# CKAD Ders Dökümanı #01 — Pod

Dil: Türkçe • Odak: Konu anlatımı + en sık kullanılan kubectl komutları

## 1) Pod Nedir?

Pod, Kubernetes üzerinde çalıştırılan en küçük deploy edilebilir birimdir. Bir Pod; bir veya birden fazla container'ı, bu container'ların paylaştığı ağ kimliğini (IP/port alanı) ve gerektiğinde paylaşılan depolamayı (volume) birlikte tanımlar. Genellikle "tek container = tek Pod" yaklaşımı yaygındır; ancak sidecar (log/agent) veya initContainer gibi desenlerde aynı Pod içinde birden fazla container bulunabilir.

### Pod'un temel özellikleri

Ephemeral: Pod yeniden oluşturulabilir; IP'si değişebilir. Kalıcı kimlik/ölçek için çoğu zaman Deployment/StatefulSet kullanılır. Paylaşımlı network namespace: Pod içindeki container'lar aynı IP'yi paylaşır, birbirine localhost ile erişebilir. Paylaşımlı volume: Container'lar aynı volume'u mount ederse dosya paylaşabilir. Scheduling: Pod bir Node'a schedule edilir; pod içindeki container'lar aynı Node'da birlikte yaşar.

## 2) Ne Zaman Pod Kullanılır?

CKAD sınavında Pod genellikle hızlı görev çözmek için kullanılır (imperative komutlar). Üretim senaryolarında ise doğrudan "çıplak Pod" kullanımı sınırlıdır; çoğunlukla Deployment/Job/CronJob gibi controller'lar tercih edilir.

### Tipik kullanım durumları

Hızlı test/debug (geçici Pod, curl/wget/dns testleri) Tek seferlik basit workload (sınav soruları gibi) Sidecar / initContainer desenleri (tek uygulama + yardımcı container)

## 3) Pod YAML Anatomisi (Temel Alanlar)

```
apiVersion: v1
kind: Pod
metadata:
  name: web
  labels:
    app: web
spec:
  containers:
  - name: nginx
    image: nginx:1.25
    ports:
    - containerPort: 80
    env:
    - name: ENV
      value: "prod"
    resources:
      requests:
        cpu: "100m"
        memory: "128Mi"
      limits:
        cpu: "200m"
```

```
      memory: "256Mi"
restartPolicy: Always
```

| Alan | Ne işe yarar? |
|---|---|
| metadata.name | Pod’un adı (namespace içinde benzersiz). |
| metadata.labels | Service/selector, filtreleme ve politika (NetworkPolicy vb.) için etiketler. |
| spec.containers[] | Pod içindeki container tanımları. |
| image | Çalıştırılacak container image’ı (tag ile birlikte). |
| ports.containerPort | Port bildirimi (okunabilirlik ve bazı araçlar için faydalı). |
| env / envFrom | Ortam değişkenleri (ConfigMap/Secret ile de beslenebilir). |
| resources | CPU/RAM requests & limits (scheduler ve limit enforcement). |
| restartPolicy | Restart davranışı (Always/OnFailure/Never). |

# 4) Pod ile İlgili En Sık Kullanılan kubectl Komutları

Aşağıdaki komutlar CKAD'de ve günlük pratikte en sık işinize yarayan Pod komutlarıdır. Her komutun yanında ne yaptığı ve örnek kullanımlar yer alır.

## 4.1 Pod oluşturma (imperative)

Hızlı Pod oluşturur. En iyi pratik: önce YAML üret, sonra düzenleyip apply et.

```
# Basit Pod oluştur
kubectl run web --image=nginx:1.25

# Belirli namespace'te oluştur
kubectl run web --image=nginx:1.25 -n dev

# YAML üret (oluşturmadan)
kubectl run web --image=nginx:1.25 --dry-run=client -o yaml

# Env ekleyerek oluştur
kubectl run web --image=nginx:1.25 --env="ENV=prod"
```

## 4.2 Pod listeleme ve filtreleme

Pod'ları listeler; namespace/label bazında filtreleme yapar; farklı çıktı formatları alırsınız.

```
kubectl get pod
kubectl get pod -A
kubectl get pod -o wide
kubectl get pod -l app=web
kubectl get pod web -o yaml
```

## 4.3 Pod detaylarını görme (describe)

Çalışmayan Pod'da ilk bakacağınız yer event'lerdir.

```
kubectl describe pod web
kubectl describe pod web -n dev
```

## 4.4 Log alma

Container loglarını gösterir. Çok container'lı Pod'da -c ile container seçin.

```
kubectl logs web
kubectl logs web -c nginx
kubectl logs -f web
kubectl logs web --previous
```

## 4.5 Pod içine girme / komut çalıştırma (exec)

Çalışan container içinde komut çalıştırır (debug).

```
kubectl exec -it web -- sh
kubectl exec -it web -c nginx -- sh
kubectl exec web -- ls -la
```

## 4.6 Port-forward

Local bilgisayarınızdan Pod portuna geçici tünel açar.

```
kubectl port-forward pod/web 8080:80
```

### 4.7 Pod silme

Pod’u siler. Controller varsa tekrar yaratılacağını unutmayın.

```
kubectl delete pod web
kubectl delete pod web -n dev
kubectl delete pod -l app=web
```

### 4.8 Pod Ready olana kadar bekleme (wait)

Otomasyon ve sınavda zaman kazandırır.

```
kubectl wait --for=condition=Ready pod/web --timeout=60s
kubectl wait --for=condition=Ready pod -l app=web --timeout=60s
```

### 4.9 JSONPath ile bilgi çekme

Pod çıktısından IP, node adı, image gibi alanları hızlı çekmek için kullanılır.

```
kubectl get pod web -o jsonpath='{.status.podIP}'
kubectl get pod web -o jsonpath='{.spec.nodeName}'
kubectl get pod web -o jsonpath='{.spec.containers[0].image}'
```

## 5) Pod Troubleshooting Hızlı Rehber

ImagePullBackOff / ErrImagePull: Image adı/tag doğru mu? Registry erişimi var mı? Önce kubectl describe pod. CrashLoopBackOff: Uygulama crash. kubectl logs ve gerekirse --previous. Pending: Kaynak yetersizliği / node selector / toleration. describe içindeki event’lere bakın. Ready değil: Readiness probe veya bağımlı servisler; trafik alamaz.

## 6) CKAD İpuçları (Pod)

Imperative → YAML: kubectl run ... --dry-run=client -o yaml ile iskelet üretin. En hızlı teşhis: kubectl describe pod + kubectl logs. Namespace’e dikkat: Görevler çoğu zaman belirli namespace’te olur.