CKAD Ders Dökümanı #18 — Rolling Update & Rollback

Dil: Türkçe • CKAD Kategorisi: Application Deployment (20%) • Güncelleme: maxSurge ve maxUnavailable (ve ilgili ayarlar) örneklerle detaylandırıldı

1) Konu Anlatımı

Deployment, uygulama sürüm geçişlerini güvenli şekilde yapabilmeniz için Kubernetes'in temel controller'ıdır. Varsayılan güncelleme stratejisi RollingUpdate'tır: yeni Pod'lar kademeli olarak ayağa kalkar, eski Pod'lar kademeli olarak kapanır. Bu sayede servis tamamen kesilmeden versiyon geçişi yapılır.

Rollback, hatalı bir rollout sonrası önceki çalışan ReplicaSet/Revision'a hızlıca dönmeyi sağlar. CKAD'de genelde image update → rollout takip → sorun varsa rollback adımları istenir.

2) maxSurge ve maxUnavailable Nedir?

RollingUpdate stratejisinde iki kritik parametre vardır: maxSurge ve maxUnavailable. Bu ikisi, rollout sırasında kapasite artışını ve kullanılamaz Pod sayısını kontrol eder.

2.1 maxSurge

maxSurge: Rollout sırasında istenen replica sayısının üzerine kaç adet ekstra Pod çıkılabileceğini belirtir. Değer sayı (örn. 1) veya yüzde (örn. 25%) olabilir.

Örnek: replicas=4 ve maxSurge: 1 ise rollout sırasında toplam Pod sayısı geçici olarak 5 olabilir. Bu, yeni Pod'u önce ayağa kaldırıp sonra eskiyi düşürerek kesintiyi azaltmaya yardımcı olur.

2.2 maxUnavailable

maxUnavailable: Rollout sırasında aynı anda kaç Pod'un kullanılamaz (Unavailable/NotReady) olmasına izin verildiğini belirtir. Değer sayı veya yüzde olabilir.

Örnek: replicas=4 ve maxUnavailable: 0 ise rollout boyunca 0 Pod unavailable olabilir. Yani Kubernetes, eski Pod'u kapatmadan önce yeni Pod'un Ready olmasını bekler. Bu ayar, yüksek erişilebilirlik gerektiren servislerde yaygındır.

2.3 Birlikte nasıl çalışırlar?

RollingUpdate genelde şu mantıkla ilerler: maxSurge kadar yeni Pod ekle → yeni Pod Ready olunca maxUnavailable sınırını aşmadan eski Pod'ları azalt. Özellikle maxUnavailable: 0 + maxSurge: 1 kombinasyonu "nereye kadar mümkünse kesintisiz" geçiş sağlar.

3) Örnekler (YAML + Senaryo)

3.1 Kesintisiz geçişe yakın ayar: maxSurge=1, maxUnavailable=0

Senaryo: replicas=3. Rollout sırasında en fazla 1 ekstra Pod açılabilir ve 0 Pod unavailable olabilir. Bu, yeni Pod'un hazır olmasını bekleyerek servis kesintisini minimize eder.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - name: nginx
        image: nginx:1.25
        ports:
        - containerPort: 80
```

3.2 Daha hızlı ama kesinti riski daha yüksek: maxSurge=0, maxUnavailable=1

Senaryo: replicas=3. Ekstra Pod açılmasına izin yok (maxSurge=0). 1 Pod'u aynı anda unavailable yapmaya izin var. Bu, daha hızlı ilerleyebilir ama kapasite düşebilir.

```
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxSurge: 0
    maxUnavailable: 1
```

3.3 Yüzde ile kullanım örneği: 25% / 25%

Büyük replica sayılarında yüzde kullanımı daha pratiktir. Örn. replicas=20 ise 25% ≈ 5 Pod demektir.

```
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxSurge: 25%
    maxUnavailable: 25%
```

4) İlgili Diğer Ayarlar ve İşlevleri

Ayar	Açıklama
strategy.type	RollingUpdate (varsayılan) veya Recreate (tüm pod'ları kapatıp yeniden a
minReadySeconds	Pod Ready olduktan sonra 'hazır sayılması' için bekleme. Hızlı Ready flip/
progressDeadlineSeconds	Rollout bu süre içinde ilerlemezse Deployment 'ProgressDeadlineExceed
revisionHistoryLimit	Rollback için tutulacak eski ReplicaSet sayısı. Çok düşük olursa eski revis
paused	Rollout'u geçici durdurma (kubectl rollout pause/resume).

4.1 minReadySeconds örneği

```
spec:
  minReadySeconds: 10
```

Pod Ready olsa bile 10 saniye stabil kalmadan 'available' sayılmaz. Özellikle readiness probe dalgalanıyorsa yararlı olabilir.

4.2 progressDeadlineSeconds örneği

```
spec:
  progressDeadlineSeconds: 120
```

Rollout 120 saniye içinde ilerleyemezse Deployment başarısız sayılır ve event'lerde nedeni görünür.

4.3 revisionHistoryLimit örneği

```
spec:
  revisionHistoryLimit: 3
```

Rollback için en fazla 3 eski revision tutulur. Daha fazlası gerekiyorsa yükseltebilirsiniz.

5) En Sık Kullanılan Komutlar (Rollout/Rollback)

5.1 Güncelleme (rollout başlat)

```
kubectl set image deployment/web nginx=nginx:1.26
# Alternatif: edit/apply ile image değiştir
kubectl edit deployment web
```

5.2 Rollout durumunu izle

```
kubectl rollout status deployment/web
kubectl get rs
kubectl get pod -l app=web
```

5.3 Geçmişi gör (history)

```
kubectl rollout history deployment/web
kubectl rollout history deployment/web --revision=2
```

5.4 Rollback

```
kubectl rollout undo deployment/web
kubectl rollout undo deployment/web --to-revision=2
```

5.5 Pause / Resume

```
kubectl rollout pause deployment/web
kubectl rollout resume deployment/web
```

6) Troubleshooting Hızlı Rehber

Rollout takıldı: Yeni Pod Ready olmuyor → kubectl describe pod + kubectl logs.Availability düşüyor: maxUnavailable yüksek olabilir; servis kapasitesini etkiler.Çok yavaş rollout: maxSurge düşük ve readiness/minReadySeconds yüksek olabilir (bilerek olabilir).Rollback sonrası da sorun: config/secret değişiklikleri veya bağımlı servisler de değişmiş olabilir; sadece image değil diğer bileşenleri kontrol et.

7) CKAD İpuçları

En sık kombinasyon: maxSurge: 1, maxUnavailable: 0 (kesintiyi minimize eder).Hızlı akış: set image → rollout status → sorun varsa rollout undo.Sorun analizinde: önce yeni Pod’lar neden Ready değil? (readiness probe, image pull, env/secret/config).Revision’ları kaybetmemek için revisionHistoryLimit çok küçük olmasın (en az 2-3 mantıklı).