# 7 Kubernetes Custom 관리 방법

## 01 Kubernetes Custom 관리 방법 소개

# Kubernetes Custom 관리 방법



1. Kubernetes Custom 관리 방법 소개
2. Custom Resource Definition(CRD) 소개
3. Kubernetes Operator 소개
4. [실습] Kubernetes Operator 적용

# 소개 내용

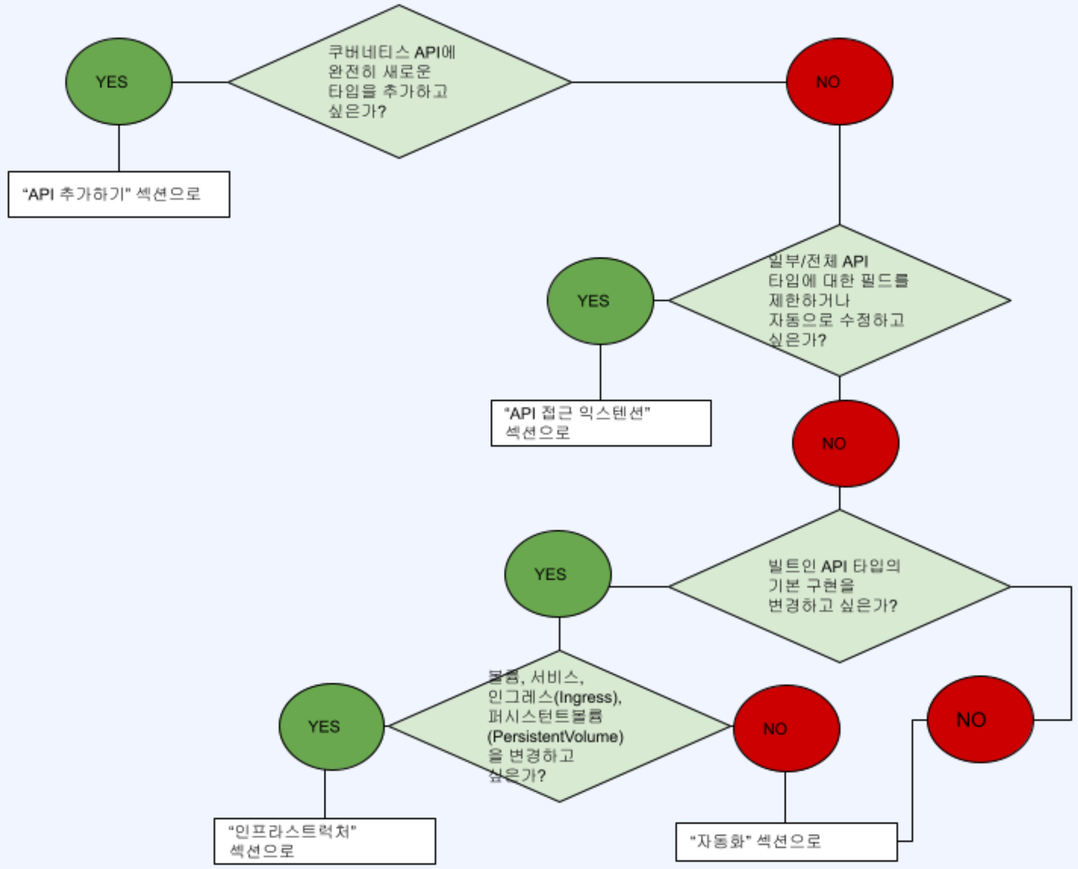


## 순서

1. Kubernetes Custom 관리 방법 소개
2. Extension 패턴
3. Extension 포인트
4. API Extension
5. 플랫폼 Extension

# 1. Kubernetes Custom 관리 방법 소개

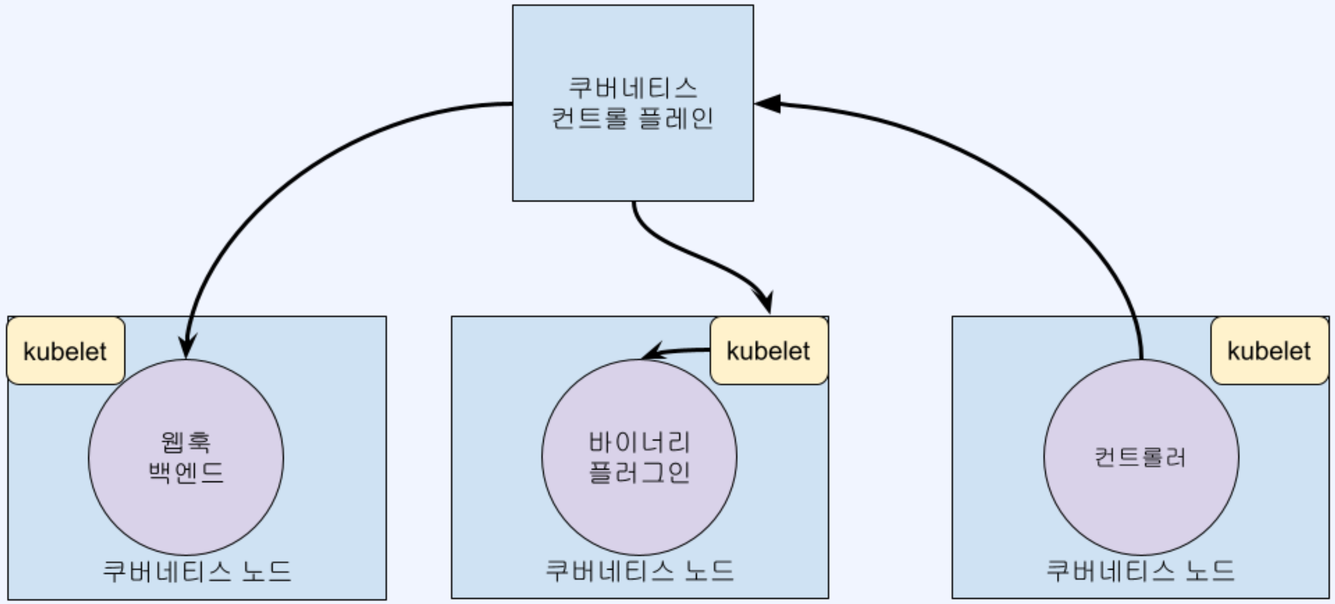
- Kubernetes Custom 관리 방법으로는 Extension을 통해 구현
- Extension은 Kubernetes를 확장하고 Kubernetes와 긴밀하게 통합되는 소프트웨어 컴포넌트 구현
- Kubernetes가 새로운 type의 리소스와 새로운 종류의 시스템을 지원할 수 있게 해줌




출처 : https://kubernetes.io/ko/docs/concepts/extend-kubernetes/

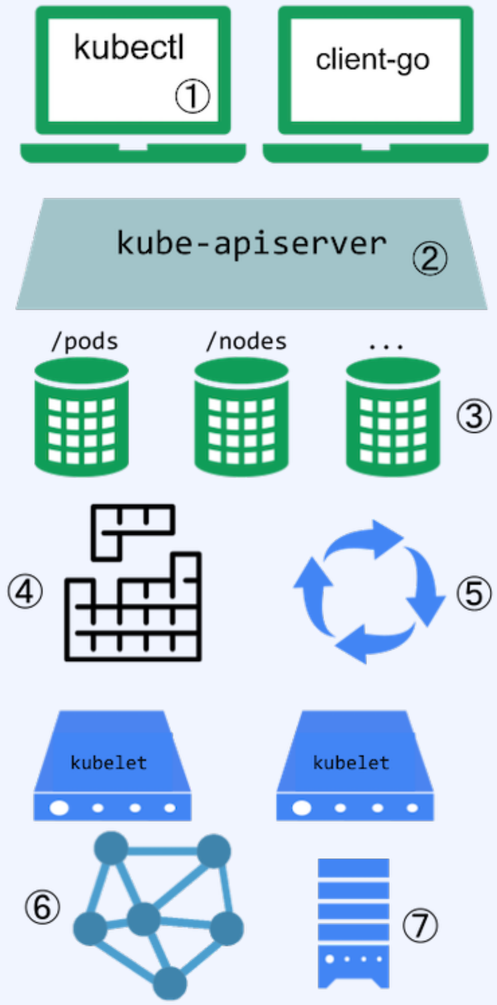
# 2. Extension 패턴

- Kubernetes는 **클라이언트** 프로그램을 작성하여 **자동화** 되도록 설계됨
- **Controller** 모델은 Kubernetes의 클라이언트로, 일반적으로 Object의 **.spec**을 읽고, 클러스터에서 적용한 다음 Object의 **.status**를 업데이트해 etcd에 갱신
- **WebHook** 모델에서는 Kubernetes가 클라이언트가 되서 특정 **원격 서비스**에 네트워크상으로 요청
- **바이너리** 플러그인 모델에서는 Kubernetes가 바이너리(프로그램)를 **실행**




출처 : https://kubernetes.io/ko/docs/concepts/extend-kubernetes/

# 3. Extension 포인트




1.사용자는 종종 kubectl을 사용하여 Kubernetes API와 상호 작용

2.apiserver는 여러 유형의 Extension 포인트는 요청을 인증하거나, 콘텐츠를 기반으로 요청을 차단하거나, 콘텐츠를 편집하고, 삭제 처리를 허용

3.apiserver는 다양한 종류의 리소스를 제공한다. 직접 정의한 리소스를 추가 및 Custom Resource라고 부르는 다른 프로젝트에서 정의한 리소스를 추가

4.Kubernetes 스케줄러는 파드를 배치할 노드를 결정

5.Kubernetes의 많은 동작은 API-Server의 클라이언트인 Controller라는 프로그램으로 구현됨. Controller는 종종 Custom Resource와 함께 사용

6.kubelet은 서버에서 실행되며, Pod가 클러스터 네트워크에서 자체 IP를 가진 가상 서버처럼 적용

7.kubelet은 컨테이너의 볼륨을 마운트 및 마운트를 해제

출처 : https://kubernetes.io/ko/docs/concepts/extend-kubernetes/

# 4. API Extension #1

## (1) 사용자 정의 유형

새 컨트롤러, 애플리케이션 구성 오브젝트 또는 기타 선언적 **API를 정의**하고 kubectl과 같은 쿠버네티스 도구를 사용하여 관리하려면 쿠버네티스에 **커스텀 리소스**를 추가

## (2) 새로운 API와 자동화의 결합

사용자 정의 리소스 API와 컨트롤 루프의 조합을 **오퍼레이터**(operator) 패턴이라고 함. 오퍼레이터 패턴은 특정 애플리케이션, 일반적으로 스테이트풀(stateful) 애플리케이션을 관리하는 데 사용됨. 이러한 **사용자 정의 API** 및 **컨트롤 루프**를 사용하여 스토리지나 정책과 같은 다른 리소스를 **제어**

# 4. API Extension #2



## (3) 빌트인 리소스 변경

기존 API 그룹을 **바꾸거나 변경할 수 없음**. API를 추가해도 기존 API(예: 파드)의 동작에 직접 **영향을 미치지는 않지만** API 접근 익스텐션은 영향을 줌

## (4) API 접근 익스텐션

요청이 쿠버네티스 API 서버에 도달하면 먼저 인증이 되고, 그런 다음 승인된 후 다양한 유형의 **어드미션 컨트롤**이 적용. 이러한 각 단계는 **익스텐션 포인트**를 제공.
쿠버네티스에는 이를 지원하는 몇 가지 빌트인 인증 방법이 있는데, 또한 인증 **프록시** 뒤에 있을 수 있으며 인증 **헤더**에서 원격 서비스로 토큰을 전송하여 확인할 수 있음(웹훅)

# 4. API Extension #3

## (5) 인증

인증은 모든 요청의 **헤더 또는 인증서**를 요청하는 클라이언트의 사용자 이름에 매핑.
쿠버네티스는 몇 가지 빌트인 인증 방법과 필요에 맞지 않는 경우 인증 **웹훅** 방법을 제공

## (6) 인가

인가는 특정 사용자가 **API 리소스**에서 읽고, 쓰고, 다른 작업을 수행할 수 있는지를 결정.
전체 리소스 레벨에서 작동하며 임의의 오브젝트 필드를 기준으로 구별하지 않는다.
빌트인 인증 옵션이 사용자의 요구를 충족시키지 못하면 인가 **웹훅**을 통해 사용자가
제공한 코드를 호출하여 인증 결정을 내림

# 4. API Extension #4



## (7) 동적 어드미션 컨트롤

요청이 승인된 후, 쓰기 작업인 경우 **어드미션 컨트롤** 단계도 수행됨. 빌트인 단계 외에도 몇 가지 익스텐션이 있음

- **이미지 정책** 웹훅은 컨테이너에서 실행할 수 있는 이미지를 제한
- 임의의 어드미션 컨트롤 결정을 내리기 위해 일반적인 **어드미션** 웹훅을 사용할 수 있음.

  어드미션 웹훅은 생성 또는 업데이트를 거부할 수 있음

# 5. 플랫폼 Extension #1

## (1) 스토리지 플러그인

**Flex Volumes**을 사용하면 Kubelet이 바이너리 플러그인을 호출하여 볼륨을 마운트하도록 함으로써 빌트인 지원 없이 볼륨 유형을 마운트 할 수 있음.
그러나 FlexVolume은 쿠버네티스 **v1.23**부터 사용 중단(deprecated)됨.
**CSI 드라이버**가 쿠버네티스에서 볼륨 드라이버를 작성할 때 추천하는 방식

## (2) 장치 플러그인

장치 플러그인은 **노드**가 장치 플러그인을 통해 새로운 **노드 리소스**(CPU 및 메모리와 같은 빌트인 자원 외에)를 발견할 수 있게 해줌

# 5. 플랫폼 Extension #2



## (3) 네트워크 플러그인

노드-레벨의 네트워크 플러그인을 통해 다양한 **네트워킹 패브릭**을 지원할 수 있음

## (4) 스케줄러 익스텐션

**스케줄러**는 파드를 감시하고 파드를 노드에 할당하는 특수한 유형의 **컨트롤러**임.
다른 쿠버네티스 컴포넌트를 계속 사용하면서 기본 스케줄러를 완전히
교체하거나, 여러 스케줄러를 동시에 실행할 수 있음.
스케줄러는 또한 **웹훅** 백엔드(스케줄러 익스텐션)가 파드에 대해 선택된 노드를
필터링하고 우선 순위를 지정할 수 있도록 하는 웹훅을 지원