CKAD Ders Dökümanı #05 — Commands and Arguments

Dil: Türkçe • CKAD Kategorisi: Application Design and Build (20%) • Odak: Konu anlatımı + YAML alanları + en sık kullanılan komutlar

1) Konu Anlatımı

Kubernetes'te bir container'ın nasıl başlatılacağını belirleyen iki temel kavram vardır: command ve args. Bunlar Docker dünyasındaki ENTRYPOINT ve CMDcommand alanı genellikle container image içindeki ENTRYPOINT'i, args ise CMD'yi override etmek için kullanılır.

CKAD'de sık senaryolar: Pod'un belirli bir komutla başlaması (ör. sleep, echo), image'ın default davranışını override etmek, aynı image'ı farklı komut/arg ile çalıştırmak, ve/veya giriş komutu içinde environment variable kullanmaktır.

Docker ↔ Kubernetes hızlı eşleştirme

Docker ENTRYPOINT ≈ Kubernetes command Docker CMD ≈ Kubernetes args Image'ın içindeki default ENTRYPOINT/CMD, Pod YAML'ında belirtilmezse aynen kullanılır. Pod YAML'ında command yazarsanız, image ENTRYPOINT'i override olur. Pod YAML'ında sadece args yazarsanız, image ENTRYPOINT korunur; CMD override olur.

2) En Sık Karşılaşılan YAML Örnekleri

2.1 Sadece args override (ENTRYPOINT korunur)

```
apiVersion: v1
kind: Pod
metadata:
  name: bb-args
spec:
  containers:
  - name: bb
    image: busybox:1.36
    args: ["sh", "-c", "echo Merhaba CKAD; sleep 3600"]
```

2.2 command + args override (ENTRYPOINT + CMD değişir)

```
apiVersion: v1
kind: Pod
metadata:
  name: bb-command-args
spec:
  containers:
  - name: bb
    image: busybox:1.36
    command: ["sh", "-c"]
    args: ["echo Başladı; date; sleep 3600"]
```

2.3 Tek satırlık komut (sleep) — sınav klasiği

```
apiVersion: v1
kind: Pod
metadata:
  name: sleeper
spec:
  containers:
  - name: sleep
    image: busybox:1.36
    command: ["sleep", "3600"]
```

2.4 env ile birlikte komut içinde değişken kullanma

```
apiVersion: v1
kind: Pod
metadata:
  name: env-cmd
spec:
  containers:
  - name: app
    image: busybox:1.36
    env:
    - name: MSG
      value: "Merhaba"
    command: ["sh","-c"]
    args: ["echo $MSG CKAD; sleep 3600"]
```

2.5 Birden fazla arg ile komut

```
containers:
- name: app
  image: nginx:1.25
  args: ["-g", "daemon off;"]
```

3) Sık Kullanılan Alanlar (Kısa Açıklamalar)

Alan	Ne işe yarar?
spec.containers[].command	Container’ın çalıştıracağı komutu (ENTRYPOINT) override eder. Dizi (arr
spec.containers[].args	Komut argümanlarını (CMD) override eder. Dizi formatı önerilir.
env	Komut içinde kullanılabilecek environment variable’lar.
workingDir	Komutun çalışacağı dizin (opsiyonel).
imagePullPolicy	Image çekme davranışı (Always/IfNotPresent/Never).

4) En Sık Kullanılan kubectl Komutları

CKAD'de command/args ile ilgili görevler genellikle Pod oluşturma ve YAML düzenleme üzerinden gelir. Aşağıdaki komutlar hızlı üretim ve doğrulama için en sık kullanılır.

4.1 Imperative Pod üretip YAML'a geçme (en hızlı yöntem)

Önce Pod iskeletini üretip YAML üzerinde command/args eklemek sınavda çok hız kazandırır.

```
# Pod iskeleti üret (oluşturmadan)
kubectl run sleeper --image=busybox:1.36 --dry-run=client -o yaml > pod.yaml

# Sonra pod.yaml içinde command/args ekle ve apply et
kubectl apply -f pod.yaml
```

4.2 Çalışan Pod'un komut/arg bilgilerini görme

Pod üzerinde gerçekten hangi command/args koştuğunu kontrol edebilirsiniz (özellikle debug için).

```
# YAML ile bak
kubectl get pod sleeper -o yaml | sed -n '/containers:/,/status:/p'

# JSONPath ile hızlı çek
kubectl get pod sleeper -o jsonpath='{.spec.containers[0].command}{"\n"}{.spec.containers[0].
```

4.3 Log ve exec ile doğrulama

Komutun çalıştığını ve doğru çıktıyı verdiğini doğrular.

```
kubectl logs env-cmd
kubectl exec -it env-cmd -- sh
```

4.4 Pod'u hızlı güncelleme (edit)

Sınavda küçük bir düzeltme gerektiğinde edit ile hızlı müdahale edebilirsiniz (bazı alanlar immutable olabilir).

```
kubectl edit pod sleeper
```

5) Troubleshooting Hızlı Rehber

Pod hemen kapanıyor: Komut bitmiş olabilir. Örn. sadece echo çalıştırırsanız container exit eder. Test için sleep ekleyin.Komut çalışmıyor / 'not found': Image içinde komut yok olabilir. busybox/alpine gibi image'larda komut seti sınırlı olabilir.Env değişkeni yazdırmıyor: shell üzerinden çalıştırdığınızdan emin olun. Örn. args: ["echo $MSG"] tek başına çalışmaz; sh -c gerekir.command/args karıştı: Sadece args yazınca ENTRYPOINT korunur. Tam override için command + args birlikte verin.

6) CKAD İpuçları

En sık pattern: busybox + sh -c + sleep (pod'u alive tutmak için).Env değişkeni kullanacaksan: command: ["sh","-c"] ve args: ["echo $VAR; sleep 3600"]Hız için: kubectl run ... --dry-run=client -o yaml ile iskelet üret, sonra YAML'da command/args ekle.Komut/args'ı mutlaka array (liste) formatında yaz. Tek string, shell parsing

farkları yaratabilir.