Cloud Native Forum

쿠버네티스 교육 과정

쿠버네티스 보안과 거버넌스

무엇을 막고 무엇을 열지 정하는 기준
보안 경계 · 접근 제어 · 워크로드 보안 · 거버넌스

교육 과정	DATE
6권 중 4권	2026-08

PUBLIC

CONTENTS

오늘 다룰 다섯 가지

보안 경계와 층 · 접근 제어와 감사 · 실행 권한 통제 · 공급망과 런타임 보안 · 정책 기반 거버넌스

01 보안 경계와 층
무엇을 어디까지 지킬 것인가

02 접근 제어와 감사
권한을 넘기고 기록으로 남기기

03 실행 권한 통제
컨테이너가 할 수 있는 일 줄이기

04 공급망과 런타임 보안
들어온 것을 확인하고 도는 것을 보기

05 정책 기반 거버넌스
점검을 사람 손에서 떼어 내기

KEY FOCUS 기본값을 그대로 두지 않는 것에서 클러스터 보안이 시작됨

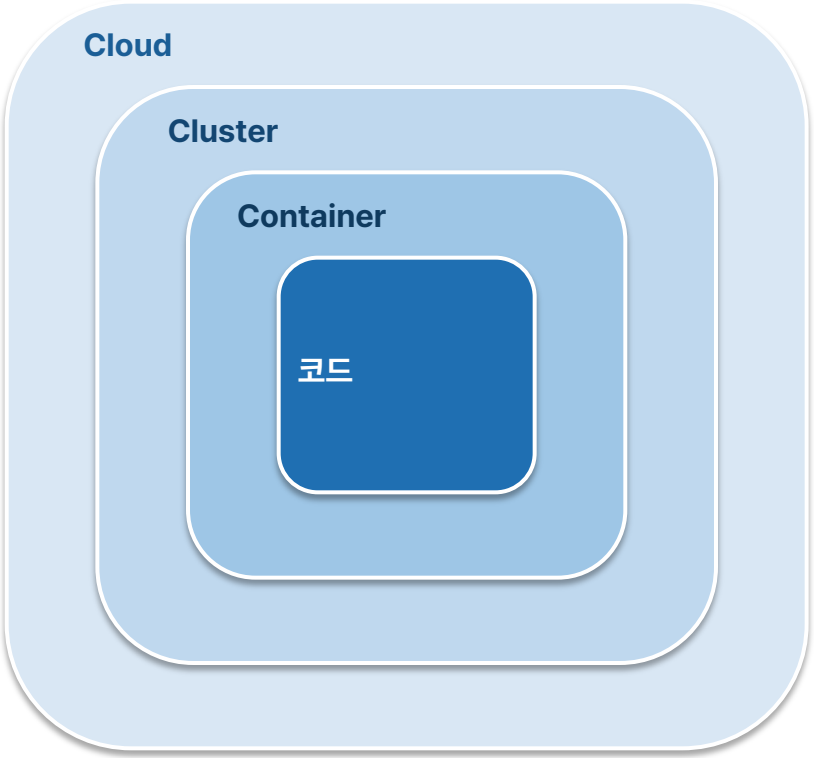
표시된 항목은 쿠버네티스 공식 아이콘으로 나타낸 구성 요소임

PART 1

보안 경계를 나누는 층

무엇을 어디까지 지켜야 하는지 층으로 갈라 봄

컨테이너 보안 바깥 층이 뚫리면 안쪽도 위험 — 네 층을 겹겹이 지켜야 성립하는 모델



Cloud

Cluster 를 배포하는 클라우드 서비스의 기반

Cluster

Container 를 관리·운영하는 컨테이너 오케스트레이터
오케스트레이션용 다양한 구성요소 포함

Container

코드를 포함하고 Cluster 에서 실행되는 컨테이너
코드 실행을 위한 라이브러리 포함

코드

Container 에 포함된 애플리케이션 코드

① 신뢰의 전제
안쪽 층의 보안은 바깥 층이 안전하다는 가정 위에서만 성립

② 강화 순서
Cloud → Cluster → Container → 코드, 바깥부터 점검해야 방어가 완성

③ 책임 주체 분리
층마다 통제 수단과 책임자가 달라 조직 전체가 함께 봐야

Source: kubernetes.io/docs/concepts — 클라우드 네이티브 보안 4C 모델

NIST SP 800-190 — 컨테이너 기술 관련 위험

컨테이너 보안 이미지 · 레지스트리 · Orchestrator · Container 네 축의 위험 — 미국 NIST 표준 문서

이미지

- 알려진 취약점 존재
- 기밀 정보 포함
- 신뢰할 수 없는 이미지 활용

Registry

- 레지스트리와의 통신 미암호화
- 취약점이 있는 이미지 저장
- 인증·승인 미흡

Orchestrator

- 클러스터 사용자에게 과도한 권한 부여
- 상이한 기밀 수준의 트래픽 혼재
- 클러스터를 구성하는 노드의 취약점

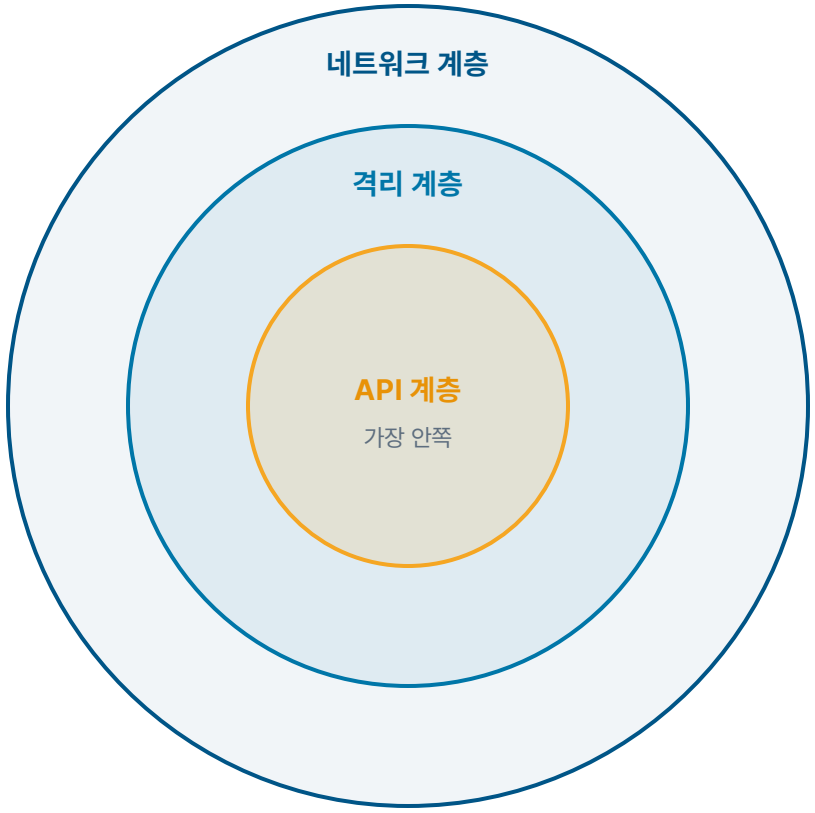
Container

- 런타임 취약점으로 인한 중단
- 컨테이너에 과도한 권한 부여
- 검사를 수행하지 않은 컨테이너 배포

Source: NIST SP 800-190 Application Container Security Guide

네트워크와 격리와 API 세 층으로 나뉘는 보안 경계

KEY INSIGHT 한 층만 걸면 나머지 두 층이 그대로 열림 — 세 층을 함께 봐야 실제 공격 경로가 닫힘



세 층의 기본값을 먼저 확인하는 것이 이번 파트의 출발점임

네트워크 계층

파드 사이 통신과 클러스터 밖으로 나가는 통신을 통제함

기본값 — 모두 허용

격리 계층

네임스페이스와 노드로 워크로드를 가르고 컨테이너 실행 권한을 제한함

기본값 — 특권 실행이 막혀 있지 않음

API 계층

누가 무엇을 할 수 있는지 통제함. 모든 변경이 여기를 지나므로 통제 지점이 됨

기본값 — 서비스 계정 토큰이 자동 주입됨

Source: Sustainable Kubernetes, builderscon (2018)

KEY INSIGHT 아무 설정도 없으면 모든 파드가 서로 닿음 — 격리는 선언해야만 생기므로 기본값을 두면 내부 이동이 자유로움

파드 하나에 대한 판정

지정하는 방식

라벨 조건으로 대상 파드를 고르고,
들어오는 통신과 나가는 통신을
따로 적음

나가는 통신은 잊기 쉬운 쪽임

⚠ 플러그인이 지원해야

폴리시를 만들어 두어도 네트워크
플러그인이 지원하지 않으면 오류 없이
무시됨

막힌 줄 알았는데 열려 있는 상태가 됨

적용을 확인하는 방법

막았어야 할 파드에서 실제로 호출해 봄 — 폴리시 파일이 있음은 사실은 차단됐다는 증거가 아님
네임스페이스마다 기본 차단 폴리시를 먼저 깔고 필요한 통신만 여는 순서가 안전함

Source: Sustainable Kubernetes, builderscon (2018)

KEY INSIGHT 논리 격리와 물리 격리의 강도가 다름 — 요구 수준을 정하지 않으면 네임스페이스로 충분한지 판단할 근거가 없음

가르고 싶은 요구 수준
아래로 갈수록 강하고 비쌈

네임스페이스

가르는 것
- 이름 공간과 권한
- 자원 쿼터

남는 공유
- 노드 · 커널 · 컨트롤 플레인

비용이 가장 낮음

노드 분리

가르는 것
- 이름 공간과 권한과 쿼터
- 커널과 하드웨어

남는 공유
- 컨트롤 플레인

여유 자원이 늘어남

클러스터 분리

가르는 것
- 앞의 모든 항목
- 컨트롤 플레인까지

남는 공유
- 없음

관리 비용이 가장 큼

고르는 원칙

요구 수준을 먼저 문장으로 적고, 그 수준을 만족하는 가장 낮은 비용의 격리를 고름 — 위에서부터 내려오는 순서로 검토함

Source: Sustainable Kubernetes, builderscon (2018)

네임스페이스만으로 부족해지는 분리 요구

KEY INSIGHT 커널과 노드를 공유한다는 사실이 남음 — 규제나 신뢰 경계가 걸리면 논리 분리로는 답이 되지 않음

가르는 대상	네임스페이스 분리	노드 분리	클러스터 분리
이름 공간과 권한	갈림	갈림	갈림
자원 쿼터	갈림	갈림	갈림
노드와 커널	공유됨	갈림	갈림
컨트롤 플레인 장애	함께 겪음	함께 겪음	갈림

아래 두 줄이 네임스페이스 분리의 한계선임

네임스페이스로 충분한 경우

같은 조직 안에서 팀이나 환경을 가르는 목적
자원 다툼과 이름 충돌을 막으려는 목적
대부분의 사내 요구가 여기에 해당함

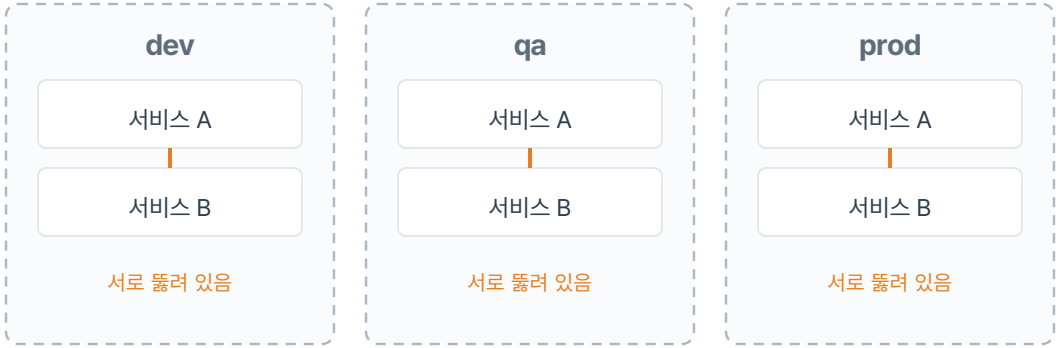
클러스터를 나눠야 하는 경우

신뢰 경계가 다른 조직이 같은 클러스터에 섞일 때
규제로 인프라 자체를 갈라야 한다고 정해졌을 때
이 경우에는 비용을 알고도 나누는 것이 맞음

Source: Sustainable Kubernetes, builderscon (2018)

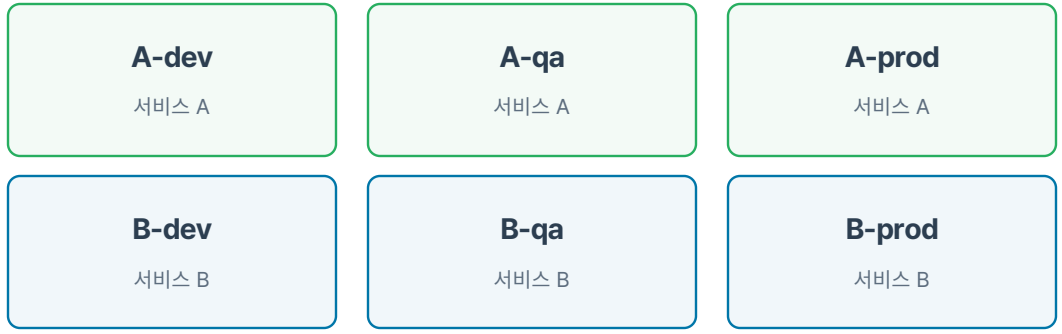
KEY INSIGHT 환경으로만 나누면 서비스 간 간섭이 남음 — 서비스명과 환경을 함께 써야 뒤따르는 정책이 서비스 단위로 걸림

환경으로만 나눈 경우



정책과 권한을 서비스 단위로 걸 자리가 없음

서비스명과 환경을 함께 쓴 경우



구획이 서비스 단위로 갈려 정책과 권한이 그대로 정렬됨

Namespace 단위로 걸 수 있는 것

자원 총량
CPU 와 메모리의 구획별 상한을 지정

통신 정책
어느 구획에서 어느 구획으로 갈 수 있는지

권한 범위
이 구획 안에서만 유효한 역할을 부여

구획으로 막지 못하는 것
클러스터 전체에 걸친 장애는 Namespace 로 막히지 않음 — 사고 반경을 더 줄이려면 클러스터를 나눠야 하고 그만큼 관리 비용이 늘어남