


# KU7s 1<sup>st</sup> Session

GDGoC [25-26] 황성조

# Table of Contents



- 쿠버네티스란?
- 쿠버네티스 컴포넌트
- 쿠버네티스 API

# 쿠버네티스란?

컨테이너화된 애플리케이션을 자동으로 배포·관리·확장해주는 오픈소스 시스템

→ 여러 서버(Node)에 흩어진 컨테이너들을 하나의 큰 시스템처럼 관리해주는 시스템

## 쿠버네티스 역할

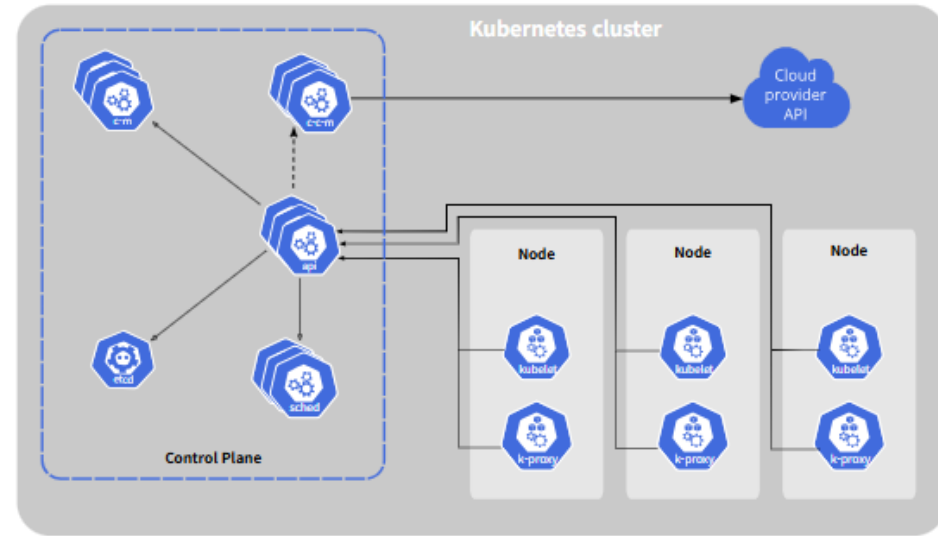
- 컨테이너 자동 배포 → `kubectl apply`
- Pod이 죽으면 자동 복구
- Node 자원에 맞게 배치 (스케줄링)
- 장애 발생 시 자동 복구
- 트래픽 급증 시 자동 확장
- 환경 설정, 비밀번호 관리
- 버전 업그레이드 중에도 서비스 중단 없이 교체 가능

→ 개발자는 코드만 작성하면 쿠버네티스가 자동으로 운영함

# 주요 개념

개념	설명
Pod	쿠버네티스의 최소 실행 단위 컨테이너 + 실행환경 정보 (ex. 네트워크(IP), 스토리지)
Node	Pod이 실행되는 실제 서버
Cluster	여러 Node를 묶은 전체 쿠버네티스 환경
Control Plane	클러스터의 제어-관리 담당
Deployment	Pod의 배포/복구/스케일링을 자동화하는 상위 리소스
Service	Pod 집합에 대한 네트워크 접근 제어 및 로드밸런싱 제공

# 컴포넌트



**Control Plane:** 클러스터 전체 관리-제어 (두뇌 역할)

API Server: 모든 동작의 시작점

etcd: 클러스터의 DB

controller manager: 클러스터 상태 감시

Scheduler: Pod를 어떤 Node에 넣을지 배치

Cloud controller manager: 클라우드 환경에서  
인프라와 연동

**Node:** 실제 컨테이너가 실행되는 물리적/가상 머신

kubelet: Node에서 Pod의 실행 상태 관리

kube-proxy: 네트워크 관리 (Pod/외부 간 통신)

Container Runtime: 컨테이너 실행 프로그램

## 흐름

사용자 → API Server → Scheduler → Node → Pod

1. 사용자 명령(kubectl → API Server)
2. API Server → etcd에 State 저장
3. k-cm가 변화 감시(Desired State != Current State이면, 자동으로 Pod 요청)
4. Scheduler가 Pod를 배치할 Node 결정
5. Node의 kubelet이 Pod 생성
6. kube-proxy가 네트워크 설정 (Pod 통신)
7. Add-ons이 보조

# API

쿠버네티스의 모든 동작은 API를 통해 제어된다.

쿠버네티스의 모든 Entity → API 리소스로 표현

`/api/v1/pods` → Pod 목록

`/apis/apps/v1/deployments` → Deployment 목록

API 그룹 이름	대표 리소스	설명/기능
core (v1)	Pod, Service(네트워크), ConfigMap(설정)	쿠버네티스 기본 리소스
apps	Deployment, DaemonSet, Statefulset	애플리케이션 실행 관리용 리소스 Pod를 묶고 자동 복구 관리
batch	Job, CronJob	일회성-주기적 작업 수행용 리소스 ex. 스케줄러 기반 작업
rbac.authorization.k8s.io	Role, RoleBinding, ClusterRole	리소스 사용량에 따라 Pod 수 조정 CPU/메모리 기반 스케일링 지원
autoscaling	HorizontalPodAutoscaler	권한 관리

## API 버전 관리

Alpha: 실험 단계   beta: 안전화 중   stable: 안정화 완료

## Open API

Open API 스펙(JSON 형식)으로 제공

→ 이를 이용해 프로그래밍 가능

ex. “REST API 서버”로 접근 가능

```
bash
```

```
GET /api/v1/namespaces/default/pods
```

## 전체 구조 정리

[사용자] (kubectl / Dashboard / Controller)

|

▼

[API Server]

|

├ [etcd] (상태 저장)

├ [Controller Manager] (감시 및 복구)

└ [Scheduler] (배치 결정)

API Server는 클러스터의 “모든 요청 통로”이자  
Kubernetes의 “RESTful 중심 API 시스템”의 핵심

# ADD Component

컴포넌트	설명
kubectl (클라이언트)	사용자가 명령을 내리는 도구 (kubectl apply, get pods 등)
Add-ons (추가 기능)	DNS, Dashboard, Monitoring, Ingress Controller 등 클러스터 기능 확장용

## Add-ons

기능 영역	주요 Add-on	설명
Network	CoreDNS	클러스터 내부에서 Pod 이름을 IP로 변환
대시보드(UI)	Kubernetes Dashboard	웹 기반 관리 UI, kubectl 없이 Pod/Deployment 등을 시각적으로 관리 가능
Monitoring	Metrics Server	CPU-메모리 사용량 수집 및 시각화
Logging	EFK Stack (Elasticsearch + Fluentd + Kibana)	클러스터 전체의 로그를 중앙 수집/검색/시각화
Network 정책	Calico, Flannel	Pod 간 네트워크 규칙 관리 및 보안 정책 적용
Ingress Controller	Nginx, Traefik	외부 트래픽을 클러스터 내부 서비스로 라우팅
Storage 관리	CSI 플러그인	외부 스토리지(AWS EBS, NFS 등) 연결 자동화

# 정리

## 쿠버네티스란?

컨테이너 운영을 자동화하고, 확장성과 안전성을 보장하는 오픈소스 시스템

## 주요 특징

- 자동화: 컨테이너 관련 과정 자동 처리
- 자가 복구: Pod 자동 복구
- 자동 확장: Pod 개수 자동 조절
- 이식성: 어디서든 동일한 환경 제공
- 선언형 관리: 사용자가 원하는 상태만

정의, 나머지는 쿠버네티스가 관리

## 전체 구조

Control Plane: 클러스터 전체 제어 (두뇌 역할)

Node: 실제 컨테이너 실행 (손발 역할)

API Server: 모든 동작의 중심 == 명령 시작점

Add-ons: 확장 기능 제공