ECSA DAY2 2025.01

Amazon EKS Overview

Woongsik Kim

AWS Professional Service

Agenda

1. Container Concept

2. Container Orchestration and Kubernetes

3. EKS Overview

4. Container Security in EKS

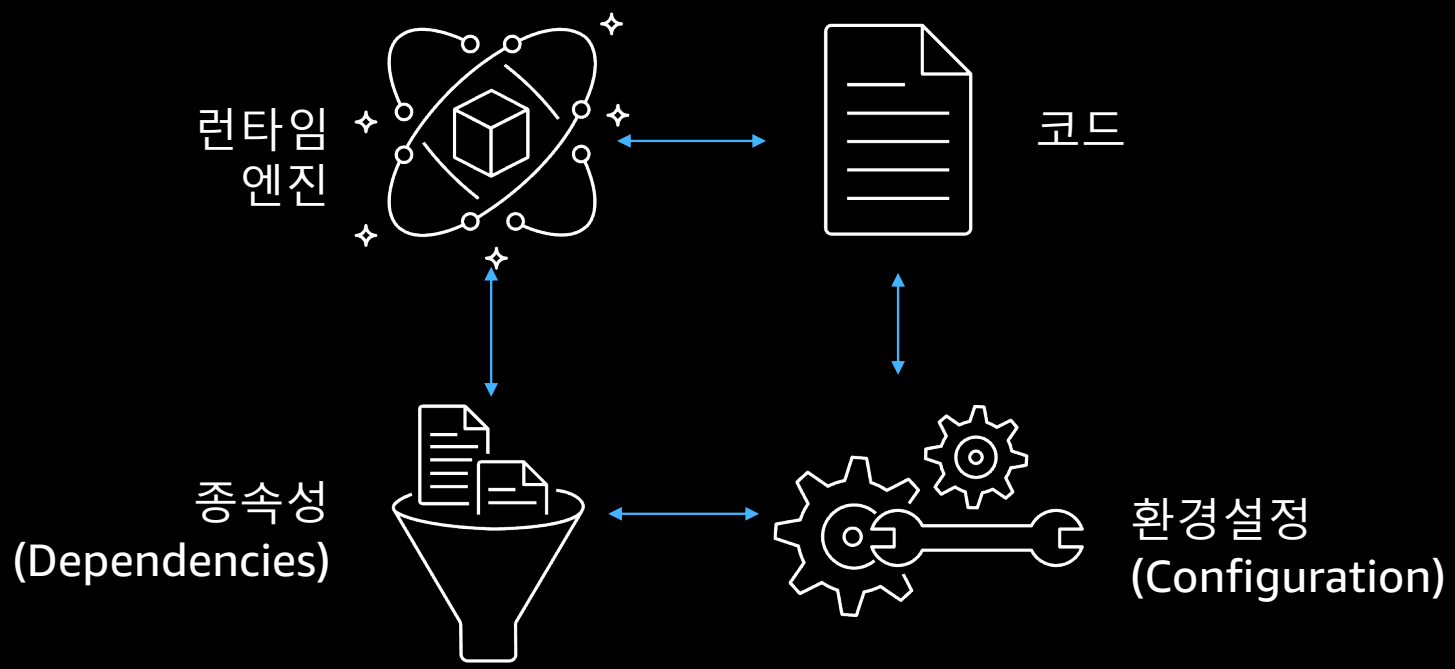
5. EKS Demo

6. Q&A

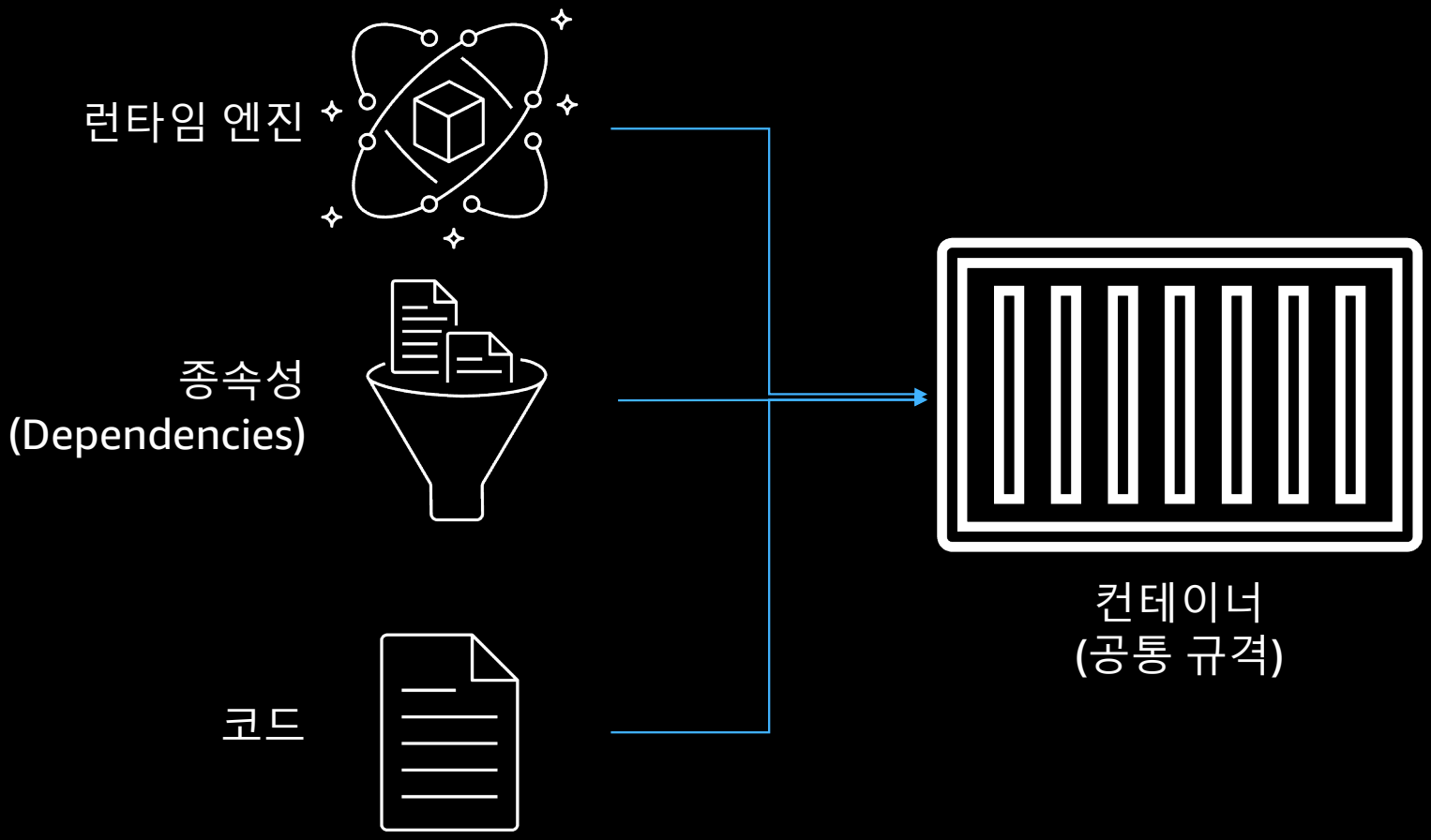
1. Container Concept

- 컨테이너의 정의와 필요성
- 컨테이너 vs 가상머신
- 컨테이너의 기반 기술
- 컨테이너의 장점

어플리케이션 환경 구성 요소

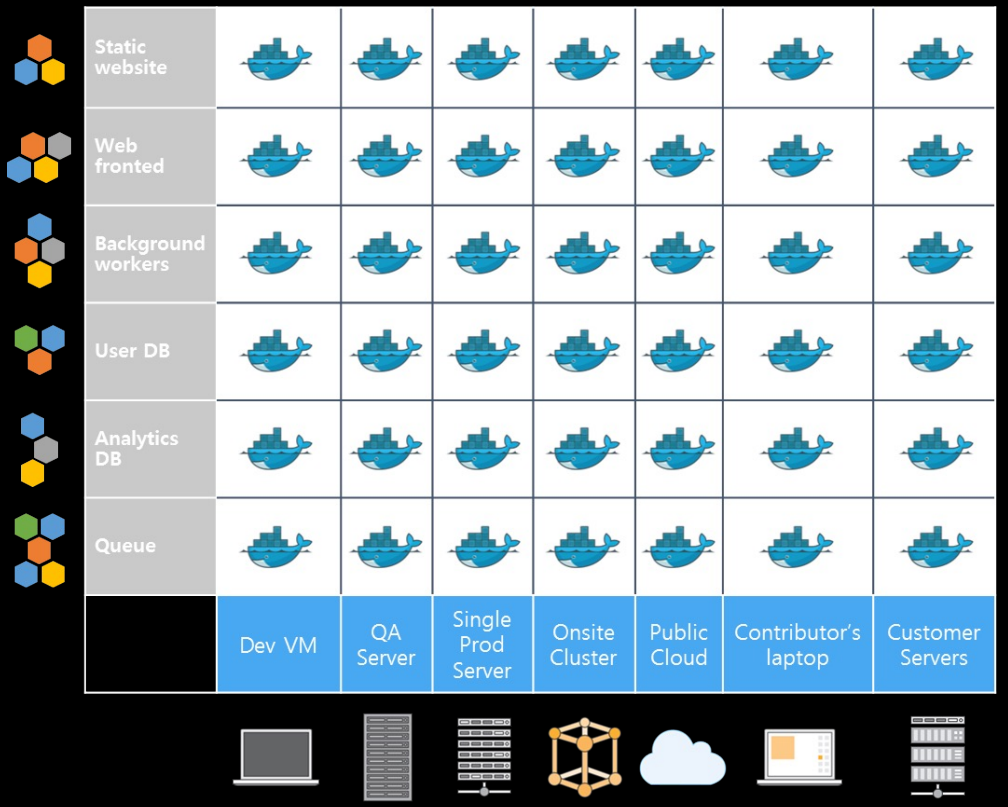


컨테이너

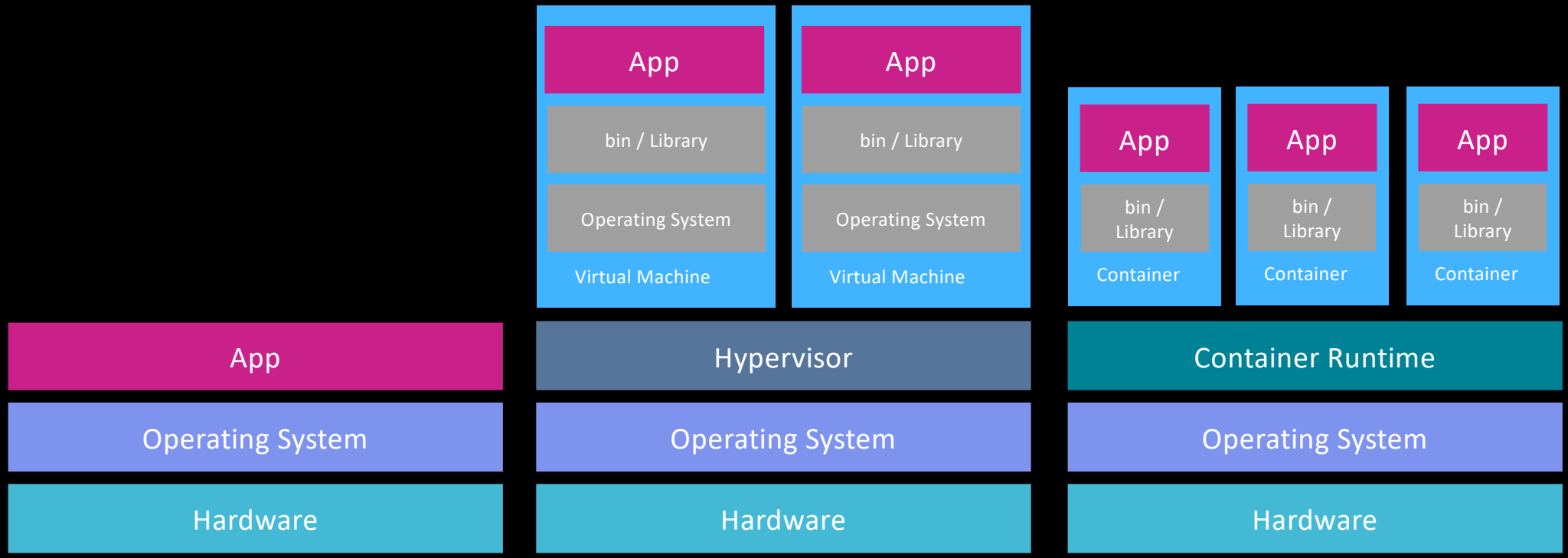


컨테이너 기술의 특징

- Container image라는 공통 규격으로 모든 SW를 패키징
- 환경에 상관없이 동일하게 배포 및 실행
- 어떤 어플리케이션이든 상관없이 동일하게 배포 및 실행

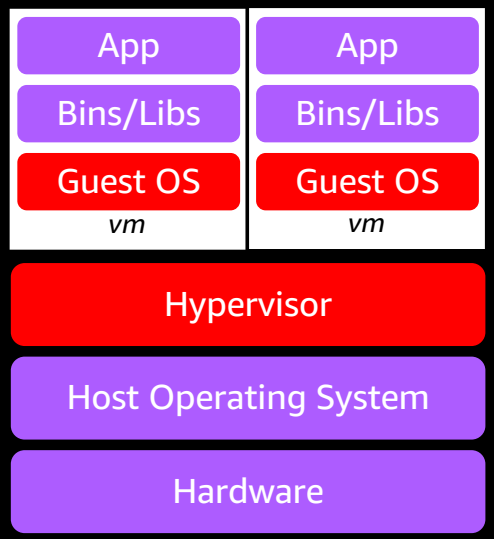


어플리케이션 개발 환경의 발전



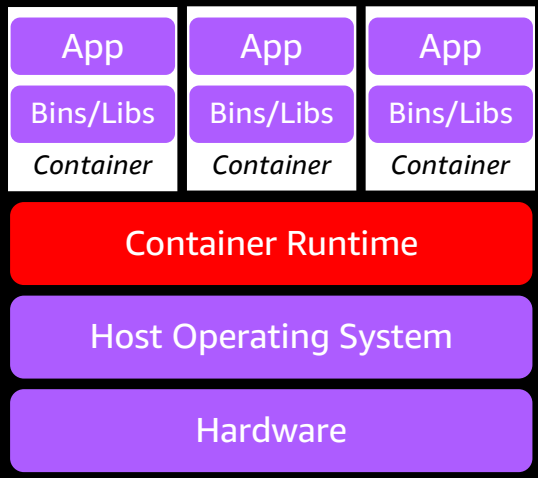
가상머신 VS 컨테이너

Virtual Machine



- Hypervisor를 이용한 H/W 가상화 기술
- Hypervisor 위에 Guest OS 별도 설치 필요
- Application간 강력한 격리 수준 제공

Container

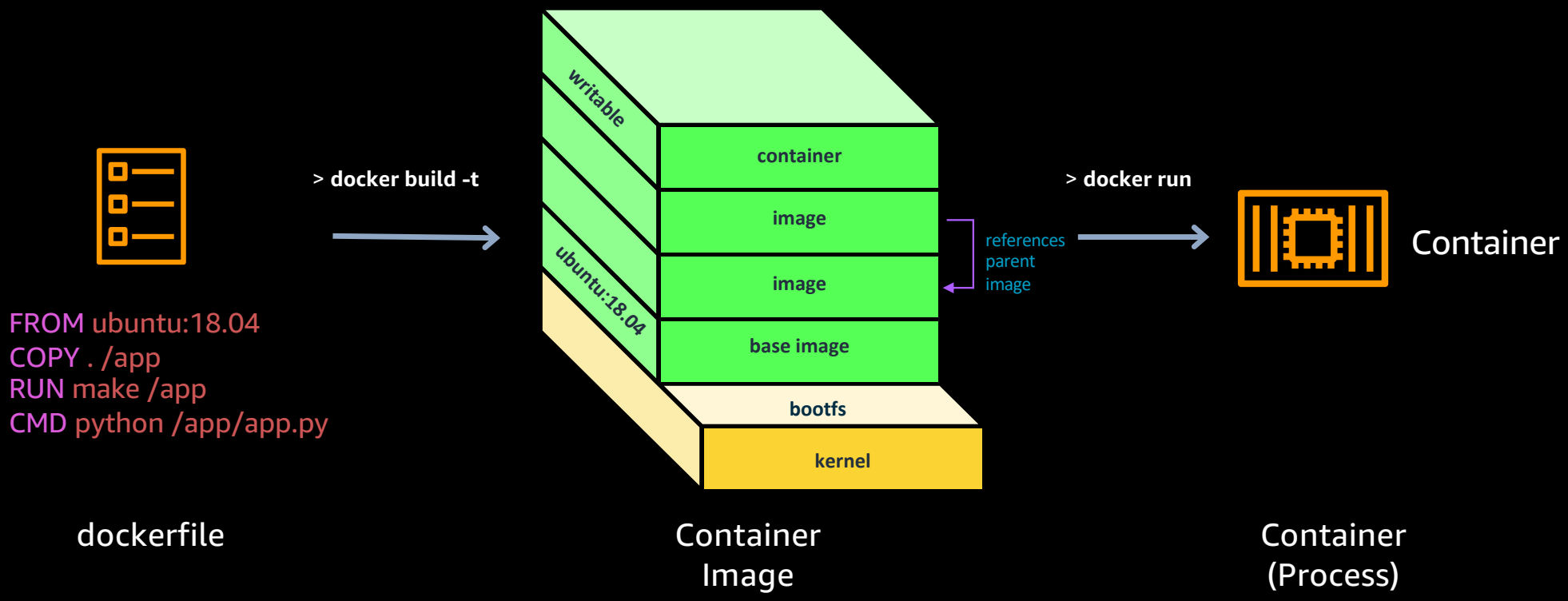


- Container Runtime을 이용한 가상화 기술
- Application이 Host OS의 Kernal을 공유
- VM대비 경량, 뛰어난 성능

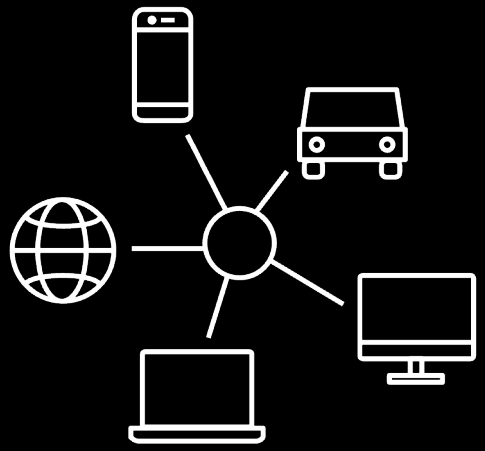
컨테이너의 기반 기술

- **cgroups**: 리소스 사용 제한, 감사, 격리
- **Namespaces**: 리소스를 격리시키기 위한 메커니즘
- **Union filesystem**: r/o & r/w 레이어들을 구현하여 효율적인 스토리지 사용
- **iptables**: 호스트에서 NAT를 사용하여 프로세스 포트를 노출
- **Security policies**: capabilities, seccomp and Linux Security Modules (LSM)

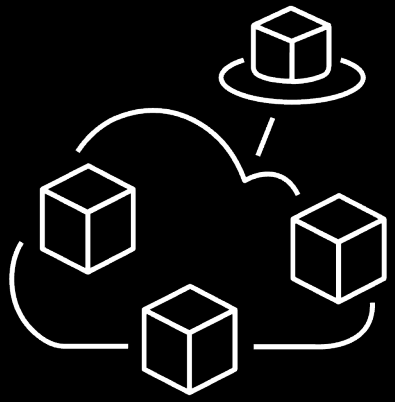
컨테이너 사용법



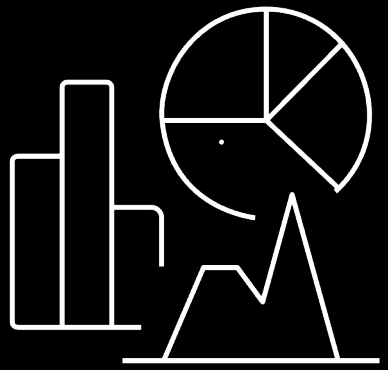
컨테이너의 장점



어디서나 안정적으로 실행



서로 다른 애플리케이션을 격리된 환경에서 독립적으로 실행

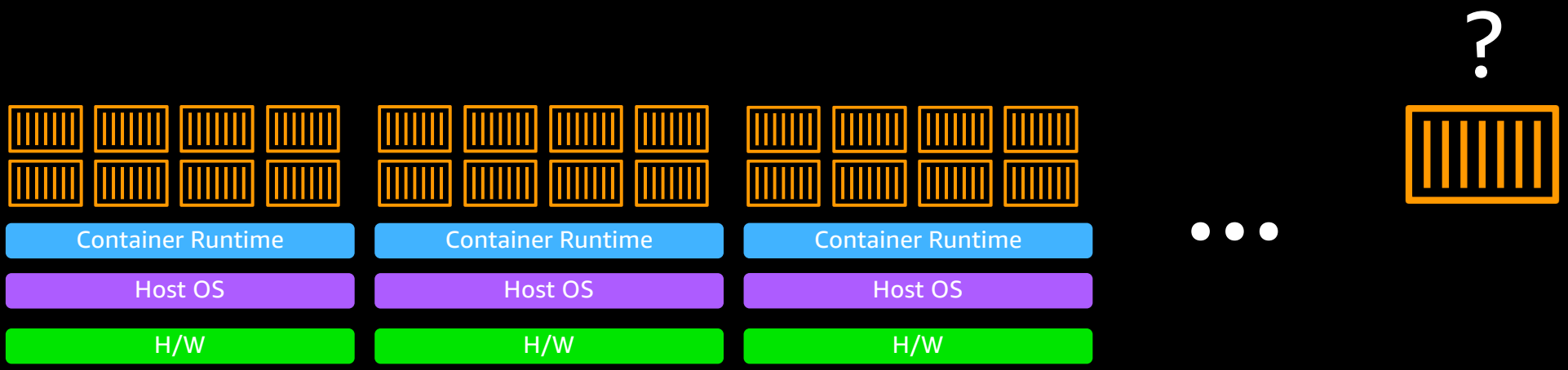


효율적인 리소스 활용

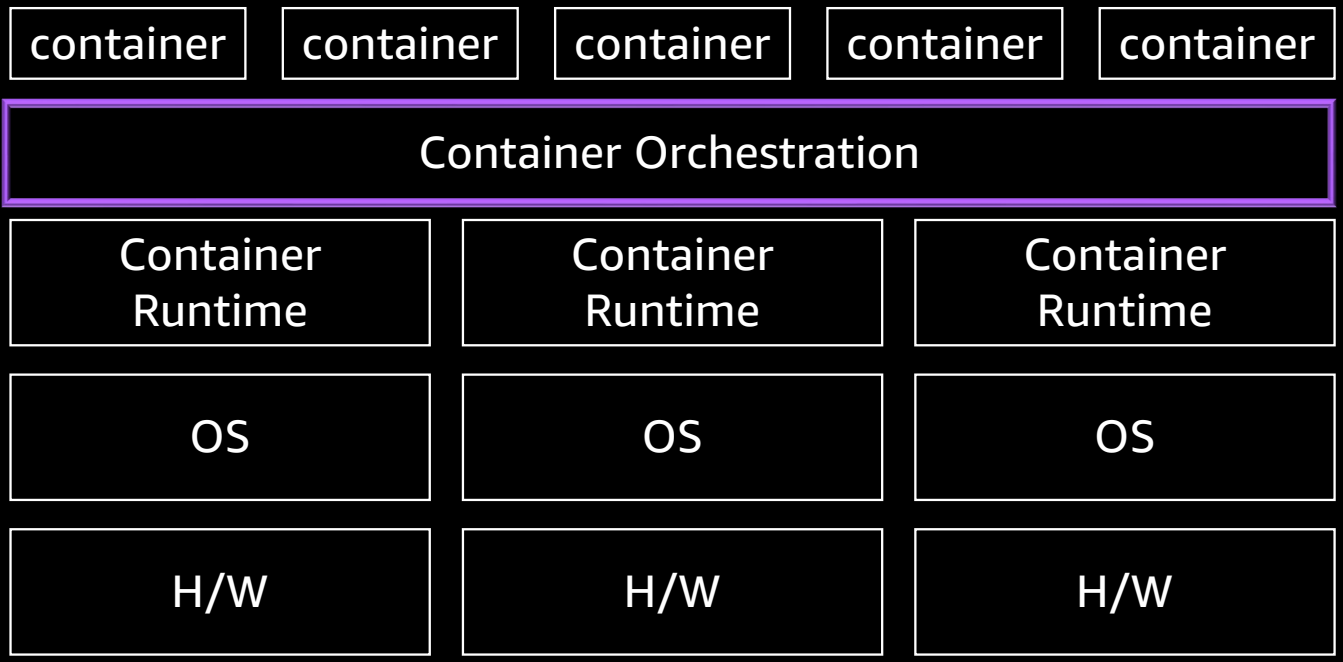
2. Container Orchestration and Kubernetes

- 컨테이너 오케스트레이션 개념 소개
- Kubernetes 주요 기능
- Kubernetes Cluster 아키텍처
- Kubernetes 사용 사례 및 이점
- Kubernetes에서의 보안 고려 사항

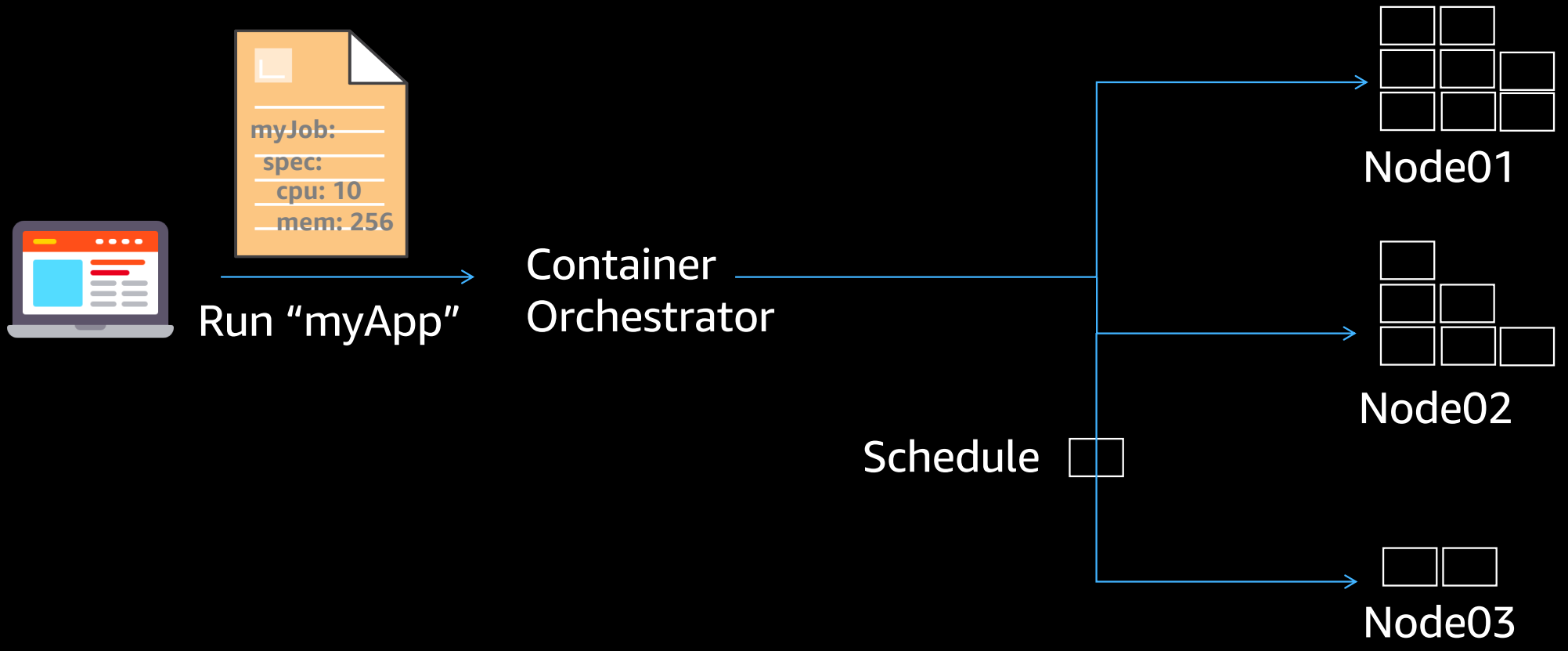
컨테이너 관리의 어려움



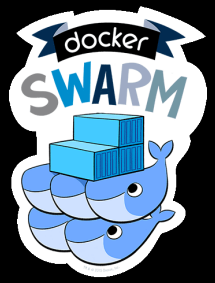
컨테이너 오케스트레이션의 등장



컨테이너 오케스트레이션의 역할

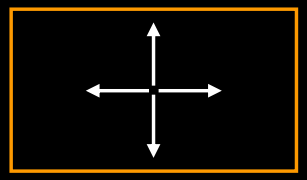


다양한 컨테이너 오케스트레이션 솔루션



Kubernetes란 무엇인가?

Open source container management platform



Helps you run containers at scale

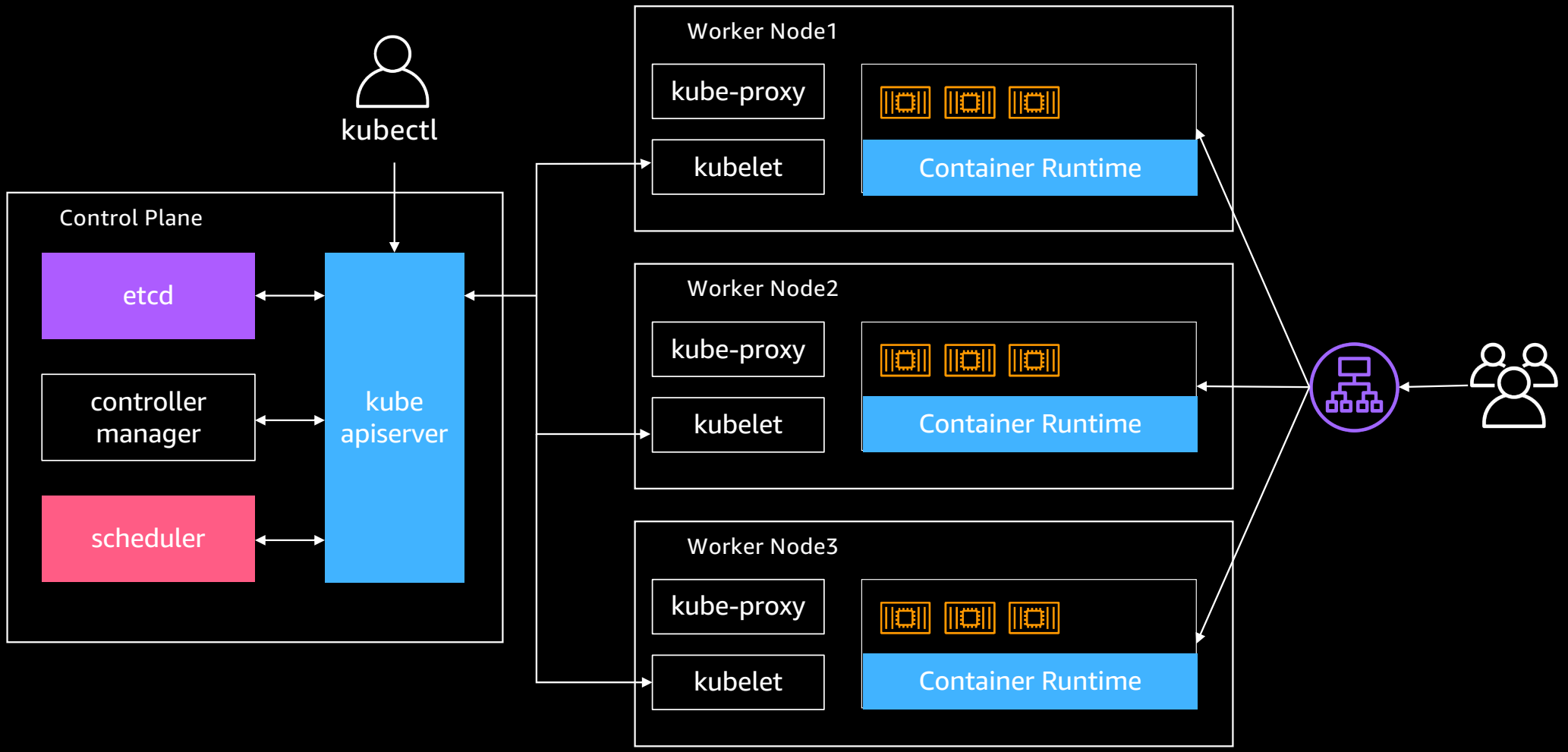
Gives you primitives for building modern applications

https://kubernetes.io/docs/tutorials/kubernetes-basics/

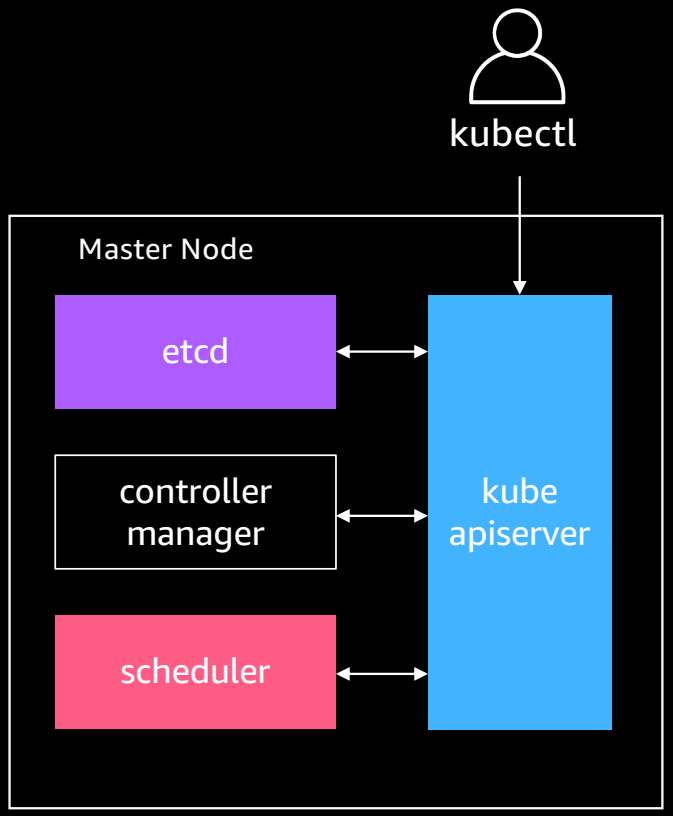
Kubernetes의 기능

- **서비스 디스커버리와 로드 밸런싱** : 컨테이너에 대한 트래픽이 증가할 때, 배포가 안정적으로 이뤄질 수 있도록 네트워크 트래픽을 로드 밸런싱
- **스토리지 오케스트레이션** : 원하는 스토리지 시스템을 자동으로 마운트할 수 있는 기능 제공
- **롤아웃/롤백** : 현재 상태를 원하는 상태로 설정한 속도에 따라 변경
- **리소스 배분**: 각 컨테이너 필요한 CPU 및 메모리 양 설정
- **자동 복구** : 실패한 컨테이너를 다시 시작하고, 교체하며 응답하지 않는 컨테이너를 종료
- **시크릿과 구성 관리** : OAuth 토큰 및 SSH 키와 같은 중요한 정보 저장 및 관리

Kubernetes Cluster 아키텍처



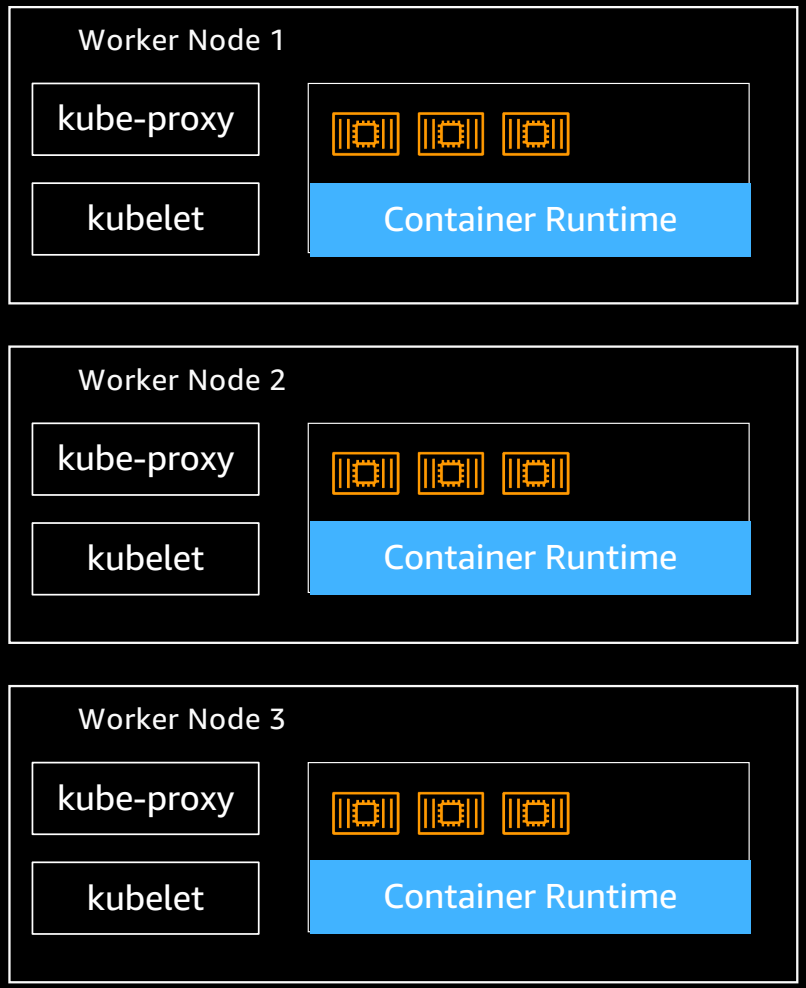
Control Plane



- **apiserver**
  - Kubernetes API를 노출하는 컴포넌트. kubectl 등으로부터 리소스를 조작 명령을 받아 controller로 전달
  - Control Plane과 Worker Node의 연결을 담당.
- **etcd**
  - 고가용성을 갖춘 분산 키-값 스토어
  - kubernetes cluster의 설정값 및 이벤트를 저장
- **scheduler**
  - 노드를 모니터링하고 pod를 배치할 적절한 노드를 선택
- **kube-controller-manager**
  - Pod를 복제하거나 노드 운영 등 각 리소스를 제어하는 컨트롤러들을 감독하고 실행

Worker Node

- **kubelet**
  - 클러스터의 각 노드에서 실행되는 에이전트
  - Control Plane의 명령을 처리
- **kube-proxy**
  - Pod의 네트워크 통신을 담당
- **container runtime**
  - 컨테이너 실행을 담당하는 소프트웨어
  - 도커(Docker), containerd 등

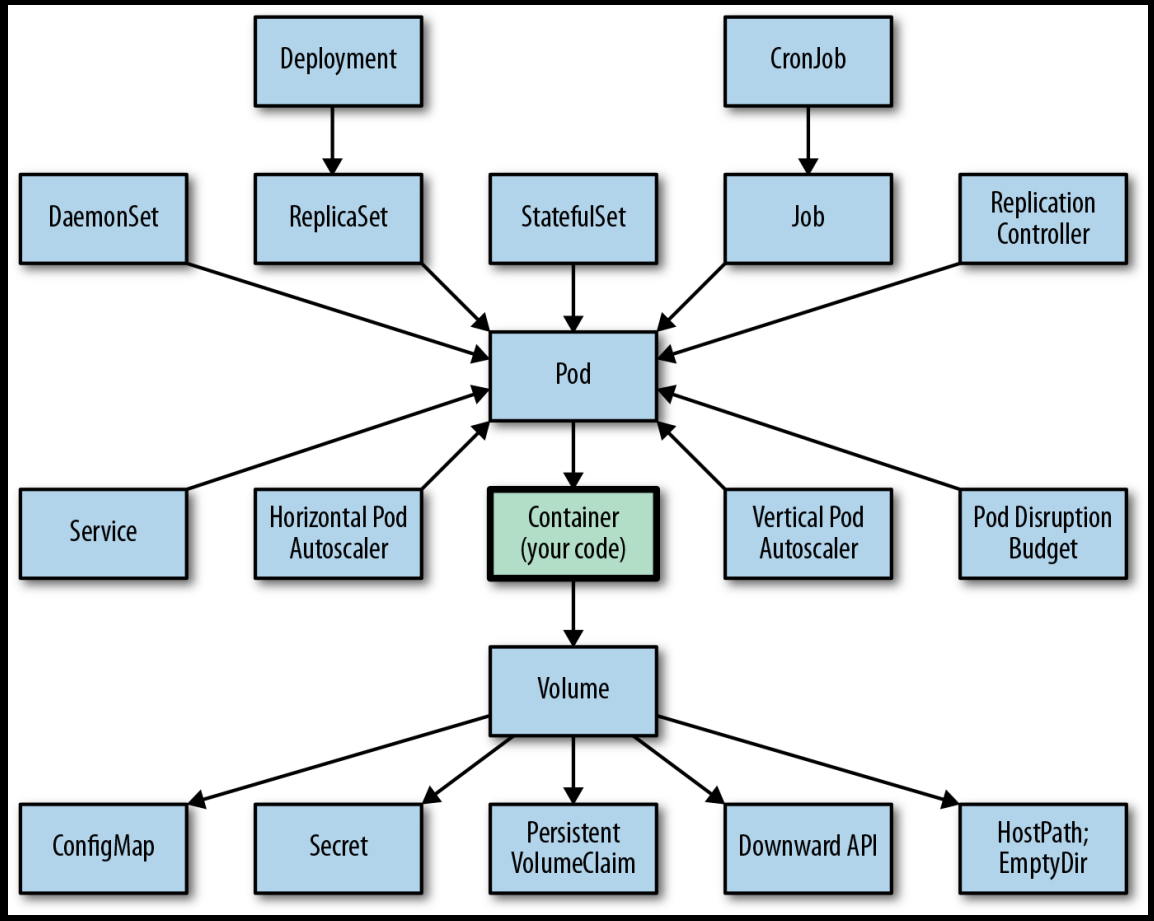


Kubernetes Manifest(Yaml)

- **Declarative Management**
  - Yaml 파일로 각 object의 최종 형태를 정의하면 Kubernetes가 그 형태를 meet하게끔 노력함.
- **오브젝트 설정을 위한 두 가지 필드**
  - Spec : 오브젝트가 가져야 할 요구되는 상태(desired status)를 기술하는 곳
  - Status : 오브젝트의 현재 상태를 기술하고 시스템에 의해 업데이트 됨.

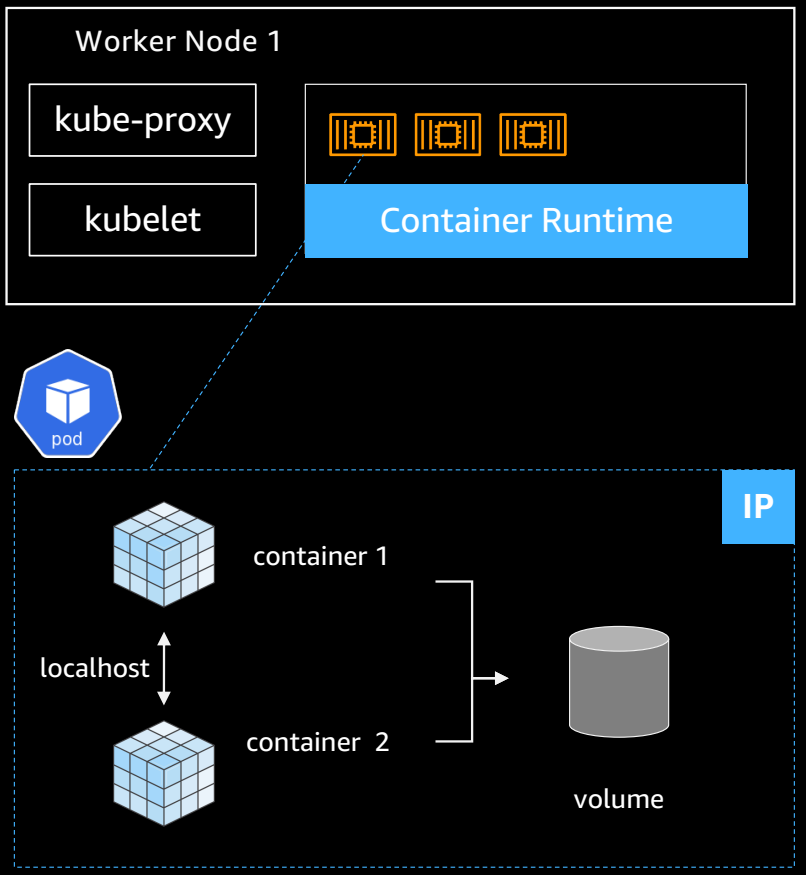
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-nodejs-backend
  namespace: default
spec:
  replicas: 3
  selector:
    matchLabels:
      app: demo-nodejs-backend
  template:
    metadata:
      labels:
        app: demo-nodejs-backend
    spec:
      containers:
        - name: demo-nodejs-backend
          image: public.ecr.aws/y7c9e1d2/joozero-repo:latest
          imagePullPolicy: Always
          ports:
            - containerPort: 3000
```

주요 Kubernetes 오브젝트



Pod

- Pod은 Kubernetes에서 생성하고 관리할 수 있는 배포 가능한 가장 작은 컴퓨팅 단위를 의미
- Pod은 애플리케이션 컨테이너(하나 또는 다수), 스토리지, 네트워크 등의 정보를 포함
- Pod의 특징
  - Pod에는 각각 고유한 private IP 할당
  - pod 안에 있는 모든 컨테이너는 pod의 IP를 공유(port로 구분)



3. EKS Overview

- Amazon EKS 소개
- EKS 아키텍처
- EKS 장점
- EKS 보안

Kubernetes 시작하기

Self hosted(https://kubernetes.io/docs/setup/production-environment/tools/)

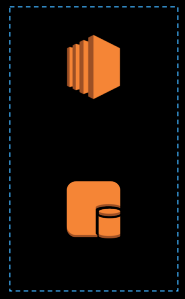
- kubeadm
- Kubespray
- Kops

Managed

- AWS: EKS(Elastic Kubernetes Service)
- Azure: AKS(Azure Kubernetes Service)
- GCP: GKE(Google Kubernetes Engine)
- NCP: NKS(Naver Kubernetes Service)

Amazon EKS

Elastic Kubernetes Service



관리형 Kubernetes 컨트롤 플레인 및 워커노드

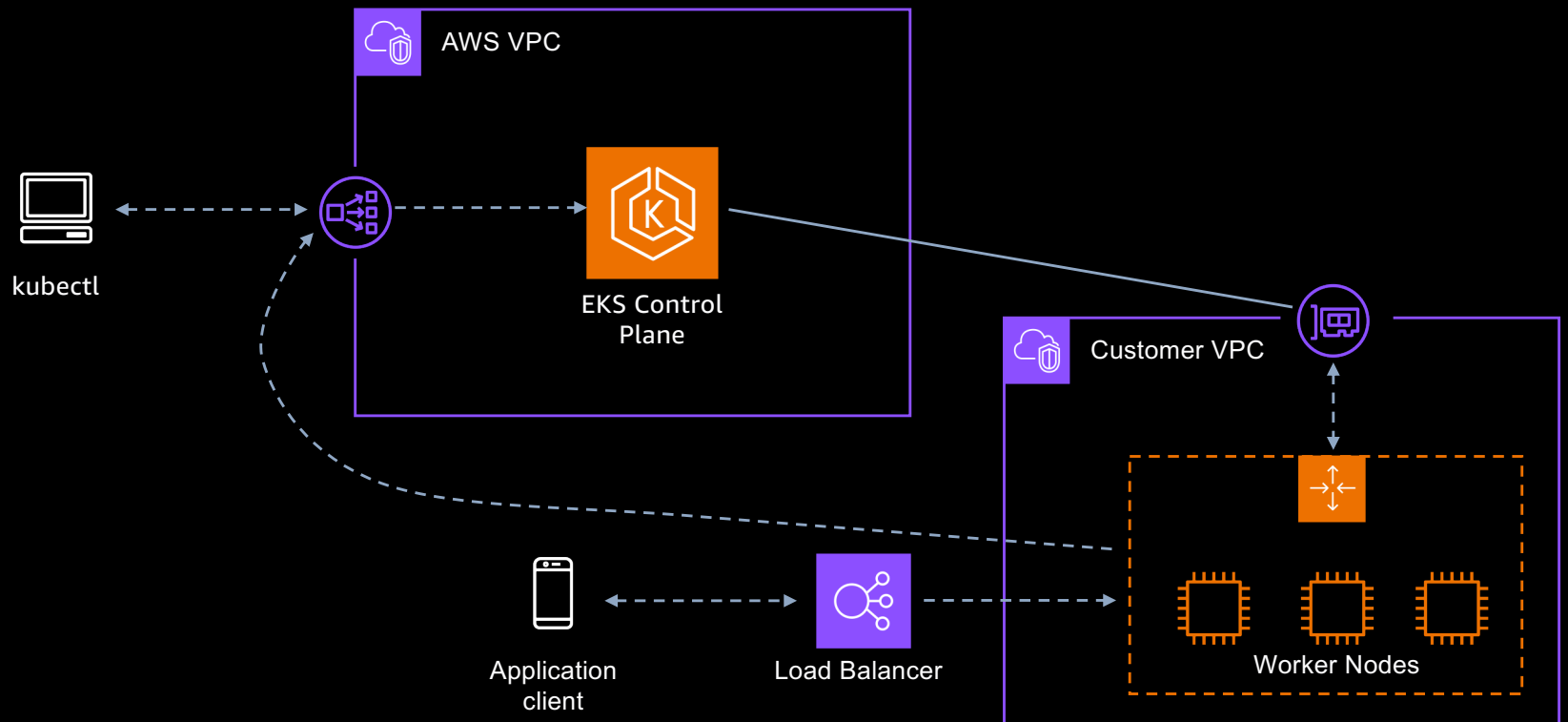
고가용성

자동 버전 업그레이드

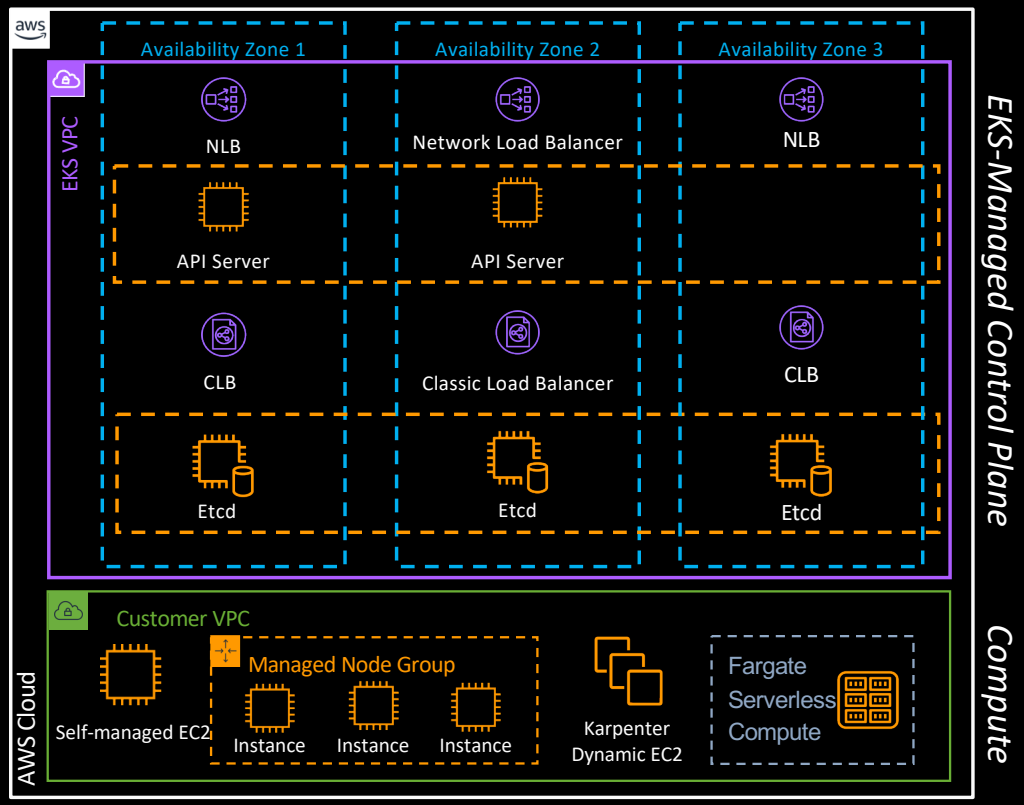
CloudTrail, CloudWatch, ELB, IAM, VPC, PrivateLink , Appmesh,

타 AWS 서비스들과의 통합

Amazon EKS High Level Architecture



Amazon EKS cluster architecture 상세



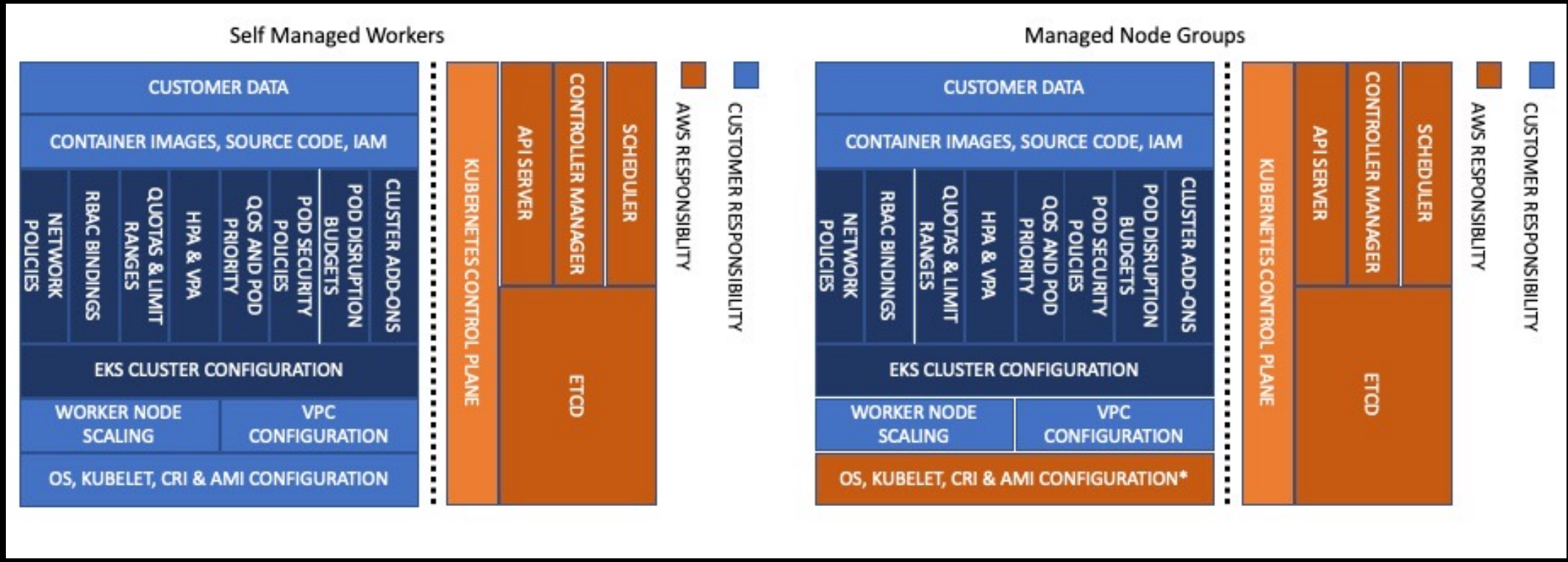
AWS managed control plane

- **API Server와 Etcd HA 구성**
- **트래픽에 따라 control plane 자동 scale-out**

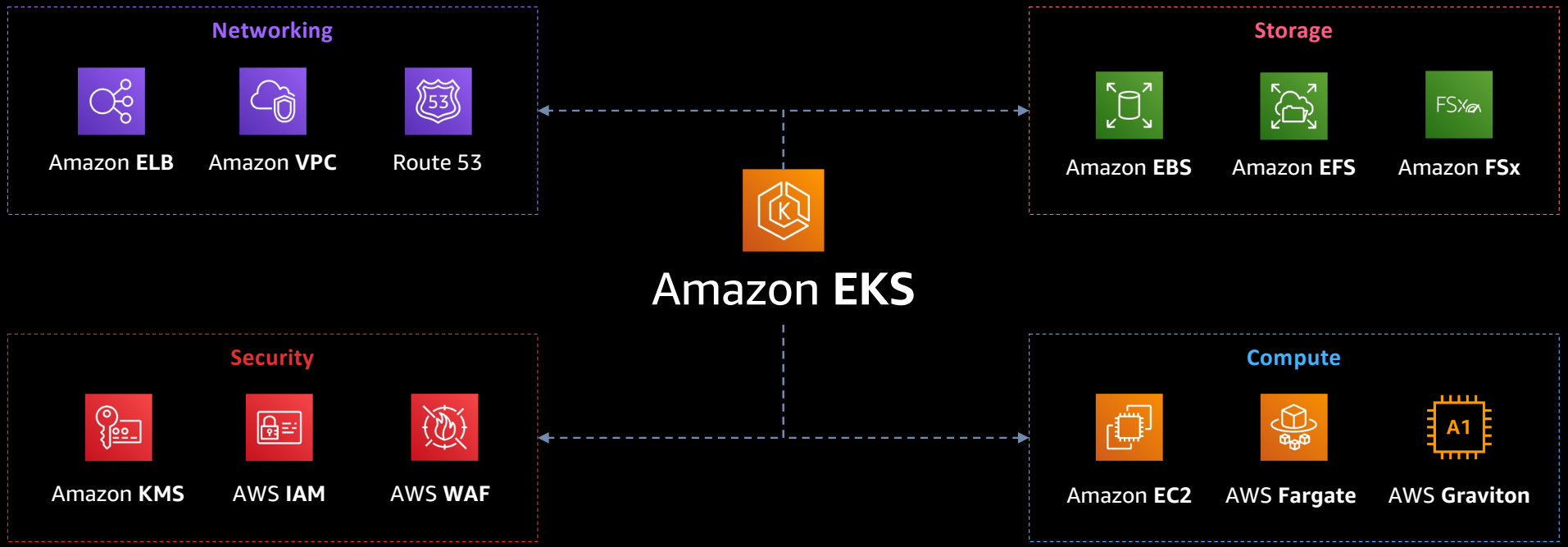
Cluster compute

- **Self-managed EC2 instances**
  사용자가 EC2 instance를 사용자 VPC에 생성하고 직접 관리
- **EKS managed nodes**
  AWS가 EC2 instance를 사용자 VPC에 생성하고 대신 관리
- **Karpenter**
  Karpenter가 EC2 instance를 사용자 VPC에 생성하고 대신 관리
- **AWS Fargate**
  instance가 생성되지 않는 serverless 타입

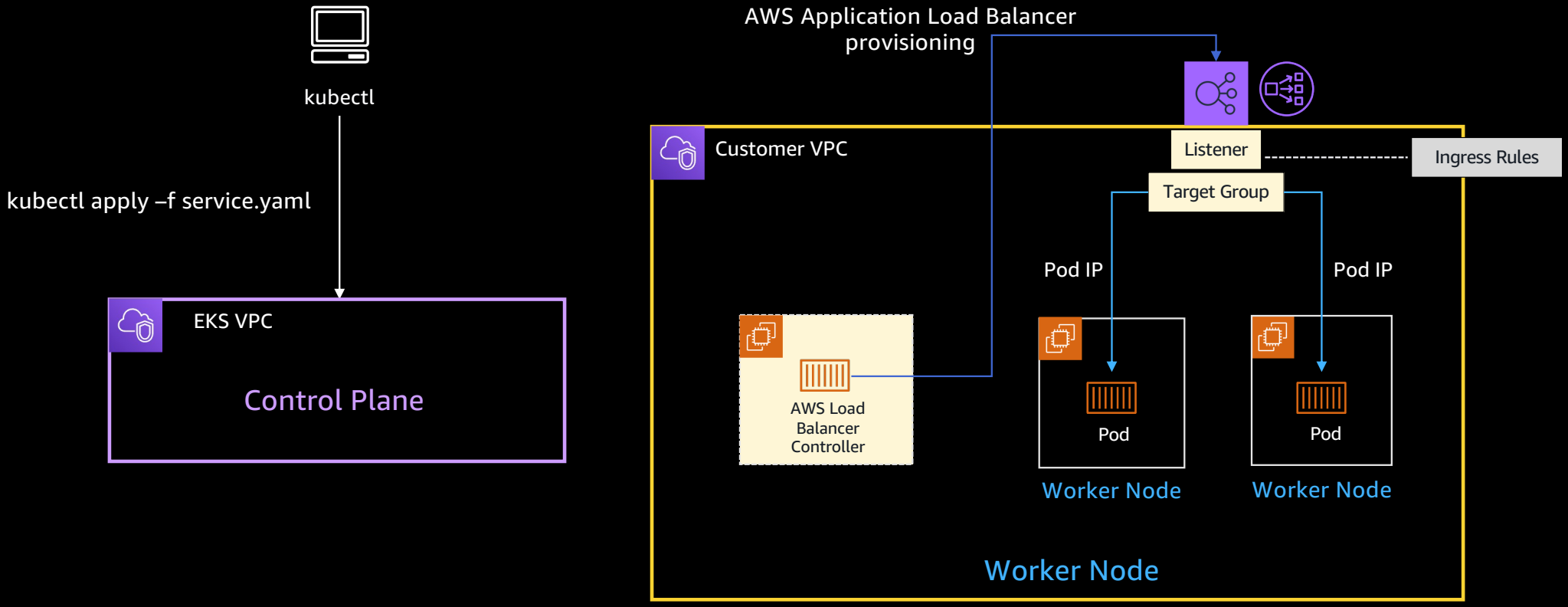
AWS EKS Shared Responsibility



타 AWS 서비스와의 연계

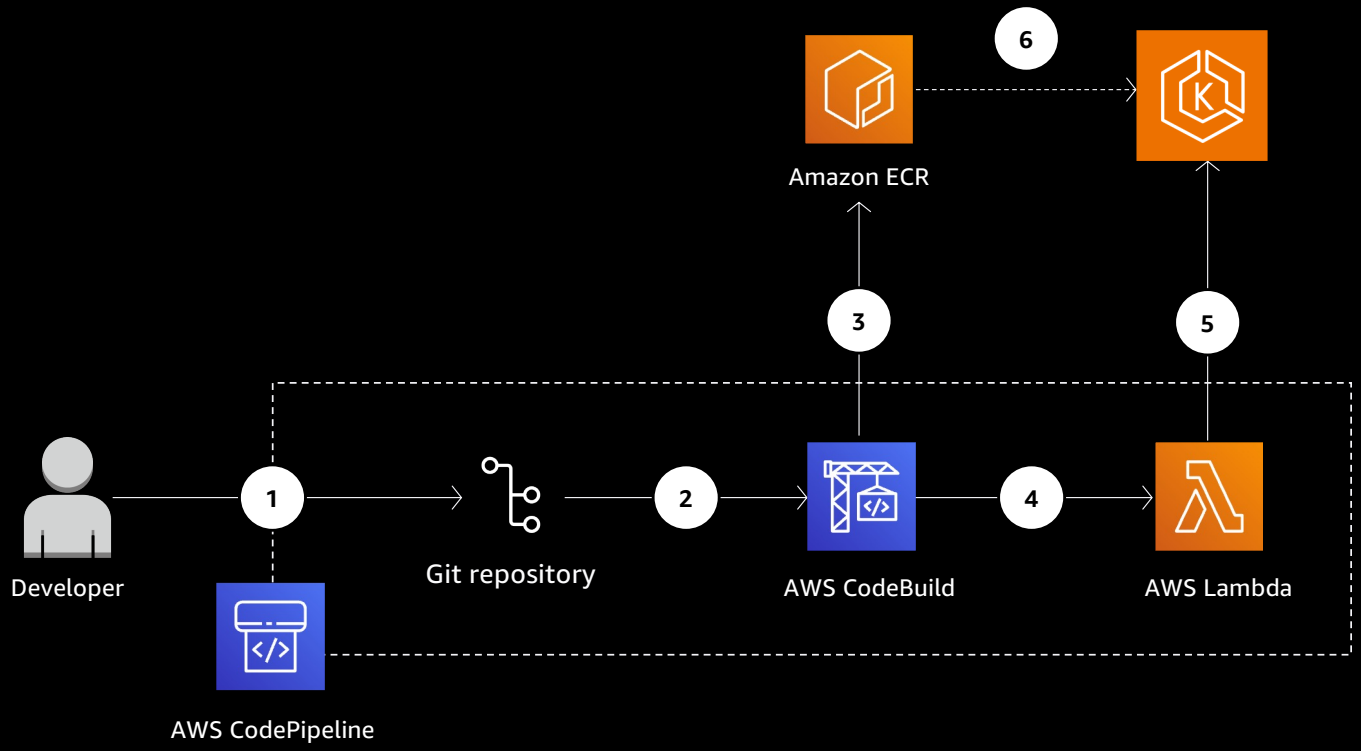


AWS Load Balancer 와의 연계



kubernetes-sigs/aws-load-balancer-controller

CI/CD Pipeline 구축 연계



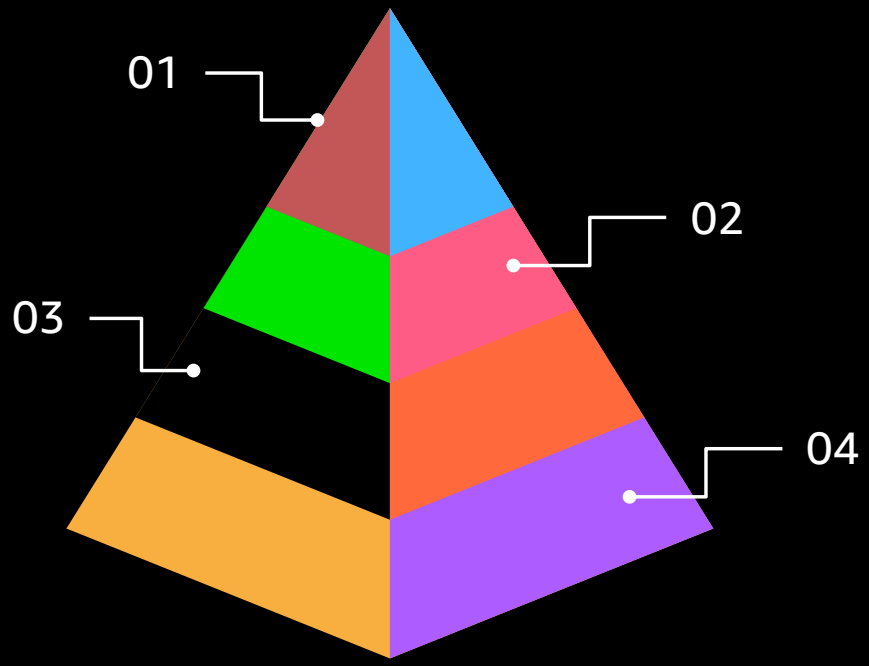
1. Developers continuously integrate changes into a release branch hosted within a repo
2. Triggers an execution of the pipeline when a new version is found, builds a new image with build id
3. Pushes the newly built image tagged with build id to ECR repo
4. Invokes a Lambda function to trigger application deployment
5. Leverages Kubernetes Python SDK to update a deployment
6. Fetches new container image and performs a rolling update of deployment

https://github.com/aws-samples/aws-kube-codesuite

4. Container Security in EKS

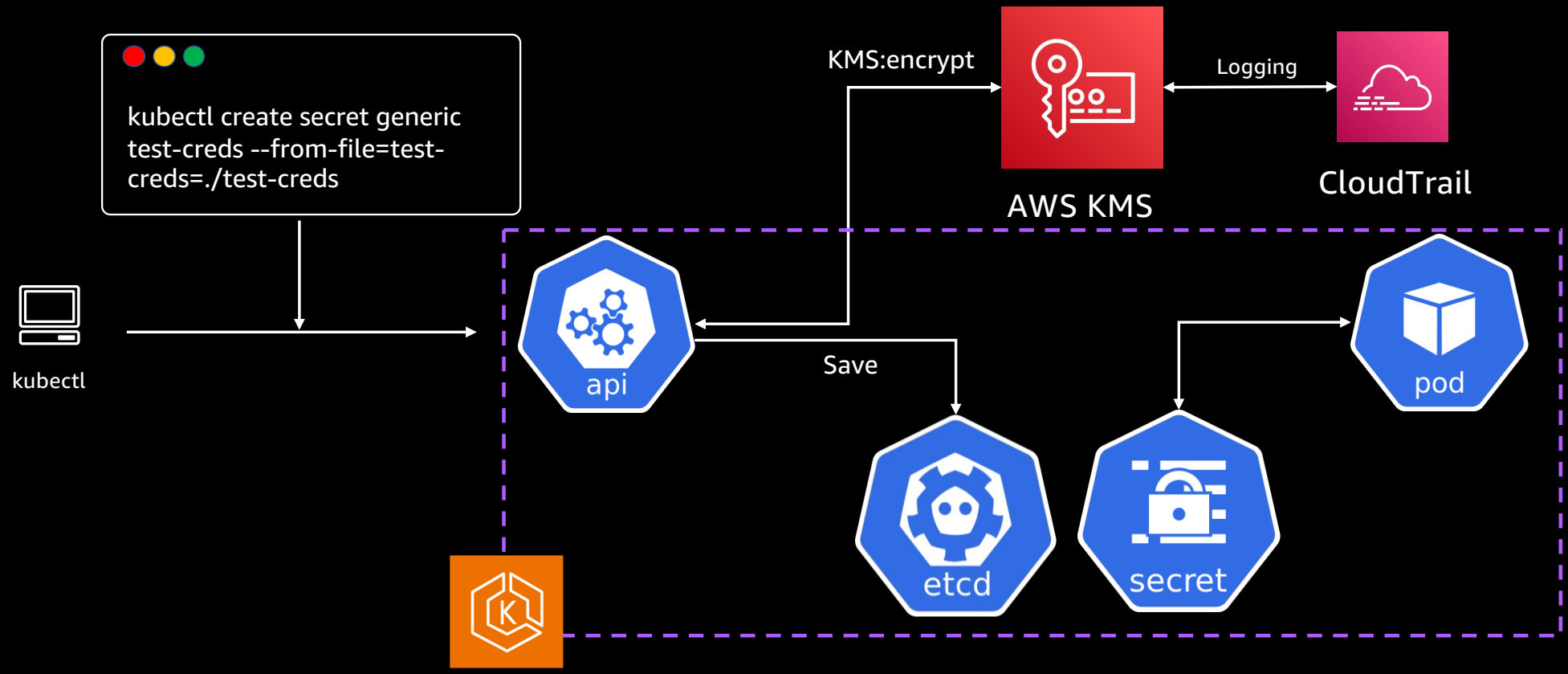
- 컨테이너 환경의 보안 4C
- EKS에서의 보안

컨테이너 환경에서 보안의 4C



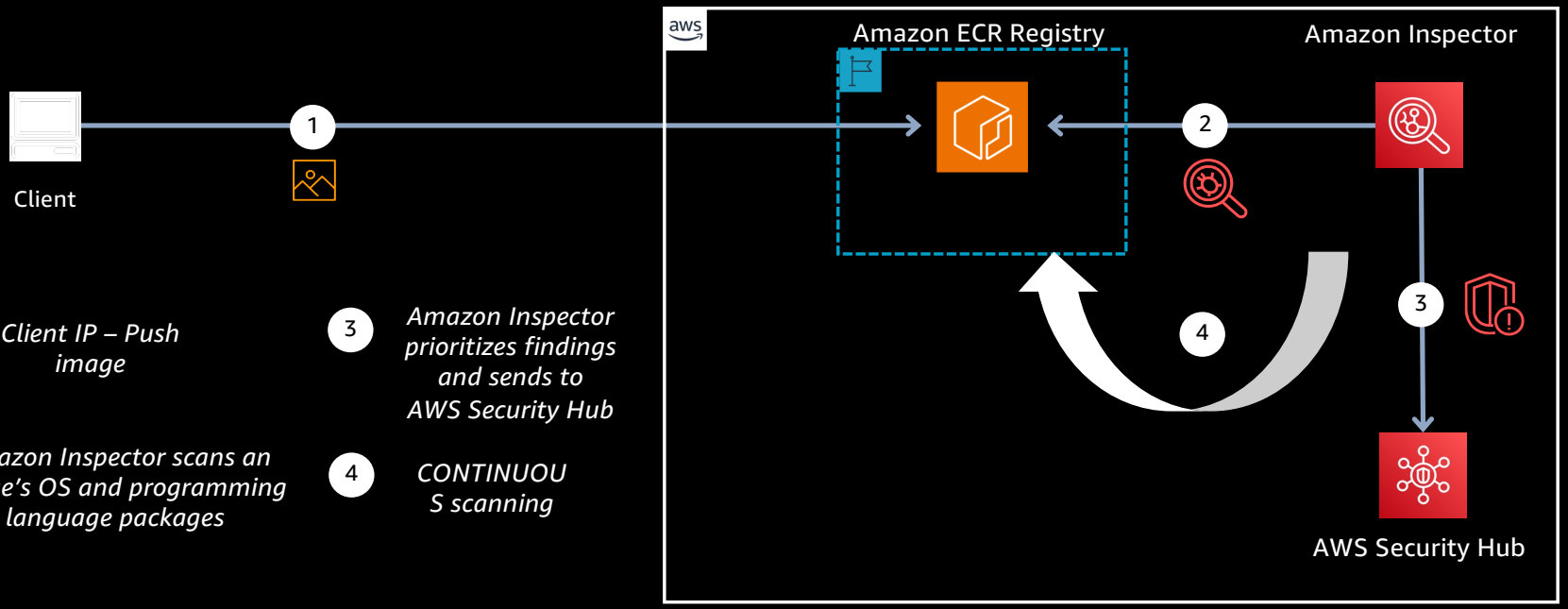
1. Code
2. Container
3. Cluster
4. Cloud

Code 보안 - AWS KMS 연계



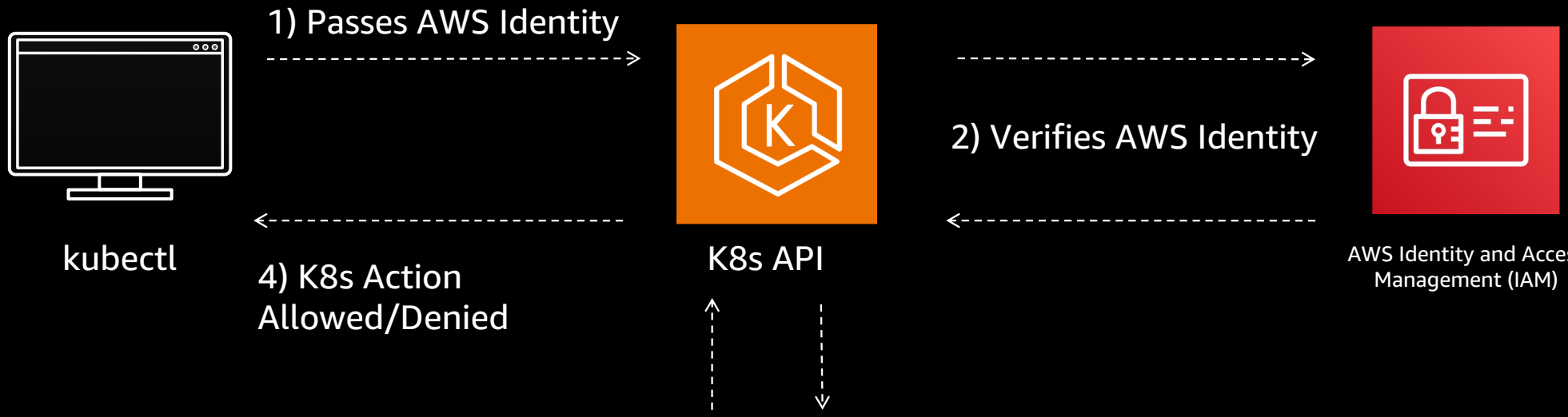
Container 보안 - AWS ECR Container Image Scanning

Continuous scanning and enhanced detections



1. *Client IP – Push image*
2. *Amazon Inspector scans an image's OS and programming language packages*
3. *Amazon Inspector prioritizes findings and sends to AWS Security Hub*
4. *CONTINUOUS scanning*

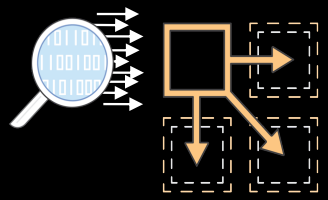
Cluster 보안 - AWS IAM Authentication 연계



Uses: [aws-iam-authenticator]

Cluster 보안 - Amazon GuardDuty 연계

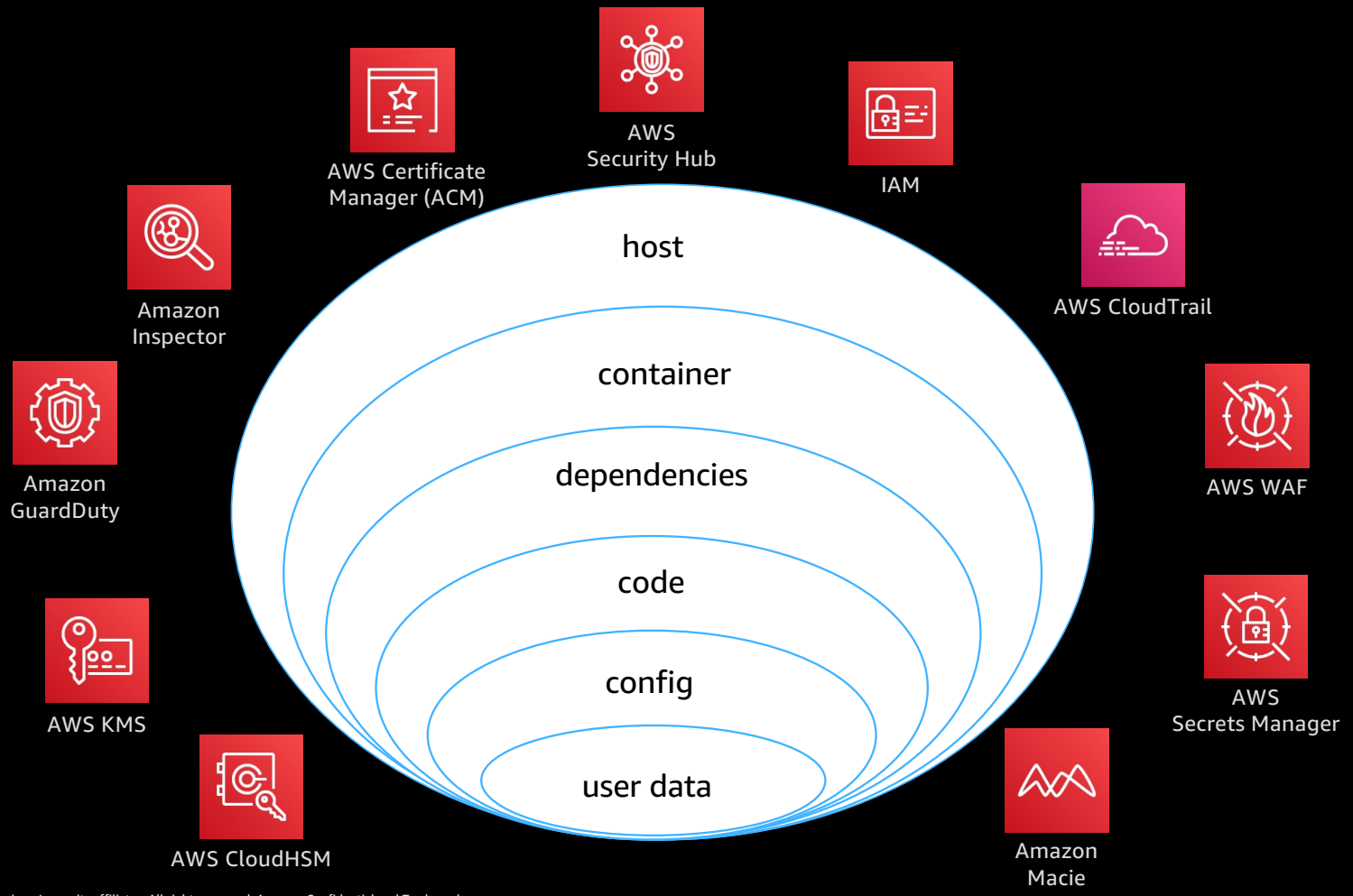
With a single click Amazon GuardDuty for Kubernetes protection will continuously monitor Amazon Elastic Kubernetes Service (Amazon EKS) cluster control plane activity by analyzing the EKS audit logs (across all accounts).



Using threat intelligence it will detect and report findings for Amazon EKS clusters that are accessed by known malicious actors or from Tor nodes, API operations performed by anonymous users that might indicate a misconfiguration, and misconfigurations that can result in unauthorized access to Amazon EKS clusters.

By using machine learning (ML) models, GuardDuty can identify patterns consistent with privilege-escalation techniques, such as a suspicious launch of a container with root-level access to the underlying Amazon Elastic Compute Cloud (Amazon EC2) host

기타 AWS security 관련 서비스 연계



5. EKS Demo

Q&A

Thank you!

Woongsik Kim
kwsryan@amazon.com