*를 클러스터 이름으로 바꿉니다. 명령의 출력이 None인 경우 사전 요구 사항을 검토합니다.

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

예제 출력은 다음과 같습니다.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

2. IAM 역할을 생성하여 Kubernetes 서비스 계정에 AssumeRoleWithWebIdentity 작업 권한을 부여합니다.

   a. 다음 콘텐츠를 *aws-s3-csi-driver-trust-policy*.json라는 파일에 복사합니다. *111122223333*을 계정 ID로 바꿉니다. *EXAMPLED539D4633E53DE1B71EXAMPLE* 및 *region-code*를 이전 단계에서 반환된 값으로 바꿉니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
```

```
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringLike": {
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:s3-csi-*",
          "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
        }
      }
    }
  ]
}
```

b. 역할을 생성합니다. *AmazonEKS_S3_CSI_DriverRole*를 다른 이름으로 변경할 수 있지만, 그렇게 할 경우 이후 단계에서도 변경해야 합니다.

```
aws iam create-role \
  --role-name AmazonEKS_S3_CSI_DriverRole \
  --assume-role-policy-document file://"aws-s3-csi-driver-trust-policy.json"
```

3. 다음 명령을 사용하여 이전에 생성한 IAM 정책을 역할에 연결합니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonS3CSIDriverPolicy \
  --role-name AmazonEKS_S3_CSI_DriverRole
```

> **Note**
>
> IAM 정책 AmazonS3CSIDriverPolicy는 이전 섹션에서 생성되었습니다.

4. 드라이버를 Amazon EKS 추가 기능으로 설치하는 경우에는 이 단계를 생략합니다. 자체 관리형 드라이버 설치의 경우 생성한 IAM 역할의 ARN으로 주석이 달린 Kubernetes 서비스 계정을 생성합니다.

a. 다음 콘텐츠를 *mountpoint-s3-service-account*.yaml이라는 파일에 저장합니다. *111122223333*을 계정 ID로 바꿉니다.

```
---
apiVersion: v1
```

```
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: aws-mountpoint-s3-csi-driver
  name: mountpoint-s3-csi-controller-sa
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn:
 arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
```

b. 클러스터에 Kubernetes 서비스 계정을 생성합니다. Kubernetes 서비스 계정(`mountpoint-s3-csi-controller-sa`)에는 *AmazonEKS_S3_CSI_DriverRole* 이름으로 생성된 IAM 역할을 사용하여 주석을 추가합니다.

```
kubectl apply -f mountpoint-s3-service-account.yaml
```

> **Note**
>
> 해당 절차에서 플러그인이 배포되면 `s3-csi-driver-sa`라는 서비스 계정을 생성하고 해당 계정을 사용하도록 구성됩니다.

Mountpoint for Amazon S3 CSI 드라이버 설치

Amazon EKS 추가 기능을 통해 Mountpoint for Amazon S3 CSI 드라이버를 설치할 수 있습니다. `eksctl`, AWS Management Console 또는 AWS CLI를 사용하여 추가 기능을 클러스터에 추가할 수 있습니다.

선택적으로 Mountpoint for Amazon S3 CSI 드라이버를 자체 관리형 설치로 설치할 수 있습니다. 자체 관리형 설치 방법에 대한 지침은 GitHub의 설치를 참조하세요.

eksctl

`eksctl`을 사용하여 Amazon S3 CSI 추가 기능 추가

다음 명령을 실행합니다. *my-cluster*를 클러스터 이름으로, *111122223333*을 계정 ID로, *AmazonEKS_S3_CSI_DriverRole*을 이전에 생성한 IAM 역할의 이름으로 바꿉니다.

```
eksctl create addon --name aws-mountpoint-s3-csi-driver --cluster my-cluster --
service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
 --force
```

기존 설정과 충돌하는 **--force** 옵션과 Amazon EKS 추가 기능 설정을 제거할 경우, Amazon EKS 추가 기능의 업데이트 작업이 실패하고 충돌을 해결하는 데 도움이 되는 오류 메시지가 표시됩니다. 이 옵션을 사용하여 이러한 설정을 덮어쓰기 때문에 이 옵션을 지정하기 전에 Amazon EKS 추가 기능이 관리해야 하는 설정을 관리하지 않는지 확인합니다. 이 설정의 기타 옵션에 대한 자세한 내용은 `eksctl` 설명서의 추가 기능을 참조하세요. Amazon EKS Kubernetes 필드 관리에 대한 자세한 내용은 Kubernetes 필드 관리 섹션을 참조하세요.

AWS Management Console

AWS Management Console을 사용하여 Amazon S3 CSI 추가 기능에 대한 Mountpoint 추가

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터를 선택합니다.
3. Amazon S3 CSI 추가 기능에 대한 Mountpoint를 구성하려는 클러스터의 이름을 선택합니다.
4. 추가 기능(Add-ons) 탭을 선택합니다.
5. 추가 기능 더 가져오기를 선택합니다.
6. 추가 기능 선택 페이지에서 다음을 수행합니다.
   a. Amazon EKS 추가 기능 섹션에서 Mountpoint for Amazon S3 CSI 드라이버 확인란을 선택합니다.
   b. 다음을 선택합니다.
7. 선택한 추가 기능 설정 구성 페이지에서 다음을 수행합니다.
   a. 사용할 버전(Version)을 선택합니다.
   b. IAM 역할 선택에서 Mountpoint for Amazon S3 CSI 드라이버 IAM 정책을 연결한 IAM 역할의 이름을 선택합니다.
   c. (선택 사항) 선택적 구성 설정을 확장할 수 있습니다. 충돌 해결 방법에서 재정의를 선택한 경우 기존 추가 기능에 대한 하나 이상의 설정을 Amazon EKS 추가 기능의 설정으로 덮어쓸 수 있습니다. 이 옵션을 사용 설정하지 않고 기존 설정과 충돌이 있는 경우 작업이 실패합니다. 결과 오류 메시지를 사용하여 충돌을 해결할 수 있습니다. 이 옵션을 선택하기 전에 Amazon EKS 추가 기능이 사용자가 자체 관리해야 하는 설정을 관리하지 않는지 확인하세요.
   d. 다음을 선택합니다.

8. 검토 및 추가 페이지에서 생성을 선택합니다. 추가 기능 설치가 완료되면 설치한 추가 기능이 표시됩니다.

AWS CLI

AWS CLI을 사용하여 Amazon S3 CSI 추가 기능에 대한 Mountpoint 추가

다음 명령을 실행합니다. *my-cluster*를 클러스터 이름으로, *111122223333*을 계정 ID로, *AmazonEKS_S3_CSI_DriverRole*을 이전에 생성한 역할의 이름으로 바꿉니다.

```
aws eks create-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-
driver \
  --service-account-role-arn
 arn:aws:iam::111122223333:role/AmazonEKS_S3_CSI_DriverRole
```

Mountpoint for Amazon S3 구성

대부분의 경우 버킷 이름만 사용하여 Mountpoint for Amazon S3를 구성할 수 있습니다. Mountpoint for Amazon S3 구성 지침은 GitHub의 Mountpoint for Amazon S3 구성을 참조하세요.

샘플 애플리케이션 배포

기존 Amazon S3 버킷의 드라이버에 정적 프로비저닝을 배포할 수 있습니다. 자세한 내용은 GitHub의 정적 프로비저닝을 참조하세요.

Mountpoint for Amazon S3 CSI 드라이버 제거

Amazon EKS 추가 기능을 제거하는 경우 다음 두 가지 옵션이 있습니다.

- 클러스터에 추가 기능 소프트웨어 보존 – 이 옵션은 모든 설정에 대한 Amazon EKS 관리를 제거합니다. 또한 업데이트를 시작한 후 Amazon EKS가 업데이트를 알리고 Amazon EKS 추가 기능을 자동으로 업데이트하는 기능을 제거합니다. 그러나 클러스터에 추가 기능 소프트웨어는 유지됩니다. 이 옵션을 사용하면 추가 기능을 Amazon EKS 추가 기능이 아닌 자체 관리형 설치 기능으로 만들 수 있습니다. 이 옵션을 사용하면 추가 기능에 대한 가동 중지 시간이 없습니다. 이 절차의 명령은 이 옵션을 사용합니다.
- 클러스터에서 추가 기능 소프트웨어 완전히 제거(Remove the add-on software entirely from your cluster) - 클러스터에 종속된 리소스가 없는 경우에만 클러스터에서 Amazon EKS 추가 기능을 제거

하는 것이 좋습니다. 이 옵션을 수행하려면 이 절차에서 사용하는 명령에서 `--preserve`를 삭제합니다.

추가 기능에 연결된 IAM 계정이 있는 경우 IAM 계정은 제거되지 않습니다.

`eksctl`, AWS Management Console 또는 AWS CLI를 사용하여 Amazon S3 CSI 추가 기능을 제거할 수 있습니다.

eksctl

`eksctl`을 사용하여 Amazon S3 CSI 추가 기능 제거

*`my-cluster`*를 클러스터 이름으로 바꾸고 다음 명령을 실행합니다.

```
eksctl delete addon --cluster my-cluster --name aws-mountpoint-s3-csi-driver --
preserve
```

AWS Management Console

AWS Management Console을 사용하여 Amazon S3 CSI 추가 기능 제거

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터를 선택합니다.
3. Amazon EBS CSI 추가 기능을 제거할 클러스터의 이름을 선택합니다.
4. 추가 기능(Add-ons) 탭을 선택합니다.
5. Mountpoint for Amazon S3 CSI 드라이버를 선택합니다.
6. 제거를 선택합니다.
7. 제거: aws-mountpoint-s3-csi-driver 확인 대화 상자에서 다음을 수행합니다.
   a. 추가 기능에 대한 Amazon EKS의 설정 관리를 중지하려면 클러스터에 보존을 선택합니다. 클러스터에 추가 소프트웨어를 유지하려면 이렇게 합니다. 이는 추가 기능의 모든 설정을 스스로 관리할 수 있도록 하기 위한 것입니다.
   b. **`aws-mountpoint-s3-csi-driver`**을 입력합니다.
   c. 제거를 선택합니다.

AWS CLI

AWS CLI을 사용하여 Amazon S3 CSI 추가 기능 제거

*my-cluster*를 클러스터 이름으로 바꾸고 다음 명령을 실행합니다.

```
aws eks delete-addon --cluster-name my-cluster --addon-name aws-mountpoint-s3-csi-driver --preserve
```

CSI 스냅샷 컨트롤러

컨테이너 스토리지 인터페이스(CSI) 스냅샷 컨트롤러를 사용하면 Amazon EBS CSI 드라이버와 같은 호환 CSI 드라이버에서 스냅샷 기능을 사용할 수 있습니다.

CSI 스냅샷 컨트롤러 사용 시 고려할 사항이 몇 가지 있습니다.

- 스냅샷 컨트롤러는 스냅샷 기능이 있는 CSI 드라이버와 함께 설치해야 합니다. Amazon EBS CSI 드라이버는 Amazon EBS CSI 관리 볼륨의 Amazon EBS 스냅샷 생성을 지원합니다. 설치 지침은 Amazon EBS CSI 드라이버 섹션을 참조하세요.
- Kubernetes는 프로비저너 `kubernetes.io/aws-ebs`에 `StorageClass`를 사용하는 Amazon EBS와 같이 CSI 마이그레이션을 통해 제공되는 볼륨의 스냅샷을 지원하지 않습니다. CSI 드라이버 프로비저너 `ebs.csi.aws.com`를 참조하는 `StorageClass`로 볼륨을 생성해야 합니다. CSI 마이그레이션에 대한 자세한 내용은 Amazon EBS CSI 마이그레이션 관련 자주 묻는 질문 섹션을 참조하세요.

Amazon EKS 관리형 추가 기능을 통해 CSI 스냅샷 컨트롤러를 설치하는 것이 좋습니다. Amazon EKS 추가 기능을 클러스터에 추가하려면 추가 기능 생성 섹션을 참조하세요. Amazon EKS 추가 기능에 대한 자세한 내용은 Amazon EKS 추가 기능 섹션을 참조하세요.

또는 Amazon EBS CSI 스냅샷 컨트롤러의 자체 관리형 설치를 원하는 경우 GitHub의 업스트림 Kubernetes `external-snapshotter`에서 Usage를 참조하세요.

Amazon EKS 네트워킹

Amazon EKS 클러스터는 VPC에서 생성됩니다. 포드 네트워킹은 Amazon VPC 컨테이너 네트워크 인터페이스(CNI) 플러그 인을 통해 제공됩니다. 이 장에서는 클러스터의 네트워킹에 대해 자세히 알아보기 위해 다음 주제를 다룹니다.

주제

Amazon EKS VPC 및 서브넷 요구 사항과 고려 사항

클러스터를 생성하는 경우 서로 다른 가용 영역에 있는 VPC 및 두 개 이상의 서브넷을 지정합니다. 이 주제에서는 클러스터에서 사용하는 VPC 및 서브넷에 대한 Amazon EKS별 요구 사항과 고려 사항을 간략하게 살펴봅니다. Amazon EKS에서 사용할 VPC가 없는 경우 Amazon EKS 제공 AWS CloudFormation 템플릿을 사용하여 생성할 수 있습니다. AWS Outposts에서 로컬 또는 확장 클러스터를 생성하는 경우 이 주제 대신 Amazon EKS 로컬 클러스터 VPC 및 서브넷 요구 사항과 고려 사항(Amazon EKS 로컬 클러스터 VPC 및 서브넷 요구 사항 및 고려 사항)를 참조하세요.

VPC 요구 사항 및 고려 사항

클러스터를 생성하는 경우 지정하는 VPC는 다음 요구 사항 및 고려 사항을 충족해야 합니다.

- VPC에는 생성하려는 클러스터, 노드 및 기타 Kubernetes 리소스에 사용 가능한 IP 주소가 충분해야 합니다. 사용하려는 VPC에 IP 주소가 충분하지 않은 경우 사용 가능한 IP 주소 수를 늘립니다.

  클러스터 구성을 업데이트하여 클러스터에서 사용하는 서브넷 및 보안 그룹을 변경하면 이 작업을 수행할 수 있습니다. AWS Management Console, AWS CLI의 최신 버전, AWS CloudFormation 및 `eksctl` 버전 `v0.164.0-rc.0` 이상에서 업데이트할 수 있습니다. 서브넷에 사용 가능한 IP 주소를 더 많이 제공하여 클러스터 버전을 성공적으로 업그레이드하려면 이 작업을 수행해야 할 수도 있습니다.

> **Important**
>
> 추가하는 모든 서브넷은 클러스터를 생성할 때 원래 제공한 것과 동일한 AZ 세트에 있어야 합니다. 새 서브넷은 다른 모든 요구 사항을 충족해야 합니다. 예를 들어 충분한 IP 주소가 있어야 합니다.
> 예를 들어 클러스터를 만들고 서브넷 4개를 지정했다고 가정합시다. 지정한 순서대로 첫 번째 서브넷은 `us-west-2a` 가용 영역에, 두 번째와 세 번째 서브넷은 `us-west-2b` 가용 영역에, 네 번째 서브넷은 `us-west-2c` 가용 영역에 있습니다. 서브넷을 변경하려면 세 가용 영역 각각에 하나 이상의 서브넷을 제공해야 하며, 서브넷은 원래 서브넷과 동일한 VPC에 있어야 합니다.

VPC의 CIDR 블록보다 많은 IP 주소가 필요한 경우, 추가적인 Classless Inter-Domain Routing(CIDR) 블록을 VPC에 연결하여 CIDR 블록을 추가할 수 있습니다. 클러스터를 생성하기 전이나 후에 프라이빗(RFC 1918) 및 퍼블릭(비RFC 1918) CIDR 블록을 VPC에 연결합니다. 클러스터가 VPC와 연결된 CIDR 블록을 인식하는 데 최대 5시간이 걸릴 수 있습니다.

공유 서비스 VPC를 통해 전송 게이트웨이를 사용하여 IP 주소 사용을 절약할 수 있습니다. 자세한 내용은 공유 서비스를 사용하여 격리된 VPC 및 하이브리드 네트워크의 Amazon EKS VPC 라우팅 가능 IP 주소 보존 패턴을 참조하세요.

- Kubernetes가 `IPv6` 주소를 Pods 및 서비스에 할당하게 하려는 경우 `IPv6` CIDR 블록을 VPC와 연결합니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 `IPv6` CIDR 블록을 VPC와 연결을 참조하세요.
- VPC는 DNS 호스트 이름과 DNS 확인을 모두 지원해야 합니다. 그렇지 않으면 노드가 클러스터에 등록할 수 없습니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 VPC에 대한 DNS 속성을 참조하세요.
- VPC에는 AWS PrivateLink를 사용하는 VPC 엔드포인트가 필요할 수 있습니다. 자세한 내용은 서브넷 요구 사항 및 고려 사항 단원을 참조하십시오.

Kubernetes `1.14` 또는 이전 버전을 사용하여 클러스터를 생성한 경우 Amazon EKS에서는 VPC에 다음 태그를 추가했습니다.

키	값
`kubernetes.io/cluster/` *`my-cluster`*	`owned`

이 태그는 Amazon EKS에서만 사용되었습니다. 서비스에 영향을 주지 않고 태그를 제거할 수 있습니다. 버전 1.15 이상인 클러스터에서는 사용되지 않습니다.

서브넷 요구 사항 및 고려 사항

클러스터를 생성하는 경우 Amazon EKS에서는 지정한 서브넷에 2~4개의 탄력적 네트워크 인터페이스를 생성합니다. 이러한 네트워크 인터페이스를 사용하면 클러스터와 VPC 간에 통신할 수 있습니다. 또한 이러한 네트워크 인터페이스를 통해 Kubernetes 및 `kubectl exec`와 같은 `kubectl logs` 기능을 사용 설정할 수 있습니다. 각 Amazon EKS 생성 네트워크 인터페이스의 설명에는 `Amazon EKS cluster-name`이 포함되어 있습니다.

클러스터를 생성하는 경우 Amazon EKS에서는 지정한 서브넷에 해당 네트워크 인터페이스를 생성합니다. 클러스터가 생성되면 Amazon EKS가 네트워크 인터페이스를 생성하는 서브넷을 변경할 수 있습니다. Kubernetes 버전의 클러스터를 업데이트하면 Amazon EKS에서는 클러스터가 생성한 원래 네트워크 인터페이스를 삭제하고 새로운 네트워크 인터페이스를 생성합니다. 이러한 네트워크 인터페이스는 원래 네트워크 인터페이스와 동일한 서브넷 또는 원래 네트워크 인터페이스와 다른 서브넷에서 생성될 수 있습니다. 네트워크 인터페이스가 생성되는 서브넷을 제어하려면 클러스터를 생성하거나 클러스터를 생성한 후 서브넷을 업데이트할 때 지정하는 서브넷 수를 2개로 제한할 수 있습니다.

클러스터의 서브넷 요구 사항

클러스터를 생성하거나 업데이트하는 경우 지정하는 서브넷은 다음 요구 사항을 충족해야 합니다.

- 각 서브넷에는 Amazon EKS에서 사용할 IP 주소가 6개 이상 있어야 합니다. 하지만 16개 이상의 IP 주소를 사용하는 것이 좋습니다.
- 서브넷은 AWS Outposts, AWS Wavelength 또는 AWS Local Zone에 상주할 수 없습니다. 그러나 VPC에 있는 경우 자체 관리형 노드 및 Kubernetes 리소스를 이러한 유형의 서브넷에 배포할 수 있습니다.
- 서브넷은 퍼블릭 또는 프라이빗일 수 있습니다. 그러나 가능한 경우 프라이빗 서브넷을 지정하는 것이 좋습니다. 퍼블릭 서브넷은 인터넷 게이트웨이로 이어지는 경로가 포함된 라우팅 테이블을 사용한 서브넷인 반면 프라이빗 서브넷은 인터넷 게이트웨이로 이어지는 경로가 포함되지 않은 라우팅 테이블을 사용한 서브넷입니다.
- 서브넷은 다음 가용 영역에 상주할 수 없습니다.

AWS 리전	지역명	허용되지 않는 가용 영역 ID
us-east-1	미국 동부(버지니아 북부)	use1-az3

AWS 리전	지역명	허용되지 않는 가용 영역 ID
us-west-1	미국 서부(캘리포니아 북부)	usw1-az2
ca-central-1	캐나다(중부)	cac1-az3

구성 요소별 IP 주소 패밀리 사용

다음 표에는 Amazon EKS의 각 구성 요소에서 사용하는 IP 주소 패밀리가 나와 있습니다. Network Address Translation(NAT) 또는 기타 호환성 시스템을 사용하여 테이블 항목 "No" 값이 있는 패밀리의 소스 IP 주소에서 이러한 구성 요소에 연결할 수 있습니다.

기능은 클러스터의 IP family(`ipFamily`) 설정에 따라 다를 수 있습니다. 이 설정은 Kubernetes에서 Services에 할당하는 CIDR 블록에 사용되는 IP 주소 유형을 변경합니다. 설정 값이 IPv4인 클러스터를 IPv4 cluster라 하고, 설정 값이 IPv6인 클러스터를 IPv6 cluster라 합니다.

구성 요소	**IPv4** 주소만	**IPv6** 주소만	이중 스택 주소
EKS API 퍼블릭 엔드포인트	예	아니요	아니요
EKS API VPC 엔드포인트	예	아니요	아니요
EKS Auth API 퍼블릭 엔드포인트	예[1]	예[1]	예[1]
EKS Auth API VPC 엔드포인트	예[1]	예[1]	예[1]
EKS 클러스터 퍼블릭 엔드포인트	예	아니요	아니요
EKS 클러스터 프라이빗 엔드포인트	예[2]	예[2]	아니요
EKS 클러스터 서브넷	예[2]	아니요	예[2]

구성 요소	**IPv4** 주소만	**IPv6** 주소만	이중 스택 주소
노드 기본 IP 주소	예[2]	아니요	예[2]
Service IP 주소의 클러스터 CIDR 범위	예[2]	예[2]	아니요
VPC CNI의 Pod IP 주소	예[2]	예[2]	아니요

Note

[1] 엔드포인트는 IPv4 및 IPv6 주소가 모두 포함된 이중 스택입니다. AWS 외부의 애플리케이션, 클러스터의 노드 및 클러스터 내부의 포드는 IPv4 또는 IPv6 중 하나를 통해 이 엔드포인트에 도달할 수 있습니다.

[2] 클러스터를 생성할 때 클러스터의 IP family(ipFamily) 설정에서 IPv4 클러스터와 IPv6 클러스터 중 하나를 선택하며, 이는 변경할 수 없습니다. 대신 다른 클러스터를 생성하고 워크로드를 마이그레이션할 때 다른 설정을 선택해야 합니다.

노드의 서브넷 요구 사항

클러스터를 생성할 때 지정하는 것과 동일한 서브넷에 노드 및 Kubernetes 리소스를 배포할 수 있습니다. 그러나 이 작업은 필요하지 않습니다. 클러스터를 생성할 때 지정하지 않은 서브넷에 노드 및 Kubernetes 리소스를 배포할 수 있기 때문입니다. 노드를 다른 서브넷에 배포하는 경우 Amazon EKS에서는 해당 서브넷에서 클러스터 네트워크 인터페이스를 생성하지 않습니다. 노드 및 Kubernetes 리소스를 배포하는 모든 서브넷은 다음 요구 사항을 충족해야 합니다.

- 서브넷에는 모든 노드와 Kubernetes 리소스를 배포할 수 있는 IP 주소가 충분해야 합니다.
- Kubernetes가 IPv6 주소를 Pods 및 서비스에 할당하도록 하려는 경우 서브넷에 연결된 IPv6 CIDR 블록 하나와 IPv4 CIDR 블록 하나가 있어야 합니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 IPv6 CIDR 블록을 서브넷과 연결을 참조하세요. 서브넷과 연결된 라우팅 테이블에는 IPv4 및 IPv6 주소로 이어지는 경로가 포함되어야 합니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 경로를 참조하세요. 포드는 IPv6 주소에만 할당됩니다. 그러나 Amazon EKS가 클러스터와 노드에 대해 생성하는 네트워크 인터페이스에는 IPv4 및 IPv6 주소가 할당됩니다.

- 인터넷에서 Pods로 인바운드 액세스가 필요한 경우 로드 밸런서와 인그레스를 배포할 수 있는 IP 주소가 충분한 퍼블릭 서브넷이 하나 이상 있어야 합니다. 로드 밸런서를 퍼블릭 서브넷에 배포할 수 있습니다. 로드 밸런서는 프라이빗 또는 퍼블릭 서브넷의 Pods에 로드 밸런싱할 수 있습니다. 가능하면 프라이빗 서브넷에 노드를 배포하는 것이 좋습니다.
- 노드를 퍼블릭 서브넷에 배포하려는 경우 서브넷이 IPv4 퍼블릭 주소 또는 IPv6 주소를 자동 할당해야 합니다. 연결된 IPv6 CIDR 블록이 있는 프라이빗 서브넷에 노드를 배포하는 경우 프라이빗 서브넷도 IPv6 주소를 자동 할당해야 합니다. 2020년 3월 26일 이후에 Amazon EKS AWS CloudFormation 템플릿을 사용하여 VPC를 배포한 경우 이 설정이 사용 설정됩니다. 템플릿을 사용하여 이 날짜 이전에 VPC를 배포했거나 자체 VPC를 사용하는 경우 이 설정을 수동으로 사용 설정해야 합니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 서브넷의 퍼블릭 IPv4 주소 지정 속성 수정 및 서브넷의 IPv6 주소 지정 속성 수정을 참조하세요.
- 노드를 배포하는 서브넷이 프라이빗 서브넷이고 해당 라우팅 테이블에 Network Address Translation(NAT) 디바이스(IPv4) 또는 발신 전용 게이트웨이(IPv6)로 이어지는 경로가 포함되지 않은 경우, AWS PrivateLink를 사용하는 VPC 엔드포인트를 사용자 VPC에 추가합니다. VPC 엔드포인트는 노드 및 Pods와 통신해야 하는 모든 AWS 서비스에 필요합니다. 예를 들면 Amazon ECR, Elastic Load Balancing, Amazon CloudWatch, AWS Security Token Service 및 Amazon Simple Storage Service(Amazon S3)가 있습니다. 엔드포인트에는 노드가 있는 서브넷이 포함되어야 합니다. 일부 AWS 서비스가 VPC 엔드포인트를 지원합니다. 자세한 내용을 알아보려면 AWS PrivateLink란? 및 AWS PrivateLink와 통합되는 AWS 서비스를 참조하세요. Amazon EKS 추가 요구 사항 목록은 프라이빗 클러스터 요구 사항 섹션을 참조하세요.
- 서브넷에 로드 밸런서를 배포하려는 경우 서브넷에는 다음 태그가 있어야 합니다.
  - 프라이빗 서브넷

키	값
kubernetes.io/role/internal-elb	1

  - 퍼블릭 서브넷

키	값
kubernetes.io/role/elb	1

이전에는 버전이 Kubernetes 이하인 1.18 클러스터가 생성되면 Amazon EKS에서 지정된 모든 서브넷에 다음 태그를 추가했습니다.

키	값
kubernetes.io/cluster/ *my-cluster*	shared

지금은 새 Kubernetes 클러스터를 생성하면 Amazon EKS에서 서브넷에 태그를 추가하지 않습니다. 태그가 이전에 1.19 이하 버전인 클러스터에서 사용된 서브넷에 있는 경우 클러스터가 최신 버전으로 업데이트될 때 태그가 서브넷에서 자동으로 제거되지 않았습니다. 버전 2.1.1 이하의 AWS Load Balancer Controller에는 이 태그가 필요합니다. 최신 버전의 Load Balancer Controller를 사용하는 경우 서비스를 중단하지 않고 태그를 제거할 수 있습니다.

eksctl 또는 Amazon EKS AWS CloudFormation VPC 템플릿 중 하나를 사용하여 VPC를 배포한 경우 다음이 적용됩니다.

- 2020년 3월 26일 또는 그 이후 - 퍼블릭 서브넷에서 퍼블릭 서브넷에 배포된 새 노드에 퍼블릭 IPv4 주소가 자동으로 할당됩니다.
- 2020년 3월 26일 이전 - 퍼블릭 서브넷에서 퍼블릭 서브넷에 배포된 새 노드에 퍼블릭 IPv4 주소가 자동으로 할당되지 않습니다.

이 변경은 다음과 같은 방식으로 퍼블릭 서브넷에 배포된 새 노드 그룹에 영향을 줍니다.

- 관리형 노드 그룹 - 2020년 4월 22일 또는 그 이후에 노드 그룹이 퍼블릭 서브넷에 배포된 경우 퍼블릭 서브넷에 퍼블릭 IP 주소가 자동으로 할당되도록 사용 설정되어 있어야 합니다. 자세한 내용을 알아보려면 서브넷의 퍼블릭 IPv4 주소 지정 속성 수정을 참조하세요.
- Linux, Windows 또는 Arm 자체 관리형 노드 그룹 - 2020년 3월 26일 또는 그 이후에 노드 그룹이 퍼블릭 서브넷에 배포된 경우 퍼블릭 서브넷에 퍼블릭 IP 주소가 자동으로 할당되도록 사용 설정되어 있어야 합니다. 그렇지 않으면 대신에 공용 IP 주소로 노드를 시작해야 합니다. 자세한 내용을 알아보려면 서브넷의 퍼블릭 IPv4 주소 지정 속성 수정 또는 인스턴스 시작 시 퍼블릭 IPv4 주소 할당을 참조하세요.

공유 서브넷 요구 사항 및 고려 사항

VPC 공유를 사용하여 서브넷을 동일한 AWS Organizations 내의 다른 AWS 계정과 공유할 수 있습니다. 다음 사항을 고려하여 공유 서브넷에서 Amazon EKS 클러스터를 생성할 수 있습니다.

- VPC 서브넷의 소유자가 참여자 계정과 서브넷을 공유해야 해당 계정에서 Amazon EKS 클러스터를 생성할 수 있습니다.
- 참여자는 VPC의 기본 보안 그룹을 사용하여 리소스를 시작할 수 없습니다. 보안 그룹은 소유자에게 속해 있기 때문입니다. 또한 참여자는 소유자 또는 다른 참여자가 소유한 보안 그룹을 사용하여 리소스를 시작할 수 없습니다.
- 공유 서브넷에서는 참여자와 소유자가 각 계정 내의 보안 그룹을 별도로 제어합니다. 서브넷 소유자는 참여자가 생성한 보안 그룹을 볼 수 있지만 이 보안 그룹에 대해 조치를 취할 수는 없습니다. 서브넷 소유자가 보안 그룹을 제거하거나 수정하고자 하는 경우 보안 그룹을 생성한 참가자가 조치를 취해야 합니다.
- 참여자가 클러스터를 생성한 경우 다음 고려 사항이 적용됩니다.
  - 클러스터 IAM 역할 및 노드 IAM 역할을 해당 계정에서 생성해야 합니다. 자세한 내용은 Amazon EKS 클러스터 IAM 역할 및 Amazon EKS 노드 IAM 역할 단원을 참조하세요.
  - 관리형 노드 그룹을 포함하여 모든 노드는 동일한 참여자가 생성해야 합니다.
- 공유 VPC 소유자는 참여자가 공유 서브넷에서 생성한 클러스터의 보기, 업데이트 또는 삭제가 불가능합니다. 여기에 더해 계정마다 액세스 권한이 다른 VPC 리소스가 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 소유자 및 참여자에 대한 책임 및 권한을 참조하세요.
- Amazon VPC CNI plugin for Kubernetes의 사용자 지정 네트워킹 기능을 사용하는 경우 소유자 계정에 나열된 가용 영역 ID 매핑을 사용하여 각각의 `ENIConfig`를 생성해야 합니다. 자세한 내용은 포드에 대한 사용자 지정 네트워킹 단원을 참조하십시오.

VPC 서브넷 공유에 관한 자세한 내용은 Amazon VPC 사용 설명서의 다른 계정과 VPC 공유를 참조하세요.

Amazon EKS 클러스터에 대해 VPC 생성

Amazon Virtual Private Cloud(Amazon VPC)를 사용하여 사용자가 정의한 가상 네트워크로 AWS 리소스를 시작할 수 있습니다. 이 가상 네트워크는 사용자의 자체 데이터 센터에서 운영하는 기존 네트워크와 매우 유사합니다. 그러나 Amazon Web Services Services의 확장 가능한 인프라를 사용하면 얻을 수 있는 이점이 있습니다. 프로덕션 Amazon EKS 클러스터를 배포하기 전에 Amazon VPC 서비스를 충분히 이해하는 것이 좋습니다. 자세한 내용은 Amazon VPC 사용 설명서를 참조하세요.

Amazon EKS 클러스터, 노드, Kubernetes 리소스는 VPC에 배포됩니다. Amazon EKS에서 기존 VPC를 사용하려는 경우 해당 VPC는 Amazon EKS VPC 및 서브넷 요구 사항과 고려 사항에서 설명하는 요구 사항을 충족해야 합니다. 이 주제에서는 Amazon EKS 제공 AWS CloudFormation 템플릿을 사용하

여 Amazon EKS 요구 사항을 충족하는 VPC를 생성하는 방법을 설명합니다. 템플릿을 배포했다면 템플릿에서 생성된 리소스를 보면서 생성된 리소스와 해당 리소스의 구성을 정확히 알 수 있습니다.

전제 조건

Amazon EKS용 VPC를 생성하려면 Amazon VPC 리소스를 생성하는 데 필요한 IAM 권한이 있어야 합니다. 이러한 리소스는 VPC, 서브넷, 보안 그룹, 라우팅 테이블 및 경로, 인터넷 및 NAT 게이트웨이입니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 퍼블릭 서브넷 예 정책을 사용하여 VPC 생성 및 서비스 승인 참조의 Amazon EC2에 대한 작업, 리소스 및 조건 키의 전체 목록을 참조하세요.

퍼블릭 및 프라이빗 서브넷을 사용하거나 퍼블릭 서브넷만 또는 프라이빗 서브넷만 있는 VPC를 생성할 수 있습니다.

Public and private subnets

이 VPC에는 2개의 퍼블릭 서브넷과 2개의 프라이빗 서브넷이 있습니다. 퍼블릭 서브넷은 인터넷 게이트웨이로 이어지는 라우팅이 있는 라우팅 테이블과 연결된 서브넷입니다. 그러나 프라이빗 서브넷의 라우팅 테이블에는 인터넷 게이트웨이로 이어지는 라우팅이 없습니다. 1개의 퍼블릭 서브넷과 1개의 프라이빗 서브넷이 동일한 가용 영역에 배포됩니다. 다른 퍼블릭 서브넷과 프라이빗 서브넷은 동일한 AWS 리전의 두 번째 가용 영역에 배포됩니다. 대부분 배포에 이 옵션을 사용하는 것이 좋습니다.

이 옵션을 사용하면 노드를 프라이빗 서브넷에 배포할 수 있습니다. 이 옵션을 사용하면 Kubernetes가 프라이빗 서브넷의 노드에서 실행되는 Pods로 트래픽을 로드 밸런싱할 수 있는 퍼블릭 서브넷에 로드 밸런서를 배포할 수 있습니다. 퍼블릭 IPv4 주소는 퍼블릭 서브넷에 배포된 노드에 자동으로 할당되지만 퍼블릭 IPv4 주소는 프라이빗 서브넷에 배포된 노드에는 할당되지 않습니다.

퍼블릭 및 프라이빗 서브넷의 노드에 IPv6 주소를 할당할 수도 있습니다. 프라이빗 서브넷의 노드는 클러스터 및 기타 AWS 서비스와 통신할 수 있으며, Pods는 IPv4 주소를 사용하여 NAT 게이트웨이를 통하거나 각 가용 영역에 배포된 IPv6 주소를 사용하여 아웃바운드 전용 인터넷 게이트웨이를 통해 인터넷으로 통신할 수 있습니다. 클러스터 또는 노드 이외의 소스에서 모든 인바운드 트래픽을 거부하지만 모든 아웃바운드 트래픽을 허용하는 규칙이 있는 보안 그룹이 배포됩니다. 서브넷에는 Kubernetes가 로드 밸런서를 배포할 수 있도록 태그가 지정됩니다.

VPC 생성

1. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
2. 탐색 모음에서 Amazon EKS를 지원하는 AWS 리전을 선택합니다.
3. 스택 생성(Create stack), 새 리소스 사용(표준)(With new resources (standard))를 선택합니다.

4. 사전 조건 - 템플릿 준비(Prerequisite - Prepare template)에서 템플릿 준비 완료(Template is ready)가 선택되었는지 확인한 다음 템플릿 지정(Specify template)에서 Amazon S3 URL을 선택합니다.
5. IPv4만 지원하는 VPC 또는 IPv4 및 IPv6를 지원하는 VPC를 생성할 수 있습니다. Amazon S3 URL 아래의 텍스트 영역에 다음 URL 중 하나를 붙여 넣고 다음(Next)을 선택합니다.

   - IPv4

     ```
     https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml
     ```

   - IPv4 및 IPv6

     ```
     https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-subnets.yaml
     ```

6. 스택 세부 정보 지정(Specify stack details) 페이지에서 파라미터를 입력하고 다음(Next)을 선택합니다.

   - 스택 이름(Stack name): AWS CloudFormation 스택에 대한 스택 이름을 선택합니다. 예를 들어 이전 단계에서 사용한 템플릿 이름을 사용할 수 있습니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다.
   - VpcBlock: VPC에 대한 IPv4 CIDR 범위를 선택합니다. 배포하는 각 노드, Pod 및 로드 밸런서에는 이 블록의 IPv4 주소가 할당됩니다. 기본 IPv4 값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 및 서브넷 크기 조정을 참조하세요. 또한 생성된 VPC CIDR 블록을 추가할 수도 있습니다. IPv6 VPC를 생성하는 경우 Amazon의 글로벌 유니캐스트 주소 공간에서 IPv6 CIDR 범위가 자동으로 할당됩니다.
   - PublicSubnet01Block: 퍼블릭 서브넷 1의 IPv4 CIDR 블록을 지정합니다. 기본값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다. IPv6 VPC를 생성하는 경우 템플릿 내에 이 블록이 지정됩니다.
   - PublicSubnet02Block: 퍼블릭 서브넷 2의 IPv4 CIDR 블록을 지정합니다. 기본값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다. IPv6 VPC를 생성하는 경우 템플릿 내에 이 블록이 지정됩니다.
   - PrivateSubnet01Block: 프라이빗 서브넷 1의 IPv4 CIDR 블록을 지정합니다. 기본값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다. IPv6 VPC를 생성하는 경우 템플릿 내에 이 블록이 지정됩니다.

- PrivateSubnet02Block: 프라이빗 서브넷 2의 IPv4 CIDR 블록을 지정합니다. 기본값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다. IPv6 VPC를 생성하는 경우 템플릿 내에 이 블록이 지정됩니다.

7. (선택 사항) 스택 옵션 구성(Configure stack options) 페이지에서 스택 리소스에 태그를 지정하고 다음(Next)을 선택합니다.
8. 검토(Review) 페이지에서 스택 생성(Create stack)을 선택합니다.
9. 스택이 생성된 후 콘솔에서 이를 선택하고 출력(Outputs)을 선택합니다.
10. 생성된 VPC의 VpcId를 기록합니다. 이 값은 클러스터 및 노드를 생성할 때 필요합니다.
11. 생성된 서브넷의 SubnetIds와 퍼블릭 서브넷 또는 프라이빗 서브넷으로 생성했는지 여부를 기록합니다. 2개 이상의 이 값은 클러스터 및 노드를 생성할 때 필요합니다.
12. IPv4 VPC를 생성한 경우 이 단계를 건너뜁니다. IPv6 VPC를 생성한 경우 템플릿으로 생성한 퍼블릭 서브넷에 대해 IPv6 주소 자동 할당 옵션을 사용 설정해야 합니다. 이 설정은 프라이빗 서브넷에 대해 이미 사용 설정되어 있습니다. 설정을 사용 설정하려면 다음 단계를 완료합니다.

    a. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
    b. 왼쪽 탐색 창에서 서브넷(Subnets)을 선택합니다.
    c. 퍼블릭 서브넷(***stack-name***/SubnetPublic01 또는 ***stack-name***/SubnetPublic02에 퍼블릭(public)이라는 단어가 포함됨) 중 하나를 선택하고 작업(Actions), 서브넷 설정 편집(Edit subnet settings)을 선택합니다.
    d. **IPv6** 주소 자동 할당 사용 설정(Enable auto-assign IPv6 address) 확인란을 선택한 다음 저장(Save)을 선택합니다.
    e. 다른 퍼블릭 서브넷에 대해 이전 단계를 다시 완료합니다.

Only public subnets

이 VPC에는 AWS 리전의 다른 가용 영역에 배포되는 3개의 퍼블릭 서브넷이 있습니다. 모든 노드에는 퍼블릭 IPv4 주소가 자동으로 할당되며 인터넷 게이트웨이를 통해 인터넷 트래픽을 송수신할 수 있습니다. 모든 인바운드 트래픽을 거부하고 모든 아웃바운드 트래픽을 허용하는 보안 그룹이 배포됩니다. 서브넷에는 Kubernetes가 로드 밸런서를 배포할 수 있도록 태그가 지정됩니다.

VPC 생성

1. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
2. 탐색 모음에서 Amazon EKS를 지원하는 AWS 리전을 선택합니다.

3. 스택 생성(Create stack), 새 리소스 사용(표준)(With new resources (standard))를 선택합니다.
4. 템플릿 준비(Prepare template)에서 템플릿 준비 완료(Template is ready)가 선택되었는지 확인한 다음 템플릿 소스(Template source)에서 Amazon S3 URL을 선택합니다.
5. Amazon S3 URL 아래의 텍스트 영역에 다음 URL을 붙여 넣고 다음(Next)을 선택합니다.

```
https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-vpc-sample.yaml
```

6. 세부 정보 지정(Specify Details) 페이지에서 파라미터를 입력하고 다음(Next)을 선택합니다.
   - 스택 이름: AWS CloudFormation 스택에 대한 스택 이름을 선택합니다. 예를 들어 ***amazon-eks-vpc-sample***라고 할 수 있습니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다.
   - VpcBlock: VPC에 대한 CIDR 블록을 선택합니다. 배포하는 각 노드, Pod 및 로드 밸런서에는 이 블록의 IPv4 주소가 할당됩니다. 기본 IPv4 값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 및 서브넷 크기 조정을 참조하세요. 또한 생성된 VPC CIDR 블록을 추가할 수도 있습니다.
   - Subnet01Block: 서브넷 1에 대한 CIDR 블록을 지정합니다. 기본값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다.
   - Subnet02Block: 서브넷 2에 대한 CIDR 블록을 지정합니다. 기본값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다.
   - Subnet03Block: 서브넷 3에 대한 CIDR 블록을 지정합니다. 기본값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다.
7. (선택 사항) 옵션(Options) 페이지에서 스택 리소스에 태그를 지정합니다. 다음을 선택합니다.
8. 검토(Review) 페이지에서 생성(Create)을 선택합니다.
9. 스택이 생성된 후 콘솔에서 이를 선택하고 출력(Outputs)을 선택합니다.
10. 생성된 VPC의 VpcId를 기록합니다. 이 값은 클러스터 및 노드를 생성할 때 필요합니다.
11. 생성된 서브넷에 대한 SubnetIds를 기록합니다. 2개 이상의 이 값은 클러스터 및 노드를 생성할 때 필요합니다.
12. (선택 사항) 이 VPC에 배포되는 모든 클러스터는 Pods 및 services에 프라이빗 IPv4 주소를 할당할 수 있습니다. 이 VPC에 배포된 클러스터가 Pods 및 services에 프라이빗 IPv6 주소를 할당하도록 하려면 VPC, 서브넷, 라우팅 테이블 및 보안 그룹을 업데이트해야 합니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 IPv4에서 IPv6로 기존 VPC 마이그레이션을

참조하세요. Amazon EKS를 사용하려면 서브넷에 `Auto-assign IPv6` 주소 옵션이 사용 설정되어 있어야 합니다. 기본적으로 사용 중지되어 있습니다.

Only private subnets

이 VPC에는 AWS 리전의 다른 가용 영역에 배포되는 3개의 프라이빗 서브넷이 있습니다. 서브넷에 배포된 리소스는 인터넷에 액세스할 수 없으며, 인터넷도 서브넷의 리소스에 액세스할 수 없습니다. 템플릿은 노드가 일반적으로 액세스해야 하는 여러 AWS 서비스에 대해 AWS PrivateLink를 사용하여 VPC 엔드포인트를 생성합니다. 노드에 아웃바운드 인터넷 액세스가 필요한 경우 VPC가 생성되면 각 서브넷의 가용 영역에서 퍼블릭 NAT 게이트웨이를 추가할 수 있습니다. 서브넷에 배포된 리소스를 제외한 모든 인바운드 트래픽을 거부하는 보안 그룹이 생성됩니다. 보안 그룹은 모든 아웃바운드 트래픽을 허용합니다. 서브넷에는 Kubernetes가 내부 로드 밸런서를 배포할 수 있도록 태그가 지정됩니다. 이 구성을 사용하여 VPC 생성하는 경우 추가 요구 사항 및 고려 사항에 대한 프라이빗 클러스터 요구 사항를 참조하세요.

VPC 생성

1. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
2. 탐색 모음에서 Amazon EKS를 지원하는 AWS 리전을 선택합니다.
3. 스택 생성(Create stack), 새 리소스 사용(표준)(With new resources (standard))를 선택합니다.
4. 템플릿 준비(Prepare template)에서 템플릿 준비 완료(Template is ready)가 선택되었는지 확인한 다음 템플릿 소스(Template source)에서 Amazon S3 URL을 선택합니다.
5. Amazon S3 URL 아래의 텍스트 영역에 다음 URL을 붙여 넣고 다음(Next)을 선택합니다.

   ```
   https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2020-10-29/amazon-eks-fully-private-vpc.yaml
   ```

6. 세부 정보 지정(Specify Details) 페이지에서 파라미터를 입력하고 다음(Next)을 선택합니다.

   - 스택 이름: AWS CloudFormation 스택에 대한 스택 이름을 선택합니다. 예를 들어 ***amazon-eks-fully-private-vpc***라고 할 수 있습니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다.
   - VpcBlock: VPC에 대한 CIDR 블록을 선택합니다. 배포하는 각 노드, Pod 및 로드 밸런서에는 이 블록의 `IPv4` 주소가 할당됩니다. 기본 `IPv4` 값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 및 서브넷 크기 조정을 참조하세요. 또한 생성된 VPC CIDR 블록을 추가할 수도 있습니다.

- PrivateSubnet01Block: 서브넷 1에 대한 CIDR 블록을 지정합니다. 기본값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다.
- PrivateSubnet02Block: 서브넷 2에 대한 CIDR 블록을 지정합니다. 기본값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다.
- PrivateSubnet03Block: 서브넷 3에 대한 CIDR 블록을 지정합니다. 기본값은 대부분의 구현에 충분한 IP 주소를 제공하지만 그렇지 않은 경우 변경할 수 있습니다.

7. (선택 사항) 옵션(Options) 페이지에서 스택 리소스에 태그를 지정합니다. 다음을 선택합니다.
8. 검토(Review) 페이지에서 생성(Create)을 선택합니다.
9. 스택이 생성된 후 콘솔에서 이를 선택하고 출력(Outputs)을 선택합니다.
10. 생성된 VPC의 VpcId를 기록합니다. 이 값은 클러스터 및 노드를 생성할 때 필요합니다.
11. 생성된 서브넷에 대한 SubnetIds를 기록합니다. 2개 이상의 이 값은 클러스터 및 노드를 생성할 때 필요합니다.
12. (선택 사항) 이 VPC에 배포되는 모든 클러스터는 Pods 및 services에 프라이빗 IPv4 주소를 할당할 수 있습니다. 이 VPC에 배포된 클러스터가 Pods 및 services에 프라이빗 IPv6 주소를 할당하도록 하려면 VPC, 서브넷, 라우팅 테이블 및 보안 그룹을 업데이트해야 합니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 IPv4에서 IPv6로 기존 VPC 마이그레이션을 참조하세요. Amazon EKS를 사용하려면 서브넷에 Auto-assign IPv6 주소 옵션이 사용 설정되어 있어야 합니다(기본적으로 사용 중지되어 있음).

Amazon EKS 보안 그룹 요구 사항 및 고려 사항

이 주제에서는 Amazon EKS 클러스터의 보안 그룹 요구 사항을 설명합니다.

클러스터를 생성할 때 Amazon EKS에서 `eks-cluster-sg-`*`my-cluster`*`-`*`uniqueID`*라는 보안 그룹을 만듭니다. 이 보안 그룹에는 다음의 기본 규칙이 있습니다.

규칙 타입	프로토콜	포트	소스(Source)	대상
인바운드	모두	모두	본인	
아웃바운드	모두	모두		0.0.0.0/0 (IPv4) or ::/0 (IPv6)

> **Important**
>
> 클러스터에 아웃바운드 규칙이 필요하지 않은 경우 해당 규칙을 제거할 수 있습니다. 규칙을 제거하더라도 클러스터 트래픽 제한에 나열된 최소 규칙은 여전히 있어야 합니다. 인바운드 규칙을 제거하면 클러스터가 업데이트될 때마다 Amazon EKS에서 해당 규칙을 다시 생성합니다.

Amazon EKS에서는 다음과 같은 태그를 보안 그룹에 추가합니다. 태그를 제거하면 클러스터가 업데이트될 때마다 Amazon EKS에서 태그를 보안 그룹에 다시 추가합니다.

키	값
kubernetes.io/cluster/ *my-cluster*	owned
aws:eks:cluster-name	*my-cluster*
Name	eks-cluster-sg- *my-cluster* -*uniqueid*

Amazon EKS는 이 보안 그룹을 생성하는 다음 리소스에 자동으로 연결합니다.

- 클러스터를 생성할 때 생성되는 2~4개의 탄력적 네트워크 인터페이스(이 문서의 나머지 부분에서는 네트워크 인터페이스로 언급)입니다.
- 생성한 모든 관리형 노드 그룹에 있는 노드의 네트워크 인터페이스입니다.

기본 규칙을 사용하면 클러스터와 노드 간에 모든 트래픽이 자유롭게 흐를 수 있으며 모든 아웃바운드 트래픽은 모든 대상에 대해 허용됩니다. 클러스터를 생성할 때 사용자 고유의 보안 그룹을 지정할 수 있습니다(선택 사항). 이렇게 하면 Amazon EKS는 사용자가 지정한 보안 그룹을 클러스터에 대해 생성하는 네트워크 인터페이스에 연결합니다. 그러나 사용자가 생성하는 노드 그룹에는 연결하지 않습니다.

클러스터의 네트워킹(Networking) 섹션의 AWS Management Console에서 해당 클러스터 보안 그룹의 ID를 확인할 수 있습니다. 또는 다음 AWS CLI 명령을 실행하여 이 작업을 수행할 수 있습니다.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId
```

클러스터 트래픽 제한

클러스터와 노드 사이에 열린 포트를 제한해야 하는 경우 기본 아웃바운드 규칙을 제거하고 클러스터에 필요한 다음과 같은 최소 규칙을 추가할 수 있습니다. 기본 인바운드 규칙을 제거하면 클러스터가 업데이트될 때마다 Amazon EKS에서 해당 규칙을 다시 생성합니다.

규칙 타입	프로토콜	Port	대상
아웃바운드	TCP	443	클러스터 보안 그룹
아웃바운드	TCP	10250	클러스터 보안 그룹
아웃바운드(DNS)	TCP 및 UDP	53	클러스터 보안 그룹

또한 다음 트래픽에 대한 규칙을 추가해야 합니다.

- 노드에서 노드 간 통신에 사용할 것으로 예상하는 모든 프로토콜 및 포트입니다.
- 노드가 실행 시 클러스터 내부 검사 및 노드 등록을 위해 Amazon EKS API에 액세스할 수 있는 아웃바운드 인터넷 액세스. 노드에 인터넷 액세스 권한이 없는 경우 추가 고려 사항으로 프라이빗 클러스터 요구 사항를 검토합니다.
- DockerHub와(과) 같이 이미지를 가져오는 데 필요한 Amazon ECR 또는 다른 컨테이너 레지스트리 API에서 컨테이너 이미지를 가져오기 위한 노드 액세스. 자세한 내용은 AWS 일반 참조의 AWS IP 주소 범위를 참조하세요.
- Amazon S3에 대한 노드 액세스.
- `IPv4` 및 `IPv6` 주소에 대해 별도의 규칙이 필요합니다.

규칙을 제한하려는 경우 변경된 규칙을 프로덕션 클러스터에 적용하기 전에 모든 Pods를 철저히 테스트하는 것이 좋습니다.

Kubernetes `1.14` 및 플랫폼 버전 `eks.3` 이하를 사용하여 클러스터를 처음 배포한 경우 다음을 고려합니다.

- 사용자에게 컨트롤 플레인과 노드 보안 그룹이 있을 수도 있습니다. 생성된 이러한 그룹에는 이전 표에 나열된 제한 규칙이 포함되어 있습니다. 이러한 보안 그룹은 더 이상 필요하지 않으며 제거할 수 있습니다. 그러나 클러스터 보안 그룹에 해당 그룹에 포함된 규칙이 들어 있는지 확인해야 합니다.

- 직접 API를 사용하여 클러스터를 배포했거나, AWS CLI 또는 AWS CloudFormation과 같은 도구를 사용하여 클러스터를 생성하고 클러스터 생성 시 보안 그룹을 지정하지 않은 경우, VPC의 기본 보안 그룹은 Amazon EKS가 생성한 클러스터 네트워크 인터페이스에 적용됩니다.

Amazon EKS 네트워킹 추가 기능

Amazon EKS 클러스터에 여러 네트워킹 추가 기능을 사용할 수 있습니다.

내장 추가 기능

> **Note**
>
> 콘솔을 사용하는 경우를 제외하고 어떤 방식으로든 클러스터를 생성하는 경우 각 클러스터에는 자체 관리형 버전의 내장 추가 기능이 함께 제공됩니다. 자체 관리형 버전은 AWS Management Console, AWS Command Line Interface 또는 SDK.에서 관리할 수 없습니다. 자체 관리형 추가 기능의 구성 및 업그레이드를 관리합니다.
> 자체 관리형 추가 기능 유형을 사용하는 대신 클러스터에 Amazon EKS 유형의 추가 기능을 추가하는 것이 좋습니다. 콘솔에서 클러스터를 생성한 경우 이러한 추가 기능의 Amazon EKS 유형이 설치됩니다.

Amazon VPC CNI plugin for Kubernetes

이 CNI 추가 기능은 탄력적 네트워크 인터페이스를 생성하여 Amazon EC2 노드에 연결합니다. 또한 이 추가 기능은 프라이빗 `IPv4` 또는 `IPv6` 주소를 VPC에서 각 Pod 및 서비스에 할당합니다. 이 추가 기능은 기본적으로 클러스터에 설치됩니다. 자세한 내용은 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 단원을 참조하십시오.

CoreDNS

CoreDNS은(는) Kubernetes 클러스터 DNS 역할을 할 수 있는 유연하고 확장 가능한 DNS 서버입니다. CoreDNS은(는) 클러스터의 모든 Pods에 대한 이름 확인 기능을 제공합니다. 이 추가 기능은 기본적으로 클러스터에 설치됩니다. 자세한 내용은 CoreDNS Amazon EKS 추가 기능을 사용한 작업 단원을 참조하십시오.

`kube-proxy`

이 추가 기능은 Amazon EC2 노드의 네트워크 규칙을 유지하고 Pods와(과)의 네트워크 통신을 활성화합니다. 이 추가 기능은 기본적으로 클러스터에 설치됩니다. 자세한 내용은 Kubernetes `kube-proxy` 추가 기능으로 작업하기 단원을 참조하십시오.

선택적 AWS 네트워킹 추가 기능

AWS Load Balancer Controller

`loadbalancer` 유형의 Kubernetes 서비스 객체를 배포하면 컨트롤러가 AWS Network Load Balancer를 생성합니다. Kubernetes 수신 객체를 생성하면 컨트롤러가 AWS Application Load Balancer를 생성합니다. Kubernetes에 내장된 레거시 클라우드 공급자 컨트롤러를 사용하는 대신 이 컨트롤러를 사용하여 Network Load Balancer를 프로비저닝하는 것이 좋습니다. 자세한 내용은 AWS Load Balancer Controller 설명서를 참조하세요.

AWS 게이트웨이 API 컨트롤러

이 컨트롤러를 통해 Kubernetes 게이트웨이 API를 사용하여 여러 Kubernetes 클러스터 간에 서비스를 연결할 수 있습니다. 컨트롤러는 Amazon VPC Lattice 서비스를 사용하여 Amazon EC2 인스턴스, 컨테이너 및 서버리스 함수에서 실행되는 Kubernetes 서비스를 연결합니다. 자세한 내용은 AWS Gateway API 컨트롤러 설명서를 참조하세요.

추가 기능에 대한 자세한 내용은 Amazon EKS 추가 기능 섹션을 참조하세요.

Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업

Amazon VPC CNI plugin for Kubernetes 추가 기능은 Amazon EKS 클러스터의 각 Amazon EC2 노드에 배포됩니다. 이 추가 기능은 탄력적 네트워크 인터페이스를 생성하여 Amazon EC2 노드에 연결합니다. 또한 이 추가 기능은 프라이빗 `IPv4` 또는 `IPv6` 주소를 VPC에서 각 Pod 및 서비스에 할당합니다.

추가 기능 버전은 클러스터의 각 Fargate 노드와 함께 배포되지만 Fargate 노드에서는 업데이트되지 않습니다. 호환되는 다른 CNI 플러그인을 Amazon EKS 클러스터에서 사용할 수 있지만 Amazon EKS에서 지원하는 유일한 CNI 플러그인입니다.

다음 표에는 각 Kubernetes 버전에 사용할 수 있는 Amazon EKS 추가 기능 유형의 최신 버전이 나열되어 있습니다.

Kubernetes 버전	**1.29**	**1.28**	**1.27**	**1.26**	**1.25**	**1.24**	**1.23**
Amazon EKS 유형의 VPC CNI 버전	v1.18.0-e	v1.18.0-e	v1.18.0-e	v1.18.0-e	v1.18.0-e	v1.18.0-e	v1.18.0-e

Kubernetes 버전	1.29	1.28	1.27	1.26	1.25	1.24	1.23
	ksbuild	ksbuild	ksbuild	ksbuild	ksbuild	ksbuild	ksbuild.1

> **Important**
>
> 이 추가 기능을 자체 관리하는 경우 표의 버전이 사용 가능한 자체 관리 버전과 다를 수 있습니다. 이 추가 기능의 자체 관리형 유형 업데이트에 대한 자세한 내용은 자체 관리형 추가 기능 업데이트 부분을 참조하세요.

필수 조건

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요.
- 클러스터에 대한 기존 AWS Identity and Access Management(IAM) OpenID Connect(OIDC) 제공업체입니다. 이미 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 부분을 참조하세요.
- AmazonEKS_CNI_Policy IAM 정책(클러스터에서 IPv4 패밀리를 사용하는 경우) 또는 IPv6 정책(클러스터에서 IPv6 패밀리를 사용하는 경우)이 연결된 IAM 역할. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오.
- CNI 플러그인 버전 1.7.0 이상을 사용하고 Amazon VPC CNI plugin for Kubernetes사용자 지정 Pod보안 정책을 사용하는 경우 기본 Amazon EKS Pod 보안 정책 삭제포드 보안 정책 부분을 참조하세요.
-

> **Important**
>
> Amazon VPC CNI plugin for Kubernetes 버전 v1.16.0~v1.16.1은 Kubernetes 버전 1.23 이하와의 호환성을 제거했습니다. VPC CNI 버전 v1.16.2는 Kubernetes 버전 1.23 이하 및 CNI 사양 v0.4.0과의 호환성을 복원합니다.

Amazon VPC CNI plugin for Kubernetes 버전 v1.16.0~v1.16.1은 CNI 사양 버전 v1.0.0을 구현합니다. CNI 사양 v1.0.0은 Kubernetes 버전 v1.24 이상을 실행하는 EKS 클러스터에서 지원됩니다. VPC CNI 버전 v1.16.0~v1.16.1 및 CNI 사양 v1.0.0은 Kubernetes 버전 v1.23 이하에

서 지원되지 않습니다. CNI 사양 `v1.0.0`에 대한 자세한 내용은 Container Network Interface (CNI) Specification을 참조하세요.

고려 사항

- 버전은 `major-version.minor-version.patch-version-eksbuild.build-number`(으)로 지정됩니다.
- 각 기능의 버전 호환성 확인

  Amazon VPC CNI plugin for Kubernetes의 각 릴리스의 일부 기능에는 특정 Kubernetes 버전이 필요합니다. 다른 Amazon EKS 기능을 사용할 때 특정 버전의 추가 기능이 필요한 경우 기능 설명서에 설명되어 있습니다. 이전 버전을 실행할 특별한 이유가 없는 한 최신 버전을 실행하는 것이 좋습니다.

Amazon EKS 추가 기능 생성

추가 기능의 Amazon EKS 유형 생성.

1. 클러스터에 설치된 추가 기능의 버전을 확인하세요.

   ```
   kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
    | cut -d : -f 3
   ```

   예제 출력은 다음과 같습니다.

   ```
   v1.12.6-eksbuild.2
   ```

2. 클러스터에 설치된 추가 기능의 유형을 확인하세요. 클러스터를 생성하는 데 사용한 도구에 따라 현재 클러스터에 Amazon EKS 추가 기능이 유형이 설치되어 있지 않을 수 있습니다. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다.

   ```
   $ aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    addon.addonVersion --output text
   ```

   버전 번호가 반환되는 경우 Amazon EKS 유형의 추가 기능이 클러스터에 설치되고 본 절차의 나머지 단계를 완료할 필요가 없습니다. 오류가 번호가 반환되는 경우 Amazon EKS 유형의 추가 기능이 클러스터에 설치되지 않습니다. 이 절차의 나머지 단계를 완료하여 설치하세요.
3. 현재 설치된 추가 기능의 구성을 저장합니다.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. AWS CLI를 사용하여 추가 기능을 생성합니다. AWS Management Console 또는 `eksctl`를 사용하여 추가 기능을 만들려면 추가 기능 생성 을 참조하여 추가 기능 이름을 `vpc-cni`로 지정하십시오. 다음 명령을 디바이스에 복사합니다. 필요에 따라 명령을 다음과 같이 수정한 다음에 수정한 명령을 실행합니다.

   - *my-cluster*를 클러스터 이름으로 바꿉니다.
   - 클러스터 버전의 최신 버전 표에 나와 있는 최신 버전으로 *v1.18.0-eksbuild.1*을(를) 교체하십시오.
   - *111122223333*을 계정 ID로, *AmazonEKSVPCCNIRole*을 생성한 기존 IAM 역할 이름으로 바꿉니다. 역할을 지정하려면 클러스터의 IAM OpenID Connect(OIDC) 제공업체가 있어야 합니다. 클러스터의 해당 제공업체가 이미 있는지 아니면 생성해야 하는지 결정하려면 클러스터에 대한 IAM OIDC 공급자 생성 부분을 참조하세요.

   ```
   aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
   version v1.18.0-eksbuild.1 \
       --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
   ```

   Amazon EKS 추가 기능의 기본 설정과 충돌하는 사용자 지정 설정을 현재 추가 기능에 적용한 경우 생성이 실패할 수 있습니다. 생성에 실패하면 문제 해결에 도움이 될 수 있는 오류를 받게 됩니다. 또는 이전 명령에 **`--resolve-conflicts OVERWRITE`**을(를) 추가할 수 있습니다. 이렇게 하면 추가 기능이 기존 사용자 지정 설정을 덮어쓸 수 있습니다. 추가 기능을 만든 후에는 사용자 지정 설정으로 업데이트할 수 있습니다.

5. 클러스터 Kubernetes 버전에 맞는 추가 기능의 최신 버전이 클러스터에 추가되었는지 확인합니다. *my-cluster*를 클러스터 이름으로 바꿉니다.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
    addon.addonVersion --output text
   ```

   추가 기능 생성이 완료되는 데 몇 초 정도 걸릴 수 있습니다.

   예제 출력은 다음과 같습니다.

   ```
   v1.18.0-eksbuild.1
   ```

6. 원래 추가 기능을 사용자 지정 설정했다면 Amazon EKS 추가 기능을 생성하기 전에 이전 단계에서 저장한 구성을 사용하여 Amazon EKS 추가 기능을 사용자 지정 설정으로 업데이트하십시오.
7. (선택 사항)`cni-metrics-helper` 클러스터에 설치합니다. 지표 헬퍼는 네트워크 인터페이스 및 IP 주소 정보를 수집하고 클러스터 수준에서 지표를 집계하고 이 지표를 Amazon CloudWatch에 게시하는 데 사용할 수 있는 도구입니다. 자세한 내용은 GitHub에서 metrics.proto를 참조하세요.

Amazon EKS 추가 기능 업데이트

추가 기능의 Amazon EKS 유형 업데이트. 클러스터에 Amazon EKS 유형의 추가 기능을 추가하지 않은 경우 이 절차를 완료하는 대신 추가하거나 자체 관리형 추가 기능 업데이트 부분을 참조하세요.

1. 클러스터에 설치된 추가 기능의 버전을 확인하세요. *my-cluster*을 클러스터 이름으로 교체합니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

예제 출력은 다음과 같습니다.

```
v1.12.6-eksbuild.2
```

반환된 버전이 최신 버전 표에 있는 클러스터의 Kubernetes 버전과 동일한 경우 클러스터에 이미 최신 버전이 설치되어 있으므로 이 절차의 나머지 부분을 완료하지 않아도 됩니다. 출력에 버전 번호 대신 오류가 표시되면 클러스터에 설치된 추가 기능의 Amazon EKS 유형이 없는 것입니다. 이 절차로 업데이트하려면 먼저 추가 기능을 생성해야 합니다.

2. 현재 설치된 추가 기능의 구성을 저장합니다.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

3. AWS CLI를 사용하여 추가 기능 업데이트. AWS Management Console 또는 `eksctl`를 사용하여 추가 기능을 업데이트하려면 추가 기능 업데이트 부분을 참조하세요. 다음 명령을 디바이스에 복사합니다. 필요에 따라 명령을 다음과 같이 수정한 다음에 수정한 명령을 실행합니다.
   - *my-cluster*를 클러스터 이름으로 바꿉니다.
   - 클러스터 버전의 최신 버전 표에 나와 있는 최신 버전으로 *v1.18.0-eksbuild.1*을(를) 교체하십시오.

- *111122223333*을 계정 ID로, *AmazonEKSVPCCNIRole*을 생성한 기존 IAM 역할 이름으로 바꿉니다. 역할을 지정하려면 클러스터의 IAM OpenID Connect(OIDC) 제공업체가 있어야 합니다. 클러스터의 해당 제공업체가 이미 있는지 아니면 생성해야 하는지 결정하려면 클러스터에 대한 IAM OIDC 공급자 생성 부분을 참조하세요.
- **--resolve-conflicts** *##* 옵션에서는 추가 기능의 기존 구성 값을 보존합니다. 추가 기능 설정에 사용자 지정 값을 설정하고 이 옵션을 사용하지 않는 경우 Amazon EKS에서는 기본값으로 해당 값을 덮어씁니다. 이 옵션을 사용하는 경우에는 추가 기능을 업데이트하기 전에 프로덕션 클러스터에서 비프로덕션 클러스터의 필드 및 값 변경 사항을 테스트하는 것이 좋습니다. 이 값을 OVERWRITE로 변경하면 모든 설정이 Amazon EKS 기본값으로 변경됩니다. 설정에 사용자 지정 값을 설정한 경우 Amazon EKS 기본값으로 해당 값을 덮어쓸 수도 있습니다. 이 값을 none으로 변경하면 Amazon EKS에서는 설정의 값을 변경하지 않지만 업데이트에 실패할 수도 있습니다. 업데이트에 실패하면 충돌 해결에 도움이 되는 오류 메시지가 표시됩니다.
- 구성 설정을 업데이트하지 않는 경우 **--configuration-values '{*"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}*}'**을(를) 명령에서 제거하십시오. 구성 설정을 업데이트하는 경우 *"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}*를 설정하려는 설정으로 바꾸십시오. 이 예에서는 AWS_VPC_K8S_CNI_EXTERNALSNAT 환경 변수가 true로 설정되어 있습니다. 지정한 값은 구성 스키마에 유효해야 합니다. 구성 스키마를 모르는 경우 *v1.18.0-eksbuild.1*을 구성을 확인할 추가 기능의 버전 번호로 바꿔 **aws eks describe-addon-configuration --addon-name vpc-cni --addon-version *v1.18.0-eksbuild.1***을 실행합니다. 스키마가 출력에 반환됩니다. 기존 사용자 지정 구성이 있는데 이를 모두 제거하고 모든 설정의 값을 Amazon EKS 기본값으로 다시 설정하려면 명령에서 *"env":{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}*를 제거하여 **{}**을(를) 비워 두십시오. 각 설정에 대한 설명은 GitHub의 CNI 구성 변수를 참조하세요.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.18.0-eksbuild.1 \
    --service-account-role-arn arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
 \
    --resolve-conflicts PRESERVE --configuration-values '{"env":
{"AWS_VPC_K8S_CNI_EXTERNALSNAT":"true"}}'
```

업데이트가 완료되는 데 몇 초 정도 걸릴 수 있습니다.

4. 추가 기능 버전이 업데이트되었는지 확인합니다. *my-cluster*를 클러스터 이름으로 바꿉니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

업데이트가 완료되는 데 몇 초 정도 걸릴 수 있습니다.

예제 출력은 다음과 같습니다.

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
        "addonVersion": "v1.18.0-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/vpc-
cni/74c33d2f-b4dc-8718-56e7-9fdfa65d14a9",
        "createdAt": "2023-04-12T18:25:19.319000+00:00",
        "modifiedAt": "2023-04-12T18:40:28.683000+00:00",
        "serviceAccountRoleArn":
 "arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole",
        "tags": {},
        "configurationValues": "{\"env\":{\"AWS_VPC_K8S_CNI_EXTERNALSNAT\":\"true
\"}}"
    }
}
```

자체 관리형 추가 기능 업데이트

> **Important**
>
> 자체 관리형 추가 기능 유형을 사용하는 대신 클러스터에 Amazon EKS 유형의 추가 기능을 추가하는 것이 좋습니다. 유형 간의 차이를 잘 모르는 경우 the section called “Amazon EKS 추가 기능” 부분을 참조하세요. Amazon EKS 추가 기능을 클러스터에 추가하는 방법에 대한 자세한 내용은 the section called “추가 기능 생성 ” 섹션을 참조하세요. Amazon EKS 추가 기능을 사용할 수 없는 경우, 사용할 수 없는 이유에 대한 문제를 컨테이너 로드맵 GitHub 리포지토리에 제출하는 것이 좋습니다.

1. 클러스터에 Amazon EKS 추가 기능 유형이 설치되어 있는지 확인하세요. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

오류 메시지가 번호가 반환되는 경우 Amazon EKS 유형의 추가 기능이 클러스터에 설치되지 않습니다. 추가 기능을 자체 관리하려면 이 절차의 나머지 단계를 완료하여 추가 기능을 업데이트하세요. 버전 번호가 반환되는 경우 Amazon EKS 유형의 추가 기능이 클러스터에 설치됩니다. 업데이트하려면 이 절차를 수행하는 대신 추가 기능 업데이트의 절차를 수행하세요. 추가 기능 유형 간의 차이를 잘 모르는 경우 Amazon EKS 추가 기능 부분을 참조하세요.

2. 클러스터에 현재 설치된 컨테이너 이미지의 버전을 확인하세요.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
 | cut -d : -f 3
```

예제 출력은 다음과 같습니다.

```
v1.12.6-eksbuild.2
```

출력에 빌드 번호가 포함되지 않을 수 있습니다.

3. 현재 설정을 백업하면 버전을 업데이트한 후에도 동일한 설정을 구성할 수 있습니다.

```
kubectl get daemonset aws-node -n kube-system -o yaml > aws-k8s-cni-old.yaml
```

4. 사용 가능한 버전을 검토하고 업데이트하려는 버전의 변경 사항을 파악하려면 GitHub에서 releases의 내용을 참조하세요. GitHub에서 최신 버전을 사용할 수 있더라도 사용 가능한 최신 버전 표에 나열된 것과 동일한 major.minor.patch 버전으로 업데이트하는 것이 좋습니다. 표에 나열된 빌드 버전은 GitHub에 나열된 자체 관리형 버전에 지정되어 있지 않습니다. 다음 옵션 중 하나로 작업을 완료하여 버전을 업데이트합니다.

   - 추가 기능에 대한 사용자 지정 설정이 없는 경우 GitHub에서 업데이트하려는 릴리스에 해당하는 `To apply this release:` 제목 아래에 있는 명령을 실행하세요.
   - 사용자 지정 설정이 있는 경우 다음 명령을 사용하여 매니페스트 파일을 다운로드하세요. *https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.18.0/config/master/aws-k8s-cni.yaml*을 업데이트하려는 GitHub 릴리스의 URL로 변경하세요.

```
curl -O https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/v1.18.0/config/
master/aws-k8s-cni.yaml
```

필요한 경우 만든 백업의 사용자 지정 설정으로 매니페스트를 수정한 다음 수정된 매니퍼스트를 클러스터에 적용합니다. 노드에서 이미지를 가져오는 프라이빗 Amazon EKS Amazon ECR 리포지토리에 액세스할 수 없는 경우(매니페스트에서 `image:`(으)로 시작하는 줄 참조) 이미지를 다운로드하여 자체 리포지토리에 복사한 다음 리포지토리에서 이미지를 가져오도록 매니페스트를 수정해야 합니다. 자세한 내용은 한 리포지토리에서 다른 리포지토리로 컨테이너 이미지 복사 단원을 참조하십시오.

```
kubectl apply -f aws-k8s-cni.yaml
```

5. 이제 클러스터에 새 버전이 설치되어 있는지 확인합니다.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
 | cut -d : -f 3
```

예제 출력은 다음과 같습니다.

```
v1.18.0
```

6. (선택 사항)`cni-metrics-helper` 클러스터에 설치합니다. 지표 헬퍼는 네트워크 인터페이스 및 IP 주소 정보를 수집하고 클러스터 수준에서 지표를 집계하고 이 지표를 Amazon CloudWatch에 게시하는 데 사용할 수 있는 도구입니다. 자세한 내용은 GitHub에서 metrics.proto를 참조하세요.

서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성

Amazon VPC CNI plugin for Kubernetes는 Amazon EKS 클러스터에 있는 Pod 네트워킹용 네트워킹 플러그 인입니다. 플러그인은 Kubernetes 노드에 VPC IP 주소를 할당하고 각 노드의 Pods에 대한 필수 네트워킹을 구성하는 역할을 합니다. 플러그 인의 기능:

- AWS Identity and Access Management(IAM) 권한이 필요합니다. 클러스터에서 IPv4 패밀리를 사용하는 경우 권한은 `AmazonEKS_CNI_Policy` AWS 관리형 정책에 지정됩니다. 클러스터에서 IPv6 패밀리를 사용하는 경우 권한이 생성하는 IAM 정책에 추가되어야 합니다. 정책을 Amazon EKS 노드 IAM 역할 또는 별도의 IAM 역할에 연결할 수 있습니다. 이 주제에 설명된 대로 별도의 역할에 할당하는 것이 좋습니다.

- 배포되면 aws-node라는 Kubernetes 서비스 계정을 생성하고 해당 계정을 사용하도록 구성됩니다. 이 서비스 계정은 aws-node라는 Kubernetes clusterrole에 바인딩되어 있습니다. 이 역할에는 필요한 Kubernetes 권한이 할당되어 있습니다.

> **Note**
>
> IMDS에 대한 액세스 권한을 차단하지 않는 한 Amazon VPC CNI plugin for Kubernetes의 Pods은 Amazon EKS 노드 IAM 역할에 할당된 권한에 액세스 권한이 있습니다. 자세한 내용은 워커 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한 섹션을 참조하세요.

필수 조건

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요.
- 클러스터에 대한 기존 AWS Identity and Access Management(IAM) OpenID Connect(OIDC) 제공업체입니다. 이미 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 부분을 참조하세요.

1단계 - Amazon VPC CNI plugin for Kubernetes IAM 역할 생성

IAM 역할 생성

1. 클러스터의 IP 패밀리를 확인합니다.

   ```
   aws eks describe-cluster --name my-cluster | grep ipFamily
   ```

   예제 출력은 다음과 같습니다.

   ```
   "ipFamily": "ipv4"
   ```

   출력에서 대신 ipv6가 반환될 수 있습니다.

2. IAM 역할을 생성합니다. eksctl 또는 kubectl 및 AWS CLI를 사용하여 IAM 역할을 생성할 수 있습니다.

eksctl

IAM 역할을 생성하고 클러스터의 IP 패밀리와 일치하는 명령을 사용하여 역할에 IAM 정책을 연결합니다. 명령은 IAM 역할을 생성하는 AWS CloudFormation 스택을 생성 및 배포하고, 여기에 지정한 정책을 연결하고, 생성된 IAM 역할의 ARN으로 기존 aws-node Kubernetes 서비스 계정에 주석을 추가합니다.

- IPv4

  *my-cluster*를 사용자의 고유한 값으로 교체합니다.

  ```
  eksctl create iamserviceaccount \
      --name aws-node \
      --namespace kube-system \
      --cluster my-cluster \
      --role-name AmazonEKSVPCCNIRole \
      --attach-policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
      --override-existing-serviceaccounts \
      --approve
  ```

- IPv6

  *my-cluster*를 사용자의 고유한 값으로 교체합니다. *111122223333*를 계정 ID로 바꾸고 *AmazonEKS_CNI_IPv6_Policy*를 IPv6 정책 이름으로 바꿉니다. IPv6 정책이 없는 경우 IPv6 패밀리를 사용하는 클러스터에 대한 IAM 정책 생성 섹션을 참조하여 생성합니다. 클러스터에서 IPv6를 사용하려면 여러 요구 사항을 충족해야 합니다. 자세한 내용은 클러스터, Pods 및 services용 IPv6 주소 단원을 참조하십시오.

  ```
  eksctl create iamserviceaccount \
      --name aws-node \
      --namespace kube-system \
      --cluster my-cluster \
      --role-name AmazonEKSVPCCNIRole \
      --attach-policy-arn
   arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
      --override-existing-serviceaccounts \
      --approve
  ```

kubectl and the AWS CLI

1. 클러스터의 OIDC 공급자 URL을 확인합니다.

```
aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text
```

예제 출력은 다음과 같습니다.

```
https://oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE
```

출력이 반환되지 않은 경우 해당 클러스터에 대한 IAM OIDC 공급자를 생성해야 합니다.

2. 다음 콘텐츠를 *vpc-cni-trust-policy.json*라는 파일에 복사합니다. *111122223333*을 계정 ID로 바꾸고, *EXAMPLED539D4633E53DE1B71EXAMPLE*을 이전 단계에서 반환된 값으로 바꿉니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:aws-node"
                }
            }
        }
    ]
}
```

3. 역할을 생성합니다. *AmazonEKSVPCCNIRole*을 선택한 모든 이름으로 바꿀 수 있습니다.

```
aws iam create-role \
  --role-name AmazonEKSVPCCNIRole \
  --assume-role-policy-document file://"vpc-cni-trust-policy.json"
```

4. 필요한 IAM 정책을 역할에 연결합니다. 클러스터의 IP 패밀리와 일치하는 명령을 실행합니다.

   - IPv4

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name AmazonEKSVPCCNIRole
```

   - IPv6

     *111122223333*를 계정 ID로 바꾸고 *AmazonEKS_CNI_IPv6_Policy*를 IPv6 정책 이름으로 바꿉니다. IPv6 정책이 없는 경우 IPv6 패밀리를 사용하는 클러스터에 대한 IAM 정책 생성 섹션을 참조하여 생성합니다. 클러스터에서 IPv6를 사용하려면 여러 요구 사항을 충족해야 합니다. 자세한 내용은 클러스터, Pods 및 services용 IPv6 주소 단원을 참조하십시오.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSVPCCNIRole
```

5. 다음 명령을 실행하여 이전에 생성한 IAM 역할의 ARN을 사용하여 aws-node 서비스 계정에 주석을 추가합니다. *example values*를 고유한 값으로 바꿉니다.

```
kubectl annotate serviceaccount \
    -n kube-system aws-node \
    eks.amazonaws.com/role-
arn=arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
```

3. (선택 사항) Kubernetes 서비스 계정에서 사용하는 AWS Security Token Service 엔드포인트 유형을 구성합니다. 자세한 내용은 서비스 계정의 AWS Security Token Service 엔드포인트 구성 단원을 참조하십시오.

2단계: Amazon VPC CNI plugin for KubernetesPods 재배포

1. 서비스 계정에 연결된 기존 Pods를 삭제하고 다시 생성하여 보안 인증 정보 환경 변수를 적용합니다. 주석은 현재 주석 없이 실행 중인 Pods에는 적용되지 않습니다. 다음 명령은 기존의 aws-node DaemonSet Pods를 삭제하고 서비스 계정 주석을 사용하여 배포합니다.

```
kubectl delete Pods -n kube-system -l k8s-app=aws-node
```

2. 모든 Pods가 다시 시작되었는지 확인합니다.

```
kubectl get pods -n kube-system -l k8s-app=aws-node
```

3. Pods 중 하나를 설명하고 AWS_WEB_IDENTITY_TOKEN_FILE 및 AWS_ROLE_ARN 환경 변수가 있는지 확인합니다. *cpjw7*을 이전 단계의 출력에서 반환된 Pods 중 하나의 이름으로 바꿉니다.

```
kubectl describe pod -n kube-system aws-node-cpjw7 | grep 'AWS_ROLE_ARN:\|
AWS_WEB_IDENTITY_TOKEN_FILE:'
```

예제 출력은 다음과 같습니다.

```
AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
      AWS_ROLE_ARN:
 arn:aws:iam::111122223333:role/AmazonEKSVPCCNIRole
      AWS_WEB_IDENTITY_TOKEN_FILE:           /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
```

Pod에 두 개의 컨테이너가 포함되어 있으므로 두 세트의 중복 결과가 반환됩니다. 두 컨테이너의 값이 같은 경우입니다.

Pod에서 AWS 리전 엔드포인트를 사용하고 있는 경우 이전 출력에서 다음 줄도 반환됩니다.

```
AWS_STS_REGIONAL_ENDPOINTS=regional
```

3단계: 노드 IAM 역할에서 CNI 정책 제거

Amazon EKS 노드 IAM 역할에 현재 AmazonEKS_CNI_Policy IAM(IPv4) 정책 또는 IPv6 정책이 연결되어 있고 별도의 IAM 역할을 생성한 경우, 정책을 대신 연결한 다음 aws-node Kubernetes 서비

스 계정에 할당한 후 클러스터의 IP 패밀리와 일치하는 AWS CLI 명령을 사용하여 노드 역할에서 정책을 제거하는 것이 좋습니다. *AmazonEKSNodeRole*을 노드 역할의 이름으로 바꿉니다.

- IPv4

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn
 arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
```

- IPv6

  *111122223333*를 계정 ID로 바꾸고 *AmazonEKS_CNI_IPv6_Policy*를 IPv6 정책 이름으로 바꿉니다.

```
aws iam detach-role-policy --role-name AmazonEKSNodeRole --policy-arn
 arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy
```

IPv6 패밀리를 사용하는 클러스터에 대한 IAM 정책 생성

IPv6 패밀리를 사용하는 클러스터를 생성했고 클러스터에 버전 1.10.1 이상의 Amazon VPC CNI plugin for Kubernetes 추가 기능이 구성되어 있으면 IAM 역할에 할당할 수 있는 IAM 정책을 생성해야 합니다. 생성할 때 IPv6 패밀리로 구성하지 않은 기존 클러스터가 있는 경우 IPv6을 사용하려면 새 클러스터를 생성해야 합니다. 클러스터에서 IPv6 사용에 대한 자세한 내용을 알아보려면 클러스터, Pods 및 services용 IPv6 주소 섹션을 참조하세요.

1. 다음 텍스트를 복사해 *vpc-cni-ipv6-policy*.json 파일에 저장합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeInstances",
                "ec2:DescribeTags",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeInstanceTypes"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
```

2. IAM 정책을 생성합니다.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-document
 file://vpc-cni-ipv6-policy.json
```

Pod 네트워킹 사용 사례 선택

Amazon VPC CNI plugin for Kubernetes는 Pods를 위한 네트워킹을 제공합니다. 다음 표는 함께 사용할 수 있는 네트워킹 사용 사례와 다양한 Amazon EKS 노드 유형에서 사용할 수 있는 기능 및 Amazon VPC CNI plugin for Kubernetes 설정을 이해하는 데 도움이 됩니다. 표의 모든 정보는 Linux `IPv4` 노드에만 적용됩니다.

Amazon EKS 노드 유형	Amazon EC2			Fargate
사용 사례	네트워크 인터페이스에 할당된 개별 IP 주소	네트워크 인터페이스에 할당된 IP 접두사	Pods의 보안 그룹	
포드에 대한 사용자 지정 네트워킹 - 노드의 서브넷과 다른 서브넷에서 IP 주소 할당	예	예	예	예(Fargate 프로파일을 통해 제어되는 서브넷)
Pods용 SNAT	예(기본값은 `false`)	예(기본값은 `false`)	예(`true`만 해당)	예(`true`만 해당)

Amazon EKS 노드 유형	Amazon EC2			Fargate
사용 사례	네트워크 인터페이스에 할당된 개별 IP 주소	네트워크 인터페이스에 할당된 IP 접두사	Pods의 보안 그룹	
기능				
보안 그룹 범위	노드	노드	포드 (POD_SECURITY_GROUP_ENFORCING_MODE =standard 및 AWS_VPC_K8S_CNI_EXTERNALSNAT =false로 설정한 경우 VPC 외부의 엔드포인트로 향하는 트래픽은 Pod's 보안 그룹이 아닌 노드의 보안 그룹 사용)	Pod
Amazon VPC 서브넷 유형	프라이빗 및 퍼블릭	프라이빗 및 퍼블릭	프라이빗 전용	프라이빗 전용
네트워크 정책 (VPC CNI)	호환됨	호환됨	호환됨 Amazon VPC CNI 플러그인 버전 1.14.0 이상만 해당	지원되지 않음

Amazon EKS 노드 유형	Amazon EC2			Fargate
사용 사례	네트워크 인터페이스에 할당된 개별 IP 주소	네트워크 인터페이스에 할당된 IP 접두사	Pods의 보안 그룹	
노드당 포드 밀도	중간	높음	낮음	1
Pod 시작 시간	상급	최상급	좋음	보통
Amazon VPC CNI 플러그인 설정(각 설정에 대한 자세한 내용은 GitHub에서 amazon-vpc-cni-k8s 참조)				
`WARM_ENI_TARGET`	예	해당 사항 없음	해당 사항 없음	해당 사항 없음
`WARM_IP_TARGET`	예	예	해당 사항 없음	해당 사항 없음
`MINIMUM_IP_TARGET`	예	예	해당 사항 없음	해당 사항 없음
`WARM_PREFIX_TARGET`	해당 사항 없음	예	해당 사항 없음	해당 사항 없음

Note

- 사용자 지정 네트워킹에서는 `IPv6`를 사용할 수 없습니다.
- `IPv6` 주소는 변환되지 않으므로 SNAT가 적용되지 않습니다.
- 연결된 보안 그룹이 있는 Pods에서 주고받는 트래픽 흐름에는 Calico 네트워크 정책이 적용되지 않으며, Amazon VPC 보안 그룹 적용으로만 제한됩니다.
- Calico 네트워크 정책 시행을 사용하는 경우 환경 변수 `ANNOTATE_POD_IP`를 `true`로 설정하여 Kubernetes 관련 알려진 문제가 발생하지 않도록 하는 것이 좋습니다. 이 기능을 사용하려면 포드에 대한 `patch` 권한을 `aws-node` ClusterRole에 추가해야 합니다. `aws-node` DaemonSet에 패치 권한을 추가하면 플러그인의 보안 범위가 증가합니다. 자세한 내용은 GitHub의 VPC CNI 리포지토리에서 ANNOTATE_POD_IP를 참조하세요.

- IP 접두사 및 IP 주소는 표준 Amazon EC2 탄력적 네트워크 인터페이스와 연결됩니다. 특정 보안 그룹이 필요한 포드에는 브랜치 네트워크 인터페이스의 기본 IP 주소가 할당됩니다. IP 주소 또는 IP 접두사의 IP 주소를 얻는 Pods와 동일한 노드에서 브랜치 네트워크 인터페이스를 얻는 Pods와 혼합할 수 있습니다.

Windows 노드

각 노드는 하나의 네트워크 인터페이스만 지원합니다. 보조 IPv4 주소와 IPv4 접두사를 사용할 수 있습니다. 기본적으로 노드에서 사용 가능한 IPv4 주소 수는 각 탄력적 네트워크 인터페이스에 할당할 수 있는 보조 IPv4 주소 수에서 1을 뺀 수와 같습니다. 그러나 IP 접두사를 활성화하여 노드에서 사용 가능한 IPv4 주소와 Pod 밀도를 늘릴 수 있습니다. 자세한 내용은 Amazon EC2 노드에 사용 가능한 IP 주소 증량 단원을 참조하십시오.

Calico 네트워크 정책은 Windows에서 지원됩니다. Windows에서는 사용자 지정 네트워킹이나 Pods용 보안 그룹을 사용할 수 없습니다.

클러스터, Pods 및 services용 **IPv6** 주소

기본적으로 Kubernetes는 Pods 및 services에 IPv4 주소를 할당합니다. Pods 및 services에 IPv4 주소를 할당하는 대신 IPv6 주소를 할당하도록 클러스터를 구성할 수 있습니다. Kubernetes가 1.23 버전 이상인 경우에도 Amazon EKS는 이중 스택 Pods 또는 services를 지원하지 않습니다. 결과적으로 Pods 및 services에 IPv4와 IPv6 주소를 모두 할당할 수 없습니다.

클러스터를 생성할 때 클러스터에 사용할 IP 패밀리를 선택합니다. 클러스터를 생성한 후에는 패밀리를 변경할 수 없습니다.

클러스터에 **IPv6** 패밀리 사용 시 고려 사항

- 새 클러스터를 생성하고 해당 클러스터에 IPv6 패밀리를 사용하도록 지정해야 합니다. 이전 버전에서 업데이트한 클러스터에 IPv6 패밀리를 사용 설정할 수 없습니다. 새 클러스터를 생성하는 방법에 대한 지침은 Amazon EKS 클러스터 생성 부분을 참조하세요.
- 클러스터에 배포하는 Amazon VPC CNI 추가 기능의 버전은 1.10.1 이상이어야 합니다. 이 버전 이상은 기본적으로 배포됩니다. 추가 기능을 배포하면 클러스터의 모든 노드 그룹에 있는 모든 노드를 제거하지 않으면 1.10.1 이전 버전에 Amazon VPC CNI 추가 기능을 다운로드할 수 없습니다.
- Windows Pods 및 services는 지원되지 않습니다.
- Amazon EC2 노드를 사용하는 경우 IP 접두사 위임 및 IPv6를 사용하여 Amazon VPC CNI 추가 기능을 구성해야 합니다. 클러스터를 생성할 때 IPv6 패밀리를 선택하면 1.10.1 버전의 추가 기능이

이 구성으로 기본 설정됩니다. 이는 자체 관리형 또는 Amazon EKS 추가 기능 모두에 해당됩니다. IP 접두사 위임에 대한 자세한 내용은 Amazon EC2 노드에 사용 가능한 IP 주소 증량 섹션을 참조하세요.

- 클러스터를 생성할 때 지정한 VPC 및 서브넷에는 지정한 VPC 및 서브넷에 할당된 IPv6 CIDR 블록이 있어야 합니다. 또한 할당된 IPv4 CIDR 블록도 있어야 합니다. IPv6만 사용하려는 경우에도 VPC가 작동하려면 IPv4 CIDR 블록이 필요하기 때문입니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 IPv6 CIDR 블록을 VPC와 연결을 참조하세요.
- 클러스터 및 노드를 생성할 때 IPv6 주소를 자동 할당하도록 구성된 서브넷을 지정해야 합니다. 그렇지 않으면 클러스터와 노드를 배포할 수 없습니다. 기본적으로 이 구성은 사용 중지되어 있습니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 서브넷의 IPv6 주소 지정 속성 수정을 참조하세요.
- 서브넷에 할당된 라우팅 테이블에는 IPv6 주소에 대한 경로가 있어야 합니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 IPv6로 마이그레이션을 참조하세요.
- 보안 그룹은 IPv6 주소를 허용해야 합니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 IPv6로 마이그레이션을 참조하세요.
- AWS Nitro 기반 Amazon EC2 또는 Fargate 노드에서만 IPv6를 사용할 수 있습니다.
- Amazon EC2 노드에서는 Pods의 보안 그룹과 함께 IPv6를 사용할 수 없습니다. 하지만 Fargate 노드에서는 사용할 수 있습니다. 개별 Pods에 대해 별도의 보안 그룹이 필요한 경우 Amazon EC2 노드와 함께 IPv4 패밀리를 계속 사용하거나 대신 Fargate 노드를 사용합니다.
- 이전에 IP 주소 소모를 완화하는 데 도움이 되도록 사용자 지정 네트워킹을 사용한 경우 대신 IPv6를 사용할 수 있습니다. IPv6에서는 사용자 지정 네트워킹을 사용할 수 없습니다. 네트워크 격리를 위해 사용자 정의 네트워킹을 사용하는 경우 클러스터에 사용자 정의 네트워킹 및 IPv4 패밀리를 계속 사용해야 할 수 있습니다.
- AWS Outposts에서는 IPv6를 사용할 수 없습니다.
- Pods 및 services만 IPv6 주소에 할당됩니다. IPv4 주소는 할당되지 않습니다. Pods는 인스턴스 자체의 NAT를 통해 IPv4 엔드포인트와 통신할 수 있으므로 DNS64 및 NAT64가 필요하지 않습니다. 트래픽에 퍼블릭 IP 주소가 필요한 경우 트래픽은 퍼블릭 IP로 변환된 소스 네트워크 주소입니다.
- Pod의 소스 IPv6 주소는 VPC 외부에서 통신할 때 노드의 IPv6 주소로 변환된 소스 네트워크 주소가 아닙니다. 인터넷 게이트웨이 또는 외부 전용 인터넷 게이트웨이를 사용하여 라우팅됩니다.
- 모든 노드에 IPv4 및 IPv6 주소가 할당됩니다.
- Amazon FSx for Lustre CSI 드라이버은(는) 지원되지 않습니다.
- 인스턴스 모드가 아닌 IP 모드에서 AWS 로드 밸런서 컨트롤러 버전 2.3.1 이상을 사용하여 애플리케이션 또는 네트워크 트래픽을 IPv6 Pods로 로드 밸런싱할 수 있습니다. 자세한 내용은 AWS Load Balancer Controller란 무엇인가요? 단원을 참조하십시오.

- IPv6 IAM 정책을 노드 IAM 또는 CNI IAM 역할에 연결해야 합니다. 두 가지 중 CNI IAM 역할에 연결하는 것이 좋습니다. 자세한 내용은 IPv6 패밀리를 사용하는 클러스터에 대한 IAM 정책 생성 및 1단계 - Amazon VPC CNI plugin for Kubernetes IAM 역할 생성 섹션을 참조하세요.
- 각 Fargate Pod는 배포된 서브넷에 대해 지정된 CIDR에서 IPv6 주소를 수신합니다. Fargate Pods를 실행하는 기본 하드웨어 장치는 하드웨어 장치가 배포된 서브넷에 할당된 CIDR에서 고유한 IPv4 및 IPv6 주소를 가져옵니다.
- IPv6 클러스터를 배포하기 전에 통합하는 애플리케이션, Amazon EKS 추가 기능 및 AWS 서비스를 철저히 평가하는 것이 좋습니다. 이는 IPv6에서 모든 것이 예상대로 작동하는지 확인하기 위한 것입니다.
- Amazon EC2 인스턴스 메타데이터 서비스 IPv6 엔드포인트 사용은 Amazon EKS에서 지원되지 않습니다.
- IPv6 패밀리를 사용하는 클러스터에서 자체 관리형 노드 그룹을 생성하는 경우 사용자 데이터에는 노드 시작 시 실행되는 bootstrap.sh 파일에 대한 다음 BootstrapArguments가 포함되어야 합니다. *your-cidr*를 클러스터 VPC의 IPv6 CIDR 범위로 교체합니다.

  ```
  --ip-family ipv6 --service-ipv6-cidr your-cidr
  ```

  클러스터의 IPv6 CIDR 범위를 모르는 경우 다음 명령을 사용하여 확인할 수 있습니다(AWS CLI 버전 2.4.9 이상 필요).

  ```
  aws eks describe-cluster --name my-cluster --query
   cluster.kubernetesNetworkConfig.serviceIpv6Cidr --output text
  ```

IPv6 클러스터 및 관리형 Amazon Linux 노드 배포

이 튜토리얼에서는 IPv6 Amazon VPC, IPv6 패밀리가 있는 Amazon EKS 클러스터 및 Amazon EC2 Amazon Linux 노드가 있는 관리형 노드 그룹을 배포합니다. IPv6 클러스터에는 Amazon EC2 Windows 노드를 배포할 수 없습니다. Fargate 노드를 클러스터에 배포할 수도 있지만 이 항목에서는 단순화를 위해 해당 지침을 제공하지 않습니다.

프로덕션 용도로 클러스터를 생성하기 전에 모든 설정을 숙지하고 요구 사항을 충족하는 설정으로 클러스터를 배포하는 것이 좋습니다. 자세한 내용은 Amazon EKS 클러스터 생성, 관리형 노드 그룹 및 이 주제에 대한 고려 사항을 참조하세요. 클러스터를 생성할 때 일부 설정만 사용 설정할 수 있습니다.

필수 조건

이 튜토리얼에서는 Amazon EKS 클러스터를 생성하고 관리할 때 필요한 다음 도구 및 리소스를 설치하고 구성해야 합니다.

- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 kubectl 설치 또는 업데이트 부분을 참조하세요.
- 사용하는 IAM 보안 주체에 Amazon EKS IAM 역할, 서비스 연결 역할, AWS CloudFormation, VPC 및 관련 리소스를 사용할 수 있는 권한이 있어야 합니다. 자세한 내용은 IAM 사용 설명서의 Amazon Elastic Kubernetes Service에 사용되는 작업, 리소스 및 조건 키와 서비스 연결 역할 사용을 참조하세요.

`eksctl` 또는 AWS CLI를 사용하여 리소스를 생성하는 절차가 제공됩니다. AWS Management Console을 사용하여 리소스를 배포할 수도 있지만 단순화를 위해 이 주제에서는 해당 지침을 제공하지 않습니다.

eksctl

전제 조건

`eksctl` 버전 `0.175.0` 이상이 컴퓨터에 설치되어 있어야 합니다. 설치하거나 업데이트하려면 `eksctl` 설명서에서 Installation을 참조하세요.

`eksctl`을 사용하여 `IPv6` 클러스터 배포

1. `ipv6-cluster.yaml` 파일을 생성합니다. 다음 명령을 디바이스에 복사합니다. 필요에 따라 명령을 다음과 같이 수정한 다음에 수정한 명령을 실행합니다.
   - *`my-cluster`*를 클러스터 이름으로 바꿉니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다.
   - *`region-code`*를 Amazon EKS에서 지원하는 AWS 리전으로 바꿉니다. AWS 리전 목록은 AWS 일반 참조 가이드의 Amazon EKS 엔드포인트 및 할당량을 참조하세요.
   - 클러스터의 버전이 포함된 `version`의 값입니다. 자세한 내용은 지원되는 Amazon EKS Kubernetes 버전을 참조하세요.
   - *`my-nodegroup`*을 노드 그룹의 이름으로 바꿉니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다.

- *t3.medium*을 AWS Nitro 시스템 인스턴스 유형으로 바꿀 수 있습니다.

```
cat >ipv6-cluster.yaml <<EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "X.XX"

kubernetesNetworkConfig:
  ipFamily: IPv6

addons:
  - name: vpc-cni
    version: latest
  - name: coredns
    version: latest
  - name: kube-proxy
    version: latest

iam:
  withOIDC: true

managedNodeGroups:
  - name: my-nodegroup
    instanceType: t3.medium
EOF
```

2. 클러스터를 생성합니다.

```
eksctl create cluster -f ipv6-cluster.yaml
```

클러스터 생성에 몇 분 정도 걸립니다. 다음 출력과 유사한 출력의 마지막 줄이 나타날 때까지 진행하지 마세요.

```
[...]
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

3. 기본 Pods에 IPv6 주소가 할당되었는지 확인합니다.

```
kubectl get pods -n kube-system -o wide
```

예제 출력은 다음과 같습니다.

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                             NODE
 NOMINATED NODE   READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::1               ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

4. 기본 서비스에 IPv6 주소가 할당되었는지 확인합니다.

```
kubectl get services -n kube-system -o wide
```

예제 출력은 다음과 같습니다.

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m
 k8s-app=kube-dns
```

5. (선택 사항) 샘플 애플리케이션을 배포하거나 AWS Load Balancer Controller 및 샘플 애플리케이션을 로드 밸런서 애플리케이션에 배포하거나 네트워크 트래픽을 IPv6 Pods에 배포합니다.

6. 이 튜토리얼용으로 생성한 클러스터 및 노드 사용을 마친 후 다음 명령을 사용하여 생성한 리소스를 정리해야 합니다.

```
eksctl delete cluster my-cluster
```

AWS CLI

전제 조건

AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요. AWS CloudShell을 사용하는 경우 AWS CloudShell에 설치된 기본 AWS CLI 버전이 이전 버전일 수 있으므로 AWS CLI의 버전 `2.12.3` 이상 또는 `1.27.160` 이상을 설치해야 할 수 있습니다.

> **Important**
>
> - 이 절차의 모든 단계를 동일한 사용자로 완료해야 합니다. 현재 사용자를 확인하려면 다음 명령을 실행합니다.
>
>   ```
>   aws sts get-caller-identity
>   ```
>
> - 이 절차의 모든 단계를 동일한 셸로 완료해야 합니다. 여러 단계에서는 이전 단계에서 설정한 변수를 사용합니다. 변수 값이 다른 셸에서 설정되면 변수를 사용하는 단계가 제대로 작동하지 않습니다. AWS CloudShell을 사용하여 다음 절차를 완료하는 경우 약 20~30분 동안 키보드나 포인터를 사용하여 상호 작용하지 않으면 셸 세션이 종료된다는 점을 기억하세요. 실행 중인 프로세스는 상호 작용으로 계산되지 않습니다.
> - 지침은 Bash 셸용으로 작성되었으며 다른 셸에서 조정해야 할 수도 있습니다.

AWS CLI를 사용하여 클러스터를 생성하려면

이 *`example values`*절차 단계의 모든 예제 값을 고유한 값으로 변경합니다.

1. 다음 명령을 실행하여 이후 단계에서 사용되는 일부 변수를 설정합니다. *region-code*를 리소스를 배포할 AWS 리전으로 변경합니다. 값은 Amazon EKS에서 지원하는 모든 AWS 리전일 수 있습니다. AWS 리전 목록은 AWS 일반 참조 가이드의 Amazon EKS 엔드포인트 및 할당량을 참조하세요. *my-cluster*를 클러스터 이름으로 바꿉니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다. *my-nodegroup*을 노드 그룹의 이름으로 바꿉니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다. *111122223333*을 계정 ID로 바꿉니다.

```
export region_code=region-code
export cluster_name=my-cluster
export nodegroup_name=my-nodegroup
export account_id=111122223333
```

2. Amazon EKS 및 IPv6 요구 사항을 충족하는 퍼블릭 및 프라이빗 서브넷이 있는 Amazon VPC를 생성합니다.

   a. 다음 명령을 실행하여 AWS CloudFormation 스택 이름에 대한 변수를 설정합니다. *my-eks-ipv6-vpc*을 선택한 모든 이름으로 바꿀 수 있습니다.

   ```
   export vpc_stack_name=my-eks-ipv6-vpc
   ```

   b. AWS CloudFormation 템플릿을 사용하여 IPv6 VPC를 생성합니다.

   ```
   aws cloudformation create-stack --region $region_code --stack-name
    $vpc_stack_name \
     --template-url https://s3.us-west-2.amazonaws.com/amazon-
   eks/cloudformation/2020-10-29/amazon-eks-ipv6-vpc-public-private-
   subnets.yaml
   ```

   스택을 만드는 데 몇 분 정도 걸립니다. 다음 명령을 실행합니다. 명령의 출력이 CREATE_COMPLETE가 될 때까지 다음 단계를 계속하지 마세요.

   ```
   aws cloudformation describe-stacks --region $region_code --stack-name
    $vpc_stack_name --query Stacks[].StackStatus --output text
   ```

   c. 생성된 퍼블릭 서브넷의 ID를 검색합니다.

   ```
   aws cloudformation describe-stacks --region $region_code --stack-name
    $vpc_stack_name \
   ```

```
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --
output text
```

예제 출력은 다음과 같습니다.

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE
```

d. 생성된 퍼블릭 서브넷에 대해 IPv6 주소 자동 할당 옵션을 사용 설정합니다.

```
aws ec2 modify-subnet-attribute --region $region_code --
subnet-id subnet-0a1a56c486EXAMPLE --assign-ipv6-address-on-
creation
aws ec2 modify-subnet-attribute --region $region_code --subnet-id
 subnet-099e6ca77aEXAMPLE --assign-ipv6-address-on-creation
```

e. 배포된 AWS CloudFormation 스택에서 템플릿으로 생성된 서브넷 및 보안 그룹의 이름을 검색하고 이후 단계에서 사용할 수 있도록 변수에 저장합니다.

```
security_groups=$(aws cloudformation describe-stacks --region $region_code
 --stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SecurityGroups`].OutputValue' --
output text)

public_subnets=$(aws cloudformation describe-stacks --region $region_code --
stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPublic`].OutputValue' --
output text)

private_subnets=$(aws cloudformation describe-stacks --region $region_code
 --stack-name $vpc_stack_name \
    --query='Stacks[].Outputs[?OutputKey==`SubnetsPrivate`].OutputValue' --
output text)

subnets=${public_subnets},${private_subnets}
```

3. 클러스터 IAM 역할을 생성하고 필요한 Amazon EKS IAM 관리형 정책을 여기에 연결합니다. Amazon EKS에서 관리하는 Kubernetes 클러스터는 사용자를 대신하여 다른 AWS 서비스를 호출하여 서비스와 함께 사용하는 리소스를 관리합니다.

a. 다음 명령을 실행해 eks-cluster-role-trust-policy.json 파일을 생성합니다.

```
cat >eks-cluster-role-trust-policy.json <<EOF
```

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

b. 다음 명령을 실행하여 역할 이름에 대한 변수를 설정합니다. *myAmazonEKSClusterRole*을 선택한 모든 이름으로 바꿀 수 있습니다.

```
export cluster_role_name=myAmazonEKSClusterRole
```

c. 역할을 생성합니다.

```
aws iam create-role --role-name $cluster_role_name --assume-role-policy-
document file://"eks-cluster-role-trust-policy.json"
```

d. IAM 역할의 ARN을 검색하고 이후 단계를 위해 변수에 저장합니다.

```
cluster_iam_role=$(aws iam get-role --role-name $cluster_role_name --
query="Role.Arn" --output text)
```

e. 필요한 Amazon EKS 관리형 IAM 정책을 역할에 연결합니다.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name $cluster_role_name
```

4. 클러스터를 생성합니다.

```
aws eks create-cluster --region $region_code --name $cluster_name --kubernetes-
version 1.XX \
   --role-arn $cluster_iam_role --resources-vpc-config subnetIds=
$subnets,securityGroupIds=$security_groups \
   --kubernetes-network-config ipFamily=ipv6
```

- > **Note**
  >
  > 요청의 가용 영역 중 하나에 Amazon EKS 클러스터를 생성하는 데 충분한 용량이 없다는 오류가 표시될 수 있습니다. 이 경우 오류 출력에는 새 클러스터를 지원할 수 있는 가용 영역이 포함됩니다. 사용자 계정의 지원 가용 영역에 있는 2개 이상의 서브넷을 사용하여 클러스터를 다시 생성합니다. 자세한 정보를 알아보려면 용량 부족을 참조하세요.

  클러스터를 생성하는 데 몇 분 정도 걸립니다. 다음 명령을 실행합니다. 명령의 출력이 `ACTIVE`가 될 때까지 다음 단계를 계속하지 마세요.

  ```
  aws eks describe-cluster --region $region_code --name $cluster_name --query
   cluster.status
  ```

5. 클러스터와 통신할 수 있도록 클러스터에 대한 `kubeconfig` 파일을 생성하거나 업데이트합니다.

   ```
   aws eks update-kubeconfig --region $region_code --name $cluster_name
   ```

   기본적으로 `config` 파일이 `~/.kube`에 생성되거나 새 클러스터의 구성이 `~/.kube`의 기존 `config` 파일에 추가됩니다.

6. 노드 IAM 역할을 생성합니다.

   a. 다음 명령을 실행해 `vpc-cni-ipv6-policy.json` 파일을 생성합니다.

   ```
   cat >vpc-cni-ipv6-policy <<EOF
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "ec2:AssignIpv6Addresses",
                   "ec2:DescribeInstances",
                   "ec2:DescribeTags",
                   "ec2:DescribeNetworkInterfaces",
                   "ec2:DescribeInstanceTypes"
               ],
   ```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ]
        }
    ]
}
EOF
```

b. IAM 정책을 생성합니다.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-
document file://vpc-cni-ipv6-policy.json
```

c. 다음 명령을 실행해 node-role-trust-relationship.json 파일을 생성합니다.

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

d. 다음 명령을 실행하여 역할 이름에 대한 변수를 설정합니다. *AmazonEKSNodeRole*을 선택한 모든 이름으로 바꿀 수 있습니다.

```
export node_role_name=AmazonEKSNodeRole
```

e. IAM 역할을 생성합니다.

```
aws iam create-role --role-name $node_role_name --assume-role-policy-
document file://"node-role-trust-relationship.json"
```

f. IAM 정책을 IAM 역할에 연결합니다.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy \
    --role-name $node_role_name
```

> **Important**
>
> 이 자습서에서는 단순화를 위해 이 IAM 역할에 정책이 연결되어 있습니다. 그러나 프로덕션 클러스터에서는 정책을 별도의 IAM 역할에 연결하는 것이 좋습니다. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성을 참조하세요.

g. 필요한 2개의 관리형 IAM 정책을 IAM 역할에 연결합니다.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly \
  --role-name $node_role_name
```

h. IAM 역할의 ARN을 검색하고 이후 단계를 위해 변수에 저장합니다.

```
node_iam_role=$(aws iam get-role --role-name $node_role_name --
query="Role.Arn" --output text)
```

7. 관리형 노드 그룹을 생성합니다.

a. 이전 단계에서 생성한 서브넷의 ID를 확인합니다.

```
echo $subnets
```

예제 출력은 다음과 같습니다.

```
subnet-0a1a56c486EXAMPLE,subnet-099e6ca77aEXAMPLE,subnet-
0377963d69EXAMPLE,subnet-0c05f819d5EXAMPLE
```

b. 노드 그룹을 생성합니다. *0a1a56c486EXAMPLE*, *099e6ca77aEXAMPLE*, *0377963d69EXAMPLE*, *0c05f819d5EXAMPLE*를 이전 단계의 결과에서 반환된 값으로 바꿉니다. 다음 명령의 이전 출력에서 서브넷 ID 사이의 쉼표를 제거해야 합니다. *t3.medium*을 AWS Nitro 시스템 인스턴스 유형으로 바꿀 수 있습니다.

```
aws eks create-nodegroup --region $region_code --cluster-name $cluster_name
 --nodegroup-name $nodegroup_name \
    --subnets subnet-0a1a56c486EXAMPLE subnet-099e6ca77aEXAMPLE
 subnet-0377963d69EXAMPLE subnet-0c05f819d5EXAMPLE \
    --instance-types t3.medium --node-role $node_iam_role
```

노드 그룹을 만드는 데 몇 분 정도 걸립니다. 다음 명령을 실행합니다. 반환되는 출력이 ACTIVE가 되면 다음 단계로 진행하세요.

```
aws eks describe-nodegroup --region $region_code --cluster-name
 $cluster_name --nodegroup-name $nodegroup_name \
    --query nodegroup.status --output text
```

8. IP 열에서 기본 Pods에 IPv6 주소가 할당되었는지 확인합니다.

```
kubectl get pods -n kube-system -o wide
```

예제 출력은 다음과 같습니다.

```
NAME                       READY   STATUS    RESTARTS   AGE     IP
                             NODE
 NOMINATED NODE   READINESS GATES
aws-node-rslts             1/1     Running   1          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
aws-node-t74jh             1/1     Running   0          5m32s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
coredns-85d5b4454c-cw7w2   1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::                ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

```
coredns-85d5b4454c-tx6n8   1/1     Running   0          56m
  2600:1f13:b66:8203:34e5::1              ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
kube-proxy-btpbk           1/1     Running   0          5m36s
  2600:1f13:b66:8200:11a5:ade0:c590:6ac8   ip-192-168-34-75.region-
code.compute.internal   <none>           <none>
kube-proxy-jjk2g           1/1     Running   0          5m33s
  2600:1f13:b66:8203:4516:2080:8ced:1ca9   ip-192-168-253-70.region-
code.compute.internal  <none>           <none>
```

9. IP 열에서 기본 서비스에 IPv6 주소가 할당되었는지 확인합니다.

```
kubectl get services -n kube-system -o wide
```

예제 출력은 다음과 같습니다.

```
NAME       TYPE        CLUSTER-IP          EXTERNAL-IP   PORT(S)         AGE
 SELECTOR
kube-dns   ClusterIP   fd30:3087:b6c2::a   <none>        53/UDP,53/TCP   57m
 k8s-app=kube-dns
```

10. (선택 사항) 샘플 애플리케이션을 배포하거나 AWS Load Balancer Controller 및 샘플 애플리케이션을 로드 밸런서 애플리케이션에 배포하거나 네트워크 트래픽을 IPv6 Pods에 배포합니다.

11. 이 튜토리얼로 생성한 클러스터 및 노드 사용을 마친 후 다음 명령을 사용하여 생성한 리소스를 정리해야 합니다. 리소스를 삭제하기 전에 이 튜토리얼 외부의 리소스를 사용하고 있지 않은지 확인합니다.

    a. 이전 단계를 완료한 것과 다른 셸에서 이 단계를 완료하는 경우 이전 단계에서 사용한 모든 변수의 값을 설정하고 *example values*을 이전 단계를 완료했을 때 지정한 값으로 바꿉니다. 이전 단계를 완료한 동일한 셸에서 이 단계를 완료하는 경우 다음 단계로 건너뛰니다.

    ```
    export region_code=region-code
    export vpc_stack_name=my-eks-ipv6-vpc
    export cluster_name=my-cluster
    export nodegroup_name=my-nodegroup
    export account_id=111122223333
    export node_role_name=AmazonEKSNodeRole
    export cluster_role_name=myAmazonEKSClusterRole
    ```

    b. 노드 그룹을 삭제합니다.

```
aws eks delete-nodegroup --region $region_code --cluster-name $cluster_name
 --nodegroup-name $nodegroup_name
```

삭제하는 데 몇 분 정도 걸립니다. 다음 명령을 실행합니다. 출력이 반환되는 경우 다음 단계로 진행하지 마세요.

```
aws eks list-nodegroups --region $region_code --cluster-name $cluster_name
 --query nodegroups --output text
```

c. 클러스터를 삭제합니다.

```
aws eks delete-cluster --region $region_code --name $cluster_name
```

클러스터를 삭제하는 데 몇 분 정도 걸립니다. 계속하기 전에 다음 명령으로 클러스터가 삭제되었는지 확인합니다.

```
aws eks describe-cluster --region $region_code --name $cluster_name
```

출력이 다음 출력과 유사할 때까지 다음 단계로 진행하지 마세요.

```
An error occurred (ResourceNotFoundException) when calling the
 DescribeCluster operation: No cluster found for name: my-cluster.
```

d. 생성한 IAM 리소스를 삭제합니다. 이전 단계에서 사용한 이름과 다른 이름을 선택한 경우 *AmazonEKS_CNI_IPv6_Policy*를 선택한 이름으로 바꿉니다.

```
aws iam detach-role-policy --role-name $cluster_role_name --policy-arn
 arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn
 arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name $node_role_name --policy-arn
 arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name $node_role_name --policy-arn
 arn:aws:iam::$account_id:policy/AmazonEKS_CNI_IPv6_Policy
aws iam delete-policy --policy-arn arn:aws:iam::
$account_id:policy/AmazonEKS_CNI_IPv6_Policy
aws iam delete-role --role-name $cluster_role_name
aws iam delete-role --role-name $node_role_name
```

e. VPC를 생성한 AWS CloudFormation 스택을 삭제합니다.

```
aws cloudformation delete-stack --region $region_code --stack-name
 $vpc_stack_name
```

Pods용 SNAT

IPv6 패밀리를 사용하여 클러스터를 배포한 경우 IPv6 주소는 네트워크로 변환되지 않기 때문에 이 주제의 정보는 클러스터에 적용되지 않습니다. 클러스터에서 IPv6 사용에 대한 자세한 내용을 알아보려면 클러스터, Pods 및 services용 IPv6 주소 섹션을 참조하세요.

기본적으로 클러스터의 각 Pod에는 Pod에 배포된 VPC와 연결된 Classless Inter-Domain Routing(CIDR) 블록의 프라이빗 IPv4 주소가 할당됩니다. 동일한 VPC의 Pods는 이러한 프라이빗 IP 주소를 엔드포인트로 사용하여 서로 통신합니다. Pod가 VPC에 연결된 CIDR 블록 내에 있지 않은 모든 IPv4 주소와 통신하는 경우 Amazon VPC CNI 플러그인(Linux 또는 Windows용 모두)은 Pod's IPv4 주소를 기본적으로 Pod가 실행 중인 노드의 기본 탄력적 네트워크 인터페이스의 기본 프라이빗 IPv4 주소로 변환합니다.*

> **Note**
>
> Windows 노드의 경우 고려해야 할 추가 세부 정보가 있습니다. 기본적으로 Windows용 VPC CNI 플러그인은 동일한 VPC 내의 대상에 대한 트래픽이 SNAT에서 제외되는 네트워킹 구성으로 정의됩니다. 이는 내부 VPC 통신에 SNAT가 비활성화되어 있고 Pod에 할당된 IP 주소를 VPC 내에서 라우팅할 수 있음을 의미합니다. 그러나 VPC 외부의 대상에 대한 트래픽은 소스 Pod IP SNAT가 인스턴스 ENI의 기본 IP 주소에 연결되어 있습니다. Windows에 대한 이 기본 구성은 포드가 호스트 인스턴스와 동일한 방식으로 VPC 외부의 네트워크에 액세스할 수 있도록 합니다.

이 동작으로 인해 다음이 발생합니다.

- 실행 중인 노드가 퍼블릭 또는 Elastic IP 주소를 할당받고 퍼블릭 서브넷에 있는 경우에만 Pods가 인터넷 리소스와 통신할 수 있습니다. 퍼블릭 서브넷은 인터넷 게이트웨이로 향하는 라우팅이 있는 라우팅 테이블과 연결된 서브넷입니다. 가능하면 프라이빗 서브넷에 노드를 배포하는 것이 좋습니다.
- 1.8.0 이전 플러그인 버전의 경우 VPC 피어링, 트랜짓 VPC 또는 AWS Direct Connect를 사용하여 클러스터 VPC에 연결된 네트워크 또는 VPC에 있는 리소스는 보조 탄력적 네트워크 인터페이스 뒤

에 있는 Pods와의 통신을 시작할 수 없습니다. 그럼에도 Pods는 이러한 리소스와 통신을 시작하고 응답을 수신할 수 있습니다.

사용자 환경에서 다음 설명 중 하나에 해당하는 경우 다음 명령을 사용하여 기본 구성을 변경하십시오.

- IPv4 주소를 사용하여 Pods와 통신을 시작해야 하는 VPC 피어링, 전송 VPC 또는 AWS Direct Connect를 사용하여 클러스터 VPC에 연결되는 네트워크 또는 VPC에 리소스가 있으며 플러그인 버전이 1.8.0보다 이전 버전입니다.
- Pods는 프라이빗 서브넷에 있으며 인터넷과 아웃바운드로 통신해야 합니다. 서브넷은 NAT 게이트웨이로 가는 경로를 가지고 있습니다.

```
kubectl set env daemonset -n kube-system aws-node AWS_VPC_K8S_CNI_EXTERNALSNAT=true
```

> **Note**
>
> AWS_VPC_K8S_CNI_EXTERNALSNAT 및 AWS_VPC_K8S_CNI_EXCLUDE_SNAT_CIDRS CNI 구성 변수는 Windows 노드에 적용할 수 없습니다. Windows에서는 SNAT를 비활성화할 수 없습니다. SNAT에서 IPv4 CIDR 목록을 제외하는 경우 Windows 부트스트랩 스크립트에서 ExcludedSnatCIDRs 파라미터를 지정하여 이를 정의할 수 있습니다. 이 파라미터 사용에 대한 자세한 내용은 부트스트랩 스크립트 구성 파라미터 섹션을 참조하세요.

* Pod's 사양에 hostNetwork=true(기본값은 false)가 포함된 경우 해당 IP 주소가 다른 주소로 변환되지 않습니다. 이 경우는 기본적으로 클러스터에서 실행되는 kube-proxy 및 Amazon VPC CNI plugin for Kubernetes Pods의 경우입니다. 이러한 Pods의 경우 IP 주소는 노드의 기본 IP 주소와 동일하므로 Pod's IP 주소는 변환되지 않습니다. Pod's hostNetwork 설정에 대한 자세한 내용을 알아보려면 Kubernetes API 참조에서 PodSpec v1 코어를 참조하세요.

클러스터에 Kubernetes 네트워크 정책 구성

기본적으로 클러스터의 Pods 또는 다른 네트워크의 Pods 및 리소스 사이에 IP 주소, 포트 또는 연결에 대해 Kubernetes에는 제한이 없습니다. Kubernetes 네트워크 정책을 사용하여 Pods에 대한 네트워크 트래픽을 제한할 수 있습니다. 자세한 내용은 Kubernetes 설명서의 네트워크 정책을 참조하세요.

클러스터에 버전 1.13 이하의 Amazon VPC CNI plugin for Kubernetes가 있는 경우 타사 솔루션을 구현하여 클러스터에 Kubernetes 네트워크 정책을 적용해야 합니다. 버전 1.14 이상의 플러그인으로 네트워크 정책을 구현할 수 있으므로 타사 솔루션을 사용할 필요가 없습니다. 이 주제에서는 타사 추가

기능을 사용하지 않고 클러스터에서 Kubernetes 네트워크 정책을 사용하도록 클러스터를 구성하는 방법을 알아봅니다.

Amazon VPC CNI plugin for Kubernetes의 네트워크 정책은 다음 구성에서 지원됩니다.

- Amazon EKS 클러스터 버전 `1.25` 이상.
- 클러스터의 Amazon VPC CNI plugin for Kubernetes 버전 1.14 이상.
- `IPv4` 또는 `IPv6` 주소에 대해 구성된 클러스터.
- Pods용 보안 그룹에 네트워크 정책을 사용할 수 있습니다. 네트워크 정책을 사용하면 클러스터 내 모든 통신을 제어할 수 있습니다. Pods용 보안 그룹을 통해 Pod 내의 애플리케이션에서 AWS 서비스에 대한 액세스를 제어할 수 있습니다.
- 사용자 지정 네트워킹 및 접두사 위임에 네트워크 정책을 사용할 수 있습니다.

고려 사항

- Amazon VPC CNI plugin for Kubernetes 사용을 통해 클러스터에 Amazon VPC CNI plugin for Kubernetes 네트워크 정책을 적용할 때 Amazon EC2 Linux 노드에만 정책을 적용할 수 있습니다. Fargate 또는 Windows 노드에는 정책을 적용할 수 없습니다.
- 클러스터에서 현재 타사 솔루션을 사용하여 Kubernetes 네트워크 정책을 관리하는 경우 Amazon VPC CNI plugin for Kubernetes에 동일한 정책을 사용할 수 있습니다. 하지만 동일한 정책을 관리하지 않도록 기존 솔루션을 제거해야 합니다.
- 여러 네트워크 정책을 동일한 Pod에 적용할 수 있습니다. 동일한 Pod를 선택한 둘 이상의 정책이 구성된 경우 모든 정책이 Pod에 적용됩니다.
- 네트워크 정책의 각 `ingress:` 또는 `egress:` 선택기에서 각 프로토콜의 고유한 포트 조합 최대 수는 24입니다.
- Kubernetes 서비스의 경우 서비스 포트는 컨테이너 포트와 동일해야 합니다. 이름이 지정된 포트를 사용하는 경우 서비스 사양에도 같은 이름을 사용합니다.
- Pod 시작 시 정책 적용

  Amazon VPC CNI plugin for Kubernetes는 포드 프로비저닝과 병행하여 포드에 대한 네트워크 정책을 구성합니다. 새 포드에 대해 모든 정책이 구성될 때까지 새 포드의 컨테이너는 기본 허용 정책으로 시작됩니다. 이를 표준 모드라고 합니다. 기본 허용 정책은 새 포드를 오가는 모든 수신 및 송신 트래픽이 허용됨을 의미합니다.

VPC CNI DaemonSet의 aws-node 컨테이너에서 환경 변수 NETWORK_POLICY_ENFORCING_MODE를 strict로 설정하여 기본 네트워크 정책을 변경할 수 있습니다.

```
env:
  - name: NETWORK_POLICY_ENFORCING_MODE
    value: "strict"
```

NETWORK_POLICY_ENFORCING_MODE 변수가 strict로 설정되면 VPC CNI를 사용하는 포드는 기본 거부 정책으로 시작하고, 정책이 구성됩니다. 이를 엄격 모드라고 합니다. 엄격 모드에서는 클러스터에서 포드가 액세스해야 하는 모든 엔드포인트에 대한 네트워크 정책이 있어야 합니다. 이 요구 사항은 CoreDNS 포드에 적용됩니다. 호스트 네트워킹을 사용하는 포드에는 기본 거부 정책이 구성되어 있지 않습니다.

- 네트워크 정책 기능에서 policyendpoints.networking.k8s.aws라는 PolicyEndpoint 사용자 지정 리소스 정의(CRD)가 생성되고 이를 사용해야 합니다.사용자 지정 참조의 PolicyEndpoint 개체는 Amazon EKS에서 관리합니다. 이러한 리소스를 수정하거나 삭제해서는 안 됩니다.
- 인스턴스 역할 IAM 보안 인증 정보를 사용하는 포드를 실행하거나 EC2 IMDS에 연결하는 경우 EC2 IMDS에 대한 액세스를 차단하는 네트워크 정책을 주의하여 확인하세요. EC2 IMDS에 대한 액세스를 허용하려면 네트워크 정책을 추가해야 할 수도 있습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스 메타데이터 및 사용자 데이터를 참조하세요.

  서비스 계정에 대한 IAM 역할을 사용하는 포드는 EC2 IMDS에 액세스하지 않습니다.
- Amazon VPC CNI plugin for Kubernetes는 각 포드의 추가 네트워크 인터페이스에 네트워크 정책을 적용하지 않고 각 포드의 기본 인터페이스(eth0)에만 적용합니다. 이는 다음 아키텍처에 영향을 미칩니다.
  - ENABLE_V4_EGRESS 변수가 true로 설정된 IPv6 포드. 이 변수를 사용하면 IPv4 송신 기능을 통해 IPv6 포드를 클러스터 외부의 항목과 같이 IPv4 엔드포인트에 연결할 수 있습니다. IPv4 송신 기능은 로컬 루프백 IPv4 주소를 사용하여 추가 네트워크 인터페이스를 생성하여 작동합니다.
  - Multus와 같은 체인 네트워크 플러그인을 사용하는 경우. 이러한 플러그인은 각 포드에 네트워크 인터페이스를 추가하므로 네트워크 정책은 체인 네트워크 플러그인에 적용되지 않습니다.
- 네트워크 정책 기능은 기본적으로 노드의 포트 8162를 지표에 사용합니다. 또한 이 기능은 상태 프로브에 포트 8163을 사용했습니다. 이러한 포트를 사용해야 하는 노드 또는 포드 내부에서 다른 애플리케이션을 실행하면 앱이 실행되지 않습니다. VPC CNI 버전 v1.14.1 이상에서는 다음 위치에서 이러한 포트 포트를 변경할 수 있습니다.

AWS Management Console

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 클러스터를 선택한 다음 Amazon VPC CNI 추가 기능을 구성할 클러스터의 이름을 선택합니다.
3. 추가 기능(Add-ons) 탭을 선택합니다.
4. 추가 기능 상자의 오른쪽 상단에 있는 상자를 선택한 다음에 Edit(편집)를 선택합니다.
5. Configure ***name of addon***(추가 기능 이름 구성) 페이지에서:
   a. 버전 드롭다운 목록에서 `v1.14.0-eksbuild.3` 이상 버전을 선택합니다.
   b. 선택적 구성 설정을 확장합니다.
   c. 구성 값에 JSON 키 `"enableNetworkPolicy":` 및 값 `"true"`를 입력합니다. 결과 텍스트는 유효한 JSON 객체여야 합니다. 이 키와 값이 텍스트 상자의 유일한 데이터인 경우 키와 값을 중괄호({})로 둘러쌉니다.

      다음 예시에서는 네트워크 정책 기능이 활성화되어 있고, 네트워크 정책 로그가 활성화되어 있고, 네트워크 정책 로그가 Amazon CloudWatch Logs로 전송되었으며, 지표와 상태 프로브가 기본 포트 번호로 설정되어 있습니다.

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true",
        "enableCloudWatchLogs": "true",
        "healthProbeBindAddr": "8163",
        "metricsBindAddr": "8162"
    }
}
```

Helm

`helm`을 통해 Amazon VPC CNI plugin for Kubernetes를 설치한 경우 구성을 업데이트하여 포트를 변경할 수 있습니다.

- 다음 명령을 실행하여 포트를 변경합니다. 키 `nodeAgent.metricsBindAddr` 또는 키 `nodeAgent.healthProbeBindAddr`의 값에 각각 포트 번호를 설정합니다.

```
helm upgrade --set nodeAgent.metricsBindAddr=8162 --set
 nodeAgent.healthProbeBindAddr=8163 aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. 편집기에서 aws-node DaemonSet를 엽니다.

```
kubectl edit daemonset -n kube-system aws-node
```

2. VPC CNI aws-node 데몬 세트 매니페스트의 aws-network-policy-agent 컨테이너에 있는 args:의 다음 명령 인수에서 포트 번호를 바꿉니다.

```
    - args:
            - --metrics-bind-addr=:8162
            - --health-probe-bind-addr=:8163
```

필수 조건

- 최소 클러스터 버전

  기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요. 클러스터는 Kubernetes 버전 1.25 이상이어야 합니다. 클러스터는 다음 표에 나열된 Kubernetes 버전 및 플랫폼 버전 중 하나를 실행해야 합니다. 나열된 것보다 이후의 모든 Kubernetes 및 플랫폼 버전도 지원됩니다. 다음 명령에서 *my-cluster*를 본인의 클러스터 이름으로 바꾼 다음 수정된 명령을 실행하여 현재 Kubernetes 버전을 확인할 수 있습니다.

```
aws eks describe-cluster
              --name my-cluster --query cluster.version --output
              text
```

Kubernetes 버전	플랫폼 버전
1.27.4	eks.5
1.26.7	eks.6

Kubernetes 버전	플랫폼 버전
1.25.12	eks.7

- 최소 VPC CNI 버전

  클러스터의 Amazon VPC CNI plugin for Kubernetes 버전 1.14 이상. 다음 명령을 사용하여 현재 버전을 확인할 수 있습니다.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni: |
 cut -d : -f 3
```

  버전이 1.14 이하인 경우 Amazon EKS 추가 기능 업데이트의 내용을 참조하여 버전을 1.14 이상으로 업그레이드합니다.

- 최소 Linux 커널 버전

  노드의 Linux 커널 버전이 5.10 이상이어야 합니다. uname -r을 사용하여 커널 버전을 확인할 수 있습니다. Amazon EKS 최적화 Amazon Linux, Amazon EKS 최적화 가속 Amazon Linux AMI, Bottlerocket AMI의 최신 버전을 사용하는 경우 이미 필수 커널 버전이 설치되어 있습니다.

  Amazon EKS 최적화 가속 Amazon Linux AMI 버전 v20231116 이상에는 커널 버전 5.10이 있습니다.

Kubernetes 네트워크 정책을 사용하도록 클러스터 구성

1. BPF 파일 시스템을 탑재합니다.

> **Note**
>
> 클러스터가 버전 1.27 이상인 경우 모든 Amazon EKS 최적화 Amazon Linux 및 Bottlerocket AMI 1.27 이상에 이미 이 기능이 있습니다.
> 다른 모든 클러스터 버전의 경우 Amazon EKS 최적화 Amazon Linux를 v20230703 이상으로 업그레이드하거나 Bottlerocket AMI를 버전 v1.0.2 이상으로 업그레이드한 경우 이 단계를 건너뛸 수 있습니다.

   a. 각 노드에 버클리 패킷 필터(BPF) 파일 시스템을 탑재합니다.

```
sudo mount -t bpf bpffs /sys/fs/bpf
```

b. 그런 다음 Amazon EC2 Auto Scaling 그룹용 시작 템플릿의 사용자 데이터에 동일한 명령을 추가합니다.

2. VPC CNI에서 네트워크 정책을 활성화합니다.

a. 클러스터에 설치된 추가 기능의 유형을 확인하세요. 클러스터를 생성하는 데 사용한 도구에 따라 현재 클러스터에 Amazon EKS 추가 기능이 유형이 설치되어 있지 않을 수 있습니다. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 addon.addonVersion --output text
```

버전 번호가 반환되는 경우 Amazon EKS 유형의 추가 기능이 클러스터에 설치되고 본 절차의 나머지 단계를 완료할 필요가 없습니다. 오류가 번호가 반환되는 경우 Amazon EKS 유형의 추가 기능이 클러스터에 설치되지 않습니다.

b. • Amazon EKS 추가 기능

AWS Management Console

a. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.

b. 왼쪽 탐색 창에서 클러스터를 선택한 다음 Amazon VPC CNI 추가 기능을 구성할 클러스터의 이름을 선택합니다.

c. 추가 기능(Add-ons) 탭을 선택합니다.

d. 추가 기능 상자의 오른쪽 상단에 있는 상자를 선택한 다음에 Edit(편집)를 선택합니다.

e. Configure ***name of addon***(추가 기능 이름 구성) 페이지에서:

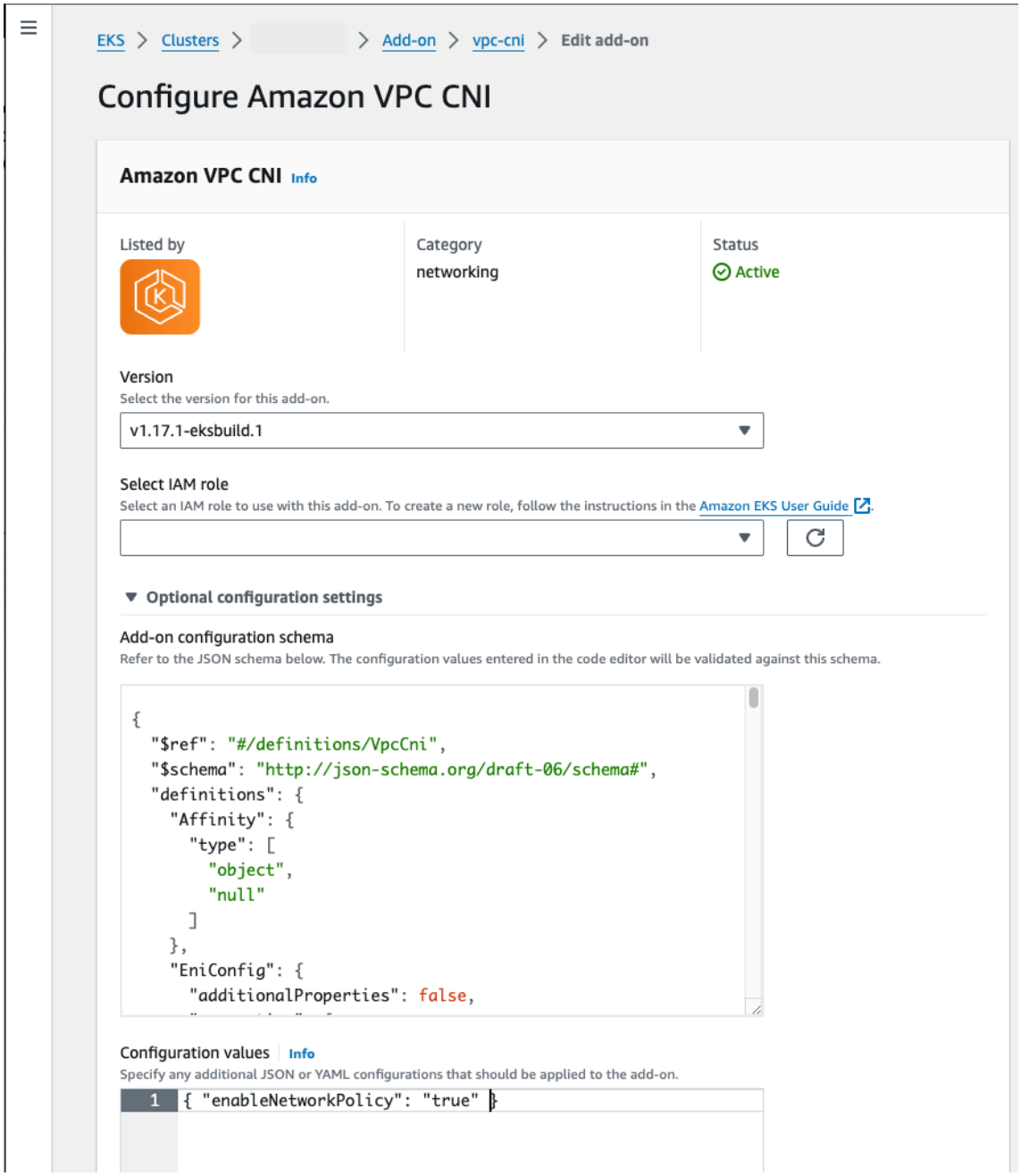
i. 버전 드롭다운 목록에서 `v1.14.0-eksbuild.3` 이상 버전을 선택합니다.

ii. 선택적 구성 설정을 확장합니다.

iii. 구성 값에 JSON 키 `"enableNetworkPolicy":` 및 값 `"true"`를 입력합니다. 결과 텍스트는 유효한 JSON 객체여야 합니다. 이 키와 값이 텍스트 상자의 유일한 데이터인 경우 키와 값을 중괄호({})로 둘러쌉니다. 다음 예시는 네트워크 정책이 활성화된 것을 보여줍니다.

```
{ "enableNetworkPolicy": "true" }
```

다음 스크린샷은 이 시나리오의 예를 보여줍니다.



AWS CLI

- 다음 AWS CLI 명령을 실행합니다. `my-cluster`를 본인의 클러스터 이름으로 바꾸고 IAM 역할 ARN을 사용 중인 역할로 바꿉니다.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni
 --addon-version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/
AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values
 '{"enableNetworkPolicy": "true"}'
```

- 자체 관리형 추가 기능

Helm

`helm`을 통해 Amazon VPC CNI plugin for Kubernetes를 설치한 경우 구성을 업데이트하여 네트워크 정책을 활성화할 수 있습니다.

- 다음 명령을 실행하여 네트워크 정책을 활성화합니다.

```
helm upgrade --set enableNetworkPolicy=true aws-vpc-cni --namespace
 kube-system eks/aws-vpc-cni
```

kubectl

a. 편집기에서 `amazon-vpc-cni` `ConfigMap`을 엽니다.

```
kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
```

b. 다음 줄을 `ConfigMap`의 `data`에 추가합니다.

```
enable-network-policy-controller: "true"
```

줄을 추가한 후에는 `ConfigMap`이 다음 예와 같을 것입니다.

```
apiVersion: v1
 kind: ConfigMap
```

```
metadata:
 name: amazon-vpc-cni
 namespace: kube-system
data:
 enable-network-policy-controller: "true"
```

c. 편집기에서 aws-node DaemonSet를 엽니다.

```
kubectl edit daemonset -n kube-system aws-node
```

d. VPC CNI aws-node 데몬 세트 매니페스트의 aws-network-policy-agent 컨테이너에 있는 args:의 명령 인수 --enable-network-policy=false에서 false를 true로 바꿉니다.

```
    - args:
       - --enable-network-policy=true
```

3. aws-node 포드가 클러스터에서 실행 중인지 확인합니다.

```
kubectl get pods -n kube-system | grep 'aws-node\|amazon'
```

예제 출력은 다음과 같습니다.

```
aws-node-gmqp7                                          2/2     Running   1 (24h
 ago)   24h
aws-node-prnsh                                          2/2     Running   1 (24h
 ago)   24h
```

네트워크 정책이 활성화된 경우 aws-node 포드에는 2개의 컨테이너입니다. 이전 버전에서 네트워크 정책이 비활성화된 경우 aws-node 포드에 하나의 컨테이너만 있습니다.

이제 클러스터에 Kubernetes 네트워크 정책을 배포할 수 있습니다. 자세한 내용은 Kubernetes 네트워크 정책 단원을 참조하십시오.

Stars 네트워크 정책 데모

이 데모는 Amazon EKS 클러스터에 프런트엔드, 백엔드 및 클라이언트 서비스를 생성합니다. 데모는 또한 각 서비스 간 이용 가능한 수신 및 발신 경로를 보여주는 관리 그래픽 사용자 인터페이스를 생성합니다. 프로덕션 워크로드를 실행하지 않는 클러스터에서 데모를 완료하는 것이 좋습니다.

네트워크 정책을 생성하기 전에 모든 서비스는 양방향으로 통신할 수 있습니다. 네트워크 정책을 적용하면 클라이언트가 프런트엔드 서비스와만 통신할 수 있고 백엔드는 프런트엔드의 트래픽만 수신하는 것을 볼 수 있습니다.

Stars 정책 데모 실행

1. 프런트엔드, 백엔드, 클라이언트 및 관리 사용자 인터페이스 서비스를 적용합니다.

```
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/namespace.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/management-ui.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/backend.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/frontend.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/client.yaml
```

2. 클러스터의 모든 Pods를 확인하세요.

```
kubectl get pods -A
```

예제 출력은 다음과 같습니다.

출력에서 다음 출력에 표시된 네임스페이스에 포드가 표시되어야 합니다. 포드의 ##과 READY 열의 포드 수는 다음 출력과 다릅니다. 이름이 비슷하고 STATUS 열에 Running이 있는 포드가 보일 때까지 계속하지 마십시오.

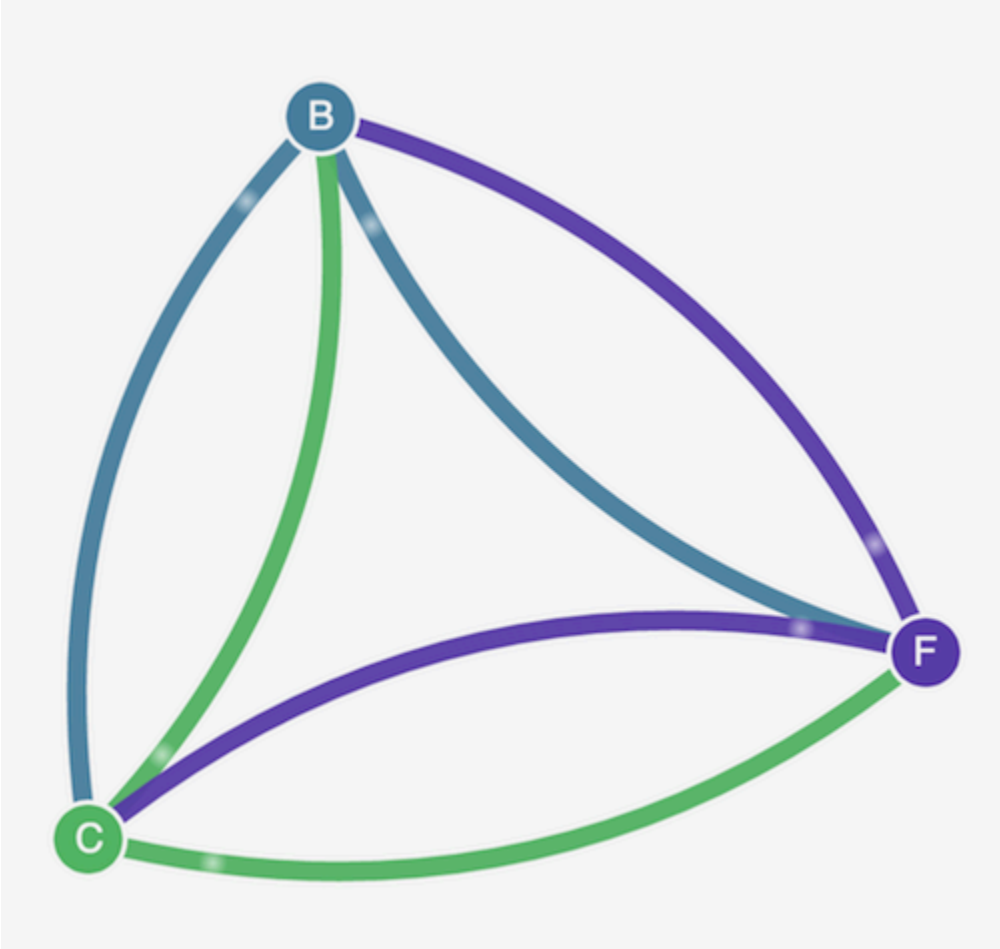
```
NAMESPACE         NAME                                       READY   STATUS
 RESTARTS   AGE
[...]
client            client-xlffc                               1/1     Running   0
       5m19s
[...]
management-ui     management-ui-qrb2g                        1/1     Running   0
       5m24s
stars             backend-sz87q                              1/1     Running   0
       5m23s
stars             frontend-cscnf                             1/1     Running   0
       5m21s
```

```
[...]
```

3. 관리 사용자 인터페이스에 연결하려면 클러스터에서 실행 중인 서비스의 EXTERNAL-IP에 연결합니다.

```
kubectl get service/management-ui -n management-ui
```

4. 브라우저를 열어 이전 단계의 위치로 이동합니다. 관리 사용자 인터페이스가 표시됩니다. C 노드는 클라이언트 서비스이고, F 노드는 프런트엔드 서비스이며, B 노드는 백엔드 서비스입니다. 각 노드는 기타 모든 노드에 대한 전체 통신 액세스를 보유합니다(진한 색상의 선으로 표시).



5. stars 및 client 네임스페이스 모두에 다음 네트워크 정책을 적용하여 서비스를 각각 격리시킵니다.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
```

```
  podSelector:
    matchLabels: {}
```

다음 명령을 사용하여 두 네임스페이스 모두에 정책을 적용할 수 있습니다.

```
kubectl apply -n stars -f https://eksworkshop.com/beginner/120_network-policies/
calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
kubectl apply -n client -f https://eksworkshop.com/beginner/120_network-policies/
calico/stars_policy_demo/apply_network_policies.files/default-deny.yaml
```

6. 브라우저를 새로 고칩니다. 관리 사용자 인터페이스가 더 이상 노드에 도달하지 않아 사용자 인터페이스에 표시되지 않는 것을 확인할 수 있습니다.
7. 다음의 다른 네트워크 정책을 적용하여 관리 사용자 인터페이스가 서비스에 액세스하도록 허용합니다. 이 정책을 적용하여 UI 허용:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

이 정책을 적용하여 클라이언트를 허용합니다.

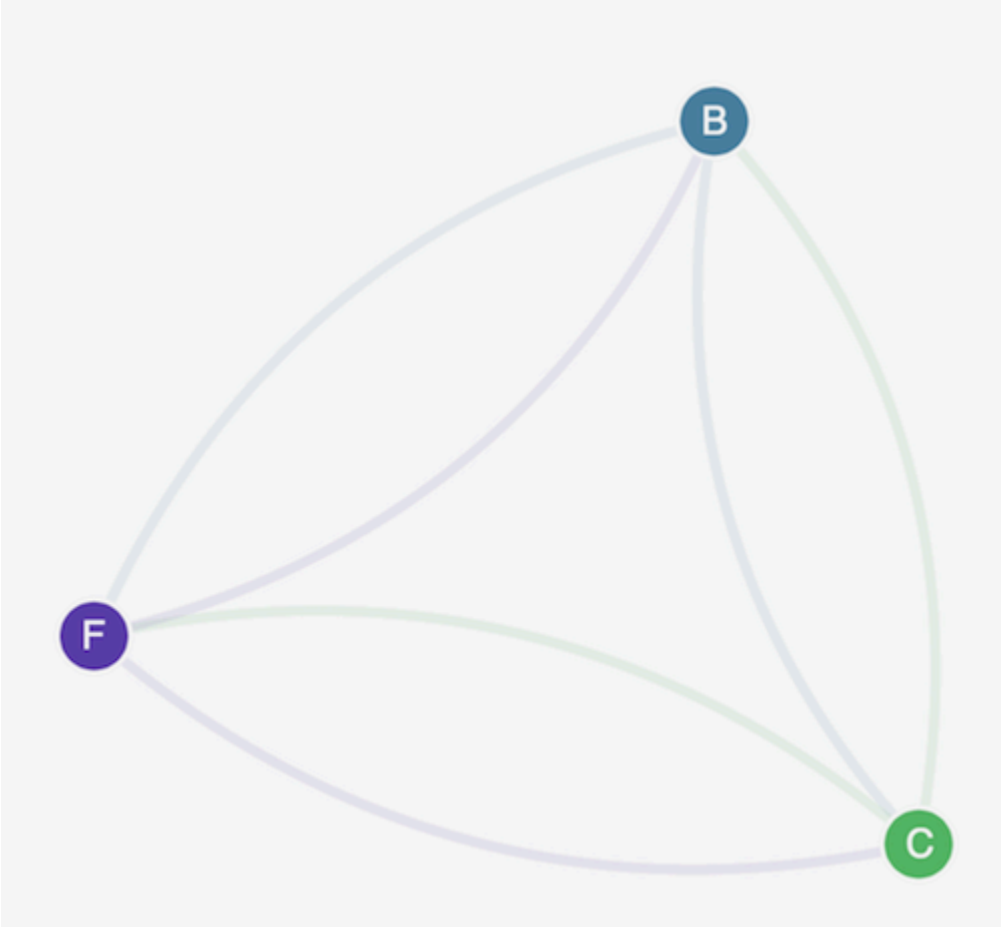
```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: client
  name: allow-ui
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
```

```
        - namespaceSelector:
            matchLabels:
              role: management-ui
```

다음 명령을 사용하여 두 정책을 모두 적용할 수 있습니다.

```
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/apply_network_policies.files/allow-ui.yaml
kubectl apply -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/apply_network_policies.files/allow-ui-client.yaml
```

8. 브라우저를 새로 고칩니다. 관리 사용자 인터페이스가 노드에 다시 도달할 수 있지만, 노드가 서로 통신할 수 없음을 확인할 수 있습니다.

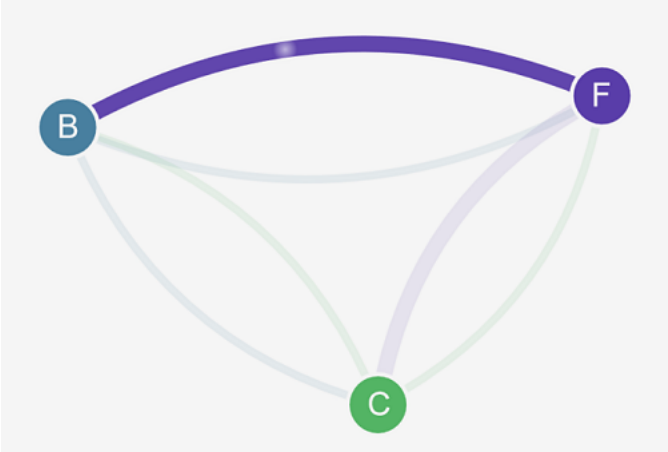


9. 다음 네트워크 정책을 적용하여 프런트엔드 서비스에서 백엔드 서비스로의 트래픽을 허용합니다.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
```

```
  namespace: stars
  name: backend-policy
spec:
  podSelector:
    matchLabels:
      role: backend
  ingress:
    - from:
        - podSelector:
            matchLabels:
              role: frontend
      ports:
        - protocol: TCP
          port: 6379
```

10. 브라우저를 새로 고칩니다. 프런트엔드가 백엔드와 통신할 수 있음을 알 수 있습니다.

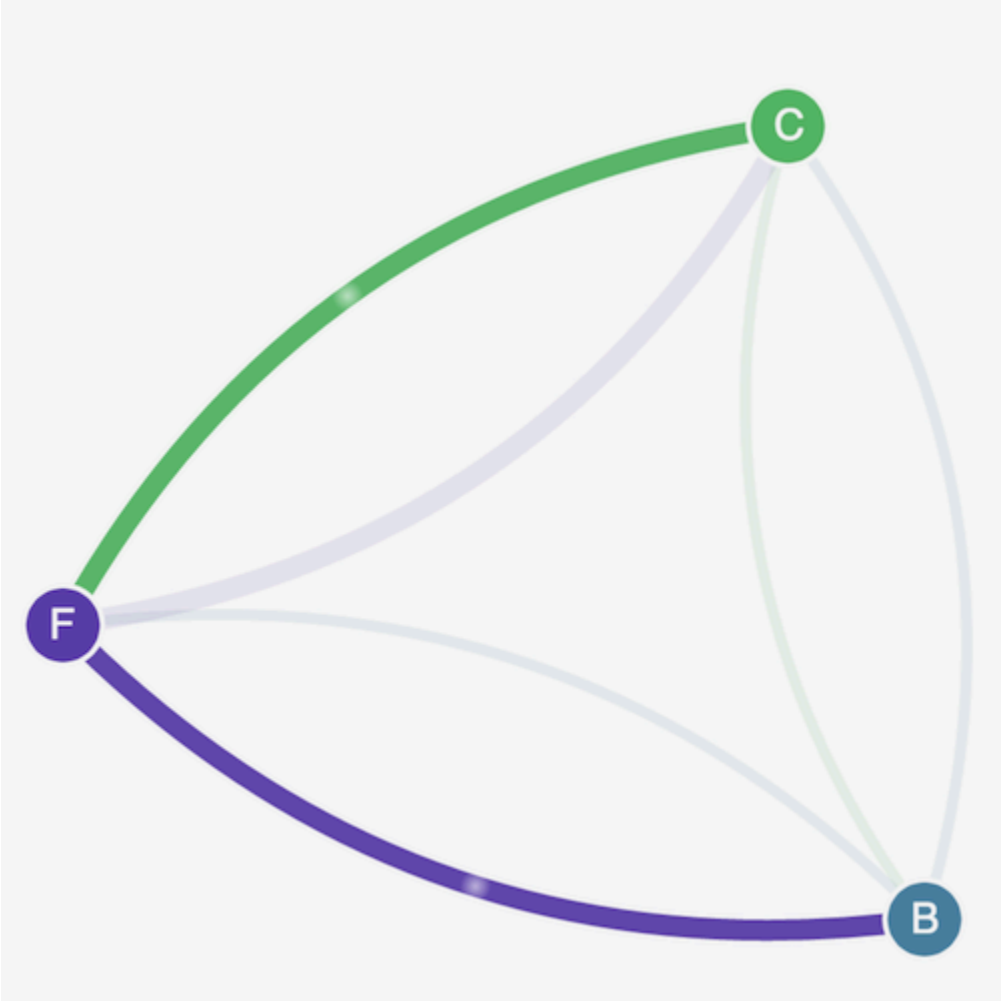


11. 다음 네트워크 정책을 적용하여 클라이언트에서 프런트엔드 서비스로의 트래픽을 허용합니다.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  namespace: stars
  name: frontend-policy
spec:
  podSelector:
    matchLabels:
      role: frontend
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
```

```
            role: client
    ports:
      - protocol: TCP
        port: 80
```

12. 브라우저를 새로 고칩니다. 클라이언트가 프런트엔드 서비스와 통신할 수 있음을 알게 됩니다. 프런트엔드 서비스는 여전히 백엔드 서비스와 통신할 수 있습니다.



13. (선택 사항) 데모를 마쳤으면 리소스를 삭제할 수 있습니다.

```
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/client.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/frontend.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/backend.yaml
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/management-ui.yaml
```

```
kubectl delete -f https://eksworkshop.com/beginner/120_network-policies/calico/
stars_policy_demo/create_resources.files/namespace.yaml
```

리소스를 삭제한 후에도 클러스터에서 예기치 않은 방식으로 네트워킹을 방해할 수 있는 네트워크 정책 엔드포인트가 노드에 있을 수 있습니다. 이러한 규칙을 제거하는 유일한 방법은 노드를 재부팅하거나 모든 노드를 종료하고 휴지통으로 이동하는 것입니다. 모든 노드를 종료하려면 Auto Scaling 그룹의 원하는 개수를 0으로 설정한 다음, 원하는 개수로 백업하거나 노드를 종료하면 됩니다.

네트워크 정책 문제 해결

네트워크 정책 로그의 내용을 읽고 eBPF SDK의 도구를 실행하여 네트워크 정책을 사용하는 네트워크 연결을 문제 해결 및 조사할 수 있습니다.

네트워크 정책 로그

네트워크 정책에 의해 연결이 허용 또는 거부되는지 여부는 흐름 로그에 기록됩니다. 각 노드의 네트워크 정책 로그에는 네트워크 정책이 있는 모든 포드의 흐름 로그가 포함됩니다. 네트워크 정책 로그는 `/var/log/aws-routed-eni/network-policy-agent.log`에 저장됩니다. 다음은 `network-policy-agent.log` 파일의 예입니다.

```
{"level":"info","timestamp":"2023-05-30T16:05:32.573Z","logger":"ebpf-
client","msg":"Flow Info: ","Src
IP":"192.168.87.155","Src Port":38971,"Dest IP":"64.6.160","Dest
Port":53,"Proto":"UDP","Verdict":"ACCEPT"}
```

네트워크 정책 로그는 기본적으로 비활성화되어 있습니다. 네트워크 정책 로그를 활성화하려면 다음 단계를 수행합니다.

> **Note**
>
> 네트워크 정책 로그에는 VPC CNI `aws-node` 데몬 세트 매니페스트의 `aws-network-policy-agent` 컨테이너용 vCPU 1개가 추가로 필요합니다.

Amazon EKS 추가 기능

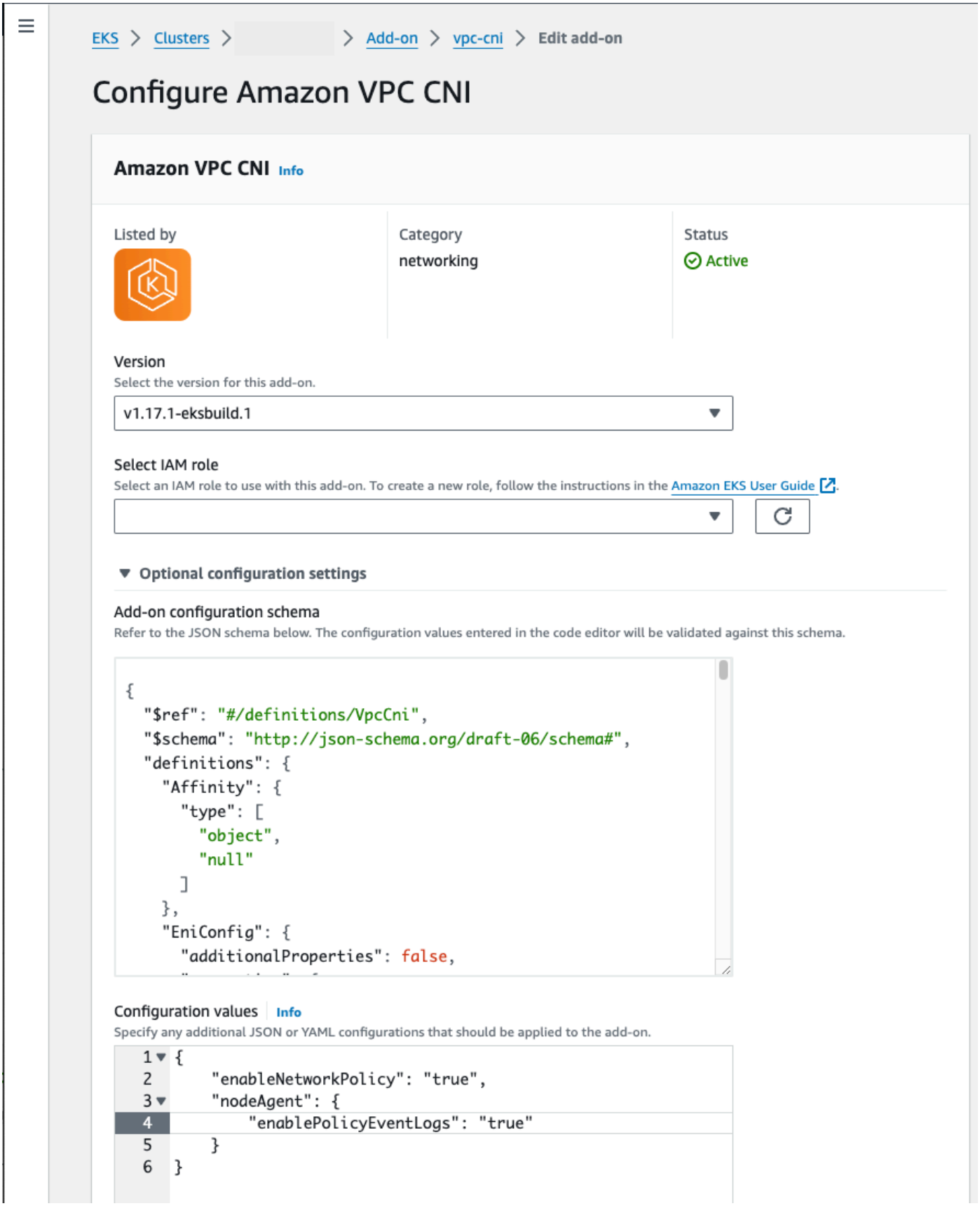
AWS Management Console

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.

2. 왼쪽 탐색 창에서 클러스터를 선택한 다음 Amazon VPC CNI 추가 기능을 구성할 클러스터의 이름을 선택합니다.
3. 추가 기능(Add-ons) 탭을 선택합니다.
4. 추가 기능 상자의 오른쪽 상단에 있는 상자를 선택한 다음에 Edit(편집)를 선택합니다.
5. Configure ***name of addon***(추가 기능 이름 구성) 페이지에서:
   a. 버전 드롭다운 목록에서 `v1.14.0-eksbuild.3` 이상 버전을 선택합니다.
   b. 선택적 구성 설정을 확장합니다.
   c. 최상위 JSON 키 `"nodeAgent":`를 입력하고 값은 키 `"enablePolicyEventLogs":`와 구성 값의 `"true"` 값이 포함된 객체입니다. 결과 텍스트는 유효한 JSON 객체여야 합니다. 다음 예시는 네트워크 정책과 네트워크 정책 로그가 활성화되어 있으며 CloudWatch Logs로 전송된 것을 보여줍니다.

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true"
    }
}
```

다음 스크린샷은 이 시나리오의 예를 보여줍니다.


EKS > Clusters > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true"
5 }
6 }

AWS CLI

- 다음 AWS CLI 명령을 실행합니다. `my-cluster`를 본인의 클러스터 이름으로 바꾸고 IAM 역할 ARN을 사용 중인 역할로 바꿉니다.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/
AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true"}}'
```

자체 관리형 추가 기능

Helm

`helm`을 통해 Amazon VPC CNI plugin for Kubernetes를 설치한 경우 구성을 업데이트하여 네트워크 정책을 쓸 수 있습니다.

- 다음 명령을 실행하여 네트워크 정책을 활성화합니다.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true aws-vpc-cni --namespace
 kube-system eks/aws-vpc-cni
```

kubectl

`kubectl`을 통해 Amazon VPC CNI plugin for Kubernetes를 설치한 경우 구성을 업데이트하여 네트워크 정책을 쓸 수 있습니다.

1. 편집기에서 `aws-node DaemonSet`를 엽니다.

```
kubectl edit daemonset -n kube-system aws-node
```

2. VPC CNI `aws-node` 데몬 세트 매니페스트의 `aws-network-policy-agent` 컨테이너에 있는 `args:`의 명령 인수 `--enable-policy-event-logs=false`에서 `false`를 `true`로 바꿉니다.

```
    - args:
```

```
        - --enable-policy-event-logs=true
```

네트워크 정책 로그를 Amazon CloudWatch Logs로 전송

Amazon CloudWatch Logs와 같은 서비스를 사용하여 네트워크 정책 로그를 모니터링할 수 있습니다. 다음 방법을 사용하여 네트워크 정책 로그를 CloudWatch Logs로 보낼 수 있습니다.

EKS 클러스터의 경우 정책 로그는 `/aws/eks/cluster-name/cluster/`에 있고 자체 관리형 K8S 클러스터의 경우 로그는 `/aws/k8s-cluster/cluster/`에 있습니다.

Amazon VPC CNI plugin for Kubernetes로 네트워크 정책 로그 전송

네트워크 정책을 활성화하면 두 번째 컨테이너가 노드 에이전트용 `aws-node` 포드에 추가됩니다. 이 노드 에이전트는 네트워크 정책 로그를 CloudWatch Logs로 보낼 수 있습니다.

> **Note**
>
> 네트워크 정책 로그는 노드 에이전트에 의해서만 전송됩니다. VPC CNI에서 생성한 다른 로그는 포함되지 않습니다.

필수 조건

- VPC CNI에 사용하는 IAM 역할에 다음 권한을 스탠자 또는 별도의 정책으로 추가합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

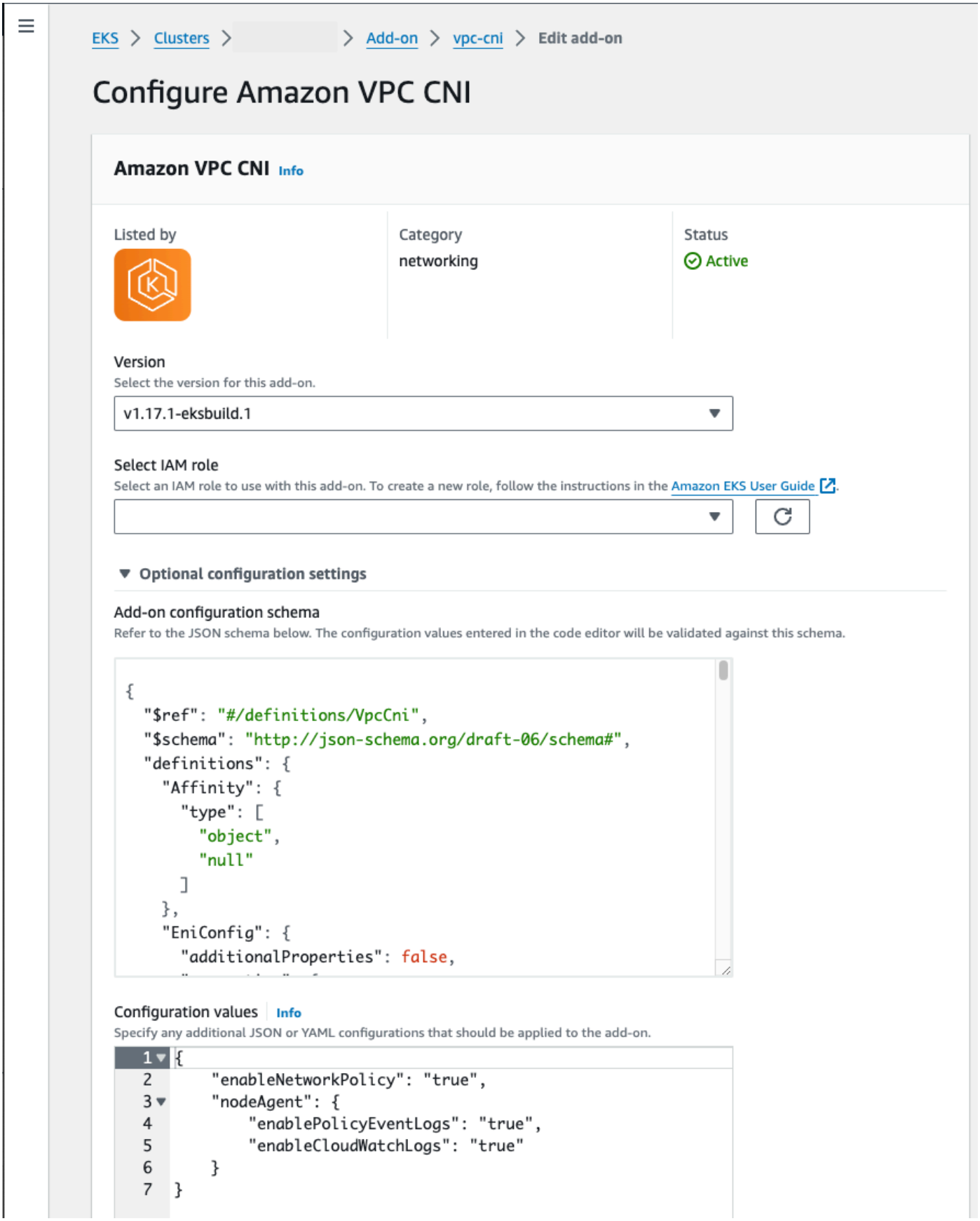
Amazon EKS 추가 기능

AWS Management Console

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 클러스터를 선택한 다음 Amazon VPC CNI 추가 기능을 구성할 클러스터의 이름을 선택합니다.
3. 추가 기능(Add-ons) 탭을 선택합니다.
4. 추가 기능 상자의 오른쪽 상단에 있는 상자를 선택한 다음에 Edit(편집)를 선택합니다.
5. Configure ***name of addon***(추가 기능 이름 구성) 페이지에서:
   a. 버전 드롭다운 목록에서 `v1.14.0-eksbuild.3` 이상 버전을 선택합니다.
   b. 선택적 구성 설정을 확장합니다.
   c. 최상위 JSON 키 `"nodeAgent":`를 입력하고 값은 키 `"enableCloudWatchLogs":`와 구성 값의 `"true"` 값이 포함된 객체입니다. 결과 텍스트는 유효한 JSON 객체여야 합니다. 다음 예시는 네트워크 정책과 네트워크 정책 로그가 활성화되어 있으며 로그가 CloudWatch Logs로 전송된 것을 보여줍니다.

```
{
    "enableNetworkPolicy": "true",
    "nodeAgent": {
        "enablePolicyEventLogs": "true",
        "enableCloudWatchLogs": "true",
    }
}
```

다음 스크린샷은 이 시나리오의 예를 보여줍니다.


EKS > Clusters > > Add-on > vpc-cni > Edit add-on
Configure Amazon VPC CNI
Amazon VPC CNI Info
Listed by
Category
networking
Status
Active
Version
Select the version for this add-on.
v1.17.1-eksbuild.1
Select IAM role
Select an IAM role to use with this add-on. To create a new role, follow the instructions in the Amazon EKS User Guide.
Optional configuration settings
Add-on configuration schema
Refer to the JSON schema below. The configuration values entered in the code editor will be validated against this schema.
{
"$ref": "#/definitions/VpcCni",
"$schema": "http://json-schema.org/draft-06/schema#",
"definitions": {
"Affinity": {
"type": [
"object",
"null"
]
},
"EniConfig": {
"additionalProperties": false,
Configuration values Info
Specify any additional JSON or YAML configurations that should be applied to the add-on.
1 {
2 "enableNetworkPolicy": "true",
3 "nodeAgent": {
4 "enablePolicyEventLogs": "true",
5 "enableCloudWatchLogs": "true"
6 }
7 }

AWS CLI

- 다음 AWS CLI 명령을 실행합니다. `my-cluster`를 본인의 클러스터 이름으로 바꾸고 IAM 역할 ARN을 사용 중인 역할로 바꿉니다.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version v1.14.0-eksbuild.3 \
    --service-account-role-arn arn:aws:iam::123456789012:role/
AmazonEKSVPCCNIRole \
    --resolve-conflicts PRESERVE --configuration-values '{"nodeAgent":
 {"enablePolicyEventLogs": "true", "enableCloudWatchLogs": "true"}}'
```

자체 관리형 추가 기능

Helm

`helm`을 통해 Amazon VPC CNI plugin for Kubernetes를 설치한 경우 구성을 업데이트하여 네트워크 정책 로그를 CloudWatch Logs로 전송할 수 있습니다.

- 다음 명령을 실행하여 네트워크 정책 로그를 활성화하고 CloudWatch Logs로 전송합니다.

```
helm upgrade --set nodeAgent.enablePolicyEventLogs=true --set
 nodeAgent.enableCloudWatchLogs=true aws-vpc-cni --namespace kube-system eks/
aws-vpc-cni
```

kubectl

1. 편집기에서 `aws-node DaemonSet`를 엽니다.

```
kubectl edit daemonset -n kube-system aws-node
```

2. VPC CNI `aws-node` 데몬 세트 매니페스트의 `aws-network-policy-agent` 컨테이너에 있는 `args:`에서 두 명령 인수 `--enable-policy-event-logs=false` 및 `--enable-cloudwatch-logs=false`에서 `false`를 `true`로 바꿉니다.

```
    - args:
       - --enable-policy-event-logs=true
       - --enable-cloudwatch-logs=true
```

Fluent Bit 데몬 세트로 네트워크 정책 로그 전송

데몬 세트에서 Fluent Bit를 사용하여 노드에서 로그를 전송하는 경우 네트워크 정책에서 네트워크 정책 로그를 포함하도록 구성을 추가할 수 있습니다. 다음 예제 구성을 사용할 수 있습니다.

```
    [INPUT]
        Name              tail
        Tag               eksnp.*
        Path              /var/log/aws-routed-eni/network-policy-agent*.log
        Parser            json
        DB                /var/log/aws-routed-eni/flb_npagent.db
        Mem_Buf_Limit     5MB
        Skip_Long_Lines   On
        Refresh_Interval  10
```

포함된 eBPF SDK

Amazon VPC CNI plugin for Kubernetes에서 노드에 eBPF SDK 도구 컬렉션을 설치합니다. eBPF SDK 도구를 사용하여 네트워크 정책 관련 문제를 식별할 수 있습니다. 예를 들어 다음 명령은 노드에서 실행 중인 프로그램을 나열합니다.

```
sudo /opt/cni/bin/aws-eks-na-cli ebpf progs
```

이 명령을 실행하려면 임의의 방법을 사용하여 노드에 연결할 수 있습니다.

Kubernetes 네트워크 정책

생성한 Kubernetes `NetworkPolicy` 개체에 Kubernetes 네트워크 정책을 구현하고 이를 클러스터에 배포하려면 `NetworkPolicy` 개체 범위는 네임스페이스로 지정됩니다. 정책을 구현하여 레이블 선택기, 네임스페이스 및 IP 주소 범위를 기반으로 Pods 사이의 트래픽을 허용 또는 거부합니다. `NetworkPolicy` 개체 생성에 관한 자세한 내용은 Kubernetes 설명서의 네트워크 정책을 참조하세요.

Kubernetes `NetworkPolicy` 개체의 적용은 Extended Berkeley Packet Filter (eBPF)를 사용하여 구현됩니다. `iptables` 기반 구현에 따라 지연 시간을 낮추고 CPU 활용률 감소 및 순차 조회 방지를 포함한 성능 특성을 제공합니다. 추가로 eBPF 프로브는 복잡한 커널 수준 문제를 디버깅하고 관찰성을 개선하는 데 도움이 되도록 컨텍스트가 풍부한 데이터에 대한 액세스를 제공합니다. Amazon EKS는 프로브를 활용하여 각 노드에 정책 결과를 기록하고 외부 로그 수집기로 데이터를 내보내 디버깅을 지원하는 eBPF 기반 익스포터를 지원합니다. 자세한 내용은 eBPF 설명서를 참조하십시오.

포드에 대한 사용자 지정 네트워킹

기본적으로 Amazon VPC CNI plugin for Kubernetes는 Amazon EC2 노드의 보조 Elastic 네트워크 인터페이스(네트워크 인터페이스)를 생성하는 경우 노드의 기본 네트워크 인터페이스와 동일한 서브넷에서 생성합니다. 또한 기본 네트워크 인터페이스에 연결된 보조 네트워크 인터페이스에 동일한 보안 그룹을 연결합니다. 다음 중 하나 이상의 이유로 플러그 인이 다른 서브넷에서 보조 네트워크 인터페이스를 생성하거나 다른 보안 그룹을 보조 네트워크 인터페이스에 연결하거나, 둘 다 하려고 할 수 있습니다.

- 기본 네트워크 인터페이스가 있는 서브넷에서 사용할 수 있는 IPv4 주소의 수는 제한되어 있습니다. 이렇게 하면 서브넷에서 생성할 수 있는 Pods의 수가 제한될 수 있습니다. 보조 네트워크 인터페이스에 다른 서브넷을 사용하면 Pods에 사용 가능한 IPv4 주소 수를 늘릴 수 있습니다.
- 보안상의 이유로 Pods는 노드의 기본 네트워크 인터페이스와 다른 서브넷 또는 보안 그룹을 사용해야 할 수 있습니다.
- 노드는 퍼블릭 서브넷에서 구성되며, Pods를 프라이빗 서브넷에 배치할 수 있습니다. 퍼블릭 서브넷과 연결된 라우팅 테이블에는 인터넷 게이트웨이로 가는 경로가 포함됩니다. 프라이빗 서브넷과 연결된 라우팅 테이블에는 인터넷 게이트웨이로 가는 경로가 포함되지 않습니다.

고려 사항

- 사용자 지정 네트워킹을 사용 설정하면 기본 네트워크 인터페이스에 할당된 IP 주소가 Pods에 할당되지 않습니다. 보조 네트워크 인터페이스의 IP 주소만 Pods에 할당됩니다.
- 클러스터에서 IPv6 패밀리를 사용하는 경우 사용자 지정 네트워킹을 사용할 수 없습니다.
- 사용자 지정 네트워킹을 사용하여 IPv4 주소 소모를 완화하려는 경우 대신 IPv6 패밀리를 사용하여 클러스터를 생성할 수 있습니다. 자세한 정보를 알아보려면 클러스터, Pods 및 services용 IPv6 주소을 참조하세요.
- 보조 네트워크 인터페이스에 지정된 서브넷에 배포된 Pods는 노드의 기본 네트워크 인터페이스와 다른 서브넷 및 보안 그룹을 사용할 수 있다고 해도 서브넷과 보안 그룹은 노드와 동일한 VPC에 있어야 합니다.

필수 조건

- Amazon VPC CNI plugin for Kubernetes가 보조 네트워크 인터페이스를 생성하고 Pods에 IP 주소를 할당하는 방법을 숙지합니다. 자세한 내용을 알아보려면 GitHub의 ENI 할당을 참조하세요.
- AWS Command Line Interface(AWS CLI) 버전 2.12.3 이상 또는 1.27.160 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **aws --version | cut**

`-d / -f2 | cut -d ' ' -f1`을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.

- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 `kubectl` 설치 또는 업데이트 부분을 참조하세요.
- Bash 셸에서 이 주제의 단계를 완료하는 것이 좋습니다. Bash 셸을 사용하지 않는 경우 줄 연속 문자 및 변수 설정 및 사용 방식과 같은 일부 스크립트 명령을 통해 셸이 조정되어야 합니다. 또한 쉘의 인용 및 이스케이프 규칙이 다를 수 있습니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI에서 문자열에 따옴표 사용을 참조하세요.

본 튜토리얼에서는 *`example values`*를 바꾸라고 언급된 경우를 제외하고는 이를 사용하는 것이 좋습니다. 프로덕션 클러스터의 단계를 완료하는 경우 모든 *`example value`*를 바꿀 수 있습니다. 동일한 터미널에서 모든 단계를 완료하는 것이 좋습니다. 이는 변수가 여러 단계에서 설정되고 사용되면서 서로 다른 터미널에 존재하지 않기 때문입니다.

이 주제의 명령은 AWS CLI 예제 사용에 나열된 규칙에 따라 형식이 지정되었습니다. 사용 중인AWS CLI 프로필에 정의된 기본 AWS 리전과 다른 AWS 리전에 있는 리소스에 대해 명령줄의 명령을 실행하는 경우에는 명령에 **`--region`** ***`region-code`***를 추가해야 합니다.

프로덕션 클러스터에 사용자 지정 네트워킹을 배포하려면 2단계: VPC 구성로 이동합니다.

1단계: 테스트 VPC 및 클러스터 생성

클러스터 생성

다음 절차를 활용하면 테스트 VPC 및 클러스터를 생성하고 해당 클러스터에 대해 사용자 지정 네트워킹을 구성할 수 있습니다. 프로덕션 클러스터에서 사용할 수 있는 관련 없는 기능들은 이 주제에서 다루지 않으므로 프로덕션 워크로드에 테스트 클러스터를 사용하지 않는 것이 좋습니다. 자세한 내용은 Amazon EKS 클러스터 생성 단원을 참조하십시오.

1. 나머지 단계에서 사용할 몇 가지 변수를 정의합니다.

```
export cluster_name=my-custom-networking-cluster
```

```
account_id=$(aws sts get-caller-identity --query Account --output text)
```

2. VPC를 생성합니다.

   1. Amazon EKS AWS CloudFormation 템플릿을 사용하여 VPC를 생성합니다.

      ```
      aws cloudformation create-stack --stack-name my-eks-custom-networking-vpc \
        --template-url https://s3.us-west-2.amazonaws.com/amazon-
      eks/cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
        --parameters ParameterKey=VpcBlock,ParameterValue=192.168.0.0/24 \
        ParameterKey=PrivateSubnet01Block,ParameterValue=192.168.0.64/27 \
        ParameterKey=PrivateSubnet02Block,ParameterValue=192.168.0.96/27 \
        ParameterKey=PublicSubnet01Block,ParameterValue=192.168.0.0/27 \
        ParameterKey=PublicSubnet02Block,ParameterValue=192.168.0.32/27
      ```

      AWS CloudFormation 스택을 만드는 데 몇 분 정도 걸립니다. 스택의 배포 상태를 확인하려면 다음 명령을 실행합니다.

      ```
      aws cloudformation describe-stacks --stack-name my-eks-custom-networking-vpc --
      query Stacks\[\].StackStatus  --output text
      ```

      명령의 출력이 `CREATE_COMPLETE`가 될 때까지 다음 단계를 계속하지 마세요.

   2. 템플릿에서 생성한 프라이빗 서브넷 ID 값으로 변수를 정의합니다.

      ```
      subnet_id_1=$(aws cloudformation describe-stack-resources --stack-name my-eks-
      custom-networking-vpc \
          --query "StackResources[?
      LogicalResourceId=='PrivateSubnet01'].PhysicalResourceId" --output text)
      subnet_id_2=$(aws cloudformation describe-stack-resources --stack-name my-eks-
      custom-networking-vpc \
          --query "StackResources[?
      LogicalResourceId=='PrivateSubnet02'].PhysicalResourceId" --output text)
      ```

   3. 이전 단계에서 검색된 서브넷의 가용 영역으로 변수를 정의합니다.

      ```
      az_1=$(aws ec2 describe-subnets --subnet-ids $subnet_id_1 --query
       'Subnets[*].AvailabilityZone' --output text)
      az_2=$(aws ec2 describe-subnets --subnet-ids $subnet_id_2 --query
       'Subnets[*].AvailabilityZone' --output text)
      ```

3. 클러스터 IAM 역할을 생성합니다.

a. 다음 명령을 실행하여 IAM 신뢰 정책 JSON 파일을 생성합니다.

```
cat >eks-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

b. Amazon EKS 클러스터 IAM 역할을 생성합니다. 필요한 경우 `eks-cluster-role-trust-policy.json` 앞에 이전 단계에서 파일을 작성한 컴퓨터의 경로를 붙입니다. 명령은 사용자가 이전 단계에서 생성한 신뢰 정책을 역할에 연결합니다. IAM 보안 주체를 생성하려면 역할을 생성하는 IAM 보안 주체에 `iam:CreateRole` 작업(권한)을 할당해야 합니다.

```
aws iam create-role --role-name myCustomNetworkingAmazonEKSClusterRole --
assume-role-policy-document file://"eks-cluster-role-trust-policy.json"
```

c. Amazon EKS 관리형 `AmazonEKSClusterPolicy`를 역할에 연결합니다. IAM 정책을 IAM 보안 주체에 연결하려면 정책을 연결하는 보안 주체에 `iam:AttachUserPolicy` 또는 `iam:AttachRolePolicy`의 IAM 작업(권한) 중 하나를 할당해야 합니다.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name myCustomNetworkingAmazonEKSClusterRole
```

4. Amazon EKS 클러스터를 생성하고 이와 통신하도록 디바이스를 구성합니다.

a. 클러스터를 생성합니다.

```
aws eks create-cluster --name my-custom-networking-cluster \
   --role-arn arn:aws:iam::$account_id:role/
myCustomNetworkingAmazonEKSClusterRole \
   --resources-vpc-config subnetIds=$subnet_id_1","$subnet_id_2
```

> **Note**
>
> 요청의 가용 영역 중 하나에 Amazon EKS 클러스터를 생성하는 데 충분한 용량이 없다는 오류가 표시될 수 있습니다. 이 경우 오류 출력에는 새 클러스터를 지원할 수 있는 가용 영역이 포함됩니다. 사용자 계정의 지원 가용 영역에 있는 2개 이상의 서브넷을 사용하여 클러스터를 다시 생성합니다. 자세한 정보를 알아보려면 용량 부족을 참조하세요.

b. 클러스터를 생성하는 데 몇 분 정도 걸립니다. 클러스터의 배포 상태를 확인하려면 다음 명령을 실행합니다.

```
aws eks describe-cluster --name my-custom-networking-cluster --query
 cluster.status
```

명령의 출력이 "ACTIVE"가 될 때까지 다음 단계를 계속하지 마세요.

c. 클러스터와 통신하도록 kubectl을 구성합니다.

```
aws eks update-kubeconfig --name my-custom-networking-cluster
```

2단계: VPC 구성

이 튜토리얼을 시작하려면 1단계: 테스트 VPC 및 클러스터 생성에서 생성된 VPC가 있어야 합니다. 프로덕션 클러스터의 경우 모든 *example values*을 사용자 값으로 바꿔 사용자 VPC 맞게 단계를 조정합니다.

1. 현재 설치된 Amazon VPC CNI plugin for Kubernetes이(가) 최신 버전인지 확인하세요. Amazon EKS 추가 기능 유형의 최신 버전을 확인하고 사용 중인 버전을 이 버전으로 업데이트하려면 추가 기능 업데이트 부분을 참조하세요. 자체 관리형 추가 기능 유형의 최신 버전을 확인하고 사용 중인 버전을 이 버전으로 업데이트하려면 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 부분을 참조하세요.
2. 클러스터 VPC ID를 검색하여 이후 단계에서 사용할 수 있도록 변수에 저장합니다. 프로덕션 클러스터의 경우 *my-custom-networking-cluster*를 클러스터의 이름으로 바꿉니다.

```
vpc_id=$(aws eks describe-cluster --name my-custom-networking-cluster --query
 "cluster.resourcesVpcConfig.vpcId" --output text)
```

3. 추가 Classless Inter-Domain Routing(CIDR) 블록을 클러스터의 VPC 연결합니다. CIDR 블록은 기존에 연결된 모든 CIDR 블록과 겹칠 수 없습니다.

   1. VPC에 연결된 현재 CIDR 블록을 봅니다.

      ```
      aws ec2 describe-vpcs --vpc-ids $vpc_id \
          --query 'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
       CidrBlockState.State}' --out table
      ```

      예제 출력은 다음과 같습니다.

      ```
      ---------------------------------
      |          DescribeVpcs          |
      +-----------------+--------------+
      |    CIDRBlock    |    State     |
      +-----------------+--------------+
      |  192.168.0.0/24 |  associated  |
      +-----------------+--------------+
      ```

   2. 추가 CIDR 블록을 VPC에 연결합니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 추가 IPv4 CIDR 블록을 VPC와 연결을 참조하세요.

      ```
      aws ec2 associate-vpc-cidr-block --vpc-id $vpc_id --cidr-block 192.168.1.0/24
      ```

   3. 새 블록이 연결되어 있는지 확인합니다.

      ```
      aws ec2 describe-vpcs --vpc-ids $vpc_id --query
       'Vpcs[*].CidrBlockAssociationSet[*].{CIDRBlock: CidrBlock, State:
       CidrBlockState.State}' --out table
      ```

      예제 출력은 다음과 같습니다.

      ```
      ---------------------------------
      |          DescribeVpcs          |
      +-----------------+--------------+
      |    CIDRBlock    |    State     |
      +-----------------+--------------+
      ```

```
|  192.168.0.0/24 |  associated  |
|  192.168.1.0/24 |  associated  |
+-----------------+--------------+
```

새 CIDR 블록의 State가 associated가 될 때까지 다음 단계를 진행하지 마세요.

4. 기존 서브넷이 있는 각 가용 영역에서 사용하려는 만큼 서브넷을 생성합니다. 이전 단계에서 VPC와 연결한 CIDR 블록 내에 있는 CIDR 블록을 지정합니다.

   1. 새 서브넷을 생성합니다. 서브넷은 기존 서브넷이 있는 것과 다른 VPC CIDR 블록에서 생성되어야 하지만 기존 서브넷과 동일한 가용 영역에서 생성되어야 합니다. 이 예제에서는 하나의 서브넷이 현재 프라이빗 서브넷이 존재하는 각 가용 영역의 새 CIDR 블록에서 생성됩니다. 생성된 서브넷의 ID는 이후 단계에서 사용할 수 있도록 변수에 저장됩니다. Name 값은 이전 단계에서 Amazon EKS VPC 템플릿을 사용하여 생성된 서브넷에 할당된 값과 일치합니다. 이름은 필요하지 않습니다. 다른 이름을 사용할 수 있습니다.

```
new_subnet_id_1=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone
 $az_1 --cidr-block 192.168.1.0/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-
custom-networking-vpc-PrivateSubnet01},{Key=kubernetes.io/role/internal-
elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
new_subnet_id_2=$(aws ec2 create-subnet --vpc-id $vpc_id --availability-zone
 $az_2 --cidr-block 192.168.1.32/27 \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=my-eks-
custom-networking-vpc-PrivateSubnet02},{Key=kubernetes.io/role/internal-
elb,Value=1}]' \
    --query Subnet.SubnetId --output text)
```

> **Important**
>
> 기본적으로 새로운 서브넷은 VPC의 기본 라우팅 테이블과 암시적으로 연결됩니다. 이 라우팅 테이블을 사용하면 VPC 배포된 모든 리소스 간에 통신할 수 있습니다. 그러나 VPC와 연결된 CIDR 블록 외부에 있는 IP 주소를 보유한 리소스와는 통신할 수 없습니다. 자체 라우팅 테이블을 서브넷에 연결하여 이 동작을 변경할 수 있습니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 서브넷 라우팅 테이블을 참조하세요.

   2. VPC에서 현재 서브넷을 확인합니다.

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" \
    --query 'Subnets[*].{SubnetId: SubnetId,AvailabilityZone:
 AvailabilityZone,CidrBlock: CidrBlock}' \
    --output table
```

예제 출력은 다음과 같습니다.

```
-------------------------------------------------------------------
|                          DescribeSubnets                        |
+------------------+--------------------+--------------------------+
| AvailabilityZone |     CidrBlock      |         SubnetId         |
+------------------+--------------------+--------------------------+
|  us-west-2d      |  192.168.0.0/27    |     subnet-example1      |
|  us-west-2a      |  192.168.0.32/27   |     subnet-example2      |
|  us-west-2a      |  192.168.0.64/27   |     subnet-example3      |
|  us-west-2d      |  192.168.0.96/27   |     subnet-example4      |
|  us-west-2a      |  192.168.1.0/27    |     subnet-example5      |
|  us-west-2d      |  192.168.1.32/27   |     subnet-example6      |
+------------------+--------------------+--------------------------+
```

생성한 `192.168.1.0` CIDR 블록의 서브넷이 `192.168.0.0` CIDR 블록의 서브넷과 동일한 가용 영역에 있는지 확인할 수 있습니다.

3단계: Kubernetes 리소스 구성

Kubernetes 리소스를 구성하려면 다음을 수행합니다.

1. `AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG` 환경 변수를 `aws-node` `DaemonSet`의 `true`로 설정합니다.

```
kubectl set env daemonset aws-node -n kube-system
 AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG=true
```

2. 클러스터 보안 그룹의 ID를 검색하여 이후 단계에서 사용할 수 있도록 변수에 저장합니다. Amazon EKS는 클러스터를 생성하는 경우 이 보안 그룹을 자동으로 생성합니다.

```
cluster_security_group_id=$(aws eks describe-cluster --name $cluster_name --query
 cluster.resourcesVpcConfig.clusterSecurityGroupId --output text)
```

3. Pods를 배포하려는 각 서브넷에 대해 ENIConfig 사용자 지정 리소스를 생성합니다.

   a. 각 네트워크 인터페이스 구성에 대해 고유한 파일을 생성합니다.

   다음 명령은 이전 단계에서 생성된 두 서브넷용 ENIConfig 파일을 별도로 생성합니다. name의 값은 고유해야 합니다. 이름은 서브넷이 있는 가용 영역과 동일합니다. 클러스터 보안 그룹은 ENIConfig에 할당됩니다.

```
cat >$az_1.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_1
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_1
EOF
```

```
cat >$az_2.yaml <<EOF
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
  name: $az_2
spec:
  securityGroups:
    - $cluster_security_group_id
  subnet: $new_subnet_id_2
EOF
```

   프로덕션 클러스터의 경우 이전 명령을 다음과 같이 변경할 수 있습니다.

   - *$cluster_security_group_id*를 각 ENIConfig에 대해 사용하려는 기존 보안 그룹 ID로 바꿉니다.
   - 가능하면 ENIConfig를 사용할 가용 영역과 동일하게 ENIConfigs의 이름을 지정하는 것이 좋습니다. 여러 이유로 가용 영역의 이름과 다른 ENIConfigs의 이름을 사용해야 할 수 있습니다. 예를 들어 동일한 가용 영역에 두 개 이상의 서브넷이 있고 사용자 지정 네트워킹에서 둘 다 사용하려는 경우 동일한 가용 영역에 대한 여러 ENIConfigs가 필요합니다. 각

ENIConfig는 고유한 이름이 필요하므로 가용 영역 이름을 사용하여 ENIConfigs에 두 개 이상의 이름을 지정할 수 없습니다.

ENIConfig 이름이 모두 가용 영역 이름과 동일하지 않은 경우 *$az_1* 및 *$az_2*를 이전 명령의 사용자 자신의 이름으로 바꾸고 이 자습서의 뒷부분에서 ENIConfig를 사용하여 노드에 주석을 답니다.

> **Note**
>
> 프로덕션 클러스터에서 사용할 유효한 보안 그룹을 지정하지 않고 다음을 사용하는 경우,
>
> - Amazon VPC CNI plugin for Kubernetes의 버전 1.8.0 이상을 사용하는 경우 노드의 기본 Elastic 네트워크 인터페이스와 연결된 보안 그룹이 사용됩니다.
> - 1.8.0 이전 버전의 Amazon VPC CNI plugin for Kubernetes를 사용하는 경우 VPC의 기본 보안 그룹이 보조 Elastic 네트워크 인터페이스에 할당됩니다.

> **Important**
>
> - AWS_VPC_K8S_CNI_EXTERNALSNAT=false는 Kubernetes용 Amazon VPC CNI 플러그인 구성의 기본 설정입니다. 기본 설정을 사용하는 경우 VPC와 연결된 CIDR 블록 중 하나에 있지 않은 IP 주소로 향하는 트래픽은 노드의 기본 네트워크 인터페이스의 보안 그룹 및 서브넷을 사용합니다. 보조 네트워크 인터페이스를 생성하는 데 사용된 ENIConfigs에서 정의된 서브넷 및 보안 그룹은 이 트래픽에는 사용되지 않습니다. 이 설정에 대한 자세한 내용을 알아보려면 Pods용 SNAT 섹션을 참조하세요.
> - Pods의 보안 그룹도 사용하는 경우 SecurityGroupPolicy에서 지정된 보안 그룹이 ENIConfigs에서 지정된 보안 그룹 대신 사용됩니다. 자세한 정보를 알아보려면 Pods의 보안 그룹을 참조하세요.

b. 다음 명령을 사용하여 생성된 각 사용자 지정 리소스 파일을 클러스터에 적용합니다.

```
kubectl apply -f $az_1.yaml
kubectl apply -f $az_2.yaml
```

4. ENIConfigs가 생성되었는지 확인합니다.

```
kubectl get ENIConfigs
```

예제 출력은 다음과 같습니다.

```
NAME         AGE
us-west-2a   117s
us-west-2d   105s
```

5. 프로덕션 클러스터에서 사용자 지정 네트워킹을 사용 설정하고 사용 중인 가용 영역 이외의 것으로 ENIConfigs의 이름을 지정한 경우 다음 단계로 이동하여 Amazon EC2 노드를 배포합니다.

   Kubernetes가 클러스터에 생성된 모든 새로운 Amazon EC2 노드에 가용 영역의 ENIConfig를 자동으로 적용할 수 있습니다.

   1. 이 튜토리얼의 테스트 클러스터의 경우 다음 단계로 이동합니다.

      프로덕션 클러스터의 경우 ENI_CONFIG_ANNOTATION_DEF 환경 변수에 대한 키 k8s.amazonaws.com/eniConfig가 포함된 annotation이 aws-node DaemonSet의 컨테이너 사양에 있는지 확인합니다.

      ```
      kubectl describe daemonset aws-node -n kube-system | grep
       ENI_CONFIG_ANNOTATION_DEF
      ```

      출력이 반환되는 경우 주석이 존재합니다. 출력이 반환되지 않으면 변수가 설정되지 않은 것입니다. 프로덕션 클러스터의 경우 이 설정이나 다음 단계의 설정 중 하나를 사용할 수 있습니다. 이 설정을 사용하는 경우 다음 단계의 설정을 재정의합니다. 이 튜토리얼에서는 다음 단계의 설정이 사용됩니다.

   2. aws-node DaemonSet를 업데이트하여 클러스터에 생성된 모든 새로운 Amazon EC2 노드에 가용 영역의 ENIConfig를 자동으로 적용합니다.

      ```
      kubectl set env daemonset aws-node -n kube-system
       ENI_CONFIG_LABEL_DEF=topology.kubernetes.io/zone
      ```

4단계: Amazon EC2 노드 배포

Amazon EC2 노드 배포

1. 노드 IAM 역할을 생성합니다.

   a. 다음 명령을 실행하여 IAM 신뢰 정책 JSON 파일을 생성합니다.

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

   b. 다음 명령을 실행하여 역할 이름에 대한 변수를 설정합니다. *myCustomNetworkingAmazonEKSNodeRole*을 선택한 모든 이름으로 바꿀 수 있습니다.

```
export node_role_name=myCustomNetworkingAmazonEKSNodeRole
```

   c. IAM 역할을 생성하고 이후 단계에서 사용할 수 있도록 반환된 Amazon 리소스 이름(ARN)을 변수에 저장합니다.

```
node_role_arn=$(aws iam create-role --role-name $node_role_name --assume-role-
policy-document file://"node-role-trust-relationship.json" \
    --query Role.Arn --output text)
```

   d. 필요한 3개의 IAM 관리형 정책을 IAM 역할에 연결합니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name $node_role_name
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
```

```
  --role-name $node_role_name
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
    --role-name $node_role_name
```

> **Important**
>
> 이 튜토리얼에서는 단순화를 위해 노드 IAM 역할에 AmazonEKS_CNI_Policy 정책이 연결되어 있습니다. 그러나 프로덕션 클러스터에서는 정책을 Amazon VPC CNI plugin for Kubernetes에서만 사용되는 별도의 IAM 역할에 연결하는 것이 좋습니다. 자세한 정보를 알아보려면 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성을 참조하세요.

2. 다음 노드 그룹 유형 중 하나를 생성합니다. 배포할 인스턴스 유형을 확인하려면 Amazon EC2 인스턴스 유형 선택 섹션을 참조하세요. 이 자습서에서는 관리형(Managed), 시작 템플릿을 사용하지 않거나 AMI ID가 지정되지 않은 시작 템플릿을 사용하는 경우(Without a launch template or with a launch template without an AMI ID specified)의 옵션을 완료합니다. 프로덕션 워크로드의 노드 그룹을 사용하려는 경우 노드 그룹을 배포하기 전에 관리형 및 자체 관리형 노드 그룹 옵션을 모두 숙지하는 것이 좋습니다.

   - 관리형(Managed) - 다음 옵션 중 하나를 사용하여 노드 그룹을 배포합니다.
     - 시작 템플릿을 사용하지 않거나 AMI ID가 지정되지 않은 시작 템플릿을 사용하는 경우(Without a launch template or with a launch template without an AMI ID specified) - 다음 명령을 실행합니다. 본 튜토리얼에서는 *example values*를 사용합니다. 프로덕션 노드 그룹의 경우 모든 *example values*을 사용자의 것으로 바꿉니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다.

       ```
       aws eks create-nodegroup --cluster-name $cluster_name --nodegroup-name my-
       nodegroup \
           --subnets $subnet_id_1 $subnet_id_2 --instance-types t3.medium --node-role
        $node_role_arn
       ```

     - AMI ID가 지정된 시작 템플릿을 사용하는 경우(With a launch template with a specified AMI ID)

       1. Amazon EKS 권장 노드의 최대 Pods 수를 확인합니다. 각 Amazon EC2 인스턴스 유형의 Amazon EKS 권장 최대 Pods 수의 지침을 따라 해당 주제의 3단계에 **--cni-custom-**

networking-enabled를 추가하세요. 다음 단계에서 사용할 수 있도록 출력을 적어 둡니다.

2. 시작 템플릿에서 Amazon EKS 최적화 AMI ID 또는 Amazon EKS 최적화 AMI에서 빌드한 사용자 지정 AMI를 지정한 다음 시작 템플릿을 사용하여 노드 그룹 배포를 지정하고 시작 템플릿에 다음 사용자 데이터를 입력합니다. 이 사용자 데이터는 인수를 `bootstrap.sh` 파일에 전달합니다. 부트스트랩 파일에 대한 자세한 내용은 GitHub에서 bootstrap.sh 섹션을 참조하세요. *20*을 이전 단계의 값(권장) 또는 고유한 값으로 바꿉니다.

```
/etc/eks/bootstrap.sh my-cluster --use-max-pods false --kubelet-extra-args
 '--max-pods=20'
```

Amazon EKS 최적화 AMI에서 빌드되지 않은 사용자 지정 AMI를 생성한 경우 직접 구성을 사용자 지정하여 생성해야 합니다.

- 자체 관리형(Self-managed)

1. Amazon EKS 권장 노드의 최대 Pods 수를 확인합니다. 각 Amazon EC2 인스턴스 유형의 Amazon EKS 권장 최대 Pods 수의 지침을 따라 해당 주제의 3단계에 **--cni-custom-networking-enabled**를 추가하세요. 다음 단계에서 사용할 수 있도록 출력을 적어 둡니다.
2. 자체 관리형 Amazon Linux 노드 시작하기의 지침에 따라 노드 그룹을 배포합니다. BootstrapArguments 파라미터에 다음 텍스트를 지정합니다. *20*을 이전 단계의 값(권장) 또는 고유한 값으로 바꿉니다.

```
--use-max-pods false --kubelet-extra-args '--max-pods=20'
```

> **Note**
>
> 프로덕션 클러스터의 노드가 훨씬 더 많은 수의 Pods를 지원하도록 하려는 경우 다시 각 Amazon EC2 인스턴스 유형의 Amazon EKS 권장 최대 Pods 수에서 스크립트를 실행합니다. 또한 **--cni-prefix-delegation-enabled** 옵션을 명령에 추가합니다. 예를 들어 `m5.large` 인스턴스 유형에 대해 *110*이 반환됩니다. 이 기능을 사용 설정하는 방법에 대한 지침은 Amazon EC2 노드에 사용 가능한 IP 주소 증량 부분을 참조하세요. 사용자 지정 네트워킹에서 이 기능을 사용할 수 있습니다.

노드 그룹을 생성하는 데 몇 분 정도 걸립니다. 다음 명령을 사용하여 관리형 노드 그룹의 생성 상태를 확인할 수 있습니다.

```
aws eks describe-nodegroup --cluster-name $cluster_name --nodegroup-name my-nodegroup --query nodegroup.status --output text
```

반환되는 출력이 ACTIVE가 될 때까지 다음 단계를 진행하지 마세요.

3. 튜토리얼에서는 이 단계를 건너뛸 수 있습니다.

프로덕션 클러스터의 경우 사용 중인 가용 영역과 동일하게 ENIConfigs의 이름을 지정하지 않은 경우 노드에서 사용해야 하는 ENIConfig 이름으로 해당 노드에 주석을 달아야 합니다. 각 가용 영역에 서브넷이 하나만 있고 가용 영역과 동일한 이름으로 ENIConfigs의 이름을 지정한 경우에는 이 단계가 필요하지 않습니다. 이는 Amazon VPC CNI plugin for Kubernetes가 이전 단계에서 그렇게 하도록 사용 설정한 경우 사용자 노드와 올바른 ENIConfig을 자동으로 연결하기 때문입니다.

a. 클러스터의 노드 목록을 가져옵니다.

```
kubectl get nodes
```

예제 출력은 다음과 같습니다.

```
NAME                                             STATUS   ROLES    AGE     VERSION
ip-192-168-0-126.us-west-2.compute.internal   Ready    <none>   8m49s
 v1.22.9-eks-810597c
ip-192-168-0-92.us-west-2.compute.internal    Ready    <none>   8m34s
 v1.22.9-eks-810597c
```

b. 각 노드가 있는 가용 영역을 확인합니다. 이전 단계에서 반환된 각 노드에 대한 다음 명령을 실행합니다.

```
aws ec2 describe-instances --filters Name=network-interface.private-dns-name,Values=ip-192-168-0-126.us-west-2.compute.internal \
--query 'Reservations[].Instances[].{AvailabilityZone:
 Placement.AvailabilityZone, SubnetId: SubnetId}'
```

예제 출력은 다음과 같습니다.

```
[
    {
        "AvailabilityZone": "us-west-2d",
        "SubnetId": "subnet-Example5"
    }
]
```

c. 각 노드에 서브넷 ID 및 가용 영역에 대해 생성한 ENIConfig로 주석을 답니다. 여러 노드에 동일한 ENIConfig로 주석을 달 수 있지만, 노드에는 하나의 ENIConfig로만 주석을 달 수 있습니다. *example values*을 사용자의 값으로 교체합니다.

```
kubectl annotate node ip-192-168-0-126.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName1
kubectl annotate node ip-192-168-0-92.us-west-2.compute.internal
 k8s.amazonaws.com/eniConfig=EniConfigName2
```

4. 사용자 지정 네트워킹 기능 사용으로 전환하기 전에 프로덕션 클러스터에 Pods를 실행 중인 노드가 있는 경우 다음 작업을 완료해야 합니다.

   a. 사용자 지정 네트워킹 기능을 사용하는 사용 가능한 노드가 있는지 확인합니다.

   b. 노드를 차단하고 드레이닝하여 Pods를 정상적으로 종료합니다. 자세한 내용을 알아보려면 Kubernetes 설명서의 안전한 노드 드레이닝을 참조하세요.

   c. 노드를 종료합니다. 노드가 기존 관리형 노드 그룹에 있는 경우 노드 그룹을 삭제할 수 있습니다. 다음 명령을 디바이스에 복사합니다. 필요에 따라 명령을 다음과 같이 수정한 다음에 수정한 명령을 실행합니다.

   - *my-cluster*를 클러스터 이름으로 변경합니다.
   - *my-nodegroup*을 노드 그룹의 이름으로 변경합니다.

```
aws eks delete-nodegroup --cluster-name my-cluster --nodegroup-name my-
nodegroup
```

   k8s.amazonaws.com/eniConfig 레이블에 등록된 새 노드만 사용자 지정 네트워킹 기능을 사용합니다.

5. Pods가 이전 단계에서 생성한 서브넷 중 하나에 연결된 CIDR 블록의 IP 주소를 할당하는지 확인합니다.

```
kubectl get pods -A -o wide
```

예제 출력은 다음과 같습니다.

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE     IP
           NODE                                            NOMINATED NODE   READINESS
 GATES
kube-system   aws-node-2rkn4             1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   aws-node-k96wp             1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
kube-system   coredns-657694c6f4-smcgr   1/1     Running   0          56m
 192.168.1.23    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   coredns-657694c6f4-stwv9   1/1     Running   0          56m
 192.168.1.28    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-jgshq           1/1     Running   0          7m19s
 192.168.0.92    ip-192-168-0-92.us-west-2.compute.internal    <none>
 <none>
kube-system   kube-proxy-wx9vk           1/1     Running   0          7m15s
 192.168.0.126   ip-192-168-0-126.us-west-2.compute.internal   <none>
 <none>
```

coredns Pods에 VPC 추가한 192.168.1.0 CIDR 블록의 IP 주소가 할당되는지 볼 수 있습니다. 사용자 지정 네트워킹을 사용하지 않는 경우 원래 CIDR 블록만 VPC와 연결되기 때문에 192.168.0.0 CIDR 블록의 주소가 할당되었을 것입니다.

Pod's spec이 hostNetwork=true를 포함하는 경우 노드의 기본 IP 주소가 할당됩니다. 추가한 서브넷의 주소는 할당되지 않습니다. 기본적으로 이 값은 false로 설정됩니다. 이 값은 클러스터에서 실행되는 kube-proxy 및 Amazon VPC CNI plugin for Kubernetes(aws-node) Pods에 대해 true로 설정됩니다. 따라서 kube-proxy 및 플러그인의 aws-node Pods에 이전 출력의 192.168.1.*x* 주소가 할당되지 않습니다. Pod's hostNetwork 설정에 대한 자세한 내용을 알아보려면 Kubernetes API 참조에서 PodSpec v1 코어를 참조하세요.

5단계: 튜토리얼 리소스 삭제

튜토리얼을 완료하면 생성한 리소스를 삭제하는 것이 좋습니다. 그런 다음 단계를 조정하여 프로덕션 클러스터에 대해 사용자 지정 네트워킹을 사용 설정할 수 있습니다.

튜토리얼 리소스 삭제

1. 생성한 노드 그룹이 테스트용일 경우 삭제하세요.

```
aws eks delete-nodegroup --cluster-name $cluster_name --nodegroup-name my-nodegroup
```

AWS CLI 출력에 클러스터가 삭제되었다고 표시된 이후에도 삭제 프로세스는 실제로 완료되지 않을 수 있습니다. 삭제 프로세스는 몇 분 정도 걸릴 수 있습니다. 다음 명령을 실행하여 완료되었는지 확인합니다.

```
aws eks describe-nodegroup --cluster-name $cluster_name --nodegroup-name my-
nodegroup --query nodegroup.status --output text
```

반환된 출력이 다음 출력과 비슷하게 될 때까지 진행하지 마세요.

```
An error occurred (ResourceNotFoundException) when calling the DescribeNodegroup
 operation: No node group found for name: my-nodegroup.
```

2. 생성한 노드 그룹이 테스트용일 경우 노드 IAM 역할을 삭제하세요.

   a. 역할에서 정책을 분리합니다.

```
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSNodeRole --
policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSNodeRole --
policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly
aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSNodeRole --
policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy
```

   b. 역할을 삭제합니다.

```
aws iam delete-role --role-name myCustomNetworkingAmazonEKSNodeRole
```

3. 클러스터를 삭제하세요.

```
aws eks delete-cluster --name $cluster_name
```

다음 명령을 사용하여 클러스터가 삭제되었는지 확인합니다.

```
aws eks describe-cluster --name $cluster_name --query cluster.status --output text
```

다음과 비슷한 출력이 반환되면 클러스터가 성공적으로 삭제됩니다.

```
An error occurred (ResourceNotFoundException) when calling the DescribeCluster
 operation: No cluster found for name: my-cluster.
```

4. 클러스터 IAM 역할을 삭제합니다.

   a. 역할에서 정책을 분리합니다.

   ```
   aws iam detach-role-policy --role-name myCustomNetworkingAmazonEKSClusterRole
    --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy
   ```

   b. 역할을 삭제합니다.

   ```
   aws iam delete-role --role-name myCustomNetworkingAmazonEKSClusterRole
   ```

5. 이전 단계에서 생성한 서브넷을 삭제합니다.

```
aws ec2 delete-subnet --subnet-id $new_subnet_id_1
aws ec2 delete-subnet --subnet-id $new_subnet_id_2
```

6. 생성한 VPC를 삭제합니다.

```
aws cloudformation delete-stack --stack-name my-eks-custom-networking-vpc
```

Amazon EC2 노드에 사용 가능한 IP 주소 증량

각 Amazon EC2 인스턴스는 최대 탄력적 네트워크 인터페이스 수와 각 네트워크 인터페이스에 할당할 수 있는 최대 IP 주소 수를 지원합니다. 각 노드에는 네트워크 인터페이스마다 IP 주소가 하나씩 필요합니다. 사용 가능한 다른 모든 IP 주소를 Pods에 할당할 수 있습니다. 각 Pod에는 자체 IP 주소가 필요합니다. 따라서 사용 가능한 컴퓨팅 및 메모리 리소스가 있지만 Pods에 할당할 IP 주소가 부족하여 추가 Pods를 수용할 수 없는 노드가 있을 수 있습니다.

이 주제에서는 노드에 개별 보조 IP 주소를 할당하는 대신 IP 접두사를 할당하여 노드가 Pods에 할당할 수 있는 IP 주소 수를 크게 늘리는 방법을 알아봅니다. 각 접두사에는 여러 IP 주소가 포함됩니다. IP 접두사 할당을 위해 클러스터를 구성하지 않는 경우 클러스터는 Pod 연결에 필요한 네트워크 인터페이스와 IP 주소를 구성하기 위해 더 많은 Amazon EC2 애플리케이션 프로그래밍 인터페이스(API)를 호출해야 합니다. 클러스터가 더 큰 규모로 성장함에 따라 이러한 API 호출의 빈도가 증가하여 Pod 및 인스턴스 시작 시간이 더 길어질 수 있습니다. 이로 인해 규모가 크고 까다로운 워크로드의 수요를 충족하기 위해 확장이 지연되고 확장 요구 사항을 충족하기 위해 추가 클러스터와 VPC를 프로비저닝해야 하므로 비용 및 관리 오버헤드가 추가됩니다. 자세한 내용은 GitHub의 Kubernetes Scalability thresholds를 참조하세요.

고려 사항

- 각 Amazon EC2 인스턴스 유형은 최대 Pods 수를 지원합니다. 관리형 노드 그룹이 여러 인스턴스 유형으로 구성된 경우, 클러스터의 인스턴스에 대한 최대 Pods 수 가운데 최소 수가 클러스터의 모든 노드에 적용됩니다.
- 기본적으로 노드에서 실행할 수 있는 최대 Pods 수는 110개이지만 이 수를 변경할 수 있습니다. 숫자를 변경하고 기존 관리형 노드 그룹이 있는 경우 노드 그룹의 다음 AMI 또는 시작 템플릿 업데이트로 인해 새 노드에 변경된 값이 나타납니다.
- IP 주소 할당에서 IP 접두사 할당으로 전환할 때 기존 노드를 순차적으로 바꾸는 대신 새 노드 그룹을 생성하여 사용 가능한 IP 주소 수를 늘리는 것이 좋습니다. IP 주소와 접두사가 모두 할당된 노드에서 Pods를 실행하면 광고된 IP 주소 용량에 불일치가 발생하여 노드의 향후 워크로드에 영향을 줄 수 있습니다. 전환을 수행하는 권장 방법은 Amazon EKS 모범 사례 안내서의 Replace all nodes when migrating from Secondary IP mode to Prefix Delegation mode or vice versa를 참조하세요.
- Linux 노드만 있는 클러스터의 경우
  - 네트워크 인터페이스에 접두사를 할당하도록 추가 기능을 구성한 후에는 클러스터의 모든 노드 그룹에 있는 모든 노드를 제거하지 않으면 `1.9.0` 또는 `1.10.1` 이전 버전에 Amazon VPC CNI plugin for Kubernetes 추가 기능을 다운로드할 수 없습니다.
  - `POD_SECURITY_GROUP_ENFORCING_MODE=standard` 및 `AWS_VPC_K8S_CNI_EXTERNALSNAT=false`로 설정하고 Pods에 보안 그룹도 사용하는 경우 Pods가 VPC 외부의 엔드포인트와 통신할 때 Pods에 할당한 보안 그룹이 아닌 노드의 보안 그룹이 사용됩니다.

    `POD_SECURITY_GROUP_ENFORCING_MODE=strict`로 설정하고 Pods에 보안 그룹도 사용하는 경우 Pods가 VPC 외부의 엔드포인트와 통신할 때 `Pod's` 보안 그룹이 사용됩니다.

사전 조건

- 기존 클러스터가 있어야 합니다. 배포하려면 Amazon EKS 클러스터 생성 부분을 참조하세요.
- Amazon EKS 노드가 있는 서브넷에는 충분한 연속 /28(IPv4 클러스터용) 또는 /80(IPv6 클러스터용) Classless Inter-Domain Routing(CIDR) 블록이 있어야 합니다. IPv6 클러스터에는 Linux 노드만 있을 수 있습니다. IP 주소가 서브넷 CIDR 전체에 분산되어 있는 경우 IP 접두사를 사용하지 못할 수 있습니다. 다음과 같이 하는 것이 좋습니다.
  - 서브넷 CIDR 예약을 사용하여 예약된 범위 내의 IP 주소가 아직 사용 중이더라도 릴리스 시 IP 주소가 다시 할당되지 않도록 합니다. 이렇게 하면 분할 없이 접두사를 할당할 수 있습니다.
  - IP 접두사가 할당된 워크로드 실행에 특별히 사용되는 새 서브넷을 사용합니다. IP 접두사를 할당할 때 Windows 및 Linux 워크로드를 모두 동일한 서브넷에서 실행할 수 있습니다.
- 노드에 IP 접두사를 할당하려면 노드가 AWS Nitro를 기반으로 해야 합니다. Nitro 기반이 아닌 인스턴스는 계속해서 개별 보조 IP 주소를 할당하지만 Nitro-based 인스턴스보다 Pods에 할당할 IP 주소 수가 상당히 적습니다.
- Linux 노드만 있는 클러스터의 경우 – 클러스터가 IPv4 패밀리용으로 구성된 경우 버전 1.9.0 이상의 Amazon VPC CNI plugin for Kubernetes 추가 기능이 설치되어 있어야 합니다. 현재 설치된 버전은 다음과 같은 명령으로 확인할 수 있습니다.

  ```
  kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/" -f 2
  ```

  클러스터가 IPv6 패밀리용으로 구성된 경우 버전 1.10.1의 추가 기능이 설치되어 있어야 합니다. 플러그인 버전이 필요한 버전보다 이전인 경우 업데이트해야 합니다. 자세한 내용은 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업의 업데이트 부분을 참조하세요.
- Windows 노드만 있는 클러스터의 경우
  - 클러스터와 해당 플랫폼 버전은 다음 표에 나와 있는 버전 이상이어야 합니다. 클러스터 버전을 업그레이드하려면 Amazon EKS 클러스터 Kubernetes 버전 업데이트 섹션을 참조하세요. 클러스터가 최소 플랫폼 버전이 아닌 경우 Amazon EKS가 플랫폼 버전을 업데이트할 때까지 노드에 IP 접두사를 할당할 수 없습니다.

Kubernetes 버전	플랫폼 버전
1.27	eks.3
1.26	eks.4

Kubernetes 버전	플랫폼 버전
1.25	eks.5

aws eks describe-cluster --name *my-cluster* --query 'cluster.{"Kubernetes Version": version, "Platform Version": platformVersion}' 명령에서 *my-cluster*를 클러스터 이름으로 바꾼 다음 수정된 명령을 실행하여 현재 Kubernetes 및 플랫폼 버전을 확인할 수 있습니다.

- 클러스터에 대해 Windows 지원이 활성화되었습니다. 자세한 내용은 Amazon EKS 클러스터에 대해 Windows 지원 사용 설정 섹션을 참조하세요.

Amazon EC2 노드에 사용 가능한 IP 주소의 증량하려면

1. 노드에 IP 주소 접두사를 할당하도록 클러스터를 구성합니다. 해당 노드의 운영 체제와 일치하는 탭에서 절차를 완료합니다.

   Linux

   1. 파라미터를 활성화하여 Amazon VPC CNI DaemonSet의 네트워크 인터페이스에 접두사를 할당합니다. 1.21 이상의 클러스터를 배포하는 경우 Amazon VPC CNI plugin for Kubernetes 추가 기능의 버전 1.10.1 이상은 함께 배포됩니다. IPv6 패밀리를 사용하여 클러스터를 생성한 경우 이 설정은 기본적으로 true로 설정되었습니다. IPv4 패밀리를 사용하여 클러스터를 생성한 경우 이 설정은 기본적으로 false로 설정되었습니다.

      ```
      kubectl set env daemonset aws-node -n kube-system
       ENABLE_PREFIX_DELEGATION=true
      ```

      Important

      서브넷에 사용 가능한 IP 주소가 있더라도 서브넷이 인접한 /28 블록을 사용할 수 없는 경우, Amazon VPC CNI plugin for Kubernetes 로그에 다음 오류가 표시됩니다.

      ```
      InsufficientCidrBlocks: The specified subnet does not have enough free
       cidr blocks to satisfy the request
      ```

이는 서브넷에 분산된 기존 보조 IP 주소의 조각화로 인해 발생할 수 있습니다. 이 오류를 해결하려면 새 서브넷을 생성하고 거기서 Pods를 시작하거나 Amazon EC2 서브넷 CIDR 예약을 사용하여 접두사 할당과 함께 사용할 서브넷 내에 공간을 예약합니다. 자세한 내용은 Amazon VPC 사용 설명서의 서브넷 CIDR 예약을 참조하세요.

2. 시작 템플릿을 사용하지 않고 관리형 노드 그룹을 배포하거나 AMI ID를 지정하지 않은 시작 템플릿을 사용하여 배포하고, 사전 조건에 나열된 버전 이상의 Amazon VPC CNI plugin for Kubernetes 버전을 사용하고 있는 경우 다음 단계로 건너뜁니다. 관리형 노드 그룹은 자동으로 최대 Pods 수를 계산합니다.

   AMI ID를 지정한 시작 템플릿을 사용하여 자체 관리형 노드 그룹 또는 관리형 노드 그룹을 배포하는 경우 Amazon EKS 권장 노드의 최대 Pods 수를 결정해야 합니다. 각 Amazon EC2 인스턴스 유형의 Amazon EKS 권장 최대 Pods 수의 지침을 따라 3단계에 **`--cni-prefix-delegation-enabled`**를 추가하세요. 이후 단계에서 사용하기 위해 출력을 적어둡니다.

   > **Important**
   >
   > 관리형 노드 그룹은 `maxPods`의 값에 최대 수를 적용합니다. vCPU가 30개 미만인 인스턴스의 경우 최대 수는 110이고 다른 모든 인스턴스의 경우 최대 수는 250입니다. 이 최대 수는 접두사 위임의 활성화 여부에 관계없이 적용됩니다.

3. IPv6용으로 구성된 `1.21` 이상 클러스터를 사용하는 경우 다음 단계로 건너뜁니다.

   다음 옵션 중 하나에서 파라미터를 지정합니다. 사용자에게 적합한 옵션과 이 옵션을 제공할 값을 결정하려면 GitHub에서 `WARM_PREFIX_TARGET`, `WARM_IP_TARGET`, `MINIMUM_IP_TARGET` 섹션을 참조하세요.

   *`example values`*을 0보다 큰 값으로 바꿀 수 있습니다.

   - `WARM_PREFIX_TARGET`

     ```
     kubectl set env ds aws-node -n kube-system WARM_PREFIX_TARGET=1
     ```

   - `WARM_IP_TARGET` 또는 `MINIMUM_IP_TARGET` - 둘 중 하나의 값이 설정되면 `WARM_PREFIX_TARGET`에 설정된 모든 값을 재정의합니다.

```
kubectl set env ds aws-node -n kube-system WARM_IP_TARGET=5
```

```
kubectl set env ds aws-node -n kube-system MINIMUM_IP_TARGET=2
```

4. 하나 이상의 Amazon EC2 Nitro Amazon Linux 2 인스턴스 유형으로 다음 유형의 노드 그룹 중 하나를 생성합니다. Nitro 인스턴스 유형의 목록은 Linux 인스턴스용 Amazon EC2 사용 설명서에서 Nitro 시스템에 구축된 인스턴스 섹션을 참조하세요. 이 기능은 Windows에서 지원되지 않습니다. *110*을 포함하는 옵션의 경우 3단계의 값(권장) 또는 고유한 값으로 바꿉니다.
   - 자체 관리형 - 자체 관리형 Amazon Linux 노드 시작하기의 지침에 따라 노드 그룹을 배포합니다. BootstrapArguments 파라미터에 다음 텍스트를 지정합니다.

     ```
     --use-max-pods false --kubelet-extra-args '--max-pods=110'
     ```

     `eksctl`을 사용하여 노드 그룹을 생성하려면 다음 명령을 사용합니다.

     ```
     eksctl create nodegroup --cluster my-cluster --managed=false --max-pods-per-node 110
     ```

   - 관리형(Managed) - 다음 옵션 중 하나를 사용하여 노드 그룹을 배포합니다.
     - 시작 템플릿을 사용하지 않거나 AMI ID가 지정되지 않은 시작 템플릿을 사용하는 경우(Without a launch template or with a launch template without an AMI ID specified) - 관리형 노드 그룹 생성의 절차를 완료합니다. 관리형 노드 그룹은 자동으로 Amazon EKS 권장 최대 `max-pods` 값을 계산합니다.
     - AMI ID가 지정된 시작 템플릿을 사용하는 경우(With a launch template with a specified AMI ID) - 시작 템플릿에서 Amazon EKS 최적화 AMI ID 또는 Amazon EKS 최적화 AMI에서 빌드한 사용자 지정 AMI를 지정한 다음 시작 템플릿을 사용하여 노드 그룹 배포를 지정하고 시작 템플릿에 다음 사용자 데이터를 입력합니다. 이 사용자 데이터는 인수를 `bootstrap.sh` 파일에 전달합니다. 부트스트랩 파일에 대한 자세한 내용은 GitHub에서 bootstrap.sh 섹션을 참조하세요.

       ```
       /etc/eks/bootstrap.sh my-cluster \
         --use-max-pods false \
         --kubelet-extra-args '--max-pods=110'
       ```

       `eksctl`을 사용하여 노드 그룹을 생성하려면 다음 명령을 사용합니다.

```
eksctl create nodegroup --cluster my-cluster --max-pods-per-node 110
```

Amazon EKS 최적화 AMI에서 빌드되지 않은 사용자 지정 AMI를 생성한 경우 직접 구성을 사용자 지정하여 생성해야 합니다.

> **Note**
>
> 또한 인스턴스의 서브넷과 다른 서브넷의 Pods에 IP 주소를 할당하려면 이 단계에서 기능을 활성화해야 합니다. 자세한 내용은 포드에 대한 사용자 지정 네트워킹 섹션을 참조하세요.

Windows

1. IP 접두사의 할당을 활성화합니다.
   a. 편집을 위해 `amazon-vpc-cni ConfigMap`을 엽니다.

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. 다음 줄을 `data` 섹션에 추가합니다.

   ```
     enable-windows-prefix-delegation: "true"
   ```

   c. 파일을 저장하고 편집기를 닫습니다.
   d. 이 줄이 `ConfigMap`에 추가되었는지 확인합니다.

   ```
   kubectl get configmap -n kube-system amazon-vpc-cni -o
    "jsonpath={.data.enable-windows-prefix-delegation}"
   ```

   반환된 출력이 `true`가 아닌 경우 오류가 있을 수 있습니다. 단계를 다시 완료해 봅니다.

> **Important**
>
> 서브넷에 사용 가능한 IP 주소가 있더라도 서브넷이 인접한 /28 블록을 사용할 수 없는 경우 노드 이벤트에 다음 오류가 표시됩니다.

```
"failed to allocate a private IP/Prefix address:
 InsufficientCidrBlocks: The specified subnet does not have enough
 free cidr blocks to satisfy the request"
```

이는 서브넷에 분산된 기존 보조 IP 주소의 조각화로 인해 발생할 수 있습니다. 이 오류를 해결하려면 새 서브넷을 생성하고 거기서 Pods를 시작하거나 Amazon EC2 서브넷 CIDR 예약을 사용하여 접두사 할당과 함께 사용할 서브넷 내에 공간을 예약합니다. 자세한 내용은 Amazon VPC 사용 설명서의 서브넷 CIDR 예약을 참조하세요.

2. (선택 사항) 클러스터의 사전 규모 조정 및 동적 규모 조정 동작을 제어하기 위한 추가 구성을 지정합니다. 자세한 내용은 GitHub의 Configuration options with Prefix Delegation mode on Windows를 참조하세요.

   a. 편집을 위해 `amazon-vpc-cni ConfigMap`을 엽니다.

   ```
   kubectl edit configmap -n kube-system amazon-vpc-cni -o yaml
   ```

   b. *`example values`*를 0보다 큰 값으로 바꾸고 필요한 항목을 `ConfigMap`의 `data` 섹션에 추가합니다. `warm-ip-target` 또한 `minimum-ip-target`에 대한 값을 설정하면 해당 값이 `warm-prefix-target`에 대해 설정된 모든 값을 재정의합니다.

   ```
   warm-prefix-target: "1"
   warm-ip-target: "5"
   minimum-ip-target: "2"
   ```

   c. 파일을 저장하고 편집기를 닫습니다.

3. 하나 이상의 Amazon EC2 Nitro 인스턴스 유형으로 Windows 노드 그룹을 생성합니다. Nitro 인스턴스 유형의 목록은 Amazon EC2 - Linux 인스턴스용 사용 설명서의 Nitro 시스템에 구축된 인스턴스를 참조하세요. 기본적으로 노드에 배포할 수 있는 최대 Pods 수는 110개입니다. 이 수를 늘리거나 줄이려면 부트스트랩 구성의 사용자 데이터에 다음을 지정합니다. *`max-pods-quantity`*를 최대 포드 수 값으로 바꿉니다.

   ```
   -KubeletExtraArgs '--max-pods=max-pods-quantity'
   ```

   관리형 노드 그룹을 배포하는 경우 시작 템플릿에 이 구성을 추가해야 합니다. 자세한 내용은 사용자 지정 시작 템플릿이 있는 관리형 노드 섹션을 참조하세요. Windows 부트스트랩

스크립트의 구성 파라미터에 대한 자세한 내용은 부트스트랩 스크립트 구성 파라미터 섹션을 참조하세요.

2. 노드가 배포되면 클러스터의 노드를 확인합니다.

```
kubectl get nodes
```

예제 출력은 다음과 같습니다.

```
NAME                                               STATUS     ROLES    AGE   VERSION
ip-192-168-22-103.region-code.compute.internal   Ready      <none>   19m   v1.XX.X-eks-6b7464
ip-192-168-97-94.region-code.compute.internal    Ready      <none>   19m   v1.XX.X-eks-6b7464
```

3. 노드 중 하나를 설명하여 노드에 대한 max-pods 값과 사용 가능한 IP 주소 수를 결정합니다. *192.168.30.193*을 이전 출력에서 반환된 노드 중 하나의 이름에 있는 IPv4 주소로 바꿉니다.

```
kubectl describe node ip-192-168-30-193.region-code.compute.internal | grep 'pods\|PrivateIPv4Address'
```

예제 출력은 다음과 같습니다.

```
pods:                                       110
vpc.amazonaws.com/PrivateIPv4Address:  144
```

이전 출력에서 110은 *144*개의 IP 주소를 사용할 수 있더라도 Kubernetes가 노드에 배포할 최대 Pods 수입니다.

Pods의 보안 그룹

Pods의 보안 그룹은 Amazon EC2 보안 그룹을 Kubernetes Pods와 통합합니다. Amazon EC2 보안 그룹을 사용하여 여러 Amazon EC2 인스턴스 유형 및 Fargate 에서 실행 중인 노드에 배포하는 Pods와의 인바운드 및 아웃바운드 네트워크 트래픽을 허용하는 규칙을 정의할 수 있습니다. 이 기능에 대한 자세한 설명은 Pods의 보안 그룹 소개 블로그 게시물을 참조하세요.

고려 사항

- Pods의 보안 그룹을 배포하기 전에 다음 제한 사항 및 조건을 고려하세요.
- Pods의 보안 그룹은 Windows 노드에서 사용할 수 없습니다.

- Pods의 보안 그룹은 Amazon VPC CNI 플러그인 버전 1.16.0 이상을 사용하여 Amazon EC2 노드가 포함된 IPv6 제품군에 대해 구성된 클러스터에서 사용할 수 있습니다. Amazon VPC CNI 플러그인 버전 1.7.7 이상을 사용하여 Fargate 노드만 포함하는 IPv6 제품군에 대해 구성된 클러스터에서 Pods의 보안 그룹을 사용할 수 있습니다. 자세한 내용은 클러스터, Pods 및 services용 IPv6 주소 단원을 참조하십시오.
- Pods에 대한 보안 그룹은 대다수 Nitro 기반 Amazon EC2 인스턴스 패밀리에서 지원하지만, 패밀리의 모든 세대에서 지원하지는 않습니다. 예를 들면 m5, c5, r5, p3, m6g, c6g 및 r6g 인스턴스 패밀리와 세대는 지원됩니다. t 패밀리의 인스턴스 유형은 지원되지 않습니다. 지원되는 인스턴스 유형의 전체 목록은 GitHub의 limits.go 파일을 참조하세요. 노드는 해당 파일에 IsTrunkingCompatible: true가 있는 나열된 인스턴스 유형 중 하나여야 합니다.
- Pod 보안 정책을 사용하여 Pod 변형에 대한 액세스 권한을 제한하는 경우에는 psp가 할당된 role의 Kubernetes ClusterRoleBinding에 eks:vpc-resource-controller Kubernetes 사용자가 지정되어야 합니다. 기본 Amazon EKS psp, role 및 ClusterRoleBinding을 사용하는 경우 eks:podsecuritypolicy:authenticated ClusterRoleBinding입니다. 예를 들어 다음 예와 같이 사용자를 subjects: 섹션에 추가합니다.

```
[...]
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: system:authenticated
  - apiGroup: rbac.authorization.k8s.io
    kind: User
    name: eks:vpc-resource-controller
  - kind: ServiceAccount
    name: eks-vpc-resource-controller
```

- Pods의 보안 그룹 및 사용자 지정 네트워킹을 함께 사용하는 경우 Pods에 대해 보안 그룹에서 지정한 보안 그룹이 ENIConfig에 지정된 보안 그룹 대신 사용됩니다.
- 버전 1.10.2 이하의 Amazon VPC CNI 플러그인을 사용하고 Pod 사양에 terminationGracePeriodSeconds 설정이 포함된 경우 설정 값은 0이 될 수 없습니다.
- 버전 1.10 이하의 Amazon VPC CNI 플러그 인이나 기본 설정인 POD_SECURITY_GROUP_ENFORCING_MODE=strict로 버전 1.11를 사용하는 경우, externalTrafficPolicy를 Local로 설정한 인스턴스 대상을 사용하는 유형 NodePort 및 LoadBalancer의 Kubernetes 서비스는 보안 그룹을 할당하는 Pods에서 지원하지 않습니다. 인스턴스 대상에서 로드 밸런서 사용에 대한 자세한 내용은 Amazon EKS의 네트워크 로드 밸런싱 섹션을 참조하세요.

- 버전 `1.10` 이하의 Amazon VPC CNI 플러그 인이나 기본 설정인 `POD_SECURITY_GROUP_ENFORCING_MODE=strict`로 버전 `1.11`를 사용하는 경우, 소스 NAT는 보안 그룹이 할당된 Pods에서 아웃바운드 트래픽에 대해 사용 중지되므로 아웃바운드 보안 그룹 규칙이 적용됩니다. 인터넷에 액세스하려면 보안 그룹이 할당된 Pods는 NAT 게이트웨이 또는 인스턴스로 구성된 프라이빗 서브넷에 배포된 노드에서 실행해야 합니다. 퍼블릭 서브넷에 배포된 보안 그룹이 할당된 Pods는 인터넷에 액세스할 수 없습니다.

  `POD_SECURITY_GROUP_ENFORCING_MODE=standard`로 설정된 버전 `1.11` 이상의 플러그 인을 사용하는 경우 VPC 외부로 향하는 Pod 트래픽은 인스턴스의 기본 네트워크 인터페이스의 IP 주소로 변환됩니다. 이 트래픽의 경우 Pod's 보안 그룹의 규칙이 아닌 기본 네트워크 인터페이스에 대한 보안 그룹의 규칙이 사용됩니다.
- 연결된 보안 그룹이 있는 Pods를 통해 Calico 네트워크 정책을 사용하려면 버전 `1.11.0` 이상의 Amazon VPC CNI 플러그 인을 사용하고 `POD_SECURITY_GROUP_ENFORCING_MODE=standard`로 설정해야 합니다. 그렇지 않으면 연결된 보안 그룹이 있는 Pods에서 주고받는 트래픽 흐름에는 Calico 네트워크 정책이 적용되지 않으며 Amazon EC2 보안 그룹으로만 적용이 제한됩니다. Amazon VPC CNI 버전을 업데이트하려면 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 섹션을 참조하세요.
- Nodelocal DNSCache를 사용하는 클러스터에서 보안 그룹을 사용하는 Amazon EC2 노드에서 실행되는 Pods는 `POD_SECURITY_GROUP_ENFORCING_MODE=standard`로 설정된 버전 `1.11.0` 이상의 Amazon VPC CNI 플러그 인에서만 지원됩니다. Amazon VPC CNI 플러그인 버전을 업데이트하려면 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 섹션을 참조하세요.
- Pods의 보안 그룹은 변동이 많은 Pods의 경우 Pod 시작 지연 시간 증가로 이어질 수 있습니다. 이는 리소스 컨트롤러의 속도 제한 때문입니다.

Pods용 보안 그룹의 Amazon VPC CNI plugin for Kubernetes 구성

Pods의 보안 그룹 배포

Fargate Pods의 보안 그룹만 사용하고 클러스터에 Amazon EC2 노드가 없는 경우 애플리케이션 예 배포로 건너뜁니다.

1. 현재 Amazon VPC CNI plugin for Kubernetes 버전은 다음 명령을 통해 확인할 수 있습니다.

```
kubectl describe daemonset aws-node --namespace kube-system | grep amazon-k8s-cni:
 | cut -d : -f 3
```

예제 출력은 다음과 같습니다.

```
v1.7.6
```

Amazon VPC CNI plugin for Kubernetes 버전이 `1.7.7` 이하인 경우 플러그인을 버전 `1.7.7` 이상으로 업데이트합니다. 자세한 내용은 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 단원을 참조하십시오.

2. `AmazonEKSVPCResourceController` 관리형 IAM 정책을 Amazon EKS 클러스터에 연결된 클러스터 역할에 추가합니다. 정책을 사용하면 역할이 네트워크 인터페이스, 프라이빗 IP 주소, 네트워크 인스턴스의 연결 및 분리를 관리할 수 있습니다.

   a. 클러스터 IAM 역할의 이름을 검색하고 변수에 저장합니다. *`my-cluster`*를 클러스터 이름으로 바꿉니다.

   ```
   cluster_role=$(aws eks describe-cluster --name my-cluster --query
    cluster.roleArn --output text | cut -d / -f 2)
   ```

   b. 책을 역할에 연결합니다.

   ```
   aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
   AmazonEKSVPCResourceController --role-name $cluster_role
   ```

3. `aws-node` DaemonSet에서 `ENABLE_POD_ENI` 변수를 `true`로 설정하여 Amazon VPC CNI 추가 기능이 Pods의 네트워크 인터페이스를 관리하도록 합니다. 이 설정이 `true`로 설정되면 클러스터의 각 노드에서 추가 기능이 cninode 사용자 지정 리소스를 만듭니다. VPC 리소스 컨트롤러는 설명 `aws-k8s-trunk-eni`를 사용하여 트렁크 네트워크 인터페이스라는 하나의 특수 네트워크 인터페이스를 생성하고 연결합니다.

```
kubectl set env daemonset aws-node -n kube-system ENABLE_POD_ENI=true
```

> **Note**
>
> 트렁크 네트워크 인터페이스는 인스턴스 유형에서 지원하는 최대 네트워크 인터페이스 수에 포함됩니다. 인스턴스 유형별로 지원되는 최대 네트워크 인터페이스 수 목록은 Linux 인스턴스용 Amazon EC2 사용 설명서에서 인스턴스 유형별 네트워크 인터페이스당 IP 주소 섹션을 참조하세요. 노드에 이미 최대 수의 표준 네트워크 인터페이스가 연결되어 있는 경우 VPC 리소스 컨트롤러가 공간을 예약합니다. 컨트롤러가 표준 네트워크 인터페이스

를 분리 및 삭제하고 트렁크 네트워크 인터페이스를 만든 다음 인스턴스에 연결할 수 있도록 실행 중인 Pods를 축소해야 합니다.

4. 다음 명령을 사용하여 CNINode 사용자 지정 리소스를 포함하는 노드를 확인할 수 있습니다. No resources found가 반환된 경우 몇 초 정도 기다렸다가 다시 시도하세요. 이전 단계에서는 몇 초 정도 걸리는 Amazon VPC CNI plugin for Kubernetes Pods를 다시 시작해야 합니다.

```
$ kubectl get cninode -A
     NAME FEATURES
     ip-192-168-64-141.us-west-2.compute.internal
 [{"name":"SecurityGroupsForPods"}]
     ip-192-168-7-203.us-west-2.compute.internal [{"name":"SecurityGroupsForPods"}]
```

1.15 이전의 VPC CNI 버전을 사용하는 경우 CNINode 사용자 지정 리소스 대신 노드 레이블이 사용됩니다. 다음 명령을 사용하여 노드 레이블 aws-k8s-trunk-eni가 true로 설정된 노드를 확인할 수 있습니다. No resources found가 반환된 경우 몇 초 정도 기다렸다가 다시 시도하세요. 이전 단계에서는 몇 초 정도 걸리는 Amazon VPC CNI plugin for Kubernetes Pods를 다시 시작해야 합니다.

```
kubectl get nodes -o wide -l vpc.amazonaws.com/has-trunk-attached=true
-
```

트렁크 네트워크 인터페이스가 만들어지면 트렁크 또는 표준 네트워크 인터페이스에서 Pods에 보조 IP 주소를 할당합니다. 노드가 삭제되면 트렁크 인터페이스가 자동으로 삭제됩니다.

이후 단계에서 Pod에 대한 보안 그룹을 배포하면 VPC 리소스 컨트롤러가 aws-k8s-branch-eni의 설명과 함께 브랜치 네트워크 인터페이스라는 특수한 네트워크 인터페이스를 생성하고 거기에 보안 그룹을 연결합니다. 브랜치 네트워크 인터페이스는 노드에 연결된 표준 및 트렁크 네트워크 인터페이스와 함께 생성됩니다.

라이브니스 또는 준비 프로브를 사용하는 경우 kubelet이 TCP를 사용하여 브랜치 네트워크 인터페이스의 Pods에 연결할 수 있도록 TCP 초기 demux를 사용 정지해야 합니다. TCP 초기 Demux를 비활성화하려면 다음 명령을 실행합니다.

```
kubectl patch daemonset aws-node -n kube-system \
  -p '{"spec": {"template": {"spec": {"initContainers": [{"env":
[{"name":"DISABLE_TCP_EARLY_DEMUX","value":"true"}],"name":"aws-vpc-cni-
init"}]}}}}'
```

> **Note**
>
> `POD_SECURITY_GROUP_ENFORCING_MODE=standard`로 설정된 `1.11.0` 이상의 Amazon VPC CNI plugin for Kubernetes 추가 기능을 사용하는 경우 다음 단계에서 설명한 대로 이전 명령을 실행할 필요가 없습니다.

5. 클러스터에서 `NodeLocal DNSCache`를 사용하거나, 자체 보안 그룹이 있는 Pods를 통해 Calico 네트워크 정책을 사용하거나, 보안 그룹을 할당하려는 Pods에 대해 `externalTrafficPolicy`를 `Local`로 설정한 인스턴스 대상을 사용하는 `NodePort` 및 `LoadBalancer` 유형의 Kubernetes 서비스가 있는 경우 버전 `1.11.0` 이상의 Amazon VPC CNI plugin for Kubernetes 추가 기능을 사용하고 있어야 하며 다음 설정을 활성화해야 합니다.

```
kubectl set env daemonset aws-node -n kube-system
 POD_SECURITY_GROUP_ENFORCING_MODE=standard
```

> **Important**
>
> - Pod 보안 그룹 규칙은 `kubelet` 또는 `nodeLocalDNS`와 같이 동일한 노드에 있는 Pods 간 또는 Pods와 services 간의 트래픽에 적용되지 않습니다. 동일한 노드에서 서로 다른 보안 그룹을 사용하는 포드는 서로 다른 서브넷에 구성되어 있기 때문에 통신할 수 없으며, 이러한 서브넷 사이에 라우팅이 비활성화되어 있습니다.
> - Pods에서 VPC 외부의 주소로 향하는 아웃바운드 트래픽은 인스턴스의 기본 네트워크 인터페이스의 IP 주소로 변환된 네트워크 주소입니다(`AWS_VPC_K8S_CNI_EXTERNALSNAT=true`로 설정하지 않은 경우 제외). 이 트래픽의 경우 Pod's 보안 그룹의 규칙이 아닌 기본 네트워크 인터페이스에 대한 보안 그룹의 규칙이 사용됩니다.
> - 이 설정을 기존 Pods에 적용하려면 Pods 또는 Pods가 실행되고 있는 노드를 다시 시작해야 합니다.

애플리케이션 예 배포

Pods의 보안 그룹을 사용하려면 다음 절차에 설명된 대로 클러스터에 기존 보안 그룹과 Amazon EKS `SecurityGroupPolicy` 배포가 있어야 합니다. 다음 단계에서는 Pod의 보안 그룹 정책을 사용하는 방법을 보여 줍니다. 달리 명시되지 않는 한, 변수가 터미널에 걸쳐 지속되지 않는 다음 단계에서 사용되므로 동일한 터미널에서 모든 단계를 완료합니다.

보안 그룹을 통해 Pod 에 배포

1. 리소스를 배포할 Kubernetes 네임스페이스를 생성합니다. 사용하려는 네임스페이스의 이름으로 *my-namespace*를 바꿀 수 있습니다.

```
kubectl create namespace my-namespace
```

2. Amazon EKS SecurityGroupPolicy를 클러스터에 배포합니다.

   a. 다음 콘텐츠를 디바이스에 복사합니다. 서비스 계정 레이블에 따라 Pods를 선택하려는 경우 *podSelector*를 **serviceAccountSelector**로 바꿀 수 있습니다. 두 선택기 중 하나만 지정해야 합니다. 비어 있는 podSelector(예: podSelector: {})는 네임스페이스의 모든 Pods를 선택합니다. *my-role*을 자신의 역할 이름으로 변경할 수 있습니다. 비어 있는 serviceAccountSelector는 네임스페이스의 모든 서비스 계정을 선택합니다. *my-security-group-policy*를 자신의 SecurityGroupPolicy에 대한 이름으로 바꾸고 SecurityGroupPolicy를 생성하려는 네임스페이스로 *my-namespace*를 바꿀 수 있습니다.

   *my_pod_security_group_id*를 기존 보안 그룹의 ID로 바꿔야 합니다. 기존 보안 그룹이 없으면 하나를 생성해야 합니다. 자세한 정보는 Linux 인스턴스용 Amazon EC2 사용 설명서의 Linux 인스턴스에 대한 Amazon EC2 보안 그룹을 참조하세요. 보안 그룹 ID는 1~5개를 지정할 수 있습니다. 둘 이상의 ID를 지정하면 모든 보안 그룹의 모든 규칙 조합이 선택한 Pods에 적용됩니다.

```
cat >my-security-group-policy.yaml <<EOF
apiVersion: vpcresources.k8s.aws/v1beta1
kind: SecurityGroupPolicy
metadata:
  name: my-security-group-policy
  namespace: my-namespace
spec:
  podSelector:
    matchLabels:
      role: my-role
  securityGroups:
    groupIds:
      - my_pod_security_group_id
EOF
```

> **Important**
>
> Pod에 대해 지정한 보안 그룹 또는 그룹은 다음 기준을 충족해야 합니다.
>
> - 존재해야 합니다. 존재하지 않을 경우 선택기와 일치하는 Pod를 배포하면 Pod가 생성 프로세스에 계속 남아 있습니다. Pod에 대해 설명하면 `An error occurred (InvalidSecurityGroupID.NotFound) when calling the CreateNetworkInterface operation: The securityGroup ID 'sg-05b1d815d1EXAMPLE' does not exist`와 유사한 오류 메시지가 표시됩니다.
> - 보안 그룹에서는 프로브를 구성한 모든 포트를 통해 노드에 적용된 보안 그룹의 인바운드 통신(kubelet의 경우)을 허용해야 합니다.
> - CoreDNS를 실행하는 Pods(또는 Pods가 실행되는 노드)에 할당된 보안 그룹에 대한 TCP 및 UDP 포트 53을 통한 아웃바운드 통신을 허용해야 합니다. CoreDNS Pods의 보안 그룹에서는 지정하는 보안 그룹의 인바운드 TCP 및 UDP 포트 53 트래픽을 허용해야 합니다.
> - 통신하려면 필요한 다른 Pods와 통신하는 데 필요한 인바운드 및 아웃바운드 규칙이 있어야 합니다.
> - Fargate에서 보안 그룹을 사용하는 경우 Pods가 Kubernetes 컨트롤 플레인과 통신할 수 있는 규칙이 있어야 합니다. 이 작업을 수행할 수 있는 가장 쉬운 방법은 클러스터 보안 그룹을 보안 그룹 중 하나로 지정하는 것입니다.
>
> 보안 그룹 정책은 새로 예약된 Pods에만 적용되며, 실행 중인 Pods에는 영향을 주지 않습니다.

b. 정책을 배포합니다.

```
kubectl apply -f my-security-group-policy.yaml
```

3. 이전 단계에서 지정한 *podSelector*의 *my-role* 값과 일치하는 레이블이 있는 샘플 애플리케이션을 배포합니다.

a. 다음 콘텐츠를 디바이스에 복사합니다. *## #*을 사용자 값으로 바꾼 다음 수정된 명령을 실행합니다. *my-role*을 바꾸는 경우 이전 단계에서 선택기에 지정한 값과 동일한지 확인합니다.

```
cat >sample-application.yaml <<EOF
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
  namespace: my-namespace
  labels:
    app: my-app
spec:
  replicas: 4
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
        role: my-role
    spec:
      terminationGracePeriodSeconds: 120
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: my-app
  namespace: my-namespace
  labels:
    app: my-app
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
EOF
```

b. 다음 명령을 사용하여 애플리케이션을 배포합니다. 애플리케이션을 배포할 때 `role` 레이블과 일치하는 Amazon VPC CNI plugin for Kubernetes과 이전 단계에서 지정한 보안 그룹이 Pod에 적용됩니다.

```
kubectl apply -f sample-application.yaml
```

4. 샘플 애플리케이션을 통해 배포된 Pods를 봅니다. 이 주제의 나머지 부분에서는 이 터미널을 `TerminalA`로 언급합니다.

```
kubectl get pods -n my-namespace -o wide
```

예제 출력은 다음과 같습니다.

```
NAME                             READY   STATUS    RESTARTS   AGE     IP
       NODE                                             NOMINATED NODE   READINESS
 GATES
my-deployment-5df6f7687b-4fbjm   1/1     Running   0          7m51s   192.168.53.48
    ip-192-168-33-28.region-code.compute.internal   <none>           <none>
my-deployment-5df6f7687b-j9fl4   1/1     Running   0          7m51s
  192.168.70.145   ip-192-168-92-33.region-code.compute.internal   <none>
 <none>
my-deployment-5df6f7687b-rjxcz   1/1     Running   0          7m51s
  192.168.73.207   ip-192-168-92-33.region-code.compute.internal   <none>
 <none>
my-deployment-5df6f7687b-zmb42   1/1     Running   0          7m51s   192.168.63.27
    ip-192-168-33-28.region-code.compute.internal   <none>           <none>
```

> **Note**
>
> - Pods가 `Waiting` 상태에서 멈춘 경우 `kubectl describe pod my-deployment-xxxxxxxxxx-xxxxx -n my-namespace`를 실행합니다. `Insufficient permissions: Unable to create Elastic Network Interface.`(권한 부족: 탄력적 네트워크 인터페이스를 생성할 수 없습니다.)가 표시되면 이전 단계에서 IAM 클러스터 역할에 IAM 정책을 추가했는지 확인합니다.
> - Pods가 `Pending` 상태로 멈춰 있는 경우 노드 인스턴스 유형이 limits.go에 나열되어 있고 인스턴스 유형에서 지원하는 최대 브랜치 네트워크 인터페이스 수의 제품에 노드 그룹의 노드 수를 곱한 값이 아직 충족되지 않았는지 확인합니다. 예를 들어 `m5.large` 인스턴스는 9개의 브랜치 네트워크 인터페이스를 지원합니다. 노드 그룹에

5개의 노드가 있는 경우 노드 그룹에 대해 최대 45개의 브랜치 네트워크 인터페이스를 생성할 수 있습니다. 배포하려는 46 번째 Pod는 연결된 보안 그룹이 있는 다른 Pod가 삭제될 때까지 Pending 상태로 남아 있습니다.

`kubectl describe pod` ***my-deployment-xxxxxxxxxx-xxxxx*** **-n** ***my-namespace***를 실행할 때 다음 메시지와 유사한 메시지가 표시될 경우 무시해도 됩니다. 이 메시지는 Amazon VPC CNI plugin for Kubernetes가 호스트 네트워킹을 설정하려고 할 때 네트워크 인터페이스가 생성되는 동안 실패할 경우 나타날 수 있습니다. 플러그인은 네트워크 인터페이스가 생성될 때까지 이 이벤트를 기록합니다.

```
Failed to create Pod sandbox: rpc error: code = Unknown desc = failed to set up
 sandbox container
 "e24268322e55c8185721f52df6493684f6c2c3bf4fd59c9c121fd4cdc894579f" network for Pod
 "my-deployment-5df6f7687b-4fbjm": networkPlugin
cni failed to set up Pod "my-deployment-5df6f7687b-4fbjm-c89wx_my-namespace"
 network: add cmd: failed to assign an IP address to container
```

인스턴스 유형에서 실행할 수 있는 최대 Pods 수를 초과할 수 없습니다. 각 인스턴스 유형에서 실행할 수 있는 최대 Pods 수 목록은 GitHub의 eni-max-pods.txt를 참조하세요. 연결된 보안 그룹이 있는 Pod를 삭제하거나 Pod가 실행되고 있는 노드를 삭제하면 VPC 리소스 컨트롤러가 브랜치 네트워크 인터페이스를 삭제합니다. 보안 그룹에 대해 Pods를 사용하는 Pods가 있는 클러스터를 삭제하면 컨트롤러가 브랜치 네트워크 인터페이스를 삭제하지 않으므로 직접 삭제해야 합니다. 네트워크 인터페이스를 삭제하는 방법에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 네트워크 인터페이스 삭제를 참조하세요.

5. 별도의 터미널에서 Pods 중 하나에 셸을 적용합니다. 이 주제의 나머지 부분에서는 이 터미널을 `TerminalB`로 언급합니다. ***5df6f7687b-4fbjm***을 이전 단계의 결과에서 반환된 Pods 중 하나의 ID로 바꿉니다.

```
kubectl exec -it -n my-namespace my-deployment-5df6f7687b-4fbjm -- /bin/bash
```

6. `TerminalB`의 셸에서 샘플 애플리케이션이 작동하는지 확인합니다.

```
curl my-app
```

예제 출력은 다음과 같습니다.

```
<!DOCTYPE html>
```

```
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

애플리케이션이 실행되는 모든 Pods는 사용자가 생성한 보안 그룹과 연결되기 때문에 이 결과가 표시됩니다. 해당 그룹에는 보안 그룹이 연결되는 모든 Pods 간에 모든 트래픽을 허용하는 규칙이 포함되어 있습니다. DNS 트래픽은 해당 보안 그룹에서 노드와 연결된 클러스터 보안 그룹으로 아웃바운드 트래픽이 허용됩니다. 노드가 Pods에서 이름 조회를 수행한 CoreDNS Pods를 실행하고 있습니다.

7. 보안 그룹에서 클러스터 보안 그룹으로의 DNS 통신을 허용하는 보안 그룹 규칙을 TerminalA에서 제거합니다. 이전 단계에서 클러스터 보안 그룹에 DNS 규칙을 추가하지 않은 경우 *$my_cluster_security_group_id*를 사용자가 규칙을 생성한 보안 그룹의 ID로 바꿉니다.

```
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_tcp_rule_id
aws ec2 revoke-security-group-ingress --group-id $my_cluster_security_group_id --
security-group-rule-ids $my_udp_rule_id
```

8. TerminalB에서 다시 애플리케이션에 액세스하려고 시도합니다.

```
curl my-app
```

예제 출력은 다음과 같습니다.

```
curl: (6) Could not resolve host: my-app
```

Pod에서 클러스터 보안 그룹을 연결해 주는 CoreDNS Pods에 더 이상 액세스할 수 없기 때문에 이 시도는 실패합니다. 클러스터 보안 그룹에는 Pod에 연결된 보안 그룹에서 DNS 통신을 허용하는 보안 그룹 규칙이 더 이상 없습니다.

이전 단계에서 Pods 중 하나에 대해 반환된 IP 주소를 사용하여 애플리케이션에 액세스하려고 시도하는 경우 보안 그룹이 연결되어 있는 Pods 간에 모든 포트가 허용되고 이름 조회가 필요 없기 때문에 응답이 계속 표시됩니다.

9. 실험이 끝나면 생성한 샘플 보안 그룹 정책, 애플리케이션 및 보안 그룹을 제거할 수 있습니다. TerminalA에서 다음 명령을 실행합니다.

```
kubectl delete namespace my-namespace
```

```
aws ec2 revoke-security-group-ingress --group-id $my_pod_security_group_id --
security-group-rule-ids $my_inbound_self_rule_id
wait
sleep 45s
aws ec2 delete-security-group --group-id $my_pod_security_group_id
```

Pods용 여러 네트워크 인터페이스

Multus CNI는 여러 네트워크 인터페이스를 Pod에 연결할 수 있는 Amazon EKS용 컨테이너 네트워크 인터페이스(CNI) 플러그 인입니다. 자세한 내용은 GitHub에서 Multus-CNI 설명서를 참조하세요.

Amazon EKS에서 각 Pod에는 Amazon VPC CNI 플러그 인에서 할당하는 하나의 네트워크 인터페이스가 있습니다. Multus를 사용하면 여러 인터페이스가 있는 멀티 호밍 Pod를 생성할 수 있습니다. 이는 여러 다른 CNI 플러그 인을 호출할 수 있는 CNI 플러그 인인 '메타 플러그 인'으로 작동하는 Multus를 통해 생성됩니다. AWS의 Multus 지원은 Amazon VPC CNI 플러그 인이 기본 위임 플러그 인으로 구성됩니다.

고려 사항

- Amazon EKS는 단일 루트 I/O 가상화(SR-IOV) 및 데이터 영역 개발 키트(DPDK) CNI 플러그 인을 구축하지도 게시하지도 않습니다. 그러나 Multus 관리형 호스트 디바이스 및 `ipvlan` 플러그 인를 통해 Amazon EC2 탄력적 네트워크 어댑터(ENA)에 직접 연결하여 패킷 가속화를 실현할 수 있습니다.
- Amazon EKS는 추가 CNI 플러그 인을 간단하게 연결할 수 있는 일반 프로세스를 제공하는 Multus를 지원합니다. Multus와 연결 프로세스가 지원되지만 AWS에서는 연결될 수 있는 호환 가능한 모든 CNI 플러그 인이나 연결 구성과 관련이 없는 CNI 플러그 인에서 발생할 수 있는 문제에 대한 지원을 제공하지 않습니다.
- Amazon EKS는 Multus 플러그 인에 대한 지원 및 수명 주기 관리를 제공하지만 추가 네트워크 인터페이스와 연결된 IP 주소 또는 추가 관리에 대해서는 책임을 지지 않습니다. Amazon VPC CNI 플러그 인을 사용하는 기본 네트워크 인터페이스의 IP 주소와 관리는 변경되지 않습니다.
- Amazon VPC CNI 플러그 인만 기본 위임 플러그 인으로 공식 지원됩니다. 기본 네트워킹에 대해 Amazon VPC CNI 플러그 인을 사용하지 않도록 선택한 경우 게시된 Multus 설치 매니페스트를 수정하여 기본 위임 플러그 인을 대체 CNI로 재구성해야 합니다.
- Multus는 Amazon VPC CNI를 기본 CNI로 사용하는 경우에만 지원됩니다. 상위 인터페이스 및 보조 또는 기타 인터페이스에 사용되는 경우 Amazon VPC CNI를 지원하지 않습니다.
- Amazon VPC CNI 플러그인이 Pods에 할당된 추가 네트워크 인터페이스를 관리하지 못하도록 하려면 네트워크 인터페이스에 다음 태그를 추가합니다.

키: `node.k8s.amazonaws.com/no_manage`

값: `true`

- Multus는 네트워크 정책과 호환되지만 Pods에 연결된 추가 네트워크 인터페이스의 일부일 수 있는 포트 및 IP 주소를 포함하도록 정책을 강화해야 합니다.

구현에 대한 자세한 내용은 GitHub에서 Multus 설정 안내서를 참조하세요.

호환 가능한 대체 CNI 플러그 인

Amazon VPC CNI plugin for Kubernetes는 Amazon EKS에서 지원하는 유일한 CNI 플러그인입니다. Amazon EKS는 업스트림 Kubernetes을 실행하므로 클러스터의 Amazon EC2 노드에 대체 호환 CNI 플러그인을 설치할 수 있습니다. 클러스터에 Fargate 노드가 있는 경우 해당 Amazon VPC CNI plugin for Kubernetes은(는) 이미 Fargate 노드에 있습니다. Fargate 노드와 함께 사용할 수 있는 유일한 CNI 플러그인입니다. Fargate 노드에 대체 CNI 플러그인을 설치하려는 시도가 실패합니다.

Amazon EC2 노드에서 대체 CNI 플러그인을 사용하려는 경우 해당 플러그인에 대한 상용 지원을 받거나 CNI 플러그인 프로젝트의 문제를 해결하고 프로젝트 수정에 기여할 수 있는 전문 지식을 내부적으로 보유하는 것이 좋습니다.

Amazon EKS는 호환 가능한 대체 CNI 플러그 인을 지원하는 파트너 네트워크와 관계를 유지합니다. 버전, 자격 요건 및 수행된 테스트에 대한 자세한 내용은 다음 파트너 설명서를 참조하세요.

파트너	제품	설명서
Tigera	Calico	설치 지침
Isovalent	Cilium	설치 지침
주니퍼	클라우드 네이티브 콘트레일 네트워킹(CN2)	설치 지침
VMware	Antrea	설치 지침

Amazon EKS는 모든 사용 사례를 포괄하는 다양한 옵션을 제공하는 것을 목표로 합니다. 여기에 나열되지 않은 상업적으로 지원되는 Kubernetes CNI 플러그인을 개발하는 경우 자세한 내용은 파트너 팀(aws-container-partners@amazon.com)으로 문의하세요.

AWS Load Balancer Controller란 무엇인가요?

AWS Load Balancer Controller는 Kubernetes 클러스터의 AWS Elastic Load Balancer를 관리합니다. 컨트롤러를 사용하여 클러스터 앱을 인터넷에 노출할 수 있습니다. 컨트롤러는 클러스터 Service 또는 Ingress 리소스를 가리키는 AWS 로드 밸런서를 프로비저닝합니다. 다시 말해 컨트롤러는 클러스터의 여러 포드를 가리키는 단일 IP 주소 또는 DNS 이름을 생성합니다.

컨트롤러는 Kubernetes Ingress 또는 Service 리소스를 감시합니다. 이에 대한 응답으로 해당 AWS Elastic Load Balancing 리소스를 생성합니다. Kubernetes 리소스에 주석을 적용하여 로드 밸런서의 특정 동작을 구성할 수 있습니다. 예를 들어 주석을 사용하여 로드 밸런서에 AWS 보안 그룹을 연결할 수 있습니다.

이 컨트롤러는 다음 리소스를 프로비저닝합니다.

Kubernetes `Ingress`

LBC는 Kubernetes `Ingress` 생성 시 AWS Application Load Balancer(ALB)를 생성합니다. Ingress 리소스에 적용할 수 있는 주석을 검토합니다.

`LoadBalancer` 유형의 Kubernetes 서비스

LBC는 `LoadBalancer` 유형의 Kubernetes 서비스 생성 시 AWS Network Load Balancer(NLB)를 생성합니다. Service 리소스에 적용할 수 있는 주석을 검토합니다.

과거에는 인스턴스 대상에 대해 Kubernetes Network Load Balancer를 사용했지만 IP 대상에 대해서는 LBC를 사용했습니다. AWS Load Balancer Controller 버전 `2.3.0` 이상에서 대상 유형 중 하나를 사용하여 NLB를 생성할 수 있습니다. NLB 대상 유형에 대한 자세한 내용은 Network Load Balancer 사용 설명서에서 대상 유형을 참조하세요.

컨트롤러는 GitHub에서 관리되는 오픈 소스 프로젝트입니다.

컨트롤러를 배포하기 전에 Amazon EKS 애플리케이션 로드 밸런싱 및 Amazon EKS의 네트워크 로드 밸런싱에서 필수 구성 요소 및 고려 사항을 검토하는 것이 좋습니다. 이러한 주제에서는 AWS 로드 밸런서가 포함된 샘플 앱을 배포합니다.

컨트롤러 배포

- the section called “Helm을 사용하여 설치” 방법을 알아봅니다. Amazon EKS를 처음 사용하는 경우 이 절차를 사용하세요. 이 절차에서는 Kubernetes용 패키지 관리자인 Helm과 `eksctl`을 사용하여 LBC 설치를 간소화합니다.

- 또는 the section called “매니페스트를 사용한 설치”. 이 절차는 고급 클러스터 구성에 해당됩니다. 여기에는 퍼블릭 컨테이너 레지스트리에 대한 네트워크 액세스가 제한된 클러스터가 포함됩니다.

사용되지 않는 버전 제거

- 사용되지 않는 AWS Load Balancer Controller 버전이 설치된 경우 the section called “사용되지 않는 컨트롤러에서 마이그레이션” 방법을 알아보세요.
- 사용되지 않는 버전은 업그레이드할 수 없습니다. 이를 제거하고 최신 AWS Load Balancer Controller 버전을 설치해야 합니다.
- 사용되지 않는 버전은 다음과 같습니다.
  - AWS ALB Ingress Controller for Kubernetes("Ingress Controller"), 이전 AWS Load Balancer Controller.
  - AWS Load Balancer Controller의 모든 `0.1.x` 버전

레거시 클라우드 공급자

Kubernetes에는 AWS에 대한 레거시 클라우드 공급자가 포함됩니다. 레거시 클라우드 공급자는 AWS Load Balancer Controller와 마찬가지로 AWS 로드 밸런서를 프로비저닝할 수 있습니다. 레거시 클라우드 공급자가 Classic Load Balancer를 생성합니다. AWS Load Balancer Controller를 설치하지 않은 경우 Kubernetes에서 기본적으로 레거시 클라우드 공급자를 사용합니다. AWS Load Balancer Controller를 설치하고 레거시 클라우드 공급자 사용을 자제해야 합니다.

> **Important**
>
> 버전 2.5 이상에서는 AWS Load Balancer Controller이(가) `type: LoadBalancer`와(과) 함께 Kubernetes 서비스 리소스의 기본 컨트롤러가 되며 각 서비스에 대한 AWS Network Load Balancer(NLB)를 만듭니다. 이는 서비스에 대한 변형 웹훅을 만들어 이를 수행하며, 이는 `type: LoadBalancer`의 새 서비스에 대한 `spec.loadBalancerClass` 필드를 `service.k8s.aws/nlb`(으)로 설정합니다. 차트 Helm 값을 `enableServiceMutatorWebhook`에서 `false`(으)로 설정하여 이 기능을 끄고 레거시 클라우드 공급자를 기본 컨트롤러로 사용하도록 되돌릴 수 있습니다. 이 기능을 끄지 않는 한 클러스터는 서비스에 새 Classic Load Balancer를 프로비저닝하지 않습니다. 기존 Classic Load Balancer는 계속 작동합니다.

Helm을 사용하여 AWS Load Balancer Controller 설치

이 주제에서는 Kubernetes용 패키지 관리자인 Helm과 `eksctl`을 사용하여 AWS Load Balancer Controller를 설치하는 방법을 설명합니다. 컨트롤러는 기본 옵션으로 설치됩니다. 주석을 사용한 구성에 관한 세부 정보를 포함한 컨트롤러에 관한 자세한 내용은 GitHub의 AWS Load Balancer Controller 설명서를 참조하세요.

다음 단계에서 *`example values`*를 사용자의 값으로 교체합니다.

필수 조건

이 튜토리얼에서는 Amazon EKS 클러스터를 생성하고 관리할 때 필요한 다음 도구 및 리소스를 설치하고 구성해야 합니다.

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요.
- 클러스터에 대한 기존 AWS Identity and Access Management(IAM) OpenID Connect(OIDC) 제공업체입니다. 이미 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 부분을 참조하세요.
- Amazon VPC CNI plugin for Kubernetes, `kube-proxy` 및 CoreDNS 추가 기능이 서비스 계정 토큰에 나열된 최소 버전인지 확인합니다.
- AWS Elastic Load Balancing을 숙지합니다. 자세한 내용은 Elastic Load Balancing 사용 설명서를 참조하세요.
- Kubernetes 서비스 및 수신 리소스를 숙지합니다.

- Helm은 로컬에 설치됩니다.

1단계: **`eksctl`**을 사용하여 IAM 역할 생성

> **Note**
>
> AWS 계정당 AWS Load Balancer Controller의 IAM 역할 하나만 생성해야 합니다. IAM 콘솔에 `AmazonEKSLoadBalancerControllerRole`이 있는지 확인합니다. 이 역할이 있는 경우 the section called “2단계: AWS Load Balancer Controller 설치” 단계로 건너뜁니다.

IAM 정책을 생성합니다.

1. 사용자 대신 AWS API를 호출할 수 있는 AWS Load Balancer Controller의 IAM 정책을 다운로드합니다.

   AWS

   ```
   $ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy.json
   ```

   AWS GovCloud (US)

   ```
   $ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy_us-gov.json
   ```

   ```
   $ mv iam_policy_us-gov.json iam_policy.json
   ```

2. 이전 단계에서 다운로드한 정책을 사용하여 IAM 정책을 만듭니다.

   ```
   $ aws iam create-policy \
       --policy-name AWSLoadBalancerControllerIAMPolicy \
       --policy-document file://iam_policy.json
   ```

   > **Note**
   >
   > AWS Management Console에서 정책을 보는 경우 콘솔에 ELB 서비스에 대한 경고는 표시되지만 ELB v2 서비스에 대한 경고는 표시되지 않습니다. 이는 정책의 작업 중 일부가 ELB v2에는 있지만 ELB에는 없기 때문에 발생합니다. ELB에 대한 경고는 무시해도 됩니다.

eksctl을 사용하여 IAM 역할 생성

- *my-cluster*를 사용자 클러스터 이름으로 바꾸고 *111122223333*을 계정 ID로 바꾼 다음 명령을 실행합니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

  ```
  $ eksctl create iamserviceaccount \
    --cluster=my-cluster \
  ```

```
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-
arn=arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve
```

2단계: AWS Load Balancer Controller 설치

Helm V3을 사용하여 AWS Load Balancer Controller 설치

1. `eks-charts` 차트 Helm 리포지토리를 추가합니다. AWS에서는 이 리포지토리를 GitHub에 유지합니다.

```
$ helm repo add eks https://aws.github.io/eks-charts
```

2. 최신 차트가 적용되도록 로컬 리포지토리를 업데이트합니다.

```
$ helm repo update eks
```

3. AWS Load Balancer Controller를 설치합니다.

   *my-cluster*를 클러스터 이름으로 바꿉니다. 다음 명령에서 `aws-load-balancer-controller`는 이전 단계에서 생성한 Kubernetes 서비스 계정입니다.

   차트 Helm 구성에 관한 자세한 내용은 GitHub에서 `values.yaml`을 참조하세요.

```
$ helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=my-cluster \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller
```

   a. Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 대해 제한적인 액세스 권한이 있는 Amazon EC2 노드에 컨트롤러를 배포하거나 Fargate에 배포하는 경우, 다음 `helm` 명령에 다음 플래그를 추가합니다.

      - **--set region=*region-code***
      - **--set vpcId=*vpc-xxxxxxxx***

   b. 차트 Helm 및 로드 밸런서 컨트롤러의 사용 가능한 버전을 보려면 다음 명령을 사용합니다.

```
helm search repo eks/aws-load-balancer-controller --versions
```

Important

배포된 차트는 보안 업데이트를 자동으로 수신하지 않습니다. 새 차트가 사용 가능해지면 수동으로 업그레이드해야 합니다. 업그레이드할 때는 이전 명령에서 *install*을 **upgrade**로 변경합니다.
`helm install` 명령은 컨트롤러의 사용자 지정 리소스 정의(CRDs)를 자동으로 설치합니다. `helm upgrade` 명령은 설치하지 않습니다. `helm upgrade`,를 사용하는 경우 CRDs를 수동으로 설치해야 합니다. 다음 명령을 실행하여 CRDs를 설치합니다.

```
wget https://raw.githubusercontent.com/aws/eks-charts/master/stable/aws-load-balancer-controller/crds/crds.yaml
kubectl apply -f crds.yaml
```

3단계: 컨트롤러가 설치되어 있는지 확인

1. 컨트롤러가 설치되어 있는지 확인합니다.

```
$ kubectl get deployment -n kube-system aws-load-balancer-controller
```

예제 출력은 다음과 같습니다.

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

Helm을 사용하여 배포한 경우 이전 출력이 표시됩니다. Kubernetes 매니페스트를 사용하여 배포한 경우 복제본이 하나만 있습니다.

2. 컨트롤러를 사용하여 AWS 리소스를 프로비저닝하기 전에 클러스터가 특정 요구 사항을 충족해야 합니다. 자세한 내용은 Amazon EKS 애플리케이션 로드 밸런싱 및 Amazon EKS의 네트워크 로드 밸런싱 섹션을 참조하세요.

Kubernetes 매니페스트를 사용하여 AWS Load Balancer Controller 추가 기능 설치

이 주제에서는 Kubernetes 매니페스트를 다운로드하고 적용하여 컨트롤러를 설치하는 방법을 설명합니다. GitHub에서 컨트롤러에 대한 전체 설명서를 볼 수 있습니다.

다음 단계에서 *example values*를 사용자의 값으로 교체합니다.

필수 조건

이 튜토리얼에서는 Amazon EKS 클러스터를 생성하고 관리할 때 필요한 다음 도구 및 리소스를 설치하고 구성해야 합니다.

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요.
- 클러스터에 대한 기존 AWS Identity and Access Management(IAM) OpenID Connect(OIDC) 제공업체입니다. 이미 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 부분을 참조하세요.
- Amazon VPC CNI plugin for Kubernetes, `kube-proxy` 및 CoreDNS 추가 기능이 서비스 계정 토큰에 나열된 최소 버전인지 확인합니다.
- AWS Elastic Load Balancing을 숙지합니다. 자세한 내용은 Elastic Load Balancing 사용 설명서를 참조하세요.
- Kubernetes 서비스 및 수신 리소스를 숙지합니다.

1단계: IAM 구성

> **Note**
>
> AWS 계정당 AWS Load Balancer Controller의 IAM 역할 하나만 생성해야 합니다. IAM 콘솔에 `AmazonEKSLoadBalancerControllerRole`이 있는지 확인합니다. 이 역할이 있는 경우 the section called “2단계: `cert-manager` 설치” 단계로 건너뜁니다.

IAM 정책을 생성합니다.

1. 사용자 대신 AWS API를 호출할 수 있는 AWS Load Balancer Controller의 IAM 정책을 다운로드합니다.

AWS

```
$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy.json
```

AWS GovCloud (US)

```
$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/install/iam_policy_us-gov.json
```

```
$ mv iam_policy_us-gov.json iam_policy.json
```

2. 이전 단계에서 다운로드한 정책을 사용하여 IAM 정책을 만듭니다.

```
$ aws iam create-policy \
    --policy-name AWSLoadBalancerControllerIAMPolicy \
    --policy-document file://iam_policy.json
```

> **Note**
>
> AWS Management Console에서 정책을 보는 경우 콘솔에 ELB 서비스에 대한 경고는 표시되지만 ELB v2 서비스에 대한 경고는 표시되지 않습니다. 이는 정책의 작업 중 일부가 ELB v2에는 있지만 ELB에는 없기 때문에 발생합니다. ELB에 대한 경고는 무시해도 됩니다.

eksctl

`eksctl`을 사용하여 IAM 역할 생성

- *`my-cluster`*를 사용자 클러스터 이름으로 바꾸고 *`111122223333`*을 계정 ID로 바꾼 다음 명령을 실행합니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

```
$ eksctl create iamserviceaccount \
  --cluster=my-cluster \
  --namespace=kube-system \
  --name=aws-load-balancer-controller \
```

```
  --role-name AmazonEKSLoadBalancerControllerRole \
  --attach-policy-
arn=arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve
```

AWS CLI and kubectl

AWS CLI 및 **kubectl**을 사용하여 IAM 역할 생성

1. 클러스터의 OIDC 제공업체 ID를 검색하고 변수에 저장합니다.

```
oidc_id=$(aws eks describe-cluster --name my-cluster --query
 "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
```

2. 클러스터 ID를 가진 IAM OIDC 제공업체가 이미 계정에 있는지 확인합니다. 클러스터와 IAM 모두에 OIDC를 구성해야 합니다.

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

   출력이 반환되면 해당 클러스터에 대한 IAM OIDC 제공업체가 이미 있는 것입니다. 출력이 반환되지 않은 경우 해당 클러스터에 대한 IAM OIDC 제공업체를 생성해야 합니다. 자세한 내용은 클러스터에 대한 IAM OIDC 공급자 생성 단원을 참조하십시오.

3. 다음 콘텐츠를 디바이스에 복사합니다. *111122223333*을 계정 ID로 바꿉니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다. *EXAMPLED539D4633E53DE1B71EXAMPLE*을 이전 단계에서 반환된 출력으로 바꿉니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다. 텍스트를 바꾼 후 수정된 명령을 실행하여 `load-balancer-role-trust-policy.json` 파일을 생성합니다.

```
cat >load-balancer-role-trust-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.region-code.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
```

```
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.region-code.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:kube-
system:aws-load-balancer-controller"
                }
            }
        }
    ]
}
EOF
```

4. IAM 역할을 생성합니다.

```
aws iam create-role \
  --role-name AmazonEKSLoadBalancerControllerRole \
  --assume-role-policy-document file://"load-balancer-role-trust-policy.json"
```

5. 필요한 Amazon EKS 관리형 IAM 정책을 IAM 역할에 연결합니다. *111122223333*을 계정 ID로 바꿉니다.

```
aws iam attach-role-policy \
  --policy-arn
 arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerIAMPolicy \
  --role-name AmazonEKSLoadBalancerControllerRole
```

6. 다음 콘텐츠를 디바이스에 복사합니다. *111122223333*을 계정 ID로 바꿉니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다. 텍스트를 바꾼 후 수정된 명령을 실행하여 `aws-load-balancer-controller-service-account.yaml` 파일을 생성합니다.

```
cat >aws-load-balancer-controller-service-account.yaml <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
```

```
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn:
 arn:aws:iam::111122223333:role/AmazonEKSLoadBalancerControllerRole
EOF
```

7. 클러스터에 Kubernetes 서비스 계정을 생성합니다. aws-load-balancer-controller라는 이름의 Kubernetes 서비스 계정에는 *AmazonEKSLoadBalancerControllerRole* 이름으로 생성된 IAM 역할을 사용하여 주석을 추가합니다.

```
$ kubectl apply -f aws-load-balancer-controller-service-account.yaml
```

2단계: **cert-manager** 설치

다음 방법 중 하나를 사용하여 cert-manager를 설치한 다음 인증서 구성을 Webhook에 주입합니다. 자세한 내용은 *cert-manager* 설명서의 시작하기를 참조하세요.

quay.io 컨테이너 레지스트리를 사용하여 cert-manager를 설치하는 것이 좋습니다. 노드가 quay.io 컨테이너 레지스트리에 액세스할 수 없는 경우 Amazon ECR을 사용하여 cert-manager를 설치하세요(아래 참조).

Quay.io

Quay.io를 사용하여 **cert-manager** 설치

- 노드에 quay.io 컨테이너 레지스트리에 대한 액세스 권한이 있는 경우, cert-manager를 설치한 다음 Webhook에 인증서 구성을 주입합니다.

```
$ kubectl apply \
    --validate=false \
    -f https://github.com/jetstack/cert-manager/releases/download/v1.13.5/cert-
manager.yaml
```

Amazon ECR

Amazon ECR을 사용하여 **cert-manager** 설치

1. 다음 방법 중 하나를 사용하여 cert-manager를 설치한 다음 인증서 구성을 Webhook에 주입합니다. 자세한 내용은 *cert-manager* 설명서의 시작하기를 참조하세요.

2. 매니페스트를 다운로드합니다.

```
curl -Lo cert-manager.yaml https://github.com/jetstack/cert-manager/releases/
download/v1.13.5/cert-manager.yaml
```

3. 다음 이미지를 가져와서 노드가 액세스할 수 있는 리포지토리로 푸시합니다. 이미지를 가져오기, 태그 지정 및 자체 리포지토리로 푸시하는 방법에 대한 자세한 내용은 한 리포지토리에서 다른 리포지토리로 컨테이너 이미지 복사 단원을 참조하세요.

```
quay.io/jetstack/cert-manager-cainjector:v1.13.5
quay.io/jetstack/cert-manager-controller:v1.13.5
quay.io/jetstack/cert-manager-webhook:v1.13.5
```

4. 자체 레지스트리 이름으로 세 개의 이미지에 대한 매니페스트 `quay.io`를 바꿉니다. 다음 명령은 개인 리포지토리의 이름이 원본 리포지토리와 같다고 가정합니다. *111122223333.dkr.ecr.region-code.amazonaws.com*을 개인 레지스트리로 바꿉니다.

```
$ sed -i.bak -e 's|quay.io|111122223333.dkr.ecr.region-code.amazonaws.com|' ./
cert-manager.yaml
```

5. 매니페스트를 적용합니다.

```
$ kubectl apply \
    --validate=false \
    -f ./cert-manager.yaml
```

3단계: AWS Load Balancer Controller 설치

Kubernetes 매니페스트를 사용하여 AWS Load Balancer Controller 설치

1. 컨트롤러 사양을 다운로드합니다. 컨트롤러에 대한 자세한 내용은 GitHub에서 설명서를 참조하세요.

```
curl -Lo v2_7_2_full.yaml https://github.com/kubernetes-sigs/aws-load-balancer-
controller/releases/download/v2.7.2/v2_7_2_full.yaml
```

2. 파일에 대해 편집한 항목:

a. v2_7_2_full.yaml 파일을 다운로드 한 경우 다음 명령을 실행하여 매니페스트에서 ServiceAccount 섹션을 제거합니다. 이 섹션을 제거하지 않으면 이전 단계에서 서비스 계정에 작성한 필수 주석이 덮어씌워집니다. 이 섹션을 제거하면 컨트롤러를 삭제할 경우 이전 단계에서 생성한 서비스 계정도 유지됩니다.

```
$ sed -i.bak -e '596,604d' ./v2_7_2_full.yaml
```

다른 파일 버전을 다운로드한 경우 편집기에서 파일을 열고 다음 줄을 제거합니다.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/component: controller
    app.kubernetes.io/name: aws-load-balancer-controller
  name: aws-load-balancer-controller
  namespace: kube-system
---
```

b. *my-cluster*를 해당 클러스터 이름으로 바꿔 파일의 Deployment spec 섹션에 있는 your-cluster-name을 해당 클러스터의 이름으로 바꿉니다.

```
$ sed -i.bak -e 's|your-cluster-name|my-cluster|' ./v2_7_2_full.yaml
```

c. 노드가 Amazon EKS Amazon ECR 이미지 리포지토리에 액세스할 수 없는 경우 다음 이미지를 가져와서 노드가 액세스할 수 있는 리포지토리로 푸시해야 합니다. 이미지를 가져오기, 태그 지정 및 자체 리포지토리로 푸시하는 방법에 대한 자세한 내용은 한 리포지토리에서 다른 리포지토리로 컨테이너 이미지 복사 단원을 참조하세요.

```
public.ecr.aws/eks/aws-load-balancer-controller:v2.7.2
```

매니페스트에 레지스트리 이름을 추가합니다. 다음 명령은 개인 리포지토리의 이름이 원본 리포지토리와 같다고 가정하고 개인 레스트리의 이름을 파일에 추가합니다. *111122223333.dkr.ecr.region-code.amazonaws.com*을 해당 레지스트리로 바꿉니다. 이 라인에서는 개인 리포지토리의 이름을 원본 리포지토리와 동일하게 지정했다고 가정합니다. 그렇지 않은 경우 개인 레지스트리 이름 뒤에 있는 eks/aws-load-balancer-controller 텍스트를 해당 리포지토리 이름으로 바꿉니다.

```
$ sed -i.bak -e 's|public.ecr.aws/eks/aws-load-balancer-
controller|111122223333.dkr.ecr.region-code.amazonaws.com/eks/aws-load-
balancer-controller|' ./v2_7_2_full.yaml
```

d. (Fargate 또는 제한된 IMDS에만 필요)

Amazon EC2 인스턴스 메타데이터 서비스(IMDS)에 대해 제한적인 액세스 권한이 있는 Amazon EC2 노드에 컨트롤러를 배포하거나 Fargate에 배포하는 경우, `- args:`에서 **`following parameters`**를 추가합니다.

```
[...]
spec:
      containers:
        - args:
            - --cluster-name=your-cluster-name
            - --ingress-class=alb
            - --aws-vpc-id=vpc-xxxxxxxx
            - --aws-region=region-code


[...]
```

3. 파일을 적용합니다.

```
$ kubectl apply -f v2_7_2_full.yaml
```

4. `IngressClass` 및 `IngressClassParams` 매니페스트를 클러스터에 다운로드합니다.

```
$ curl -Lo v2_7_2_ingclass.yaml https://github.com/kubernetes-sigs/aws-load-
balancer-controller/releases/download/v2.7.2/v2_7_2_ingclass.yaml
```

5. 매니페스트를 클러스터에 적용합니다.

```
$ kubectl apply -f v2_7_2_ingclass.yaml
```

4단계: 컨트롤러가 설치되어 있는지 확인

1. 컨트롤러가 설치되어 있는지 확인합니다.

```
$ kubectl get deployment -n kube-system aws-load-balancer-controller
```

예제 출력은 다음과 같습니다.

```
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           84s
```

Helm을 사용하여 배포한 경우 이전 출력이 표시됩니다. Kubernetes 매니페스트를 사용하여 배포한 경우 복제본이 하나만 있습니다.

2. 컨트롤러를 사용하여 AWS 리소스를 프로비저닝하기 전에 클러스터가 특정 요구 사항을 충족해야 합니다. 자세한 내용은 Amazon EKS 애플리케이션 로드 밸런싱 및 Amazon EKS의 네트워크 로드 밸런싱 섹션을 참조하세요.

사용되지 않는 컨트롤러에서 마이그레이션

이 주제에서는 사용되지 않는 컨트롤러 버전에서 마이그레이션하는 방법을 설명합니다. 보다 구체적으로 AWS Load Balancer Controller의 사용되지 않는 버전을 제거하는 방법을 설명합니다.

- 사용되지 않는 버전은 업그레이드할 수 없습니다. 이를 제거하고 최신 LBC 버전을 설치해야 합니다.
- 사용되지 않는 버전은 다음과 같습니다.
  - AWS ALB Ingress Controller for Kubernetes("Ingress Controller"), 이전 AWS Load Balancer Controller.
  - AWS Load Balancer Controller의 모든 `0.1.x` 버전

사용되지 않는 컨트롤러 버전 제거

> **Note**
>
> Helm을 사용하거나 Kubernetes 매니페스트를 사용하여 수동으로 사용되지 않는 버전을 설치했을 수 있습니다. 원래 설치한 도구를 사용하여 절차를 완료합니다.

Helm을 사용하여 Ingress Controller 제거

1. `incubator/aws-alb-ingress-controller` 차트 Helm이 설치된 경우 제거합니다.

```
$ helm delete aws-alb-ingress-controller -n kube-system
```

2. `eks-charts/aws-load-balancer-controller` 차트의 버전 `0.1.x`이 설치된 경우 제거합니다. 버전 `0.1.x`에서 `1.0.0`으로의 업그레이드는 웹후크 API 버전과 호환되지 않아 작동되지 않습니다.

```
$ helm delete aws-load-balancer-controller -n kube-system
```

Kubernetes 매니페스트를 사용하여 Ingress Controller 제거

1. 이 컨트롤러가 현재 설치되어 있는지 확인합니다.

```
$ kubectl get deployment -n kube-system alb-ingress-controller
```

컨트롤러가 설치되어 있지 않은 경우의 출력입니다.

서버에서 발생한 오류(찾을 수 없음): deployments.apps "alb-ingress-controller"를 찾을 수 없음

컨트롤러가 설치된 경우의 출력입니다.

```
NAME                     READY UP-TO-DATE AVAILABLE AGE
alb-ingress-controller 1/1   1          1         122d
```

2. 다음 명령을 입력하여 컨트롤러를 제거합니다.

```
$ kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-
ingress-controller/v1.1.8/docs/examples/alb-ingress-controller.yaml
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-
ingress-controller/v1.1.8/docs/examples/rbac-role.yaml
```

AWS Load Balancer Controller으로 마이그레이션

ALB Ingress Controller for Kubernetes에서 AWS Load Balancer Controller로 마이그레이션하려면 다음을 수행해야 합니다.

1. ALB Ingress Controller를 제거합니다(위 내용 참조).
2. AWS Load Balancer Controller를 설치합니다.

3. LBC에서 사용하는 IAM 역할에 정책을 추가합니다. 이 정책은 LBC가 ALB Ingress Controller for Kubernetes에서 생성한 리소스를 관리할 수 있도록 허용합니다.

AWS Load Balancer Controller IAM 역할에 마이그레이션 정책을 추가합니다.

1. IAM 정책을 다운로드합니다. 이 정책은 LBC가 ALB Ingress Controller for Kubernetes에서 생성한 리소스를 관리할 수 있도록 허용합니다. 정책을 볼 수도 있습니다.

```
$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-
controller/v2.7.2/docs/install/iam_policy_v1_to_v2_additional.json
```

2. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다..

```
$ sed -i.bak -e 's|arn:aws:|arn:aws-us-gov:|' iam_policy_v1_to_v2_additional.json
```

3. IAM 정책을 생성하고 반환된 ARN 적어둡니다.

```
$ aws iam create-policy \
  --policy-name AWSLoadBalancerControllerAdditionalIAMPolicy \
  --policy-document file://iam_policy_v1_to_v2_additional.json
```

4. IAM 정책을 LBC에서 사용하는 IAM 역할에 연결합니다. *your-role-name*을 역할의 이름(예: `AmazonEKSLoadBalancerControllerRole`)으로 바꿉니다.

   `eksctl`을 사용하여 역할을 생성한 경우 생성된 역할 이름을 찾으려면 AWS CloudFormation 콘솔을 열고 eksctl-*my-cluster*-addon-iamserviceaccount-kube-system-aws-load-balancer-controller 스택을 선택합니다. 리소스(Resources) 탭을 선택합니다. 실제 ID(Physical ID) 열에 역할 이름이 있습니다. 클러스터가 AWS GovCloud(미국 동부) 또는 AWS GovCloud(미국 서부) AWS 리전에 있는 경우 `arn:aws:`를 `arn:aws-us-gov:`로 바꿉니다.

```
$ aws iam attach-role-policy \
  --role-name your-role-name \
  --policy-arn
 arn:aws:iam::111122223333:policy/AWSLoadBalancerControllerAdditionalIAMPolicy
```

CoreDNS Amazon EKS 추가 기능을 사용한 작업

CoreDNS는 Kubernetes 클러스터 DNS로 사용할 수 있는 유연하고 확장 가능한 DNS 서버입니다. 하나 이상의 노드가 있는 Amazon EKS 클러스터를 시작하면 클러스터에 배포된 노드 수에 관계없이 CoreDNS 이미지의 복제본 2개가 기본적으로 배포됩니다. CoreDNS Pods는 클러스터의 모든 Pods의 이름을 확인합니다. 클러스터에 CoreDNS `deployment`의 네임 스페이스와 일치하는 네임스페이스의 AWS Fargate 프로파일이 포함되어 있는 경우 CoreDNS Pods를 Fargate 노드에 배포할 수 있습니다. CoreDNS에 대한 자세한 내용은 Kubernetes 설명서의 서비스 검색에 CoreDNS 사용을 참조하세요.

다음 표에는 각 Kubernetes 버전에 사용할 수 있는 Amazon EKS 추가 기능 유형의 최신 버전이 나열되어 있습니다.

Kubernetes 버전	1.29	1.28	1.27	1.26	1.25	1.24	1.23
	v1.11.1-eksbuild	v1.10.1-eksbuild	v1.10.1-eksbuild	v1.9.3-eksbuild.1	v1.9.3-eksbuild.1	v1.9.3-eksbuild.1	v1.8.7-eksbuild.10

> **Important**
>
> 이 추가 기능을 자체 관리하는 경우 표의 버전이 사용 가능한 자체 관리 버전과 다를 수 있습니다. 이 추가 기능의 자체 관리형 유형 업데이트에 대한 자세한 내용은 자체 관리형 추가 기능 업데이트 부분을 참조하세요.

중요 CoreDNS 업그레이드 고려 사항

- CoreDNS Deployment의 안정성과 가용성을 개선하기 위해 버전 `v1.9.3-eksbuild.5` 이상과 `v1.10.1-eksbuild.2`가 `PodDisruptionBudget`과 함께 배포됩니다. 기존 `PodDisruptionBudget` 버전을 배포한 경우 이러한 버전으로의 업그레이드가 실패할 수 있습니다. 업그레이드가 실패할 경우 다음 작업 중 하나를 수행하면 문제가 해결됩니다.
  - Amazon EKS 추가 기능을 업그레이드할 때 충돌 해결 옵션으로 기존 설정을 재정의하도록 선택합니다. Deployment에 다른 사용자 지정 설정을 지정한 경우 업그레이드 후 다른 사용자 지정 설정을 다시 적용할 수 있도록 업그레이드하기 전에 설정을 백업해야 합니다.
  - 기존 `PodDisruptionBudget`을 제거하고 업그레이드를 다시 시도하세요.

- EKS 추가 기능 버전 v1.9.3-eksbuild.3 이상 및 v1.10.1-eksbuild.6 이상에서는 CoreDNS Deployment는 /ready 엔드포인트를 사용하도록 readinessProbe를 설정합니다. 이 엔드포인트는 CoreDNS의 Corefile 구성 파일에서 활성화됩니다.

  사용자 지정 Corefile을 사용하는 경우 프로브가 사용할 /ready 엔드포인트가 CoreDNS에서 활성화되도록 구성에 ready 플러그인을 추가해야 합니다.

- EKS 추가 기능 버전 v1.9.3-eksbuild.7 이상 및 v1.10.1-eksbuild.4 이상에서는 PodDisruptionBudget을 변경할 수 있습니다. 다음 예제의 필드를 사용하여 선택적 구성 설정에서 추가 기능을 편집하고 이러한 설정을 변경할 수 있습니다. 이 예제는 기본 PodDisruptionBudget을 보여줍니다.

  ```
  {
      "podDisruptionBudget": {
          "enabled": true,
          "maxUnavailable": 1
          }
  }
  ```

  maxUnavailable 또는 minAvailable을 설정할 수 있지만 단일 PodDisruptionBudget에서 둘 다 설정할 수는 없습니다. PodDisruptionBudgets에 대한 자세한 내용은 Kubernetes 설명서의 PodDisruptionBudget 지정을 참조하세요.

  enabled을 false으로 설정한 경우 PodDisruptionBudget은 제거되지 않는다는 점에 유의하세요. 이 필드를 false(으)로 설정한 후에는 PodDisruptionBudget 객체를 삭제해야 합니다. 마찬가지로, PodDisruptionBudget이 있는 버전으로 업그레이드한 후 이전 버전의 추가 기능을 사용하도록 추가 기능을 편집(추가 기능 다운그레이드)해도 PodDisruptionBudget은 제거되지 않습니다. PodDisruptionBudget을 삭제하려면 다음 명령을 실행할 수 있습니다.

  ```
  kubectl delete poddisruptionbudget coredns -n kube-system
  ```

- EKS 추가 기능 버전 v1.10.1-eksbuild.5 이상에서는 KEP 2067을 준수하도록 기본 허용 범위를 node-role.kubernetes.io/master:NoSchedule에서 node-role.kubernetes.io/control-plane:NoSchedule로 변경합니다. KEP 2067에 대한 자세한 내용은 GitHub의 Kubernetes Enhancement Proposals (KEPs)에서 KEP-2067: Rename the kubeadm "master" label and taint를 참조하세요.

  EKS 추가 기능 버전 v1.8.7-eksbuild.8 이상 및 v1.9.3-eksbuild.9 이상에서 허용 범위는 모든 Kubernetes 버전과 호환되도록 설정됩니다.

- EKS 추가 기능 버전 `v1.9.3-eksbuild.11` 및 `v1.10.1-eksbuild.7` 이상에서는 CoreDNS Deployment가 `topologySpreadConstraints`에 대한 기본값을 설정합니다. 기본값은 사용 가능한 여러 가용 영역에 노드가 있는 경우 CoreDNS Pods가 가용 영역에 분산되도록 합니다. 기본값 대신 사용될 사용자 지정 값을 설정할 수 있습니다. 기본값은 다음과 같습니다.

```
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    whenUnsatisfiable: ScheduleAnyway
    labelSelector:
      matchLabels:
        k8s-app: kube-dns
```

CoreDNS v**1.11** 업그레이드 고려 사항

- EKS 추가 기능 버전 `v1.11.1-eksbuild.4` 이상에서 컨테이너 이미지는 최소 패키지를 포함하고 셸이 없는 Amazon EKS Distro에서 유지 관리하는 최소 기본 이미지를 기반으로 합니다. 자세한 내용은 Amazon EKS Distro를 참조하세요. CoreDNS 이미지의 사용법과 문제 해결은 동일하게 유지됩니다.

Amazon EKS 추가 기능 생성

추가 기능의 Amazon EKS 유형 생성. Check]를 선택합니다

필수 조건

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요.

1. 클러스터에 설치된 추가 기능의 버전을 확인하세요.

```
kubectl describe deployment coredns --namespace kube-system | grep coredns: | cut -d : -f 3
```

예제 출력은 다음과 같습니다.

```
v1.10.1-eksbuild.7
```

2. 클러스터에 설치된 추가 기능의 유형을 확인하세요. 클러스터를 생성하는 데 사용한 도구에 따라 현재 클러스터에 Amazon EKS 추가 기능이 유형이 설치되어 있지 않을 수 있습니다. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

버전 번호가 반환되는 경우 Amazon EKS 유형의 추가 기능이 클러스터에 설치되고 본 절차의 나머지 단계를 완료할 필요가 없습니다. 오류가 번호가 반환되는 경우 Amazon EKS 유형의 추가 기능이 클러스터에 설치되지 않습니다. 이 절차의 나머지 단계를 완료하여 설치하세요.

3. 현재 설치된 추가 기능의 구성을 저장합니다.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

4. AWS CLI를 사용하여 추가 기능을 생성합니다. AWS Management Console 또는 `eksctl`를 사용하여 추가 기능을 만들려면 추가 기능 생성 을 참조하여 추가 기능 이름을 `coredns`로 지정하십시오. 다음 명령을 디바이스에 복사합니다. 필요에 따라 명령을 다음과 같이 수정한 다음에 수정한 명령을 실행합니다.

   - *my-cluster*를 클러스터 이름으로 바꿉니다.
   - 클러스터 버전의 최신 버전 표에 나와 있는 최신 버전으로 *v1.11.1-eksbuild.6*을 바꿉니다.

```
aws eks create-addon --cluster-name my-cluster --addon-name coredns --addon-
version v1.11.1-eksbuild.6
```

Amazon EKS 추가 기능의 기본 설정과 충돌하는 사용자 지정 설정을 현재 추가 기능에 적용한 경우 생성이 실패할 수 있습니다. 생성에 실패하면 문제 해결에 도움이 될 수 있는 오류를 받게 됩니다. 또는 이전 명령에 **`--resolve-conflicts OVERWRITE`**을(를) 추가할 수 있습니다. 이렇게 하면 추가 기능이 기존 사용자 지정 설정을 덮어쓸 수 있습니다. 추가 기능을 만든 후에는 사용자 지정 설정으로 업데이트할 수 있습니다.

5. 클러스터 Kubernetes 버전에 맞는 추가 기능의 최신 버전이 클러스터에 추가되었는지 확인합니다. *my-cluster*를 클러스터 이름으로 바꿉니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 addon.addonVersion --output text
```

추가 기능 생성이 완료되는 데 몇 초 정도 걸릴 수 있습니다.

예제 출력은 다음과 같습니다.

```
v1.11.1-eksbuild.6
```

6. 원래 추가 기능을 사용자 지정 설정했다면 Amazon EKS 추가 기능을 생성하기 전에 이전 단계에서 저장한 구성을 사용하여 Amazon EKS 추가 기능을 사용자 지정 설정으로 업데이트하십시오.

Amazon EKS 추가 기능 업데이트

추가 기능의 Amazon EKS 유형 업데이트. 클러스터에 Amazon EKS 유형의 추가 기능을 추가하지 않은 경우 이 절차를 완료하는 대신 추가하거나 자체 관리형 추가 기능 업데이트 부분을 참조하세요.

1. 클러스터에 설치된 추가 기능의 버전을 확인하세요. *my-cluster*을 클러스터 이름으로 교체합니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
 "addon.addonVersion" --output text
```

예제 출력은 다음과 같습니다.

```
v1.10.1-eksbuild.7
```

반환된 버전이 최신 버전 표에 있는 클러스터의 Kubernetes 버전과 동일한 경우 클러스터에 이미 최신 버전이 설치되어 있으므로 이 절차의 나머지 부분을 완료하지 않아도 됩니다. 출력에 버전 번호 대신 오류가 표시되면 클러스터에 설치된 추가 기능의 Amazon EKS 유형이 없는 것입니다. 이 절차로 업데이트하려면 먼저 추가 기능을 생성해야 합니다.

2. 현재 설치된 추가 기능의 구성을 저장합니다.

```
kubectl get deployment coredns -n kube-system -o yaml > aws-k8s-coredns-old.yaml
```

3. AWS CLI를 사용하여 추가 기능 업데이트. AWS Management Console 또는 `eksctl`를 사용하여 추가 기능을 업데이트하려면 추가 기능 업데이트 부분을 참조하세요. 다음 명령을 디바이스에 복사합니다. 필요에 따라 명령을 다음과 같이 수정한 다음에 수정한 명령을 실행합니다.

   - *my-cluster*를 클러스터 이름으로 바꿉니다.
   - 클러스터 버전의 최신 버전 표에 나와 있는 최신 버전으로 *v1.11.1-eksbuild.6*을 바꿉니다.

- **--resolve-conflicts** *##* 옵션에서는 추가 기능의 기존 구성 값을 보존합니다. 추가 기능 설정에 사용자 지정 값을 설정하고 이 옵션을 사용하지 않는 경우 Amazon EKS에서는 기본값으로 해당 값을 덮어씁니다. 이 옵션을 사용하는 경우에는 추가 기능을 업데이트하기 전에 프로덕션 클러스터에서 비프로덕션 클러스터의 필드 및 값 변경 사항을 테스트하는 것이 좋습니다. 이 값을 OVERWRITE로 변경하면 모든 설정이 Amazon EKS 기본값으로 변경됩니다. 설정에 사용자 지정 값을 설정한 경우 Amazon EKS 기본값으로 해당 값을 덮어쓸 수도 있습니다. 이 값을 none으로 변경하면 Amazon EKS에서는 설정의 값을 변경하지 않지만 업데이트에 실패할 수도 있습니다. 업데이트에 실패하면 충돌 해결에 도움이 되는 오류 메시지가 표시됩니다.
- 구성 설정을 업데이트하지 않는 경우 **--configuration-values '{*"replicaCount":3*}'**을(를) 명령에서 제거하십시오. 구성 설정을 업데이트하는 경우, *"ReplicaCount":3#* 설정하려는 설정으로 바꾸십시오. 이 예제에서는 CoreDNS의 복제본 수가 3로 설정 되어 있습니다. 지정한 값은 구성 스키마에 유효해야 합니다. 구성 스키마를 모르는 경우 **aws eks describe-addon-configuration --addon-name coredns --addon-version *v1.11.1-eksbuild.6***을 실행하고 구성을 확인할 추가 기능의 버전 번호로 *v1.11.1-eksbuild.6*을 바꿉니다. 스키마가 출력에 반환됩니다. 기존 사용자 지정 구성이 있는데 이를 모두 제거하고 모든 설정의 값을 Amazon EKS 기본값으로 다시 설정하려면 명령에서 *"replicaCount":3*를 제거하여 **{}**을(를) 비워 두십시오. CoreDNS설정에 대한 자세한 내용은Kubernetes 설명서의 DNS 서비스 사용자 지정을 참조하십시오.

```
aws eks update-addon --cluster-name my-cluster --addon-name coredns --addon-
version v1.11.1-eksbuild.6 \
    --resolve-conflicts PRESERVE --configuration-values '{"replicaCount":3}'
```

업데이트가 완료되는 데 몇 초 정도 걸릴 수 있습니다.

4. 추가 기능 버전이 업데이트되었는지 확인합니다. *my-cluster*를 클러스터 이름으로 바꿉니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name coredns
```

업데이트가 완료되는 데 몇 초 정도 걸릴 수 있습니다.

예제 출력은 다음과 같습니다.

```
{
    "addon": {
        "addonName": "coredns",
        "clusterName": "my-cluster",
        "status": "ACTIVE",
```

```
        "addonVersion": "v1.11.1-eksbuild.6",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:region:111122223333:addon/my-cluster/coredns/
d2c34f06-1111-2222-1eb0-24f64ce37fa4",
        "createdAt": "2023-03-01T16:41:32.442000+00:00",
        "modifiedAt": "2023-03-01T18:16:54.332000+00:00",
        "tags": {},
        "configurationValues": "{\"replicaCount\":3}"
    }
}
```

자체 관리형 추가 기능 업데이트

> **Important**
>
> 자체 관리형 추가 기능 유형을 사용하는 대신 클러스터에 Amazon EKS 유형의 추가 기능을 추가하는 것이 좋습니다. 유형 간의 차이를 잘 모르는 경우 the section called “Amazon EKS 추가 기능” 부분을 참조하세요. Amazon EKS 추가 기능을 클러스터에 추가하는 방법에 대한 자세한 내용은 the section called “추가 기능 생성 ” 섹션을 참조하세요. Amazon EKS 추가 기능을 사용할 수 없는 경우, 사용할 수 없는 이유에 대한 문제를 컨테이너 로드맵 GitHub 리포지토리에 제출하는 것이 좋습니다.

1. 클러스터에 자체 관리형 추가 기능 유형이 설치되어 있는지 확인하세요. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다.

   ```
   aws eks describe-addon --cluster-name my-cluster --addon-name coredns --query
    addon.addonVersion --output text
   ```

   오류 메시지가 반환되면 자체 관리형 추가 기능 유형이 클러스터에 설치됩니다. 이 절차의 나머지 단계를 완료하여 설치하세요. 버전 번호가 반환되는 경우 Amazon EKS 유형의 추가 기능이 클러스터에 설치됩니다. Amazon EKS의 추가 기능을 업데이트하려면 이 절차를 수행하는 대신 Amazon EKS 추가 기능 업데이트의 절차를 수행하세요. 추가 기능 유형 간의 차이를 잘 모르는 경우 Amazon EKS 추가 기능 부분을 참조하세요.
2. 클러스터에 현재 설치된 컨테이너 이미지의 버전을 확인하세요.

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

예제 출력은 다음과 같습니다.

```
v1.8.7-eksbuild.2
```

3. 현재 CoreDNS 버전이 v1.5.0 이상이지만 CoreDNS 버전 표에 나열된 버전보다 낮은 경우 이 단계를 건너뜁니다. 현재 버전이 1.5.0 버전보다 낮은 경우 프록시 추가 기능 대신 전달 추가 기능을 사용하도록 CoreDNS의 ConfigMap을 수정해야 합니다.

   1. 다음 명령을 통해 Configmap을 엽니다.

      ```
      kubectl edit configmap coredns -n kube-system
      ```

   2. 다음 줄의 proxy를 forward로 바꿉니다. 파일을 저장하고 편집기를 종료합니다.

      ```
      proxy . /etc/resolv.conf
      ```

4. 원래 Kubernetes 1.17 또는 이전 버전에서 클러스터를 배포한 경우 CoreDNS 매니페스트에서 더는 사용되지 않는 줄을 제거해야 할 수 있습니다.

   > **Important**
   >
   > CoreDNS 버전 1.7.0으로 업데이트하기 전에 이 단계를 완료해야 하지만, 이전 버전으로 업데이트하는 경우에도 이 단계를 완료하는 것이 좋습니다.

   1. CoreDNS 매니페스트에 이 줄이 있는지 확인합니다.

      ```
      kubectl get configmap coredns -n kube-system -o jsonpath='{$.data.Corefile}' |
       grep upstream
      ```

      결과가 반환되지 않으면 매니페스트에 해당 줄이 없는 것이므로 다음 단계로 건너뛰어 CoreDNS를 업데이트할 수 있습니다. 출력이 반환되면 줄을 제거해야 합니다.

   2. 다음 명령을 사용하여 ConfigMap을 편집하고 파일에서 upstream 단어가 포함된 줄을 제거합니다. 파일에서 다른 것을 변경하지 마세요. 이 줄을 제거한 후 변경 사항을 저장하세요.

```
kubectl edit configmap coredns -n kube-system -o yaml
```

5. 현재 CoreDNS 이미지 버전을 검색합니다.

```
kubectl describe deployment coredns -n kube-system | grep Image
```

예제 출력은 다음과 같습니다.

```
602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.8.7-eksbuild.2
```

6. CoreDNS 1.8.3 이상으로 업데이트하는 경우 `system:coredns` Kubernetes `clusterrole`에 `endpointslices` 권한을 추가해야 합니다.

```
kubectl edit clusterrole system:coredns -n kube-system
```

파일의 `rules` 부분의 기존 권한 줄 아래에 다음 줄을 추가합니다.

```
[...]
- apiGroups:
  - discovery.k8s.io
  resources:
  - endpointslices
  verbs:
  - list
  - watch
[...]
```

7. *602401143452*와 *region-code*를 이전 단계에서 반환된 출력의 값으로 바꿔 CoreDNS 추가 기능을 업데이트합니다. Kubernetes버전의 최신 버전 표에 나와 있는 CoreDNS최신 버전으로 *v1.11.1-eksbuild.6*을(를) 교체하십시오.

```
kubectl set image deployment.apps/coredns -n kube-system
 coredns=602401143452.dkr.ecr.region-code.amazonaws.com/eks/coredns:v1.11.1-
eksbuild.6
```

예제 출력은 다음과 같습니다.

```
deployment.apps/coredns image updated
```

8. 컨테이너 이미지 버전을 다시 확인하여 이전 단계에서 지정한 버전으로 업데이트되었는지 확인합니다.

```
kubectl describe deployment coredns -n kube-system | grep Image | cut -d ":" -f 3
```

예제 출력은 다음과 같습니다.

```
v1.11.1-eksbuild.6
```

CoreDNS 지표

EKS 추가 기능 CoreDNS는 `kube-dns` 서비스의 Prometheus 형식으로 포트 9153의 CoreDNS에서 지표를 노출합니다. Prometheus, Amazon CloudWatch 에이전트 또는 기타 호환 시스템을 사용하여 이러한 지표를 스크레이프(수집)할 수 있습니다.

Prometheus 및 CloudWatch 에이전트 모두와 호환되는 스크레이프 구성의 예시는 Amazon CloudWatch 사용 설명서의 Prometheus에 대한 CloudWatch 에이전트 구성을 참조하세요.

Kubernetes **`kube-proxy`** 추가 기능으로 작업하기

> **Important**
>
> 자체 관리형 추가 기능 유형을 사용하는 대신 클러스터에 Amazon EKS 유형의 추가 기능을 추가하는 것이 좋습니다. 유형 간의 차이를 잘 모르는 경우 the section called “Amazon EKS 추가 기능” 부분을 참조하세요. Amazon EKS 추가 기능을 클러스터에 추가하는 방법에 대한 자세한 내용은 the section called “추가 기능 생성 ” 섹션을 참조하세요. Amazon EKS 추가 기능을 사용할 수 없는 경우, 사용할 수 없는 이유에 대한 문제를 컨테이너 로드맵 GitHub 리포지토리에 제출하는 것이 좋습니다.

`kube-proxy` 추가 기능은 Amazon EKS 클러스터의 각 Amazon EC2 노드에 배포됩니다. 노드에 대한 네트워크 규칙을 유지하고 Pods와(과)의 네트워크 통신을 가능하게 합니다. 이 추가 기능은 클러스터의 Fargate 노드에 배포되지 않습니다. 자세한 내용은 Kubernetes 설명서의 `kube-proxy`을(를) 참조하세요.

다음 표에는 각 Kubernetes 버전에 사용할 수 있는 Amazon EKS 추가 기능 유형의 최신 버전이 나열되어 있습니다.

Kubernetes 버전	1.29	1.28	1.27	1.26	1.25	1.24	1.23
	v1.29.1-eksbuild	v1.28.6-eksbuild	v1.27.1(eksbuil(2	v1.26.1:eksbuil(2	v1.25.1(eksbuil(3	v1.24.1:eksbuil(8	v1.23.17-eksbuild.9

> **Important**
>
> 이전 버전의 설명서가 올바르지 않았습니다. kube-proxy 버전 v1.28.5, v1.27.9, v1.26.12는 사용할 수 없습니다.
> 이 추가 기능을 자체 관리하는 경우 표의 버전이 사용 가능한 자체 관리 버전과 다를 수 있습니다.

각 Amazon EKS 클러스터 버전에서 사용할 수 있는 2가지 유형의 kube-proxy 컨테이너 이미지가 있습니다.

- Default(기본) - 이 이미지 유형은 Kubernetes 업스트림 커뮤니티에서 유지 관리하는 Debian 기반 도커 이미지를 기반으로 합니다.
- Minimal(최소) - 이 이미지 유형은 Amazon EKS Distro에서 유지 관리하는 최소 기본 이미지를 기반으로 하며, 최소한의 패키지가 포함되고 쉘이 없습니다. 자세한 내용은 Amazon EKS Distro를 참조하세요.

각 Amazon EKS 클러스터 버전에 사용할 수 있는 최신 자체 관리형 **kube-proxy** 컨테이너 이미지 버전

Image type(이미지 유형)	1.29	1.28	1.27	1.26	1.25	1.24	1.23
kube-proxy (기본 유형)	최소 유형만 사용할 수 있습니다.	최소 유형만 사용할 수 있습니다.	최소 유형만 사용할 수 있습니다.	최소 유형만 사용할 수 있습니다.	최소 유형만 사용할 수 있습니다.	v1.24.1(eksbuil(2	v1.23.16-eksbuild.2

Image type(이미지 유형)	1.29	1.28	1.27	1.26	1.25	1.24	1.23
kube-proxy (최소 유형)	v1.29.1-minimal-eksbuild.1	v1.28.6-minimal-eksbuild.1	v1.27.10-minimal-eksbuild	v1.26.13-minimal-eksbuild	v1.25.16-minimal-eksbuild	v1.24.17-minimal-eksbuild	v1.23.17-minimal-eksbuild.5

Important

- Kubernetes 버전 1.25 이상에서는 기본 이미지 유형을 사용할 수 없습니다. 최소 이미지 유형을 사용해야 합니다.
- Amazon EKS 추가 기능 유형을 업데이트할 때 유효한 Amazon EKS 추가 기능 버전을 지정합니다. 이 버전은 이 표에 나와 있는 버전이 아닐 수도 있습니다. Amazon EKS 추가 기능 버전이 이 추가 기능의 자체 관리형 유형을 업데이트할 때 지정한 컨테이너 이미지 버전과 항상 일치하지는 않기 때문입니다. 이 추가 기능의 자체 관리형 유형을 업데이트할 때는 이 표에 나열된 유효한 컨테이너 이미지 버전을 지정합니다.

필수 조건

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요.

고려 사항

- Amazon EKS 클러스터의 Kube-proxy에는 Kubernetes와(과) 동일한 호환성 및 스큐 정책이 있습니다. 추가 기능 버전 호환성 검색 방법에 대해 알아봅니다.
- Kube-proxy은(는) Amazon EC2 노드의 kubelet와(과) 동일한 마이너 버전이어야 합니다.
- Kube-proxy은(는) 클러스터 컨트롤 플레인의 마이너 버전보다 이후일 수 없습니다.
- 최근에 클러스터를 새 Kubernetes 마이너 버전으로 업데이트한 경우에는 노드와 동일한 마이너 버전으로 Amazon EC2 노드를 업데이트하기 전에 kube-proxy을(를) 노드와 동일한 마이너 버전으로 업데이트합니다.

kube-proxy 자체 관리형 추가 기능 업데이트

1. 클러스터에 자체 관리형 추가 기능 유형이 설치되어 있는지 확인하세요. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name kube-proxy --query
 addon.addonVersion --output text
```

   오류 메시지가 반환되면 자체 관리형 추가 기능 유형이 클러스터에 설치됩니다. 이 주제의 나머지 단계는 추가 기능의 자체 관리형 유형을 업데이트하는 단계입니다. 버전 번호가 반환되는 경우 Amazon EKS 유형의 추가 기능이 클러스터에 설치됩니다. 업데이트하려면 이 주제의 절차를 수행하는 대신 추가 기능 업데이트의 절차를 수행하세요. 추가 기능 유형 간의 차이를 잘 모르는 경우 Amazon EKS 추가 기능 섹션을 참조하세요.
2. 클러스터에 현재 설치된 컨테이너 이미지의 버전을 확인하세요.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image
```

   예제 출력은 다음과 같습니다.

```
Image:    602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.25.6-
minimal-eksbuild.2
```

   예제 출력에서 *v1.25.6-minimal-eksbuild.2*가 클러스터에 설치된 버전입니다.
3. 출력의 값으로 *602401143452* 및 *region-code*을(를) 대체하여 kube-proxy 추가 기능을 업데이트합니다. 이전 단계에서 각 Amazon EKS 클러스터에 사용 가능한 최신 자체 관리형 kube-proxy 컨테이너 이미지 버전 표에 나열된 kube-proxy 버전으로 *v1.26.2-minimal-eksbuild.2*을(를) 대체합니다. 기본 또는 최소 이미지 유형에 대한 버전 번호를 지정할 수 있습니다.

```
kubectl set image daemonset.apps/kube-proxy -n kube-system kube-
proxy=602401143452.dkr.ecr.region-code.amazonaws.com/eks/kube-proxy:v1.26.2-
minimal-eksbuild.2
```

   예제 출력은 다음과 같습니다.

```
daemonset.apps/kube-proxy image updated
```

4. 이제 클러스터에 새 버전이 설치되어 있는지 확인합니다.

```
kubectl describe daemonset kube-proxy -n kube-system | grep Image | cut -d ":" -f 3
```

예제 출력은 다음과 같습니다.

```
v1.26.2-minimal-eksbuild.2
```

5. 동일한 클러스터에서 x86 및 Arm 노드를 사용하고 있고 클러스터가 2020년 8월 17일 이전에 배포된 경우 다음 명령을 사용하여 여러 하드웨어 아키텍처에 대한 노드 셀렉터를 포함하도록 kube-proxy 매니페스트를 편집합니다. 이는 일회성 작업입니다. 매니페스트에 셀렉터를 추가한 후에는 추가 기능을 업데이트할 때마다 이 옵션을 추가할 필요가 없습니다. 클러스터가 2020년 8월 17일 또는 이후에 배포된 경우 kube-proxy은(는) 이미 멀티 아키텍처를 지원합니다.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

다음 노드 셀렉터를 편집기에서 파일에 추가한 후 파일을 저장합니다. 편집기에서 이 텍스트를 포함할 위치에 대한 예는 GitHub에서 CNI 매니페스트 파일을 참조하세요. 이를 통해 Kubernetes은(는) 노드의 하드웨어 아키텍처를 기반으로 올바른 하드웨어 이미지를 가져올 수 있습니다.

```
- key: "kubernetes.io/arch"
  operator: In
  values:
  - amd64
  - arm64
```

6. 원래 클러스터가 Kubernetes 버전 1.14 이상으로 생성된 경우 kube-proxy이(가) 이미 Affinity Rule에 포함되어 있으므로 이 단계를 건너뛸 수 있습니다. 원래 1.13 또는 이전 버전의 Kubernetes을(를) 사용하여 Amazon EKS 클러스터를 생성했으며 클러스터에서 Fargate 노드를 사용하려는 경우에는 Fargate 노드에서 kube-proxy Pods이(가) 예약되지 않도록 kube-proxy 매니페스트를 편집하여 NodeAffinity 규칙을 포함합니다. 이는 일회성 편집입니다. Affinity Rule을(를) 매니페스트에 추가하면 추가 기능을 업데이트할 때마다 추가할 필요가 없습니다. kube-proxy DaemonSet을(를) 편집합니다.

```
kubectl edit -n kube-system daemonset/kube-proxy
```

편집기에서 파일의 DaemonSet spec에 다음 Affinity Rule을(를) 추가한 후 파일을 저장합니다. 편집기에서 이 텍스트를 포함할 위치에 대한 예는 GitHub에서 CNI 매니페스트 파일을 참조하세요.

```
- key: eks.amazonaws.com/compute-type
  operator: NotIn
  values:
  - fargate
```

인터페이스 엔드포인트를 사용하여 Amazon Elastic Kubernetes Service 액세스(AWS PrivateLink)

AWS PrivateLink를 사용하여 VPC와 Amazon Elastic Kubernetes Service 사이에 프라이빗 연결을 생성할 수 있습니다. 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결을 사용하지 않고 VPC에 있는 것처럼 Amazon EKS에 액세스할 수 있습니다. VPC의 인스턴스에서 Amazon EKS에 액세스하는 데 퍼블릭 IP 주소가 필요하지 않습니다.

AWS PrivateLink에서 제공되는 인터페이스 엔드포인트를 생성하여 이 프라이빗 연결을 설정합니다. 인터페이스 엔드포인트에 대해 사용 설정하는 각 서브넷에서 엔드포인트 네트워크 인터페이스를 생성합니다. 이는 Amazon EKS로 향하는 트래픽의 진입점 역할을 하는 요청자 관리형 네트워크 인터페이스입니다.

자세한 내용은 AWS PrivateLink 가이드의 AWS PrivateLink를 통해 AWS 서비스에 액세스를 참조하세요.

Amazon EKS 고려 사항

- Amazon EKS에 대한 인터페이스 엔드포인트를 설정하려면 먼저 AWS PrivateLink 가이드의 Considerations(고려 사항)를 검토합니다.
- Amazon EKS에서는 인터페이스 엔드포인트를 통해 모든 API 작업에 대한 호출 수행을 지원하지만, Kubernetes API는 해당하지 않습니다. Kubernetes API 서버에서는 이미 프라이빗 엔드포인트를 지원합니다. Kubernetes API 서버 프라이빗 엔드포인트에서는 클러스터와 통신하는 데 사용하는 Kubernetes API 서버에 대한 프라이빗 엔드포인트를 생성합니다(`kubectl`과 같은 Kubernetes 관리 도구 사용). 노드와 API 서버 사이의 모든 통신이 VPC 내에 유지되도록 Kubernetes API 서버에 대한 프라이빗 액세스를 활성화할 수 있습니다. Amazon EKS API용 AWS PrivateLink는 트래픽을 퍼블릭 인터넷에 노출하지 않고 VPC의 Amazon EKS API를 호출하는 데 도움이 됩니다.
- 인터페이스 엔드포인트를 통해서만 액세스하도록 Amazon EKS를 구성할 수 없습니다.
- AWS PrivateLink의 표준 요금이 Amazon EKS의 인터페이스 엔드포인트에 적용됩니다. 각 가용 영역에서 인터페이스 엔드포인트가 프로비저닝된 각 시간과 인터페이스 엔드포인트를 통해 처리된 데이터에 대해 요금이 청구됩니다. 자세한 내용은 AWS PrivateLink 요금을 참조하세요.

- Amazon EKS에는 VPC 엔드포인트 정책이 지원되지 않습니다. 기본적으로 인터페이스 엔드포인트를 통해 Amazon EKS에 대한 전체 액세스가 허용됩니다. 또는 보안 그룹을 엔드포인트 네트워크 인터페이스와 연결하여 인터페이스 엔드포인트를 통해 Amazon EKS로 향하는 트래픽을 제어할 수 있습니다.
- VPC 흐름 로그를 사용하여 네트워크 인터페이스에서 송수신되는 IP 트래픽에 대한 정보를 캡처할 수 있습니다(인터페이스 엔드포인트 포함). 흐름 로그 데이터는 Amazon CloudWatch 또는 Amazon S3에 게시할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 흐름 로그를 사용하여 IP 트래픽 로깅을 참조하세요.
- Amazon EKS API를 인터페이스 엔드포인트가 있는 VPC에 연결하여 온프레미스 데이터 센터에서 Amazon EKS API에 액세스할 수 있습니다. AWS Direct Connect 또는 AWS Site-to-Site VPN을 사용하여 온프레미스 사이트를 VPC에 연결할 수 있습니다.
- AWS Transit Gateway 또는 VPC 피어링을 사용하여 인터페이스 엔드포인트가 있는 VPC에 다른 VPC를 연결할 수 있습니다. VPC 피어링은 두 VPC 간의 네트워킹 연결입니다. 사용자의 VPC 사이 또는 다른 계정의 VPC와 VPC 피어링 연결을 설정할 수 있습니다. VPC는 서로 다른 AWS 리전에 있을 수 있습니다. 피어링된 VPC 간 트래픽은 AWS 네트워크에 유지됩니다. 트래픽은 퍼블릭 인터넷을 통과하지 않습니다. 전송 게이트웨이는 VPC를 상호 연결하는 데 사용할 수 있는 네트워크 전송 허브입니다. VPC와 Transit Gateway 간 트래픽은 AWS 글로벌 프라이빗 네트워크에 남아 있습니다. 트래픽은 퍼블릭 인터넷에 노출되지 않습니다.
- Amazon EKS의 VPC 인터페이스 엔드포인트에는 `IPv4`를 통해서만 액세스할 수 있습니다. `IPv6`는 지원되지 않습니다.
- AWS PrivateLink 지원은 아시아 태평양(하이데라바드), 아시아 태평양(자카르타), 아시아 태평양(멜버른), 아시아 태평양(오사카), 캐나다 서부(캘거리), 유럽(스페인), 유럽(취리히), 이스라엘(텔아비브) 또는 중동(UAE) AWS 리전에서 제공되지 않습니다.

Amazon EKS용 인터페이스 엔드포인트 생성

Amazon VPC 콘솔 또는 AWS Command Line Interface(AWS CLI)를 사용하여 Amazon EKS의 인터페이스 엔드포인트를 생성할 수 있습니다. 자세한 정보는 AWS PrivateLink 가이드의 VPC 엔드포인트 생성을 참조하세요.

다음과 같은 서비스 이름을 사용하여 Amazon EKS의 인터페이스 엔드포인트를 생성합니다.

```
com.amazonaws.region-code.eks
```

프라이빗 DNS 기능은 Amazon EKS 및 기타 AWS 서비스의 인터페이스 엔드포인트를 생성할 때 기본적으로 활성화됩니다. 그러나 다음과 같은 VPC 속성이`true`: `enableDnsHostnames` 및

`enableDnsSupport`로 설정되었는지 확인해야 합니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC에 대한 DNS 속성 보기 및 업데이트를 참조하세요. 인터페이스 엔드포인트에 프라이빗 DNS 기능이 활성화된 경우:

- 기본 리전 DNS 이름을 사용하여 Amazon EKS에 API를 요청할 수 있습니다. 예: `eks.region.amazonaws.com`. API 목록은 Amazon EKS API 참조의 Actions(작업)를 참조하세요.
- EKS API를 호출하는 애플리케이션은 변경할 필요가 없습니다.
- Amazon EKS 기본 서비스 엔드포인트에 적용된 호출은 프라이빗 AWS 네트워크에서 인터페이스 엔드포인트를 통해 자동으로 라우팅됩니다.

워크로드

워크로드는 Kubernetes의 Pods에 배포되는 컨테이너에 배포됩니다. Pod에는 하나 이상의 컨테이너가 포함됩니다. 일반적으로 동일한 서비스를 제공하는 하나 이상의 Pods가 Kubernetes 서비스에 배포됩니다. 동일한 서비스를 제공하는 여러 Pods를 배포한 후에는 다음을 수행할 수 있습니다.

- AWS Management Console을 사용하여 각 클러스터에서 실행 주인 워크로드에 대한 정보 봅니다.
- Kubernetes Vertical Pod Autoscaler를 사용하여 Pods를 수직으로 크기 조정합니다.
- Kubernetes Horizontal Pod Autoscaler를 사용하여 수요에 따라 필요한 수의 Pods를 수평으로 크기 조정합니다.
- 외부(인터넷에 액세스할 수 있는 Pods의 경우) 또는 내부(프라이빗 Pods의 경우) Network Load Balancer를 사용하여 여러 Pods에 걸쳐 네트워크 트래픽의 균형을 맞출 수 있습니다. 로드 밸런서는 OSI 모델의 계층 4에서 트래픽을 라우팅합니다.
- Amazon EKS 애플리케이션 로드 밸런싱를 생성하여 여러 Pods에 걸쳐 애플리케이션 트래픽의 균형을 맞출 수 있습니다. Application Load Balancer는 OSI 모델의 계층 7에서 트래픽을 라우팅합니다.
- Kubernetes를 처음 사용한다면 이 주제를 통해 샘플 애플리케이션 배포하는 방법을 알아볼 수 있습니다.
- `externalIPs`를 사용하여 서비스에 할당할 수 있는 IP 주소 제한할 수 있습니다.

샘플 애플리케이션 배포

이 주제에서는 샘플 애플리케이션을 생성하고 클러스터에 배포합니다.

사전 조건

- 노드가 하나 이상 있는 기존 Kubernetes 클러스터. 기존 Amazon EKS 클러스터가 없는 경우 Amazon EKS 시작하기 가이드 중 하나를 사용하여 Amazon EKS 클러스터를 배포할 수 있습니다. Windows 애플리케이션을 배포하는 경우 클러스터와 하나 이상의 Amazon EC2 Windows 노드에 대해 Windows 지원이 사용 설정되어 있어야 합니다.
- 컴퓨터에 설치된 `Kubectl`. 자세한 내용은 `kubectl` 설치 또는 업데이트 섹션을 참조하세요.
- 클러스터와 통신하도록 구성된 `Kubectl`. 자세한 내용은 Amazon EKS 클러스터용 `kubeconfig` 파일 생성 또는 업데이트 섹션을 참조하세요.
- 샘플 워크로드를 Fargate에 배포하려는 경우 이름을 변경하지 않는 한 이 자습서에서 생성한 것과 동일한 네임스페이스(`eks-sample-app`)를 포함하는 기존 Fargate 프로파일이 있어야 합니다. 시작 안내서 중 하나를 사용하여 클러스터를 생성한 경우 시작 안내서에서 생성된 프로파일이 이 자습

서에서 사용하는 네임스페이스를 지정하지 않기 때문에 새 프로파일을 생성하거나 기존 프로파일에 네임스페이스를 추가해야 합니다. 또한 VPC에 프라이빗 서브넷이 하나 이상 있어야 합니다.

샘플 애플리케이션 배포

다음 단계에서 많은 변수를 변경할 수 있지만 지정된 경우에만 변수 값을 변경하는 것이 좋습니다. Kubernetes Pods, 배포, 서비스를 더 잘 이해하면 다른 값을 변경해 볼 수 있습니다.

1. 네임스페이스를 생성합니다. 네임스페이스를 사용하면 Kubernetes에서 리소스를 그룹화할 수 있습니다. 자세한 내용은 Kubernetes 설명서의 네임스페이스를 참조하세요. 샘플 애플리케이션을 AWS Fargate에 배포하려는 경우 AWS Fargate 프로파일의 `namespace` 값이 `eks-sample-app`인지 확인합니다.

   ```
   kubectl create namespace eks-sample-app
   ```

2. Kubernetes 배포를 생성합니다. 이 샘플 배포는 퍼블릭 리포지토리에서 컨테이너 이미지를 가져와 클러스터에 3개의 복제본(개별 Pods)을 배포합니다. 자세한 내용은 Kubernetes 설명서의 배포를 참조하세요. 애플리케이션을 Linux 또는 Windows 노드에 배포할 수 있습니다. Fargate에 배포하는 경우 Linux 애플리케이션만 배포할 수 있습니다.

   a. 다음 콘텐츠를 `eks-sample-deployment.yaml`이라는 파일에 저장합니다. 샘플 애플리케이션의 컨테이너는 네트워크 스토리지를 사용하지 않지만 필요한 애플리케이션이 있을 수 있습니다. 자세한 내용은 스토리지 섹션을 참조하세요.

      Linux

      `kubernetes.io/arch` 키 아래의 `amd64` 또는 `arm64` `values`는 애플리케이션을 하드웨어 아키텍처 중 하나에 배포할 수 있음을 의미합니다(클러스터에 둘 다 있는 경우). 이는 이 이미지가 다중 아키텍처 이미지이지만 모두가 그렇지는 않기 때문에 가능합니다. 이미지를 가져오는 리포지토리에서 이미지 세부 정보를 확인하여 이미지가 지원되는 하드웨어 아키텍처를 결정할 수 있습니다. 하드웨어 아키텍처 유형을 지원하지 않거나 이미지를 배포하지 않으려는 이미지를 배포하는 경우 매니페스트에서 해당 유형을 제거합니다. 자세한 내용은 Kubernetes 설명서의 잘 알려진 레이블, 주석, 테인트를 참조하세요.

      `kubernetes.io/os: linux` `nodeSelector`는 예를 들어 클러스터에 Linux 및 Windows 노드가 있는 경우 이미지가 Linux 노드에만 배포됨을 의미합니다. 자세한 내용은 Kubernetes 설명서의 잘 알려진 레이블, 주석, 테인트를 참조하세요.

      ```
      apiVersion: apps/v1
      ```

```
kind: Deployment
metadata:
  name: eks-sample-linux-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-linux-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-linux-app
  template:
    metadata:
      labels:
        app: eks-sample-linux-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: kubernetes.io/arch
                operator: In
                values:
                - amd64
                - arm64
      containers:
      - name: nginx
        image: public.ecr.aws/nginx/nginx:1.23
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
      nodeSelector:
        kubernetes.io/os: linux
```

Windows

`kubernetes.io/os: windows` nodeSelector는 예를 들어 클러스터에 Windows 및 Linux 노드가 있는 경우 이미지가 Windows 노드에만 배포됨을 의미합니다. 자세한 내용은 Kubernetes 설명서의 잘 알려진 레이블, 주석, 테인트를 참조하세요.

```
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: eks-sample-windows-deployment
  namespace: eks-sample-app
  labels:
    app: eks-sample-windows-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: eks-sample-windows-app
  template:
    metadata:
      labels:
        app: eks-sample-windows-app
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: beta.kubernetes.io/arch
                operator: In
                values:
                - amd64
      containers:
      - name: windows-server-iis
        image: mcr.microsoft.com/windows/servercore:ltsc2019
        ports:
        - name: http
          containerPort: 80
        imagePullPolicy: IfNotPresent
        command:
        - powershell.exe
        - -command
        - "Add-WindowsFeature Web-Server; Invoke-WebRequest -UseBasicParsing
 -Uri 'https://dotnetbinaries.blob.core.windows.net/servicemonitor/2.0.1.6/
ServiceMonitor.exe' -OutFile 'C:\\ServiceMonitor.exe'; echo
 '<html><body><br/><br/><marquee><H1>Hello EKS!!!<H1><marquee></body><html>'
 > C:\\inetpub\\wwwroot\\default.html; C:\\ServiceMonitor.exe 'w3svc'; "
      nodeSelector:
        kubernetes.io/os: windows
```

b. 배포 매니페스트를 클러스터에 적용합니다.

```
kubectl apply -f eks-sample-deployment.yaml
```

3. 서비스를 생성합니다. 서비스를 사용하면 단일 IP 주소 또는 이름을 통해 모든 복제본에 액세스할 수 있습니다. 자세한 내용은 Kubernetes 설명서의 서비스를 참조하세요. 샘플 애플리케이션에서는 구현되지 않았지만 다른 AWS 서비스와 상호 작용해야 하는 애플리케이션이 있는 경우 Pods에 대한 Kubernetes 서비스 계정을 생성하고 이를 AWS IAM 계정에 연결하는 것이 좋습니다. 서비스 계정을 지정하면 Pods가 다른 서비스와 상호 작용하기 위해 지정한 최소 권한만 갖게 됩니다. 자세한 내용은 서비스 계정에 대한 IAM 역할 섹션을 참조하세요.

   a. 다음 콘텐츠를 `eks-sample-service.yaml`이라는 파일에 저장합니다. Kubernetes는 클러스터 내에서만 액세스할 수 있는 자체 IP 주소를 서비스에 할당합니다. 클러스터 외부에서 서비스에 액세스하려면 AWS Load Balancer Controller 로드 밸런서 애플리케이션 또는 네트워크 트래픽을 서비스에 배포합니다.

      Linux

      ```
      apiVersion: v1
      kind: Service
      metadata:
        name: eks-sample-linux-service
        namespace: eks-sample-app
        labels:
          app: eks-sample-linux-app
      spec:
        selector:
          app: eks-sample-linux-app
        ports:
          - protocol: TCP
            port: 80
            targetPort: 80
      ```

      Windows

      ```
      apiVersion: v1
      kind: Service
      metadata:
        name: eks-sample-windows-service
        namespace: eks-sample-app
        labels:
          app: eks-sample-windows-app
      ```

```
spec:
  selector:
    app: eks-sample-windows-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

b. 서비스 매니페스트를 클러스터에 적용합니다.

```
kubectl apply -f eks-sample-service.yaml
```

4. eks-sample-app 네임스페이스에 있는 모든 리소스를 봅니다.

```
kubectl get all -n eks-sample-app
```

예제 출력은 다음과 같습니다.

Windows 리소스를 배포한 경우 다음 출력에서 모든 *linux* 인스턴스는 windows입니다. 다른 ### 은 출력과 다를 수 있습니다.

```
NAME                                                 READY   STATUS    RESTARTS   AGE
pod/eks-sample-linux-deployment-65b7669776-m6qxz     1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-mmxvd     1/1     Running   0          27m
pod/eks-sample-linux-deployment-65b7669776-qzn22     1/1     Running   0          27m

NAME                                 TYPE         CLUSTER-IP      EXTERNAL-IP
 PORT(S)   AGE
service/eks-sample-linux-service     ClusterIP    10.100.74.8     <none>        80/
TCP    32m

NAME                                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/eks-sample-linux-deployment    3/3     3            3           27m

NAME                                                      DESIRED   CURRENT   READY
   AGE
replicaset.apps/eks-sample-linux-deployment-776d8f8fd8    3         3         3
   27m
```

이 출력에는 이전 단계에서 배포된 샘플 매니페스트에 지정된 서비스 및 배포가 표시됩니다. 3개의 Pods도 확인할 수 있습니다. 동일한 매니페스트에 3 replicas가 지정되었기 때문입

니다. Pods에 대한 자세한 내용은 Kubernetes 설명서의 포드를 참조하세요. Kubernetes는 샘플 매니페스트에서 지정되지 않은 경우에도 자동으로 `replicaset` 리소스를 생성합니다. `ReplicaSets`에 대한 자세한 내용은 Kubernetes 설명서의 ReplicaSet를 참조하세요.

> **Note**
>
> Kubernetes는 매니페스트에 지정된 복제본의 수를 유지합니다. 프로덕션 배포이고 Kubernetes가 복제본 수를 수평적 또는 수직적으로 조정하여 Pods의 컴퓨팅 리소스를 조정하도록 하려는 경우 Horizontal Pod Autoscaler 및 Vertical Pod Autoscaler를 사용합니다.

5. 배포된 서비스의 세부 정보를 확인합니다. Windows 서비스를 배포한 경우 *`linux`*를 **`windows`**로 바꿉니다.

```
kubectl -n eks-sample-app describe service eks-sample-linux-service
```

예제 출력은 다음과 같습니다.

Windows 리소스를 배포한 경우 다음 출력에서 모든 *`linux`* 인스턴스는 `windows`입니다. 다른 *`###`*은 출력과 다를 수 있습니다.

```
Name:              eks-sample-linux-service
Namespace:         eks-sample-app
Labels:            app=eks-sample-linux-app
Annotations:       <none>
Selector:          app=eks-sample-linux-app
Type:              ClusterIP
IP Families:       <none>
IP:                10.100.74.8
IPs:               10.100.74.8
Port:              <unset>  80/TCP
TargetPort:        80/TCP
Endpoints:         192.168.24.212:80,192.168.50.185:80,192.168.63.93:80
Session Affinity:  None
Events:            <none>
```

이전 출력에서 `IP:`의 값은 클러스터 내의 모든 노드 또는 Pod에서 도달할 수 있는 고유한 IP 주소이지만 클러스터 외부에서는 도달할 수 없습니다. `Endpoints`의 값은 VPC 내에서 서비스의 일부인 Pods에 할당된 IP 주소입니다.

6. 이전 단계에서 네임스페이스를 볼 때 출력에 나열된 Pods 중 하나의 세부 정보를 봅니다. Windows 앱을 배포한 경우 *linux*를 **windows**로 바꾸고 *776d8f8fd8-78w66*을 Pods 중 하나에 대해 반환된 값으로 바꿉니다.

```
kubectl -n eks-sample-app describe pod eks-sample-linux-deployment-65b7669776-m6qxz
```

간략한 출력

Windows 리소스를 배포한 경우 다음 출력에서 모든 *linux* 인스턴스는 windows입니다. 다른 *example values*는 출력과 다를 수 있습니다.

```
Name:         eks-sample-linux-deployment-65b7669776-m6qxz
Namespace:    eks-sample-app
Priority:     0
Node:         ip-192-168-45-132.us-west-2.compute.internal/192.168.45.132
[...]
IP:           192.168.63.93
IPs:
  IP:           192.168.63.93
Controlled By:  ReplicaSet/eks-sample-linux-deployment-65b7669776
[...]
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
[...]
Events:
  Type    Reason     Age    From
 Message
  ----    ------     ----   ----
 -------
  Normal  Scheduled  3m20s  default-scheduler
 Successfully assigned eks-sample-app/eks-sample-linux-deployment-65b7669776-m6qxz
 to ip-192-168-45-132.us-west-2.compute.internal
[...]
```

이전 출력에서 IP:의 값은 노드가 있는 서브넷에 할당된 CIDR 블록에서 Pod에 할당되는 고유한 IP입니다. Pods에 다른 CIDR 블록의 IP 주소를 할당하려면 기본 동작을 변경할 수 있습니다. 자세

한 내용은 포드에 대한 사용자 지정 네트워킹 섹션을 참조하세요. Kubernetes 스케줄러가 IP 주소 *192.168.45.132*를 사용하여 Node의 Pod를 예약했는지 확인할 수도 있습니다.

> **Tip**
>
> 명령줄을 사용하는 대신 AWS Management Console에서 Pods, 서비스, 배포, 기타 Kubernetes 리소스에 대한 많은 세부 정보를 볼 수 있습니다. 자세한 내용은 Kubernetes 리소스 보기 섹션을 참조하세요.

7. 이전 단계에서 설명한 Pod에서 셸을 실행하여 *65b7669776-m6qxz*를 Pods 중 하나의 ID로 바꿉니다.

   Linux

   ```
   kubectl exec -it eks-sample-linux-deployment-65b7669776-m6qxz -n eks-sample-app
    -- /bin/bash
   ```

   Windows

   ```
   kubectl exec -it eks-sample-windows-deployment-65b7669776-m6qxz -n eks-sample-
   app -- powershell.exe
   ```

8. Pod 셸에서 이전 단계에서 배포와 함께 설치된 웹 서버의 출력을 봅니다. 서비스 이름만 지정하면 됩니다. 기본적으로 Amazon EKS 클러스터와 함께 배포되는 CoreDNS에 의해 서비스의 IP 주소로 확인됩니다.

   Linux

   ```
   curl eks-sample-linux-service
   ```

   예제 출력은 다음과 같습니다.

   ```
   <!DOCTYPE html>
   <html>
   <head>
   <title>Welcome to nginx!</title>
   [...]
   ```

Windows

```
Invoke-WebRequest -uri eks-sample-windows-service/default.html -UseBasicParsing
```

예제 출력은 다음과 같습니다.

```
StatusCode        : 200
StatusDescription : OK
Content           : < h t m l > < b o d y > < b r / > < b r / > < m a r q u e e
 > < H 1 > H e l l o
                        E K S ! ! ! < H 1 > < m a r q u e e > < / b o d y > < h t
 m l >
```

9. Pod 셸에서 Pod의 DNS 서버를 봅니다.

Linux

```
cat /etc/resolv.conf
```

예제 출력은 다음과 같습니다.

```
nameserver 10.100.0.10
search eks-sample-app.svc.cluster.local svc.cluster.local cluster.local us-
west-2.compute.internal
options ndots:5
```

이전 출력에서 `10.100.0.10`은 클러스터에 배포된 모든 Pods의 `nameserver`로 자동 할당됩니다.

Windows

```
Get-NetIPConfiguration
```

간략한 출력

```
InterfaceAlias       : vEthernet
[...]
IPv4Address          : 192.168.63.14
[...]
```

```
DNSServer            : 10.100.0.10
```

이전 출력에서 `10.100.0.10`은 클러스터에 배포된 모든 Pods의 DNS 서버로 자동 할당됩니다.

10. `exit`를 입력하여 Pod에서 연결을 해제합니다.
11. 샘플 애플리케이션 사용이 끝나면 다음 명령을 사용하여 샘플 네임스페이스, 서비스 및 배포를 제거할 수 있습니다.

```
kubectl delete namespace eks-sample-app
```

다음 단계

샘플 애플리케이션을 배포한 후 다음 연습 중 일부를 시도해 볼 수 있습니다.

- the section called “애플리케이션 로드 밸런싱”
- the section called “네트워크 로드 밸런싱”

Vertical Pod Autoscaler

Kubernetes Vertical Pod Autoscaler는 Pods에 대한 CPU 및 메모리 예약을 자동으로 조정하여 애플리케이션의 ‘크기를 적절히 조정’할 수 있게 지원합니다. 이러한 조정을 통해 클러스터 리소스 사용률을 개선하고 다른 Pods를 위한 CPU 및 메모리를 확보할 수 있습니다. 이 주제에서는 Vertical Pod Autoscaler를 클러스터에 배포하고 제대로 작동하는지 확인하는 방법을 안내합니다.

사전 조건

- 기존 Amazon EKS 클러스터가 있습니다. 그렇지 않은 경우 Amazon EKS 시작하기 섹션을 참조하세요.
- Kubernetes 지표 서버가 설치되어 있습니다. 자세한 내용은 Kubernetes 지표 서버 설치 섹션을 참조하세요.
- Amazon EKS 클러스터와 통신하도록 구성된 `kubectl` 클라이언트를 사용 중입니다.
- OpenSSL `1.1.1` 이상이 디바이스에 설치되어 있습니다.

Vertical Pod Autoscaler 배포

이 섹션에서는 클러스터에 Vertical Pod Autoscaler를 배포합니다.

Vertical Pod Autoscaler를 배포하려면

1. 터미널 창을 열고 최신 Vertical Pod Autoscaler 소스 코드를 다운로드할 디렉터리로 이동합니다.
2. kubernetes/autoscaler GitHub 리포지토리를 복제합니다.

```
git clone https://github.com/kubernetes/autoscaler.git
```

3. 디렉터리를 `vertical-pod-autoscaler`로 변경합니다.

```
cd autoscaler/vertical-pod-autoscaler/
```

4. (선택 사항) 다른 버전의 Vertical Pod Autoscaler를 이미 배포했다면 다음 명령을 사용하여 제거합니다.

```
./hack/vpa-down.sh
```

5. 노드에 `registry.k8s.io` 컨테이너 레지스트리에 대한 노드에 인터넷 액세스 권한이 없는 경우, 다음 이미지를 가져와 자체 개인 리포지토리로 푸시해야 합니다. 이미지 가져오기, 자체 개인 리포지토리로 푸시하는 방법에 대한 자세한 내용은 한 리포지토리에서 다른 리포지토리로 컨테이너 이미지 복사 단원을 참조하세요.

```
registry.k8s.io/autoscaling/vpa-admission-controller:0.10.0
registry.k8s.io/autoscaling/vpa-recommender:0.10.0
registry.k8s.io/autoscaling/vpa-updater:0.10.0
```

이미지를 개인 Amazon ECR 리포지토리에 푸시하는 경우 매니페스트의 `registry.k8s.io`를 레지스트리로 바꿉니다. *111122223333*을 계정 ID로 바꿉니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다. 다음 명령은 해당 리포지토리의 이름이 매니페스트의 리포지토리 이름과 같다고 가정합니다. 리포지토리의 이름을 다르게 지정하면 리포지토리도 변경해야 합니다.

```
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-
code.amazonaws.com/' ./deploy/admission-controller-deployment.yaml
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-
code.amazonaws.com/' ./deploy/recommender-deployment.yaml
```

```
sed -i.bak -e 's/registry.k8s.io/111122223333.dkr.ecr.region-
code.amazonaws.com/' ./deploy/updater-deployment.yaml
```

6. 다음 명령을 사용하여 클러스터에 Vertical Pod Autoscaler를 배포합니다.

```
./hack/vpa-up.sh
```

7. Vertical Pod Autoscaler Pods가 성공적으로 생성되었는지 확인합니다.

```
kubectl get pods -n kube-system
```

예제 출력은 다음과 같습니다.

```
NAME                                        READY   STATUS    RESTARTS   AGE
[...]
metrics-server-8459fc497-kfj8w              1/1     Running   0          83m
vpa-admission-controller-68c748777d-ppspd   1/1     Running   0          7s
vpa-recommender-6fc8c67d85-gljpl            1/1     Running   0          8s
vpa-updater-786b96955c-bgp9d                1/1     Running   0          8s
```

Vertical Pod Autoscaler 설치 테스트

이 섹션에서는 샘플 애플리케이션을 배포하여 Vertical Pod Autoscaler가 작동 중인지 확인합니다.

Vertical Pod Autoscaler 설치를 테스트하려면

1. 다음 명령을 사용하여 예시 `hamster.yaml` Vertical Pod Autoscaler를 배포합니다.

```
kubectl apply -f examples/hamster.yaml
```

2. `hamster` 예제 애플리케이션에서 Pods를 가져옵니다.

```
kubectl get pods -l app=hamster
```

예제 출력은 다음과 같습니다.

```
hamster-c7d89d6db-rglf5   1/1     Running   0          48s
hamster-c7d89d6db-znvz5   1/1     Running   0          48s
```

3. 여러 Pods 중 하나를 설명하여 cpu 및 memory 예약을 확인합니다. *c7d89d6db-rglf5*를 이전 단계의 출력에서 반환된 ID 중 하나로 바꿉니다.

```
kubectl describe pod hamster-c7d89d6db-rglf5
```

예제 출력은 다음과 같습니다.

```
[...]
Containers:
  hamster:
    Container ID:  docker://
e76c2413fc720ac395c33b64588c82094fc8e5d590e373d5f818f3978f577e24
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:35:16 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        100m
      memory:     50Mi
[...]
```

원래 Pod에서는 100millicpu의 CPU와 50메비바이트의 메모리를 예약한 것을 알 수 있습니다. 이 예시 애플리케이션의 경우 100millicpu는 Pod가 실행해야 하는 것보다 적으므로 CPU가 제한되어 있습니다. 또한 필요한 메모리보다 훨씬 적은 메모리를 예약합니다. Vertical Pod Autoscaler vpa-recommender 배포에서는 hamster Pods를 분석하여 CPU 및 메모리 요구 사항이 적절한지 확인합니다. 조정이 필요하면 vpa-updater는 업데이트된 값으로 Pods를 다시 시작합니다.

4. vpa-updater가 새로운 hamster Pod를 시작할 때까지 기다립니다. 1~2분 정도 걸립니다. 다음 명령을 사용하여 Pods를 모니터링할 수 있습니다.

> **Note**
>
> 새 Pod가 시작되었는지 알 수 없다면 Pod 이름을 이전 목록과 비교하십시오. 새 Pod가 시작되면 새 Pod 이름이 표시됩니다.

```
kubectl get --watch Pods -l app=hamster
```

5. 새 hamster Pod가 시작되면 포드를 설명하여 업데이트된 CPU 및 메모리 예약을 확인합니다.

```
kubectl describe pod hamster-c7d89d6db-jxgfv
```

예제 출력은 다음과 같습니다.

```
[...]
Containers:
  hamster:
    Container ID:
 docker://2c3e7b6fb7ce0d8c86444334df654af6fb3fc88aad4c5d710eac3b1e7c58f7db
    Image:         registry.k8s.io/ubuntu-slim:0.1
    Image ID:      docker-pullable://registry.k8s.io/ubuntu-
slim@sha256:b6f8c3885f5880a4f1a7cf717c07242eb4858fdd5a84b5ffe35b1cf680ea17b1
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
    Args:
      -c
      while true; do timeout 0.5s yes >/dev/null; sleep 0.5s; done
    State:          Running
      Started:      Fri, 27 Sep 2019 10:37:08 -0700
    Ready:          True
    Restart Count:  0
    Requests:
      cpu:        587m
      memory:     262144k
[...]
```

이전 출력에서, cpu 예약이 원래 값의 5배가 넘는 587밀리cpu로 증가했음을 알 수 있습니다. memory은 원래 값의 5배에 달하는 262,144킬로바이트(약 250메비바이트)로 증가했습니다. 이

Pod는 리소스가 부족하므로 Vertical Pod Autoscaler는 추정치를 훨씬 더 적절한 값으로 수정하였습니다.

6. hamster-vpa 리소스를 설명하여 새로운 권장 사항을 확인하십시오.

```
kubectl describe vpa/hamster-vpa
```

예제 출력은 다음과 같습니다.

```
Name:         hamster-vpa
Namespace:    default
Labels:       <none>
Annotations:  kubectl.kubernetes.io/last-applied-configuration:
                {"apiVersion":"autoscaling.k8s.io/
v1beta2","kind":"VerticalPodAutoscaler","metadata":{"annotations":
{},"name":"hamster-vpa","namespace":"d...
API Version:  autoscaling.k8s.io/v1beta2
Kind:         VerticalPodAutoscaler
Metadata:
  Creation Timestamp:  2019-09-27T18:22:51Z
  Generation:          23
  Resource Version:    14411
  Self Link:           /apis/autoscaling.k8s.io/v1beta2/namespaces/default/
verticalpodautoscalers/hamster-vpa
  UID:                 d0d85fb9-e153-11e9-ae53-0205785d75b0
Spec:
  Target Ref:
    API Version:  apps/v1
    Kind:         Deployment
    Name:         hamster
Status:
  Conditions:
    Last Transition Time:  2019-09-27T18:23:28Z
    Status:                True
    Type:                  RecommendationProvided
  Recommendation:
    Container Recommendations:
      Container Name:  hamster
      Lower Bound:
        Cpu:     550m
        Memory:  262144k
      Target:
        Cpu:     587m
```

```
        Memory:  262144k
      Uncapped Target:
        Cpu:     587m
        Memory:  262144k
      Upper Bound:
        Cpu:     21147m
        Memory:  387863636
Events:          <none>
```

7. 샘플 애플리케이션에 대한 실험이 끝나면 다음 명령으로 이를 삭제할 수 있습니다.

```
kubectl delete -f examples/hamster.yaml
```

Horizontal Pod Autoscaler

Kubernetes Horizontal Pod Autoscaler는 배포, 복제 컨트롤러 또는 복제본 집합에 있는 Pods의 수를 해당 리소스의 CPU 사용률에 따라 자동으로 조정합니다. 이를 통해 애플리케이션은 증가된 수요를 충족하기 위해 규모를 확장하거나 리소스가 필요 없을 때 규모를 축소할 수 있어 다른 애플리케이션을 위한 노드를 확보할 수 있습니다. 목표 CPU 사용률(백분율)을 설정하면 Horizontal Pod Autoscaler가 이 목표를 충족하기 위해 애플리케이션을 축소 또는 확장합니다.

Horizontal Pod Autoscaler는 Kubernetes의 표준 API 리소스로서, 지표 소스(예: Kubernetes 지표 서버)를 작업할 Amazon EKS 클러스터에 설치하기만 하면 됩니다. 애플리케이션 규모를 조정하기 위해 Horizontal Pod Autoscaler를 클러스터에 배포하거나 설치할 필요가 없습니다. 자세한 내용은 Kubernetes 설명서의 Horizontal Pod Autoscaler를 참조하세요.

이 주제를 통해 Amazon EKS 클러스터용 Horizontal Pod Autoscaler를 준비하고 샘플 애플리케이션에서 작동 중인지 확인하세요.

> **Note**
>
> 이 주제는 Kubernetes 설명서의 Horizontal Pod autoscaler 연습에 기반을 두고 있습니다.

사전 조건

- 기존 Amazon EKS 클러스터가 있습니다. 그렇지 않은 경우 Amazon EKS 시작하기 섹션을 참조하세요.

- Kubernetes 지표 서버가 설치되어 있습니다. 자세한 내용은 Kubernetes 지표 서버 설치 섹션을 참조하세요.
- Amazon EKS 클러스터와 통신하도록 구성된 `kubectl` 클라이언트를 사용 중입니다.

Horizontal Pod Autoscaler 테스트 애플리케이션 실행

이 섹션에서는 샘플 애플리케이션을 실행하여 Horizontal Pod Autoscaler가 작동 중인지 확인합니다.

> **Note**
>
> 이 예제는 Kubernetes 설명서의 Horizontal Pod 자동 스케일러 연습에 기반을 두고 있습니다.

Horizontal Pod Autoscaler 설치를 테스트하려면

1. 다음 명령을 사용하여 간단한 Apache 웹 서버 애플리케이션을 배포합니다.

   ```
   kubectl apply -f https://k8s.io/examples/application/php-apache.yaml
   ```

   이 Apache 웹 서버 Pod에는 500 millicpu CPU 제한이 지정되며 포트 80에서 제공됩니다.
2. `php-apache` 배포를 위해 Horizontal Pod Autoscaler 리소스를 생성합니다.

   ```
   kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10
   ```

   이 명령을 통해 최소 1개의 Pod에서 최대 10개의 Pods로 배포에 대해 50퍼센트의 CPU 사용률을 달성하려는 자동 스케일러가 생성됩니다. 평균 CPU 로드가 50퍼센트 이하인 경우 자동 스케일러는 배포 시 Pods의 수를 최소 1개로 줄이려고 합니다. 로드가 50퍼센트보다 큰 경우 자동 스케일러는 배포 시 Pods의 수를 최대 10개로 늘이려고 합니다. 자세한 내용은 Kubernetes 설명서의 HorizontalPodAutoscaler의 작동 방식을 참조하세요.
3. 자동 조정기의 세부 정보를 볼 수 있는 다음 명령을 사용하여 자동 조정기를 설명하십시오.

   ```
   kubectl get hpa
   ```

   예제 출력은 다음과 같습니다.

   ```
   NAME         REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
   php-apache   Deployment/php-apache   0%/50%    1         10        1          51s
   ```

보시다시피 서버에 아직 로드가 없으므로 현재 CPU 로드가 0%입니다. Pod 수가 이미 가장 낮은 경계(1)에 있으므로 스케일 인할 수 없습니다.

4. 컨테이너를 실행하여 웹 서버의 로드를 생성합니다.

```
kubectl run -i \
    --tty load-generator \
    --rm --image=busybox \
    --restart=Never \
    -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"
```

5. 배포 확장을 보려면 이전 단계를 실행한 터미널과는 별도의 터미널에서 다음 명령을 주기적으로 실행합니다.

```
kubectl get hpa php-apache
```

예제 출력은 다음과 같습니다.

```
NAME         REFERENCE                TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache   250%/50%   1         10        5
 4m44s
```

복제본 수가 증가하는 데 1분 이상 걸릴 수 있습니다. 실제 CPU 백분율이 목표 백분율보다 높으면 복제본 수가 최대 10까지 증가합니다. 이 경우 해당 백분율이 250%이므로 REPLICAS 수가 계속 증가합니다.

> **Note**
>
> 복제본 수가 최대값에 도달하기까지 몇 분 정도 걸릴 수 있습니다. 예를 들어 CPU 로드가 50% 이하로 유지되는 데 복제본이 6개만 필요한 경우 로드가 6개 복제본을 초과해서 확장되지 않습니다.

6. 로드를 중지합니다. 로드를 생성 중인 터미널 창에서 Ctrl+C 키를 누른 채 로드를 중지합니다. 축소가 관찰되는 터미널에서 다음 명령을 다시 실행하여 복제본이 1로 축소되는 것을 확인할 수 있습니다.

```
kubectl get hpa
```

예제 출력은 다음과 같습니다.

```
NAME         REFERENCE                  TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache      0%/50%    1         10        1          25m
```

> **Note**
>
> 다시 축소하기 위한 기본 기간은 5분이므로 현재 CPU 백분율이 0%인 경우에도 복제본 수가 다시 1에 도달하기까지 시간이 걸릴 수 있습니다. 시간 프레임은 수정할 수 있습니다. 자세한 내용은 Kubernetes 설명서의 Horizontal Pod Autoscaler를 참조하세요.

7. 샘플 애플리케이션에 대한 실험이 끝났으면 php-apache 리소스를 삭제하십시오.

```
kubectl delete deployment.apps/php-apache service/php-apache
 horizontalpodautoscaler.autoscaling/php-apache
```

Amazon EKS의 네트워크 로드 밸런싱

네트워크 트래픽은 OSI 모델의 L4에서 로드 밸런싱됩니다. L7에서 애플리케이션 트래픽을 로드 밸런싱하려면 Kubernetes `ingress`를 배포합니다. 이는 AWS Application Load Balancer를 프로비저닝합니다. 자세한 내용은 Amazon EKS 애플리케이션 로드 밸런싱 단원을 참조하십시오. 두 가지 로드 밸런싱 간의 차이점을 자세히 알아보려면 AWS 웹 사이트에서 Elastic 로드 밸런싱 기능 부분을 참조하세요.

Kubernetes 유형의 `Service` `LoadBalancer`를 생성하면 AWS 클라우드 공급자 로드 밸런서 컨트롤러가 기본적으로 AWS Classic 로드 밸런서를 생성하지만 AWS Network Load Balancer도 생성할 수 있습니다. 이 컨트롤러는 향후 중요한 버그 수정만 수신하고 있습니다. AWS 클라우드 공급자 로드 밸런서 사용에 대한 자세한 내용은 Kubernetes 설명서의 AWS 클라우드 공급자 로드 밸런서 컨트롤러를 참조하세요. 이 주제에서는 이에 대해 다루지 않습니다.

AWS 클라우드 공급자 로드 밸런서 컨트롤러 대신 AWS Load Balancer Controller의 버전 `2.7.2` 이상을 사용하는 것이 좋습니다. AWS Load Balancer Controller는 AWS Network Load Balancer를 생성하지만 AWS Classic 로드 밸런서는 생성하지 않습니다. 이 주제의 나머지 부분은 AWS 로드 밸런서 컨트롤러 사용에 대한 것입니다.

AWS Network Load Balancer는 Amazon EC2 AWS Fargate IP 및 인스턴스 대상 또는 Pods IP 대상에 배포된 네트워크 트래픽을 로드 밸런싱할 수 있습니다. 자세한 내용은 GitHub의 AWS 로드 밸런서 컨트롤러를 참조하세요.

필수 조건

AWS Load Balancer Controller를 사용하여 네트워크 트래픽을 로드 밸런싱하려면 먼저 다음 요구 사항을 충족해야 합니다.

- 기존 클러스터를 보유해야 합니다. 기존 클러스터가 없는 경우 Amazon EKS 시작하기 부분을 참조하세요. 기존 클러스터의 버전을 업데이트해야 하는 경우에는 Amazon EKS 클러스터 Kubernetes 버전 업데이트 부분을 참조하세요.
- 클러스터에 배포된 AWS Load Balancer Controller가 있습니다. 자세한 내용은 AWS Load Balancer Controller란 무엇인가요? 단원을 참조하십시오. 버전 `2.7.2` 이상을 사용하는 것이 좋습니다.
- 하나 이상의 서브넷. 가용 영역에서 태그가 지정된 서브넷이 여러 개 있는 경우 컨트롤러는 서브넷 ID가 사전 순으로 가장 먼저 표시되는 서브넷을 선택합니다. 또한 서브넷에는 최소 8개의 사용 가능한 IP 주소가 있어야 합니다.
- AWS Load Balancer Controller 버전 `2.1.1` 이하를 사용하는 경우 서브넷에 다음과 같이 태깅되어야 합니다. 버전 `2.1.2` 이상을 사용하는 경우 이 태그는 선택 사항입니다. 동일한 VPC에서 여러 클러스터를 실행 중이거나 AWS 서비스가 VPC에서 서브넷을 공유하는 경우 로드 밸런서가 클러스터별로 프로비저닝되는 위치를 보다 효과적으로 제어하려면 서브넷에 태깅할 수 있습니다. 서브넷 ID를 서비스 객체에 대한 주석으로 명시적으로 지정하면 Kubernetes 및 AWS Load Balancer Controller는 이러한 서브넷을 직접 사용하여 로드 밸런서를 생성합니다. 로드 밸런서를 프로비저닝하는 데 이 방법을 사용하도록 선택한 경우에는 서브넷 태깅이 필요하지 않으며 다음과 같은 프라이빗 및 퍼블릭 서브넷 태깅 요구 사항을 건너뛸 수 있습니다. *`my-cluster`*을 클러스터 이름으로 교체합니다.
  - 키 - `kubernetes.io/cluster/`*`my-cluster`*
  - 값 - `shared` 또는 `owned`
- 서비스 또는 수신 객체에 대한 주석으로 서브넷 ID를 명시적으로 지정하지 않는 한 퍼블릭 및 프라이빗 서브넷은 다음 요구 사항을 충족해야 합니다. 서브넷 ID를 서비스 또는 수신 객체에 대한 주석으로 명시적으로 지정하여 로드 밸런서를 프로비저닝하면 Kubernetes 및 AWS Load Balancer Controller는 이러한 서브넷을 직접 사용하여 로드 밸런서를 생성하며 다음 태그가 필요하지 않습니다.
  - 프라이빗 서브넷(Private subnets) - 다음 형식으로 태그를 지정해야 합니다. 이렇게 하면 Kubernetes 및 AWS 로드 밸런서 컨트롤러가 서브넷을 내부 로드 밸런서에 사용할 수 있음을 알 수 있습니다. 2020년 3월 26일 이후에 `eksctl` 또는 Amazon EKS AWS AWS CloudFormation 템플릿을 사용하여 VPC 생성하는 경우 서브넷은 생성될 때 적절하게 태깅됩니다. Amazon EKS AWS AWS CloudFormation VPC 템플릿에 대한 자세한 내용은 Amazon EKS 클러스터에 대해 VPC 생성를 참조하세요.
    - 키 - `kubernetes.io/role/internal-elb`

  - 값 - 1
- 퍼블릭 서브넷(Public subnets) - 다음 형식으로 태그를 지정해야 합니다. 이렇게 하면 Kubernetes가 각 가용 영역에서 퍼블릭 서브넷을 선택하는 대신 외부 로드 밸런서에 대해 이러한 서브넷만 사용하는 것을 알 수 있습니다(서브넷 ID를 기준으로 사전순으로). 2020년 3월 26일 이후에 `eksctl` 또는 Amazon EKS AWS CloudFormation 템플릿을 사용하여 VPC 생성하는 경우 서브넷은 생성될 때 적절하게 태깅됩니다. Amazon EKS AWS CloudFormation VPC 템플릿에 대한 자세한 내용은 Amazon EKS 클러스터에 대해 VPC 생성를 참조하세요.
  - 키 - `kubernetes.io/role/elb`
  - 값 - 1

  서브넷 역할 태그가 명시적으로 추가되지 않은 경우 Kubernetes 서비스 컨트롤러는 클러스터 VPC 서브넷의 라우팅 테이블을 검사하여 서브넷이 프라이빗 또는 퍼블릭 서브넷인지 확인합니다. 하지만 프라이빗 또는 퍼블릭 역할 태그를 명시적으로 추가하는 것을 권장합니다. AWS Load Balancer Controller는 라우팅 테이블을 검사하지 않으며 성공적인 자동 검색을 위해서는 프라이빗 및 퍼블릭 태그가 있어야 합니다.

고려 사항

- 로드 밸런서의 구성은 서비스의 매니페스트에 추가된 주석에 의해 제어됩니다. AWS Load Balancer Controller를 사용할 때와 AWS 클라우드 공급자 로드 밸런서 컨트롤러를 사용할 때의 서비스 주석이 다릅니다. 서비스를 배포하기 전에 AWS Load Balancer Controller에 대한 주석을 검토해야 합니다.
- Amazon VPC CNI plugin for Kubernetes를 사용하는 경우 AWS Load Balancer Controller는 Amazon EC2 IP 또는 인스턴스 대상과 Fargate IP 대상으로 로드 밸런싱할 수 있습니다. 호환 가능한 대체 CNI 플러그 인을 사용할 때 컨트롤러는 인스턴스 대상으로만 로드 밸런싱을 수행할 수 있습니다. Network Load Balancer 대상 유형에 대한 자세한 내용은 Network Load Balancer 사용 설명서에서 대상 유형을 참조하세요.
- 로드 밸런서가 생성될 때 또는 그 이후에 로드 밸런서에 태그를 추가하려면 서비스 사양에 다음 주석을 추가합니다. 자세한 내용을 알아보려면 AWS Load Balancer Controller 설명서의 AWS 리소스 태그를 참조하세요.

  ```
  service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags
  ```

- 다음 주석을 추가하여 Network Load Balancer에 Elastic IP 주소를 할당할 수 있습니다. *`example values`*를 Elastic IP 주소의 `Allocation IDs`로 바꿉니다. `Allocation IDs` 수는 로드 밸런서에 사용되는 서브넷 수와 일치해야 합니다. 자세한 내용은 AWS Load Balancer Controller 설명서를 참조하세요.

```
service.beta.kubernetes.io/aws-load-balancer-eip-allocations:
 eipalloc-xxxxxxxxxxxxxxxxx,eipalloc-yyyyyyyyyyyyyyyyy
```

- Amazon EKS는 클라이언트 트래픽에 대한 노드의 보안 그룹에 인바운드 규칙 하나를 추가하고, 생성하는 각 Network Load Balancer의 상태 확인을 위해 VPC에 각 로드 밸런서 서브넷에 대한 규칙 하나를 추가합니다. `LoadBalancer` 유형의 서비스 배포는 Amazon EKS가 보안 그룹에 허용된 최대 규칙 수에 대한 할당량을 초과하는 규칙을 생성하려고 하면 실패할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 Amazon VPC 할당량에 있는 보안 그룹을 참조하세요. 보안 그룹에 대한 최대 규칙 수를 초과할 가능성을 최소화하려면 다음 옵션을 고려합니다.
  - 보안 그룹 할당량당 규칙 증가를 요청합니다. 자세한 내용은 Service Quotas 사용 설명서에서 할당량 증가 요청을 참조하세요.
  - 인스턴스 대상이 아닌 IP 대상을 사용합니다. IP 대상을 사용하면 동일한 대상 포트에 대해 규칙을 공유할 수 있습니다. 로드 밸런서 서브넷은 주석을 사용하여 수동으로 지정할 수 있습니다. 자세한 내용은 GitHub에서 주석을 참조하세요.
  - `LoadBalancer` 유형의 서비스 대신 수신을 사용하여 서비스에 트래픽을 전송할 수 있습니다. AWS Application Load Balancer에는 Network Load Balancer보다 적은 수의 규칙이 필요합니다. 여러 수신 간에 ALB를 공유할 수 있습니다. 자세한 내용은 Amazon EKS 애플리케이션 로드 밸런싱 단원을 참조하십시오. 여러 서비스에서 Network Load Balancer를 공유할 수 없습니다.
  - 클러스터를 여러 계정에 배포합니다.
- Pods가 Amazon EKS 클러스터의 Windows에서 실행되는 경우 로드 밸런서가 있는 단일 서비스는 최대 1,024개의 백엔드 Pods를 지원할 수 있습니다. 각 Pod에는 고유한 IP 주소가 있습니다.
- AWS Load Balancer Controller를 사용하여 새 Network Load Balancer만 생성하는 것이 좋습니다. AWS 클라우드 공급자 로드 밸런서 컨트롤러로 생성된 기존 Network Load Balancer를 교체하려고 하면 여러 Network Load Balancer가 발생하여 애플리케이션 가동 중지를 유발할 수 있습니다.

Network Load Balancer 생성

IP 또는 인스턴스 대상을 사용하여 Network Load Balancer를 생성할 수 있습니다.

IP targets

Amazon EC2 노드 또는 Fargate에 배포된 Pods와 함께 IP 대상을 사용할 수 있습니다. Kubernetes 서비스는 `LoadBalancer` 유형으로 생성되어야 합니다. 자세한 내용은 Kubernetes 설명서에서 LoadBalancer 유형을 참조하세요.

IP 대상을 사용하는 로드 밸런서를 생성하려면 서비스 매니페스트에 다음 주석을 추가하고 서비스를 배포합니다. `aws-load-balancer-type`의 `external` 값은 AWS 클라우드 공급자 로드 밸런서 컨트롤러가 아닌 AWS Load Balancer Controller가 Network Load Balancer를 생성하도록 하는 원인입니다. 주석이 있는 샘플 서비스 매니페스트를 볼 수 있습니다.

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "ip"
```

> **Note**
>
> IPv6 Pods에 로드 밸런싱하는 경우 다음 주석을 추가합니다. IPv6를 통해서는 인스턴스 대상이 아닌 IP 대상으로만 로드 밸런싱할 수 있습니다. 이 주석이 없으면 로드 밸런싱은 IPv4를 통해 이루어집니다.
>
> ```
> service.beta.kubernetes.io/aws-load-balancer-ip-address-type: dualstack
> ```

Network Load Balancer는 기본적으로 `internal` `aws-load-balancer-scheme`으로 생성됩니다. 클러스터를 생성했을 때 지정되지 않았던 서브넷을 포함하여 클러스터 VPC의 모든 서브넷에서 Network Load Balancer를 시작할 수 있습니다.

Kubernetes는 서브넷의 라우팅 테이블을 검사하여 퍼블릭인지 또는 프라이빗인지 판단합니다. 퍼블릭 서브넷에는 인터넷 게이트웨이를 사용해 인터넷으로 직접 연결되는 경로가 있지만 프라이빗 서브넷은 그렇지 않습니다.

Amazon EC2 노드로 로드 밸런싱하기 위해 퍼블릭 서브넷에 Network Load Balancer를 생성하려는 경우(Fargate는 프라이빗만 가능) 다음 주석과 함께 `internet-facing`을 지정합니다.

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Note**
>
> 이전 버전과의 호환성을 위해 `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` 주석이 여전히 지원됩니다. 그러나 `service.beta.kubernetes.io/aws-load-balancer-type: "nlb-ip"` 대신 새 로드 밸런서에 이전 주석을 사용하는 것이 좋습니다.

> **Important**
>
> 서비스를 생성한 후에는 주석을 편집하지 마세요. 수정해야 하는 경우 서비스 객체를 삭제하고 이 주석에 대해 원하는 값으로 다시 작성하세요.

Instance targets

AWS 클라우드 공급자 로드 밸런서 컨트롤러는 인스턴스 대상만으로 Network Load Balancer를 생성합니다. AWS 로드 밸런서 컨트롤러의 버전 `2.2.0` 이상도 인스턴스 대상을 사용하여 Network Load Balancer를 생성합니다. AWS 클라우드 공급자 로드 밸런서 컨트롤러보다 이를 사용하여 새 Network Load Balancer를 생성하는 것이 좋습니다. Amazon EC2 노드에 배포된 Pods와 함께 Network Load Balancer 인스턴스 대상을 사용할 수 있지만 Fargate에는 사용할 수 없습니다. Fargate에 배포된 Pods에서 네트워크 트래픽을 로드 밸런싱하려면 IP 대상을 사용해야 합니다.

프라이빗 서브넷에 Network Load Balancer를 배포하려면 서비스 사양에 다음과 같은 주석이 있어야 합니다. 주석이 있는 샘플 서비스 매니페스트를 볼 수 있습니다. `aws-load-balancer-type`의 `external` 값은 AWS 클라우드 공급자 로드 밸런서 컨트롤러가 아닌 AWS 로드 밸런서 컨트롤러가 Network Load Balancer를 생성하도록 하는 원인입니다.

```
service.beta.kubernetes.io/aws-load-balancer-type: "external"
service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "instance"
```

Network Load Balancer는 기본적으로 `internal` `aws-load-balancer-scheme`으로 생성됩니다. 내부 Network Load Balancer의 경우 Amazon EKS 클러스터는 VPC에서 하나 이상의 프라이빗 서브넷을 사용하도록 구성되어야 합니다. Kubernetes는 서브넷의 라우팅 테이블을 검사하여 퍼블릭인지 또는 프라이빗인지 판단합니다. 퍼블릭 서브넷에는 인터넷 게이트웨이를 사용해 인터넷으로 직접 연결되는 경로가 있지만 프라이빗 서브넷은 그렇지 않습니다.

Amazon EC2 노드로 로드 밸런싱하기 위해 퍼블릭 서브넷에 Network Load Balancer를 생성하려는 경우 다음 주석과 함께 `internet-facing`을 지정합니다.

```
service.beta.kubernetes.io/aws-load-balancer-scheme: "internet-facing"
```

> **Important**
>
> 서비스를 생성한 후에는 주석을 편집하지 마세요. 수정해야 하는 경우 서비스 객체를 삭제하고 이 주석에 대해 원하는 값으로 다시 작성하세요.

(선택 사항) 샘플 애플리케이션을 배포합니다.

필수 조건

- 클러스터 VPC의 퍼블릭 또는 프라이빗 서브넷 하나 이상 보유해야 합니다.
- 클러스터에 배포된 AWS Load Balancer Controller가 있습니다. 자세한 내용은 AWS Load Balancer Controller란 무엇인가요? 단원을 참조하십시오. 버전 `2.7.2` 이상을 사용하는 것이 좋습니다.

샘플 애플리케이션 배포

1. Fargate에 배포하는 경우 VPC에 사용 가능한 프라이빗 서브넷이 있는지 확인하고 Fargate 프로파일을 생성합니다. Fargate에 배포하지 않는 경우 이 단계를 건너뜁니다. 다음 명령을 실행하여 프로필을 생성하거나, AWS Management Console에서 명령의 `name` 및 `namespace`에 동일한 값을 사용하여 프로필을 생성할 수 있습니다. *`example values`*을 사용자의 값으로 교체합니다.

   ```
   eksctl create fargateprofile \
       --cluster my-cluster \
       --region region-code \
       --name nlb-sample-app \
       --namespace nlb-sample-app
   ```

2. 샘플 애플리케이션을 배포합니다.

   a. 애플리케이션에 대한 네임스페이스를 생성합니다.

   ```
   kubectl create namespace nlb-sample-app
   ```

   b. 다음 콘텐츠를 컴퓨터에서 *`sample-deployment`*`.yaml`이라는 파일에 저장합니다.

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: nlb-sample-app
     namespace: nlb-sample-app
   spec:
     replicas: 3
     selector:
       matchLabels:
         app: nginx
     template:
       metadata:
   ```

```
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: public.ecr.aws/nginx/nginx:1.23
          ports:
            - name: tcp
              containerPort: 80
```

c. 매니페스트를 클러스터에 적용합니다.

```
kubectl apply -f sample-deployment.yaml
```

3. IP 대상에 대한 로드 밸런싱을 수행하는 인터넷이 연결된 Network Load Balancer를 사용하여 서비스를 생성합니다.

a. 다음 콘텐츠를 컴퓨터에서 *sample-service*.yaml이라는 파일에 저장합니다. Fargate 노드에 배포하는 경우 `service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing` 줄을 제거합니다.

```
apiVersion: v1
kind: Service
metadata:
  name: nlb-sample-service
  namespace: nlb-sample-app
  annotations:
    service.beta.kubernetes.io/aws-load-balancer-type: external
    service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: ip
    service.beta.kubernetes.io/aws-load-balancer-scheme: internet-facing
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: LoadBalancer
  selector:
    app: nginx
```

b. 매니페스트를 클러스터에 적용합니다.

```
kubectl apply -f sample-service.yaml
```

4. 서비스가 배포되었는지 확인합니다.

```
kubectl get svc nlb-sample-service -n nlb-sample-app
```

예제 출력은 다음과 같습니다.

```
NAME             TYPE           CLUSTER-IP         EXTERNAL-IP
                                                  PORT(S)        AGE
sample-service   LoadBalancer   10.100.240.137
 k8s-nlbsampl-nlbsampl-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
  80:32400/TCP   16h
```

> **Note**
>
> *10.100.240.137* 및 *xxxxxxxxxx-xxxxxxxxxxxxxxxx*의 값은 예제 출력과 다르며(로드 밸런서마다 고유함) 클러스터가 있는 AWS 리전에 따라 *us-west-2*이 다를 수 있습니다.

5. Amazon EC2 AWS Management Console을 엽니다. 왼쪽 탐색 창의 로드 밸런싱(Load Balancing) 아래에서 대상 그룹(Target Groups)을 선택합니다. 이름(Name) 열에서 로드 밸런서(Load balancer) 열의 값이 이전 단계에서 출력의 EXTERNAL-IP 열에 있는 이름의 일부와 일치하는 대상 그룹의 이름을 선택합니다. 예를 들어 출력이 이전 출력과 동일한 경우 k8s-default-samplese-*xxxxxxxxxx*라는 대상 그룹을 선택합니다. 샘플 서비스 매니페스트에 지정되었기 때문에 대상 유형(Target type)이 IP입니다.

6. 대상 그룹(Target group)을 선택한 다음 대상(Targets) 탭을 선택합니다. 등록된 대상(Registered targets)에서 이전 단계에서 배포한 3개 복제본의 IP 주소 3개가 표시되어야 합니다. 계속하기 전에 모든 대상의 상태가 정상이 될 때까지 기다립니다. 모든 대상이 healthy가 될 때까지 몇 분 정도 걸릴 수 있습니다. 대상은 healthy 상태로 변경하기 전에 unhealthy 상태일 수 있습니다.

7. *xxxxxxxxxx-xxxxxxxxxxxxxxxx*와 *us-west-2*를 EXTERNAL-IP에 대한 이전 단계의 출력에서 반환된 값으로 바꿔 서비스로 트래픽을 전송합니다. 프라이빗 서브넷에 배포한 경우 Bastion Host와 같은 VPC 내의 디바이스에서 페이지를 확인해야 합니다. 자세한 내용은 AWS 기반 Linux Bastion Host를 참조하세요.

```
curl k8s-default-samplese-xxxxxxxxxx-xxxxxxxxxxxxxxxx.elb.region-code.amazonaws.com
```

예제 출력은 다음과 같습니다.

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
[...]
```

8. 샘플 배포, 서비스 및 네임스페이스가 완료되면 제거합니다.

```
kubectl delete namespace nlb-sample-app
```

Amazon EKS 애플리케이션 로드 밸런싱

Kubernetes `ingress`를 생성할 때 애플리케이션 트래픽을 로드 밸런싱하는 AWS Application Load Balancer(ALB)가 프로비저닝 됩니다. 자세한 내용은 Application Load Balancer 사용 설명서의 Application Load Balancer란 무엇인가요? 및 Kubernetes 설명서의 인그레스를 참조하세요. ALB는 노드 또는 AWS Fargate에 배포되는 Pods와 함께 사용할 수 있습니다. ALB를 퍼블릭 또는 프라이빗 서브넷에 배포할 수 있습니다.

애플리케이션 트래픽은 OSI 모델의 L7에서 로드 밸런싱됩니다. L4에서 네트워크 트래픽을 로드 밸런싱하려면 `LoadBalancer` 유형의 Kubernetes `service`를 배포합니다. 이 유형은 AWS Network Load Balancer를 프로비저닝 합니다. 자세한 내용은 Amazon EKS의 네트워크 로드 밸런싱 단원을 참조하십시오. 두 가지 로드 밸런싱 간의 차이점을 자세히 알아보려면 AWS 웹 사이트에서 Elastic 로드 밸런싱 기능 부분을 참조하세요.

필수 조건

애플리케이션 트래픽을 애플리케이션으로 로드 밸런싱 하기 전 다음 요구 사항을 충족해야 합니다.

- 기존 클러스터를 보유해야 합니다. 기존 클러스터가 없는 경우 Amazon EKS 시작하기 부분을 참조하세요. 기존 클러스터의 버전을 업데이트해야 하는 경우에는 Amazon EKS 클러스터 Kubernetes 버전 업데이트 부분을 참조하세요.
- 클러스터에 배포된 AWS Load Balancer Controller가 있습니다. 자세한 내용은 AWS Load Balancer Controller란 무엇인가요? 단원을 참조하십시오. 버전 `2.7.2` 이상을 사용하는 것이 좋습니다.
- 서로 다른 가용 영역에 두 개 이상의 서브넷을 보유해야 합니다. AWS Load Balancer Controller는 각 가용 영역에서 서브넷을 하나씩 선택합니다. 가용 영역에서 태그가 지정된 서브넷이 여러 개 있는 경우 컨트롤러는 서브넷 ID가 사전 순으로 가장 먼저 표시되는 서브넷을 선택합니다. 각 서브넷에는 최소 8개의 사용 가능한 IP 주소가 있어야 합니다.

작업자 노드에 연결된 보안 그룹을 여러 개 사용하는 경우 정확히 하나의 보안 그룹에 다음과 같이 태그를 지정해야 합니다. *my-cluster*을 클러스터 이름으로 교체합니다.

- 키 - kubernetes.io/cluster/*my-cluster*
- 값 - shared 또는 owned

- AWS Load Balancer Controller 버전 2.1.1 이하를 사용하는 경우 서브넷에 다음과 같은 형식으로 태그를 지정해야 합니다. 버전 2.1.2 이상을 사용하는 경우 태그 지정은 선택 사항입니다. 그러나 다음 중 하나라도 해당되는 경우 서브넷에 태그를 지정하는 것이 좋습니다. 동일한 VPC에서 실행 중인 클러스터가 여러 개 있거나 VPC에서 서브넷을 공유하는 AWS 서비스가 여러 개 있습니다. 또는 각 클러스터에 대해 로드 밸런서가 프로비저닝되는 위치에 대한 더 정밀한 제어가 필요합니다. *my-cluster*을 클러스터 이름으로 교체합니다.
  - 키 - kubernetes.io/cluster/*my-cluster*
  - 값 - shared 또는 owned
- 퍼블릭 및 프라이빗 서브넷은 다음 요구 사항을 충족해야 합니다. 서비스 또는 수신 개체에서 서브넷 ID를 주석으로 명시적으로 지정하지 않는 한 마찬가지입니다. 서브넷 ID를 서비스 또는 수신 개체에 주석으로 명시적으로 지정하여 로드 밸런서를 프로비저닝한다고 가정합니다. 이 경우, Kubernetes 및 AWS 로드 밸런서 컨트롤러는 이러한 서브넷을 직접 사용하여 로드 밸런서를 생성하며 다음 태그는 필요하지 않습니다.
  - 프라이빗 서브넷(Private subnets) - 다음 형식으로 태그를 지정해야 합니다. 이렇게 하면 Kubernetes 및 AWS 로드 밸런서 컨트롤러가 서브넷을 내부 로드 밸런서에 사용할 수 있음을 알 수 있습니다. 2020년 3월 26일 이후에 eksctl 또는 Amazon EKS AWS CloudFormation 템플릿을 통해 VPC를 생성하는 경우 서브넷은 적절하게 태그 지정됩니다. Amazon EKS AWS CloudFormation VPC 템플릿에 대한 자세한 내용은 Amazon EKS 클러스터에 대해 VPC 생성를 참조하세요.
    - 키 - kubernetes.io/role/internal-elb
    - 값 - 1
  - 퍼블릭 서브넷(Public subnets) - 다음 형식으로 태그를 지정해야 합니다. 이렇게 하면 Kubernetes에서 외부 로드 밸런서에 지정된 서브넷만 사용할 수 있음을 알 수 있습니다. 따라서 Kubernetes는 각 가용 영역에서 퍼블릭 서브넷을 선택하지 않습니다(서브넷 ID 기준 사전 순으로). 2020년 3월 26일 이후에 eksctl 또는 Amazon EKS AWS CloudFormation 템플릿을 통해 VPC를 생성하는 경우 서브넷은 적절하게 태그 지정됩니다. Amazon EKS AWS CloudFormation VPC 템플릿에 대한 자세한 내용은 Amazon EKS 클러스터에 대해 VPC 생성를 참조하세요.
    - 키 - kubernetes.io/role/elb
    - 값 - 1

서브넷 역할 태그가 명시적으로 추가되지 않은 경우 Kubernetes 서비스 컨트롤러는 클러스터 VPC 서브넷의 라우팅 테이블을 검사합니다. 이는 서브넷이 프라이빗 또는 퍼블릭 인지를 확인하기 위함입니다. 이 동작에는 의존하지 않는 것이 좋습니다. 대신 프라이빗 또는 퍼블릭 역할 태그를 명시적으로 추가합니다. AWS Load Balancer Controller에서는 라우팅 테이블을 검사하지 않습니다. 또한 자동 검색에 성공하려면 프라이빗 및 퍼블릭 태그가 있어야 합니다.

고려 사항

- AWS 로드 밸런서 컨트롤러는 `kubernetes.io/ingress.class: alb`주석을 사용하여 클러스터에 Kubernetes Kubernetes인그레스 리소스가 생성될 때마다 ALB 및 필요한 지원 AWS 리소스를 생성합니다. 인그레스 리소스는 ALB를 구성하여 HTTP 또는 HTTPS 트래픽을 클러스터 내 다른 Pods로 라우팅합니다. 인그레스 사양에서 AWS Load Balancer Controller를 사용하는지 확인하려면 Kubernetes 수신 사양에 다음 주석을 추가합니다. 자세한 내용은 GitHub의 인그레스 사양을 참조하세요.

```
annotations:
    kubernetes.io/ingress.class: alb
```

> **Note**
>
> IPv6 Pods로 로드 밸런싱하는 경우, 인그레스 사양에 다음 주석을 추가합니다. IPv6를 통해서는 인스턴스 대상이 아닌 IP 대상으로만 로드 밸런싱할 수 있습니다. 이 주석이 없으면 로드 밸런싱은 IPv4를 통해 이루어집니다.
>
> ```
> alb.ingress.kubernetes.io/ip-address-type: dualstack
> ```

- AWS Load Balancer Controller는 다음과 같은 트래픽 모드를 지원합니다.
  - 인스턴스 - 클러스터 내의 노드를 ALB의 대상으로 등록합니다. ALB에 도달하는 트래픽은 서비스를 위해 Pods로 라우팅된 다음 NodePort로 프록시됩니다. 이것이 기본 트래픽 모드입니다. `alb.ingress.kubernetes.io/target-type: instance` 주석을 사용하여 명시적으로 지정할 수도 있습니다.

> **Note**
>
> Kubernetes 서비스에서 `NodePort` 또는 이 트래픽 모드를 사용할 '로드 밸런서' 유형을 지정해야 합니다.

  - IP - Pods를 ALB의 대상으로 등록합니다. ALB에 도달하는 트래픽은 서비스를 위해 Pods로 직접 라우팅됩니다. 이 트래픽 모드를 사용하려면 `alb.ingress.kubernetes.io/target-type: ip` 주석을 지정해야 합니다. Fargate에서 대상 Pods가 실행 중인 경우 IP 대상 유형이 필요합니다.
- 컨트롤러에 의해 생성 된 ALB에 태그를 지정하려면 컨트롤러에 다음 주석을 추가합니다. `alb.ingress.kubernetes.io/tags`. AWS Load Balancer Controller에서 지원하는 모든 사용 가능한 주석의 목록은 GitHub의 인그레스 주석을 참조하세요.
- ALB 컨트롤러 버전을 업그레이드하거나 다운그레이드하면 해당 버전에서 활용하는 기능에 새로운 변경 사항이 발생할 수 있습니다. 각 릴리스에서 발생하는 새로운 변경 사항에 대한 자세한 내용은 GitHub의 ALB 컨트롤러 릴리스 정보를 참조하세요.

`IngressGroups`를 이용하여 여러 서비스 리소스 간에 애플리케이션 로드 밸런서를 공유합니다.

그룹 인그레스에 등록하려면 Kubernetes 인그레스 리소스 사양에 다음 주석을 추가합니다.

```
alb.ingress.kubernetes.io/group.name: my-group
```

그룹 이름은 다음과 같아야 합니다.

- 63자 이하의 길이여야 합니다.
- 소문자, 숫자, `-` 및 `.`으로 구성되어야 합니다.
- 문자나 숫자로 시작하여 끝나야 합니다.

컨트롤러는 동일한 그룹에 있는 모든 인그레스에 대한 규칙을 자동으로 병합합니다. 단일 ALB로 지원합니다. 인그레스에 정의된 대부분의 주석은 해당 인그레스에 의해 정의된 경로에만 적용됩니다. 기본적으로 인그레스 리소스는 인그레스 그룹에 속하지 않습니다.

> **Warning**
>
> 잠재적 보안 위험: 인그레스 리소스를 만들거나 수정할 수 있는 RBAC 권한이 있는 모든 Kubernetes 사용자가 동일한 신뢰 경계 내에 있는 경우에만 인그레스에 대한 수신 그룹을 지정

합니다. 그룹 이름을 사용하여 주석을 추가하는 경우 다른 Kubernetes 사용자가 동일한 인그레스 그룹에 속하도록 해당 인그레스를 생성하거나 수정할 수 있습니다. 이렇게 하면 기존 규칙을 우선 순위가 높은 규칙으로 덮어쓰는 것과 같이 바람직하지 않은 동작이 발생할 수 있습니다.

수신 리소스의 처리 순서를 추가할 수 있습니다.

```
alb.ingress.kubernetes.io/group.order: '10'
```

숫자는 1~1,000 사이어야 합니다. 동일한 인그레스 그룹의 모든 인그레스 중 가장 낮은 수가 먼저 평가됩니다. 이 주석이 없는 모든 인그레스는 값이 0인 것으로 평가됩니다. 숫자가 높은 중복 규칙은 낮은 수의 규칙을 덮어쓸 수 있습니다. 기본적으로 동일한 인그레스 그룹 내의 규칙 순서는 네임스페이스 및 이름을 기준으로 사전 순으로 결정됩니다.

Important

동일한 인그레스 그룹의 각 인그레스에 고유한 우선 순위 번호가 있는지 확인합니다. 인그레스에서는 중복 주문 번호를 사용할 수 없습니다.

(선택 사항) 샘플 애플리케이션을 배포합니다.

필수 조건

- 클러스터 VPC의 퍼블릭 또는 프라이빗 서브넷 하나 이상 보유해야 합니다.
- 클러스터에 배포된 AWS Load Balancer Controller가 있습니다. 자세한 내용은 AWS Load Balancer Controller란 무엇인가요? 단원을 참조하십시오. 버전 `2.7.2` 이상을 사용하는 것이 좋습니다.

샘플 애플리케이션 배포

Amazon EC2 노드가 있거나 Fargate Pods가 있거나 둘 다 있는 클러스터에서 샘플 애플리케이션을 실행할 수 있습니다.

1. Fargate에 배포하지 않는 경우 이 단계를 건너뜁니다. Fargate 에 배포하는 경우 Fargate 프로필을 생성합니다. 다음 명령을 실행하여 프로필을 생성하거나, AWS Management Console에서 명령의 `name` 및 `namespace`에 동일한 값을 사용하여 프로필을 생성할 수 있습니다. *`example values`*을 사용자의 값으로 교체합니다.

```
eksctl create fargateprofile \
    --cluster my-cluster \
    --region region-code \
    --name alb-sample-app \
    --namespace game-2048
```

2. 게임 2048을 샘플 애플리케이션으로 배포하여 AWS Load Balancer Controller가 인그레스 대상의 결과로 AWS ALB를 생성하는지 확인합니다. 배포하려는 서브넷 유형에 대한 단계를 완료합니다.

   a. IPv6 패밀리로 생성한 클러스터의 Pods에 배포하는 경우 다음 단계로 건너뜁니다.

      - 퍼블릭

        ```
        kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/examples/2048/2048_full.yaml
        ```

      - 프라이빗

        1. 매니페스트를 다운로드합니다.

           ```
           curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/examples/2048/2048_full.yaml
           ```

        2. 파일을 편집하고 `alb.ingress.kubernetes.io/scheme: internet-facing` 줄을 검색합니다.
        3. *internet-facing*을 **internal**로 변경하고 파일을 저장합니다.
        4. 매니페스트를 클러스터에 적용합니다.

           ```
           kubectl apply -f 2048_full.yaml
           ```

   b. IPv6 패밀리로 생성한 클러스터의 Pods에 배포하는 경우 다음 단계를 수행합니다.

      1. 매니페스트를 다운로드합니다.

         ```
         curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/examples/2048/2048_full.yaml
         ```

      2. 편집기에서 파일을 열고 인그레스 사양의 주석에 다음 줄을 추가합니다.

         ```
         alb.ingress.kubernetes.io/ip-address-type: dualstack
         ```

3. 인터넷 연결 Pods가 아니라 내부 Pods로 로드 밸런싱하는 경우 `alb.ingress.kubernetes.io/scheme: internet-facing` 줄을 `alb.ingress.kubernetes.io/scheme: internal`로 변경합니다.
4. 파일을 저장합니다.
5. 매니페스트를 클러스터에 적용합니다.

```
kubectl apply -f 2048_full.yaml
```

3. 몇 분 후, 인그레스 리소스가 다음 명령으로 생성되었는지 확인합니다.

```
kubectl get ingress/ingress-2048 -n game-2048
```

예제 출력은 다음과 같습니다.

```
NAME           CLASS    HOSTS   ADDRESS
                       PORTS   AGE
ingress-2048   <none>   *       k8s-game2048-ingress2-xxxxxxxxxx-yyyyyyyyyy.region-
code.elb.amazonaws.com   80      2m32s
```

> **Note**
>
> 프라이빗 서브넷에서 로드 밸런서를 생성한 경우 이전 출력에서 ADDRESS 아래의 값은 `internal-`로 시작합니다.
> 몇 분 후에도 수신이 생성되지 않은 경우 다음 명령을 실행하여 AWS Load Balancer Controller 로그를 확인합니다. 이러한 로그에는 배포 문제를 진단하는 데 사용할 수 있는 오류 메시지가 포함될 수 있습니다.
>
> ```
> kubectl logs -f -n kube-system -l app.kubernetes.io/instance=aws-load-
> balancer-controller
> ```

4. 퍼블릭 서브넷에 배포한 경우 브라우저를 열고 이전 명령 출력에서 ADDRESS URL로 이동하여 샘플 애플리케이션을 봅니다. 아무것도 표시되지 않으면 새로고침 한 후 다시 시도하세요. 프라이빗 서브넷에 배포한 경우 Bastion Host와 같은 VPC 내의 디바이스에서 페이지를 확인해야 합니다. 자세한 내용은 AWS 기반 Linux Bastion Host를 참조하세요.
5. 샘플 애플리케이션에 대한 실험이 끝나면 다음 명령 중 하나를 실행하여 이를 삭제합니다.

- 매니페스트를 적용했다면 다운로드한 사본을 적용하는 대신 다음 명령을 사용합니다.

```
kubectl delete -f https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.7.2/docs/examples/2048/2048_full.yaml
```

- 매니페스트를 다운로드하여 편집한 경우 다음 명령을 사용합니다.

```
kubectl delete -f 2048_full.yaml
```

서비스에 할당할 수 있는 외부 IP 주소 제한

Kubernetes 서비스는 클러스터 내부에서 다음을 통해 연결할 수 있습니다.

- Kubernetes가 자동으로 할당하는 클러스터 IP 주소
- 서비스 사양에서 `externalIPs` 속성에 대해 지정하는 모든 IP 주소. 외부 IP 주소는 Kubernetes에서 관리하지 않으며 클러스터 관리자의 책임입니다. `externalIPs`로 지정된 외부 IP 주소는 클라우드 공급자가 지정하여 `LoadBalancer` 서비스 유형에 할당되는 외부 IP 주소와 다릅니다.

Kubernetes 서비스에 대한 자세한 내용은 Kubernetes 설명서의 서비스를 참조하세요. 서비스 사양에서 `externalIPs`에 대해 지정할 수 있는 IP 주소를 제한할 수 있습니다.

서비스 사양에서 **`externalIPs`**에 대해 지정할 수 있는 IP 주소를 제한하려면

1. `cert-manager`를 배포하여 Webhook 인증서를 관리합니다. 자세한 내용은 `cert-manager` 설명서를 참조하세요.

```
kubectl apply -f https://github.com/jetstack/cert-manager/releases/download/v1.5.4/cert-manager.yaml
```

2. `cert-manager` Pods가 실행 중인지 확인하세요.

```
kubectl get pods -n cert-manager
```

예제 출력은 다음과 같습니다.

```
NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-58c8844bb8-nlx7q              1/1     Running   0          15s
cert-manager-cainjector-745768f6ff-696h5   1/1     Running   0          15s
cert-manager-webhook-67cc76975b-4v4nk      1/1     Running   0          14s
```

3. 기존 서비스를 검토하여 주소를 제한하려는 CIDR 블록 내에 포함되지 않은 외부 IP 주소가 할당되지 않았는지 확인합니다.

```
kubectl get services -A
```

예제 출력은 다음과 같습니다.

```
NAMESPACE                       NAME                                    TYPE
   CLUSTER-IP       EXTERNAL-IP     PORT(S)         AGE
cert-manager                    cert-manager                            ClusterIP
   10.100.102.137   <none>          9402/TCP        20m
cert-manager                    cert-manager-webhook                    ClusterIP
   10.100.6.136     <none>          443/TCP         20m
default                         kubernetes                              ClusterIP
   10.100.0.1       <none>          443/TCP         2d1h
externalip-validation-system    externalip-validation-webhook-service   ClusterIP
   10.100.234.179   <none>          443/TCP         16s
kube-system                     kube-dns                                ClusterIP
   10.100.0.10      <none>          53/UDP,53/TCP   2d1h
my-namespace                    my-service                              ClusterIP
   10.100.128.10    192.168.1.1     80/TCP          149m
```

값이 액세스를 제한하려는 블록 내에 있지 않은 IP 주소인 경우 블록 내에 포함되도록 주소를 변경하고 서비스를 다시 배포해야 합니다. 예를 들어 이전 출력의 `my-service` 서비스에는 5단계에서 CIDR 블록 예제 내에 없는 외부 IP 주소가 할당되어 있습니다.

4. 외부 IP Webhook 매니페스트를 다운로드합니다. 또한 GitHub에서 웹후크의 소스 코드를 볼 수도 있습니다.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/externalip-webhook.yaml
```

5. CIDR 블록을 지정합니다. 편집기에서 다운로드한 파일을 열고 다음 행의 시작 부분에 있는 #을 제거합니다.

```
#args:
#- --allowed-external-ip-cidrs=10.0.0.0/8
```

`10.0.0.0/8`을 실제 CIDR 블록으로 바꿉니다. 원하는 만큼 블록을 지정할 수 있습니다. 여러 블록을 지정하는 경우 블록 사이에 쉼표를 추가합니다.

6. 클러스터가 us-west-2 AWS 리전에 없는 경우 파일의 us-west-2, 602401143452 및 amazonaws.com을 다음 명령으로 . 명령을 실행하려면 먼저 *region-code* 및 *111122223333*을 Amazon 컨테이너 이미지 레지스트리의 목록의 AWS 리전 값으로 바꿉니다.

```
sed -i.bak -e 's|602401143452|111122223333|' externalip-webhook.yaml
sed -i.bak -e 's|us-west-2|region-code|' externalip-webhook.yaml
sed -i.bak -e 's|amazonaws.com||' externalip-webhook.yaml
```

7. 매니페스트를 클러스터에 적용합니다.

```
kubectl apply -f externalip-webhook.yaml
```

CIDR 블록 지정 단계에서 지정한 블록에 포함되어 있지 않은 externalIPs의 지정된 IP 주소를 사용하여 클러스터에 서비스를 배포하려고 시도할 경우 실패합니다.

한 리포지토리에서 다른 리포지토리로 컨테이너 이미지 복사

이 항목에서는 노드가 액세스할 수 없는 리포지토리에서 컨테이너 이미지를 가져와 노드가 액세스할 수 있는 리포지토리에 그 이미지를 푸시하는 방법에 대해 설명합니다. 이미지를 Amazon ECR 또는 노드가 액세스할 수 있는 대체 리포지토리로 푸시할 수 있습니다.

사전 조건

- 컴퓨터에 설치 및 구성된 Docker 엔진. 자세한 내용은 Docker 설명서의 Docker 엔진 설치 부분을 참조하세요.
- AWS Command Line Interface(AWS CLI) 버전 2.12.3 이상 또는 1.27.160 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **aws --version | cut -d / -f2 | cut -d ' ' -f1**을 사용합니다. macOS용 yum, apt-get 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.
- 노드가 Amazon 네트워크를 통해 프라이빗 Amazon ECR 리포지토리에서 컨테이너 이미지를 가져오거나 프라이빗 Amazon ECR 리포지토리에 컨테이너 이미지를 푸시하도록 하려는 경우 사용하는 Amazon ECR용 인터페이스 VPC 엔드포인트입니다. 자세한 내용은 Amazon Elastic Container Registry 사용 설명서의 Amazon ECR 인터페이스 VPC 엔드포인트 생성 단원을 참조하세요.

다음 단계를 수행하여 리포지토리에서 컨테이너 이미지를 가져와 자체 리포지토리로 푸시합니다. 이 항목에서 제공하는 다음 예제에서는Amazon VPC CNI plugin for Kubernetes metrics helper 에 대한 이미지를 가져옵니다. 다음 단계를 수행할 때 *example values*을 자체 값으로 교체합니다.

한 리포지토리에서 다른 리포지토리로 컨테이너 이미지 복사 방법

1. Amazon ECR 리포지토리나 기타 리포지토리가 아직 없다면 노드가 액세스할 수 있는 리포지토리를 생성합니다. 다음 명령은 Amazon ECR 프라이빗 리포지토리를 생성합니다. Amazon ECR 프라이빗 리포지토리 이름은 문자로 시작해야 합니다. 소문자, 숫자, 하이픈(-), 밑줄(_), 슬래시(/)만 포함할 수 있습니다. 자세한 내용은 Amazon Elastic Container Registry 사용 설명서의 프라이빗 리포지토리 생성 을 참조하세요.

   *cni-metrics-helper*을 원하는 것으로 바꿀 수 있습니다. 가장 좋은 방법은 각 이미지에 대해 별도의 리포지토리를 생성하는 것입니다. 리포지토리 내에서 이미지 태그가 고유해야 하므로 이 옵션을 사용하는 것이 좋습니다. *region-code*를 Amazon ECR이 지원하는 AWS 리전으로 변경합니다.

   ```
   aws ecr create-repository --region region-code --repository-name cni-metrics-helper
   ```

2. 노드가 가져와야 하는 이미지의 레지스트리, 리포지토리 및 태그(선택 사항)를 결정합니다. 이 정보는 `registry/repository[:tag]` 형식을 갖습니다.

   이미지 설치에 대한 Amazon EKS 주제 중 다수는 매니페스트 파일을 적용하거나 Helm 차트를 사용하여 이미지를 설치해야 합니다. 그러나 매니페스트 파일을 적용하거나 Helm 차트를 설치하기 전에 먼저 매니페스트 또는 차트의 `values.yaml` 파일 내용을 검토하십시오. 그렇게 하면 가져올 레지스트리, 리포지토리 및 태그를 결정할 수 있습니다.

   예를 들어, Amazon VPC CNI plugin for Kubernetes metrics helper에 대한 매니페스트 파일에서 다음 라인을 찾을 수 있습니다. 레지스트리는 `602401143452.dkr.ecr.us-west-2.amazonaws.com`이며, 이는 Amazon ECR 프라이빗 레지스트리입니다. 리포지토리는 `cni-metrics-helper`입니다.

   ```
   image: "602401143452.dkr.ecr.us-west-2.amazonaws.com/cni-metrics-helper:v1.12.6"
   ```

   이미지 위치에 대해 다음과 같은 변형이 나타날 수 있습니다.

   - `repository-name:tag`만 해당합니다. 이 경우 `docker.io`는 일반적으로 레지스트리이지만 레지스트리가 지정되지 않은 경우 Kubernetes가 기본적으로 리포지토리 이름 앞에 추가하기 때문에 지정되지 않습니다.

- `repository-name/repository-namespace/repository:tag`. 리포지토리 네임스페이스는 선택 사항이지만 이미지를 범주화하기 위해 리포지토리 소유자가 지정하는 경우도 있습니다. 예를 들어 모든 Amazon ECR Public Gallery의 Amazon EC2 이미지는 `aws-ec2` 네임스페이스를 사용합니다.

Helm을 사용하여 이미지를 설치하기 전에 Helm `values.yaml` 파일을 검토하여 이미지 위치를 결정합니다. 예를 들어, Amazon VPC CNI plugin for Kubernetes metrics helper에 대한 `values.yaml` 파일에는 다음과 같은 내용이 포함됩니다.

```
image:
  region: us-west-2
  tag: v1.12.6
  account: "602401143452"
  domain: "amazonaws.com"
```

3. 매니페스트 파일에 지정된 컨테이너 이미지를 가져옵니다.

   a. Amazon ECR Public Gallery와 같은 퍼블릭 레지스트리에서 가져오는 경우에는 인증이 필요하지 않으므로 다음 단계로 넘어가도 됩니다. 이 예에서는 CNI 지표 헬퍼 이미지에 대한 리포지토리를 포함하는 Amazon ECR 프라이빗 레지스트리에 대해 인증합니다. Amazon EKS는 Amazon 컨테이너 이미지 레지스트리에 나열된 각 레지스트리에서 이미지를 관리합니다. *602401143452* 및 *region-code*를 다른 레지스트리에 대한 정보로 바꿔 모든 레지스트리에 대해 인증할 수 있습니다. 각각의 Amazon EKS가 지원되는 AWS 리전에 대해 별도의 레지스트리가 있습니다.

   ```
   aws ecr get-login-password --region region-code | docker login --username AWS --
   password-stdin 602401143452.dkr.ecr.region-code.amazonaws.com
   ```

   b. 이미지를 가져옵니다. 이 예제에서는 이전 하위 단계에서 인증한 레지스트리에서 가져옵니다. *602401143452* 및 *region-code*를 이전 하위 단계에서 제공한 정보로 바꿉니다.

   ```
   docker pull 602401143452.dkr.ecr.region-code.amazonaws.com/cni-metrics-
   helper:v1.12.6
   ```

4. 레지스트리, 리포지토리 및 태그로 가져온 이미지에 태그를 지정합니다. 다음 예제에서는 매니페스트 파일에서 이미지를 가져와서 첫 번째 단계에서 생성한 Amazon ECR 프라이빗 리포지토리로 푸시한다고 가정합니다. *111122223333*을 계정 ID로 바꿉니다. *region-code*를 Amazon ECR 프라이빗 리포지토리를 생성한 AWS 리전으로 바꿉니다.

```
docker tag cni-metrics-helper:v1.12.6 111122223333.dkr.ecr.region-
code.amazonaws.com/cni-metrics-helper:v1.12.6
```

5. 레지스트리를 인증합니다. 이 예제에서는 첫 번째 단계에서 생성한 Amazon ECR 프라이빗 레지스트리에 대해 인증합니다. 자세한 내용은 Amazon Elastic Container Registry 사용 설명서의 레지스트리 권한을 참조하세요.

```
aws ecr get-login-password --region region-code | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.region-code.amazonaws.com
```

6. 리포지토리에 이미지를 푸시합니다. 이 예제에서는 첫 번째 단계에서 생성한 Amazon ECR 프라이빗 레지스트리에 이미지를 푸시합니다. 자세한 내용은 Amazon Elastic Container Registry 사용 설명서의 Docker 이미지 푸시하기 부분을 참조하세요.

```
docker push 111122223333.dkr.ecr.region-code.amazonaws.com/cni-metrics-
helper:v1.12.6
```

7. 푸시한 이미지에 대해 이전 단계에서 이미지를 결정하는데 `registry/repository:tag`와 함께 사용된 매니페스트 파일을 업데이트합니다. Helm 차트를 사용하여 설치하는 경우 종종 `registry/repository:tag`를 지정하는 옵션이 있습니다. 차트를 설치할 때 리포지토리에 푸시한 이미지에 대한 `registry/repository:tag`를 지정합니다.

Amazon 컨테이너 이미지 레지스트리

AWS Amazon EKS 추가 기능을 클러스터에 배포하면 노드는 추가 기능의 설치 메커니즘에 지정된 레지스트리에서 설치 매니페스트나 Helm `values.yaml` 파일과 같은 필수 컨테이너 이미지를 가져옵니다. Amazon EKS Amazon ECR 프라이빗 리포지토리에서 이미지를 생성합니다. Amazon EKS는 각 Amazon EKS가 지원하는 AWS 리전 리포지토리에 이미지를 복제합니다. 노드는 다음 레지스트리 중에서 컨테이너 이미지를 인터넷을 통해 가져올 수 있습니다. 또는 VPC에서 Amazon ECR에 대한 인터페이스 VPC 엔드포인트(AWS PrivateLink)를 생성한 경우 노드는 Amazon 네트워크를 통해 이미지를 가져올 수도 있습니다. 레지스트리는 AWS IAM 계정을 통한 인증이 필요합니다. 노드는 관련된 AmazonEC2ContainerRegistryReadOnly 관리형 IAM 정책의 권한을 갖는 Amazon EKS 노드 IAM 역할을 이용하여 인증합니다.

AWS 리전	레지스트리
af-south-1	877085696533.dkr.ecr.af-south-1.amazonaws.com
ap-east-1	800184023465.dkr.ecr.ap-east-1.amazonaws.com
ap-northeast-1	602401143452.dkr.ecr.ap-northeast-1.amazonaws.com
ap-northeast-2	602401143452.dkr.ecr.ap-northeast-2.amazonaws.com
ap-northeast-3	602401143452.dkr.ecr.ap-northeast-3.amazonaws.com
ap-south-1	602401143452.dkr.ecr.ap-south-1.amazonaws.com
ap-south-2	900889452093.dkr.ecr.ap-south-2.amazonaws.com
ap-southeast-1	602401143452.dkr.ecr.ap-southeast-1.amazonaws.com/
ap-southeast-2	602401143452.dkr.ecr.ap-southeast-2.amazonaws.com
ap-southeast-3	296578399912.dkr.ecr.ap-southeast-3.amazonaws.com
ap-southeast-4	491585149902.dkr.ecr.ap-southeast-4.amazonaws.com
ca-central-1	602401143452.dkr.ecr.ca-central-1.amazonaws.com

AWS 리전	레지스트리
ca-west-1	761377655185.dkr.ecr.ca-west-1.amazonaws.com
cn-north-1	918309763551.dkr.ecr.cn-north-1.amazonaws.com.cn
cn-northwest-1	961992271922.dkr.ecr.cn-northwest-1.amazonaws.com.cn
eu-central-1	602401143452.dkr.ecr.eu-central-1.amazonaws.com
eu-central-2	900612956339.dkr.ecr.eu-central-2.amazonaws.com
eu-north-1	602401143452.dkr.ecr.eu-north-1.amazonaws.com
eu-south-1	590381155156.dkr.ecr.eu-south-1.amazonaws.com
eu-south-2	455263428931.dkr.ecr.eu-south-2.amazonaws.com
eu-west-1	602401143452.dkr.ecr.eu-west-1.amazonaws.com
eu-west-2	602401143452.dkr.ecr.eu-west-2.amazonaws.com
eu-west-3	602401143452.dkr.ecr.eu-west-3.amazonaws.com
il-central-1	066635153087.dkr.ecr.il-central-1.amazonaws.com
me-south-1	558608220178.dkr.ecr.me-south-1.amazonaws.com

AWS 리전	레지스트리
me-central-1	759879836304.dkr.ecr.me-central-1.amazonaws.com
sa-east-1	602401143452.dkr.ecr.sa-east-1.amazonaws.com
us-east-1	602401143452.dkr.ecr.us-east-1.amazonaws.com
us-east-2	602401143452.dkr.ecr.us-east-2.amazonaws.com
us-gov-east-1	151742754352.dkr.ecr.us-gov-east-1.amazonaws.com
us-gov-west-1	013241004608.dkr.ecr.us-gov-west-1.amazonaws.com
us-west-1	602401143452.dkr.ecr.us-west-1.amazonaws.com
us-west-2	602401143452.dkr.ecr.us-west-2.amazonaws.com

Amazon EKS 추가 기능

추가 기능은 Kubernetes 애플리케이션에 대한 지원 운영 기능을 제공하는 소프트웨어이지만 애플리케이션에만 국한되지는 않습니다. 여기에는 기본 AWS 리소스를 사용하여 네트워킹, 컴퓨팅 및 스토리지를 관리할 수 있도록 지원하는 관찰 가능성 에이전트 또는 Kubernetes 드라이버가 포함됩니다. 추가 기능 소프트웨어는 일반적으로 Kubernetes 커뮤니티, AWS와 같은 클라우드 공급자 또는 타사 공급업체에 의해 구축 및 유지 관리됩니다. Amazon EKS는 모든 클러스터에 대해 Amazon VPC CNI plugin for Kubernetes, `kube-proxy` 및 CoreDNS와 같은 자체 관리형 추가 기능을 자동으로 설치합니다. 원하는 경우 추가 기능의 기본 구성을 변경하고 업데이트할 수 있습니다.

Amazon EKS 추가 기능은 Amazon EKS 클러스터에 대해 큐레이팅된 추가 기능 집합을 설치하고 관리합니다. 모든 Amazon EKS 추가 기능에는 최신 보안 패치, 버그 수정 사항이 포함되어 있으

며, Amazon EKS와 작동하도록 AWS에서 검증을 거쳤습니다. Amazon EKS 추가 기능을 사용하면 Amazon EKS 클러스터가 안전하고 안정적이도록 일관되게 보장할 수 있으며, 추가 기능을 설치하고 구성하고 업데이트하기 위해 수행해야 하는 작업량을 줄일 수 있습니다. 자체 관리형 추가 기능(예: `kube-proxy`)이 클러스터에서 이미 실행 중이고 Amazon EKS 추가 기능으로 사용할 수 있는 경우, `kube-proxy` Amazon EKS 추가 기능을 설치하여 Amazon EKS 추가 기능의 이점을 활용할 수 있습니다.

Amazon EKS API를 통해 Amazon EKS 추가 기능에 대한 특정 Amazon EKS 관리형 구성 필드를 업데이트할 수 있습니다. 추가 기능이 시작되면 Amazon EKS가 관리하지 않는 구성 필드를 Kubernetes 클러스터 내에서 직접 수정할 수도 있습니다. 해당하는 경우, 여기에는 추가 기능에 대한 특정 구성 필드를 정의하는 작업이 포함됩니다. 이러한 변경 사항은 Amazon EKS에 의해 재정의되지 않습니다. 이는 Kubernetes 서버 측 적용 기능을 사용하여 지원됩니다. 자세한 내용은 Kubernetes 필드 관리 단원을 참조하십시오.

모든 Amazon EKS 노드 유형에서 Amazon EKS 추가 기능을 사용할 수 있습니다.

고려 사항

- 클러스터의 추가 기능을 구성하려면 IAM 보안 주체에게 추가 기능을 사용할 수 있는 IAM 권한이 있어야 합니다. 자세한 내용은 Amazon Elastic Kubernetes Service에서 정의한 작업에서 해당 이름의 `Addon` 작업을 참조하세요.
- Amazon EKS 추가 기능은 클러스터용으로 프로비저닝하거나 구성하는 노드에서 실행됩니다. 노드 유형에는 Amazon EC2 인스턴스와 Fargate가 포함됩니다.
- Amazon EKS가 관리하지 않는 필드를 수정하여 Amazon EKS 추가 기능의 설치를 사용자 지정할 수 있습니다. 자세한 내용은 Kubernetes 필드 관리 단원을 참조하십시오.
- AWS Management Console을 사용하여 클러스터를 생성하는 경우, Amazon EKS `kube-proxy`, Amazon VPC CNI plugin for Kubernetes 및 CoreDNS Amazon EKS 추가 기능이 클러스터에 자동으로 추가됩니다. `eksctl`을 사용하여 `config` 파일이 있는 클러스터를 생성하는 경우 `eksctl`은 또한 Amazon EKS 추가 기능을 사용하여 클러스터를 생성할 수 있습니다. `config` 파일 없이 `eksctl`을 사용하거나 다른 도구를 사용하여 클러스터를 생성할 경우 Amazon EKS 추가 기능이 아니라 자체 관리형 `kube-proxy`, Amazon VPC CNI plugin for Kubernetes 및 CoreDNS 추가 기능이 설치됩니다. 클러스터 생성 후 Amazon EKS 추가 기능을 수동으로 추가하거나 직접 관리할 수 있습니다.
- `eks:addon-cluster-admin` `ClusterRoleBinding`에서는 `cluster-admin` `ClusterRole`을 `eks:addon-manager` Kubernetes 자격 증명에 바인딩합니다. Kubernetes 네임스페이스를 생성하고 추가 기능을 네임스페이스에 설치하는 `eks:addon-manager` 자격 증명에 필요한 권한이 역할에 있습니다. `eks:addon-cluster-admin` `ClusterRoleBinding`이 제거되는 경우 Amazon EKS

클러스터는 계속 작동하지만, Amazon EKS에서 더 이상 추가 기능을 관리할 수 없습니다. 다음과 같은 플랫폼 버전으로 시작하는 모든 클러스터에서 새 `ClusterRoleBinding`을 사용합니다.

Kubernetes 버전	플랫폼 버전
1.20	eks.12
1.21	eks.14
1.22	eks.9
1.23	eks.5
1.24	eks.3

Amazon EKS API, AWS Management Console, AWS CLI 및 `eksctl`을 사용하여 Amazon EKS 추가 기능을 추가, 업데이트 또는 삭제할 수 있습니다. 자세한 내용은 Amazon EKS 추가 기능 관리 단원을 참조하십시오. 또한 AWS CloudFormation을 사용하여 Amazon EKS 추가 기능을 생성할 수도 있습니다.

Amazon EKS에서 사용 가능한 Amazon EKS 추가 기능

다음과 같은 Amazon EKS 추가 기능을 사용하여 클러스터를 생성할 수 있습니다. `eksctl`, AWS Management Console 또는 AWS CLI을(를) 사용하여 사용 가능한 추가 기능의 최신 목록을 언제든지 볼 수 있습니다. 사용 가능한 모든 추가 기능을 보거나 추가 기능을 설치하려면 추가 기능 생성 섹션을 참조하세요. 추가 기능에 IAM 권한이 필요한 경우 클러스터에 대한 IAMOpenID Connect(OIDC) 공급자가 있어야 합니다. 제공업체가 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요. 추가 기능을 설치한 후에는 추가 기능을 업데이트하거나 삭제할 수 있습니다.

추가 기능과 설치 요구 사항에 대해 자세히 알아보려면 해당 추가 기능을 선택합니다.

Amazon VPC CNI plugin for Kubernetes

- 명칭 – `vpc-cni`
- 설명 – 클러스터에 네이티브 VPC 네트워킹을 제공하는 Kubernetes 컨테이너 네트워크 인터페이스(CNI) 플러그인입니다. 이 추가 기능의 자체 관리형 또는 관리형 유형은 기본적으로 각 Amazon EC2 노드에 설치됩니다.
- 필수 IAM 권한 – 이 추가 기능은 Amazon EKS의 서비스 계정에 대한 IAM 역할 기능을 활용합니다. 클러스터에서 `IPv4` 패밀리를 사용하는 경우 AmazonEKS_CNI_Policy의 권한이 필요합니다. 클러스터에서 `IPv6` 패밀리를 사용하는 경우 IPv6 모드에서 해당 권한이 있는 IAM 정책을 만들어야 합니다. 다음 명령을 사용하여 IAM 역할을 생성하고, 정책 중 하나를 이 역할에 연결하고, 추가 기능에서 사용하는 Kubernetes 서비스 계정에 주석을 달 수 있습니다.

  *`my-cluster`*를 클러스터 이름으로 바꾸고, *`AmazonEKSVPCCNIRole`*을 역할의 이름으로 바꿉니다. 클러스터에서 `IPv6` 패밀리를 사용하는 경우 *`AmazonEKS_CNI_Policy`*를 생성한 정책의 이름으로 바꿉니다. 이 명령을 사용하려면 디바이스에 `eksctl`가 설치되어 있어야 합니다. 다른 도구를 사용하여 역할을 생성하고 이 역할에 정책을 연결하고, Kubernetes 서비스 계정에 주석을 추가해야 하는 경우 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 섹션을 참조하세요.

  ```
  eksctl create iamserviceaccount --name aws-node --namespace kube-system --cluster my-cluster --role-name AmazonEKSVPCCNIRole \
      --role-only --attach-policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy --approve
  ```

- 추가 정보 – 추가 기능의 구성 가능한 설정에 대해 자세히 알아보려면 GitHub에서 aws-vpc-cni-k8s를 참조하세요. 플러그인에 대해 자세히 알아보려면 제안: AWS VPC를 통한 Kubernetes 네트워킹을 위한 CNI 플러그인를 참조하세요. 추가 기능 생성에 대한 자세한 내용은 Amazon EKS 추가 기능 생성 부분을 참조하세요.
- 정보 업데이트 – 마이너 버전은 한 번에 하나만 업데이트할 수 있습니다. 예를 들어 현재 버전이 `1.27.`*`x`*`-eksbuild.`*`y`*인데 `1.29.`*`x`*`-eksbuild.`*`y`*(으)로 업데이트하려는 경우 현재 버전을 `1.28.`*`x`*`-eksbuild.`*`y`*(으)로 먼저 업데이트한 다음에 `1.29.`*`x`*`-eksbuild.`*`y`*(으)로 업데이트해야 합니다. 추가 기능을 업데이트하는 방법에 대한 자세한 내용은 Amazon EKS 추가 기능 업데이트 부분을 참조하세요.

CoreDNS

- 명칭 – `coredns`

- 설명 – Kubernetes 클러스터 DNS로 제공할 수 있는 유연하고 확장 가능한 DNS 서버입니다. 이 추가 기능의 자체 관리형 또는 관리형 유형은 기본적으로 클러스터를 만들 때 설치되었습니다. 하나 이상의 노드가 있는 Amazon EKS 클러스터를 시작하면 클러스터에 배포된 노드 수에 관계없이 CoreDNS 이미지의 복제본 2개가 기본적으로 배포됩니다. CoreDNS Pods는 클러스터의 모든 Pods의 이름을 확인합니다. 클러스터에 CoreDNS `deployment`의 네임스페이스와 일치하는 네임스페이스가 있는 AWS Fargate 프로파일이(가) 포함되어 있는 경우 CoreDNS Pods을(를) Fargate 노드에 배포할 수 있습니다.
- 필수 IAM 권한 – 이 추가 기능에는 권한이 필요하지 않습니다.
- 추가 정보 – CoreDNS에 대해 자세히 알아보려면 Kubernetes 설명서에서 서비스 검색에 CoreDNS 사용 및 DNS 서비스 사용자 지정을 참조하세요.

`Kube-proxy`

- 명칭 – `kube-proxy`
- 설명 – 각 Amazon EC2 노드에서 네트워크 규칙을 관리합니다. 이를 통해 Pods와의 네트워크 통신이 가능합니다. 이 추가 기능의 자체 관리형 또는 관리형 유형은 기본적으로 클러스터의 각 Amazon EC2 노드에 설치됩니다.
- 필수 IAM 권한 – 이 추가 기능에는 권한이 필요하지 않습니다.
- 추가 정보 – `kube-proxy`에 대해 자세히 알아보려면 Kubernetes 설명서에서 `kube-proxy`을(를) 참조하세요.
- 정보 업데이트 – 현재 버전을 업데이트하기 전에 다음 요구 사항을 고려하세요.
  - Amazon EKS 클러스터의 `Kube-proxy`에는 Kubernetes와(과) 동일한 호환성 및 스큐 정책이 있습니다.
  - `Kube-proxy`은(는) Amazon EC2 노드의 `kubelet`와(과) 동일한 마이너 버전이어야 합니다.
  - `Kube-proxy`은(는) 클러스터 컨트롤 플레인의 마이너 버전보다 이후일 수 없습니다.
  - Amazon EC2 노드의 `kube-proxy` 버전은 컨트롤 플레인보다 이전의 마이너 버전이 2개를 초과할 수 없습니다. 예를 들어, 컨트롤 플레인에서 Kubernetes 1.29를 실행하고 있는 경우 `kube-proxy` 마이너 버전은 1.27보다 이전 버전일 수 없습니다.
  - 최근에 클러스터를 새 Kubernetes 마이너 버전으로 업데이트한 경우에는 노드와 동일한 마이너 버전으로 Amazon EC2 노드를 업데이트하기 전에 `kube-proxy`을(를) 노드와 동일한 마이너 버전으로 업데이트합니다.

Amazon EBS CSI 드라이버

- 명칭 – `aws-ebs-csi-driver`
- 설명 – 클러스터에 Amazon EBS 스토리지를 제공하는 Kubernetes 컨테이너 스토리지 인터페이스(CSI) 플러그인입니다.
- 필수 IAM 권한 – 이 추가 기능은 Amazon EKS의 서비스 계정에 대한 IAM 역할 기능을 활용합니다. AmazonEBSCSIDriverPolicy AWS 관리형 정책의 권한이 필요합니다. IAM 역할을 생성하고 다음 명령을 사용하여 관리형 정책을 연결할 수 있습니다. *my-cluster*를 클러스터 이름으로 바꾸고, *AmazonEKS_EBS_CSI_DriverRole*을 역할의 이름으로 바꿉니다. 이 명령을 사용하려면 디바이스에 eksctl가 설치되어 있어야 합니다. 다른 도구를 사용해야 하거나 암호화에 사용자 지정 KMS 키를 사용해야 하는 경우 Amazon EBS CSI 드라이버 IAM 역할 생성 섹션을 참조하세요.

```
eksctl create iamserviceaccount \
    --name ebs-csi-controller-sa \
    --namespace kube-system \
    --cluster my-cluster \
    --role-name AmazonEKS_EBS_CSI_DriverRole \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEBSCSIDriverPolicy
 \
    --approve
```

- 추가 정보 – 추가 기능에 대해 자세히 알아보려면 Amazon EBS CSI 드라이버 섹션을 참조하세요.

Amazon EFS CSI 드라이버

- 명칭 – `aws-efs-csi-driver`
- 설명 – 클러스터에 Amazon EFS 스토리지를 제공하는 Kubernetes 컨테이너 스토리지 인터페이스(CSI) 플러그인입니다.
- 필수 IAM 권한 – 이 추가 기능은 Amazon EKS의 서비스 계정에 대한 IAM 역할 기능을 활용합니다. AmazonEFSCSIDriverPolicy AWS 관리형 정책의 권한이 필요합니다. IAM 역할을 생성하고 다음 명령을 사용하여 관리형 정책을 연결할 수 있습니다. *my-cluster*를 클러스터 이름으로 바꾸고, *AmazonEKS_EFS_CSI_DriverRole*을 역할의 이름으로 바꿉니다. 이 명령을 사용하려면 디바이스에 eksctl이(가) 설치되어 있어야 합니다. 다른 도구를 사용해야 하는 경우 IAM 역할 생성을(를) 참조하세요.

```
export cluster_name=my-cluster
export role_name=AmazonEKS_EFS_CSI_DriverRole
```

```
eksctl create iamserviceaccount \
    --name efs-csi-controller-sa \
    --namespace kube-system \
    --cluster $cluster_name \
    --role-name $role_name \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/service-role/AmazonEFSCSIDriverPolicy
 \
    --approve
TRUST_POLICY=$(aws iam get-role --role-name $role_name --query
 'Role.AssumeRolePolicyDocument' | \
    sed -e 's/efs-csi-controller-sa/efs-csi-*/' -e 's/StringEquals/StringLike/')
aws iam update-assume-role-policy --role-name $role_name --policy-document
 "$TRUST_POLICY"
```

- 추가 정보 – 추가 기능에 대해 자세히 알아보려면 Amazon EFS CSI 드라이버 섹션을 참조하세요.

Mountpoint for Amazon S3 CSI 드라이버

- 명칭 – `aws-mountpoint-s3-csi-driver`
- 설명 – 클러스터에 Amazon S3 스토리지를 제공하는 Kubernetes 컨테이너 스토리지 인터페이스(CSI) 플러그인입니다.
- 필수 IAM 권한 – 이 추가 기능은 Amazon EKS의 서비스 계정에 대한 IAM 역할 기능을 활용합니다. 생성된 IAM 역할에는 S3에 대한 액세스 권한을 부여하는 정책이 필요합니다. 정책을 생성할 때 Mountpoint IAM 권한 권장 사항을 따르세요. 또는 AWS 관리형 정책 AmazonS3FullAccess를 사용할 수 있지만 이 관리형 정책은 Mountpoint에 필요한 것보다 더 많은 권한을 부여합니다.

  IAM 역할을 생성하고 다음 명령을 사용하여 정책을 연결할 수 있습니다. *my-cluster*를 클러스터의 이름, *region-code*을 올바른 AWS 리전 코드, *AmazonEKS_S3_CSI_DriverRole*을 역할 이름, *AmazonEKS_S3_CSI_DriverRole_ARN*을 역할 ARN으로 바꿉니다. 이 명령을 사용하려면 디바이스에 `eksctl`이(가) 설치되어 있어야 합니다. IAM 콘솔 또는 AWS CLI 사용에 대한 지침은 IAM 역할 생성 섹션을 참조하세요.

```
CLUSTER_NAME=my-cluster
REGION=region-code
ROLE_NAME=AmazonEKS_S3_CSI_DriverRole
POLICY_ARN=AmazonEKS_S3_CSI_DriverRole_ARN
eksctl create iamserviceaccount \
    --name s3-csi-driver-sa \
    --namespace kube-system \
```

```
    --cluster $CLUSTER_NAME \
    --attach-policy-arn $POLICY_ARN \
    --approve \
    --role-name $ROLE_NAME \
    --region $REGION \
    --role-only
```

- 추가 정보 – 추가 기능에 대해 자세히 알아보려면 Mountpoint for Amazon S3 CSI 드라이버 섹션을 참조하세요.

CSI 스냅샷 컨트롤러

- 명칭 – `snapshot-controller`
- 설명 - 컨테이너 스토리지 인터페이스(CSI) 스냅샷 컨트롤러를 사용하면 Amazon EBS CSI 드라이버와 같은 호환 CSI 드라이버에서 스냅샷 기능을 사용할 수 있습니다.
- 필수 IAM 권한 – 이 추가 기능에는 권한이 필요하지 않습니다.
- 추가 정보 – 추가 기능에 대해 자세히 알아보려면 CSI 스냅샷 컨트롤러 섹션을 참조하세요.

AWS Distro for OpenTelemetry

- 명칭 – `adot`
- 설명 - AWS Distro for OpenTelemetry(ADOT)는 OpenTelemetry 프로젝트의 안전하고, 프로덕션 준비가 된 AWS 지원 배포입니다.
- 필수 IAM 권한 - 이 추가 기능에는 고급 구성을 통해 옵트인할 수 있는 사전 구성된 사용자 지정 리소스 중 하나를 사용하는 경우에만 IAM 권한이 필요합니다.
- 추가 정보 – 자세한 내용은 AWS Distro for OpenTelemetry 설명서에서 EKS 추가 기능을 사용하여 AWS Distro for OpenTelemetry 시작하기를 참조하세요.

  ADOT를 사용하려면 `cert-manager`가 필수 구성 요소로 클러스터에 배포되어야 하며, 그렇지 않으면 Amazon EKS Terraform `cluster_addons` 속성을 사용하여 직접 배포하는 경우 이 추가 기능이 작동하지 않습니다. 자세한 요구 사항은 AWS Distro for OpenTelemetry 설명서에서 EKS 추가 기능을 사용하여 AWS Distro for OpenTelemetry 시작하기 요구 사항을 참조하세요.

Amazon GuardDuty 에이전트

- 명칭 – `aws-guardduty-agent`

- 설명 — Amazon GuardDuty는AWS CloudTrail 관리 이벤트 및 Amazon VPC 흐름 로그를 비롯한 기본 데이터 소스를 분석하고 처리하는 보안 모니터링 서비스입니다. 또한 Amazon GuardDuty는 Kubernetes 감사 로그 및 런타임 모니터링과 같은 기능을 처리합니다.
- 필수 IAM 권한 – 이 추가 기능에는 권한이 필요하지 않습니다.
- 추가 정보 - 자세한 내용은 Amazon GuardDuty의 Runtime Monitoring for Amazon EKS clusters를 참조하십시오.
  - Amazon EKS 클러스터에서 잠재적인 보안 위협을 탐지하려면 Amazon GuardDuty 런타임 모니터링을 활성화하고 GuardDuty 보안 에이전트를 Amazon EKS 클러스터에 배포하십시오.

Amazon CloudWatch Observability 에이전트

- 명칭 – `amazon-cloudwatch-observability`
- 설명: Amazon CloudWatch 에이전트는 AWS에서 제공하는 모니터링 및 관찰성 서비스입니다. 이 추가 기능은 CloudWatch 에이전트를 설치하고 Amazon EKS의 향상된 관찰성을 통해 CloudWatch Application Signals와 CloudWatch Container Insights를 모두 활성화합니다.
- 필수 IAM 권한 – 이 추가 기능은 Amazon EKS의 서비스 계정에 대한 IAM 역할 기능을 활용합니다. AWSXrayWriteOnlyAccess 및 CloudWatchAgentServerPolicy AWS 관리형 정책의 권한이 필요합니다. 다음 명령을 사용하여 IAM 역할을 생성하고, 관리형 정책을 이 역할에 연결하고, 추가 기능에서 사용하는 Kubernetes 서비스 계정에 주석을 달 수 있습니다. *`my-cluster`*를 클러스터 이름으로 바꾸고, *`AmazonEKS_Observability_role`*을 역할의 이름으로 바꿉니다. 이 명령을 사용하려면 디바이스에 `eksctl`가 설치되어 있어야 합니다. 다른 도구를 사용하여 역할을 생성하고 이 역할에 정책을 연결하고, Kubernetes 서비스 계정에 주석을 추가해야 하는 경우 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 섹션을 참조하세요.

```
eksctl create iamserviceaccount \
    --name cloudwatch-agent \
    --namespace amazon-cloudwatch \
    --cluster my-cluster \
    --role-name AmazonEKS_Observability_Role \
    --role-only \
    --attach-policy-arn arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess \
    --attach-policy-arn arn:aws:iam::aws:policy/CloudWatchAgentServerPolicy \
    --approve
```

- 추가 정보 - 자세한 내용은 CloudWatch 에이전트 설치를 참조하세요.

Amazon EKS Pod Identity Agent

- 명칭 – `eks-pod-identity-agent`
- 설명 - Amazon EKS Pod Identity는 Amazon EC2 인스턴스 프로파일이 EC2 인스턴스에 자격 증명을 제공하는 것과 비슷한 방식으로 애플리케이션에 대한 자격 증명을 관리하는 기능을 제공합니다.
- 필수 IAM 권한 - 이 추가 기능에는 Amazon EKS 노드 IAM 역할의 사용자 권한이 필요합니다.
- 정보 업데이트 – 마이너 버전은 한 번에 하나만 업데이트할 수 있습니다. 예를 들어 현재 버전이 `1.27.`*x*`-eksbuild.`*y*인데 `1.29.`*x*`-eksbuild.`*y*(으)로 업데이트하려는 경우 현재 버전을 `1.28.`*x*`-eksbuild.`*y*(으)로 먼저 업데이트한 다음에 `1.29.`*x*`-eksbuild.`*y*(으)로 업데이트해야 합니다. 추가 기능을 업데이트하는 방법에 대한 자세한 내용은 Amazon EKS 추가 기능 업데이트 부분을 참조하세요.

독립 소프트웨어 공급업체의 추가 Amazon EKS 추가 기능

이전 Amazon EKS 추가 기능 목록 외에 독립 소프트웨어 공급업체의 작업용 소프트웨어 Amazon EKS 추가 기능을 폭넓게 선택하여 추가할 수도 있습니다. 추가 기능과 설치 요구 사항에 대해 자세히 알아보려면 해당 추가 기능을 선택합니다.

AWS Marketplace에서 추가 기능 찾기, 조달 및 Amazon EKS에 배포(YouTube).

Accuknox

- 게시자 – Accuknox
- 명칭 – `accuknox_kubearmor`
- 네임스페이스 – `kubearmor`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 – KubeArmor 설명서의 Getting Started with KubeArmor를 참조하세요.

NetApp

- 게시자 – NetApp
- 명칭 – `netapp_trident-operator`
- 네임스페이스 – `trident`

- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 – NetApp 설명서의 Configure the Astra Trident EKS add-on을 참조하세요.

Calyptia

- 게시자 – Calyptia
- 명칭 – `calyptia_fluent-bit`
- 네임스페이스 – `calytia-fluentbit`
- 서비스 계정 이름 – `clyptia-fluentbit`
- AWS 관리형 IAM 정책 – AWSMarketplaceMeteringRegisterUsage.
- 필수 IAM 역할을 생성하는 명령 – 다음과 같은 명령에는 클러스터의 IAM OpenID Connect(OIDC) 제공업체가 필요합니다. 제공업체가 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요. *my-cluster*를 클러스터 이름으로 바꾸고, *my-calyptia-role*을 역할의 이름으로 바꿉니다. 이 명령을 사용하려면 디바이스에 `eksctl`가 설치되어 있어야 합니다. 다른 도구를 사용하여 역할을 생성하고 Kubernetes 서비스 계정에 주석을 추가해야 하는 경우 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 섹션을 참조하세요.

  ```
  eksctl create iamserviceaccount --name service-account-name  --namespace calyptia-
  fluentbit --cluster my-cluster --role-name my-calyptia-role \
      --role-only --attach-policy-arn arn:aws:iam::aws:policy/
  AWSMarketplaceMeteringRegisterUsage --approve
  ```

- 설정 및 사용 지침 - Calyptia 설명서의 Calyptia for Fluent Bit를 참조하세요.

Cribl

- 게시자 – Cribl
- 명칭 – `cribl_cribledge`
- 네임스페이스 – `cribledge`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.

- 설정 및 사용 지침 - Cribl 설명서의 Installing the Cribl Amazon EKS Add-on for Edge를 참조하세요.

Dynatrace

- 게시자 – Dynatrace
- 명칭 – `dynatrace_dynatrace-operator`
- 네임스페이스 – `dynatrace`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 – dynatrace 설명서의 Kubernetes monitoring(Kubernetes 모니터링)을 참조하세요.

Datree

- 게시자 – Datree
- 명칭 – `datree_engine-pro`
- 네임스페이스 – `datree`
- 서비스 계정 이름 - datree-webhook-server-awsmp
- AWS 관리형 IAM 정책 – AWSLicenseManagerConsumptionPolicy.
- 필수 IAM 역할을 생성하는 명령 – 다음과 같은 명령에는 클러스터의 IAM OpenID Connect(OIDC) 제공업체가 필요합니다. 제공업체가 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요. *`my-cluster`*를 클러스터 이름으로 바꾸고, *`my-datree-role`*을 역할의 이름으로 바꿉니다. 이 명령을 사용하려면 디바이스에 `eksctl`가 설치되어 있어야 합니다. 다른 도구를 사용하여 역할을 생성하고 Kubernetes 서비스 계정에 주석을 추가해야 하는 경우 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 섹션을 참조하세요.

```
eksctl create iamserviceaccount --name datree-webhook-server-awsmp --namespace datree
 --cluster my-cluster --role-name my-datree-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 – Datree 설명서의 Amazon EKS-intergration을 참조하세요.

Datadog

- 게시자 – Datadog
- 명칭 – `datadog_operator`
- 네임스페이스 – `datadog-agent`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설치 및 사용 지침 - Datadog 설명서의 Installing the Datadog Agent on Amazon EKS with the Datadog Operator Add-on을 참조하세요.

Groundcover

- 게시자 – groundcover
- 명칭 – `groundcover_agent`
- 네임스페이스 – `groundcover`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 - groundcover 설명서의 Installing the groundcover Amazon EKS Add-on을 참조하세요.

Grafana Labs

- 게시자 – Grafana Labs
- 명칭 – `grafana-labs_kubernetes-monitoring`
- 네임스페이스 – `monitoring`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 - Grafana Labs 설명서에서 Configure Kubernetes Monitoring as an Add-on with Amazon EKS를 참조하세요.

HA Proxy

- 게시자 – HA Proxy
- 명칭 – `haproxy-technologies_kubernetes-ingress-ee`
- 네임스페이스 – `haproxy-controller`
- 서비스 계정 이름 – `customer defined`
- AWS 관리형 IAM 정책 – AWSLicenseManagerConsumptionPolicy.
- 필수 IAM 역할을 생성하는 명령 – 다음과 같은 명령에는 클러스터의 IAM OpenID Connect(OIDC) 제공업체가 필요합니다. 제공업체가 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요. *my-cluster*를 클러스터 이름으로 바꾸고, *my-haproxy-role*을 역할의 이름으로 바꿉니다. 이 명령을 사용하려면 디바이스에 `eksctl`가 설치되어 있어야 합니다. 다른 도구를 사용하여 역할을 생성하고 Kubernetes 서비스 계정에 주석을 추가해야 하는 경우 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 섹션을 참조하세요.

  ```
  eksctl create iamserviceaccount --name service-account-name  --namespace haproxy-
  controller --cluster my-cluster --role-name my-haproxy-role \
      --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
  AWSLicenseManagerConsumptionPolicy --approve
  ```
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 - HAProxy 문서의 AWS에서 Amazon EKS의 HAProxy Enterprise Kubernetes Ingress 컨트롤러 설치를 참조하세요.

Kpow

- 게시자 – Factorhouse
- 명칭 – `factorhouse_kpow`
- 네임스페이스 – `factorhouse`
- 서비스 계정 이름 – `kpow`
- AWS 관리형 IAM 정책 – AWSLicenseManagerConsumptionPolicy
- 필수 IAM 역할을 생성하는 명령 – 다음과 같은 명령에는 클러스터의 IAM OpenID Connect(OIDC) 제공업체가 필요합니다. 제공업체가 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요. *my-cluster*를 클러스터 이름으로 바꾸고, *my-kpow-role*을 역할의 이름으로 바꿉니다. 이 명령을 사용하려면 디바이스에 `eksctl`가 설치되어 있어야 합니다. 다른 도구를 사용하여 역할을 생성하고 Kubernetes 서비스 계정에 주석을 추가해야 하는 경우 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 섹션을 참조하세요.

```
eksctl create iamserviceaccount --name kpow --namespace factorhouse --cluster my-
cluster --role-name my-kpow-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 – Kpow 설명서의 AWS Marketplace LM을 참조하세요.

Kubecost

- 게시자 – Kubecost
- 명칭 – `kubecost_kubecost`
- 네임스페이스 – `kubecost`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 – Kubecost 설명서의 AWS 클라우드 결제 통합을 참조하세요.
- 클러스터가 `1.23` 또는 이후 버전인 경우 클러스터에 the section called “Amazon EBS CSI 드라이버”가 설치되어 있어야 합니다. 그렇지 않으면 오류가 발생합니다.

Kasten

- 게시자 – Kasten by Veeam
- 명칭 – `kasten_k10`
- 네임스페이스 – `kasten-io`
- 서비스 계정 이름 – `k10-k10`
- AWS 관리형 IAM 정책 – AWSLicenseManagerConsumptionPolicy.
- 필수 IAM 역할을 생성하는 명령 – 다음과 같은 명령에는 클러스터의 IAM OpenID Connect(OIDC) 제공업체가 필요합니다. 제공업체가 있는지 아니면 생성해야 하는지 확인하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요. *my-cluster*를 클러스터 이름으로 바꾸고, *my-kasten-role*을 역할의 이름으로 바꿉니다. 이 명령을 사용하려면 디바이스에 `eksctl`가 설치되어 있어야 합니다. 다른 도구를 사용하여 역할을 생성하고 Kubernetes 서비스 계정에 주석을 추가해야 하는 경우 IAM 역할을 수임하도록 Kubernetes 서비스 계정 구성 섹션을 참조하세요.

```
eksctl create iamserviceaccount --name k10-k10 --namespace kasten-io --cluster my-
cluster --role-name my-kasten-role \
    --role-only --attach-policy-arn arn:aws:iam::aws:policy/service-role/
AWSLicenseManagerConsumptionPolicy --approve
```

- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 - Kasten 설명서의 Installing K10 on AWS using Amazon EKS Add-on을 참조하세요.
- 추가 정보 - Amazon EKS 클러스터가 버전 Kubernetes `1.23` 이상인 경우 기본 `StorageClass`를 사용하여 클러스터에 Amazon EBS CSI 드라이버가 설치되어 있어야 합니다.

Kong

- 게시자 – Kong
- 명칭 – `kong_konnect-ri`
- 네임스페이스 – `kong`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 - Kong 설명서의 Installing the Kong Gateway EKS Add-on을 참조하세요.

LeakSignal

- 게시자 – LeakSignal
- 명칭 – `leaksignal_leakagent`
- 네임스페이스 – `leakagent`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설치 및 사용 지침 - LeakSignal 설명서에서 Install the LeakAgent add-on을 참조하세요.

New Relic

- 게시자 – New Relic

- 명칭 – `new-relic_kubernetes-operator`
- 네임스페이스 – `newrelic`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 - New Relic 설명서의 Installing the New Relic Add-on for EKS를 참조하세요.

Rafay

- 게시자 – Rafay
- 명칭 – `rafay-systems_rafay-operator`
- 네임스페이스 – `rafay-system`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 - Rafay 설명서의 Installing the Rafay Amazon EKS Add-on을 참조하세요.

Solo.io

- 게시자 – Solo.io
- 명칭 – `solo-io_istio-distro`
- 네임스페이스 – `istio-system`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설치 및 사용 지침 — Solo.io 설명서의 Istio 설치를 참조하세요.

Stormforge

- 게시자 – Stormforge
- 명칭 – `stormforge_optimize-Live`

- 네임스페이스 – `stormforge-system`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 - StormForge 설명서의 Installing the StormForge Agent를 참조하세요.

Splunk

- 게시자 – Splunk
- 명칭 – `splunk_splunk-otel-collector-chart`
- 네임스페이스 – `splunk-monitoring`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 - Splunk 설명서에서 Install the Splunk add-on for Amazon EKS를 참조하세요.

Teleport

- 게시자 – Teleport
- 명칭 – `teleport_teleport`
- 네임스페이스 – `teleport`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 – Teleport 설명서의 How Teleport Works를 참조하세요.

Tetrate

- 게시자 – Tetrate Io
- 명칭 – `tetrate-io_istio-distro`
- 네임스페이스 – `istio-system`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.

- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설치 및 사용 지침 – Tetrate Istio Distro 웹 사이트를 참조하세요.

Upbound Universal Crossplane

- 게시자 – Upbound
- 명칭 – `upbound_universal-crossplane`
- 네임스페이스 – `upbound-system`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 – Upbound 설명서의 Upbound Universal Crossplane(UXP)을 참조하세요.

Upwind

- 게시자 – Upwind
- 명칭 – `upwind`
- 네임스페이스 – `upwind`
- 서비스 계정 이름 – 서비스 계정은 이 추가 기능에서 사용하지 않습니다.
- AWS 관리형 IAM 정책 – 관리형 정책은 이 추가 기능에서 사용하지 않습니다.
- 사용자 지정 IAM 권한 – 사용자 지정 권한은 이 추가 기능에서 사용하지 않습니다.
- 설정 및 사용 지침 – Upwind 설명서의 설치 단계를 참조하세요.

Amazon EKS 추가 기능 관리

Amazon EKS 추가 기능은 Amazon EKS 클러스터용으로 선별된 추가 기능 소프트웨어 세트입니다. 모든 Amazon EKS 추가 기능:

- 최신 보안 패치 및 버그 수정이 포함됩니다.
- Amazon EKS와 연동하도록 AWS에서 검증했습니다.
- 추가 기능 소프트웨어 관리에 필요한 작업량이 감소합니다.

Amazon EKS 추가 기능의 새 버전을 사용할 수 있으면 AWS Management Console에 알림이 표시됩니다. 간단히 업데이트를 시작할 수 있으며 Amazon EKS에서 추가 기능 소프트웨어가 업데이트됩니다.

사용 가능한 추가 기능 목록은 Amazon EKS에서 사용 가능한 Amazon EKS 추가 기능 섹션을 참조하세요. Kubernetes 필드 관리에 대한 자세한 내용은 Kubernetes 필드 관리 부분을 참조하세요.

필수 조건

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요.
- Kubernetes 서비스 계정과 IAM 역할을 사용하는 추가 기능을 생성하는 경우에는 클러스터의 AWS Identity and Access Management(IAM) OpenID Connect(OIDC) 제공업체가 있어야 합니다. 클러스터의 해당 제공업체가 이미 있는지 아니면 생성해야 하는지 결정하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요.

추가 기능 생성

`eksctl`, AWS Management Console 또는 AWS CLI를 사용하여 Amazon EKS 추가 기능을 생성할 수 있습니다. 추가 기능에 IAM 역할이 필요한 경우 역할 생성에 대한 자세한 내용은 Amazon EKS에서 사용 가능한 Amazon EKS 추가 기능에서 특정 추가 기능에 대한 세부 정보를 참조하십시오.

eksctl

전제 조건

디바이스 또는 `0.175.0`에 설치된 버전 AWS CloudShell 이상의 `eksctl` 명령줄 도구. `eksctl`을 설치 또는 업그레이드하려면 `eksctl` 설명서에서 Installation을 참조하세요.

`eksctl`을 사용하여 Amazon EKS 추가 기능을 생성하는 방법

1. 클러스터 버전에 사용할 수 있는 추가 기능의 이름을 조회합니다. *`1.29`*를 클러스터의 버전으로 바꿉니다.

```
eksctl utils describe-addon-versions --kubernetes-version 1.29 | grep AddonName
```

예제 출력은 다음과 같습니다.

```
"AddonName": "aws-ebs-csi-driver",
                        "AddonName": "coredns",
                        "AddonName": "kube-proxy",
```

```
                    "AddonName": "vpc-cni",
                    "AddonName": "adot",
                    "AddonName": "dynatrace_dynatrace-operator",
                    "AddonName": "upbound_universal-crossplane",
                    "AddonName": "teleport_teleport",
                    "AddonName": "factorhouse_kpow",
                    [...]
```

2. 생성하려는 추가 기능에 사용할 수 있는 버전을 조회합니다. *1.29*를 클러스터의 버전으로 바꿉니다. 버전을 조회하려는 추가 기능의 이름으로 *name-of-addon*을 바꿉니다. 이름은 이전 단계에서 반환된 이름 중 하나여야 합니다.

```
eksctl utils describe-addon-versions --kubernetes-version 1.29 --name name-of-addon | grep AddonVersion
```

다음과 같은 출력은 vpc-cni라는 추가 기능에 대해 반환되는 내용의 예입니다. 사용할 수 있는 여러 버전이 추가 기능에 있는지 참조할 수 있습니다.

```
"AddonVersions": [
    "AddonVersion": "v1.12.0-eksbuild.1",
    "AddonVersion": "v1.11.4-eksbuild.1",
    "AddonVersion": "v1.10.4-eksbuild.1",
    "AddonVersion": "v1.9.3-eksbuild.1",
```

3. 생성하려는 추가 기능이 Amazon EKS 또는 AWS Marketplace 추가 기능인지 결정합니다. AWS Marketplace에는 추가 기능을 생성하려면 추가 단계를 완료해야 하는 타사 추가 기능이 있습니다.

```
eksctl utils describe-addon-versions --kubernetes-version 1.29 --name name-of-addon | grep ProductUrl
```

출력이 반환되지 않는다면 추가 기능이 Amazon EKS입니다. 출력이 반환된다면 추가 기능이 AWS Marketplace 추가 기능입니다. 다음은 teleport_teleport라는 추가 기능의 출력입니다.

```
"ProductUrl": "https://aws.amazon.com/marketplace/pp?
sku=3bda70bb-566f-4976-806c-f96faef18b26"
```

반환된 URL을 통해 AWS Marketplace에서 추가 기능에 대해 자세히 알아볼 수 있습니다. 추가 기능에 구독이 필요한 경우 AWS Marketplace를 통해 추가 기능을 구독할 수 있습니다. AWS

Marketplace 중에서 추가 기능을 생성하려는 경우 추가 기능 생성에 사용하는 IAM 보안 주체에 AWSServiceRoleForAWSLicenseManagerRole 서비스 연결 역할을 생성하는 권한이 있어야 합니다. IAM 개체에 권한을 할당하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 제거를 참조하세요.

4. Amazon EKS 추가 기능을 생성합니다. 다음 명령을 디바이스에 복사합니다. 필요에 따라 명령을 다음과 같이 수정한 다음에 수정한 명령을 실행합니다.

   - *my-cluster*를 클러스터 이름으로 바꿉니다.
   - *name-of-addon*을 생성하려는 추가 기능 이름으로 바꿉니다.
   - 최신 버전보다 이전의 추가 기능 버전을 원하는 경우에는 사용하려는 이전 단계의 출력에서 반환된 버전 번호로 *latest*를 바꿉니다.
   - 추가 기능에서 서비스 계정 역할을 사용하는 경우 *111122223333*을 계정 ID로 바꾸고 *role-name*을 역할 이름으로 바꿉니다. 서비스 계정의 역할 생성에 대한 지침은 생성 중인 추가 기능의 설명서를 참조하세요. 서비스 계정 역할을 지정하려면 클러스터의 IAM OpenID Connect(OIDC) 제공업체가 있어야 합니다. 클러스터의 해당 제공업체가 이미 있는지 아니면 생성해야 하는지 결정하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요.

     추가 기능에서 서비스 계정 역할을 사용하지 않으면 ***--service-account-role-arn*** **arn:aws:iam::*111122223333*:role/*role-name*을** 삭제합니다.
   - 이 예시 명령에서는 추가 기능의 기존 자체 관리형 버전(있는 경우)의 구성을 덮어씁니다. 기존 자체 관리형 추가 기능의 구성을 덮어쓰지 않으려면 *--force* 옵션을 제거합니다. 옵션을 제거하고 Amazon EKS 추가 기능에 기존 자체 관리형 추가 기능 구성을 덮어써야 하는 경우 Amazon EKS 추가 기능의 생성에 실패하고 충돌 해결에 도움이 되는 오류 메시지가 표시됩니다. 이 옵션을 사용하여 이러한 설정을 덮어쓰기 때문에 이 옵션을 지정하기 전에 Amazon EKS 추가 기능이 관리해야 하는 설정을 관리하지 않는지 확인합니다.

     ```
     eksctl create addon --cluster my-cluster --name name-of-addon --version latest
      \
        --service-account-role-arn arn:aws:iam::111122223333:role/role-name --
     force
     ```

명령에 사용할 수 있는 모든 옵션의 목록을 참조할 수 있습니다.

```
eksctl create addon --help
```

사용할 수 있는 옵션에 대한 자세한 내용은 `eksctl` 설명서의 Addons(추가 기능)를 참조하세요.

AWS Management Console

AWS Management Console을 사용하여 Amazon EKS 추가 기능을 생성하는 방법

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Clusters(클러스터)를 선택한 다음에 추가 기능을 생성할 클러스터의 이름을 선택합니다.
3. 추가 기능(Add-ons) 탭을 선택합니다.
4. 추가 기능 더 가져오기를 선택합니다.
5. 클러스터를 추가할 추가 기능을 선택합니다. Amazon EKS add-ons(Amazon EKS 추가 기능)와 AWS Marketplace add-ons(추가 기능)를 필요한 만큼 추가할 수 있습니다.

   AWS Marketplace 추가 기능의 경우 추가 기능을 생성하는 데 사용 중인 IAM 보안 주체 AWS LicenseManager에서 추가 기능에 대한 자격을 읽는 권한이 있어야 합니다. AWS LicenseManager에는 AWS 리소스에서 자동으로 라이선스를 관리할 수 있는 AWSServiceRoleForAWSLicenseManagerRole SLR(서비스 연결 역할)이 필요합니다. SLR은 계정당 한 번만 필요하므로 각 추가 기능 또는 각 클러스터에 대해 별도의 SLR을 생성할 필요가 없습니다. IAM 보안 주체에 권한을 할당하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 삭제를 참조하세요.

   설치하려는 AWS Marketplace 추가 기능이 목록에 없는 경우 검색 상자에 텍스트를 입력하여 사용 가능한 추가 기능을 검색할 수 있습니다. 필터링 옵션에서 카테고리, 공급업체 또는 요금 모델별로 필터링한 다음에 검색 결과 중에서 추가 기능을 선택할 수도 있습니다. 설치할 추가 기능을 선택했으면 Next(다음)를 선택합니다.
6. Configure selected add-ons settings(선택한 추가 기능 설정 구성) 페이지에서:
   - 구독 옵션 보기를 선택하여 구독 옵션 양식을 엽니다. 가격 세부 정보 및 법률 섹션을 검토한 다음 구독 버튼을 선택하여 계속하세요.
   - Version(버전)의 경우 설치할 버전을 선택합니다. 생성 중인 개별 추가 기능에서 다른 버전이 권장되지 않으면 latest(최신)로 표시된 버전이 좋습니다. 추가 기능에 권장 버전이 있는지 확인하려면 생성 중인 추가 기능의 설명서를 참조하세요.
   - 선택한 모든 추가 기능의 Status(상태) 아래에 Requires subscription(구독 필요)이 있으면 Next(다음)를 선택합니다. 클러스터를 생성한 후 구독할 때까지는 추가로 해당 추가 기능을 구성할 수 없습니다. Status(상태) 아래에 Requires subscription(구독 필요)이 없는 추가 기능의 경우:

- Select IAM role(IAM 역할 선택)의 경우 추가 기능에 IAM 권한이 필요하지 않으면 기본 옵션을 수락합니다. 추가 기능에 AWS 권한이 필요한 경우 노드의 IAM 역할(설정되지 않음) 또는 추가 기능에 사용하려고 생성한 기존 역할을 선택할 수 있습니다. 선택할 역할이 없는 경우에는 기존 역할이 없는 것입니다. 선택하는 옵션과 관계없이 IAM 정책을 생성하고 역할에 연결하려고 생성 중인 추가 기능의 설명서를 참조하세요. IAM 역할을 선택하려면 클러스터에 대한 IAM OpenID Connect(OIDC) 제공업체가 있어야 합니다. 클러스터의 해당 제공업체가 이미 있는지 아니면 생성해야 하는지 결정하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요.
- Optional configuration settings(선택적 구성 설정)를 선택합니다.
  - 추가 기능에 구성이 필요하면Configuration values(구성 값) 상자에 해당 구성을 입력합니다. 추가 기능에 구성 정보가 필요한지 확인하려면 생성 중인 추가 기능의 설명서를 참조하세요.
  - Conflict resolution method(충돌 해결 방법)에 사용할 수 있는 옵션 중 하나를 선택합니다.
- 다음을 선택합니다.

7. 검토 및 추가 페이지에서 생성을 선택합니다. 추가 기능 설치가 완료되면 설치한 추가 기능이 표시됩니다.
8. 설치한 추가 기능 중 구독이 필요한 추가 기능이 있으면 다음과 같은 단계를 완료합니다.
   1. 추가 기능의 오른쪽 하단에 있는 Subscribe(구독) 버튼을 선택합니다. AWS Marketplace의 추가 기능 페이지가 나타납니다. Product Overview(제품 개요) 및 Pricing Information(요금 정보)과 같은 추가 기능에 대한 정보를 읽어보세요.
   2. 추가 기능 페이지의 오른쪽 상단에 있는 Continue to Subscribe(계속 구독) 버튼을 선택합니다.
   3. Terms and Conditions(이용 약관)를 모두 읽어보세요. 동의하는 경우 Accept Terms(약관 수락)를 선택합니다. 구독을 처리하는 데 몇 분 정도 걸릴 수 있습니다. 구독이 처리되는 동안에는 Return to Amazon EKS Console(Amazon EKS 콘솔로 돌아가기) 버튼이 회색으로 표시됩니다.
   4. 구독 처리가 완료되면 Return to Amazon EKS Console(Amazon EKS 콘솔로 돌아가기) 버튼이 더는 회색으로 표시되지 않습니다. 버튼을 선택하여 클러스터의 Amazon EKS 콘솔 Add-ons(추가 기능) 탭으로 돌아갑니다.
   5. 구독한 추가 기능의 경우 Remove and reinstall(제거 및 재설치)을 선택한 다음에 Reinstall add-on(추가 기능 재설치)을 선택합니다. 추가 기능을 설치하는 데 몇 분 정도 걸릴 수 있습니다. 설치가 완료되면 추가 기능을 구성할 수 있습니다.

AWS CLI

전제 조건

AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.

AWS CLI를 사용하여 Amazon EKS 추가 기능을 생성하는 방법

1. 사용할 수 있는 추가 기능을 확인합니다. 사용할 수 있는 모든 추가 기능, 해당 유형 및 게시자를 참조할 수 있습니다. 사용할 수 있는 추가 기능의 URL도 AWS Marketplace를 통해 참조할 수 있습니다. *`1.29`*를 클러스터의 버전으로 바꿉니다.

```
aws eks describe-addon-versions --kubernetes-version 1.29 \
    --query 'addons[].{MarketplaceProductUrl: marketplaceInformation.productUrl,
 Name: addonName, Owner: owner Publisher: publisher, Type: type}' --output table
```

예제 출력은 다음과 같습니다.

```
-------------------------------------------------------------------------------------------
|
 DescribeAddonVersions
       |
+-----------------------------------------------------------
+--------------------------------+------------------+--------------
+---------------------+
|                     MarketplaceProductUrl                      |
 Name              |      Owner       |  Publisher   |        Type         |
+-----------------------------------------------------------
+--------------------------------+------------------+--------------
+---------------------+
|  None                                                          |  aws-ebs-csi-
driver           |  aws             |  eks         |  storage            |
|  None                                                          |  coredns
                |  aws             |  eks         |  networking         |
```

```
|  None                                                                  |  kube-proxy
                |  aws             |  eks         |  networking          |
|  None                                                                  |  vpc-cni
                |  aws             |  eks         |  networking          |
|  None                                                                  |  adot
                |  aws             |  eks         |  observability       |
|  https://aws.amazon.com/marketplace/pp/prodview-brb73nceicv7u |
 dynatrace_dynatrace-operator |  aws-marketplace |  dynatrace   |  monitoring
      |
|  https://aws.amazon.com/marketplace/pp/prodview-uhc2iwi5xysoc |
 upbound_universal-crossplane |  aws-marketplace |  upbound     |  infra-
management   |
|  https://aws.amazon.com/marketplace/pp/prodview-hd2ydsrgqy4li |
 teleport_teleport            |  aws-marketplace |  teleport    |  policy-
management  |
|  https://aws.amazon.com/marketplace/pp/prodview-vgghgqdsplhvc |
 factorhouse_kpow             |  aws-marketplace |  factorhouse |  monitoring
      |
|  [...]                                                                 |  [...]
                |  [...]           |  [...]       |  [...]               |
+------------------------------------------------------------------
+-------------------------------+------------------+--------------
+---------------------+
```

출력은 다를 수도 있습니다. 이 예시 출력에는 networking 유형에 사용할 수 있는 3가지 추가 기능과 eks 유형의 게시자가 있는 5가지 추가 기능이 있습니다. Owner 열에 aws-marketplace가 있는 추가 기능을 설치하려면 구독이 필요할 수 있습니다. URL을 방문하여 추가 기능에 대해 자세히 알아보고 구독할 수 있습니다.

2. 각 추가 기능에 어떤 버전을 사용할 수 있는지 확인할 수 있습니다. *1.29*을 클러스터의 버전으로 바꾸고 *vpc-cni*를 이전 단계에서 반환된 추가 기능 이름으로 바꿉니다.

```
aws eks describe-addon-versions --kubernetes-version 1.29 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

예제 출력은 다음과 같습니다.

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
```

```
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

기본적으로 Defaultversion 열에 True가 있는 버전이 추가 기능을 생성한 버전입니다.

3. (선택 사항) 다음과 같은 명령을 실행하여 선택한 추가 기능의 구성 옵션을 찾습니다.

```
aws eks describe-addon-configuration --addon-name vpc-cni --addon-
version v1.12.0-eksbuild.1
```

```
{
    "addonName": "vpc-cni",
    "addonVersion": "v1.12.0-eksbuild.1",
    "configurationSchema": "{\"$ref\":\"#/definitions/VpcCni\",\"$schema
\":\"http://json-schema.org/draft-06/schema#\",\"definitions\":{\"Cri\":
{\"additionalProperties\":false,\"properties\":{\"hostPath\":{\"$ref\":
\"#/definitions/HostPath\"}},\"title\":\"Cri\",\"type\":\"object\"},\"Env
\":{\"additionalProperties\":false,\"properties\":{\"ADDITIONAL_ENI_TAGS
\":{\"type\":\"string\"},\"AWS_VPC_CNI_NODE_PORT_SUPPORT\":{\"format\":
\"boolean\",\"type\":\"string\"},\"AWS_VPC_ENI_MTU\":{\"format\":\"integer
\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CONFIGURE_RPFILTER\":{\"format
\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG\":
{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_EXTERNALSNAT
\":{\"format\":\"boolean\",\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOGLEVEL
\":{\"type\":\"string\"},\"AWS_VPC_K8S_CNI_LOG_FILE\":{\"type
\":\"string\"},\"AWS_VPC_K8S_CNI_RANDOMIZESNAT\":{\"type\":
\"string\"},\"AWS_VPC_K8S_CNI_VETHPREFIX\":{\"type\":\"string
\"},\"AWS_VPC_K8S_PLUGIN_LOG_FILE\":{\"type\":\"string\"},
\"AWS_VPC_K8S_PLUGIN_LOG_LEVEL\":{\"type\":\"string\"},\"DISABLE_INTROSPECTION
\":{\"format\":\"boolean\",\"type\":\"string\"},\"DISABLE_METRICS\":{\"format
\":\"boolean\",\"type\":\"string\"},\"DISABLE_NETWORK_RESOURCE_PROVISIONING
\":{\"format\":\"boolean\",\"type\":\"string\"},\"ENABLE_POD_ENI\":{\"format
\":\"boolean\",\"type\":\"string\"},\"ENABLE_PREFIX_DELEGATION\":{\"format
\":\"boolean\",\"type\":\"string\"},\"WARM_ENI_TARGET\":{\"format\":\"integer
\",\"type\":\"string\"},\"WARM_PREFIX_TARGET\":{\"format\":\"integer\",
\"type\":\"string\"}},\"title\":\"Env\",\"type\":\"object\"},\"HostPath\":
{\"additionalProperties\":false,\"properties\":{\"path\":{\"type\":\"string\"}},
\"title\":\"HostPath\",\"type\":\"object\"},\"Limits\":{\"additionalProperties
\":false,\"properties\":{\"cpu\":{\"type\":\"string\"},\"memory\":{\"type
```

```
\":\"string\"}},\"title\":\"Limits\",\"type\":\"object\"},\"Resources\":
{\"additionalProperties\":false,\"properties\":{\"limits\":{\"$ref\":\"#/
definitions/Limits\"},\"requests\":{\"$ref\":\"#/definitions/Limits\"}},
\"title\":\"Resources\",\"type\":\"object\"},\"VpcCni\":{\"additionalProperties
\":false,\"properties\":{\"cri\":{\"$ref\":\"#/definitions/Cri\"},\"env\":
{\"$ref\":\"#/definitions/Env\"},\"resources\":{\"$ref\":\"#/definitions/
Resources\"}},\"title\":\"VpcCni\",\"type\":\"object\"}}}"
}
```

출력은 표준 JSON 스키마입니다.

다음은 위의 스키마와 연동하는 유효한 구성 값의 예시입니다.

```
{
  "resources": {
    "limits": {
      "cpu": "100m"
    }
  }
}
```

다음은 위의 스키마와 연동하는 유효한 구성 값의 예시입니다.

```
  resources:
    limits:
      cpu: 100m
```

4. Amazon EKS 추가 기능을 생성합니다. 다음 명령을 디바이스에 복사합니다. 필요에 따라 명령을 다음과 같이 수정한 다음에 수정한 명령을 실행합니다.

   - *my-cluster*를 클러스터 이름으로 바꿉니다.
   - 생성하려는 이전 단계의 출력에서 반환된 추가 기능 이름으로 *vpc-cni*를 바꿉니다.
   - 사용하려는 이전 단계의 출력에서 반환된 버전으로 *version-number*를 바꿉니다.
   - 추가 기능에서 Kubernetes 서비스 계정과 IAM 역할을 사용하는 경우 계정 ID로 *111122223333*을 바꾸고 생성한 기존 IAM 역할의 이름으로 *role-name*을 바꿉니다. 역할 생성에 대한 지침은 생성 중인 추가 기능의 설명서를 참조하세요. 서비스 계정 역할을 지정하려면 클러스터의 IAM OpenID Connect(OIDC) 제공업체가 있어야 합니다. 클러스터의 해당 제공업체가 이미 있는지 아니면 생성해야 하는지 결정하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요.

추가 기능에서 Kubernetes 서비스 계정과 IAM 역할을 사용하지 않으면 **--service-account-role-arn arn:aws:iam::*111122223333*:role/*role-name*을** 삭제합니다.

- 이들 예시 명령에서는 추가 기능의 기존 자체 관리형 버전(있는 경우)의 `--configuration-values` 옵션을 덮어씁니다. 이 값을 문자열이나 파일 입력과 같은 원하는 구성 값으로 바꾸십시오. 구성 값을 제공하지 않으려면 `--configuration-values` 파라미터를 삭제합니다. AWS CLI로 기존 자체 관리형 추가 기능의 구성을 덮어쓰지 않으려면 *`--resolve-conflicts OVERWRITE`* 옵션을 제거합니다. 옵션을 제거하고 Amazon EKS 추가 기능에 기존 자체 관리형 추가 기능 구성을 덮어써야 하는 경우 Amazon EKS 추가 기능의 생성에 실패하고 충돌 해결에 도움이 되는 오류 메시지가 표시됩니다. 이 옵션을 사용하여 이러한 설정을 덮어쓰기 때문에 이 옵션을 지정하기 전에 Amazon EKS 추가 기능이 관리해야 하는 설정을 관리하지 않는지 확인합니다.

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version version-number \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name --
configuration-values '{"resources":{"limits":{"cpu":"100m"}}}' --resolve-
conflicts OVERWRITE
```

```
aws eks create-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version version-number \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name --
configuration-values 'file://example.yaml' --resolve-conflicts OVERWRITE
```

사용할 수 있는 옵션의 전체 목록은 Amazon EKS 명령줄 참조의 create-addon 부분을 참조하세요. 생성한 추가 기능에 이전 단계의 Owner 열에 나열된 aws-marketplace가 있으면 생성에 실패하고 다음과 같은 오류와 비슷한 오류 메시지가 표시될 수 있습니다.

```
{
    "addon": {
        "addonName": "addon-name",
        "clusterName": "my-cluster",
        "status": "CREATE_FAILED",
        "addonVersion": "version",
        "health": {
            "issues": [
                {
```

```
                    "code": "AddonSubscriptionNeeded",
                    "message": "You are currently not subscribed to this add-
on. To subscribe, visit the AWS Marketplace console, agree to the seller EULA,
 select the pricing type if required, then re-install the add-on"
[...]
```

이전 출력의 오류와 비슷한 오류가 표시되는 경우 이전 단계의 출력에 있는 URL을 방문하여 추가 기능을 구독합니다. 구독한 후 `create-addon` 명령을 다시 실행합니다.

추가 기능 업데이트

Amazon EKS에서는 새 버전이 릴리스되거나 새 Kubernetes 마이너 버전으로 클러스터를 업데이트한 후 추가 기능을 자동으로 업데이트하지 않습니다. 기존 클러스터의 추가 기능을 업데이트하려면 업데이트를 시작해야 합니다. 사용자가 업데이트를 시작하면 Amazon EKS에서 추가 기능을 업데이트합니다. 추가 기능을 업데이트하기 전에 추가 기능에 대한 최신 설명서를 검토합니다. 사용 가능한 추가 기능 목록은 Amazon EKS에서 사용 가능한 Amazon EKS 추가 기능 부분을 참조하세요. 추가 기능에 IAM 역할이 필요한 경우 역할 생성에 대한 자세한 내용은 Amazon EKS에서 사용 가능한 Amazon EKS 추가 기능에서 특정 추가 기능에 대한 세부 정보를 참조하십시오.

`eksctl`, AWS Management Console 또는 AWS CLI를 사용하여 Amazon EKS 추가 기능을 업데이트할 수 있습니다.

eksctl

전제 조건

디바이스 또는 `0.175.0`에 설치된 버전 AWS CloudShell 이상의 `eksctl` 명령줄 도구. `eksctl`을 설치 또는 업그레이드하려면 `eksctl` 설명서에서 Installation을 참조하세요.

`eksctl`을 사용하여 Amazon EKS 추가 기능을 업데이트하는 방법

1. 클러스터에 설치된 현재 추가 기능과 추가 기능 버전을 확인합니다. ***`my-cluster`***를 클러스터 이름으로 바꿉니다.

```
eksctl get addon --cluster my-cluster
```

예제 출력은 다음과 같습니다.

```
NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
coredns     v1.8.7-eksbuild.2    ACTIVE  0
```

```
kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0                 v1.23.8-eksbuild.2
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0                 v1.12.0-
eksbuild.1,v1.11.4-eksbuild.1,v1.11.3-eksbuild.1,v1.11.2-eksbuild.1,v1.11.0-
eksbuild.1
```

클러스터에 있는 추가 기능과 버전에 따라 출력이 다르게 보일 수 있습니다. 이전 예시 출력에서는 클러스터의 기존 추가 기능2가지에 사용할 수 있는 더 새로운 버전이 UPDATE AVAILABLE 열에 있는 것을 볼 수 있습니다.

2. 추가 기능을 업데이트합니다.

   1. 다음 명령을 디바이스에 복사합니다. 필요에 따라 명령을 다음과 같이 수정합니다.

      - *my-cluster*를 클러스터 이름으로 바꿉니다.
      - *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다.
      - 업데이트하려는 이전 단계의 출력에서 반환된 추가 기능 이름으로 *vpc-cni*를 바꿉니다.
      - 사용할 수 있는 최신 버전보다 이전의 버전으로 업데이트하려는 경우에는 사용하려는 이전 단계의 출력에서 반환된 버전 번호로 *latest*를 바꿉니다. 일부 추가 기능에는 권장 버전이 있습니다. 자세한 내용은 업데이트 중인 추가 기능의 설명서를 참조하세요.
      - 추가 기능에서 Kubernetes 서비스 계정과 IAM 역할을 사용하는 경우 계정 ID로 *111122223333*을 바꾸고 생성한 기존 IAM 역할의 이름으로 *role-name*을 바꿉니다. 역할 생성에 대한 지침은 생성 중인 추가 기능의 설명서를 참조하세요. 서비스 계정 역할을 지정하려면 클러스터의 IAM OpenID Connect(OIDC) 제공업체가 있어야 합니다. 클러스터의 해당 제공업체가 이미 있는지 아니면 생성해야 하는지 결정하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요.

        추가 기능에서 Kubernetes 서비스 계정과 IAM 역할을 사용하지 않으면 **serviceAccountRoleARN: arn:aws:iam::*111122223333*:role/*role-name*** 줄을 삭제합니다.
      - *preserve* 옵션은 추가 기능의 기존 값을 보존합니다. 추가 기능 설정에 사용자 지정 값을 설정하고 이 옵션을 사용하지 않는 경우 Amazon EKS에서는 기본값으로 해당 값을 덮어씁니다. 이 옵션을 사용하는 경우에는 추가 기능을 업데이트하기 전에 프로덕션 클러스터에서 비프로덕션 클러스터의 필드 및 값 변경 사항을 테스트하는 것이 좋습니다. 이 값을 overwrite로 변경하면 모든 설정이 Amazon EKS 기본값으로 변경됩니다. 설정에 사용자 지정 값을 설정한 경우 Amazon EKS 기본값으로 해당 값을 덮어쓸 수도 있습니다. 이 값을 none으로 변경하면 Amazon EKS에서는 설정의 값을 변경하지 않지만 업데이트에 실패할 수도 있습니다. 업데이트에 실패하면 충돌 해결에 도움이 되는 오류 메시지가 표시됩니다.

```
cat >update-addon.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: my-cluster
  region: region-code

addons:
- name: vpc-cni
  version: latest
  serviceAccountRoleARN: arn:aws:iam::111122223333:role/role-name
  resolveConflicts: preserve
EOF
```

2. 수정한 명령을 실행하여 `update-addon.yaml` 파일을 생성합니다.
3. 클러스터에 구성 파일을 적용합니다.

```
eksctl update addon -f update-addon.yaml
```

추가 기능 업데이트에 대한 자세한 내용은 `eksctl` 설명서의 Addons(추가 기능)를 참조하세요.

AWS Management Console

AWS Management Console를 사용하여 Amazon EKS 추가 기능을 업데이트하는 방법

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Clusters(클러스터)를 선택한 다음에 추가 기능을 구성할 클러스터의 이름을 선택합니다.
3. 추가 기능(Add-ons) 탭을 선택합니다.
4. 추가 기능 상자의 오른쪽 상단에 있는 상자를 선택한 다음에 Edit(편집)를 선택합니다.
5. Configure ***name of addon***(추가 기능 이름 구성) 페이지에서:
   - 사용할 Version(버전)을 선택합니다. 추가 기능에 권장 버전이 있을 수도 있습니다. 자세한 내용은 업데이트 중인 추가 기능의 설명서를 참조하세요.
   - IAM 역할 선택의 경우 노드의 IAM 역할(설정되지 않음) 또는 추가 기능에 사용하려고 생성한 기존 역할을 선택할 수 있습니다. 선택할 역할이 없는 경우에는 기존 역할이 없는 것

입니다. 선택하는 옵션과 관계없이 IAM 정책을 생성하고 역할에 연결하려고 생성 중인 추가 기능의 설명서를 참조하세요. IAM 역할을 선택하려면 클러스터에 대한 IAM OpenID Connect(OIDC) 제공업체가 있어야 합니다. 클러스터의 해당 제공업체가 이미 있는지 아니면 생성해야 하는지 결정하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요.

- `Code editor`의 경우 추가 기능에 특정 구성 정보를 입력합니다. 자세한 내용은 업데이트 중인 추가 기능의 설명서를 참조하세요.
- 충돌 해결 방법의 옵션 중 하나를 선택합니다. 추가 기능 설정에 사용자 지정 값을 설정한 경우 Preserve(보존) 옵션이 좋습니다. 이 옵션을 선택하지 않으면 Amazon EKS에서 기본값으로 해당 값을 덮어씁니다. 이 옵션을 사용하는 경우에는 추가 기능을 업데이트하기 전에 프로덕션 클러스터에서 비프로덕션 클러스터의 필드 및 값 변경 사항을 테스트하는 것이 좋습니다.

6. 업데이트를 선택합니다.

AWS CLI

전제 조건

AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.

AWS CLI를 사용하여 Amazon EKS 추가 기능을 업데이트하는 방법

1. 설치한 추가 기능 목록을 참조합니다. *`my-cluster`*를 클러스터 이름으로 바꿉니다.

```
aws eks list-addons --cluster-name my-cluster
```

예제 출력은 다음과 같습니다.

```
{
    "addons": [
        "coredns",
```

```
        "kube-proxy",
        "vpc-cni"
    ]
}
```

2. 업데이트할 추가 기능의 현재 버전을 조회합니다. *my-cluster*를 클러스터 이름으로 바꾸고 업데이트할 추가 기능의 이름으로 *vpc-cni*를 바꿉니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni --query
 "addon.addonVersion" --output text
```

예제 출력은 다음과 같습니다.

```
v1.10.4-eksbuild.1
```

3. 클러스터의 버전에 어떤 추가 기능의 버전을 사용할 수 있는지 확인할 수 있습니다. *1.29*를 클러스터의 버전으로 바꾸고 업데이트할 추가 기능의 이름으로 *vpc-cni*를 바꿉니다.

```
aws eks describe-addon-versions --kubernetes-version 1.29 --addon-name vpc-cni \
    --query 'addons[].addonVersions[].{Version: addonVersion, Defaultversion:
 compatibilities[0].defaultVersion}' --output table
```

예제 출력은 다음과 같습니다.

```
-----------------------------------------
|          DescribeAddonVersions         |
+-----------------+----------------------+
| Defaultversion  |       Version        |
+-----------------+----------------------+
|  False          |  v1.12.0-eksbuild.1  |
|  True           |  v1.11.4-eksbuild.1  |
|  False          |  v1.10.4-eksbuild.1  |
|  False          |  v1.9.3-eksbuild.1   |
+-----------------+----------------------+
```

기본적으로 `Defaultversion` 열에 `True`가 있는 버전이 추가 기능을 생성한 버전입니다.

4. 추가 기능을 업데이트합니다. 다음 명령을 디바이스에 복사합니다. 필요에 따라 명령을 다음과 같이 수정한 다음에 수정한 명령을 실행합니다.

   - *my-cluster*를 클러스터 이름으로 바꿉니다.

- 업데이트하려는 이전 단계의 출력에서 반환된 추가 기능 이름으로 *vpc-cni*를 바꿉니다.
- 업데이트하려는 이전 단계의 출력에서 반환된 버전으로 *version-number*를 바꿉니다. 일부 추가 기능에는 권장 버전이 있습니다. 자세한 내용은 업데이트 중인 추가 기능의 설명서를 참조하세요.
- 추가 기능에서 Kubernetes 서비스 계정과 IAM 역할을 사용하는 경우 계정 ID로 *111122223333*을 바꾸고 생성한 기존 IAM 역할의 이름으로 *role-name*을 바꿉니다. 역할 생성에 대한 지침은 생성 중인 추가 기능의 설명서를 참조하세요. 서비스 계정 역할을 지정하려면 클러스터의 IAM OpenID Connect(OIDC) 제공업체가 있어야 합니다. 클러스터의 해당 제공업체가 이미 있는지 아니면 생성해야 하는지 결정하려면 클러스터에 대한 IAM OIDC 공급자 생성 섹션을 참조하세요.

  추가 기능에서 Kubernetes 서비스 계정과 IAM 역할을 사용하지 않으면 **serviceAccountRoleARN: arn:aws:iam::*111122223333*:role/*role-name*** 줄을 삭제합니다.
- **--resolve-conflicts** *PRESERVE*(보존) 옵션에서는 추가 기능의 기존 값을 보존합니다. 추가 기능 설정에 사용자 지정 값을 설정하고 이 옵션을 사용하지 않는 경우 Amazon EKS에서는 기본값으로 해당 값을 덮어씁니다. 이 옵션을 사용하는 경우에는 추가 기능을 업데이트하기 전에 프로덕션 클러스터에서 비프로덕션 클러스터의 필드 및 값 변경 사항을 테스트하는 것이 좋습니다. 이 값을 overwrite로 변경하면 모든 설정이 Amazon EKS 기본값으로 변경됩니다. 설정에 사용자 지정 값을 설정한 경우 Amazon EKS 기본값으로 해당 값을 덮어쓸 수도 있습니다. 이 값을 none으로 변경하면 Amazon EKS에서는 설정의 값을 변경하지 않지만 업데이트에 실패할 수도 있습니다. 업데이트에 실패하면 충돌 해결에 도움이 되는 오류 메시지가 표시됩니다.

- 모든 사용자 지정 구성을 제거하려면 *--configuration-values '{}'*옵션을 사용하여 업데이트를 수행하십시오. 그러면 모든 사용자 지정 구성이 기본값으로 되돌아갑니다. 사용자 지정 구성을 변경하지 않으려는 경우에는 *--configuration-values* 플래그를 제공하지 마세요. 사용자 지정 구성을 조정하려는 경우에는 *{}* 부분을 새 파라미터로 바꿉니다. 파라미터 목록을 참조하려면 추가 기능 생성 부분의 구성 스키마 보기 단계를 참조하세요.

```
aws eks update-addon --cluster-name my-cluster --addon-name vpc-cni --addon-
version version-number \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name --
configuration-values '{}' --resolve-conflicts PRESERVE
```

5. 업데이트 상태를 확인합니다. *my-cluster*를 클러스터 이름으로 바꾸고 업데이트 중인 추가 기능의 이름으로 *vpc-cni*를 바꿉니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name vpc-cni
```

예제 출력은 다음과 같습니다.

```
{
    "addon": {
        "addonName": "vpc-cni",
        "clusterName": "my-cluster",
        "status": "UPDATING",
[...]
```

상태가 ACTIVE라면 업데이트가 완료된 것입니다.

추가 기능 삭제

Amazon EKS 추가 기능을 삭제하는 경우:

- 추가 기능에서 제공되는 기능의 가동 중지 시간이 없습니다.
- 추가 기능에 연결된 IAM 역할이 있는 경우 IAM 역할은 제거되지 않습니다.
- Amazon EKS에서는 추가 기능의 설정 관리를 중지합니다.
- 새 버전이 사용할 수 있을 때 콘솔의 알림 표시가 중지됩니다.
- AWS 도구 또는 API를 사용하여 추가 기능을 업데이트할 수 없습니다.
- 자체 관리할 수 있도록 클러스터에 추가 기능 소프트웨어 남겨두기를 선택하거나 클러스터에서 추가 기능 소프트웨어를 제거할 수 있습니다. 추가 기능에서 제공되는 기능에 종속되는 리소스가 클러스터에 없는 경우에만 클러스터에서 추가 기능 소프트웨어을 제거해야 합니다.

`eksctl`, AWS Management Console 또는 AWS CLI를 사용하여 클러스터에서 Amazon EKS 추가 기능을 삭제할 수 있습니다.

eksctl

전제 조건

디바이스 또는 `0.175.0`에 설치된 버전 AWS CloudShell 이상의 `eksctl` 명령줄 도구. `eksctl`을 설치 또는 업그레이드하려면 `eksctl` 설명서에서 Installation을 참조하세요.

eksctl을 사용하여 Amazon EKS 추가 기능을 삭제하는 방법

1. 클러스터에 설치된 현재 추가 기능을 확인합니다. *my-cluster*를 클러스터 이름으로 바꿉니다.

```
eksctl get addon --cluster my-cluster
```

예제 출력은 다음과 같습니다.

```
NAME        VERSION              STATUS  ISSUES  IAMROLE  UPDATE AVAILABLE
coredns     v1.8.7-eksbuild.2    ACTIVE  0
kube-proxy  v1.23.7-eksbuild.1   ACTIVE  0
vpc-cni     v1.10.4-eksbuild.1   ACTIVE  0
[...]
```

클러스터에 있는 추가 기능과 버전에 따라 출력이 다르게 보일 수 있습니다.

2. 추가 기능을 삭제합니다. *my-cluster*를 클러스터의 이름으로 바꾸고 제거하려는 이전 단계의 출력에서 반환된 추가 기능의 이름으로 *name-of-add-on*을 바꿉니다. ***--preserve*** 옵션을 제거하면 Amazon EKS에서 더는 추가 기능을 관리하지 않을 뿐만 아니라 클러스터에서 추가 기능 소프트웨어가 제거됩니다.

```
eksctl delete addon --cluster my-cluster --name name-of-addon --preserve
```

AWS Management Console

AWS Management Console를 사용하여 Amazon EKS 추가 기능을 삭제하는 방법

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 Clusters(클러스터)를 선택한 다음에 Amazon EKS 추가 기능을 제거하려는 클러스터의 이름을 선택합니다.
3. 추가 기능(Add-ons) 탭을 선택합니다.
4. 추가 기능 상자의 오른쪽 상단에서 확인란을 선택한 다음에 Remove(제거)를 선택합니다. Amazon EKS에서 추가 기능의 설정 관리를 중지하되 추가 기능의 모든 설정을 자체 관리할 수 있도록 클러스터에 추가 기능 소프트웨어를 유지하려는 경우 Preserve on the cluster(클러스터에 보존)를 선택합니다. 추가 기능 이름을 입력한 다음에Remove(제거)를 선택합니다.

AWS CLI

전제 조건

디바이스 또는 `0.175.0`에 설치된 버전 AWS CloudShell 이상의 `eksctl` 명령줄 도구. `eksctl`을 설치 또는 업그레이드하려면 `eksctl` 설명서에서 Installation을 참조하세요.

AWS CLI를 사용하여 Amazon EKS 추가 기능을 삭제하는 방법

1. 설치한 추가 기능 목록을 참조합니다. *my-cluster*를 클러스터 이름으로 바꿉니다.

```
aws eks list-addons --cluster-name my-cluster
```

예제 출력은 다음과 같습니다.

```
{
    "addons": [
        "coredns",
        "kube-proxy",
        "vpc-cni",
        "name-of-addon"
    ]
}
```

2. 설치한 추가 기능을 삭제합니다. *my-cluster*를 클러스터 이름으로 바꾸고 제거할 추가 기능의 이름으로 *name-of-add-on*을 바꿉니다. ***--preserve***를 제거하면 클러스터에서 추가 기능 소프트웨어가 제거됩니다.

```
aws eks delete-addon --cluster-name my-cluster --addon-name name-of-addon --
preserve
```

축약한 예시 출력은 다음과 같습니다.

```
{
    "addon": {
        "addonName": "name-of-add-on",
        "clusterName": "my-cluster",
        "status": "DELETING",
[...]
```

3. 삭제 상태를 확인합니다. *my-cluster*를 클러스터 이름으로 바꾸고 제거할 추가 기능의 이름으로 *name-of-addon*를 바꿉니다.

```
aws eks describe-addon --cluster-name my-cluster --addon-name name-of-addon
```

추가 기능이 삭제된 후의 예시 출력은 다음과 같습니다.

```
An error occurred (ResourceNotFoundException) when calling the DescribeAddon
 operation: No addon: name-of-addon found in cluster: my-cluster
```

추가 기능 버전 호환성 검색

describe-addon-verisions API를 사용하여 EKS 추가 기능의 사용 가능한 버전과 각 추가 기능 버전에서 지원하는 Kubernetes 버전을 나열합니다.

추가 기능 버전 호환성 검색(AWS CLI)

1. `aws sts get-caller-identity` 명령으로 AWS CLI가 설치되어 작동하고 있는지 확인합니다. 이 명령이 작동하지 않는 경우 AWS CLI 시작 방법을 알아보십시오.
2. 버전 호환성 정보를 검색하려는 추가 기능의 이름(예: `amazon-cloudwatch-observability`)을 확인합니다.
3. 클러스터의 Kubernetes 버전(예: `1.28`)을 확인합니다.
4. AWS CLI를 사용하여 클러스터의 Kubernetes 버전과 호환되는 추가 기능 버전을 검색합니다.

```
aws eks describe-addon-versions --addon-name amazon-cloudwatch-observability --
kubernetes-version 1.29
```

예제 출력은 다음과 같습니다.

```
{
    "addons": [
        {
            "addonName": "amazon-cloudwatch-observability",
            "type": "observability",
            "addonVersions": [
                {
                    "addonVersion": "v1.5.0-eksbuild.1",
                    "architecture": [
```

```
                        "amd64",
                        "arm64"
                    ],
                    "compatibilities": [
                        {
                            "clusterVersion": "1.28",
                            "platformVersions": [
                                "*"
                            ],
                            "defaultVersion": true
                        }
                    ],
[...]
```

이 출력은 추가 기능 버전 `v1.5.0-eksbuild.1`이 Kubernetes 클러스터 버전 `1.28`과 호환됨을 보여줍니다.

Kubernetes 필드 관리

Amazon EKS 추가 기능은 표준 모범 사례 구성을 사용하여 클러스터에 설치됩니다. Amazon EKS 추가 기능을 클러스터에 추가하는 방법에 대한 자세한 내용은 Amazon EKS 추가 기능 섹션을 참조하세요.

고급 기능을 사용하도록 Amazon EKS 추가 기능의 구성을 사용자 지정하려 할 수 있습니다. Amazon EKS는 Kubernetes 서버 측 적용 기능을 사용하여 Amazon EKS가 관리하지 않는 설정에 대한 구성을 덮어쓰지 않고도 Amazon EKS의 추가 기능을 관리할 수 있습니다. 자세한 내용을 알아보려면 Kubernetes 설명서의 서버 측 적용이란?을 참조하세요. 이를 위해 Amazon EKS는 설치하는 모든 추가 기능에 대해 최소한의 필드 세트를 관리합니다. Amazon EKS가 관리하지 않는 모든 필드 또는 다른 Kubernetes 컨트롤 플레인 프로세스(예: `kube-controller-manager`)를 문제 없이 수정할 수 있습니다.

> **Important**
>
> Amazon EKS가 관리하는 필드를 수정하면 Amazon EKS가 추가 기능을 관리할 수 없으며, 추가 기능이 업데이트될 때 변경 내용이 덮어써질 수 있습니다.

필드 관리 상태 보기

kubectl을 사용하여 Amazon EKS 추가 기능을 위해 Amazon EKS가 관리하는 필드를 볼 수 있습니다.

필드의 관리 상태를 보려면

1. 조사할 추가 기능을 확인합니다. 클러스터에 배포된 모든 deployments 및 DaemonSets를 보려면 Kubernetes 리소스 보기 섹션을 참조하세요.
2. 다음 명령을 실행하여 추가 기능에 대한 관리형 필드를 확인합니다.

```
kubectl get type/add-on-name -n add-on-namespace -o yaml
```

예를 들어 다음 명령을 사용하여 CoreDNS 추가 기능에 대한 관리형 필드를 볼 수 있습니다.

```
kubectl get deployment/coredns -n kube-system -o yaml
```

필드 관리는 반환된 출력의 다음 섹션에 나열되어 있습니다.

```
[...]
managedFields:
  - apiVersion: apps/v1
    fieldsType: FieldsV1
    fieldsV1:
[...]
```

> **Note**
>
> 출력에 managedFields가 표시되지 않으면 명령에 **--show-managed-fields**를 추가하고 다시 실행합니다. 사용 중인 kubectl 버전에 따라 관리형 필드가 기본적으로 반환되는지 여부가 결정됩니다.

Kubernetes API의 필드 관리 구문 이해

Kubernetes 객체에 대한 세부 정보를 볼 때 관리형 및 비관리형 필드 모두 출력에 반환됩니다. 관리형 필드는 다음 유형 중 하나일 수 있습니다.

- 완전관리형(Fully managed) - 필드의 모든 키를 Amazon EKS에서 관리합니다. 어떤 값이든 수정하면 충돌이 발생합니다.
- 부분관리형(Partially managed) - 필드의 일부 키를 Amazon EKS에서 관리합니다. Amazon EKS에서 명시적으로 관리하는 키를 수정하면 충돌이 발생합니다.

두 유형의 필드 모두 `manager: eks`를 사용하여 태그가 지정됩니다.

각 키는 항상 빈 세트에 매핑되는 필드 자체를 나타내는 `.`이거나, 하위 필드 또는 항목을 나타내는 문자열입니다. 필드 관리에 대한 출력은 다음과 같은 유형의 선언으로 구성됩니다.

- `f:`*`name`*, 여기서 *`name`*은 목록의 필드 이름입니다.
- `k:`*`keys`*, 여기서 *`keys`*는 목록 항목의 필드 맵입니다.
- `v:`*`value`*, 여기서 *`value`*는 목록 항목의 정확한 JSON 형식 값입니다.
- `i:`*`index`*, 여기서 *`index`*는 목록에서 항목의 위치입니다.

CoreDNS 추가 기능에 대한 다음 출력 부분은 이전 선언을 보여줍니다.

- 완전관리형 필드(Fully managed fields) - 관리형 필드에 `f:`(필드)가 지정되고 `k:`(키)가 지정되지 않은 경우 전체 필드가 관리됩니다. 이 필드의 값을 수정하면 충돌이 발생합니다.

  다음 출력에서는 `coredns`라는 컨테이너가 `eks`에 의해 관리되는 것을 확인할 수 있습니다. `args`, `image` 및 `imagePullPolicy` 하위 필드도 `eks`에서 관리합니다. 이러한 필드의 값을 수정하면 충돌이 발생합니다.

  ```
  [...]
  f:containers:
    k:{"name":"coredns"}:
    .: {}
    f:args: {}
    f:image: {}
    f:imagePullPolicy: {}
  [...]
  manager: eks
  [...]
  ```

- 부분관리형 필드(Partially managed fields) - 관리형 키에 값이 지정되면 해당 필드에 대해 선언된 키가 관리됩니다. 지정된 키를 수정하면 충돌이 발생합니다.

다음 출력에서는 eks가 name 키로 설정된 config-volume 및 tmp 볼륨을 관리하는 것을 확인할 수 있습니다.

```
[...]
f:volumes:
  k:{"name":"config-volume"}:
    .: {}
    f:configMap:
      f:items: {}
      f:name: {}
    f:name: {}
  k:{"name":"tmp"}:
    .: {}
    f:name: {}
[...]
manager: eks
[...]
```

- 부분관리형 필드에 키 추가(Adding keys to partially managed fields) - 특정 키 값만 관리되는 경우, 충돌을 일으키지 않고 인수와 같은 추가 키를 필드에 안전하게 추가할 수 있습니다. 다른 키를 추가하는 경우 먼저 필드가 관리되지 않는지 확인합니다. 관리되는 값을 추가하거나 수정하면 충돌이 발생합니다.

  다음 출력에서는 name 키와 name 필드가 모두 관리되는 것을 알 수 있습니다. 컨테이너 이름을 추가하거나 수정하면 이 관리 키와 충돌이 발생합니다.

```
[...]
f:containers:
  k:{"name":"coredns"}:
[...]
    f:name: {}
[...]
manager: eks
[...]
```

배포 중 컨테이너 이미지 확인

AWS Signer를 사용하고 배포 시 서명된 컨테이너 이미지를 확인하려는 경우 다음 솔루션 중 하나를 사용할 수 있습니다.

- Gatekeeper 및 Ratify – Gatekeeper를 승인 컨트롤러로 사용하고 AWS Signer 플러그인으로 구성된 Ratify를 서명 검증을 위한 웹 후크로 사용합니다.
- Kyverno –서명 검증을 위해 AWS Signer 플러그인으로 구성된 Kubernetes 정책 엔진입니다.

> **Note**
>
> 컨테이너 이미지 서명을 확인하기 전에 선택한 승인 컨트롤러의 필요에 따라 Notation 트러스트 스토어 및 신뢰 정책을 구성합니다.

Elastic Fabric Adapter를 사용한 기계 학습 훈련

이 주제에서는 Amazon EKS 클러스터에 배포된 Pods와 Elastic Fabric Adapter(EFA)를 통합하는 방법에 대해 설명합니다. Elastic Fabric Adapter(EFA)는 Amazon EC2 인스턴스용 네트워크 인터페이스로, AWS에서 대규모로 높은 수준의 노드 간 통신을 필요로 하는 애플리케이션을 실행할 수 있도록 지원합니다. 맞춤형으로 구축된 운영 체제 바이패스 하드웨어 인터페이스는 인스턴스 간 통신의 성능을 향상시켜 애플리케이션을 확장하는 데 매우 중요합니다. EFA를 사용하면 메시지 전달 인터페이스(MPI)를 사용하는 고성능 컴퓨팅(HPC) 애플리케이션과 NVIDIA Collective Communications Library(NCCL)를 사용하는 기계 학습(ML) 애플리케이션을 수천 개의 CPU 또는 GPU로 확장할 수 있습니다. 따라서 온프레미스 HPC 클러스터의 애플리케이션 성능을 AWS 클라우드의 온디맨드 탄력성과 유연성을 통해 실현할 수 있습니다. Amazon EKS 클러스터에서 실행되는 애플리케이션과 EFA를 통합하면 클러스터에 인스턴스를 추가하지 않고도 대규모 분산 훈련 워크로드를 완료하는 데 걸리는 시간을 줄일 수 있습니다. EFA에 대한 자세한 내용은 Elastic Fabric Adapter(EFA) 섹션을 참조하세요.

이 주제에서 설명하는 EFA 플러그 인은 클라우드 분산 기계 학습의 최신 기술을 보여주는 Amazon EC2 P4d 인스턴스를 완벽하게 지원합니다. 각 `p4d.24xlarge` 인스턴스에는 8개의 NVIDIA A100 GPU가 있고 EFA를 통해 400Gbps GPUDirectRDMA를 지원합니다. GPUDirectRDMA를 사용하면 CPU 바이패스를 통해 노드 간에 직접 GPU와 GPU 간 통신이 가능하므로 집단적 통신 대역폭이 증가하고 대기 시간이 줄어듭니다. P4d 인스턴스와 Amazon EKS 및 EFA의 통합은 분산 기계 학습 훈련을 위해 최고 성능의 Amazon EC2 컴퓨팅 인스턴스를 활용할 수 있는 완벽한 방법을 제공합니다.

필수 조건

- 기존 Amazon EKS 클러스터. 기존 클러스터가 없는 경우 Amazon EKS 시작하기 가이드 중 하나를 사용하여 생성합니다. 노드를 배포할 수 있는 충분한 IP 주소가 있는 프라이빗 서브넷이 하나 이상 있는 VPC에 클러스터를 배포해야 합니다. 프라이빗 서브넷에는 NAT 게이트웨이와 같은 외부 디바이스에서 제공하는 아웃바운드 인터넷 액세스가 있어야 합니다.

`eksctl`을 사용하여 노드 그룹을 생성하려는 경우 `eksctl`이 클러스터를 생성할 수도 있습니다.

- AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.
- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 `kubectl` 설치 또는 업데이트 부분을 참조하세요.
- `p4d.24xlarge`와 같이 여러 Elastic Fabric Adapter를 지원하는 워커 노드를 시작하기 전에 Amazon VPC CNI plugin for Kubernetes 버전 `1.7.10` 이상이 설치되어 있어야 합니다. Amazon VPC CNI plugin for Kubernetes 버전 업데이트에 대한 자세한 내용은 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 섹션을 참조하세요.

노드 그룹 생성

다음 절차를 따르면 EFA 인터페이스 및 GPUDirect RDMA를 사용하여 `p4d.24xlarge` 지원 노드 그룹으로 노드 그룹을 생성하고, EFA를 사용하여 다중 노드 NCCL 성능에 대한 예시 NVIDIA Collective Communications Library(NCCL) 테스트를 실행하는 데 도움이 됩니다. 이 예제는 EFA를 사용하는 Amazon EKS에 대한 분산 딥 러닝 훈련용 템플릿으로 사용할 수 있습니다.

1. 노드를 배포하려는 AWS 리전에서 사용할 수 있는 EFA를 지원하는 Amazon EC2 인스턴스 유형을 결정합니다. *`region-code`*를 노드 그룹을 배포할 AWS 리전으로 바꿉니다.

   ```
   aws ec2 describe-instance-types --region region-code --filters Name=network-
   info.efa-supported,Values=true \
       --query "InstanceTypes[*].[InstanceType]" --output text
   ```

   노드를 배포할 때 배포하려는 인스턴스 유형은 클러스터가 속한 AWS 리전에서 사용할 수 있어야 합니다.

2. 배포하려는 인스턴스 유형이 사용 가능한 가용 영역을 결정합니다. 이 자습서에서는 `p4d.24xlarge` 인스턴스 유형이 사용되며 이전 단계에서 지정한 AWS 리전에 대한 출력으로 반환되어야 합니다. 프로덕션 클러스터에 노드를 배포할 때 *`p4d.24xlarge`*를 이전 단계에서 반환된 인스턴스 유형으로 바꿉니다.

```
aws ec2 describe-instance-type-offerings --region region-code --location-type
 availability-zone --filters Name=instance-type,Values=p4d.24xlarge \
    --query 'InstanceTypeOfferings[*].Location' --output text
```

예제 출력은 다음과 같습니다.

```
us-west-2a    us-west-2c    us-west-2b
```

이후 단계에서 사용하기 위해 반환된 가용 영역을 기록해 둡니다. 클러스터에 노드를 배포할 때 VPC에는 출력에 반환된 가용 영역 중 하나에 사용 가능한 IP 주소가 있는 서브넷이 있어야 합니다.

3. `eksctl` 또는 AWS CLI 및 AWS CloudFormation을 사용하여 노드 그룹을 생성합니다.

eksctl

전제 조건

디바이스 또는 `0.175.0`에 설치된 버전 AWS CloudShell 이상의 `eksctl` 명령줄 도구. `eksctl`을 설치 또는 업그레이드하려면 `eksctl` 설명서에서 Installation을 참조하세요.

1. 다음 콘텐츠를 *`efa-cluster.yaml`*라는 파일에 복사합니다. *`example values`*을(를) 사용자의 값으로 교체합니다. *`p4d.24xlarge`*을(를) 다른 인스턴스로 바꿀 수 있지만, 그렇게 할 경우 `availabilityZones`의 값이 1단계에서 인스턴스 유형에 대해 반환된 가용 영역이어야 합니다.

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-efa-cluster
  region: region-code
  version: "1.XX"

iam:
  withOIDC: true
```

```
availabilityZones: ["us-west-2a", "us-west-2c"]

managedNodeGroups:
  - name: my-efa-ng
    instanceType: p4d.24xlarge
    minSize: 1
    desiredCapacity: 2
    maxSize: 3
    availabilityZones: ["us-west-2a"]
    volumeSize: 300
    privateNetworking: true
    efaEnabled: true
```

2. 기존 클러스터에 관리형 노드 그룹을 생성합니다.

```
eksctl create nodegroup -f efa-cluster.yaml
```

기존 클러스터가 없는 경우 다음 명령을 실행하여 클러스터와 노드 그룹을 생성할 수 있습니다.

```
eksctl create cluster -f efa-cluster.yaml
```

> **Note**
>
> 이 예제에 사용된 인스턴스 유형에는 GPU가 있으므로 `eksctl`은 각 인스턴스에 NVIDIA Kubernetes 디바이스 플러그인을 자동으로 설치합니다.

AWS CLI and AWS CloudFormation

EFA 네트워킹에는 EFA용 보안 그룹 생성, Amazon EC2 배치 그룹 생성, 하나 이상의 EFA 인터페이스를 지정하는 시작 템플릿 생성 등의 몇 가지 요구 사항이 있으며, Amazon EC2 사용자 데이터의 일부로 EFA 드라이버를 설치하는 것을 포함합니다. EFA 요구 사항에 대해 자세히 알아보려면 Linux 인스턴스용 Amazon EC2 사용 설명서에서 EFA 및 MPI 시작하기 섹션을 참조하세요. 다음 단계에서는 이 모든 것을 생성합니다. 모든 [*## #(example values)*]을 사용자의 값으로 바꿉니다.

1. 이후 단계에서 사용되는 몇 가지 변수를 설정합니다. 모든 *example values*를 실제 값으로 교체합니다. *my-cluster*을 기존 클러스터 이름으로 바꿉니다.

node_group_resources_name의 값은 나중에 AWS CloudFormation 스택을 생성하는 데 사용됩니다. node_group_name의 값은 나중에 클러스터에 노드 그룹을 생성하는 데 사용됩니다.

```
cluster_name="my-cluster"
cluster_region="region-code"
node_group_resources_name="my-efa-nodegroup-resources"
node_group_name="my-efa-nodegroup"
```

2. VPC에서 배포하려는 인스턴스 유형과 동일한 가용 영역에 있는 프라이빗 서브넷을 식별합니다.

   a. 클러스터의 버전을 검색하여 이후 단계에서 사용할 수 있도록 변수에 저장합니다.

   ```
   cluster_version=$(aws eks describe-cluster \
       --name $cluster_name \
       --query "cluster.version" \
       --output text)
   ```

   b. 클러스터가 있는 VPC ID를 검색하여 이후 단계에서 사용할 수 있도록 변수에 저장합니다.

   ```
   vpc_id=$(aws eks describe-cluster \
       --name $cluster_name \
       --query "cluster.resourcesVpcConfig.vpcId" \
       --output text)
   ```

   c. 클러스터의 제어 영역 보안 그룹의 ID를 검색하여 이후 단계에서 사용할 수 있도록 변수에 저장합니다.

   ```
   control_plane_security_group=$(aws eks describe-cluster \
       --name $cluster_name \
       --query "cluster.resourcesVpcConfig.clusterSecurityGroupId" \
       --output text)
   ```

   d. 1단계에서 반환된 가용 영역에 있는 VPC 서브넷 ID 목록을 가져옵니다.

   ```
   aws ec2 describe-subnets \
       --filters "Name=vpc-id,Values=$vpc_id" "Name=availability-
   zone,Values=us-west-2a" \
       --query 'Subnets[*].SubnetId' \
       --output text
   ```

반환되는 출력이 없으면 1단계에서 반환된 다른 가용 영역을 시도합니다. 1단계에서 반환된 가용 영역에 서브넷이 없는 경우 1단계에서 반환된 가용 영역에 서브넷을 생성해야 합니다. VPC에 다른 서브넷을 생성할 공간이 없는 경우 VPC에 CIDR 블록을 추가하고 새 CIDR 블록에 서브넷을 생성하거나 새 VPC에 새 클러스터를 생성할 수 있습니다.

e. 서브넷의 라우팅 테이블을 확인하여 서브넷이 프라이빗 서브넷인지 확인합니다.

```
aws ec2 describe-route-tables \
    --filter Name=association.subnet-id,Values=subnet-0d403852a65210a29 \
    --query "RouteTables[].Routes[].GatewayId" \
    --output text
```

예제 출력은 다음과 같습니다.

```
local
```

출력이 `local` `igw-02adc64c1b72722e2`이면 서브넷은 퍼블릭 서브넷입니다. 1단계에서 반환된 가용 영역에서 프라이빗 서브넷을 선택해야 합니다. 프라이빗 서브넷을 식별했으면 이후 단계에서 사용할 수 있도록 ID를 적어 둡니다.

f. 이후 단계에서 사용할 수 있도록 이전 단계의 프라이빗 서브넷 ID를 사용하여 변수를 설정합니다.

```
subnet_id=your-subnet-id
```

3. AWS CloudFormation 템플릿을 다운로드합니다.

```
curl -O https://raw.githubusercontent.com/aws-samples/aws-efa-eks/main/
cloudformation/efa-p4d-managed-nodegroup.yaml
```

4. 다음 텍스트를 컴퓨터에 복사합니다. *p4d.24xlarge*을(를) 1단계의 인스턴스 유형으로 바꿉니다. *subnet-0d403852a65210a29*을(를) 2.b.v 단계에서 식별한 프라이빗 서브넷의 ID로 바꿉니다. *path-to-downloaded-cfn-template*을(를) 이전 단계에서 다운로드한 `efa-p4d-managed-nodegroup.yaml`의 경로로 바꿉니다. *your-public-key-name*을(를) 퍼블릭 키 이름으로 바꿉니다. 다 바꾼 후 수정된 명령을 실행합니다.

```
aws cloudformation create-stack \
    --stack-name ${node_group_resources_name} \
    --capabilities CAPABILITY_IAM \
    --template-body file://path-to-downloaded-cfn-template \
```

```
    --parameters \
       ParameterKey=ClusterName,ParameterValue=${cluster_name} \
       ParameterKey=ClusterControlPlaneSecurityGroup,ParameterValue=
${control_plane_security_group} \
       ParameterKey=VpcId,ParameterValue=${vpc_id} \
       ParameterKey=SubnetId,ParameterValue=${subnet_id} \
       ParameterKey=NodeGroupName,ParameterValue=${node_group_name} \
       ParameterKey=NodeImageIdSSMParam,ParameterValue=/aws/service/eks/
optimized-ami/${cluster_version}/amazon-linux-2-gpu/recommended/image_id \
       ParameterKey=KeyName,ParameterValue=your-public-key-name \
       ParameterKey=NodeInstanceType,ParameterValue=p4d.24xlarge
```

5. 이전 단계에서 배포한 스택이 배포되는 시기를 결정합니다.

```
aws cloudformation wait stack-create-complete --stack-name
 $node_group_resources_name
```

이전 명령의 출력은 없지만 스택이 생성될 때까지 쉘 프롬프트가 반환되지 않습니다.

6. 이전 단계에서 AWS CloudFormation 스택에 의해 생성된 리소스를 사용하여 노드 그룹을 생성합니다.

   a. 배포된 AWS CloudFormation 스택에서 정보를 검색하여 변수에 저장합니다.

```
node_instance_role=$(aws cloudformation describe-stacks \
    --stack-name $node_group_resources_name \
    --query='Stacks[].Outputs[?OutputKey==`NodeInstanceRole`].OutputValue'
 \
    --output text)
launch_template=$(aws cloudformation describe-stacks \
    --stack-name $node_group_resources_name \
    --query='Stacks[].Outputs[?OutputKey==`LaunchTemplateID`].OutputValue'
 \
    --output text)
```

   b. 이전 단계에서 생성된 시작 템플릿과 노드 IAM 역할을 사용하는 관리형 노드 그룹을 생성합니다.

```
aws eks create-nodegroup \
    --cluster-name $cluster_name \
    --nodegroup-name $node_group_name \
    --node-role $node_instance_role \
    --subnets $subnet_id \
```

```
    --launch-template id=$launch_template,version=1
```

c. 노드가 생성되었는지 확인합니다.

```
aws eks describe-nodegroup \
   --cluster-name ${cluster_name} \
   --nodegroup-name ${node_group_name} | jq -r .nodegroup.status
```

이전 명령에서 반환된 상태가 ACTIVE가 될 때까지 더 이상 진행하지 마세요. 노드가 준비되려면 몇 분 정도 걸릴 수 있습니다.

7. GPU 인스턴스 유형을 선택한 경우 Kubernetes용 NVIDIA 디바이스 플러그인을 배포해야 합니다. 다음 명령을 실행하기 전에 *vX.X.X*을(를) 원하는 NVIDIA/k8s-device-plugin 버전으로 교체합니다.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-
plugin/vX.X.X/nvidia-device-plugin.yml
```

4. EFA Kubernetes 디바이스 플러그 인을 배포합니다.

EFA Kubernetes 디바이스 플러그 인은 EFA 인터페이스를 Kubernetes에 할당 가능한 리소스로 감지하고 제공합니다. 애플리케이션은 CPU 및 메모리처럼 Pod 요청 사양에 확장 리소스 유형 vpc.amazonaws.com/efa을 사용할 수 있습니다. 자세한 내용은 Kubernetes 설명서의 확장 리소스 사용을 참조하세요. 요청되면 플러그인은 자동으로 EFA 인터페이스를 할당하고 Pod에 탑재합니다. 디바이스 플러그인을 사용하면 EFA 설정이 간소화되고 권한 있는 모드에서 Pod를 실행할 필요가 없습니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws-samples/aws-efa-eks/main/
manifest/efa-k8s-device-plugin.yml
```

(선택 사항) 샘플 EFA 호환 애플리케이션을 배포합니다.

Kubeflow MPI 연산자 배포

NCCL 테스트를 위해 Kubeflow MPI 연산자를 적용할 수 있습니다. MPI 연산자를 사용하면 Kubernetes에서 Allreduce 스타일의 분산 훈련을 쉽게 실행할 수 있습니다. 자세한 내용은 GitHub에서 MPI 연산자를 참조하세요.

```
kubectl apply -f https://raw.githubusercontent.com/kubeflow/mpi-operator/master/deploy/v2beta1/mpi-operator.yaml
```

GPUDirectRDMA/EFA 검증을 위한 다중 노드 NCCL 성능 테스트 실행

EFA를 통한 GPUDirectRDMA를 사용하여 NCCL 성능을 검증하려면 표준 NCCL 성능 테스트를 실행합니다. 자세한 내용은 GitHub에서 공식 NCCL-Tests 리포지토리를 참조하세요. NVIDIA CUDA 11.2 및 EFA 최신 버전용으로 사전 구축되어 이 테스트와 함께 제공되는 샘플 Dockerfile을 사용할 수 있습니다.

또는 Amazon ECR 리포지토리에서 사용 가능한 AWS Docker 이미지를 다운로드할 수 있습니다.

> **Important**
>
> Kubernetes에 EFA를 도입하는 데 필요한 중요한 고려 사항은 Huge Page를 클러스터에서 리소스로 구성하고 관리하는 것입니다. 자세한 내용은 Kubernetes 설명서의 대용량 페이지 관리를 참조하세요. EFA 드라이버가 설치된 Amazon EC2 인스턴스는 5128 2M Huge Page를 미리 할당하며, 작업 사양에 따라 사용할 리소스로 요청할 수 있습니다.

2개 노드 NCCL 성능 테스트를 실행하려면 다음 단계를 완료합니다. 예제 NCCL 테스트 작업에서 각 작업자는 8개의 GPU, 5210Mi의 hugepages-2Mi, 4개의 EFA 및 8000Mi 메모리를 요청합니다. 이는 실질적으로 각 작업자가 `p4d.24xlarge` 인스턴스의 리소스를 모두 사용한다는 것을 의미합니다.

1. NCCL 테스트 작업을 생성합니다.

   ```
   kubectl apply -f https://raw.githubusercontent.com/aws-samples/aws-efa-eks/main/examples/simple/nccl-efa-tests.yaml
   ```

   예제 출력은 다음과 같습니다.

   ```
   mpijob.kubeflow.org/nccl-tests-efa 생성
   ```

2. 실행 중인 Pods를 봅니다.

   ```
   kubectl get pods
   ```

   예제 출력은 다음과 같습니다.

   ```
   NAME                              READY   STATUS     RESTARTS   AGE
   ```

```
nccl-tests-efa-launcher-nbql9  0/1     Init:0/1   0          2m49s
nccl-tests-efa-worker-0          1/1     Running    0          2m49s
nccl-tests-efa-worker-1          1/1     Running    0          2m49s
```

MPI 연산자는 런처 Pod와 작업자 Pods 2개(각 노드에 하나씩)를 생성합니다.

3. `efa-launcher` Pod에 대한 로그를 확인합니다. *`wzr8j`*를 출력된 값과 바꿉니다.

```
kubectl logs -f nccl-tests-efa-launcher-nbql9
```

더 많은 예제는 GitHub에서 Amazon EKS EFA 샘플 리포지토리를 참조하세요.

AWS Inferentia를 사용한 기계 학습 추론

이 주제에서는 Amazon EC2 Inf1 인스턴스를 실행하는 노드가 있는 Amazon EKS 클러스터를 생성하고 샘플 애플리케이션을 배포(선택 사항)하는 방법을 설명합니다. Amazon EC2 Inf1 인스턴스는 AWS Inferentia 칩으로 구동됩니다. 이 칩은 클라우드에서 고성능의 가장 저렴한 추론을 제공하기 위해 AWS에서 사용자 지정으로 구축됩니다. 기계 학습 모델은 Inferentia 칩의 기계 학습 추론 성능을 최적화하는 컴파일러, 런타임 및 프로파일링 도구로 구성된 특수 소프트웨어 개발 키트(SDK)인 AWS Neuron을 사용하여 컨테이너에 배포됩니다. AWS Neuron은 TensorFlow, PyTorch, MXNet과 같은 인기 있는 기계 학습 프레임워크를 지원합니다.

> **Note**
>
> Neuron 디바이스 논리적 ID는 연속적이어야 합니다. Neuron 디바이스를 여러 개 요청하는 Pod가 `inf1.6xlarge` 또는 `inf1.24xlarge` 인스턴스 유형(둘 이상의 Neuron 디바이스가 있음)에서 예약된 경우 Kubernetes 스케줄러가 불연속적인 디바이스 ID를 선택하면 해당 Pod가 시작되지 않습니다. 자세한 내용은 GitHub의 Device logical IDs must be contiguous를 참조하세요.

필수 조건

- `eksctl`이 컴퓨터에 설치되어 있어야 합니다. 설치되지 않은 경우 `eksctl` 설명서에서 Installation을 참조하세요.
- `kubectl`이 컴퓨터에 설치되어 있어야 합니다. 자세한 내용은 kubectl 설치 또는 업데이트 단원을 참조하십시오.

- (선택 사항) python3이 컴퓨터에 설치되어 있어야 합니다. 설치되어 있지 않은 경우 Python downloads를 참조하여 설치 지침을 확인하십시오.

클러스터 생성

Inf1 Amazon EC2 인스턴스 노드를 사용하여 클러스터를 생성하려면

1. Inf1 Amazon EC2 인스턴스 노드를 사용하여 클러스터를 생성합니다. *inf1.2xlarge*를 모든 Inf1 인스턴스 유형으로 바꿀 수 있습니다. eksctl 유틸리티는 Inf1 인스턴스 유형을 사용하여 노드 그룹을 시작한다는 것을 감지하고 Amazon EKS 최적화 가속 Amazon Linux AMI 중 하나를 사용하여 노드를 시작합니다.

   > **Note**
   >
   > TensorFlow Serving에서는 서비스 계정에 IAM 역할을 사용할 수 없습니다.

   ```
   eksctl create cluster \
       --name inferentia \
       --region region-code \
       --nodegroup-name ng-inf1 \
       --node-type inf1.2xlarge \
       --nodes 2 \
       --nodes-min 1 \
       --nodes-max 4 \
       --ssh-access \
       --ssh-public-key your-key \
       --with-oidc
   ```

   > **Note**
   >
   > 출력의 다음 행의 값을 적어 둡니다. 이후 (선택 사항) 단계에서 이 값을 사용합니다.
   >
   > ```
   > [9]  adding identity "arn:aws:iam::111122223333:role/
   > eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09" to auth
   >  ConfigMap
   > ```

`Inf1` 인스턴스로 노드 그룹을 시작하면 `eksctl`은 AWS Neuron Kubernetes 디바이스 플러그인을 자동으로 설치합니다. 이 플러그인은 Neuron 디바이스를 Kubernetes 스케줄러에 시스템 리소스로 알립니다. 이 스케줄러는 컨테이너에서 요청할 수 있습니다. 기본 Amazon EKS 노드 IAM 정책 외에도 Amazon S3 읽기 전용 액세스 정책이 추가되어 이후 단계에서 다루는 샘플 애플리케이션이 Amazon S3에서 훈련된 모델을 로드할 수 있습니다.

2. 모든 Pods가 올바르게 시작되었는지 확인합니다.

```
kubectl get pods -n kube-system
```

간략한 출력:

```
NAME                                  READY   STATUS    RESTARTS   AGE
[...]
neuron-device-plugin-daemonset-6djhp   1/1     Running   0          5m
neuron-device-plugin-daemonset-hwjsj   1/1     Running   0          5m
```

(선택 사항) TensorFlow Serving 애플리케이션 이미지 배포

훈련된 모델은 Inferentia 인스턴스에서 배포하기 전에 Inferentia 대상에 컴파일되어야 합니다. 계속하려면, Amazon S3 저장된 Neuron 최적화 TensorFlow 모델이 필요합니다. 아직 SavedModel이 없는 경우 Neuron 호환 ResNet50 모델 생성에 대한 자습서를 따르고 그 결과로 생성된 SavedModel을 S3에 업로드합니다. ResNet-50은 이미지 인식 작업에 사용되는 인기 있는 기계 학습 모델입니다. Neuron 모델 컴파일에 대한 자세한 내용은 AWS Deep Learning AMI 개발자 가이드에서 DLAMI를 사용한 AWS Inferentia 칩 섹션을 참조하세요.

샘플 배포 매니페스트는 AWS Deep Learning Containers에서 제공되는 사전 구축된 TensorFlow용 추론 Serving 컨테이너를 관리합니다. 이 컨테이너 내부에는 AWS Neuron 런타임 및 TensorFlow Serving 애플리케이션이 있습니다. Neuron에 최적화된 사전 구축된 딥 러닝 컨테이너의 전체 목록은 사용 가능한 이미지아래 GitHub에서 관리됩니다. 시작 시 DLC가 Amazon S3에서 모델을 가져오고, 저장된 모델로 Neuron TensorFlow Serving을 시작하고, 예측 요청을 기다립니다.

Serving 애플리케이션에 할당된 Neuron 디바이스의 수는 배포 yaml에서 `aws.amazon.com/neuron` 리소스를 변경하여 조정할 수 있습니다. TensorFlow Serving과 Neuron 런타임 간의 통신은 GRPC를 통해 발생하므로 `IPC_LOCK` 기능을 컨테이너에 전달해야 합니다.

1. 클러스터 생성의 1단계에서 생성한 노드 인스턴스 역할에 AmazonS3ReadOnlyAccess IAM 정책을 추가합니다. 이는 샘플 애플리케이션이 Amazon S3에서 훈련된 모델을 로드하기 위해 필요합니다.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \
    --role-name eksctl-inferentia-nodegroup-ng-in-NodeInstanceRole-FI7HIYS3BS09
```

2. 다음 콘텐츠를 가진 rn50_deployment.yaml이라는 파일을 생성합니다: 원하는 설정과 일치하도록 리전 코드 및 모델 경로를 업데이트합니다. 모델 이름은 클라이언트가 TensorFlow 서버에 요청할 때 식별을 위한 것입니다. 이 예에서는 모델 이름을 사용하여 예측 요청을 보내는 이후 단계에서 사용할 샘플 ResNet50 클라이언트 스크립트와 일치시킵니다.

```
aws ecr list-images --repository-name neuron-rtd --registry-id 790709498068 --
region us-west-2
```

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
    role: master
spec:
  replicas: 2
  selector:
    matchLabels:
      app: eks-neuron-test
      role: master
  template:
    metadata:
      labels:
        app: eks-neuron-test
        role: master
    spec:
      containers:
        - name: eks-neuron-test
          image: 763104351884.dkr.ecr.us-east-1.amazonaws.com/tensorflow-inference-
neuron:1.15.4-neuron-py37-ubuntu18.04
          command:
            - /usr/local/bin/entrypoint.sh
```

```
          args:
            - --port=8500
            - --rest_api_port=9000
            - --model_name=resnet50_neuron
            - --model_base_path=s3://your-bucket-of-models/resnet50_neuron/
          ports:
            - containerPort: 8500
            - containerPort: 9000
          imagePullPolicy: IfNotPresent
          env:
            - name: AWS_REGION
              value: "us-east-1"
            - name: S3_USE_HTTPS
              value: "1"
            - name: S3_VERIFY_SSL
              value: "0"
            - name: S3_ENDPOINT
              value: s3.us-east-1.amazonaws.com
            - name: AWS_LOG_LEVEL
              value: "3"
          resources:
            limits:
              cpu: 4
              memory: 4Gi
              aws.amazon.com/neuron: 1
            requests:
              cpu: "1"
              memory: 1Gi
          securityContext:
            capabilities:
              add:
                - IPC_LOCK
```

3. 모델을 배포합니다.

```
kubectl apply -f rn50_deployment.yaml
```

4. 다음 콘텐츠를 가진 `rn50_service.yaml`이라는 파일을 생성합니다: HTTP 및 gRPC 포트가 예측 요청을 수락하기 위해 열립니다.

```
kind: Service
apiVersion: v1
metadata:
```

```
  name: eks-neuron-test
  labels:
    app: eks-neuron-test
spec:
  type: ClusterIP
  ports:
    - name: http-tf-serving
      port: 8500
      targetPort: 8500
    - name: grpc-tf-serving
      port: 9000
      targetPort: 9000
  selector:
    app: eks-neuron-test
    role: master
```

5. TensorFlow 모델 Serving 애플리케이션을 위한 Kubernetes 서비스를 생성합니다.

```
kubectl apply -f rn50_service.yaml
```

(선택 사항) TensorFlow Serving 서비스에 대한 예측

1. 로컬로 테스트하려면 gRPC 포트를 eks-neuron-test 서비스로 전달합니다.

```
kubectl port-forward service/eks-neuron-test 8500:8500 &
```

2. 다음 콘텐츠를 통해 tensorflow-model-server-infer.py라는 Python 스크립트를 생성합니다. 이 스크립트는 서비스 프레임워크인 gRPC를 통해 추론을 실행합니다.

```
import numpy as np
   import grpc
   import tensorflow as tf
   from tensorflow.keras.preprocessing import image
   from tensorflow.keras.applications.resnet50 import preprocess_input
   from tensorflow_serving.apis import predict_pb2
   from tensorflow_serving.apis import prediction_service_pb2_grpc
   from tensorflow.keras.applications.resnet50 import decode_predictions

   if __name__ == '__main__':
       channel = grpc.insecure_channel('localhost:8500')
       stub = prediction_service_pb2_grpc.PredictionServiceStub(channel)
```

```
        img_file = tf.keras.utils.get_file(
            "./kitten_small.jpg",
            "https://raw.githubusercontent.com/awslabs/mxnet-model-server/master/
docs/images/kitten_small.jpg")
        img = image.load_img(img_file, target_size=(224, 224))
        img_array = preprocess_input(image.img_to_array(img)[None, ...])
        request = predict_pb2.PredictRequest()
        request.model_spec.name = 'resnet50_inf1'
        request.inputs['input'].CopyFrom(
            tf.make_tensor_proto(img_array, shape=img_array.shape))
        result = stub.Predict(request)
        prediction = tf.make_ndarray(result.outputs['output'])
        print(decode_predictions(prediction))
```

3. 스크립트를 실행하여 서비스에 예측을 제출합니다.

```
python3 tensorflow-model-server-infer.py
```

예제 출력은 다음과 같습니다.

```
[[(u'n02123045', u'tabby', 0.68817204), (u'n02127052', u'lynx', 0.12701613),
 (u'n02123159', u'tiger_cat', 0.08736559), (u'n02124075', u'Egyptian_cat',
 0.063844085), (u'n02128757', u'snow_leopard', 0.009240591)]]
```

클러스터 관리

이 장에서는 클러스터를 관리하는 데 도움이 되는 다음 주제를 다룹니다. AWS Management Console을 사용하여 Kubernetes 리소스의 정보를 볼 수도 있습니다.

- Kubernetes Dashboard는 Kubernetes 클러스터용 범용 웹 기반 UI입니다. 이를 통해 사용자는 클러스터에서 실행 중인 애플리케이션을 관리하고 문제를 해결할 수 있을 뿐만 아니라 클러스터 자체를 관리할 수 있습니다. 자세한 내용은 Kubernetes Dashboard GitHub 리포지토리를 참조하세요.
- Kubernetes 지표 서버 설치 - Kubernetes 지표 서버는 클러스터에서 리소스 사용량 데이터의 집계자입니다. 클러스터에 기본적으로 배포되지 않지만 Kubernetes 대시보드 및 Horizontal Pod Autoscaler와 같은 Kubernetes 추가 기능에 사용됩니다. 이 주제에서는 Metrics Server를 설치하는 방법을 알아봅니다.
- Amazon EKS에 Helm 사용 - Kubernetes용 Helm 패키지 관리자는 Kubernetes 클러스터에서 애플리케이션을 설치하고 관리하는 데 도움이 됩니다. 이 주제에서는 Helm 바이너리를 설치하고 실행하는 방법을 설명하므로 로컬 컴퓨터에서 Helm CLI를 사용하여 차트를 설치하고 관리할 수 있습니다.
- Amazon EKS 리소스 태깅 - Amazon EKS 리소스 관리를 지원하기 위해 태그 형식으로 각 리소스에 고유한 메타데이터를 할당할 수 있습니다. 이 주제에서는 태그를 설명하고 태그를 생성하는 방법을 보여 줍니다.
- Amazon EKS 서비스 할당량 - AWS 계정에는 각 AWS 서비스에 대한 기본 할당량(이전에는 제한이라고 함)이 있습니다. Amazon EKS 할당량과 이러한 할당량을 높이는 방법을 알아봅니다.

비용 모니터링

비용 모니터링은 Amazon EKS에서 Kubernetes 클러스터를 관리할 때 필수적인 요소입니다. 클러스터 비용에 대한 가시성을 확보하면 리소스 사용률을 최적화하고, 예산을 설정하고, 배포에 대한 데이터 기반 결정을 내릴 수 있습니다. Amazon EKS는 비용을 효과적으로 추적하고 할당할 수 있도록 도와주는 두 가지 비용 모니터링 솔루션을 제공하며, 각 솔루션에는 고유한 장점이 있습니다.

Amazon EKS에 대한 AWS 빌링 분할 비용 할당 데이터 - 이 기본 기능은 AWS 빌링 콘솔과 원활하게 통합되므로 다른 AWS 서비스에서 사용하는 것과 동일한 친숙한 인터페이스와 워크플로를 사용하여 비용을 분석하고 할당할 수 있습니다. 비용 할당 분할을 사용하면 다른 AWS 지출과 함께 Kubernetes 비용을 직접 파악할 수 있으므로 AWS 환경 전반에서 비용을 전체적으로 최적화하기가 더 쉬워집니다. 또한 비용 범주 및 비용 이상 징후 감지와 같은 기존 AWS 빌링 기능을 활용하여 비용 관리 기능을 더욱 향상시킬 수 있습니다. 자세한 내용은 AWS 빌링 사용 설명서의 분할 비용 할당 데이터에 대한 이해를 참조하세요.

Kubecost - Amazon EKS는 Kubernetes 비용 모니터링 도구인 Kubecost를 지원합니다. Kubecost는 Kubernetes 네이티브 방식으로 기능이 풍부한 비용 모니터링 접근 방식을 제공하며 Kubernetes 리소스별로 세분화된 비용 분석, 비용 최적화 권장 사항, 즉시 사용 가능한 대시보드 및 보고서를 제공합니다. 또한 Kubecost는 AWS 비용 및 사용 보고서를 통합하여 정확한 요금 데이터를 가져오므로 고객이 Amazon EKS 비용을 정확하게 파악할 수 있습니다. Kubecost 설치 방법을 알아봅니다.

AWS 빌링 - 분할 비용 할당

Amazon EKS에 대한 AWS 분할 비용 할당 데이터를 사용한 비용 모니터링

Amazon EKS에 대한 AWS 분할 비용 할당 데이터를 사용하여 Amazon EKS 클러스터의 비용을 세부적으로 파악할 수 있습니다. 이를 통해 Kubernetes 애플리케이션의 비용 및 사용량을 분석하고 최적화하며 차지백할 수 있습니다. Kubernetes 애플리케이션에서 사용하는 Amazon EC2 CPU 및 메모리 리소스를 기반으로 개별 사업부 및 팀에 애플리케이션 비용을 할당합니다. Amazon EKS에 대한 분할 비용 할당 데이터를 사용하면 포드당 비용을 파악할 수 있고 네임스페이스, 클러스터 및 기타 Kubernetes 기본 요소를 사용하여 포드당 비용 데이터를 집계할 수 있습니다. 다음은 Amazon EKS 비용 할당 데이터를 분석하는 데 사용할 수 있는 Kubernetes 기본 요소의 예입니다.

- 클러스터 이름
- 배포
- 네임스페이스
- 노드
- 워크로드 이름
- 워크로드 유형

분할 비용 할당 데이터에 대한 자세한 내용은 AWS 빌링 사용 설명서의 분할 비용 할당 데이터에 대한 이해를 참조하세요.

비용 및 사용 보고서 설정

비용 관리 콘솔, AWS Command Line Interface 또는 AWS SDK에서 EKS에 대한 분할 비용 할당 데이터를 설정할 수 있습니다.

분할 비용 할당 데이터에 대해 다음을 사용합니다.

1. 분할 비용 할당 데이터를 옵트인합니다. 자세한 내용은 AWS Cost and Usage Report 사용 설명서의 분할 비용 할당 데이터 활성화를 참조하세요.
2. 새 보고서나 기존 보고서에 데이터를 포함합니다.

3. 보고서를 봅니다. 결제 및 비용 관리 콘솔을 사용하거나 Amazon Simple Storage Service에서 보고서 파일을 볼 수 있습니다.

Kubecost

Amazon EKS는 Pods, 노드, 네임스페이스 및 레이블을 포함한 Kubernetes 리소스별로 분류된 비용을 모니터링하는 데 사용할 수 있는 Kubecost를 지원합니다. Kubernetes 플랫폼 관리자 및 재무 리더는 Kubecost를 사용하여 Amazon EKS 청구 내역을 시각화하고, 비용을 할당하고, 애플리케이션 팀과 같은 조직 단위를 청구할 수 있습니다. 내부 팀과 사업부에 실제 AWS 청구서를 기반으로 투명하고 정확한 비용 데이터를 제공할 수 있습니다. 또한 인프라 환경 및 클러스터 내 사용 패턴을 기반으로 비용 최적화를 위한 맞춤형 권장 사항을 가져올 수도 있습니다. Kubecost에 대한 자세한 내용은 Kubecost 설명서를 참조하십시오.

Amazon EKS는 클러스터 비용 가시성을 위해 AWS에 최적화된 Kubecost 번들을 제공합니다. 기존 AWS 지원 계약을 사용하여 지원을 받을 수 있습니다.

필수 조건

- 기존 Amazon EKS 클러스터. 배포하려면 Amazon EKS 시작하기 섹션을 참조하세요. Fargate 노드에서는 Kubecost를 실행할 수 없으므로 클러스터에 Amazon EC2 노드가 있어야 합니다.
- 디바이스 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 `kubectl` 설치 또는 업데이트 부분을 참조하세요.
- 디바이스 또는 AWS CloudShell에 구성된 Helm 버전이 3.9.0 이상이어야 합니다. Helm을 설치 또는 업데이트하려면 the section called “Helm 사용” 부분을 참조하세요.
- 클러스터가 버전 `1.23` 이상인 경우 클러스터에 the section called “Amazon EBS CSI 드라이버”가 설치되어 있어야 합니다.

Kubecost 설치 방법

1. 설치할 Kubecost의 버전을 결정합니다. Amazon ECR Public Gallery의 kubecost/cost-analyzer에서 사용 가능한 버전을 확인할 수 있습니다. Kubecost 버전과 Amazon EKS의 호환성에 대한 자세한 내용은 Kubecost 설명서의 Environment Requirements를 참조하세요.
2. 다음 명령으로 Kubecost를 설치합니다. *`kubecost-version`*을 ECR에서 검색한 값(예: *`1.108.1`*)으로 바꿉니다.

```
helm upgrade -i kubecost oci://public.ecr.aws/kubecost/cost-analyzer --
version kubecost-version \
    --namespace kubecost --create-namespace \
    -f https://raw.githubusercontent.com/kubecost/cost-analyzer-helm-chart/develop/
cost-analyzer/values-eks-cost-monitoring.yaml
```

Kubecost는 주기적으로 새 버전을 출시합니다. `helm upgrade`를 사용하여 버전을 업데이트할 수 있습니다. 기본적으로 설치에는 로컬 Prometheus 서버와 `kube-state-metrics`가 포함됩니다. 설명서의 Integrating with Amazon EKS cost monitoring(Amazon EKS 비용 모니터링과 통합)에 따라 Amazon Managed Service for Prometheus를 사용하도록 배포를 사용자 지정할 수 있습니다. 구성 가능한 다른 모든 설정 목록은 GitHub의 샘플 구성 파일를 참조하세요.

3. 필요한 Pods가 실행 중인지 확인합니다.

```
kubectl get pods -n kubecost
```

예제 출력은 다음과 같습니다.

```
NAME                                          READY   STATUS    RESTARTS   AGE
kubecost-cost-analyzer-b9788c99f-5vj5b        2/2     Running   0          3h27m
kubecost-kube-state-metrics-99bb8c55b-bn2br   1/1     Running   0          3h27m
kubecost-prometheus-server-7d9967bfc8-9c8p7   2/2     Running   0          3h27m
```

4. 디바이스에서 포트 포워딩을 활성화하여 Kubecost 대시보드를 노출합니다.

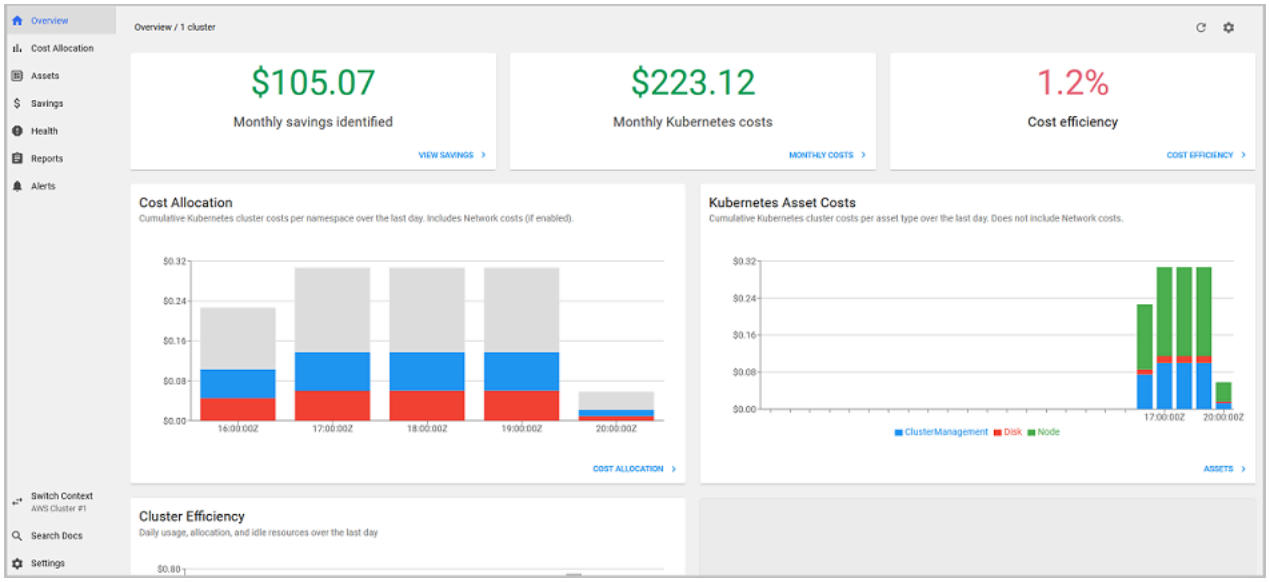
```
kubectl port-forward --namespace kubecost deployment/kubecost-cost-analyzer 9090
```

또는 AWS Load Balancer Controller를 사용하여 Kubecost 를 노출하고 인증, 권한 부여 및 사용자 관리를 위해 Amazon Cognito를 사용할 수 있습니다. 자세한 내용은 Application Load Balancer 및 Amazon Cognito를 사용하여 Kubernetes 웹 앱에 대한 사용자를 인증하는 방법을 참조하세요.

5. 이전 단계를 완료한 동일한 디바이스에서 웹 브라우저를 열고 다음 주소를 입력합니다.

```
http://localhost:9090
```

브라우저에 Kubecost Overview(개요) 페이지가 표시됩니다. Kubecost가 지표를 수집하는 데 5~10분 정도 걸릴 수 있습니다. 누적 클러스터 비용, 관련 Kubernetes 자산 비용, 월별 집계 비용을 포함한 Amazon EKS 지출을 확인할 수 있습니다.



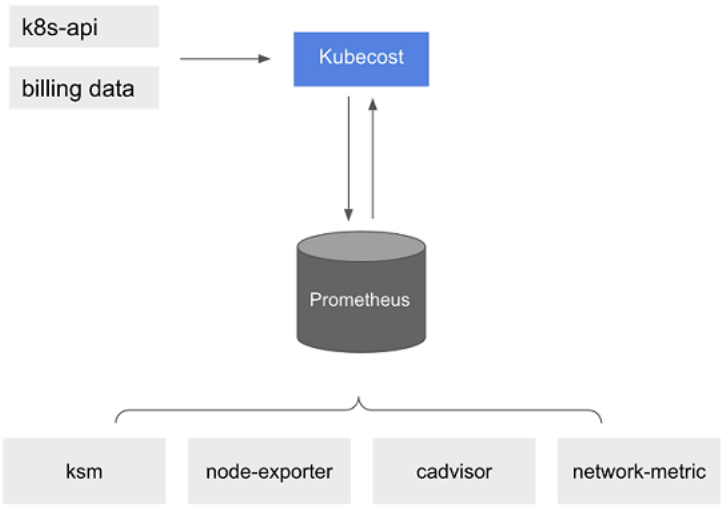
6. 클러스터 수준에서 비용을 추적하려면 청구할 Amazon EKS 리소스에 태그를 지정합니다. 자세한 내용은 리소스에 결제용 태그 지정 단원을 참조하십시오.

대시보드의 왼쪽 창에서 다음 정보를 선택하여 볼 수도 있습니다.

- Cost allocation(비용 할당) - 지난 7일 동안 각 네임스페이스와 기타 차원에 대한 월별 Amazon EKS 비용 및 누적 비용을 확인합니다. 이는 애플리케이션의 어느 부분이 Amazon EKS 지출에 기여하는지 파악하는 데 유용합니다.
- Assets(자산) - Amazon EKS 리소스와 연결된 AWS 인프라 자산의 비용을 확인합니다.

기타 기능

- Export cost metrics(비용 지표 내보내기) – Amazon EKS 최적화 비용 모니터링은 오픈 소스 모니터링 시스템 및 시계열 데이터베이스인 Kubecost 및 Prometheus와 함께 배포됩니다. Kubecost는 Prometheus에서 지표를 읽은 다음 비용 할당 계산을 수행하고 지표를 다시 Prometheus에 씁니다. Kubecost 프런트엔드는 Prometheus에서 지표를 읽고 Kubecost 사용자 인터페이스에 표시합니다. 다음 다이어그램에 아키텍처가 나와 있습니다.



Prometheus가 사전 설치되어 있으면 추가 분석을 위해 Kubecost 데이터를 현재 비즈니스 인텔리전스 시스템으로 모으는 쿼리를 작성할 수 있습니다. 또한 이를 현재 Grafana 대시보드의 데이터 소스로 사용하여 내부 팀이 잘 알고 있는 Amazon EKS 클러스터 비용을 표시할 수 있습니다. Prometheus 쿼리를 작성하는 방법에 대해 자세히 알아보려면 GitHub의 Prometheus 구성 `readme` 파일을 참조하거나 Kubecost Github 리포지토리의 예제 Grafana JSON 모델을 참조로 사용하세요.

- AWS Cost and Usage Report 통합 – Amazon EKS 클러스터에 대한 비용 할당 계산을 수행하기 위해 Kubecost는 AWS Price List API에서 AWS 서비스 및 AWS 리소스의 공개 가격 정보를 검색합니다. 또한 Kubecost를 AWS Cost and Usage Report와 통합하여 AWS 계정과 관련된 요금 정보의 정확성을 높일 수 있습니다. 이 정보에는 엔터프라이즈 할인 프로그램, 예약 인스턴스 사용량, 절감형 플랜 및 스팟 사용량이 포함됩니다. AWS Cost and Usage Report 통합 작동 방식에 대한 자세한 내용은 Kubecost 설명서의 AWS 클라우드 결제 통합을 참조하세요.

Kubecost 제거

다음 명령을 사용하여 클러스터에서 Kubecost를 제거할 수 있습니다.

```
helm uninstall kubecost --namespace kubecost
kubectl delete ns kubecost
```

자주 묻는 질문(FAQ)

Amazon EKS에서 Kubecost 사용에 대한 다음과 같은 일반적인 질문과 답변을 참조하세요.

Kubecost의 사용자 지정 번들과 무료 버전의 Kubecost(OpenCost(이)라고도 함)의 차이점은 무엇인가요?

AWS와 Kubecost는 협력하여 사용자 지정 버전의 Kubecost를 제공했습니다. 이 버전에는 추가 요금 없이 상용 기능의 하위 집합이 포함되어 있습니다. Kubecost의 사용자 지정 번들에 포함된 기능은 다음 표를 참조하세요.

기능	Kubecost 프리 티어	Amazon EKS 최적화 Kubecost 사용자 지정 번들	Kubecost Enterprise
배포	사용자 호스팅	사용자 호스팅	사용자 호스팅 또는Kubecost 호스팅 (SaaS)
지원되는 클러스터의 수	무제한	무제한	무제한
지원되는 데이터베이스	로컬 Prometheus	로컬 Prometheus 및 Amazon Managed Service for Prometheus	Prometheus, Amazon Managed Service for Prometheus, Cortex 또는 Thanos
데이터베이스 보존 지원	15일	무제한 이력 데이터	무제한 이력 데이터
Kubecost API 보존 (ETL)	15일	15일	무제한 이력 데이터
클러스터 비용 가시성	단일 클러스터	통합 멀티 클러스터	통합 멀티 클러스터
하이브리드 클라우드 가시성	-	Amazon EKS 및 Amazon EKS Anywhere 클러스터	멀티 클라우드 및 하이브리드 클라우드 지원
경고 및 반복 보고서	-	효율성 알림, 예산 알림, 지출 변경 알림 등 지원	효율성 알림, 예산 알림, 지출 변경 알림 등 지원

기능	Kubecost 프리 티어	Amazon EKS 최적화 Kubecost 사용자 지정 번들	Kubecost Enterprise
저장된 보고서	-	15일 데이터를 사용한 보고서	무제한 이력 데이터를 사용하는 보고서
클라우드 결제 통합	각 개별 클러스터에 필요	AWS에 대한 사용자 지정 가격 지원(여러 클러스터 및 여러 계정 포함)	AWS에 대한 사용자 지정 가격 지원(여러 클러스터 및 여러 계정 포함)
절감 권장 사항	단일 클러스터 인사이트	단일 클러스터 인사이트	멀티 클러스터 인사이트
거버넌스: 감사	-	-	이력 비용 이벤트 감사
Single Sign-On(SSO) 지원	-	Amazon Cognito 지원	Okta, Auth0, PingID, KeyCloak
SAML이 포함된 역할 기반 액세스 제어(RBAC) **2.0**	-	-	Okta, Auth0, PingID, Keycloak
기업 교육 및 온보딩	-	-	풀 서비스 교육 및 FinOps 온보딩

Kubecost API 보존(ETL) 기능은 무엇입니까?

Kubecost ETL 기능은 지표를 집계 및 구성하여 다양한 수준의 세분화에서 비용 가시성을 제공합니다(예: `namespace-level`, `pod-level` 및`deployment-level`). 사용자 지정 Kubecost 번들의 경우 고객은 지난 15일간의 지표를 통해 데이터와 인사이트를 얻을 수 있습니다.

알림 및 반복 보고서 기능이란 무엇입니까? 여기에는 어떤 알림 및 보고서가 포함됩니까?

Kubecost 알림을 통해 팀은 실시간 Kubernetes 지출 및 클라우드 지출에 대한 업데이트를 받을 수 있습니다. 반복 보고서를 통해 팀은 이력 Kubernetes 및 클라우드 지출에 대한 사용자 지정 보기를 받을 수 있습니다. 둘 다 Kubecost UI 또는 Helm 값을 사용하여 구성할 수 있습니다. 팀은 이메일, Slack 및 Microsoft Teams을(를) 지원합니다.

저장된 보고서에는 어떤 내용이 포함됩니까?

Kubecost 저장된 보고서는 비용 및 효율성 지표의 사전 정의된 보기입니다. 여기에는 클러스터, 네임스페이스, 라벨 등에 따른 비용이 포함됩니다.

클라우드 결제 통합이란 무엇입니까?

AWS 청구 API와 통합하면 Kubecost이(가) 클러스터 외 비용(예: Amazon S3)을 표시할 수 있습니다. 또한, 이를 통해 Kubecost은(는) Kubecost의 클러스터 내 예측을 실제 결제 데이터와 일치시켜 스팟 사용량, 절감형 플랜 및 기업 할인을 설명할 수 있습니다.

절감 권장 사항에는 무엇이 포함됩니까?

Kubecost은(는) 사용자가 자신의 Kubernetes 인프라 및 지출을 최적화하는 데 도움이 되도록 인사이트와 자동화를 제공합니다.

이 기능은 유료인가요?

아니요. 추가 비용 없이 이 버전의 Kubecost를 사용할 수 있습니다. 이 번들에 포함되지 않은 추가 Kubecost 기능이 필요한 경우 AWS Marketplace를 통해 또는 Kubecost에서 직접 Kubecost의 엔터프라이즈 라이선스를 구입할 수 있습니다.

지원을 받을 수 있나요?

예. AWS에 문의하기에서 AWS Support 팀에 지원 사례를 열 수 있습니다.

Amazon EKS 통합에서 제공하는 Kubecost 기능을 사용하려면 라이선스가 필요하나요?

아니요.

보다 정확한 보고를 위해 Kubecost를 AWS Cost and Usage Report와 통합할 수 있나요?

예. AWS Cost and Usage Report에서 데이터를 수집하도록 Kubecost를 구성하여 할인, 스팟 요금, 예약 인스턴스 요금 등의 정확한 비용 가시성을 얻을 수 있습니다. 자세한 내용은 Kubecost 설명서의 AWS 클라우드 결제 통합을 참조하세요.

이 버전에서 Amazon EC2에서 자체 관리형 Kubernetes 클러스터의 비용 관리를 지원하나요?

아니요. 이 버전은 Amazon EKS 클러스터와만 호환됩니다.

Kubecost가 AWS Fargate에서 Amazon EKS에 대한 비용을 추적할 수 있나요?

Kubecost는 Fargate의 Amazon EKS에 대한 클러스터 비용 가시성을 보여주기 위해 최선을 다하지만 Amazon EC2의 Amazon EKS보다 정확도가 낮습니다. 이는 주로 사용량에 대한 요금 청구 방식

의 차이 때문입니다. Fargate의 Amazon EKS를 사용하면 사용한 리소스에 대한 요금이 청구됩니다. Amazon EC2 노드의 Amazon EKS를 사용하면 프로비저닝된 리소스에 대한 요금이 청구됩니다. Kubecost는 CPU, RAM 및 임시 스토리지를 포함하는 노드 사양을 기반으로 Amazon EC2 노드의 비용을 계산합니다. Fargate를 사용하면 Fargate Pods에 대해 요청된 리소스를 기준으로 비용이 계산됩니다.

Kubecost의 업데이트와 새 버전을 받으려면 어떻게 해야 하나요?

표준 Helm 업그레이드 절차를 사용하여 Kubecost 버전을 업그레이드할 수 있습니다. 최신 버전은 Amazon ECR Public Gallery에 있습니다.

`kubectl-cost` CLI가 지원되나요? 어떻게 설치하나요?

예. `kubectl-cost`는 Kubernetes 비용 할당 지표에 대한 CLI 액세스를 제공하는 Kubecost(Apache 2.0 라이선스)의 오픈 소스 도구입니다. `kubectl-cost`를 설치하려면 GitHub의 Installation(설치)을 참조하세요.

Kubecost 사용자 인터페이스가 지원되나요? 액세스하려면 어떻게 해야 하나요?

Kubecost는 `kubectl` 포트 전달, 수신 또는 로드 밸런서를 통해 액세스할 수 있는 웹 대시보드를 제공합니다. AWS Load Balancer Controller를 사용하여 Kubecost를 노출하고 인증, 권한 부여 및 사용자 관리를 위해 Amazon Cognito를 사용할 수 있습니다. 자세한 내용은 AWS 블로그의 How to use Application Load Balancer and Amazon Cognito to authenticate users for your Kubernetes web apps(애플리케이션 로드 밸런서 및 Amazon Cognito를 사용하여 웹 앱에 대한 사용자를 인증하는 방법)를 참조하세요.

Amazon EKS Anywhere가 지원되나요?

아니요.

Kubernetes 지표 서버 설치

Kubernetes 지표 서버는 클러스터에서 리소스 사용량 데이터의 집계자이며, 기본적으로 Amazon EKS 클러스터에 배포되어 있지 않습니다. 자세한 내용은 GitHub에서 Kubernetes지표 서버를 참조하세요. 지표 서버는 다른 Kubernetes 추가 기능(예: Horizontal Pod Autoscaler 또는 Kubernetes 대시보드)에서 일반적으로 사용됩니다. 자세한 내용은 Kubernetes 설명서의 리소스 지표 파이프라인을 참조하세요. 이 주제에서는 Amazon EKS 클러스터에 Kubernetes 지표 서버를 배포하는 방법을 설명합니다.

> **Important**
>
> 지표는 시점 분석을 위한 것이며 기록 분석을 위한 정확한 소스가 아닙니다. 모니터링 솔루션이나 기타 Auto Scaling 이외의 용도로는 사용할 수 없습니다. 모니터링 도구에 대한 자세한 내용은 Amazon EKS의 관찰 가능성 섹션을 참조하세요.

지표 서버 배포

1. 다음 명령을 사용하여 지표 서버를 배포합니다.

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
```

2. 다음 명령을 사용하여 `metrics-server` 배포에서 원하는 수의 Pods를 실행하고 있는지 확인합니다.

```
kubectl get deployment metrics-server -n kube-system
```

예제 출력은 다음과 같습니다.

```
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           6m
```

Amazon EKS에 Helm 사용

Kubernetes용 Helm 패키지 관리자는 Kubernetes 클러스터에서 애플리케이션을 설치하고 관리하는 데 도움이 됩니다. 자세한 내용은 Helm 설명서를 참조하십시오. 이 주제에서는 Helm 바이너리를 설치하고 실행하는 방법을 설명하므로 로컬 시스템에서 Helm CLI를 사용하여 차트를 설치하고 관리할 수 있습니다.

> **Important**
>
> Amazon EKS 클러스터에 Helm 차트를 설치하기 전에 Amazon EKS에 대해 작동하도록 `kubectl`을 구성해야 합니다. 아직 구성하지 않은 경우 계속하기 전에 Amazon EKS 클러스터용 `kubeconfig` 파일 생성 또는 업데이트 단원을 참조하십시오. 클러스터에 대해 다음 명령이 성공한 경우 적절하게 구성한 것입니다.

```
kubectl get svc
```

로컬 시스템에 Helm 바이너리를 설치하려면

1. 클라이언트 운영 체제에 맞는 명령을 실행합니다.
   - Homebrew와 함께 macOS를 사용하는 경우 다음 명령으로 바이너리를 설치합니다.

     ```
     brew install helm
     ```
   - Chocolatey와 함께 Windows를 사용하는 경우 다음 명령으로 바이너리를 설치합니다.

     ```
     choco install kubernetes-helm
     ```
   - Linux를 사용하는 경우 다음 명령을 사용하여 바이너리를 설치합니다.

     ```
     curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 >
      get_helm.sh
     chmod 700 get_helm.sh
     ./get_helm.sh
     ```

     > **Note**
     >
     > openssl을 먼저 설치해야 한다는 메시지가 나타나는 경우 다음 명령으로 설치할 수 있습니다.
     >
     > ```
     > sudo yum install openssl
     > ```
2. PATH에서 새 바이너리를 선택하려면 현재 터미널 창을 닫고 새 창을 엽니다.
3. 설치한 Helm 버전을 확인합니다.

   ```
   helm version | cut -d + -f 1
   ```

   예제 출력은 다음과 같습니다.

   ```
   v3.9.0
   ```

4. 이 시점에서 Helm 명령(예: `helm install` *`chart-name`*)을 실행하여 클러스터에서 Helm 차트를 설치, 수정, 삭제 또는 쿼리할 수 있습니다. Helm을 처음 사용하는 경우 설치할 특정 차트가 없으면 다음을 수행할 수 있습니다.

   - 예제 차트를 설치하여 실험합니다. Helm Quickstart 가이드에서 예제 차트 설치를 참조하세요.
   - 예제 차트를 생성하여 Amazon ECR로 푸시합니다. 자세한 내용은 Amazon Elastic 컨테이너 레지스트리 사용 설명서의 Helm 차트 푸시를 참조하세요.
   - eks-charts GitHub 리포지토리 또는 ArtifactHub에서 Amazon EKS 차트를 설치합니다.

Amazon EKS 리소스 태깅

태그를 사용하여 Amazon EKS 리소스를 관리할 수 있습니다. 이 항목에서는 태그 함수에 대한 개요를 제공하고 태그를 생성하는 방법을 보여 줍니다.

주제

> **Note**
>
> 태그는 Kubernetes 레이블 및 주석과 분리된 메타데이터 유형입니다. 이러한 다른 메타데이터 유형에 대한 자세한 내용을 알아보려면 Kubernetes 설명서의 다음 섹션을 참조하세요.
>
> - 레이블 및 선택기
> - 주석

태그 기본 사항

태그는 AWS 리소스에 할당하는 레이블입니다. 각 태그는 키와 값(선택 사항)으로 구성됩니다.

태그를 사용하여 AWS 리소스를 범주화할 수 있습니다. 예를 들어 용도, 소유자 또는 환경 기준별로 리소스를 범주화할 수 있습니다. 동일한 유형의 리소스가 많은 경우 특정 리소스에 할당한 태그를 사용하여 해당 리소스를 빠르게 식별할 수 있습니다. 예를 들어 Amazon EKS 클러스터에 태그 집합을 정의하면 클러스터의 소유자 및 스택 수준을 추적하는 데 도움이 됩니다. 각 리소스 유형에 대해 일관된 태그 키 집합을 고안하는 것이 좋습니다. 그러면 추가하는 태그에 따라 리소스를 검색하고 필터링할 수 있습니다.

태그를 추가한 후에는 언제든지 태그 키와 값을 편집하거나 리소스에서 태그를 제거할 수 있습니다. 리소스를 삭제하면 리소스 태그도 삭제됩니다.

태그는 Amazon EKS에는 아무런 의미가 없으며 엄격하게 문자열로 해석됩니다. 태그 값을 빈 문자열로 설정할 수 있습니다. 그러나 태그 값을 null로 설정할 수는 없습니다. 해당 리소스에 대해 키가 기존 태그와 동일한 태그를 추가하는 경우 새 값이 이전 값을 덮어씁니다.

AWS Identity and Access Management(IAM)를 사용하는 경우 AWS 계정에서 태그를 관리할 수 있는 권한을 가진 사용자를 제어할 수 있습니다.

리소스에 태그 지정

다음 Amazon EKS 리소스는 태그를 지원합니다.

- 클러스터
- 관리형 노드 그룹
- Fargate 프로필

다음을 사용하여 이러한 리소스에 태그를 지정할 수 있습니다.

- Amazon EKS 콘솔을 사용하는 경우 새 리소스 또는 기존 리소스에 태그를 언제든지 적용할 수 있습니다. 관련 리소스 페이지의 [태그(Tags)] 탭을 사용하여 이 작업을 수행할 수 있습니다. 자세한 내용은 콘솔을 사용한 태그 작업 섹션을 참조하세요.
- `eksctl`을 사용하는 경우 리소스가 생성될 때 `--tags` 옵션을 사용하여 리소스에 태그를 적용할 수 있습니다.
- AWS CLI, Amazon EKS API 또는 AWS SDK를 사용 중인 경우 관련 API 작업의 `tags` 파라미터를 사용하여 새 리소스에 태그를 적용할 수 있습니다. `TagResource` API 작업을 사용하여 기존 리소스에 태그를 적용할 수도 있습니다. 자세한 내용은 TagResource를 참조하세요.

일부 리소스 생성 작업을 사용할 때 리소스를 생성함과 동시에 리소스에 대한 태그를 지정할 수도 있습니다. 리소스를 생성하는 동안 태그를 적용할 수 없는 경우 리소스를 생성할 수 없습니다. 이 메커니즘

에서는 태그를 지정하려는 리소스가 지정된 태그와 함께 생성되거나 전혀 생성되지 않습니다. 리소스를 생성할 때 태그를 지정하면 리소스를 생성한 후 사용자 지정 태그 지정 스크립트를 실행할 필요가 없습니다.

태그는 생성한 리소스와 연결된 다른 리소스로 전파되지 않습니다. 예를 들어, Fargate 프로파일 태그는 Fargate 프로파일과 연결된 다른 리소스(예: 예약된 Pods)로 전파되지 않습니다.

태그 제한

태그에 적용되는 제한은 다음과 같습니다.

- 최대 50개의 태그를 리소스와 연결할 수 있습니다.
- 한 리소스에 대해 태그 키를 반복할 수 없습니다. 각 태그 키는 고유해야 하며 하나의 값만 가질 수 있습니다.
- 키는 최대 128자(UTF-8 형식)까지 가능합니다.
- 값은 최대 256자(UTF-8 형식)까지 가능합니다.
- 여러 AWS 서비스 및 리소스에서 태그 지정 스키마를 사용하는 경우 사용하는 문자 유형을 제한합니다. 일부 서비스는 허용되는 문자에 제한이 있을 수 있습니다. 일반적으로 허용되는 문자는 문자, 숫자, 공백 및 특수 문자 **+ - = . _ : / @**입니다.
- 태그 키와 값은 대소문자를 구분합니다.
- 키 또는 값에 `aws:`, `AWS:` 또는 이러한 접두사의 대문자 또는 소문자 조합을 사용하지 마세요. 이러한 이름은 AWS 전용으로 예약되어 있습니다. 이 접두사가 지정된 태그 키나 값은 편집하거나 삭제할 수 없습니다. 이 접두사가 지정된 태그는 리소스당 태그 수 제한에 포함되지 않습니다.

리소스에 결제용 태그 지정

Amazon EKS 클러스터에 태그를 적용하면 Cost & Usage Reports에서 비용 할당에 태그를 사용할 수 있습니다. Cost & Usage Reports의 측정 데이터는 모든 Amazon EKS 클러스터에서의 사용량을 보여줍니다. 자세한 내용은 AWS Billing 사용 설명서의 AWS 비용 및 사용 보고서를 참조하세요.

AWS에서 생성한 비용 할당 태그, 특히 `aws:eks:cluster-name`을 사용하면 Cost Explorer에서 Amazon EC2 인스턴스 비용을 개별 Amazon EKS 클러스터별로 분류할 수 있습니다. 그러나 이 태그는 컨트롤 플레인 비용을 캡처하지 않습니다. 태그는 Amazon EKS 클러스터에 참여하는 Amazon EC2 인스턴스에 자동으로 추가됩니다. 이 동작은 인스턴스가 Amazon EKS 관리형 노드 그룹인 Karpenter를 사용하여 프로비저닝되었는지 아니면 Amazon EC2를 사용하여 직접 프로비저닝되었는지 여부에 관계없이 발생합니다. 이 특정 태그는 50개 태그 한도에 포함되지 않습니다. 태그를 사용하려면 계정

소유자가 AWS Billing 콘솔에서 또는 API를 사용하여 태그를 활성화해야 합니다. AWS Organizations 관리 계정 소유자가 태그를 활성화하면 태그는 모든 조직 멤버 계정에 대해서도 활성화됩니다.

태그 키 값이 동일한 리소스를 기준으로 결제 정보를 구성할 수도 있습니다. 예를 들어, 특정 애플리케이션 이름으로 여러 리소스에 태그를 지정한 다음 결제 정보를 구성할 수 있습니다. 이렇게 하면 여러 서비스에서 해당 애플리케이션의 총 비용을 볼 수 있습니다. 태그를 사용한 비용 할당 보고서 설정에 대한 자세한 내용은 AWS Billing 사용 설명서에서 월간 비용 할당 보고서를 참조하세요.

> **Note**
>
> 방금 보고서를 활성화한 경우, 24시간 후에 이번 달의 데이터를 볼 수 있습니다.

Cost Explorer는 AWS 프리 티어의 일부로 제공되는 보고 도구입니다. Cost Explorer를 사용하여 지난 13개월 동안의 Amazon EKS 리소스 차트를 볼 수 있습니다. 또한 향후 3개월 동안의 지출을 예상해볼 수 있습니다. 시간 경과에 따른 AWS 리소스의 지출 패턴을 확인할 수 있습니다. 예를 들어 Cost Explorer를 사용하여 추가 조사가 필요한 영역을 알아내고, 비용 이해에 사용할 수 있는 추세를 파악할 수 있습니다. 또한 데이터의 시간 범위를 지정하고 일별 또는 월별 시간 데이터를 볼 수도 있습니다.

콘솔을 사용한 태그 작업

Amazon EKS 콘솔을 사용하면 신규/기존 클러스터 및 관리형 노드 그룹과 연결된 태그를 관리할 수 있습니다.

Amazon EKS 콘솔에서 리소스 관련 페이지를 선택하면 페이지에 이러한 리소스의 목록이 표시됩니다. 예를 들어, 왼쪽 탐색 창에서 클러스터(Clusters)를 선택하는 경우, 콘솔에 Amazon EKS 클러스터 목록이 표시됩니다. 이러한 목록 중 하나(예: 특정 클러스터)에서 리소스를 선택할 때 해당 리소스가 태그를 지원하면 태그(Tags) 탭에서 해당 태그를 보고 관리할 수 있습니다.

또한 태그를 관리할 수 있는 통합 방법을 제공하는 태그 편집기(Tag Editor)를 AWS Management Console에서 사용할 수 있습니다. 자세한 내용은 AWS Tag Editor 사용 안내서의 Tag Editor로 AWS 리소스 태그 지정을 참조하세요.

생성 시 리소스에 대한 태그 추가

Amazon EKS 클러스터, 관리형 노드 그룹 및 Fargate 프로필을 생성할 때 태그를 추가할 수 있습니다. 자세한 내용은 Amazon EKS 클러스터 생성 섹션을 참조하세요.

리소스에 대한 태그 추가 및 삭제

리소스 페이지에서 직접 클러스터와 연결된 태그를 추가하거나 삭제할 수 있습니다.

개별 리소스에서 태그를 추가하거나 삭제하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 탐색 모음에서 사용할 AWS 리전을 선택합니다.
3. 좌측 탐색 창에서 클러스터를 선택합니다.
4. 특정 클러스터를 선택합니다.
5. 태그(Tags) 탭을 선택한 후 태그 관리(Manage tags)를 선택합니다.
6. 태그 관리(Manage tags) 페이지에서 필요에 따라 태그를 추가하거나 삭제합니다.
   - 태그를 추가하려면 태그 추가(Add tag)를 선택합니다. 그런 다음 각 태그의 키와 값을 지정합니다.
   - 태그를 삭제하려면 Remove tag(태그 제거)를 선택합니다.
7. 추가 또는 삭제하려는 각 태그에 대해 이 프로세스를 반복합니다.
8. Update(업데이트)를 선택하여 완료합니다.

CLI, API 또는 `eksctl`를 사용한 태그 작업

다음 AWS CLI 명령 또는 Amazon EKS API 작업을 사용하여 리소스에 대한 태그를 추가, 업데이트, 나열 및 삭제합니다. `eksctl`만 사용하여 하나의 명령으로 새 리소스를 생성하는 동시에 태그를 추가할 수 있습니다.

Amazon EKS 리소스 태깅 지원

작업	AWS CLI	AWS Tools for Windows PowerShell	API 작업
하나 이상의 태그를 추가하거나 덮어씁니다.	`tag-resource`	`Add-EKSResourceTag`	`TagResource`
하나 이상의 태그를 삭제합니다.	`untag-resource`	`Remove-EKSResourceTag`	`UntagResource`

다음 예제는 AWS CLI를 사용하여 리소스에 태그를 지정하거나 태그를 제거하는 방법을 보여줍니다.

예제 1: 기존 클러스터에 태그 지정

다음 명령은 기존 클러스터에 태그를 지정합니다.

```
aws eks tag-resource --resource-arn resource_ARN --tags team=devs
```

예제 2: 기존 클러스터에서 태그 제거

다음 명령은 기존 클러스터에서 태그를 삭제합니다.

```
aws eks untag-resource --resource-arn resource_ARN --tag-keys tag_key
```

예제 3: 리소스의 태그 나열

다음 명령은 기존 리소스와 연결된 태그를 나열합니다.

```
aws eks list-tags-for-resource --resource-arn resource_ARN
```

일부 리소스 생성 작업을 사용할 때 리소스를 생성하는 동시에 태그를 지정할 수 있습니다. 리소스를 생성할 때 태그를 지원하는 작업은 다음과 같습니다.

작업	AWS CLI	AWS Tools for Windows PowerShell	API 작업	**eksctl**
클러스터 생성	create-cluster	New-EKSCluster	CreateCluster	create cluster
관리형 노드 그룹 생성*	create-nodegroup	New-EKSNodegroup	CreateNodegroup	create nodegroup
Fargate 프로파일 생성	create-fargate-profile	New-EKSFargateProfile	CreateFargateProfile.html	create fargateprofile

* 관리형 노드 그룹을 생성할 때 Amazon EC2 인스턴스에도 태그를 지정하려면 시작 템플릿을 사용하여 관리형 노드 그룹을 생성합니다. 자세한 내용은 Amazon EC2 인스턴스 태깅 섹션을 참조하세요. 인스턴스가 이미 있는 경우 인스턴스에 수동으로 태깅할 수 있습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서에서 리소스 태깅을 참조하세요.

Amazon EKS 서비스 할당량

Amazon EKS는 중앙 위치에서 할당량을 보고 관리할 수 있는 AWS 서비스인 Service Quotas와 통합됩니다. 자세한 내용은 Service Quotas 사용 설명서의 Service Quotas는 무엇입니까?를 참조하세요. Service Quotas 통합과 함께 AWS Management Console 및 AWS CLI를 사용하여 Amazon EKS 및 AWS Fargate 서비스 할당량 값을 빠르게 조회할 수 있습니다.

AWS Management Console

AWS Management Console을 사용하여 Amazon EKS 및 Fargate 서비스 할당량을 보려면

1. https://console.aws.amazon.com/servicequotas/에서 Service Quotas 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 AWS 서비스를 선택합니다.
3. AWS 서비스 목록에서 Amazon Elastic Kubernetes Service(Amazon EKS) 또는 AWS Fargate를 검색하고 선택합니다.

   Service quotas(서비스 할당량) 목록에서 서비스 할당량 이름, 적용된 값(제공된 경우), AWS 기본 할당량 및 할당량 값 조정 가능 여부를 확인할 수 있습니다.
4. 설명 등 서비스 할당량에 대한 추가 정보를 보려면 할당량 이름을 선택합니다.
5. (선택 사항) 할당량 증가를 요청하려면 증가시킬 할당량을 선택하고 할당량 증가 요청(Request quota increase)을 선택한 다음 필요한 정보를 입력하거나 선택한 다음 요청(Request)을 선택합니다.

AWS Management Console을 사용하여 서비스 할당량에 대한 추가 작업을 수행하려면 Service Quotas 사용 설명서를 참조하세요. 할당량 증가를 요청하려면 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요.

AWS CLI

AWS CLI을 사용하여 Amazon EKS 및 Fargate 서비스 할당량을 보려면

다음 명령을 실행하여 Amazon EKS 할당량을 확인합니다.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code eks \
    --output table
```

다음 명령을 실행하여 Fargate 할당량을 확인합니다.

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code fargate \
    --output table
```

> **Note**
>
> 반환된 할당량은 현재 AWS 리전의 이 계정에서 Fargate에서 동시에 실행할 수 있는 Amazon ECS 태스크 또는 Amazon EKS Pods의 수입니다.

AWS CLI를 사용하여 서비스 할당량에 대한 추가 작업을 수행하려면 AWS CLI Command Reference의 `service-quotas`를 참조하세요. 할당량 증가를 요청하려면 AWS CLI 명령 참조에서 `request-service-quota-increase` 명령을 참조하세요.

서비스 할당량

명칭	기본값	조정 가능	설명
클러스터당 액세스 항목	지원되는 각 리전: 3,000개	아니요	클러스터당 최대 액세스 항목 수입니다.
클러스터	지원되는 각 리전: 100개	예	현재 리전에 있는 이 계정의 최대 EKS 클러스터 수입니다.
클러스터당 컨트롤 플레인 보안 그룹	지원되는 각 리전: 4개	아니요	클러스터당 컨트롤 플레인 보안 그룹의 최대 수입니다(클러스터 생성 시 지정됨).

명칭	기본값	조정 가능	설명
EKS Anywhere 엔터프라이즈 구독	지원되는 각 리전: 10개	예	현재 리전에 있는 이 계정의 최대 EKS Anywhere 엔터프라이즈 구독 수입니다.
클러스터당 Fargate 프로파일	지원되는 각 리전: 10개	예	클러스터당 최대 Fargate 프로파일 수입니다.
Fargate 프로파일 셀렉터당 레이블 페어	지원되는 각 리전: 5개	예	Fargate 프로파일 셀렉터당 최대 레이블 페어 수입니다.
클러스터당 관리형 노드 그룹	지원되는 각 리전: 30개	예	클러스터당 최대 관리형 노드 그룹 수입니다.
관리형 노드 그룹당 노드	지원되는 각 리전: 450개	예	관리형 노드 그룹당 최대 노드 수입니다.
클러스터당 퍼블릭 엔드포인트 액세스 CIDR 범위	지원되는 각 리전: 40개	아니요	클러스터당 퍼블릭 엔드포인트 액세스 CIDR 범위의 최대 수입니다(클러스터 생성 또는 업데이트 시 지정됨).
등록된 클러스터	지원되는 각 리전: 10개	예	현재 리전에 있는 이 계정의 최대 등록된 클러스터 수입니다.
Fargate 프로파일당 셀렉터	지원되는 각 리전: 5개	예	Fargate 프로파일당 최대 셀렉터 수입니다.

> **Note**
>
> 기본값은 AWS에서 설정한 초기 할당량입니다. 이러한 기본값은 실제 적용된 할당량 값 및 가능한 최대 서비스 할당량과 별개입니다. 자세한 내용은 Service Quotas 사용 설명서의 Service Quotas 용어를 참조하십시오.

이러한 서비스 할당량은 Service Quotas 콘솔의 Amazon Elastic Kubernetes Service(Amazon EKS) 아래에 나열됩니다. 조정 가능한 것으로 표시되는 값에 대한 할당량 증가를 요청하려면 Service Quotas User Guide(Service Quotas 사용 설명서)의 할당량 증가 요청을 참조하십시오.

AWS Fargate 서비스 할당량

Service Quotas 콘솔의 AWS Fargate 서비스는 다양한 서비스 할당량을 나열합니다. 다음 표에서는 Amazon EKS에 적용되는 할당량만 설명합니다. 사용량이 서비스 할당량에 가까워지면 경고하는 경보를 구성할 수 있습니다. 자세한 내용은 Fargate 리소스 사용량 지표 모니터링을 위한 CloudWatch 경보 생성 단원을 참조하십시오.

새 AWS 계정 계정의 초기 할당량은 낮지만 시간이 지남에 따라 증가할 수 있습니다. Fargate는 각 AWS 리전 내의 계정 사용량을 지속적으로 모니터링한 다음 사용량에 따라 할당량을 자동으로 늘립니다. 조정 가능한 것으로 표시된 값에 대한 할당량 증가를 요청할 수도 있습니다. 자세한 내용은 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하십시오.

명칭	기본값	조정 가능	설명
Fargate 온디맨드 vCPU 리소스 수	6	예	현재 지역의 이 계정에서 Fargate 온디맨드로 동시에 실행할 수 있는 Fargate vCPU 수입니다.

> **Note**
>
> 기본값은 AWS에서 설정한 초기 할당량입니다. 이러한 기본값은 실제 적용된 할당량 값 및 가능한 최대 서비스 할당량과 별개입니다. 자세한 내용은 Service Quotas 사용 설명서의 Service Quotas 용어를 참조하십시오.

> **Note**
>
> Fargate는 Amazon ECS 태스크 및 Amazon EKS Pods 시작률 할당량을 추가로 시행합니다. 자세한 내용은 Amazon ECS 가이드의 AWS Fargate 스로틀링 할당량을 참조하십시오.

Amazon EKS의 보안

AWS는 클라우드 보안을 가장 중요하게 생각합니다. AWS 고객으로서 여러분은 가장 높은 보안 요구 사항을 충족하기 위해 설계된 데이터 센터 및 네트워크 아키텍처의 혜택을 받게 됩니다.

보안은 AWS과 사용자의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

- 클라우드의 보안 – AWS는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라 보호를 책임집니다. Amazon EKS의 경우 AWS가 컨트롤 플레인 노드와 `etcd` 데이터베이스를 포함한 Kubernetes 컨트롤 플레인을 담당합니다. 서드 파티 감사원은 정기적으로 AWS규정 준수 프로그램의 일환으로 보안 효과를 테스트하고 검증합니다. Amazon EKS에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하세요.
- 클라우드 내부의 보안 - 고객의 책임에는 다음 영역이 포함됩니다.
  - Amazon EKS 제어 영역에서 고객 VPC로 트래픽을 전달할 수 있는 보안 그룹의 구성을 포함한 데이터 영역의 보안 구성
  - 노드와 컨테이너 자체의 구성
  - 노드 운영 체제(업데이트 및 보안 패치 포함)
  - 기타 관련 애플리케이션 소프트웨어:
    - 방화벽 규칙과 같은 네트워크 컨트롤 설정 및 관리
    - IAM을 단독 또는 추가로 이용하여 플랫폼 수준의 자격 증명 및 액세스 관리
  - 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정

이 문서는 Amazon EKS를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목적에 맞게 Amazon EKS를 구성하는 방법을 보여줍니다. 또한 Amazon EKS 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 배우게 됩니다.

> **Note**
>
> Linux 컨테이너는 컨테이너가 액세스할 수 있는 항목을 제한하는 데 도움이 되는 제어 그룹(cgroups)과 네임스페이스로 구성되지만 모든 컨테이너는 호스트 Amazon EC2 인스턴스와 동일한 Linux 커널을 공유합니다. 컨테이너를 루트 사용자(UID 0)로 실행하거나 호스트 네트워크 또는 호스트 PID 네임스페이스와 같은 호스트 리소스 또는 네임스페이스에 대한 액세스 권한

을 컨테이너에 부여하는 것은 권장하지 않습니다. 이렇게 하면 컨테이너가 제공하는 격리의 효과가 떨어지기 때문입니다.

주제

인증서 서명

Kubernetes Certificates API를 통해 X.509 보안 인증 프로비저닝이 자동화됩니다. CA(인증 기관)에서 X.509 인증서를 요청하고 획득할 Kubernetes API 클라이언트의 명령줄 인터페이스가 API에서 추천됩니다. 표시된 서명자가 인증서에 서명하도록 요청하는 데 `CertificateSigningRequest`(CSR) 리소스를 사용할 수 있습니다. 요청은 서명되기 전에 승인 또는 거부됩니다. Kubernetes에서는 잘 정의된 동작이 있는 기본 제공 서명자와 사용자 지정 서명자를 모두 지원합니다. 이렇게 하면 클라이언트는 CSR에 어떤 일이 일어나는지 예측할 수 있습니다. 인증서 서명에 대한 자세한 내용을 알아보려면 서명 요청을 참조하세요.

기본 제공 서명자 중 하나는 `kubernetes.io/legacy-unknown`입니다. CSR 리소스의 `v1beta1` API는 이 legacy-unknown 서명자를 준수했습니다. 그러나 CSR의 안정적인 `v1` API에서는 `signerName`이 `kubernetes.io/legacy-unknown`으로 설정되도록 허용하지 않습니다.

Amazon EKS `1.21` 및 이전 버전에서는 `legacy-unknown` 값을 `v1beta1` CSR API의 `signerName`으로 허용했습니다. 이 API를 사용하면 Amazon EKS CA(인증 기관)에서 인증서를 생성할 수 있습니다. 그러나 Kubernetes 버전 `1.22`에서는 `v1beta1` CSR API가 `v1` CSR API로 바뀌었습니다. 이 API는 “legacy-unknown”의 signerName을 지원하지 않습니다. Amazon EKS CA를 사용하여

클러스터에서 인증서를 생성하려는 경우 사용자 지정 서명자를 사용해야 합니다. Amazon EKS 버전 1.22에 도입되었습니다. CSR v1 API 버전을 사용하여 새 인증서를 생성하려면 기존 매니페스트 및 API 클라이언트를 마이그레이션해야 합니다. 기존 v1beta1 API를 사용하여 생성된 기존 인증서는 유효하며 인증서가 만료될 때까지 기능합니다. 다음 내용이 포함됩니다:

- 신뢰 배포: 없음. Kubernetes 클러스터에는 이 서명자에 대한 표준 트러스트나 배포가 없습니다.
- 허용된 과목: 모두
- 허용된 x509 확장 기능: subjectAltName 및 키 사용 확장 기능을 준수하고 다른 확장 기능을 폐기합니다.
- 허용된 키 사용: ["키 암호화", "디지털 서명", "서버 인증"] 이외의 용도는 포함하지 않아야 합니다.

> **Note**
>
> 클라이언트 인증서 서명은 지원되지 않습니다.

- 만료/인증서 수명: 1년(기본 및 최대)
- CA 비트 허용됨/허용되지 않음: 허용되지 않음

signerName을 사용한 CSR 생성 예

다음 단계에서는 signerName: beta.eks.amazonaws.com/app-serving을 사용하여 myserver.default.svc DNS 이름에 대한 서빙 인증서를 생성하는 방법을 보여줍니다. 이 정보를 사용자 환경에 대한 가이드로 사용합니다.

1. openssl genrsa -out myserver.key 2048 명령을 실행하여 RSA 프라이빗 키를 생성합니다.

```
openssl genrsa -out myserver.key 2048
```

2. 다음 명령을 실행하여 인증 요청을 생성합니다.

```
openssl req -new -key myserver.key -out myserver.csr -subj "/
CN=myserver.default.svc"
```

3. CSR 요청에 대한 base64 값을 생성하고 이후 단계에서 사용할 수 있도록 변수에 저장합니다.

```
base_64=$(cat myserver.csr | base64 -w 0 | tr -d "\n")
```

4. 다음 명령을 실행하여 mycsr.yaml이라는 파일을 생성합니다. 다음 예에서 beta.eks.amazonaws.com/app-serving은 signerName입니다.

```
cat >mycsr.yaml <<EOF
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: myserver
spec:
  request: $base_64
  signerName: beta.eks.amazonaws.com/app-serving
  usages:
    - digital signature
    - key encipherment
    - server auth
EOF
```

5. CSR을 제출합니다.

```
kubectl apply -f mycsr.yaml
```

6. 서빙 인증서를 승인합니다.

```
kubectl certificate approve myserver
```

7. 인증서가 발급되었는지 확인합니다.

```
kubectl get csr myserver
```

예제 출력은 다음과 같습니다.

```
NAME       AGE     SIGNERNAME                              REQUESTOR
 CONDITION
myserver   3m20s   beta.eks.amazonaws.com/app-serving   kubernetes-admin
 Approved,Issued
```

8. 발급된 인증서를 내보냅니다.

```
kubectl get csr myserver -o jsonpath='{.status.certificate}'| base64 -d
 > myserver.crt
```

클러스터를 Kubernetes 1.24로 업그레이드하기 전의 인증서 서명 고려 사항

Kubernetes `1.23` 및 이전 버전에서는 `kubelet`에서 제공되는 확인할 수 없는 IP와 DNS SAN(주체 대체 이름)이 있는 인증서가 확인할 수 없는 SAN과 함께 자동으로 발급됩니다. SAN은 프로비저닝된 인증서에서 생략됩니다. `1.24` 및 이후 버전의 클러스터에서는 SAN을 확인할 수 없는 경우 `kubelet`에서 제공되는 인증서가 발급되지 않습니다. 이렇게 하면 `kubectl exec` 및 `kubectl logs` 명령이 작동하지 않습니다.

클러스터를 `1.24`(으)로 업그레이드하기 전에 다음 단계를 완료하여 승인되지 않은 CSR(인증서 서명 요청)이 있는지 확인하세요.

1. 다음 명령을 실행합니다.

```
kubectl get csr -A
```

예제 출력은 다음과 같습니다.

```
NAME        AGE   SIGNERNAME                            REQUESTOR
                          REQUESTEDDURATION   CONDITION
csr-7znmf   90m   kubernetes.io/kubelet-serving
 system:node:ip-192-168-42-149.region.compute.internal      <none>
 Approved
csr-9xx5q   90m   kubernetes.io/kubelet-serving
 system:node:ip-192-168-65-38.region.compute.internal      <none>
 Approved, Issued
```

반환된 출력에 노드에 대한 `Issued`이(가) 아니라 `Approved`인 `kubernetes.io/kubelet-serving` 서명자와 함께 CSR이 표시되면 요청을 승인해야 합니다.

2. CSR을 수동으로 승인합니다. `csr-7znmf`를 사용자의 고유한 값으로 교체합니다.

```
kubectl certificate approve csr-7znmf
```

앞으로 CSR을 자동 승인하려면 Amazon EKS에서 확인할 수 없는 IP 또는 DNS SAN이 포함된 CSR을 자동으로 검증하고 승인할 수 있는 승인 컨트롤러를 작성하는 것이 좋습니다.

Amazon EKS용 자격 증명 및 액세스 관리

AWS Identity and Access Management(IAM)는 관리자가 AWS리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 어떤 사용자가 Amazon EKS 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지를 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

고객

AWS Identity and Access Management(IAM)를 사용하는 방법은 Amazon EKS에서 수행하는 작업에 따라 달라집니다.

서비스 사용자 – Amazon EKS 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 다른 Amazon EKS 기능을 사용하여 작업을 수행한다면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. Amazon EKS의 기능에 액세스할 수 없다면 IAM 문제 해결 부분을 참조하세요.

서비스 관리자 – 사내 Amazon EKS 리소스를 책임지고 있다면 Amazon EKS에 대한 완전한 액세스 권한이 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 Amazon EKS 기능과 리소스를 결정합니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해합니다. 귀사에서 Amazon EKS와 함께 IAM을 사용하는 방법에 대해 자세히 알아보려면 Amazon EKS가 IAM과 작동하는 방식 부분을 참조하세요.

IAM 관리자 - IAM 관리자라면 Amazon EKS에 대한 액세스 관리 정책 작성 방법을 자세히 알고 싶을 수도 있습니다. IAM에서 사용할 수 있는 Amazon EKS 자격 증명 기반 정책 예제를 보려면 Amazon EKS 자격 증명 기반 정책 예제 부분을 참조하세요.

자격 증명을 통한 인증

인증은 ID 보안 인증을 사용하여 AWS에 로그인하는 방식입니다. AWS 계정 루트 사용자나 IAM 사용자로 또는 IAM 역할을 수임하여 인증(AWS에 로그인)되어야 합니다.

자격 증명 소스 AWS IAM Identity Center를 통해 제공된 보안 인증 정보를 사용하여 연동 ID로 AWS에 로그인할 수 있습니다. (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증, Google 또는 Facebook 자격 증명이 연동 ID의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 연동을 사용하여 AWS에 액세스하면 간접적으로 역할을 수임합니다.

사용자 유형에 따라 AWS Management Console 또는AWS 액세스 포털에 로그인할 수 있습니다. AWS에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인사용 설명서의 AWS 계정에 로그인하는 방법을 참조하십시오.

AWS에 프로그래밍 방식으로 액세스하는 경우, AWS에서는 보안 인증 정보를 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK) 및 명령줄 인터페이스(CLI)를 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청에 서명을 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS은 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center사용 설명서의 다중 인증 및 IAM 사용 설명서의 AWS에서 다중 인증(MFA) 사용을 참조하세요.

AWS 계정 루트 사용자

AWS 계정을 생성할 때는 해당 계정의 모든 AWS 서비스및 리소스에 대한 완전한 액세스 권한이 있는 단일 로그인 ID로 시작합니다. 이 ID는 AWS 계정 루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 전체 작업 목록은 IAM 사용 설명서의 루트 사용자 보안 인증이 필요한 태스크를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정내 ID입니다. 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 사용자 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 보안 인증 정보를 가지고 있지만, 역할은 임시 보안 인증 정보만 제공합니다. 자세한 정보는 IAM 사용 설명서의 IAM 사용자(역할을 대신하여)를 만들어야 하는 경우를 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가지고 있는 AWS 계정계정 내 ID입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할 전환하여 AWS Management Console에서 IAM 역할을 임시로 수임할 수 있습니다. AWS CLI 또는 AWSAPI 태스크를 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용자 설명서의 IAM 역할 사용을 참조하세요.

임시 보안 인증 정보가 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 연동 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 서드 파티 자격 증명 공급자의 역할 만들기 부분을 참조하세요. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 자격 증명이 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center사용 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 서비스를 사용하면 정책을 리소스에 직접 연결할 수 있습니다(역할을 프록시로 사용하는 대신). 크로스 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.
- 교차 서비스 액세스: 일부 AWS 서비스는 다른 AWS 서비스의 기능을 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.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 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 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.

- Amazon EC2에서 실행 중인 애플리케이션: IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 보안 인증을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS역할을 할당하고 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 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 사용자 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하세요.

정책을 사용한 액세스 관리

정책을 생성하고 AWSID 또는 리소스에 연결하여 AWS내 액세스를 제어합니다. 정책은 ID 또는 리소스와 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. AWS는 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청을 보낼 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 AWS에 JSON 문서로 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용자 설명서의 JSON 정책 개요를 참조하세요.

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는 지를 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 정책이 있는 사용자는 AWS Management Console, AWS CLI또는 AWSAPI에서 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 생성 단원을 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 AWS 계정에 속한 다수의 사용자, 그룹 및 역할에 독립적으로 추가할 수 있는 정책입니다. 관리형 정책에는 AWS관리형 정책과 고객 관리형 정책이 포함되어 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용자 설명서의 관리형 정책과 인라인 정책 사이의 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 또는 AWS 서비스가 포함될 수 있습니다.

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

액세스 제어 목록(ACLs)

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.

Amazon S3, AWS WAF 및 Amazon VPC는 ACL을 지원하는 대표적인 서비스입니다. ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS는 비교적 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 ID 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 사용자 설명서의 IAM 엔터티에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCP): SCP는 AWS Organizations에서 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations는 기업이 소유하는 여러 개의 AWS 계정을 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 특성을 활성화할 경우, 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 AWS 계정 루트 사용자을 비롯하여 멤버 계정의 엔터티에 대한 권한을 제한합니다. 조직 및 SCP에 대한 자세한 정보는 AWS Organizations사용 설명서의 SCP 작동 방식을 참조하십시오.
- 세션 정책: 세션 정책은 역할 또는 연합된 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 사용자 설명서의 세션 정책을 참조하세요.

여러 정책 타입

여러 정책 타입이 요청에 적용되는 경우 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련될 때 AWS가 요청을 허용할지를 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

Amazon EKS가 IAM과 작동하는 방식

IAM을 사용하여 Amazon EKS에 대한 액세스를 관리하기 전에 Amazon EKS에서 사용할 수 있는 IAM 기능을 이해해야 합니다. Amazon EKS 및 기타 AWS 서비스에서 IAM을 사용하는 방법을 자세히 알아보려면 IAM 사용 설명서의 IAM으로 작업하는 AWS 서비스를 참조하세요.

주제

- Amazon EKS 자격 증명 기반 정책
- Amazon EKS 리소스 기반 정책
- Amazon EKS 태그를 기반으로 권한 부여
- Amazon EKS IAM 역할

Amazon EKS 자격 증명 기반 정책

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업와 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. Amazon EKS는 특정 작업, 리소스 및 조건 키를 지원합니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하십시오.

작업

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는 지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 일반적으로 정책 작업의 이름은 연결된 AWSAPI 작업의 이름과 동일합니다. 일치하는 API 작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 종속 작업이라고 합니다.

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

Amazon EKS의 정책 작업은 작업 앞에 `eks:` 접두사를 사용합니다. 예를 들어 Amazon EKS 클러스터에 대한 기술 정보를 다운로드할 수 있는 권한을 부여하려면 정책에 `DescribeCluster` 작업을 포함합니다. 정책 문에는 `Action` 또는 `NotAction` 요소가 포함되어야 합니다.

명령문 하나에 여러 태스크를 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": ["eks:action1", "eks:action2"]
```

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

```
"Action": "eks:Describe*"
```

Amazon EKS 작업의 목록을 보려면 서비스 인증 참조의 Amazon Elastic Kubernetes Service에서 정의한 작업을 참조하세요.

리소스

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

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 개체를 지정합니다. 문장에는 `Resource`또는 `NotResource`요소가 반드시 추가되어야 합니다. 모범 사례에 따라 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 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": "*"
```

Amazon EKS 클러스터 리소스의 ARN은 다음과 같습니다.

```
arn:aws:eks:region-code:account-id:cluster/cluster-name
```

ARN 형식에 대한 자세한 내용은 Amazon 리소스 이름(ARN) 및 AWS 서비스 네임스페이스를 참조하세요.

예를 들어, 명령문에 *my-cluster*라는 이름의 클러스터를 지정하려면 다음 ARN을 사용합니다.

```
"Resource": "arn:aws:eks:region-code:111122223333:cluster/my-cluster"
```

특정 계정 및 AWS 리전에 속하는 모든 클러스터를 지정하려면 와일드카드(*)를 사용합니다.

```
"Resource": "arn:aws:eks:region-code:111122223333:cluster/*"
```

리소스 생성 작업과 같은 일부 Amazon EKS 작업은 특정 리소스에서 수행할 수 없습니다. 이러한 경우, 와일드카드(*)를 사용해야 합니다.

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

Amazon EKS 리소스 유형 및 해당 ARN의 목록을 보려면 서비스 인증 참조의 Amazon Elastic Kubernetes Service에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon Elastic Kubernetes Service에서 정의한 작업을 참조하세요.

조건 키

Amazon EKS는 자체 조건 키 세트를 정의하며 일부 글로벌 조건 키 사용도 지원합니다. 모든 AWS전역 조건 키를 보려면 IAM 사용 설명서의 AWS전역 조건 컨텍스트 키를 참조하세요.

OpenID Connect 공급자를 클러스터에 연결할 때 조건 키를 설정할 수 있습니다. 자세한 정보는 IAM 정책 예제을 참조하세요.

모든 Amazon EC2 작업은 `aws:RequestedRegion` 및 `ec2:Region` 조건 키를 지원합니다. 자세한 내용은 예제: 특정 AWS 리전으로 액세스 제한을 참조하세요.

Amazon EKS 조건 키 목록을 보려면 서비스 인증 참조의 Amazon Elastic Kubernetes Service의 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon Elastic Kubernetes Service에서 정의한 작업을 참조하세요.

예제

Amazon EKS 자격 증명 기반 정책 예제를 보려면 Amazon EKS 자격 증명 기반 정책 예제 섹션을 참조하세요.

Amazon EKS 클러스터를 생성할 경우 클러스터를 생성하는 IAM 보안 주체에게는 Amazon EKS 제어 영역의 클러스터 역할 기반 액세스 제어(RBAC) 구성에 `system:masters` 권한이 자동으로 부여됩니다. 이 보안 주체는 표시되는 구성에 나타나지 않으므로 클러스터를 원래 생성한 보안 주체를 추적해야 합니다. 추가 IAM 보안 주체에 클러스터와 상호 작용할 수 있는 기능을 부여하려면 Kubernetes 내에서 `aws-auth ConfigMap`을 편집하고 `aws-auth ConfigMap`에 지정하는 `group`의 이름으로 Kubernetes `rolebinding` 또는 `clusterrolebinding`을 생성해야 합니다.

ConfigMap 작업에 대한 자세한 내용은 Kubernetes API에 대한 액세스 권한 부여 세션을 참조하세요.

Amazon EKS 리소스 기반 정책

Amazon EKS는 리소스 기반 정책을 지원하지 않습니다.

Amazon EKS 태그를 기반으로 권한 부여

Amazon EKS 리소스에 태그를 연결하거나 Amazon EKS에 대한 요청에서 태그를 전달할 수 있습니다. 태그를 기반으로 액세스를 제어하려면 `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`* 또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다. Amazon EKS 리소스 태깅에 대한 자세한 내용은 Amazon EKS 리소스 태깅 섹션을 참조하세요. 조건 키의 태그를 사용할 수 있는 작업에 대한 자세한 내용은 Service Authorization Reference에서 Amazon EKS가 정의한 작업을 참조하세요.

Amazon EKS IAM 역할

IAM 역할은 특정 권한을 가지고 있는 AWS계정 내 엔터티입니다.

Amazon EKS에서 임시 자격 증명 사용

임시 보안 인증을 사용하여 연동을 통해 로그인하거나, IAM 역할을 맡거나, 교차 계정 역할을 맡을 수 있습니다. AssumeRole 또는 GetFederationToken 같은 AWS STS API 작업을 호출하여 임시 보안 보안 인증을 가져옵니다.

Amazon EKS는 임시 자격 증명 사용을 지원합니다.

서비스 연결 역할

서비스 연결 역할을 사용하면 AWS 서비스에서 다른 서비스의 리소스에 액세스하여 사용자 대신 작업을 완료할 수 있습니다. 서비스 연결 역할은 IAM 계정에 나타나고 서비스가 소유합니다. 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

Amazon EKS는 서비스 연결 역할을 지원합니다. Amazon EKS 서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 Amazon EKS에 대해 서비스 연결 역할 사용 섹션을 참조하세요.

서비스 역할

이 기능을 사용하면 서비스가 사용자를 대신하여 서비스 역할을 수임할 수 있습니다. 이 역할을 사용하면 서비스가 다른 서비스의 리소스에 액세스해 사용자를 대신해 작업을 완료할 수 있습니다. 서비스 역할은 IAM 계정에 나타나고, 해당 계정이 소유합니다. 즉, IAM 관리자가 이 역할에 대한 권한을 변경할 수 있습니다. 그러나 권한을 변경하면 서비스의 기능이 손상될 수 있습니다.

Amazon EKS는 서비스 역할을 지원합니다. 자세한 내용은 Amazon EKS 클러스터 IAM 역할 및 Amazon EKS 노드 IAM 역할 섹션을 참조하세요.

Amazon EKS에서 IAM 역할 선택

Amazon EKS에서 클러스터 리소스를 생성할 경우 Amazon EKS가 사용자 대신 다른 여러 AWS 리소스에 액세스하도록 허용하는 역할을 선택해야 합니다. 이전에 서비스 역할을 생성한 경우 Amazon EKS는 선택할 수 있는 역할 목록을 제공합니다. Amazon EKS 관리형 정책이 연결된 역할을 선택하는 것이 중요합니다. 자세한 내용은 기존 클러스터 역할 확인 및 기존 노드 역할 확인 섹션을 참조하세요.

Amazon EKS 자격 증명 기반 정책 예제

기본적으로 IAM 사용자 및 역할은 Amazon EKS 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS CLI또는 AWSAPI를 사용해 태스크를 수행할 수 없습니다. IAM 관리자는 지정된 리소스에서 특정 API 태스크를 수행할 수 있는 권한을 사용자와 역할에게 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 IAM 사용자 또는 그룹에 이러한 정책을 연결해야 합니다.

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

Amazon EKS 클러스터를 생성할 경우 클러스터를 생성하는 IAM 보안 주체에게는 Amazon EKS 제어 영역의 클러스터 역할 기반 액세스 제어(RBAC) 구성에 `system:masters` 권한이 자동으로 부여됩니다. 이 보안 주체는 표시되는 구성에 나타나지 않으므로 클러스터를 원래 생성한 보안 주체를 추적해야 합니다. 추가 IAM 보안 주체에 클러스터와 상호 작용할 수 있는 기능을 부여하려면 Kubernetes 내에서 `aws-auth ConfigMap`을 편집하고 `aws-auth ConfigMap`에 지정하는 `group`의 이름으로 Kubernetes `rolebinding` 또는 `clusterrolebinding`을 생성해야 합니다.

ConfigMap 작업에 대한 자세한 내용은 Kubernetes API에 대한 액세스 권한 부여 세션을 참조하세요.

주제

- 정책 모범 사례
- Amazon EKS 콘솔 사용
- IAM 사용자가 자체 권한을 볼 수 있도록 허용
- AWS 클라우드에서 Kubernetes 클러스터 생성
- Outpost에서 로컬 Kubernetes 클러스터 생성
- Kubernetes 클러스터 업데이트
- 모든 클러스터 나열 또는 설명

정책 모범 사례

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

- AWS 관리형 정책으로 시작하고 최소 권한을 향해 나아가기: 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. AWS 계정에서 사용할 수 있습니다. 사용 사례에 고유한 AWS고객 관리형 정책을 정의하여 권한을 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. AWS CloudFormation와 같이, 특정 AWS 서비스를 통해 사용되는 경우에만 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 정보는 IAM 사용자 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.
- 다중 인증(MFA) 필요: AWS 계정 계정에 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 설정합니다. 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 사용자 설명서의 IAM의 보안 모범 사례를 참조하세요.

Amazon EKS 콘솔 사용

Amazon EKS 콘솔에 액세스하려면 IAM 보안 주체는 최소한의 권한 집합이 있어야 합니다. 이러한 권한은 보안 주체가 AWS 계정에서 Amazon EKS 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용합니다. 최소 필수 권한보다 더 제한적인 보안 인증 정보 기반 정책을 만들면 콘솔이 해당 정책이 연결된 보안 주체에 대해 의도대로 작동하지 않습니다.

이러한 IAM 보안 주체가 Amazon EKS 콘솔을 계속 사용할 수 있도록 하려면 `AmazonEKSAdminPolicy`와 같은 고유한 자체 이름으로 정책을 생성합니다. 정책을 보안 주체에게 연결하세요. 자세한 정보는 IAM 사용 설명서의 IAM 자격 증명 권한 추가 및 제거 섹션을 참조하세요.

> **Important**
>
> 다음 정책 예를 사용하면 보안 주체는 콘솔의 구성 탭에 표시되는 정보를 볼 수 있습니다. AWS Management Console에 있는 개요와 리소스 탭에서 정보를 보려면 보안 주체에는 Kubernetes 권한도 필요합니다. 자세한 내용은 필요한 권한 단원을 참조하십시오.

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

```
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "eks.amazonaws.com"
                }
            }
        }
    ]
}
```

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

IAM 사용자가 자체 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 AWS CLI나 AWSAPI를 사용하여 프로그래밍 방식으로 이 태스크를 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
```

```
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 클라우드에서 Kubernetes 클러스터 생성

이 예제 정책에는 *us-west-2* AWS 리전에서 *my-cluster*라는 Amazon EKS 클러스터를 생성하는 데 필요한 최소 권한이 포함되어 있습니다. AWS 리전을 클러스터를 생성할 AWS 리전으로 바꿀 수 있습니다. AWS Management Console에서 정책의 작업이 리소스 수준 권한을 지원하지 않으므로 **`All resources`**를 선택해야 합니다.라는 경고가 표시되는 경우 이 오류는 무시해도 안전합니다. 계정에 이미 *AWSServiceRoleForAmazonEKS* 역할이 있는 경우 정책에서 `iam:CreateServiceLinkedRole` 작업을 제거할 수 있습니다. 계정에 Amazon EKS 클러스터를 생성한 적이 있다면 삭제하지 않는 한 이 역할이 이미 존재합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/
eks.amazonaws.com/AWSServiceRoleForAmazonEKS",
```

```
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "iam:AWSServiceName": "eks"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        }
    ]
}
```

Outpost에서 로컬 Kubernetes 클러스터 생성

이 예제 정책에는 *us-west-2* AWS 리전의 Outpost에서 *my-cluster*라는 Amazon EKS 로컬 클러스터를 생성하는 데 필요한 최소 권한이 포함되어 있습니다. AWS 리전을 클러스터를 생성할 AWS 리전으로 바꿀 수 있습니다. AWS Management Console에서 정책의 작업이 리소스 수준 권한을 지원하지 않으므로 **All resources**를 선택해야 합니다.라는 경고가 표시되는 경우 이 오류는 무시해도 안전합니다. 계정에 이미 AWSServiceRoleForAmazonEKSLocalOutpost 역할이 있는 경우 정책에서 iam:CreateServiceLinkedRole 작업을 제거할 수 있습니다. 계정의 Outpost에 Amazon EKS 로컬 클러스터를 생성한 적이 있다면 삭제하지 않는 한 이 역할이 이미 존재합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:CreateCluster",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        },
        {
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "iam:GetRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
```

```
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::111122223333:role/aws-service-role/outposts.eks-
local.amazonaws.com/AWSServiceRoleForAmazonEKSLocalOutpost"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole",
                "iam:ListAttachedRolePolicies"
            ]
            "Resource": "arn:aws:iam::111122223333:role/cluster-role-name"
        },
        {
            "Action": [
                "iam:CreateInstanceProfile",
                "iam:TagInstanceProfile",
                "iam:AddRoleToInstanceProfile",
                "iam:GetInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:RemoveRoleFromInstanceProfile"
            ],
            "Resource": "arn:aws:iam::*:instance-profile/eks-local-*",
            "Effect": "Allow"
        },
    ]
}
```

Kubernetes 클러스터 업데이트

이 예제 정책에는 us-west-2 AWS 리전에서 *my-cluster*라는 클러스터를 업데이트하는 데 필요한 최소 권한이 포함되어 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "eks:UpdateClusterVersion",
            "Resource": "arn:aws:eks:us-west-2:111122223333:cluster/my-cluster"
        }
    ]
}
```

모든 클러스터 나열 또는 설명

이 예제 정책에는 계정의 모든 클러스터를 나열하고 설명하는 데 필요한 최소 권한이 포함되어 있습니다. IAM 보안 주체는 `update-kubeconfig` AWS CLI 명령을 사용하기 위해 클러스터를 나열하고 설명할 수 있어야 합니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:DescribeCluster",
                "eks:ListClusters"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon EKS에 대해 서비스 연결 역할 사용

Amazon Elastic Kubernetes Service는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Amazon EKS에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon EKS에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.

주제

- Amazon EKS 클러스터에 대한 역할 사용
- Amazon EKS 노드 그룹에 대한 역할 사용
- Amazon EKS Fargate 프로필에 역할 사용
- 역할을 사용하여 Amazon EKS에 Kubernetes 클러스터 연결
- Outpost에서 Amazon EKS 로컬 클러스터에 대한 역할 사용

Amazon EKS 클러스터에 대한 역할 사용

Amazon Elastic Kubernetes Service는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Amazon EKS에 직접 연결된 고유한 유형의 IAM 역할입니다. 서

비스 연결 역할은 Amazon EKS에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 Amazon EKS를 더 쉽게 설정할 수 있습니다. Amazon EKS에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한, Amazon EKS만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 삭제할 수 없기 때문에 Amazon EKS 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-Linked Role) 열에 예(Yes)가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Amazon EKS에 대한 서비스 연결 역할 권한

Amazon EKS는 `AWSServiceRoleForAmazonEKS`라는 서비스 연결 역할을 사용합니다. 이 역할은 Amazon EKS가 사용자 계정의 클러스터를 관리하도록 허용합니다. 연결된 정책은 이 역할이 네트워크 인터페이스, 보안 그룹, 로그 및 VPC와 같은 리소스를 관리하도록 허용합니다.

> **Note**
>
> `AWSServiceRoleForAmazonEKS` 서비스 연결 역할은 클러스터 생성에 필요한 역할과 다릅니다. 자세한 내용은 Amazon EKS 클러스터 IAM 역할 단원을 참조하십시오.

`AWSServiceRoleForAmazonEKS` 서비스 연결 역할은 역할을 위임하기 위해 다음 서비스를 신뢰합니다.

- `eks.amazonaws.com`

역할 권한 정책은 Amazon EKS가 지정된 리소스에서 다음 작업을 완료하도록 허용합니다.

- `AmazonEKSServiceRolePolicy`

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

Amazon EKS에 대한 서비스 연결 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console, AWS CLI 또는 AWS API에서 클러스터를 생성하면 Amazon EKS에서 서비스 연결 역할이 자동으로 생성됩니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 클러스터를 생성할 때 Amazon EKS에서는 서비스 연결 역할을 다시 생성합니다.

Amazon EKS에 대한 서비스 연결 역할 편집

Amazon EKS에서는 `AWSServiceRoleForAmazonEKS` 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 링크 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

Amazon EKS에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권합니다. 그렇게 하면 적극적으로 모니터링하거나 유지 관리하지 않은 미사용 엔터티가 없습니다. 단, 서비스 연결 역할을 정리해야 수동으로 삭제할 수 있습니다.

서비스 연결 역할을 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에서 사용되는 리소스를 삭제해야 합니다.

> **Note**
>
> 리소스를 삭제하려고 할 때 Amazon EKS 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

`AWSServiceRoleForAmazonEKS` 역할에 의해 사용되는 Amazon EKS 리소스를 삭제하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터를 선택합니다.
3. 클러스터에 노드 그룹 또는 Fargate 프로필이 있는 경우 클러스터를 삭제하기 전에 이를 삭제해야 합니다. 자세한 내용을 알아보려면 관리형 노드 그룹 삭제 및 Fargate 프로파일 삭제 섹션을 참조하세요.
4. [클러스터(Clusters)] 페이지에서 삭제하려는 클러스터를 선택하고 [삭제(Delete)]를 선택합니다.

5. 삭제 확인 창에 클러스터 이름을 입력한 다음 [삭제(Delete)]를 선택합니다.
6. 계정의 다른 클러스터에 대해 이 절차를 반복합니다. 모든 삭제 작업이 완료될 때까지 기다립니다.

수동으로 서비스 연결 역할 삭제

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 `AWSServiceRoleForAmazonEKS` 서비스 연결 역할을 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하십시오.

Amazon EKS 서비스 연결 역할을 지원하는 리전

Amazon EKS는 서비스가 제공되는 모든 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용을 알아보려면 Amazon EKS 엔드포인트 및 할당량을 참조하세요.

Amazon EKS 노드 그룹에 대한 역할 사용

Amazon EKS는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Amazon EKS에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon EKS에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 Amazon EKS를 더 쉽게 설정할 수 있습니다. Amazon EKS에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한, Amazon EKS만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 삭제할 수 없기 때문에 Amazon EKS 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-Linked Role) 열에 예(Yes)가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Amazon EKS에 대한 서비스 연결 역할 권한

Amazon EKS는 `AWSServiceRoleForAmazonEKSNodegroup`이라는 서비스 연결 역할을 사용합니다. 이 역할은 Amazon EKS가 사용자 계정의 노드 그룹을 관리하도록 허용합니다. 연결된 정책은 이 역할이 Auto Scaling 그룹, 보안 그룹, 시작 템플릿 및 IAM 인스턴스 프로필과 같은 리소스를 관리하도록 허용합니다.

`AWSServiceRoleForAmazonEKSNodegroup` 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- eks-nodegroup.amazonaws.com

역할 권한 정책은 Amazon EKS가 지정된 리소스에서 다음 작업을 완료하도록 허용합니다.

- AWSServiceRoleForAmazonEKSNodegroup

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

Amazon EKS에 대한 서비스 연결 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console, AWS CLI 또는 AWS API에서 관리형 노드 그룹을 생성하면 Amazon EKS에서 서비스 연결 역할이 자동으로 생성됩니다.

> **Important**
>
> 이러한 서비스 연결 역할은 해당 역할이 지원하는 기능을 사용하는 다른 서비스에서 작업을 완료했을 경우 계정에 나타날 수 있습니다. Amazon EKS 서비스가 서비스 연결 역할을 지원하기 시작한 2017년 1월 1일 이전에 이 서비스를 사용했다면 Amazon EKS가 사용자 계정에 AWSServiceRoleForAmazonEKSNodegroup 역할을 이미 생성했습니다. 자세한 내용은 내 IAM 계정에 표시되는 새 역할을 참조하십시오.

Amazon EKS(AWS API)에서 서비스 연결 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console, AWS CLI 또는 AWS API에서 관리형 노드 그룹을 생성하면 Amazon EKS에서 서비스 연결 역할이 자동으로 생성됩니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 다른 관리형 노드 그룹을 생성할 때 Amazon EKS가 서비스 연결 역할을 다시 생성합니다.

Amazon EKS에 대한 서비스 연결 역할 편집

Amazon EKS에서는 AWSServiceRoleForAmazonEKSNodegroup 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 링크 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

Amazon EKS에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권합니다. 그렇게 하면 적극적으로 모니터링하거나 유지 관리하지 않은 미사용 엔터티가 없습니다. 단, 서비스 연결 역할을 정리해야 수동으로 삭제할 수 있습니다.

서비스 연결 역할을 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에서 사용되는 리소스를 삭제해야 합니다.

> **Note**
>
> 리소스를 삭제하려고 할 때 Amazon EKS 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

`AWSServiceRoleForAmazonEKSNodegroup` 역할에 의해 사용되는 Amazon EKS 리소스를 삭제하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터를 선택합니다.
3. 컴퓨팅(Compute) 탭을 선택합니다.
4. 노드 그룹(Node Groups) 섹션에서 삭제할 노드 그룹을 선택합니다.
5. 삭제 확인 창에 노드 그룹 이름을 입력한 다음 [삭제(Delete)]를 선택합니다.
6. 클러스터의 다른 노드 그룹에 대해 이 절차를 반복합니다. 모든 삭제 작업이 완료될 때까지 기다립니다.

수동으로 서비스 연결 역할 삭제

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 `AWSServiceRoleForAmazonEKSNodegroup` 서비스 연결 역할을 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하십시오.

Amazon EKS 서비스 연결 역할을 지원하는 리전

Amazon EKS는 서비스가 제공되는 모든 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용을 알아보려면 Amazon EKS 엔드포인트 및 할당량을 참조하세요.

Amazon EKS Fargate 프로필에 역할 사용

Amazon EKS는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Amazon EKS에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon EKS에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 Amazon EKS를 더 쉽게 설정할 수 있습니다. Amazon EKS에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한, Amazon EKS만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 삭제할 수 없기 때문에 Amazon EKS 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-Linked Role) 열에 예(Yes)가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Amazon EKS에 대한 서비스 연결 역할 권한

Amazon EKS는 `AWSServiceRoleForAmazonEKSForFargate`라는 서비스 연결 역할을 사용합니다. 이 역할은 Amazon EKS Fargate가 Fargate Pods에 필요한 VPC 네트워킹을 구성하도록 허용합니다. 연결된 정책을 통해 역할은 탄력적 네트워크 인터페이스를 생성 및 삭제하고 탄력적 네트워크 인터페이스 및 리소스를 설명할 수 있습니다.

`AWSServiceRoleForAmazonEKSForFargate` 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `eks-fargate.amazonaws.com`

역할 권한 정책은 Amazon EKS가 지정된 리소스에서 다음 작업을 완료하도록 허용합니다.

- `AmazonEKSForFargateServiceRolePolicy`

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

Amazon EKS에 대한 서비스 연결 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console, AWS CLI 또는 AWS API에서 Fargate 프로필을 생성하면 Amazon EKS에서 서비스 연결 역할이 자동으로 생성됩니다.

> **Important**
>
> 이러한 서비스 연결 역할은 해당 역할이 지원하는 기능을 사용하는 다른 서비스에서 작업을 완료했을 경우 계정에 나타날 수 있습니다. Amazon EKS 서비스가 서비스 연결 역할을 지원하기 시작한 2019년 12월 13일 이전에 이 서비스를 사용했다면 Amazon EKS가 사용자 계정에 AWSServiceRoleForAmazonEKSForFargate 역할을 이미 생성했습니다. 자세한 내용을 알아보려면 내 IAM 계정에 표시되는 새 역할을 참조하세요.

Amazon EKS(AWS API)에서 서비스 연결 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console, AWS CLI 또는 AWS API에서 Fargate 프로필을 생성하면 Amazon EKS에서 서비스 연결 역할이 자동으로 생성됩니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 다른 관리형 노드 그룹을 생성할 때 Amazon EKS가 서비스 연결 역할을 다시 생성합니다.

Amazon EKS에 대한 서비스 연결 역할 편집

Amazon EKS에서는 `AWSServiceRoleForAmazonEKSForFargate` 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 링크 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

Amazon EKS에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권합니다. 그렇게 하면 적극적으로 모니터링하거나 유지 관리하지 않은 미사용 엔터티가 없습니다. 단, 서비스 연결 역할을 정리해야 수동으로 삭제할 수 있습니다.

서비스 연결 역할을 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에서 사용되는 리소스를 삭제해야 합니다.

> **Note**
>
> 리소스를 삭제하려고 할 때 Amazon EKS 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

`AWSServiceRoleForAmazonEKSForFargate` 역할에 의해 사용되는 Amazon EKS 리소스를 삭제하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터를 선택합니다.
3. 클러스터 페이지에서 클러스터를 선택합니다.
4. 컴퓨팅(Compute) 탭을 선택합니다.
5. Fargate 프로필(Fargate Profiles) 섹션에 Fargate 프로필이 있는 경우 각각 개별적으로 선택한 다음 삭제(Delete)를 선택합니다.
6. 삭제 확인 창에 프로필 이름을 입력한 다음 [삭제(Delete)]를 선택합니다.
7. 클러스터의 다른 Fargate 프로필과 계정의 다른 클러스터에 대해 이 절차를 반복합니다.

수동으로 서비스 연결 역할 삭제

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 AWSServiceRoleForAmazonEKSForFargate 서비스 연결 역할을 삭제합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하십시오.

Amazon EKS 서비스 연결 역할을 지원하는 리전

Amazon EKS는 서비스가 제공되는 모든 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용을 알아보려면 Amazon EKS 엔드포인트 및 할당량을 참조하세요.

역할을 사용하여 Amazon EKS에 Kubernetes 클러스터 연결

Amazon EKS는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Amazon EKS에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon

EKS에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 Amazon EKS를 더 쉽게 설정할 수 있습니다. Amazon EKS에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한, Amazon EKS만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 삭제할 수 없기 때문에 Amazon EKS 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-Linked Role) 열에 예(Yes)가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Amazon EKS에 대한 서비스 연결 역할 권한

Amazon EKS는 `AWSServiceRoleForAmazonEKSConnector`라는 서비스 연결 역할을 사용합니다. 이 역할은 Amazon EKS가 Kubernetes 클러스터를 연결하도록 허용합니다. 연결된 정책을 사용하면 역할이 등록된 Kubernetes 클러스터에 연결하는 데 필요한 리소스를 관리할 수 있습니다.

`AWSServiceRoleForAmazonEKSConnector` 서비스 연결 역할은 역할을 수임하기 위해 다음 서비스를 신뢰합니다.

- `eks-connector.amazonaws.com`

역할 권한 정책은 Amazon EKS가 지정된 리소스에서 다음 작업을 완료하도록 허용합니다.

- `AmazonEKSConnectorServiceRolePolicy`

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

Amazon EKS에 대한 서비스 연결 역할 생성

클러스터 연결을 위해 서비스 연결 역할을 수동으로 생성할 필요가 없습니다. AWS Management Console, AWS CLI, `eksctl` 또는 AWS API에서 클러스터를 연결하면 Amazon EKS에서 서비스 연결 역할이 자동으로 생성됩니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 클러스터를 연결할 때 Amazon EKS에서는 서비스 연결 역할을 다시 생성합니다.

Amazon EKS에 대한 서비스 연결 역할 편집

Amazon EKS에서는 `AWSServiceRoleForAmazonEKSConnector` 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 링크 역할을 생성한 후에는 다양한 개체가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

Amazon EKS에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권합니다. 그렇게 하면 적극적으로 모니터링하거나 유지 관리하지 않은 미사용 엔터티가 없습니다. 단, 서비스 연결 역할을 정리해야 수동으로 삭제할 수 있습니다.

서비스 연결 역할을 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에서 사용되는 리소스를 삭제해야 합니다.

> **Note**
>
> 리소스를 삭제하려고 할 때 Amazon EKS 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

`AWSServiceRoleForAmazonEKSConnector` 역할에 의해 사용되는 Amazon EKS 리소스를 삭제하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 좌측 탐색 창에서 클러스터를 선택합니다.
3. 클러스터 페이지에서 클러스터를 선택합니다.
4. 등록 취소 탭을 선택한 다음 확인 탭을 선택합니다.

수동으로 서비스 연결 역할 삭제

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 AWSServiceRoleForAmazonEKSConnector 서비스 연결 역할을 삭제합니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하십시오.

Outpost에서 Amazon EKS 로컬 클러스터에 대한 역할 사용

Amazon Elastic Kubernetes Service는 AWS Identity and Access Management(IAM) 서비스 연결 역할을 사용합니다. 서비스 연결 역할은 Amazon EKS에 직접 연결된 고유한 유형의 IAM 역할입니다. 서비스 연결 역할은 Amazon EKS에서 사전 정의하며 서비스에서 다른 AWS 서비스를 자동으로 호출하기 위해 필요한 모든 권한을 포함합니다.

필요한 권한을 수동으로 추가할 필요가 없으므로 서비스 연결 역할은 Amazon EKS를 더 쉽게 설정할 수 있습니다. Amazon EKS에서 서비스 연결 역할의 권한을 정의하므로 다르게 정의되지 않은 한, Amazon EKS만 해당 역할을 수임할 수 있습니다. 정의된 권한에는 신뢰 정책과 권한 정책이 포함되며 이 권한 정책은 다른 IAM 엔터티에 연결할 수 없습니다.

먼저 관련 리소스를 삭제한 후에만 서비스 연결 역할을 삭제할 수 있습니다. 이렇게 하면 리소스에 대한 액세스 권한을 실수로 삭제할 수 없기 때문에 Amazon EKS 리소스가 보호됩니다.

서비스 연결 역할을 지원하는 기타 서비스에 대한 자세한 내용은 IAM으로 작업하는 AWS 서비스를 참조하고 서비스 연결 역할(Service-Linked Role) 열에 예(Yes)가 있는 서비스를 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예 링크를 선택합니다.

Amazon EKS에 대한 서비스 연결 역할 권한

Amazon EKS는 `AWSServiceRoleForAmazonEKSLocalOutpost`라는 서비스 연결 역할을 사용합니다. 이 역할은 Amazon EKS가 사용자 계정의 로컬 클러스터를 관리하도록 허용합니다. 연결된 정책은 이 역할이 네트워크 인터페이스, 보안 그룹, 로그 및 Amazon EC2 인스턴스와 같은 리소스를 관리하도록 허용합니다.

> **Note**
>
> `AWSServiceRoleForAmazonEKSLocalOutpost` 서비스 연결 역할은 클러스터 생성에 필요한 역할과 다릅니다. 자세한 내용은 Amazon EKS 클러스터 IAM 역할 단원을 참조하십시오.

`AWSServiceRoleForAmazonEKSLocalOutpost` 서비스 연결 역할은 역할을 위임하기 위해 다음 서비스를 신뢰합니다.

- `outposts.eks-local.amazonaws.com`

역할 권한 정책은 Amazon EKS가 지정된 리소스에서 다음 작업을 완료하도록 허용합니다.

- `AmazonEKSServiceRolePolicy`

IAM 엔터티(사용자, 그룹, 역할 등)가 서비스 링크 역할을 생성하고 편집하거나 삭제할 수 있도록 권한을 구성할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 권한을 참조하세요.

Amazon EKS에 대한 서비스 연결 역할 생성

서비스 링크 역할은 수동으로 생성할 필요가 없습니다. AWS Management Console, AWS CLI 또는 AWS API에서 클러스터를 생성하면 Amazon EKS에서 서비스 연결 역할이 자동으로 생성됩니다.

이 서비스 연결 역할을 삭제했다가 다시 생성해야 하는 경우 동일한 프로세스를 사용하여 계정에서 역할을 다시 생성할 수 있습니다. 클러스터를 생성할 때 Amazon EKS에서는 서비스 연결 역할을 다시 생성합니다.

Amazon EKS에 대한 서비스 연결 역할 편집

Amazon EKS에서는 `AWSServiceRoleForAmazonEKSLocalOutpost` 서비스 연결 역할을 편집하도록 허용하지 않습니다. 서비스 연결 역할을 생성한 후에는 다양한 엔터티가 역할을 참조할 수 있기 때문에 역할 이름을 변경할 수 없습니다. 하지만 IAM을 사용하여 역할의 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하세요.

Amazon EKS에 대한 서비스 연결 역할 삭제

서비스 연결 역할이 필요한 기능 또는 서비스가 더 이상 필요 없는 경우에는 해당 역할을 삭제할 것을 권합니다. 그렇게 하면 적극적으로 모니터링하거나 유지 관리하지 않은 미사용 엔터티가 없습니다. 단, 서비스 연결 역할을 정리해야 수동으로 삭제할 수 있습니다.

서비스 연결 역할을 정리

IAM을 사용하여 서비스 연결 역할을 삭제하기 전에 먼저 역할에서 사용되는 리소스를 삭제해야 합니다.

> **Note**
>
> 리소스를 삭제하려고 할 때 Amazon EKS 서비스가 역할을 사용 중이면 삭제에 실패할 수 있습니다. 이 문제가 발생하면 몇 분 기다렸다가 작업을 다시 시도하세요.

`AWSServiceRoleForAmazonEKSLocalOutpost` 역할에 의해 사용되는 Amazon EKS 리소스를 삭제하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.

2. 왼쪽 탐색 창에서 Amazon EKS 클러스터(Clusters)를 선택합니다.
3. 클러스터에 노드 그룹 또는 Fargate 프로필이 있는 경우 클러스터를 삭제하기 전에 이를 삭제해야 합니다. 자세한 내용을 알아보려면 관리형 노드 그룹 삭제 및 Fargate 프로파일 삭제 섹션을 참조하세요.
4. [클러스터(Clusters)] 페이지에서 삭제하려는 클러스터를 선택하고 [삭제(Delete)]를 선택합니다.
5. 삭제 확인 창에 클러스터 이름을 입력한 다음 [삭제(Delete)]를 선택합니다.
6. 계정의 다른 클러스터에 대해 이 절차를 반복합니다. 모든 삭제 작업이 완료될 때까지 기다립니다.

수동으로 서비스 연결 역할 삭제

IAM 콘솔, AWS CLI 또는 AWS API를 사용하여 `AWSServiceRoleForAmazonEKSLocalOutpost` 서비스 연결 역할을 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제를 참조하십시오.

Amazon EKS 서비스 연결 역할을 지원하는 리전

Amazon EKS는 서비스가 제공되는 모든 리전에서 서비스 연결 역할 사용을 지원합니다. 자세한 내용을 알아보려면 Amazon EKS 엔드포인트 및 할당량을 참조하세요.

Amazon EKS 클러스터 IAM 역할

각 클러스터는 Amazon EKS 클러스터 IAM 역할이 필수입니다. Amazon EKS에서 관리하는 Kubernetes 클러스터는 이 역할을 사용하여 노드를 관리하고, 기존 클라우드 공급자는 이 역할을 사용하여 서비스용 Elastic Load Balancing으로 로드 밸런서를 생성합니다.

Amazon EKS 클러스터를 생성하기 전에 다음 IAM 정책 중 하나를 사용하여 IAM 역할을 생성해야 합니다.

- `AmazonEKSClusterPolicy`
- 사용자 지정 IAM 정책. 다음 최소 권한으로는 Kubernetes 클러스터가 노드를 관리할 수 있지만, 기존 클라우드 공급자가 Elastic Load Balancing으로 로드 밸런서를 생성하는 것은 허용되지 않습니다. 사용자 지정 IAM 정책에는 적어도 다음과 같은 권한이 있어야 합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
        "ec2:CreateTags"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:TagKeys": "kubernetes.io/cluster/*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcs",
        "ec2:DescribeDhcpOptions",
        "kms:DescribeKey"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 2023년 10월 3일 이전에는 각 클러스터의 IAM 역할에 AmazonEKSClusterPolicy이 필요했습니다.
> 2020년 4월 16일 이전에는 AmazonEKSServicePolicy도 필요했으며 제안된 이름은 `eksServiceRole`이었습니다. `AWSServiceRoleForAmazonEKS` 서비스 연결 역할을 사용하면 2020년 4월 16일 이후에 생성된 클러스터에 해당 정책이 더 이상 필요하지 않습니다.

기존 클러스터 역할 확인

다음 절차를 사용하여 계정에 이미 Amazon EKS 클러스터 역할이 있는지 확인할 수 있습니다.

IAM 콘솔에서 **`eksClusterRole`**을 확인하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.

3. 역할 목록에서 eksClusterRole(을)를 검색합니다. eksClusterRole을 포함하는 역할이 존재하지 않을 경우 Amazon EKS 클러스터 역할 생성 섹션을 참조하여 역할을 생성합니다. eksClusterRole을 포함하는 역할이 존재하지 않을 경우, 연결된 정책을 볼 역할을 선택합니다.
4. 권한을 선택합니다.
5. [AmazonEKSClusterPolicy] 관리형 정책이 역할에 연결되었는지 확인합니다. 정책이 연결된 경우 Amazon EKS 클러스터 역할이 적절히 구성된 것입니다.
6. 신뢰 관계(Trust relationships)를 선택한 후 신뢰 정책 편집(Edit trust policy)을 선택합니다.
7. 신뢰 관계에 다음 정책이 포함되어 있는지 확인합니다. 신뢰 관계가 다음 정책과 일치하는 경우, 취소(Cancel)를 선택합니다. 신뢰 관계가 일치하지 않는 경우 정책을 정책 문서(Policy Document) 창에 복사하고 신뢰 정책 업데이트(Update Trust Policy)를 선택합니다.

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

Amazon EKS 클러스터 역할 생성

클러스터 역할을 생성하려면 AWS Management Console 또는 AWS CLI를 사용할 수 있습니다.

AWS Management Console

IAM 콘솔에서 Amazon EKS 클러스터 역할을 생성하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 역할을 선택한 다음 역할 생성을 선택합니다.
3. 신뢰할 수 있는 엔터티 유형(Trusted entity type)에서 AWS 서비스(service)를 선택합니다.
4. 기타 AWS 서비스에 대한 사용 사례(Use cases for other) 드롭다운 목록에서 EKS를 선택합니다.

5. 사용 사례에 대한 EKS - 클러스터(EKS - Cluster)를 선택한 후 다음(Next)을 선택합니다.
6. 권한 추가(Add permissions) 탭에서 다음(Next)을 선택합니다.
7. 역할 이름(Role name)에 역할의 고유한 이름(예: **eksClusterRole**)을 입력합니다.
8. 설명(Description)에서 **Amazon EKS - Cluster role**과 같은 설명 텍스트를 입력합니다.
9. 역할 생성을 선택합니다.

AWS CLI

1. 다음 콘텐츠를 *cluster-trust-policy.json*라는 파일에 복사합니다.

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

2. 역할을 생성합니다. ***eksClusterRole***을 선택한 모든 이름으로 바꿀 수 있습니다.

```
aws iam create-role \
  --role-name eksClusterRole \
  --assume-role-policy-document file://"cluster-trust-policy.json"
```

3. 필요한 IAM 정책을 역할에 연결합니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name eksClusterRole
```

Amazon EKS 노드 IAM 역할

Amazon EKS 노드 `kubelet` 데몬은 사용자를 대신하여 AWS API를 호출합니다. 노드는 IAM 인스턴스 프로파일 및 연결 정책을 통해 이 API 호출에 대한 권한을 수신합니다. 노드를 시작해 클러스터에

등록하려면 시작할 때 노드에서 사용할 IAM 역할을 생성해야 합니다. 이 요구 사항은 Amazon이 제공하는 Amazon EKS 최적화 AMI 또는 사용하려는 다른 노드 AMI를 사용하여 시작된 노드에 적용됩니다. 또한 이 요구 사항은 관리형 노드 그룹과 자체 관리형 노드 모두에 적용됩니다.

> **Note**
>
> 클러스터를 생성하는 데 사용된 것과 동일한 역할을 사용할 수 없습니다.

노드를 생성하려면 먼저 다음 권한을 사용하여 IAM 역할을 생성해야 합니다.

- `AmazonEKSWorkerNodePolicy` 정책에서 제공하는 것과 같이 VPC의 Amazon EC2 리소스를 설명할 수 있는 `kubelet` 권한. 이 정책은 Amazon EKS Pod Identity Agent에 대한 권한도 제공합니다.
- Amazon Elastic Container Registry(Amazon ECR)의 컨테이너 이미지를 사용할 수 있는 `kubelet` 권한(`AmazonEC2ContainerRegistryReadOnly` 정책에서 제공하는 권한). 네트워킹용 내장 애드온이 Amazon ECR의 컨테이너 이미지를 사용하는 포드를 실행하기 때문에 Amazon Elastic Container Registry(Amazon ECR)의 컨테이너 이미지를 사용할 수 있는 권한이 필요합니다.
- (선택 사항) Amazon EKS Pod Identity Agent에서 `eks-auth:AssumeRoleForPodIdentity` 작업을 사용하여 포드에 대한 보안 인증 정보를 검색할 수 있는 권한. AmazonEKSWorkerNodePolicy를 사용하지 않는 경우 EKS Pod Identity를 사용할 수 있는 EC2 권한 외에도 이 권한을 제공해야 합니다.
- (선택 사항) IRSA 또는 EKS Pod Identity를 사용하여 VPC CNI 포드에 권한을 부여하지 않는 경우 인스턴스 역할에서 VPC CNI에 대한 권한을 제공해야 합니다. `AmazonEKS_CNI_Policy` 관리형 정책(IPv4 제품군으로 클러스터를 생성하는 경우) 또는 사용자가 생성한 IPv6 정책(IPv6 제품군으로 클러스터를 생성하는 경우)을 사용할 수 있습니다. 그러나 이 역할에 정책을 연결하는 대신 Amazon VPC CNI 추가 기능에 특별히 사용되는 별도의 역할에 정책을 연결하는 것이 좋습니다. Amazon VPC CNI 추가 기능에 대한 별도의 역할 생성에 대한 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 섹션을 참조하세요.

> **Note**
>
> 2023년 10월 3일 이전에는 각 관리형 노드 그룹의 IAM 역할에 `AmazonEKSWorkerNodePolicy`와(과) `AmazonEC2ContainerRegistryReadOnly`이(가) 필요했습니다.
> Amazon EC2 노드 그룹에는 Fargate 프로파일과 다른 IAM 역할이 있어야 합니다. 자세한 내용은 Amazon EKS Pod 실행 IAM 역할 단원을 참조하십시오.

기존 노드 역할 확인

다음 절차를 사용하여 계정에 이미 Amazon EKS 노드 역할이 있는지 확인할 수 있습니다.

IAM 콘솔에서 **eksNodeRole**을 확인하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 역할 목록에서 eksNodeRole, AmazonEKSNodeRole 또는 NodeInstanceRole을 검색합니다. 이러한 이름 중 하나를 가진 역할이 없으면 Amazon EKS 노드 IAM 역할 생성 섹션을 참조하여 역할을 만듭니다. eksNodeRole, AmazonEKSNodeRole 또는 NodeInstanceRole을 포함하는 역할이 존재하는 경우 역할을 선택하여 연결된 정책을 확인합니다.
4. 권한을 선택합니다.
5. [AmazonEKSWorkerNodePolicy] 및 [AmazonEC2ContainerRegistryReadOnly] 관리형 정책이 역할에 연결되어 있는지 또는 사용자 지정 정책이 최소 권한으로 연결되어 있는지 확인합니다.

> **Note**
>
> AmazonEKS_CNI_Policy 정책이 역할에 연결되어 있는 경우 해당 역할을 제거하고 대신 aws-node Kubernetes 서비스 계정에 매핑된 IAM 역할에 연결하는 것이 좋습니다. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오.

6. 신뢰 관계(Trust relationships)를 선택한 후 신뢰 정책 편집(Edit trust policy)을 선택합니다.
7. 신뢰 관계에 다음 정책이 포함되어 있는지 확인합니다. 신뢰 관계가 다음 정책과 일치하는 경우, 취소(Cancel)를 선택합니다. 신뢰 관계가 일치하지 않는 경우 정책을 정책 문서(Policy Document) 창에 복사하고 신뢰 정책 업데이트(Update Trust Policy)를 선택합니다.

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

```
}
```

Amazon EKS 노드 IAM 역할 생성

AWS Management Console 또는 AWS CLI를 사용하여 노드 IAM 역할을 생성할 수 있습니다.

AWS Management Console

IAM 콘솔에서 Amazon EKS 노드 역할을 생성하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 역할(Roles) 페이지에서 역할 생성(Create role)을 선택합니다.
4. 신뢰할 수 있는 엔터티 선택(Select trusted entity) 페이지에서 다음을 수행합니다.
   a. 신뢰할 수 있는 엔터티 유형(Trusted entity type) 섹션에서 AWS 서비스를 선택합니다.
   b. 사용 사례(Use case)에서 EC2를 선택합니다.
   c. 다음을 선택합니다.
5. 권한 추가 페이지에서 사용자 지정 정책을 연결하거나 다음과 같이 합니다.
   a. 필터 정책(Filter policies) 상자에 **AmazonEKSWorkerNodePolicy**를 입력합니다.
   b. 검색 결과의 AmazonEKSWorkerNodePolicy 왼쪽에 있는 확인란을 선택합니다.
   c. 필터 지우기(Clear filters)를 선택합니다.
   d. 필터 정책(Filter policies) 상자에 **AmazonEC2ContainerRegistryReadOnly**를 입력합니다.
   e. 검색 결과의 AmazonEC2ContainerRegistryReadOnly 왼쪽에 있는 확인란을 선택합니다.

      생성하는 AmazonEKS_CNI_Policy 관리형 정책이나 IPv6 정책은 또한 이 역할 또는 aws-node Kubernetes 서비스 계정에 매핑된 다른 역할에 연결되어야 합니다. 이 역할에 정책을 할당하는 대신 Kubernetes 서비스 계정에 연결된 역할에 정책을 할당하는 것이 좋습니다. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오.
   f. Next(다음)를 선택합니다.
6. 이름, 검토 및 생성(Name, review, and create) 페이지에서 다음을 수행합니다.
   a. 역할 이름(Role name)에 역할의 고유한 이름(예: **AmazonEKSNodeRole**)을 입력합니다.

b. 설명(Description)에서 현재 텍스트를 설명이 포함된 텍스트(예: **Amazon EKS - Node role**)로 바꿉니다.

c. 태그 추가(선택 사항)에서 태그를 키-값 페어로 연결하여 메타데이터를 역할에 추가합니다. IAM에서 태그 사용에 대한 자세한 내용을 알아보려면 IAM 사용 설명서의 IAM 리소스에 태그 지정을 참조하세요.

d. 역할 생성을 선택합니다.

AWS CLI

1. 다음 명령을 실행해 `node-role-trust-relationship.json` 파일을 생성합니다.

```
cat >node-role-trust-relationship.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

2. IAM 역할을 생성합니다.

```
aws iam create-role \
  --role-name AmazonEKSNodeRole \
  --assume-role-policy-document file://"node-role-trust-relationship.json"
```

3. 필요한 2개의 관리형 IAM 정책을 IAM 역할에 연결합니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name AmazonEKSNodeRole
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name AmazonEKSNodeRole
```

4. 클러스터를 생성한 IP 패밀리에 따라 다음 IAM 정책 중 하나를 IAM 역할에 연결합니다. 이 정책은 이 역할에 연결되거나 Amazon VPC CNI plugin for Kubernetes에 사용되는 Kubernetes `aws-node` 서비스 계정에 연결된 역할에 연결되어야 합니다. Kubernetes 서비스 계정에 연결된 역할에 정책을 할당하는 것이 좋습니다. Kubernetes 서비스 계정에 연결된 역할에 정책을 할당하려면 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 섹션을 참조하세요.

   - IPv4

     ```
     aws iam attach-role-policy \
       --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
       --role-name AmazonEKSNodeRole
     ```

   - IPv6

     1. 다음 텍스트를 복사해 *vpc-cni-ipv6-policy*.json 파일에 저장합니다.

        ```
        {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "ec2:AssignIpv6Addresses",
                        "ec2:DescribeInstances",
                        "ec2:DescribeTags",
                        "ec2:DescribeNetworkInterfaces",
                        "ec2:DescribeInstanceTypes"
                    ],
                    "Resource": "*"
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "ec2:CreateTags"
                    ],
                    "Resource": [
                        "arn:aws:ec2:*:*:network-interface/*"
                    ]
                }
            ]
        }
        ```

2. IAM 정책을 생성합니다.

```
aws iam create-policy --policy-name AmazonEKS_CNI_IPv6_Policy --policy-
document file://vpc-cni-ipv6-policy.json
```

3. IAM 정책을 IAM 역할에 연결합니다. *111122223333*을 계정 ID로 바꿉니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::111122223333:policy/AmazonEKS_CNI_IPv6_Policy \
  --role-name AmazonEKSNodeRole
```

Amazon EKS Pod 실행 IAM 역할

Pod 인프라에서 Pods를 실행하려면 Amazon EKS AWS Fargate 실행 역할이 필요합니다.

클러스터가 AWS Fargate 인프라에서 Pods를 생성하는 경우 Fargate 인프라에서 실행되는 구성 요소는 사용자를 대신하여 AWS API를 호출해야 합니다. 이를 통해 Amazon ECR에서 컨테이너 이미지를 가져오거나 로그를 다른 AWS 서버로 라우팅하는 등의 작업을 수행할 수 있습니다. Amazon EKS Pod 실행 역할은 이 작업을 수행할 수 있는 IAM 권한을 제공합니다.

Fargate 프로필을 생성할 때 프로필을 사용하여 Fargate 인프라에서 실행되는 Amazon EKS 구성 요소의 Pod 실행 역할을 지정해야 합니다. 이 역할은 권한 부여를 위해 클러스터의 Kubernetes 역할 기반 액세스 제어(RBAC)에 추가됩니다. 이렇게 하면 Fargate 인프라에서 실행 중인 `kubelet`이 Amazon EKS 클러스터에 등록되어 클러스터에 노드로 표시될 수 있습니다.

> **Note**
>
> Fargate 프로파일에는 Amazon EC2 노드 그룹과 다른 IAM 역할이 있어야 합니다.

> **Important**
>
> Fargate Pod에서 실행 중인 컨테이너는 Pod 실행 역할과 연결된 IAM 권한을 맡을 수 없습니다. Fargate Pod의 컨테이너에 다른 AWS 서비스에 대한 액세스 권한을 부여하려면 서비스 계정에 대한 IAM 역할을 사용해야 합니다.

Fargate 프로필을 생성하기 전에 AmazonEKSFargatePodExecutionRolePolicy를 사용하여 IAM 역할을 생성해야 합니다.

올바르게 구성된 기존 Pod 실행 역할 확인

다음 절차를 사용하여 계정에 이미 올바로 구성된 Amazon EKS Pod 실행 역할이 있는지 확인할 수 있습니다. 대리인 보안 문제의 혼동을 방지하려면 역할이 `SourceArn`에 따라 액세스를 제한하는 것이 중요합니다. 필요에 따라 실행 역할을 수정하여 다른 클러스터의 Fargate 프로필에 대한 지원을 포함할 수 있습니다.

IAM 콘솔에서 Amazon EKS Pod 실행 역할 확인

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할(Roles)을 선택합니다.
3. 역할(Roles) 페이지에서 AmazonEKSFargatePodExecutionRole에 대한 역할 목록을 검색합니다. 역할이 존재하지 않을 경우 Amazon EKS Pod 실행 역할 생성을 참조하여 역할을 생성합니다. 역할이 존재하지 않을 경우 역할을 선택합니다.
4. AmazonEKSFargatePodExecutionRole 페이지에서 다음을 수행합니다.
   a. 권한(Permissions)을 선택합니다.
   b. AmazonEKSFargatePodExecutionRolePolicy Amazon 관리형 정책이 역할에 연결되어 있는지 확인합니다.
   c. 신뢰 관계(Trust relationships)를 선택합니다.
   d. 신뢰 정책 편집(Edit trust policy)을 선택합니다.
5. 신뢰 정책 편집(Edit trust policy) 페이지에서 신뢰 관계에 다음 정책이 포함되어 있고 클러스터의 Fargate 프로필에 대한 줄이 있는지 확인합니다. 그렇다면 취소(Cancel)를 선택합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
```

```
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

정책이 일치하지만 클러스터의 Fargate 프로필을 지정하는 줄이 없는 경우 다음 줄을 `ArnLike` 객체의 상단에 추가할 수 있습니다. *region-code*를 클러스터가 있는 AWS 리전으로, *111122223333*을 계정 ID로, *my-cluster*를 클러스터 이름으로 바꿉니다.

```
"aws:SourceArn": "arn:aws:eks:region-code:111122223333:fargateprofile/my-cluster/
*",
```

정책이 일치하지 않는 경우 이전 정책 전체를 양식에 복사하고 업데이트 정책(Update policy)을 선택합니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다. 계정의 모든 AWS 리전에서 동일한 역할을 사용하려는 경우 *region-code*을 *로 바꾸세요. *111122223333*를 계정 ID로, *my-cluster*를 클러스터의 이름으로 바꿉니다. 계정의 모든 클러스터에 대해 동일한 역할을 사용하려는 경우 *를 *my-cluster*로 바꾸세요.

Amazon EKS Pod 실행 역할 생성

클러스터에 대한 Amazon EKS Pod 실행 역할이 아직 없는 경우 AWS Management Console 또는 AWS CLI를 사용하여 생성할 수 있습니다.

AWS Management Console

AWS Management Console을 사용하여 AWS FargatePod 실행 역할 생성

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 역할(Roles) 페이지에서 역할 생성(Create role)을 선택합니다.
4. 신뢰할 수 있는 엔터티 선택(Select trusted entity) 페이지에서 다음을 수행합니다.
   a. 신뢰할 수 있는 엔터티 유형(Trusted entity type) 섹션에서 AWS 서비스(service)를 선택합니다.
   b. 기타 AWS 서비스에 대한 사용 사례(Use cases for other) 드롭다운에서 EKS를 선택합니다.

c. EKS - Fargate Pod를 선택합니다.

d. 다음을 선택합니다.

5. 권한 추가(Add permissions) 페이지에서 다음(Next)을 선택합니다.

6. 이름, 검토 및 생성(Name, review, and create) 페이지에서 다음을 수행합니다.

a. 역할 이름(Role name)에 역할의 고유한 이름(예: **AmazonEKSFargatePodExecutionRole**)을 입력합니다.

b. 태그 추가(선택 사항)에서 태그를 키-값 페어로 연결하여 메타데이터를 역할에 추가합니다. IAM에서 태그 사용에 대한 자세한 내용을 알아보려면 IAM 사용 설명서의 IAM 리소스에 태그 지정을 참조하세요.

c. 역할 생성을 선택합니다.

7. 역할(Roles) 페이지에서 AmazonEKSFargatePodExecutionRole에 대한 역할 목록을 검색합니다. 역할을 선택합니다.

8. AmazonEKSFargatePodExecutionRole 페이지에서 다음을 수행합니다.

a. 신뢰 관계(Trust relationships)를 선택합니다.

b. 신뢰 정책 편집(Edit trust policy)을 선택합니다.

9. 신뢰 정책 편집(Edit trust policy) 페이지에서 다음 작업을 수행합니다.

a. 다음 내용을 복사하여 신뢰 정책 편집(Edit trust policy) 양식에 붙여 넣습니다. *region-code*을 해당 클러스터가 있는 AWS 리전으로 바꿉니다. 계정의 모든 AWS 리전에서 동일한 역할을 사용하려는 경우 *region-code*을 *로 바꾸세요. *111122223333*를 계정 ID로, *my-cluster*를 클러스터의 이름으로 바꿉니다. 계정의 모든 클러스터에 대해 동일한 역할을 사용하려는 경우 *my-cluster*를 *로 바꾸세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

b. 정책 업데이트를 선택합니다.

AWS CLI

AWS CLI을 사용하여 AWS FargatePod 실행 역할 생성

1. 다음 내용을 복사하여 *pod-execution-role-trust-policy*.json라는 파일에 붙여넣습니다. *region-code*을 해당 클러스터가 있는 AWS 리전으로 바꿉니다. 계정의 모든 AWS 리전에서 동일한 역할을 사용하려는 경우 *region-code*을 *로 바꾸세요. *111122223333*를 계정 ID로, *my-cluster*를 클러스터의 이름으로 바꿉니다. 계정의 모든 클러스터에 대해 동일한 역할을 사용하려는 경우 *를 *my-cluster*로 바꾸세요.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Condition": {
         "ArnLike": {
            "aws:SourceArn": "arn:aws:eks:region-
code:111122223333:fargateprofile/my-cluster/*"
         }
      },
      "Principal": {
        "Service": "eks-fargate-pods.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Pod 실행 IAM 역할을 생성합니다.

```
aws iam create-role \
  --role-name AmazonEKSFargatePodExecutionRole \
```

```
  --assume-role-policy-document file://"pod-execution-role-trust-policy.json"
```

3. 필요한 Amazon EKS 관리형 IAM 정책을 역할에 연결합니다.

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSFargatePodExecutionRolePolicy \
  --role-name AmazonEKSFargatePodExecutionRole
```

Amazon EKS Connector IAM 역할

Kubernetes 클러스터를 연결하여 AWS Management Console에서 볼 수 있습니다. Kubernetes 클러스터에 연결하려면 IAM 역할을 생성합니다.

기존 EKS 커넥터 역할 확인

다음 절차를 사용하여 계정에 이미 Amazon EKS Connector 역할이 있는지 확인할 수 있습니다.

IAM 콘솔에서 **AmazonEKSConnectorAgentRole**(을)를 확인하는 방법

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 역할을 선택합니다.
3. 역할 목록에서 AmazonEKSConnectorAgentRole(을)를 검색합니다. AmazonEKSConnectorAgentRole을 포함하는 역할이 존재하지 않을 경우 Amazon EKS Connector 에이전트 역할 생성 섹션을 참조하여 역할을 생성합니다. AmazonEKSConnectorAgentRole을 포함하는 역할이 존재하지 않을 경우, 연결된 정책을 볼 역할을 선택합니다.
4. 권한을 선택합니다.
5. AmazonEKSConnectorAgentPolicy 관리형 정책이 역할에 연결되었는지 확인합니다. 정책이 연결된 경우 Amazon EKS 커넥터 역할이 적절히 구성된 것입니다.
6. 신뢰 관계(Trust relationships)를 선택한 후 신뢰 정책 편집(Edit trust policy)을 선택합니다.
7. 신뢰 관계에 다음 정책이 포함되어 있는지 확인합니다. 신뢰 관계가 다음 정책과 일치하는 경우, 취소(Cancel)를 선택합니다. 신뢰 관계가 일치하지 않는 경우 정책을 정책 문서(Policy Document) 창에 복사하고 신뢰 정책 업데이트(Update Trust Policy)를 선택합니다.

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

```
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Amazon EKS Connector 에이전트 역할 생성

AWS Management Console 또는 AWS CloudFormation을 사용하여 커넥터 에이전트 역할을 생성할 수 있습니다.

AWS CLI

1. IAM 역할에 사용할 다음과 같은 JSON이 포함된 `eks-connector-agent-trust-policy.json`이라는 이름의 파일을 생성합니다.

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

2. IAM 역할에 사용할 다음과 같은 JSON이 포함된 `eks-connector-agent-policy.json`이라는 이름의 파일을 생성합니다.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SsmControlChannel",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateControlChannel"
            ],
            "Resource": "arn:aws:eks:*:*:cluster/*"
        },
        {
            "Sid": "ssmDataplaneOperations",
            "Effect": "Allow",
            "Action": [
                "ssmmessages:CreateDataChannel",
                "ssmmessages:OpenDataChannel",
                "ssmmessages:OpenControlChannel"
            ],
            "Resource": "*"
        }
    ]
}
```

3. 이전 목록 항목에서 생성한 신뢰 정책 및 정책을 사용하여 Amazon EKS Connector 에이전트 역할을 생성합니다.

```
aws iam create-role \
     --role-name AmazonEKSConnectorAgentRole \
     --assume-role-policy-document file://eks-connector-agent-trust-policy.json
```

4. Amazon EKS Connector 에이전트 역할에 정책을 연결합니다.

```
aws iam put-role-policy \
     --role-name AmazonEKSConnectorAgentRole \
     --policy-name AmazonEKSConnectorAgentPolicy \
     --policy-document file://eks-connector-agent-policy.json
```

AWS CloudFormation

AWS CloudFormation을 사용하여 Amazon EKS Connector 에이전트 역할을 생성하려면

1. 다음 AWS CloudFormation 템플릿을 로컬 시스템의 텍스트 파일에 저장합니다.

> **Note**
>
> 이 템플릿은 또한 `registerCluster` API가 호출될 때 생성되는 서비스 링크 역할도 생성합니다. 세부 정보는 역할을 사용하여 Amazon EKS에 Kubernetes 클러스터 연결 섹션을 참조하세요.

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Provisions necessary resources needed to register clusters in EKS'
Parameters: {}
Resources:
  EKSConnectorSLR:
    Type: AWS::IAM::ServiceLinkedRole
    Properties:
      AWSServiceName: eks-connector.amazonaws.com

  EKSConnectorAgentRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Action: [ 'sts:AssumeRole' ]
            Principal:
              Service: 'ssm.amazonaws.com'

  EKSConnectorAgentPolicy:
    Type: AWS::IAM::Policy
    Properties:
      PolicyName: EKSConnectorAgentPolicy
      Roles:
        - {Ref: 'EKSConnectorAgentRole'}
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateControlChannel' ]
            Resource:
            - Fn::Sub: 'arn:${AWS::Partition}:eks:*:*:cluster/*'
```

```
          - Effect: 'Allow'
            Action: [ 'ssmmessages:CreateDataChannel',
 'ssmmessages:OpenDataChannel', 'ssmmessages:OpenControlChannel' ]
            Resource: "*"
Outputs:
  EKSConnectorAgentRoleArn:
    Description: The agent role that EKS connector uses to communicate with AWS ##
#.
    Value: !GetAtt EKSConnectorAgentRole.Arn
```

2. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
3. 스택 생성(Create stack)(새 리소스 또는 기존 리소스 사용)을 선택합니다.
4. 템플릿 지정(Specify template)에서 템플릿 파일 업로드(Upload a template file)를 선택한 다음 파일 선택(Choose file)을 선택합니다.
5. 이전에 생성한 파일을 선택한 후 다음(Next)을 선택합니다.
6. 스택 이름(Stack name)에 `eksConnectorAgentRole`과 같은 역할 이름을 입력하고 다음(Next)을 선택합니다.
7. 스택 옵션 구성 페이지에서 다음(Next)을 선택합니다.
8. 검토 페이지에서 정보를 검토하고, 스택이 IAM 리소스를 생성할 수 있음을 확인한 다음 스택 생성(Create stack)을 선택합니다.

Amazon Elastic Kubernetes Service에 대한 AWS 관리형 정책

AWS 관리형 정책은 AWS에서 생성되고 관리되는 독립 실행형 정책입니다. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있기 때문에 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에서 정의한 권한은 변경할 수 없습니다. 만약 AWS가 AWS 관리형 정책에 정의된 권한을 업데이트할 경우 정책이 연결되어 있는 모든 보안 주체 엔터티(사용자, 그룹 및 역할)에도 업데이트가 적용됩니다. 새로운 AWS 서비스를 시작하거나 새로운 API 작업을 기존 서비스에 이용하는 경우 AWS가 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: AmazonEKS_CNI_Policy

`AmazonEKS_CNI_Policy`를 IAM 엔터티에 연결할 수 있습니다. Amazon EC2 노드 그룹을 생성하기 전에 이 정책을 노드 IAM 역할에 연결하거나 Amazon VPC CNI plugin for Kubernetes에서 명시적으로 사용하는 IAM 역할에 연결해야 합니다. 이는 사용자를 대신하여 작업을 수행할 수 있도록 합니다. 플러그 인에서만 사용되는 역할에 정책을 연결하는 것이 좋습니다. 자세한 내용은 Amazon VPC CNI plugin for Kubernetes Amazon EKS 추가 기능을 사용한 작업 및 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 섹션을 참조하세요.

권한 세부 정보

이 정책에는 Amazon EKS가 다음 태스크를 완료할 수 있도록 허용하는 다음과 같은 권한이 포함되어 있습니다.

- **`ec2:*NetworkInterface`** 및 **`ec2:*PrivateIpAddresses`** - Amazon VPC CNI 플러그인이 Amazon EKS에서 실행되는 애플리케이션에 네트워킹을 제공하기 위해 Pods의 탄력적 네트워크 인터페이스 및 IP 주소 프로비저닝과 같은 작업을 수행하도록 허용합니다.
- **`ec2`** 읽기 작업 - Amazon VPC CNI 플러그인이 Amazon VPC 서브넷의 사용 가능한 IP 주소 양을 확인하기 위해 인스턴스 및 서브넷 설명과 같은 작업을 수행하도록 허용합니다. VPC CNI는 각 서브넷의 사용 가능한 IP 주소를 사용하여 탄력적 네트워크 인터페이스를 생성할 때 사용 가능한 IP 주소가 가장 많은 서브넷을 선택할 수 있습니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AmazonEKS_CNI_Policy를 참조하십시오.

AWS 관리형 정책: AmazonEKSClusterPolicy

`AmazonEKSClusterPolicy`를 IAM 엔터티에 연결할 수 있습니다. 클러스터를 생성하기 전에 이 정책이 연결된 클러스터 IAM 역할이 있어야 합니다. Amazon EKS에서 관리되는 Kubernetes 클러스터는 사용자 대신 다른 AWS 서비스를 호출합니다. 이러한 작업을 수행하여 서비스에 사용하는 리소스를 관리합니다.

이 정책에는 Amazon EKS가 다음 태스크를 완료할 수 있도록 허용하는 다음과 같은 권한이 포함되어 있습니다.

- **`autoscaling`** - Auto Scaling 그룹의 구성을 읽고 업데이트합니다. 이러한 권한은 Amazon EKS에서 사용되지 않지만 이전 버전과의 호환성을 위해 정책에 그대로 유지됩니다.

- **ec2** - Amazon EC2 노드에 연결된 볼륨 및 네트워크 리소스로 작업합니다. 이는 Kubernetes 컨트롤 플레인이 인스턴스를 클러스터에 조인하고 Kubernetes 영구 볼륨에서 요청한 Amazon EBS 볼륨을 동적으로 프로비저닝 및 관리할 수 있도록 하기 위해 필요합니다.
- **elasticloadbalancing** - Elastic 로드 밸런서를 사용하여 노드를 대상으로 추가합니다. 이는 Kubernetes 컨트롤 플레인이 Kubernetes 서비스에서 요청한 Elastic Load Balancer를 동적으로 프로비저닝할 수 있도록 하기 위해 필요합니다.
- **iam** - 서비스 연결 역할 생성 이는 Kubernetes 컨트롤 플레인이 Kubernetes 서비스에서 요청한 Elastic Load Balancer를 동적으로 프로비저닝할 수 있도록 하기 위해 필요합니다.
- **kms** - AWS KMS에서 키를 읽습니다. 이는 Kubernetes 컨트롤 플레인이 `etcd`에 저장된 Kubernetes 비밀의 비밀 암호화를 지원하는 데 필요합니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AmazonEKSClusterPolicy를 참조하십시오.

AWS 관리형 정책: AmazonEKSFargatePodExecutionRolePolicy

`AmazonEKSFargatePodExecutionRolePolicy`를 IAM 엔터티에 연결할 수 있습니다. Fargate 프로필을 생성하려면 먼저 Fargate Pod 실행 역할을 생성하고 이 정책을 해당 역할에 연결해야 합니다. 자세한 내용을 알아보려면 Fargate Pod 실행 역할 생성 및 AWS Fargate 프로파일 섹션을 참조하세요.

이 정책은 Fargate에서 Amazon EKS Pods를 실행하는 데 필요한 다른 AWS 서비스 리소스에 대한 액세스를 제공하는 권한을 역할에 부여합니다.

권한 세부 정보

이 정책에는 Amazon EKS가 다음 태스크를 완료할 수 있도록 허용하는 다음과 같은 권한이 포함되어 있습니다.

- **ecr** - Fargate에서 실행 중인 포드가 Amazon ECR에 저장된 컨테이너 이미지를 가져오도록 허용합니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AmazonEKSFargatePodExecutionRolePolicy를 참조하십시오.

AWS 관리형 정책: AmazonEKSForFargateServiceRolePolicy

`AmazonEKSForFargateServiceRolePolicy`를 IAM 엔터티에 연결할 수 없습니다. 이 정책은 Amazon EKS에 사용자를 대신하여 작업을 수행할 수 있도록 서비스 연결 역할에 연결됩니다. 자세한 내용은 AWSServiceRoleforAmazonEKSForFargate를 참조하세요.

이 정책은 Amazon EKS에 Fargate 태스크를 실행하는 데 필요한 권한을 부여합니다. 이 정책은 Fargate 노드가 있는 경우에만 사용됩니다.

권한 세부 정보

이 정책에는 Amazon EKS가 다음 태스크를 완료할 수 있도록 허용하는 다음과 같은 권한이 포함되어 있습니다.

- **ec2** – Elastic 네트워크 인터페이스를 생성 및 삭제하고 Elastic 네트워크 인터페이스 및 리소스를 설명합니다. 이는 Amazon EKS Fargate 서비스가 Fargate 포드에 필요한 VPC 네트워킹을 구성할 수 있도록 하기 위해 필요합니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AmazonEKSForFargateServiceRolePolicy를 참조하십시오.

AWS 관리형 정책: AmazonEKSServicePolicy

`AmazonEKSServicePolicy`를 IAM 엔터티에 연결할 수 있습니다. 2020년 4월 16일 이전에 생성된 클러스터에서는 IAM 역할을 만들어 이 정책을 역할에 연결해야 했습니다. 2020년 4월 16일 이후에 생성된 클러스터에서는 역할을 생성할 필요가 없으며 이 정책을 할당할 필요가 없습니다. `iam:CreateServiceLinkedRole` 권한이 있는 IAM 보안 주체를 사용하여 클러스터를 생성하는 경우 AWS ServiceRoleforAmazonEKS 서비스 연결 역할이 자동으로 생성됩니다. 이 서비스 연결 역할에는 AWS 관리형 정책: AmazonEKSServiceRolePolicy가 연결되어 있습니다.

이 정책을 통해 Amazon EKS는 Amazon EKS 클러스터를 운영하는 데 필요한 리소스를 생성하고 관리할 수 있습니다.

권한 세부 정보

이 정책에는 Amazon EKS가 다음 태스크를 완료할 수 있도록 허용하는 다음과 같은 권한이 포함되어 있습니다.

- **eks** - 사용자가 업데이트를 시작한 후 클러스터의 Kubernetes 버전을 업데이트합니다. 이 권한은 Amazon EKS에 사용되지 않지만 이전 버전과의 호환성을 위해 정책에 그대로 유지됩니다.

- **ec2** - Elastic 네트워크 인터페이스 및 기타 네트워크 리소스와 태그를 사용합니다. 이는 Amazon EKS에서 노드와 Kubernetes 컨트롤 플레인 간의 통신을 용이하게 하는 네트워킹을 구성하는 데 필요합니다.
- **route53** - VPC를 호스팅 영역에 연결합니다. 이는 Amazon EKS에서 Kubernetes 클러스터 API 서버에 프라이빗 엔드포인트 네트워킹을 활성화하는 데 필요합니다.
- **logs** - 로그 이벤트 이는 Amazon EKS가 Kubernetes 컨트롤 플레인 로그를 CloudWatch로 전송하기 위해 필요합니다.
- **iam** - 서비스 연결 역할 생성 이는 Amazon EKS가 사용자를 대신하여 AWSServiceRoleForAmazonEKS 서비스 연결 역할을 생성하는 데 필요합니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AmazonEKSServicePolicy를 참조하십시오.

AWS 관리형 정책: AmazonEKSServiceRolePolicy

AmazonEKSServiceRolePolicy를 IAM 엔터티에 연결할 수 없습니다. 이 정책은 Amazon EKS에 사용자를 대신하여 작업을 수행할 수 있도록 서비스 연결 역할에 연결됩니다. 자세한 내용은 Amazon EKS에 대한 서비스 연결 역할 권한 단원을 참조하십시오. iam:CreateServiceLinkedRole 권한이 있는 IAM 보안 주체를 사용하여 클러스터를 생성하는 경우 AWS ServiceRoleforAmazonEKS 서비스 연결 역할이 자동으로 생성되며 이 정책이 연결됩니다.

이 정책을 사용하면 서비스 연결 역할이 사용자를 대신하여 AWS 서비스를 호출할 수 있습니다.

권한 세부 정보

이 정책에는 Amazon EKS가 다음 태스크를 완료할 수 있도록 허용하는 다음과 같은 권한이 포함되어 있습니다.

- **ec2** - 탄력적 네트워크 인터페이스와 Amazon EC2 인스턴스, 클러스터 보안 그룹 및 클러스터를 생성하는 데 필요한 VPC를 생성하고 설명합니다.
- **iam** - IAM 역할에 연결된 모든 관리형 정책을 나열합니다. 이는 Amazon EKS가 클러스터 생성에 필요한 모든 관리형 정책 및 권한을 나열하고 검증하는 데 필요합니다.
- 호스팅 영역과 VPC 연결 - 이는 Amazon EKS에서 Kubernetes 클러스터 API 서버에 프라이빗 엔드포인트 네트워킹을 사용 설정하는 데 필요합니다.
- 로그 이벤트 - 이는 Amazon EKS가 Kubernetes 컨트롤 플레인 로그를 CloudWatch로 전송하기 위해 필요합니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AmazonEKSServiceRolePolicy를 참조하십시오.

AWS 관리형 정책: AmazonEKSVPCResourceController

`AmazonEKSVPCResourceController` 정책을 IAM 보안 인증에 연결할 수 있습니다. Pods의 보안 그룹을 사용하는 경우 사용자를 대신하여 작업을 수행하도록 이 정책을 Amazon EKS 클러스터 IAM 역할에 연결해야 합니다.

이 정책은 노드의 탄력적 네트워크 인터페이스 및 IP 주소를 관리하기 위한 클러스터 역할에 권한을 부여합니다.

권한 세부 정보

이 정책에는 Amazon EKS가 다음 태스크를 완료할 수 있도록 허용하는 다음과 같은 권한이 포함되어 있습니다.

- **ec2** - Elastic 네트워크 인터페이스 및 IP 주소를 관리하여 Pod 보안 그룹 및 Windows 노드를 지원합니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AmazonEKSVPCResourceController를 참조하십시오.

AWS 관리형 정책: AmazonEKSWorkerNodePolicy

`AmazonEKSWorkerNodePolicy`를 IAM 엔터티에 연결할 수 있습니다. 이 정책을 Amazon EC2 노드 생성 시에 지정한, Amazon EKS에서 사용자를 대신하여 작업을 수행하도록 허용하는 IAM 역할에 연결해야 합니다. `eksctl`을 사용하여 노드 그룹을 생성하는 경우, 노드 IAM 역할을 생성하고 이 정책을 역할에 자동으로 연결합니다.

이 정책은 Amazon EKS Amazon EC2 노드에 Amazon EKS 클러스터에 연결할 수 있는 권한을 부여합니다.

권한 세부 정보

이 정책에는 Amazon EKS가 다음 태스크를 완료할 수 있도록 허용하는 다음과 같은 권한이 포함되어 있습니다.

- **ec2** - 인스턴스 볼륨 및 네트워크 정보를 읽습니다. 이는 Kubernetes 노드가 Amazon EKS 클러스터에 가입하는 데 필요한 Amazon EC2 리소스에 대한 정보를 설명하는 데 필요합니다.

- **eks** - 선택적으로 클러스터를 노드 부트스트랩의 일부로 설명합니다.
- **eks-auth:AssumeRoleForPodIdentity** - 노드에서 EKS 워크로드에 대한 보안 인증 정보 검색을 허용합니다. EKS Pod Identity가 제대로 작동하려면 필요합니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AmazonEKSWorkerNodePolicy를 참조하십시오.

AWS 관리형 정책: AWSServiceRoleForAmazonEKSNodegroup

AWSServiceRoleForAmazonEKSNodegroup를 IAM 엔터티에 연결할 수 없습니다. 이 정책은 Amazon EKS에 사용자를 대신하여 작업을 수행할 수 있도록 서비스 연결 역할에 연결됩니다. 자세한 내용은 Amazon EKS에 대한 서비스 연결 역할 권한 단원을 참조하십시오.

이 정책은 계정에서 Amazon EC2 노드 그룹을 생성하고 관리하도록 허용하는 AWSServiceRoleForAmazonEKSNodegroup 역할 권한을 부여합니다.

권한 세부 정보

이 정책에는 Amazon EKS가 다음 태스크를 완료할 수 있도록 허용하는 다음과 같은 권한이 포함되어 있습니다.

- **ec2** - 보안 그룹, 태그 및 시작 템플릿을 사용합니다. Amazon EKS 관리 노드 그룹이 원격 액세스 구성을 활성화하는 데 필요합니다. 또한 Amazon EKS 관리 노드 그룹은 사용자를 대신하여 시작 템플릿을 생성합니다. 이는 각 관리형 노드 그룹을 백업하는 Amazon EC2 Auto Scaling 그룹을 구성하기 위한 것입니다.
- **iam** - 서비스 연결 역할을 생성하고 역할을 전달합니다. 이는 Amazon EKS 관리형 노드 그룹이 관리형 노드 그룹을 생성할 때 전달되는 역할의 인스턴스 프로필을 관리하는 데 필요합니다. 이 인스턴스 프로필은 관리형 노드 그룹의 일부로 시작된 Amazon EC2 인스턴스에 사용됩니다. Amazon EKS는 Amazon EC2 Auto Scaling 그룹과 같은 다른 서비스에 대한 서비스 연결 역할을 생성해야 합니다. 이러한 권한은 관리 노드 그룹 생성에 사용됩니다.
- **autoscaling** - 보안 Auto Scaling 그룹을 사용합니다. 이는 Amazon EKS 관리 노드 그룹이 각 관리 노드 그룹을 백업하는 Amazon EC2 Auto Scaling 그룹을 관리하는 데 필요합니다. 또한 노드 그룹 업데이트 중에 노드가 종료되거나 재활용될 때 Pods 제거와 같은 기능을 지원하는 데에도 사용됩니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AWSServiceRoleForAmazonEKSNodegroup를 참조하십시오.

AWS 관리형 정책: AmazonEBSCSIDriverPolicy

AmazonEBSCSIDriverPolicy 정책은 Amazon EBS Container Storage Interface(CSI) 드라이버가 사용자를 대신하여 볼륨을 생성, 수정, 연결, 분리 및 삭제할 수 있도록 허용합니다. 또한 EBS CSI 드라이버에 스냅샷을 생성 및 삭제하고 인스턴스, 볼륨 및 스냅샷을 나열할 수 있는 권한을 부여합니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AmazonEKS_CNI_Policy를 참조하십시오.

AWS 관리형 정책: AmazonEFSCSIDriverPolicy

AmazonEFSCSIDriverPolicy 정책을 통해 Amazon EFS 컨테이너 스토리지 인터페이스(CSI)가 사용자를 대신하여 액세스 포인트를 생성하고 삭제할 수 있습니다. 또한 Amazon EFS CSI 드라이버에 액세스 포인트, 파일 시스템, 탑재 대상 및 Amazon EC2 가용 영역을 나열할 수 있는 권한을 부여합니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AmazonEFSCSIDriverServiceRolePolicy를 참조하세요.

AWS 관리형 정책: AmazonEKSLocalOutpostClusterPolicy

IAM 엔터티에 이 정책을 연결할 수 있습니다. 로컬 클러스터를 생성하기 전에 이 정책을 클러스터 역할에 연결해야 합니다. Amazon EKS에서 관리하는 Kubernetes 클러스터는 사용자를 대신하여 다른 AWS 서비스를 호출합니다. 이러한 작업을 수행하여 서비스에 사용하는 리소스를 관리합니다.

AmazonEKSLocalOutpostClusterPolicy에는 다음 권한이 포함되어 있습니다.

- **ec2** - Amazon EC2 인스턴스가 성공적으로 클러스터에 컨트롤 플레인 인스턴스로 조인하는 데 필요한 권한입니다.
- **ssm** – Amazon EKS가 계정의 로컬 클러스터를 통신하고 관리하는 데 사용하는 컨트롤 플레인 인스턴스에 대한 Amazon EC2 Systems Manager 연결을 허용합니다.
- **logs** - 인스턴스가 Amazon CloudWatch에 로그를 푸시하도록 허용합니다.
- **secretsmanager** - 인스턴스가 AWS Secrets Manager에서 안전하게 컨트롤 플레인 인스턴스에 대한 부트스트랩 데이터를 가져오고 삭제하도록 허용합니다.
- **ecr** – 컨트롤 플레인 인스턴스에서 실행 중인 Pods 및 컨테이너가 Amazon Elastic Container Registry에 저장된 컨테이너 이미지를 끌어오도록 허용합니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AmazonEKS_CNI_Policy를 참조하십시오.

AWS 관리형 정책: AmazonEKSLocalOutpostServiceRolePolicy

IAM 엔터티에 이 정책을 연결할 수 없습니다. `iam:CreateServiceLinkedRole` 권한이 있는 IAM 보안 주체를 사용하여 클러스터를 생성하면 Amazon EKS가 자동으로 `AWSServiceRoleforAmazonEKSLocalOutpost` 서비스 연결 역할을 생성하고 이 정책을 연결합니다. 이 정책을 사용하면 서비스 연결 역할이 로컬 클러스터에 대해 사용자를 대신하여 AWS 서비스를 호출할 수 있습니다.

`AmazonEKSLocalOutpostServiceRolePolicy`에는 다음 권한이 포함되어 있습니다.

- **`ec2`** - Amazon EKS가 보안, 네트워크 및 기타 리소스와 함께 작동하여 계정에서 컨트롤 플레인 인스턴스를 성공적으로 시작하고 관리하도록 허용합니다.
- **`ssm`** – Amazon EKS가 계정의 로컬 클러스터를 통신하고 관리하는 데 사용하는 컨트롤 플레인 인스턴스에 대한 Amazon EC2 Systems Manager 연결을 허용합니다.
- **`iam`** - Amazon EKS가 컨트롤 플레인 인스턴스와 연결된 인스턴스 프로파일을 관리하도록 허용합니다.
- **`secretsmanager`** - 인스턴스 부트스트랩 중 안전하게 참조할 수 있게 Amazon EKS가 컨트롤 플레인 인스턴스에 대한 부트스트랩 데이터를 AWS Secrets Manager에 저장하도록 허용합니다.
- **`outposts`** - Amazon EKS가 계정에서 Outpost 정보를 가져와 Outpost에서 로컬 클러스터를 성공적으로 시작할 수 있도록 합니다.

최신 버전의 JSON 정책 문서를 보려면AWS 관리형 정책 참조 안내서의 AmazonEKS_CNI_Policy를 참조하십시오.

AWS 관리형 정책에 대한 Amazon EKS 업데이트

이 서비스에서 이러한 변경 사항 추적을 시작한 이후 Amazon EKS에 대한 AWS 관리형 정책 업데이트에 대한 세부 정보를 확인합니다. 이 페이지의 변경 사항에 대한 자동 알림을 받아보려면 Amazon EKS 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경 사항	설명	날짜
AmazonEKS_CNI_Policy – 기존 정책 업데이트	Amazon EKS에 Amazon VPC CNI plugin for Kubernetes에서 Amazon VPC 서브넷의 사용 가능한 IP 주소 양	2024년 3월 4일

변경 사항	설명	날짜
	을 볼 수 있도록 허용하는 `ec2:DescribeSubnets` 권한이 새로 추가되었습니다. VPC CNI는 각 서브넷의 사용 가능한 IP 주소를 사용하여 탄력적 네트워크 인터페이스를 생성할 때 사용 가능한 IP 주소가 가장 많은 서브넷을 선택할 수 있습니다.	
AmazonEKSWorkerNodePolicy - 기존 정책에 대한 업데이트	Amazon EKS에서 EKS Pod Identity를 허용하는 새로운 권한을 추가했습니다. Amazon EKS Pod Identity Agent는 노드 역할을 사용합니다.	2023년 11월 26일
AmazonEFSCSIDriverPolicy를 도입했습니다.	AWS에서는 `AmazonEFSCSIDriverPolicy` 를 도입했습니다.	2023년 7월 26일
AmazonEKSClusterPolicy에 권한을 추가했습니다.	Amazon EKS에서 로드 밸런서를 생성하는 동안 서브넷 자동 검색 중에 AZ 세부 정보를 가져올 수 있는 `ec2:DescribeAvailabilityZones` 권한이 추가되었습니다.	2023년 2월 7일
AmazonEBSCSIDriverPolicy의 정책 조건을 업데이트했습니다.	`StringLike` 키 필드에 와일드카드 문자가 있는 잘못된 정책 조건을 제거했습니다. 또한 in-tree 플러그인에서 생성된 볼륨을 EBS CSI 드라이버에서 삭제할 수 있는 새 조건 `ec2:ResourceTag/kubernetes.io/created-for/pvc/name: "*"` 를 `ec2:DeleteVolume` 에 추가했습니다.	2022년 11월 17일

변경 사항	설명	날짜
AmazonEKSLocalOutpostServiceRolePolicy에 권한이 추가되었습니다.	더 나은 사전 조건 검증 및 관리형 수명 주기 제어를 허용하기 위해 `ec2:DescribeVPCAttribute` , `ec2:GetConsoleOutput` 및 `ec2:DescribeSecret` 을 추가했습니다. 또한 Outposts의 컨트롤 플레인 Amazon EC2 인스턴스의 배치 제어를 지원하는 `ec2:RunInstances` 에 `ec2:DescribePlacementGroups` 및 `"arn:aws:ec2:*:*:placement-group/*"` 을 추가했습니다.	2022년 10월 24일
AmazonEKSLocalOutpostClusterPolicy의 Amazon Elastic Container Registry 권한을 업데이트합니다.	모든 리소스 부분에서 범위가 지정된 부분으로 `ecr:GetDownloadUrlForLayer` 작업을 이동했습니다. `arn:aws:ecr:*:*:repository/eks/*` 리소스를 추가했습니다. `arn:aws:ecr:*:*:repository/eks/eks-certificates-controller-public` 리소스를 제거했습니다. 이 리소스에는 추가한 `arn:aws:ecr:*:*:repository/eks/*` 리소스가 적용됩니다.	2022년 10월 20일
AmazonEKSLocalOutpostClusterPolicy에 권한이 추가되었습니다.	클러스터 컨트롤 플레인 인스턴스가 일부 `kubelet` 인수를 업데이트할 수 있도록 `arn:aws:ecr:*:*:repository/kubelet-config-updater` Amazon Elastic Container Registry 리포지토리가 추가되었습니다.	2022년 8월 31일
AmazonEKSLocalOutpostClusterPolicy가 도입되었습니다.	AWS에서는 `AmazonEKSLocalOutpostClusterPolicy` 를 도입했습니다.	2022년 8월 24일

변경 사항	설명	날짜
AmazonEKSLocalOutpostServiceRolePolicy가 도입되었습니다.	AWS에서는 AmazonEKSLocalOutpostServiceRolePolicy 를 도입했습니다.	2022년 8월 23일
AmazonEBSCSIDriverPolicy를 도입했습니다.	AWS에서는 AmazonEBSCSIDriverPolicy 를 도입했습니다.	2022년 4월 4일
AmazonEKSWorkerNodePolicy에 권한을 추가했습니다.	인스턴스 수준 속성을 자동으로 검색할 수 있는 Amazon EKS 최적화 AMI를 사용 설정하도록 ec2:DescribeInstanceTypes 를 추가했습니다.	2022년 3월 21일
AWSServiceRoleForAmazonEKSNodegroup에 권한을 추가했습니다.	Amazon EKS가 지표 수집을 사용할 수 있도록 autoscaling:EnableMetricsCollection 권한이 추가되었습니다.	2021년 12월 13일
AmazonEKSClusterPolicy에 권한을 추가했습니다.	Amazon EKS가 Network Load Balancer에 대한 서비스 연결 역할을 생성하도록 허용하는 ec2:DescribeAccountAttributes , ec2:DescribeAddresses 및 ec2:DescribeInternetGateways 권한이 추가됨.	2021년 6월 17일
Amazon EKS가 변경 사항 추적 시작.	Amazon EKS가 AWS 관리형 정책에 대한 변경 내용 추적을 시작했습니다.	2021년 6월 17일

IAM 문제 해결

이 주제에서는 IAM과 함께 Amazon EKS를 사용하는 동안 발생할 수 있는 몇 가지 일반적 오류와 이를 해결하는 방법을 다룹니다.

AccessDeniedException

AWS API 작업을 호출할 때 `AccessDeniedException`이 발생하면 사용하고 있는 IAM 보안 주체 자격 증명에 해당 호출을 수행할 필수 권한이 없는 것입니다.

```
An error occurred (AccessDeniedException) when calling the DescribeCluster operation:
User: arn:aws:iam::111122223333:user/user_name is not authorized to perform:
eks:DescribeCluster on resource: arn:aws:eks:region:111122223333:cluster/my-cluster
```

이전 예제 메시지에서 사용자는 Amazon EKS `DescribeCluster` API 작업을 호출할 권한이 없습니다. IAM 보안 주체에게 Amazon EKS 관리 권한을 부여하려면 Amazon EKS 자격 증명 기반 정책 예제 섹션을 참조하세요.

IAM에 자세한 내용은 IAM 사용 설명서에서 정책을 사용하여 액세스 제어를 참조하세요.

컴퓨팅 탭에서 노드 또는 리소스 탭에서 아무것도 볼 수 없으며 AWS Management Console에서 오류가 발생합니다.

`Your current user or role does not have access to Kubernetes objects on this EKS cluster`라는 콘솔 오류 메시지가 나타날 수 있습니다. AWS Management Console를 함께 사용하는 IAM 보안 주체 사용자에게 필요한 권한이 있는지 확인하세요. 자세한 내용은 필요한 권한 단원을 참조하십시오.

aws-auth **`ConfigMap`**은 클러스터에 대한 액세스 권한을 부여하지 않습니다.

AWS IAM Authenticator는 `ConfigMap`에 사용된 역할 ARN의 경로를 허용하지 않습니다. 따라서 `rolearn`를 지정하기 전에 경로를 제거하십시오. 예를 들면 `arn:aws:iam::111122223333:role/team/developers/eks-admin`를 `arn:aws:iam::111122223333:role/eks-admin`로 변경합니다.

iam:PassRole을 수행하도록 인증되지 않음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Amazon EKS에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 해당 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 Amazon EKS에서 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 필요한 경우 AWS관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 AWS 계정 외부의 사용자가 내 Amazon EKS 리소스에 액세스하도록 허용하려고 합니다.

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

자세히 알아보려면 다음을 참조하세요.

- Amazon EKS에서 이러한 기능을 지원하는지 알아보려면 Amazon EKS가 IAM과 작동하는 방식 섹션을 참조하세요.
- 소유하고 있는 AWS 계정의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용자 설명서의 자신이 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 권한 제공을 참조하세요.
- 리소스에 대한 액세스 권한을 타사 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 소유한 AWS 계정에 대한 액세스 제공을 참조하세요.
- 보안 인증 연동을 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.

포드 컨테이너는 다음과 같은 오류가 발생합니다. **An error occurred (SignatureDoesNotMatch) when calling the GetCallerIdentity operation: Credential should be scoped to a valid region**

애플리케이션이 명시적으로 AWS STS 글로벌 엔드포인트(`https://sts.amazonaws`)를 요청하고 Kubernetes 서비스 계정이 리전별 엔드포인트를 사용하도록 구성된 경우 컨테이너에 이 오류가 표시됩니다. 다음 옵션 중 하나를 사용하여 문제를 해결할 수 있습니다.

- 애플리케이션 코드를 업데이트하여 AWS STS 전역 엔드포인트에 대한 명시적 호출을 제거합니다.
- 애플리케이션 코드를 업데이트하여 `https://sts.us-west-2.amazonaws.com`와 같은 리전별 엔드포인트를 명시적으로 호출합니다. 애플리케이션에는 해당 AWS 리전에서 서비스 장애가 발생한 경우 다른 AWS 리전을 선택하기 위해 기본 제공되는 중복성이 있어야 합니다. 자세한 내용은 IAM 사용 설명서의 AWS 리전에서 AWS STS 관리를 참조하세요.
- 전역 엔드포인트를 사용하도록 서비스 계정을 구성합니다. `1.22` 이하인 모든 버전은 기본적으로 글로벌 엔드포인트를 사용했지만 버전 `1.22` 이상의 클러스터는 기본적으로 리전별 엔드포인트를 사용합니다. 자세한 내용은 서비스 계정의 AWS Security Token Service 엔드포인트 구성 단원을 참조하십시오.

기본 Amazon EKS에서 생성한 Kubernetes 역할 및 사용자

Kubernetes 클러스터를 생성하면 Kubernetes 클러스터가 제대로 작동하도록 해당 클러스터에 여러 개의 기본 Kubernetes 자격 증명이 생성됩니다. Amazon EKS는 각 기본 구성 요소에 대한Kubernetes ID를 생성합니다. 자격 증명에서는 클러스터 구성 요소에 대한 Kubernetes RBAC(역할 기반 권한 부여 제어)가 제공됩니다. 자세한 내용은 Kubernetes 설명서의 RBAC 승인 사용을 참조하세요.

클러스터에 선택적 추가 기능을 설치하면 클러스터에 추가 Kubernetes 자격 증명이 추가될 수도 있습니다. 이 주제에서 다루지 않는 자격 증명에 대한 자세한 내용은 추가 기능에 대한 설명서를 참조하세요.

Amazon EKS에서 클러스터에 생성한 Kubernetes 자격 증명 목록은 AWS Management Console 또는 `kubectl` 명령줄 도구를 사용하여 볼 수 있습니다. 모든 사용자 자격 증명은 Amazon CloudWatch를 통해 사용할 수 있는 kube 감사 로그에 표시됩니다.

AWS Management Console

전제 조건

필요한 권한에서 설명한 권한이 사용하는 IAM 보안 주체에 있어야 합니다.

AWS Management Console을 사용하여 Amazon EKS에서 생성한 자격 증명을 보는 방법

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 보려는 자격 증명이 포함된 클러스터를 Clusters(클러스터) 목록에서 선택합니다.
3. 리소스 탭을 선택합니다.
4. Resource types(리소스 유형)에서 Authorization(권한 부여)을 선택합니다.
5. ClusterRoles, ClusterRoleBindings, Roles 또는 RoleBindings를 선택합니다. 앞에 eks로 시작하는 모든 리소스는 Amazon EKS에서 생성합니다. Amazon EKS에서 생성한 추가 자격 증명 리소스는 다음과 같습니다.

   - ClusterRole 및 aws-node라는 ClusterRoleBinding. aws-node 리소스에서는 Amazon EKS에서 모든 클러스터에 설치하는 Amazon VPC CNI plugin for Kubernetes을 지원합니다.
   - vpc-resource-controller-role이라는 ClusterRole 및 vpc-resource-controller-rolebinding이라는 ClusterRoleBinding. 이러한 리소스에서는 Amazon EKS에서 모든 클러스터에 설치하는 Amazon VPC 리소스 컨트롤러를 지원합니다.

   콘솔에 표시되는 리소스 외에 다음과 같은 특별한 사용자 자격 증명이 클러스터에 있지만, 클러스터의 구성에는 표시되지 않습니다.

   - **`eks:cluster-bootstrap`** – 클러스터 부트스트랩 중 `kubectl` 작업에 사용됩니다.
   - **`eks:support-engineer`** – 클러스터 관리 작업에 사용됩니다.
6. 특정 리소스를 선택하면 해당 세부 정보를 볼 수 있습니다. 기본적으로 Structured view(구조적 뷰)에 정보가 표시됩니다. 세부 정보 페이지의 오른쪽 상단에서 Raw view(원시 뷰)를 선택하여 리소스에 대한 모든 정보를 볼 수 있습니다.

Kubectl

전제 조건

클러스터의 Kubernetes 리소스를 나열하는 데 사용하는 엔터티(AWS Identity and Access Management(IAM) 또는 OpenID Connect(OIDC))는 IAM 또는 OIDC 자격 증명 제공업체에서 인증해야 합니다. 엔터티를 연동하려는 클러스터의 `Role`, `ClusterRole`, `RoleBinding` 및 `ClusterRoleBinding` 리소스에 Kubernetes `get` 및 `list` 동사를 사용하는 권한이 엔터티에 부여되어야 합니다. IAM 엔터티에 클러스터 액세스 권한을 부여하는 자세한 내용은 the section called “Kubernetes API에 대한 액세스 권한 부여” 섹션을 참조하세요. 자체 OIDC 제공업체에서 인

증한 엔터티에 클러스터 액세스 권한을 부여하는 것에 대한 자세한 내용은 OpenID Connect 자격 증명 공급자에서 클러스터에 대해 사용자 인증 섹션을 참조하세요.

kubectl을 사용하여 Amazon EKS에서 생성한 자격 증명을 보는 방법

표시하려는 리소스 유형에 대한 명령을 실행합니다. eks로 시작하는 반환된 모든 리소스는 Amazon EKS에서 생성합니다. 출력에서 반환되는 리소스 외에 다음과 같은 특별한 사용자 자격 증명이 클러스터에 있지만, 클러스터의 구성에는 표시되지 않습니다.

- **eks:cluster-bootstrap** – 클러스터 부트스트랩 중 kubectl 작업에 사용됩니다.
- **eks:support-engineer** – 클러스터 관리 작업에 사용됩니다.

ClusterRoles – ClusterRoles의 범위가 클러스터로 지정되므로 역할에 부여된 모든 권한이 클러스터의 모든 Kubernetes 네임스페이스에 있는 리소스에 적용됩니다.

다음과 같은 명령에서는 Amazon EKS에서 생성한 클러스터의 Kubernetes ClusterRoles를 모두 반환합니다.

```
kubectl get clusterroles | grep eks
```

출력에서 반환되어 앞에 붙는 ClusterRoles 외에 다음과 같은 ClusterRoles가 있습니다.

- **aws-node** – 이 ClusterRole에서는 Amazon EKS에서 모든 클러스터에 설치하는 Amazon VPC CNI plugin for Kubernetes를 지원합니다.
- **vpc-resource-controller-role** – 이 ClusterRole에서는 Amazon EKS에서 모든 클러스터에 설치하는 Amazon VPC 리소스 컨트롤러를 지원합니다.

ClusterRole의 사양을 표시하려면 다음과 같은 명령의 *eks:k8s-metrics*를 이전 명령의 출력에서 반환된 ClusterRole로 바꿉니다. 다음 예시에서는 *eks:k8s-metrics* ClusterRole에 대한 사양을 반환합니다.

```
kubectl describe clusterrole eks:k8s-metrics
```

예제 출력은 다음과 같습니다.

```
Name:         eks:k8s-metrics
Labels:       <none>
```

```
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names  Verbs
  ---------         -----------------  --------------  -----
                    [/metrics]         []              [get]
  endpoints         []                 []              [list]
  nodes             []                 []              [list]
  pods              []                 []              [list]
  deployments.apps  []                 []              [list]
```

ClusterRoleBindings – `ClusterRoleBindings`의 범위가 클러스터로 지정됩니다.

다음과 같은 명령에서는 Amazon EKS에서 생성한 클러스터의 Kubernetes `ClusterRoleBindings`를 모두 반환합니다.

```
kubectl get clusterrolebindings | grep eks
```

출력에서 반환된 `ClusterRoleBindings` 외에 다음과 같은 `ClusterRoleBindings`가 있습니다.

- **`aws-node`** – 이 `ClusterRoleBinding`에서는 Amazon EKS에서 모든 클러스터에 설치하는 Amazon VPC CNI plugin for Kubernetes를 지원합니다.
- **`vpc-resource-controller-rolebinding`** – 이 `ClusterRoleBinding`에서는 Amazon EKS에서 모든 클러스터에 설치하는 Amazon VPC 리소스 컨트롤러를 지원합니다.

`ClusterRoleBinding`의 사양을 표시하려면 다음과 같은 명령의 ***`eks:k8s-metrics`***를 이전 명령의 출력에서 반환된 `ClusterRoleBinding`로 바꿉니다. 다음 예시에서는 ***`eks:k8s-metrics`*** `ClusterRoleBinding`에 대한 사양을 반환합니다.

```
kubectl describe clusterrolebinding eks:k8s-metrics
```

예제 출력은 다음과 같습니다.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  eks:k8s-metrics
Subjects:
```

```
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Roles – Roles의 범위가 Kubernetes 네임스페이스로 지정됩니다. Amazon EKS에서 생성한 모든 Roles의 범위가 kube-system 네임스페이스로 지정됩니다.

다음과 같은 명령에서는 Amazon EKS에서 생성한 클러스터의 Kubernetes Roles를 모두 반환합니다.

```
kubectl get roles -n kube-system | grep eks
```

Role의 사양을 표시하려면 다음과 같은 명령의 *eks:k8s-metrics*를 이전 명령의 출력에서 반환된 Role의 이름으로 변경합니다. 다음 예시에서는 *eks:k8s-metrics* Role에 대한 사양을 반환합니다.

```
kubectl describe role eks:k8s-metrics -n kube-system
```

예제 출력은 다음과 같습니다.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources         Non-Resource URLs  Resource Names              Verbs
  ---------         -----------------  --------------              -----
  daemonsets.apps   []                 [aws-node]                  [get]
  deployments.apps  []                 [vpc-resource-controller]  [get]
```

RoleBindings – RoleBindings의 범위가 Kubernetes 네임스페이스로 지정됩니다. Amazon EKS에서 생성한 모든 RoleBindings의 범위가 kube-system 네임스페이스로 지정됩니다.

다음과 같은 명령에서는 Amazon EKS에서 생성한 클러스터의 Kubernetes RoleBindings를 모두 반환합니다.

```
kubectl get rolebindings -n kube-system | grep eks
```

RoleBinding의 사양을 표시하려면 다음과 같은 명령의 *eks:k8s-metrics*를 이전 명령의 출력에서 반환된 RoleBinding으로 바꿉니다. 다음 예시에서는 *eks:k8s-metrics* RoleBinding에 대한 사양을 반환합니다.

```
kubectl describe rolebinding eks:k8s-metrics -n kube-system
```

예제 출력은 다음과 같습니다.

```
Name:         eks:k8s-metrics
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  Role
  Name:  eks:k8s-metrics
Subjects:
  Kind  Name             Namespace
  ----  ----             ---------
  User  eks:k8s-metrics
```

Amazon Elastic Kubernetes Service에 대한 규정 준수 검증

AWS 서비스가 특정 규정 준수 프로그램의 범위에 포함되는지 알아보려면 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반적인 정보는 AWS 규정 준수 프로그램을 참조하세요.

AWS Artifact를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 AWS Artifact에서 보고서 다운로드를 참조하세요.

AWS 서비스 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 – 이 배포 안내서에서는 아키텍처 고려 사항에 대해 설명하고 보안 및 규정 준수에 중점을 둔 기본 AWS 환경을 배포하기 위한 단계를 제공합니다.
- Amazon Web Services에서 HIPAA 보안 및 규정 준수 기술 백서 설계 - 이 백서는 기업에서 AWS를 사용하여 HIPAA를 준수하는 애플리케이션을 만드는 방법을 설명합니다.

> **Note**
>
> 모든 AWS 서비스에 HIPAA 자격이 있는 것은 아닙니다. 자세한 내용은 HIPAA 적격 서비스 참조를 참조하세요.

- AWS 규정 준수 리소스 - 고객 조직이 속한 산업 및 위치에 적용될 수 있는 워크북 및 가이드 콜렉션입니다.

- AWS 고객 규정 준수 가이드 - 규정 준수의 관점에서 공동 책임 모델을 이해합니다. 이 가이드에서는 AWS 서비스를 보호하기 위한 모범 사례를 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 사용하여 리소스 평가 – AWS Config 서비스는 내부 사례, 산업 지침 및 규제에 대한 리소스 구성의 준수 상태를 평가합니다.
- AWS Security Hub – 이 AWS 서비스는 AWS 내의 보안 상태에 대한 포괄적인 보기를 제공합니다. 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 참조하세요.
- AWS Audit Manager - 이 AWS 서비스는 AWS 사용을 지속해서 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 간소화할 수 있도록 지원합니다.

Amazon EKS의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전에서는 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며 이러한 가용 영역은 짧은 대기 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 복수 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

Amazon EKS는 여러 AWS 가용 영역에 걸쳐 Kubernetes 컨트롤 플레인을 실행하고 크기 조정하여 높은 가용성을 보장합니다. Amazon EKS는 로드에 따라 제어 영역 인스턴스의 크기를 자동으로 조정하고, 비정상 제어 영역 인스턴스를 감지하고 교체하며, 자동으로 제어 영역을 패치합니다. 버전 업데이트를 시작하면 Amazon EKS가 제어 영역을 업데이트하여 업데이트 중에 제어 영역의 고가용성을 유지합니다.

이 제어 플레인은 2개 이상의 API 서버 인스턴스와 AWS 리전 내 3개의 가용 영역에서 실행되는 3개의 `etcd` 인스턴스로 구성됩니다. Amazon EKS:

- 컨트롤 플레인 인스턴스의 로드를 적극적으로 모니터링하고 자동으로 확장하여 높은 성능을 보장합니다.
- 비정상 제어 플레인 인스턴스를 자동으로 감지하고 교체하여 필요에 따라 AWS 리전 내의 가용 영역에서 다시 시작합니다.
- AWS 리전의 아키텍처를 활용하여 고가용성을 유지합니다. 따라서 Amazon EKS는 API 서버 엔드포인트 가용성을 위한 SLA를 제공할 수 있습니다.

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

Amazon EKS의 인프라 보안

관리형 서비스인 Amazon Elastic Kubernetes Service는 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스와 AWS의 인프라 보호 방법에 대한 자세한 내용은 AWS 클라우드 보안을 참조하세요. 인프라 보안에 대한 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS에서 게시한 API 호출을 사용하여 네트워크를 통해 Amazon EKS에 액세스합니다. 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 인증 정보를 생성하여 요청에 서명할 수 있습니다.

Amazon EKS 클러스터를 생성할 때 클러스터가 사용할 VPC 서브넷을 지정합니다. Amazon EKS에는 최소 2개의 가용 영역에 있는 서브넷이 필요합니다. Kubernetes가 프라이빗 서브넷에 있는 노드에서 실행되는 Pods로 트래픽을 로드 밸런싱하는 퍼블릭 서브넷에 퍼블릭 로드 밸런서를 생성할 수 있도록 퍼블릭 및 프라이빗 서브넷이 있는 VPC를 사용하는 것이 좋습니다.

VPC 고려 사항에 대한 자세한 내용은 Amazon EKS VPC 및 서브넷 요구 사항과 고려 사항 단원을 참조하십시오.

Amazon EKS 시작하기 시연에 제공된 AWS CloudFormation 템플릿을 사용하여 VPC 및 노드를 생성하는 경우 제어 영역 및 노드 보안 그룹은 권장 설정으로 구성됩니다.

보안 그룹 고려 사항에 대한 자세한 내용은 Amazon EKS 보안 그룹 요구 사항 및 고려 사항 단원을 참조하십시오.

새 클러스터를 생성할 때 Amazon EKS에서는 클러스터와 통신하는 데 사용하는 관리형 Kubernetes API 서버에 대한 엔드포인트를 생성합니다(`kubectl`과 같은 Kubernetes 관리 도구 사용). 기본적으로 이 API 서버 엔드포인트는 인터넷에 공개되어 있으며, API 서버에 대한 액세스는 AWS Identity and

Access Management(IAM) 및 기본 Kubernetes 역할 기반 액세스 제어(RBAC)의 조합을 통해 보호됩니다.

노드와 API 서버 간의 모든 통신이 VPC 내에 유지되도록 Kubernetes API 서버에 대한 프라이빗 액세스를 활성화할 수 있습니다. 인터넷에서 API 서버로 액세스하는 IP 주소를 제한하거나 API 서버로의 인터넷 액세스를 완전히 비활성화할 수 있습니다.

클러스터 엔드포인트 액세스 수정에 대한 자세한 내용은 클러스터 엔드포인트 액세스 수정 단원을 참조하십시오.

Amazon VPC CNI 또는 Project Calico와 같은 타사 도구로 Kubernetes 네트워크 정책을 구현할 수 있습니다. 네트워크 정책에 맞는 Amazon VPC CNI 사용에 관한 자세한 내용은 클러스터에 Kubernetes 네트워크 정책 구성 섹션을 참조하세요. Project Calico는 타사 오픈 소스 프로젝트입니다. 자세한 내용은 Project Calico 설명서를 참조하세요.

Amazon EKS의 구성 및 취약성 분석

보안은 Kubernetes 클러스터 및 애플리케이션을 구성하고 유지 관리하는 데 중요한 고려 사항입니다. Center for Internet Security(CIS) Kubernetes 벤치마크는 Amazon EKS 노드 보안 구성에 대한 지침을 제공합니다. 이 벤치마크의 기능:

- Kubernetes 구성 요소의 보안 구성을 담당하는 Amazon EC2 노드(관리형 노드 및 자체 관리형 노드 모두)에 적용할 수 있습니다.
- Amazon EKS를 사용할 때 Kubernetes 클러스터와 노드를 안전하게 구성할 수 있도록 커뮤니티에서 승인한 표준 방법을 제공합니다.
- 컨트롤 플레인 로깅 구성, 노드 보안 구성, 정책 및 관리형 서비스의 4개 섹션으로 구성됩니다.
- 현재 Amazon EKS에서 사용할 수 있는 모든 Kubernetes 버전을 지원하며, Kubernetes 클러스터에서 CIS 벤치마크를 사용하여 구성을 확인하기 위한 표준 오픈 소스 도구인 kube-bench를 사용하여 실행할 수 있습니다.

자세한 내용은 CIS Amazon EKS 벤치마크 소개를 참조하세요.

Amazon EKS 플랫폼 버전은 활성화된 Kubernetes API 서버 플래그와 현재 Kubernetes 패치 버전 등 클러스터 컨트롤 플레인의 기능을 나타냅니다. 새 클러스터는 최신 플랫폼 버전과 함께 배포됩니다. 자세한 내용은 Amazon EKS 플랫폼 버전 단원을 참조하세요.

새로운 Kubernetes 버전으로 Amazon EKS 클러스터를 업데이트할 수 있습니다. Amazon EKS에서 새로운 Kubernetes 버전을 사용할 수 있게 되면 미리 클러스터를 업데이트하여 최신 버전을 사용하는 것

이 좋습니다. EKS의 Kubernetes 버전에 대한 자세한 내용은 Amazon EKS Kubernetes 버전 섹션을 참조하세요.

Amazon Linux 보안 센터에서 Amazon Linux 2의 보안 또는 프라이버시 이벤트를 추적하거나 관련 RSS 피드를 구독하세요. 보안 및 프라이버시 이벤트에는 영향을 받는 문제의 개요, 패키지, 인스턴스 업데이트로 문제를 해결하는 방법이 포함됩니다.

Amazon Inspector를 사용하여 노드의 의도하지 않은 네트워크 액세스 가능성과 이러한 Amazon EC2 인스턴스 취약성을 검사할 수 있습니다.

Amazon EKS에 대한 보안 모범 사례

Amazon EKS 보안 모범 사례는 Github(https://aws.github.io/aws-eks-best-practices/security/docs/)에서 유지 관리됩니다.

포드 보안 정책

Kubernetes Pod 보안 정책 승인 컨트롤러는 일련의 규칙을 기준으로 Pod 생성 및 업데이트 요청을 검증합니다. 기본적으로 Amazon EKS 클러스터는 제한 없이 완전히 허용되는 보안 정책과 함께 제공됩니다. 자세한 내용은 Kubernetes 설명서의 포드 보안 정책을 참조하세요.

> **Note**
>
> `PodSecurityPolicy`(PSP)는 Kubernetes 버전 `1.21`에서 사용 중단되었으며 Kubernetes `1.25`에서 제거되었습니다. PSPs는 PSS(포드 보안 표준)에 간략히 설명된 보안 컨트롤을 구현하는 기본 제공 승인 컨트롤러인 PSA(포드 보안 승인)로 바뀌고 있습니다. PSA와 PSS는 모두 베타 기능 상태에 도달했으며, 기본적으로 Amazon EKS에서 활성화되어 있습니다. `1.25`의 PSP 제거를 해결하려면 Amazon EKS에서 PSS를 구현하는 것이 좋습니다. 자세한 내용은 AWS 블로그의 Implementing Pod Security Standards in Amazon EKS(Amazon EKS에서 포드 보안 표준 구현)를 참조하세요.

Amazon EKS 기본 Pod 보안 정책

Kubernetes 버전 `1.13` 또는 이후 버전의 Amazon EKS 클러스터에는 `eks.privileged`라는 기본 Pod 보안 정책이 있습니다. 이 정책에는 시스템에 수용할 수 있는 Pod 종류에 대한 제한이 없기 때문에 `PodSecurityPolicy` 컨트롤러가 비활성화된 상태로 Kubernetes를 실행하는 것과 같습니다.

> **Note**
>
> 이 정책은 PodSecurityPolicy 컨트롤러가 활성화되지 않은 클러스터와의 이전 버전 호환성을 유지하기 위해 만들어졌습니다. 클러스터와 개별 네임스페이스 및 서비스 계정에 대해 더 제한적인 정책을 만든 다음 기본 정책을 삭제해 더 제한적인 정책을 활성화할 수 있습니다.

다음 명령을 사용하여 기본 정책을 볼 수 있습니다.

```
kubectl get psp eks.privileged
```

예제 출력은 다음과 같습니다.

```
NAME             PRIV   CAPS   SELINUX    RUNASUSER   FSGROUP    SUPGROUP
 READONLYROOTFS   VOLUMES
eks.privileged   true   *      RunAsAny   RunAsAny    RunAsAny   RunAsAny   false
      *
```

세부 정보를 추가하려면 다음 명령을 사용하여 정책을 설명할 수 있습니다.

```
kubectl describe psp eks.privileged
```

예제 출력은 다음과 같습니다.

```
Name:  eks.privileged

Settings:
  Allow Privileged:                       true
  Allow Privilege Escalation:             0xc0004ce5f8
  Default Add Capabilities:               <none>
  Required Drop Capabilities:             <none>
  Allowed Capabilities:                   *
  Allowed Volume Types:                   *
  Allow Host Network:                     true
  Allow Host Ports:                       0-65535
  Allow Host PID:                         true
  Allow Host IPC:                         true
  Read Only Root Filesystem:              false
  SELinux Context Strategy: RunAsAny
    User:                                 <none>
    Role:                                 <none>
```

```
    Type:                                 <none>
    Level:                                <none>
  Run As User Strategy: RunAsAny
    Ranges:                               <none>
  FSGroup Strategy: RunAsAny
    Ranges:                               <none>
  Supplemental Groups Strategy: RunAsAny
    Ranges:                               <none>
```

기본 Pod 보안 정책을 설치 또는 복원에서 `eks.privileged` Pod 보안 정책, 클러스터 역할, 클러스터 역할 바인딩을 위한 완전한 YAML 파일을 볼 수 있습니다.

기본 Amazon EKS Pod 보안 정책 삭제

Pods에 대해 더 제한적인 정책을 생성하는 경우 그런 다음 기본 Amazon EKS `eks.privileged` Pod 보안 정책을 삭제하여 사용자 지정 정책을 사용 설정할 수 있습니다.

> **Important**
>
> CNI 플러그인 버전 `1.7.0` 이상을 사용하고 있고 DaemonSet에 의해 배포된 `aws-node` Pods에 사용되는 `aws-node` Kubernetes 서비스 계정에 사용자 지정 Pod 보안 정책을 할당한 경우, 정책의 `allowedCapabilities`섹션에 `NET_ADMIN`이 있고 정책의 `spec`에 `hostNetwork: true`와 `privileged: true`가 있어야 합니다.

기본 Pod 보안 정책을 삭제하려면 다음을 수행합니다.

1. 기본 Pod 보안 정책을 설치 또는 복원에 예제 파일의 내용으로 *`privileged-podsecuritypolicy.yaml`*이라는 파일을 생성합니다.
2. 다음 명령을 사용하여 YAML을 삭제합니다. 이렇게 하면 기본 Pod 보안 정책인 `ClusterRole` 및 이와 연결된 `ClusterRoleBinding`이 삭제됩니다.

   ```
   kubectl delete -f privileged-podsecuritypolicy.yaml
   ```

기본 Pod 보안 정책을 설치 또는 복원

이전 버전의 Kubernetes에서 업그레이드하거나 기본 Amazon EKS `eks.privileged` Pod 보안 정책을 수정 또는 삭제한 경우 다음 단계에 따라 복원할 수 있습니다.

기본 Pod 보안 정책을 설치하거나 복원하려면 다음을 수행합니다.

1. 다음 콘텐츠로 *privileged-podsecuritypolicy.yaml*이라는 파일을 생성합니다.

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: eks.privileged
  annotations:
    kubernetes.io/description: 'privileged allows full unrestricted access to
      Pod features, as if the PodSecurityPolicy controller was not enabled.'
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: '*'
  labels:
    kubernetes.io/cluster-service: "true"
    eks.amazonaws.com/component: pod-security-policy
spec:
  privileged: true
  allowPrivilegeEscalation: true
  allowedCapabilities:
  - '*'
  volumes:
  - '*'
  hostNetwork: true
  hostPorts:
  - min: 0
    max: 65535
  hostIPC: true
  hostPID: true
  runAsUser:
    rule: 'RunAsAny'
  seLinux:
    rule: 'RunAsAny'
  supplementalGroups:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'
  readOnlyRootFilesystem: false

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:podsecuritypolicy:privileged
```

```
  labels:
    kubernetes.io/cluster-service: "true"
    eks.amazonaws.com/component: pod-security-policy
rules:
- apiGroups:
  - policy
  resourceNames:
  - eks.privileged
  resources:
  - podsecuritypolicies
  verbs:
  - use

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:podsecuritypolicy:authenticated
  annotations:
    kubernetes.io/description: 'Allow all authenticated users to create privileged
 Pods.'
  labels:
    kubernetes.io/cluster-service: "true"
    eks.amazonaws.com/component: pod-security-policy
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:podsecuritypolicy:privileged
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: system:authenticated
```

2. 다음 명령을 사용하여 YAML을 적용합니다.

```
kubectl apply -f privileged-podsecuritypolicy.yaml
```

포드 보안 정책 (PSP) 제거 FAQ

PodSecurityPolicy은 Kubernetes1.21에서 더 이상 사용되지 않으며 Kubernetes1.25에서 제거되었습니다. 클러스터에서 PodSecurityPolicy를 사용하는 경우, 워크로드 중단을 방지하기 위해 클러

스터를 버전 **1.25**으로 업그레이드하기 전에 내장된 Kubernetes Pod Security Standards (PSS) 또는 policy-as-code 솔루션으로 마이그레이션해야 합니다. 자세히 알아보려면 자주 묻는 질문을 선택하세요.

PSP란 무엇입니까?

PodSecurityPolicy는 클러스터 관리자가 Pod 사양의 보안에 민감한 측면을 제어할 수 있는 내장형 승인 컨트롤러입니다. Pod가 PSP의 요구 사항을 충족하면 해당 Pod는 평소와 같이 클러스터에 승인됩니다. Pod가 PSP 요구 사항을 충족하지 않으면 해당 Pod는 거부되어 실행할 수 없습니다.

PSP 제거는 Amazon EKS에만 해당됩니까, 아니면 업스트림 Kubernetes에서 제거됩니까?

이는 Kubernetes 프로젝트의 업스트림 변경이며 Amazon EKS에서 변경한 내용이 아닙니다. PSP은(는) Kubernetes `1.21`에서 더 이상 사용되지 않고 Kubernetes `1.25`에서 제거되었습니다. Kubernetes 커뮤니티에서 PSP를 포함한 심각한 사용성 문제를 확인했습니다. 여기에는 의도한 것보다 더 광범위한 권한을 실수로 부여하거나 PSPs이(가) 특정 상황에 해당하는 권한을 검사하기 어려운 경우가 포함되었습니다. 이러한 문제는 주요 변경 없이는 해결할 수 없습니다. 이것이 Kubernetes 커뮤니티가 PSP를 제거하기로 결정한 주된 이유입니다.

Amazon EKS 클러스터의 PSPs를 사용하고 있는지 어떻게 확인할 수 있나요?

클러스터에서 PSPs를 사용 중인지 확인하려면 다음 명령을 실행하면 확인할 수 있습니다.

```
kubectl get psp
```

클러스터의 PSPs가 영향을 주는 Pods를 보려면 다음 명령을 실행합니다. 이 명령은 Pod 이름, 네임스페이스 및 PSPs를 출력합니다.

```
kubectl get pod -A -o jsonpath='{range.items[?(@.metadata.annotations.kubernetes
\.io/psp)]}{.metadata.name}{"\t"}{.metadata.namespace}{"\t"}
{.metadata.annotations.kubernetes\.io/psp}{"\n"}'
```

Amazon EKS 클러스터의 PSPs를 사용하는 경우 무엇을 할 수 있습니까?

클러스터를 `1.25`로 업그레이드하기 전에 다음 대안 중 하나로 PSPs를 마이그레이션해야 합니다.

- Kubernetes PSS.

- Kubernetes 환경에서 제공되는 Policy-as-code 솔루션.

PSP 지원 중단과 처음부터 Pod 보안을 제어해야 하는 지속적인 필요성에 대응하여 Kubernetes 커뮤니티는 PSS 및 Pod Security Admission(PSA)을 갖춘 기본 제공 솔루션을 만들었습니다. PSA 웹후크는 PSS에 정의된 컨트롤을 구현합니다.

EKS 모범 사례 가이드에서 PSPs를 내장형 PSS로 마이그레이션하기 위한 모범 사례를 검토할 수 있습니다. 또한 Amazon EKS에서 포드 보안 표준 구현 블로그를 검토하세요. 추가 레퍼런스로는 PodSecurityPolic에서 빌트인 PodSecurity Admission Controller로 마이그레이션하기 및 PodSecurityPolicies를 포드 보안 표준으로 매핑하기 등이 있습니다.

Policy-as-code 솔루션은 클러스터 사용자를 안내하고 사전 정의된 자동 제어를 통해 원치 않는 동작을 방지하는 가드레일을 제공합니다. Policy-as-code 솔루션은 일반적으로 Kubernetes 동적 승인 컨트롤러를 사용하여 웹후크 호출을 통해 Kubernetes API 서버 요청 흐름을 차단합니다. Policy-as-code 솔루션은 코드로 작성 및 저장된 정책을 기반으로 요청 페이로드를 변경하고 검증합니다.

Kubernetes에 사용할 수 있는 오픈 소스 policy-as-code 솔루션이 몇 가지 있습니다. PSPs를 policy-as-code 솔루션으로 마이그레이션하기 위한 모범 사례를 검토하려면 GitHub의 Pod Security 페이지의 Policy-as-code 섹션을 참조하세요.

클러스터에 PSP 호출 **eks.privileged**이 표시됩니다. 이것이 무엇이고 어떻게 해야 하나요?

Kubernetes 버전 `1.13` 또는 이후 버전의 Amazon EKS 클러스터에는 `eks.privileged`라는 기본 PSP가 있습니다. 이 정책은 `1.24` 및 이전 클러스터에서 생성됩니다. `1.25` 및 이후 클러스터에서는 사용되지 않습니다. Amazon EKS는 이 PSP를 PSS 기반 집행 기관으로 자동 마이그레이션합니다. 사용자는 아무 작업도 수행할 필요가 없습니다.

클러스터를 버전 **1.25**으로 업데이트할 때 Amazon EKS에서 기존 클러스터에 존재하는 PSPs를 변경하나요?

아니요. 또한 Amazon EKS에서 생성된 PSP인 `eks.privileged` 이외에도 `1.25`로 업그레이드할 때 클러스터의 다른 PSPs에는 변경 사항이 적용되지 않습니다.

PSP으로 마이그레이션하지 않은 경우 Amazon EKS에서 클러스터를 버전 **1.25**으로 업데이트할 수 없나요?

아니요. Amazon EKS는 아직 PSP으로 마이그레이션하지 않은 경우 클러스터의 버전 `1.25` 업데이트를 막지 않습니다.

내 클러스터를 버전 1.25으로 업데이트하기 전에 PSPs을 PSS/PSA 또는 policy-as-code 솔루션으로 마이그레이션하는 것을 잊은 경우 어떻게 합니까? 클러스터를 업데이트한 후에 마이그레이션할 수 있나요?

PSP가 포함된 클러스터를 Kubernetes 버전 1.25로 업그레이드하면 API 서버가 1.25의 PSP 리소스를 인식하지 못합니다. 이로 인해 Pods에 잘못된 보안 범위가 지정될 수 있습니다. 이와 관련된 전체 목록은 PodSecurityPolicy에서 빌트인 PodSecurity Admission Controller로 마이그레이션하기를 참조하세요.

이 변경은 Windows 워크로드의 포드 보안에 어떤 영향을 미칩니까?

Windows 워크로드에 대한 구체적인 영향은 예상되지 않습니다. PodSecurityContext에는 Winodws Pods용 PodSpec v1 API에 windowsOptions라는 필드가 있습니다. 이는 Kubernetes 1.25의 PSS를 사용합니다. Windows 워크로드용 PSS 적용에 대한 자세한 내용과 모범 사례는 EKS 모범 사례 가이드 및 Kubernetes 설명서를 참조하세요.

Kubernetes가 있는 AWS Secrets Manager 비밀 사용

Secrets Manager의 보안 암호 및 파라미터 스토어의 파라미터를 Amazon EKS Pods에 탑재된 파일로 표시하려면 Kubernetes 보안 정보 스토어 CSI 드라이버에 대한 AWS 보안 정보 및 구성 공급자(ASCP)를 사용할 수 있습니다.

ASCP를 사용하면 Secrets Manager 보안 암호를 저장 및 관리한 후 Amazon EKS에서 실행되는 워크로드를 통해 해당 검색할 수 있습니다. IAM 역할 및 정책을 사용하여 보안 암호에 대한 액세스 권한을 클러스터의 특정 Kubernetes Pods로 제한할 수 있습니다. ASCP는 Pod 자격 증명을 검색하고 IAM 역할에 대한 자격 증명을 교환합니다. ASCP는 Pod의 IAM 역할을 가정한 후 해당 역할에 대해 인증된 Secrets Manager에서 비밀을 검색할 수 있습니다.

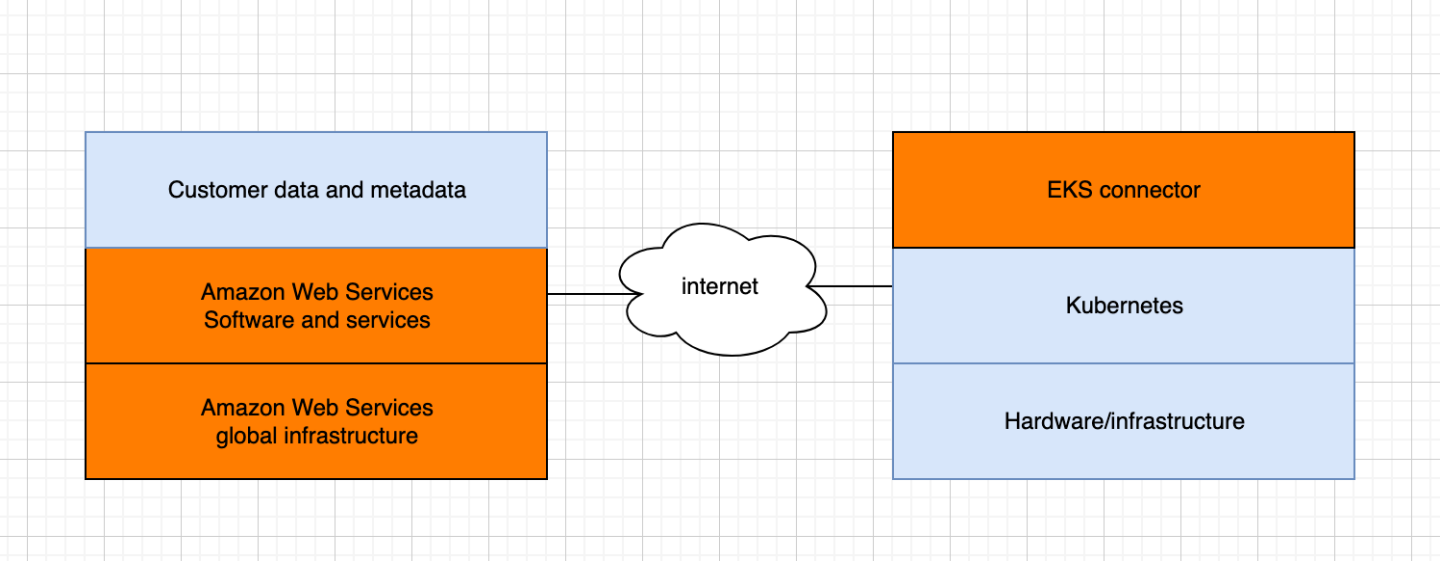
보안 암호에 대해 Secrets Manager 자동 교체를 사용하는 경우 보안 암호 스토어 CSI 드라이버 교체 조정자 기능을 사용하여 Secrets Manager에서 최신 보안 암호를 검색할 수도 있습니다.

자세한 내용은 AWS Secrets Manager 사용 설명서의 Amazon EKS에서 Secrets Manager 보안 암호 사용을 참조하세요.

Amazon EKS Connector 고려 사항

Amazon EKS Connector는 Kubernetes 클러스터에서 실행되는 오픈 소스 구성 요소입니다. 이 클러스터는 AWS 환경의 외부에 위치해 있을 수 있습니다. 이로 인해 보안 책임에 대한 추가 고려 사항이 생깁

니다. 다음 다이어그램에서 이 구성을 볼 수 있습니다. 오렌지색은 AWS 책임을 표시하며, 파란색은 고객 책임을 나타냅니다.



이 주제에서는 연결된 클러스터가 AWS의 외부에 있는 경우 책임 모델의 차이점에 대해 설명합니다.

AWS 책임

- 고객의 Kubernetes 클러스터에서 실행되고 AWS와 통신하는 오픈 소스 구성 요소인 Amazon EKS Connector의 유지 관리, 구축 및 전달.
- 연결된 Kubernetes 클러스터 및 AWS 서비스 간의 전송 및 애플리케이션 계층 통신 보안 유지 관리.

고객 책임

- 특히 다음 줄이 표시되는 Kubernetes 클러스터 특정 보안.
  - Kubernetes 보안 정보는 적절히 암호화되고 보호되어야 합니다.
  - `eks-connector` 네임스페이스에 대한 액세스 잠금.
- AWS에서 IAM 보안 주체를 관리하기 위한 역할 기반 액세스 제어(RBAC) 권한 구성. 지침은 클러스터에서 Kubernetes 리소스를 보도록 IAM 보안 주체에게 액세스 권한 부여 섹션을 참조하세요.
- Amazon EKS Connector 설치 및 업그레이드.
- 연결된 Kubernetes 클러스터를 지원하는 하드웨어, 소프트웨어 및 인프라 유지 관리.
- 해당 AWS 계정 보안(예: 루트 사용자 보안 인증 정보 보호를 통해).

Kubernetes 리소스 보기

AWS Management Console을 사용하면 클러스터에 배포된 Kubernetes 리소스를 볼 수 있습니다. AWS CLI 또는 `eksctl`로는 Kubernetes 리소스를 볼 수 없습니다. 명령줄 도구를 사용하여 Kubernetes 리소스를 보려면 `kubectl`을 사용합니다.

전제 조건

AWS Management Console의 컴퓨팅 탭에서 리소스 탭과 노드 섹션을 보려면 사용 중인 IAM 보안 주체에 특정 IAM 및 Kubernetes 권한이 있어야 합니다. 자세한 내용은 필요한 권한 단원을 참조하십시오.

AWS Management Console로 Kubernetes 리소스를 보려면 다음을 수행합니다.

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 클러스터(Clusters) 목록에서 보려는 Kubernetes 리소스를 포함하는 클러스터를 선택합니다.
3. 리소스(Resources) 탭을 선택합니다.
4. 보려는 리소스에 대한 리소스 유형(Resource type) 그룹을 선택합니다(예: 워크로드(Workloads). 해당 그룹의 리소스 유형 목록이 표시됩니다.
5. 워크로드(Workloads) 그룹에서 리소스 유형을 선택합니다(예: 배포(Deployments)). 리소스 유형 설명, 리소스 유형의 자세한 내용이 있는 Kubernetes 설명서에 대한 링크, 클러스터에 배포된 해당 유형의 리소스 목록이 표시됩니다. 목록이 비어 있는 경우 클러스터에 배포된 해당 유형의 리소스가 없습니다.
6. 이에 대한 자세한 내용을 확인하려면 리소스를 선택합니다. 다음 예를 시도해 보세요.

   - 워크로드(Workloads) 그룹을 선택하고, 배포(Deployments) 리소스 유형을 선택한 다음 coredns 리소스를 선택합니다. 리소스를 선택하면 기본적으로 구조적 뷰(Structured view)에 있습니다. 일부 리소스 유형의 경우 구조적 뷰(Structured view)의 포드(Pod) 섹션이 표시됩니다. 이 섹션에서는 워크로드에 의해 관리되는 Pods가 나열됩니다. Pod에 대한 정보를 보려면 나열된 모든 Pod를 선택할 수 있습니다. 일부 리소스 유형의 정보가 구조적 뷰(Structured View)에 표시됩니다. 리소스 페이지의 오른쪽 상단 모서리에서 원시 보기(Raw view)를 선택하는 경우 리소스에 대한 Kubernetes API의 전체 JSON 응답이 표시됩니다.
   - 클러스터(Cluster) 그룹을 선택한 다음 노드(Nodes) 리소스 유형을 선택합니다. 클러스터에 모든 노드 목록이 표시됩니다. 노드는 모든 Amazon EKS 노드 유형일 수 있습니다. 이 목록은 클러스터의 컴퓨팅(Compute) 탭을 선택하는 경우 노드(Nodes) 섹션에 표시되는 것과 동일한 목록입니다. 목록에서 노드 리소스를 선택합니다. 구조적 뷰(Structured view)에 포드(Pod) 섹션도 표시됩니다. 이 섹션에서는 노드에서 실행되는 모든 Pods를 보여줍니다.

필요한 권한

AWS Management Console의 컴퓨팅 탭에서 리소스 탭과 노드 섹션을 보려면 사용 중인 IAM 보안 주체에 최소한 특정 IAM 및 Kubernetes 권한이 있어야 합니다. IAM 보안 주체에 필요한 권한을 할당하려면 다음 단계를 완료하세요.

1. `eks:AccessKubernetesApi` 및 Kubernetes 리소스를 보는 데 필요한 기타 IAM 권한이 사용 중인 IAM 보안 주체에 할당되었는지 확인하십시오. IAM 보안 주체의 권한을 편집하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 보안 주체에 대한 액세스 제어 섹션을 참조하세요. 역할의 권한을 편집하는 방법에 대한 자세한 내용을 알아보려면 IAM 사용 설명서의 역할 권한 정책(콘솔) 수정을 참조하세요.

   다음 정책 예에는 계정의 모든 클러스터에 대한 Kubernetes 리소스를 보는 데 필요한 보안 주체의 권한이 포함되어 있습니다. *111122223333*을 AWS 계정 ID로 바꿉니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:ListFargateProfiles",
                "eks:DescribeNodegroup",
                "eks:ListNodegroups",
                "eks:ListUpdates",
                "eks:AccessKubernetesApi",
                "eks:ListAddons",
                "eks:DescribeCluster",
                "eks:DescribeAddonVersions",
                "eks:ListClusters",
                "eks:ListIdentityProviderConfigs",
                "iam:ListRoles"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:GetParameter",
            "Resource": "arn:aws:ssm:*:111122223333:parameter/*"
        }
    ]
```

```
}
```

연결된 클러스터에서 노드를 보려면 Amazon EKS 커넥터 IAM 역할이 클러스터에서 보안 주체를 가장할 수 있어야 합니다. 이렇게 하면 Amazon EKS Connector은(는) 보안 주체를 Kubernetes 사용자에게 매핑할 수 있습니다.

2. Kubernetes 리소스를 보는 데 필요한 권한이 있는 Kubernetes `role` 또는 `clusterrole`에 바인딩된 Kubernetes `rolebinding` 또는 `clusterrolebinding`을 만듭니다. Kubernetes 역할 및 역할 바인딩에 대한 자세한 내용을 알아보려면 Kubernetes 설명서의 RBAC 권한 부여 사용을 참조하세요. 필요한 Kubernetes 권한을 사용하여 `role` 및 `rolebinding` 또는 `clusterrole` 및 `clusterrolebinding`을 생성하는 클러스터에 다음 매니페스트 중 하나를 적용할 수 있습니다.

   모든 네임스페이스의 Kubernetes 리소스 보기

   파일의 그룹 이름은 `eks-console-dashboard-full-access-group`입니다. 다음 명령을 사용하여 클러스터에 매니페스트를 적용합니다.

   ```
   kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
   ```

   특정 네임스페이스의 Kubernetes 리소스 보기

   이 파일의 네임스페이스는 `default`입니다. 파일의 그룹 이름은 `eks-console-dashboard-restricted-access-group`입니다. 다음 명령을 사용하여 클러스터에 매니페스트를 적용합니다.

   ```
   kubectl apply -f https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-restricted-access.yaml
   ```

   Kubernetes 그룹 이름, 네임스페이스, 권한 또는 파일의 다른 모든 구성을 변경해야 하는 경우 파일을 다운로드하여 편집한 후 클러스터에 적용합니다.

   1. 다음 명령 중 하나를 사용하여 파일을 다운로드합니다.

      ```
      curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-full-access.yaml
      ```

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/docs/eks-console-
restricted-access.yaml
```

2. 필요에 따라 파일을 편집합니다.
3. 다음 명령 중 하나를 사용하여 클러스터에 매니페스트를 적용합니다.

```
kubectl apply -f eks-console-full-access.yaml
```

```
kubectl apply -f eks-console-restricted-access.yaml
```

3. IAM 보안 주체를 aws-auth ConfigMap의 Kubernetes 사용자 또는 그룹에 매핑합니다. eksctl과 같은 도구를 사용하여 ConfigMap을 업데이트하거나 편집하여 수동으로 업데이트할 수 있습니다.

> **Important**
>
> eksctl 또는 다른 도구를 사용하여 ConfigMap을 편집하는 것이 좋습니다. 사용할 수 있는 다른 도구에 대한 자세한 내용을 알아보려면 Amazon EKS 모범 사례 가이드의 도구를 사용하여 aws-authConfigMap을 변경을 참조하세요. 부적절하게 형식이 지정된 aws-auth ConfigMap으로 인해 클러스터에 대한 액세스 권한을 상실할 수 있습니다.

eksctl

전제 조건

디바이스 또는 0.175.0에 설치된 버전 AWS CloudShell 이상의 eksctl 명령줄 도구. eksctl을 설치 또는 업그레이드하려면 eksctl 설명서에서 Installation을 참조하세요.

1. ConfigMap에서 현재 매핑을 확인합니다. *my-cluster*를 클러스터 이름으로 바꿉니다. *region-code*를 클러스터가 있는 AWS 리전으로 바꿉니다.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

예제 출력은 다음과 같습니다.

```
ARN
                    USERNAME                                GROUPS
            ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
```

2. 역할에 대한 매핑을 추가합니다. 이 예에서는 첫 번째 단계의 IAM 권한을 *my-console-viewer-role*라는 역할에 연결했다고 가정합니다. *111122223333*을 계정 ID로 바꿉니다.

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:role/my-console-viewer-role \
    --group eks-console-dashboard-full-access-group \
    --no-duplicate-arns
```

> **Important**
>
> 역할 ARN에는 role/my-team/developers/my-role과 같은 경로가 포함될 수 없습니다. ARN 형식은 arn:aws:iam::*111122223333*:role/*my-role*이어야 합니다. 이 예에서는 my-team/developers/를 제거해야 합니다.

예제 출력은 다음과 같습니다.

```
[...]
2022-05-09 14:51:20 [#]  adding identity "arn:aws:iam::111122223333:role/my-
console-viewer-role" to auth ConfigMap
```

3. 사용자에 대한 매핑을 추가합니다. IAM 모범 사례에서는 되도록 사용자 대신 역할에 권한을 부여하세요. 이 예에서는 첫 번째 단계의 IAM 권한을 *my-user*라는 사용자에 연결했다고 가정합니다. *111122223333*을 계정 ID로 바꿉니다.

```
eksctl create iamidentitymapping \
    --cluster my-cluster \
    --region=region-code \
    --arn arn:aws:iam::111122223333:user/my-user \
    --group eks-console-dashboard-restricted-access-group \
```

```
    --no-duplicate-arns
```

예제 출력은 다음과 같습니다.

```
[...]
2022-05-09 14:53:48 [#]  adding identity "arn:aws:iam::111122223333:user/my-
user" to auth ConfigMap
```

4. ConfigMap에서 매핑을 다시 확인합니다.

```
eksctl get iamidentitymapping --cluster my-cluster --region=region-code
```

예제 출력은 다음과 같습니다.

```
ARN
                  USERNAME                                GROUPS
                    ACCOUNT
arn:aws:iam::111122223333:role/eksctl-my-cluster-my-nodegroup-
NodeInstanceRole-1XLS7754U3ZPA    system:node:{{EC2PrivateDNSName}}
 system:bootstrappers,system:nodes
arn:aws:iam::111122223333:role/my-console-viewer-role
                                                          eks-console-
dashboard-full-access-group
arn:aws:iam::111122223333:user/my-user
                                                          eks-console-
dashboard-restricted-access-group
```

Edit ConfigMap manually

aws-auth ConfigMap에 사용자 또는 역할을 추가하는 방법에 대한 자세한 내용은 Amazon EKS 클러스터에 IAM 보안 주체 추가 섹션을 참조하세요.

1. 편집을 위해 aws-auth ConfigMap을 엽니다.

```
kubectl edit -n kube-system configmap/aws-auth
```

2. aws-auth ConfigMap에 매핑을 추가하지만 모든 기존 매핑을 대체하지는 마세요. 다음 예에서는 첫 번째 단계에서 추가된 권한과 이전 단계에서 생성된 Kubernetes 그룹을 사용하여 IAM 보안 주체 간에 매핑을 추가합니다.

- *my-console-viewer-role* 역할 및 eks-console-dashboard-full-access-group.
- *my-user* 사용자 및 eks-console-dashboard-restricted-access-group.

이 예에서는 첫 번째 단계의 IAM 권한을 *my-console-viewer-role*이라는 역할 및 *my-user*라는 사용자에 연결했다고 가정합니다. *111122223333*을 AWS 계정 ID로 바꿉니다.

```
apiVersion: v1
data:
mapRoles: |
  - groups:
    - eks-console-dashboard-full-access-group
    rolearn: arn:aws:iam::111122223333:role/my-console-viewer-role
    username: my-console-viewer-role
mapUsers: |
  - groups:
    - eks-console-dashboard-restricted-access-group
    userarn: arn:aws:iam::111122223333:user/my-user
    username: my-user
```

> **Important**
>
> 역할 ARN에는 role/my-team/developers/my-console-viewer-role과 같은 경로가 포함될 수 없습니다. ARN 형식은 arn:aws:iam::*111122223333*:role/*my-console-viewer-role*이어야 합니다. 이 예에서는 my-team/developers/를 제거해야 합니다.

3. 파일을 저장하고 텍스트 편집기를 종료합니다.

Amazon EKS의 관찰 가능성

사용 가능한 많은 모니터링 또는 로깅 도구를 사용하여 Amazon EKS에서 데이터를 관찰할 수 있습니다. 데이터 분석을 위해 Amazon EKS 로그 데이터를 AWS 서비스 또는 파트너 도구로 스트리밍할 수 있습니다. Amazon EKS 문제 해결을 위한 데이터를 제공하는 AWS Management Console에서 사용할 수 있는 서비스가 많습니다.

Amazon EKS 콘솔의 왼쪽 탐색 창에서 클러스터를 선택한 후 클러스터 이름을 선택하여 클러스터 상태와 세부 정보를 볼 수 있습니다. 클러스터에 배포된 기존의 모든 Kubernetes 리소스에 대한 세부 정보를 보려면 Kubernetes 리소스 보기 섹션을 참조하세요.

모니터링은 Amazon EKS와 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다. AWS 솔루션의 모든 부분으로부터 모니터링 데이터를 수집하는 것이 좋습니다. 그렇게 하면 다중 지점 실패가 발생하는 경우 보다 쉽게 디버깅할 수 있습니다. Amazon EKS 모니터링을 시작하기 전에 모니터링 계획이 다음 문제를 해결할 수 있는지 확인합니다.

- 목표가 무엇인가요? 클러스터가 급격히 확장될 경우 실시간 알림이 필요하나요?
- 관찰해야 할 리소스는 무엇인가요?
- 이러한 리소스를 얼마나 자주 관찰해야 하나요? 귀사는 위험에 신속하게 대응하고 싶나요?
- 어떤 도구를 사용하고 싶나요? AWS Fargate를 시작의 일부로 이미 실행한 경우에는 기본 제공 로그 라우터를 사용할 수 있습니다.
- 누가 모니터링 작업을 수행하기를 바라나요?
- 문제가 발생하면 누구에게 알림을 보내고 싶나요?

Amazon EKS의 로깅 및 모니터링

Amazon EKS는 로깅 및 모니터링을 위한 기본 제공 도구를 제공합니다. 컨트롤 플레인 로깅은 클러스터에 대한 모든 API 호출, 사용자가 클러스터에 어떤 작업을 수행했는지 캡처하는 감사 정보 및 역할 기반 정보를 기록합니다. 자세한 내용은 AWS Prescriptive Guidance의 Logging and monitoring on Amazon EKS를 참조하세요.

Amazon EKS 컨트롤 플레인 로깅은 Amazon EKS 컨트롤 플레인에서 계정의 CloudWatch Logs로 감사 및 진단 로그를 직접 제공합니다. 이러한 로그를 통해 클러스터를 쉽게 보호하고 실행할 수 있습니다. 필요한 로그 유형을 정확하게 선택할 수 있으며, 로그는 CloudWatch의 각 Amazon EKS 클러스터에 대한 그룹에 로그 스트림으로 전송됩니다. 자세한 내용은 Amazon EKS 컨트롤 플레인 로깅 섹션을 참조하세요.

> **Note**
>
> Amazon CloudWatch에서 Amazon EKS 인증자 로그를 확인하면 다음 텍스트 예와 유사한 텍스트가 포함된 항목이 표시됩니다.
>
> ```
> level=info msg="mapping IAM role" groups="[]"
>  role="arn:aws:iam::111122223333:role/XXXXXXXXXXXXXXXXXX-
> NodeManagerRole-XXXXXXXX" username="eks:node-manager"
> ```
>
> 이 텍스트를 포함하는 항목이 필요합니다. `username`은 관리형 노드 그룹 및 Fargate에 대해 특정 작업을 수행하는 Amazon EKS 내부 서비스 역할입니다.
> 낮은 수준의 사용자 지정 가능한 로깅의 경우 Kubernetes 로깅을 사용할 수 있습니다.

Amazon EKS는 Amazon EKS에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 Amazon EKS에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 Amazon EKS 콘솔로부터의 호출과 Amazon EKS API 작업에 대한 코드 호출이 포함됩니다. 자세한 내용은 AWS CloudTrail을 사용하여 Amazon EKS API 호출 로깅 섹션을 참조하세요.

Kubernetes API 서버는 모니터링 및 분석에 유용한 많은 지표를 표시합니다. 자세한 내용은 Prometheus 지표 섹션을 참조하세요.

사용자 지정 Amazon CloudWatch 로그에 대해 Fluent Bit를 구성하려면 Amazon CloudWatch 사용 설명서의 Fluent Bit 설정을 참조하세요.

Amazon EKS의 로깅 및 모니터링 도구

Amazon Web Services는 Amazon EKS를 모니터링하는 데 사용할 수 있는 다양한 도구를 제공합니다. 일부 도구를 구성하여 자동 모니터링을 설정할 수 있지만 일부는 수동 호출이 필요합니다. 사용 중인 환경 및 기존 도구 집합이 허용하는 만큼 모니터링 작업을 자동화하는 것이 좋습니다.

로깅 도구

영역	도구	설명	설치
애플리케이션	Amazon CloudWatc	컨테이너식 애플리케이션 및 마이크로서비스의 지	설정 절차

영역	도구	설명	설치
	h Container Insights	표 및 로그를 수집하고 종합하며 요약합니다.	
컨트롤 플레인	AWS CloudTrail	사용자, 역할 또는 서비스를 통한 API 호출을 기록합니다.	설정 절차
AWS Fargate 인스턴스의 여러 영역	AWS Fargate 로그 라우터	AWS Fargate의 경우 AWS 서비스 또는 파트너 도구로 로그를 스트리밍합니다. Fluent Bit용 AWS를 사용합니다. 로그를 다른 AWS 서비스 또는 파트너 도구로 스트리밍할 수 있습니다.	설정 절차

모니터링 도구

영역	도구	설명	설치
애플리케이션	CloudWatch Container Insights	CloudWatch Container Insights는 컨테이너 애플리케이션 및 마이크로서비스의 지표 및 로그를 수집하고 종합하며 요약합니다.	설정 절차

영역	도구	설명	설치
애플리케이션	AWS Distro for OpenTelemetry (ADOT)	상호 연관된 지표, 추적 데이터 및 메타데이터를 수집해서 AWS 모니터링 서비스 또는 파트너로 전송합니다. CloudWatch Container Insights를 통해 설정할 수 있습니다.	설정 절차
애플리케이션	Amazon DevOps Guru	노드 수준의 운영 성능 및 가용성을 감지합니다.	설정 절차
애플리케이션	AWS X-Ray	애플리케이션에 대한 추적 데이터를 수신합니다. 이 추적 데이터에는 수신 및 발신 요청과 요청에 대한 메타데이터가 포함됩니다. Amazon EKS의 경우 구현하려면 OpenTelemetry 추가 기능이 필요합니다.	설정 절차

영역	도구	설명	설치
애플리케이션	Amazon CloudWatch Observability Operator	Amazon CloudWatch Observability Operator는 지표, 로그 및 추적 데이터를 수집합니다. 이를 Amazon CloudWatch 및 AWS X-Ray로 보냅니다.	설정 절차
컨트롤 플레인	Prometheus	CloudWatch Logs 수집, 아카이브 스토리지 및 데이터 검색 요율이 활성화된 컨트롤 플레인 로그에 적용됩니다.	설정 절차

Prometheus 지표

Prometheus는 엔드포인트를 스크레이프하는 모니터링 및 시계열 데이터베이스입니다. 수집된 데이터를 쿼리, 집계 및 저장하는 기능을 제공합니다. 알림 및 알림 집계에도 사용할 수 있습니다. 이 주제에서는 관리형 또는 오픈 소스 옵션으로 Prometheus를 설정하는 방법을 설명합니다. Amazon EKS 컨트롤 플레인 지표 모니터링은 일반적인 사용 사례입니다.

Amazon Managed Service for Prometheus는 컨테이너화된 애플리케이션과 인프라를 대규모로 쉽게 모니터링할 수 있는 Prometheus 호환 모니터링 및 알림 서비스입니다. 지표의 수집, 스토리지, 쿼리 및 알림을 자동으로 확장하는 완전관리형 서비스입니다. 또한 AWS 보안 서비스와 통합되어 데이터에 빠르고 안전하게 액세스할 수 있습니다. 오픈 소스 PromQL 쿼리 언어를 사용하여 지표를 쿼리하고 지표에 대해 알릴 수 있습니다.

활성화 후 Prometheus 지표를 사용하는 방법에 대한 자세한 내용은 Amazon Managed Service for Prometheus 사용 설명서를 참조하세요.

클러스터를 생성할 때 Prometheus 지표 켜기

> **Important**
>
> Amazon Managed Service for Prometheus 리소스는 클러스터 수명 주기를 벗어나며 클러스터에 독립적으로 유지 관리되어야 합니다. 클러스터를 삭제할 때 해당 스크레이퍼도 모두 삭제하여 관련 비용을 줄여야 합니다. 자세한 내용은 Amazon Managed Service for Prometheus 사용자 가이드의 스크레이퍼 찾기 및 삭제를 참조하세요.

새 클러스터를 생성할 때 지표를 Prometheus로 보내는 옵션을 켤 수 있습니다. AWS Management Console에서 이 옵션은 새 클러스터 생성의 관찰성 구성 단계에 있습니다. 자세한 내용은 Amazon EKS 클러스터 생성 단원을 참조하십시오.

Prometheus는 스크레이핑이라는 풀 기반 모델을 통해 클러스터에서 지표를 검색하고 수집합니다. 스크레이퍼는 클러스터 인프라 및 컨테이너화된 애플리케이션에서 데이터를 수집하도록 설정됩니다.

Prometheus 지표 전송 옵션을 켜면 Amazon Managed Service for Prometheus에서 에이전트 없는 완전 관리형 스크레이퍼를 제공합니다. 다음 고급 구성 옵션을 사용하여 필요에 따라 기본 스크레이퍼를 사용자 지정합니다.

스크레이퍼 별칭

(선택 사항) 스크레이퍼의 고유한 별칭을 입력합니다.

대상

Amazon Managed Service for Prometheus 작업 영역을 선택합니다. 작업 영역은 Prometheus 지표 보관 및 쿼리를 위한 전용 논리 공간입니다. 이 작업 영역을 사용하면 액세스 권한이 있는 계정에서 Prometheus 지표를 볼 수 있습니다. 새 작업 영역 생성 옵션은 Amazon EKS에 제공한 작업 영역 별칭을 사용하여 대신 작업 영역을 생성하도록 지시합니다. 기존 작업 영역 선택 옵션을 사용하면 드롭다운 목록에서 기존 작업 영역을 선택할 수 있습니다. 작업 영역에 대한 자세한 내용은 Amazon Managed Service for Prometheus 사용 설명서의 작업 영역 관리를 참조하세요.

서비스 액세스

이 섹션에는 Prometheus 지표를 전송할 때 부여하는 권한이 요약되어 있습니다.

- Amazon Managed Service for Prometheus에 스크레이프된 Amazon EKS 클러스터 설명 허용
- Amazon Managed Prometheus 작업 영역에 원격 쓰기 허용

`AmazonManagedScraperRole`이 이미 있는 경우 스크레이퍼가 이를 사용합니다. `AmazonManagedScraperRole` 링크를 선택하여 권한 세부 정보를 확인합니다. `AmazonManagedScraperRole`이 아직 없는 경우 권한 세부 정보 보기 링크를 선택하여 Prometheus 지표를 전송하여 부여하는 특정 권한을 확인합니다.

서브넷

스크레이퍼가 상속할 서브넷을 확인합니다. 변경해야 하는 경우 클러스터 생성 네트워킹 지정 단계로 돌아갑니다.

보안 그룹

스크레이퍼가 상속할 보안 그룹을 확인합니다. 변경해야 하는 경우 클러스터 생성 네트워킹 지정 단계로 돌아갑니다.

스크레이퍼 구성

필요에 따라 YAML 형식으로 스크레이퍼 구성을 수정합니다. 이렇게 하려면 양식을 사용하거나 대체 YAML 파일을 업로드합니다. 자세한 내용은 Amazon Managed Service for Prometheus 사용 설명서의 스크레이퍼 구성을 참조하세요.

Amazon Managed Service for Prometheus는 클러스터와 함께 AWS 관리형 컬렉터로 생성되는 에이전트 없는 스크레이퍼를 말합니다. AWS 관리형 컬렉터에 대한 자세한 내용은 Amazon Managed Service for Prometheus 사용 설명서의 AWS 관리형 컬렉터를 참조하세요.

> **Important**
>
> 스크레이퍼에 클러스터 내 권한을 부여하려면 `aws-auth ConfigMap`을 설정해야 합니다. 자세한 내용은 Amazon Managed Service for Prometheus 사용 설명서의 Amazon EKS 클러스터 구성을 참조하세요.

Prometheus 스크레이퍼 세부 정보 보기

Prometheus 지표 옵션이 켜진 상태에서 클러스터를 생성한 후 Prometheus 스크레이퍼 세부 정보를 볼 수 있습니다. AWS Management Console에서 클러스터를 볼 때 관찰성 탭을 선택합니다. 테이블에는 스크레이퍼 ID, 별칭, 상태 및 생성 날짜와 같은 정보를 포함하여 클러스터의 스크레이퍼 목록이 표시됩니다.

스크레이퍼에 관한 추가 세부 정보를 확인하려면 스크레이퍼 ID 링크를 선택합니다. 예를 들어 스크레이퍼 구성, Amazon 리소스 이름(ARN), 원격 쓰기 URL 및 네트워킹 정보를 볼 수 있습니다. 스크레이

퍼 ID를 `DescribeScraper` 및 `DeleteScraper`와 같은 Amazon Managed Service for Prometheus API 작업의 입력으로 사용할 수 있습니다. API를 사용하여 더 많은 스크레이퍼를 생성할 수도 있습니다.

Prometheus API 사용에 대한 자세한 내용은 Amazon Managed Service for Prometheus API 참조를 참조하세요.

Helm 사용을 통한 Prometheus 배포

또는 Helm V3를 사용하여 클러스터에 Prometheus를 배포할 수도 있습니다. 이미 Helm이 설치되어 있는 경우 `helm version` 명령을 사용하여 버전을 확인할 수 있습니다. Helm은 Kubernetes 클러스터에 대한 패키지 관리자입니다. Helm 및 설치 방법에 대한 자세한 내용은 Amazon EKS에 Helm 사용 섹션을 참조하세요.

Amazon EKS 클러스터에 대해 Helm을 구성하면 이를 사용하여 다음과 같은 단계로 Prometheus를 배포할 수 있습니다.

Helm 사용을 통해 Prometheus 배포

1. Prometheus 네임스페이스를 생성합니다.

```
kubectl create namespace prometheus
```

2. `prometheus-community` 차트 리포지토리를 추가합니다

```
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
```

3. Prometheus를 배포합니다.

```
helm upgrade -i prometheus prometheus-community/prometheus \
    --namespace prometheus \
    --set
 alertmanager.persistentVolume.storageClass="gp2",server.persistentVolume.storageClass="gp2
```

> **Note**
>
> 이 명령을 실행할 때 `Error: failed to download "stable/prometheus" (hint: running `helm repo update` may help)` 오류가 발생

하면 `helm repo update prometheus-community`를 실행한 다음 2단계 명령을 다시 실행해봅니다.

`Error: rendered manifests contain a resource that already exists` 오류가 발생하면 `helm uninstall your-release-name -n namespace`를 실행한 다음 3단계 명령을 다시 실행해봅니다.

4. `prometheus` 네임스페이스의 모든 Pods가 `READY` 상태인지 확인합니다.

```
kubectl get pods -n prometheus
```

예제 출력은 다음과 같습니다.

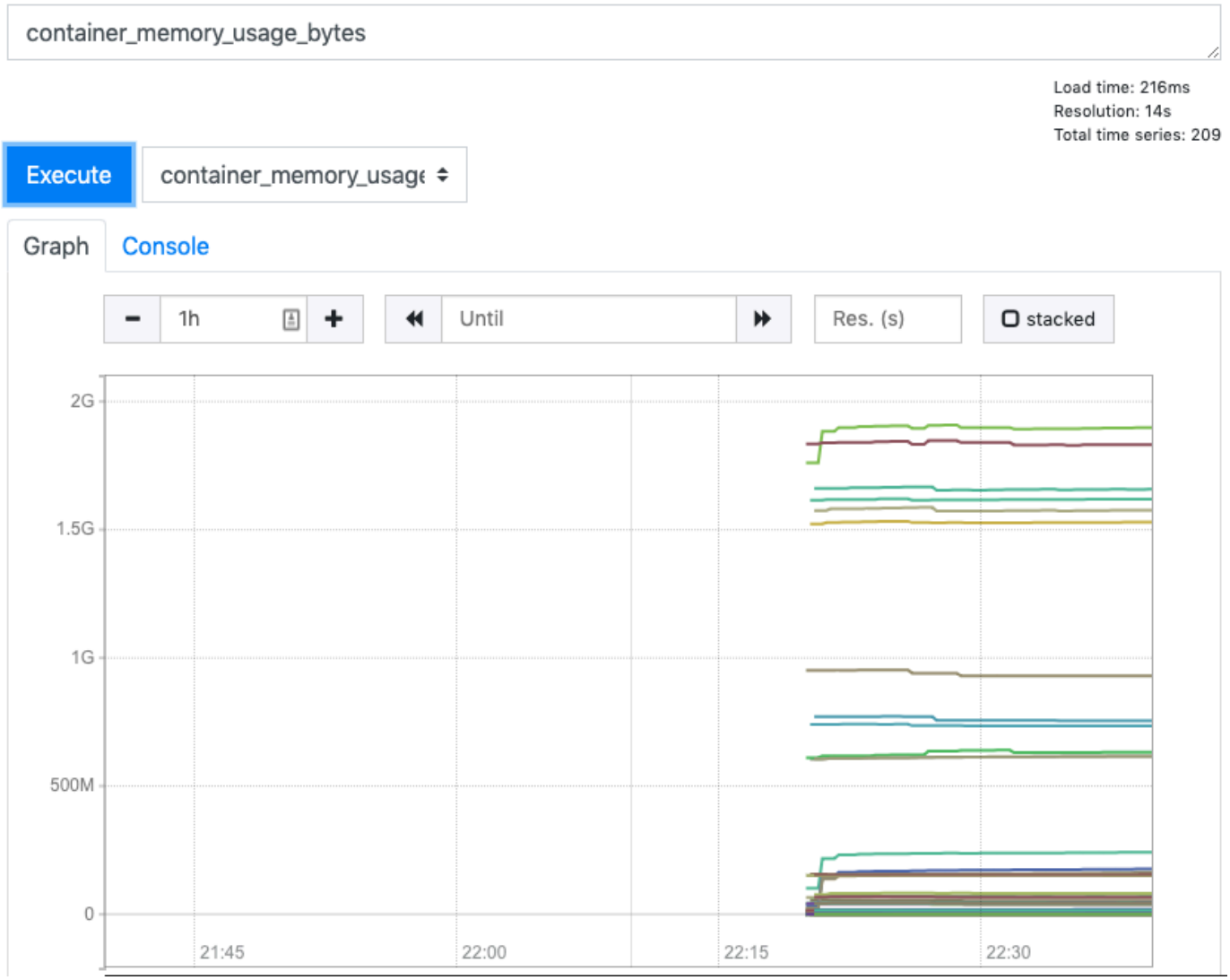
```
NAME                                             READY   STATUS    RESTARTS   AGE
prometheus-alertmanager-59b4c8c744-r7bgp         1/2     Running   0          48s
prometheus-kube-state-metrics-7cfd87cf99-jkz2f   1/1     Running   0          48s
prometheus-node-exporter-jcjqz                   1/1     Running   0          48s
prometheus-node-exporter-jxv2h                   1/1     Running   0          48s
prometheus-node-exporter-vbdks                   1/1     Running   0          48s
prometheus-pushgateway-76c444b68c-82tnw          1/1     Running   0          48s
prometheus-server-775957f748-mmht9               1/2     Running   0          48s
```

5. `kubectl`을 사용하여 포트를 통해 로컬 시스템에 Prometheus 콘솔을 전송합니다.

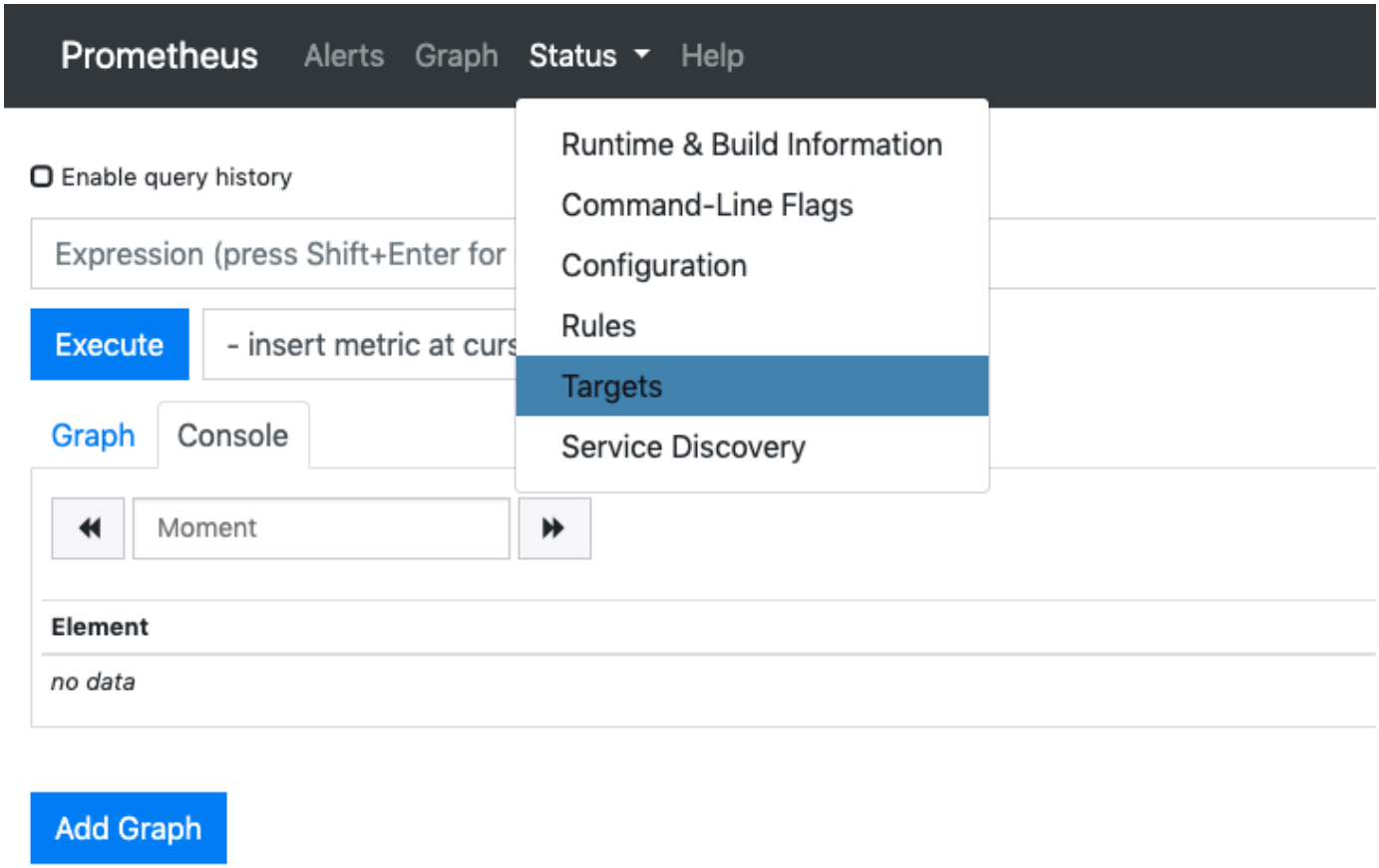
```
kubectl --namespace=prometheus port-forward deploy/prometheus-server 9090
```

6. 웹 브라우저에서 `http://localhost:9090`으로 이동하여 Prometheus 콘솔을 표시합니다.

7. - insert metric at cursor(커서에 지표 삽입) 메뉴에서 지표를 선택한 후 Execute(실행)를 선택합니다. Graph(그래프) 탭을 선택하여 시간에 따른 지표를 표시합니다. 다음 이미지는 시간에 따른 `container_memory_usage_bytes`를 표시합니다.



8. 상단 탐색 모음에서 Status(상태)를 선택한 후 Targets(대상)를 선택합니다.



서비스 검색을 사용하여 Prometheus에 연결된 모든 Kubernetes 엔드포인트가 표시됩니다.

컨트롤 플레인 원시 지표 보기

Prometheus 배포의 대안으로 Kubernetes API 서버는 Prometheus 형식으로 표시되는 여러 지표를 노출합니다. 이러한 지표는 모니터링 및 분석에 유용합니다. `/metrics` HTTP API를 나타내는 지표 엔드포인트를 통해 내부적으로 표시됩니다. 다른 엔드포인트와 마찬가지로 이 엔드포인트도 Amazon EKS 제어 영역에 표시됩니다. 이 엔드포인트는 주로 특정 지표를 살펴보는 데 유용합니다. 시간 경과에 따른 지표를 분석하려면 Prometheus 배포가 권장됩니다.

원시 지표 출력을 보려면 `--raw` 플래그와 함께 `kubectl`을 사용합니다. 이 명령을 사용하면 HTTP 경로를 전달할 수 있으며, 원시 응답이 반환됩니다.

```
kubectl get --raw /metrics
```

예제 출력은 다음과 같습니다.

```
[...]
# HELP rest_client_requests_total Number of HTTP requests, partitioned by status code,
 method, and host.
# TYPE rest_client_requests_total counter
rest_client_requests_total{code="200",host="127.0.0.1:21362",method="POST"} 4994
rest_client_requests_total{code="200",host="127.0.0.1:443",method="DELETE"} 1
rest_client_requests_total{code="200",host="127.0.0.1:443",method="GET"} 1.326086e+06
rest_client_requests_total{code="200",host="127.0.0.1:443",method="PUT"} 862173
rest_client_requests_total{code="404",host="127.0.0.1:443",method="GET"} 2
rest_client_requests_total{code="409",host="127.0.0.1:443",method="POST"} 3
rest_client_requests_total{code="409",host="127.0.0.1:443",method="PUT"} 8
# HELP ssh_tunnel_open_count Counter of ssh tunnel total open attempts
# TYPE ssh_tunnel_open_count counter
ssh_tunnel_open_count 0
# HELP ssh_tunnel_open_fail_count Counter of ssh tunnel failed open attempts
# TYPE ssh_tunnel_open_fail_count counter
ssh_tunnel_open_fail_count 0
```

이 원시 출력은 API 서버가 표시하는 축자를 반환합니다. 다양한 지표가 줄별로 나열되며 각 줄에는 지표 이름, 태그 및 값이 포함됩니다.

```
metric_name{"tag"="value"[,...]}
            value
```

Amazon CloudWatch에 대한 Amazon EKS 추가 기능 지원

Amazon CloudWatch Observability는 실시간 로그, 지표 및 추적 데이터를 수집합니다. Amazon CloudWatch 및 AWS X-Ray로 보냅니다. 이 추가 기능을 설치하여 Amazon EKS의 향상된 관찰성을 통해 CloudWatch Application Signals와 CloudWatch Container Insights를 모두 활성화할 수 있습니다. 이를 통해 인프라 및 컨테이너화된 애플리케이션의 상태와 성능을 모니터링할 수 있습니다. Amazon CloudWatch Observability Operator는 필요한 구성 요소를 설치하고 구성하도록 설계되었습니다.

Amazon EKS는 Amazon EKS 추가 기능으로 Amazon CloudWatch Observability Operator를 지원합니다. 이 추가 기능을 통해 클러스터의 Linux 및 Windows 워커 노드 모두에서 Container Insights를 사용할 수 있습니다. Windows에서 Container Insights를 활성화하려면 Amazon EKS 추가 기능 버전이 `1.5.0` 이상이어야 합니다. 현재 Amazon EKS Windows에서는 CloudWatch Application Signals가 지원되지 않습니다.

아래 주제에서는 Amazon EKS 클러스터에서 Amazon CloudWatch Observability Operator 사용을 시작하는 방법을 설명합니다.

- 이 추가 기능 설치 지침은 Amazon CloudWatch 사용 설명서의 CloudWatch Observability Amazon EKS 추가 기능을 사용하여 CloudWatch 에이전트 설치를 참조하세요.
- CloudWatch Application Signals에 대한 자세한 내용은 Application Signals를 참조하세요.
- Container Insights에 대한 자세한 정보는 Amazon CloudWatch 사용 설명서의 Container Insights 사용을 참조하세요.

Amazon EKS 컨트롤 플레인 로깅

Amazon EKS 컨트롤 플레인 로깅은 Amazon EKS 컨트롤 플레인에서 계정의 CloudWatch Logs로 감사 및 진단 로그를 직접 제공합니다. 이러한 로그를 통해 클러스터를 쉽게 보호하고 실행할 수 있습니다. 필요한 로그 유형을 정확하게 선택할 수 있으며, 로그는 CloudWatch의 각 Amazon EKS 클러스터에 대한 그룹에 로그 스트림으로 전송됩니다. 자세한 내용은 Amazon CloudWatch 로깅을 참조하세요.

각각의 새로운 또는 기존 Amazon EKS 클러스터에 대해 활성화하려는 로그 유형을 선택하여 Amazon EKS 컨트롤 플레인 로깅의 사용을 시작할 수 있습니다. AWS Management Console, AWS CLI(버전 `1.16.139` 이상)를 사용하거나 Amazon EKS API를 통해 클러스터 단위로 각 로그 유형을 활성화하거나 비활성화할 수 있습니다. 활성화하면 Amazon EKS 클러스터에서 동일한 계정의 CloudWatch Logs로 로그가 자동으로 전송됩니다.

Amazon EKS 컨트롤 플레인 로깅을 사용하면 실행하는 각 클러스터에 대해 표준 Amazon EKS 요금이 부과됩니다. 클러스터에서 CloudWatch Logs로 전송한 모든 로그에 대해 표준 CloudWatch Logs 데이터 수집 및 스토리지 비용이 부과됩니다. 또한 클러스터의 일부로 프로비저닝한 AWS 리소스(예: Amazon EC2 인스턴스 또는 Amazon EBS 볼륨)에 대해서도 요금이 부과됩니다.

사용할 수 있는 클러스터 컨트롤 플레인 로그 유형은 다음과 같습니다. 각 로그 유형은 Kubernetes 컨트롤 플레인의 구성 요소에 해당합니다. 이러한 구성 요소에 대한 자세한 내용은 Kubernetes 설명서의 Kubernetes 구성 요소를 참조하세요.

API 서버(**`api`**)

클러스터의 API 서버는 Kubernetes API가 표시되는 컨트롤 플레인 구성 요소입니다. 클러스터를 시작할 때 또는 직후에 API 서버 로그를 활성화하면 API 서버를 시작하는 데 사용된 API 서버 플래그가 로그에 포함됩니다. 자세한 내용은 Kubernetes 설명서의 `kube-apiserver` 및 감사 정책 부분을 참조하세요.

감사(**`audit`**)

Kubernetes 감사 로그는 클러스터에 영향을 준 개별 사용자, 관리자 또는 시스템 구성 요소에 대한 기록을 제공합니다. 자세한 내용은 Kubernetes 설명서의 감사를 참조하세요.

인증자(**authenticator**)

인증자 로그는 Amazon EKS에 고유합니다. 이러한 로그는 Amazon EKS가 IAM 자격 증명을 사용한 Kubernetes 역할 기반 액세스 컨트롤(RBAC) 인증에 사용하는 컨트롤 플레인 구성 요소를 나타냅니다. 자세한 내용은 클러스터 관리 섹션을 참조하세요.

컨트롤러 관리자(**controllerManager**)

컨트롤러 관리자는 Kubernetes와 함께 제공되는 핵심 컨트롤 루프를 관리합니다. 자세한 내용은 Kubernetes 설명서의 kube-controller-manager를 참조하세요.

스케줄러(**scheduler**)

스케줄러 구성 요소는 클러스터에서 Pods를 실행하는 시기와 위치를 관리합니다. 자세한 내용은 Kubernetes 설명서의 kube-scheduler를 참조하세요.

컨트롤 플레인 로그 활성화 및 비활성화

기본적으로 클러스터 컨트롤 플레인 로그는 CloudWatch Logs로 전송되지 않습니다. 클러스터에 대해 로그를 전송하려면 각 로그 유형을 개별적으로 활성화해야 합니다. CloudWatch Logs 수집, 아카이브 스토리지 및 데이터 검색 요율이 활성화된 컨트롤 플레인 로그에 적용됩니다. 자세한 내용은 CloudWatch 요금을 참조하십시오.

컨트롤 플레인 로깅 구성을 업데이트하려면 Amazon EKS에서 각 서브넷에 최대 5개의 사용 가능한 IP 주소가 필요합니다. 로그 유형을 활성화하면 로그가 로그 세부 수준 2로 전송됩니다.

AWS Management Console

AWS Management Console를 통해 컨트롤 플레인 로그를 활성화 또는 비활성화하려면

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 클러스터 이름을 선택하여 클러스터 정보를 표시합니다.
3. 관찰성 탭을 선택합니다.
4. 컨트롤 플레인 로깅 섹션에서 로깅 관리를 선택합니다.
5. 각 개별 로그 유형에 대해 로그 유형이 켜져 있거나 꺼져 있어야 하는지를 선택합니다. 기본적으로 각 로그 유형은 해제되어 있습니다.
6. 변경 사항 저장(Save changes)을 선택하여 종료합니다.

AWS CLI

AWS CLI를 통해 컨트롤 플레인 로그를 활성화 또는 비활성화하려면

1. AWS CLI 버전은 다음 명령을 통해 확인할 수 있습니다.

```
aws --version
```

AWS CLI 버전이 `1.16.139` 이하인 경우 먼저 최신 버전으로 업데이트해야 합니다. AWS CLI를 설치 또는 업그레이드하려면 AWS Command Line Interface 사용 설명서에서 AWS Command Line Interface 설치를 참조하세요.

2. 다음 AWS CLI 명령을 사용하여 클러스터의 컨트롤 플레인 로그 내보내기 구성을 업데이트합니다. *my-cluster*를 클러스터 이름으로 바꾸고 원하는 엔드포인트 액세스 값을 지정합니다.

> **Note**
>
> 다음 명령은 사용 가능한 모든 로그 유형을 CloudWatch Logs로 전송합니다.

```
aws eks update-cluster-config \
    --region region-code \
    --name my-cluster \
    --logging '{"clusterLogging":[{"types":
["api","audit","authenticator","controllerManager","scheduler"],"enabled":true}]}'
```

예제 출력은 다음과 같습니다.

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "InProgress",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
```

```
        "errors": []
    }
}
```

3. 이전 명령에서 반환된 클러스터 이름과 업데이트 ID를 사용하여 다음 명령으로 로그 구성 업데이트의 상태를 모니터링합니다. 상태가 Successful로 표시되면 업데이트가 완료된 것입니다.

```
aws eks describe-update \
    --region region-code\
    --name my-cluster \
    --update-id 883405c8-65c6-4758-8cee-2a7c1340a6d9
```

예제 출력은 다음과 같습니다.

```
{
    "update": {
        "id": "883405c8-65c6-4758-8cee-2a7c1340a6d9",
        "status": "Successful",
        "type": "LoggingUpdate",
        "params": [
            {
                "type": "ClusterLogging",
                "value": "{\"clusterLogging\":[{\"types\":[\"api\",\"audit\",
\"authenticator\",\"controllerManager\",\"scheduler\"],\"enabled\":true}]}"
            }
        ],
        "createdAt": 1553271814.684,
        "errors": []
    }
}
```

클러스터 컨트롤 플레인 로그 보기

Amazon EKS 클러스터에 대한 컨트롤 플레인 로그 유형을 활성화하면 CloudWatch 콘솔에서 해당 유형을 볼 수 있습니다.

CloudWatch에서 로그 보기, 분석 및 관리에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하세요.

CloudWatch 콘솔에서 클러스터 컨트롤 플레인 로그를 보려면

1. CloudWatch 콘솔을 엽니다. 이 링크는 콘솔을 열고 현재 사용 가능한 로그 그룹을 표시하며 `/aws/eks` 접두사로 해당 로그 그룹을 필터링합니다.
2. 로그를 보려는 클러스터를 선택합니다. 로그 그룹 이름 형식은 `/aws/eks/`*`my-cluster`*`/cluster`입니다.
3. 보려는 로그 스트림을 선택합니다. 다음 목록에는 각 로그 유형의 로그 스트림 이름 형식에 대한 설명이 나와 있습니다.

   > **Note**
   >
   > 로그 스트림 데이터가 증가함에 따라 로그 스트림 이름이 교체됩니다. 특정 로그 유형에 대해 여러 로그 스트림이 있는 경우 최신 Last Event Time(마지막 이벤트 시간)으로 로그 스트림 이름을 찾아 최신 로그 스트림을 볼 수 있습니다.

   - Kubernetes API 서버 구성 요소 로그(**api**) – `kube-apiserver-`*`1234567890abcdef01234567890abcde`*
   - 감사(**audit**) - `kube-apiserver-audit-`*`1234567890abcdef01234567890abcde`*
   - 인증자(**authenticator**) - `authenticator-`*`1234567890abcdef01234567890abcde`*
   - 컨트롤러 관리자(**controllerManager**) - `kube-controller-manager-`*`1234567890abcdef01234567890abcde`*
   - 스케줄러(**scheduler**) - `kube-scheduler-`*`1234567890abcdef01234567890abcde`*
4. 로그 스트림의 이벤트를 살펴봅니다.

   예를 들어 상위 `kube-apiserver-`*`1234567890abcdef01234567890abcde`*를 볼 때 클러스터의 초기 API 서버 플래그가 표시되어야 합니다.

   > **Note**
   >
   > 로그 스트림의 시작 부분에 API 서버 로그가 표시되지 않으면 서버에서 API 서버 로깅을 활성화하기 전에 서버에서 API 서버 로그 파일이 교체되었을 수 있습니다. API 서버 로깅이 활성화되기 전에 교체되는 모든 로그 파일은 CloudWatch로 내보낼 수 없습니다. 그러나 동일한 Kubernetes 버전으로 새 클러스터를 생성하고 클러스터를 생성할 때 API 서버 로깅을 활성화할 수 있습니다. 동일한 플랫폼 버전의 클러스터에는 동일한 플래그가

활성화되어 있으므로 플래그가 새 클러스터의 플래그와 일치해야 합니다. CloudWatch에서 새 클러스터에 대한 플래그 보기를 마치면 새 클러스터를 삭제할 수 있습니다.

AWS CloudTrail을 사용하여 Amazon EKS API 호출 로깅

Amazon EKS는 AWS CloudTrail에 통합됩니다. CloudTrail은 Amazon EKS에서 사용자, 역할 또는 AWS 서비스가 수행한 작업의 레코드를 제공하는 서비스입니다. CloudTrail은 Amazon EKS에 대한 모든 API 호출을 이벤트로 캡처합니다. 여기에는 Amazon EKS 콘솔 호출과 Amazon EKS API 작업에 대한 코드 호출이 포함됩니다.

추적을 생성하면 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 여기에는 Amazon EKS 이벤트가 포함됩니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집하는 정보를 사용하여 요청에 대한 몇 가지 세부 정보를 확인할 수 있습니다. 예를 들어 언제 Amazon EKS에 요청했는지, 어떤 IP 주소에서 요청했는지 그리고 누가 요청했는지를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

주제

- CloudTrail의 Amazon EKS 정보
- Amazon EKS 로그 파일 항목 이해
- 오토 스케일링 지표 수집 활성화

CloudTrail의 Amazon EKS 정보

CloudTrail은 AWS 계정 생성 시 AWS 계정에서 사용 설정합니다. Amazon EKS에서 어떤 활동이 발생하는 경우 해당 활동이 이벤트 기록(Event history)의 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.

Amazon EKS에 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 AWS 리전의 이벤트를 로깅하고 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 또는 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 정보는 다음 리소스를 참조하세요.

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

모든 Amazon EKS 작업은 CloudTrail에서 로깅되며 Amazon EKS API 참조에 설명되어 있습니다. 예를 들어, `CreateCluster`, `ListClusters` 및 `DeleteCluster`을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 또는 로그 항목에는 요청이 이루어지고 자격 증명이 사용되는 IAM 자격 증명 유형에 관한 정보가 포함됩니다. 임시 자격 증명이 사용되는 경우 요소는 자격 증명을 획득하는 방법을 보여 줍니다.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

Amazon EKS 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 소스로부터의 단일 요청을 나타내며 요청 작업에 대한 정보가 들어 있습니다. 여기에는 작업 날짜와 시간, 사용된 요청 파라미터에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음은 `CreateCluster` 작업을 보여 주는 CloudTrail 로그 항목이 나타낸 예제입니다.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/username",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "username"
  },
  "eventTime": "2018-05-28T19:16:43Z",
  "eventSource": "eks.amazonaws.com",
  "eventName": "CreateCluster",
  "awsRegion": "region-code",
  "sourceIPAddress": "205.251.233.178",
```

```
  "userAgent": "PostmanRuntime/6.4.0",
  "requestParameters": {
    "resourcesVpcConfig": {
      "subnetIds": [
        "subnet-a670c2df",
        "subnet-4f8c5004"
      ]
    },
    "roleArn": "arn:aws:iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
    "clusterName": "test"
  },
  "responseElements": {
    "cluster": {
      "clusterName": "test",
      "status": "CREATING",
      "createdAt": 1527535003.208,
      "certificateAuthority": {},
      "arn": "arn:aws:eks:region-code:111122223333:cluster/test",
      "roleArn": "arn:aws:iam::111122223333:role/AWSServiceRoleForAmazonEKS-
CAC1G1VH3ZKZ",
      "version": "1.10",
      "resourcesVpcConfig": {
        "securityGroupIds": [],
        "vpcId": "vpc-21277358",
        "subnetIds": [
          "subnet-a670c2df",
          "subnet-4f8c5004"
        ]
      }
    }
  },
  "requestID": "a7a0735d-62ab-11e8-9f79-81ce5b2b7d37",
  "eventID": "eab22523-174a-499c-9dd6-91e7be3ff8e3",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```


Amazon EKS 서비스 연결 역할에 대한 로그 항목

Amazon EKS 서비스 연결 역할은 AWS 리소스에 대한 API 호출을 수행합니다. Amazon EKS 서비스 연결 역할에서 수행한 호출의 경우 CloudTrail 로

그 항목에 username: AWSServiceRoleForAmazonEKS 및 username: AWSServiceRoleForAmazonEKSNodegroup이 표시됩니다. Amazon EKS 및 서비스 연결 역할에 대한 자세한 내용은 Amazon EKS에 대해 서비스 연결 역할 사용 섹션을 참조하세요.

다음 예에서는 sessionContext에 표시된 AWSServiceRoleForAmazonEKSNodegroup 서비스 연결 역할에서 생성된 DeleteInstanceProfile 작업을 보여 주는 CloudTrail 로그 항목을 표시합니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA3WHGPEZ7SJ2CW55C5:EKS",
        "arn": "arn:aws:sts::111122223333:assumed-role/
AWSServiceRoleForAmazonEKSNodegroup/EKS",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA3WHGPEZ7SJ2CW55C5",
                "arn": "arn:aws:iam::111122223333:role/aws-service-role/eks-
nodegroup.amazonaws.com/AWSServiceRoleForAmazonEKSNodegroup",
                "accountId": "111122223333",
                "userName": "AWSServiceRoleForAmazonEKSNodegroup"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-02-26T00:56:33Z"
            }
        },
        "invokedBy": "eks-nodegroup.amazonaws.com"
    },
    "eventTime": "2020-02-26T00:56:34Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "DeleteInstanceProfile",
    "awsRegion": "region-code",
    "sourceIPAddress": "eks-nodegroup.amazonaws.com",
    "userAgent": "eks-nodegroup.amazonaws.com",
    "requestParameters": {
        "instanceProfileName": "eks-11111111-2222-3333-4444-abcdef123456"
    },
```

```
    "responseElements": null,
    "requestID": "11111111-2222-3333-4444-abcdef123456",
    "eventID": "11111111-2222-3333-4444-abcdef123456",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

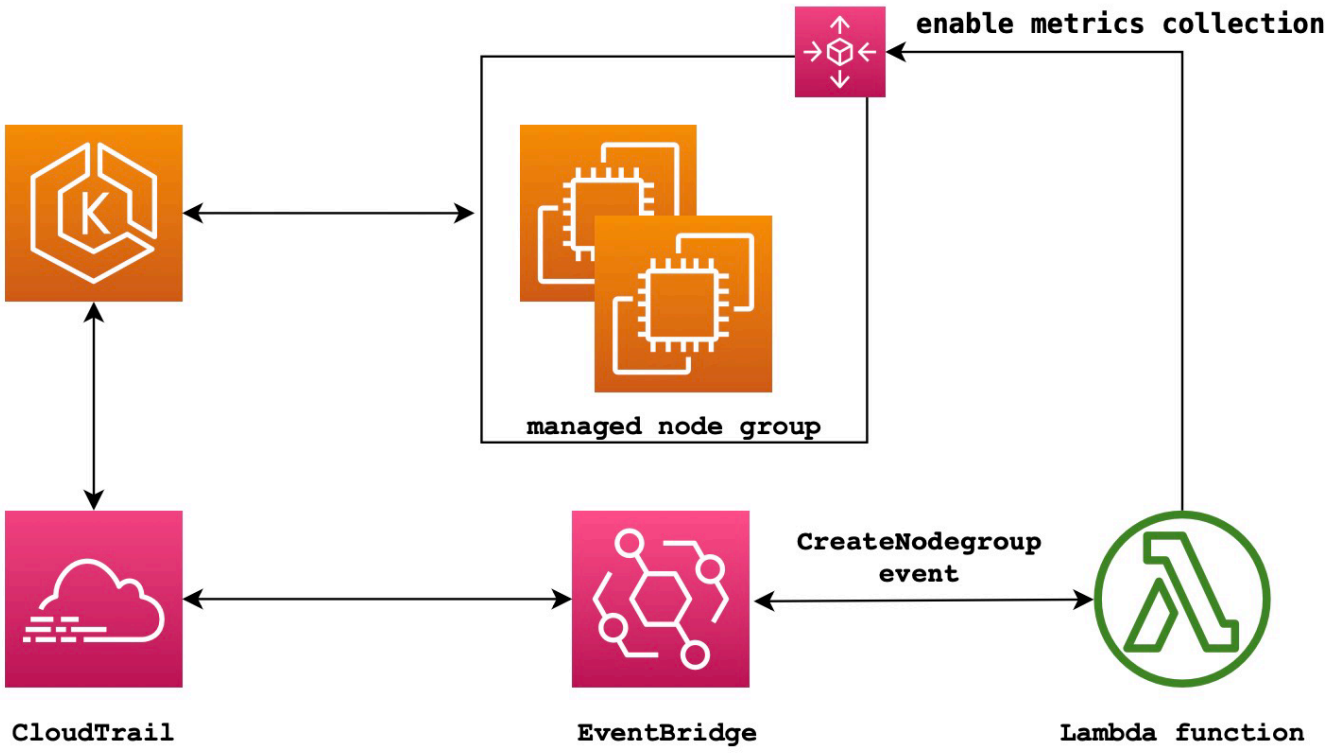
오토 스케일링 지표 수집 활성화

이 주제에서는 어떻게 AWS Lambda와 AWS CloudTrail을 사용하여 오토 스케일링 지표 수집을 활성화할 수 있는지 설명합니다. Amazon EKS에서는 관리형 노드용으로 생성된 오토 스케일링에 대한 그룹 지표 수집이 자동으로 활성화되지 않습니다.

오토 스케일링 지표를 사용하여 오토 스케일링의 변경 사항을 추적하고 임계값에 대한 경보를 설정할 수 있습니다. 오토 스케일링 지표는 Auto Scaling 콘솔 또는 Amazon CloudWatch 콘솔에서 사용할 수 있습니다. 활성화하면 오토 스케일링에서는 샘플링한 데이터를 1분마다 Amazon CloudWatch로 보냅니다. 이러한 지표를 활성화하는 데는 요금이 부과되지 않습니다.

오토 스케일링 지표 수집을 활성화하면 관리형 노드 그룹의 크기 조정을 모니터링할 수 있습니다. 오토 스케일링 지표를 통해 오토 스케일링의 최소, 최대 및 원하는 크기가 보고됩니다. 노드 그룹의 노드 수가 최소 크기 미만으로 감소하면 비정상 노드 그룹을 나타내는 경보를 생성할 수 있습니다. 노드 그룹 크기를 추적하면 데이터 영역의 용량이 부족하지 않도록 최대 개수를 조정하는 데도 유용합니다.

관리형 노드 그룹을 생성하면 AWS CloudTrail에서 Amazon EventBridge로 `CreateNodegroup` 이벤트를 보냅니다. `CreateNodegroup` 이벤트와 일치하는 Amazon EventBridge 규칙을 생성하면 관리형 노드 그룹과 연결된 오토 스케일링에 대한 그룹 지표 수집을 활성화하는 Lambda 함수가 트리거됩니다.



오토 스케일링 지표 수집 활성화 방법

1. Lambda용 IAM 역할을 생성합니다.

```
LAMBDA_ROLE=$(aws iam create-role \
  --role-name lambda-asg-enable-metrics \
  --assume-role-policy-document '{"Version": "2012-10-17","Statement":
 [{ "Effect": "Allow", "Principal": {"Service": "lambda.amazonaws.com"}, "Action":
 "sts:AssumeRole"}]}' \
  --output text \
  --query 'Role.Arn')
echo $LAMBDA_ROLE
```

2. Amazon EKS 노드 그룹을 설명하고 오토 스케일링 지표 수집을 활성화할 수 있는 정책을 생성합니다.

```
cat > /tmp/lambda-policy.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
```

```
      {
        "Effect": "Allow",
        "Action": [
            "eks:DescribeNodegroup",
            "autoscaling:EnableMetricsCollection"
            ],
        "Resource": [
            "*"
        ]
      }
    ]
}
EOF
LAMBDA_POLICY_ARN=$(aws iam create-policy \
  --policy-name lambda-asg-enable-metrics-policy \
  --policy-document file:///tmp/lambda-policy.json \
  --output text \
  --query 'Policy.Arn')
echo $LAMBDA_POLICY_ARN
```

3. 정책을 Lambda용 IAM 역할에 연결합니다.

```
aws iam attach-role-policy \
  --policy-arn $LAMBDA_POLICY_ARN \
  --role-name lambda-asg-enable-metrics
```

4. 함수를 통해 CloudWatch Logs에 로그를 쓰는 데 필요한 권한이 있는 `AWSLambdaBasicExecutionRole` 관리형 정책을 추가합니다.

```
aws iam attach-role-policy \
  --role-name lambda-asg-enable-metrics \
  --policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole
```

5. Lambda 코드를 생성합니다.

```
cat > /tmp/lambda-handler.py <<EOF
import json
import boto3
import time
import logging

eks = boto3.client('eks')
autoscaling = boto3.client('autoscaling')
```

```
logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    ASG_METRICS_COLLLECTION_TAG_NAME = "ASG_METRICS_COLLLECTION_ENABLED"
    initial_retry_delay = 10
    attempts = 0

    #print(event)

    if not event["detail"]["eventName"] == "CreateNodegroup":
        print("invalid event.")
        return -1

    clusterName = event["detail"]["requestParameters"]["name"]
    nodegroupName = event["detail"]["requestParameters"]["nodegroupName"]
    try:
        metricsCollectionEnabled = event["detail"]["requestParameters"]["tags"]
[ASG_METRICS_COLLLECTION_TAG_NAME]
    except KeyError:
        print(ASG_METRICS_COLLLECTION_TAG_NAME, "tag not found.")
        return

    # Check if metrics collection is enabled in tags
    if metricsCollectionEnabled.lower() != "true":
        print("Metrics collection is not enabled in nodegroup tags.")
        return

    # Get the name of the associated autoscaling group
    print("Getting the autoscaling group name for nodegroup=", nodegroupName, ",
 cluster=", clusterName )
    for i in range(0,10):
        try:
            autoScalingGroup =
 eks.describe_nodegroup(clusterName=clusterName,nodegroupName=nodegroupName)
["nodegroup"]["resources"]["autoScalingGroups"][0]["name"]
        except:
            attempts += 1
            print("Failed to obtain the associated autoscaling group for
 nodegroup", nodegroupName, "Retrying in", initial_retry_delay*attempts,
 "seconds.")
            time.sleep(initial_retry_delay*attempts)
        else:
```

```
            break

    print("Enabling metrics collection on autoscaling group ", autoScalingGroup)

    # Enable metrics collection in the autoscaling group
    try:
        enableMetricsCollection =
 autoscaling.enable_metrics_collection(AutoScalingGroupName=autoScalingGroup,Granularity="1
    except:
        print("Unable to enable metrics collection on nodegroup=",nodegroup)
    print("Enabled metrics collection on nodegroup", nodegroupName)
EOF
```

6. 배포 패키지를 만듭니다.

```
cd /tmp
zip function.zip lambda-handler.py
```

7. Lambda 함수를 생성합니다.

```
LAMBDA_ARN=$(aws lambda create-function --function-name asg-enable-metrics-
collection \
  --zip-file fileb://function.zip --handler lambda-handler.lambda_handler \
  --runtime python3.9 \
  --timeout 600 \
  --role $LAMBDA_ROLE \
  --output text \
  --query 'FunctionArn')
echo $LAMBDA_ARN
```

8. EventBridge 규칙을 생성합니다.

```
RULE_ARN=$(aws events put-rule --name CreateNodegroupRuleToLambda \
  --event-pattern "{\"source\":[\"aws.eks\"],\"detail-type\":[\"AWS API Call via
 CloudTrail\"],\"detail\":{\"eventName\":[\"CreateNodegroup\"],\"eventSource\":
[\"eks.amazonaws.com\"]}}" \
  --output text \
  --query 'RuleArn')
echo $RULE_ARN
```

9. Lambda 함수 대상으로 추가합니다.

```
aws events put-targets --rule CreateNodegroupRuleToLambda \
```

```
  --targets "Id"="1","Arn"="$LAMBDA_ARN"
```

10. EventBridge에서 Lambda 함수를 호출할 수 있는 정책을 추가합니다.

```
aws lambda add-permission \
  --function-name asg-enable-metrics-collection \
  --statement-id CreateNodegroupRuleToLambda \
  --action 'lambda:InvokeFunction' \
  --principal events.amazonaws.com \
  --source-arn $RULE_ARN
```

TRUE로 설정된 ASG_METRICS_COLLLECTION_ENABLED로 태그를 지정하는 관리형 노드 그룹의 오토 스케일링 지표 수집이 Lambda 함수를 통해 활성화됩니다. 오토 스케일링 지표 수집이 활성화되었는지 확인하려면 Amazon EC2 콘솔의 관련 오토 스케일링으로 이동합니다. Monitoring(모니터링) 탭의 Enable(활성화) 확인란이 활성화되어 있어야 합니다.

ADOT 연산자에 대한 Amazon EKS 추가 기능 지원

Amazon EKS는 AWS Management Console, AWS CLI, Amazon EKS API를 사용하여 AWS Distro for OpenTelemetry(ADOT) 연산자를 설치하고 관리합니다. 이로 인해 Amazon EKS에서 실행되는 애플리케이션에서 지표 및 추적 데이터를 Amazon CloudWatch, Prometheus 및 X-Ray와 같은 여러 모니터링 서비스 옵션으로 전송할 수 있습니다.

자세한 내용은 AWS Distro for OpenTelemetry 설명서에서 EKS 추가 기능을 사용하여 AWS Distro for OpenTelemetry 시작하기를 참조하세요.

Amazon EKS와 통합된 추가 AWS 서비스

다른 섹션에서 다루는 서비스 외에도 Amazon EKS는 더 많은 AWS 서비스를 사용하여 추가 솔루션을 제공합니다. 이 주제에서는 Amazon EKS를 사용하여 기능을 추가하는 몇 가지 서비스 또는 Amazon EKS에서 작업을 수행하는 데 사용하는 서비스에 대해 살펴봅니다.

주제

AWS CloudFormation을 사용하여 Amazon EKS 리소스 생성

Amazon EKS는 리소스 및 인프라를 생성하고 관리하는 데 소요되는 시간을 줄일 수 있도록 AWS 리소스를 모델링하고 설정하는 데 도움이 되는 서비스인 AWS CloudFormation과 통합됩니다. 필요한 모든 AWS 리소스(예: Amazon EKS 클러스터)를 설명하는 템플릿을 생성하면 AWS CloudFormation에서 해당 리소스의 프로비저닝과 구성을 담당합니다.

AWS CloudFormation을 사용할 때 템플릿을 재사용하여 Amazon EKS 리소스를 일관되고 반복적으로 설정할 수 있습니다. 리소스를 한 번만 설명하고 여러 AWS 계정 및 리전에서 동일한 리소스를 반복적으로 프로비저닝할 수 있습니다.

Amazon EKS 및 AWS CloudFormation 템플릿

Amazon EKS 및 관련 서비스에 대한 리소스를 프로비저닝하고 구성하려면 AWS CloudFormation 템플릿을 이해해야 합니다. 템플릿은 JSON 또는 YAML로 서식 지정된 텍스트 파일입니다. 이 템플릿은 AWS CloudFormation 스택에서 프로비저닝할 리소스에 대해 설명합니다. JSON 또는 YAML에 익숙하지 않은 경우 AWS CloudFormation Designer를 사용하면 AWS CloudFormation 템플릿을 시작하는 데 도움이 됩니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 AWS CloudFormation Designer란 무엇입니까?를 참조하세요.

Amazon EKS는 AWS CloudFormation에서 클러스터 및 노드 그룹 생성을 지원합니다. Amazon EKS 리소스에 대한 JSON 및 YAML 템플릿의 예를 포함한 자세한 내용은 AWS CloudFormation 사용 설명서의 Amazon EKS 리소스 유형 참조를 참조하세요.

AWS CloudFormation에 대해 자세히 알아보기

AWS CloudFormation에 대한 자세한 내용은 다음 리소스를 참조하세요.

- AWS CloudFormation
- AWS CloudFormation 사용 설명서
- AWS CloudFormation 명령줄 인터페이스 사용 설명서

Amazon EKS 및 AWS Local Zones

AWS Local Zone은 사용자와 지리적으로 근접한 AWS 리전의 확장입니다. Local Zones는 인터넷에 대한 자체 연결을 가지고 있으며 AWS Direct Connect를 지원합니다. Local Zones에서 생성된 리소스는 대기 시간이 매우 짧은 통신으로 로컬 사용자에게 제공될 수 있습니다. 자세한 내용은 Local Zones를 참조하세요.

Amazon EKS는 Local Zones의 특정 리소스를 지원합니다. 여기에는 자체 관리형 Amazon EC2 노드, Amazon EBS 볼륨 및 Application Load Balers(ALB)가 포함됩니다. Amazon EKS 클러스터의 일부로 Local Zones를 사용할 때 다음을 고려하는 것이 좋습니다.

Nodes(노드)

Amazon EKS를 사용하여 Local Zones에서 관리형 노드 그룹 또는 Fargate 노드를 생성할 수 없습니다. 그러나 Amazon EC2 API, AWS CloudFormation 또는 `eksctl`을 사용하여 Local Zones에서 자체 관리형 Amazon EC2 노드를 생성할 수 있습니다. 자세한 내용은 자체 관리형 노드 섹션을 참조하세요.

네트워크 아키텍처

- Amazon EKS 관리형 Kubernetes 컨트롤 플레인은 항상 AWS 리전에서 실행됩니다. Amazon EKS 관리형 Kubernetes 컨트롤 플레인은 로컬 영역에서 실행될 수 없습니다. Local Zones는 VPC 내에 서브넷으로 표시되므로 Kubernetes는 로컬 영역 리소스를 해당 서브넷의 일부로 인식합니다.
- Amazon EKS Kubernetes 클러스터는 Amazon EKS 관리형 탄력적 네트워크 인터페이스를 사용하여 AWS 리전 또는 로컬 영역을 실행하는 Amazon EC2 인스턴스와 통신합니다. Amazon EKS 네트워킹 아키텍처에 대한 자세한 내용은 Amazon EKS 네트워킹 섹션을 참조하세요.

- 리전 서브넷과 달리 Amazon EKS는 로컬 영역 서브넷에는 네트워크 인터페이스를 배치할 수 없습니다. 즉, 클러스터를 만들 때 로컬 영역 서브넷을 지정하지 않아야 합니다.

Deep Learning Containers

AWS Deep Learning Containers는 Amazon EKS 및 Amazon Elastic Container Service(Amazon ECS)의 TensorFlow에서 모델을 훈련하고 제공하기 위한 Docker 이미지 세트입니다. Deep Learning Containers는 TensorFlow, NVIDIA CUDA(GPU 인스턴스용) 및 Intel MKL(CPU 인스턴스용) 라이브러리에 최적의 환경을 제공하며, Amazon ECR에서 사용할 수 있습니다.

Amazon EKS에서 AWS Deep Learning Containers 사용을 시작하려면 AWS Deep Learning Containers 개발자 가이드에서 Amazon EKS 설정을 참조하세요.

Amazon VPC Lattice

Amazon VPC Lattice는 AWS 네트워킹 인프라에 직접 구축된 완전관리형 애플리케이션 네트워킹 서비스로, 여러 계정과 Virtual Private Cloud(VPC)에서 서비스를 연결, 보호 및 모니터링하는 데 사용할 수 있습니다. Amazon EKS를 사용하면 Kubernetes Gateway API의 구현인 AWS Gateway API 컨트롤러를 사용하여 Amazon VPC Lattice를 활용할 수 있습니다. Amazon VPC Lattice를 사용하면 간단하고 일관된 방식으로 표준 Kubernetes 의미 체계를 사용하여 클러스터 간 연결을 설정할 수 있습니다. Amazon EKS에서 Amazon VPC Lattice를 사용하기 시작하려면 AWS Gateway API 컨트롤러 사용 설명서를 참조하세요.

AWS Resilience Hub

AWS Resilience Hub는 인프라를 분석하여 Amazon EKS 클러스터의 복원력을 평가합니다. AWS Resilience Hub는 Kubernetes 역할 기반 액세스 제어(RBAC) 구성을 사용하여 클러스터에 배포된 Kubernetes 워크로드를 평가합니다. 자세한 내용은 AWS Resilience Hub 사용 설명서의 Amazon EKS 클러스터에 대한 AWS Resilience Hub 액세스 활성화를 참조하세요.

Amazon GuardDuty

Amazon GuardDuty의 EKS Protection은 AWS 환경 내에서 Amazon EKS 클러스터를 보호하는 데 도움이 되는 위협 탐지 범위를 제공합니다. EKS Protection에는 EKS 감사 로그 모니터링과 EKS 런타임 모니터링이 포함됩니다. 자세한 내용은 Amazon GuardDuty 사용 설명서의 Amazon GuardDuty의 EKS 보호를 참조하세요. GuardDuty 에이전트를 클러스터에 Amazon EKS 추가 기능으로 설치할 수 있습니다. 자세한 내용은 Amazon EKS에서 사용 가능한 Amazon EKS 추가 기능 섹션을 참조하세요.

Amazon EKS에서 Amazon Security Lake 사용

Amazon Security Lake는 Amazon EKS를 비롯한 다양한 소스의 보안 데이터를 중앙 집중화하는 데 사용할 수 있는 완전 관리형 보안 데이터 레이크 서비스입니다. Amazon EKS와 Security Lake를 통합하면 Kubernetes 리소스에서 수행되는 활동에 대한 심층적인 통찰력을 얻고 Amazon EKS 클러스터의 보안 태세를 강화할 수 있습니다.

> **Note**
>
> Amazon EKS와 함께 Security Lake를 사용하고 데이터 소스를 설정하는 방법에 대한 자세한 내용은 Amazon Security Lake 설명서를 참조하세요.

Amazon EKS에서 Security Lake를 사용하여 얻을 수 있는 이점

보안 데이터 중앙 집중화 - Security Lake는 다른 AWS 서비스, SaaS 공급자, 온프레미스 소스 및 타사 소스의 데이터와 함께 Amazon EKS 클러스터의 보안 데이터를 자동으로 수집하고 중앙 집중화합니다. 이를 통해 조직 전반의 보안 태세에 대한 포괄적인 뷰를 제공합니다.

데이터 형식 표준화 - Security Lake는 수집된 데이터를 표준 오픈 소스 스키마인 OCSF(Open Cyber Security Schema Framework) 형식으로 변환합니다. 이러한 정규화를 통해 분석과 다른 보안 도구 및 서비스와의 통합이 더 쉬워집니다.

위협 탐지 개선 - Amazon EKS 컨트롤 플레인 로그를 비롯한 중앙 집중식 보안 데이터를 분석하여 Amazon EKS 클러스터 내에서 잠재적으로 의심스러운 활동을 보다 효과적으로 탐지할 수 있습니다. 이를 통해 보안 사고를 즉시 식별하고 이에 대응할 수 있습니다.

데이터 관리 간소화 - Security Lake는 사용자 지정 가능한 보존 및 복제 설정을 통해 보안 데이터의 수명 주기를 관리합니다. 이를 통해 데이터 관리 작업이 간소화되고 규정 준수 및 감사 목적으로 필요한 데이터를 보존할 수 있습니다.

Amazon EKS에서 Security Lake 활성화

Amazon EKS에서 Security Lake 사용을 시작하려면 다음 단계를 따르십시오.

1. EKS 클러스터에서 Amazon EKS 컨트롤 플레인 로깅을 활성화합니다. 자세한 지침은 컨트롤 플레인 로그 활성화 및 비활성화를 참조하세요.

2. Amazon EKS 감사 로그를 Security Lake의 소스로 추가합니다. 그러면 Security Lake가 EKS 클러스터에서 실행되고 있는 Kubernetes 리소스에서 수행된 활동에 대한 심층적인 정보를 수집하기 시작합니다.
3. 요구 사항에 따라 Security Lake의 보안 데이터에 대한 보존 및 복제 설정을 구성합니다.
4. Security Lake에 저장된 정규화된 OCSF 데이터를 인시던트 대응, 보안 분석 및 다른 AWS 서비스 또는 타사 도구와의 통합을 위해 사용합니다. 예를 들어, Amazon OpenSearch Ingestion을 사용하여 Amazon Security Lake 데이터로부터 보안 인사이트를 생성할 수 있습니다.

Security Lake에서 EKS 로그 분석

Security Lake는 EKS 로그 이벤트를 OCSF 형식으로 정규화하여 데이터를 더 쉽게 분석하고 다른 보안 이벤트와 상호 연관시킬 수 있도록 만듭니다. Amazon Athena, Amazon QuickSight 또는 타사 보안 분석 도구와 같은 다양한 도구 및 서비스를 사용하여 정규화된 데이터를 쿼리하고 시각화할 수 있습니다.

EKS 로그 이벤트의 OCSF 매핑에 대한 자세한 내용은 OCSF GitHub 리포지토리의 매핑 참조를 참조하세요.

Amazon Detective

Amazon Detective는 사용자가 보안 조사 결과 또는 의심스러운 활동의 근본 원인을 분석 및 조사하고 신속하게 식별하는 데 도움이 됩니다. Detective는 AWS 리소스에서 로그 데이터를 자동으로 수집합니다. 그런 다음 기계 학습, 통계 분석 및 그래프 이론을 사용하여 더 빠르고 효율적으로 보안 조사를 수행할 수 있도록 시각화를 생성합니다. Detective의 사전 구축된 데이터 집계, 요약 및 컨텍스트는 가능한 보안 문제의 특성과 범위를 신속하게 분석하고 확인하는 데 도움이 됩니다. 자세한 내용은 Amazon Detective 사용 설명서를 참조하세요.

Detective는 Kubernetes와(과) AWS 데이터를 다음과 같은 조사 결과로 정리합니다.

- 클러스터를 생성한 IAM ID 및 클러스터의 서비스 역할을 포함한 Amazon EKS 클러스터 세부 정보. Detective를 사용하여 이러한 IAM 자격 증명의 AWS 및 Kubernetes API 활동을 조사할 수 있습니다.
- 컨테이너 세부 정보(예: 이미지 및 보안 컨텍스트). 종료된 Pods에 대한 세부 정보를 검토할 수도 있습니다.
- API 활동의 전체 추세와 특정 API 호출에 대한 세부 정보를 포함한 Kubernetes API 활동. 예를 들어 선택한 시간 범위 동안 실행된 성공 및 실패한 Kubernetes API 호출 수를 표시할 수 있습니다. 또한 새로 관찰된 API 호출에 대한 섹션은 의심스러운 활동을 식별하는 데 도움이 될 수 있습니다.

Amazon EKS 감사 로그는 Detective 행동 그래프에 추가할 수 있는 선택적 데이터 소스 패키지입니다. 계정에서 사용 가능한 선택적 소스 패키지와 해당 상태를 볼 수 있습니다. 자세한 내용은 Amazon Detective 사용 설명서의 Detective에 대한 Amazon EKS 감사 로그를 참조하세요.

Amazon EKS와 함께 Amazon Detective 사용

Amazon EKS 클러스터에 대한 조사사 결과를 검토하는 방법

조사 결과를 검토하기 전에 클러스터가 있는 곳과 같은 AWS 리전에서 최소 48시간 동안 Detective를 활성화해야 합니다. 자세한 내용은 Amazon Detective 사용 설명서의 Amazon Detective 설정을 참조하세요.

1. https://console.aws.amazon.com/detective/에서 Detective 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 검색을 선택합니다.
3. [유형 선택]을 선택한 다음 [EKS 클러스터]를 선택합니다.
4. 클러스터 이름 또는 ARN을 입력한 다음 검색을 선택합니다.
5. 검색 결과에서 활동을 볼 클러스터 이름을 선택합니다. 확인할 수 있는 내용에 대한 자세한 내용은 Amazon Detective 사용 설명서의 Amazon EKS 클러스터와 관련된 전체 Kubernetes API 활동을 참조하세요.

Amazon EKS 문제 해결

이 장에서는 Amazon EKS를 사용하는 동안 발생할 수 있는 몇 가지 일반 오류와 이를 해결하는 방법을 다룹니다. 특정 Amazon EKS 영역의 문제를 해결해야 하는 경우 별도의 IAM 문제 해결. Amazon EKS 커넥터의 문제 해결 및 EKS 추가 기능을 사용하여 ADOT 문제 해결 주제를 참조하세요.

기타 문제 해결 정보는 AWS re:Post의 Amazon 엘라스틱 쿠버네티스 서비스에 대한 지식 센터 콘텐츠를 참조하세요.

용량 부족

Amazon EKS 클러스터를 생성하려는 도중 다음 오류가 표시되는 경우 지정된 가용 영역 중 하나에 클러스터를 지원하는 충분한 용량이 없습니다.

```
Cannot create cluster 'example-cluster' because region-1d, the targeted
Availability Zone, does not currently have sufficient capacity to support
the cluster. Retry and choose from these Availability Zones: region-1a,
region-1b, region-1c
```

이 오류 메시지를 반환한 가용 영역에서 호스팅하는 클러스터 VPC의 서브넷을 사용하여 클러스터를 재생성하십시오.

클러스터가 상주할 수 없는 가용 영역이 있습니다. 서브넷이 있는 가용 영역을 서브넷 요구 사항 및 고려 사항의 가용 영역 목록과 비교합니다.

노드가 클러스터 조인에 실패

노드가 클러스터에 조인하지 못하는 몇 가지 일반적인 원인은 다음과 같습니다.

- 노드가 관리형 노드인 경우 Amazon EKS는 노드 그룹을 생성할 때 항목을 `aws-auth ConfigMap`에 추가합니다. 항목이 제거되거나 수정된 경우 항목을 다시 추가해야 합니다. 자세한 내용을 보려면 터미널에 **`eksctl create iamidentitymapping --help`**를 입력합니다. 다음 명령에서 *`my-cluster`*를 클러스터 이름으로 바꾸고 수정된 명령(**`eksctl get iamidentitymapping --cluster`** ***`my-cluster`***)을 실행하여 현재 `aws-auth ConfigMap` 항목을 확인할 수 있습니다. 지정하는 역할의 ARN에 `/` 이외의 경로를 포함할 수 없습니다. 예를 들어, 역할 이름이 `development/apps/my-role`인 경우 역할에 대한 ARN을 지정할 때 역할 이름을

my-role로 변경해야 합니다. 노드 IAM 역할 ARN(인스턴스 프로파일 ARN이 아님)을 지정해야 합니다.

자체 관리형 노드이고 노드의 IAM 역할 ARN에 대한 액세스 항목을 생성하지 않은 경우 관리형 노드에 대해 나열된 것과 동일한 명령을 실행합니다. 노드 IAM 역할의 ARN에 대한 액세스 항목을 생성한 경우 액세스 항목에서 올바르게 구성되지 않을 수 있습니다. aws-auth ConfigMap 항목 또는 액세스 항목에서 노드 IAM 역할 ARN(인스턴스 프로파일 ARN이 아님)이 보안 주체 ARN으로 지정되어야 합니다. 액세스 항목에 대한 자세한 내용은 액세스 항목 관리 섹션을 참조하십시오.

- 노드 AWS CloudFormation템플릿에서 ClusterName이 조인할 클러스터의 이름과 정확히 일치하지 않습니다. 이 필드에 올바르지 않은 값을 전달하면 노드의 /var/lib/kubelet/kubeconfig 파일의 올바르지 않은 구성이 발생하고, 노드가 클러스터에 조인되지 않습니다.
- 노드가 클러스터에서 소유하는 것으로 태그가 지정되지 않았습니다. 노드에는 다음 태그가 적용되어야 합니다. 여기서 *my-cluster*은 클러스터의 이름으로 교체됩니다.

키	값
kubernetes.io/cluster/ *my-cluster*	owned

- 노드는 퍼블릭 IP 주소를 사용하여 클러스터에 액세스하지 못할 수 있습니다. 퍼블릭 서브넷에 배포된 노드에 퍼블릭 IP 주소가 할당되었는지 확인합니다. 그렇지 않은 경우 시작 후 탄력적 IP 주소를 노드에 연결할 수 있습니다. 자세한 내용은 실행 중인 인스턴스 또는 네트워크 인터페이스와 탄력적 IP 주소 연결을 참조하세요. 퍼블릭 IP 주소를 배포된 인스턴스에 자동으로 할당하도록 퍼블릭 서브넷이 설정되어 있지 않은 경우, 해당 설정을 사용하도록 설정하는 것이 좋습니다. 자세한 내용을 알아보려면 서브넷의 퍼블릭 IPv4 주소 지정 속성 수정을 참조하세요. 노드가 프라이빗 서브넷에 배포된 경우, 서브넷에는 퍼블릭 IP 주소가 할당된 NAT 게이트웨이에 대한 경로가 있어야 합니다.
- 노드를 배포할 AWS 리전의 AWS STS 엔드포인트가 계정에서 활성화되지 않았습니다. 지역을 활성화하려면 AWS 리전에서 AWS STS 활성화 및 비활성화를 참조하세요.
- 노드에 프라이빗 DNS 항목이 없으므로 kubelet 로그에 node "" not found 오류가 포함됩니다. 노드가 생성된 VPC에 대해 DHCP options set에서 Options로 domain-name 및 domain-name-servers 값이 설정되었는지 확인합니다. 기본값은 domain-name:<region>.compute.internal 및 domain-name-servers:AmazonProvidedDNS입니다. 자세한 내용은 Amazon VPC 사용 설명서에서 DHCP 옵션 세트를 참조하세요.
- 관리형 노드 그룹의 노드가 15분 내에 클러스터에 연결되지 않으면 “NodeCreationFailure”라는 상태 문제가 발생하고 콘솔 상태가 Create failed로 설정됩니다. 시작 시간이 느린 Windows AMI의 경우 빠른 실행을 사용하여 이 문제를 해결할 수 있습니다.

작업자 노드의 클러스터 조인을 방해하는 일반적인 원인을 식별하고 문제를 해결하기 위해 `AWSSupport-TroubleshootEKSWorkerNode` 실행서를 사용할 수 있습니다. 자세한 내용은 AWS Systems Manager Automation 실행서 참조에서 `AWSSupport-TroubleshootEKSWorkerNode`를 참조하세요.

권한이 없거나 액세스가 거부됨(`kubectl`)

`kubectl` 명령을 실행할 때 다음 오류 중 하나가 발생하는 경우 `kubectl`이 Amazon EKS에 대해 올바르게 구성되지 않았거나 사용하고 있는 IAM 보안 주체(역할 또는 사용자)의 보안 인증 정보가 Amazon EKS 클러스터의 Kubernetes 객체에 대해 충분한 권한을 보유한 Kubernetes 사용자 이름에 매핑되지 않은 것입니다.

- `could not get token: AccessDenied: Access denied`
- `error: You must be logged in to the server (Unauthorized)`
- `error: the server doesn't have a resource type "svc"`

이유는 다음 중 하나 때문일 수 있습니다.

- 클러스터가 하나의 IAM 보안 주체에 대한 보안 인증 정보로 생성되었지만 `kubectl`이 다른 IAM 보안 주체의 보안 인증 정보를 사용하도록 구성되었기 때문입니다. 이 문제를 해결하려면 클러스터를 생성한 보안 인증 정보를 사용하도록 `kube config` 파일을 업데이트합니다. 자세한 내용은 Amazon EKS 클러스터용 `kubeconfig` 파일 생성 또는 업데이트 단원을 참조하십시오.
- 클러스터가 액세스 항목 관리의 사전 조건 섹션에 나온 최소 플랫폼 요구 사항을 충족하는 경우 IAM 보안 주체에 액세스 항목이 존재하지 않습니다. 존재하는 경우 필수 Kubernetes 그룹 이름이 정의되지 않았거나 적절한 액세스 정책이 연결되지 않은 것입니다. 자세한 내용은 액세스 항목 관리 단원을 참조하십시오.
- 클러스터가 액세스 항목 관리의 최소 플랫폼 요구 사항을 충족하지 않는 경우 IAM 보안 주체가 포함된 항목이 `aws-auth ConfigMap`에 존재하지 않습니다. 존재하는 경우 필요한 권한을 보유한 Kubernetes `Role` 또는 `ClusterRole`에 바인딩된 Kubernetes 그룹 이름에 매핑되지 않은 것입니다. Kubernetes 역할 기반 권한 부여(RBAC) 객체에 대한 자세한 내용은 Kubernetes 설명서에서 Using RBAC authorization을 참조하세요. 다음 명령에서 ***`my-cluster`***를 클러스터 이름으로 바꾸고 수정된 명령(**`eksctl get iamidentitymapping --cluster my-cluster`**)을 실행하여 현재 `aws-auth ConfigMap` 항목을 확인할 수 있습니다. IAM 보안 주체의 ARN이 포함된 항목이 `ConfigMap`에 없는 경우 터미널에 **`eksctl create iamidentitymapping --help`**를 입력하여 생성 방법을 알아보세요.

AWS CLI를 설치하고 구성하면 사용할 IAM 자격 증명을 구성할 수 있습니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 AWS CLI 구성을 참조하세요. 클러스터의 Kubernetes 객체에 액세스하기 위해 IAM 역할을 수임하는 경우 IAM 역할을 사용하도록 kubectl을 구성할 수도 있습니다. 자세한 내용은 Amazon EKS 클러스터용 kubeconfig 파일 생성 또는 업데이트 단원을 참조하십시오.

hostname doesn't match

시스템의 Python 버전이 2.7.9 이상이어야 합니다. 그렇지 않은 경우 Amazon EKS에 대한 AWS CLI 호출과 함께 hostname doesn't match 오류가 표시됩니다. 자세한 내용은 Python Requests Frequently Asked Questions에서 What are "hostname doesn't match" errors?를 참조하세요.

getsockopt: no route to host

Docker는 Amazon EKS 클러스터의 172.17.0.0/16 CIDR 범위에서 실행됩니다. 클러스터의 VPC 서브넷이 이 범위와 겹치지 않게 하는 것이 좋습니다. 그렇지 않으면 다음 오류가 발생합니다.

```
Error: : error upgrading connection: error dialing backend: dial tcp
 172.17.<nn>.<nn>:10250: getsockopt: no route to host
```

Instances failed to join the Kubernetes cluster

AWS Management Console에서 Instances failed to join the Kubernetes cluster 오류 메시지가 표시되면 클러스터의 프라이빗 엔드포인트 액세스가 활성화되어 있는지 또는 퍼블릭 엔드포인트 액세스에 대해 CIDR 블록을 올바르게 구성했는지 확인합니다. 자세한 내용은 Amazon EKS 클러스터 엔드포인트 액세스 제어 단원을 참조하십시오.

관리형 노드 그룹 오류

관리형 노드 그룹에 하드웨어 상태 문제가 발생하면 Amazon EKS는 문제 진단에 도움이 되는 오류 메시지를 반환합니다. 이러한 상태 확인은 Amazon EC2 상태 확인을 기반으로 하기 때문에 소프트웨어 문제를 감지하지 못합니다. 다음 목록에서는 오류 코드에 대해 설명합니다.

AccessDenied

Amazon EKS 또는 하나 이상의 관리형 노드가 Kubernetes 클러스터 API 서버로 인증하거나 인증하지 못합니다. 일반적인 응답 헤더에 대한 자세한 내용은 관리형 노드 그룹의 일반적

인AccessDenied 오류 원인 해결을 참조하십시오. 프라이빗Windows AMI로 인해 오류 메시지와 함께 이 오류 코드가 발생할 수도 있습니다.`Not authorized for images` 자세한 내용은 `Not authorized for images` 단원을 참조하십시오.

AmiIdNotFound

시작 템플릿과 연결된 AMI ID를 찾을 수 없습니다. AMI가 존재하고 계정과 공유되는지 확인합니다.

AutoScalingGroupNotFound

관리형 노드 그룹과 연결된 Auto Scaling 그룹을 찾을 수 없습니다. 복구하려면 동일한 설정으로 Auto Scaling 그룹을 다시 만들어야 합니다.

ClusterUnreachable

Amazon EKS 또는 하나 이상의 관리형 노드가 Kubernetes 클러스터 API 서버와 통신할 수 없습니다. 네트워크 중단이 있거나 API 서버가 요청 처리 시간을 초과한 경우 이 문제가 발생할 수 있습니다.

Ec2SecurityGroupNotFound

클러스터의 클러스터 보안 그룹을 찾을 수 없습니다. 클러스터를 다시 생성해야 합니다.

Ec2SecurityGroupDeletionFailure

관리형 노드 그룹의 원격 액세스 보안 그룹을 삭제할 수 없습니다. 보안 그룹에서 종속성을 제거합니다.

Ec2LaunchTemplateNotFound

관리형 노드 그룹의 Amazon EC2 시작 템플릿을 찾을 수 없습니다. 복구하려면 노드 그룹을 다시 생성해야 합니다.

Ec2LaunchTemplateVersionMismatch

관리형 노드 그룹의 Amazon EC2 시작 템플릿 버전이 Amazon EKS에서 생성한 버전과 일치하지 않습니다. 복구하려면 Amazon EKS에서 생성한 버전으로 되돌려야 합니다.

IamInstanceProfileNotFound

관리형 노드 그룹의 IAM 인스턴스 프로파일을 찾을 수 없습니다. 복구하려면 동일한 설정으로 인스턴스 프로파일을 다시 만들어야 합니다.

IamNodeRoleNotFound

관리형 노드 그룹의 IAM 역할을 찾을 수 없습니다. 복구하려면 동일한 설정으로 IAM 역할을 다시 만들어야 합니다.

AsgInstanceLaunchFailures

인스턴스를 시작하려고 시도하는 중에 Auto Scaling 그룹에 오류가 발생했습니다.

NodeCreationFailure

시작된 인스턴스를 Amazon EKS 클러스터에 등록할 수 없습니다. 이러한 오류의 일반적인 원인은 노드 IAM 역할 권한이 충분하지 않거나 노드에 대한 아웃바운드 인터넷 액세스가 부족하기 때문입니다. 노드는 다음 요구 사항 중 하나를 충족해야 합니다.

- 퍼블릭 IP 주소를 사용하여 인터넷에 액세스할 수 있습니다. 노드가 있는 서브넷에 연결된 보안 그룹은 통신을 허용해야 합니다. 자세한 내용은 서브넷 요구 사항 및 고려 사항 및 Amazon EKS 보안 그룹 요구 사항 및 고려 사항 단원을 참조하세요.
- 노드와 VPC는 프라이빗 클러스터 요구 사항의 요구 사항을 충족해야 합니다.

InstanceLimitExceeded

AWS 계정에서 해당 인스턴스 유형의 인스턴스를 더 이상 시작할 수 없습니다. 복구하려면 Amazon EC2 인스턴스 제한의 증가를 요청해야 합니다.

InsufficientFreeAddresses

관리형 노드 그룹과 연결된 하나 이상의 서브넷에 새 노드에 사용 가능한 IP 주소가 충분하지 않습니다.

InternalFailure

이러한 오류는 일반적으로 Amazon EKS 서버 측 문제로 인해 발생합니다.

관리형 노드 그룹의 일반적인**AccessDenied** 오류 원인 해결

`AccessDenied` 오류의 가장 흔한 원인은 관리형 노드 그룹에서 작업을 수행할 때 `eks:node-manager` `ClusterRole` 또는 `ClusterRoleBinding`이 누락되는 것입니다. Amazon EKS는 관리형 노드 그룹을 사용한 온보딩의 일부로 클러스터에 이러한 리소스를 설정하며 이러한 리소스는 노드 그룹을 관리하는 데 필요합니다.

`ClusterRole`은 시간이 지남에 따라 변경될 수 있지만 다음 예제와 비슷해야 합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
```

```
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
  - create
```

ClusterRoleBinding은 시간이 지남에 따라 변경될 수 있지만 다음 예제와 비슷해야 합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

eks:node-manager ClusterRole이 존재하는지 확인합니다.

```
kubectl describe clusterrole eks:node-manager
```

존재하는 경우, 출력을 이전 ClusterRole 예와 비교합니다.

eks:node-manager ClusterRoleBinding이 존재하는지 확인합니다.

```
kubectl describe clusterrolebinding eks:node-manager
```

존재하는 경우, 출력을 이전 ClusterRoleBinding 예와 비교합니다.

관리형 노드 그룹 작업을 요청하는 동안 누락되거나 손상된 ClusterRole 또는 ClusterRoleBinding이 AcessDenied 오류의 원인으로 확인된 경우 복원할 수 있습니다. 다음 콘텐츠를 *eks-node-manager-role.yaml*이라는 파일에 저장합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: eks:node-manager
rules:
- apiGroups:
  - ''
  resources:
  - pods
  verbs:
  - get
  - list
  - watch
  - delete
- apiGroups:
  - ''
  resources:
  - nodes
  verbs:
  - get
  - list
  - watch
  - patch
- apiGroups:
  - ''
  resources:
  - pods/eviction
  verbs:
```

```
  - create
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: eks:node-manager
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: eks:node-manager
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: eks:node-manager
```

파일을 적용합니다.

```
kubectl apply -f eks-node-manager-role.yaml
```

노드 그룹 작업을 다시 시도하여 문제가 해결되었는지 확인합니다.

Not authorized for images

Not authorized for images오류 메시지의 잠재적 원인 중 하나는 프라이빗 Amazon EKSWindows AMI를 사용하여Windows 관리형 노드 그룹을 시작하는 것입니다. 새 Windows AMI를 릴리스한 후 AWS에서는 4개월이 지난 AMI를 프라이빗으로 설정하여 더 이상 액세스할 수 없도록 합니다. 관리형 노드 그룹이 프라이빗 Windows AMI를 사용하는 경우 Windows 관리형 노드 그룹 업데이트를 고려해 보세요. 프라이빗으로 설정된 AMI에 대한 액세스 제공을 보장할 수는 없지만 AWS Support에 티켓을 제출하여 액세스를 요청할 수 있습니다. 자세한 내용은 Windows 인스턴스용 Amazon EC2 사용 설명서의 패치, 보안 업데이트 및 AMI ID를 참조하세요.

노드가 NotReady 상태임

노드가 NotReady 상태인 경우 해당 노드가 비정상이어서 새 Pods를 예약할 수 없을 가능성이 있습니다. 이는 노드에 CPU 리소스, 메모리 또는 사용 가능한 디스크 공간이 충분하지 않은 등의 다양한 이유로 발생할 수 있습니다.

Amazon EKS 최적화 Windows AMI의 경우 kubelet 구성에 기본적으로 지정된 컴퓨팅 리소스를 예약할 수 없습니다. 리소스 문제를 방지하는 데 도움이 되도록 kubelet에 kube-reserved 및/또는

system-reserved의 구성 값을 제공하여 시스템 프로세스에 컴퓨팅 리소스를 예약할 수 있습니다. 부트스트랩 스크립트의 -KubeletExtraArgs 명령줄 파라미터를 사용하여 이를 수행합니다. 자세한 내용은 Kubernetes 설명서의 Reserve Compute Resources for System Daemons와 이 사용 설명서의 부트스트랩 스크립트 구성 파라미터를 참조하세요.

CNI 로그 수집 도구

Amazon VPC CNI plugin for Kubernetes에는 /opt/cni/bin/aws-cni-support.sh의 노드에서 자체 문제 해결 스크립트를 사용할 수 있습니다. 이 스크립트를 사용하여 지원 사례 및 일반 문제 해결에 대한 진단 로그를 수집할 수 있습니다.

다음 명령을 사용하여 노드에서 스크립트를 실행합니다.

```
sudo bash /opt/cni/bin/aws-cni-support.sh
```

> **Note**
>
> 해당 위치에 스크립트가 없으면 CNI 컨테이너를 실행할 수 없습니다. 다음 명령을 사용하여 스크립트를 수동으로 다운로드하고 실행할 수 있습니다.
>
> ```
> curl -O https://raw.githubusercontent.com/awslabs/amazon-eks-ami/master/log-collector-script/linux/eks-log-collector.sh
> sudo bash eks-log-collector.sh
> ```

이 스크립트는 다음 진단 정보를 수집합니다. 배포한 CNI 버전은 스크립트 버전보다 이전 버전일 수 있습니다.

```
      This is version 0.6.1. New versions can be found at https://github.com/awslabs/amazon-eks-ami

Trying to collect common operating system logs...
Trying to collect kernel logs...
Trying to collect mount points and volume information...
Trying to collect SELinux status...
Trying to collect iptables information...
Trying to collect installed packages...
Trying to collect active system services...
Trying to collect Docker daemon information...
```

```
Trying to collect kubelet information...
Trying to collect L-IPAMD information...
Trying to collect sysctls information...
Trying to collect networking information...
Trying to collect CNI configuration information...
Trying to collect running Docker containers and gather container data...
Trying to collect Docker daemon logs...
Trying to archive gathered information...

 Done... your bundled logs are located in /var/
log/eks_i-0717c9d54b6cfaa19_2020-03-24_0103-UTC_0.6.1.tar.gz
```

진단 정보는 다음에 수집되고 저장됩니다.

```
/var/log/eks_i-0717c9d54b6cfaa19_2020-03-24_0103-UTC_0.6.1.tar.gz
```

컨테이너 런타임 네트워크가 준비되지 않음

다음과 같은 `Container runtime network not ready` 오류 및 권한 부여 오류가 나타날 수 있습니다.

```
4191 kubelet.go:2130] Container runtime network not ready: NetworkReady=false
 reason:NetworkPluginNotReady message:docker: network plugin is not ready: cni config
 uninitialized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
4191 kubelet_node_status.go:106] Unable to register node
 "ip-10-40-175-122.ec2.internal" with API server: Unauthorized
4191 reflector.go:205] k8s.io/kubernetes/pkg/kubelet/kubelet.go:452: Failed to list
 *v1.Service: Unauthorized
```

이유는 다음 중 하나 때문일 수 있습니다.

1. 클러스터에 `aws-auth ConfigMap`이 없거나 노드를 구성하는 데 사용한 IAM 역할 항목이 포함되어 있지 않습니다.

   노드가 다음 기준 중 하나를 충족하는 경우 이 `ConfigMap` 항목이 필요합니다.

   - Kubernetes 또는 플랫폼 버전을 사용하는 클러스터의 관리형 노드.
   - 액세스 항목 관리 주제의 사전 조건 섹션에 나열된 플랫폼 버전보다 이전 버전인 클러스터의 자체 관리형 노드.

이 문제를 해결하기 위해 다음 명령에서 *my-cluster*를 클러스터 이름으로 바꾸고 수정된 명령(**eksctl get iamidentitymapping --cluster *my-cluster***)을 실행하여 ConfigMap에서 기존 항목을 확인합니다. 명령에서 오류 메시지를 받은 경우 클러스터에 aws-auth ConfigMap이 없기 때문일 수 있습니다. 다음 명령은 ConfigMap에 항목을 추가합니다. ConfigMap이 없으면 명령에서 생성합니다. *111122223333*을 IAM 역할의 AWS 계정 ID로, *myAmazonEKSNodeRole*을 노드 역할 이름으로 바꿉니다.

```
eksctl create iamidentitymapping --cluster my-cluster \
    --arn arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --group
 system:bootstrappers,system:nodes \
    --username system:node:{{EC2PrivateDNSName}}
```

지정하는 역할의 ARN에 / 이외의 경로를 포함할 수 없습니다. 예를 들어, 역할 이름이 development/apps/my-role인 경우 역할에 대한 ARN을 지정할 때 역할 이름을 my-role로 변경해야 합니다. 노드 IAM 역할 ARN(인스턴스 프로파일 ARN이 아님)을 지정해야 합니다.

2. 자체 관리형 노드는 액세스 항목 관리 주제의 사전 조건에 나열된 최소 버전의 플랫폼 버전에 포함되어 있지만, 노드 IAM 역할에 대한 항목이 aws-auth ConfigMap에 나열되지 않거나(이전 항목 참조) 역할에 대한 액세스 항목이 없습니다. 이 문제를 해결하기 위해 다음 명령에서 *my-cluster*를 클러스터 이름으로 바꾸고 수정된 명령(**aws eks list-access-entries --cluster-name *my-cluster***)을 실행하여 기존 액세스 항목을 확인합니다. 다음 명령은 노드의 IAM 역할에 대한 액세스 항목을 추가합니다. *111122223333*을 IAM 역할의 AWS 계정 ID로, *myAmazonEKSNodeRole*을 노드 역할 이름으로 바꿉니다. Windows 노드가 있는 경우 *EC2_Linux*를 **EC2_Windows**로 바꿉니다. 노드 IAM 역할 ARN(인스턴스 프로파일 ARN이 아님)을 지정해야 합니다.

```
aws eks create-access-entry --cluster-name my-cluster --principal-arn
 arn:aws:iam::111122223333:role/myAmazonEKSNodeRole --type EC2_Linux
```

TLS 핸드셰이크 시간 초과

노드에서 퍼블릭 API 서버 엔드포인트에 연결할 수 없는 경우 다음과 유사한 오류가 발생할 수 있습니다.

```
server.go:233] failed to run Kubelet: could not init cloud provider "aws": error
 finding instance i-1111f2222f333e44c: "error listing AWS instances: \"RequestError:
 send request failed\\ncaused by: Post  net/http: TLS handshake timeout\""
```

kubelet 프로세스가 지속적으로 다시 생성되어 API 서버 엔드포인트를 테스트합니다. 이 오류는 구성 변경 또는 버전 업데이트와 같이 컨트롤 플레인에서 클러스터의 롤링 업데이트를 수행하는 절차 중에 일시적으로 발생할 수도 있습니다.

이 문제를 해결하려면 라우팅 테이블과 보안 그룹을 점검하여 노드의 트래픽이 퍼블릭 엔드포인트에 도달할 수 있는지 확인합니다.

InvalidClientTokenId

중국 AWS 리전에 있는 클러스터에 배포된 Pod 또는 DaemonSet의 서비스 계정에 IAM 역할을 사용하고 사양에 AWS_DEFAULT_REGION 환경 변수를 설정하지 않은 경우 Pod 또는 DaemonSet에 다음과 같은 오류가 발생할 수 있습니다.

```
An error occurred (InvalidClientTokenId) when calling the GetCallerIdentity operation:
 The security token included in the request is invalid
```

이 문제를 해결하려면 다음 예의 Pod 사양에 표시된 것처럼 Pod 또는 DaemonSet 사양에 AWS_DEFAULT_REGION 환경 변수를 추가해야 합니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: envar-demo
  labels:
    purpose: demonstrate-envars
spec:
  containers:
  - name: envar-demo-container
    image: gcr.io/google-samples/node-hello:1.0
    env:
    - name: AWS_DEFAULT_REGION
      value: "region-code"
```

VPC 승인 webhook 인증서 만료

VPC 승인 웹후크에 서명하는 데 사용된 인증서가 만료되면 새 Windows Pod 배포의 상태는 ContainerCreating으로 남아 있습니다.

데이터 영역에 레거시 Windows 지원이 있는 경우 이 문제를 해결하려면 VPC 승인 Webhook 인증서 갱신 부분을 참조하세요. 클러스터 및 플랫폼 버전이 Windows 지원 사전 조건에 나열된 버전보다 이후

인 경우 레거시 Windows 지원을 데이터 영역에서 제거하고 컨트롤 플레인에 대해 사용 설정하는 것이 좋습니다. 그러면 웹후크 인증서를 관리할 필요가 없습니다. 자세한 내용은 Amazon EKS 클러스터에 대해 Windows 지원 사용 설정 단원을 참조하십시오.

컨트롤 플레인을 업데이트하기 전에 노드 그룹이 Kubernetes 버전과 일치해야 합니다.

컨트롤 플레인을 새 Kubernetes 버전으로 업그레이드하기 전에 클러스터에 있는 관리형 노드 및 Fargate 노드의 부 버전이 컨트롤 플레인의 현재 버전과 동일해야 합니다. Amazon EKS `update-cluster-version` API는 모든 Amazon EKS 관리형 노드를 현재 클러스터 버전으로 업그레이드할 때까지 요청을 거부합니다. Amazon EKS는 관리형 노드 업그레이드를 위한 API를 제공합니다. 관리형 노드 그룹의 Kubernetes 버전 업그레이드에 대한 자세한 내용은 관리형 노드 그룹 업데이트 섹션을 참조하세요. Fargate 노드의 버전을 업그레이드하려면 컨트롤 플레인을 업그레이드한 후 노드가 나타내는 pod를 삭제하고 pod를 다시 배포합니다. 자세한 내용은 Amazon EKS 클러스터 Kubernetes 버전 업데이트 단원을 참조하십시오.

많은 노드를 시작할 때 **`Too Many Requests`** 오류가 있습니다.

많은 노드를 동시에 시작하는 경우 Amazon EC2 사용자 데이터 실행 로그에 `Too Many Requests`라는 오류 메시지가 표시될 수 있습니다. 컨트롤 플레인이 `describeCluster` 호출로 오버로드되기 때문일 수 있습니다. 오버로드는 제한, 부트스트랩 스크립트를 실행하지 못하는 노드, 클러스터에 참여하지 못하는 노드를 초래합니다.

`--apiserver-endpoint`, `--b64-cluster-ca` 및 `--dns-cluster-ip` 인수가 노드의 부트스트랩 스크립트에 전달되고 있는지 확인합니다. 이러한 인수를 포함하면 부트스트랩 스크립트가 `describeCluster`를 호출할 필요가 없으므로 컨트롤 플레인 오버로드를 방지하는 데 도움이 됩니다. 자세한 내용은 Amazon EKS 최적화 Linux/Bottlerocket AMI에 포함된 `bootstrap.sh` 파일에 인수를 전달하기 위해 사용자 데이터를 제공함 단원을 참조하십시오.

Kubernetes API 서버 요청에 대한 HTTP 401 권한 없음 오류 응답

이러한 오류는 Pod의 서비스 계정 토큰이 클러스터에서 만료된 경우 표시됩니다.

Amazon EKS 클러스터 Kubernetes API 서버는 90일이 지난 토큰을 사용한 요청을 거부합니다. 이전 Kubernetes 버전에는 토큰에 만료 기간이 없었습니다. 즉, 이러한 토큰에 의존하는 클라이언트는 1시간 이내에 토큰을 새로 고쳐야 합니다. Kubernetes API 서버가 유효하지 않은 토큰으로 인해 요청을 거

부하는 것을 방지하려면 워크로드에서 사용하는 Kubernetes 클라이언트 SDK 버전은 다음 버전과 동일하거나 이상 버전이어야 합니다.

- Go 버전 `0.15.7` 이상
- Python 버전 `12.0.0` 이상
- Java 버전 `9.0.0` 이상
- JavaScript 버전 `0.10.3` 이상
- Ruby `master` 브랜치
- Haskell 버전 `0.3.0.0`
- C# 버전 `7.0.5` 이상

오래된 토큰을 사용하는 클러스터의 모든 기존 Pods를 식별할 수 있습니다. 자세한 내용을 알아보려면 Kubernetes 서비스 계정을 참조하세요.

Amazon EKS 플랫폼 버전이 현재 플랫폼 버전보다 두 버전 이상 뒤떨어져 있음

이는 Amazon EKS가 클러스터의 플랫폼 버전을 자동으로 업데이트할 수 없는 경우에 발생할 수 있습니다. 여기에는 여러 가지 원인이 있지만 일반적인 원인 중 일부는 다음과 같습니다. 이러한 문제가 클러스터에 적용되는 경우 여전히 작동할 수 있으며 플랫폼 버전은 Amazon EKS에서 업데이트되지 않습니다.

문제

클러스터 IAM 역할이 삭제됨 – 이 역할은 클러스터가 생성될 때 지정되었습니다. 다음 명령을 사용하여 지정된 역할을 확인할 수 있습니다. *`my-cluster`*를 해당 클러스터의 이름으로 바꿉니다.

```
aws eks describe-cluster --name my-cluster --query cluster.roleArn --output text | cut
 -d / -f 2
```

예제 출력은 다음과 같습니다.

```
eksClusterRole
```

Solution

동일한 이름으로 새 클러스터 IAM 역할을 생성합니다.

문제

클러스터 생성 중에 지정된 서브넷이 삭제됨 - 클러스터와 함께 사용할 서브넷이 클러스터 생성 중에 지정되었습니다. 다음 명령을 사용하여 지정된 서브넷을 확인할 수 있습니다. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다.

```
aws eks describe-cluster --name my-cluster --query cluster.resourcesVpcConfig.subnetIds
```

예제 출력은 다음과 같습니다.

```
[
"subnet-EXAMPLE1",
"subnet-EXAMPLE2"
]
```

Solution

계정에 서브넷 ID가 있는지 확인합니다.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$vpc_id" --query
 "Subnets[*].SubnetId"
```

예제 출력은 다음과 같습니다.

```
[
"subnet-EXAMPLE3",
"subnet-EXAMPLE4"
]
```

출력에 반환된 서브넷 ID가 클러스터 생성 시 지정된 서브넷 ID와 일치하지 않는 경우 Amazon EKS가 클러스터를 업데이트하도록 하려면 클러스터에서 사용하는 서브넷을 변경해야 합니다. 클러스터를 생성할 때 서브넷을 3개 이상 지정한 경우 Amazon EKS는 새로운 탄력적 네트워크 인터페이스를 생성하기 위해 지정한 서브넷을 무작위로 선택하기 때문입니다. 이러한 네트워크 인터페이스를 사용하면 컨트롤 플레인이 노드와 통신할 수 있습니다. Amazon EKS는 선택한 서브넷이 없는 경우 클러스터를 업데이트하지 않습니다. Amazon EKS가 새 네트워크 인터페이스를 생성하기 위해 선택하는 클러스터 생성 시 지정한 서브넷을 컨트롤할 수 없습니다.

클러스터에 대한 Kubernetes 버전 업데이트를 시작하면 같은 이유로 업데이트가 실패할 수 있습니다.

문제

클러스터 생성 중에 지정된 보안 그룹이 삭제됨 - 클러스터 생성 중에 보안 그룹을 지정한 경우 다음 명령을 사용하여 보안 그룹의 ID를 볼 수 있습니다. *my-cluster*를 해당 클러스터의 이름으로 바꿉니다.

```
aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.securityGroupIds
```

예제 출력은 다음과 같습니다.

```
[
    "sg-EXAMPLE1"
]
```

[]이 반환되면 클러스터 생성 시 보안 그룹이 지정되지 않았으며 보안 그룹이 누락된 것이 문제가 아닙니다. 보안 그룹이 반환되면 계정에 보안 그룹이 있는지 확인합니다.

Solution

계정에 이러한 보안 그룹이 있는지 확인합니다.

```
vpc_id=$(aws eks describe-cluster --name my-cluster --query
 cluster.resourcesVpcConfig.vpcId --output text)
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$vpc_id" --query
 "SecurityGroups[*].GroupId"
```

예제 출력은 다음과 같습니다.

```
[
"sg-EXAMPLE2"
]
```

출력에 반환된 보안 그룹 ID가 클러스터 생성 시 지정된 보안 그룹 ID와 일치하지 않는 경우 Amazon EKS가 클러스터를 업데이트하도록 하려면 클러스터에서 사용하는 보안 그룹을 변경해야 합니다. 클러스터 생성 시 지정된 보안 그룹 ID가 없는 경우 Amazon EKS는 클러스터를 업데이트하지 않습니다.

클러스터에 대한 Kubernetes 버전 업데이트를 시작하면 같은 이유로 업데이트가 실패할 수 있습니다.

Amazon EKS가 클러스터의 플랫폼 버전을 업데이트하지 않는 기타 이유

- 클러스터를 생성할 때 지정한 각 서브넷에 최소 6개(16개 권장)의 사용 가능한 IP 주소가 없습니다. 서브넷에 사용 가능한 IP 주소가 충분하지 않은 경우 서브넷에서 IP 주소를 확보하거나 클러스터에서 사용하는 서브넷을 변경하여 가용 IP 주소가 충분한 서브넷을 사용해야 합니다.
- 클러스터를 생성할 때 비밀 암호화를 활성화했으며 지정한 AWS KMS 키가 삭제되었습니다. Amazon EKS가 클러스터를 업데이트하도록 하려면 새 클러스터를 생성해야 합니다.

클러스터 상태 FAQ 및 오류 코드(해결 경로 포함)

Amazon EKS는 EKS 클러스터 및 클러스터 인프라에서 문제를 감지하여 클러스터 상태에 저장합니다. 클러스터 상태 정보를 사용하여 클러스터 문제를 더 빠르게 감지하고 해결할 수 있습니다. 이를 통해 보다 안전하고 최신 상태인 애플리케이션 환경을 구축할 수 있습니다. 또한 필요한 인프라 또는 클러스터 구성 문제로 인해 Amazon EKS에서 성능이 저하된 클러스터에 보안 업데이트를 설치하거나 Kubernetes의 최신 버전으로 업그레이드할 수 없습니다. Amazon EKS에서 문제 자체 또는 문제가 해결되었음을 감지하는 데 3시간이 걸릴 수 있습니다.

Amazon EKS 클러스터의 상태에 대해서는 Amazon EKS와 해당 사용자가 공동으로 책임을 집니다. 사용자는 IAM 역할 및 Amazon VPC 서브넷의 필수 인프라와 사전에 제공해야 하는 기타 필수 인프라를 책임져야 합니다. Amazon EKS는 이 인프라 및 클러스터의 구성 변경을 감지합니다.

Amazon EKS 콘솔에서 클러스터 상태에 액세스하려면 Amazon EKS 클러스터 세부 정보 페이지의 개요 탭에서 상태 문제 섹션을 살펴봅니다. 이 데이터는 EKS API에서 `DescribeCluster` 작업을 직접 호출하여 사용할 수도 있습니다(예: AWS Command Line Interface 내에서 수행).

왜 이 기능을 사용해야 하나요?

디버깅하거나 AWS 지원 사례를 여는 데 시간을 소비하지 않고도 Amazon EKS 클러스터의 상태를 더 잘 파악하고 문제를 신속하게 진단 및 수정할 수 있습니다. 예: 실수로 Amazon EKS 클러스터의 서브넷을 삭제한 경우 Amazon EKS는 크로스 계정 네트워크 인터페이스 및 Kubernetes AWS CLI 명령(예: `kubectl exec`) 또는 `kubectl` 로그를 생성할 수 없습니다. 다음과 같은 오류와 함께 실패합니다. `Error from server: error dialing backend: remote error: tls: internal error.` 이제 다음과 같은 Amazon EKS 상태 문제가 표시됩니다. `subnet-da60e280 was deleted: could not create network interface.`

이 기능은 다른 AWS 서비스와 어떻게 관련되었거나 어떻게 작동하나요?

IAM 역할과 Amazon VPC 서브넷은 클러스터 상태에서 문제를 감지하는 필수 인프라의 두 가지 예입니다. 이 기능은 리소스가 제대로 구성되지 않은 경우 자세한 정보를 반환합니다.

상태 문제가 있는 클러스터에서 요금이 발생하나요?

예. 모든 아마존 EKS 클러스터에는 표준 Amazon EKS 요금이 청구됩니다. 클러스터 상태 기능은 추가 비용 없이 사용할 수 있습니다.

이 기능은 AWS Outposts의 Amazon EKS 클러스터에서 작동하나요?

예. AWS Outposts의 확장 클러스터 및 AWS Outposts의 로컬 클러스터를 포함하여 AWS 클라우드의 EKS 클러스터에서 클러스터 문제가 감지됩니다. 클러스터 상태에서는 Amazon EKS Anywhere 또는 Amazon EKS Distro(EKS-D) 관련 문제를 감지하지 못합니다.

새로운 문제가 감지되면 알림을 받을 수 있나요?

아니요. Amazon EKS 콘솔을 확인하거나 EKS `DescribeCluster` API를 직접 호출해야 합니다.

콘솔에서 상태 문제에 대한 경고를 제공하나요?

예. 상태 문제가 있는 모든 클러스터에는 콘솔 상단에 배너가 포함됩니다.

처음 두 열은 API 응답 값에 필요한 항목입니다. Health ClusterIssue 객체의 세 번째 필드는 resourceIds이며, 반환되는 내용은 문제 유형에 따라 달라집니다.

코드	메시지	ResourceIds	클러스터를 복구할 수 있나요?
SUBNET_NOT_FOUND	현재 클러스터와 연결된 하나 이상의 서브넷을 찾을 수 없습니다. Amazon EKS update-cluster-config API를 직접 호출하여 서브넷을 업데이트합니다.	서브넷 ID	예
SECURITY_GROUP_NOT_FOUND	현재 클러스터와 연결된 하나 이상의 보안 그룹을 찾을 수 없습니다. Amazon EKS update-cluster-config API를 직접 호출하여 보안 그룹 업데이트	보안 그룹 ID	예
IP_NOT_AVAILABLE	클러스터와 연결된 하나 이상의 서브넷에서 Amazon EKS가 클러스터	서브넷 ID	예

코드	메시지	ResourceIds	클러스터를 복구할 수 있나요?	
	관리 작업을 수행하는 데 사용할 수 있는 IP 주소가 충분하지 않습니다. Amazon EKS update-cluster-config API를 사용하여 서브넷의 주소를 확보하거나 다른 서브넷을 클러스터에 연결합니다.			
VPC_NOT_FOUND	클러스터와 연결된 VPC를 찾을 수 없습니다. 클러스터를 삭제하고 다시 생성해야 합니다.	VPC ID	아니요	
ASSUME_ROLE_ACCESS_DENIED	클러스터는 Amazon EKS 서비스 연결 역할을 사용하지 않습니다. 필수 Amazon EKS 관리 작업을 수행할 클러스터와 연결된 관련된 역할을 수임할 수 없습니다. 역할이 존재하고 필요한 신뢰 정책이 있는지 확인합니다.	클러스터 IAM 역할	예	
PERMISSION_ACCESS_DENIED	클러스터는 Amazon EKS 서비스 연결 역할을 사용하지 않습니다. 클러스터와 연결된 역할이 Amazon EKS가 필수 관리 작업을 수행할 수 있는 충분한 권한을 부여하지 않습니다. 클러스터 역할에 연결된 정책을 확인하고 별도의 거부 정책이 적용되어 있는지 확인합니다.	클러스터 IAM 역할	예	
ASSUME_ROLE_ACCESS_DENIED_USING_SLR	Amazon EKS 클러스터 관리 서비스 연결 역할을 수임할 수 없었습니다. 역할이 존재하고 필요한 신뢰 정책이 있는지 확인합니다.	Amazon EKS 서비스 연결 역할	예	

코드	메시지	ResourceIds	클러스터를 복구할 수 있나요?
PERMISSION_ACCESS_DENIED_USING_SLR	Amazon EKS 클러스터 관리 서비스 연결 역할이 Amazon EKS가 필수 관리 작업을 수행할 수 있는 충분한 권한을 부여하지 않습니다. 클러스터 역할에 연결된 정책을 확인하고 별도의 거부 정책이 적용되어 있는지 확인합니다.	Amazon EKS 서비스 연결 역할	예
OPT_IN_REQUIRED	계정에 Amazon EC2 서비스 구독이 없습니다. 계정 설정 페이지에서 계정 구독을 업데이트합니다.	N/A	예
STS_REGIONAL_ENDPOINT_DISABLED	STS 리전 엔드포인트가 비활성화되었습니다. Amazon EKS의 엔드포인트가 필수 클러스터 관리 작업을 수행할 수 있도록 합니다.	N/A	예
KMS_KEY_DISABLED	클러스터와 연결된 AWS KMS 키가 비활성화되었습니다. 키를 다시 활성화하여 클러스터를 복구합니다.	KMS Key Arn은	예
KMS_KEY_NOT_FOUND	클러스터와 연결된 AWS KMS 키를 찾을 수 없습니다. 클러스터를 삭제하고 다시 생성해야 합니다.	KMS Key ARN은	아니요
KMS_GRANT_REVOKED	클러스터와 연결된 AWS KMS 키에 부여된 권한이 취소됩니다. 클러스터를 삭제하고 다시 생성해야 합니다.	KMS Key Arn은	아니요

Amazon EKS Connector

Amazon EKS Connector를 사용하여 적합한 Kubernetes 클러스터를 AWS에 등록하고 연결할 수 있으며 Amazon EKS 콘솔에서 시각화할 수 있습니다. 클러스터가 연결되면 Amazon EKS 콘솔에서 해당 클러스터의 상태, 구성 및 워크로드를 확인할 수 있습니다. 이 기능을 사용하여 Amazon EKS 콘솔에서 연결된 클러스터를 볼 수 있지만 관리할 수는 없습니다. Amazon EKS Connector에는 Github의 오픈 소스 프로젝트인 에이전트가 필요합니다. FAQ 및 문제 해결을 포함한 추가 기술 콘텐츠는 Amazon EKS 커넥터의 문제 해결 섹션을 참조하세요.

Amazon EKS Connector는 다음 유형의 Kubernetes 클러스터를 Amazon EKS에 연결할 수 있습니다.

- 온프레미스 Kubernetes 클러스터
- Amazon EC2에서 실행 중인 자체 관리형 클러스터
- 다른 클라우드 공급자의 관리형 클러스터

Amazon EKS Connector 고려 사항

Amazon EKS Connector를 사용하기 전에 다음 사항에 대한 이해가 필요합니다.

- 클러스터를 Amazon EKS에 연결하려면 Kubernetes 클러스터에 대한 관리 권한이 있어야 합니다.
- Kubernetes 클러스터에는 연결하기 전에 Linux 64비트(x86) 워커 노드가 있어야 합니다. ARM 작업자 노드는 지원되지 않습니다.
- Kubernetes 클러스터에 `ssm.` 및 `ssmmessages.` 시스템 관리자 엔드포인트에 대한 아웃바운드 액세스 권한이 있는 워커 노드가 있어야 합니다. 자세한 내용은 AWS 일반 참조의 Systems Manager 엔드포인트를 참조하세요.
- 기본적으로 리전에서 최대 10개의 클러스터를 연결할 수 있습니다. 서비스 할당량 콘솔을 통해 증가를 요청할 수 있습니다. 자세한 내용은 할당량 증가 요청을 참조하세요.
- Amazon EKS `RegisterCluster`, `ListClusters`, `DescribeCluster`, 및 `DeregisterCluster` API만 외부 Kubernetes 클러스터에 대해 지원됩니다.
- 클러스터를 등록하려면 다음 권한이 있어야 합니다.
  - eks:RegisterCluster
  - ssm:CreateActivation
  - ssm:DeleteActivation

  - iam:PassRole
- 클러스터를 등록 취소하려면 다음 권한이 있어야 합니다.
  - eks:DeregisterCluster
  - ssm:DeleteActivation
  - ssm:DeregisterManagedInstance

Amazon EKS Connector에 필요한 IAM 역할

Amazon EKS Connector를 사용하려면 다음 두 가지 IAM 역할이 필요합니다.

- Amazon EKS Connector 서비스 연결 역할은 클러스터를 처음 등록할 때 생성됩니다.
- Amazon EKS Connector 에이전트 IAM 역할을 생성해야 합니다. 세부 정보는 Amazon EKS Connector IAM 역할을 참조하십시오.

IAM 보안 주체에 대한 클러스터 및 워크로드 보기 권한을 사용 설정하려면 클러스터에 `eks-connector` 및 Amazon EKS Connector 클러스터 역할을 적용해야 합니다. 클러스터에서 Kubernetes 리소스를 보도록 IAM 보안 주체에게 액세스 권한 부여 단원의 단계를 따르십시오.

외부 클러스터 연결

다음 프로세스에서 여러 방법을 사용하여 외부 Kubernetes 클러스터를 Amazon EKS에 연결할 수 있습니다. 이 프로세스에는 두 단계로 이루어집니다. Amazon EKS에 클러스터를 등록하고 클러스터에 `eks-connector` 에이전트를 설치하는 것입니다.

> **Important**
>
> 첫 번째 단계를 완료한 후 등록이 만료되는 3일 이내에 두 번째 단계를 완료해야 합니다.

커넥터 방법

클러스터를 등록하는 각 방법 이후에 에이전트를 설치하는 모든 방법을 사용할 수 있는 것은 아닙니다. 다음 표에는 각 등록 방법과 사용할 수 있는 에이전트 설치 방법이 나와 있습니다.

단계	메서드		
클러스터 등록	AWS Management Console	AWS Command Line Interface	`eksctl`
에이전트 설치	Helm, YAML 매니페스트	Helm, YAML 매니페스트	YAML 매니페스트

사전 조건

- Amazon EKS Connector 에이전트 역할이 생성되었는지 확인합니다. Amazon EKS Connector 에이전트 역할 생성 섹션의 단계를 따릅니다.
- 클러스터를 등록하려면 다음 권한이 있어야 합니다.
  - `eks:RegisterCluster`
  - `ssm:CreateActivation`
  - `ssm:DeleteActivation`
  - `iam:PassRole`

1단계: 클러스터 등록

AWS CLI

사전 조건

- AWS CLI가 설치되어 있어야 합니다. 설치하거나 업그레이드하려면 AWS CLI 설치를 참조하세요.

클러스터를 AWS CLI에 등록하려면

- 커넥터 구성의 경우 Amazon EKS Connector 에이전트 IAM 역할을 지정합니다. 자세한 내용은 Amazon EKS Connector에 필요한 IAM 역할 섹션을 참조하세요.

```
aws eks register-cluster \
     --name my-first-registered-cluster \
     --connector-config roleArn=arn:aws:iam::111122223333:role/
AmazonEKSConnectorAgentRole,provider="OTHER" \
```

```
    --region aws-region
```

예제 출력은 다음과 같습니다.

```
{
    "cluster": {
        "name": "my-first-registered-cluster",
        "arn": "arn:aws:eks:region:111122223333:cluster/my-first-registered-
cluster",
        "createdAt": 1627669203.531,
        "ConnectorConfig": {
            "activationId": "xxxxxxxxACTIVATION_IDxxxxxxxx",
            "activationCode": "xxxxxxxxACTIVATION_CODExxxxxxxx",
            "activationExpiry": 1627672543.0,
            "provider": "OTHER",
            "roleArn": "arn:aws:iam::111122223333:role/
AmazonEKSConnectorAgentRole"
        },
        "status": "CREATING"
    }
}
```

다음 단계에서 `aws-region`, `activationId` 및 `activationCode` 값을 사용합니다.

AWS Management Console

콘솔에 Kubernetes 클러스터를 등록하려면 다음을 수행합니다.

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 클러스터 추가(Add cluster)를 선택하고 등록(Register)을 선택하여 구성 페이지를 불러옵니다.
3. 클러스터 구성(Configure cluster) 페이지에서 다음 필드를 작성합니다.
   - 이름 - 클러스터에 대한 고유한 이름
   - 공급자(Provider) - 선택하여 Kubernetes 클러스터 공급자의 드롭다운 목록을 표시합니다. 특정 공급자를 모르는 경우 기타를 선택합니다.
   - EKS Connector 역할(EKS Connector role) - 클러스터 연결에 사용할 역할을 선택합니다.
4. 클러스터 등록(Register cluster)을 선택합니다.

5. 클러스터 개요 페이지가 표시됩니다. 차트 Helm을 사용하려면 `helm install` 명령을 복사하고 다음 단계로 넘어갑니다. YAML 매니페스트를 사용하려면 YAML 파일 다운로드를 선택하여 매니페스트 파일을 로컬 드라이브에 다운로드합니다.

   > **Important**
   >
   > - `helm install` 명령을 복사하거나 이 파일을 다운로드할 수 있는 유일한 기회입니다. 링크에 액세스할 수 없으며 클러스터를 등록 취소하고 처음부터 단계를 시작해야 하므로 이 페이지에서 벗어나지 마세요.
   > - 명령 또는 매니페스트 파일은 등록된 클러스터에 대해 한 번만 사용할 수 있습니다. Kubernetes 클러스터에서 리소스를 삭제하는 경우 클러스터를 다시 등록하고 새 매니페스트 파일을 가져와야 합니다.

   매니페스트 파일을 Kubernetes 클러스터에 적용하려면 다음 단계로 이동합니다.

eksctl

사전 조건

- `eksctl` 버전 `0.68` 이상을 설치해야 합니다. 설치하거나 업그레이드하려면 Amazon EKS 시작하기 - `eksctl` 섹션을 참조하세요.

클러스터를 **`eksctl`**에 등록하려면 다음을 수행합니다.

1. 이름, 공급자 및 리전을 제공하여 클러스터를 등록합니다.

```
eksctl register cluster --name my-cluster --provider my-provider --
region region-code
```

출력 예제:

```
2021-08-19 13:47:26 [#]  creating IAM role "eksctl-20210819194112186040"
2021-08-19 13:47:26 [#]  registered cluster "<name>" successfully
2021-08-19 13:47:26 [#]  wrote file eks-connector.yaml to <current directory>
2021-08-19 13:47:26 [#]  wrote file eks-connector-clusterrole.yaml to <current
 directory>
```

```
2021-08-19 13:47:26 [#]  wrote file eks-connector-console-dashboard-full-access-
group.yaml to <current directory>
2021-08-19 13:47:26 [!]  note: "eks-connector-clusterrole.yaml" and "eks-
connector-console-dashboard-full-access-group.yaml" give full EKS Console access
 to IAM identity "<aws-arn>", edit if required; read https://eksctl.io/usage/
eks-connector for more info
2021-08-19 13:47:26 [#]  run `kubectl apply -f eks-connector.yaml,eks-connector-
clusterrole.yaml,eks-connector-console-dashboard-full-access-group.yaml` before
 expiry> to connect the cluster
```

이렇게 하면 로컬 컴퓨터에 파일이 생성됩니다. 이 파일을 3일 이내에 외부 클러스터에 적용해야 하며, 그러지 않으면 등록이 만료됩니다.

2. 클러스터에 액세스할 수 있는 터미널에서 `eks-connector-binding.yaml` 파일 적용:

```
kubectl apply -f eks-connector-binding.yaml
```

2단계: **eks-connector** 에이전트 설치

Helm chart

1. 이전 단계에서 AWS CLI를 사용한 경우 다음 명령에서 `ACTIVATION_CODE` 및 `ACTIVATION_ID`를 각각 `activationId` 및 `activationCode` 값으로 대체합니다. `aws-region`을 이전 단계에서 사용한 AWS 리전으로 바꿉니다. 그런 다음 명령을 실행하여 등록하는 클러스터에 `eks-connector` 에이전트를 설치합니다.

```
$ helm install eks-connector \
  --namespace eks-connector \
  oci://public.ecr.aws/eks-connector/eks-connector-chart \
  --set eks.activationCode=ACTIVATION_CODE \
  --set eks.activationId=ACTIVATION_ID \
  --set eks.agentRegion=aws-region
```

이전 단계에서 AWS Management Console을 사용한 경우 이전 단계에서 복사하고 이러한 값이 채워진 명령을 사용합니다.

2. 설치된 `eks-connector` 배포의 상태를 확인하고 Amazon EKS에 등록된 클러스터의 상태가 `ACTIVE`가 될 때까지 기다립니다.

YAML manifest

Amazon EKS Connector 매니페스트 파일을 Kubernetes 클러스터에 적용하여 연결 작업을 완료합니다. 이렇게 하려면 앞서 설명한 방법을 사용해야 합니다. 3일 이내에 매니페스트가 적용되지 않으면 Amazon EKS Connector 등록이 만료됩니다. 클러스터 연결이 만료되면 먼저 클러스터의 등록을 취소한 후 클러스터를 다시 연결해야 합니다.

1. Amazon EKS Connector YAML 파일을 다운로드합니다.

```
curl -O https://amazon-eks.s3.us-west-2.amazonaws.com/eks-connector/manifests/
eks-connector/latest/eks-connector.yaml
```

2. Amazon EKS Connector YAML 파일을 편집하여 %AWS_REGION%, %EKS_ACTIVATION_ID%, %EKS_ACTIVATION_CODE%의 모든 참조를 이전 단계 출력의 aws-region, activationId 및 activationCode로 바꿉니다.

   다음 예제 명령은 이러한 값을 대체할 수 있습니다.

```
sed -i "s~%AWS_REGION%~$aws-region~g; s~%EKS_ACTIVATION_ID
%~$EKS_ACTIVATION_ID~g; s~%EKS_ACTIVATION_CODE%~$(echo -n $EKS_ACTIVATION_CODE |
 base64)~g" eks-connector.yaml
```

   > **Important**
   >
   > 정품 인증 코드가 base64 형식인지 확인합니다.

3. 클러스터에 액세스할 수 있는 터미널에서 다음 명령을 실행하여 업데이트된 매니페스트 파일을 적용할 수 있습니다.

```
kubectl apply -f eks-connector.yaml
```

4. Amazon EKS Connector 매니페스트 및 역할 바인딩 YAML 파일이 Kubernetes 클러스터에 적용된 후 클러스터가 이제 연결되었는지 확인합니다.

```
aws eks describe-cluster \
     --name "my-first-registered-cluster" \
     --region AWS_REGION
```

   출력에 status=ACTIVE가 포함되어야 합니다.

5. (선택 사항) 클러스터에 태그를 추가합니다. 자세한 내용은 Amazon EKS 리소스 태깅 섹션을 참조하세요.

다음 단계

이 단계에서 문제가 있는 경우 Amazon EKS 커넥터의 문제 해결의 내용을 참조하세요.

연결된 클러스터에서 Kubernetes 리소스를 볼 수 있도록 추가 IAM 보안 주체에게 Amazon EKS 콘솔에 대한 액세스 권한을 부여하려면 클러스터에서 Kubernetes 리소스를 보도록 IAM 보안 주체에게 액세스 권한 부여 섹션을 참조하세요.

클러스터에서 Kubernetes 리소스를 보도록 IAM 보안 주체에게 액세스 권한 부여

IAM 보안 주체에게 Amazon EKS 콘솔에 대한 액세스 권한을 부여하여 연결된 클러스터에서 실행 중인 Kubernetes 리소스에 대한 정보를 봅니다.

사전 조건

AWS Management Console 액세스에 사용하는 IAM 보안 주체에서는 다음과 같은 요구 사항을 충족해야 합니다.

- `eks:AccessKubernetesApi` IAM 권한이 있어야 합니다.
- Amazon EKS Connector 서비스 계정은 클러스터에서 IAM 보안 주체를 가장할 수 있습니다. 그러면 Amazon EKS Connector에서 IAM 보안 주체를 Kubernetes 사용자에게 매핑할 수 있습니다.

Amazon EKS Connector 클러스터 역할을 생성하고 적용하려면

1. `eks-connector` 클러스터 역할 템플릿을 다운로드합니다.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-clusterrole.yaml
```

2. 클러스터 역할 템플릿 YAML 파일을 편집합니다. `%IAM_ARN%`의 참조를 IAM 보안 주체의 Amazon 리소스 이름(ARN)으로 바꿉니다.
3. Amazon EKS Connector 클러스터 역할 YAML을 Kubernetes 클러스터에 적용합니다.

```
kubectl apply -f eks-connector-clusterrole.yaml
```

IAM 보안 주체가 Amazon EKS 콘솔에서 Kubernetes 리소스를 시각화하려면 보안 주체가 이러한 리소스를 읽는 데 필요한 권한을 통해 Kubernetes Kubernetes `role` 또는 `clusterrole`과 연결되어 있어야 합니다. 자세한 내용은 Kubernetes 설명서의 RBAC 승인 사용을 참조하세요.

연결된 클러스터에 액세스하도록 IAM 보안 주체를 구성하려면

1. 예제 매니페스트 파일을 다운로드하여 `clusterrole`과 `clusterrolebinding` 또는 `role`과 `rolebinding`을 각각 생성할 수 있습니다.

   모든 네임스페이스의 Kubernetes 리소스 보기

   `eks-connector-console-dashboard-full-access-clusterrole` 클러스터 역할을 사용하면 콘솔에서 시각화할 수 있는 모든 네임스페이스와 리소스에 액세스할 수 있습니다. 먼저 `role`, `clusterrole` 및 해당 바인딩의 이름을 변경해야 이를 클러스터에 적용할 수 있습니다. 다음 명령을 사용하여 샘플 파일을 다운로드합니다.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-full-access-group.yaml
   ```

   특정 네임스페이스의 Kubernetes 리소스 보기

   이 파일의 네임스페이스는 `default`이므로 다른 네임스페이스를 지정하려면 클러스터에 적용하기 전에 파일을 편집합니다. 다음 명령을 사용하여 샘플 파일을 다운로드합니다.

   ```
   curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/eks-connector/manifests/eks-connector-console-roles/eks-connector-console-dashboard-restricted-access-group.yaml
   ```

2. 전체 액세스 또는 제한된 액세스 YAML 파일을 편집하고, `%IAM_ARN%`의 참조를 IAM 보안 주체의 Amazon 리소스 이름(ARN)으로 바꿉니다.
3. Kubernetes 클러스터에 전체 액세스 또는 제한된 액세스 YAML 파일을 적용합니다. YAML 파일 값을 사용자의 고유한 값으로 교체합니다.

   ```
   kubectl apply -f eks-connector-console-dashboard-full-access-group.yaml
   ```

연결된 클러스터에서 Kubernetes 리소스를 보려면 Kubernetes 리소스 보기 섹션을 참조하세요. 리소스 탭에서 일부 리소스 유형에 대한 데이터는 연결된 클러스터에서 사용할 수 없습니다.

클러스터 등록 취소

연결된 클러스터 사용을 마치면 등록 취소할 수 있습니다. 등록이 취소된 후에는 클러스터가 Amazon EKS 콘솔에 더이상 표시되지 않습니다.

deregisterCluster API를 호출하려면 다음 권한이 있어야 합니다.

- `eks:DeregisterCluster`
- `ssm:DeleteActivation`
- `ssm:DeregisterManagedInstance`

이 프로세스는 두 단계로 이루어집니다. Amazon EKS에서 클러스터 등록을 취소하고 클러스터에서 eks-connector 에이전트를 제거하는 것입니다.

Kubernetes 클러스터 등록 취소

AWS CLI

사전 조건

- AWS CLI가 설치되어 있어야 합니다. 설치하거나 업그레이드하려면 AWS CLI 설치를 참조하세요.
- Amazon EKS Connector 에이전트 IAM 역할이 생성되었는지 확인하세요.

연결된 클러스터의 등록을 취소합니다.

```
aws eks deregister-cluster \
    --name my-cluster \
    --region region-code
```

AWS Management Console

1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
2. 클러스터를 선택하십시오.

3. 클러스터 페이지에서 연결된 클러스터를 선택한 다음 등록 취소를 선택합니다.
4. 클러스터 등록을 취소할지 확인합니다.

eksctl

사전 조건

- `eksctl` 버전 `0.68` 이상을 설치해야 합니다. 설치하거나 업그레이드하려면 Amazon EKS 시작하기 - `eksctl` 섹션을 참조하세요.
- Amazon EKS Connector 에이전트 IAM 역할이 생성되었는지 확인하세요.

`eksctl`을 사용하여 클러스터를 등록 취소하려면 다음을 수행합니다.

- 커넥터 구성의 경우 Amazon EKS Connector 에이전트 IAM 역할을 지정합니다. 자세한 내용은 Amazon EKS Connector에 필요한 IAM 역할 섹션을 참조하세요.

```
eksctl deregister cluster --name my-cluster
```

Kubernetes 클러스터의 리소스 정리

Helm

- 다음 명령을 실행하여 에이전트를 제거합니다.

```
helm -n eks-connector uninstall eks-connector
```

YAML manifest

1. Kubernetes 클러스터에서 Amazon EKS Connector YAML 파일을 삭제합니다.

```
kubectl delete -f eks-connector.yaml
```

2. 추가 IAM 보안 주체가 클러스터에 액세스할 수 있도록 `clusterrole` 또는 `clusterrolebindings`을 만든 경우 Kubernetes 클러스터에서 삭제해야 합니다.

Amazon EKS 커넥터의 문제 해결

이 주제에서는 해결 방법에 대한 지침과 해결 방법을 포함하여 Amazon EKS 커넥터를 사용하는 동안 발생할 수 있는 몇 가지 일반적인 오류를 설명합니다.

기본 문제 해결

이 섹션에서는 명확하지 않은 문제를 진단하는 단계를 설명합니다.

Amazon EKS 커넥터 상태 확인

Amazon EKS 커넥터 상태를 확인합니다.

```
kubectl get pods -n eks-connector
```

Amazon EKS 커넥터 로그 검사

Amazon EKS 커넥터 Pod는 3개의 컨테이너로 구성됩니다. 이러한 컨테이너를 모두 검사할 수 있도록 전체 로그를 검색하려면 다음 명령을 실행합니다.

- `connector-init`

```
kubectl logs eks-connector-0 --container connector-init -n eks-connector
kubectl logs eks-connector-1 --container connector-init -n eks-connector
```

- `connector-proxy`

```
kubectl logs eks-connector-0 --container connector-proxy -n eks-connector
kubectl logs eks-connector-1 --container connector-proxy -n eks-connector
```

- `connector-agent`

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
kubectl exec eks-connector-1 --container connector-agent -n eks-connector -- cat /
var/log/amazon/ssm/amazon-ssm-agent.log
```

유효 클러스터 이름 가져오기

Amazon EKS 클러스터는 단일 AWS 계정 및 AWS 리전 내에서 `clusterName`으로 고유하게 식별됩니다. Amazon EKS에 연결된 클러스터가 여러 개 있는 경우 현재 Kubernetes 클러스터가 등록된 Amazon EKS 클러스터를 확인할 수 있습니다. 이렇게 하려면 다음을 입력하여 현재 클러스터의 `clusterName`을 찾습니다.

```
kubectl exec eks-connector-0 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
kubectl exec eks-connector-1 --container connector-agent -n eks-connector \
  -- cat /var/log/amazon/ssm/amazon-ssm-agent.log | grep -m1 -oE "eks_c:[a-zA-Z0-9_-]+"
 | sed -E "s/^.*eks_c:([a-zA-Z0-9_-]+)_[a-zA-Z0-9]+.*$/\1/"
```

기타 명령

다음 명령은 문제를 해결하는 데 필요한 정보를 검색하는 데 유용합니다.

- 다음 명령을 사용하여 Amazon EKS 커넥터의 Pods에서 사용하는 이미지를 수집합니다.

  ```
  kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.containers[*].image}"
   | tr -s '[[:space:]]' '\n'
  ```

- 다음과 같은 명령을 사용하여 Amazon EKS 커넥터가 실행 중인 노드 이름을 확인합니다.

  ```
  kubectl get pods -n eks-connector -o jsonpath="{.items[*].spec.nodeName}" | tr -s
   '[[:space:]]' '\n'
  ```

- 다음 명령을 실행하여 Kubernetes 클라이언트 및 서버 버전을 가져옵니다.

  ```
  kubectl version
  ```

- 다음과 같은 명령을 실행하여 노드에 대한 정보를 가져옵니다.

  ```
  kubectl get nodes -o wide --show-labels
  ```

Helm 문제: 403 Forbidden

helm install 명령을 실행할 때 다음 오류가 발생하는 경우:

```
Error: INSTALLATION FAILED: unexpected status from HEAD request to https://
public.ecr.aws/v2/eks-connector/eks-connector-chart/manifests/0.0.6: 403 Forbidden
```

다음 줄을 실행하여 문제를 해결할 수 있습니다.

```
docker logout public.ecr.aws
```

콘솔 오류: 클러스터가 보류 중 상태에서 멈춰 있음

클러스터를 등록한 후 Amazon EKS 콘솔에서 클러스터가 Pending 상태로 멈추는 경우 Amazon EKS 커넥터가 아직 클러스터를 AWS에 성공적으로 연결하지 않았기 때문일 수 있습니다. 등록된 클러스터의 경우 Pending 상태는 연결이 성공적으로 설정되지 않았음을 의미합니다. 이 문제를 해결하려면 대상 Kubernetes 클러스터에 매니페스트를 적용했는지 확인하세요. 클러스터에 적용했지만 클러스터가 여전히 Pending 상태인 경우에는 eks-connector statefulset가 비정상 상태일 수 있습니다. 이 문제를 해결하려면 이 주제의 Amazon EKS 커넥터 Pods가 크래시 반복됨 섹션을 참조하세요.

콘솔 오류: 클러스터 범위의 **User “system:serviceaccount:eks-connector:eks-connector” can't impersonate resource “users” in API group “”**

Amazon EKS 커넥터는 Kubernetes 사용자 가장을 사용하여 AWS Management Console에서 IAM 보안 주체를 대신합니다. AWS eks-connector 서비스 계정에서 Kubernetes API에 액세스하는 각 보안 주체에 IAM ARN을 Kubernetes 사용자 이름으로 사용하여 해당 Kubernetes 사용자를 가장할 수 있는 권한이 부여되어야 합니다. 다음 예에서 IAM ARN은 Kubernetes 사용자에게 매핑됩니다.

- AWS 계정 *111122223333*의 IAM 사용자 *john*은 Kubernetes 사용자에게 매핑됩니다. IAM 모범 사례에서는 되도록 사용자 대신 역할에 권한을 부여하세요.

  ```
  arn:aws:iam::111122223333:user/john
  ```

- AWS 계정 *111122223333*의 IAM 역할 *admin*은 Kubernetes 사용자에게 매핑됩니다.

  ```
  arn:aws:iam::111122223333:role/admin
  ```

  결과는 AWS STS 세션 ARN 대신 IAM 역할 ARN입니다.

매핑된 사용자를 가장할 수 있는 eks-connector 서비스 계정 권한을 부여하도록 ClusterRole 및 ClusterRoleBinding을 구성하는 방법에 대한 지침은 클러스터에서 Kubernetes 리소스를 보도록 IAM 보안 주체에게 액세스 권한 부여 섹션을 참조하세요. 템플릿에서 %IAM_ARN%가 AWS Management Console IAM 보안 주체의 IAM ARN으로 바뀌었는지 확인합니다.

콘솔 오류: 클러스터 범위의 **[...] is forbidden: User [...] cannot list resource "[...] in API group"**

다음 문제를 고려하세요. Amazon EKS 커넥터가 대상 Kubernetes 클러스터에서 요청하는 AWS Management Console IAM 보안 주체를 성공적으로 가장했습니다. 그러나 가장된 보안 주체에게 Kubernetes API 작업에 대한 RBAC 권한이 없습니다.

이 문제를 해결하려면 두 가지 방법으로 추가 사용자에게 권한을 부여할 수 있습니다. 이전에 차트 Helm을 통해 eks-connector를 설치한 경우 다음 명령을 실행하여 사용자에게 액세스 권한을 쉽게 부여할 수 있습니다. userARN1 및 userARN2를 Kubernetes 리소스에 대한 보기 권한을 부여하는 IAM 역할의 ARN 목록으로 대체합니다.

```
helm upgrade eks-connector oci://public.ecr.aws/eks-connector/eks-connector-chart \
    --reuse-values \
    --set 'authentication.allowedUserARNs={userARN1,userARN2}'
```

클러스터 관리자로서 개별 Kubernetes 사용자에게 적절한 수준의 RBAC 권한을 부여하세요. 자세한 정보와 지침은 클러스터에서 Kubernetes 리소스를 보도록 IAM 보안 주체에게 액세스 권한 부여 섹션을 참조하세요.

콘솔 오류: Amazon EKS가Kubernetes 클러스터 API 서버와 통신할 수 없습니다. 연결에 성공하려면 클러스터가 ACTIVE 상태여야 합니다. 몇 분 후에 다시 시도하세요.

Amazon EKS 서비스가 대상 클러스터의 Amazon EKS 커넥터와 통신할 수 없는 경우 다음과 같은 사유 중 하나 때문일 수 있습니다.

- 대상 클러스터의 Amazon EKS 커넥터가 비정상입니다.
- 대상 클러스터와 AWS 리전 간의 연결 상태가 좋지 않거나 연결이 중단되었습니다.

이 문제를 해결하려면 Amazon EKS 커넥터 로그를 확인하세요. Amazon EKS 커넥터에 대한 오류가 표시되지 않으면 몇 분 후에 연결을 다시 시도하세요. 대상 클러스터에 대해 긴 대기 시간 또는 간헐적인 연결이 정기적으로 발생하는 경우 가까운 AWS 리전에 클러스터를 다시 등록하는 것이 좋습니다.

Amazon EKS 커넥터 Pods가 크래시 반복됨

여러 가지 사유로 Amazon EKS 커넥터 Pod가 `CrashLoopBackOff` 상태로 전환될 수 있습니다. 이 문제는 `connector-init` 컨테이너와 관련이 있을 수 있습니다. Amazon EKS 커넥터 Pod의 상태를 확인합니다.

```
kubectl get pods -n eks-connector
```

예제 출력은 다음과 같습니다.

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:CrashLoopBackOff   1          7s
```

출력이 이전 출력과 비슷한 경우 Amazon EKS 커넥터 로그 검사의 내용을 참조하여 문제를 해결합니다.

Failed to initiate eks-connector: InvalidActivation

Amazon EKS 커넥터를 처음 시작할 때 Amazon Web Services에 `activationId` 및 `activationCode`를 등록합니다. 등록에 실패하여 다음과 같은 오류와 비슷한 오류로 `connector-init` 컨테이너가 충돌할 수 있습니다.

```
F1116 20:30:47.261469       1 init.go:43] failed to initiate eks-connector:
 InvalidActivation:
```

이 문제를 해결하려면 다음 원인과 권장 수정 사항을 고려하세요.

- `activationId` 및 `activationCode`가 매니페스트 파일에 없기 때문에 등록에 실패했을 수 있습니다. 이 경우 `RegisterCluster` API 작업에서 반환된 올바른 값이고 `activationCode`가 매니페스트 파일에 있는지 확인합니다. `activationCode`는 Kubernetes 보안 암호에 추가되므로 `base64`로 인코딩되어야 합니다. 자세한 내용은 1단계: 클러스터 등록 단원을 참조하십시오.
- 정품 인증이 만료되어 등록에 실패했을 수 있습니다. 보안상의 이유로 클러스터를 등록한 후 3일 이내에 Amazon EKS 커넥터를 활성화해야 하기 때문입니다. 이 문제를 해결하려면 만료 날짜 및 시간 전에 Amazon EKS 커넥터 매니페스트가 대상 Kubernetes 클러스터에 적용되었는지 확인하세요. 정품 인증 만료 날짜를 확인하려면 `DescribeCluster` API 작업을 호출하세요.

```
aws eks describe-cluster --name my-cluster
```

다음과 같은 예시 응답에서는 만료 날짜 및 시간이 2021-11-12T22:28:51.101000-08:00으로 기록됩니다.

```
{
    "cluster": {
        "name": "my-cluster",
        "arn": "arn:aws:eks:region:111122223333:cluster/my-cluster",
        "createdAt": "2021-11-09T22:28:51.449000-08:00",
        "status": "FAILED",
        "tags": {
        },
        "connectorConfig": {
            "activationId": "00000000-0000-0000-0000-000000000000",
            "activationExpiry": "2021-11-12T22:28:51.101000-08:00",
            "provider": "OTHER",
            "roleArn": "arn:aws:iam::111122223333:role/my-connector-role"
        }
    }
}
```

activationExpiry가 지나면 클러스터를 등록 취소하고 다시 등록합니다. 이렇게 하면 새 정품 인증이 생성됩니다.

클러스터 노드에 아웃바운드 연결이 없습니다.

제대로 작동하려면 Amazon EKS 커넥터에 여러 AWS 엔드포인트에 대한 아웃바운드 연결이 필요합니다. 대상 AWS 리전에 대한 아웃바운드 연결 없이 프라이빗 클러스터를 연결할 수 없습니다. 이 문제를 해결하려면 필요한 아웃바운드 연결을 추가해야 합니다. 커넥터 요구 사항에 대한 자세한 내용은 Amazon EKS Connector 고려 사항 단원을 참조하십시오.

Amazon EKS 커넥터 Pods의 상태가 **ImagePullBackOff**입니다.

get pods 명령을 실행하고 Pods의 상태가 ImagePullBackOff이면 제대로 작동하지 않습니다. Amazon EKS 커넥터 Pods의 상태가 ImagePullBackOff이면 제대로 작동하지 않습니다. Amazon EKS 커넥터 Pods의 상태를 확인합니다.

```
kubectl get pods -n eks-connector
```

예제 출력은 다음과 같습니다.

```
NAME              READY   STATUS                  RESTARTS   AGE
eks-connector-0   0/2     Init:ImagePullBackOff   0          4s
```

기본 Amazon EKS 커넥터 매니페스트 파일은 Amazon ECR Public Gallery의 이미지를 참조합니다. 대상 Kubernetes 클러스터가 Amazon ECR Public Gallery에서 이미지를 가져올 수 없을 가능성이 있습니다. Amazon ECR Public Gallery 이미지 풀 문제를 해결하거나 선택한 프라이빗 컨테이너 레지스트리의 이미지 미러링을 고려하세요.

FAQ

Q: Amazon EKS 커넥터의 기본 기술은 어떻게 작동하나요?

A: Amazon EKS 커넥터는 AWS Systems Manager(System Manager) 에이전트를 기반으로 합니다. Amazon EKS 커넥터는 Kubernetes 클러스터에서 `StatefulSet`로 실행됩니다. 연결을 설정하고 클러스터의 API 서버와 Amazon Web Services 간의 통신을 프록시합니다. EKS 커넥터는 AWS에서 클러스터 연결을 끊을 때까지 Amazon EKS 콘솔에 클러스터 데이터를 표시하기 위해 이 작업을 수행합니다. Systems Manager 에이전트는 오픈 소스 프로젝트입니다. 이 프로젝트에 대한 자세한 내용은 GitHub 프로젝트 페이지를 참조하세요.

Q: 연결하려는 온프레미스 Kubernetes 클러스터가 있습니다. 연결하려면 방화벽 포트를 열어야 하나요?

A: 아니요, 방화벽 포트를 열 필요가 없습니다. Kubernetes 클러스터는 AWS 리전에 대한 아웃바운드 연결만 필요합니다. AWS 서비스는 온프레미스 네트워크의 리소스에 절대 액세스하지 않습니다. Amazon EKS 커넥터는 클러스터에서 실행되고 AWS에 대한 연결을 시작합니다. 클러스터 등록이 완료되면 AWS는 클러스터에 있는 Kubernetes API 서버의 정보가 필요한 작업을 Amazon EKS 콘솔에서 시작한 후에만 Amazon EKS 커넥터에 명령을 실행합니다.

Q: Amazon EKS 커넥터가 내 클러스터에서 AWS로 어떤 데이터를 전송하나요?

A: Amazon EKS 커넥터는 클러스터를 AWS에 등록하는 데 필요한 기술 정보를 전송합니다. 또한 고객이 요청하는 Amazon EKS 콘솔 기능에 대한 클러스터 및 워크로드 메타데이터를 전송합니다. Amazon EKS 커넥터는 데이터를 AWS로 전송해야 하는 Amazon EKS 콘솔 또는 Amazon EKS API에서 작업을 시작하는 경우에만 이 데이터를 수집하거나 전송합니다. Kubernetes 버전 번호 외에 AWS는 기본적으로 데이터를 저장하지 않습니다. 권한을 부여받은 경우에만 데이터를 저장합니다.

Q: AWS 리전 외부의 클러스터를 연결할 수 있나요?

A: 예. 어느 위치에서든 클러스터를 Amazon EKS에 연결할 수 있습니다. 또한 Amazon EKS 서비스는 모든 AWS 퍼블릭 상용 AWS 리전에 위치할 수 있습니다. 이는 클러스터에서 대상 AWS 리전으로의 유효한 네트워크 연결에서 작동합니다. UI 성능 최적화를 위해 클러스터 위치에서 가장 가까운 AWS 리전을 선택하는 것이 좋습니다. 예를 들어 도쿄에서 실행 중인 클러스터가 있는 경우 짧은 대기 시간을 위해 클러스터를 도쿄의 AWS 리전(즉, `ap-northeast-1` AWS 리전)에 연결합니다. 중국 또는 GovCloud AWS 리전을 제외한 모든 퍼블릭 상용 AWS 리전의 Amazon EKS에 모든 위치의 클러스터를 연결할 수 있습니다.

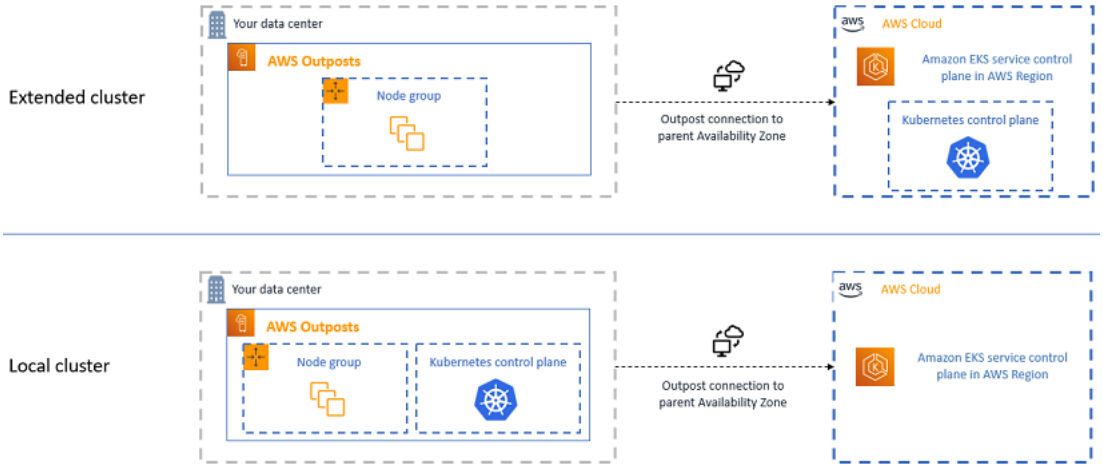
AWS Outposts에 대한 Amazon EKS

Amazon EKS를 사용하여 AWS Outposts에서 온프레미스 Kubernetes 애플리케이션을 실행할 수 있습니다. 다음과 같은 방법으로 Outposts에 Amazon EKS를 배포할 수 있습니다.

- 확장 클러스터 - AWS 리전에서 Kubernetes 컨트롤 플레인을 실행하고 Outpost에서 노드를 실행합니다.
- 로컬 클러스터 - Outpost에서 Kubernetes 컨트롤 플레인과 노드를 실행합니다.

두 배포 옵션 모두 AWS에서 Kubernetes 컨트롤 플레인을 완전히 관리합니다. 클라우드에서 사용하는 것과 동일한 Amazon EKS API, 도구 및 콘솔을 사용하여 Outposts에서 Amazon EKS를 생성하고 실행할 수 있습니다.

다음 다이어그램은 이러한 배포 옵션을 보여줍니다.



각 배포 옵션을 사용해야 하는 경우

로컬 클러스터와 확장 클러스터 모두 범용 배포 옵션이며 다양한 용도에 사용할 수 있습니다.

로컬 클러스터를 통해 전체 Amazon EKS 클러스터를 Outposts에서 로컬로 실행할 수 있습니다. 이 옵션을 통해 클라우드에 대한 일시적인 네트워크 연결 해제로 인해 발생할 수 있는 애플리케이션 가동 중지 위험을 완화할 수 있습니다. 광케이블 절단 또는 기상 현상이 이러한 네트워크 연결 해제의 원인일 수 있습니다. 전체 Amazon EKS 클러스터가 Outposts에서 로컬로 실행되기 때문에 애플리케이션을 계속 사용할 수 있습니다. 클라우드에 대한 네트워크 연결 해제 중 클러스터 작업을 수행할 수 있습니다. 자세한 내용은 네트워크 연결 해제 준비 섹션을 참조하세요. Outposts부터 상위 AWS 리전까지 네트워

크 연결 품질이 우려되고 네트워크 연결 해제를 통한 고가용성이 필요한 경우 로컬 클러스터 배포 옵션을 사용합니다.

확장 클러스터가 있으면 Kubernetes 컨트롤 플레인이 상위 AWS 리전에서 실행되기 때문에 Outpost의 용량을 절약할 수 있습니다. Outpost부터 AWS 리전까지 안정적이고 중복된 네트워크 연결에 투자할 수 있으면 이 옵션이 적합합니다. 이 옵션에는 네트워크 연결 품질이 매우 중요합니다. Kubernetes에서 Kubernetes 컨트롤 플레인과 노드 간 네트워크 연결 해제를 처리하는 방법에서는 애플리케이션 가동 중지가 발생할 수도 있습니다. Kubernetes의 동작에 대한 자세한 내용을 알아보려면 Kubernetes 문서의 스케줄링, 선점(Preemption), 축출(Eviction)를 참조하세요.

배포 옵션 비교

다음 표에 두 옵션의 차이점이 비교되어 있습니다.

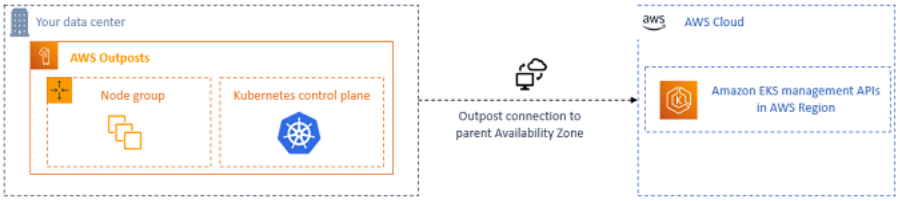
기능	확장 클러스터	로컬 클러스터
Kubernetes 컨트롤 플레인 위치	AWS 리전	Outpost
Kubernetes 컨트롤 플레인 계정	AWS 계정	사용자 계정
지역별 가용성	서비스 엔드포인트 참조	미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(캘리포니아 북부), 미국 서부(오레곤), 아시아 태평양(서울), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 캐나다(중부), 유럽(프랑크푸르트), 유럽(아일랜드), 유럽(런던), 중동(바레인) 및 남아메리카(상파울루)
Kubernetes 마이너 버전	지원되는 Amazon EKS 버전	지원되는 Amazon EKS 버전
플랫폼 버전	Amazon EKS 플랫폼 버전 섹션 참조	Amazon EKS 로컬 클러스터 플랫폼 버전 섹션 참조

기능	확장 클러스터	로컬 클러스터
Outpost 폼 팩터	Outpost 랙	Outpost 랙
사용자 인터페이스	AWS Management Console, AWS CLI, Amazon EKS API, `eksctl`, AWS CloudFormation 및 Terraform	AWS Management Console, AWS CLI, Amazon EKS API, `eksctl`, AWS CloudFormation 및 Terraform
관리형 정책	AmazonEKSClusterPolicy 및 AmazonEKSServiceRolePolicy	AmazonEKSLocalOutpostClusterPolicy 및 AmazonEKSLocalOutpostServiceRolePolicy
클러스터 VPC 및 서브넷	Amazon EKS VPC 및 서브넷 요구 사항과 고려 사항 섹션 참조	Amazon EKS 로컬 클러스터 VPC 및 서브넷 요구 사항과 고려 사항 섹션 참조
클러스터 엔드포인트 액세스	퍼블릭이나 프라이빗 또는 둘 다	프라이빗 전용
Kubernetes API 서버 인증	AWS Identity and Access Management(IAM) 및 OIDC	IAM 및 `x.509` 인증서
노드 유형	자체 관리형만	자체 관리형만
노드 컴퓨팅 유형	Amazon EC2 온디맨드	Amazon EC2 온디맨드
노드 스토리지 유형	Amazon EBS gp2 및 로컬 NVMe SSD	Amazon EBS gp2 및 로컬 NVMe SSD
Amazon EKS 최적화 AMI	Amazon Linux Windows 및 Bottlerocket	Amazon Linux만 해당
IP 버전	`IPv4` 전용	`IPv4` 전용
추가 기능	Amazon EKS 추가 기능 또는 자체 관리형 추가 기능	자체 관리형 추가 기능만

기능	확장 클러스터	로컬 클러스터
기본 컨테이너 네트워크 인터페이스	Amazon VPC CNI plugin for Kubernetes	Amazon VPC CNI plugin for Kubernetes
Kubernetes 컨트롤 플레인 로그	Amazon CloudWatch Logs	Amazon CloudWatch Logs
로드 밸런싱	AWS Load Balancer Controller를 사용하여 Application Load Balancer만 프로비저닝(Network Load Balancer 없음)	AWS Load Balancer Controller를 사용하여 Application Load Balancer만 프로비저닝(Network Load Balancer 없음)
보안 암호 봉투 암호화	기존 클러스터에서 비밀 암호화 활성화 섹션 참조	지원되지 않음
서비스 계정에 대한 IAM 역할	서비스 계정에 대한 IAM 역할 섹션 참조	지원되지 않음
문제 해결	Amazon EKS 문제 해결 섹션 참조	AWS Outposts의 Amazon EKS에 대한 로컬 클러스터 문제 해결 부분 참조

AWS Outposts의 Amazon EKS용 로컬 클러스터

로컬 클러스터를 사용하여 전체 Amazon EKS 클러스터를 AWS Outposts에서 로컬로 실행할 수 있습니다. 그러면 클라우드에 대한 일시적인 네트워크 연결 해제로 인해 발생할 수 있는 애플리케이션 가동 중지 위험 완화에 도움이 됩니다. 광케이블 절단 또는 기상 현상이 이러한 연결 해제의 원인일 수 있습니다. 전체 Kubernetes 클러스터가 Outposts에서 로컬로 실행되기 때문에 애플리케이션을 계속 사용할 수 있습니다. 클라우드에 대한 네트워크 연결 해제 중 클러스터 작업을 수행할 수 있습니다. 자세한 내용은 네트워크 연결 해제 준비 섹션을 참조하세요. 다음 다이어그램은 로컬 클러스터 배포를 보여줍니다.



로컬 클러스터는 일반적으로 Outposts 랙과 함께 사용할 수 있습니다.

지원되는 AWS 리전

미국 동부(오하이오), 미국 동부(버지니아 북부), 미국 서부(캘리포니아 북부), 미국 서부(오레곤), 아시아 태평양(서울), 아시아 태평양(싱가포르), 아시아 태평양(시드니), 아시아 태평양(도쿄), 캐나다(중부), 유럽(프랑크푸르트), 유럽(아일랜드), 유럽(런던), 중동(바레인) 및 남아메리카(상파울루) AWS 리전에서 로컬 클러스터를 생성할 수 있습니다. 지원되는 기능에 대한 자세한 내용을 알아보려면 배포 옵션 비교 섹션을 참조하세요.

주제

Outpost에서 로컬 클러스터 생성

이 주제에서는 Outpost에서 로컬 클러스터를 실행할 때 고려할 내용의 개요를 제공합니다. 이 주제에서는 Outpost에 로컬 클러스터를 배포하는 방법에 대한 지침도 제공합니다.

고려 사항

> **Important**
>
> - 이러한 고려 사항은 관련 Amazon EKS 설명서에 나와 있지 않습니다. 기타 Amazon EKS 설명서 주제가 여기의 고려 사항과 충돌하는 경우 여기의 고려 사항을 따릅니다.
> - 이러한 고려 사항은 변경될 수 있으며 자주 변경될 수 있습니다. 따라서 이 주제를 주기적으로 검토하는 것이 좋습니다.
> - AWS 클라우드에서 클러스터를 생성하기 위한 고려 사항과 많은 고려 사항이 다릅니다.

- 로컬 클러스터는 Outpost 랙만 지원합니다. 단일 로컬 클러스터는 단일 논리적 Outpost를 구성하는 여러 물리적 Outpost 랙에서 실행할 수 있습니다. 단일 로컬 클러스터는 여러 논리적 Outposts에서 실행할 수 없습니다. 각 논리적 Outpost에는 단일 Outpost ARN이 있습니다.

- 로컬 클러스터는 Outpost의 계정에서 Kubernetes 컨트롤 플레인을 실행하고 관리합니다. Kubernetes 컨트롤 플레인 인스턴스에 대한 워크로드를 실행하거나 Kubernetes 컨트롤 플레인 구성 요소를 수정할 수 없습니다. 이러한 노드는 Amazon EKS 서비스에서 관리합니다. Kubernetes 컨트롤 플레인에 대한 변경 사항은 패치와 같은 자동 Amazon EKS 관리 작업 시 유지되지 않습니다.
- 로컬 클러스터는 자체 관리형 추가 기능 및 자체 관리형 Amazon Linux 노드 그룹을 지원합니다. Amazon VPC CNI plugin for Kubernetes, `kube-proxy` 및 CoreDNS 추가 기능은 로컬 클러스터에 자동으로 설치됩니다.
- 로컬 클러스터는 Outposts에서 Amazon EBS를 사용해야 합니다. Outpost에는 Kubernetes 컨트롤 플레인 스토리지에 사용할 수 있는 Amazon EBS가 있어야 합니다.
- 로컬 클러스터는 Outposts에서 Amazon EBS를 사용합니다. Outpost에는 Kubernetes 컨트롤 플레인 스토리지에 사용할 수 있는 Amazon EBS가 있어야 합니다. Outposts는 Amazon EBS gp2 볼륨만 지원합니다.
- Amazon EBS 지원 Kubernetes `PersistentVolumes`은 Amazon EBS CSI 드라이버를 사용하여 지원됩니다.

필수 조건

- Outposts 배포 옵션, 용량 고려 사항 및 Amazon EKS 로컬 클러스터 VPC 및 서브넷 요구 사항과 고려 사항을 숙지합니다.
- 기존 Outpost. 자세한 내용은 AWS Outposts란 무엇인가요?를 참조하세요.
- 컴퓨터 또는 AWS CloudShell에 설치된 `kubectl` 명령줄 도구. 버전은 클러스터의 Kubernetes 버전과 동일하거나 최대 하나 이전 또는 이후의 마이너 버전일 수 있습니다. 예를 들어 클러스터 버전이 `1.28`인 경우 `kubectl` 버전 `1.27`, `1.28` 또는 `1.29`를 함께 사용할 수 있습니다. `kubectl`을 설치하거나 업그레이드하려면 `kubectl` 설치 또는 업데이트 부분을 참조하세요.
- AWS Command Line Interface(AWS CLI) 버전 `2.12.3` 이상 또는 `1.27.160` 이상이 디바이스나 AWS CloudShell에 설치 및 구성되어 있습니다. 현재 버전을 확인하려면 **`aws --version | cut -d / -f2 | cut -d ' ' -f1`**을 사용합니다. macOS용 `yum`, `apt-get` 또는 Homebrew와 같은 패키지 관리자는 최신 버전의 AWS CLI보다 여러 버전 이전인 경우가 많습니다. 최신 버전을 설치하려면 AWS Command Line Interface 사용 설명서의 AWS CLI 설치, 업데이트 및 제거와 aws configure를 통한 빠른 구성을 참조하세요. AWS CloudShell에 설치된 AWS CLI 버전도 최신 버전보다 여러 버전 이전일 수도 있습니다. 업데이트하려면 AWS CloudShell 사용 설명서의 홈 디렉터리에 AWS CLI 설치를 참조하세요.
- Amazon EKS 클러스터에 대한 `create` 및 `describe` 작업을 수행할 수 있는 권한이 있는 IAM 보안 주체(사용자 또는 역할). 자세한 내용은 Outpost에서 로컬 Kubernetes 클러스터 생성 및 모든 클러스터 나열 또는 설명 단원을 참조하세요.

로컬 Amazon EKS 클러스터가 생성될 때 클러스터를 생성하는 IAM 보안 주체가 영구적으로 추가됩니다. 보안 주체는 특히 Kubernetes RBAC 권한 부여 표에 관리자로 추가됩니다. 이 엔터티에는 `system:masters` 권한이 있습니다. 이 엔터티의 ID는 클러스터 구성에 표시되지 않습니다. 따라서 클러스터를 생성한 엔터티를 기록하고 절대로 삭제하지 않도록 하는 것이 중요합니다. 처음에는 서버를 생성한 보안 주체만 `kubectl`를 사용하여 Kubernetes API 서버를 호출할 수 있습니다. 콘솔을 사용하여 클러스터를 생성하는 경우 클러스터에서 `kubectl` 명령을 실행할 때 동일한 IAM 보안 인증 정보가 AWS SDK 보안 인증 정보 체인에 있어야 합니다. 클러스터가 생성되면 클러스터에 대한 액세스 권한을 다른 IAM 보안 주체에 부여할 수 있습니다.

로컬 Amazon EKS 로컬 클러스터 생성

`eksctl`, AWS Management Console, AWS CLI, Amazon EKS API, AWS SDK, AWS CloudFormation 또는 Terraform을 사용하여 로컬 클러스터를 생성할 수 있습니다.

1. 로컬 클러스터를 생성합니다.

   eksctl

   전제 조건

   디바이스 또는 `0.175.0`에 설치된 버전 AWS CloudShell 이상의 `eksctl` 명령줄 도구. `eksctl`을 설치 또는 업그레이드하려면 `eksctl` 설명서에서 Installation을 참조하세요.

   `eksctl`를 사용하여 클러스터를 생성하려면

   1. 다음 콘텐츠를 디바이스에 복사합니다. 다음 값을 바꾼 다음 수정된 명령을 실행하여 `outpost-control-plane.yaml` 파일을 생성합니다.
      - *`region-code`*를 클러스터를 생성할 지원되는 AWS 리전으로 바꿉니다.
      - *`my-cluster`*를 클러스터 이름으로 바꿉니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다. 이름은 클러스터를 생성하는 AWS 리전과 AWS 계정 내에서 고유해야 합니다.
      - *`vpc-ExampleID1`*과 *`subnet-ExampleID1`*을 기존 VPC 및 서브넷의 ID로 바꿉니다. VPC와 서브넷은 Amazon EKS 로컬 클러스터 VPC 및 서브넷 요구 사항과 고려 사항의 요구 사항을 충족해야 합니다.
      - *`uniqueid`*를 Outpost의 ID로 바꿉니다.
      - *`m5.large`*를 Outpost에서 사용 가능한 인스턴스 유형으로 바꿉니다. 인스턴스 유형을 선택하기 전에 용량 고려 사항 섹션을 참조하세요. 3개의 컨트롤 플레인 인스턴스가 배포됩니다. 이 수는 변경할 수 없습니다.

```
cat >outpost-control-plane.yaml <<EOF
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: my-cluster
  region: region-code
  version: "1.24"

vpc:
  clusterEndpoints:
    privateAccess: true
  id: "vpc-vpc-ExampleID1"
  subnets:
    private:
      outpost-subnet-1:
        id: "subnet-subnet-ExampleID1"

outpost:
  controlPlaneOutpostARN: arn:aws:outposts:region-code:111122223333:outpost/
op-uniqueid
  controlPlaneInstanceType: m5.large
EOF
```

사용 가능한 모든 옵션 및 기본값의 전체 목록은 eksctl 설명서의 AWS Outposts 지원과 구성 파일 스키마를 참조하세요.

2. 이전 단계에서 생성한 구성 파일을 사용하여 클러스터를 생성합니다. eksctl은 클러스터를 배포할 Outpost에 VPC와 서브넷 하나를 생성합니다.

```
eksctl create cluster -f outpost-control-plane.yaml
```

클러스터 프로비저닝에는 몇 분 정도 걸립니다. 클러스터가 생성되는 동안 여러 줄의 출력이 나타납니다. 출력의 마지막 줄은 다음 예제와 유사합니다.

```
[#]  EKS cluster "my-cluster" in "region-code" region is ready
```

> **Tip**
>
> `eksctl`을 사용하여 클러스터를 생성할 때 지정할 수 있는 대부분의 옵션을 보려면 **`eksctl create cluster --help`** 명령을 사용합니다. 사용 가능한 옵션을 모두 확인하려면 `config` 파일을 사용할 수 있습니다. 자세한 내용은 `eksctl` 설명서에서 config 파일 사용 및 config 파일 스키마 부분을 참조하세요. GitHub에서 구성 파일 예제를 찾을 수 있습니다.

`Eksctl`에서는 클러스터를 생성한 IAM 보안 주체(사용자 또는 역할)에 대한 액세스 항목을 자동으로 생성하고 IAM 보안 주체 관리자에게 클러스터의 Kubernetes 객체에 대한 권한을 부여합니다. 클러스터 생성자가 클러스터의 Kubernetes 객체에 대한 관리자 액세스 권한을 갖지 않도록 하려면 이전 구성 파일에 **`bootstrapClusterCreatorAdminPermissions: false`** 텍스트를 추가합니다(`metadata`, `vpc` 및 `outpost`와 같은 수준). 옵션을 추가한 경우 클러스터를 생성한 후 하나 이상의 IAM 보안 주체에 대한 액세스 항목을 생성해야 합니다. 그렇지 않으면 어떤 IAM 보안 주체도 클러스터의 Kubernetes 객체에 액세스할 수 없습니다.

AWS Management Console

전제 조건

Amazon EKS 요구 사항을 충족하는 기존 VPC 및 서브넷. 자세한 내용은 Amazon EKS 로컬 클러스터 VPC 및 서브넷 요구 사항과 고려 사항 단원을 참조하십시오.

AWS Management Console를 사용하여 클러스터를 생성하려면

1. 이미 로컬 클러스터 IAM 역할이 있거나 `eksctl`을 사용하여 클러스터를 생성하려는 경우 이 단계를 건너뛸 수 있습니다. 기본적으로 `eksctl`은 사용자를 위한 역할을 생성합니다.
   a. 다음 명령을 실행하여 IAM 신뢰 정책 JSON 파일을 생성합니다.

```
cat >eks-local-cluster-role-trust-policy.json <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

b. Amazon EKS 클러스터 IAM 역할을 생성합니다. IAM 보안 주체를 생성하려면 역할을 생성하는 IAM 보안 주체에 `iam:CreateRole` 작업(권한)을 할당해야 합니다.

```
aws iam create-role --role-name myAmazonEKSLocalClusterRole --assume-role-policy-document file://"eks-local-cluster-role-trust-policy.json"
```

c. Amazon EKS 관리형 `AmazonEKSLocalOutpostClusterPolicy`를 역할에 연결합니다. IAM 정책을 IAM 보안 주체에 연결하려면 정책을 연결하는 보안 주체에 `iam:AttachUserPolicy` 또는 `iam:AttachRolePolicy`의 IAM 작업(권한) 중 하나를 할당해야 합니다.

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/AmazonEKSLocalOutpostClusterPolicy --role-name myAmazonEKSLocalClusterRole
```

2. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
3. 콘솔 화면 상단에서 지원되는 AWS 리전을 선택했는지 확인합니다.
4. 클러스터 추가(Add cluster)를 선택하고 생성(Create)을 선택합니다.
5. Configure cluster(클러스터 구성) 페이지에서 다음 필드의 값을 입력하거나 선택합니다.
   - Kubernetes 컨트롤 플레인 위치 - AWS Outposts를 선택합니다.
   - Outpost ID - 컨트롤 플레인을 생성할 Outpost의 ID를 선택합니다.
   - Instance type(인스턴스 유형) - 인스턴스 유형을 선택합니다. Outpost에서 사용 가능한 인스턴스 유형만 표시됩니다. 드롭다운 목록에서 각 인스턴스 유형은 인스턴스 유형이 권장되는 노드 수를 설명합니다. 인스턴스 유형을 선택하기 전에 용량 고려 사항 부분을 참조하세요. 모든 복제본은 동일한 인스턴스 유형을 사용하여 배포됩니다. 클러스터가 생성된 후에는 인스턴스 유형을 변경할 수 없습니다. 3개의 컨트롤 플레인 인스턴스가 배포됩니다. 이 수는 변경할 수 없습니다.
   - 이름(Name) - 클러스터의 이름 AWS 계정에서 고유해야 합니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다. 이름은 클러스터를 생성하는 AWS 리전과 AWS 계정 내에서 고유해야 합니다.
   - Kubernetes 버전 – 클러스터에 사용하려는 Kubernetes 버전을 선택합니다. 이전 버전이 필요한 경우가 아니면 최신 버전을 선택하는 것이 좋습니다.

- Cluster service role(클러스터 서비스 역할) - 이전 단계에서 생성한 Amazon EKS 클러스터 IAM 역할을 선택하여 Kubernetes 컨트롤 플레인에서 AWS 리소스를 관리하게 할 수 있습니다.
- Kubernetes 클러스터 관리자 액세스 - 클러스터를 생성하는 IAM 보안 주체(역할 또는 사용자)가 클러스터의 Kubernetes 객체에 대한 관리자 액세스 권한을 갖도록 하려면 기본값(허용)을 그대로 수락합니다. Amazon EKS는 IAM 보안 주체에 대한 액세스 항목을 생성하고 클러스터 관리자에게 액세스 항목에 대한 권한을 부여합니다. 액세스 항목에 대한 자세한 내용은 액세스 항목 관리 섹션을 참조하십시오.

  클러스터를 생성하는 주체가 아닌 다른 IAM 보안 주체가 Kubernetes 클러스터 객체에 대한 관리자 액세스 권한을 갖도록 하려면 거부 옵션을 선택합니다. 클러스터를 생성한 후 액세스 항목을 생성할 수 있는 IAM 권한이 있는 IAM 보안 주체는 Kubernetes 클러스터 객체에 액세스해야 하는 모든 IAM 보안 주체에 대한 액세스 항목을 추가할 수 있습니다. 필수 IAM 권한에 대한 자세한 내용은 서비스 권한 부여 참조에서 Actions defined by Amazon Elastic Kubernetes Service를 참조하세요. 거부 옵션을 선택하고 액세스 항목을 생성하지 않으면 어떤 IAM 보안 주체도 클러스터의 Kubernetes 객체에 액세스할 수 없습니다.
- 태그(Tags) - (선택 사항) 클러스터에 태그를 추가합니다. 자세한 내용은 Amazon EKS 리소스 태깅 단원을 참조하십시오.

이 페이지를 모두 완료하면 다음을 선택합니다.

6. 네트워킹 지정(Specify networking) 페이지에서 다음 필드의 값을 선택합니다.
   - VPC – 기존 VPC를 선택합니다. VPC에는 생성하려는 클러스터, 노드 및 기타 Kubernetes 리소스에 사용 가능한 IP 주소가 충분해야 합니다. VPC에서는 VPC 요구 사항 및 고려 사항의 요구 사항을 충족해야 합니다.
   - 서브넷(Subnets) - 기본적으로 이전 필드에서 지정한 VPC에서 사용 가능한 모든 서브넷이 사전 선택됩니다. 선택한 서브넷은 서브넷 요구 사항 및 고려 사항의 요구 사항을 충족해야 합니다.

     보안 그룹(Security groups) – (선택 사항) 생성하는 네트워크 인터페이스에 Amazon EKS가 연결하려는 보안 그룹을 하나 이상 지정합니다. Amazon EKS에서는 클러스터와 VPC 간 통신을 활성화하는 보안 그룹을 자동으로 생성합니다. Amazon EKS는 이 보안 그룹과 사용자가 선택한 보안 그룹을 생성하는 네트워크 인터페이스에 연결합니다. Amazon EKS에서 생성하는 클러스터 보안 그룹에 대한 자세한 내용을 알아보려면 Amazon EKS 보안 그룹 요구 사항 및 고려 사항 부분을 참조하세요. Amazon EKS가 생성하는 클러스터 보안 그룹에서 규칙을 수정할 수 있습니다. 자체 보안 그룹을 추가하도록 선택한 경우 클

러스터 생성 후 선택한 보안 그룹은 변경할 수 없습니다. 온프레미스 호스트가 클러스터 엔드포인트와 통신하려면 클러스터 보안 그룹의 인바운드 트래픽을 허용해야 합니다. 수신 및 송신 인터넷 연결이 없는 클러스터(프라이빗 클러스터라고도 함)의 경우 다음 중 하나를 수행해야 합니다.

- 필수 VPC 엔드포인트와 연결된 보안 그룹을 추가합니다. 필수 엔드포인트에 대한 자세한 내용은 서브넷 액세스AWS 서비스의 VPC 엔드포인트를 참조하세요.
- VPC 엔드포인트와 연결된 보안 그룹의 트래픽을 허용하려면 Amazon EKS에서 생성한 보안 그룹을 수정합니다.

이 페이지를 모두 완료하면 다음을 선택합니다.

7. 관찰성 구성 페이지에서 설정할 지표 및 컨트롤 플레인 로깅 옵션을 선택할 수 있습니다. 기본적으로 각 로그 유형은 해제되어 있습니다.
   - Prometheus 지표 옵션에 자세한 내용은 클러스터를 생성할 때 Prometheus 지표 켜기 섹션을 참조하세요.
   - 컨트롤 플레인 로깅 옵션에 대한 자세한 내용은 Amazon EKS 컨트롤 플레인 로깅 섹션을 참조하세요.

   이 페이지를 모두 완료하면 다음을 선택합니다.
8. 검토 및 생성 페이지에서 이전 페이지에서 입력하거나 선택한 정보를 검토합니다. 변경해야 하는 경우 편집(Edit)을 선택합니다 만족하는 경우 생성(Create)을 선택합니다. 클러스터가 프로비저닝되는 동안 상태(Status) 필드에는 생성 중(CREATING)이 표시됩니다.

   클러스터 프로비저닝에는 몇 분 정도 걸립니다.

2. 클러스터가 생성된 후 생성된 Amazon EC2 컨트롤 플레인 인스턴스를 볼 수 있습니다.

```
aws ec2 describe-instances --query 'Reservations[*].Instances[*].{Name:Tags[?
Key==`Name`]|[0].Value}' | grep my-cluster-control-plane
```

예제 출력은 다음과 같습니다.

```
"Name": "my-cluster-control-plane-id1"
"Name": "my-cluster-control-plane-id2"
"Name": "my-cluster-control-plane-id3"
```

컨트롤 플레인 인스턴스에서 워크로드가 예약되지 않도록 각 인스턴스는 `node-role.eks-local.amazonaws.com/control-plane`으로 테인트됩니다. 테인트에 대한 자세한 내용을 알아보려면 Kubernetes 문서의 테인트(Taints)와 톨러레이션(Tolerations)을 참조하세요. Amazon

EKS는 로컬 클러스터의 상태를 지속적으로 모니터링합니다. 보안 패치 및 비정상 인스턴스 복구와 같은 자동 관리 작업을 수행합니다. 로컬 클러스터가 클라우드에서 연결 해제되면 재연결 시 클러스터가 정상 상태로 복구되도록 조치를 완료합니다.

3. `eksctl`을 사용하여 클러스터를 생성한 경우 이 단계를 건너뛸 수 있습니다. `eksctl`에서 자동으로 이 단계를 완료합니다. `kubectl config` 파일에 새 컨텍스트를 추가하여 `kubectl`이 클러스터와 통신하도록 사용 설정합니다. 파일 생성 설치 및 업데이트 방법에 대한 지침은 Amazon EKS 클러스터용 kubeconfig 파일 생성 또는 업데이트 부분을 참조하세요.

```
aws eks update-kubeconfig --region region-code --name my-cluster
```

예제 출력은 다음과 같습니다.

```
Added new context arn:aws:eks:region-code:111122223333:cluster/my-cluster to /home/username/.kube/config
```

4. 로컬 클러스터의 Kubernetes API 서버에 연결하려면 서브넷의 로컬 게이트웨이에 액세스하거나 VPC 내에서 연결해야 합니다. Outpost 랙을 온프레미스 네트워크에 연결하는 방법을 자세히 알아보려면 AWS Outposts 사용 설명서의 랙용 로컬 게이트웨이 작동 방식을 참조하세요. Direct VPC Routing을 사용하며 Outpost 서브넷에 로컬 게이트웨이에 대한 경로가 있으면 Kubernetes 컨트롤 플레인 인스턴스의 프라이빗 IP 주소가 로컬 네트워크를 통해 자동으로 브로드캐스트됩니다. 로컬 클러스터의 Kubernetes API 서버 엔드포인트는 Amazon Route 53(Route 53)에서 호스팅됩니다. API 서비스 엔드포인트는 퍼블릭 DNS 서버에서 Kubernetes API 서버의 프라이빗 IP 주소로 확인할 수 있습니다.

로컬 클러스터의 Kubernetes 컨트롤 플레인 인스턴스는 클러스터 수명 주기 동안 변경되지 않는 고정 프라이빗 IP 주소를 사용하여 정적 탄력적 네트워크 인터페이스로 구성됩니다. Kubernetes API 서버와 상호 작용하는 머신은 네트워크 연결이 해제된 동안 Route 53에 연결되지 않을 수도 있습니다. 이 경우 계속되는 작업을 위해 고정 프라이빗 IP 주소로 `/etc/hosts`를 구성하는 것이 좋습니다. 또한 로컬 DNS 서버를 설정하고 Outpost에 연결하는 것이 좋습니다. 자세한 내용은 AWS Outposts 설명서를 참조하십시오. 다음 명령을 실행하여 클러스터와 설정된 통신을 확인합니다.

```
kubectl get svc
```

예제 출력은 다음과 같습니다.

```
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
```

```
kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   28h
```

5. (선택 사항) 로컬 클러스터가 AWS 클라우드에서 연결 해제된 상태일 때 인증을 테스트합니다. 지침은 네트워크 연결 해제 준비 을 참조하세요.

내부 리소스

Amazon EKS는 클러스터에 다음 리소스를 생성합니다. 리소스는 Amazon EKS 내부용입니다. 클러스터가 제대로 작동하려면 이러한 리소스를 편집하거나 수정하지 않습니다.

- 다음 미러 Pods:
  - `aws-iam-authenticator-node-hostname`
  - `eks-certificates-controller-node-hostname`
  - `etcd-node-hostname`
  - `kube-apiserver-node-hostname`
  - `kube-controller-manager-node-hostname`
  - `kube-scheduler-node-hostname`
- 다음 자체 관리형 추가 기능:
  - `kube-system/coredns`
  - `kube-system/kube-proxy`(첫 번째 노드를 추가할 때까지 생성되지 않음)
  - `kube-system/aws-node`(첫 번째 노드를 추가할 때까지 생성되지 않음). 로컬 클러스터는 클러스터 네트워킹에 Amazon VPC CNI plugin for Kubernetes 플러그인을 사용합니다. 컨트롤 플레인 인스턴스(`aws-node-controlplane-*`라는 포드)의 구성을 변경하지 않습니다. 플러그인에서 새 네트워크 인터페이스를 생성할 때 기본값 변경에 사용할 수 있는 구성 변수가 있습니다. 자세한 내용을 알아보려면 GitHub의 설명서를 참조하세요.
- 다음 서비스:
  - `default/kubernetes`
  - `kube-system/kube-dns`
- 이름이 `eks.system`인 `PodSecurityPolicy`
- 이름이 `eks:system:podsecuritypolicy`인 `ClusterRole`
- 이름이 `eks:system`인 `ClusterRoleBinding`
- 기본 포드 보안 정책
- 클러스터 보안 그룹 외에도 Amazon EKS에서는 AWS 계정에 이름이 `eks-local-internal-do-not-use-or-edit-cluster-name-uniqueid`인 보안 그룹을 생성합니다. 이 보안 그룹을 사용

하면 컨트롤 플레인 인스턴스에서 실행되는 Kubernetes 구성 요소 간에 트래픽이 자유롭게 흐를 수 있습니다.

권장되는 다음 단계:

- 클러스터를 생성한 IAM 보안 주체에 AWS Management Console에서 Kubernetes 리소스를 볼 수 있는 필요한 권한 부여
- 클러스터에 IAM 엔터티에 액세스 권한 부여 엔터티가 Amazon EKS 콘솔에서 Kubernetes 리소스를 보도록 하려면 엔티티에 필요한 권한을 부여합니다.
- 클러스터에 대한 로깅 구성
- 네트워크 연결 해제 중 어떤 일이 발생하는지 숙지합니다.
- 클러스터에 노드 추가
- `etcd`에 대한 백업 계획 설정을 고려합니다. Amazon EKS는 로컬 클러스터에 대한 `etcd`의 자동 백업 및 복원을 지원하지 않습니다. 자세한 내용은 Kubernetes 설명서의 `etcd` 클러스터 백업을 참조하세요. 두 가지 주요 옵션은 `etcdctl`을 사용하여 스냅샷 생성 또는 Amazon EBS 스토리지 볼륨 백업 사용을 자동화합니다.

Amazon EKS 로컬 클러스터 플랫폼 버전

로컬 클러스터 플랫폼 버전은 AWS Outposts에 있는 Amazon EKS 클러스터의 기능을 나타냅니다. 버전에는 Kubernetes API 서버 플래그가 활성화된 Kubernetes 컨트롤 플레인에서 실행되는 구성 요소가 포함됩니다. 현재 Kubernetes 패치 버전도 포함됩니다. 각 Kubernetes 마이너 버전에는 하나 이상의 연결된 플랫폼 버전이 있습니다. 서로 다른 Kubernetes 마이너 버전에 대한 플랫폼 버전은 독립적입니다. 클라우드의 로컬 클러스터와 Amazon EKS 클러스터의 플랫폼 버전은 독립적입니다.

`1.28`과 같은 로컬 클러스터에 새 Kubernetes 마이너 버전을 사용할 수 있는 경우 해당 Kubernetes 마이너 버전의 초기 플랫폼 버전은 `eks-local-outposts.1`에서 시작합니다. 그러나 Amazon EKS에서는 새 Kubernetes 컨트롤 플레인 설정을 활성화하고 보안 수정 사항을 제공하기 위해 새 플랫폼 버전을 주기적으로 릴리스합니다.

마이너 버전에 대해 새 로컬 클러스터 플랫폼 버전을 사용할 수 있게 된 경우 다음과 같은 사항이 발생합니다.

- 플랫폼 버전 번호가 증가합니다(`eks-local-outposts.n+1`).
- Amazon EKS에서 기존의 모든 로컬 클러스터를 해당하는 Kubernetes 마이너 버전에 대한 최신 플랫폼 버전으로 자동으로 업데이트합니다. 기존 플랫폼 버전의 자동 업데이트는 증분 방식으로 롤아

웃됩니다. 이 롤아웃 프로세스는 다소 시간이 걸릴 수 있습니다. 최신 플랫폼 버전 기능이 즉시 필요한 경우 새 로컬 클러스터를 생성하는 것이 좋습니다.

- Amazon EKS에서는 해당 패치 버전으로 새 AMI를 게시할 수 있습니다. 모든 패치 버전은 단일 Kubernetes 마이너 버전에 대한 노드 AMI와 Kubernetes 컨트롤 플레인 간에 호환됩니다.

새 플랫폼 버전은 주요 변경 사항을 도입하거나 서비스 중단을 초래하지 않습니다.

로컬 클러스터는 항상 지정된 Kubernetes 버전에 사용 가능한 최신 플랫폼 버전(`eks-local-outposts.n`)으로 생성됩니다.

현재 및 최근 플랫폼 버전은 다음 표에 설명되어 있습니다.

Kubernetes 버전 `1.28`

다음 승인 컨트롤러는 모든 `1.28` 플랫폼 버전(`CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `DefaultStorageClass`, `DefaultTolerationSeconds`, `ExtendedResourceToleration`, `LimitRanger`, `MutatingAdmissionWebhook`, `NamespaceLifecycle`, `NodeRestriction`, `PersistentVolumeClaimResize`, `Priority`, `PodSecurity`, `ResourceQuota`, `RuntimeClass`, `ServiceAccount`, `StorageObjectInUseProtection`, `TaintNodesByCondition`, `ValidatingAdmissionPolicy`, `ValidatingAdmissionWebhook`)에 대해 활성화됩니다.

Kubernetes 버전	Amazon EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
`1.28.1`	`eks-local-outposts.1`	Outposts의 로컬 Amazon EKS 클러스터 Kubernetes 버전 `1.28`의 최초 릴리스.	2023년 10월 4일

Kubernetes 버전 `1.27`

다음 승인 컨트롤러는 모든 `1.27` 플랫폼 버전(`CertificateApproval`, `CertificateSigning`, `CertificateSubjectRestriction`, `DefaultIngressClass`, `DefaultStorageClass`, `DefaultTolerationSeconds`, `ExtendedResourceToleration`, `LimitRanger`, `MutatingAdmissionWebhook`, `NamespaceLifecycle`, `NodeRestriction`,

PersistentVolumeClaimResize, Priority, PodSecurity, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionPolicy, ValidatingAdmissionWebhook)에 대해 활성화됩니다.

Kubernetes 버전	Amazon EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.27.3	eks-local-outposts.3	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전. kube-proxy 은(는) v1.27.3(으)로 업데이트되었습니다. Kubernetes용 Amazon VPC CNI 플러그인은 v1.13.2(으)로 업데이트되었습니다.	2023년 7월 14일
1.27.1	eks-local-outposts.2	CoreDNS 이미지가 v1.10.1로 업데이트됨	2023년 6월 22일
1.27.1	eks-local-outposts.1	Outposts의 로컬 Amazon EKS 클러스터 Kubernetes 버전 1.27의 최초 릴리스.	2023년 5월 30일

Kubernetes 버전 **1.26**

다음 승인 컨트롤러는 모든 1.26 플랫폼 버전(CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, PersistentVolumeClaimResize, Priority, PodSecurity, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionPolicy, ValidatingAdmissionWebhook)에 대해 활성화됩니다.

Kubernetes 버전	Amazon EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.26.6	eks-local-outposts.4	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전. kube-proxy 은(는) v1.26.6(으)로 업데이트되었습니다. Kubernetes용 Amazon VPC CNI 플러그인은 v1.13.2(으)로 업데이트되었습니다.	2023년 7월 14일
1.26.4	eks-local-outposts.3	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 7월 13일
1.26.2	eks-local-outposts.2	Bottlerocket 버전을 1.13.2로 업데이트했습니다.	2023년 5월 2일
1.26.2	eks-local-outposts.1	Outposts의 로컬 Amazon EKS 클러스터 Kubernetes 버전 1.26의 최초 릴리스.	2023년 4월 11일

Kubernetes 버전 **1.25**

다음 승인 컨트롤러는 모든 1.25 플랫폼 버전(CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, ExtendedResourceToleration, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, PersistentVolumeClaimResize, Priority, PodSecurity, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionWebhook)에 대해 활성화됩니다.

Kubernetes 버전	Amazon EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.25.11	eks-local-outposts.6	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전. kube-proxy 은(는) v1.25.11(으)로	2023년 7월 14일

Kubernetes 버전	Amazon EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
		업데이트되었습니다. Kubernetes용 Amazon VPC CNI 플러그인은 v1.13.2(으)로 업데이트되었습니다.	
1.25.9	eks-local-outposts.5	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 7월 13일
1.25.6	eks-local-outposts.4	Bottlerocket 버전을 1.13.2로 업데이트했습니다.	2023년 5월 2일
1.25.6	eks-local-outposts.3	Amazon EKS 컨트롤 플레인 인스턴스 운영 체제가 Bottlerocket 버전 v1.13.1로 업데이트되고 Amazon VPC CNI plugin for Kubernetes는 버전 v1.12.6으로 업데이트되었습니다.	2023년 4월 14일
1.25.6	eks-local-outposts.2	Kubernetes컨트롤 플레인 인스턴스에 대한 진단 컬렉션이 개선되었습니다.	2023년 3월 8일
1.25.6	eks-local-outposts.1	Outposts의 로컬 Amazon EKS 클러스터 Kubernetes 버전 1.25의 최초 릴리스.	2023년 3월 1일

Kubernetes 버전 **1.24**

다음 승인 컨트롤러는 모든 1.24 플랫폼 버전(DefaultStorageClass, DefaultTolerationSeconds, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, ResourceQuota, ServiceAccount, ValidatingAdmissionWebhook, PodSecurityPolicy, TaintNodesByCondition, StorageObjectInUseProtection, PersistentVolumeClaimResize, ExtendedResourceToleration, CertificateApproval, PodPriority,

CertificateSigning, CertificateSubjectRestriction, RuntimeClass, DefaultIngressClass)에 대해 활성화됩니다.

Kubernetes 버전	Amazon EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.24.15	eks-local-outposts.6	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전. kube-proxy 은(는) v1.24.15(으)로 업데이트되었습니다. Kubernetes용 Amazon VPC CNI 플러그인은 v1.13.2(으)로 업데이트되었습니다.	2023년 7월 14일
1.24.13	eks-local-outposts.5	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 7월 13일
1.24.9	eks-local-outposts.4	Bottlerocket 버전을 1.13.2로 업데이트했습니다.	2023년 5월 2일
1.24.9	eks-local-outposts.3	Amazon EKS 컨트롤 플레인 인스턴스 운영 체제가 Bottlerocket 버전 v1.13.1로 업데이트되고 Amazon VPC CNI plugin for Kubernetes는 버전 v1.12.6으로 업데이트되었습니다.	2023년 4월 14일
1.24.9	eks-local-outposts.2	Kubernetes컨트롤 플레인 인스턴스에 대한 진단 컬렉션이 개선되었습니다.	2023년 3월 8일
1.24.9	eks-local-outposts.1	Outposts의 로컬 Amazon EKS 클러스터 Kubernetes 버전 1.24의 최초 릴리스.	2023년 1월 17일

Kubernetes 버전 **1.23**

다음 승인 컨트롤러는 모든 1.23 플랫폼 버전(DefaultStorageClass, DefaultTolerationSeconds, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, NodeRestriction, ResourceQuota, ServiceAccount, ValidatingAdmissionWebhook, PodSecurityPolicy, TaintNodesByCondition, StorageObjectInUseProtection, PersistentVolumeClaimResize, ExtendedResourceToleration, CertificateApproval, PodPriority, CertificateSigning, CertificateSubjectRestriction, RuntimeClass, DefaultIngressClass)에 대해 활성화됩니다.

Kubernetes 버전	Amazon EKS 플랫폼 버전	릴리스 정보	릴리스 날짜
1.23.17	eks-local-outposts.5	보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전.	2023년 7월 13일
1.23.15	eks-local-outposts.4	Bottlerocket 버전을 1.13.2로 업데이트했습니다.	2023년 5월 2일
1.23.15	eks-local-outposts.3	Amazon EKS 컨트롤 플레인 인스턴스 운영 체제가 Bottlerocket 버전 v1.13.1로 업데이트되고 Amazon VPC CNI plugin for Kubernetes는 버전 v1.12.6으로 업데이트되었습니다.	2023년 4월 14일
1.23.15	eks-local-outposts.2	Kubernetes컨트롤 플레인 인스턴스에 대한 진단 컬렉션이 개선되었습니다.	2023년 3월 8일
1.23.15	eks-local-outposts.1	Outposts의 로컬 Amazon EKS 클러스터 Kubernetes 버전 1.23의 최초 릴리스.	2023년 1월 17일

Amazon EKS 로컬 클러스터 VPC 및 서브넷 요구 사항과 고려 사항

로컬 클러스터를 생성할 때 Outposts에서 실행되는 하나 이상의 프라이빗 서브넷과 VPC를 지정합니다. 이 주제에서는 로컬 클러스터에 대한 VPC 및 서브넷 요구 사항 및 고려 사항을 간략하게 살펴봅니다.

VPC 요구 사항 및 고려 사항

로컬 클러스터를 생성하는 경우 지정하는 VPC는 다음 요구 사항 및 고려 사항을 충족해야 합니다.

- 생성하려는 로컬 클러스터, 노드 및 기타 Kubernetes 리소스에 대한 충분한 IP 주소가 VPC에 있는지 확인합니다. 사용하려는 VPC에 충분한 IP 주소가 없는 경우 사용 가능한 IP 주소를 증량합니다. 이 작업은 해당 VPC를 통해 추가 Classless Inter-Domain Routing(CIDR) 블록을 연결하여 수행할 수 있습니다. 클러스터를 생성하기 전이나 후에 프라이빗(RFC 1918) 및 퍼블릭(비RFC 1918) CIDR 블록을 VPC에 연결할 수 있습니다. 클러스터가 VPC와 연결된 CIDR 블록을 인식하는 데 최대 5시간이 걸릴 수 있습니다.
- VPC에는 할당된 IP 접두사 또는 IPv6 CIDR 블록이 있을 수 없습니다. 이러한 제약으로 인해 Amazon EC2 노드에 사용 가능한 IP 주소 증량 및 클러스터, Pods 및 services용 IPv6 주소에 설명된 정보는 VPC에 적용되지 않습니다.
- 활성화된 DNS 호스트 이름과 DNS 확인이 VPC에 있습니다. 이러한 기능이 없으면 로컬 클러스터 생성에 실패하므로 기능을 활성화하고 클러스터를 다시 생성해야 합니다. 자세한 내용을 알아보려면 Amazon VPC 사용 설명서의 VPC에 대한 DNS 속성을 참조하세요.
- 로컬 네트워크를 통해 로컬 클러스터에 액세스하려면 VPC가 Outpost의 로컬 게이트웨이 라우팅 테이블과 연결되어 있어야 합니다. 자세한 내용을 알아보려면 AWS Outposts 사용 설명서의 VPC 연결을 참조하세요.

서브넷 요구 사항 및 고려 사항

클러스터를 생성할 때 하나 이상의 프라이빗 서브넷을 지정합니다. 서브넷을 여러 개 지정하면 Kubernetes 컨트롤 플레인 인스턴스가 서브넷 전체에 고르게 분산됩니다. 2개 이상의 서브넷이 지정된 경우 서브넷이 동일한 Outpost에 있어야 합니다. 아울러 서로 통신하는 적절한 경로와 보안 그룹 권한도 서브넷에 있어야 합니다. 로컬 클러스터를 생성할 때 지정하는 서브넷에서 다음과 같은 요구 사항을 충족해야 합니다.

- 서브넷이 모두 동일한 논리적 Outpost에 있습니다.
- 서브넷에 전체적으로 Kubernetes 컨트롤 플레인 인스턴스에 대해 사용 가능한 IP 주소가 3개 이상 있어야 합니다. 3개의 서브넷이 지정된 경우 각 서브넷에 사용 가능한 IP 주소가 1개 이상 있어야 합

니다. 2개의 서브넷이 지정된 경우 각 서브넷에 사용 가능한 IP 주소가 2개 이상 있어야 합니다. 1개의 서브넷이 지정된 경우 서브넷에 사용 가능한 IP 주소가 3개 이상 있어야 합니다.

- 로컬 네트워크를 통해 Kubernetes API 서버에 액세스하기 위한 Outpost 랙의 로컬 게이트웨이에 대한 경로가 서브넷에 있어야 합니다. 서브넷에 Outpost 랙의 로컬 게이트웨이에 대한 경로가 없는 경우 VPC 내에서 Kubernetes API 서버와 통신해야 합니다.
- 서브넷에서는 IP 주소 기반 이름 지정을 사용해야 합니다. Amazon EC2 리소스 기반 이름 지정은 Amazon EKS에서 지원되지 않습니다.

서브넷 액세스AWS 서비스

Outst에 있는 로컬 클러스터의 사설 서브넷은 있어야 합니다AWS 서비스. 아웃바운드 인터넷 액세스를 위해 NAT 게이트웨이를 사용하거나, VPC 내에서 모든 트래픽을 비공개로 유지하려는 경우 인터페이스 VPC 엔드포인트를 사용하여 이를 달성할 수 있습니다.

NAT 게이트웨이 태그 지정

Outpost의 상위 가용 영역에 있는 퍼블릭 서브넷에서 실행되는 NAT 게이트웨이에 대한 경로가 있는 연결된 라우팅 테이블이 서브넷에 있습니다. 퍼블릭 서브넷에 인터넷 게이트웨이에 대한 경로가 있어야 합니다. 이 경로는 아웃바운드 인터넷 액세스를 활성화하고 인터넷에서 Outpost의 인스턴스로의 원치 않는 인바운드 연결을 방지합니다.

인터페이스 VPC 종단점 사용

Outst에 있는 로컬 클러스터의 사설 서브넷에 송신 인터넷 연결이 없는 경우에는, 또는 송신 인터넷 연결이 없는 경우에는 클러스터를 생성하기 전에 다음과 같은 인터페이스 VPC 엔드포인트와 게이트웨이 엔드포인트를 생성해야 합니다.

엔드포인트	[엔드포인트 유형]
com.amazonaws. *region-code* .ssm	인터페이스
com.amazonaws. *region-code* .ssmmessages	인터페이스
com.amazonaws. *region-code* .ec2messages	인터페이스
com.amazonaws. *region-code* .ec2	인터페이스

엔드포인트	[엔드포인트 유형]
com.amazonaws. *region-code* .secretsmanager	인터페이스
com.amazonaws. *region-code* .logs	인터페이스
com.amazonaws. *region-code* .sts	인터페이스
com.amazonaws. *region-code* .ecr.api	인터페이스
com.amazonaws. *region-code* .ecr.dkr	인터페이스
com.amazonaws. *region-code* .s3	게이트웨이

이름은 다음 요구 사항을 충족해야 합니다.

- Outpost의 상위 가용 영역에 있는 프라이빗 서브넷에서 생성됨
- 프라이빗 DNS 이름 활성화
- 사설 송신 서브넷의 CIDR 범위에서 오는 인바운드 HTTPS 트래픽을 허용하는 연결된 보안 그룹이 있습니다.

엔드포인트를 생성하면 요금이 부과됩니다. 자세한 내용은 AWS PrivateLink 요금을 참조하십시오. 기타 AWS 서비스에 대한 액세스 권한이 Pods에 필요한 경우에는 추가 엔드포인트를 생성해야 합니다. 포괄적인 엔드포인트 목록은 AWS PrivateLink와 통합되는 AWS 서비스를 참조하세요.

VPC 생성

다음과 같은 AWS CloudFormation 템플릿 중 하나를 사용하여 이전 요구 사항을 충족하는 VPC를 생성할 수 있습니다.

- 템플릿 1 - 이 템플릿에서는 Outpost의 프라이빗 서브넷 1개와 AWS 리전의 퍼블릭 서브넷 1개로 VPC를 생성합니다. 프라이빗 서브넷에는 AWS 리전의 퍼블릭 서브넷에 상주하는 NAT 게이트웨이를 통해 인터넷에 연결되는 경로가 있습니다. 이 템플릿은 송신 인터넷 액세스 권한이 있는 서브넷에 로컬 클러스터를 생성하는 데 사용할 수 있습니다.
- 템플릿 2 - 이 템플릿에서는 Outpost의 프라이빗 서브넷 1개와 수신 또는 송신 인터넷 액세스 권한이 없는 서브넷에 로컬 클러스터(프라이빗 서브넷이라고도 함)를 생성하는 데 필요한 최소 VPC 엔드포인트 집합이 있는 VPC를 생성합니다.

네트워크 연결 해제 준비

로컬 네트워크와 AWS 클라우드의 연결이 끊어진 경우 Outpost에서 로컬 Amazon EKS 클러스터를 계속 사용할 수 있습니다. 이 주제에서는 로컬 클러스터에서 네트워크 연결 해제에 대비하는 방법과 관련 고려 사항을 설명합니다.

로컬 클러스터에서 네트워크 연결 해제에 대비할 때 고려할 사항:

- 로컬 클러스터는 계획되지 않은 일시적인 네트워크 연결 해제 동안 안정성과 지속적인 운영을 가능하게 합니다. AWS Outposts는 데이터 센터에서 AWS 클라우드의 확장 역할을 하는 완전히 연결된 제품으로 남아 있습니다. Outpost와 AWS 클라우드 간 네트워크 연결이 해제된 경우 연결 복원을 시도하는 것이 좋습니다. 지침은 AWS Outposts 사용 설명서의 AWS Outposts 랙 네트워크 문제 해결 체크리스트를 참조하세요. 로컬 클러스터 문제를 해결하는 방법에 대한 자세한 내용은 AWS Outposts의 Amazon EKS에 대한 로컬 클러스터 문제 해결 섹션을 참조하세요.
- Outposts는 Outpost의 연결 상태를 모니터링하는 데 사용할 수 있는 `ConnectedStatus` 지표를 내보냅니다. 자세한 내용은 AWS Outposts 사용 설명서의 Outposts 지표를 참조하세요.
- 로컬 클러스터는 Kubernetes용 AWS Identity and Access Management 인증자를 사용하여 IAM을 기본 인증 메커니즘으로 사용합니다. 네트워크 연결 해제 중에는 IAM을 사용할 수 없습니다. 따라서 로컬 클러스터에서는 네트워크 연결 해제 중 클러스터에 연결하는 데 사용할 수 있는 `x.509` 인증서를 사용하는 대체 인증 메커니즘을 지원합니다. 클러스터에 대한 `x.509` 인증서를 받고 사용하는 방법에 대한 내용은 네트워크 연결 해제 중 로컬 클러스터에 인증 섹션을 참조하세요.
- 네트워크 연결 해제 중 Route 53에 액세스할 수 없는 경우 온프레미스 환경에서 로컬 DNS 서버를 사용하는 것이 좋습니다. Kubernetes 컨트롤 플레인 인스턴스는 고정 IP 주소를 사용합니다. 로컬 DNS 서버를 사용하는 대신 엔드포인트 호스트 이름과 IP 주소를 사용하여 클러스터에 연결하는 데 사용하는 호스트를 구성할 수 있습니다. 자세한 내용은 AWS Outposts 사용 설명서에서 DNS를 참조하세요.
- 네트워크 연결 해제 중 애플리케이션 트래픽이 증가할 것으로 예상되는 경우 클라우드에 연결될 때 클러스터에 여분의 컴퓨팅 용량을 프로비저닝할 수 있습니다. Amazon EC2 인스턴스는 AWS Outposts의 가격에 포함됩니다. 따라서 여분의 인스턴스를 실행해도 AWS 사용 비용에 영향을 미치지 않습니다.
- 네트워크 연결 해제 중 워크로드에 대한 생성, 업데이트 및 조정 작업을 활성화하려면 로컬 네트워크를 통해 애플리케이션의 컨테이너 이미지에 액세스할 수 있어야 하고 클러스터에 충분한 용량이 있어야 합니다. 로컬 클러스터는 컨테이너 레지스트리를 호스팅하지 않습니다. Pods가 이전에 해당 노드에서 실행된 경우 컨테이너 이미지가 노드에서 캐시됩니다. 일반적으로 클라우드의 Amazon ECR에서 애플리케이션의 컨테이너 이미지를 가져오는 경우 로컬 캐시 또는 레지스트리를 실행해 보세

요. 네트워크 연결 해제 중 워크로드 리소스에 대한 생성, 업데이트 또는 조정 작업이 필요한 경우 로컬 캐시 또는 레지스트리가 도움이 됩니다.

- 로컬 클러스터는 Amazon EBS를 영구 볼륨의 기본 스토리지 클래스로 사용하고 Amazon EBS CSI 드라이버를 사용하여 Amazon EBS 영구 볼륨의 수명 주기를 관리합니다. 네트워크 연결 해제 중 Amazon EBS에서 지원하는 Pods는 생성, 업데이트 또는 조정할 수 없습니다. 이러한 작업에는 클라우드의 Amazon EBS API에 대한 호출이 필요하기 때문입니다. 로컬 클러스터에 상태 저장 워크로드를 배포하고 네트워크 연결 해제 중 생성, 업데이트 또는 조정 작업이 필요한 경우 대체 스토리지 메커니즘을 사용하는 것이 좋습니다.
- AWS Outposts에서 관련 AWS 리전 내 API(예: Amazon EBS 또는 Amazon S3용 API)에 액세스할 수 없는 경우 Amazon EBS 스냅샷을 생성하거나 삭제할 수 없습니다.
- ALB(수신)를 AWS Certificate Manager(ACM)와 통합하면 인증서가 푸시되어 AWS Outposts ALB Compute 인스턴스의 메모리에 저장됩니다. AWS 리전에서 연결이 해제되는 경우 현재 TLS 종료는 계속 작동합니다. 이 컨텍스트에서는 변형 작업(예: 새 수신 정의, 새 ACM 기반 인증서 API 작업, ALB 컴퓨팅 크기 조정 또는 인증서 교체)에 실패합니다. 자세한 내용은 AWS Certificate Manager 사용 설명서의 관리형 인증서 갱신 문제 해결을 참조하세요.
- Amazon EKS 컨트롤 플레인 로그는 네트워크 연결 해제 중 Kubernetes 컨트롤 플레인 인스턴스에서 로컬로 캐시됩니다. 다시 연결되면 로그가 상위 AWS 리전의 CloudWatch Logs로 전송됩니다. Prometheus, Grafana 또는 Amazon EKS 파트너 솔루션을 사용하여 Kubernetes API 서버의 지표 엔드포인트를 사용하거나 로그에 Fluent Bit를 사용하여 로컬로 클러스터를 모니터링할 수 있습니다.
- 애플리케이션 트래픽을 위해 Outposts에서 AWS Load Balancer Controller를 사용하는 경우 AWS Load Balancer Controller 뒤에 있는 기존 Pods는 네트워크 연결 해제 중 트래픽을 계속 수신합니다. 네트워크 연결 해제 중 생성된 새 Pods는 Outpost가 AWS 클라우드에 다시 연결될 때까지 트래픽을 수신하지 않습니다. 네트워크 연결 해제 중 조정 요구 사항을 수용할 수 있도록 AWS 클라우드에 연결되어 있는 동안 애플리케이션의 복제본 수를 설정하는 것이 좋습니다.
- Amazon VPC CNI plugin for Kubernetes는 기본적으로 보조 IP 모드입니다. `WARM_ENI_TARGET=1`로 구성되어 플러그인이 사용 가능한 IP 주소의 '전체 탄력적 네트워크 인터페이스'를 사용 가능한 상태로 유지할 수 있습니다. 연결 해제된 상태에서 조정 요구 사항에 따라 `WARM_ENI_TARGET`, `WARM_IP_TARGET` 및 `MINIMUM_IP_TARGET` 값을 변경하는 것을 고려하세요. 자세한 내용을 알아보려면 GitHub에서 플러그인의 `readme` 파일을 참조하세요. 각 인스턴스 유형별로 지원되는 최대 Pods 수 목록은 GitHub의 `eni-max-pods.txt` 파일을 참조하세요.

네트워크 연결 해제 중 로컬 클러스터에 인증

네트워크 연결 해제 중에는 AWS Identity and Access Management(IAM)을 사용할 수 없습니다. 연결이 해제된 동안에는 IAM 자격 증명을 사용하여 로컬 클러스터에 인증할 수 없습니다. 그러나 연결이 해

제되면 x509 인증서를 사용하여 로컬 네트워크를 통해 클러스터에 연결할 수 있습니다. 연결 해제 중 사용할 클라이언트 X509 인증서를 다운로드하여 저장해야 합니다. 이 주제에서는 연결 해제된 상태일 때 클러스터에 인증하기 위해 인증서를 생성하고 사용하는 방법을 알아봅니다.

1. 인증서 서명 요청을 생성합니다.

   a. 인증서 서명 요청을 생성합니다.

   ```
   openssl req -new -newkey rsa:4096 -nodes -days 365 \
       -keyout admin.key -out admin.csr -subj "/CN=admin"
   ```

   b. Kubernetes에서 인증서 서명 요청을 생성합니다.

   ```
   BASE64_CSR=$(cat admin.csr | base64 -w 0)
   cat << EOF > admin-csr.yaml
   apiVersion: certificates.k8s.io/v1
   kind: CertificateSigningRequest
   metadata:
     name: admin-csr
   spec:
     signerName: kubernetes.io/kube-apiserver-client
     request: ${BASE64_CSR}
     usages:
     - client auth
   EOF
   ```

2. kubectl을 사용하여 인증서 서명 요청을 생성합니다.

   ```
   kubectl create -f admin-csr.yaml
   ```

3. 인증서 서명 요청의 상태를 확인합니다.

   ```
   kubectl get csr admin-csr
   ```

   예제 출력은 다음과 같습니다.

   ```
   NAME       AGE   REQUESTOR                      CONDITION
   admin-csr  11m   kubernetes-admin               Pending
   ```

   Kubernetes에서 인증서 서명 요청을 생성했습니다.

4. 인증서 서명 요청을 승인합니다.

```
kubectl certificate approve admin-csr
```

5. 인증서 서명 요청의 승인 상태를 다시 확인합니다.

```
kubectl get csr admin-csr
```

예제 출력은 다음과 같습니다.

```
NAME        AGE   REQUESTOR                      CONDITION
admin-csr   11m   kubernetes-admin               Approved
```

6. 인증서를 검색하고 확인합니다.

a. 인증서를 검색합니다.

```
kubectl get csr admin-csr -o jsonpath='{.status.certificate}' | base64 --decode
 > admin.crt
```

b. 인증서를 확인합니다.

```
cat admin.crt
```

7. admin 사용자에 대한 클러스터 역할 바인딩을 생성합니다.

```
kubectl create clusterrolebinding admin --clusterrole=cluster-admin \
    --user=admin --group=system:masters
```

8. 연결 해제된 상태에 대한 사용자 범위 kubeconfig를 생성합니다.

다운로드한 admin 인증서를 사용하여 kubeconfig 파일을 생성할 수 있습니다. 다음 명령에서 *my-cluster*와 *apiserver-endpoint*를 바꿉니다.

```
aws eks describe-cluster --name my-cluster \
    --query "cluster.certificateAuthority" \
    --output text | base64 --decode > ca.crt
```

```
kubectl config --kubeconfig admin.kubeconfig set-cluster my-cluster \
    --certificate-authority=ca.crt --server apiserver-endpoint --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-credentials admin \
    --client-certificate=admin.crt --client-key=admin.key --embed-certs
```

```
kubectl config --kubeconfig admin.kubeconfig set-context admin@my-cluster \
    --cluster my-cluster --user admin
```

```
kubectl config --kubeconfig admin.kubeconfig use-context admin@my-cluster
```

9. kubeconfig 파일을 봅니다.

```
kubectl get nodes --kubeconfig admin.kubeconfig
```

10. Outpost에서 서비스가 이미 프로덕션에 있는 경우 이 단계를 건너뜁니다. Amazon EKS가 Outpost에서 실행되는 유일한 서비스이고 Outpost가 현재 프로덕션에 없는 경우 네트워크 연결 해제를 시뮬레이션할 수 있습니다. 로컬 클러스터를 사용하여 프로덕션에 들어가기 전에 연결 해제를 시뮬레이션하여 연결 해제된 상태에서도 클러스터에 계속 액세스할 수 있는지 확인해야 합니다.

    a. Outpost를 AWS 리전에 연결하는 네트워킹 디바이스에 방화벽 규칙을 적용합니다. 그러면 Outpost의 서비스 링크 연결이 해제됩니다. 새 인스턴스를 생성할 수 없습니다. 현재 실행 중인 인스턴스는 AWS 리전 및 인터넷 연결이 끊어집니다.

    b. x509 인증서를 사용하여 연결이 해제된 동안 로컬 클러스터 연결을 테스트할 수 있습니다. kubeconfig를 이전 단계에서 생성한 admin.kubeconfig로 변경해야 합니다. *my-cluster*를 로컬 클러스터의 이름으로 바꿉니다.

    ```
    kubectl config use-context admin@my-cluster --kubeconfig admin.kubeconfig
    ```

    연결 해제된 상태에서 로컬 클러스터에 문제가 있는 경우 지원 티켓을 여는 것이 좋습니다.

용량 고려 사항

이 주제에서는 Kubernetes 컨트롤 플레인 인스턴스 유형 선택 및 Outpost의 로컬 Amazon EKS 클러스터에 대한 고가용성 요구 사항을 충족하기 위한 배치 그룹 사용(선택 사항)에 대한 지침을 제공합니다.

Outposts에서 로컬 클러스터의 Kubernetes 컨트롤 플레인에 사용할 인스턴스 유형(예: m5, c5 또는 r5)을 선택하기 전에 Outpost 구성에서 사용 가능한 인스턴스 유형을 확인합니다. 사용 가능한 인스턴

스 유형 식별 후 워크로드에 필요한 노드 수에 따라 인스턴스 크기(예: `large`, `xlarge` 또는 `2xlarge`)를 선택합니다. 다음 표에는 인스턴스 크기 선택에 대한 권장 사항이 나와 있습니다.

> **Note**
>
> 인스턴스 크기는 Outposts에 배정되어 있어야 합니다. 로컬 클러스터의 수명 동안 Outposts에서 사용할 수 있는 크기의 인스턴스 3개에 대한 용량이 충분한지 확인합니다. 사용 가능한 Amazon EC2 인스턴스 유형 목록은 AWS Outposts 랙 기능의 컴퓨팅 및 스토리지 섹션을 참조하세요.

노드 수	Kubernetes 컨트롤 플레인 인스턴스 크기
1~20	`large`
21~100	`xlarge`
101~250	`2xlarge`
251~500	`4xlarge`

Kubernetes 컨트롤 플레인용 스토리지에는 `etcd`의 필수 IOPS를 충족하기 위해 로컬 클러스터마다 246GB의 Amazon EBS 스토리지가 필요합니다. 로컬 클러스터가 생성될 때 Amazon EBS 볼륨이 자동으로 프로비저닝됩니다.

컨트롤 플레인 배치

`OutpostConfig.ControlPlanePlacement.GroupName` 속성으로 배치 그룹을 지정하지 않으면 Kubernetes 컨트롤 플레인용으로 프로비저닝된 Amazon EC2 인스턴스에는 Outpost에서 사용할 수 있는 기본 용량에 대한 특정 하드웨어 배치가 적용되지 않습니다.

배치 그룹을 사용하여 Outpost의 로컬 Amazon EKS 클러스터에 대한 고가용성 요구 사항을 충족할 수 있습니다. 클러스터 생성 중에 배치 그룹을 지정하면 Kubernetes 컨트롤 플레인 인스턴스의 배치에 영향을 줍니다. 인스턴스가 독립적인 기본 하드웨어(랙 또는 호스트) 전체에 분산되어 하드웨어 장애 발생 시 상관 관계가 있는 인스턴스 영향을 최소화합니다.

요구 사항

구성할 수 있는 분산 유형은 배포되어 있는 Outpost 랙의 수에 따라 다릅니다.

- 하나의 논리적 Outpost에 1개 또는 2개의 물리적 랙이 있는 배포 – Kubernetes 컨트롤 플레인 인스턴스에 대해 선택하는 인스턴스 유형으로 구성되는 호스트가 3개 이상 있어야 합니다. 호스트 수준 분산을 사용하는 분산 배치 그룹은 모든 Kubernetes 컨트롤 플레인 인스턴스가 Outpost 배포에서 사용할 수 있는 기본 랙 내의 개별 호스트에서 실행되도록 합니다.
- 하나의 논리적 Outpost에 3개 이상의 물리적 랙이 있는 배포 – Kubernetes 컨트롤 플레인 인스턴스에 대해 선택하는 인스턴스 유형으로 구성된 호스트가 3개 이상 있어야 합니다. 랙 수준 분산을 사용하는 분산 배치 그룹은 모든 Kubernetes 컨트롤 플레인 인스턴스가 Outpost 배포의 개별 랙에서 실행되도록 합니다. 이전 옵션에서 설명한 대로 호스트 수준 분산 배치 그룹을 사용할 수도 있습니다.

원하는 배치 그룹을 생성하는 것은 본인 책임입니다. `CreateCluster` API를 호출할 때 배치 그룹을 지정합니다. 배치 그룹과 생성 방법에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 배치 그룹을 참조하세요.

고려 사항

- 배치 그룹을 지정할 때 로컬 Amazon EKS 클러스터를 생성하려면 Outpost에 사용 가능한 배정된 용량이 있어야 합니다. 용량은 호스트 유형을 사용하는지 아니면 랙 분산 유형을 사용하는지에 따라 달라집니다. 용량이 부족하면 클러스터가 `Creating` 상태로 유지됩니다. `DescribeCluster` API 응답의 상태 필드에서 `Insufficient Capacity Error`를 확인할 수 있습니다. 생성 프로세스를 진행하려면 용량을 확보해야 합니다.
- Amazon EKS 로컬 클러스터 플랫폼 및 버전 업데이트 중에는 클러스터의 Kubernetes 컨트롤 플레인 인스턴스가 롤링 업데이트 전략을 통해 새 인스턴스로 바뀝니다. 이 대체 프로세스 중에 각 컨트롤 플레인 인스턴스가 종료되어 해당 슬롯이 비워집니다. 업데이트한 새 인스턴스가 대신에 프로비저닝됩니다. 업데이트한 인스턴스는 릴리스된 슬롯에 배치될 수도 있습니다. 관련 없는 다른 인스턴스에서 슬롯을 사용하고 필요한 분산 토폴로지 요구 사항을 충족하는 용량이 더는 남아 있지 않으면 클러스터가 `Updating` 상태로 유지됩니다. `DescribeCluster` API 응답의 상태 필드에서 각 `Insufficient Capacity Error`를 확인할 수 있습니다. 업데이트 프로세스를 진행하고 이전의 고가용성 수준을 다시 설정할 수 있도록 용량을 확보해야 합니다.
- 각 AWS 리전에서 계정당 최대 500개의 배치 그룹을 생성할 수 있습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 일반 규칙 및 제한 사항을 참조하세요.

AWS Outposts의 Amazon EKS에 대한 로컬 클러스터 문제 해결

이 주제에서는 로컬 클러스터를 사용하는 동안 발생할 수 있는 몇 가지 일반적 오류와 문제를 해결하는 방법을 설명합니다. 로컬 클러스터는 클라우드의 Amazon EKS 클러스터와 유사하지만, Amazon EKS 서비스에서 관리하는 방식에는 몇 가지 차이점이 있습니다.

API 동작

로컬 클러스터는 Amazon EKS API를 통해 생성되지만, 비동기 방식으로 실행됩니다. 즉, Amazon EKS API에 대한 요청이 로컬 클러스터에 즉시 반환됩니다. 그러나 이러한 요청은 성공하거나, 입력 검증 오류로 인해 빠르게 실패하거나, 실패하고 설명이 포함된 검증 오류가 있을 수 있습니다. 이 동작은 Kubernetes API와 유사합니다.

로컬 클러스터는 `FAILED` 상태로 전환되지 않습니다. Amazon EKS에서는 클러스터 상태를 사용자가 요청한 원하는 상태와 지속적으로 조정하려고 시도합니다. 따라서 로컬 클러스터가 근본적인 문제가 해결될 때까지 장기간 `CREATING` 상태로 유지될 수도 있습니다.

클러스터 상태 필드 설명

로컬 클러스터 문제는 `describe-cluster` Amazon EKS AWS CLI 명령을 사용하여 검색할 수 있습니다. 로컬 클러스터 문제는 `describe-cluster` 명령 응답의 `cluster.health` 필드에 표시됩니다. 이 필드에 포함된 메시지에는 오류 코드, 설명이 포함된 메시지 및 관련 리소스 ID가 포함됩니다. 이 정보는 Amazon EKS API와 AWS CLI를 통해서만 제공됩니다. 다음 예제에서 *my-cluster*를 로컬 클러스터의 이름으로 바꿉니다.

```
aws eks describe-cluster --name my-cluster --query 'cluster.health'
```

예제 출력은 다음과 같습니다.

```
{
    "issues": [
        {
            "code": "ConfigurationConflict",
            "message": "The instance type 'm5.large' is not supported in Outpost 'my-
outpost-arn'.",
            "resourceIds": [
                "my-cluster-arn"
            ]
        }
    ]
}
```

문제를 복구할 수 없는 경우 로컬 클러스터를 삭제하고 새 클러스터를 생성해야 할 수 있습니다. 예를 들어 Outpost에서 사용할 수 없는 인스턴스 유형으로 클러스터를 프로비저닝하려고 합니다. 다음 표에는 일반적인 상태 관련 오류가 포함되어 있습니다.

오류 시나리오	코드	메시지	ResourceIds
제공된 서브넷을 찾을 수 없습니다.	ResourceNotFound	The subnet ID *subnet-id* does not exist	제공된 모든 서브넷 ID
제공된 서브넷이 동일한 VPC에 속하지 않습니다.	ConfigurationConflict	Subnets specified must belong to the same VPC	제공된 모든 서브넷 ID
제공된 일부 서브넷이 지정된 Outpost에 속하지 않습니다.	ConfigurationConflict	Subnet *subnet-id* expected to be in *outpost-arn* , but is in *other-outpost-arn*	문제가 있는 서브넷 ID
제공된 일부 서브넷이 어느 Outpost에도 속하지 않습니다.	ConfigurationConflict	Subnet *subnet-id* is not part of any Outpost	문제가 있는 서브넷 ID
제공된 일부 서브넷에 컨트롤 플레인 인스턴스용 탄력적 네트워크 인터페이스를 생성하기에 충분한 여유 주소가 없습니다.	ResourceLimitExceeded	The specified subnet does not have enough free addresses to satisfy the request.	문제가 있는 서브넷 ID
지정된 컨트롤 플레인 인스턴스 유형이 Outpost에서 지원되지 않습니다.	ConfigurationConflict	The instance type *type* is not supported in Outpost *outpost-arn*	클러스터 ARN
컨트롤 플레인 Amazon EC2 인스턴스를 종료했거나 run-instance 에	InternalFailure	EC2 instance state "Terminated" is unexpected	클러스터 ARN

오류 시나리오	코드	메시지	ResourceIds
성공했지만, 관찰된 상태가 `Terminated` 로 변경됩니다. 이는 Outpost가 다시 연결되고 Amazon EBS 내부 오류로 인해 Amazon EC2 내부 워크플로가 실패한 후 일정 기간 동안 발생할 수 있습니다.			
Outpost의 용량이 부족합니다. 이는 Outpost가 AWS 리전에서 연결 해제된 경우 클러스터를 생성 중일 때 발생할 수도 있습니다.	`ResourceLimitExceeded`	`There is not enough capacity on the Outpost to launch or start the instance.`	클러스터 ARN
계정에서 보안 그룹 할당량을 초과했습니다.	`ResourceLimitExceeded`	Amazon EC2 API에서 반환된 오류 메시지	대상 VPC ID
계정에서 탄력적 네트워크 인터페이스 할당량을 초과했습니다.	`ResourceLimitExceeded`	Amazon EC2 API에서 반환된 오류 메시지	대상 서브넷 ID

오류 시나리오	코드	메시지	ResourceIds
컨트롤 플레인 인스턴스가 AWS Systems Manager를 통해 연결되지 않았습니다. 해결 방법은 컨트롤 플레인 인스턴스는 AWS Systems Manager를 통해 연결할 수 없습니다. 섹션을 참조하세요.	`ClusterUnreachable`	Amazon EKS control plane instances are not reachable through SSM. Please verify your SSM and network configuration, and reference the EKS on Outposts troubleshooting documentation.(Amazon EKS 컨트롤 플레인 인스턴스는 SSM을 통해 연결할 수 없습니다. SSM 및 네트워크 구성을 확인하고 Outposts의 EKS 문제 해결 문서를 참조하세요.)	Amazon EC2 인스턴스 ID
관리형 보안 그룹 또는 탄력적 네트워크 인터페이스에 대한 세부 정보를 가져오는 동안 오류가 발생했습니다.	Amazon EC2 클라이언트 오류 코드 기준	Amazon EC2 API에서 반환된 오류 메시지	모든 관리형 보안 그룹 ID
보안 그룹 수신 규칙을 승인하거나 취소하는 동안 오류가 발생했습니다. 이는 클러스터 및 컨트롤 플레인 보안 그룹 모두에 적용됩니다.	Amazon EC2 클라이언트 오류 코드 기준	Amazon EC2 API에서 반환된 오류 메시지	문제가 있는 보안 그룹 ID

오류 시나리오	코드	메시지	ResourceIds
컨트롤 플레인 인스턴스에 대한 탄력적 네트워크 인터페이스를 삭제하는 동안 오류가 발생했습니다.	Amazon EC2 클라이언트 오류 코드 기준	Amazon EC2 API에서 반환된 오류 메시지	문제가 있는 탄력적 네트워크 인터페이스 ID

다음 표에는 `describe-cluster` 응답의 상태 필드에 표시되는 다른 AWS 서비스의 오류가 나열되어 있습니다.

Amazon EC2 오류 코드	클러스터 상태 문제 코드	설명
`AuthFailure`	`AccessDenied`	이 오류는 여러 가지 이유로 발생할 수 있습니다. 가장 일반적인 이유는 서비스 연결 역할 정책의 범위를 좁히기 위해 서비스에서 사용하는 태그를 컨트롤 플레인에서 실수로 제거했기 때문입니다. 이렇게 되면 Amazon EKS에서 더는 이러한 AWS 리소스를 관리하고 모니터링할 수 없습니다.
`UnauthorizedOperation`	`AccessDenied`	이 오류는 여러 가지 이유로 발생할 수 있습니다. 가장 일반적인 이유는 서비스 연결 역할 정책의 범위를 좁히기 위해 서비스에서 사용하는 태그를 컨트롤 플레인에서 실수로 제거했기 때문입니다. 이렇게 되면 Amazon EKS에서 더는 이러한 AWS 리소스를 관리하고 모니터링할 수 없습니다.

Amazon EC2 오류 코드	클러스터 상태 문제 코드	설명
InvalidSubnetID.NotFound	ResourceNotFound	이 오류는 보안 그룹의 수신 규칙에 대한 서브넷 ID를 찾을 수 없을 때 발생합니다.
InvalidPermission.NotFound	ResourceNotFound	이 오류는 보안 그룹의 수신 규칙에 대한 권한이 올바르지 않을 때 발생합니다.
InvalidGroup.NotFound	ResourceNotFound	이 오류는 보안 그룹의 수신 규칙 그룹을 찾을 수 없을 때 발생합니다.
InvalidNetworkInterfaceID.NotFound	ResourceNotFound	이 오류는 보안 그룹의 수신 규칙에 대한 네트워크 인터페이스 ID를 찾을 수 없을 때 발생합니다.
InsufficientFreeAddressesInSubnet	ResourceLimitExceeded	이 오류는 서브넷 리소스 할당량을 초과했을 때 발생합니다.
InsufficientCapacityOnOutpost	ResourceLimitExceeded	이 오류는 출력 용량 할당량을 초과했을 때 발생합니다.
NetworkInterfaceLimitExceeded	ResourceLimitExceeded	이 오류는 탄력적 네트워크 인터페이스 할당량을 초과했을 때 발생합니다.
SecurityGroupLimitExceeded	ResourceLimitExceeded	이 오류는 보안 그룹 할당량을 초과했을 때 발생합니다.

Amazon EC2 오류 코드	클러스터 상태 문제 코드	설명
VcpuLimitExceeded	ResourceLimitExceeded	이는 새 계정에서 Amazon EC2 인스턴스를 생성할 때 관찰됩니다. 오류는 다음과 비슷할 수 있습니다. "You have requested more vCPU capacity than your current vCPU limit of 32 allows for the instance bucket that the specified instance type belongs to. Please visit http://aws.amazon.com/contact-us/ec2-request to request an adjustment to this limit."
InvalidParameterValue	ConfigurationConflict	지정된 인스턴스 유형이 Outpost에서 지원되지 않는 경우 Amazon EC2가 이 오류 코드를 반환합니다.
기타 모든 오류	InternalFailure	None

클러스터를 생성하거나 수정할 수 없음

클라우드에서 호스팅되는 Amazon EKS 클러스터와 다른 권한과 정책이 로컬 클러스터에 필요합니다. 클러스터가 생성되지 않고 InvalidPermissions 오류가 발생하면 사용 중인 클러스터 역할에 AmazonEKSLocalOutpostClusterPolicy 관리형 정책이 연결되어 있는지 다시 확인합니다. 다른 모든 API 호출에는 클라우드의 Amazon EKS 클러스터와 동일한 권한 집합이 필요합니다.

클러스터가 **CREATING** 상태에서 멈춤

로컬 클러스터 생성에 걸리는 시간은 여러 가지 요인에 따라 다릅니다. 네트워크 구성, Outpost 구성 및 클러스터의 구성이 이러한 요인에 포함됩니다. 일반적으로 15~20분 이내에 로컬 클러스터가 생성되고

ACTIVE 상태로 변경됩니다. 로컬 클러스터가 CREATING 상태로 유지되는 경우 cluster.health 출력 필드에서 원인에 대한 정보의 describe-cluster를 호출할 수 있습니다.

가장 일반적인 문제는 다음과 같습니다.

AWS Systems Manager(Systems Manager)에서 다음과 같은 문제가 발생합니다.

- 클러스터에서 Systems Manager가 있는 AWS 리전의 컨트롤 플레인 인스턴스에 연결할 수 없습니다. 리전 내 Bastion Host에서 `aws ssm start-session --target instance-id`를 호출하여 이를 확인할 수 있습니다. 명령이 작동하지 않으면 Systems Manager가 컨트롤 플레인 인스턴스에서 실행 중인지 확인합니다. 클러스터를 삭제한 다음에 다시 생성하여 해결하는 방법도 있습니다.
- Systems Manager 컨트롤 플레인 인스턴스에 인터넷 액세스 권한이 없을 수도 있습니다. 클러스터를 생성할 때 제공한 서브넷에 NAT 게이트웨이와 VPC(인터넷 게이트웨이 포함)가 있는지 확인합니다. VPC 연결성 분석기를 사용하여 컨트롤 플레인 인스턴스가 인터넷 게이트웨이에 연결할 수 있는지 확인합니다. 자세한 내용을 알아보려면 Getting started with VPC Reachability Analyzer(VPC Reachability Analyzer 시작하기)를 참조하세요.
- 제공한 역할 ARN에 정책이 없습니다. 역할에서 AWS 관리형 정책: AmazonEKSLocalOutpostClusterPolicy가 제거되었는지 확인합니다. AWS CloudFormation 스택이 잘못 구성된 경우에도 이 문제가 발생할 수 있습니다.

클러스터가 생성될 때 여러 서브넷이 잘못 구성되고 지정되었음:

- 제공된 모든 서브넷이 동일한 Outpost와 연결되어야 하며 서로 연결되어야 합니다. 클러스터가 생성될 때 여러 서브넷이 지정되면 Amazon EKS에서는 컨트롤 플레인 인스턴스를 여러 서브넷에 분산하려고 시도합니다.
- Amazon EKS 관리형 보안 그룹은 탄력적 네트워크 인터페이스에 적용됩니다. 그러나 NACL 방화벽 규칙과 같은 기타 구성 요소가 탄력적 네트워크 인터페이스의 규칙과 충돌할 수도 있습니다.

VPC 및 서브넷 DNS 구성이 잘못 구성되었거나 누락됨

Amazon EKS 로컬 클러스터 VPC 및 서브넷 요구 사항과 고려 사항 섹션을 검토합니다.

클러스터에 노드를 조인할 수 없음

일반적인 원인:

- AMI 문제:

  - 지원되지 않는 AMI를 사용하고 있습니다. Amazon EKS 최적화 Amazon Linux AMI Amazon EKS 최적화 Amazon Linux에 v20220620 이상을 사용해야 합니다.
  - AWS CloudFormation 템플릿을 사용하여 노드를 생성한 경우 지원되지 않는 AMI를 사용하고 있지 않은지 확인합니다.
- AWS IAM Authenticator `ConfigMap` 누락 – 누락된 경우 생성해야 합니다. 자세한 정보는 클러스터에 `aws-auth` `ConfigMap` 적용을 참조하세요.
- 잘못된 보안 그룹 사용 - 워커 노드의 보안 그룹에 `eks-cluster-sg-`*`cluster-name`*`-`*`uniqueid`*를 사용해야 합니다. 선택한 보안 그룹은 스택이 사용될 때마다 새 보안 그룹을 허용하도록 AWS CloudFormation에 의해 변경됩니다.
- 예상치 못한 프라이빗 링크 VPC 단계 수행 – 잘못된 CA 데이터(`--b64-cluster-ca`) 또는 API 엔드포인트(`--apiserver-endpoint`)가 전달됩니다.
- 잘못 구성된 Pod 보안 정책:
  - 노드가 클러스터와 조인하고 통신하도록 노드에서 CoreDNS 및 Amazon VPC CNI plugin for Kubernetes Daemonsets가 실행되어야 합니다.
  - Amazon VPC CNI plugin for Kubernetes가 제대로 작동하려면 일부 권한 있는 네트워킹 기능이 필요합니다. `kubectl describe psp eks.privileged` 명령을 사용하여 권한 있는 네트워킹 기능을 볼 수 있습니다.

  기본 포드 보안 정책을 수정하지 않는 것이 좋습니다. 자세한 내용은 포드 보안 정책 섹션을 참조하세요.

로그 수집

Outpost가 연결된 AWS 리전에서 연결 해제되면 Kubernetes 클러스터는 정상적으로 계속 작동할 수 있습니다. 단, 클러스터가 제대로 작동하지 않는 경우 네트워크 연결 해제 준비의 문제 해결 단계를 따릅니다. 기타 문제가 발생하면 AWS Support에 문의하세요. AWS Support에서는 로그 수집 도구를 다운로드하고 실행하는 방법을 안내할 수 있습니다. 그렇게 하면 Kubernetes 클러스터 컨트롤 플레인 인스턴스에서 로그를 수집하여 추가 조사를 위해 AWS Support 지원에 보낼 수 있습니다.

컨트롤 플레인 인스턴스는 AWS Systems Manager를 통해 연결할 수 없습니다.

AWS Systems Manager(Systems Manager)를 통해 Amazon EKS 컨트롤 플레인 인스턴스에 연결할 수 없는 경우 Amazon EKS는 클러스터에 대해 다음 오류를 표시합니다.

```
Amazon EKS control plane instances are not reachable through SSM. Please verify your
 SSM and network configuration, and reference the EKS on Outposts troubleshooting
 documentation.
```

이 문제를 해결하려면 VPC와 서브넷이 Amazon EKS 로컬 클러스터 VPC 및 서브넷 요구 사항과 고려 사항의 요구 사항을 충족하고 AWS Systems Manager 사용 설명서의 Session Manager 설정 단계를 완료했는지 확인합니다.

Outpost에서 자체 관리형 Amazon Linux 노드 시작하기

이 주제에서는 Amazon EKS 클러스터에 등록하는 Outpost에서 Amazon Linux 노드의 Auto Scaling을 시작하는 방법을 설명합니다. 클러스터는 AWS 클라우드 또는 Outpost에 있을 수 있습니다.

필수 조건

- 기존 Outpost. 자세한 내용은 AWS Outposts란 무엇인가요?를 참조하세요.
- 기존 Amazon EKS 클러스터. AWS 클라우드에 클러스터를 배포하려면 Amazon EKS 클러스터 생성 부분을 참조하세요. Outpost에 클러스터를 배포하려면 AWS Outposts의 Amazon EKS용 로컬 클러스터 부분을 참조하세요.
- AWS 클라우드의 클러스터에서 노드를 생성 중이며 활성화된 AWS Outposts, AWS Wavelength 또는 AWS 로컬 영역이 있는 AWS 리전에 서브넷이 있다고 가정합니다. 그러면 클러스터를 생성할 때 해당 서브넷이 전달되지 않았어야 합니다. Outpost의 클러스터에 노드를 생성하는 경우 클러스터 생성 시 Outpost 서브넷을 전달해야 합니다.
- (AWS 클라우드의 클러스터에 권장됨) 필요한 IAM 정책이 연결된 자체 IAM 역할로 구성된 Amazon VPC CNI plugin for Kubernetes 추가 기능입니다. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오. 로컬 클러스터는 서비스 계정에 대한 IAM 역할을 지원하지 않습니다.

`eksctl` 또는 AWS Management Console(AWS CloudFormation 템플릿 사용)을 사용하여 자체 관리형 Amazon Linux 노드 그룹을 생성할 수 있습니다. Terraform도 사용할 수 있습니다.

eksctl

전제 조건

디바이스 또는 `0.175.0`에 설치된 버전 AWS CloudShell 이상의 `eksctl` 명령줄 도구. `eksctl`을 설치 또는 업그레이드하려면 `eksctl` 설명서에서 Installation을 참조하세요.

eksctl을 사용하여 자체 관리형 Linux 노드를 시작하려면 다음을 수행합니다.

1. 클러스터가 AWS 클라우드에 있고 AmazonEKS_CNI_Policy 관리형 IAM 정책이 Amazon EKS 노드 IAM 역할에 연결된 경우 대신 Kubernetes aws-node 서비스 계정에 연결하는 IAM 역할에 할당하는 것이 좋습니다. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오. 클러스터가 Outpost에 있는 경우 정책이 노드 역할에 연결되어야 합니다.
2. 다음 명령은 기존 클러스터의 노드 그룹을 생성합니다. eksctl을 사용하여 클러스터가 생성되었어야 합니다. *al-nodes*을 노드 그룹의 이름으로 바꿉니다. 노드 그룹 이름은 63자를 초과할 수 없습니다. 문자나 숫자로 시작하되, 나머지 문자의 경우 하이픈과 밑줄을 포함할 수 있습니다. *my-cluster*를 클러스터 이름으로 바꿉니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다. 클러스터가 Outpost에 있는 경우 *id*를 Outpost 서브넷의 ID로 바꿉니다. 클러스터가 AWS 클라우드에 있는 경우 *id*를 클러스터를 생성할 때 지정하지 않은 서브넷의 ID로 바꿉니다. instance-type을 Outpost에서 지원하는*instance-type* 인스턴스 유형으로 바꿉니다. 나머지 *example values*를 고유한 값으로 바꿉니다. 노드는 기본적으로 제어 영역과 동일한 Kubernetes 버전으로 작성됩니다.

   *instance-type*를 Outpost에서 사용 가능한 인스턴스 유형으로 바꿉니다.

   *my-key*를 Amazon EC2 키 페어 또는 퍼블릭 키 이름으로 바꿉니다. 이 키는 노드를 시작한 후 SSH로 연결하는 데 사용됩니다. Amazon EC2 키 페어가 아직 없는 경우 AWS Management Console에서 새로 생성할 수 있습니다. 자세한 내용을 알아보려면 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하세요.

   다음 명령을 사용하여 노드 그룹을 생성합니다.

   ```
   eksctl create nodegroup --cluster my-cluster --name al-nodes --node-
   type instance-type \
       --nodes 3 --nodes-min 1 --nodes-max 4 --managed=false --node-volume-type gp2
    --subnet-ids subnet-id
   ```

   AWS 클라우드에 클러스터가 배포된 경우:

   - 배포하는 노드 그룹에서 인스턴스의 블록과 다른 CIDR 블록의 Pods에 IPv4 주소를 할당할 수 있습니다. 자세한 내용은 포드에 대한 사용자 지정 네트워킹 단원을 참조하십시오.
   - 배포하는 노드 그룹에는 아웃바운드 인터넷 액세스 권한이 필요하지 않습니다. 자세한 내용은 프라이빗 클러스터 요구 사항 단원을 참조하십시오.

사용 가능한 모든 옵션과 기본값의 전체 목록은 `eksctl` 설명서의 AWS Outposts 지원을 참조하세요.

노드가 클러스터에 조인하지 못하는 경우 Amazon EKS 문제 해결의 노드가 클러스터 조인에 실패 섹션과 AWS Outposts의 Amazon EKS에 대한 로컬 클러스터 문제 해결의 클러스터에 노드를 조인할 수 없음 섹션을 참조하세요.

예제 출력은 다음과 같습니다. 노드가 생성되는 동안 여러 줄이 출력됩니다. 출력의 마지막 줄 중 하나는 다음 예제 줄과 유사합니다.

```
[#]  created 1 nodegroup(s) in cluster "my-cluster"
```

3. (선택 사항) 샘플 애플리케이션을 배포하여 클러스터 및 Linux 노드를 테스트합니다.

AWS Management Console

1단계: AWS Management Console을 사용하여 자체 관리형 Amazon Linux 노드 시작

1. AWS CloudFormation 템플릿의 최신 버전 다운로드

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/cloudformation/2022-12-23/amazon-eks-nodegroup.yaml
```

2. AWS CloudFormation 콘솔(https://console.aws.amazon.com/cloudformation)을 엽니다.
3. 스택 생성(Create stack)을 선택한 다음 새 리소스 사용(표준)(With new resources(standard))을 선택합니다.
4. 템플릿 지정(Specify template)에서 템플릿 파일 업로드(Upload a template file)를 선택한 다음 파일 선택(Choose file)을 선택합니다. 이전 단계에서 다운로드한 `amazon-eks-nodegroup.yaml` 파일을 선택한 후 Next(다음)를 선택합니다.
5. 스택 세부 정보 지정(Specify stack details) 페이지에서 이에 따라 다음 파라미터를 입력한 후 다음(Next)을 선택합니다.
   - 스택 이름: AWS CloudFormation 스택에 대한 스택 이름을 선택합니다. 예를 들어 ***al-nodes***라고 할 수 있습니다. 이름에는 영숫자(대소문자 구분)와 하이픈만 사용할 수 있습니다. 영문자로 시작해야 하며 100자 이하여야 합니다.
   - ClusterName: 클러스터의 이름을 입력합니다. 이 이름이 클러스터 이름과 일치하지 않으면 노드가 클러스터에 조인할 수 없습니다.

- ClusterControlPlaneSecurityGroup: VPC를 생성할 때 생성한 AWS CloudFormation 출력에서 SecurityGroups값을 선택합니다.

  다음 단계에서는 해당 그룹을 검색하는 한 가지 작업을 보여줍니다.

  1. https://console.aws.amazon.com/eks/home#/clusters에서 Amazon EKS 콘솔을 엽니다.
  2. 클러스터의 이름을 선택합니다.
  3. 네트워킹 탭을 선택합니다.
  4. ClusterControlPlaneSecurityGroup 드롭다운 목록에서 선택할 때 추가 보안 그룹(Additional Security Group) 값을 참조로 사용하세요.
- NodeGroupName: 노드 그룹의 이름을 입력합니다. 이 이름은 나중에 노드에 대해 생성된 Auto Scaling 노드 그룹을 식별하는 데 사용할 수 있습니다.
- NodeAutoScalingGroupMinSize: Auto Scaling 그룹이 축소할 수 있는 노드의 최소 노드 수를 입력합니다.
- NodeAutoScalingGroupDesiredCapacity: 스택을 생성할 때 조정할 원하는 노드 수를 입력합니다.
- NodeAutoScalingGroupMaxSize: Auto Scaling 그룹이 확장할 수 있는 노드의 최대 노드 수를 입력합니다.
- NodeInstanceType: 노드에 대한 인스턴스 유형을 선택합니다. 클러스터가 AWS 클라우드에서 실행 중인 경우 자세한 내용을 알아보려면 Amazon EC2 인스턴스 유형 선택 부분을 참조하세요. 클러스터가 Outpost에서 실행 중인 경우 Outpost에서 사용할 수 있는 인스턴스 유형만 선택할 수 있습니다.
- NodeImageIdSSMParam: 가변 Kubernetes 버전용 최신 Amazon EKS 최적화 AMI의 Amazon EC2 Systems Manager 파라미터로 미리 채워집니다. Amazon EKS에서 지원되는 다른 Kubernetes 마이너 버전을 사용하려면 *1.XX*를 다른 지원되는 버전으로 바꿉니다. 클러스터와 동일한 Kubernetes 버전을 지정하는 것이 좋습니다.

  Amazon EKS 최적화 가속 AMI를 사용하려면 *amazon-linux-2*를 **amazon-linux-2-gpu**로 바꿉니다. Amazon EKS 최적화 Arm AMI를 사용하려면 *amazon-linux-2*를 **amazon-linux-2-arm64**로 바꿉니다.

  > **Note**
  >
  > Amazon EKS 노드 AMI는 Amazon Linux를 기반으로 합니다. Amazon Linux 보안 센터에서 Amazon Linux 2의 보안 또는 프라이버시 이벤트를 추적하거나 관련 RSS 피

드를 구독하세요. 보안 및 프라이버시 이벤트에는 문제의 개요, 영향을 받는 패키지 및 인스턴스를 업데이트하여 문제를 해결하는 방법이 포함됩니다.

- NodeImageId: (선택 사항) Amazon EKS 최적화 AMI 대신 사용자 정의 AMI를 사용 중인 경우 해당 AWS 리전에 노드 AMI ID를 입력합니다. 여기에서 값을 지정하면 NodeImageIdSSMParam 필드에 있는 모든 값이 재정의됩니다.
- NodeVolumeSize: 노드에 대해 루트 볼륨 크기를 GiB 단위로 지정합니다.
- NodeVolumeType: 노드의 루트 볼륨 유형을 지정합니다.
- KeyName: 시작 이후 SSH를 사용하여 노드에 연결하는 데 사용할 수 있는 Amazon EC2 SSH 키 페어 이름을 입력합니다. Amazon EC2 키 페어가 아직 없는 경우 AWS Management Console에서 새로 생성할 수 있습니다. 자세한 내용을 알아보려면 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하세요.

> **Note**
>
> 여기에 키 페어를 입력하지 않으면 AWS CloudFormation 스택이 생성되지 않습니다.

- BootstrapArguments: 노드에 전달할 수 있는 몇 가지 선택적 인수가 있습니다. 자세한 내용은 GitHub의 부트스트랩 스크립트 사용 정보를 참조하세요. 수신 및 송신 인터넷 연결이 없는 클러스터(프라이빗 클러스터라고도 함)에 노드를 추가하는 경우에는 다음과 같은 부트스트랩 인수를 한 줄로 제공해야 합니다.

```
--b64-cluster-ca ${CLUSTER_CA} --apiserver-endpoint https://
${APISERVER_ENDPOINT} --enable-local-outpost true --cluster-id ${CLUSTER_ID}
```

- DisableIMDSv1: 기본적으로 각 노드는 인스턴스 메타데이터 서비스 버전 1(IMDSv1) 및 IMDSv2를 지원합니다. IMDSv1을 사용 중지할 수 있습니다. 향후 노드 그룹의 노드와 Pods가 IMDSv1을 사용하지 못하게 하려면 DisableIMDSv1을 true로 설정합니다. IMDS에 대한 자세한 내용은 인스턴스 메타데이터 서비스 구성을 참조하세요. 노드의 IMDS 액세스 제한에 대한 자세한 내용은 작업자 노드에 할당된 인스턴스 프로파일에 대한 액세스 제한을 참조하세요.
- VpcId: 생성한 VPC에 대한 ID를 입력합니다. VPC를 선택하기 전에 VPC 요구 사항 및 고려 사항을 검토하세요.
- Subnets(서브넷): 클러스터가 Outpost에 있는 경우 VPC에서 프라이빗 서브넷을 하나 이상 선택합니다. 서브넷을 선택하기 전에 을 검토하세요. 클러스터의 네트워킹 탭에서 각 서브넷 링크를 열면 어떤 서브넷이 프라이빗인지 확인할 수 있습니다.

6. 스택 옵션 구성(Configure stack options) 페이지에서 원하는 선택 항목을 선택한 후 다음(Next)을 선택합니다.
7. AWS CloudFormation에서 IAM 리소스를 생성할 수 있음을 인정합니다.(I acknowledge that might create IAM resources.)의 왼쪽에 있는 확인란을 선택한 다음 스택 생성(Create stack)을 선택합니다.
8. 스택이 생성된 후 콘솔에서 이를 선택하고 출력(Outputs)을 선택합니다.
9. 생성된 노드 그룹에 대해 NodeInstanceRole을 기록합니다. Amazon EKS 노드를 구성할 때 필요합니다.

2단계: 노드가 클러스터에 조인하도록 하려면

1. `aws-auth ConfigMap`이 이미 있는지 확인합니다.

```
kubectl describe configmap -n kube-system aws-auth
```

2. `aws-auth ConfigMap`이 표시되면 필요에 따라 이를 업데이트합니다.

   a. 편집을 위해 `ConfigMap`을 엽니다.

```
kubectl edit -n kube-system configmap/aws-auth
```

   b. 필요에 따라 새 `mapRoles` 항목을 추가합니다. `rolearn` 값을 이전 절차에서 기록한 NodeInstanceRole 값으로 설정합니다.

```
[...]
data:
  mapRoles: |
    - rolearn: <ARN of instance role (not instance profile)>
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
[...]
```

   c. 파일을 저장하고 텍스트 편집기를 종료합니다.

3. "`Error from server (NotFound): configmaps "aws-auth" not found`라는 오류 메시지가 표시되면 스톡 `ConfigMap`을 적용합니다.

   a. 구성 맵을 다운로드합니다.

```
curl -O https://s3.us-west-2.amazonaws.com/amazon-
eks/cloudformation/2020-10-29/aws-auth-cm.yaml
```

b. aws-auth-cm.yaml 파일에서 rolearn을 이전 절차에서 기록한 NodeInstanceRole 값으로 설정합니다. 이 작업은 텍스트 편집기를 사용하거나 *my-node-instance-role*을 대체하고 다음 명령을 실행하여 수행할 수 있습니다.

```
sed -i.bak -e 's|<ARN of instance role (not instance profile)>|my-node-
instance-role|' aws-auth-cm.yaml
```

c. 구성을 적용합니다. 이 명령을 완료하는 데 몇 분이 걸릴 수 있습니다.

```
kubectl apply -f aws-auth-cm.yaml
```

4. 노드의 상태를 확인하고 Ready 상태가 될 때까지 대기합니다.

```
kubectl get nodes --watch
```

Ctrl+C를 입력하여 쉘 프롬프트로 돌아갑니다.

> **Note**
>
> 권한 부여 또는 리소스 유형 오류가 표시되는 경우 문제 해결 주제의 권한이 없거나 액세스가 거부됨(kubectl) 부분을 참조하세요.

노드가 클러스터에 조인하지 못하는 경우 Amazon EKS 문제 해결의 노드가 클러스터 조인에 실패 섹션과 AWS Outposts의 Amazon EKS에 대한 로컬 클러스터 문제 해결의 클러스터에 노드를 조인할 수 없음 섹션을 참조하세요.

5. Amazon EBS CSI 드라이버를 설치합니다. 자세한 내용을 알아보려면 GitHub에서 Installation(설치)을 참조하세요. Set up driver permission(드라이버 권한 설정) 섹션에서 Using IAM instance profile(IAM 인스턴스 프로파일 사용) 옵션에 대한 지침을 따릅니다. gp2 스토리지 클래스를 사용해야 합니다. gp3 스토리지 클래스는 지원되지 않습니다.

   클러스터에 gp2 스토리지 클래스를 생성하려면 다음 단계를 수행합니다.

   1. 다음 명령을 실행해 gp2-storage-class.yaml 파일을 생성합니다.

```
cat >gp2-storage-class.yaml <<EOF
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
  name: ebs-sc
provisioner: ebs.csi.aws.com
volumeBindingMode: WaitForFirstConsumer
parameters:
  type: gp2
  encrypted: "true"
allowVolumeExpansion: true
EOF
```

2. 매니페스트를 클러스터에 적용합니다.

```
kubectl apply -f gp2-storage-class.yaml
```

6. (GPU 노드만 해당) GPU 인스턴스 유형과 Amazon EKS 최적화 가속 AMI를 선택한 경우 클러스터에 Kubernetes용 NVIDIA 디바이스 플러그인을 DaemonSet(으)로 적용해야 합니다. 다음 명령을 실행하기 전에 *vX.X.X*을(를) 원하는 NVIDIA/k8s-device-plugin 버전으로 교체합니다.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-
plugin/vX.X.X/nvidia-device-plugin.yml
```

3단계: 추가 작업

1. (선택 사항) 샘플 애플리케이션을 배포하여 클러스터 및 Linux 노드를 테스트합니다.
2. 클러스터가 Outpost에 배포된 경우 이 단계를 건너뜁니다. 클러스터가 AWS 클라우드에 배포된 경우 다음 정보는 선택 사항입니다. AmazonEKS_CNI_Policy 관리형 IAM 정책이 Amazon EKS 노드 IAM 역할에 연결되어 있는 경우 Kubernetes aws-node 서비스 계정에 연결하는 IAM 역할에 대신 할당하는 것이 좋습니다. 자세한 내용은 서비스 계정용 IAM 역할(IRSA)을 사용하도록 Amazon VPC CNI plugin for Kubernetes 구성 단원을 참조하십시오.

관련 프로젝트

이러한 오픈 소스 프로젝트는 Amazon EKS에서 관리하는 클러스터를 비롯하여 AWS 외부에서 실행되는 Kubernetes 클러스터의 기능을 확장합니다.

관리 도구

Amazon EKS 및 Kubernetes 클러스터용 관련 관리 도구입니다.

eksctl

`eksctl`은 Amazon EKS에 클러스터를 생성하기 위한 간단한 CLI 도구입니다.

- 프로젝트 URL
- 프로젝트 설명서
- AWS 오픈 소스 블로그: eksctl: Amazon EKS cluster with one command

Kubernetes용 AWS 컨트롤러

Kubernetes용 AWS 컨트롤러를 사용하여 AWS 리소스를 Kubernetes 클러스터에서 직접 생성하고 관리할 수 있습니다.

- 프로젝트 URL
- AWS 오픈 소스 블로그: Kubernetes용 AWS 서비스 연산자 이제 사용 가능

Flux CD

Flux는 Git을 사용하여 클러스터 구성을 관리하는 데 사용할 수 있는 도구입니다. 클러스터의 운영자를 사용하여 Kubernetes 내부에서 배포를 트리거합니다. 연산자에 대한 자세한 내용은 GitHub의 OperatorHub.io를 참조하세요.

- 프로젝트 URL
- 프로젝트 설명서

Kubernetes용 CDK

Kubernetes용 CDK(cdk8s)를 사용하면 익숙한 프로그래밍 언어로 Kubernetes 앱과 구성 요소를 정의할 수 있습니다. cdk8s 앱은 표준 Kubernetes 매니페스트로 합성되며 모든 Kubernetes 클러스터에 적용할 수 있습니다.

- 프로젝트 URL
- 프로젝트 설명서
- AWS 컨테이너 블로그: cdk8s+ 소개: Kubernetes 객체를 위한 의도 기반 API

네트워킹

Amazon EKS 및 Kubernetes 클러스터와 관련된 네트워킹 프로젝트입니다.

Amazon VPC CNI plugin for Kubernetes

Amazon EKS는 Amazon VPC CNI plugin for Kubernetes를 통한 기본 VPC 네트워킹을 지원합니다. 이 플러그인은 VPC IP 주소를 각 Pod에 할당합니다.

- 프로젝트 URL
- 프로젝트 설명서

Kubernetes용 AWS Load Balancer Controller

AWS Load Balancer Controller는 Kubernetes 클러스터의 AWS Elastic Load Balancer를 관리합니다. AWS Application Load Balancer를 프로비저닝하여 Kubernetes 수신 리소스를 충족합니다. AWS Network Load Balancer를 프로비저닝하여 Kubernetes 서비스 리소스를 충족합니다.

- 프로젝트 URL
- 프로젝트 설명서

ExternalDNS

ExternalDNS는 노출된 Kubernetes 서비스 및 수신을 Amazon Route 53 및 AWS Service Discovery를 포함한 DNS 공급자와 동기화합니다.

- 프로젝트 URL
- 프로젝트 설명서

기계 학습

Amazon EKS 및 Kubernetes 클러스터와 관련된 기계 학습 프로젝트입니다.

Kubeflow

Kubernetes용 기계 학습 도구 키트입니다.

- 프로젝트 URL
- 프로젝트 설명서
- AWS 오픈 소스 블로그: Kubeflow on Amazon EKS

Auto Scaling

Amazon EKS 및 Kubernetes 클러스터와 관련된 오토 스케일링 프로젝트입니다.

Cluster autoscaler

Cluster Autoscaler는 CPU 및 메모리 압력에 따라 Kubernetes 클러스터의 크기를 자동으로 조정하는 도구입니다.

- 프로젝트 URL
- 프로젝트 설명서
- Amazon EKS 워크숍: https://www.eksworkshop.com/

Escalator

Escalator는 Kubernetes용 배치 또는 작업에 최적화된 수평 자동 조정기입니다.

- 프로젝트 URL
- 프로젝트 설명서

모니터링

Amazon EKS 및 Kubernetes 클러스터와 관련된 모니터링 프로젝트입니다.

Prometheus

Prometheus는 오픈 소스 시스템 모니터링 및 알림 도구 키트입니다.

- 프로젝트 URL
- 프로젝트 설명서
- Amazon EKS 워크숍: https://eksworkshop.com/intermediate/240_monitoring/

연속 통합/연속 배포

Amazon EKS 및 Kubernetes 클러스터와 관련된 CI/CD 프로젝트입니다.

Jenkins X

Amazon EKS 및 Kubernetes 클러스터에 실행되는 현대적 클라우드 애플리케이션의 CI/CD 솔루션입니다.

- 프로젝트 URL
- 프로젝트 설명서

Amazon EKS 새로운 기능 및 로드맵

새로운 Amazon EKS 기능에 대해 알아보려면 AWS의 새로운 기능 페이지로 스크롤합니다. 그뿐만 아니라 GitHub에서 로드맵을 검토할 수도 있습니다. 향후 Amazon EKS를 사용할 방법을 계획할 수 있도록 예정된 기능 및 우선 순위에 대해 알 수 있습니다. 로드맵 우선 순위에 대해 직접 피드백을 제공할 수 있습니다.

Amazon EKS 문서 기록

다음 표에서 Amazon EKS 사용 설명서의 중요한 업데이트 및 새로운 기능이 나와 있습니다. 사용자로부터 받은 의견을 수렴하기 위해 설명서가 자주 업데이트됩니다.

변경 사항	설명	날짜
Windows에 대한 CloudWatch Container Insights 지원	이제 Amazon CloudWatch Observability Operator 추가 기능을 통해 클러스터의 Windows 워커 노드에서 Container Insights를 사용할 수 있습니다.	2024년 4월 10일
Kubernetes 개념	새 Kubernetes 개념 주제를 추가했습니다.	2024년 4월 5일
액세스 및 IAM 콘텐츠 재구성	인증 구성 맵, 액세스 항목, 포드 ID, IRSA 등 액세스 및 IAM 주제와 관련된 기존 페이지를 새 섹션으로 이동합니다. 개요 콘텐츠를 수정합니다.	2024년 4월 2일
Bottlerocket Amazon S3 CSI 드라이버에 대한 OS 지원	Mountpoint for Amazon S3 CSI 드라이버는 현재 Bottlerocket과 호환됩니다.	2024년 3월 13일
AWS 관리형 정책 업데이트 - 기존 정책에 대한 업데이트	Amazon EKS에서는 기존 AWS 관리형 정책을 업데이트했습니다.	2024년 3월 4일
Amazon Linux 2023	Amazon Linux 2023(AL2023)은 클라우드 애플리케이션에 안전하고 안정적이면서 고성능의 환경을 제공하도록 설계된 새로운 Linux 기반 운영 체제입니다.	2024년 2월 29일

Kubernetes1.29의 EKS Pod Identity와 IRSA 지원 사이드카	Kubernetes 1.29에서는 Amazon EKS 클러스터에서 사이드카 컨테이너를 사용할 수 있습니다. 서비스 계정 또는 EKS Pod Identity에 대한 IAM 역할에서는 사이드카 컨테이너가 지원됩니다. 사이드카에 대한 자세한 내용은 Kubernetes 설명서의 Sidecar Containers를 참조하세요.	2024년 2월 26일
Kubernetes 버전 1.29	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.29 지원이 추가되었습니다.	2024년 1월 23일
전체 릴리스: Kubernetes 버전에 대한 Amazon EKS 확장 지원	확장 Kubernetes 버전 지원을 통해 특정 Kubernetes 버전을 14개월 이상 유지할 수 있습니다.	2024년 1월 16일
AWS 클라우드에서 Amazon EKS 클러스터 상태 감지	Amazon EKS는 Amazon EKS 클러스터 및 클러스터 사전 조건 인프라에서 문제를 감지하여 클러스터 상태에 저장합니다. EKS API에서 클러스터의 `health` 및 AWS Management Console에서 EKS 클러스터 문제를 확인할 수 있습니다. 콘솔에서 감지되고 콘솔에 표시되는 문제 외에도 이 문제를 확인할 수 있습니다. 이전에는 AWS Outposts의 로컬 클러스터에서만 클러스터 상태를 사용할 수 있었습니다.	2023년 12월 28일

Amazon EKS AWS 리전 확장	이제 Amazon EKS를 캐나다 서부(캘거리)(ca-west-1) AWS 리전에서 사용할 수 있습니다.	2023년 12월 20일
클러스터 인사이트	이제 반복 검사를 기반으로 클러스터에 대한 권장 사항을 얻을 수 있습니다.	2023년 12월 20일
이제 액세스 항목을 사용하여 IAM 역할 및 사용자에게 클러스터에 대한 액세스 권한을 부여할 수 있습니다.	액세스 항목을 도입하기 전에는 aws-auth ConfigMap 에 항목을 추가하여 IAM 역할 및 사용자에게 클러스터에 대한 액세스 권한을 부여했습니다. 이제 각 클러스터에는 액세스 모드가 있으며, 일정에 따라 액세스 항목을 사용하도록 전환할 수 있습니다. 모드를 전환한 후 AWS CLI, AWS CloudFormation 및 AWS SDK에 액세스 항목을 추가하여 사용자를 추가할 수 있습니다.	2023년 12월 18일
Amazon EKS 플랫폼 버전 업데이트	이 업데이트는 보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전입니다. 여기에는 Kubernetes 1.28.4, 1.27.8, 1.26.11, 1.25.16의 새 패치 버전이 포함됩니다.	2023년 12월 12일
Mountpoint for Amazon S3 CSI 드라이버	이제 Amazon EKS 클러스터에 Mountpoint for Amazon S3 CSI 드라이버를 설치할 수 있습니다.	2023년 11월 27일

클러스터를 생성할 때 Prometheus 지표 켜기	이제 AWS Management Console에서 클러스터를 생성할 때 Prometheus 지표를 켤 수 있습니다. 또한 관찰성 탭에서 Prometheus 스크레이퍼 세부 정보를 볼 수 있습니다.	2023년 11월 26일
Amazon EKS Pod Identity	Amazon EKS Pod Identity는 IAM 역할을 Kubernetes 서비스 계정에 연결합니다. 이 기능을 사용하면 노드 IAM 역할에 확장된 권한을 더 이상 제공할 필요가 없습니다. 이렇게 하면 해당 노드의 Pods가 AWS API를 호출할 수 있습니다. 서비스 계정에 대한 IAM 역할과 달리 EKS Pod Identity는 완전히 EKS 내에 있으므로 OIDC 자격 증명 공급자가 필요하지 않습니다.	2023년 11월 26일
AWS 관리형 정책 업데이트 - 기존 정책에 대한 업데이트	Amazon EKS에서는 기존 AWS 관리형 정책을 업데이트했습니다.	2023년 11월 26일
CSI 스냅샷 컨트롤러	이제 Amazon EBS CSI 드라이버와 같은 호환 CSI 드라이버에 사용할 CSI 스냅샷 컨트롤러를 설치할 수 있습니다.	2023년 11월 17일

ADOT Operator 주제 재작성	ADOT Operator에 대한 Amazon EKS 추가 기능 지원 섹션이 AWS Distro for OpenTelemetry 설명서와 중복되었습니다. 오래되고 일관적이지 않은 정보를 줄이기 위해 나머지 필수 정보를 해당 리소스로 마이그레이션했습니다.	2023년 11월 14일
Prometheus 지표에 대한 CoreDNS EKS 추가 기능 지원	CoreDNS에 대한 EKS 추가 기능의 `v1.10.1-eksbuild.5` , `v1.9.3-eksbuild.9` , `v1.8.7-eksbuild.8` 버전은 `kube-dns` 서비스에 CoreDNS에서 지표를 게시하는 포트가 노출됩니다. 이를 통해 모니터링 시스템에서 CoreDNS 지표를 더욱 쉽게 포함할 수 있습니다.	2023년 11월 10일
Amazon EKS CloudWatch Observability Operator 추가 기능	Amazon EKS CloudWatch Observability Operator 페이지가 추가되었습니다.	2023년 11월 6일
미국 동부(오하이오)의 자체 관리형 P5 인스턴스용 용량 블록	미국 동부(오하이오)에서는 이제 자체 관리형 P5 인스턴스에 용량 블록을 사용할 수 있습니다.	2023년 10월 31일

클러스터는 서브넷 및 보안 그룹을 수정하도록 지원합니다	클러스터를 업데이트하여 클러스터에서 사용하는 서브넷과 보안 그룹을 변경할 수 있습니다. AWS Management Console, AWS CLI의 최신 버전, AWS CloudFormation 및 `eksctl` 버전 `v0.164.0-rc.0` 이상에서 업데이트할 수 있습니다. 서브넷에 사용 가능한 IP 주소를 더 많이 제공하여 클러스터 버전을 성공적으로 업그레이드하려면 이 작업을 수행해야 할 수도 있습니다.	2023년 10월 24일
클러스터 역할 및 관리형 노드 그룹 역할은 고객 관리형 AWS Identity and Access Management 정책을 지원합니다	클러스터 역할에서 `AmazonEKSClusterPolicy` AWS 관리형 정책 대신 사용자 지정 IAM 정책을 사용할 수 있습니다. 또한 `AmazonEKSWorkerNodePolicy` AWS 관리형 정책 대신 관리형 노드 그룹의 노드 역할에 사용자 지정 IAM 정책을 사용할 수 있습니다. 엄격한 규정 준수 요구 사항을 충족할 수 있는 최소 권한이 있는 정책을 생성하려면 이렇게 하세요.	2023년 10월 23일
eksctl 설치 링크 수정	페이지가 이동된 후 eksctl의 설치 링크를 수정합니다.	2023년 10월 6일
평가판 릴리스: Kubernetes 버전에 대한 Amazon EKS 확장 지원	확장 Kubernetes 버전 지원을 통해 특정 Kubernetes 버전을 14개월 이상 유지할 수 있습니다.	2023년 10월 4일

AWS App Mesh 통합에 대한 참조 제거	AWS App Mesh과의 Amazon EKS 통합은 App Mesh를 사용하는 기존 고객만 이용할 수 있습니다.	2023년 9월 29일
Kubernetes 버전 1.28	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.28 지원이 추가되었습니다.	2023년 9월 26일
기존 클러스터는 Amazon VPC CNI plugin for Kubernetes에서 Kubernetes 네트워크 정책 시행을 지원합니다	타사 솔루션을 필요로 하는 대신 Amazon VPC CNI plugin for Kubernetes를 사용하는 기존 클러스터에서 Kubernetes 네트워크 정책을 사용할 수 있습니다.	2023년 9월 15일
CoreDNS Amazon EKS 추가 기능은 PDB 수정을 지원합니다	버전 v1.9.3-eksbuild.7 이상 및 v1.10.1-eksbuild.4 이상에서 CoreDNS에 대한 EKS 추가 기능의 PodDisruptionBudget 을 수정할 수 있습니다.	2023년 9월 15일
공유 서브넷에 대한 Amazon EKS 지원	공유 서브넷에서 Amazon EKS 클러스터를 만드는 것에 대한 새로운 공유 서브넷 요구 사항 및 고려 사항.	2023년 9월 7일
Amazon EKS란 무엇입니까? 업데이트	새로운 일반 사용 사례 및 아키텍쳐 주제 추가. 기타 주제 갱신.	2023년 9월 6일
Amazon VPC CNI plugin for Kubernetes의 Kubernetes 네트워크 정책 시행	타사 솔루션을 필요로 하는 대신 Amazon VPC CNI plugin for Kubernetes에 Kubernetes 네트워크 정책을 사용할 수 있습니다.	2023년 8월 29일

Amazon EKS AWS 리전 확장	Amazon EKS는 이제 이스라엘(텔아비브)(`il-central-1`) AWS 리전에서 사용할 수 있습니다.	2023년 8월 1일
구성 가능한 Fargate용 임시 스토리지	Amazon EKS Fargate에서 각 Pod 실행용 임시 스토리지 총량을 늘릴 수 있습니다.	2023년 7월 31일
Amazon EFS CSI 드라이버에 대한 추가 기능 지원	이제 AWS Management Console, AWS CLI 및 API를 사용하여 Amazon EFS CSI 드라이버를 관리할 수 있습니다.	2023년 7월 26일
AWS 관리형 정책 업데이트 - 새 정책	Amazon EKS에서는 새 AWS 관리형 정책을 추가했습니다.	2023년 7월 26일
이제 AWS Outposts의 로컬 클러스터에서 1.27, 1.26, 1.25 및 1.24용 Kubernetes 버전 업데이트를 사용할 수 있습니다	이제 AWS Outposts의 로컬 클러스터에서 1.27.3, 1.26.6, 1.25.11 및 1.24.15용 Kubernetes 버전 업데이트를 사용할 수 있습니다	2023년 7월 20일
Windows 노드에 대한 IP 접두사 지원	노드에 IP 접두사를 할당하면 노드에 개별 보조 IP 주소를 할당할 때보다 훨씬 더 많은 수의 Pods를 노드에서 호스팅할 수 있습니다.	2023년 7월 6일
Amazon FSx for OpenZFS CSI 드라이버	이제 Amazon EKS 클러스터에 Amazon FSx for OpenZFS CSI 드라이버를 설치할 수 있습니다.	2023년 6월 30일
`IPv4` 클러스터의 Linux 노드에 있는 Pods가 이제 `IPv6` 엔드포인트와 통신 가능	노드에 IPv6 주소를 할당한 후 Pods의 `IPv4` 주소는 실행 중인 노드의 `IPv6` 주소로 변환된 네트워크 주소입니다.	2023년 6월 19일

AWS GovCloud (US) Regions의 Windows 관리형 노드 그룹	AWS GovCloud (US) Regions에서 Amazon EKS 관리형 노드 그룹이 이제 Windows 컨테이너를 실행할 수 있습니다.	2023년 5월 30일
Kubernetes 버전 1.27	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.27 지원이 추가되었습니다.	2023년 5월 24일
Kubernetes 버전 1.26	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.26 지원이 추가되었습니다.	2023년 4월 11일
도메인 없는 gMSA	이제 Windows Pods에서 도메인 없는 gMSA를 사용할 수 있습니다.	2023년 3월 27일
Amazon EKS AWS 리전 확장	이제 아시아 태평양(서울)(ap-southeast-4) AWS 리전에서 Amazon EKS를 사용할 수 있습니다.	2023년 3월 10일
Amazon File Cache CSI 드라이버	이제 Amazon EKS 클러스터에 Amazon 파일 캐시 CSI 드라이버를 설치할 수 있습니다.	2023년 3월 3일
이제 Kubernetes 버전 1.25를 AWS Outposts의 로컬 클러스터에 사용할 수 있습니다.	이제 Outpost에서 Kubernetes 버전 1.22 – 1.25을(를) 사용하여 Amazon EKS 로컬 클러스터를 생성할 수 있습니다.	2023년 3월 1일
Kubernetes 버전 1.25	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.25 지원이 추가되었습니다.	2023년 2월 22일
AWS 관리형 정책 업데이트 - 기존 정책에 대한 업데이트	Amazon EKS에서는 기존 AWS 관리형 정책을 업데이트했습니다.	2023년 2월 7일

Amazon EKS AWS 리전 확장	Amazon EKS는 이제 아시아 태평양(하이데라바드)(ap-south-2), 유럽(취리히)(eu-central-2) 및 유럽(스페인)(eu-south-2) AWS 리전에서 사용할 수 있습니다.	2023년 2월 6일
Kubernetes 버전 1.21 – 1.24은(는) 이제 AWS Outposts의 로컬 클러스터에 사용할 수 있습니다.	이제 Outpost에서 Kubernetes 버전 1.21 – 1.24을(를) 사용하여 Amazon EKS 로컬 클러스터를 생성할 수 있습니다. 이전에는 1.21 버전만 사용할 수 있었습니다.	2023년 1월 17일
Amazon EKS에서 현재 AWS PrivateLink 지원	AWS PrivateLink를 사용하여 VPC와 Amazon EKS 사이에 프라이빗 연결을 생성할 수 있습니다.	2022년 12월 16일
관리형 노드 그룹 Windows 지원	이제 Amazon EKS 관리형 노드 그룹에 Windows를 사용할 수 있습니다.	2022년 12월 15일
독립 소프트웨어 공급업체의 Amazon EKS 추가 기능을 이제 AWS Marketplace에서 사용할 수 있습니다.	이제 AWS Marketplace를 통해 독립 소프트웨어 공급업체의 Amazon EKS 추가 기능을 검색하고 구독할 수 있습니다.	2022년 11월 28일
AWS 관리형 정책 업데이트 - 기존 정책에 대한 업데이트	Amazon EKS에서는 기존 AWS 관리형 정책을 업데이트했습니다.	2022년 11월 17일
Kubernetes 버전 1.24	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.24 지원이 추가되었습니다.	2022년 11월 15일

Amazon EKS AWS 리전 확장	이제 중동(UAE)(me-central-1) AWS 리전에서 Amazon EKS를 사용할 수 있습니다.	2022년 11월 3일
AWS 관리형 정책 업데이트 - 기존 정책에 대한 업데이트	Amazon EKS에서는 기존 AWS 관리형 정책을 업데이트했습니다.	2022년 10월 24일
AWS 관리형 정책 업데이트 - 기존 정책에 대한 업데이트	Amazon EKS에서는 기존 AWS 관리형 정책을 업데이트했습니다.	2022년 10월 20일
현재 AWS Outposts의 로컬 클러스터 사용 가능	이제 Outpost에서 Amazon EKS 로컬 클러스터를 생성할 수 있습니다.	2022년 9월 19일
Fargate vCPU 기반 할당량	Fargate는 Pod 기반 할당량에서 vCPU 기반 할당량으로 전환하고 있습니다.	2022년 9월 8일
AWS 관리형 정책 업데이트 - 기존 정책에 대한 업데이트	Amazon EKS에서는 기존 AWS 관리형 정책을 업데이트했습니다.	2022년 8월 31일
비용 모니터링	Amazon EKS는 이제 Pods, 노드, 네임스페이스 및 레이블을 포함한 Kubernetes 리소스별로 분류된 비용을 모니터링하는 데 사용할 수 있는 Kubecost를 지원합니다.	2022년 8월 24일
AWS 관리형 정책 업데이트 - 새 정책	Amazon EKS에서는 새 AWS 관리형 정책을 추가했습니다.	2022년 8월 24일
AWS 관리형 정책 업데이트 - 새 정책	Amazon EKS에서는 새 AWS 관리형 정책을 추가했습니다.	2022년 8월 23일

리소스에 결제용 태그 지정	모든 클러스터에 대한 `aws:eks:cluster-name` 생성 비용 할당 태그 지원이 추가되었습니다.	2022년 8월 16일
Fargate 프로파일 와일드카드	네임스페이스, 레이블 키 및 레이블 값에 대한 선택기 기준에서 Fargate 프로파일 와일드카드에 대한 지원이 추가되었습니다.	2022년 8월 16일
Kubernetes 버전 `1.23`	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 `1.23` 지원이 추가되었습니다.	2022년 8월 11일
AWS Management Console에서 Kubernetes 리소스 보기	이제 AWS Management Console을 사용하여 클러스터에 배포된 Kubernetes 리소스에 대한 정보를 볼 수 있습니다.	2022년 5월 3일
Amazon EKS AWS 리전 확장	이제 아시아 태평양(자카르타) (`ap-southeast-3`) AWS 리전에서 Amazon EKS를 사용할 수 있습니다.	2022년 5월 2일
Observability 페이지 및 ADOT 추가 기능 지원	관찰 가능성 페이지 및 AWS Distro for OpenTelemetry(ADOT)가 추가되었습니다.	2022년 4월 21일
Kubernetes 버전 `1.22`	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 `1.22` 지원이 추가되었습니다.	2022년 4월 4일
AWS 관리형 정책 업데이트 - 새 정책	Amazon EKS에서는 새 AWS 관리형 정책을 추가했습니다.	2022년 4월 4일

Fargate Pod 패치 세부 정보 추가	Fargate Pods를 업그레이드하면 Amazon EKS는 먼저 Pod 중단 예산에 따라 Pods를 제거하려고 시도합니다. 실패한 제거에 대응하는 이벤트 규칙을 생성한 후에 Pods를 삭제할 수 있습니다.	2022년 4월 1일
정식 릴리스: Amazon EBS CSI 드라이버에 대한 추가 기능 지원	이제 AWS Management Console, AWS CLI 및 API를 사용하여 Amazon EBS CSI 드라이버를 관리할 수 있습니다.	2022년 3월 31일
AWS Outposts 내용 업데이트	AWS Outposts에서 Amazon EKS 클러스터를 배포하는 방법.	2022년 3월 22일
AWS 관리형 정책 업데이트 - 기존 정책에 대한 업데이트	Amazon EKS에서는 기존 AWS 관리형 정책을 업데이트했습니다.	2022년 3월 21일
Windows `containerd` 지원	이제 Windows 노드용 `containerd` 런타임을 선택할 수 있습니다.	2022년 3월 14일
Amazon EKS Connector 고려 사항이 보안 문서에 추가	연결된 클러스터와 관련된 공동 책임 모델을 설명합니다.	2022년 2월 25일
Pods 및 서비스에 IPv6 주소 할당	이제 Pods와 서비스에 IPv6 주소를 할당하는 `1.21` 이상 클러스터를 생성할 수 있습니다.	2022년 1월 6일
AWS 관리형 정책 업데이트 - 기존 정책에 대한 업데이트	Amazon EKS에서는 기존 AWS 관리형 정책을 업데이트했습니다.	2021년 12월 13일

미리 보기 릴리스: Amazon EBS CSI 드라이버에 대한 추가 기능 지원	이제 AWS Management Console, AWS CLI 및 API를 통해 미리 보기하여 Amazon EBS CSI 드라이버를 관리할 수 있습니다.	2021년 12월 9일
Karpenter Autoscaler 지원	이제 Karpenter 오픈 소스 프로젝트를 사용하여 노드를 자동 확장할 수 있습니다.	2021년 11월 29일
Fargate 로깅에서 Fluent Bit Kubernetes 필터 지원	이제 Fargate 로깅과 함께 Fluent Bit Kubernetes 필터를 사용할 수 있습니다.	2021년 11월 10일
컨트롤 플레인에서 Windows 지원 사용 가능	이제 컨트롤 플레인에서 Windows 지원을 사용할 수 있습니다. 더 이상 데이터 영역에서 Windows 지원을 사용 설정할 필요가 없습니다.	2021년 11월 9일
관리형 노드 그룹에 대한 AMI 유형으로 Bottlerocket 추가	이전에는 Bottlerocket을 자체 관리형 노드 옵션으로만 사용할 수 있었습니다. 이제 관리형 노드 그룹으로 구성할 수 있어 노드 규정 준수 요구 사항을 충족하는 데 필요한 노력을 줄일 수 있습니다.	2021년 10월 28일
DL1 드라이버 지원	사용자 정의 Amazon Linux AMI는 이제 Amazon Linux 2용 딥 러닝 워크로드를 지원합니다. 이를 통해 일반 온프레미스 또는 클라우드 기준 구성이 허용됩니다.	2021년 10월 25일

VT1 비디오 지원	사용자 지정 Amazon Linux AMI는 이제 일부 배포에 대해 VT1을 지원합니다. 이 활성화를 통해 Amazon EKS 클러스터에 Xilinx U30 디바이스를 알립니다.	2021년 9월 13일
이제 Amazon EKS 커넥터를 사용할 수 있습니다.	Amazon EKS Connector를 사용하여 적합한 Kubernetes 클러스터를 AWS에 등록하고 연결할 수 있으며 Amazon EKS 콘솔에서 시각화할 수 있습니다.	2021년 9월 8일
이제 Amazon EKS Anywhere를 사용할 수 있습니다.	Amazon EKS Anywhere는 온프레미스에서 Kubernetes 클러스터를 생성하고 운영하기 위해 사용할 수 있는 Amazon EKS에 대한 새로운 배포 옵션입니다.	2021년 9월 8일
Amazon FSx for NetApp ONTAP CSI 드라이버	Amazon FSx for NetApp ONTAP CSI 드라이버에 대해 간략히 설명하고 다른 참조에 대한 링크를 제공하는 주제가 추가되었습니다.	2021년 9월 2일
이제 관리형 노드 그룹은 노드에 대한 Amazon EKS 권장 최대 Pods 수를 자동으로 계산합니다.	이제 관리형 노드 그룹은 시작 템플릿 없이 배포하거나 AMI ID를 지정하지 않은 시작 템플릿으로 배포하는 노드에 대한 Amazon EKS 최대 Pods 수를 자동으로 계산합니다.	2021년 8월 30일

Amazon EKS 추가 기능 소프트웨어를 제거하지 않고 추가 기능 설정의 Amazon EKS 관리 제거	이제 클러스터에서 추가 기능 소프트웨어를 제거하지 않고도 Amazon EKS 추가 기능을 제거할 수 있습니다.	2021년 8월 20일
Multus를 사용하여 멀티홈 Pods 생성	이제 Multus를 사용하여 여러 개의 네트워크 인터페이스를 Pod에 추가할 수 있습니다.	2021년 8월 2일
Linux Amazon EC2 노드에 더 많은 IP 주소 추가	Linux Amazon EC2 노드에 훨씬 더 많은 IP 주소를 추가할 수 있습니다. 즉, 각 노드에서 더 높은 구성으로 Pods를 실행할 수 있습니다.	2021년 7월 27일
`containerd` 런타임 부트스트랩	Amazon EKS 최적화 가속 Amazon Linux Amazon Machine Image(AMI)에는 이제 부트스트랩 플래그가 포함되어 있어 `containerd` 런타임을 Amazon EKS 최적화 및 Bottlerocket AMI에서 사용 설정할 수 있습니다. 이 플래그는 지원되는 모든 Kubernetes 버전의 AMI에서 사용할 수 있습니다.	2021년 7월 19일
Kubernetes 버전 `1.21`	Kubernetes 버전 `1.21` 지원이 추가 되었습니다.	2021년 7월 19일
관리형 정책 주제 추가	2021년 6월 17일 이후 적용된 모든 Amazon EKS IAM 관리형 정책 및 변경 사항의 목록입니다.	2021년 6월 17일

Fargate에서 Pods에 보안 그룹 사용	이제 Fargate에서 Pods에 보안 그룹을 사용할 수 있을 뿐 아니라 Amazon EC2 노드와 함께 사용할 수도 있습니다.	2021년 6월 1일
CoreDNS 및 kube-proxy Amazon EKS 추가 기능 추가	Amazon EKS는 이제 CoreDNS 및 클러스터용 kube-proxy Amazon EKS 추가 기능을 제공합니다.	2021년 5월 19일
Kubernetes 버전 1.20	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.20 지원이 추가되었습니다.	2021년 5월 18일
AWS Load Balancer Controller r2.2.0 릴리스	이제 AWS Load Balancer Controller를 사용하여 인스턴스 또는 IP 대상을 사용하여 Elastic 로드 밸런서를 생성할 수 있습니다.	2021년 5월 14일
관리형 노드 그룹의 노드 테인트	Amazon EKS는 이제 관리 노드 그룹에 노트 테인트를 추가할 수 있도록 지원합니다.	2021년 5월 11일
기존 클러스터에 대한 비밀 암호화	Amazon EKS는 이제 기존 클러스터에 비밀 암호화를 추가할 수 있도록 지원합니다.	2021년 2월 26일
Kubernetes 버전 1.19	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.19 지원이 추가되었습니다.	2021년 2월 16일
Amazon EKS는 이제 버전 1.16 이상의 클러스터에서 사용자를 인증하는 방법으로 OpenID Connect(OIDC) ID 공급자를 지원합니다.	OIDC 자격 증명 공급자는 AWS Identity and Access Management(IAM)와 함께 사용하거나 그 대안으로 사용할 수 있습니다.	2021년 2월 12일

AWS Management Console에서 노드 및 워크로드 리소스 보기	이제 관리형, 자체 관리형 및 Fargate 노드, 그리고 AWS Management Console에 배포된 Kubernetes 워크로드에 대한 세부 정보를 볼 수 있습니다.	2020년 12월 1일
관리형 노드 그룹에 스팟 인스턴스 유형 배포	이제 관리형 노드 그룹에 여러 스팟 또는 온디맨드 인스턴스 유형을 배포할 수 있습니다.	2020년 12월 1일
이제 Amazon EKS가 클러스터의 특정 추가 기능을 관리할 수 있습니다.	추가 기능을 직접 관리하거나 Amazon EKS가 Amazon EKS API를 통해 추가 기능의 시작 및 버전을 제어하도록 할 수 있습니다.	2020년 12월 1일
여러 수신 간에 ALB 공유	이제 여러 Kubernetes 수신 간에 AWS Application Load Balancer(ALB)를 공유할 수 있습니다. 과거에는 각 수신마다 별도의 ALB를 배포해야 했습니다.	2020년 10월 23일
NLB IP 대상 지원	이제 IP 대상을 사용하여 Network Load Balancer를 배포할 수 있습니다. 즉, NLB를 사용하여 네트워크 트래픽을 Fargate Pods로, Amazon EC2 노드에서 실행되는 Pods로 직접 로드 밸런싱할 수 있습니다.	2020년 10월 23일
Kubernetes 버전 1.18	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.18 지원이 추가되었습니다.	2020년 10월 13일

Kubernetes 서비스 IP 주소 할당에 대한 사용자 지정 CIDR 블록을 지정합니다.	이제 Kubernetes가 서비스 IP 주소를 할당하는 사용자 지정 CIDR 블록을 지정할 수 있습니다.	2020년 9월 29일
개별 Pods에 보안 그룹 할당	이제 여러 Amazon EC2 인스턴스 유형에서 실행되는 일부 개별 Pods에 서로 다른 보안 그룹을 연결할 수 있습니다.	2020년 9월 9일
노드에서 Bottlerocket 배포	이제 Bottlerocket을 실행하는 노드를 배포할 수 있습니다.	2020년 8월 31일
Arm 노드를 시작하는 기능 정식 지원	이제 관리형 노드 그룹과 자체 관리형 노드 그룹에서 Arm 노드를 시작할 수 있습니다.	2020년 8월 17일
관리 노드 그룹 시작 템플릿 및 사용자 지정 AMI	이제 Amazon EC2 시작 템플릿을 사용하는 관리형 노드 그룹을 배포할 수 있습니다. 원하는 경우 시작 템플릿에서 사용자 지정 AMI 지정할 수 있습니다.	2020년 8월 17일
AWS Fargate에 대한 EFS 지원	이제 AWS Fargate에서 Amazon EFS를 사용할 수 있습니다.	2020년 8월 17일
Amazon EKS 플랫폼 버전 업데이트	이 업데이트는 보안 수정 및 개선 사항이 적용된 새로운 플랫폼 버전입니다. 여기에는 Kubernetes 버전 `1.15` 이상으로 Network Load Balancer를 사용할 때 `LoadBalancer` 유형의 서비스에 대한 UDP 지원이 포함됩니다. 자세한 내용은 GitHub에서 AWS Network Load Balancer에 대해 UDP 허용 문제를 참조하세요.	2020년 8월 12일

Amazon EKS AWS 리전 확장	Amazon EKS는 이제 아프리카(케이프타운)(af-south-1) 및 EU(밀라노)(eu-south-1) AWS 리전에서 사용할 수 있습니다.	2020년 8월 6일
Fargate 사용량 지표	AWS Fargate는 Fargate 온디맨드 리소스의 계정 사용에 대한 가시성을 제공하는 CloudWatch 사용량 지표를 제공합니다.	2020년 8월 3일
Kubernetes 버전 1.17	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.17 지원이 추가되었습니다.	2020년 7월 10일
Kubernetes용 App Mesh 컨트롤러를 사용하여 Kubernetes 내에서 App Mesh 리소스 생성 및 관리	Kubernetes 내에서 App Mesh 리소스를 생성하고 관리할 수 있습니다. 또한 컨트롤러는 사용자가 배포한 Pods에 Envoy 프록시 및 init 컨테이너를 자동으로 삽입합니다.	2020년 6월 18일
Amazon EKS에서 Amazon EC2 Inf1 노드 지원	Amazon EC2 Inf1 노드를 클러스터에 추가할 수 있습니다.	2020년 6월 4일
Amazon EKS AWS 리전 확장	이제 AWS GovCloud(미국 동부)(us-gov-east-1) 및 AWS GovCloud(미국 서부)(us-gov-west-1) AWS 리전에서 Amazon EKS를 사용할 수 있습니다.	2020년 5월 13일

Amazon EKS에서 Kubernetes1.12가 더 이상 지원되지 않음	Amazon EKS에서 Kubernetes 버전 1.12가 더 이상 지원되지 않습니다. 서비스 중단이 발생하지 않도록 모든 1.12 클러스터를 버전 1.13 이상으로 업데이트하세요.	2020년 5월 12일
Kubernetes 버전 1.16	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.16 지원이 추가되었습니다.	2020년 4월 30일
AWSServiceRoleForAmazonEKS 서비스 연결 역할 추가	AWSServiceRoleForAmazonEKS 서비스 연결 역할이 추가되었습니다.	2020년 4월 16일
Kubernetes 버전 1.15	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.15 지원이 추가되었습니다.	2020년 3월 10일
Amazon EKS AWS 리전 확장	이제 베이징(cn-north-1) 및 닝샤(cn-northwest-1) AWS 리전에서 Amazon EKS를 사용할 수 있습니다.	2020년 2월 26일
FSx for Lustre CSI 드라이버	Kubernetes 1.14 Amazon EKS 클러스터에 FSX for Lustre CSI 드라이버를 설치하기 위한 항목이 추가되었습니다.	2019년 12월 23일
클러스터의 퍼블릭 액세스 엔드포인트에 대한 네트워크 액세스 제한	이 업데이트를 통해 Amazon EKS를 사용하여 Kubernetes API 서버의 퍼블릭 액세스 엔드포인트와 통신할 수 있는 CIDR 범위를 제한할 수 있습니다.	2019년 12월 20일

VPC 외부에서 클러스터의 프라이빗 액세스 엔드포인트 주소 확인	이 업데이트를 통해 Amazon EKS를 사용하여 VPC 외부에서 Kubernetes API 서버의 프라이빗 액세스 엔드포인트를 확인할 수 있습니다.	2019년 12월 13일
(베타) Amazon EC2 A1 Amazon EC2 인스턴스 노드	Amazon EKS 클러스터에 등록하는 Amazon EC2 A1 Amazon EC2 인스턴스 노드를 시작합니다.	2019년 12월 4일
AWS Outpost에 클러스터 생성	이제 Amazon EKS가 AWS Outposts에 클러스터를 생성하도록 지원합니다.	2019년 12월 3일
Amazon EKS의 AWS Fargate	Amazon EKS Kubernetes 클러스터는 이제 Fargate에서 실행 중인 Pods를 지원합니다.	2019년 12월 3일
Amazon EKS AWS 리전 확장	Amazon EKS는 이제 캐나다(중부)(`ca-central-1`) AWS 리전에서 사용할 수 있습니다.	2019년 11월 21일
관리형 노드 그룹	Amazon EKS 관리형 노드 그룹은 Amazon EKS Kubernetes 클러스터의 노드(Amazon EC2 인스턴스) 프로비저닝 및 수명 주기 관리를 자동화합니다.	2019년 11월 18일
Amazon EKS 플랫폼 버전 업데이트	`CVE-2019-11253` 을 해결하기 위한 새로운 플랫폼 버전	2019년 11월 6일
Amazon EKS에서 Kubernetes1.11가 더 이상 지원되지 않음	Amazon EKS에서 Kubernetes 버전 `1.11`이 더 이상 지원되지 않습니다. 서비스 중단이 발생하지 않도록 모든 `1.11` 클러스터를 버전 `1.12` 이상으로 업데이트하세요.	2019년 11월 4일

Amazon EKS AWS 리전 확장	이제 남아메리카(상파울루) (`sa-east-1`) AWS 리전에서 Amazon EKS를 사용할 수 있습니다.	2019년 10월 16일
Windows 지원	이제 Kubernetes 버전 `1.14`를 실행하는 Amazon EKS 클러스터에서 Windows 워크로드를 지원합니다.	2019년 10월 7일
AutoScaling	Amazon EKS 클러스터에서 지원되는 다양한 유형의 Kubernetes 자동 크기 조정 중 일부에 관해 설명하는 챕터를 추가했습니다.	2019년 9월 30일
Kubernetes 대시보드 업데이트	베타 `2.0` 버전을 사용하기 위해 Amazon EKS 클러스터에 Kubernetes 대시보드를 설치하는 방법에 관한 주제를 업데이트했습니다.	2019년 9월 28일
Amazon EFS CSI 드라이버	Kubernetes `1.14` Amazon EKS 클러스터에 Amazon EFS CSI 드라이버를 설치하는 방법에 관한 주제를 추가했습니다.	2019년 9월 19일
Amazon EKS 최적화 AMI ID에 대한 Amazon EC2 Systems Manager 파라미터	Amazon EC2 Systems Manager 파라미터를 사용해 Amazon EKS 최적화 AMI ID를 검색하는 방법에 관한 주제를 추가했습니다. 이 파라미터를 사용하면 AMI ID를 조회할 필요가 없습니다.	2019년 9월 18일
Amazon EKS 리소스 태깅	Amazon EKS 클러스터의 태깅을 관리할 수 있습니다.	2019년 9월 16일

Amazon EBS CSI 드라이버	Kubernetes 1.14 Amazon EKS 클러스터에 Amazon EBS CSI 드라이버를 설치하는 방법에 관한 주제를 추가했습니다.	2019년 9월 9일
CVE-2019-9512 및 CVE-2019-9514 에 패치된 새 Amazon EKS 최적화 AMI	Amazon EKS는 CVE-2019-9512 및 CVE-2019-9514 를 해결하기 위해 Amazon EKS 최적화 AMI를 업데이트했습니다.	2019년 9월 6일
Amazon EKS에서 Kubernetes1.11 사용 중단 발표	Amazon EKS는 2019년 11월 4일에 Kubernetes 버전 1.111 에 대한 지원을 중단했습니다.	2019년 9월 4일
Kubernetes 버전 1.14	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.14 지원이 추가되었습니다.	2019년 9월 3일
서비스 계정에 대한 IAM 역할	Amazon EKS 클러스터의 서비스 계정에 대한 IAM 역할과 함께 IAM 역할을 Kubernetes 서비스 계정에 연결할 수 있습니다. 이 기능을 사용하면 노드 IAM 역할에 확장된 권한을 더 이상 제공할 필요가 없습니다. 이렇게 하면 해당 노드의 Pods가 AWS API를 호출할 수 있습니다.	2019년 9월 3일
Amazon EKS AWS 리전 확장	이제 중동(바레인)(me-south-1) AWS 리전에서 Amazon EKS를 사용할 수 있습니다.	2019년 8월 29일
Amazon EKS 플랫폼 버전 업데이트	CVE-2019-9512 및 CVE-2019-9514 를 해결하기 위한 새로운 플랫폼 버전	2019년 8월 28일

Amazon EKS 플랫폼 버전 업데이트	CVE-2019-11247 및 CVE-2019-11249 를 해결하기 위한 새로운 플랫폼 버전	2019년 8월 5일
Amazon EKS 지역 확장	이제 아시아 태평양(홍콩)(ap-east-1) AWS 리전에서 Amazon EKS를 사용할 수 있습니다.	2019년 7월 31일
Amazon EKS에서 Kubernetes1.10이 더 이상 지원되지 않음	Amazon EKS에서 Kubernetes 버전 1.10이 더 이상 지원되지 않습니다. 서비스 중단이 발생하지 않도록 모든 1.10 클러스터를 버전 1.11 이상으로 업데이트합니다.	2019년 7월 30일
ALB 인그레스 컨트롤러에 대한 주제 추가	Kubernetes에 대한 AWS ALB 수신 컨트롤러는 인그레스 리소스가 생성될 때 ALB가 생성되도록 하는 컨트롤러입니다.	2019년 7월 11일
새로운 Amazon EKS 최적화 AMI	AMI에서 불필요한 kubectl 바이너리 제거	2019년 7월 3일
Kubernetes 버전 1.13	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 1.13 지원이 추가되었습니다.	2019년 6월 18일
AWS-2019-005 에 패치된 새 Amazon EKS 최적화 AMI	Amazon EKS는 AWS-2019-005 에서 설명하는 취약성을 해결하기 위해 Amazon EKS 최적화 AMI를 업데이트했습니다.	2019년 6월 17일
Amazon EKS에서 Kubernetes1.10 지원 중단 발표	Amazon EKS는 2019년 7월 22일에 Kubernetes 버전 1.10 지원을 중단했습니다.	2019년 5월 21일

Amazon EKS 플랫폼 버전 업데이트	`kubelet` 인증서에서 사용자 지정 DNS 이름을 지원하고 `etcd` 성능을 개선하기 위한 새로운 Kubernetes `1.11` 및 `1.10` 클러스터용 플랫폼 버전입니다.	2019년 5월 21일
AWS CLI `get-token` 명령	`aws eks get-token` 명령이 AWS CLI에 추가되었습니다. 클러스터 API 서버 통신에 사용할 클라이언트 보안 토큰을 생성하기 위해 AWS IAM Authenticator for Kubernetes를 더 이상 설치할 필요가 없습니다. 이 새 기능을 사용하려면 AWS CLI 설치를 최신 버전으로 업그레이드하세요. 자세한 내용을 알아보려면 AWS Command Line Interface 사용자 가이드에서 AWS Command Line Interface 설치를 참조하세요.	2019년 5월 10일
`eksctl` 시작하기	이 시작 가이드에서는 `eksctl`을 사용하여 Amazon EKS를 시작하는 데 필요한 모든 리소스를 설치할 수 있는 방법에 대해 설명합니다. 이 유틸리티는 Amazon EKS에서 Kubernetes 클러스터를 생성하고 관리하기 위한 간단한 명령줄 유틸리티입니다.	2019년 5월 10일

Amazon EKS 플랫폼 버전 업데이트	`kubelet` 인증서에서 사용자 지정 DNS 이름을 지원하고 `etcd` 성능을 개선하기 위한 새로운 Kubernetes `1.12` 클러스터용 플랫폼 버전입니다. 이 버전에서 노드 `kubelet` 데몬이 몇 초마다 새로운 인증서를 요청하는 버그가 수정되었습니다.	2019년 5월 8일
Prometheus 튜토리얼	Amazon EKS 클러스터에 Prometheus를 배포하는 방법에 대한 주제가 추가되었습니다.	2019년 4월 5일
Amazon EKS 컨트롤 영역 로깅	이 업데이트를 통해 Amazon EKS 제어판에서 감사 및 진단 로그를 직접 가져올 수 있습니다. 해당 계정의 이러한 CloudWatch 로그를 클러스터 보안 및 실행에 대한 참조로 사용할 수 있습니다.	2019년 4월 4일
Kubernetes 버전 `1.12`	새 클러스터 및 버전 업그레이드를 위한 Kubernetes 버전 `1.12` 지원이 추가되었습니다.	2019년 3월 28일
App Mesh 시작 가이드 추가	App Mesh 및 Kubernetes를 시작하기 위한 문서가 추가되었습니다.	2019년 3월 27일
Amazon EKS API 서버 엔드포인트 프라이빗 액세스	Amazon EKS 클러스터의 Kubernetes API 서버 엔드포인트에 대한 퍼블릭 액세스를 비활성화하기 위한 문서가 추가되었습니다.	2019년 3월 19일

Kubernetes 지표 서버를 설치하기 위한 주제 추가	Kubernetes 지표 서버는 클러스터에서 리소스 사용량 데이터의 집계입니다.	2019년 3월 18일
관련 오픈 소스 프로젝트의 목록 추가	이러한 오픈 소스 프로젝트는 Amazon EKS에서 관리하는 클러스터를 비롯하여 AWS에서 실행되는 Kubernetes 클러스터의 기능을 확장합니다.	2019년 3월 15일
Helm을 로컬로 설치하기 위한 주제 추가	Kubernetes용 `helm` 패키지 관리자는 Kubernetes 클러스터에서 애플리케이션을 설치하고 관리하는 데 도움이 됩니다. 이 주제에서는 `helm` 및 `tiller` 바이너리를 로컬에서 설치하고 실행하는 방법을 설명합니다. 이렇게 하면 로컬 시스템에서 Helm CLI를 사용하여 차트를 설치하고 관리할 수 있습니다.	2019년 3월 11일
Amazon EKS 플랫폼 버전 업데이트	`CVE-2019-1002100` 을 해결하기 위해 Amazon EKS Kubernetes `1.11` 클러스터를 패치 수준 `1.11.8`로 업데이트하는 새로운 플랫폼 버전.	2019년 3월 8일
클러스터 한도 향상	Amazon EKS는 AWS 리전에서 생성할 수 있는 클러스터의 수를 3개에서 50개로 늘렸습니다.	2019년 2월 13일

Amazon EKS AWS 리전 확장	이제 EU(런던) (eu-west-2), EU(파리)(eu-west-3) 및 아시아 태평양(뭄바이)(ap-south-1) AWS 리전에서 Amazon EKS를 사용할 수 있습니다.	2019년 2월 13일
ALAS-2019-1156 에 패치된 새 Amazon EKS 최적화 AMI	Amazon EKS는 ALAS-2019-1156 에서 설명하는 취약성을 해결하기 위해 Amazon EKS 최적화 AMI 업데이트했습니다.	2019년 2월 11일
ALAS2-2019-1141 에 패치된 새 Amazon EKS 최적화 AMI	Amazon EKS는 ALAS2-2019-1141 에서 참조하는 CVE를 해결하기 위해 Amazon EKS 최적화 AMI 업데이트했습니다.	2019년 1월 9일
Amazon EKS AWS 리전 확장	이제 아시아 태평양(서울)(ap-northeast-2) AWS 리전에서 Amazon EKS를 사용할 수 있습니다.	2019년 1월 9일
Amazon EKS 지역 확장	이제 추가적으로 EU(프랑크푸르트)(eu-central-1), 아시아 태평양(도쿄)(ap-northeast-1), 아시아 태평양(싱가포르)(ap-southeast-1) 및 아시아 태평양(시드니)(ap-southeast-2) AWS 리전에서 Amazon EKS를 사용할 수 있습니다.	2018년 12월 19일

Amazon EKS 클러스터 업데이트	Amazon EKS 클러스터 Kubernetes 버전 업데이트 및 노드 교체에 대한 문서를 추가했습니다.	2018년 12월 12일
Amazon EKS AWS 리전 확장	이제 EU(스톡홀름)(eu-north-1) AWS 리전 지역에서 Amazon EKS를 사용할 수 있습니다.	2018년 12월 11일
Amazon EKS 플랫폼 버전 업데이트	Kubernetes를 패치 레벨 1.10.11로 업데이트하여 CVE-2018-1002105 를 해결하는 새 플랫폼 버전.	2018년 12월 4일
ALB 인그레스 컨트롤러에 대한 버전 1.0.0 지원 추가	ALB 인그레스 컨트롤러는 AWS의 공식적인 지원과 함께 버전 1.0.0을 릴리스합니다.	2018년 11월 20일
CNI 네트워크 구성에 대한 지원 추가	이제 Amazon VPC CNI plugin for Kubernetes 버전 1.2.1에서 보조 Pod 네트워크 인터페이스에 대한 사용자 지정 네트워크 구성이 지원됩니다.	2018년 10월 16일
MutatingAdmissionWebhook 및 ValidatingAdmissionWebhook 에 대한 지원 추가	이제 Amazon EKS 플랫폼 버전 1.10-eks.2 가 MutatingAdmissionWebhook 및 ValidatingAdmissionWebhook 승인 컨트롤러를 지원합니다.	2018년 10월 10일
파트너 AMI 정보 추가	Canonical은 Amazon EKS와 협력하여 클러스터에서 사용할 수 있는 노드 AMI를 만들었습니다.	2018년 10월 3일

AWS CLIupdate-kubeconfig 명령에 대한 지침 추가	Amazon EKS는 클러스터에 액세스하는 데 필요한 kubeconfig 파일을 만드는 과정을 단순화하기 위해 update-kubeconfig 를 AWS CLI에 추가했습니다.	2018년 9월 21일
새로운 Amazon EKS 최적화 AMI	Amazon EKS는 다양한 보안 수정 및 AMI 최적화를 제공하기 위해 Amazon EKS 최적화 AMI(GPU 지원 유무와 무관)를 업데이트했습니다.	2018년 9월 13일
Amazon EKS AWS 리전 확장	이제 EU(아일랜드)(eu-west-1) 리전에서 Amazon EKS를 사용할 수 있습니다.	2018년 9월 5일
Amazon EKS 플랫폼 버전 업데이트	Kubernetes 집계 계층 및 Horizontal Pod Autoscaler(HPA)를 지원하는 새로운 플랫폼 버전.	2018년 8월 31일
새로운 Amazon EKS 최적화 AMI 및 GPU 지원	Amazon EKS는 새로운 AWS CloudFormation 노드 템플릿 및 부트스트랩 스크립트를 사용하도록 Amazon EKS 최적화 AMI를 업데이트했습니다. 또한 새로운 Amazon EKS 최적화 AMI(GPU 지원 포함)도 이용할 수 있습니다.	2018년 8월 22일
ALAS2-2018-1058 에 패치된 새 Amazon EKS 최적화 AMI	Amazon EKS는 ALAS2-2018-1058 에서 참조하는 CVE를 해결하기 위해 Amazon EKS 최적화 AMI 업데이트했습니다.	2018년 8월 14일

Amazon EKS 최적화 AMI 빌드 스크립트	Amazon EKS는 Amazon EKS 최적화 AMI를 빌드하는 데 사용되는 빌드 스크립트를 오픈 소스로 제공합니다. 이러한 빌드 스크립트를 이제 GitHub에서 사용할 수 있습니다.	2018년 7월 10일
Amazon EKS 최초 릴리스	서비스 출시에 대한 최초 설명서	2018년 6월 5일