발표 시작

오늘 이 자리에 모인 여러분 모두 환영합니다. 이 자리에서는 쿠버네티스에 대한 깊이 있는 내용을 다룰 예정입니다. 쿠버네티스는 최근 널리 사용되는 컨테이너 오케스트레이션 플랫폼으로, 안정적이고 확장 가능한 애플리케이션 배포와 관리를 가능하게 합니다.

by 환준 남

스케줄러가 없는 경우 포드 다루기

1

스케줄러 없이 포드 실행하기

쿠버네티스 클러스터에 스케줄러가 없는 경우, 포드를 직접 노드에 할당하여 실행해야 합니다. 이를 통해 스케줄러의 기본 동작을 거치지 않고 포드를 관리할 수 있습니다.

2

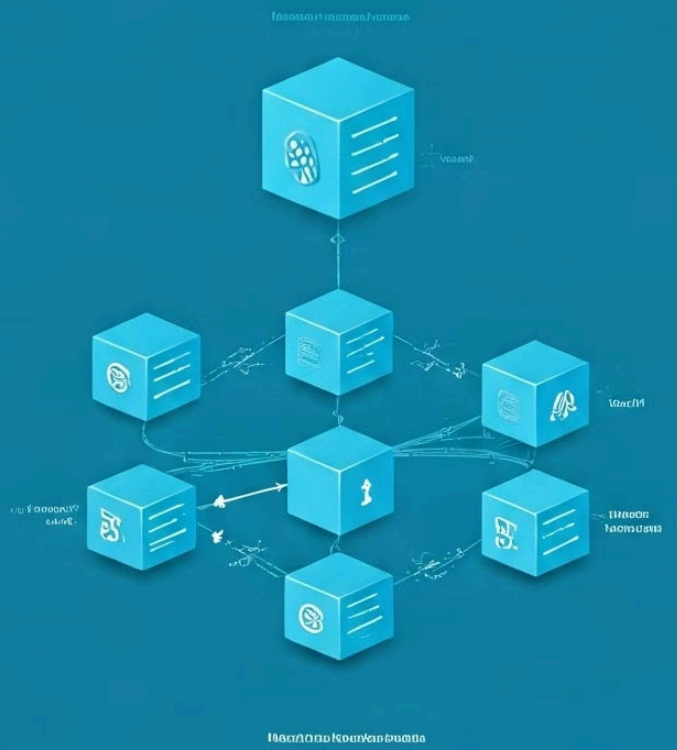
포드 명세 파일의 Node Name 필드

포드 정의 파일에는 기본적으로 Node Name 필드가 있습니다. 이 필드가 비어있는 포드는 스케줄러에 의해 자동으로 노드에 바인딩됩니다. 스케줄러가 없으면 이 필드를 직접 설정해야 합니다.

3

바인딩 객체로 포드 할당하기

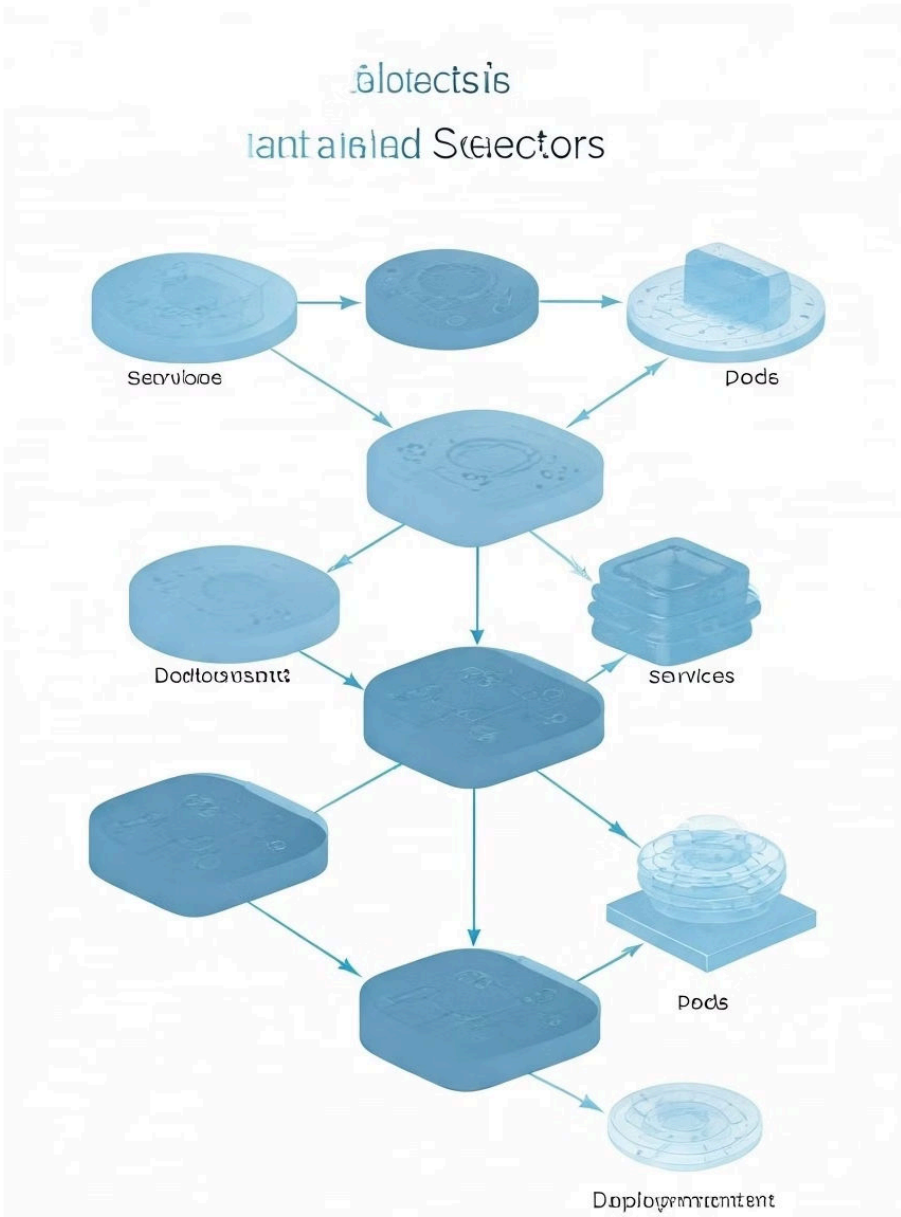
포드 생성 시 Node Name 필드를 설정하거나, 기존 포드에 바인딩 객체를 생성하여 특정 노드에 할당할 수 있습니다. 이를 통해 스케줄러 없이도 수동으로 포드를 관리할 수 있습니다.



Label과 Selector 이해하기

라벨과 셀렉터는 쿠버네티스에서 객체를 그룹화하고 필터링하는 핵심적인 개념입니다. 라벨은 각 객체에 부여되는 키-값 쌍의 속성이며, 셀렉터는 특정 라벨 조건에 따라 객체를 선택하는 데 사용됩니다. 이를 통해 수많은 객체 중에서 원하는 항목을 손쉽게 찾을 수 있습니다.

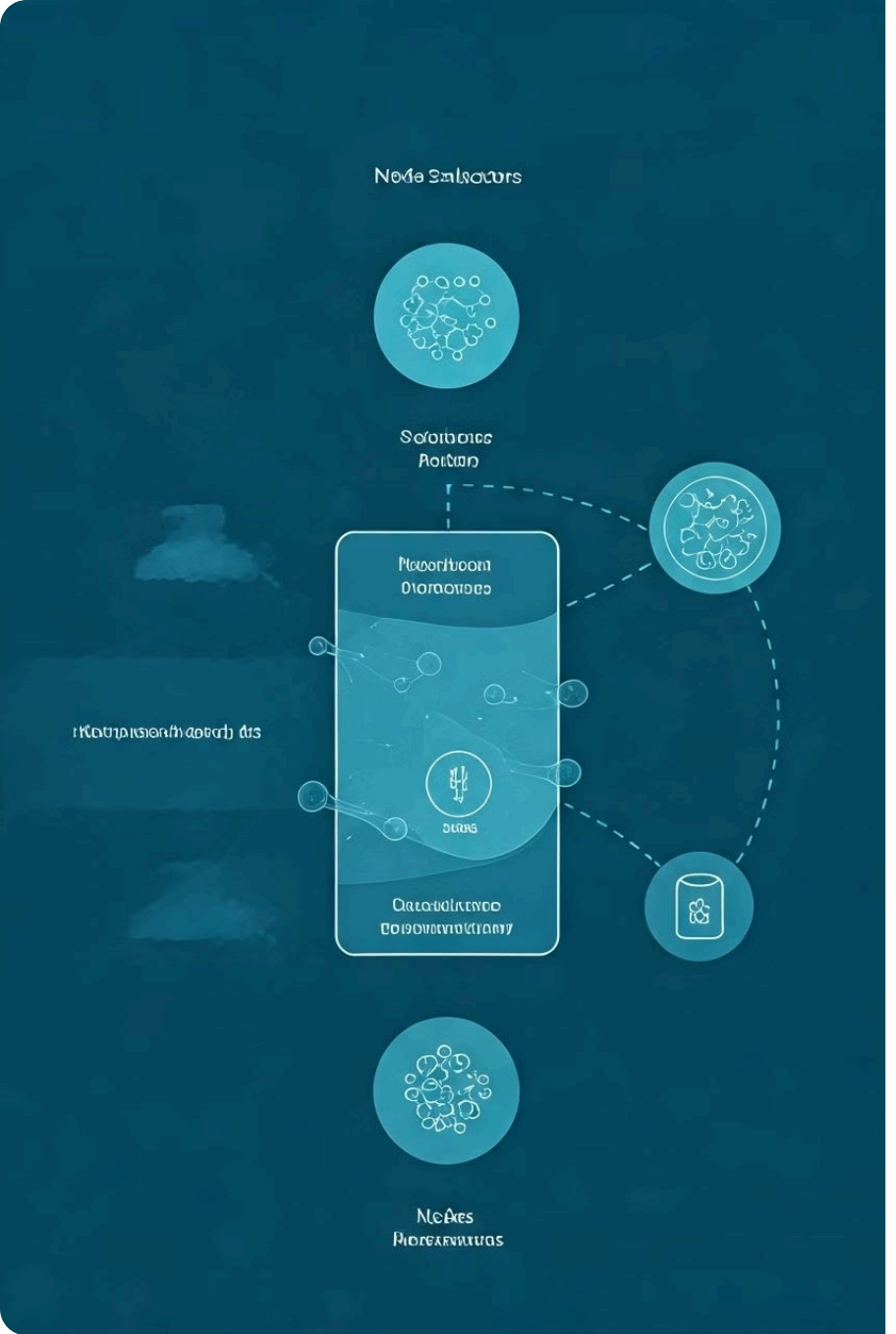
쿠버네티스에서는 포드, 서비스, 레플리카셋, 디플로이먼트 등 다양한 리소스를 관리하게 됩니다. 라벨을 통해 이들 객체를 가령 서로 다른 애플리케이션, 버전, 환경 등으로 구분할 수 있습니다. 셀렉터를 사용하여 이러한 라벨 기준으로 객체를 필터링하고 조회할 수 있습니다.



Taints와 Tolerations 이해하기

Taints와 Tolerations는 쿠버네티스에서 노드와 포드 간의 관계를 제어하는 중요한 개념입니다. 이를 이해하기 위해 벌레와 퇴치제를 사용한 비유를 활용해 볼 수 있습니다. 노드에 **taint(퇴치제)**를 설정하면 특정 포드가 이를 **toleration(내성)**하지 못해 해당 노드에 배치되지 않습니다. 반면 일부 포드는 이 taint를 견딜 수 있어 해당 노드에 배치될 수 있습니다.

쿠버네티스에서는 노드에 taint를 설정하고, 포드에 toleration을 설정하여 이를 구현합니다. 이를 통해 특정 노드를 특정 애플리케이션 전용으로 사용할 수 있습니다. 예를 들어, 노드 1에 `app=blue` taint를 설정하면 기본적으로 모든 포드가 이를 견딜 수 없어 해당 노드에 배치되지 않습니다. 하지만 포드 D에 `app=blue` toleration을 추가하면 노드 1에 배치될 수 있습니다.



노드 셀렉터와 노드 어피니티

1

노드 셀렉터

노드 셀렉터는 포드를 특정 노드에 배치하는 간단한 방법입니다. 예를 들어, 리소스가 풍부한 큰 노드에만 데이터 처리 워크로드를 배치할 수 있습니다.

2

노드 셀렉터의 한계

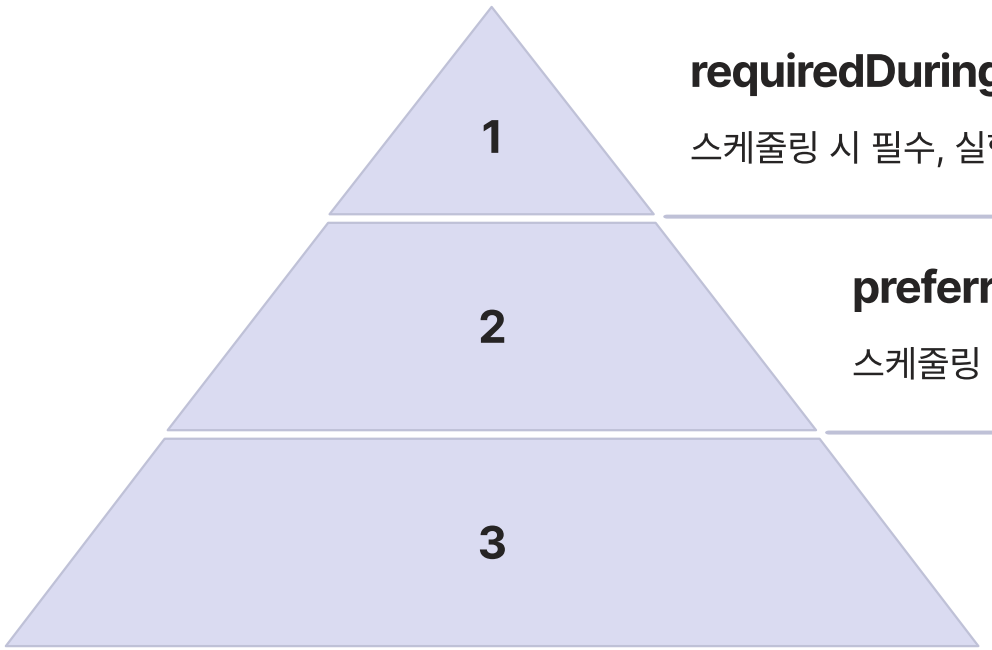
노드 셀렉터는 단일 라벨과 셀렉터만을 사용합니다. 요구사항이 더 복잡해지면 이 방법만으로는 부족할 수 있습니다. 예를 들어, 여러 가지 조건을 동시에 만족시키는 노드에 포드를 배치하고 싶을 때 이를 처리할 수 없습니다.

3

노드 어피니티와 안티-어피니티

이러한 한계를 극복하기 위해 노드 어피니티와 안티-어피니티 기능이 도입되었습니다. 이를 통해 더 복잡한 배치 규칙을 구현할 수 있습니다.

고급 노드 어피니티 개념



requiredDuringSchedulingIgnoredDuringExecution

스케줄링 시 필수, 실행 중에 무시

preferredDuringSchedulingIgnoredDuringExecution

스케줄링 시 선호, 실행 중에 무시

Future: requiredDuringExecution

실행 중에도 필수

노드 어피니티의 핵심 목적은 특정 포드를 원하는 노드에 배치하는 것입니다. 이를 위해 다양한 유형의 노드 어피니티를 제공합니다. requiredDuringSchedulingIgnoredDuringExecution은 스케줄링 시에만 규칙을 강제하고, preferredDuringSchedulingIgnoredDuringExecution는 규칙을 따르려고 하지만 실패해도 실행합니다. 향후에는 실행 중에도 규칙을 강제하는 requiredDuringExecution 유형이 추가될 예정입니다.

리소스 요청과 한계 in Kubernetes

1

리소스 요청

포드가 최소한으로 필요한 CPU와 메모리 양

2

리소스 한계

포드가 사용할 수 있는 최대 CPU와 메모리 양

3

리소스 단위

CPU: 1 vCPU, 메모리: Mi, Gi

쿠버네티스 클러스터의 각 노드는 특정 양의 CPU와 메모리 리소스를 가지고 있습니다. 포드는 실행을 위해 특정 리소스를 요구하며, 스케줄러는 이러한 리소스 요구사항을 고려하여 적절한 노드에 포드를 배치합니다. 리소스 요청은 포드가 최소한으로 필요한 리소스 양을, 리소스 한계는 포드가 사용할 수 있는 최대 리소스 양을 정의합니다. 리소스 단위로는 CPU, 메모리 등이 사용됩니다.

DaemonSet 이해하기

1

DaemonSet 개요

각 노드에 포드 배포

2

DaemonSet 사용 사례

모니터링, 네트워킹, kube-proxy

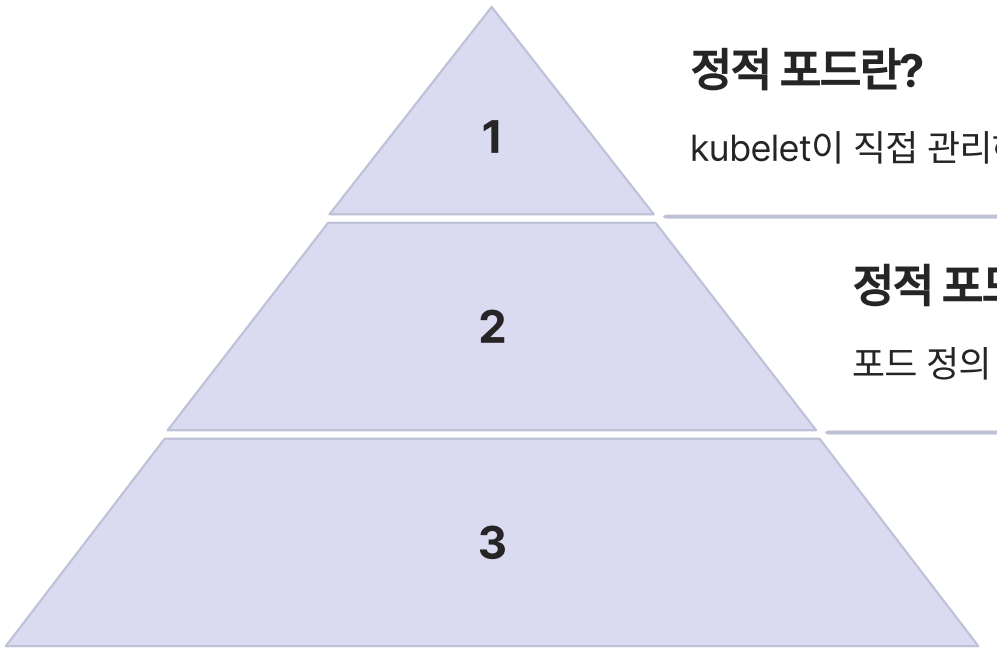
3

DaemonSet 작동 방식

노드 어피니티를 이용한 스케줄링

DaemonSet은 ReplicaSet과 유사하지만, 클러스터의 각 노드에 하나의 포드를 배포하는 데 중점을 둡니다. 새로운 노드가 클러스터에 추가되면 해당 노드에 자동으로 포드가 생성되고, 노드가 제거되면 포드도 자동으로 삭제됩니다. DaemonSet은 모니터링 에이전트, 로그 수집기, 네트워킹 솔루션 등을 클러스터 전체에 배포하는 데 유용합니다. 최신 버전의 쿠버네티스에서는 DaemonSet이 노드 어피니티 규칙을 사용하여 포드를 스케줄링합니다.

정적 포드(Static Pod)의 이해



정적 포드란?

kubelet이 직접 관리하는 포드

정적 포드 동작 방식

포드 정의 파일 배치 → kubelet이 주기적으로 관리

정적 포드 생성 방법

kubelet 설정을 통해 포드 정의 파일 경로 지정

정적 포드는 쿠버네티스 컨트롤 플레인 구성 요소(kube-apiserver, kube-scheduler 등)와 독립적으로 kubelet이 직접 관리하는 포드입니다. kubelet은 지정된 디렉토리에서 포드 정의 파일을 주기적으로 확인하여 포드를 생성하고 관리합니다. 이를 통해 컨트롤 플레인 장애 시에도 kubelet은 정적 포드를 계속해서 실행할 수 있습니다. 정적 포드는 주로 컨트롤 플레인 구성 요소 자체를 배포하는 데 사용됩니다.