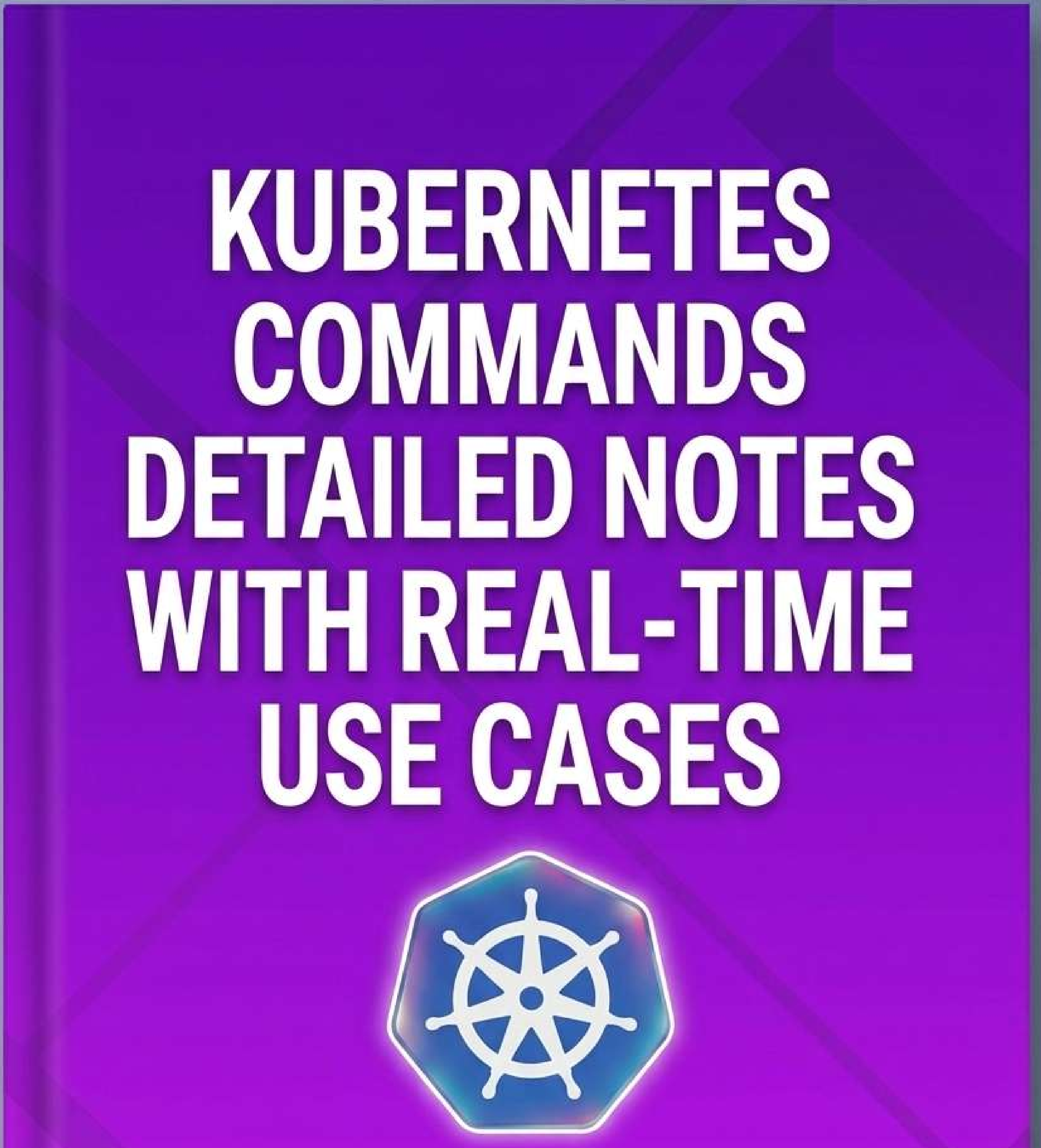

KUBERNETES COMMANDS DETAILED NOTES WITH REAL-TIME USE CASES

By Maaz Mohd

📘 Kubernetes Commands – Detailed Notes with Real-Time Use Cases

---

1️⃣ Cluster & Context Management

🔹 Check Cluster Info

```
kubectl cluster-info
```

Use case:
✓ Verify API server & CoreDNS
✓ First command after accessing a new cluster

Real-time scenario:
You joined a new project → need to confirm you're connected to the correct cluster.

🔹 View Nodes

```
kubectl get nodes
```

```
kubectl get nodes -o wide
```

Use case:
✓ Check node readiness
✓ Identify node IPs & OS

Real-time scenario:
Pods stuck in `Pending` → check if nodes are `NotReady`.

🔹 Switch Context

```
kubectl config get-contexts
kubectl config use-context prod-cluster
```

Use case:
✓ Avoid deploying to wrong cluster (VERY critical)

⚠️ **Production mistake:**
Many outages happen because engineers run commands in **prod instead of staging**.

2️⃣ Namespace Management

🔹 List Namespaces

```
kubectl get ns
```

🔹 Work in a Namespace

```
kubectl get pods -n dev
```

🔹 Set Default Namespace

```
kubectl config set-context --current --namespace=dev
```

Real-time scenario:
Microservices split across namespaces like:

- `dev`
- `qa`
- `prod`
- `monitoring`

3️⃣ Pod Management (MOST IMPORTANT)

🔹 List Pods

```
kubectl get pods
```

```
kubectl get pods -o wide
```

Use case:
✔ Pod status
✔ Node placement

🔹 Describe Pod (DEBUGGING KING 👑)

```
kubectl describe pod my-pod
```

Shows:
✔ Events
✔ Resource limits
✔ Image pull errors
✔ OOMKilled reason

Real-time scenario:
Pod in `CrashLoopBackOff` → this is your **first command**.

🔹 Pod Logs

```
kubectl logs my-pod
```

Multi-container pod:

```
kubectl logs my-pod -c app
```

Previous crash:

```
kubectl logs my-pod --previous
```

Real-time scenario:
Application crashes after deployment → logs reveal config/env issues.

🔹 Exec into Pod

```
kubectl exec -it my-pod -- /bin/sh
```

Use case:
✓ Check files
✓ Test DB connectivity
✓ Curl internal services

Production rule:
Use exec only for **debugging**, not permanent fixes.

4️⃣ Deployment Management

🔹 List Deployments

```
kubectl get deploy
```

🔹 Describe Deployment

```
kubectl describe deploy my-app
```

🔹 Scale Deployment

```
kubectl scale deploy my-app --replicas=5
```

Real-time scenario:
Traffic spike → manually scale until HPA kicks in.

🔹 Rollout Status

```
kubectl rollout status deploy my-app
```

🔹 Rollback Deployment

```
kubectl rollout undo deploy my-app
```

Real-time scenario:
New version breaks production → **instant rollback** saves SLA.

5️⃣ Service & Networking

🔹 List Services

```
kubectl get svc
```

🔹 Describe Service

```
kubectl describe svc my-service
```

Service Types:

- ClusterIP → internal
- NodePort → basic external
- LoadBalancer → cloud-managed

Real-time scenario:
Application running but not accessible → check service type & endpoints.

🔹 Check Endpoints

```
kubectl get endpoints my-service
```

✓ Empty endpoints = selector mismatch (VERY common issue)

6️⃣ ConfigMaps & Secrets

🔹 List ConfigMaps

```
kubectl get cm
```

🔹 View ConfigMap

```
kubectl describe cm app-config
```

🔹 List Secrets

```
kubectl get secrets
```

🔹 Decode Secret

```
kubectl get secret db-secret -o yaml
echo "encoded_value" | base64 --decode
```

Real-time scenario:

App failing DB connection → secret value mismatch.

7️⃣ Resource & Performance Debugging

🔹 Pod Resource Usage

```
kubectl top pod
```

🔹 Node Resource Usage

```
kubectl top node
```

Requires: metrics-server

🔹 OOMKilled Debug

```
kubectl describe pod my-pod
```

Look for:

```
Last State: Terminated
Reason: OOMKilled
```

Fix:

Increase memory limits or optimize app.

8️⃣ Events & Troubleshooting

🔹 View Events

```
kubectl get events --sort-by=.metadata.creationTimestamp
```

Real-time scenario:
ImagePullBackOff, FailedScheduling, Probe failures.

9️⃣ YAML & Apply Flow (GitOps Friendly)

🔹 Apply YAML

```
kubectl apply -f deployment.yaml
```

🔹 Dry Run

```
kubectl apply -f deployment.yaml --dry-run=client
```

🔹 Delete Resource

```
kubectl delete -f deployment.yaml
```

🔟 Common Production Debug Flow (INTERVIEW GOLD ⭐)

When **Pod not running**:

```
kubectl get pods
kubectl describe pod
kubectl logs
kubectl exec
kubectl get events
```

When **Service not accessible**:

```
kubectl get svc
kubectl get endpoints
kubectl describe svc
kubectl exec → curl service
```

When **High CPU/Memory**:

```
kubectl top pod
kubectl describe pod
Check limits & requests
```