



kubernetes

쿠버네티스 개요

Autumn 2024



AI융합학과

Seongbok Baik

sbbaik@dju.ac.kr

00 Contents



- 쿠버네티스 개념
- 핵심 구성 요소

01 쿠버네티스(Kubernetes)?

1. 컨테이너 클러스터 관리 서비스

- 컨테이너를 그룹화하고 배포, 확장, 운영을 자동화하여 관리
- 대량의 컨테이너를 관리함으로써 발생하는 복잡성을 효율적으로 처리하는 도구

2. 구글에서 오픈 소스로 공개

- 구글 내부 기술을 바탕으로 개발
- 2014년 오픈 소스로 공개, 글로벌 커뮤니티가 기여

3. 다양한 플랫폼 지원

- Google Cloud Platform(GCE), CoreOS, Microsoft Azure, VMware vSphere
- 클라우드와 온프레미스 환경 모두 지원

4. 근본적인 목적은 도커(Docker) 컨테이너의 효율적인 관리

- 기본 컨테이너 런타임으로 도커 사용
- 도커 외 다양한 컨테이너 런타임도 지원 가능

02 쿠버네티스(Kubernetes)의 어원과 역사

1. 쿠버네티스

- 컨테이너화된 애플리케이션의 자동 배포, 확장 및 관리 역할
- 오픈소스(Open Source) 플랫폼

2. 쿠버네티스 어원

- Kubernetes, 고대 그리스어, 배의 조타수 (Helmsman)를 의미함
- K8s로 표기하기도 함

3. 구글에서 오픈 소스로 공개

- 구글 내부 기술(Borg 클러스터 매니저)을 바탕으로 개발
- Borg : 수천 개의 서버에서 수백만개의 작업을 실행하고 관리하는 시스템
- 2014년 오픈 소스로 공개, 글로벌 커뮤니티 협업

03 쿠버네티스 주요 구성 요소 (High-Level Components)

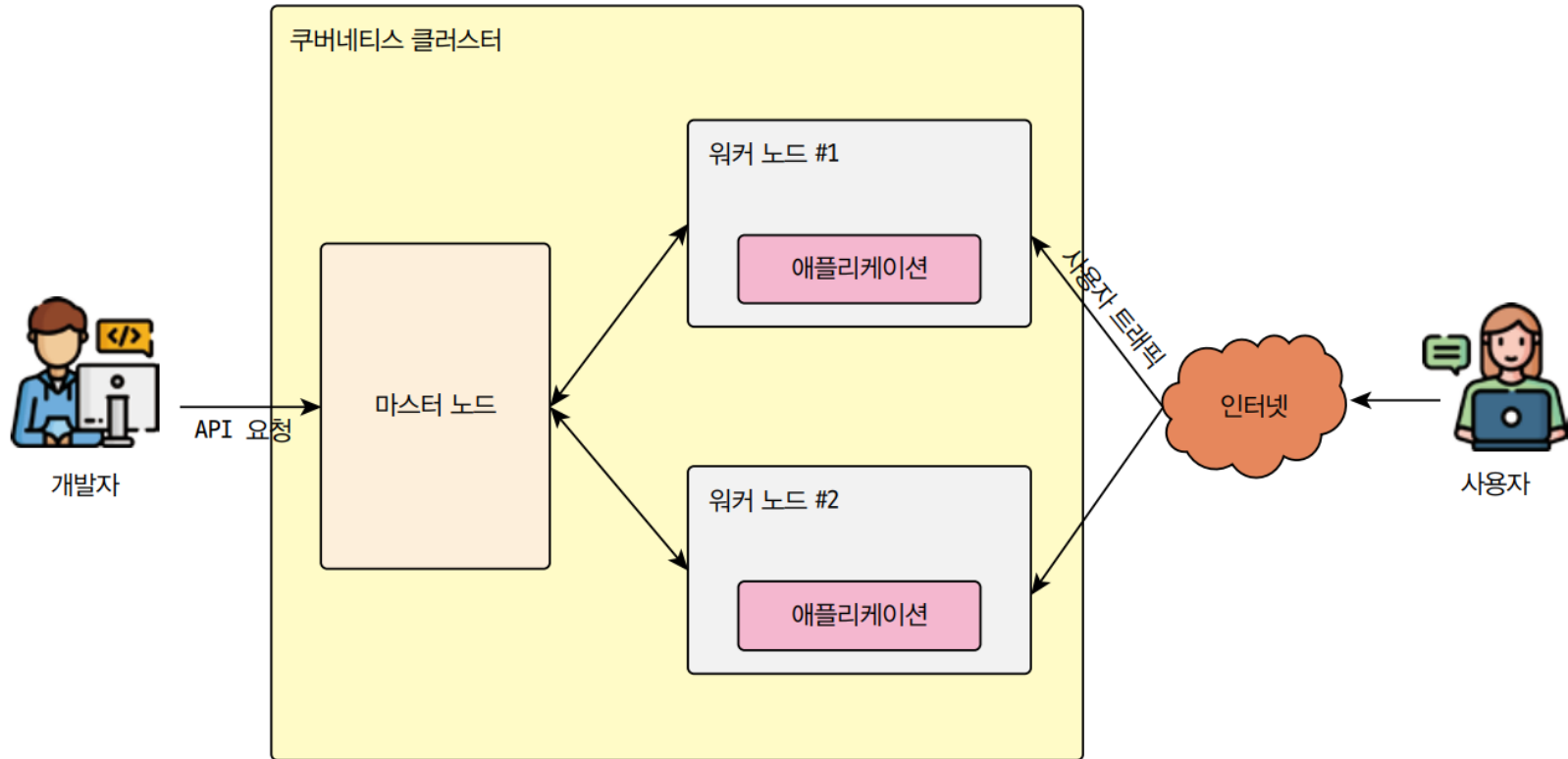
1. Master Node

- 중앙 제어 장치로, 클러스터의 전반적인 관리와 오케스트레이션을 담당
- 주요 역할:
 - 사용자 명령(kubecfg) 처리
 - 작업 스케줄링 및 분배
 - 상태 모니터링 및 클러스터 관리

2. Worker Node

- 애플리케이션 컨테이너를 실제로 실행
- Master로부터 명령을 받아 작업 수행
- 클러스터 내에서 분산 작업 처리

04 쿠버네티스 구성 요소 (High-Level Components)



CNI(Container Network Interfaces; 컨테이너 네트워크 인터페이스)

- 마스터 노드와 워커 노드 간 유기적인 통신을 위한 인터페이스
- CNI용 쿠버네티스 네트워크 플러그인: Flannel, Calico

05 쿠버네티스의 핵심 개념 (Key Concepts)

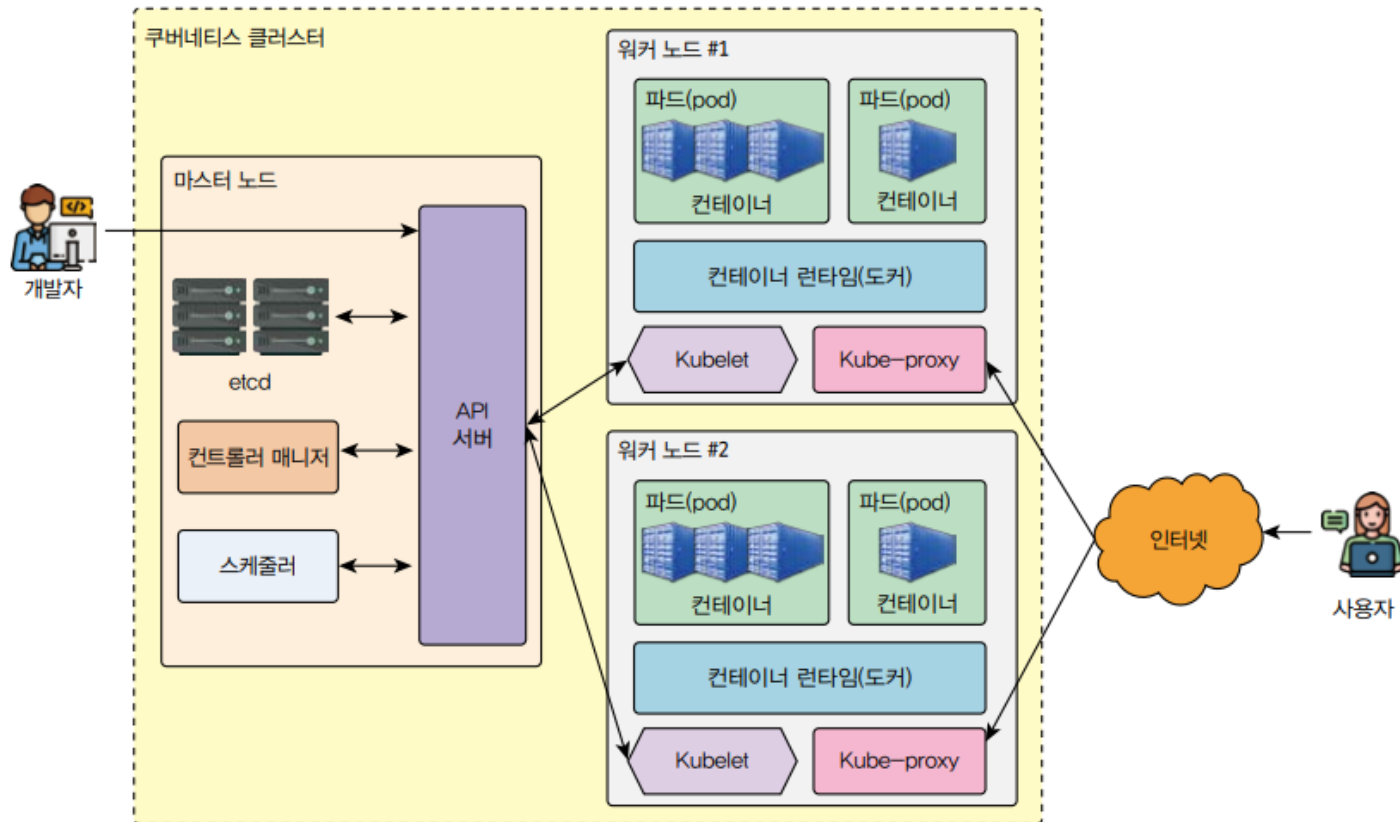
1. 주요 개념

- **Master Node:** 클러스터를 제어하고 관리하는 중앙 노드
- **Worker Node:** 애플리케이션 컨테이너를 실행하는 작업 노드
- **Pod:** 하나 이상의 컨테이너를 그룹화한 단위, 쿠버네티스의 배포 최소 단위
- **Service와 Labels:**
 - **Service:** Pod 간의 네트워크 접근을 추상화한 논리적 단위
 - **Labels:** 리소스를 그룹화하거나 식별하기 위한 키-값 쌍
- **Container:** 가상화된 애플리케이션 실행 환경
- **Kubernetes Node:**
 - **Kubelet:** Node의 상태를 관리하고 Master와 통신
 - **Kubernetes Proxy:** Pod 네트워킹과 로드 밸런싱 처리

2. 쿠버네티스 제어 패널 (Kubernetes Control Panel)

- **API Server:**
 - 클러스터와 상호작용하는 진입점, REST API를 통해 명령 처리
- **Controller Manager:**
 - 클러스터의 상태를 원하는 상태로 유지하는 컨트롤러 관리
- **Persistent Store:**
 - 클러스터 데이터의 영구 저장소, 일반적으로 etcd 사용

06 쿠버네티스 클러스터 구조



07 클러스터 동작 과정

•개발자:

- API 서버에 애플리케이션 배포 명령 전송

•마스터 노드:

- API 서버에서 요청을 받아 클러스터 관리
- 스케줄러가 작업을 적절한 워커 노드에 배치

•워커 노드:

- 할당받은 작업(pod)을 실행
- Pod 내부에서 컨테이너가 실행됨
- Kube-proxy를 통해 네트워크 트래픽을 라우팅

•사용자:

- 인터넷을 통해 워커 노드에서 실행 중인 애플리케이션에 접근

08 쿠버네티스(Kubernetes) 클러스터 구성요소

•마스터 노드 (Master Node):

- 클러스터의 중앙 관리 역할 수행
- Kubectl 명령어를 통해 마스터 노드의 kube-apiserver에게 API 요청을 보냄
- 주요 구성 요소:
 - **API 서버**: 클라이언트와의 통신을 관리하고 클러스터의 진입점 역할 (쿠버네티스 컨트롤 플레인에서의 프론트엔드 역할)
 - **etcd**: 분산 키-값 저장소로 클러스터 상태 데이터 저장 (쿠버네티스 클러스터에 존재하는 모든 데이터를 key-value 쌍으로 저장)
 - **컨트롤러 매니저**: 쿠버네티스의 자원 관리/제어, 클러스터 상태를 유지 및 조정 (Deployment 컨트롤러, Service 컨트롤러, ReplicaSet 컨트롤러, 등)
 - **스케줄러**: 작업(pod)을 적절한 워커 노드에 할당

09 쿠버네티스(Kubernetes) 클러스터 구성요소

•워커 노드 (Worker Node):

- 실제 애플리케이션 워크로드가 실행되는 노드
- 주요 구성 요소:
 - **Kubelet**: 마스터 노드와 통신하며, 컨테이너 생성/관리
 - **컨테이너 런타임**: 컨테이너 실행 환경 (예: Docker)
 - **Kube-proxy**: 네트워킹과 로드 밸런싱을 관리

10 노드 구성 요소-1

1. Kubelet

- 각 워커 노드에서 실행되는 필수 컴포넌트
- 주요 역할:
 - 마스터 노드(API 서버)와 통신
 - Pod의 상태 모니터링 및 관리
 - 컨테이너 생성, 시작, 종료 등의 작업 지시
 - Pod 명세 파일(PodSpec)을 기반으로 컨테이너 실행

2. Kube-proxy

- 클러스터 내부 네트워크 구성 및 관리
- 주요 역할:
 - 네트워크 트래픽의 라우팅 및 로드 밸런싱
 - Pod 간 통신을 위한 가상 IP 관리
 - 외부 요청을 적절한 Pod로 전달 (서비스 로드 밸런싱)

11 노드 구성 요소-2

3. 컨테이너 런타임

- 컨테이너 실행 환경을 제공
- 주요 역할:
 - 컨테이너 실행 및 관리를 담당
 - 지원되는 런타임 예시:
 - Docker / containerd / CRI-O
 - Kubelet과 연동하여 Pod 내 컨테이너 생성 및 관리

4. Pod (파드)

- 쿠버네티스의 최소 배포 단위
- 주요 특징:
 - 컨테이너들 간 자원 공유 (네트워크, 스토리지)
 - Pod 내부 컨테이너 간에는 **로컬 네트워크**로 통신
 - 짧은 수명: 필요한 경우 생성 및 삭제가 반복됨

12 쿠버네티스 특징

- **확장성:** 새로운 워커 노드를 추가하여 클러스터 확장 가능
- **복원성:** 마스터 노드가 클러스터 상태를 지속적으로 점검 및 복구
- **유연성:** 다양한 컨테이너 런타임 지원

[마스터 노드의 특징]

- 클러스터의 상태를 지속적으로 점검하고 유지
- 모든 노드와 Pod의 동작을 제어
- 개발자와 클러스터 간의 인터페이스 역할 수행 (API 서버를 통해)

[워커 노드의 특징]

- 모든 구성 요소가 조화를 이루어 애플리케이션 워크로드
- 사용자 요청을 처리하고, Pod에 트래픽 전달
- 마스터 노드의 명령에 따라 리소스 자동 관리

13 워크로드(Workload)-1

- 쿠버네티스에서 실행되는 애플리케이션
- 파드 내부에서 하나의 컴포넌트 또는 다수의 컴포넌트로 실행
- 파드: 실행 중인 컨테이너 집합

[레플리카셋]

- 파드의 복제를 관리
- 클라이언트가 요구하는 개수 만큼 파드 복제, 모니터링, 관리

[디플로이먼트]

- 애플리케이션의 배포와 스케이링 관리

[스테이트풀셋(StatefulSet)]

- 파드 간 순서와 고유성이 보장되어야 하는 경우 사용

14 워크로드(Workload)-2

[데몬셋(DaemonSet)]

- 쿠버네티스를 구성하는 모든 노드가 파드의 복사본을 실행하도록 관리
- 쿠버네티스 클러스터에 새로운 노드가 추가되면 파드 역시 추가됨
- 주로 로깅, 모니터링, 스토리지 등 시스템 수준의 서비스 실행

[잡과 크론잡(Job and Cronjob)]

- 작업(Task)이 정상적으로 완료되고 종료되는 것을 담당
- 파드가 정상 종료되지 않은 경우 재실행
- Job: 한 번 실행하고 종료되는 작업을 담당
- Cronjob: 스케줄에 따라 동일한 작업을 여러 번 수행하는 경우 사용 (리눅스의 크론 탭 (Crontab)과 유사함)

15 쿠버네티스 네트워크-1

[기본 원칙]

- Pod 간 통신 가능: 모든 Pod는 NAT 없이 다른 모든 Pod와 통신할 수 있음.
- 노드와 Pod 간 통신 가능: 모든 노드는 NAT 없이 모든 Pod와 통신 가능하며, 반대로 동일.
- 일관된 IP 주소: Pod가 스스로 인식하는 IP는 다른 Pod나 노드에서 보는 IP와 동일.

[CNI(Container Network Interface)]

- 정의:
 - 쿠버네티스의 네트워크 설정을 담당하는 플러그인 인터페이스.
 - Pod와 노드 간 통신, IP 할당 및 라우팅 관리.
- 설치 위치:
 - 각 노드(Node)에 설치되어 Pod의 네트워크 환경을 설정.
 - Pod 내부에는 설치되지 않음

16 쿠버네티스 네트워크-2

[Calico]

- Calico는 가장 널리 사용되는 CNI 플러그인 중 하나:
 - 고성능 네트워크 구현 (네이티브 L3 라우팅 지원).
 - 네트워크 정책(Network Policy) 설정 및 관리.
 - Pod에 고유한 IP 주소를 부여하고, 노드와 Pod 간 통신을 설정.
- 주요 기능:
 - BGP(Border Gateway Protocol) 기반 네트워크 라우팅.
 - 네트워크 정책을 통해 Pod 간 트래픽 제어 가능.
 - 대규모 클러스터에서도 효율적인 네트워크 관리.
- Calico의 역할
 - Pod 간 통신: Calico는 Pod 간 통신을 NAT 없이 수행하도록 네트워크를 설정.
 - 노드-Pod 통신: 각 노드에 Pod 네트워크 서브넷을 연결하여 NAT 없이 데이터 전달.
 - IP 관리: Pod와 노드 간 IP 주소 일관성을 보장

17 쿠버네티스 네트워크-3

[서비스]

- 파드를 여러 개 묶어 하나의 네트워크 엔드포인트로 관리
- 클러스터 외부로 애플리케이션 노출
- 주요 특징:
 - 파드 수정 불필요: 이미 실행 중인 파드를 외부에 노출할 때 변경 작업 필요 없음
 - 클라이언트와의 통신을 쉽게 관리
- 로드 밸런싱 지원: 여러 파드로 트래픽 분배

[인그레스(Ingress)]

- HTTP/HTTPS 요청을 쿠버네티스 클러스터 내부로 라우팅
- 주요 특징:
 - 서비스로 트래픽 전달: 클러스터 외부에서 내부 서비스를 접근 가능하게 함
 - URL 기반 라우팅, HTTPS 인증서 관리 기능 제공
- 클러스터 외부 트래픽을 관리하는 진입점 역할

18 쿠버네티스 네트워크-4

[활용]

- 이 기본 규칙을 기반으로 **서비스, 인그레스, 그리고 로드 밸런싱** 설계 가능.
- 클러스터 내 네트워크 디버깅 및 트래픽 관리의 핵심 원칙.
- 쿠버네티스의 네트워킹은 **단순하고 일관된 통신 모델** 제공
- 외부 네트워크 설정 필요 없이 클러스터 내에서 자동 연결
- Pod와 노드 간의 투명한 통신을 통해 클러스터 내 서비스 간 효율적인 데이터 교환 가능

19 쿠버네티스 스토리지-1

[컨테이너 내부의 파일 수명]

- 컨테이너 내 파일은 수명이 짧음.
- 주요 이유:
 - 컨테이너가 삭제되거나 재시작되면 내부 파일이 모두 삭제됨.
 - 실행 중인 컨테이너의 파일이 영구적으로 보존되지 않음.

[쿠버네티스 스토리지의 필요성]

- 파일 영구 저장:
 - Pod나 컨테이너의 상태와 무관하게 파일 보관 가능.
 - 데이터를 지속적으로 저장하거나 복구가 필요한 애플리케이션에서 사용.
- 스토리지 유형:
 - Persistent Volume (PV): 클러스터와 독립적인 스토리지
 - Persistent Volume Claim (PVC): Pod가 스토리지 요청 시 사용하는 추상화 계층

20 쿠버네티스 스토리지-2

[활용 예시]

- 데이터베이스 애플리케이션: 재시작 시에도 데이터 유지
- 로그 파일 저장: 분석 및 문제 해결을 위해 로그 데이터 영구 보존

21 정리-1

[쿠버네티스의 역할]

- 컨테이너화된 애플리케이션의 배포, 확장, 관리를 자동화.
- 대규모 클러스터 관리의 복잡성을 해결하여 효율적인 운영 가능.
- 다양한 플랫폼(클라우드 및 온프레미스) 지원

[주요 구성 요소]

- **마스터 노드:**
 - 클러스터의 중앙 관리, 스케줄링 및 상태 유지.
 - 주요 컴포넌트: API 서버, etcd, 스케줄러, 컨트롤러 매니저.
- **워커 노드:**
 - 애플리케이션 워크로드 실행.
 - 주요 컴포넌트: Kubelet, Kube-proxy, 컨테이너 런타임.
- **Pod:** 컨테이너 그룹화 단위로, 최소 배포 단위

22 정리-2

[네트워크와 스토리지]

- **네트워크:**
 - CNI (예: Calico)를 통해 Pod와 노드 간 통신 설정.
 - 서비스: Pod 간 네트워크 추상화 및 로드 밸런싱 지원.
 - 인그레스(Ingress): 클러스터 외부와 HTTP/HTTPS 트래픽 연결
- **스토리지:**
 - Pod와 독립적인 Persistent Volume(PV) 제공.
 - 데이터 영구 저장 및 복구 지원

[쿠버네티스의 장점]

- 확장성: 새로운 노드를 추가하여 클러스터 확장.
- 유연성: 다양한 컨테이너 런타임 및 플랫폼과 호환

서로에게, 자신에게 친절합니다 - 허준이

