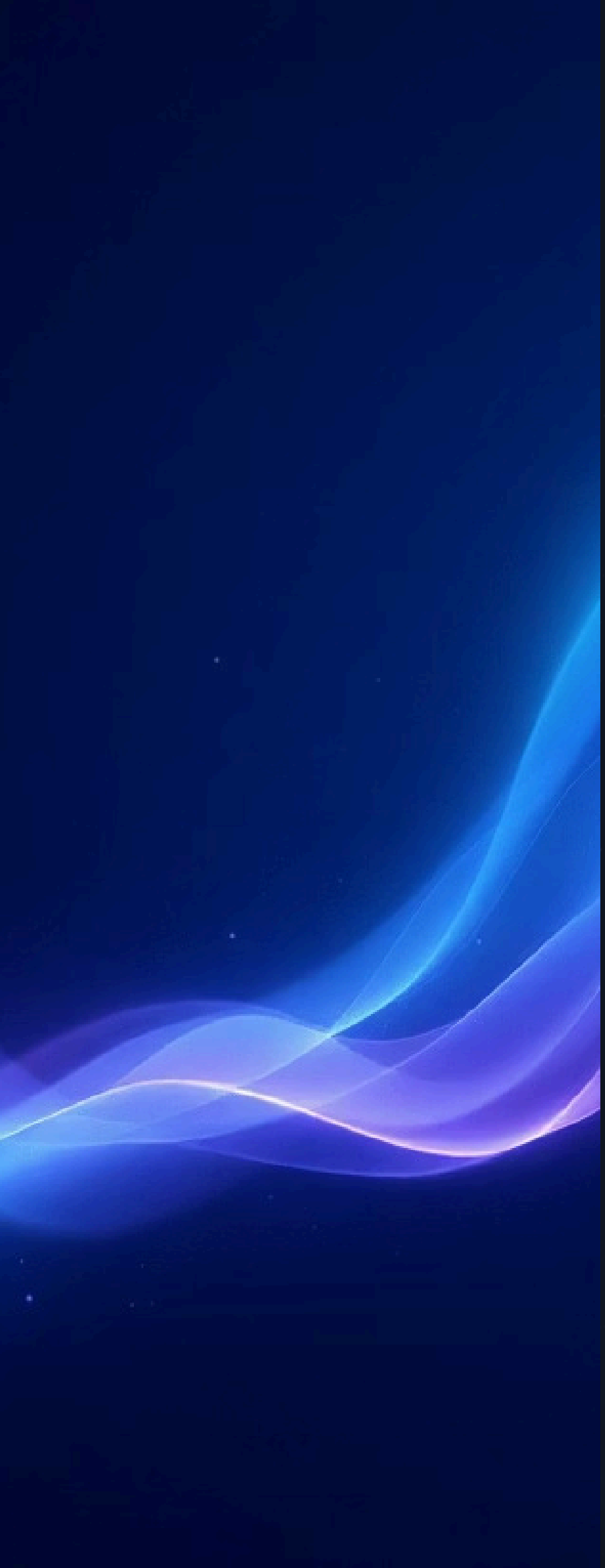

Kubernetes'te Garbage Collection (GC) Nedir?

En basit tanım

Garbage Collection (GC), Kubernetes kümelerindeki gereksiz veya artık istenmeyen objelerin **otomatik olarak temizlenmesi** işlemidir. Bu objeler genellikle **sahipsiz** kalmış, **kullanım dışı** olmuş veya kümenin **hedef durumuna uymayan** öğelerdir.

Kubernetes, her zaman tanımlanmış **Desired State (İstenen Durum)** üzerinde çalışır. Bir obje bu istenen durumda yer almıyorsa veya artık ona ihtiyaç duyulmuyorsa, sistem tarafından gereksiz ("çöp") olarak kabul edilir ve GC tarafından hedef alınır.

Kubernetes GC, bu temizleme sürecinde başlıca iki mekanizma kullanır: **OwnerReference'lar** ve **Finalizer'lar**.

- **OwnerReference'lar:** Bir Kubernetes objesi (örneğin bir Pod) oluşturulduğunda, genellikle onu yöneten başka bir objeye (örneğin bir ReplicaSet) bir **OwnerReference** ile bağlanır. Bu, "sahip-alt obje" ilişkisi kurar. Sahip obje silindiğinde, ona bağlı tüm alt objeler de otomatik olarak (kaskat silme yoluyla) kaldırılır. Böylece, bir Deployment silindiğinde, ilişkili ReplicaSet'leri ve Pod'ları da beraberinde temizlenir, sistemde artık gereksiz kaynak kalmaz.
- **Finalizer'lar:** Bir objenin tamamen silinmeden önce, dış kaynakların temizlenmesi gibi belirli ön koşulların yerine getirilmesini sağlayan özel etiketlerdir. Bu etiketler, güvenli ve eksiksiz bir silme işlemi için ek kontrol sağlar.

Bu otomatik temizleme mekanizması, Kubernetes'teki birçok obje türü için geçerlidir; yalnızca Pod'lar ve ReplicaSet'lerle sınırlı değildir. Deployment'lar, StatefulSet'ler, DaemonSet'ler, Job'lar, Service'lar, ConfigMap'ler, Secret'lar ve hatta belirli durumlarda PersistentVolume'lar dahi GC kapsamına girer. Özellikle tamamlanmış Job'ların artıkları veya silinmiş Deployment'ların ardında kalan ReplicaSet'ler gibi objeler, GC'nin temel hedefleridir.

Garbage Collection'ın verimli çalışması, Kubernetes kümesinin sağlığı için kritik öneme sahiptir. Gereksiz veya istenmeyen objelerin temizlenmemesi, zamanla kaynak sızıntılarına, performans düşüşlerine ve kümede beklenmedik davranışlara yol açabilir. GC, kümenin her zaman temiz, verimli ve belirlenen **Desired State**'e uygun kalmasını sağlayarak operasyonel yükü azaltır ve stabil bir çalışma ortamı sunar.

🧠 Kubernetes'te GC NEDEN GEREKLİ?

Kubernetes'te sürekli:

- Pod'lar ölür
- ReplicaSet'ler değişir
- Job'lar biter
- Node'lar gider gelir

Ama **bunlar otomatik silinmezse**:

- API server şişer
- etcd büyür
- Cluster yavaşlar

GC burada devreye girer.

🧩 GC HANGİ KATMANLARDA VAR?

Kubernetes'te **tek bir GC yok**, **3 ana GC mekanizması** var:

1️⃣ OWNER REFERENCES (En Önemlisi)

Mantık

Bir obje, başka bir objeye **aitse**,
sahibi silinince **çocuklar da silinir**.

Örnek

```
Deployment
  └──── ReplicaSet
     └──── Pod
```

Deployment silinirse:

- ReplicaSet gider
- Pod'lar gider

Teknik olarak

Her Kubernetes objesinde şu alan vardır:

```
metadata:
  ownerReferences:
  - apiVersion: apps/v1
    kind: ReplicaSet
    name: web-xyz
```

```
kubectl get pod <pod> -o yaml | grep ownerReferences -A 5
```

> "Kubernetes sahipsiz şeyleri sevmez."

2️⃣ CASCADING DELETE (Zincirleme Silme)

Default davranış

Bir üst obje silinince altındakiler de silinir.

```
kubectl delete deployment web
```

Bu:

- ReplicaSet'leri
- Pod'ları

otomatik siler.

Ama...

İstersen bunu durdurabilirsin.

Orphan delete

```
kubectl delete deployment web --cascade=orphan
```

Sonuç:

- Deployment gider
- ReplicaSet + Pod'lar kalır

> Bu genelde **debug** veya **özel durumlar** için kullanılır.

3️⃣ TTL BASED GC (Job & Pod Temizliği)

Problem

- Job biter
- Pod Completed
- Ama Pod kalır

Çözüm

TTL Controller

```
spec:
  ttlSecondsAfterFinished: 300
```

Örnek Job

```
apiVersion: batch/v1
kind: Job
metadata:
  name: example-job
spec:
  ttlSecondsAfterFinished: 300
  template:
    spec:
      containers:
      - name: job
        image: busybox
        command: ["sh", "-c", "echo Hello"]
      restartPolicy: Never
```

🎯 Job biter → 5 dk sonra Pod + Job silinir

🧹 Kubernetes'te GC NELERİ TEMİZLER?

Obje	GC ile silinir mi?
Pod	✅
ReplicaSet	✅
Job	✅
Completed Pod	✅
Deployment	❌ (sen silersin)
PVC	❌ (bilerek silinmez)
PV	❌ (veri riski)

> **PVC & PV asla otomatik silinmez**
> Çünkü veri kaybı riski vardır.