



# **AWS Builders Korea Program 200**

## **AWS 관리형 쿠버네티스 컨테이너 서비스 Amazon EKS로 애플리케이션 배포 및 운영하기**

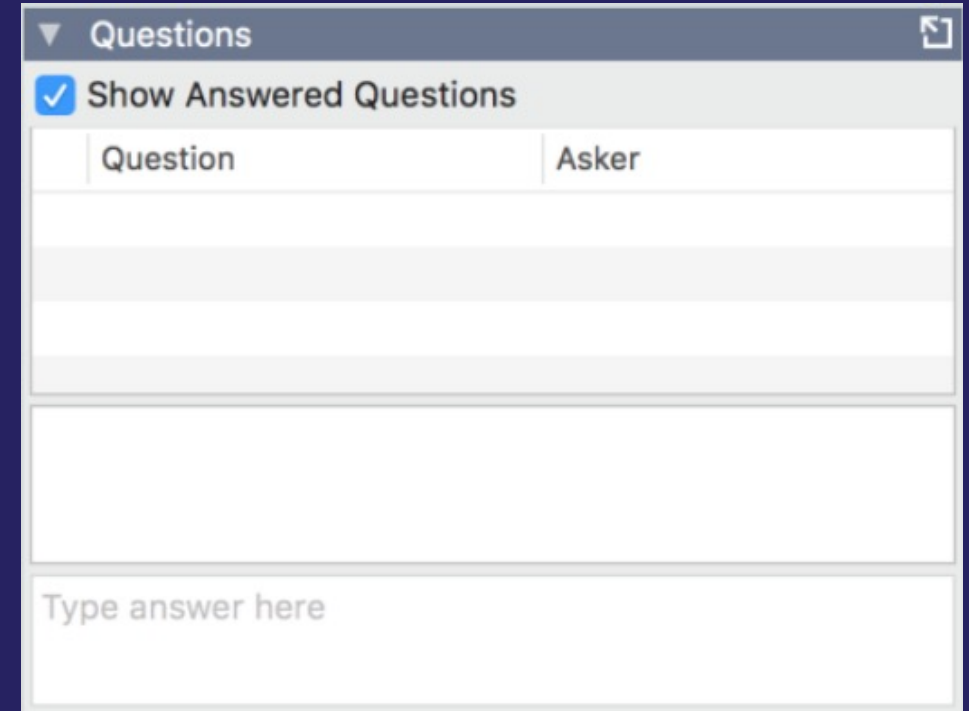
최 우 형 (whchoi@amazon.com)

Principal Solutions Architect / AWS



# 강연 중 질문하는 방법

- AWS Builders Go to Webinar “Questions” 창에 자신이 질문한 내역이 표시됩니다. 기본적으로 모든 질문은 공개로 답변됩니다만 본인만 답변을 받고 싶으면 (비공개)라고 하고 질문해 주시면 됩니다.



Questions

☒ Show Answered Questions

Question	Asker

Type answer here

## 고지 사항 (Disclaimer)

본 콘텐츠는 고객의 편의를 위해 AWS 서비스 설명을 위해 온라인 세미나용으로 별도로 제작, 제공된 것입니다. 만약 AWS 사이트와 콘텐츠 상에서 차이나 불일치가 있을 경우, AWS 사이트([aws.amazon.com](https://aws.amazon.com))가 우선합니다. 또한 AWS 사이트 상에서 한글 번역문과 영어 원문에 차이나 불일치가 있을 경우(번역의 지체로 인한 경우 등 포함), 영어 원문이 우선합니다.

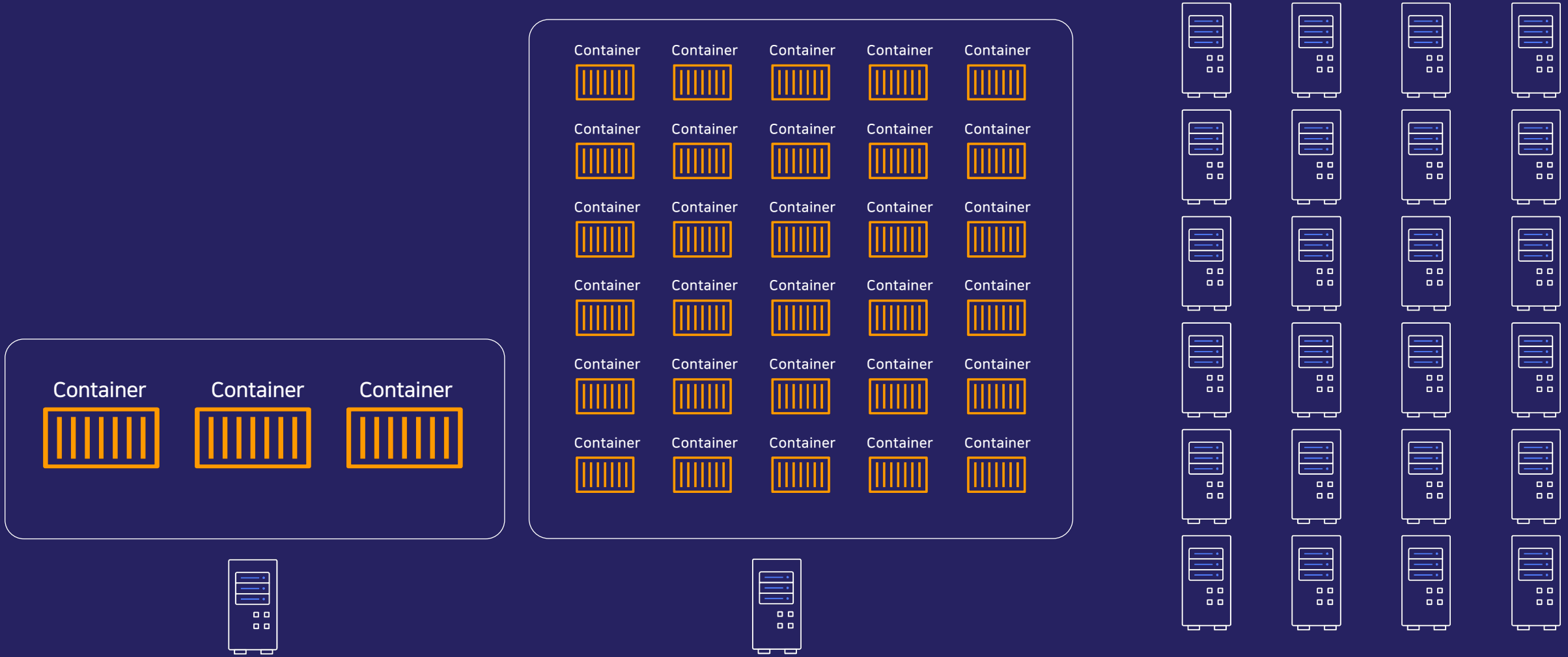
AWS는 본 콘텐츠에 포함되거나 콘텐츠를 통하여 고객에게 제공된 일체의 정보, 콘텐츠, 자료, 제품(소프트웨어 포함) 또는 서비스를 이용함으로써 인하여 발생하는 여하한 종류의 손해에 대하여 어떠한 책임도 지지 아니하며, 이는 직접 손해, 간접 손해, 부수적 손해, 징벌적 손해 및 결과적 손해를 포함하되 이에 한정되지 아니합니다.



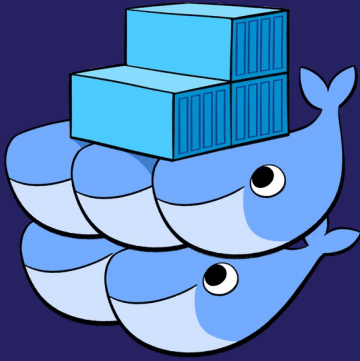
# Kubernetes Overview



# Container Orchestrator 가 필요한 이유?



# Container Orchestration Tool



Docker Swarm



Kubernetes



Mesos



# Kubernetes 가 필요한 이유?

Scheduling .....

수 많은 서버 중 최적의 서버에 컨테이너 배포

Self Healing .....

배포된 컨테이너에 문제가 생겼을 경우 자동으로 대응

Load Balancing .....

서비스 트래픽을 배포된 컨테이너에 균등하게 분산

Auto Scaling .....

서비스 제공을 위해 필요한 경우 자동으로 용량 증설

Rolling Update .....

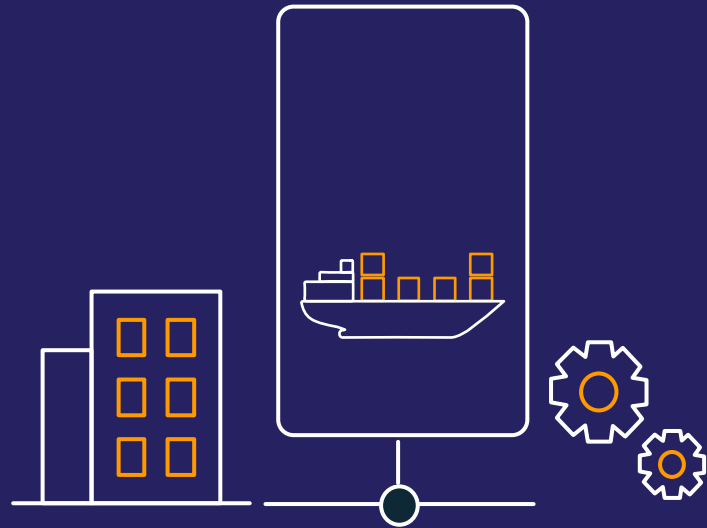
배포된 컨테이너 환경에 대한 업데이트

Portability .....

서로 다른 운영 환경에 대한 자유로운 이동



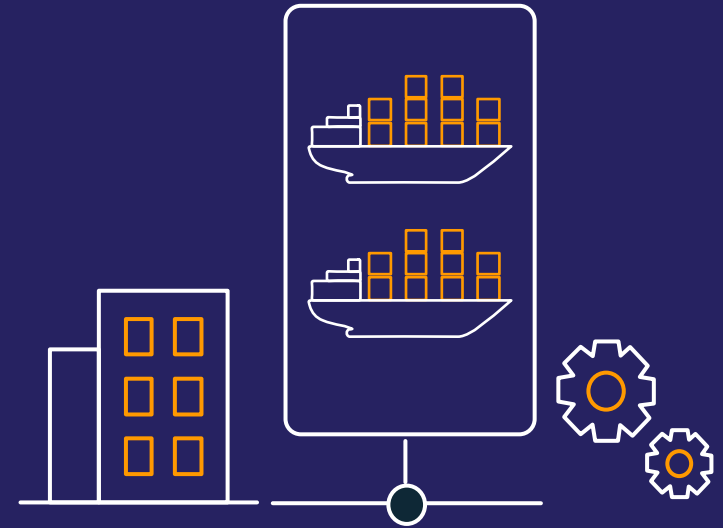
# Kubernetes 의 목표



Current State



YAML



Desired State

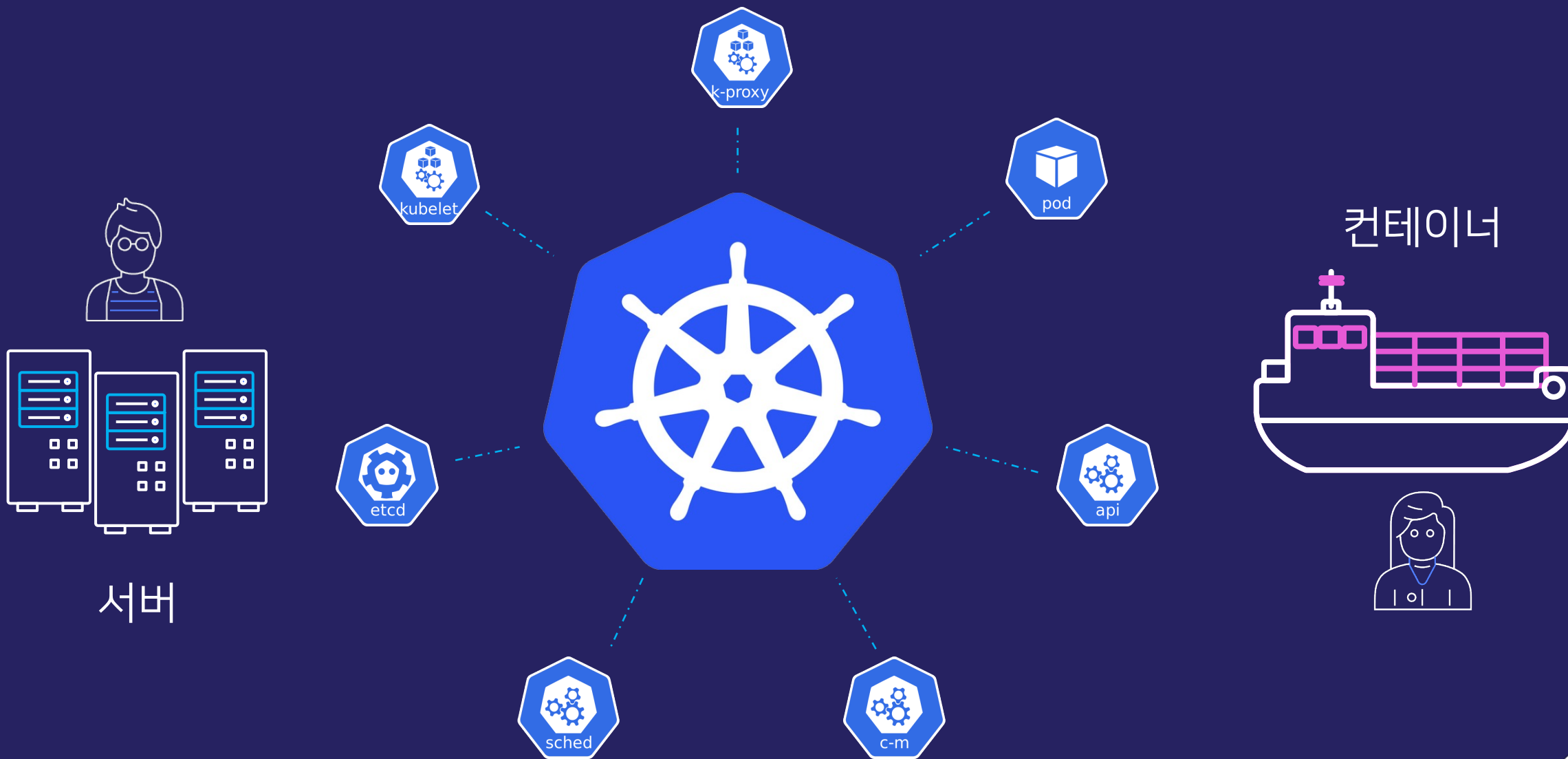


# Kubernetes 아키텍처 소개

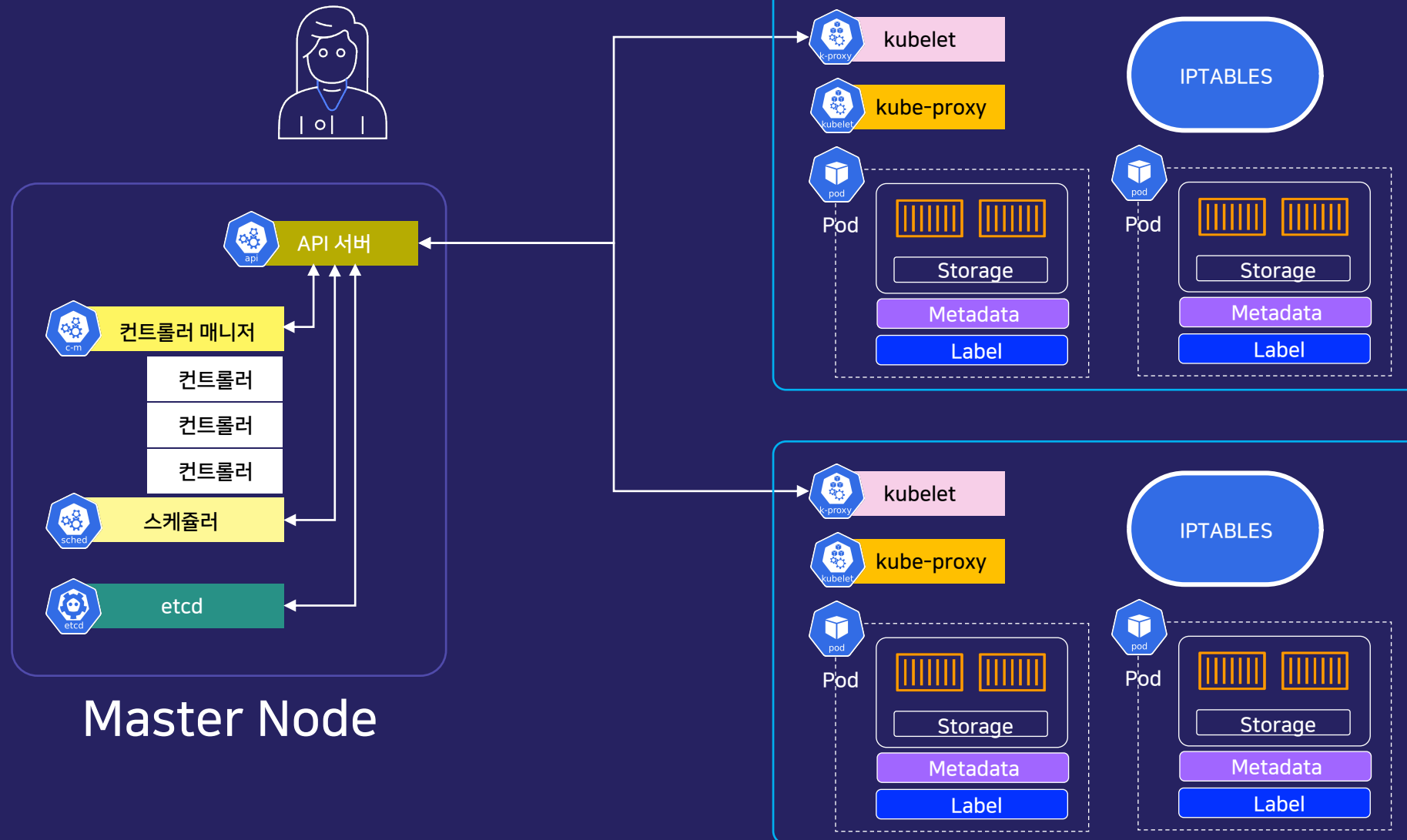




# Kubernetes, K8S, 쿠버네티스



# Kubernetes 클러스터

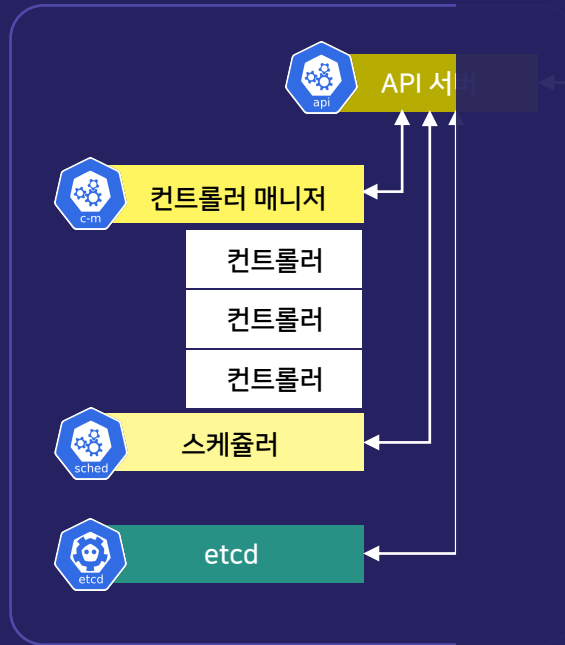


Worker Node

Worker Node



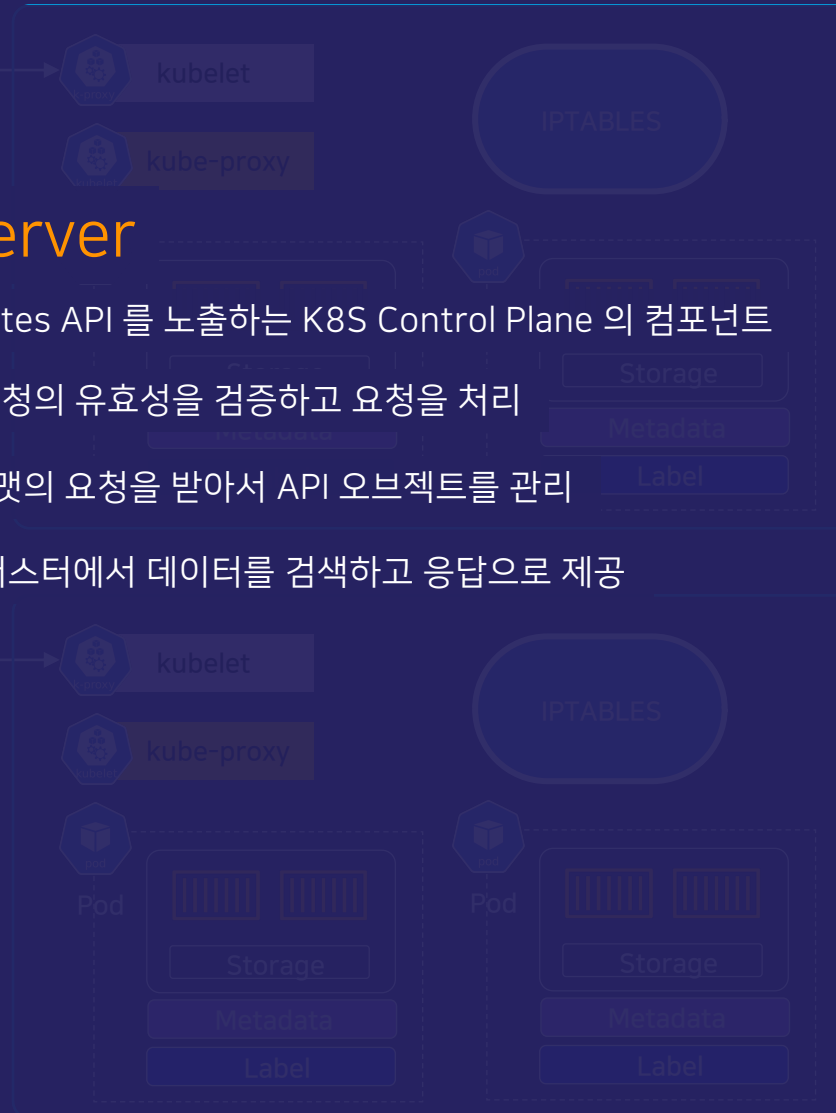
# Kubernetes 클러스터 구성요소 - Control Plane



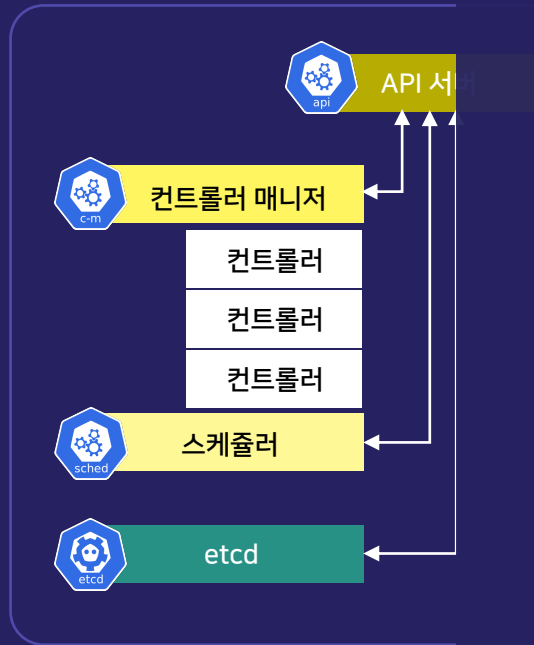
Master Node

## API Server

Kubernetes API 를 노출하는 K8S Control Plane 의 컴포넌트  
사용자 요청의 유효성을 검증하고 요청을 처리  
JSON 포맷의 요청을 받아서 API 오브젝트를 관리  
etcd 클러스터에서 데이터를 검색하고 응답으로 제공



# Kubernetes 클러스터 구성요소 - Control Plane



Master Node

## Controller Manager

Pod를 복제하거나 노드 운영 등 각 리소스를 제어하는 컨트롤러들을 감독하고 실행  
핵심 제어 루프를 포함하는 데몬  
서비스 운영에 필요한 다양한 컨트롤러를 통합  
Deployment Controller, ReplicaSet Controller, Node Controller 등등

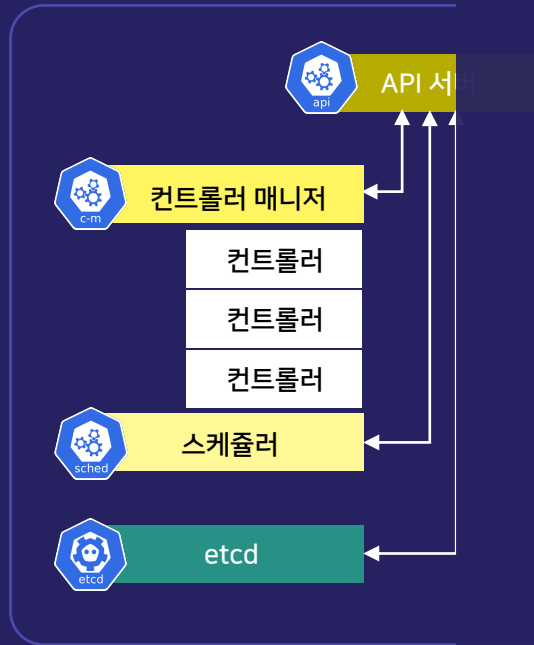


Worker Node

Worker Node



# Kubernetes 클러스터 구성요소 - Control Plane



Master Node

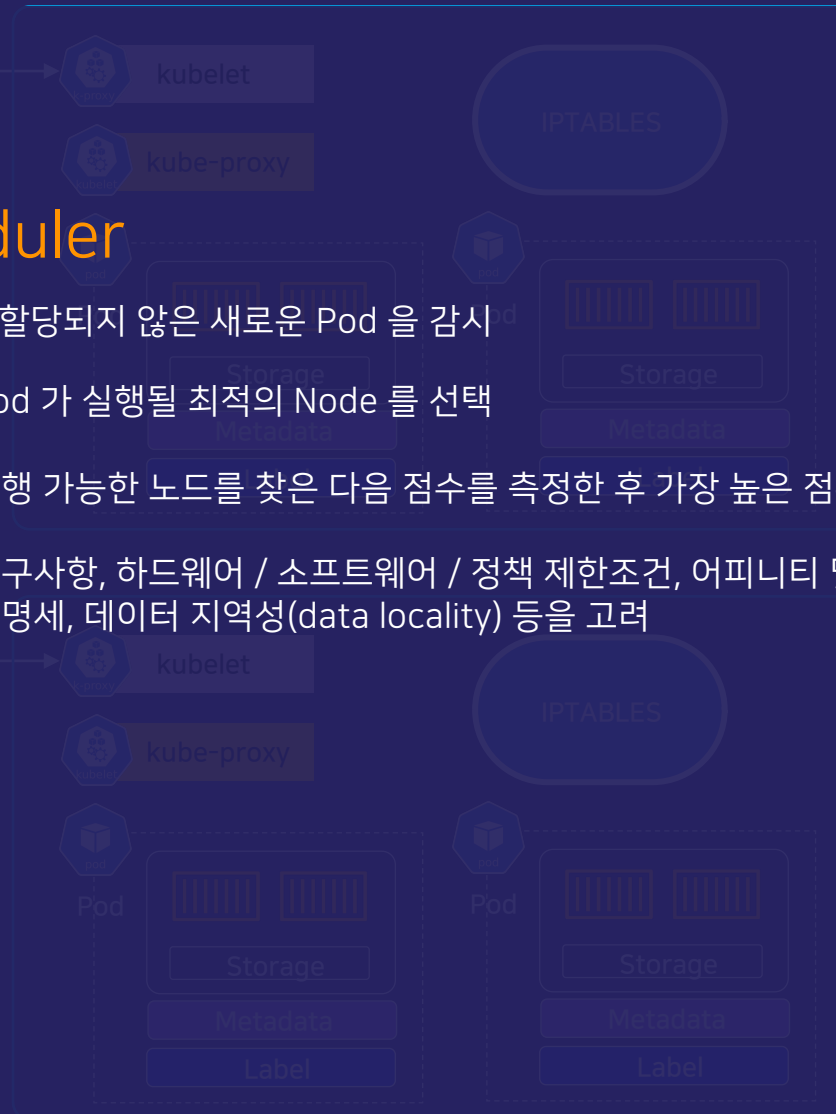
## Scheduler

Node 가 할당되지 않은 새로운 Pod 을 감시

새로운 Pod 가 실행될 최적의 Node 를 선택

파드가 실행 가능한 노드를 찾은 다음 점수를 측정한 후 가장 높은 점수를 가진 노드를 선택

리소스 요구사항, 하드웨어 / 소프트웨어 / 정책 제한조건, 어피니티 및 안티-어피니티 명세, 데이터 지역성(data locality) 등을 고려



# Kubernetes 클러스터 구성요소 - Control Plane



# Kubernetes 클러스터 구성요소 - Control Plane

## kube-proxy

Worker Node 에서 실행되는 네트워크 프록시

Node 의 네트워크 규칙을 유지 관리

서비스와 Endpoint 에 대한 연결을 구성

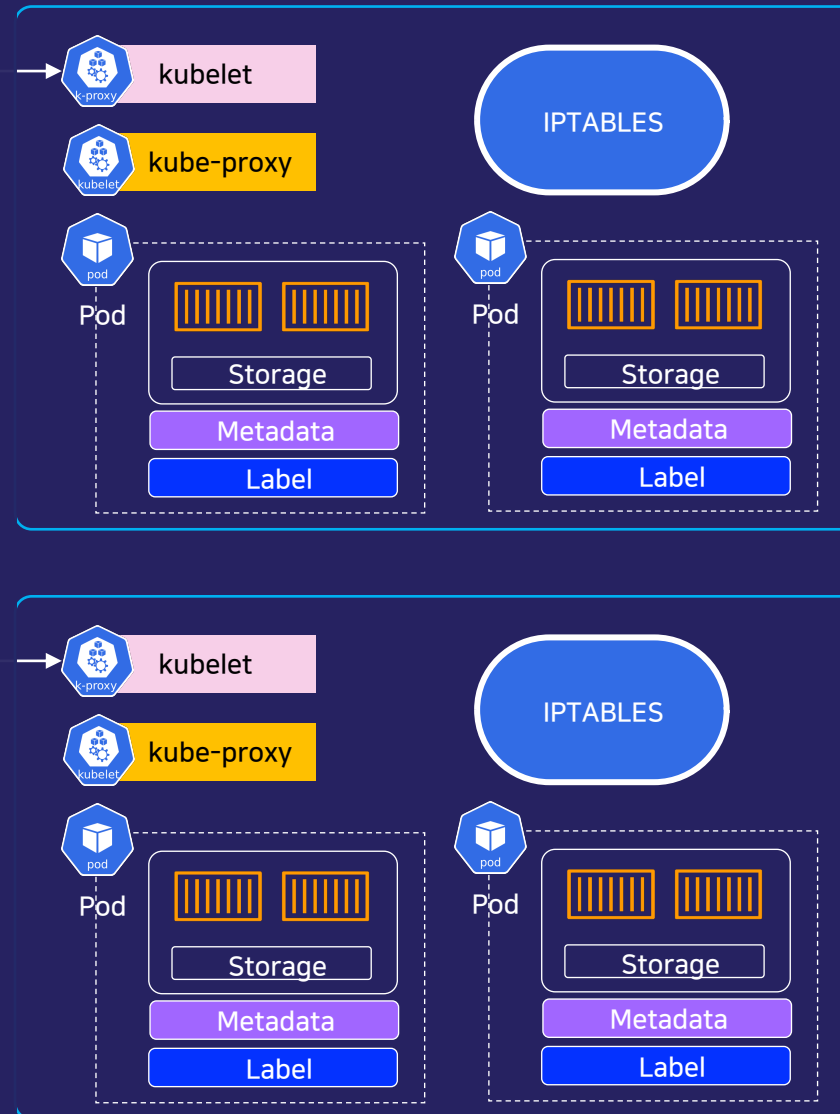
OS 에서 제공하는 패킷 필터링 기능을 사용

DaemonSet 형태로 배포

Worker Node

Worker Node

Master Node



# Kubernetes 클러스터 구성요소 - Control Plane

## kubelet

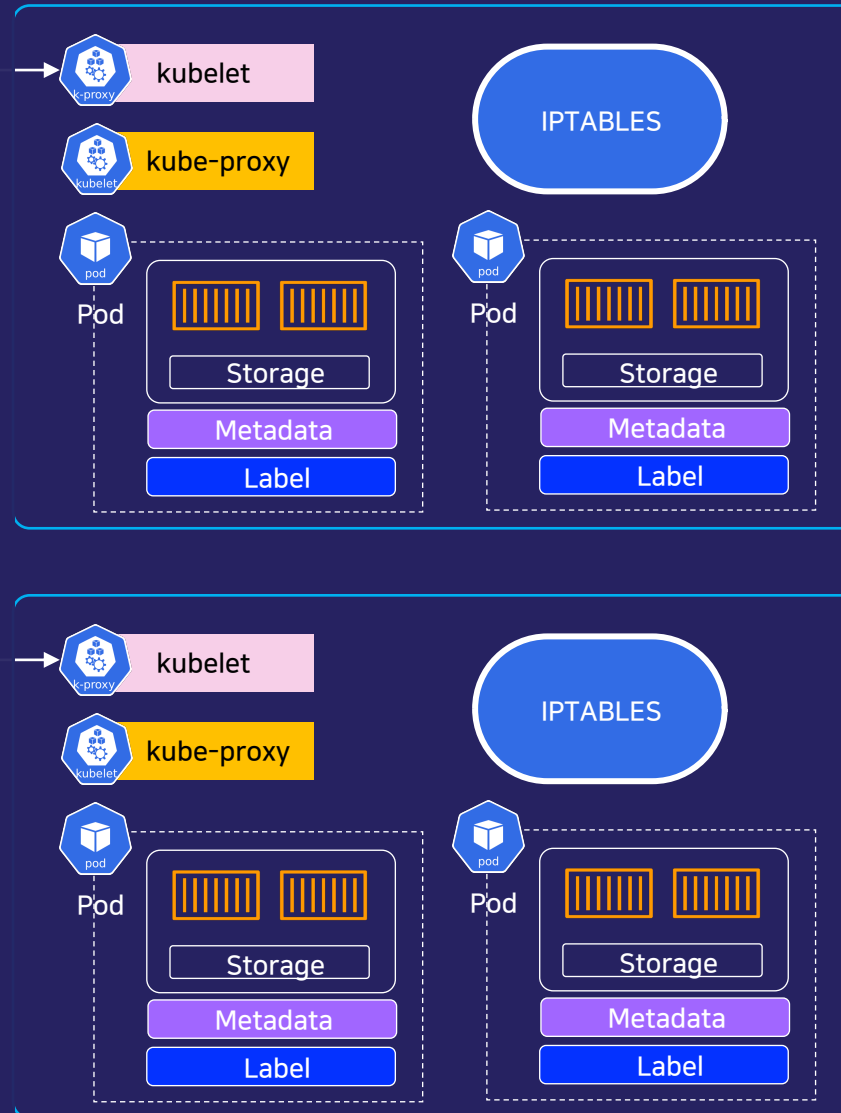
클러스터 각 노드에서 실행되는 Agent

Kubelet은 Pod에서 컨테이너가 동작하도록 관리

Pod Spec 기반으로 동작하는지 확인

Worker Node

Worker Node





# Kubernetes 클러스터 구성요소 - Control Plane

## Pod

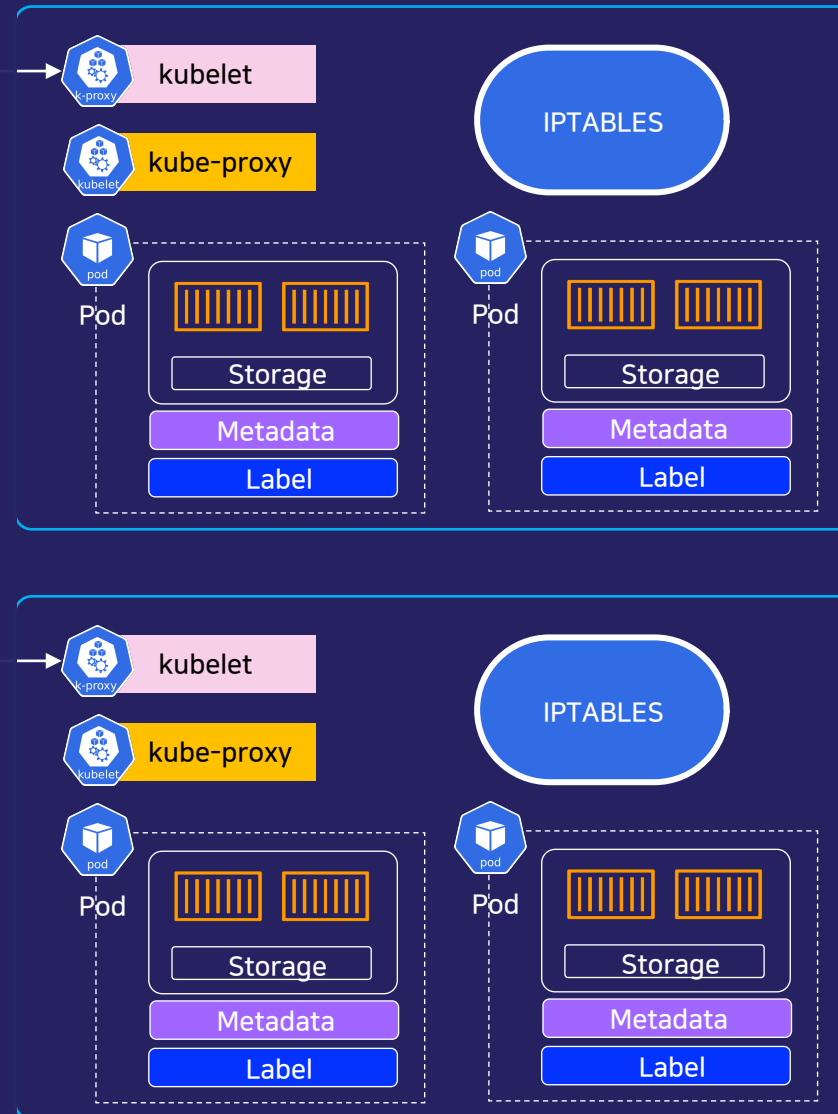
Kubernetes 의 가장 작은 Object 단위

하나 이상의 컨테이너를 포함

고유의 사설 IP 를 할당 받아서 사용

Pod 내의 컨테이너는 Pod 의 IP 를 공유해서 사용

필수 컨테이너는 Side Car 형태로 각 Pod 에 동시 배포 가능



Worker Node

Worker Node



# Workload 생성을 위한 기본 컨셉



Pod

Kubernetes 내에서  
정의되는 가장 작은 단위의  
워크로드 혹은 관리 리소스



ReplicaSet

Selector 를 기반으로하여  
Pod 의 스케줄, 스케일링,  
삭제를 담당



Deployment

Pod 과 ReplicaSet 에 대한  
선언적 관리 방법



# Kubernetes Configuration

## Imperative(명령형)

무언가를 작업하기 위한 방법을 정의하는 것

run, create, expose, edit, scale, set, create  
-f, replace -f, delete -f

NGINX Image 를 사용하는 Pod 3개를  
실행해주세요.

```
kubectl run nginxapp --image nginx:latest --replicas 3  
kubectl create service nodeport
```

## Declarative(선언형)

무언가를 작업하기 위하여 어떻게 진행할  
것인지를 나열하는 것

“Desired State” 를 YAML 로 정의하고 실행

NGINX Image 를 사용하는 Pod 3개가 실행되어  
있는 상태를 만들어주세요

```
kubectl apply -f nginx.yaml
```

정의

사용방법

예시

차이점

idempotent(멱등성)



# Kubernetes Control Loop



Desired State

spec:



Current State

status:



# Kubernetes YAML



```
⚡ apiVersion: apps/v1
⚡ kind: Deployment
⚡ metadata:
  name: nginx-deployment
  annotations:
    description: frontend
⚡ labels:
  app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
status:
```

## apiVersion

kind 에 지정된 Kubernetes Object 에 사용되는 API 버전을 지정

## kind

YAML 을 이용해 생성하고자 하는 Kubernetes Object 를 지정

## metadata

Object 를 구분지어 줄 수 있는 데이터 정보 (이름, Label, Namespace 등)

## annotations

쿠버네티스 오브젝트에 메타데이터를 첨부할 때 사용. 라벨처럼 map 형태로 구성되지만 사람보다는 machine을 위한 용도로 사용되며 검색이 불가능한 metadata를 지정하는데 사용



# Kubernetes YAML

```
⚡ apiVersion: apps/v1
⚡ kind: Deployment
⚡ metadata:
  name: nginx-deployment
  annotations:
    description: frontend
⚡ labels:
  app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
status:
```

## label

Kubernetes Object 에 첨부된 key-value 값으로 오브젝트의 특성을 식별하는데 사용

## spec

Kubernetes 에 배포될 자원에 대한 요구 사항을 명시

## selector

효율적으로 레이블을 이용해 오브젝트를 쉽게 식별할 수 있게 함. 컨트롤러가 모니터링해야하는 대상을 명시

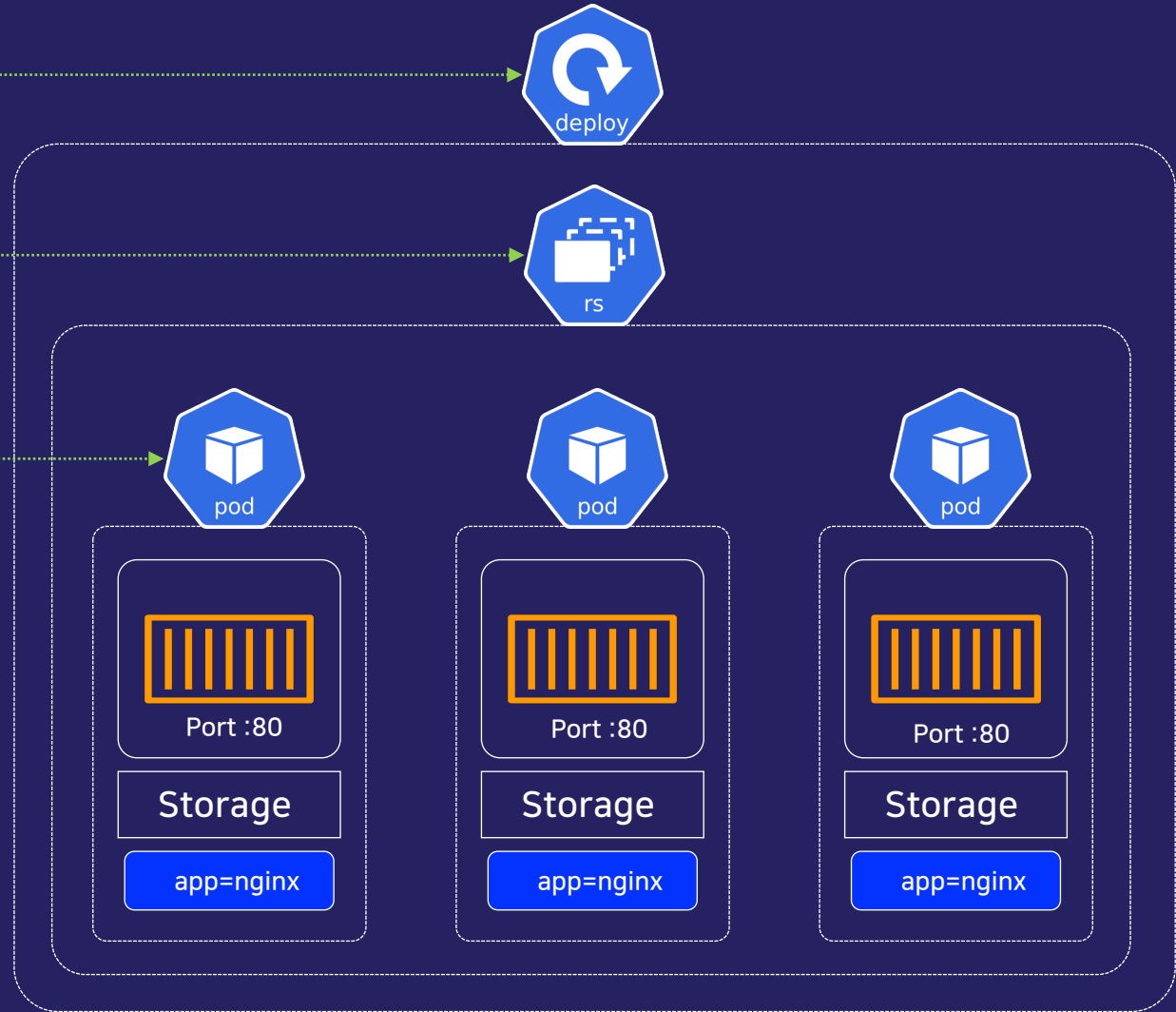
## template

새 pod를 런칭하는데 사용할 템플릿. selectors의 값이 template의 labels과 일치해야 관리되는 파드를 제대로 선택.

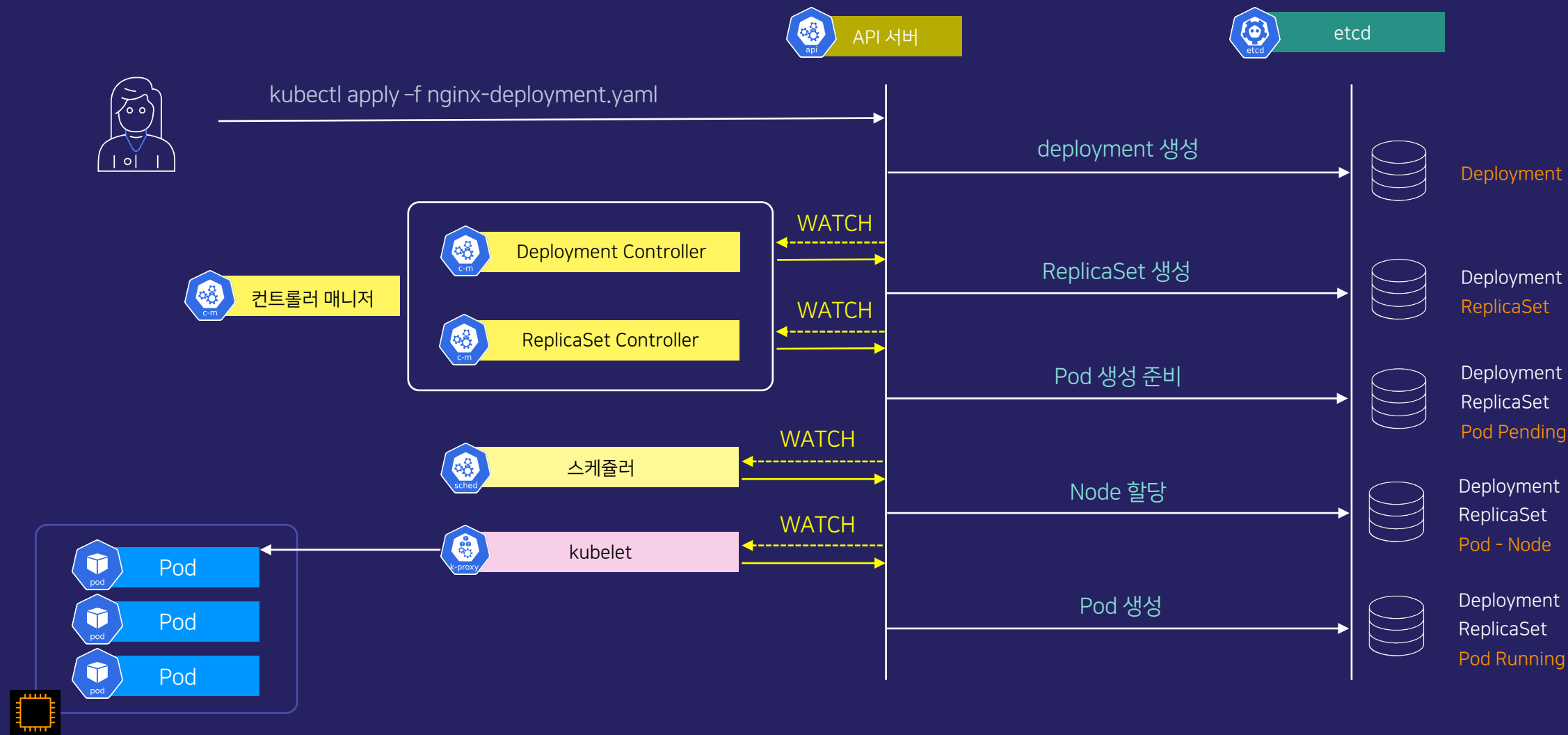


# Declarative Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```



# K8S Pod 생성 흐름

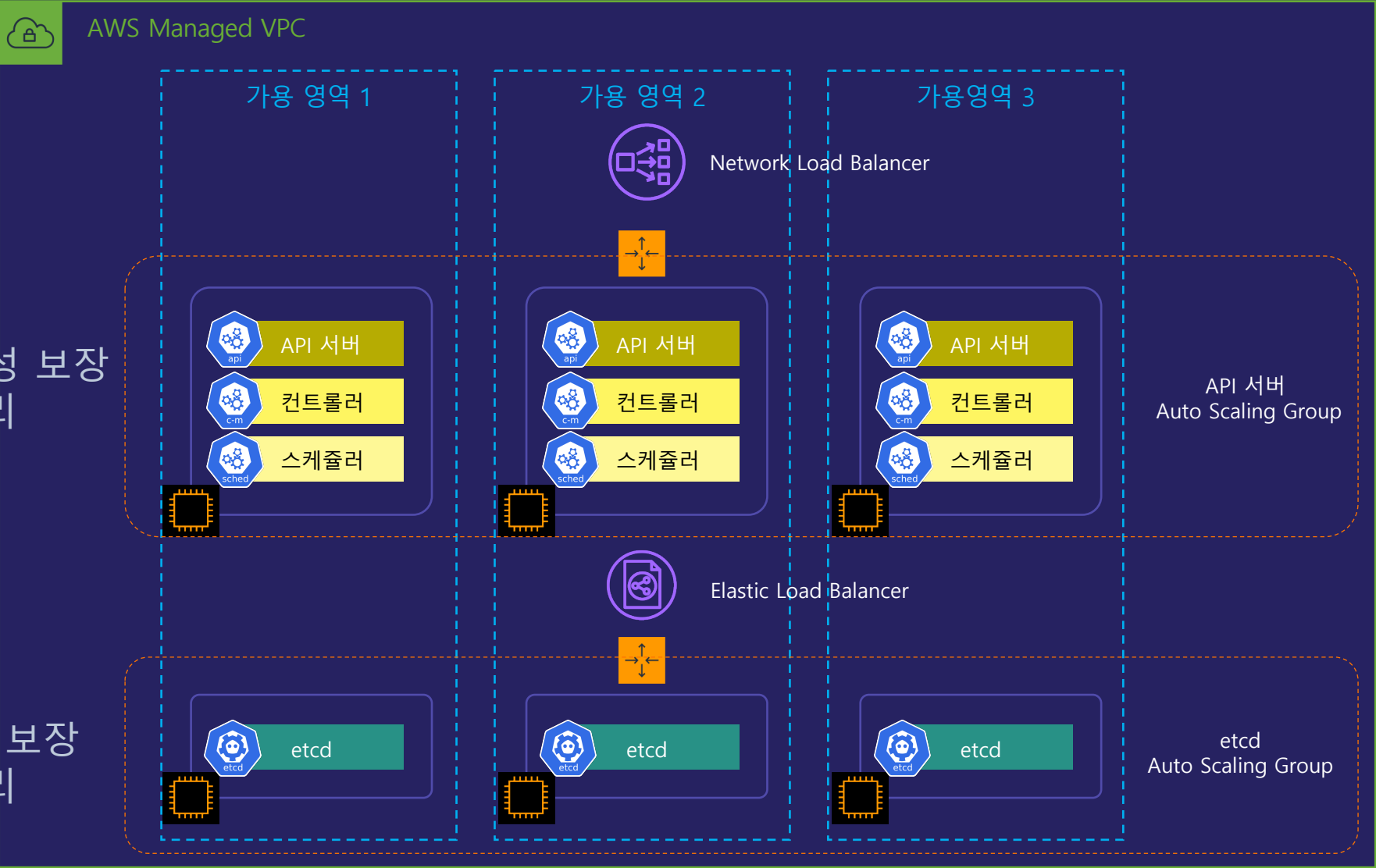




# Kubernetes Clustering 과 인증



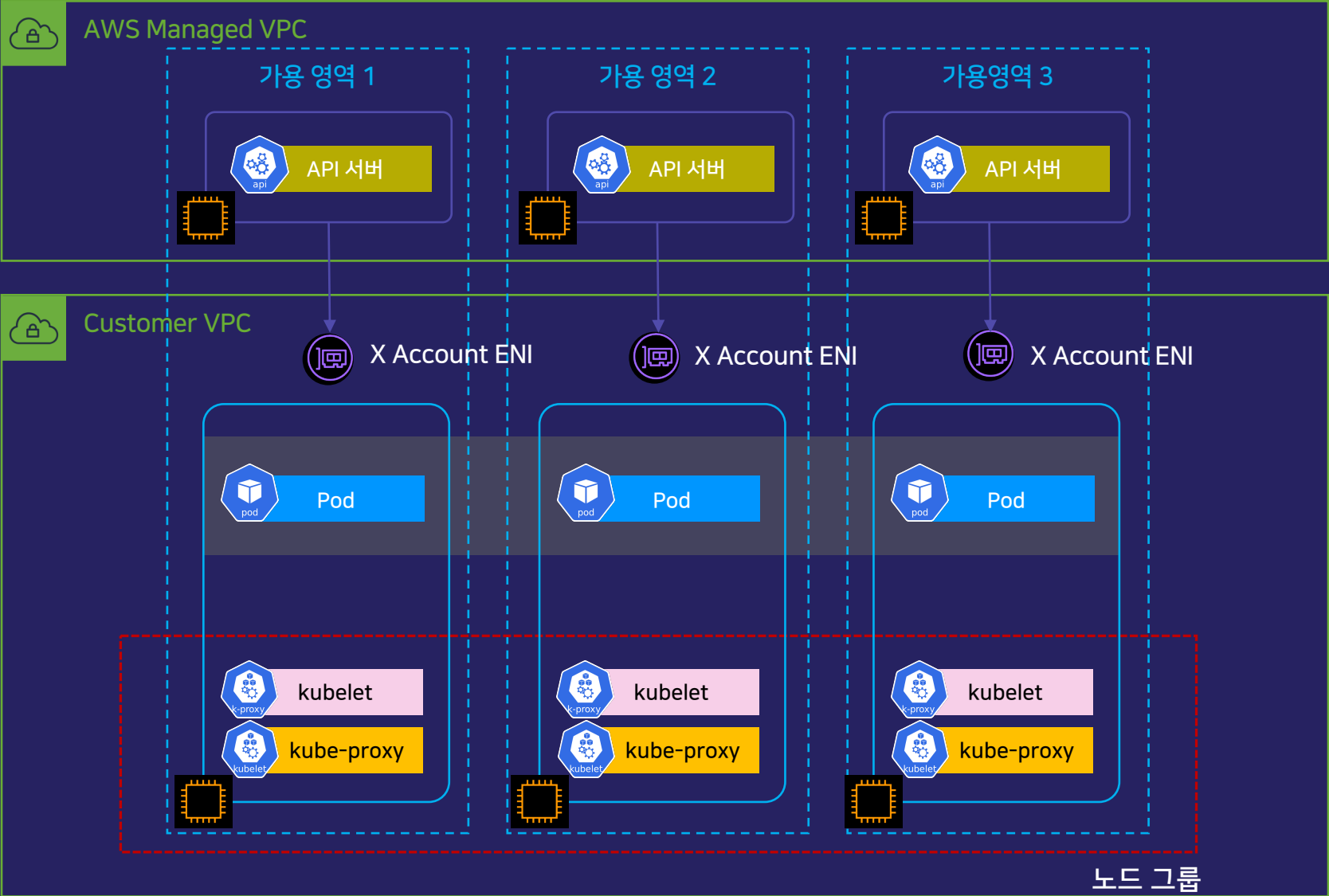
# Amazon EKS 컨트롤 플레인



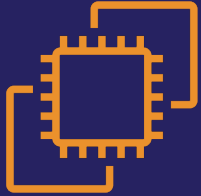
# Amazon EKS 데이터 플레인

AZ 별 교차계정 ENI 할당

관리형 노드 그룹

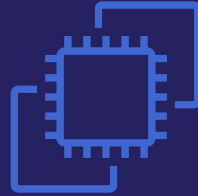


# EKS 데이터 플레인 옵션



## Self-Managed 노드 그룹

Custom AMI 를 이용하여 직접 관리하는 AutoScaling Group 을 사용. OS 에 대한 기본 구성이나 패치에 대한 책임은 고객의 책임 영역



## Managed 노드 그룹

AWS EKS 에 의해 고객의 VPC 에 프로비저닝. 최신의 EKS Optimized AMI 를 사용하며 새로운 AMI 에 대한 배포 및 구버전 AMI 제거 등을 모두 자동화하여 AWS 에서 처리

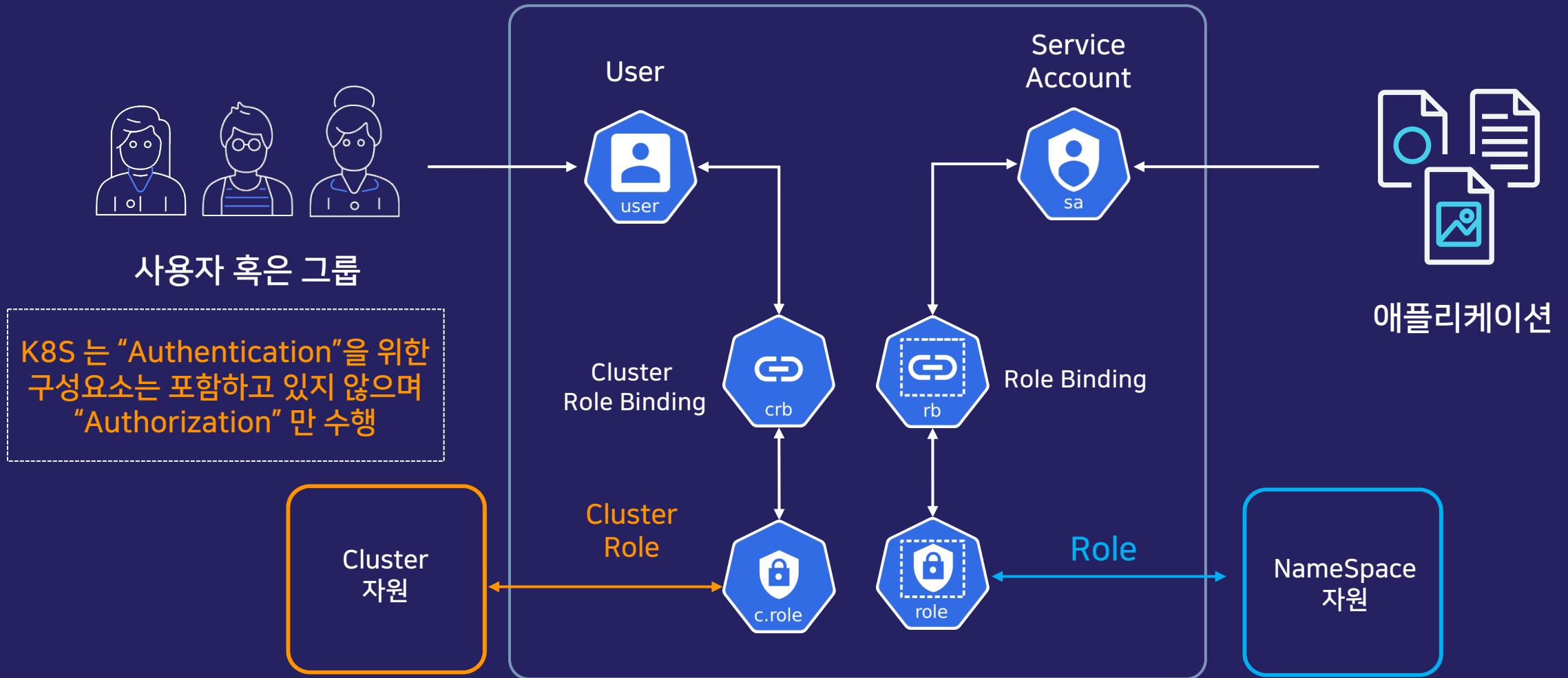


## AWS Fargate

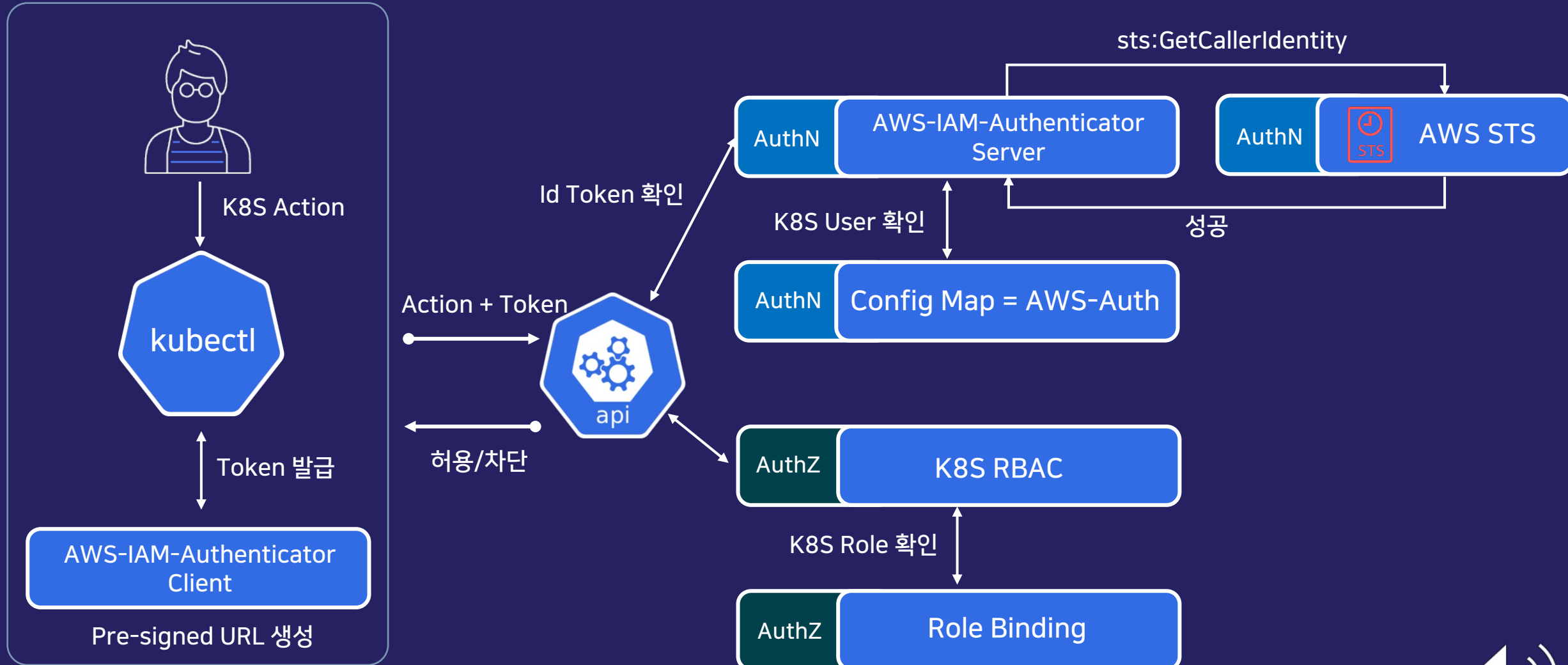
별도로 관리하는 EC2 인스턴스 없이 Fargate 환경에서 제공하는 Micro VM 을 이용하여 Pod 별 VM 할당



# K8S User & Service Account



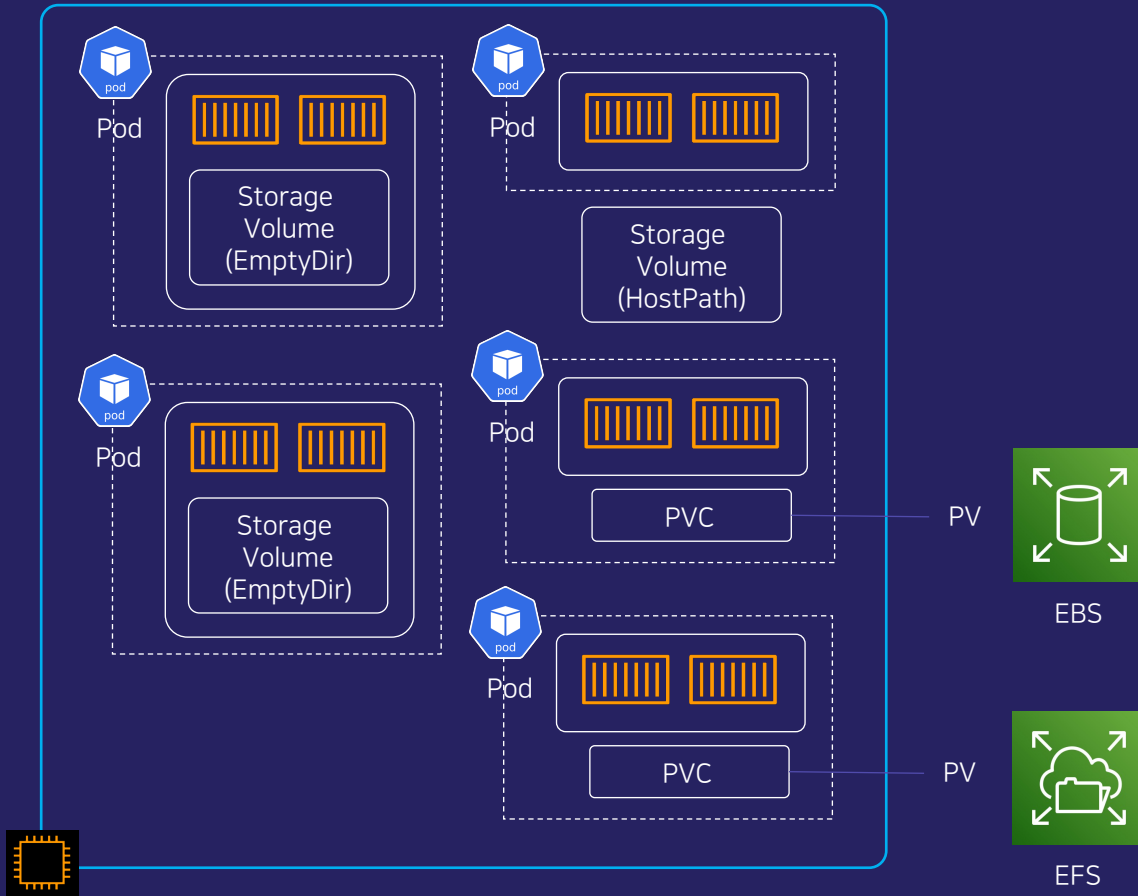
# EKS: IAM Authentication



# Kubernetes Storage



# Volume



# EmptyDir

## Pod가 실행되는 동안 사용되는 임시 저장 공간

# HostPath

Pod가 실행되는 Node 의 파일이나 경로를 마운트해서 사용

# Persistent Volume

## Pod 와 독립적으로 존재하는 리소스

## PVC 를 생성한 후 PV 를 연결하는 방식으로 사용

## Storage Class 를 이용하여 동적으로 PV 를 할당 가능

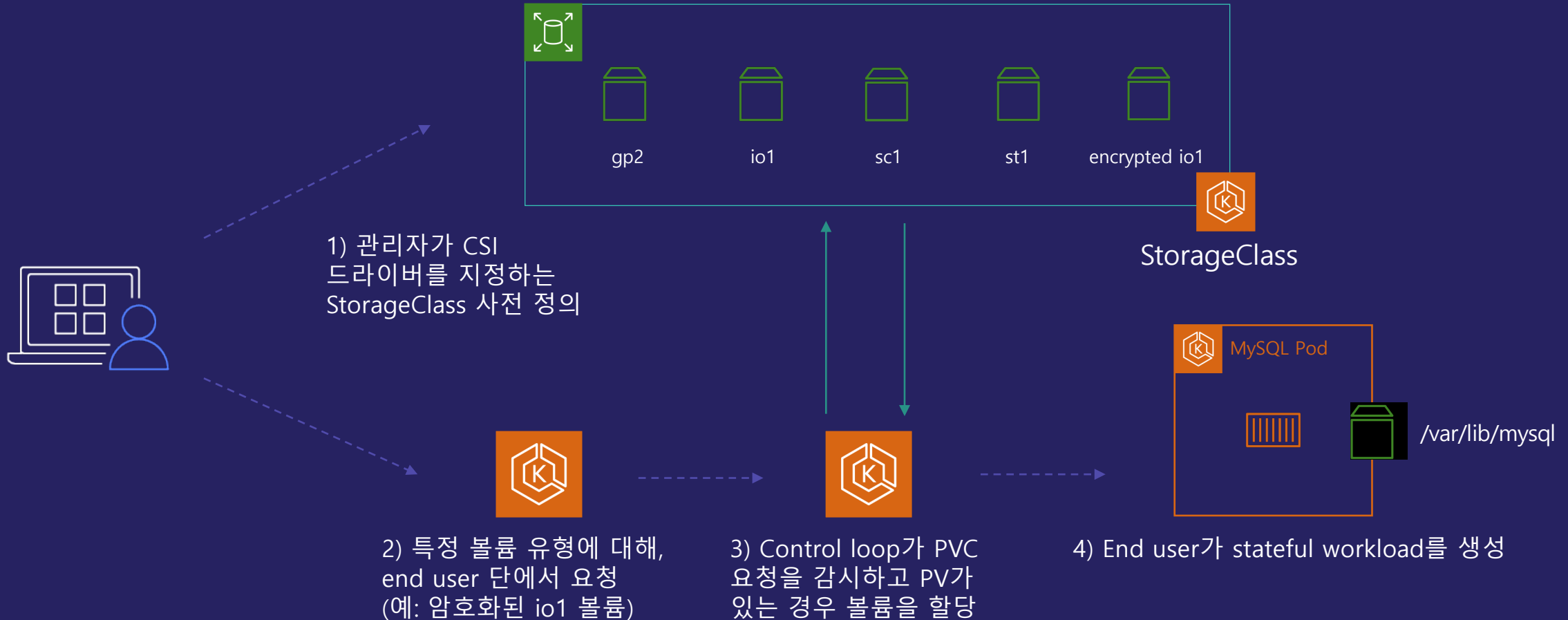
## configMap, Secret, downwardAPI

## 특정 K8S 자원이나 정보를 Pod 에 노출하기 위한 특수 목적 Volume





# AWS Container Storage Interface (CSI) Driver



# Kubernetes Scaling



© 2022, Amazon Web Services, Inc. or its affiliates.

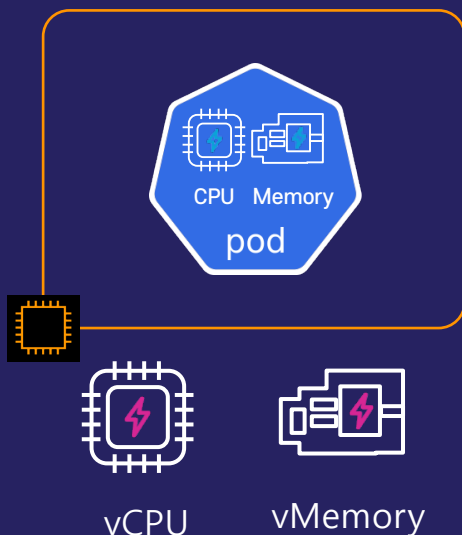


# Pod/Node 자동 확장



Cluster Auto-Scaler

Pod 을 배포할 Node 가 부족한 경우 신규 Node Provisioning



Horizontal Pod Auto-Scaler

서비스를 처리할 Pod 자원이 부족한 경우 신규 Pod Provisioning



Vertical Pod Auto-Scaler

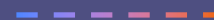
서비스를 처리할 Pod 자원이 부족한 경우 Pod 교체 (자동 or 수동)



Scale Out



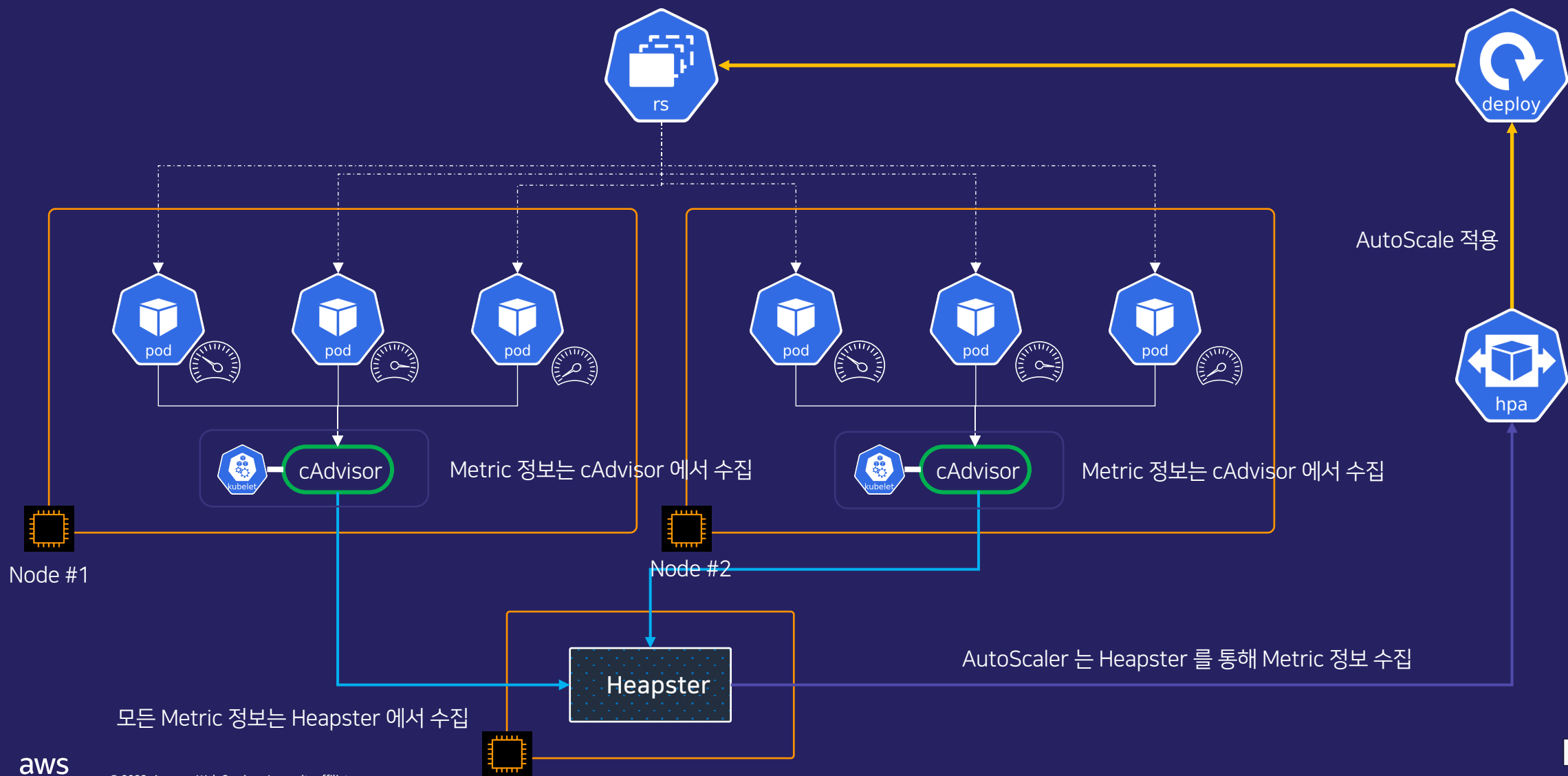
Scale Out



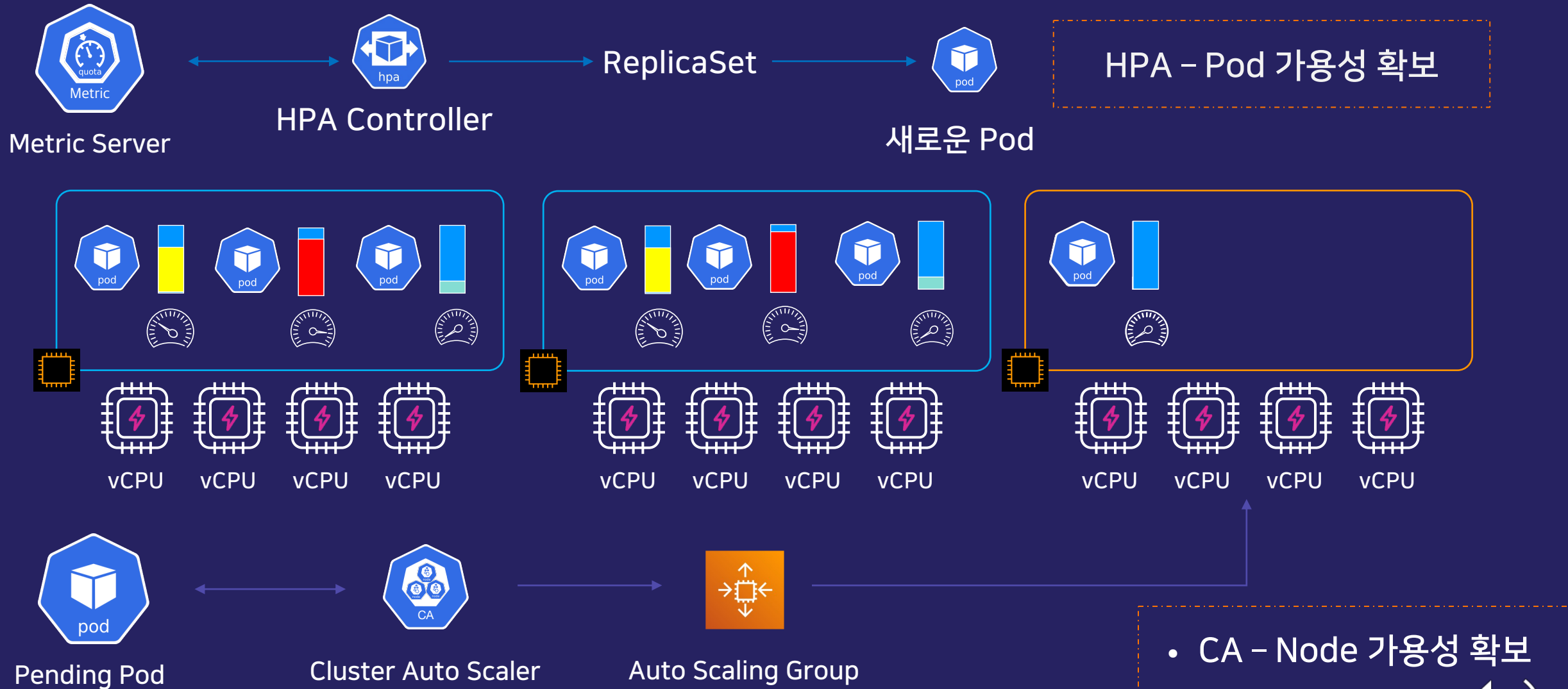
Scale Up



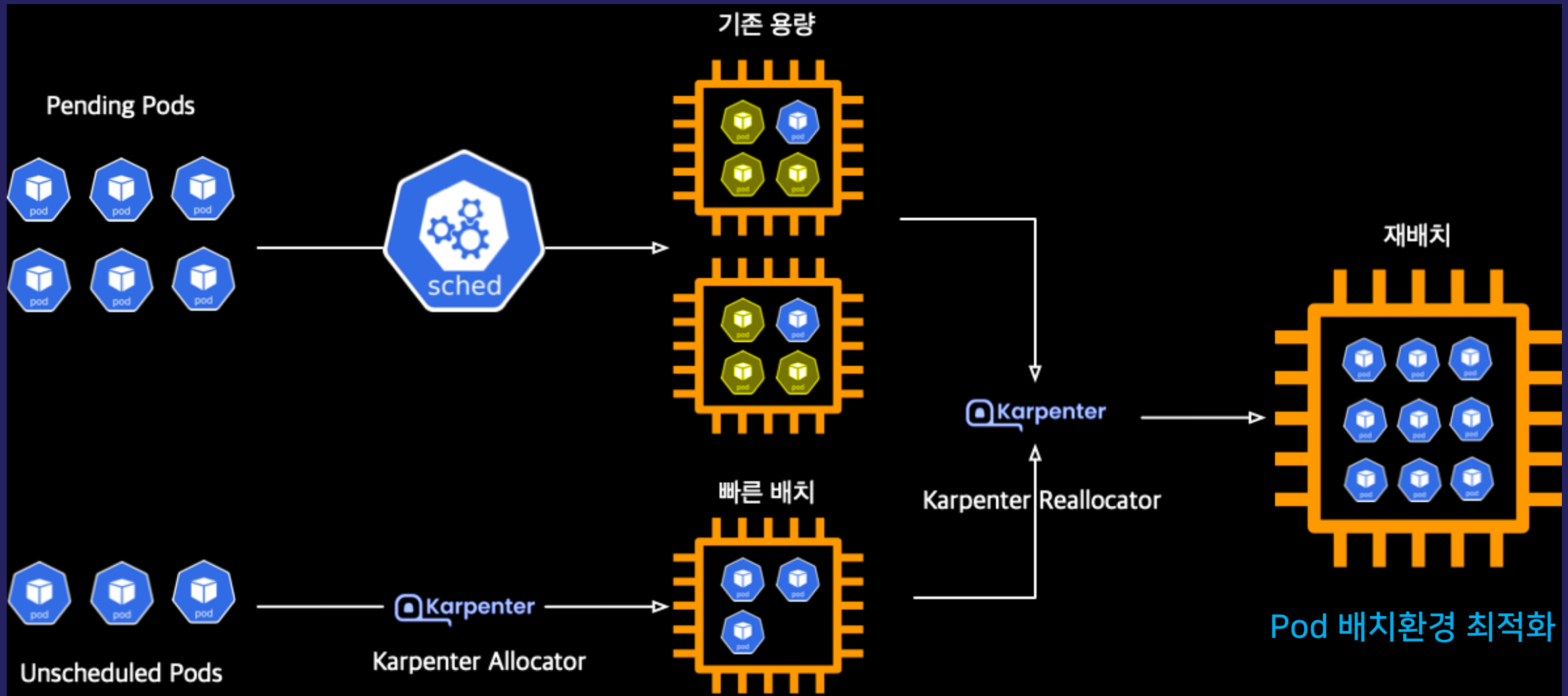
# 자동 확장을 위한 메트릭 수집



# EKS 환경에서의 Auto Scaling



# Karpenter(Groupless Node ASG)



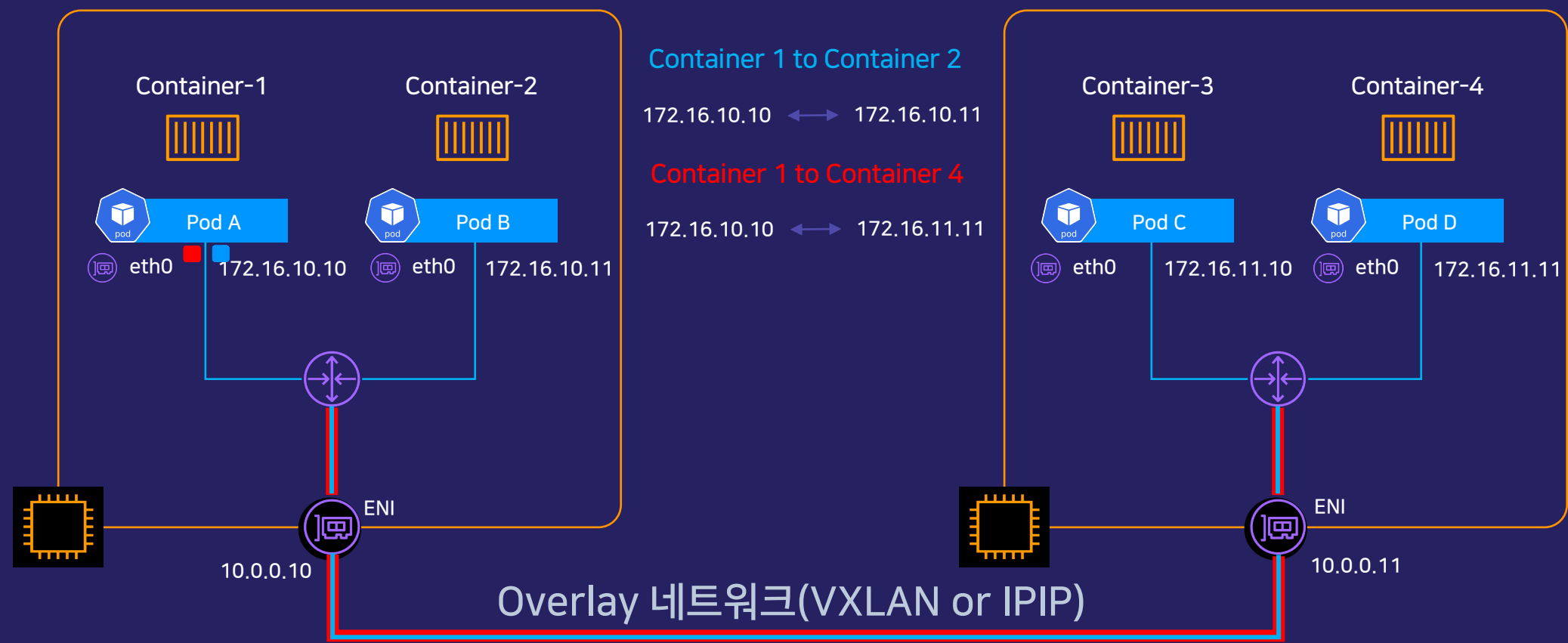
# Kubernetes Networking



© 2022, Amazon Web Services, Inc. or its affiliates.



# Kubernetes 네트워킹

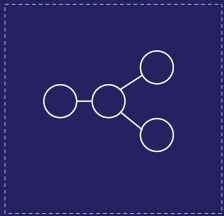
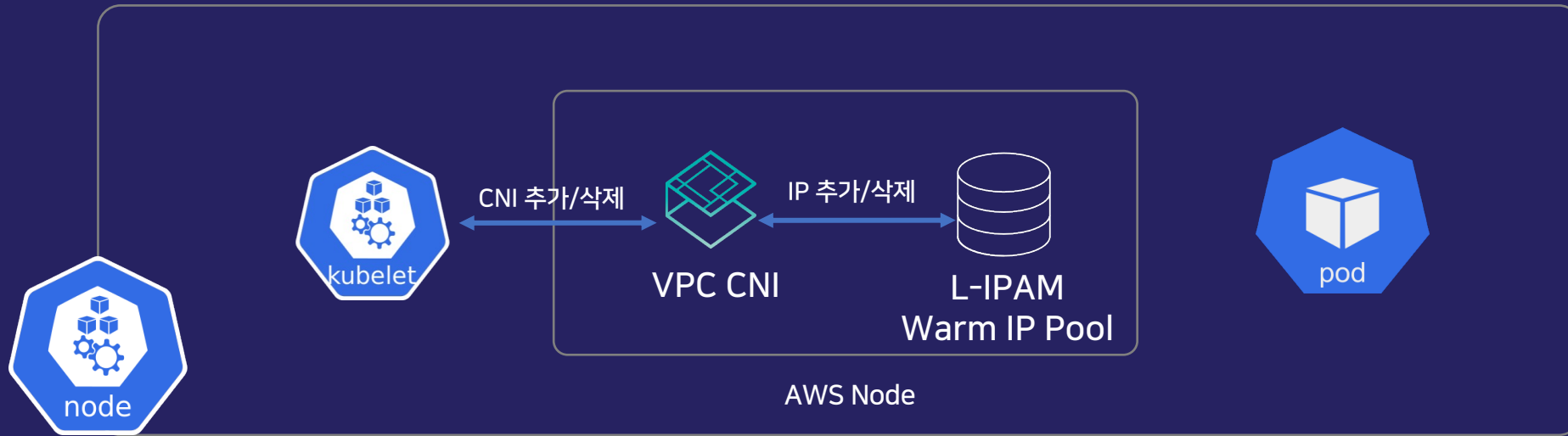


Header	Outer IP	Inner IP
Source	10.0.0.10	172.16.10.10
Destination	10.0.0.11	172.16.11.11

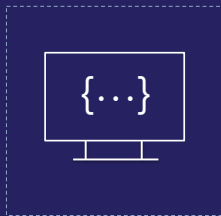




# Amazon VPC CNI(Container Network Interface) 플러그인



CNI 플러그인을 사용하는  
기본 VPC 네트워킹



파드는 VPC와 동일한  
주소를 파드가 가짐



간편하고 안전한 네트워킹



GitHub에서 유지 관리되는  
오픈 소스 프로젝트

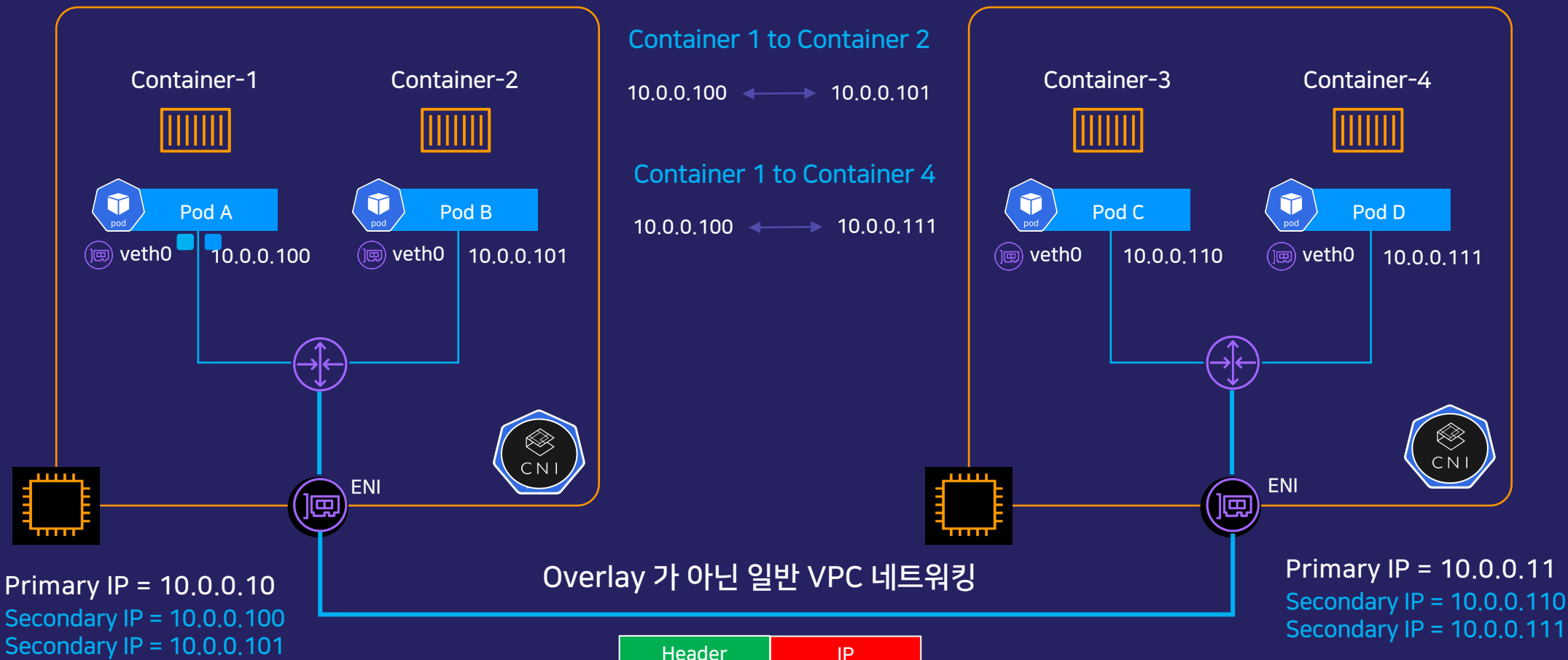
<https://github.com/aws/amazon-vpc-cni-k8s>



© 2022, Amazon Web Services, Inc. or its affiliates.



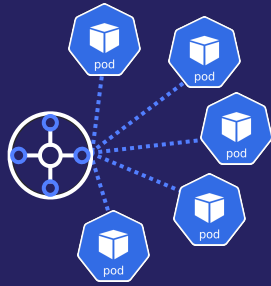
# EKS VPC CNI 네트워킹



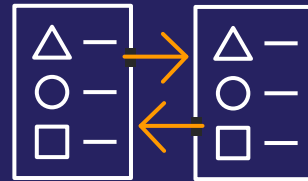
Header	IP
Source	10.0.0.100
Destination	10.0.0.111



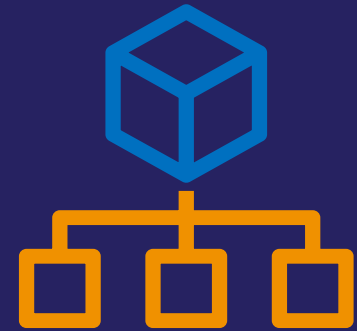
# K8S 서비스 타입



Cluster IP



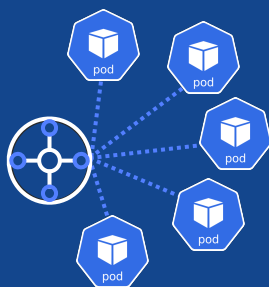
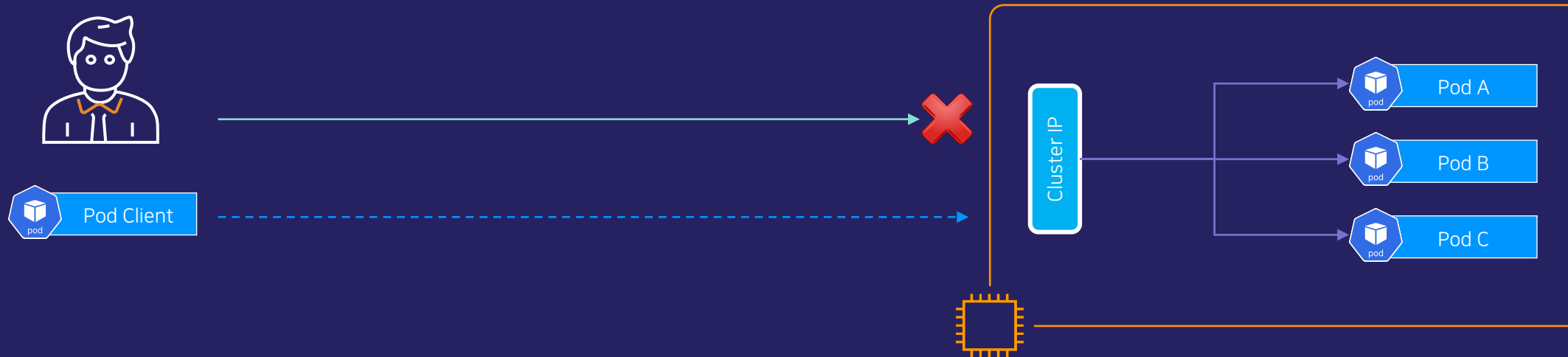
Node Port



Load Balancer



# K8S서비스 타입별 트래픽 흐름(Cluster IP)



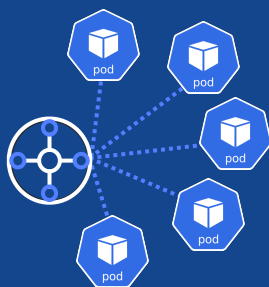
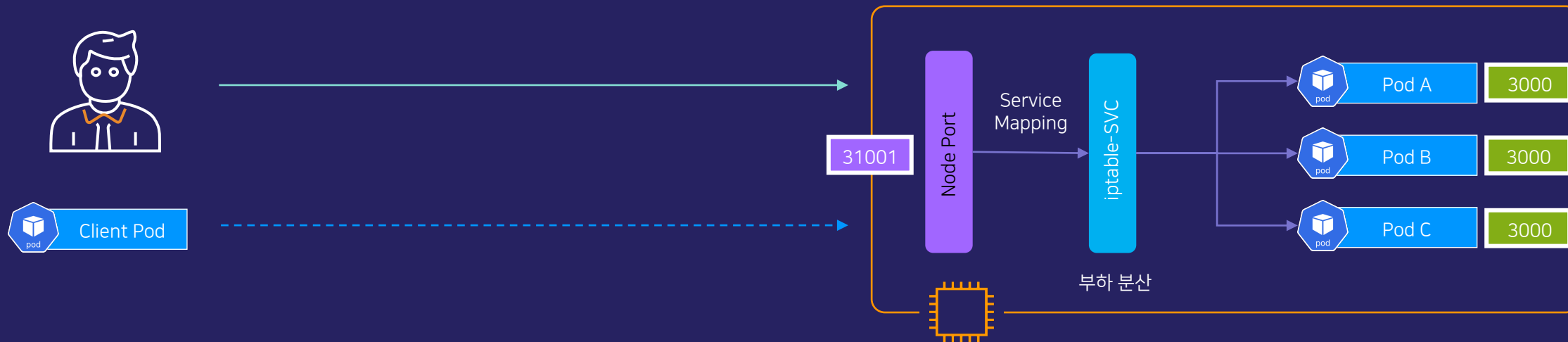
Cluster IP 는 Cluster 내부에서 서비스에 접속하고자 하는 용도로 사용

Cluster IP 에는 Pod 에 할당되지 않은 네트워크의 IP 주소를 할당

부하 분산에는 iptables(기본값) 이나 IPVS(IP Virtual Server) 를 선택적으로 사용



# K8S서비스 타입별 트래픽 흐름(NodePort)



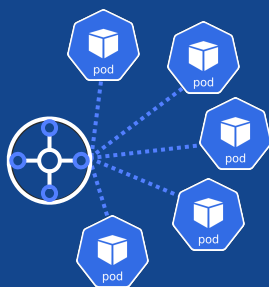
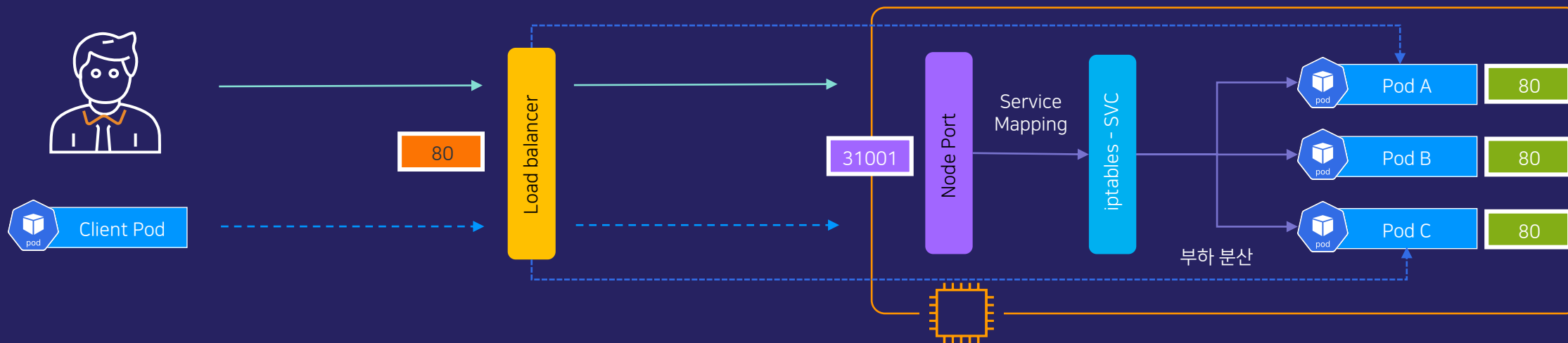
Worker Node 외부에 Port 를 Open 하여 내부 Pod 과 연결

서로 다른 Node 간의 Pool 선택이 가능하며 필요 시 externalTrafficPolicy 를 Local 로 설정

Node Port 의 Port Range = 30000 ~ 32767



# K8S서비스 타입별 트래픽 흐름(Load Balancer)



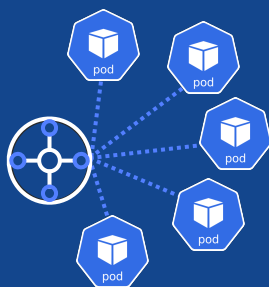
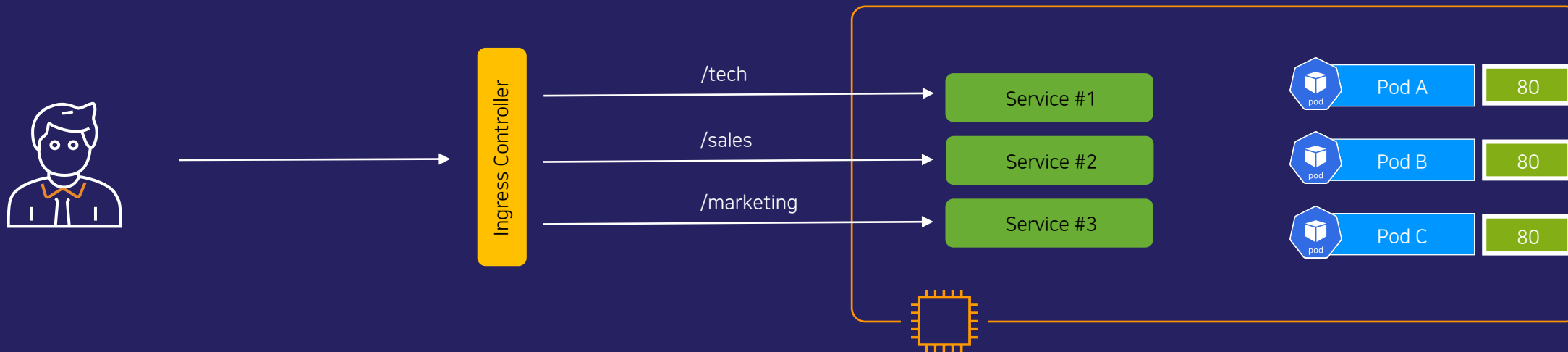
외부 접속용 Endpoint 역할을 수행할 Load Balancer 를 생성

외부 Load Balancer 는 각 Node 의 Node Port 를 Target 으로 등록(ALB/NLB 의 경우 IP 등록 지원)

Pod 의 부하 분산은 iptables에서 수행(Load Balancer 의 제공 기능에 따라 직접 분산도 가능)



# K8S서비스 타입별 트래픽 흐름(Ingress)



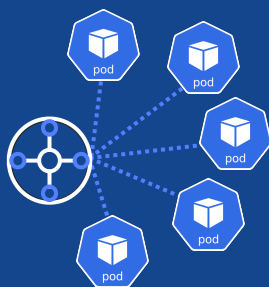
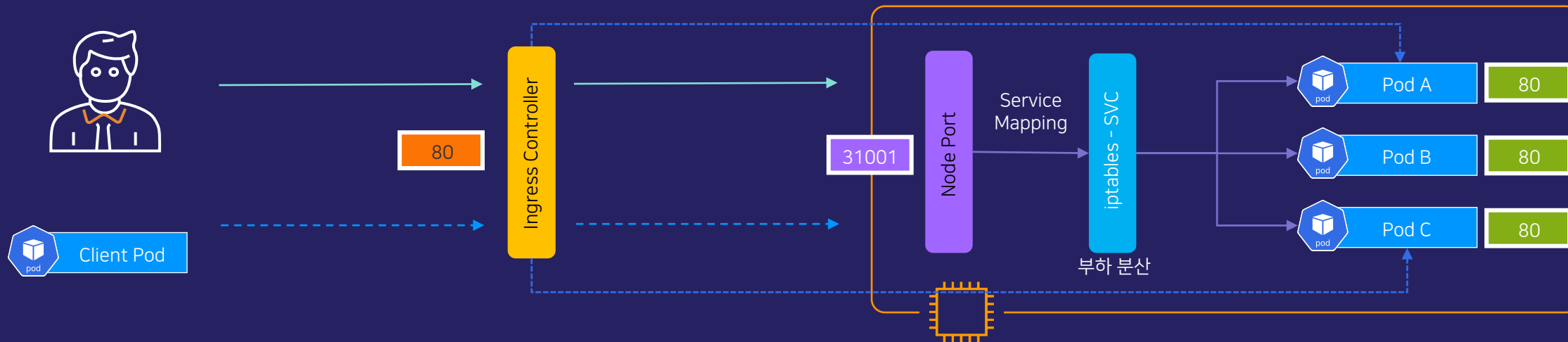
L7 영역의 관리자 정의 규칙을 기반으로 한 서비스 공개가 필요한 경우 사용

Ingress 를 통해 L7 영역에 대한 트래픽처리 규칙을 정의하고 Ingress Controller 를 통해 처리

Ingress Controller 는 NIGNX Controller 나 CSP 에서 제공하는 Controller 등 선택 가능



# K8S서비스 타입별 트래픽 흐름(Ingress - EKS)



외부 Endpoint 역할을 하는 Ingress Controller(ALB)를 자동으로 생성

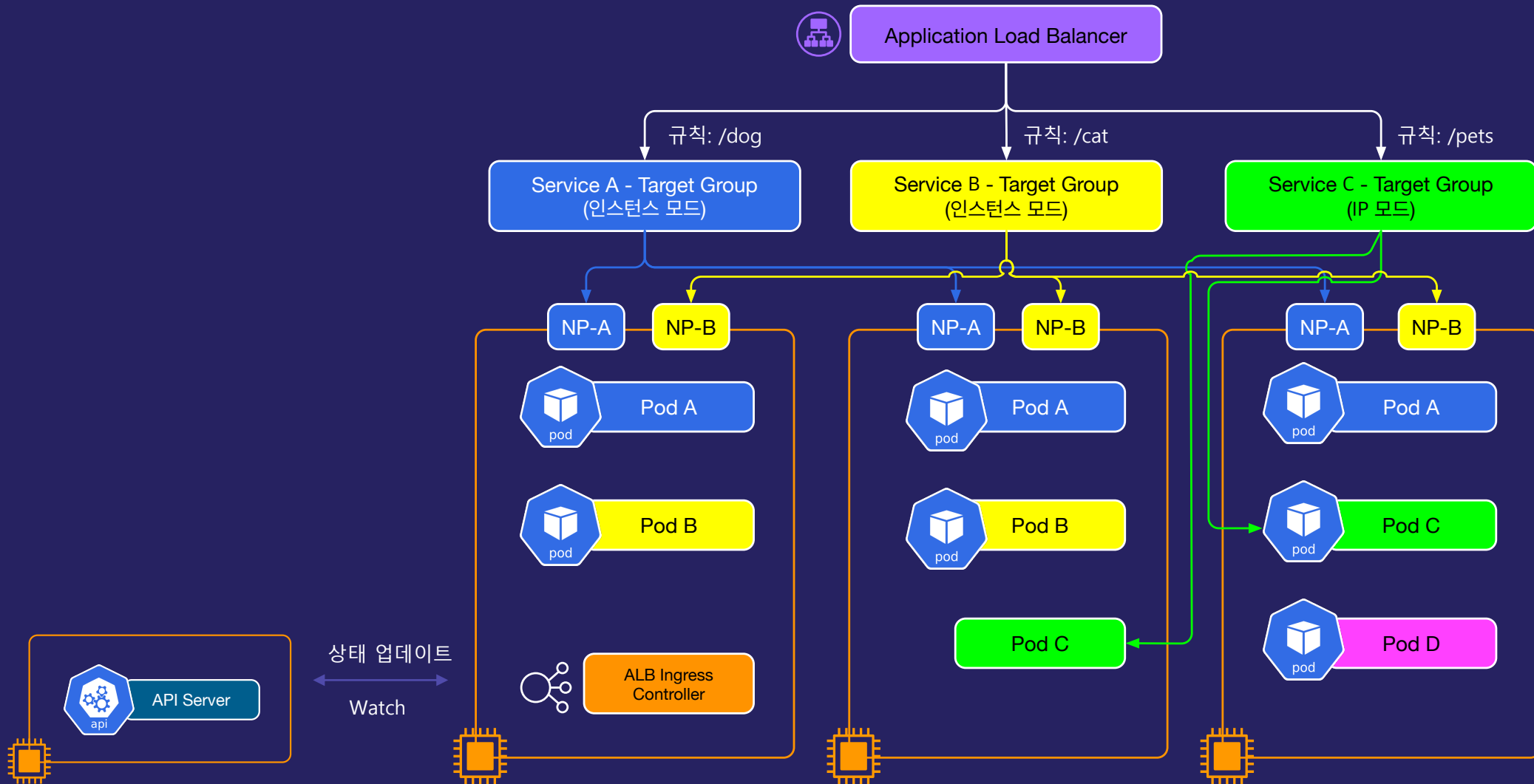
ALB 의 Target 은 Instance Mode 혹은 IP Mode 로 등록 가능(Sticky 필요 시 IP Mode 사용)

ALB 생성을 위하여 ELB 권한을 갖는 IAM Role 생성 및 연결 필요





# AWS Load Balancer Controller



# Amazon EKS 구성 및 설치



# Amazon EKS Cluster 배포 방식



AWS CloudFormation &  
AWS CDK



eksctl



pulumi

Terraform, Pulumi, Rancher  
... more

# eksctl로 쿠버네티스 클러스터 배포하는 예시

```
beta_zero:~/environment $ cat eksworkshop.yaml
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

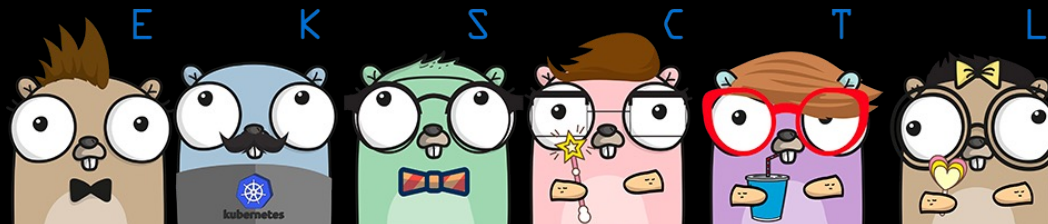
metadata:
  name: eksworkshop-eksctl
  region: ap-southeast-1
  version: "1.17"

availabilityZones: ["ap-southeast-1a", "ap-southeast-1b", "ap-southeast-1c"]

managedNodeGroups:
- name: nodegroup
  desiredCapacity: 3
  ssh:
    allow: true
    publicKeyName: eksworkshop

# To enable all of the control plane logs, uncomment below:
# cloudWatch:
#   clusterLogging:
#     enableTypes: ["*"]

secretsEncryption:
  keyARN: arn:aws:kms:ap-southeast-1:123456789012:key/93516110-3af6-4bcf-80ca-9140b703828d
beta_zero:~/environment $ eksctl create cluster -f eksworkshop.yaml
```





더 나은 세미나를 위해  
여러분의 의견을 남겨주세요!

▶ 질문에 대한 답변 드립니다.





# Thank you!

