Amazon EKS

사용 설명서

Amazon EKS: 사용 설명서

Copyright © 2019 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Amazon EKS란 무엇입니까? 1
- Amazon EKS 작동 방식 2

시작하기 3
- Amazon EKS 필수 조건 3
  - Amazon EKS 서비스 역할 생성 3
  - Amazon EKS 클러스터 VPC 생성 3
  - Amazon EKS에 대한 kubectl을 설치하고 구성합니다. 4
  - (선택 사항) 최신 AWS CLI 다운로드 및 설치 6
- 1단계: Amazon EKS 클러스터 생성 6
- 2단계: Amazon EKS용 `kubectl` 구성 8
- 3단계: Amazon EKS 작업자 노드 시작 및 구성 9
- 4단계: 게스트 북 애플리케이션 시작 12
- 5단계: 게스트 북 객체 정리 14

클러스터 16
- 클러스터 생성 16
- 클러스터 업데이트 19
- 클러스터 삭제 21
- 플랫폼 버전 22
  - Kubernetes 1.11 플랫폼 버전 22
  - Kubernetes 1.10 플랫폼 버전 22

작업자 노드 24
- Amazon EKS 최적화 AMI 24
  - Amazon EKS 최적화 AMI 빌드 스크립트 25
  - GPU 지원이 포함된 Amazon EKS 최적화 AMI 25
- 파트너 AMI 27
- Amazon EKS 작업자 노드 시작 28
- 작업자 노드 업데이트 31
  - 새 작업자 노드 그룹으로 마이그레이션 31
  - 기존 작업자 노드 그룹 업데이트 35

스토리지 클래스 37
로드 밸런싱 39
네트워킹 40
- 클러스터 VPC 고려 사항 40
  - VPC 태그 지정 요건 41
  - 서브넷 태그 지정 요건 41
  - 내부 로드 밸런서에 대한 프라이빗 서브넷 태그 지정 요구사항 41
- 클러스터 보안 그룹 고려 사항 41
- 포드 네트워킹 43
- CNI 구성 변수 45
- CoreDNS 설치 46
- 외부 SNAT 48
- CNI 사용자 지정 네트워킹 51
- CNI 업그레이드 52
- Amazon EKS에 Calico 설치 53
  - Stars Policy 데모 54

클러스터 인증 관리 58
- `kubectl` 설치 58
  - MacOS 59
  - 리눅스 59
  - Windows가 설치된 60
- `aws-iam-authenticator` 설치 61
- Amazon EKS에 대한 `kubeconfig` 생성 62
- 클러스터의 사용자 또는 IAM 역할 관리 65

서비스 제한 69

IAM 정책, 역할 및 권한 70
정책 구조 70
정책 구문 70
Amazon EKS 작업 71
권한 테스트 71
IAM 정책 만들기 72
Amazon EKS 서비스 IAM 역할 72
기존 AWSServiceRoleForAmazonEKS 역할 확인 73
AWSServiceRoleForAmazonEKS 역할 만들기 73
자습서: Kubernetes 대시보드 배포 75
사전 요구 사항 76
1단계: 대시보드 배포 77
2단계: eks-admin 서비스 계정 및 클러스터 역할 바인딩 생성 78
3단계: 대시보드에 연결 79
4단계: 다음 절차 80
자습서: Amazon EKS에 대한 VPC 생성 81
1단계: NAT 게이트웨이에 대한 탄력적 IP 주소 만들기 81
2단계: VPC 마법사 실행 81
3단계: 추가 서브넷 생성 82
4단계: 프라이빗 서브넷 태그 지정 82
5단계: 제어 플레인 보안 그룹 생성 83
다음 단계 83
CloudTrail 84
CloudTrail의 Amazon EKS 정보 84
Amazon EKS 로그 파일 항목 이해 85
공동 책임 86
문제 해결 87
용량 부족 87
aws-iam-authenticator 찾을 수 없음 87
작업자 노드와 클러스터 조인 실패 87
권한이 없거나 액세스가 거부됨 87
hostname doesn't match 88
CNI 로그 수집 도구 88
문서 기록 90
AWS Glossary 92

Amazon EKS란 무엇입니까?

Amazon Elastic Container Service for Kubernetes(Amazon EKS)는 Kubernetes 제어 플레인을 설치하고 운영할 필요 없이 AWS에서 Kubernetes를 손쉽게 실행하도록 하는 관리형 서비스입니다. Kubernetes는 컨테이너화된 애플리케이션의 배포, 조정 및 관리 자동화를 위한 오픈 소스 시스템입니다.

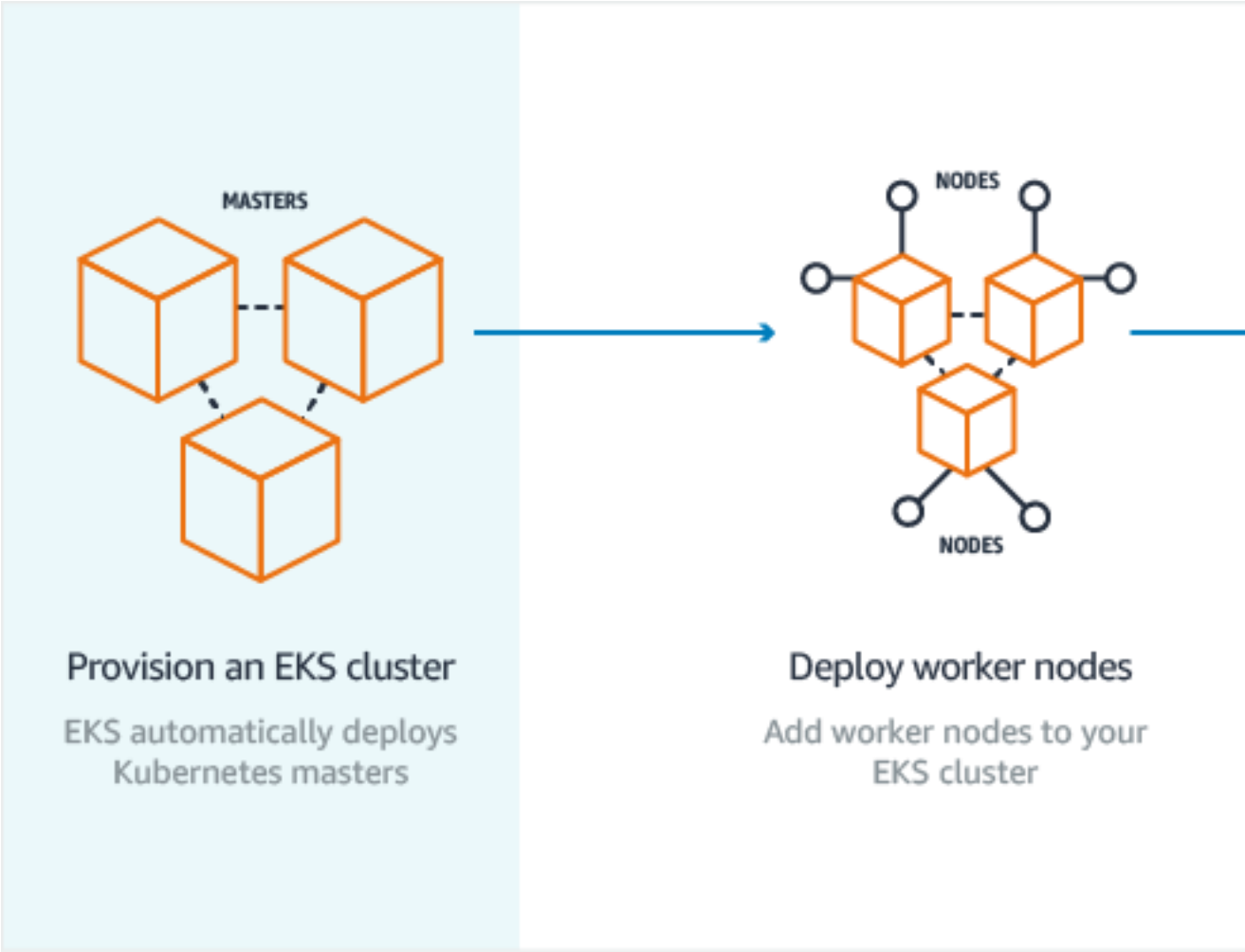
Amazon EKS는 여러 가용 영역에서 Kubernetes 제어 플레인 인스터스를 실행하여 고가용성을 보장합니다. Amazon EKS는 비정상 제어 플레인 인스턴스를 자동으로 감지하고 교체합니다. 또한 자동화된 버전 업그레이드를 제공할 뿐 아니라 이에 대한 버전 업그레이드 및 패치를 자동으로 제공합니다.

Amazon EKS는 또한 여러 AWS 서비스와 통합되어 다음을 포함한 애플리케이션에 대한 확장성과 보안을 제공합니다.

- 컨테이너 이미지용 Amazon ECR
- 로드 배포용 Elastic Load Balancing
- 인증용 IAM
- 격리용 Amazon VPC

Amazon EKS는 오픈 소스 Kubernetes 소프트웨어의 최신 버전을 실행하므로 Kubernetes 커뮤니티에서 모든 기존 플러그인과 도구를 사용할 수 있습니다. 온프레미스 데이터 센터에서 실행 중인지 퍼블릭 클라우드에서 실행 중인지에 상관없이, Amazon EKS에서 실행 중인 애플리케이션은 표준 Kubernetes 환경에서 실행 중인 애플리케이션과 완벽하게 호환됩니다. 즉, 필요한 코드를 수정하지 않고 표준 Kubernetes 애플리케이션을 Amazon EKS로 쉽게 마이그레이션할 수 있습니다.

Amazon EKS 작동 방식



Amazon EKS 손쉽게 시작하기:

1. 우선 AWS Management 콘솔이나 AWS CLI 또는 AWS SDK 중 하나를 사용하여 Amazon EKS 클러스터를 생성합니다.
2. 그런 다음 작업자 노드를 시작하여 Amazon EKS 클러스터를 등록합니다. 노드를 자동으로 구성하는 AWS CloudFormation 템플릿이 제공됩니다.
3. 클러스터가 준비되면 원하는 Kubernetes 도구(예: kubectl)를 사용하여 클러스터와 통신할 수 있습니다.
4. 다른 Kubernetes 환경에서와 마찬가지로 Amazon EKS 클러스터에 애플리케이션을 배포 및 관리합니다.

필요한 리소스 생성 및 최초 Amazon EKS 클러스터 생성에 대한 자세한 내용은 Amazon EKS 시작하기 (p. 3) 단원을 참조하십시오.

Amazon EKS 시작하기

이 시작 안내서는 Amazon EKS 시작하기에 필요한 모든 리소스를 생성하는 데 도움이 됩니다.

Amazon EKS 필수 조건

Amazon EKS 클러스터를 생성하기 전에 Kubernetes가 AWS 리소스 생성 시 수임할 수 있는 IAM 역할을 생성해야 합니다. 예를 들어 로드 밸런서가 생성될 때 Kubernetes는 역할을 수임하여 계정에 Elastic Load Balancing 로드 밸런서를 생성합니다. 한 번만 수행해야 하고 여러 EKS 클러스터에 대해 사용할 수 있습니다.

클러스터가 사용할 보안 그룹과 VPC를 생성해야 합니다. 여러 클러스터에 대해 VPC 및 보안 그룹을 사용할 수 있지만 더욱 효율적인 네트워크 격리를 제공하기 위해 각 EKS 클러스터에 별도의 VPC를 사용하는 것이 좋습니다.

이 단원은 또한 kubectl 바이너리를 설치하고 Amazon EKS를 사용하도록 구성하는 데 도움이 됩니다.

Amazon EKS 서비스 역할 생성

IAM 콘솔에서 Amazon EKS 서비스 역할을 만들려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. [Roles]를 선택한 다음 [Create role]을 선택합니다.
3. 서비스 목록에서 EKS를 선택한 다음 사용 사례에 대해 Allows Amazon EKS to manage your clusters on your behalf(EKS에서 사용자를 대신하여 클러스터를 관리하도록 허용)를 선택하고 Next: Permissions(다음: 권한)를 선택합니다.
4. [Next: Review]를 선택합니다.
5. Role name(역할 이름)에서 역할에 대한 고유 이름(예: `eksServiceRole`)을 입력한 다음 Create role(역할 생성)을 선택합니다.

Amazon EKS 클러스터 VPC 생성

클러스터 VPC를 생성하려면

1. https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
2. 탐색 모음에서 Amazon EKS를 지원하는 리전을 선택합니다.

   Note

   현재 다음 리전에서 Amazon EKS를 사용할 수 있습니다.

   - 미국 서부(오리건) (`us-west-2`)
   - 미국 동부(버지니아 북부) (`us-east-1`)
   - 미국 동부(오하이오)(`us-east-2`)
   - EU(아일랜드) (`eu-west-1`)
3. [Create stack]을 선택합니다.

4. 템플릿 선택에서 Amazon S3 템플릿 URL 지정을 선택합니다.
5. 텍스트 영역에 다음 URL을 붙여넣고 다음을 선택합니다.

```
https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2018-11-07/amazon-eks-vpc-sample.yaml
```

6. Specify Details(세부 정보 지정) 페이지에서 파라미터를 입력하고 다음을 선택합니다.
   - 스택 이름: AWS CloudFormation 스택에 대한 스택 이름을 선택합니다. 예를 들어 eks-vpc로 사용할 수 있습니다.
   - VpcBlock: VPC에 대한 CIDR 범위를 선택합니다. 기본값을 유지할 수 있습니다.
   - Subnet01Block: 서브넷 1에 대한 CIDR 범위를 선택합니다. 기본값을 유지할 수 있습니다.
   - Subnet02Block: 서브넷 2에 대한 CIDR 범위를 선택합니다. 기본값을 유지할 수 있습니다.
   - Subnet03Block: 서브넷 3에 대한 CIDR 범위를 선택합니다. 기본값을 유지할 수 있습니다.
7. (선택 사항) 옵션 페이지에서 스택 리소스에 태그를 지정합니다. [Next]를 선택합니다.
8. [Review] 페이지에서 [Create]을 선택합니다.
9. 스택이 생성된 후 콘솔에서 이를 선택하고 출력을 선택합니다.
10. 생성된 보안 그룹에 대한 SecurityGroups 값을 기록합니다. 이 값은 EKS 클러스터를 생성할 때 필요합니다. 이 보안 그룹은 서브넷에 생성된 계정 간 탄력적 네트워크 인터페이스에 적용되고, 이를 통해 Amazon EKS 제어 플레인이 작업자 노드와 통신할 수 있습니다.
11. 생성된 VPC의 VpcId를 기록합니다. 작업자 노드 그룹 템플릿을 시작할 때 필요합니다.
12. 생성된 서브넷에 대한 SubnetIds를 기록합니다. EKS 클러스터를 생성할 때 필요합니다. 작업자 노드가 시작되는 서브넷입니다.

Amazon EKS에 대한 kubectl을 설치하고 구성합니다.

Kubernetes는 클러스터 API 서버와 통신하기 위해 `kubectl`이라는 명령줄 유틸리티를 사용합니다. 또한 Amazon EKS 클러스터는 Kubernetes 클러스터에 대한 IAM 인증을 위해 Kubernetes용 AWS IAM Authenticator를 필요로 합니다. Kubernetes 버전 1.10부터 Kubernetes용 AWS IAM Authenticator를 설치하고 인증에 사용할 kubectl 구성 파일을 사용하여 Amazon EKS를 사용할 kubectl 클라이언트를 구성할 수 있습니다.

Amazon EKS는 동일한 버전의 업스트림 aws-iam-authenticator 바이너리와 동일하게 사용할 수 있는 aws-iam-authenticator 바이너리를 판매합니다. 또는 go get을 사용하여 GitHub의 Kubernetes용 AWS IAM Authenticator 프로젝트에서 바이너리를 가져올 수 있습니다.

Amazon EKS용 kubectl를 설치하려면

- 운영 체제에 따라 kubectl을 다운로드하고 설치하는 다양한 옵션이 있습니다.
  - `kubectl` 바이너리는 많은 운영 체제 패키지 관리자에 제공되며, 이 옵션은 대체로 수동 다운로드 및 설치 프로세스보다 훨씬 쉽습니다. 해당 운영 체제 또는 패키지 관리자에 대한 Kubernetes 문서의 지침에 따라 설치할 수 있습니다.
  - Amazon EKS는 동일한 버전의 업스트림 kubectl 바이너리와 동일하게 사용할 수 있는 kubectl 바이너리도 판매합니다. 운영 체제에 해당하는 Amazon EKS 판매 바이너리를 설치하려면 `kubectl` 설치 (p. 58)를 참조하십시오.

Amazon EKS용 **`aws-iam-authenticator`**를 설치하려면

1. Amazon S3에서 Amazon EKS 판매 `aws-iam-authenticator` 바이너리를 다운로드합니다.
   - Linux: https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/aws-iam-authenticator

- MacOS: https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/darwin/amd64/aws-iam-authenticator
- Windows: https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/windows/amd64/aws-iam-authenticator.exe

아래 명령을 사용하여 바이너리를 다운로드하고 플랫폼에 맞는 올바른 URL로 교체합니다. 아래 예는 MacOS 클라이언트에 해당됩니다.

```
curl -o aws-iam-authenticator https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/darwin/amd64/aws-iam-authenticator
```

2. (선택 사항) 동일한 버킷 접두사에 제공된 SHA-256 합계를 사용하여 다운로드한 바이너리를 확인하고 플랫폼에 맞는 올바른 URL로 바꿉니다.

   a. 해당 시스템에 맞는 SHA-256 합계를 다운로드하십시오. 아래 예는 MacOS 클라이언트의 SHA-256 합계를 다운로드합니다.

   ```
   curl -o aws-iam-authenticator.sha256 https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/darwin/amd64/aws-iam-authenticator.sha256
   ```

   b. 다운로드한 바이너리에 대해 SHA-256 합계를 확인합니다. 아래의 예제 `openssl` 명령은 MacOS 및 Ubuntu 클라이언트에 대해 테스트되었습니다. 해당 운영 체제에서 다른 명령이나 구문을 사용하여 SHA-256 합계를 검사할 수도 있습니다. 필요한 경우 해당 운영 체제 설명서를 참조하십시오.

   ```
   openssl sha1 -sha256 aws-iam-authenticator
   ```

   c. 명령 출력에 생성된 SHA-256 합계를 다운로드한 `aws-iam-authenticator.sha256` 파일과 비교합니다. 두 값이 일치해야 합니다.

3. 바이너리에 실행 권한을 적용합니다.

   ```
   chmod +x ./aws-iam-authenticator
   ```

4. 바이너리를 `$PATH`의 폴더에 복사합니다. `$HOME/bin/aws-iam-authenticator`를 생성하고 `$PATH`가 `$HOME/bin`로 시작하는 것이 좋습니다.

   ```
   cp ./aws-iam-authenticator $HOME/bin/aws-iam-authenticator && export PATH=$HOME/bin:$PATH
   ```

5. `PATH` 환경 변수에 `$HOME/bin`을 추가합니다.

   - MacOS의 Bash 셸의 경우:

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

   - Linux의 Bash 셸의 경우:

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

6. `aws-iam-authenticator` 바이너리가 작동하는지 테스트합니다.

   ```
   aws-iam-authenticator help
   ```

(선택 사항) 최신 AWS CLI 다운로드 및 설치

AWS CLI가 Amazon EKS 사용을 명시적으로 요구하지는 않지만, update-kubeconfig 명령을 사용하면 kubeconfig 생성 프로세스가 크게 간소화됩니다. AWS CLI에서 Amazon EKS를 사용하려면 1.16.18 버전 이상의 AWS CLI가 설치되어 있어야 합니다. AWS CLI를 설치 또는 업그레이드하려면 AWS Command Line Interface 사용 설명서의 AWS 명령줄 인터페이스 설치를 참조하십시오.

> **Important**
>
> yum, apt-get 또는 Homebrew 등의 MacOS용 패키지 관리자는 흔히 AWS CLI의 여러 버전 뒤에 있습니다. 최신 버전이 설치되어 있는지 확인하려면 AWS Command Line Interface 사용 설명서의 AWS 명령줄 인터페이스 설치를 참조하십시오.

AWS CLI 버전은 다음 명령을 통해 확인할 수 있습니다.

```
aws --version
```

> **Note**
>
> 시스템의 Python 버전이 Python 3 또는 Python 2.7.9 이상이어야 합니다. 그렇지 않은 경우 Amazon EKS에 대한 AWS CLI 호출과 함께 `hostname doesn't match` 오류가 표시됩니다. 자세한 내용은 Python 요청 FAQ의 What are "hostname doesn't match" errors?를 참조하십시오.

1단계: Amazon EKS 클러스터 생성

이제 Amazon EKS 클러스터를 생성할 수 있습니다.

> **Important**
>
> Amazon EKS 클러스터가 생성될 때 클러스터를 생성하는 IAM 엔터티(사용자 또는 역할)가 Kubernetes RBAC 권한 부여 표에 관리자(`system:master` 권한이 있음)로 추가됩니다. 처음에는 해당 IAM 사용자만이 kubectl을 사용하여 Kubernetes API를 호출할 수 있습니다. 자세한 정보는 클러스터의 사용자 또는 IAM 역할 관리 (p. 65) 단원을 참조하십시오. 또한 Kubernetes용 AWS IAM Authenticator에서는 Go용 AWS SDK을 사용하여 Amazon EKS 클러스터에 대한 인증을 수행합니다. 콘솔을 사용하여 클러스터를 생성하는 경우 클러스터에서 kubectl 명령을 실행할 때 동일한 IAM 사용자 자격 증명이 AWS SDK 자격 증명 체인에 있어야 합니다.
> AWS CLI를 설치하고 구성하면 사용자에 대한 IAM 자격 증명을 구성할 수 있습니다. 이 자격 증명은 Kubernetes용 AWS IAM Authenticator에 대해서도 유효합니다. AWS CLI가 사용자에 대해 적절하게 구성된 경우 Kubernetes용 AWS IAM Authenticator에서도 이 자격 증명을 찾을 수 있습니다. 자세한 정보는 AWS Command Line Interface 사용 설명서의 AWS CLI 구성을 참조하십시오.

콘솔을 사용하여 클러스터를 생성하려면

1. Open the Amazon EKS console at https://console.aws.amazon.com/eks/home#/clusters.
2. [Create cluster]를 선택합니다.

   > **Note**
   >
   > IAM 사용자에게 관리 권한이 없는 경우 해당 사용자가 Amazon EKS API 작업을 호출하는 권한을 명시적으로 추가해야 합니다. 자세한 내용은 Amazon EKS IAM 정책 만들기 (p. 72) 단원을 참조하십시오.
3. 클러스터 생성 페이지에서 다음 필드를 입력한 다음 생성을 선택합니다.
   - Cluster name: 클러스터에 대해 고유한 이름입니다.
   - Kubernetes version: 클러스터에 대해 사용할 Kubernetes 버전. 기본적으로 최신 버전이 선택됩니다.
   - 역할 ARN: Amazon EKS 서비스 역할 생성 (p. 3) 사용을 통해 생성한 IAM 역할을 선택합니다.

- VPC: Amazon EKS 클러스터 VPC 생성 (p. 3) 사용을 통해 생성한 VPC. 드롭다운 목록에서 VPC의 이름을 찾을 수 있습니다.
- 서브넷: Amazon EKS 클러스터 VPC 생성 (p. 3) 사용을 통해 생성된 AWS CloudFormation 출력의 SubnetIds 값(쉼표로 구분). 기본적으로 위 VPC에서 사용 가능한 서브넷이 사전 선택됩니다.
- 보안 그룹: Amazon EKS 클러스터 VPC 생성 (p. 3) 사용을 통해 생성된 AWS CloudFormation 출력의 SecurityGroups 값. 이 보안 그룹의 드롭다운 이름에는 ControlPlaneSecurityGroup이 있습니다.

  Important

  작업자 노드 AWS CloudFormation 템플릿이 여기서 사용자가 지정하는 보안 그룹을 수정하므로 클러스터 제어 플레인에 대한 전용 보안 그룹을 사용하는 것이 바람직합니다. 다른 리소스와 공유할 경우 이러한 리소스를 방해하거나 막을 수 있습니다.

  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.

4. 클러스터 페이지에서 새로 생성한 클러스터의 이름을 선택하고 클러스터 정보를 확인합니다.
5. Status(상태) 필드는 클러스터 프로비저닝 프로세스가 완료될 때까지 CREATING(생성 중)으로 표시됩니다.

AWS CLI를 사용하여 클러스터를 생성하려면

1. 다음 명령을 사용하여 클러스터를 생성합니다. 클러스터 이름, Amazon EKS 서비스 역할 생성 (p. 3)에서 생성한 Amazon EKS 서비스 역할의 Amazon 리소스 이름(ARN), Amazon EKS 클러스터 VPC 생성 (p. 3)에서 생성한 VPC에 대한 서브넷 및 보안 그룹 ID를 교체합니다.

```
aws eks --region region create-cluster --name devel --role-
arn arn:aws:iam::111122223333:role/eks-service-role-AWSServiceRoleForAmazonEKS-
EXAMPLEBKZRQR --resources-vpc-config subnetIds=subnet-
a9189fe2,subnet-50432629,securityGroupIds=sg-f5c54184
```

   Important

   다음과 유사한 구문 오류를 수신한 경우 Amazon EKS용 AWS CLI의 미리 보기 버전을 사용할 수 있습니다. 퍼블릭 서비스가 시작된 이후 여러 명령의 구문이 변경되었습니다. AWS CLI 버전을 사용 가능한 최신 버전으로 업데이트하고 `~/.aws/models/eks`에서 사용자 지정 서비스 모델 디렉터리를 반드시 삭제하십시오.

```
aws: error: argument --cluster-name is required
```

   Note

   IAM 사용자에게 관리 권한이 없는 경우 해당 사용자가 Amazon EKS API 작업을 호출하는 권한을 명시적으로 추가해야 합니다. 자세한 내용은 Amazon EKS IAM 정책 만들기 (p. 72) 단원을 참조하십시오.

   결과:

```
{
    "cluster": {
        "name": "devel",
        "arn": "arn:aws:eks:us-west-2:111122223333:cluster/devel",
```

```
        "createdAt": 1527785885.159,
        "version": "1.10",
        "roleArn": "arn:aws:iam::111122223333:role/eks-service-role-
AWSServiceRoleForAmazonEKS-AFNL4H8HB71F",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-a9189fe2",
                "subnet-50432629"
            ],
            "securityGroupIds": [
                "sg-f5c54184"
            ],
            "vpcId": "vpc-a54041dc"
        },
        "status": "CREATING",
        "certificateAuthority": {}
    }
}
```

2. 클러스터 프로비저닝은 일반적으로 10분 미만이 걸립니다. 다음 명령을 사용하여 클러스터의 상태를 쿼리할 수 있습니다. 클러스터 상태가 `ACTIVE`인 경우 계속 진행할 수 있습니다.

```
aws eks --region region describe-cluster --name devel --query cluster.status
```

2단계: Amazon EKS용 kubectl 구성

이 단원에서는 AWS CLI update-kubeconfig 명령으로 클러스터에 대한 `kubeconfig` 파일을 생성합니다. AWS CLI를 설치하고 싶지 않거나, kubeconfig를 수동으로 생성하거나 업데이트하려는 경우 Amazon EKS에 대한 kubeconfig 생성 (p. 62)를 참조하십시오.

AWS CLI로 **kubeconfig**를 생성하려면

1. 1.16.18 버전 이상의 AWS CLI가 설치되어 있어야 합니다. AWS CLI를 설치 또는 업그레이드하려면 AWS Command Line Interface 사용 설명서의 AWS 명령줄 인터페이스 설치를 참조하십시오.

   Note

   시스템의 Python 버전이 Python 3 또는 Python 2.7.9 이상이어야 합니다. 그렇지 않은 경우 Amazon EKS에 대한 AWS CLI 호출과 함께 `hostname doesn't match` 오류가 표시됩니다. 자세한 내용은 Python 요청 FAQ의 What are "hostname doesn't match" errors?를 참조하십시오.

   AWS CLI 버전은 다음 명령을 통해 확인할 수 있습니다.

```
aws --version
```

   Important

   yum, apt-get 또는 Homebrew 등의 MacOS용 패키지 관리자는 흔히 AWS CLI의 여러 버전 뒤에 있습니다. 최신 버전이 설치되어 있는지 확인하려면 AWS Command Line Interface 사용 설명서의 AWS 명령줄 인터페이스 설치를 참조하십시오.

2. AWS CLI update-kubeconfig 명령을 사용하여 클러스터에 대한 kubeconfig를 생성하거나 업데이트합니다.

   - 기본적으로, 구성 파일은 홈 디렉터리의 기본 kubeconfig 경로(`.kube/config`)에 생성되거나 해당 위치의 기존 kubeconfig와 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 엔터티가 사용됩니다. aws sts get-caller-identity 명령을 실행하여 기본 AWS CLI 또는 SDK ID를 확인할 수 있습니다.
- 자세한 내용은 aws eks update-kubeconfig help 명령의 도움말 페이지 또는 AWS CLI Command Reference의 update-kubeconfig를 참조하십시오.

```
aws eks --region region update-kubeconfig --name cluster_name
```

3. 구성을 테스트합니다.

```
kubectl get svc
```

Note

`"aws-iam-authenticator": executable file not found in $PATH` 오류가 표시되는 경우 Amazon EKS에 대해 kubectl이 구성되지 않았습니다. 자세한 내용은 ??? 단원을 참조하십시오.

결과:

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

3단계: Amazon EKS 작업자 노드 시작 및 구성

VPC 및 Kubernetes 제어 플레인이 생성되었으므로 작업자 노드를 시작 및 구성할 수 있습니다.

Important

Amazon EKS 작업자 노드는 표준 Amazon EC2 인스턴스이고, 일반 Amazon EC2 온디맨드 인스턴스 가격을 기반으로 비용이 청구됩니다. 자세한 내용은 Amazon EC2 요금 단원을 참조하십시오.

작업자 노드를 시작하려면

1. 클러스터 상태가 `ACTIVE`가 되기를 기다립니다. 클러스터가 활성화되기 전에 작업자 노드를 시작하면 작업자 노드가 클러스터에 등록되지 않고 작업자 노드를 다시 시작해야 합니다.
2. https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
3. 탐색 모음에서 Amazon EKS를 지원하는 리전을 선택합니다.

   Note

   현재 다음 리전에서 Amazon EKS를 사용할 수 있습니다.

   - 미국 서부(오리건) (`us-west-2`)
   - 미국 동부(버지니아 북부) (`us-east-1`)
   - 미국 동부(오하이오)(`us-east-2`)
   - EU(아일랜드) (`eu-west-1`)
4. [Create stack]을 선택합니다.
5. 템플릿 선택에서 Amazon S3 템플릿 URL 지정을 선택합니다.
6. 텍스트 영역에 다음 URL을 붙여넣고 다음을 선택합니다.

```
https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2018-11-07/amazon-eks-nodegroup.yaml
```

7. Specify Details(세부 정보 지정) 페이지에서 다음 파라미터를 입력하고 다음을 선택합니다.

   - 스택 이름: AWS CloudFormation 스택에 대한 스택 이름을 선택합니다. 예를 들어 ***<cluster-name>***-worker-nodes로 사용할 수 있습니다.
   - ClusterName: Amazon EKS 클러스터 생성 시 사용할 이름을 입력합니다.

     Important

     이 이름은 1단계: Amazon EKS 클러스터 생성 (p. 6)에서 사용하는 이름과 정확하게 일치해야 합니다. 그렇지 않은 경우 작업자 노드가 클러스터에 조인할 수 없습니다.
   - ClusterControlPlaneSecurityGroup: Amazon EKS 클러스터 VPC 생성 (p. 3) 사용을 통해 생성된 AWS CloudFormation 출력의 SecurityGroups 값을 선택합니다.
   - NodeGroupName: 노드 그룹의 이름을 입력합니다. 이 이름은 나중에 작업자 노드에 대해 생성된 Auto Scaling 노드 그룹을 식별하는 데 사용할 수 있습니다.
   - NodeAutoScalingGroupMinSize: Auto Scaling 그룹이 축소할 수 있는 작업자 노드의 최소 노드 수를 입력합니다.
   - NodeAutoScalingGroupDesiredCapacity: 스택을 생성할 때 조정할 원하는 노드 수를 입력합니다.
   - NodeAutoScalingGroupMaxSize: Auto Scaling 그룹이 확장할 수 있는 작업자 노드의 최대 노드 수를 입력합니다.
   - NodeInstanceType: 작업자 노드에 대한 인스턴스 유형을 선택합니다.
   - NodeImageId: 리전에 대한 현재 Amazon EKS 작업자 노드 AMI ID를 입력합니다. 최신 Amazon EKS 최적화 AMI(GPU 지원 (p. 25) 유무는 상이)의 AMI ID는 다음 표에 나와 있습니다.

     Note

     GPU 지원이 포함된 Amazon EKS 최적화 AMI는 P2 및 P3 인스턴스 유형만 지원합니다. 작업자 노드 AWS CloudFormation 템플릿에서 이러한 인스턴스 유형을 지정해야 합니다. 이 AMI에는 최종 사용자 라이선스 계약(EULA)이 필요한 타사 소프트웨어가 포함되므로 작업자 노드 그룹에서 AMI를 사용하려면 먼저 AWS Marketplace에서 AMI에 가입하고 EULA에 동의해야 합니다. AMI에 가입하려면 AWS Marketplace를 방문하십시오.

리전	Amazon EKS 최적화 AMI	(GPU 지원 포함)
미국 서부(오리건) (us-west-2)	ami-0f54a2f7d2e9c88b3	ami-08156e8fd65879a13
미국 동부(버지니아 북부) (us-east-1)	ami-0a0b913ef3249b655	ami-0c974dde3f6d691a1
미국 동부(오하이오)(us-east-2)	ami-0958a76db2d150238	ami-089849e811ace242f
EU(아일랜드)(eu-west-1)	ami-00c3b2d35bddd4f5c	ami-0c3479bcd739094f0

     Note

     Amazon EKS 작업자 노드 AMI는 Amazon Linux 2 기반입니다. Amazon Linux 보안 센터에서 Amazon Linux 2에 대한 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.

- KeyName: 시작 이후 SSH를 사용하여 작업자 노드에 연결하는 데 사용할 수 있는 Amazon EC2 SSH 키 페어 이름을 입력합니다. Amazon EC2 키 페어가 아직 없는 경우 AWS Management 콘솔에서 새로 생성할 수 있습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하십시오.

  Note

  여기에 키 페어를 입력하지 않으면 AWS CloudFormation 스택이 생성되지 않습니다.
- BootstrapArguments: 별도의 kubelet 인수와 같이 작업자 노드 부트스트랩 스크립트에 전달할 선택적 인수를 지정합니다. 자세한 내용은 https://github.com/awslabs/amazon-eks-ami/blob/master/files/bootstrap.sh에서 부트스트랩 스크립트 사용 정보를 참조하십시오.
- VpcId: Amazon EKS 클러스터 VPC 생성 (p. 3)에서 생성한 VPC에 대한 ID를 입력합니다.
- Subnets: Amazon EKS 클러스터 VPC 생성 (p. 3)에서 생성한 서브넷을 선택합니다.

8. 옵션 페이지에서 스택 리소스에 태그를 지정할 수 있습니다. [Next]를 선택합니다.
9. 검토 페이지에서 정보를 검토하고, 스택이 IAM 리소스를 생성할 수 있음을 인지한 다음 생성을 선택합니다.
10. 스택이 생성된 후 콘솔에서 이를 선택하고 출력 탭을 선택합니다.
11. 생성된 노드 그룹에 대해 NodeInstanceRole을 기록합니다. Amazon EKS 작업자 노드를 구성할 때 필요합니다.

작업자 노드가 클러스터에 조인하도록 하려면

1. AWS Authenticator 구성 맵 다운로드, 편집 및 적용:

   a. 구성 맵을 다운로드합니다.

   ```
   curl -O https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2018-11-07/aws-auth-cm.yaml
   ```

   b. 즐겨찾는 텍스트 편집기에서 파일을 엽니다. 인스턴스 역할의 *<ARN(##### ####)>* 조각을 이전 절차에서 기록한 NodeInstanceRole 값으로 교체하고 파일을 저장합니다.

   Important

   이 파일에서 어떠한 행도 수정하지 마십시오.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: aws-auth
     namespace: kube-system
   data:
     mapRoles: |
       - rolearn: <ARN of instance role (not instance profile)>
         username: system:node:{{EC2PrivateDNSName}}
         groups:
           - system:bootstrappers
           - system:nodes
   ```

   c. 구성을 적용합니다. 이 명령을 완료하는 데 몇 분이 걸릴 수 있습니다.

   ```
   kubectl apply -f aws-auth-cm.yaml
   ```

Note

`"aws-iam-authenticator": executable file not found in $PATH` 오류가 표시되는 경우 Amazon EKS에 대해 kubectl이 구성되지 않았습니다. 자세한 내용은 ??? 단원을 참조하십시오.

2. 노드의 상태를 확인하고 `Ready` 상태가 될 때까지 대기합니다.

```
kubectl get nodes --watch
```

3. (GPU 작업자만 해당) P2 또는 P3 인스턴스 유형과 GPU 지원이 포함된 Amazon EKS 최적화 AMI를 선택한 경우, 다음 명령을 이용해 클러스터에 Kubernetes용 NVIDIA 디바이스 플러그인을 데몬 세트로 적용해야 합니다.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v1.10/
nvidia-device-plugin.yml
```

4단계: 게스트 북 애플리케이션 시작

이 단원에서는 샘플 게스트 북 애플리케이션을 생성하고 새 클러스터를 테스트합니다.

Note

게스트 북 예제 설정에 대한 자세한 내용은 Kubernetes 문서의 https://github.com/kubernetes/examples/blob/master/guestbook-go/README.md를 참조하십시오.

게스트 북 애플리케이션을 생성하려면

1. Redis 마스터 복제 컨트롤러를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/kubernetes/v1.10.3/
examples/guestbook-go/redis-master-controller.json
```

Note

`"aws-iam-authenticator": executable file not found in $PATH` 오류가 표시되는 경우 Amazon EKS에 대해 kubectl이 구성되지 않았습니다. 자세한 내용은 ??? 단원을 참조하십시오.

결과:

```
replicationcontroller "redis-master" created
```

2. Redis 마스터 서비스를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/kubernetes/v1.10.3/
examples/guestbook-go/redis-master-service.json
```

결과:

```
service "redis-master" created
```

3. Redis 슬레이브 복제 컨트롤러를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/kubernetes/v1.10.3/
examples/guestbook-go/redis-slave-controller.json
```

결과:

```
replicationcontroller "redis-slave" created
```

4. Redis 슬레이브 서비스를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/kubernetes/v1.10.3/
examples/guestbook-go/redis-slave-service.json
```

결과:

```
service "redis-slave" created
```

5. 게스트 북 복제 컨트롤러를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/kubernetes/v1.10.3/
examples/guestbook-go/guestbook-controller.json
```

결과:

```
replicationcontroller "guestbook" created
```

6. 게스트 북 서비스를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/kubernetes/v1.10.3/
examples/guestbook-go/guestbook-service.json
```

결과:

```
service "guestbook" created
```

7. `guestbook` 서비스의 External IP 열이 채워질 때까지 클러스터에서 서비스를 쿼리합니다.

   Note

   IP 주소가 사용 가능할 때까지 몇 분이 걸릴 수 있습니다.

```
kubectl get services -o wide
```

8. 외부 IP 주소가 이용 가능한 이후 웹 브라우저를 포트 3000의 해당 주소로 가리켜 게스트 북을 확인합니다. 예: *http://a7a95c2b9e69711e7b1a3022fdcfdf2e-1985673473.us-west-2.elb.amazonaws.com:3000*

   Note

   DNS가 전파되고 게스트 북이 표시되기까지 몇 분이 걸릴 수 있습니다.

Important

브라우저를 사용하여 외부 IP 주소에 연결할 수 없는 경우 사내 방화벽이 3000과 같은 비표준 포트를 차단하지 않도록 해야 합니다. 게스트 네트워크로 전환하여 확인할 수 있습니다.

5단계: 게스트 북 객체 정리

게스트 북 애플리케이션으로 실험을 완료한 이후 이에 대해 생성한 리소스를 정리해야 합니다. 다음 명령은 게스트 북 애플리케이션에 대한 모든 서비스 및 복제 컨트롤러를 삭제합니다.

```
kubectl delete rc/redis-master rc/redis-slave rc/guestbook svc/redis-master svc/redis-slave
 svc/guestbook
```

Note

`"aws-iam-authenticator": executable file not found in $PATH` 오류가 표시되는 경우 Amazon EKS에 대해 kubectl이 구성되지 않았습니다. 자세한 내용은 ??? 단원을 참조하십시오.

Amazon EKS 클러스터 사용을 완료한 후에는 추가 비용이 발생하지 않도록 클러스터 및 연결된 리소스를 삭제해야 합니다. 자세한 내용은 클러스터 삭제 (p. 21) 단원을 참조하십시오.

Amazon EKS 클러스터

Amazon EKS 클러스터에는 두 가지 기본 구성 요소가 있습니다.

- Amazon EKS 제어 플레인
- 제어 플레인을 사용하여 등록된 Amazon EKS 작업자 노드

Amazon EKS 제어 플레인에는 `etcd` 및 Kubernetes API 서버와 같은 Kubernetes 소프트웨어를 실행하는 제어 플레인 노드가 포함되어 있습니다. 제어 플레인은 AWS에서 관리하는 계정에서 실행됩니다. 또한 Kubernetes API는 클러스터와 연결된 Amazon EKS 엔드포인트를 통해 노출됩니다.

Amazon EKS 작업자 노드는 AWS 계정에서 실행되고, API 서버 엔드포인트 및 클러스터에 대해 생성된 인증 파일을 통해 제어 플레이엔 연결합니다.

클러스터 제어 플레인은 여러 가용 영역에 프로비저닝되고 Elastic Load Balancing, Network Load Balancer와 마주합니다. Amazon EKS는 또한 VPC 서브넷의 탄력적 네트워크 인터페이스를 프로비저닝하여 제어 플레인 인스턴스로부터 작업자 노드로의 연결을 제공합니다(예: kubectl exec, logs, proxy 데이터 흐름 지원).

항목

- Amazon EKS 클러스터 생성 (p. 16)
- Amazon EKS 클러스터 업데이트 (p. 19)
- 클러스터 삭제 (p. 21)
- 플랫폼 버전 (p. 22)

Amazon EKS 클러스터 생성

이 주제에서는 Amazon EKS 클러스터를 생성하는 절차를 살펴봅니다.

처음 Amazon EKS 클러스터를 생성하는 경우 Amazon EKS 시작하기 (p. 3) 가이드를 따르는 것이 좋습니다. 이 가이드는 Amazon EKS 클러스터 생성부터 샘플 Kubernetes 애플리케이션 배포에 이르기까지 종합적인 연습을 제공합니다.

이 주제의 사전 요구 사항은 다음과 같습니다.

- Amazon EKS 클러스터의 요건을 충족하는 VPC 및 전용 보안 그룹을 생성했습니다. 자세한 내용은 클러스터 VPC 고려 사항 (p. 40) 및 클러스터 보안 그룹 고려 사항 (p. 41) 단원을 참조하십시오. Amazon EKS 시작하기 (p. 3) 가이드는 요건을 충족하는 VPC를 생성합니다. 또는 자습서: Amazon EKS 클러스터에 대해 퍼블릭 및 프라이빗 서브넷이 있는 VPC 생성 (p. 81) 가이드를 따라 수동으로 생성할 수 있습니다.
- Amazon EKS 서비스 역할을 생성하여 클러스터에 적용했습니다. Amazon EKS 시작하기 (p. 3) 가이드는 서비스 역할을 생성합니다. 또는 Amazon EKS 서비스 IAM 역할 (p. 72) 가이드를 따라 수동으로 생성할 수 있습니다.

  Important

  Amazon EKS 클러스터가 생성될 때 클러스터를 생성하는 IAM 엔터티(사용자 또는 역할)가 Kubernetes RBAC 권한 부여 표에 관리자(`system:master` 권한이 있음)로 추가됩니다. 처음에는 해당 IAM 사용자만이 kubectl을 사용하여 Kubernetes API를 호출할 수 있습니다. 자세한 정보는 클러스터의 사용자 또는 IAM 역할 관리 (p. 65) 단원을 참조하십시오. 또한 Kubernetes용 AWS IAM Authenticator에서는 Go용 AWS SDK을 사용하여 Amazon EKS 클러스터에 대한 인증을 수행합니다. 콘솔을 사용하여 클러스터를 생성하는 경우 클러스터에서 kubectl 명령을 실행할 때 동일한 IAM 사용자 자격 증명이 AWS SDK 자격 증명 체인에 있어야 합니다.

AWS CLI를 설치하고 구성하면 사용자에 대한 IAM 자격 증명을 구성할 수 있습니다. 이 자격 증명은 Kubernetes용 AWS IAM Authenticator에 대해서도 유효합니다. AWS CLI가 사용자에 대해 적절하게 구성된 경우 Kubernetes용 AWS IAM Authenticator에서도 이 자격 증명을 찾을 수 있습니다. 자세한 정보는 AWS Command Line Interface 사용 설명서의 AWS CLI 구성을 참조하십시오.

콘솔을 사용하여 클러스터를 생성하려면

1. Open the Amazon EKS console at https://console.aws.amazon.com/eks/home#/clusters.
2. [Create cluster]를 선택합니다.

   Note

   IAM 사용자에게 관리 권한이 없는 경우 해당 사용자가 Amazon EKS API 작업을 호출하는 권한을 명시적으로 추가해야 합니다. 자세한 내용은 Amazon EKS IAM 정책 만들기 (p. 72) 단원을 참조하십시오.
3. 클러스터 생성 페이지에서 다음 필드를 입력한 다음 생성을 선택합니다.
   - Cluster name: 클러스터에 대해 고유한 이름입니다.
   - Kubernetes version: 클러스터에 대해 사용할 Kubernetes 버전. 기본적으로 최신 버전이 선택됩니다.
   - 역할 ARN: Amazon EKS 서비스 역할의 Amazon 리소스 이름(ARN). 자세한 내용은 Amazon EKS 서비스 IAM 역할 (p. 72) 단원을 참조하십시오.
   - VPC: 클러스터에 대해 사용할 VPC.
   - Subnets: 클러스터에서 사용할 위 VPC 내 서브넷. 기본적으로 위 VPC에서 사용 가능한 서브넷이 사전 선택됩니다. 서브넷이 Amazon EKS 클러스터 요구 사항을 충족해야 합니다. 자세한 내용은 클러스터 VPC 고려 사항 (p. 40) 단원을 참조하십시오.
   - Security Groups: 클러스터에 대해 여러 계정의 탄력적 네트워크 인터페이스에 적용할 위 VPC 내 하나 이상(최대 5개)의 보안 그룹을 지정합니다. 클러스터 및 작업자 노드 보안 그룹은 Amazon EKS 클러스터 요구 사항을 충족해야 합니다. 자세한 내용은 클러스터 보안 그룹 고려 사항 (p. 41) 단원을 참조하십시오.

     Important

     작업자 노드 AWS CloudFormation 템플릿이 여기서 사용자가 지정하는 보안 그룹을 수정하므로 클러스터 제어 플레인에 대한 전용 보안 그룹을 사용하는 것이 바람직합니다. 다른 리소스와 공유할 경우 이러한 리소스를 방해하거나 막을 수 있습니다.

     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.
4. 클러스터 페이지에서 새로 생성한 클러스터의 이름을 선택하고 클러스터 정보를 확인합니다.
5. Status(상태) 필드는 클러스터 프로비저닝 프로세스가 완료될 때까지 CREATING(생성 중)으로 표시됩니다. 클러스터 프로비저닝이 완료될 때(보통 10분 이내) API server endpoint(API 서버 엔드포인트) 및 인증 기관 값을 기록합니다. 이 값은 kubectl 구성에 사용됩니다.
6. 클러스터를 생성했으므로 이제 `aws-iam-authenticator` 설치 (p. 61) 및 Amazon EKS에 대한 `kubeconfig` 생성 (p. 62)의 절차를 따라 새 클러스터와의 통신을 활성화합니다.

AWS CLI를 사용하여 클러스터를 생성하려면

1. 다음 명령을 사용하여 클러스터를 생성합니다. 클러스터 이름, Amazon EKS 서비스 역할 생성 (p. 3)에서 생성한 Amazon EKS 서비스 역할의 Amazon 리소스 이름(ARN), Amazon EKS 클러스터 VPC 생성 (p. 3)에서 생성한 VPC에 대한 서브넷 및 보안 그룹 ID를 교체합니다.

```
aws eks --region region create-cluster --name devel --role-
arn arn:aws:iam::111122223333:role/eks-service-role-AWSServiceRoleForAmazonEKS-
EXAMPLEBKZRQR --resources-vpc-config subnetIds=subnet-
a9189fe2,subnet-50432629,securityGroupIds=sg-f5c54184
```

Important

다음과 유사한 구문 오류를 수신한 경우 Amazon EKS용 AWS CLI의 미리 보기 버전을 사용할 수 있습니다. 퍼블릭 서비스가 시작된 이후 여러 명령의 구문이 변경되었습니다. AWS CLI 버전을 사용 가능한 최신 버전으로 업데이트하고 `~/.aws/models/eks`에서 사용자 지정 서비스 모델 디렉터리를 반드시 삭제하십시오.

```
aws: error: argument --cluster-name is required
```

Note

IAM 사용자에게 관리 권한이 없는 경우 해당 사용자가 Amazon EKS API 작업을 호출하는 권한을 명시적으로 추가해야 합니다. 자세한 내용은 Amazon EKS IAM 정책 만들기 (p. 72) 단원을 참조하십시오.

결과:

```
{
    "cluster": {
        "name": "devel",
        "arn": "arn:aws:eks:us-west-2:111122223333:cluster/devel",
        "createdAt": 1527785885.159,
        "version": "1.10",
        "roleArn": "arn:aws:iam::111122223333:role/eks-service-role-
AWSServiceRoleForAmazonEKS-AFNL4H8HB71F",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-a9189fe2",
                "subnet-50432629"
            ],
            "securityGroupIds": [
                "sg-f5c54184"
            ],
            "vpcId": "vpc-a54041dc"
        },
        "status": "CREATING",
        "certificateAuthority": {}
    }
}
```

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.

2. 클러스터 프로비저닝은 일반적으로 10분 미만이 걸립니다. 다음 명령을 사용하여 클러스터의 상태를 쿼리할 수 있습니다. 클러스터 상태가 `ACTIVE`인 경우 계속 진행할 수 있습니다.

```
aws eks --region region describe-cluster --name devel --query cluster.status
```

3. 클러스터 프로비저닝이 완료되면 다음 명령을 사용하여 `endpoint` 및 `certificateAuthority.data` 값을 검색합니다. 클러스터와 통신할 수 있도록 kubectl 구성에 추가되어야 합니다.

   a. `endpoint` 검색:

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

   b. `certificateAuthority.data` 검색:

   ```
   aws eks --region region describe-cluster --name devel  --query
    cluster.certificateAuthority.data --output text
   ```

4. 클러스터를 생성했으므로 이제 aws-iam-authenticator 설치 (p. 61) 및 Amazon EKS에 대한 kubeconfig 생성 (p. 62)의 절차를 따라 새 클러스터와의 통신을 활성화합니다.

Amazon EKS 클러스터 업데이트

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

새 Kubernetes 버전에는 중요한 변경 사항이 도입되므로, 프로덕션 클러스터에서 업데이트를 수행하기 전에 새 Kubernetes 버전에 대해 애플리케이션의 동작을 테스트하는 것이 좋습니다. 새 Kubernetes 버전으로 이동하기 전에 애플리케이션 동작을 종단 간으로 테스트하기 위한 지속적인 통합 워크플로우를 구축하여 이 작업을 수행할 수 있습니다.

Note

Amazon EKS는 가용성 높은 제어 플레인을 실행하지만, 업데이트 중에 심각하지 않은 서비스 중단이 발생할 수 있습니다. 예를 들면, API 서버를 종료하고 새 버전의 Kubernetes를 실행하는 새로운 API 서버로 교체하기 직전 또는 직후에 API 서버에 연결하려고 하면 API 호출 오류 또는 연결 문제가 발생할 수 있습니다. 이 문제가 발생하는 경우 성공할 때까지 API 작업을 재시도하십시오.

콘솔을 사용하여 기존 클러스터를 업데이트하려면

1. Open the Amazon EKS console at https://console.aws.amazon.com/eks/home#/clusters.
2. 업데이트할 클러스터의 이름을 선택하고 Update cluster version(클러스터 버전 업데이트)를 선택합니다.
3. Kubernetes version(Kubernetes 버전)에서 클러스터를 업데이트할 버전을 선택하고 Update(업데이트)를 선택합니다.
4. Cluster name(클러스터 이름)에 클러스터의 이름을 입력하고 Confirm(확인)을 선택합니다.

   Note

   클러스터 업데이트가 몇 분 내에 완료되어야 합니다.

5. `kube-proxy` 데몬 세트를 패치하여 현재 클러스터의 Kubernetes 버전(이 예에서는 *1.11.5*)에 해당하는 이미지를 사용합니다.

   ```
   kubectl patch daemonset kube-proxy \
   -n kube-system \
   -p '{"spec": {"template": {"spec": {"containers": [{"image": "602401143452.dkr.ecr.us-
   west-2.amazonaws.com/eks/kube-proxy:v1.11.5","name":"kube-proxy"}]}}}}'
   ```

6. Kubernetes 버전 1.10으로 생성된 클러스터는 `kube-dns`를 기본 DNS 및 서비스 검색 공급자로 사용하여 출고되었습니다. 1.10 클러스터를 1.11로 업데이트한 경우 CoreDNS를 DNS와 서비스 검색에 사용하

려면 CoreDNS를 설치하고 `kube-dns`를 제거해야 합니다. 자세한 내용은 CoreDNS 설치 (p. 46) 단원을 참조하십시오.

7. 클러스터 업데이트가 완료된 후 업데이트한 클러스터와 동일한 Kubernetes 버전으로 작업자 노드를 업데이트합니다. 자세한 내용은 작업자 노드 업데이트 (p. 31) 단원을 참조하십시오.

AWS CLI를 사용하여 기존 클러스터를 업데이트하려면

1. 다음 AWS CLI 명령을 사용하여 클러스터를 업데이트합니다. 클러스터 이름과 원하는 Kubernetes 마이너 버전을 대체합니다.

```
aws eks --region region update-cluster-version --name prod --kubernetes-version 1.11
```

결과:

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "InProgress",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.11"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
        "createdAt": 1544051347.305,
        "errors": []
    }
}
```

2. 이전 명령에서 반환된 클러스터 이름과 업데이트 ID를 사용하여 다음 명령으로 클러스터 업데이트의 상태를 모니터링합니다. 상태가 `Successful`로 표시되면 업데이트가 완료된 것입니다.

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

결과:

```
{
    "update": {
        "id": "b5f0ba18-9a87-4450-b5a0-825e6e84496f",
        "status": "Successful",
        "type": "VersionUpdate",
        "params": [
            {
                "type": "Version",
                "value": "1.11"
            },
            {
                "type": "PlatformVersion",
                "value": "eks.1"
            }
        ],
        "createdAt": 1544051347.305,
        "errors": []
```

```
    }
}
```

3. `kube-proxy` 데몬 세트를 패치하여 현재 클러스터의 Kubernetes 버전(이 예에서는 *1.11.5*)에 해당하는 이미지를 사용합니다.

```
kubectl patch daemonset kube-proxy \
-n kube-system \
-p '{"spec": {"template": {"spec": {"containers": [{"image": "602401143452.dkr.ecr.us-west-2.amazonaws.com/eks/kube-proxy:v1.11.5","name":"kube-proxy"}]}}}}'
```

4. Kubernetes 버전 1.10으로 생성된 클러스터는 `kube-dns`를 기본 DNS 및 서비스 검색 공급자로 사용하여 출고되었습니다. 1.10 클러스터를 1.11로 업데이트한 경우 CoreDNS를 DNS와 서비스 검색에 사용하려면 CoreDNS를 설치하고 `kube-dns`를 제거해야 합니다. 자세한 내용은 CoreDNS 설치 (p. 46) 단원을 참조하십시오.
5. 클러스터 업데이트가 완료된 후 업데이트한 클러스터와 동일한 Kubernetes 버전으로 작업자 노드를 업데이트합니다. 자세한 내용은 작업자 노드 업데이트 (p. 31) 단원을 참조하십시오.

클러스터 삭제

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

> **Important**
>
> 클러스터의 활성 서비스가 로드 밸런서와 연결된 경우 로드 밸런서가 적절하게 삭제되도록 클러스터 삭제 전에 해당 서비스를 삭제해야 합니다. 그렇지 않은 경우 VPC에 분리된 리소스가 발생하여 VPC를 삭제할 수 없습니다.

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 밸런서와 연결된 리소스가 적절하게 릴리스되어야 합니다.

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

3. 작업자 노드 AWS CloudFormation 스택 삭제:
   a. https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
   b. 삭제할 작업자 노드 스택을 선택하려면 Actions(작업), 스택 삭제를 선택합니다.
   c. 스택 삭제 확인 화면에서 Yes, Delete(예, 삭제합니다)를 선택합니다.
4. 클러스터를 삭제하십시오:
   a. Open the Amazon EKS console at https://console.aws.amazon.com/eks/home#/clusters.
   b. 삭제할 클러스터를 선택하고 삭제를 선택합니다.
   c. 클러스터 삭제 확인 화면에서 삭제를 선택합니다.
5. (선택 사항) VPC AWS CloudFormation 스택 삭제:
   a. 삭제할 VPC 스택을 선택하고 Actions(작업), 스택 삭제를 선택합니다.
   b. 스택 삭제 확인 화면에서 Yes, Delete(예, 삭제합니다)를 선택합니다.

플랫폼 버전

새로운 클러스터에 Kubernetes 집계 계층 릴리스 지원을 시작하면서, Amazon EKS에서는 플랫폼 버전 개념을 도입하여 클러스터에서 현재 활성화되어 있는 기능을 식별할 수 있게 되었습니다. Amazon EKS 플랫폼 버전은 현재 Kubernetes 패치 버전뿐만 아니라 활성화된 Kubernetes API 서버 플래그 등, 클러스터 제어 플레인의 기능을 나타냅니다.

새로운 클러스터는 지정된 Kubernetes 버전에 사용 가능한 최신 플랫폼 버전으로 생성됩니다. 새로운 플랫폼 버전을 Kubernetes 버전에 사용할 수 있게 되면 기존 클러스터는 점진적 롤아웃을 통해 해당 Kubernetes 버전용 최신 플랫폼 버전으로 자동으로 업데이트됩니다. 최신 플랫폼 버전 기능을 즉시 활용하기 위해 새로운 Amazon EKS 클러스터를 만들 수 있습니다.

현재 및 최근 Amazon EKS 플랫폼 버전은 다음 표에 설명되어 있습니다.

Kubernetes 1.11 플랫폼 버전

Kubernetes 버전	Kubernetes 패치 버전	Amazon EKS 플랫폼 버전	활성화된 승인 컨트롤러	릴리스 정보
1.11	1.11.5	eks.1	Initializers, NamespaceLifecycle, LimitRanger, ServiceAccount, DefaultStorageClass, ResourceQuota, DefaultTolerationSeconds, NodeRestriction, MutatingAdmissionWebhook, ValidatingAdmissionWebhook	Amazon EKS용 Kubernetes 1.11의 최초 릴리스입니다.

Kubernetes 1.10 플랫폼 버전

Kubernetes 버전	Kubernetes 패치 버전	Amazon EKS 플랫폼 버전	활성화된 승인 컨트롤러	릴리스 정보
1.10	1.10.11	eks.3	Initializers, NamespaceLifecycle, LimitRanger, ServiceAccount, DefaultStorageClass, ResourceQuota, DefaultTolerationSeconds, NodeRestriction, MutatingAdmissionWebhook, ValidatingAdmissionWebhook	CVE-2018-1002105를 해결하기 위해 Kubernet을 패치 수준 1.10.11로 업데이트하는 새로운 플랫폼 버전
1.10	1.10.3	eks.2	Initializers, NamespaceLifecycle, LimitRanger, ServiceAccount, DefaultStorageClass, ResourceQuota, DefaultTolerationSeconds,	• Kubernetes 집계 계층에 대한 지원을 추가했습니다. • Kubernetes Horizontal Pod Autoscaler(HPA)

Kubernetes 버전	Kubernetes 패치 버전	Amazon EKS 플랫폼 버전	활성화된 승인 컨트롤러	릴리스 정보
			`NodeRestriction`, `MutatingAdmissionWebhook`, `ValidatingAdmissionWebhook`	에 대한 지원을 추가했습니다. • Kubernetes Metrics Server 0.3.0 이상은 EKS 플랫폼 버전 `eks.2`와 호환됩니다.
`1.10`	`1.10.3`	`eks.1`	`Initializers`, `NamespaceLifecycle`, `LimitRanger`, `ServiceAccount`, `DefaultStorageClass`, `ResourceQuota`, `DefaultTolerationSeconds`, `NodeRestriction`	Amazon EKS의 초기 시작

작업자 노드

Kubernetes의 작업자 머신을 노드라고 합니다. Amazon EKS 작업자 노드는 AWS 계정에서 실행되고, 클러스터 API 서버 엔드포인트를 통해 제어 플레이에 연결합니다.

Amazon EKS 작업자 노드는 표준 Amazon EC2 인스턴스이고, 일반 EC2 온디맨드 가격을 기반으로 비용이 청구됩니다. 자세한 내용은 Amazon EC2 요금을 참조하십시오.

기본적으로 Amazon EKS는 Amazon EKS 클러스터에서 작업자 노드를 가동하는 AWS CloudFormation 템플릿을 제공합니다. 이 AMI는 Amazon Linux 2를 기반으로 하고, Amazon EKS 작업자 노드에 대한 기본 이미지 역할을 하도록 구성됩니다. AMI는 Amazon EKS를 즉시 사용하도록 구성되며, 여기에는 도커, kubelet, AWS IAM Authenticator가 포함됩니다. AMI는 또한 특수 부트스트랩 스크립트도 포함하여 클러스터의 제어 플레인을 자동으로 찾고 연결합니다.

> **Note**
>
> Amazon Linux 보안 센터에서 Amazon Linux 2에 대한 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.

또한 AWS CloudFormation 작업자 노드 템플릿은 특수 부트스트랩 스크립트를 트리거하는 특수 Amazon EC2 사용자 데이터와 함께 작업자 노드를 시작하여 클러스터의 제어 플레인을 자동으로 찾고 연결합니다. 자세한 내용은 Amazon EKS 작업자 노드 시작 (p. 28) 단원을 참조하십시오.

일반 Kubernetes 관점에서 작업자 노드에 대한 자세한 내용은 Kubernetes 문서의 Nodes를 참조하십시오.

항목

- Amazon EKS 최적화 AMI (p. 24)
- Amazon EKS 파트너 AMI (p. 27)
- Amazon EKS 작업자 노드 시작 (p. 28)
- 작업자 노드 업데이트 (p. 31)

Amazon EKS 최적화 AMI

Amazon EKS 최적화 AMI는 Amazon Linux 2를 기반으로 하고, Amazon EKS 작업자 노드에 대한 기본 이미지 역할을 하도록 구성됩니다. AMI는 Amazon EKS를 즉시 사용하도록 구성되며, 여기에는 도커, kubelet, AWS IAM Authenticator가 포함됩니다.

> **Note**
>
> Amazon Linux 보안 센터에서 Amazon Linux 2에 대한 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.

최신 Amazon EKS 최적화 AMI(GPU 지원 (p. 25) 유무는 상이)의 AMI ID는 다음 표에 나와 있습니다.

> **Note**
>
> GPU 지원이 포함된 Amazon EKS 최적화 AMI는 P2 및 P3 인스턴스 유형만 지원합니다. 작업자 노드 AWS CloudFormation 템플릿에서 이러한 인스턴스 유형을 지정해야 합니다. 이 AMI에는 최종 사용자 라이선스 계약(EULA)이 필요한 타사 소프트웨어가 포함되므로 작업자 노드 그룹에서 AMI를 사용하려면 먼저 AWS Marketplace에서 AMI에 가입하고 EULA에 동의해야 합니다. AMI에 가입하려면 AWS Marketplace를 방문하십시오.

리전	Amazon EKS 최적화 AMI	(GPU 지원 포함)
미국 서부(오리건) (us-west-2)	ami-0f54a2f7d2e9c88b3	ami-08156e8fd65879a13
미국 동부(버지니아 북부) (us-east-1)	ami-0a0b913ef3249b655	ami-0c974dde3f6d691a1
미국 동부(오하이오)(us-east-2)	ami-0958a76db2d150238	ami-089849e811ace242f
EU(아일랜드)(eu-west-1)	ami-00c3b2d35bddd4f5c	ami-0c3479bcd739094f0

Important

이러한 AMI에는 최신 AWS CloudFormation 작업자 노드 템플릿이 필요합니다. 이전 버전의 작업자 노드 템플릿으로는 이러한 AMI를 사용할 수 없으며, AMI가 클러스터에 조인하는 데 실패합니다. 이러한 AMI를 사용하기 전에 반드시 최신 템플릿(아래 표시된 URL)으로 기존 AWS CloudFormation 작업자 스택을 업그레이드해야 합니다.

```
https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2018-11-07/amazon-eks-nodegroup.yaml
```

또한 AWS CloudFormation 작업자 노드 템플릿은 특수 부트스트랩 스크립트를 트리거하는 Amazon EC2 사용자 데이터와 함께 작업자 노드를 시작하여 클러스터의 제어 플레인을 자동으로 찾고 연결합니다. 자세한 내용은 Amazon EKS 작업자 노드 시작 (p. 28) 단원을 참조하십시오.

Amazon EKS 최적화 AMI 빌드 스크립트

Amazon Elastic Container Service for Kubernetes(Amazon EKS)는 Amazon EKS 최적화 AMI를 빌드하는 데 사용되는 빌드 스크립트를 오픈 소스로 제공합니다. 이러한 빌드 스크립트를 이제 GitHub에서 사용할 수 있습니다.

Amazon EKS 최적화 AMI는 Amazon Linux 2를 기반으로, 특히 Amazon EKS 클러스터에서 작업자 노드로 사용하기 위해 빌드됩니다. 이 리포지토리를 사용해 Amazon EKS 팀이 kubelet, Docker, Kubernetes용 AWS IAM Authenticator 등을 구성하는 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 EKS 팀이 공식 AMI에 사용하는 것과 동일한 버전의 Docker를 자체 AMI에 사용하기를 원할 수 있습니다.

또한 GitHub 리포지토리는 부팅 시 실행되어 인스턴스의 인증 데이터, 제어 플레인 엔드포인트, 클러스터 이름 등을 구성하는 특수 부트스트랩 스크립트도 포함합니다.

또한 GitHub 리포지토리는 Amazon EKS 작업자 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 참조하십시오.

GPU 지원이 포함된 Amazon EKS 최적화 AMI

GPU를 지원하는 Amazon EKS 최적화 AMI는 표준 Amazon EKS 최적화 AMI 기반으로 구축되며, GPU 워크로드를 지원하는 Amazon EKS 작업자 노드의 선택적 이미지로 사용하도록 구성됩니다.

표준 Amazon EKS 최적화 AMI 구성 외에도 GPU AMI에는 다음이 포함됩니다.

- NVIDIA 드라이버
- `nvidia-docker2` 패키지
- `nvidia-container-runtime`(기본 실행 시간으로)

GPU가 지원되는 최신 Amazon EKS 최적화 AMI의 AMI ID는 다음 표에 나와 있습니다.

> **Note**
>
> GPU 지원이 포함된 Amazon EKS 최적화 AMI는 P2 및 P3 인스턴스 유형만 지원합니다. 작업자 노드 AWS CloudFormation 템플릿에서 이러한 인스턴스 유형을 지정해야 합니다. 이 AMI에는 최종 사용자 라이선스 계약(EULA)이 필요한 타사 소프트웨어가 포함되므로 작업자 노드 그룹에서 AMI를 사용하려면 먼저 AWS Marketplace에서 AMI에 가입하고 EULA에 동의해야 합니다. AMI에 가입하려면 AWS Marketplace를 방문하십시오.

리전	GPU 지원이 포함된 Amazon EKS 최적화 AMI
미국 서부(오리건) (`us-west-2`)	`ami-08156e8fd65879a13`
미국 동부(버지니아 북부) (`us-east-1`)	`ami-0c974dde3f6d691a1`
미국 동부(오하이오)(`us-east-2`)	`ami-089849e811ace242f`
EU(아일랜드)(`eu-west-1`)	`ami-0c3479bcd739094f0`

> **Important**
>
> 이러한 AMI에는 최신 AWS CloudFormation 작업자 노드 템플릿이 필요합니다. 이전 버전의 작업자 노드 템플릿으로는 이러한 AMI를 사용할 수 없으며, AMI가 클러스터에 조인하는 데 실패합니다. 이러한 AMI를 사용하기 전에 반드시 최신 템플릿(아래 표시된 URL)으로 기존 AWS CloudFormation 작업자 스택을 업그레이드해야 합니다.

```
https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2018-11-07/amazon-eks-nodegroup.yaml
```

또한 AWS CloudFormation 작업자 노드 템플릿은 특수 부트스트랩 스크립트를 트리거하는 Amazon EC2 사용자 데이터와 함께 작업자 노드를 시작하여 클러스터의 제어 플레인을 자동으로 찾고 연결합니다. 자세한 내용은 Amazon EKS 작업자 노드 시작 (p. 28) 단원을 참조하십시오.

GPU 작업자 노드가 클러스터에 조인한 후, 다음 명령을 이용해 클러스터에 Kubernetes용 NVIDIA 디바이스 플러그인을 데몬 세트로 적용해야 합니다.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v1.10/nvidia-device-plugin.yml
```

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 매니페스트 예

이 섹션에서는 GPU 작업자가 제대로 구성되었는지 알아볼 수 있는 포드 매니페스트 예를 제공합니다.

Example **`nvidia-smi`** 출력 가져오기

Example

이 포드 매니페스트 예는 작업자 노드에서 `nvidia-smi`를 실행하는 Cuda 컨테이너를 시작합니다. `nvidia-smi.yaml`이라는 파일을 만들고, 다음 매니페스트를 복사하여 붙여 넣은 후 파일을 저장합니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: nvidia-smi
spec:
  restartPolicy: OnFailure
  containers:
  - name: nvidia-smi
    image: nvidia/cuda:9.2-devel
    args:
    - "nvidia-smi"
    resources:
      limits:
        nvidia.com/gpu: 1
```

다음 명령으로 매니페스트를 적용합니다.

```
kubectl apply -f nvidia-smi.yaml
```

포드 실행이 끝난 후, 다음 명령을 사용하여 로그를 확인합니다.

```
kubectl logs nvidia-smi
```

결과:

```
Mon Aug  6 20:23:31 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 396.26                 Driver Version: 396.26                    |
|-------------------------------+----------------------+----------------------+
| 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 파트너 AMI

공식 Amazon EKS 최적화 외에도, Canonical은 Amazon EKS와 협력하여 클러스터에서 사용할 수 있는 작업자 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 내

용은 Ubuntu 및 Amazon Elastic Container Service for Kubernetes(EKS) 및 Ubuntu 18.04에서 최적화된 Amazon EKS 지원을 참조하십시오.

Amazon EKS 작업자 노드 시작

이 주제는 Amazon EKS 클러스터에 등록하는 작업자 노드의 Auto Scaling 그룹을 시작하는 데 도움이 됩니다. 노드가 클러스터에 조인한 이후 Kubernetes 애플리케이션을 배포할 수 있습니다.

처음으로 Amazon EKS 작업자 노드를 시작하는 경우 Amazon EKS 시작하기 (p. 3) 가이드를 따르는 것이 좋습니다. 이 가이드는 Amazon EKS 클러스터 생성부터 샘플 Kubernetes 애플리케이션 배포에 이르기까지 종합적인 연습을 제공합니다.

> **Important**
>
> Amazon EKS 작업자 노드는 표준 Amazon EC2 인스턴스이고, 일반 Amazon EC2 온디맨드 인스턴스 가격을 기반으로 비용이 청구됩니다. 자세한 내용은 Amazon EC2 요금 단원을 참조하십시오.

이 주제의 사전 요구 사항은 다음과 같습니다.

- Amazon EKS 클러스터의 요건을 충족하는 VPC 및 보안 그룹을 생성했습니다. 자세한 내용은 클러스터 VPC 고려 사항 (p. 40) 및 클러스터 보안 그룹 고려 사항 (p. 41) 단원을 참조하십시오. Amazon EKS 시작하기 (p. 3) 가이드는 요건을 충족하는 VPC를 생성합니다. 또는 자습서: Amazon EKS 클러스터에 대해 퍼블릭 및 프라이빗 서브넷이 있는 VPC 생성 (p. 81) 가이드를 따라 수동으로 생성할 수 있습니다.
- Amazon EKS 클러스터를 생성하고 위 VPC 및 보안 그룹에 사용한다고 지정했습니다. 자세한 내용은 Amazon EKS 클러스터 생성 (p. 16) 단원을 참조하십시오.

작업자 노드를 시작하려면

1. 클러스터 상태가 `ACTIVE`가 되기를 기다립니다. 클러스터가 활성화되기 전에 작업자 노드를 시작하면 작업자 노드가 클러스터에 등록되지 않고 작업자 노드를 다시 시작해야 합니다.
2. https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
3. 탐색 모음에서 Amazon EKS를 지원하는 리전을 선택합니다.

   > **Note**
   >
   > 현재 다음 리전에서 Amazon EKS를 사용할 수 있습니다.
   >
   > - 미국 서부(오리건) (`us-west-2`)
   > - 미국 동부(버지니아 북부) (`us-east-1`)
   > - 미국 동부(오하이오)(`us-east-2`)
   > - EU(아일랜드) (`eu-west-1`)

4. [Create stack]을 선택합니다.
5. 템플릿 선택에서 Amazon S3 템플릿 URL 지정을 선택합니다.
6. 텍스트 영역에 다음 URL을 붙여넣고 다음을 선택합니다.

   ```
   https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2018-11-07/amazon-eks-nodegroup.yaml
   ```

7. Specify Details(세부 정보 지정) 페이지에서 다음 파라미터를 입력하고 다음을 선택합니다.
   - 스택 이름: AWS CloudFormation 스택에 대한 스택 이름을 선택합니다. 예를 들어 ***<cluster-name>***-worker-nodes로 사용할 수 있습니다.
   - ClusterName: Amazon EKS 클러스터 생성 시 사용할 이름을 입력합니다.

Important

이 이름은 Amazon EKS 클러스터 이름과 정확하게 일치해야 합니다. 그렇지 않은 경우 작업자 노드가 조인할 수 없습니다.

- ClusterControlPlaneSecurityGroup: Amazon EKS 클러스터를 생성할 때 사용한 보안 그룹 또는 그룹을 입력합니다. 이 AWS CloudFormation 템플릿은 지정된 클러스터 제어 플레인 보안 그룹의 트래픽을 허용하는 작업자 노드 보안 그룹을 생성합니다.
- NodeGroupName: 노드 그룹의 이름을 입력합니다. 이 이름은 나중에 작업자 노드에 대해 생성된 Auto Scaling 노드 그룹을 식별하는 데 사용할 수 있습니다.
- NodeAutoScalingGroupMinSize: Auto Scaling 그룹이 축소할 수 있는 작업자 노드의 최소 노드 수를 입력합니다.
- NodeAutoScalingGroupDesiredCapacity: 스택을 생성할 때 조정할 원하는 노드 수를 입력합니다.
- NodeAutoScalingGroupMaxSize: Auto Scaling 그룹이 확장할 수 있는 작업자 노드의 최대 노드 수를 입력합니다. 이 값은 원하는 용량보다 최소한 노드 1개 이상 더 커야 합니다. 그러면 업데이트 중에 노드 수를 줄이지 않고 작업자 노드의 롤링 업데이트를 수행할 수 있습니다.
- NodeInstanceType: 작업자 노드에 대한 인스턴스 유형을 선택합니다.
- NodeImageId: 리전에 대한 현재 Amazon EKS 작업자 노드 AMI ID를 입력합니다. 최신 Amazon EKS 최적화 AMI(GPU 지원 (p. 25) 유무는 상이)의 AMI ID는 다음 표에 나와 있습니다. 원하는 Kubernetes 버전과 AWS 리전에 대한 올바른 AMI ID를 선택해야 합니다.

  Note

  GPU 지원이 포함된 Amazon EKS 최적화 AMI는 P2 및 P3 인스턴스 유형만 지원합니다. 작업자 노드 AWS CloudFormation 템플릿에서 이러한 인스턴스 유형을 지정해야 합니다. 이 AMI에는 최종 사용자 라이선스 계약(EULA)이 필요한 타사 소프트웨어가 포함되므로 작업자 노드 그룹에서 AMI를 사용하려면 먼저 AWS Marketplace에서 AMI에 가입하고 EULA에 동의해야 합니다. AMI에 가입하려면 AWS Marketplace를 방문하십시오.

리전	Amazon EKS 최적화 AMI	(GPU 지원 포함)
미국 서부(오리건) (us-west-2)	ami-0f54a2f7d2e9c88b3	ami-08156e8fd65879a13
미국 동부(버지니아 북부) (us-east-1)	ami-0a0b913ef3249b655	ami-0c974dde3f6d691a1
미국 동부(오하이오)(us-east-2)	ami-0958a76db2d150238	ami-089849e811ace242f
EU(아일랜드)(eu-west-1)	ami-00c3b2d35bddd4f5c	ami-0c3479bcd739094f0

  Note

  Amazon EKS 작업자 노드 AMI는 Amazon Linux 2 기반입니다. Amazon Linux 보안 센터에서 Amazon Linux 2에 대한 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.

- KeyName: 시작 이후 SSH를 사용하여 작업자 노드에 연결하는 데 사용할 수 있는 Amazon EC2 SSH 키 페어 이름을 입력합니다. Amazon EC2 키 페어가 아직 없는 경우 AWS Management 콘솔에서 새로 생성할 수 있습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 Amazon EC2 키 페어를 참조하십시오.

  Note

  여기에 키 페어를 입력하지 않으면 AWS CloudFormation 스택이 생성되지 않습니다.

- BootstrapArguments: 별도의 kubelet 인수와 같이 작업자 노드 부트스트랩 스크립트에 전달할 선택적 인수를 지정합니다. 자세한 내용은 https://github.com/awslabs/amazon-eks-ami/blob/master/files/bootstrap.sh에서 부트스트랩 스크립트 사용 정보를 참조하십시오.
- VpcId: 작업자 노드에서 시작해야 하는 VPC의 ID를 입력합니다.
- Subnets: 작업자 노드에서 시작해야 하는 위 VPC 내 서브넷을 선택합니다.

8. 옵션 페이지에서 스택 리소스에 태그를 지정할 수 있습니다. [Next]를 선택합니다.
9. 검토 페이지에서 정보를 검토하고, 스택이 IAM 리소스를 생성할 수 있음을 인지한 다음 생성을 선택합니다.
10. 스택이 생성된 후 콘솔에서 이를 선택하고 출력을 선택합니다.
11. 생성된 노드 그룹에 대해 NodeInstanceRole을 기록합니다. Amazon EKS 작업자 노드를 구성할 때 필요합니다.

작업자 노드가 클러스터에 조인하도록 하려면

1. AWS IAM Authenticator 구성 맵을 다운로드, 편집 및 적용합니다.

   a. 구성 맵 다운로드:

   ```
   curl -O https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2018-11-07/aws-auth-cm.yaml
   ```

   b. 즐겨찾는 텍스트 편집기에서 파일을 엽니다. 인스턴스 역할의 *<ARN(##### ####)>* 조각을 이전 절차에서 기록한 NodeInstanceRole 값으로 교체하고 파일을 저장합니다.

   > **Important**
   >
   > 이 파일에서 어떠한 행도 수정하지 마십시오.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: aws-auth
     namespace: kube-system
   data:
     mapRoles: |
       - rolearn: <ARN of instance role (not instance profile)>
         username: system:node:{{EC2PrivateDNSName}}
         groups:
           - system:bootstrappers
           - system:nodes
   ```

   c. 구성을 적용합니다. 이 명령을 완료하는 데 몇 분이 걸릴 수 있습니다.

   ```
   kubectl apply -f aws-auth-cm.yaml
   ```

   > **Note**
   >
   > `"aws-iam-authenticator": executable file not found in $PATH` 오류가 표시되는 경우 Amazon EKS에 대해 kubectl이 구성되지 않았습니다. 자세한 내용은 ??? 단원을 참조하십시오.

2. 노드의 상태를 확인하고 `Ready` 상태가 될 때까지 대기합니다.

   ```
   kubectl get nodes --watch
   ```

3. (GPU 작업자만 해당) P2 또는 P3 인스턴스 유형과 GPU 지원이 포함된 Amazon EKS 최적화 AMI를 선택한 경우, 다음 명령을 이용해 클러스터에 Kubernetes용 NVIDIA 디바이스 플러그인을 데몬 세트로 적용해야 합니다.

```
kubectl apply -f https://raw.githubusercontent.com/NVIDIA/k8s-device-plugin/v1.10/nvidia-device-plugin.yml
```

작업자 노드 업데이트

새로운 Amazon EKS 최적화 AMI가 릴리스되면 작업자 노드 그룹의 노드를 새 AMI로 교체하는 것을 고려해야 합니다. 마찬가지로, Amazon EKS 클러스터용 Kubernetes 버전을 업데이트한 경우 동일한 Kubernetes 버전이 있는 작업자 노드를 사용하도록 작업자 노드도 업데이트해야 합니다.

새 AMI를 사용하도록 클러스터의 작업자 노드를 업데이트하는 두 가지 기본 방법이 있습니다. 새 작업자 노드 그룹을 생성하고 포드를 해당 그룹으로 마이그레이션하거나, 새 AMI를 사용하도록 기존 작업자 노드 그룹용 AWS CloudFormation 스택을 업데이트하는 것입니다. 새 작업자 노드 그룹으로 마이그레이션하는 방법은 단순히 기존 AWS CloudFormation 스택의 AMI ID를 업데이트하는 방법보다 더 안정적입니다. 새 작업자 노드 그룹으로 마이그레이션하는 방법은 이전 노드 그룹을 `NoSchedule`로 테인트한 다음 새 스택이 기존 포드 워크로드를 수락할 준비가 되면 노드를 드레이닝하기 때문입니다.

항목

- 새 작업자 노드 그룹으로 마이그레이션 (p. 31)
- 기존 작업자 노드 그룹 업데이트 (p. 35)

새 작업자 노드 그룹으로 마이그레이션

이 주제에서는 새로운 작업자 노드 그룹을 생성하고, 기존 애플리케이션을 새 그룹으로 안정적으로 마이그레이션한 다음, 이전 작업자 노드 그룹을 클러스터에서 제거하는 방법을 설명합니다.

애플리케이션을 새 작업자 노드 그룹으로 마이그레이션하려면

1. Amazon EKS 작업자 노드 시작 (p. 28) 단원에 설명된 단계에 따라 새 작업자 노드 그룹을 시작합니다.
2. 스택이 생성된 후 콘솔에서 이를 선택하고 출력을 선택합니다.
3. 생성된 노드 그룹에 대해 NodeInstanceRole을 기록합니다. 새 Amazon EKS 작업자 노드를 클러스터에 추가하려면 이 정보가 필요합니다.

   Note

   추가 IAM 정책을 이전 노드 그룹 IAM 역할에 연결한 경우(예를 들면 Kubernetes Cluster Autoscaler에 대한 권한을 추가하기 위해), 새 그룹에서 해당 기능을 유지하려면 동일한 정책을 새 노드 그룹 IAM 역할에 연결해야 합니다.
4. 두 작업자 그룹 노드가 서로 통신할 수 있도록 두 작업자 노드 그룹에 대한 보안 그룹을 모두 업데이트합니다. 자세한 내용은 클러스터 보안 그룹 고려 사항 (p. 41) 단원을 참조하십시오.

   a. 두 작업자 노드 그룹에 대한 보안 그룹 ID를 모두 기록합니다. 이 정보는 AWS CloudFormation 스택 출력에 NodeSecurityGroup 값으로 표시됩니다.

      다음 AWS CLI 명령을 사용하여 스택 이름에서 보안 그룹 ID를 가져옵니다. 이러한 명령에서 `oldNodes`는 이전 작업자 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)
```

b. 각 작업자 노드 보안 그룹이 서로의 트래픽을 수락하도록 각 작업자 노드 보안 그룹에 수신 규칙을 추가합니다.

   다음 AWS CLI 명령은 상대 보안 그룹의 모든 프로토콜에서 모든 트래픽을 허용하는 수신 규칙을 각 보안 그룹에 추가합니다. 이렇게 하면 워크로드를 새 그룹으로 마이그레이션하는 동안 각 작업자 노드 그룹의 포드가 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 작업자 노드 인스턴스 역할을 매핑합니다.

```
kubectl edit configmap -n kube-system aws-auth
```

   새 작업자 노드 그룹에 대한 새 `mapRoles` 항목을 추가합니다.

```
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/workers-1-10-NodeInstanceRole-
U11V27W93CX5
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

   <#### ##(#### ##### ##)# ARN> 조각을 Step 3 (p. 31)에서 기록한 NodeInstanceRole 값으로 바꾼 다음, 파일을 저장하고 닫아서 업데이트된 configmap을 적용합니다.

6. 노드의 상태를 보면서 새 작업자 노드가 클러스터에 조인하고 `Ready` 상태에 도달할 때까지 기다립니다.

```
kubectl get nodes --watch
```

7. (선택 사항) Kubernetes Cluster Autoscaler를 사용하는 경우 배포를 0개의 복제본으로 축소하여 조정 작업의 충돌을 방지합니다.

```
kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
```

8. 다음 명령을 사용하여 제거하려는 각 노드를 `NoSchedule`로 테인트합니다(그러면 교체하는 노드에서 새 포드가 예약되거나 다시 예약되지 않음).

```
kubectl taint nodes node_name key=value:NoSchedule
```

작업자 노드를 새 Kubernetes 버전으로 업그레이드하는 경우 다음 코드 조각을 사용하여 특정 Kubernetes 버전(이 경우 1.10.3)의 모든 노드를 식별하고 테인트합니다.

```
K8S_VERSION=1.10.3
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
```

출력(이 클러스터는 DNS 확인에 `kube-dns`를 사용하고 있지만, 클러스터는 그 대신 `coredns`를 반환할 수 있음):

```
NAME       DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kube-dns   1         1         1            1           31m
```

10. 현재 배포가 2개 미만의 복제본을 실행하고 있는 경우 배포를 2개의 복제본으로 확장합니다. `kube-dns`를 `coredns`로 대체합니다(이전 명령 출력에서 해당 값이 대신 반환된 경우).

```
kubectl scale deployments/kube-dns --replicas=2 -n kube-system
```

11. 다음 명령을 사용하여 클러스터에서 제거할 각 노드를 드레이닝합니다.

```
kubectl drain node_name --ignore-daemonsets --delete-local-data
```

작업자 노드를 새 Kubernetes 버전으로 업그레이드하는 경우 다음 코드 조각을 사용하여 특정 Kubernetes 버전(이 경우 1.10.3)의 모든 노드를 식별하고 드레이닝합니다.

```
K8S_VERSION=1.10.3
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 작업자 노드가 드레이닝을 완료한 후 위에서 권한 부여한 보안 그룹 수신 규칙을 취소하고 AWS CloudFormation 스택을 삭제하여 인스턴스를 종료합니다.

    Note

    추가 IAM 정책을 기존 노드 그룹 IAM 역할에 연결한 경우(예를 들면 Kubernetes Cluster Autoscaler에 대한 권한을 추가하기 위해), AWS CloudFormation 스택을 삭제하려면 먼저 해당 추가 정책을 역할에서 분리해야 합니다.

    a. 위에서 작업자 노드 보안 그룹에 대해 생성한 수신 규칙을 취소합니다. 이러한 명령에서 `oldNodes`는 이전 작업자 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. https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.

c. 이전 작업자 노드 스택을 선택합니다.

d. Actions(작업)를 선택한 다음 Delete stack(스택 삭제)을 선택합니다.

13. `aws-auth` configmap을 편집하여 RBAC에서 이전 작업자 노드 인스턴스 역할을 제거합니다.

```
kubectl edit configmap -n kube-system aws-auth
```

이전 작업자 노드 그룹에 대한 `mapRoles` 항목을 삭제합니다.

```
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::111122223333:role/workers-1-11-NodeInstanceRole-
W70725MZQFF8
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
    - rolearn: arn:aws:iam::111122223333:role/workers-1-10-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 1개의 복제본으로 축소합니다.

    Note

    또한 새 Auto Scaling 그룹을 적절하게 태그 지정하고(예: `k8s.io/cluster-autoscaler/enabled,k8s.io/cluster-autoscaler/<YOUR CLUSTER NAME>`) 새로 태그 지정된 Auto Scaling 그룹을 가리키도록 Cluster Autoscaler 배포를 업데이트해야 합니다. 자세한 내용은 AWS의 Cluster Autoscaler를 참조하십시오.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

15. 클러스터가 DNS 확인에 `kube-dns`를 사용하는 경우(Step 9 (p. 33) 단계 참조) `kube-dns` 배포를 1개의 복제본으로 축소합니다.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

기존 작업자 노드 그룹 업데이트

이 주제에서는 기존 AWS CloudFormation 작업자 노드 스택을 새 AMI로 업데이트하는 방법을 설명합니다. 이 절차를 사용하여 클러스터 업데이트 이후 작업자 노드를 새 버전의 Kubernetes로 업데이트하거나, 기존 Kubernetes 버전용 최신 Amazon EKS 최적화 AMI로 업데이트할 수 있습니다.

최신 Amazon EKS 작업자 노드 AWS CloudFormation 템플릿은 클러스터에서 한 번에 하나씩 새 AMI가 있는 인스턴스를 시작한 후 이전 인스턴스를 제거하도록 구성되므로, 롤링 업데이트 중 클러스터에 항상 원하는 수의 Auto Scaling 그룹 활성 인스턴스가 있습니다.

기존 작업자 노드 그룹을 업데이트하려면

1. 클러스터의 DNS 공급자를 결정합니다.

   ```
   kubectl get deployments -l k8s-app=kube-dns -n kube-system
   ```

   출력(이 클러스터는 DNS 확인에 `kube-dns`를 사용하고 있지만, 클러스터는 그 대신 `coredns`를 반환할 수 있음):

   ```
   NAME       DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
   kube-dns   1         1         1            1           31m
   ```

2. 현재 배포가 2개 미만의 복제본을 실행하고 있는 경우 배포를 2개의 복제본으로 확장합니다. `kube-dns`를 `coredns`로 대체합니다(이전 명령 출력에서 해당 값이 대신 반환된 경우).

   ```
   kubectl scale deployments/kube-dns --replicas=2 -n kube-system
   ```

3. (선택 사항) Kubernetes Cluster Autoscaler를 사용하는 경우 배포를 0개의 복제본으로 축소하여 조정 작업의 충돌을 방지합니다.

   ```
   kubectl scale deployments/cluster-autoscaler --replicas=0 -n kube-system
   ```

4. 현재 작업자 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 Launch Configurations(시작 구성)를 선택하고 기존 작업자 노드 시작 구성에 대한 인스턴스 유형을 기록해 둡니다.

   c. 왼쪽 탐색 창에서 Auto Scaling Groups(Auto Scaling 그룹)를 선택하고 기존 작업자 노드 Auto Scaling 그룹에 대해 Desired(원하는) 인스턴스 수를 기록해 둡니다.

5. https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
6. 작업자 노드 그룹 스택을 선택한 다음 Actions(작업), Update stack(스택 업데이트)을 선택합니다.
7. 템플릿 선택에서 Amazon S3 템플릿 URL 지정을 선택합니다.
8. 다음 URL을 텍스트 영역에 붙여 넣고(최신 버전의 작업자 노드 AWS CloudFormation 템플릿을 사용하고 있는지 확인하기 위해) Next(다음)를 선택합니다.

   ```
   https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2018-11-07/amazon-eks-nodegroup.yaml
   ```

9. Specify Details(세부 정보 지정) 페이지에서 다음 파라미터를 입력하고 다음을 선택합니다.

   - NodeAutoScalingGroupDesiredCapacity: Step 4 (p. 35)에서 기록한 원하는 인스턴스 수를 입력하거나, 스택을 업데이트할 때 조정할 원하는 새 노드 수를 입력합니다.

- NodeAutoScalingGroupMaxSize: Auto Scaling 그룹이 확장할 수 있는 작업자 노드의 최대 노드 수를 입력합니다. 이 값은 원하는 용량보다 최소한 노드 1개 이상 더 커야 합니다. 그러면 업데이트 중에 노드 수를 줄이지 않고 작업자 노드의 롤링 업데이트를 수행할 수 있습니다.
- NodeInstanceType: Step 4 (p. 35)에서 기록한 인스턴스 유형을 선택하거나, 작업자 노드에 대해 다른 인스턴스 유형을 선택합니다.
- NodeImageId: 리전에 대한 현재 Amazon EKS 작업자 노드 AMI ID를 입력합니다. 최신 Amazon EKS 최적화 AMI(GPU 지원 (p. 25) 유무는 상이)의 AMI ID는 다음 표에 나와 있습니다.

  Note

  GPU 지원이 포함된 Amazon EKS 최적화 AMI는 P2 및 P3 인스턴스 유형만 지원합니다. 작업자 노드 AWS CloudFormation 템플릿에서 이러한 인스턴스 유형을 지정해야 합니다. 이 AMI에는 최종 사용자 라이선스 계약(EULA)이 필요한 타사 소프트웨어가 포함되므로 작업자 노드 그룹에서 AMI를 사용하려면 먼저 AWS Marketplace에서 AMI에 가입하고 EULA에 동의해야 합니다. AMI에 가입하려면 AWS Marketplace를 방문하십시오.

리전	Amazon EKS 최적화 AMI	(GPU 지원 포함)
미국 서부(오리건) (us-west-2)	ami-0f54a2f7d2e9c88b3	ami-08156e8fd65879a13
미국 동부(버지니아 북부) (us-east-1)	ami-0a0b913ef3249b655	ami-0c974dde3f6d691a1
미국 동부(오하이오)(us-east-2)	ami-0958a76db2d150238	ami-089849e811ace242f
EU(아일랜드)(eu-west-1)	ami-00c3b2d35bddd4f5c	ami-0c3479bcd739094f0

  Note

  Amazon EKS 작업자 노드 AMI는 Amazon Linux 2 기반입니다. Amazon Linux 보안 센터에서 Amazon Linux 2에 대한 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.

10. (선택 사항) 옵션 페이지에서 스택 리소스에 태그를 지정합니다. [Next]를 선택합니다.
11. Review(검토) 페이지에서 정보를 검토하고, 스택이 IAM 리소스를 생성할 수 있는지 확인한 다음, Update(업데이트)를 선택합니다.

    Note

    업데이트가 완료될 때까지 기다린 후 다음 단계를 수행합니다.
12. 클러스터의 DNS 공급자가 `kube-dns`인 경우 `kube-dns` 배포를 1개의 복제본으로 축소합니다.

```
kubectl scale deployments/kube-dns --replicas=1 -n kube-system
```

13. (선택 사항) Kubernetes Cluster Autoscaler를 사용하는 경우 배포를 다시 1개의 복제본으로 축소합니다.

```
kubectl scale deployments/cluster-autoscaler --replicas=1 -n kube-system
```

스토리지 클래스

Kubernetes 버전 1.11 이전에 생성한 Amazon EKS 클러스터는 스토리지 클래스와 함께 생성되지 않았습니다. 클러스터가 사용할 스토리지 클래스를 정의해야 하고, 영구 볼륨 클레임에 대한 기본 스토리지 클래스를 정의해야 합니다. 자세한 내용은 Kubernetes 문서의 Storage Classes(스토리지 클래스)를 참조하십시오.

Amazon EKS 클러스터에 대한 AWS 스토리지 클래스를 생성하려면

1. 스토리지 클래스에 대한 AWS 스토리지 클래스 매니페스트 파일을 생성합니다. 아래 예제는 `gp2`라는 스토리지 클래스를 정의합니다. 이 클래스는 Amazon EBS `gp2` 볼륨 유형을 사용합니다. AWS 스토리지 클래스에 대해 사용 가능한 옵션에 대한 자세한 내용은 Kubernetes 문서의 AWS를 참조하십시오. 이 예제의 경우 파일은 `gp2-storage-class.yaml`입니다.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: gp2
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
mountOptions:
  - debug
```

2. kubectl을 사용하여 매니페스트 파일로부터 스토리지 클래스를 생성합니다.

```
kubectl create -f gp2-storage-class.yaml
```

결과:

```
storageclass "gp2" created
```

기본 스토리지 클래스를 정의하려면

1. 클러스터에 대한 기존 스토리지 클래스를 나열합니다. 기본값으로 설정하기 전에 스토리지 클래스를 정의해야 합니다.

```
kubectl get storageclass
```

결과:

```
NAME      PROVISIONER             AGE
gp2       kubernetes.io/aws-ebs   8m
sc1       kubernetes.io/aws-ebs   6s
```

2. 스토리지 클래스를 선택하고 `storageclass.kubernetes.io/is-default-class=true` 주석을 설정하여 이를 기본값으로 설정합니다.

```
kubectl patch storageclass gp2 -p '{"metadata": {"annotations":
{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

결과:

```
storageclass "gp2" patched
```

3. 스토리지 클래스가 이제 기본값으로 설정되었는지 확인합니다.

```
kubectl get storageclass
```

결과:

```
gp2 (default)   kubernetes.io/aws-ebs   12m
sc1             kubernetes.io/aws-ebs   4m
```

로드 밸런싱

Amazon EKS는 `LoadBalancer` 유형의 Kubernetes 서비스를 통해 Network Load Balancer 및 Classic Load Balancer를 지원합니다. 로드 밸런서의 구성은 서비스의 매니페스트에 추가된 주석에 의해 제어됩니다.

기본적으로 Classic Load Balancer는 `LoadBalancer` 유형 서비스에 사용됩니다. 대신 Network Load Balancer를 사용하려면 서비스에 다음 주석을 적용하십시오.

```
service.beta.kubernetes.io/aws-load-balancer-type: nlb
```

Network Load Balancer의 Kubernetes 사용에 대한 자세한 내용은 Kubernetes 설명서의 AWS의 Network Load Balancer 지원을 참조하십시오.

내부 로드 밸런서의 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.

내부 로드 밸런서를 사용하려면 서비스에 다음 주석을 적용하십시오.

```
service.beta.kubernetes.io/aws-load-balancer-internal: 0.0.0.0/0
```

Amazon EKS 네트워킹

이 장은 Amazon EKS에서 Kubernetes를 실행하는 데 따른 네트워킹 고려 사항을 다룹니다.

항목

- 클러스터 VPC 고려 사항 (p. 40)
- 클러스터 보안 그룹 고려 사항 (p. 41)
- 포드 네트워킹 (p. 43)
- CNI 구성 변수 (p. 45)
- CoreDNS 설치 (p. 46)
- 외부 SNAT(Source Network Address Translation) (p. 48)
- CNI 사용자 지정 네트워킹 (p. 51)
- Kubernetes 업그레이드용 Amazon VPC CNI 플러그인 (p. 52)
- Amazon EKS에 Calico 설치 (p. 53)

클러스터 VPC 고려 사항

Amazon EKS 클러스터를 생성할 때 클러스터가 사용할 Amazon VPC 서브넷을 지정합니다. Amazon EKS의 경우 2개 이상의 가용 영역에서 서브넷이 필요합니다. 내부에서 인터넷 경계 로드 밸런서를 생성하려면 작업자 노드에 대해 프라이빗 서브넷을 사용하고 Kubernetes에 대해 퍼블릭 서브넷을 사용하는 네트워크 아키텍처를 권장합니다. 클러스터를 만들 때 해당 클러스터를 위해 리소스를 호스팅할 모든 서브넷을 지정합니다(예: 작업자 노드와 로드 밸런서).

> **Note**
>
> 인터넷 경계 로드 밸런서는 클러스터에 퍼블릭 서브넷을 필요로 합니다. 작업자 노드는 또한 실행 시 클러스터 내부 검사 및 노드 등록을 위해 Amazon EKS API에 대한 아웃바운드 인터넷 액세스를 필요로 합니다. 컨테이너 이미지를 가져오려면 Amazon S3 및 Amazon ECR API(그리고 DockerHub 등 다른 컨테이너 레지스트리)에 대한 액세스가 필요합니다. 자세한 정보는 AWS General Reference의 클러스터 보안 그룹 고려 사항 (p. 41) 및 AWS IP 주소 범위를 참조하십시오.

클러스터를 생성할 때 전달하는 서브넷은 Amazon EKS가 제어 플레인에서 작업자 노드 통신에 사용하는 탄력적 네트워크 인터페이스를 배치하는 경우 영향을 미칩니다.

클러스터를 생성할 때 퍼블릭 또는 프라이빗 서브넷만을 지정할 수 있지만 이러한 구성과 관련하여 몇 가지 제한 사항이 있습니다.

- Private-only: 프라이빗 서브넷 및 Kubernetes에서 실행하는 모든 것이 포드에 대한 인터넷 경계 로드 밸런서를 생성할 수 없습니다.
- Public-only: 작업자 노드를 포함한 모든 것이 퍼블릭 서브넷에서 실행됩니다.

Amazon EKS는 프라이빗 서브넷의 탄력적 네트워크 인터페이스를 생성하여 작업자 노드에 대한 통신을 원활하게 합니다. 이 통신 채널은 kubectl exec 및 kubectl logs 등의 Kubernetes 기능을 지원합니다. 클러스터를 생성할 때 지정하는 보안 그룹은 클러스터 제어 플레인에 대해 생성된 탄력적 네트워크 인터페이스에 적용됩니다.

VPC는 DNS 호스트 이름과 DNS 확인을 모두 지원해야 합니다. 그렇지 않으면 작업자 노드가 클러스터에 등록되지 않습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC에서 DNS 사용을 참조하십시오.

VPC 태그 지정 요건

Amazon EKS 클러스터를 생성할 때 Amazon EKS는 Kubernetes에서 식별할 수 있도록 사용자가 지정하는 서브넷을 포함하는 VPC에 다음 방법으로 태그를 지정합니다.

키	값
kubernetes.io/cluster/*<cluster-name>*	shared

- 키: *<cluster-name>* 값은 Amazon EKS 클러스터의 이름과 일치합니다.
- 값: `shared` 값을 통해 2개 이상의 클러스터가 이 VPC를 사용할 수 있습니다.

서브넷 태그 지정 요건

Amazon EKS 클러스터를 생성할 때 Amazon EKS는 Kubernetes에서 식별할 수 있도록 다음 방법으로 서브넷에 태그를 지정합니다.

키	값
kubernetes.io/cluster/*<cluster-name>*	shared

- 키: *<cluster-name>* 값은 Amazon EKS 클러스터와 일치합니다.
- 값: `shared` 값을 통해 2개 이상의 클러스터가 이 서브넷을 사용할 수 있습니다.

내부 로드 밸런서에 대한 프라이빗 서브넷 태그 지정 요구사항

Kubernetes가 내부 로드 밸런서에 VPC의 프라이빗 서브넷을 사용할 수 있음을 알 수 있도록 태그가 지정되어야 합니다.

키	값
kubernetes.io/role/internal-elb	1

클러스터 보안 그룹 고려 사항

Amazon EKS 시작하기 (p. 3) 연습에 제공된 AWS CloudFormation 템플릿을 사용하여 VPC 및 작업자 노드를 생성하는 경우 제어 플레인 및 작업자 노드 보안 그룹은 권장 설정으로 구성됩니다.

작업자 노드에 대한 보안 그룹과 작업자 노드로의 제어 플레인 통신에 대한 보안 그룹이 설정되어 작업자 노드 내 권한이 있는 포트와의 통신을 방지합니다. 애플리케이션에 제어 플레인 또는 작업자 노드로부터 인바

운드 또는 아웃바운드 액세스 추가를 필요로 하는 경우 이러한 규칙을 클러스터와 연결된 보안 그룹에 추가해야 합니다. 자세한 정보는 Amazon VPC 사용 설명서의 VPC의 보안 그룹 단원을 참조하십시오.

> Note
>
> 권한이 있는 포트에서 프록시 기능을 허용하거나 CNCF 신뢰성 테스트를 실행하려면 제어 플레인 및 작업자 노드에 대한 보안 그룹을 편집해야 합니다. 작업자 노드 측의 보안 그룹은 제어 플레인의 0-65535 포트에 대한 인바운드 액세스를 허용해야 하고, 제어 플레인 측은 포트 0-65535의 작업자 노드에 대한 아웃바운드 액세스를 허용해야 합니다.

다음 표는 클러스터의 제어 플레인 및 작업자 노드 보안 그룹에 대한 최소 요구 및 권장 보안 그룹 설정을 보여줍니다.

제어 플레인 보안 그룹

	프로토콜	포트 범위	소스	Destination
최소 인바운드 트래픽	TCP	443	모든 작업자 노드 보안 그룹	
권장 인바운드 트래픽	TCP	443	모든 작업자 노드 보안 그룹	
최소 아웃바운드 트래픽	TCP	10250		모든 작업자 노드 보안 그룹
권장 아웃바운드 트래픽	TCP	1025-65535		모든 작업자 노드 보안 그룹

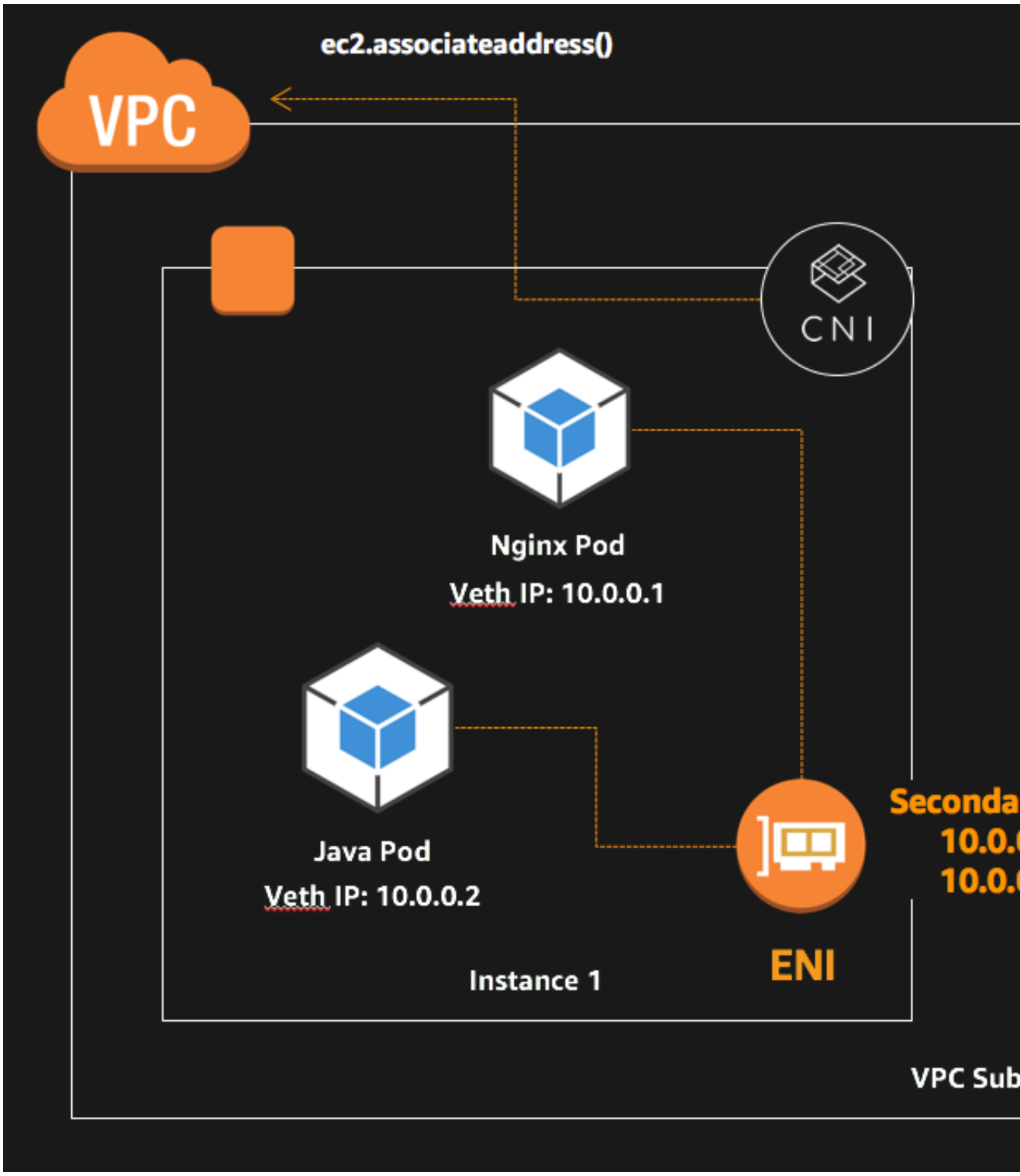
작업자 노드 보안 그룹

	프로토콜	포트 범위	소스	Destination
최소 인바운드 트래픽(다른 작업자 노드로부터)	작업자 간 통신을 위해 작업자 노드가 사용할 것으로 예상하는 모든 프로토콜	작업자 간 통신을 위해 작업자 노드가 사용할 것으로 예상하는 모든 포트	모든 작업자 노드 보안 그룹	
최소 인바운드 트래픽(제어 플레인으로부터)	TCP	10250	제어 플레인 보안 그룹	
권장 인바운드 트래픽	모두 TCP	모두 443, 1025-65535	모든 작업자 노드 보안 그룹 제어 플레인 보안 그룹	
최소 아웃바운드 트래픽*	TCP	443		제어 플레인 보안 그룹
권장 아웃바운드 트래픽	전체	모두		0.0.0.0/0

* 작업자 노드는 또한 실행 시 클러스터 내부 검사 및 노드 등록을 위해 Amazon EKS API에 대한 아웃바운드 인터넷 액세스를 필요로 합니다. 컨테이너 이미지를 가져오려면 Amazon S3 및 Amazon ECR API(그리고 DockerHub 등 다른 컨테이너 레지스트리)에 대한 액세스가 필요합니다. 자세한 내용은 AWS General Reference의 AWS IP 주소 범위를 참조하십시오.

포드 네트워킹

Amazon EKS는 Kubernetes용 Amazon VPC CNI 플러그인을 통한 기본 VPC 네트워킹을 지원합니다. 이 CNI 플러그인을 사용하면 Kubernetes 포드에서 VPC 네트워크에서와 마찬가지로 포드 내 동일한 IP 주소를 보유합니다. CNI 플러그인은 GitHub에서 유지 관리하는 오픈 소스 프로젝트입니다.



CNI 플러그인은 Kubernetes 노드에 VPC IP 주소를 할당하고 각 노드의 포드에 대한 필수 네트워킹을 구성하는 역할을 합니다. 플러그인에는 두 가지 기본 구성 요소가 있습니다.

- L-IPAM 데몬은 탄력적 네트워크 인터페이스와 인스턴스의 연결, 탄력적 네트워크 인터페이스에 보조 IP 주소 할당 및 예약 시 Kubernetes 포드에 할당할 각 노드에 있는 IP 주소의 "웜 풀" 유지 관리를 책임집니다.
- CNI 플러그인 자체는 호스트 네트워크 연결(예: 인터페이스 및 가상 이더넷 페어 구성) 및 포드 네임스페이스에 대한 올바른 인터페이스 추가를 책임집니다.

디자인 및 네트워킹 구성에 대한 자세한 내용은 Proposal: CNI plugin for Kubernetes networking over AWS VPC를 참조하십시오.

Amazon EC2 인스턴스 유형별 탄력적 네트워크 인터페이스 및 보조 IP 주소 제한이 해당됩니다. 일반적으로 인스턴스가 클수록 더 많은 IP 주소를 지원할 수 있습니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스 유형별/네트워크 인터페이스당 IP 주소 단원을 참조하십시오.

CNI 구성 변수

Kubernetes용 Amazon VPC CNI 플러그인은 환경 변수를 통해 설정되는 여러 구성 옵션을 지원합니다. 다음 환경 변수를 사용할 수 있으며 모두 선택적입니다.

`AWS_VPC_CNI_NODE_PORT_SUPPORT`

유형: 부울

기본값: `true`

작업자 노드의 기본 네트워크 인터페이스에서 `NodePort` 서비스가 활성화되어 있는지 여부를 지정합니다. 이렇게 하려면 추가 `iptables` 규칙이 필요하며, 기본 인터페이스에 있는 커널의 역경로 필터가 `loose`로 설정되어야 합니다.

`AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG`

유형: 부울

기본값: `false`

포드가 클러스터의 `resourcesVpcConfig`와 독립적인 서브넷 및 보안 그룹(컨트롤 플레인 리소스와 동일한 VPC 내에 있음)을 사용할 수 있는지 여부를 지정합니다. 기본적으로 포드는 작업자 노드의 기본 인터페이스와 동일한 서브넷 및 보안 그룹을 공유합니다. 이 변수를 `true`로 설정하면 `ipamD`에서 탄력적 네트워크 인터페이스 할당을 위해 작업자 노드의 `ENIConfig`에 있는 보안 그룹과 서브넷을 사용합니다. 포드가 있는 각 서브넷에 대해 `ENIConfig` 사용자 지정 리소스 정의를 생성한 다음 특정 `ENIConfig`를 사용하도록 각 작업자 노드에 주석을 달아야 합니다(동일한 `ENIConfig`를 사용하여 여러 작업자 노드에 주석을 달 수 있음). 한 번에 단일 `ENIConfig`를 사용하여 작업자 노드에 주석을 달 수 있으며, `ENIConfig`의 서브넷은 작업자 노드가 있는 가용 영역과 동일한 가용 영역에 속해야 합니다. 자세한 내용은 CNI 사용자 지정 네트워킹 (p. 51) 단원을 참조하십시오.

`AWS_VPC_K8S_CNI_EXTERNALSNAT`

유형: 부울

기본값: `false`

보조 ENI IP 주소의 SNAT를 제공하기 위해 외부 NAT 게이트웨이를 사용해야 하는지 여부를 지정합니다. `true`로 설정하면 SNAT `iptables` 규칙 및 오프 VPC IP 규칙이 적용되지 않으며, 해당 규칙이 이미 적용된 경우에는 제거됩니다.

외부 VPN, 직접 연결 및 외부 VPC에서 포드와의 인바운드 연결을 허용해야 하는 경우와 포드가 인터넷 게이트웨이를 통해 직접 인터넷에 액세스할 필요가 없는 경우 SNAT를 비활성화합니다. 그러나 노드는

프라이빗 서브넷에서 실행되어야 하며, AWS NAT 게이트웨이 또는 다른 외부 NAT 디바이스를 통해 연결되어야 합니다.

자세한 내용은 외부 SNAT(Source Network Address Translation) (p. 48)단원을 참조하십시오.

`WARM_ENI_TARGET`

유형: 정수

기본값: `1`

`ipamD` 데몬이 노드에 대한 포드 할당을 사용 가능한 상태로 유지하도록 시도해야 하는 무료 탄력적 네트워크 인터페이스(및 사용 가능한 모든 IP 주소)의 수를 지정합니다. 기본적으로 `ipamD`는 탄력적 네트워크 인터페이스 하나와 모든 IP 주소를 포드 할당에 사용 가능한 상태로 유지하려고 시도합니다.

Note

네트워크 인터페이스당 IP 주소의 수는 인스턴스 유형에 따라 달라집니다. 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스 유형별/네트워크 인터페이스당 IP 주소 단원을 참조하십시오.

예를 들어, `m4.4xlarge`는 네트워크 인터페이스 하나와 IP 주소 30개로 시작합니다. 노드에 포드 다섯 개가 있으며 준비된 IP 주소 풀에서 무료 IP 주소 다섯 개가 제거된 경우 `ipamD`는 `WARM_ENI_TARGET` 무료 인터페이스가 노드에서 사용 가능한 상태가 될 때까지 더 많은 인터페이스를 할당하려고 시도합니다.

Note

`WARM_IP_TARGET`이 설정되면 이 환경 변수는 무시되며, `WARM_IP_TARGET` 동작이 대신 사용됩니다.

`WARM_IP_TARGET`

유형: 정수

기본값: None

`ipamD` 데몬이 노드에 대한 포드 할당을 사용 가능한 상태로 유지하도록 시도해야 하는 무료 IP 주소의 수를 지정합니다. 예를 들어, `WARM_IP_TARGET`이 10으로 설정되면 `ipamD`는 항상 10개의 무료 IP 주소가 사용 가능한 상태가 되도록 유지하려고 시도합니다. 노드의 탄력적 네트워크 인터페이스가 이러한 무료 주소를 제공할 수 없는 경우 `ipamD`는 `WARM_IP_TARGET` 무료 IP 주소가 사용 가능한 상태가 될 때까지 더 많은 인터페이스를 할당하려고 시도합니다.

Note

이 환경 변수는 `WARM_ENI_TARGET` 동작을 재정의합니다.

CoreDNS 설치

Kubernetes 버전 1.11로 생성된 새로운 Amazon EKS 클러스터는 CoreDNS를 기본 DNS 및 서비스 검색 공급자로 사용하여 출고됩니다. Kubernetes 버전 1.10으로 생성된 클러스터는 `kube-dns`를 기본 DNS 및 서비스 검색 공급자로 사용하여 출고되었습니다. 1.10 클러스터를 1.11로 업데이트한 경우 CoreDNS를 DNS와 서비스 검색에 사용하려면 CoreDNS를 설치하고 `kube-dns`를 제거해야 합니다.

다음 명령을 사용하여 클러스터가 CoreDNS를 이미 실행하고 있는지 확인할 수 있습니다.

```
kubectl get pod -n kube-system -l k8s-app=kube-dns
```

출력의 포드 이름에 coredns가 표시되면 클러스터에서 CoreDNS를 이미 실행하고 있는 것입니다. 그렇지 않은 경우 다음 절차를 사용하여 DNS 및 서비스 검색 공급자를 CoreDNS로 업데이트합니다.

업데이트된 Amazon EKS 클러스터에 CoreDNS를 설치하려면

1. {"eks.amazonaws.com/component": "kube-dns"} 선택기를 클러스터의 kube-dns 배포에 추가합니다(이 단계는 동일한 레이블 세트를 제어하기 위해 두 개의 DNS 배포가 경쟁하지 않도록 하기 위한 것입니다).

   ```
   kubectl patch -n kube-system deployment/kube-dns --patch \
   '{"spec":{"selector":{"matchLabels":{"eks.amazonaws.com/component":"kube-dns"}}}}'
   ```

2. CoreDNS를 클러스터에 배포합니다.

   a. 클러스터의 DNS IP 주소를 DNS_CLUSTER_IP 환경 변수로 설정합니다.

   ```
   export DNS_CLUSTER_IP=$(kubectl get svc -n kube-system kube-dns -o
    jsonpath='{.spec.clusterIP}')
   ```

   b. 클러스터의 AWS 리전을 REGION 환경 변수로 설정합니다.

   ```
   export REGION="us-west-2"
   ```

   c. Amazon EKS 리소스 버킷에서 CoreDNS 매니페스트를 다운로드합니다.

   ```
   curl -O https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2018-11-07/
   dns.yaml
   ```

   d. dns.yaml 파일의 변수 자리 표시자를 해당 환경 변수 값으로 바꾸고 업데이트된 매니페스트를 클러스터에 적용합니다. 다음 명령은 이 작업을 한 단계로 완료합니다.

   ```
   cat dns.yaml | sed -e "s/REGION/$REGION/g" | sed -e "s/DNS_CLUSTER_IP/
   $DNS_CLUSTER_IP/g" | kubectl apply -f -
   ```

   e. 클러스터에서 coredns 포드 이름을 가져옵니다.

   ```
   COREDNS_POD=$(kubectl get pod -n kube-system -l eks.amazonaws.com/component=coredns
    \
   -o jsonpath='{.items[0].metadata.name}')
   ```

   f. coredns 포드를 쿼리하여 요청을 수신하고 있는지 확인합니다.

   ```
   kubectl get --raw /api/v1/namespaces/kube-system/pods/$COREDNS_POD:9153/proxy/
   metrics \
   | grep 'coredns_dns_request_count_total'
   ```

   Note

   예상 출력이 제대로 반환되려면 클러스터의 DNS 요청 속도에 따라 몇 분 정도 걸릴 수 있습니다.

   예상 출력(빨강색 숫자는 DNS 요청 수 합계):

   ```
   # HELP coredns_dns_request_count_total Counter of DNS requests made per zone,
    protocol and family.
   # TYPE coredns_dns_request_count_total counter
   coredns_dns_request_count_total{family="1",proto="udp",server="dns://:53",zone="."} 23
   ```

3. `kube-dns` 배포를 0개의 복제본으로 축소합니다.

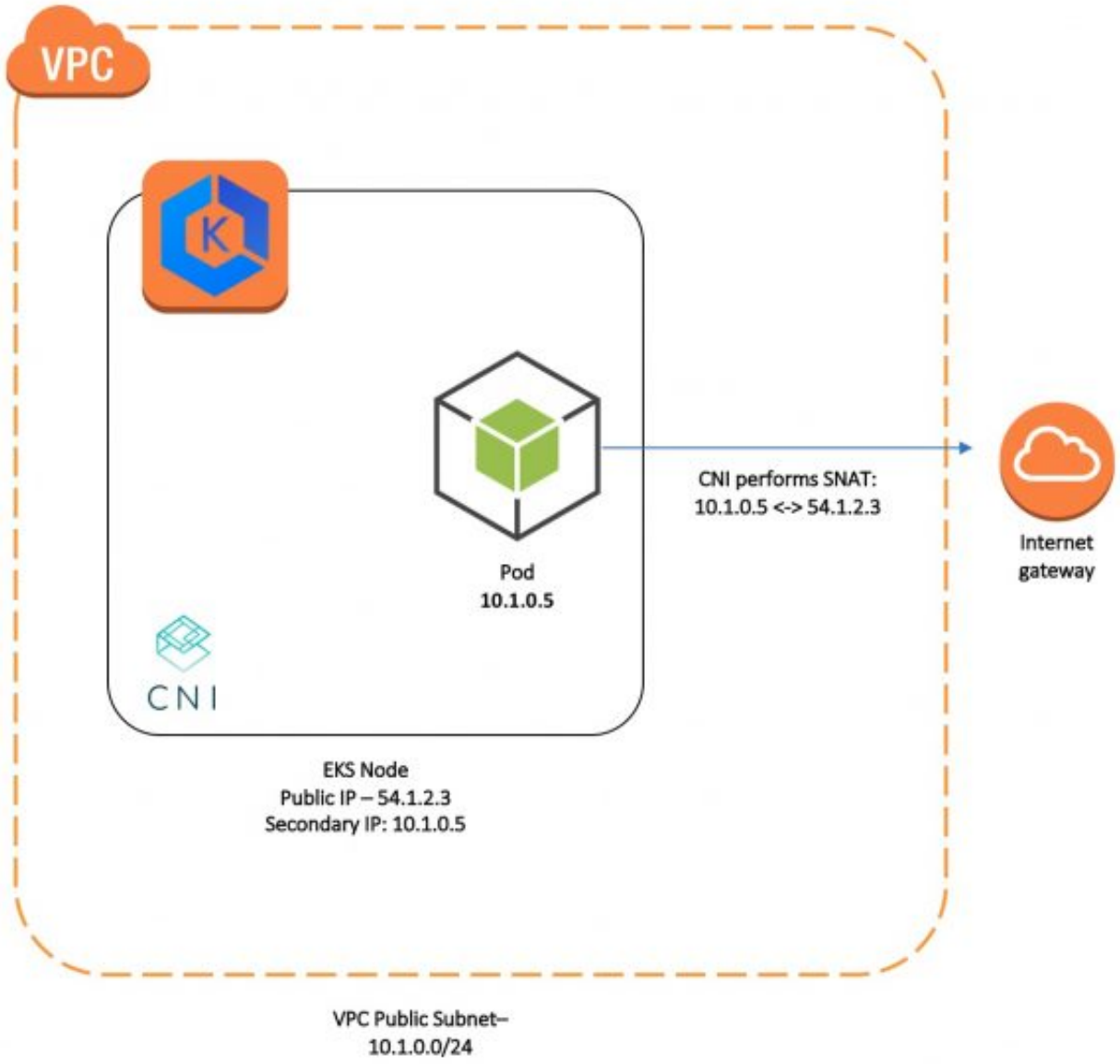
```
kubectl  scale -n kube-system deployment/kube-dns --replicas=0
```

4. 이전 `kube-dns` 리소스를 정리합니다.

```
kubectl delete -n kube-system deployment/kube-dns serviceaccount/kube-dns configmap/
kube-dns
```

외부 SNAT(Source Network Address Translation)

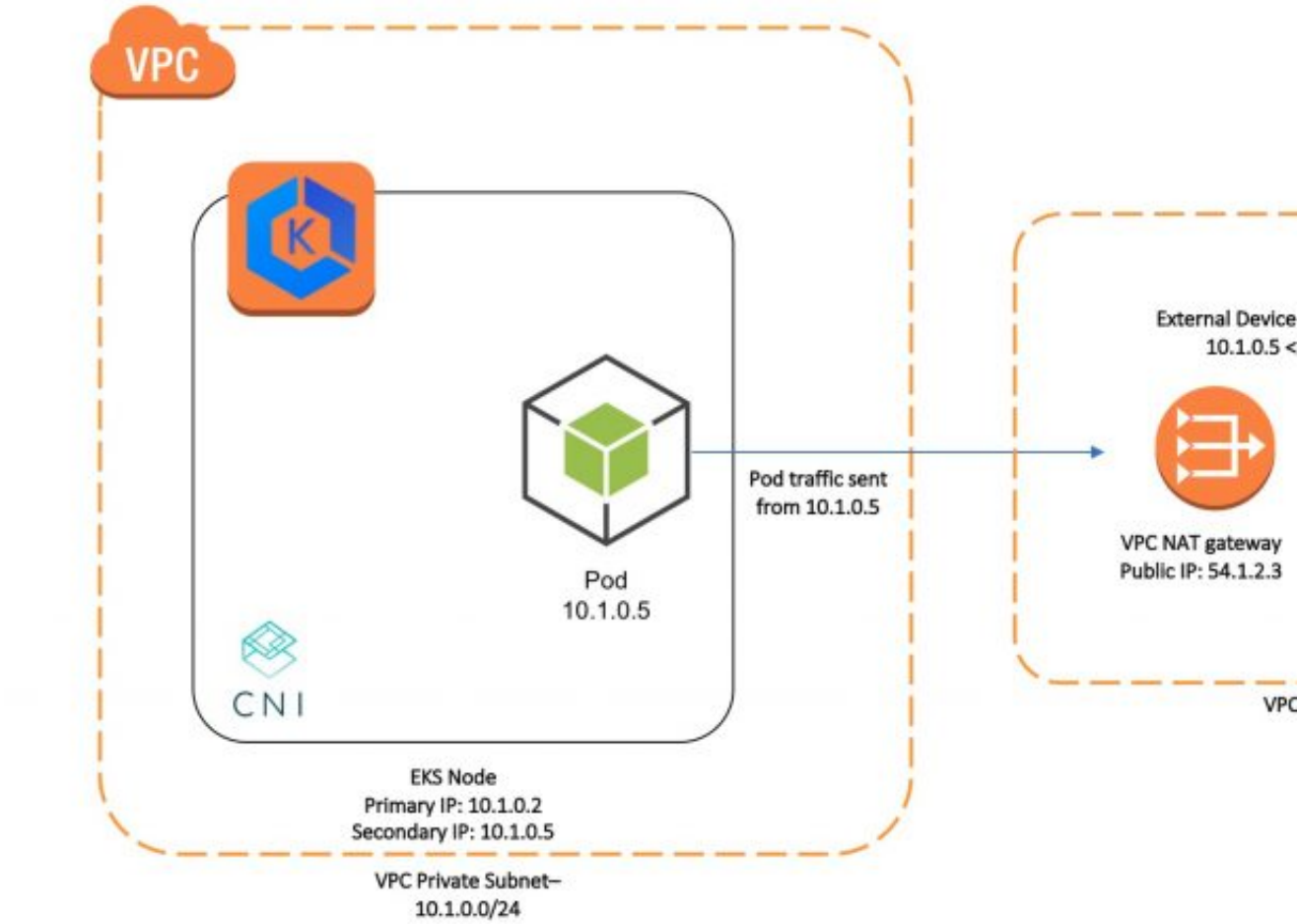
기본적으로 Kubernetes용 Amazon VPC CNI 플러그인은 SNAT(Source Network Address Translation)가 활성화된 포드를 구성합니다. 이는 패킷의 반환 주소를 인스턴스의 기본 퍼블릭 IP로 설정하고 인터넷과의 통신을 허용합니다. 이 기본 구성에서 인터넷 게이트웨이 및 퍼블릭 주소를 사용하면 반송 패킷이 올바른 Amazon EC2 인스턴스로 라우팅됩니다.



그러나 다른 프라이빗 IP 공간(예: VPC 피어링, 전송 VPC 또는 직접 연결)의 트래픽이 Amazon EC2 인스턴스의 기본 탄력적 네트워크 인터페이스에 연결되지 않은 포드와 직접 통신하려고 시도할 경우 SNAT로 인해 문제가 발생할 수 있습니다. NAT가 외부 디바이스(예: 인스턴스 자체가 아닌 NAT 게이트웨이)에서 처리되도록 지정하기 위해 `AWS_VPC_K8S_CNI_EXTERNALSNAT` 환경 변수를 `true`로 설정하여 인스턴스에서 SNAT를 비활성화할 수 있습니다. 외부 VPN, 직접 연결 및 외부 VPC에서 포드와의 인바운드 통신을 허용해야 하는 경우와 포드가 인터넷 게이트웨이를 통해 직접 인터넷에 액세스할 필요가 없는 경우 SNAT를 비활성화합니다.

Note

퍼블릭 서브넷에 있는 노드의 경우 SNAT가 필요합니다. 외부 SNAT를 사용하려면 노드가 프라이빗 서브넷에 있어야 하며, NAT 게이트웨이 또는 다른 외부 NAT 디바이스를 통해 인터넷에 연결되어야 합니다.



작업자 노드에서 SNAT를 비활성화하려면 다음을 수행합니다.

1. `aws-node` ConfigMap을 편집합니다.

```
kubectl edit daemonset -n kube-system aws-node
```

2. `AWS_VPC_K8S_CNI_EXTERNALSNAT` 환경 변수를 노드 컨테이너 사양에 추가하고 `true`로 설정합니다.

```
...
    spec:
      containers:
      - env:
        - name: AWS_VPC_K8S_CNI_EXTERNALSNAT
          value: "true"
```

```
        - name: AWS_VPC_K8S_CNI_LOGLEVEL
          value: DEBUG
        - name: MY_NODE_NAME
...
```

3. 파일을 저장하고 텍스트 편집기를 종료합니다.

CNI 사용자 지정 네트워킹

기본적으로 새 네트워크 인터페이스가 포드에 할당되면 ipamD는 작업자 노드의 기본 탄력적 네트워크 인터페이스 보안 그룹 및 서브넷을 사용합니다. 그러나 포드 네트워크 인터페이스에서 컨트롤 플레인 보안 그룹과 동일한 VPC 내에 있는 다른 보안 그룹 또는 서브넷을 사용해야 하는 사용 사례가 있습니다.

> **Note**
>
> 이 기능을 사용하려면 Kubernetes용 Amazon VPC CNI 플러그인 버전 1.2.1 이상이 필요합니다. CNI 버전을 확인하고 필요한 경우 업그레이드하려면 Kubernetes 업그레이드용 Amazon VPC CNI 플러그인 (p. 52) 단원을 참조하십시오.

- 서브넷에서 사용 가능한 IP 주소의 수는 제한되어 있습니다. 이러한 포드 수 제한은 클러스터에서 생성할 수 있습니다. 포드 그룹에 대해 다른 서브넷을 사용하면 사용 가능한 IP 주소의 수를 늘릴 수 있습니다.
- 보안상의 이유로 포드는 노드의 기본 네트워크 인터페이스와 다른 보안 그룹 또는 서브넷을 사용해야 합니다.

CNI 사용자 지정 네트워킹을 구성하려면 다음을 수행합니다.

1. `aws-node` ConfigMap을 편집합니다.

```
kubectl edit daemonset -n kube-system aws-node
```

2. `AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG` 환경 변수를 노드 컨테이너 사양에 추가하고 `true`로 설정합니다.

```
...
    spec:
      containers:
      - env:
        - name: AWS_VPC_K8S_CNI_CUSTOM_NETWORK_CFG
          value: "true"
        - name: AWS_VPC_K8S_CNI_LOGLEVEL
          value: DEBUG
        - name: MY_NODE_NAME
...
```

3. 파일을 저장하고 텍스트 편집기를 종료합니다.
4. 클러스터에 대한 새 `ENIConfig` 사용자 지정 리소스를 정의합니다.

   a. `ENIConfig.yaml`이라는 파일을 생성하고 다음 콘텐츠를 붙여 넣습니다.

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: eniconfigs.crd.k8s.amazonaws.com
spec:
  scope: Cluster
  group: crd.k8s.amazonaws.com
  version: v1alpha1
  names:
```

```
    plural: eniconfigs
    singular: eniconfig
    kind: ENIConfig
```

b. 다음 명령을 사용하여 클러스터에 파일을 적용합니다.

```
kubectl apply -f ENIConfig.yaml
```

5. 포드가 있는 각 서브넷에 대해 `ENIConfig` 사용자 지정 리소스 정의를 생성합니다.

a. 다음 정보와 함께 사용할 각 탄력적 네트워크 인터페이스 구성에 대한 고유 파일을 생성합니다. 서브넷 및 보안 그룹 ID를 사용자 고유의 값으로 바꿉니다. 이 예제의 경우 파일은 *group1-pod-netconfig*`.yaml`입니다.

Note

각 서브넷 및 보안 그룹 조합에는 자체 사용자 지정 리소스 정의가 필요합니다.

```
apiVersion: crd.k8s.amazonaws.com/v1alpha1
kind: ENIConfig
metadata:
 name: group1-pod-netconfig
spec:
 subnet: subnet-0c4678ec01ce68b24
 securityGroups:
 - sg-066c7927a794cf7e7
 - sg-08f5f22cfb70d8405
 - sg-0bb293fc16f5c2058
```

b. 다음 명령을 사용하여 이전에 생성한 각 사용자 지정 리소스 정의 파일을 클러스터에 적용합니다.

```
kubectl apply -f group1-pod-netconfig.yaml
```

6. 클러스터의 각 노드에 대해 사용할 사용자 지정 네트워크 구성으로 노드에 주석을 추가합니다. 작업자 노드에는 한 번에 `ENIConfig` 값 하나로만 주석을 추가할 수 있습니다. `ENIConfig`의 서브넷은 작업자 노드가 있는 것과 동일한 가용 영역에 속해야 합니다.

```
kubectl annotate node <nodename>.<region>.compute.internal k8s.amazonaws.com/
eniConfig=group1-pod-netconfig
```

Kubernetes 업그레이드용 Amazon VPC CNI 플러그인

Amazon EKS 클러스터를 실행하는 경우 클러스터에 Kubernetes용 Amazon VPC CNI 플러그인의 최신 버전이 적용됩니다(새 클러스터에서 버전 사용을 위한 전환이 이루어지기 전에 잠깐 동안 플러그인의 확실한 최신 버전을 GitHub에서 확인할 수 있습니다). 그러나 Amazon EKS에서는 새 버전이 릴리스될 때 클러스터의 CNI 플러그인을 자동으로 업그레이드하지 않습니다. 기존 클러스터에 대해 최신 버전을 가져오려면 CNI 플러그인을 수동으로 업그레이드해야 합니다.

최신 CNI 버전은 1.2.1입니다. 플러그인의 다른 버전을 확인하거나, 각 버전의 출시 정보를 읽어 보려면 GitHub를 참조하십시오.

CNI 버전을 확인하고 최신 버전으로 업그레이드하려면 다음 절차를 따르십시오.

Kubernetes용 Amazon VPC CNI 플러그인 버전을 확인하려면

- 다음 명령을 사용하여 클러스터의 CNI 버전을 인쇄합니다.

```
kubectl describe daemonset aws-node --namespace kube-system | grep Image | cut -d "/" -
f 2
```

결과:

```
amazon-k8s-cni:1.1.0
```

이 예제 출력에서 CNI 버전은 1.1.0이며, 최신 버전인 1.2.1의 전 버전입니다. 다음 절차에 따라 CNI를 업그레이드하십시오.

Kubernetes용 Amazon VPC CNI 플러그인을 업그레이드하려면

- 다음 명령을 사용하여 CNI 버전을 최신 버전으로 업그레이드합니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/master/
config/v1.2/aws-k8s-cni.yaml
```

Amazon EKS에 Calico 설치

Project Calico는 Kubernetes에 대한 네트워크 정책 엔진입니다. Calico 네트워크 정책 집행을 통해 네트워크 세분화 및 테넌트 격리를 구현할 수 있습니다. 이는 테넌트를 각각 격리해야 하는 다중 테넌트 환경에서 또는 개발, 스테이징 및 프로덕션에 별도의 환경을 생성하고자 하는 경우 유용합니다. 네트워크 정책은 네트워크 수신 및 발신 규칙을 생성할 수 있다는 점에서 AWS 보안 그룹과 유사하지만, 인스턴스를 보안 그룹에 할당하는 대신 포드 선택기 및 레이블을 사용하여 네트워크 정책을 포드에 할당합니다. 다음 절차에서는 Amazon EKS 클러스터에 Calico를 설치하는 방법을 보여줍니다.

Amazon EKS 클러스터에 Calico를 설치하려면

1. aws/amazon-vpc-cni-k8s GitHub 프로젝트에서 Calico 매니페스트를 적용합니다. 이 매니페스트는 kube-system 네임스페이스에 데몬 세트를 생성합니다.

```
kubectl apply -f https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/master/
config/v1.2/calico.yaml
```

2. kube-system 데몬 세트를 확인하고 calico-node 데몬 세트가 DESIRED 수만큼 포드를 보유하고 READY 상태가 되기까지 대기합니다. 이 경우 Calico가 작동 중입니다.

```
kubectl get daemonset calico-node --namespace kube-system
```

결과:

```
NAME          DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   NODE SELECTOR
 AGE
calico-node   3         3         3         3            3           <none>
 38s
```

Amazon EKS 클러스터에서 Calico를 삭제하려면

- Amazon EKS 클러스터에서 Calico를 사용을 마치면, 다음 명령을 사용하여 데몬 세트를 삭제할 수 있습니다.

```
kubectl delete daemonset calico-node --namespace kube-system
```

결과:

```
daemonset.extensions "calico-node" deleted
```

Stars Policy 데모

이 단원은 Project Calico 문서에서 제공하는 Stars Policy 데모를 연습합니다. 데모는 Amazon EKS 클러스터에 프런트엔드, 백엔드 및 클라이언트 서비스를 생성합니다. 데모는 또한 각 서비스 간 이용 가능한 수신 및 발신 경로를 보여주는 관리 GUI를 생성합니다.

네트워크 정책을 생성하기 전에 모든 서비스는 양방향으로 통신할 수 있습니다. 네트워크 정책을 적용한 후 클라이언트가 프런트엔드 서비스와의 통신만 가능하고, 백엔드의 경우 프런트엔드와의 통신만 가능한 것을 확인할 수 있습니다.

Stars Policy 데모를 실행하려면

1. 프런트엔드, 백엔드, 클라이언트 및 관리 UI 서비스를 적용합니다.

```
kubectl apply -f https://docs.projectcalico.org/v3.1/getting-started/kubernetes/
tutorials/stars-policy/manifests/00-namespace.yaml
kubectl apply -f https://docs.projectcalico.org/v3.1/getting-started/kubernetes/
tutorials/stars-policy/manifests/01-management-ui.yaml
kubectl apply -f https://docs.projectcalico.org/v3.1/getting-started/kubernetes/
tutorials/stars-policy/manifests/02-backend.yaml
kubectl apply -f https://docs.projectcalico.org/v3.1/getting-started/kubernetes/
tutorials/stars-policy/manifests/03-frontend.yaml
kubectl apply -f https://docs.projectcalico.org/v3.1/getting-started/kubernetes/
tutorials/stars-policy/manifests/04-client.yaml
```

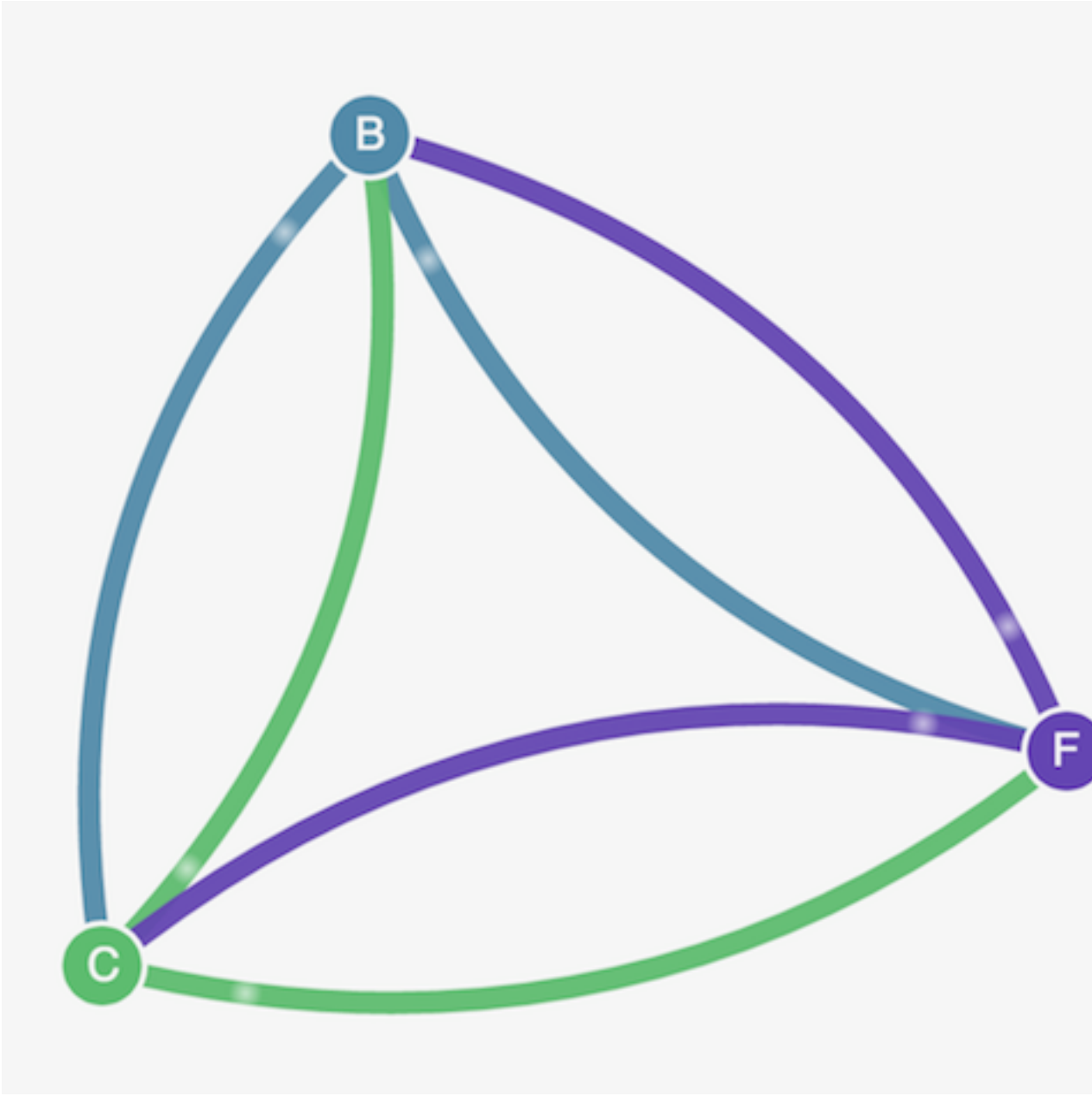
2. 모든 포드가 `Running` 상태가 될 때까지 대기합니다.

```
kubectl get pods --all-namespaces --watch
```

3. 관리 UI에 연결하려면 로컬 포트 9001을 클러스터를 실행하는 `management-ui` 서비스로 전달합니다.

```
kubectl port-forward service/management-ui -n management-ui 9001
```

4. 로컬 시스템에서 브라우저를 열고 http://localhost:9001/을 가리킵니다. 관리 UI가 표시됩니다. C 노드는 클라이언트 서비스이고, F 노드는 프런트엔드 서비스이며, B 노드는 백엔드 서비스입니다. 각 노드에는 기타 모든 노드에 대한 전체 통신 액세스를 보유합니다(진한 색상의 선으로 표시됨).



5. 다음 네트워크 정책을 적용하여 서비스를 각각 격리시킵니다.

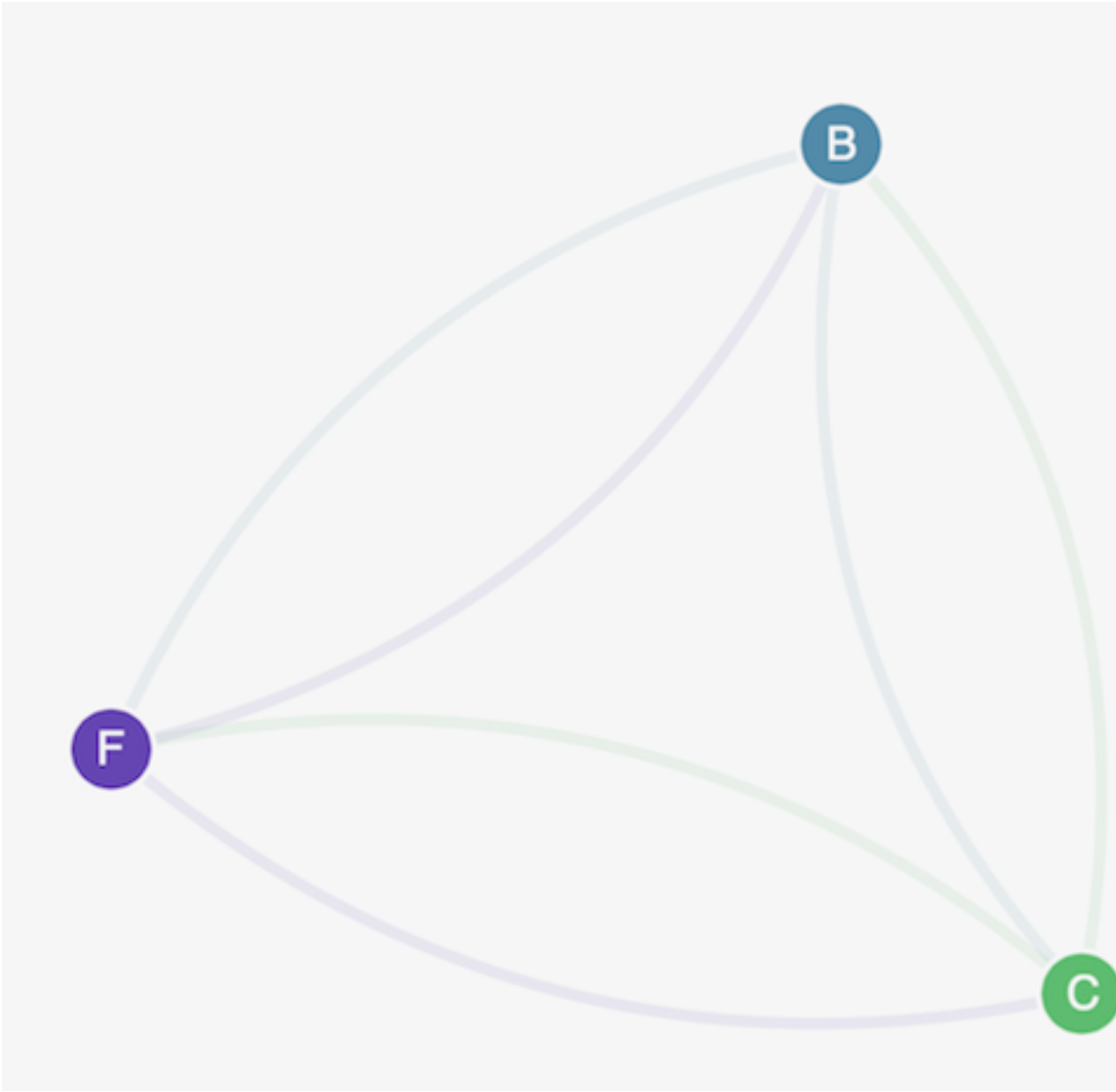
```
kubectl apply -n stars -f https://docs.projectcalico.org/v3.1/getting-started/
kubernetes/tutorials/stars-policy/policies/default-deny.yaml
kubectl apply -n client -f https://docs.projectcalico.org/v3.1/getting-started/
kubernetes/tutorials/stars-policy/policies/default-deny.yaml
```

6. 브라우저를 새로 고칩니다. 관리 UI가 더 이상 노드에 도달하지 않아 UI에 표시되지 않는 것을 확인할 수 있습니다.
7. 다음 네트워크 정책을 적용하여 관리 UI가 서비스에 액세스하도록 허용합니다.

```
kubectl apply -f https://docs.projectcalico.org/v3.1/getting-started/kubernetes/
tutorials/stars-policy/policies/allow-ui.yaml
```

```
kubectl apply -f https://docs.projectcalico.org/v3.1/getting-started/kubernetes/
tutorials/stars-policy/policies/allow-ui-client.yaml
```

8. 브라우저를 새로 고칩니다. 관리 UI가 노드에 다시 도달할 수 있지만, 노드가 서로 통신할 수 없음을 확인할 수 있습니다.

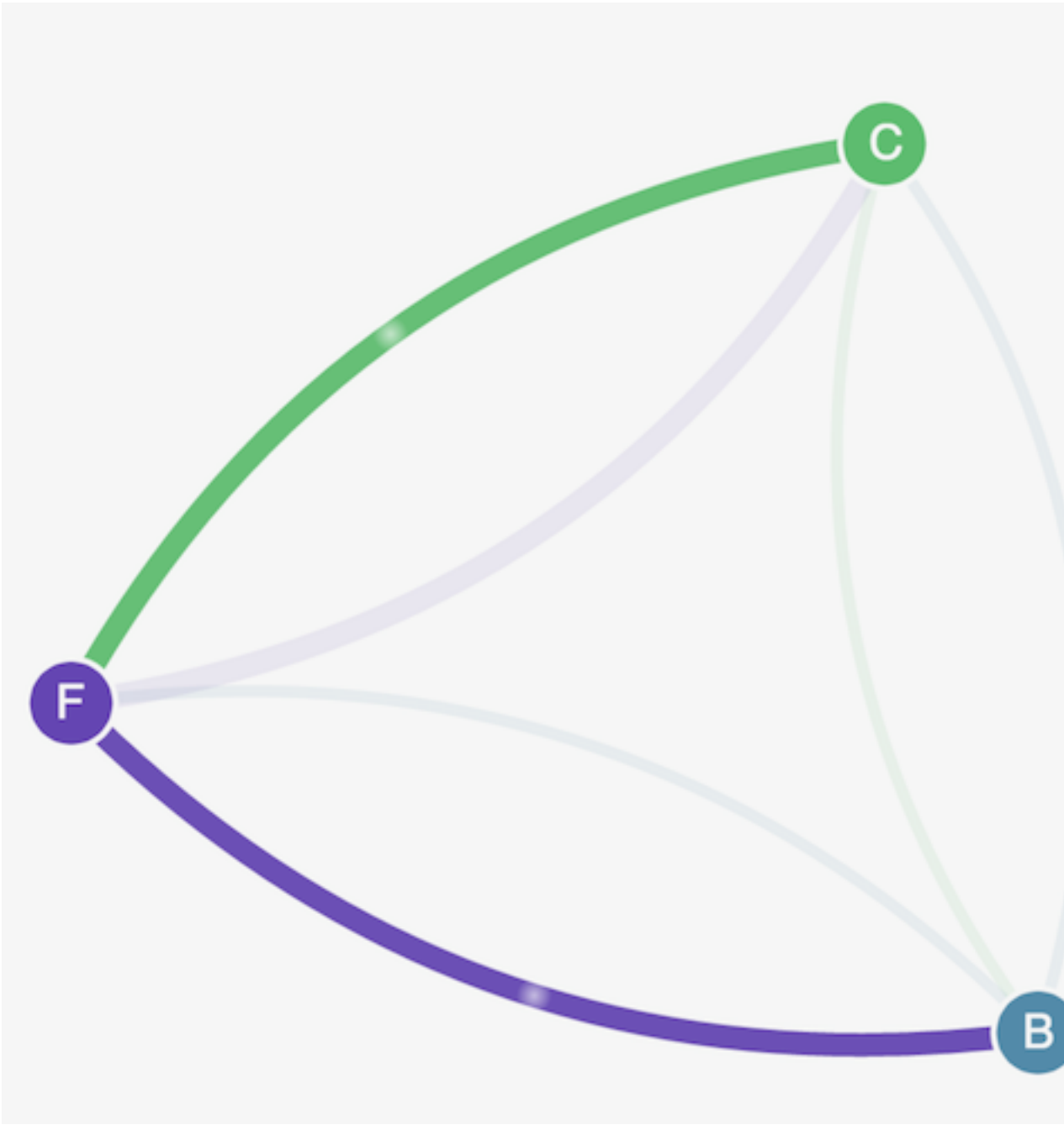


9. 다음 네트워크 정책을 적용하여 프런트엔드 서비스에서 백엔드 서비스로의 트래픽을 허용합니다.

```
kubectl apply -f https://docs.projectcalico.org/v3.1/getting-started/kubernetes/
tutorials/stars-policy/policies/backend-policy.yaml
```

10. 다음 네트워크 정책을 적용하여 `client` 네임스페이스에서 프런트엔드 서비스로의 트래픽을 허용합니다.

```
kubectl apply -f https://docs.projectcalico.org/v3.1/getting-started/kubernetes/
tutorials/stars-policy/policies/frontend-policy.yaml
```

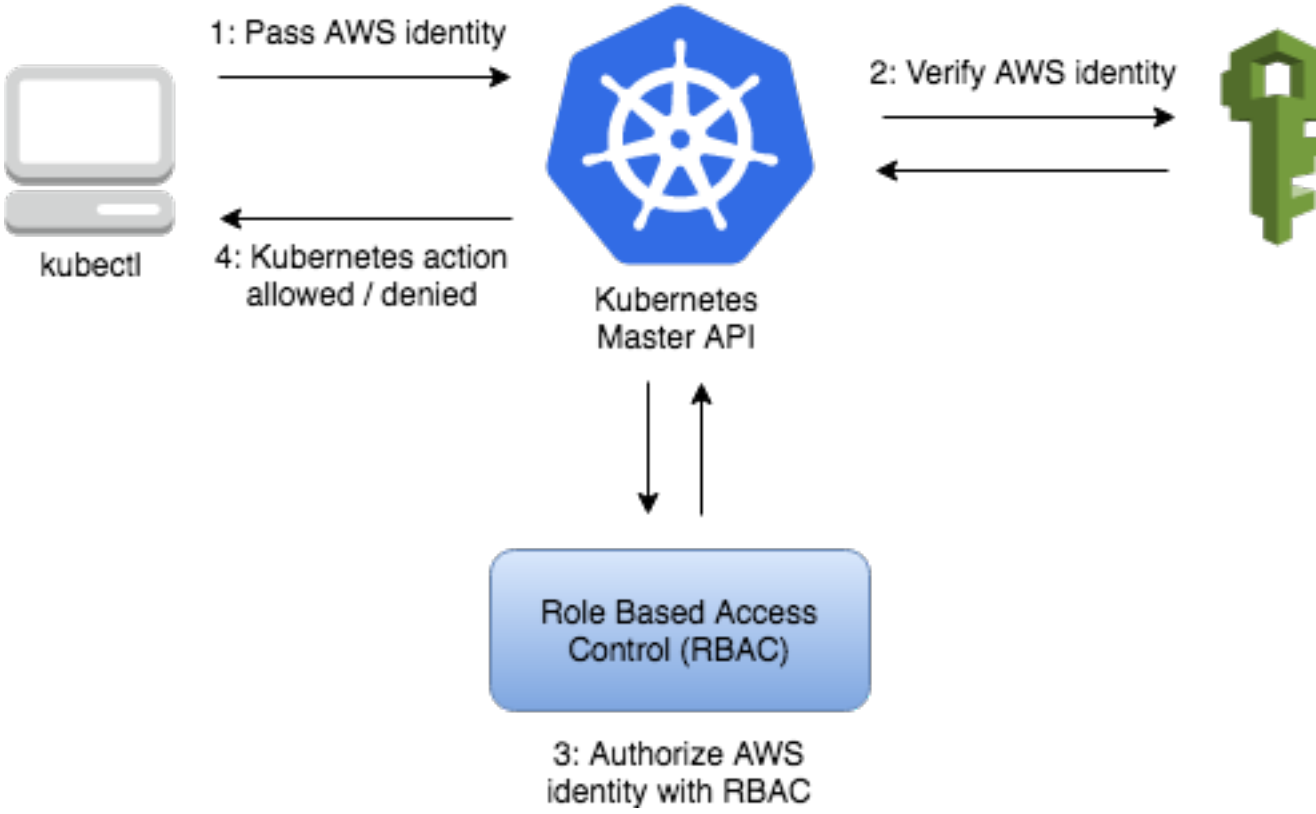


11. (선택 사항) 데모를 완료할 때 다음 명령을 사용하여 리소스를 삭제할 수 있습니다.

```
kubectl delete ns client stars management-ui
```

클러스터 인증 관리

Amazon EKS는 IAM을 사용하여 Kubernetes 클러스터에 인증을 제공(Kubernetes용 AWS IAM Authenticator 사용)하지만 인증에 대해 기본 Kubernetes 역할 기반 액세스 제어(RBAC)를 사용합니다. 따라서 IAM은 유효한 IAM 엔터티의 인증에만 사용됩니다. Amazon EKS 클러스터의 Kubernetes API와의 상호 작용에 대한 모든 권한은 기본 Kubernetes RBAC 시스템을 통해 관리합니다.



항목

- kubectl 설치 (p. 58)
- aws-iam-authenticator 설치 (p. 61)
- Amazon EKS에 대한 kubeconfig 생성 (p. 62)
- 클러스터의 사용자 또는 IAM 역할 관리 (p. 65)

kubectl 설치

Kubernetes는 클러스터 API 서버와 통신하기 위해 `kubectl`이라는 명령줄 유틸리티를 사용합니다. `kubectl` 바이너리는 많은 운영 체제 패키지 관리자에 제공되며, 이 옵션은 대체로 수동 다운로드 및 설치 프로세스보다 훨씬 쉽습니다. 해당 운영 체제 또는 패키지 관리자에 대한 Kubernetes 문서의 지침에 따라 설치할 수 있습니다.

이 주제는 MacOS, Linux 및 Windows 운영 체제용 Amazon EKS 판매 kubectl 바이너리를 다운로드하고 설치하는 데 도움이 됩니다.

항목

- MacOS (p. 59)
- 리눅스 (p. 59)
- Windows가 설치된 (p. 60)

MacOS

이 섹션은 MacOS 클라이언트용 `kubectl` 설치에 도움이 됩니다.

MacOS에 **`kubectl`**을 설치하려면

1. Amazon S3에서 Amazon EKS 판매 kubectl 바이너리를 다운로드합니다.

```
curl -o kubectl https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/darwin/amd64/kubectl
```

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

   a. MacOS용 SHA-256 합계 다운로드:

```
curl -o kubectl.sha256 https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/darwin/amd64/kubectl.sha256
```

   b. 다운로드한 바이너리에 대해 SHA-256 합계를 확인합니다.

```
openssl sha1 -sha256 kubectl
```

   c. 명령 출력에 생성된 SHA-256 합계를 다운로드한 SHA-256 파일과 비교합니다. 두 값이 일치해야 합니다.

3. 바이너리에 실행 권한을 적용합니다.

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

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

```
mkdir $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
```

6. kubectl을 설치한 이후 다음 명령을 사용하여 버전을 확인할 수 있습니다.

```
kubectl version --short --client
```

리눅스

이 섹션은 Linux 클라이언트용 `kubectl` 설치에 도움이 됩니다.

Linux에 **`kubectl`**을 설치하려면

1. Amazon S3에서 Amazon EKS 판매 kubectl 바이너리를 다운로드합니다.

```
curl -o kubectl https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/kubectl
```

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

   a. Linux용 SHA-256 합계 다운로드:

   ```
   curl -o kubectl.sha256 https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/kubectl.sha256
   ```

   b. 다운로드한 바이너리에 대해 SHA-256 합계를 확인합니다.

   ```
   openssl sha1 -sha256 kubectl
   ```

   c. 명령 출력에 생성된 SHA-256 합계를 다운로드한 SHA-256 파일과 비교합니다. 두 값이 일치해야 합니다.

3. 바이너리에 실행 권한을 적용합니다.

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

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

```
mkdir $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
```

6. kubectl을 설치한 이후 다음 명령을 사용하여 버전을 확인할 수 있습니다.

```
kubectl version --short --client
```

Windows가 설치된

이 섹션은 PowerShell을 이용한 Windows 클라이언트용 `kubectl` 설치에 도움이 됩니다.

Windows에서 **`kubectl`**을 설치하려면

1. PowerShell 터미널을 엽니다.
2. Amazon S3에서 Amazon EKS 판매 kubectl 바이너리를 다운로드합니다.

```
curl -o kubectl.exe https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/windows/amd64/kubectl.exe
```

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

   a. Windows용 SHA-256 합계 다운로드:

```
curl -o kubectl.exe.sha256 https://amazon-eks.s3-us-
west-2.amazonaws.com/1.10.3/2018-07-26/bin/windows/amd64/kubectl.exe.sha256
```

b. 다운로드한 바이너리에 대해 SHA-256 합계를 확인합니다.

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

c. 명령 출력에 생성된 SHA-256 합계를 다운로드한 SHA-256 파일과 비교합니다. PowerShell 출력은 대문자이지만 이 두 가지는 일치해야 합니다.

4. 바이너리를 `PATH`의 폴더에 복사합니다. PATH에 명령줄 유틸리티에 사용하는 기존 디렉터리가 있으면 해당 디렉터리로 바이너리를 복사하십시오. 그렇지 않은 경우 다음 단계를 완료합니다.

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

   b. `kubectl.exe` 바이너리를 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.

5. kubectl을 설치한 이후 다음 명령을 사용하여 버전을 확인할 수 있습니다.

```
kubectl version --short --client
```

aws-iam-authenticator 설치

Amazon EKS는 IAM을 사용하여 Kubernetes용 AWS IAM Authenticator를 통해 Kubernetes 클러스터에 인증을 제공합니다. Kubernetes 버전 1.10부터 Kubernetes용 AWS IAM Authenticator를 설치하고 인증에 사용할 kubectl 구성 파일을 사용하여 Amazon EKS를 사용할 stock kubectl 클라이언트를 구성할 수 있습니다.

Amazon EKS용 kubectl를 설치하려면

- 운영 체제에 따라 kubectl을 다운로드하고 설치하는 다양한 옵션이 있습니다.
  - `kubectl` 바이너리는 많은 운영 체제 패키지 관리자에 제공되며, 이 옵션은 대체로 수동 다운로드 및 설치 프로세스보다 훨씬 쉽습니다. 해당 운영 체제 또는 패키지 관리자에 대한 Kubernetes 문서의 지침에 따라 설치할 수 있습니다.
  - Amazon EKS는 동일한 버전의 업스트림 kubectl 바이너리와 동일하게 사용할 수 있는 kubectl 바이너리도 판매합니다. 운영 체제에 해당하는 Amazon EKS 판매 바이너리를 설치하려면 `kubectl` 설치 (p. 58)를 참조하십시오.

Amazon EKS용 **`aws-iam-authenticator`**를 설치하려면

1. Amazon S3에서 Amazon EKS 판매 `aws-iam-authenticator` 바이너리를 다운로드합니다.
   - Linux: https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/linux/amd64/aws-iam-authenticator
   - MacOS: https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/darwin/amd64/aws-iam-authenticator
   - Windows: https://amazon-eks.s3-us-west-2.amazonaws.com/1.10.3/2018-07-26/bin/windows/amd64/aws-iam-authenticator.exe

   아래 명령을 사용하여 바이너리를 다운로드하고 플랫폼에 맞는 올바른 URL로 교체합니다. 아래 예는 MacOS 클라이언트에 해당됩니다.

```
curl -o aws-iam-authenticator https://amazon-eks.s3-us-
west-2.amazonaws.com/1.10.3/2018-07-26/bin/darwin/amd64/aws-iam-authenticator
```

2. (선택 사항) 동일한 버킷 접두사에 제공된 SHA-256 합계를 사용하여 다운로드한 바이너리를 확인하고 플랫폼에 맞는 올바른 URL로 바꿉니다.

   a. 해당 시스템에 맞는 SHA-256 합계를 다운로드하십시오. 아래 예는 MacOS 클라이언트의 SHA-256 합계를 다운로드합니다.

   ```
   curl -o aws-iam-authenticator.sha256 https://amazon-eks.s3-us-
   west-2.amazonaws.com/1.10.3/2018-07-26/bin/darwin/amd64/aws-iam-
   authenticator.sha256
   ```

   b. 다운로드한 바이너리에 대해 SHA-256 합계를 확인합니다. 아래의 예제 `openssl` 명령은 MacOS 및 Ubuntu 클라이언트에 대해 테스트되었습니다. 해당 운영 체제에서 다른 명령이나 구문을 사용하여 SHA-256 합계를 검사할 수도 있습니다. 필요한 경우 해당 운영 체제 설명서를 참조하십시오.

   ```
   openssl sha1 -sha256 aws-iam-authenticator
   ```

   c. 명령 출력에 생성된 SHA-256 합계를 다운로드한 `aws-iam-authenticator.sha256` 파일과 비교합니다. 두 값이 일치해야 합니다.

3. 바이너리에 실행 권한을 적용합니다.

```
chmod +x ./aws-iam-authenticator
```

4. 바이너리를 `$PATH`의 폴더에 복사합니다. `$HOME/bin/aws-iam-authenticator`를 생성하고 `$PATH`가 `$HOME/bin`로 시작하는 것이 좋습니다.

```
cp ./aws-iam-authenticator $HOME/bin/aws-iam-authenticator && export PATH=$HOME/bin:
$PATH
```

5. `PATH` 환경 변수에 `$HOME/bin`을 추가합니다.

   - MacOS의 Bash 셸의 경우:

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

   - Linux의 Bash 셸의 경우:

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

6. `aws-iam-authenticator` 바이너리가 작동하는지 테스트합니다.

```
aws-iam-authenticator help
```

기존 Amazon EKS 클러스터가 있는 경우 해당 클러스터에 대한 `kubeconfig` 파일을 생성합니다. 자세한 내용은 Amazon EKS에 대한 `kubeconfig` 생성 (p. 62) 단원을 참조하십시오. 그렇지 않은 경우 새 Amazon EKS 클러스터를 생성하려면 Amazon EKS 클러스터 생성 (p. 16) 단원을 참조하십시오.

Amazon EKS에 대한 `kubeconfig` 생성

이 단원에서는 클러스터에 대한 `kubeconfig` 파일을 생성(또는 기존 파일을 업데이트)합니다.

이 단원에서는 kubeconfig를 생성하거나 업데이트하는 두 가지 절차를 제공합니다. 첫 번째 절차와 같이 AWS CLI update-kubeconfig 명령을 사용하면 kubeconfig를 빠르게 만들거나 업데이트할 수 있고, 두 번째 절차를 사용하면 kubeconfig를 수동으로 만들 수 있습니다.

Amazon EKS는 클러스터 인증을 위한 kubectl과 함께 Kubernetes용 AWS IAM Authenticator를 사용하는데, 이 Authenticator는 AWS CLI 및 AWS SDK와 동일한 기본 AWS 자격 증명 공급자 체인을 사용합니다. 시스템에 AWS CLI를 설치한 경우에는 기본적으로 Kubernetes용 AWS IAM Authenticator에서 다음 명령을 통해 반환되는 것과 동일한 자격 증명을 사용합니다.

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

자세한 내용은 AWS CLI의 AWS Command Line Interface 사용 설명서 구성을 참조하십시오.

AWS CLI로 **kubeconfig**를 생성하려면

1. 1.16.18 버전 이상의 AWS CLI가 설치되어 있어야 합니다. AWS CLI를 설치 또는 업그레이드하려면 AWS Command Line Interface 사용 설명서의 AWS 명령줄 인터페이스 설치를 참조하십시오.

   Note

   시스템의 Python 버전이 Python 3 또는 Python 2.7.9 이상이어야 합니다. 그렇지 않은 경우 Amazon EKS에 대한 AWS CLI 호출과 함께 `hostname doesn't match` 오류가 표시됩니다. 자세한 내용은 Python 요청 FAQ의 What are "hostname doesn't match" errors?를 참조하십시오.

   AWS CLI 버전은 다음 명령을 통해 확인할 수 있습니다.

   ```
   aws --version
   ```

   Important

   yum, apt-get 또는 Homebrew 등의 MacOS용 패키지 관리자는 흔히 AWS CLI의 여러 버전 뒤에 있습니다. 최신 버전이 설치되어 있는지 확인하려면 AWS Command Line Interface 사용 설명서의 AWS 명령줄 인터페이스 설치를 참조하십시오.

2. AWS CLI update-kubeconfig 명령을 사용하여 클러스터에 대한 kubeconfig를 생성하거나 업데이트합니다.

   - 기본적으로, 구성 파일은 홈 디렉터리의 기본 kubeconfig 경로(`.kube/config`)에 생성되거나 해당 위치의 기존 kubeconfig와 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 엔터티가 사용됩니다. aws sts get-caller-identity 명령을 실행하여 기본 AWS CLI 또는 SDK ID를 확인할 수 있습니다.
   - 자세한 내용은 aws eks update-kubeconfig help 명령의 도움말 페이지 또는 AWS CLI Command Reference의 update-kubeconfig를 참조하십시오.

   ```
   aws eks --region region update-kubeconfig --name cluster_name
   ```

3. 구성을 테스트합니다.

   ```
   kubectl get svc
   ```

Note

"aws-iam-authenticator": executable file not found in $PATH 오류가 표시되는 경우 Amazon EKS에 대해 kubectl이 구성되지 않았습니다. 자세한 내용은 ??? 단원을 참조하십시오.

결과:

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

kubeconfig 파일을 수동으로 생성하려면

1. 이미 존재하지 않는 경우 기본 ~/.kube 디렉터리를 생성합니다.

```
mkdir -p ~/.kube
```

2. 원하는 텍스트 편집기를 열고 아래의 kubeconfig 코드 블록을 여기에 복사합니다.

```
apiVersion: v1
clusters:
- cluster:
    server: <endpoint-url>
    certificate-authority-data: <base64-encoded-ca-cert>
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: aws
  name: aws
current-context: aws
kind: Config
preferences: {}
users:
- name: aws
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1alpha1
      command: aws-iam-authenticator
      args:
        - "token"
        - "-i"
        - "<cluster-name>"
        # - "-r"
        # - "<role-arn>"
      # env:
        # - name: AWS_PROFILE
        #   value: "<aws-profile>"
```

3. <endpoint-url>을 클러스터에 대해 생성한 엔드포인트 URL로 교체합니다.
4. <base64-encoded-ca-cert>을 클러스터에 대해 생성한 certificateAuthority.data로 교체합니다.
5. <cluster-name>를 클러스터 사용자 이름으로 바꿉니다.
6. (선택 사항) 기본 AWS 자격 증명 공급자 체인 대신, Kubernetes용 AWS IAM Authenticator가 클러스터 작업을 수행하는 역할을 맡도록 하려면, -r 및 <role-arn> 줄의 주석 처리를 해제하고 사용할 IAM 역할 ARN을 사용자로 바꿉니다.

7. (선택 사항) Kubernetes용 AWS IAM Authenticator가 항상 특정 이름의 AWS 자격 증명 프로필을 사용하도록 하려면(기본 AWS 자격 증명 공급자 체인 대신), `env` 줄의 주석 처리를 해제하고 *<aws-profile>*을 사용할 프로필 이름으로 교체합니다.
8. 파일을 기본 kubectl 폴더에 저장합니다. 파일 이름에 클러스터 이름이 포함됩니다. 예를 들어 클러스터 이름이 *devel*인 경우 파일을 `~/.kube/config-`*devel*에 저장합니다.
9. `KUBECONFIG` 환경 변수에 파일 경로를 추가하면 kubectl이 클러스터 구성을 찾아야 하는 곳을 알게 됩니다.

```
export KUBECONFIG=$KUBECONFIG:~/.kube/config-devel
```

10. (선택 사항) 셸 초기화 파일에 구성을 추가하면 셸을 열 때 구성됩니다.
    - MacOS의 Bash 셸의 경우:

```
echo 'export KUBECONFIG=$KUBECONFIG:~/.kube/config-devel' >> ~/.bash_profile
```

    - Linux의 Bash 셸의 경우:

```
echo 'export KUBECONFIG=$KUBECONFIG:~/.kube/config-devel' >> ~/.bashrc
```

11. 구성을 테스트합니다.

```
kubectl get svc
```

> **Note**
>
> `"aws-iam-authenticator": executable file not found in $PATH` 오류가 표시되는 경우 Amazon EKS에 대해 kubectl이 구성되지 않았습니다. 자세한 내용은 ??? 단원을 참조하십시오.

결과:

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

클러스터의 사용자 또는 IAM 역할 관리

Amazon EKS 클러스터를 생성할 경우, 클러스터를 생성하는 IAM 개체 사용자나 역할(예: 연합된 사용자)에는 클러스터의 RBAC 구성에서 `system:master` 권한이 자동으로 부여됩니다. AWS 사용자나 역할에 클러스터와 상호 작용할 추가 권한을 부여하려면 Kubernetes 내에서 `aws-auth` ConfigMap을 편집해야 합니다.

> **Note**
>
> 다른 IAM 자격 증명에 대한 자세한 내용은 IAM 사용 설명서의 자격 증명(사용자, 그룹 및 역할)을 참조하십시오. Kubernetes RBAC 구성에 대한 자세한 내용은 Using RBAC Authorization(RBAC 승인 사용)을 참조하십시오.

`aws-auth` ConfigMap은 Amazon EKS 시작하기 (p. 3) 가이드의 일환으로 적용됩니다. 이 가이드는 Amazon EKS 클러스터 생성부터 샘플 Kubernetes 애플리케이션 배포에 이르기까지 종합적인 연습을 제공합니다. 이 ConfigMap은 처음에는 작업자 노드가 클러스터에 참여하도록 하기 위해 만들어졌으나 이 ConfigMap을 사용하여 IAM 사용자 및 역할 RBAC 액세스를 추가할 수도 있습니다. 아직 작업자 노드를 시작하지 않았고 `aws-auth` ConfigMap을 적용하지 않았다면 다음 절차를 수행하면 됩니다.

클러스터에 **`aws-auth`** ConfigMap을 적용하려면

1. `aws-auth` ConfigMap을 이미 적용하였는지 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 재고 ConfigMap을 적용합니다.

2. AWS Authenticator 구성 맵을 다운로드, 편집 및 적용합니다.

   a. 구성 맵 다운로드:

   ```
   curl -O https://amazon-eks.s3-us-west-2.amazonaws.com/cloudformation/2018-11-07/aws-auth-cm.yaml
   ```

   b. 즐겨찾는 텍스트 편집기에서 파일을 엽니다. 인스턴스 역할의 *<ARN(##### ####)>* 조각을 이전 절차에서 기록한 NodeInstanceRole 값으로 교체하고 파일을 저장합니다.

   > **Important**
   >
   > 이 파일에서 어떠한 행도 수정하지 마십시오.

   ```
   apiVersion: v1
   kind: ConfigMap
   metadata:
     name: aws-auth
     namespace: kube-system
   data:
     mapRoles: |
       - rolearn: <ARN of instance role (not instance profile)>
         username: system:node:{{EC2PrivateDNSName}}
         groups:
           - system:bootstrappers
           - system:nodes
   ```

   c. 구성을 적용합니다. 이 명령을 완료하는 데 몇 분이 걸릴 수 있습니다.

   ```
   kubectl apply -f aws-auth-cm.yaml
   ```

   > **Note**
   >
   > "aws-iam-authenticator": executable file not found in $PATH 오류가 표시되는 경우 Amazon EKS에 대해 kubectl이 구성되지 않았습니다. 자세한 내용은 ??? 단원을 참조하십시오.

3. 노드의 상태를 확인하고 Ready 상태가 될 때까지 대기합니다.

   ```
   kubectl get nodes --watch
   ```

IAM 사용자 또는 역할을 Amazon EKS 클러스터에 추가하려면

1. kubectl이 사용 중인 AWS 자격 증명이 이미 클러스터에 대해 인증되어야 합니다. 클러스터를 생성한 IAM 사용자는 기본적으로 이러한 권한을 갖습니다.
2. aws-auth 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, 앞의 절차를 통해 재고 ConfigMap을 적용합니다.

예제 ConfigMap:

```
# 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: |
    - rolearn: arn:aws:iam::111122223333:role/doc-test-worker-nodes-NodeInstanceRole-
WDO5P42N3ETB
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"mapRoles":"- rolearn: arn:aws:iam::111122223333:role/
doc-test-worker-nodes-NodeInstanceRole-WDO5P42N3ETB\n  username: system:node:
{{EC2PrivateDNSName}}\n  groups:\n    - system:bootstrappers\n    -
 system:nodes\n"},"kind":"ConfigMap","metadata":{"annotations":{},"name":"aws-
auth","namespace":"kube-system"}}
  creationTimestamp: 2018-04-04T18:49:10Z
  name: aws-auth
  namespace: kube-system
  resourceVersion: "780"
  selfLink: /api/v1/namespaces/kube-system/configmaps/aws-auth
  uid: dcc31de5-3838-11e8-af26-02e00430057c
```

3. IAM 사용자, 역할 또는 AWS 계정을 configMap에 추가합니다.

- IAM 사용자 추가: `data`에서 ConfigMap의 `mapUsers` 섹션에 사용자 세부 정보를 추가합니다. 파일에 이미 존재하지 않는 경우 이 섹션을 추가합니다. 각 항목은 다음 파라미터를 지원합니다.
  - userarn: 추가할 IAM 사용자의 ARN.
  - username: Kubernetes 내에서 IAM 사용자에게 매핑할 사용자 이름. 기본적으로 사용자 이름은 IAM 사용자의 ARN입니다.
  - groups: Kubernetes 내에서 사용자가 매핑되는 그룹 목록. 자세한 내용은 Kubernetes 문서의 Default Roles and Role Bindings(기본 역할 및 역할 바인딩)를 참조하십시오.
- IAM 역할(예: 연합된 사용자) 추가: ConfigMap의 `mapRoles` 섹션에서 `data` 아래에 역할 세부 정보를 추가합니다. 파일에 이미 존재하지 않는 경우 이 섹션을 추가합니다. 각 항목은 다음 파라미터를 지원합니다.
  - rolearn: 추가할 IAM 역할의 ARN.
  - username: Kubernetes 내에서 IAM 역할에게 매핑할 사용자 이름. 기본적으로 사용자 이름은 IAM 역할의 ARN입니다.
  - groups: Kubernetes 내에서 역할이 매핑되는 그룹 목록. 자세한 내용은 Kubernetes 문서의 Default Roles and Role Bindings(기본 역할 및 역할 바인딩)를 참조하십시오.

예를 들어 아래 블록에는 다음이 포함됩니다.

- 작업자 노드가 클러스터에 등록할 수 있도록 작업자 노드 인스턴스 역할을 추가하는 `mapRoles` 섹션.

- 기본 AWS 계정의 AWS 사용자 `admin`이 포함된 `mapUsers` 섹션, 또 다른 AWS 계정의 `ops-user`. 두 사용자 모두 `system:masters` 그룹에 추가됩니다.

```
# 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: |
    - rolearn: arn:aws:iam::555555555555:role/devel-worker-nodes-
NodeInstanceRole-74RF4UBDUKL6
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
  mapUsers: |
    - userarn: arn:aws:iam::555555555555:user/admin
      username: admin
      groups:
        - system:masters
    - userarn: arn:aws:iam::111122223333:user/ops-user
      username: ops-user
      groups:
        - system:masters
```

4. 파일을 저장하고 텍스트 편집기를 종료합니다.

Amazon EKS 서비스 제한

다음 표는 AWS 계정에 대한 Amazon EKS의 기본 제한 사항을 보여주며, 이 제한 사항은 변경될 수 있습니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS 서비스 제한을 참조하십시오.

리소스	기본 한도
최대 Amazon EKS 클러스터 수	3

다음 표에는 Amazon EKS에 대한 변경 불가능한 제한이 나와 있습니다.

리소스	기본 제한
클러스터당 제어 플레인 보안 그룹의 최대 수(클러스터 생성 시 지정됨)	5

Amazon EKS IAM 정책, 역할 및 권한

기본적으로 IAM 사용자에게는 Amazon EKS 리소스를 생성 또는 수정하거나 Amazon EKS API를 사용하여 작업을 수행할 권한이 없습니다. Amazon EKS 콘솔이나 AWS CLI를 사용하더라도 마찬가지입니다. IAM 사용자에게 클러스터 생성 또는 수정을 허용하려면 IAM 사용자에게 필요한 특정 리소스 및 API 작업을 사용할 권한을 부여하는 IAM 정책을 생성하고, 해당 권한을 필요로 하는 IAM 사용자 또는 그룹에게 정책을 연결해야 합니다.

사용자 또는 사용자 그룹에 정책을 연결하면 지정된 리소스에 대해 지정된 작업을 수행할 권한이 허용되거나 거부됩니다. 자세한 내용은 IAM 사용 설명서에서 권한 및 정책을 참조하십시오. 사용자 지정 IAM 정책 관리 및 생성에 대한 자세한 내용은 IAM 정책 관리 단원을 참조하십시오.

마찬가지로 Amazon EKS는 다른 AWS 서비스를 호출하므로 서비스에 해당 자격 증명을 사용하여 인증을 받아야 합니다. 이 인증은 이러한 권한을 제공할 수 있는 IAM 역할 및 정책을 생성한 후 컴퓨팅 환경을 생성할 때 이 역할을 컴퓨팅 환경에 연결함으로써 수행됩니다. 자세한 내용은 Amazon EKS 서비스 IAM 역할 (p. 72) 및 IAM 사용 설명서의 IAM 역할도 참조하십시오.

시작하기

IAM 정책은 하나 이상의 Amazon EKS 작업을 사용할 권한을 허용하거나 거부해야 합니다.

항목

- 정책 구조 (p. 70)
- Amazon EKS IAM 정책 만들기 (p. 72)
- Amazon EKS 서비스 IAM 역할 (p. 72)

정책 구조

다음 항목에서는 IAM 정책의 구조에 대해 설명합니다.

항목

- 정책 구문 (p. 70)
- Amazon EKS 작업 (p. 71)
- 사용자에게 필요한 권한이 있는지 확인 (p. 71)

정책 구문

IAM 정책은 하나 이상의 명령문으로 구성된 JSON 문서입니다. 각 명령문의 구조는 다음과 같습니다.

```
{
  "Statement":[{
    "Effect":"effect",
    "Action":"action",
    "Resource":"arn",
    "Condition":{
      "condition":{
        "key":"value"
        }
      }
    }
  ]
}
```

명령문을 이루는 요소는 다양합니다.

- [Effect]: effect는 `Allow` 또는 `Deny`일 수 있습니다. 기본적으로 IAM 사용자에게는 리소스 및 API 작업을 사용할 권한이 없으므로 모든 요청이 거부됩니다. 명시적 허용은 기본 설정을 무시합니다. 명시적 거부는 모든 허용을 무시합니다.
- 작업: 이 작업은 권한을 부여하거나 거부할 특정 API 작업입니다.
- Resource: 작업의 영향을 받는 리소스입니다. Amazon EKS API 작업은 현재 리소스 수준 권한을 지원하지 않으므로, 모든 리소스에 작업이 적용될 수 있도록 와일드카드(*)를 사용하여 지정해야 합니다.
- Condition: Condition은 선택 사항으로서 정책이 적용되는 시점을 제어하는 데 사용할 수 있습니다.

IAM용 예제 Amazon EKS 정책 명령문에 대한 자세한 내용은 Amazon EKS IAM 정책 만들기 (p. 72) 섹션을 참조하십시오.

Amazon EKS 작업

IAM 정책 명령문에는 IAM을 지원하는 모든 서비스의 모든 API 작업을 지정할 수 있습니다. Amazon EKS의 경우 `eks:` 접두사와 함께 API 작업 이름을 사용합니다. 예를 들면 `eks:CreateCluster` 및 `eks:DeleteCluster` 등입니다.

명령문 하나에 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": ["eks:action1", "eks:action2"]
```

와일드카드를 사용하여 여러 작업을 지정할 수도 있습니다. 예를 들어 다음과 같이 이름이 "Describe"로 시작되는 모든 작업을 지정할 수 있습니다.

```
"Action": "eks:Describe*"
```

모든 Amazon EKS API 작업을 지정하려면 다음과 같이 * 와일드카드를 사용합니다.

```
"Action": "eks:*"
```

사용자에게 필요한 권한이 있는지 확인

IAM 정책을 생성한 후에는 사용자에게 필요한 특정 API 작업 및 리소스를 사용할 권한이 제대로 부여되는지를 확인한 후에 정책을 실무에 적용하는 것이 좋습니다.

우선 테스트용으로 IAM 사용자를 생성하고 앞서 생성한 IAM 정책을 연결하여 사용자를 테스트합니다. 그런 다음 테스트 사용자 자격으로 요청을 수행합니다. 콘솔 또는 AWS CLI에서 테스트 요청을 수행할 수 있습니다.

> **Note**
>
> IAM 정책 시뮬레이터로도 정책을 테스트할 수 있습니다. 정책 시뮬레이터에 대한 자세한 내용은 IAM 사용 설명서의 IAM 정책 시뮬레이터 작업을 참조하십시오.

정책이 사용자에게 정상적으로 권한을 부여하지 못하거나 권한을 과도하게 부여하는 경우, 원하는 결과가 나올 때까지 정책을 조정하고 다시 테스트할 수 있습니다.

> **Important**
>
> 변경된 정책이 전파되어 효력을 발휘하려면 몇 분이 걸릴 수 있습니다. 따라서 정책을 업데이트한 경우 5분간 기다린 후에 테스트하는 것이 좋습니다.

요청 시 권한 부여 확인에 실패하면 진단 정보가 포함된 인코딩 메시지가 반환됩니다. `DecodeAuthorizationMessage` 작업을 사용하여 메시지를 디코딩할 수 있습니다. 자세한 내용은

AWS Security Token Service API Reference의 DecodeAuthorizationMessage 및 AWS CLI Command Reference의 decode-authorization-message 단원을 참조하십시오.

Amazon EKS IAM 정책 만들기

특정 IAM 정책을 생성하여 계정의 사용자가 액세스할 수 있는 호출 및 리소스를 제한한 다음, 해당 정책을 IAM 사용자에게 연결합니다.

사용자 또는 사용자 그룹에 정책을 연결하면 지정된 리소스에 대해 지정된 작업을 수행할 권한이 허용되거나 거부됩니다. 자세한 내용은 IAM 사용 설명서에서 권한 및 정책을 참조하십시오. 사용자 지정 IAM 정책 관리 및 생성에 대한 자세한 내용은 IAM 정책 관리 단원을 참조하십시오.

IAM 사용자에게 관리 권한이 없는 경우 해당 사용자가 Amazon EKS API 작업을 호출하는 권한을 명시적으로 추가해야 합니다.

Amazon EKS 관리 사용자를 위한 IAM 정책을 생성하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Policies(정책)를 선택한 후 Create policy(정책 생성)를 선택합니다.
3. JSON 탭에 다음 정책을 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*"
            ],
            "Resource": "*"
        }
    ]
}
```

4. [Review policy]를 선택합니다.
5. Name 필드에 `AmazonEKSAdminPolicy`와 같은 사용자 고유의 이름을 입력합니다.
6. Create Policy를 선택하여 완료합니다.

IAM 정책을 사용자에 연결하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Users를 선택한 다음 정책을 연결하려는 사용자를 선택합니다.
3. [Permissions]를 선택하고, [Add permissions]를 선택합니다.
4. [Grant permissions] 섹션에서 [Attach existing policies directly]를 선택합니다.
5. 이전 절차에서 생성한 사용자 지정 정책을 선택한 후 [Next: Review]를 선택합니다.
6. 세부 정보를 검토한 후 [Add permissions]를 선택하여 종료합니다.

Amazon EKS 서비스 IAM 역할

Amazon EKS는 사용자를 대신하여 다른 AWS 서비스를 호출하여 서비스에 사용하는 리소스를 관리합니다. 서비스를 사용하려면 먼저 다음 IAM 정책을 사용하여 IAM 역할을 만들어야 합니다.

- `AmazonEKSServicePolicy`

- AmazonEKSClusterPolicy

기존 AWSServiceRoleForAmazonEKS 역할 확인

다음 절차를 사용하여 계정에 이미 Amazon EKS 서비스 역할이 있는지 확인할 수 있습니다.

IAM 콘솔에서 **AWSServiceRoleForAmazonEKS**를 확인하려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. 탐색 창에서 Roles를 선택합니다.
3. 역할 목록에서 AWSServiceRoleForAmazonEKS을 검색합니다. 역할이 존재하지 않을 경우, AWSServiceRoleForAmazonEKS 역할 만들기 (p. 73)을 참조하여 역할을 생성합니다. 역할이 존재하지 않을 경우, 연결된 정책을 볼 역할을 선택합니다.
4. Permissions를 선택합니다.
5. AmazonEKSServicePolicy 및 AmazonEKSClusterPolicy 관리형 정책이 역할에 연결되어 있는지 확인합니다. 정책이 연결된 경우, Amazon EKS 서비스 역할이 적절히 구성된 것입니다.
6. Trust Relationships와 Edit Trust Relationship을 차례로 선택합니다.
7. 신뢰 관계에 다음 정책이 포함되어 있는지 확인합니다. 신뢰 관계가 아래 정책과 일치하는 경우, Cancel을 선택합니다. 신뢰 관계가 일치하지 않는 경우, 정책을 Policy Document 창에 복사하고 Update Trust Policy를 선택합니다.

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

AWSServiceRoleForAmazonEKS 역할 만들기

계정에 Amazon EKS 서비스 역할이 없는 경우 다음 절차를 사용하여 새로 만들 수 있습니다.

IAM 콘솔에서 Amazon EKS 서비스 역할을 만들려면

1. https://console.aws.amazon.com/iam/에서 IAM 콘솔을 엽니다.
2. [Roles]를 선택한 다음 [Create role]을 선택합니다.
3. 서비스 목록에서 EKS를 선택한 다음 사용 사례에 대해 Allows Amazon EKS to manage your clusters on your behalf(EKS에서 사용자를 대신하여 클러스터를 관리하도록 허용)를 선택하고 Next: Permissions(다음: 권한)를 선택합니다.
4. [Next: Review]를 선택합니다.
5. Role name(역할 이름)에서 역할에 대한 고유 이름(예: eksServiceRole)을 입력한 다음 Create role(역할 생성)을 선택합니다.

AWS CloudFormation으로 Amazon EKS 서비스 역할을 생성하려면

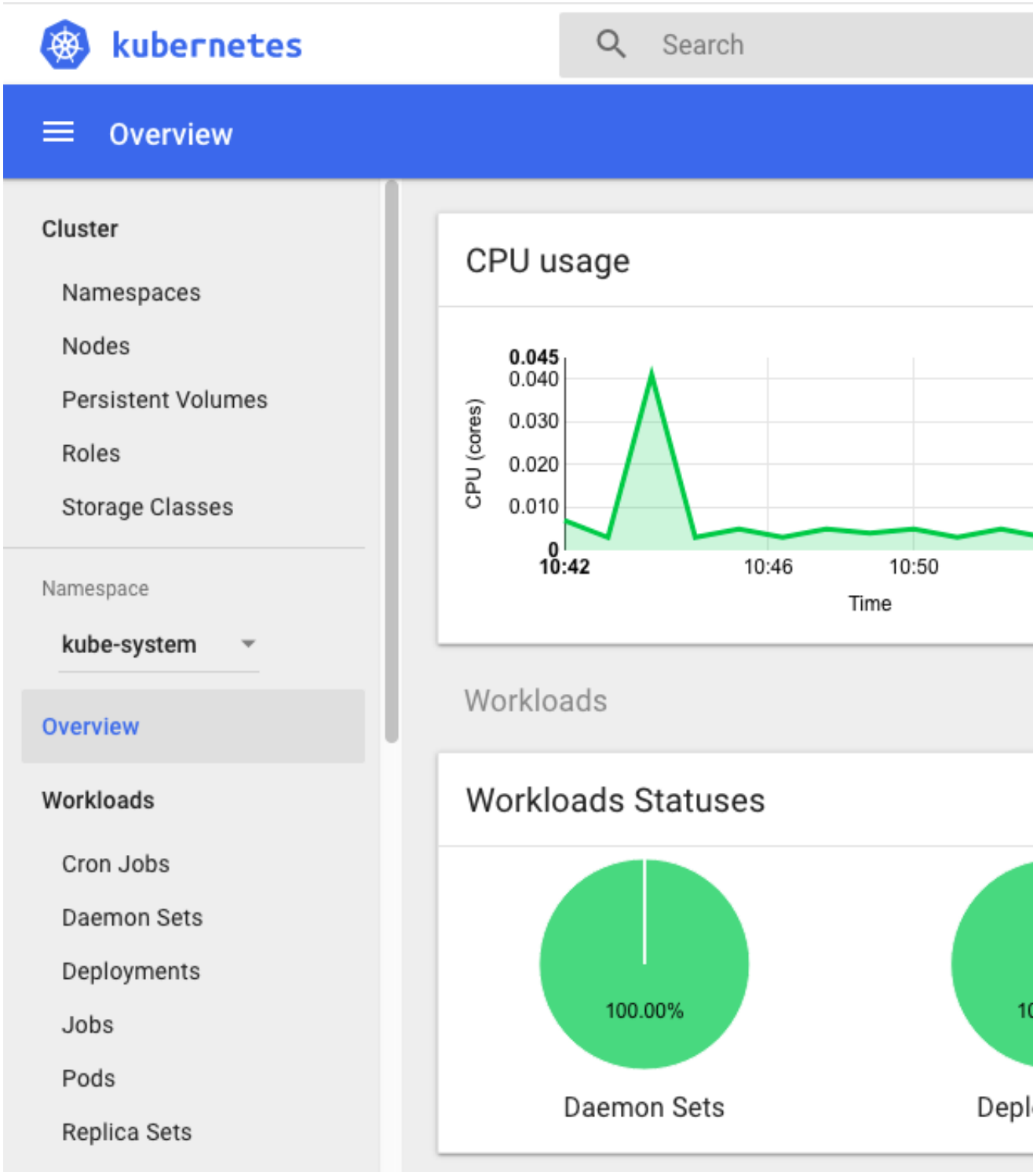
1. 다음 AWS CloudFormation 템플릿을 로컬 시스템의 텍스트 파일에 저장합니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: 'Amazon EKS Service Role'


Resources:

  AWSServiceRoleForAmazonEKS:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - eks.amazonaws.com
          Action:
          - sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonEKSServicePolicy
        - arn:aws:iam::aws:policy/AmazonEKSClusterPolicy

Outputs:

  RoleArn:
    Description: The role that EKS will use to create AWS resources for Kubernetes
 clusters
    Value: !GetAtt AWSServiceRoleForAmazonEKS.Arn
    Export:
      Name: !Sub "${AWS::StackName}-RoleArn"
```

2. https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
3. [Create stack]을 선택합니다.
4. 템플릿 선택에서 Amazon S3에 템플릿 업로드를 선택한 후 찾아보기를 선택합니다.
5. 이전에 생성한 파일을 선택한 후 다음을 선택합니다.
6. 스택 이름의 경우 `eksServiceRole`과 같은 역할 이름을 입력하고 다음을 선택합니다.
7. [Options] 페이지에서 [Next]를 선택합니다.
8. 검토 페이지에서 정보를 검토하고, 스택이 IAM 리소스를 생성할 수 있음을 인지한 다음 생성을 선택합니다.

자습서: Kubernetes 웹 UI 배포(대시보드)

이 자습서는 Amazon EKS 클러스터에 Kubernetes 대시보드를 배포하고 CPU 및 메모리 지표로 완료하는 방법을 안내합니다. 또한 대시보드에 보안 연결하여 클러스터를 보고 제어하는 데 사용할 수 있는 Amazon EKS 관리자 서비스 계정 생성에 도움이 됩니다.



사전 요구 사항

이 자습서는 다음과 같이 가정합니다.

- Amazon EKS 시작하기 (p. 3)의 단계에 따라 Amazon EKS 클러스터를 생성했습니다.
- 제어 플레인 탄력적 네트워크 인터페이스 및 작업자 노드에 대한 보안 그룹은 클러스터 보안 그룹 고려 사항 (p. 41)의 권장 설정을 따릅니다.
- Amazon EKS 클러스터를 사용하여 통신하도록 구성된 (p. 8) kubectl 클라이언트를 사용 중입니다.

1단계: 대시보드 배포

다음 단계를 따라 Kubernetes 대시보드, `heapster` 및 CPU와 메모리 지표에 대한 `influxdb` 백엔드를 클러스터에 배포합니다.

Kubernetes 대시보드를 배포하려면

1. 클러스터에 Kubernetes 대시보드 배포:

   ```
   kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml
   ```

   결과:

   ```
   secret "kubernetes-dashboard-certs" created
   serviceaccount "kubernetes-dashboard" created
   role "kubernetes-dashboard-minimal" created
   rolebinding "kubernetes-dashboard-minimal" created
   deployment "kubernetes-dashboard" created
   service "kubernetes-dashboard" created
   ```

2. `heapster`를 배포하여 컨테이너 클러스터 모니터링 및 클러스터의 성능 분석 활성화:

   ```
   kubectl apply -f https://raw.githubusercontent.com/kubernetes/heapster/master/deploy/kube-config/influxdb/heapster.yaml
   ```

   Note

   `heapster`가 사용되지 않는 경우에도 현재 Kubernetes 대시보드에 유일하게 지표를 제공합니다. 자세한 내용은 https://github.com/kubernetes/dashboard/issues/2986을 참조하십시오.

   결과:

   ```
   serviceaccount "heapster" created
   deployment "heapster" created
   service "heapster" created
   ```

3. `heapster`에 대한 `influxdb` 백엔드를 클러스터에 배포:

   ```
   kubectl apply -f https://raw.githubusercontent.com/kubernetes/heapster/master/deploy/kube-config/influxdb/influxdb.yaml
   ```

   결과:

   ```
   deployment "monitoring-influxdb" created
   service "monitoring-influxdb" created
   ```

4. 대시보드에 대한 `heapster` 클러스터 역할 바인딩 생성:

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/heapster/master/deploy/
kube-config/rbac/heapster-rbac.yaml
```

결과:

```
clusterrolebinding "heapster" created
```

2단계: eks-admin 서비스 계정 및 클러스터 역할 바인딩 생성

기본적으로 Kubernetes 대시보드 사용자의 권한은 제한되어 있습니다. 이 단원에서는 eks-admin 서비스 계정 및 클러스터 역할 바인딩을 생성하고, 이를 사용하여 관리자 수준 권한으로 대시보드에 보안 연결할 수 있습니다. 자세한 내용은 Kubernetes 문서의 Managing Service Accounts(서비스 계정 관리)를 참조하십시오.

eks-admin 서비스 계정 및 클러스터 역할 바인딩을 생성하려면

Important

이 절차를 사용하여 생성된 예제 서비스 계정은 클러스터에 대해 완전한 cluster-admin(수퍼유저) 권한을 갖습니다. 자세한 내용은 Kubernetes 문서의 Using RBAC Authorization(RBAC 승인 사용)을 참조하십시오.

1. 아래 텍스트가 포함된 eks-admin-service-account.yaml이라는 파일을 생성합니다. 이 매니페스트는 eks-admin이라는 서비스 계정 및 클러스터 역할 바인딩을 정의합니다.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eks-admin
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: eks-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: eks-admin
  namespace: kube-system
```

2. 서비스 계정 및 클러스터 역할 바인딩을 클러스터에 적용합니다.

```
kubectl apply -f eks-admin-service-account.yaml
```

결과:

```
serviceaccount "eks-admin" created
clusterrolebinding.rbac.authorization.k8s.io "eks-admin" created
```

3단계: 대시보드에 연결

클러스터에 Kubernetes 대시보드가 배포되었고, 클러스터를 보고 제어하는 데 사용할 수 있는 관리자 서비스 계정을 가졌기 때문에 이제 이 서비스 계정을 사용하여 대시보드에 연결할 수 있습니다.

Kubernetes 대시보드에 연결하려면

1. `eks-admin` 서비스 계정에 대한 인증 토큰을 조회합니다. 출력에서 *<authentication_token>* 값을 복사합니다. 이 토큰을 사용하여 대시보드에 연결합니다.

   ```
   kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep eks-admin | awk '{print $1}')
   ```

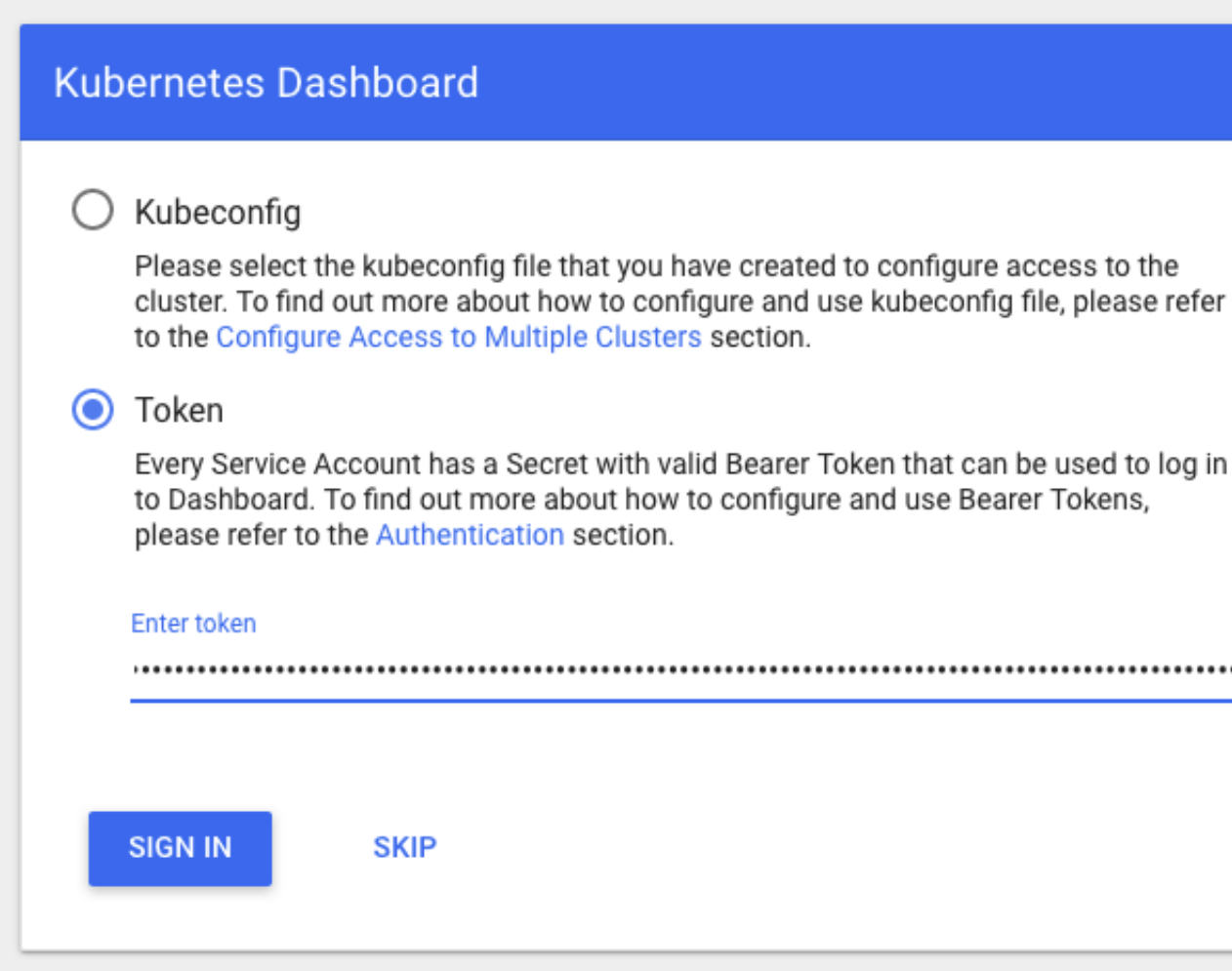
   결과:

   ```
   Name:         eks-admin-token-b5zv4
   Namespace:    kube-system
   Labels:       <none>
   Annotations:  kubernetes.io/service-account.name=eks-admin
                 kubernetes.io/service-account.uid=bcfe66ac-39be-11e8-97e8-026dce96b6e8

   Type:  kubernetes.io/service-account-token

   Data
   ====
   ca.crt:     1025 bytes
   namespace:  11 bytes
   token:      <authentication_token>
   ```

2. kubectl proxy를 시작합니다.

   ```
   kubectl proxy
   ```

3. 웹 브라우저로 다음 링크를 열어 대시보드 엔드포인트에 액세스합니다. http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#!/login
4. 토큰을 선택하고, 이전 명령의 *<authentication_token>* 출력을 토큰 필드에 복사한 다음 SIGN IN(로그인)을 선택합니다.



Note

대시보드에 CPU 및 메모리 지표가 표시되기까지 몇 분이 걸릴 수 있습니다.

4단계: 다음 절차

Kubernetes 클러스터 대시보드에 연결한 이후 `eks-admin` 서비스 계정을 사용하여 클러스터를 보고 제어할 수 있습니다. 대시보드 사용에 대한 자세한 내용은 GitHub 프로젝트 문서를 참조하십시오.

자습서: Amazon EKS 클러스터에 대해 퍼블릭 및 프라이빗 서브넷이 있는 VPC 생성

이 자습서에서는 NAT 게이트웨이를 통해 인터넷 액세스를 제공하는 퍼블릭 서브넷 두 개와 프라이빗 서브넷 두 개가 있는 VPC를 생성하는 방법을 안내합니다. Amazon EKS 클러스터에 대한 VPC를 사용할 수 있습니다. 내부에서 퍼블릭 로드 밸런서를 생성하려면 작업자 노드에 대해 프라이빗 서브넷을 사용하고 Kubernetes에 대해 퍼블릭 서브넷을 사용하는 네트워크 아키텍처를 권장합니다.

항목

- 1단계: NAT 게이트웨이에 대한 탄력적 IP 주소 만들기 (p. 81)
- 2단계: VPC 마법사 실행 (p. 81)
- 3단계: 추가 서브넷 생성 (p. 82)
- 4단계: 프라이빗 서브넷 태그 지정 (p. 82)
- 5단계: 제어 플레인 보안 그룹 생성 (p. 83)
- 다음 단계 (p. 83)

1단계: NAT 게이트웨이에 대한 탄력적 IP 주소 만들기

프라이빗 서브넷의 작업자 노드에는 아웃바운드 인터넷 액세스를 위한 NAT 게이트웨이가 필요합니다. NAT 게이트웨이에는 퍼블릭 서브넷의 탄력적 IP 주소가 필요하지만, VPC 마법사에서는 이를 만들어 주지 않습니다. VPC 마법사를 실행하기 전에 탄력적 IP 주소를 만듭니다.

탄력적 IP 주소를 만들려면

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 왼쪽 탐색 창에서 [Elastic IPs]를 선택합니다.
3. [Allocate new address], [Allocate], [Close]를 선택합니다.
4. 새로 만든 탄력적 IP 주소의 [Allocation ID]를 적어 둡니다. 나중에 VPC 마법사에서 이를 입력합니다.

2단계: VPC 마법사 실행

VPC 마법사는 대부분의 VPC 리소스를 자동으로 만들고 구성해 줍니다.

VPC 마법사를 실행하려면

1. 왼쪽 탐색 창에서 [VPC Dashboard]를 선택합니다.
2. [Start VPC Wizard], [VPC with Public and Private Subnets], [Select]를 선택합니다.
3. [VPC name]에서 VPC에 고유 이름을 부여합니다.
4. [Elastic IP Allocation ID]에서 앞서 만든 탄력적 IP 주소의 ID를 선택합니다.

5. [Create VPC]를 선택합니다.
6. 마법사를 마치면 [OK]를 선택합니다. VPC 서브넷이 생성된 가용 영역을 적어 둡니다. 추가 서브넷은 다른 가용 영역에 생성해야 합니다.

3단계: 추가 서브넷 생성

마법사가 단일 가용 영역에서 퍼블릭 서브넷과 프라이빗 서브넷이 하나씩 있는 VPC를 만듭니다. 가용성을 높이려면 다른 가용 영역에 각 서브넷 유형을 하나 이상 만들어 VPC에 두 영역에 걸쳐 퍼블릭 및 프라이빗 서브넷이 모두 있도록 해야 합니다.

프라이빗 서브넷을 추가로 생성하려면

1. 왼쪽 탐색 창에서 [Subnets], [Create Subnet]을 선택합니다.
2. [Name tag]에 서브넷 이름(예: Private subnet)을 입력합니다.
3. [VPC]에서 앞서 만든 VPC를 선택합니다.
4. 가용 영역에서 VPC의 원래 서브넷과 다른 영역을 선택합니다.
5. [IPv4 CIDR block]에 유효한 CIDR 블록을 입력합니다. 예를 들어 마법사는 기본적으로 10.0.0.0/24 및 10.0.1.0/24에 CIDR 블록을 만듭니다. 두 번째 프라이빗 서브넷에 10.0.3.0/24를 사용할 수 있습니다.
6. [Yes, Create]를 선택합니다.

퍼블릭 서브넷을 추가로 생성하려면

1. 왼쪽 탐색 창에서 [Subnets], [Create Subnet]을 선택합니다.
2. [Name tag]에 서브넷 이름(예: Public subnet)을 입력합니다.
3. [VPC]에서 앞서 만든 VPC를 선택합니다.
4. 가용 영역에서 이전 절차에서 생성한 추가 프라이빗 서브넷과 동일한 영역을 선택합니다.
5. [IPv4 CIDR block]에 유효한 CIDR 블록을 입력합니다. 예를 들어 마법사는 기본적으로 10.0.0.0/24 및 10.0.1.0/24에 CIDR 블록을 만듭니다. 두 번째 퍼블릭 서브넷에 10.0.2.0/24를 사용할 수 있습니다.
6. [Yes, Create]를 선택합니다.
7. 방금 생성한 퍼블릭 서브넷을 선택하고, [Route Table], [Edit]를 선택합니다.
8. 기본적으로 프라이빗 라우팅 테이블이 선택되어 있습니다. 0.0.0.0/0 대상이 인터넷 게이트웨이(igw-**xxxxxxxx**)로 라우팅되도록 사용 가능한 다른 라우팅 테이블을 선택하고, [Save]를 선택합니다.
9. 두 번째 퍼블릭 서브넷이 여전히 선택된 상태에서 [Subnet Actions], [Modify auto-assign IP settings]를 선택합니다.
10. [Enable auto-assign public IPv4 address]를 선택하고, [Save], [Close]를 선택합니다.

4단계: 프라이빗 서브넷 태그 지정

Kubernetes가 내부 로드 밸런서에 VPC의 프라이빗 서브넷을 사용할 수 있음을 알 수 있도록 태그가 지정되어야 합니다.

키	값
`kubernetes.io/role/internal-elb`	`1`

프라이빗 서브넷에 태그를 지정하려면

1. 프라이빗 서브넷 중 하나를 선택하고 태그, 태그 추가/편집을 선택합니다.

2. 태그 생성을 선택합니다.
3. 키에 `kubernetes.io/role/internal-elb`를 입력합니다.
4. 값에 `1`를 입력합니다.
5. 저장을 선택하고, VPC의 다른 프라이빗 서브넷에 대해서도 이 절차를 반복합니다.

5단계: 제어 플레인 보안 그룹 생성

Amazon EKS 클러스터를 생성할 때 클러스터 제어 플레인은 서브넷에 탄력적 네트워크 인터페이스를 생성하여 작업자 노드와의 통신이 가능하도록 합니다. Amazon EKS 클러스터 제어 플레인 전용 보안 그룹을 생성해야 인바운드 및 아웃바운드 규칙을 적용하여 연결 간 허용되는 트래픽을 제어할 수 있습니다. 클러스터를 생성할 때 이 보안 그룹을 지정하고, 서브넷에 생성된 탄력적 네트워크 인터페이스에 적용됩니다.

3단계: Amazon EKS 작업자 노드 시작 및 구성 (p. 9)에서 사용되는 작업자 노드 AWS CloudFormation 템플릿은 작업자 노드 보안 그룹을 생성하고, 필요한 규칙을 적용하여 제어 플레인과의 자동 통신을 허용합니다. 하지만 해당 템플릿에서 스택을 생성할 때 제어 플레인 보안 그룹을 지정해야 합니다.

제어 플레인 보안 그룹을 생성하려면

1. 왼쪽 탐색 창의 Filter by VPC(VPC로 필터링)에서 VPC를 선택하고, 보안 그룹, Create Security Group(보안 그룹 생성)을 선택합니다.

   Note

   새 VPC가 보이지 않는 경우 페이지를 새로 고치십시오.
2. 다음 필드를 입력하고 Yes, Create(예, 생성합니다)를 선택합니다.
   - Name tag(이름 태그)에 보안 그룹의 이름을 입력합니다. 예, *<cluster-name>*`-control-plane`.
   - Description(설명)에서 나중에 식별하는 데 도움이 되도록 보안 그룹의 설명을 입력합니다.
   - VPC에서 Amazon EKS 클러스터에 대해 사용 중인 VPC를 선택합니다.

다음 단계

VPC를 생성한 이후 Amazon EKS 시작하기 (p. 3) 연습을 시도할 수 있습니다. 하지만 Amazon EKS 클러스터 VPC 생성 (p. 3) 단원을 건너뛰고 클러스터에 대해 이러한 서브넷 및 보안 그룹을 사용할 수 있습니다.

AWS CloudTrail을 사용하여 Amazon EKS API 호출 로깅

Amazon EKS은 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 호출이 포함됩니다. 추적을 생성하면 Amazon EKS에 대한 이벤트를 비롯하여 CloudTrail 이벤트를 Amazon S3 버킷으로 지속적으로 배포할 수 있습니다. 추적을 구성하지 않은 경우 이벤트 기록에서 CloudTrail 콘솔의 최신 이벤트를 볼 수도 있습니다. CloudTrail에서 수집하는 정보를 사용하여 Amazon EKS에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail User Guide을 참조하십시오.

CloudTrail의 Amazon EKS 정보

CloudTrail은 계정 생성 시 AWS 계정에서 활성화됩니다. Amazon EKS에서 활동이 수행되면 해당 활동은 이벤트 기록에서 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 CloudTrail 이벤트 기록에서 이벤트 보기를 참조하십시오.

Amazon EKS 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려는 경우 추적을 생성합니다. 추적은 CloudTrail이 Amazon S3 버킷으로 로그 파일을 전송할 수 있도록 합니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또는 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하십시오.

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

모든 Amazon EKS 작업이 CloudTrail에서 로깅되고 Amazon EKS API Reference에 문서화됩니다. 예를 들어 `CreateCluster`, `ListClusters` 및 `DeleteCluster` 섹션을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

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

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

자세한 정보는 CloudTrail userIdentity 요소를 참조하십시오.

Amazon EKS 로그 파일 항목 이해

추적은 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 제공할 수 있도록 해 주는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함됩니다. 이벤트는 어떤 소스로부터의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 포함되어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문 스택 추적이 아니므로 특정 순서로 표시되지 않습니다.

다음은 `CreateCluster` 작업을 보여 주는 CloudTrail 로그 항목이 표시된 예제입니다.


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/ericn",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "ericn"
  },
  "eventTime": "2018-05-28T19:16:43Z",
  "eventSource": "eks.amazonaws.com",
  "eventName": "CreateCluster",
  "awsRegion": "us-west-2",
  "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:us-west-2:111122223333: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 공동 책임 모델

보안과 규정 준수는 AWS와 고객의 공동 책임입니다. 이 공유 모델을 통해 고객은 운영 부담을 덜 수 있습니다. AWS가 호스트 운영 체제 및 가상화 계층에서 서비스 운영 시설의 물리적 보안에 이르기까지 구성 요소를 운영, 관리, 제어하기 때문입니다. 자세한 내용은 공식 AWS 공동 책임 모델 세부 정보 페이지를 참조하십시오.

AWS는 AWS 클라우드에 제공된 모든 서비스를 실행하는 인프라를 보호해야 합니다. 이 인프라는 AWS 클라우드 서비스를 실행하는 하드웨어, 소프트웨어, 네트워킹 및 시설로 구성됩니다. Amazon EKS의 경우, AWS가 제어 플레인 노드와 `etcd` 데이터베이스를 포함한 Kubernetes 제어 플레인을 책임집니다.

사용자는 다음 사항을 책임지고 관리합니다.

- Amazon EKS 제어 플레인에서 고객 VPC로 트래픽을 전달할 수 있는 보안 그룹의 구성을 포함한 데이터 영역의 보안 구성
- 작업자 노드와 컨테이너 자체의 구성
- 작업자 노드 게스트 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 ID 및 액세스 관리

Amazon EKS와 AWS IAM을 함께 사용하는 방법에 대한 자세한 내용은 Amazon EKS IAM 정책, 역할 및 권한 (p. 70) 단원을 참조하십시오.

Amazon EKS 클러스터를 이용한 기본 Kubernetes 역할 기반 액세스 제어(RBAC) 관리에 대한 자세한 내용은 클러스터 인증 관리 (p. 58)를 참조하십시오.

Amazon EKS는 사용자, 역할 또는 Amazon EKS의 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail와 통합됩니다. CloudTrail는 Amazon EKS에 대한 API 호출을 모두 이벤트로 캡처합니다. 자세한 내용은 AWS CloudTrail을 사용하여 Amazon EKS API 호출 로깅 (p. 84) 단원을 참조하십시오.

Amazon EKS 문제 해결

이 장에서는 Amazon EKS를 사용하는 동안 발생할 수 있는 몇 가지 일반 오류와 이를 해결하는 방법을 다룹니다.

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 us-east-1d, the targeted
availability zone, does not currently have sufficient capacity to support
the cluster. Retry and choose from these availability zones: us-east-1a, us-
east-1b, us-east-1c
```

이 오류 메시지를 반환한 가용 영역에서 호스팅하는 클러스터 VPC의 서브넷을 사용하여 클러스터를 재생성하십시오.

aws-iam-authenticator 찾을 수 없음

`"aws-iam-authenticator": executable file not found in $PATH` 오류가 표시되는 경우 Amazon EKS에 대해 kubectl이 구성되지 않았습니다. 자세한 내용은 aws-iam-authenticator 설치 (p. 61) 단원을 참조하십시오.

작업자 노드와 클러스터 조인 실패

작업자 노드의 클러스터 조인을 막는 2가지 일반적인 원인이 있습니다.

- `aws-auth-cm.yaml` 파일에 작업자 노드에 대한 올바른 IAM 역할 ARN이 없습니다. 작업자 노드 IAM 역할 ARN(인스턴스 프로파일 ARN이 아님)이 `aws-auth-cm.yaml` 파일에서 지정되어야 합니다. 자세한 내용은 Amazon EKS 작업자 노드 시작 (p. 28) 단원을 참조하십시오.
- 작업자 노드 AWS CloudFormation 템플릿의 ClusterName이 작업자 노드가 조인하고자 하는 클러스터의 이름과 정확히 일치하지 않습니다. 이 필드에 올바르지 않은 값을 전달하면 작업자 노드의 `/var/lib/kubelet/kubeconfig` 파일의 올바르지 않은 구성이 발생하고, 노드가 클러스터에 조인하지 않습니다.

권한이 없거나 액세스가 거부됨

kubectl 명령 실행 시 다음 오류 중 하나가 표시되는 경우, Amazon EKS에 대해 kubectl이 올바르게 구성되지 않았음을 의미합니다.

- `could not get token: AccessDenied: Access denied`
- `error: You must be logged in to the server (Unauthorized)`

이는 클러스터가 AWS 자격 증명 집합(IAM 사용자 또는 역할에서)을 사용해 생성되었고 kubectl은 다른 자격 증명 집합을 사용하고 있기 때문입니다.

Amazon EKS 클러스터가 생성될 때 클러스터를 생성하는 IAM 엔터티(사용자 또는 역할)가 Kubernetes RBAC 권한 부여 표에 관리자(`system:master` 권한이 있음)로 추가됩니다. 처음에는 해당 IAM 사용자만이 kubectl을 사용하여 Kubernetes API를 호출할 수 있습니다. 자세한 정보는 클러스터의 사용자 또는 IAM 역할 관리 (p. 65) 단원을 참조하십시오. 또한 Kubernetes용 AWS IAM Authenticator에서는 Go용 AWS SDK을 사용하여 Amazon EKS 클러스터에 대한 인증을 수행합니다. 콘솔을 사용하여 클러스터를 생성하는 경우 클러스터에서 kubectl 명령을 실행할 때 동일한 IAM 사용자 자격 증명이 AWS SDK 자격 증명 체인에 있어야 합니다.

AWS CLI를 설치하고 구성하면 사용자에 대한 IAM 자격 증명을 구성할 수 있습니다. 이 자격 증명은 Kubernetes용 AWS IAM Authenticator에 대해서도 유효합니다. AWS CLI가 사용자에 대해 적절하게 구성된 경우 Kubernetes용 AWS IAM Authenticator에서도 이 자격 증명을 찾을 수 있습니다. 자세한 내용은 AWS CLI의 AWS Command Line Interface 사용 설명서 구성을 참조하십시오.

Amazon EKS 클러스터를 생성하는 역할을 부여할 경우 kubectl이 동일한 역할을 담당할 수 있도록 구성해야 합니다. 다음 명령을 실행해 kubeconfig 파일을 업데이트하여 IAM 역할을 사용합니다. 자세한 내용은 Amazon EKS에 대한 `kubeconfig` 생성 (p. 62) 단원을 참조하십시오.

```
aws --region region eks update-kubeconfig --name cluster_name --role-arn
 arn:aws:iam::aws_account_id:role/role_name
```

hostname doesn't match

시스템의 Python 버전이 Python 3 또는 Python 2.7.9 이상이어야 합니다. 그렇지 않은 경우 Amazon EKS에 대한 AWS CLI 호출과 함께 `hostname doesn't match` 오류가 표시됩니다. 자세한 내용은 Python 요청 FAQ의 What are "hostname doesn't match" errors?를 참조하십시오.

CNI 로그 수집 도구

Kubernetes용 Amazon VPC CNI 플러그인의 경우, 지원 사례 및 일반적인 문제 해결을 위한 진단 로그를 수집하는 데 사용할 수 있는 자체 문제 해결 스크립트(`/opt/cni/bin/aws-cni-support.sh`의 작업자 노드에서 사용 가능)가 있습니다.

이 스크립트는 다음 진단 정보를 수집합니다.

- L-IPAMD 내부 검사 데이터
- 측정치
- Kubelet 내부 검사 데이터
- `ifconfig` 출력
- `ip rule show` 출력
- `iptables-save` 출력
- `iptables -nvL` 출력
- `iptables -nvL -t nat` 출력
- CNI 구성의 덤프
- Kubelet 로그
- 저장된 `/var/log/messages`
- 작업자 노드의 라우팅 테이블 정보(`ip route`를 통해)
- `/proc/sys/net/ipv4/conf/{all,default,eth0}/rp_filter`의 `sysctls` 출력

다음 명령을 사용하여 작업자 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 https://raw.githubusercontent.com/aws/amazon-vpc-cni-k8s/master/scripts/aws-cni-support.sh | sudo bash
```

진단 정보는 `/var/log/aws-routed-eni/aws-cni-support.tar.gz`에 수집되고 저장됩니다.

Amazon EKS 문서 기록

다음 표에서 Amazon EKS 사용 설명서의 중요한 업데이트 및 새 기능이 나와 있습니다. 사용자로부터 받은 의견을 수렴하기 위해 설명서가 자주 업데이트됩니다.

update-history-change	update-history-description	update-history-date
새롭게 CVE-2019-5736용으로 패치된 Amazon EKS 최적화 AMI	CVE-2019-5736에 설명된 취약성을 해결하기 위해 Amazon EKS가 Amazon EKS 최적화 AMI를 업데이트했습니다.	February 11, 2019
Amazon EKS 리전 확장 (p. 90)	이제 아시아 태평양(서울)(`ap-northeast-2`) 리전에서 Amazon EKS를 사용할 수 있습니다.	January 9, 2019
ALAS2-2019-1141용으로 패치된 새로운 Amazon EKS 최적화 AMI	ALAS2-2019-1141에 참조된 CVE를 해결하기 위해 Amazon EKS가 Amazon EKS 최적화 AMI를 업데이트했습니다.	January 9, 2019
Amazon EKS 리전 확장 (p. 90)	이제 EU(프랑크푸르트)(`eu-central-1`), 아시아 태평양(도쿄)(`ap-northeast-1`), 아시아 태평양(싱가포르)(`ap-southeast-1`), 아시아 태평양(시드니)(`ap-southeast-2`) 등의 추가 리전에서 Amazon EKS를 사용할 수 있습니다.	December 19, 2018
Amazon EKS 클러스터 업데이트	Amazon EKS 클러스터 Kubernetes 버전 업데이트 및 작업자 노드 교체에 대한 설명서를 추가했습니다.	December 12, 2018
Amazon EKS 리전 확장 (p. 90)	이제 EU(스톡홀름)(`eu-north-1`) 리전에서 Amazon EKS를 사용할 수 있습니다.	December 11, 2018
Amazon EKS 플랫폼 버전 업데이트	CVE-2018-1002105를 해결하기 위해 Kubernet을 패치 수준 1.10.11로 업데이트하는 새로운 플랫폼 버전	December 4, 2018
Application Load Balancer 수신 컨트롤러에 대한 버전 1.0.0 지원 추가	Application Load Balancer 수신 컨트롤러는 AWS의 공식적인 지원과 함께 1.0.0을 릴리스합니다.	November 20, 2018
CNI 네트워크 구성에 대한 지원이 추가되었습니다.	이제 Kubernetes용 Amazon VPC CNI 플러그인 버전 1.2.1에서 보조 포드 네트워크 인터페이스에 대한 사용자 지정 네트워크 구성이 지원됩니다.	October 16, 2018

MutatingAdmissionWebhook 및 ValidatingAdmissionWebhook에 대한 지원이 추가되었습니다.	이제 Amazon EKS 플랫폼 버전 `1.10-eks.2`가 `MutatingAdmissionWebhook` 및 `ValidatingAdmissionWebhook` 승인 컨트롤러를 지원합니다.	October 10, 2018
추가된 파트너 AMI 정보	Canonical은 Amazon EKS와 협력하여 클러스터에서 사용할 수 있는 작업자 노드 AMI를 만들었습니다.	October 3, 2018
AWS CLI update-kubeconfig 명령에 대한 추가 지침	Amazon EKS는 클러스터에 액세스하는 데 필요한 `kubeconfig` 파일을 만드는 과정을 단순화하기 위해 `update-kubeconfig`를 AWS CLI에 추가했습니다.	September 21, 2018
새로운 Amazon EKS 최적화 AMI	Amazon EKS는 다양한 보안 픽스 및 AMI 최적화를 제공하기 위해 Amazon EKS 최적화 AMI(GPU 지원 유무는 상이)를 업데이트했습니다.	September 13, 2018
Amazon EKS 리전 확장 (p. 90)	Amazon EKS는 이제 EU(아일랜드)(`eu-west-1`) 지역에서 사용할 수 있습니다.	September 5, 2018
Amazon EKS 플랫폼 버전 업데이트	Kubernetes 집계 계층 및 Horizontal Pod Autoscaler(HPA)를 지원하는 새로운 플랫폼 버전.	August 31, 2018
새로운 Amazon EKS 최적화 AMI 및 GPU 지원	Amazon EKS는 새로운 AWS CloudFormation 작업자 노드 템플릿과 부트스트랩 스크립트를 사용하기 위해 Amazon EKS 최적화 AMI를 업데이트했습니다. 또한 새로운 Amazon EKS 최적화 AMI 및 GPU 지원도 이용할 수 있습니다.	August 22, 2018
ALAS2-2018-1058용의 새로운 Amazon EKS 최적화 AMI 패치	ALAS2-2018-1058에 참조된 CVE를 해결하기 위해 Amazon EKS가 Amazon EKS 최적화 AMI를 업데이트했습니다.	August 14, 2018
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.	July 10, 2018
Amazon EKS 최초 릴리스 (p. 90)	서비스 출시에 대한 최초 문서	June 5, 2018

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the AWS General Reference.