Kubernetes Examples & Syntax Explained (Roman Urdu Edition)

Ye PDF Minikube-focused examples aur har YAML line-by-line samjhaai ke sath hai. Roman Urdu slang me likha gaya taake seedha samajh aaye.

1) Pod - Simple Example

Pod sabse chhoti unit hai. Ek pod me ek ya zyada containers rehte hain. Yeh example ek simple nginx pod banata hai.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: nginx
spec:
  containers:
    - name: nginx
      image: nginx:stable
      ports:
        - containerPort: 80
```

Line-by-line (Roman Urdu):

apiVersion: v1 — Bhai ye batata hai ke Pod object Kubernetes v1 API use kar raha hai.

kind: Pod — Ye type batata hai ke hum Pod bana rahe hain.

metadata: name: nginx-pod — Pod ka naam; labels: app: nginx — label jo service ya selector me use hoga.

spec: containers: — Pod ke andar chalne wale containers list; - name: nginx — container ka friendly naam.

image: nginx:stable — Docker image jo container run karega. tag stable fix karna acha practice hai.

ports: containerPort: 80 — Container ke andar jo port app use kar rahi hai (Nginx 80).

2) Deployment - Manage Pods & Scaling

Deployment ReplicaSets banata hai aur pods ko manage karta hai. Rolling updates aur scaling yahin se hota hai.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-deploy
  labels:
    app: web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web
  template:
    metadata:
```

```
      labels:
        app: web
    spec:
      containers:
        - name: web
          image: nginx:stable
          ports:
            - containerPort: 80
```

Line-by-line (Roman Urdu):

apiVersion: apps/v1 — Deployment apps API group me aata hai.

kind: Deployment — Ye ek Deployment object hai jo ReplicaSet manage karega.

replicas: 3 — Kitne copies chahiye (scale).

selector.matchLabels — Deployment kis pods ko manage karega; template ke labels se match hona chahiye.

template — Ye woh Pod template hai jo ReplicaSet se pods banega.

image: nginx:stable — Deployment me image specify karte ho; update karne se new pods nikalte hain.

3) Service - ClusterIP & NodePort (Minikube)

Service pods ke upar stable networking layer deti hai. ClusterIP internal hota hai; NodePort local testing ke liye use hota hai.

```
apiVersion: v1
kind: Service
metadata:
  name: web-clusterip
spec:
  selector:
    app: web
  ports:
    - port: 80
      targetPort: 80
  type: ClusterIP
```

ClusterIP service internal communication ke liye. selector: app: web — ye service unhi pods ko route karega jinpe label app=web ho.

```
apiVersion: v1
kind: Service
metadata:
  name: web-nodeport
spec:
  selector:
    app: web
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30080
  type: NodePort
```

NodePort service se tum apne host machine (Minikube) pe port 30080 se access kar sakte ho: :30080.

4) Namespace - Logical Separation

Namespace se alag environments banate ho: dev, test, prod. Resources same name ke sath alag namespaces me exist kar sakte hain.

```
apiVersion: v1
kind: Namespace
metadata:
  name: dev-ns
```

namespace create karne ke baad resources apply karte waqt -n dev-ns use karo ya metadata me namespace: add karo.

5) Ingress - Single Entry Point (Minikube)

Ingress ek HTTP router hai. Minikube me pehle ingress addon enable karo: minikube addons enable ingress. Phir Ingress rules laga ke multiple services ek domain se route karo.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: app-ingress
  namespace: dev-ns
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
    - host: myapp.local
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: frontend-svc
                port:
                  number: 80
          - path: /api
            pathType: Prefix
            backend:
              service:
                name: backend-svc
                port:
                  number: 5000
```

Host ko /etc/hosts me map karo: myapp.local. Annotations nginx rules ko help karte hain (rewrite target).

6) Volumes - emptyDir & hostPath

emptyDir pod ke life-time tak data rakhta hai. hostPath node ke filesystem se bind karta hai (local testing).

```
apiVersion: v1
kind: Pod
metadata:
  name: empty-pod
```

```
spec:
  containers:
    - name: writer
      image: busybox
      command: ["sh", "-c", "echo hello > /data/msg.txt && sleep 3600"]
      volumeMounts:
        - name: data
          mountPath: /data
  volumes:
    - name: data
      emptyDir: {}
```

emptyDir: {} — Pod chalne pe directory banti hai node pe aur pod delete hone pe remove ho jati hai.

```
apiVersion: v1
kind: Pod
metadata:
  name: hostpath-pod
spec:
  containers:
    - name: hostwriter
      image: busybox
      command: ["sh", "-c", "echo hi > /host/data.txt && sleep 3600"]
      volumeMounts:
        - name: hostvol
          mountPath: /host
  volumes:
    - name: hostvol
      hostPath:
        path: /tmp/k8s-hostpath
        type: DirectoryOrCreate
```

hostPath direct node filesystem pe likhta hai — sirf local testing ke liye use karo, production me PV/PVC better hai.

7) PersistentVolume (PV) & PersistentVolumeClaim (PVC)

PVC se app request karta hai storage ke liye; PV admin ya cluster provide karta hai. Minikube me hostPath PV demo dekhte hain.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-demo
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /mnt/data/pv-demo
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-demo
spec:
  accessModes:
    - ReadWriteOnce
```

```
  resources:
    requests:
      storage: 500Mi
```

PV admin ne create kiya (hostPath), PVC user ne request ki — bind hone par PVC se pod me volume mount karoge.

8) Job & CronJob - One-time & Scheduled Tasks

Job ek baar chalne wali task hai; CronJob scheduler jaisa hai (cron expression).

```
apiVersion: batch/v1
kind: Job
metadata:
  name: one-time-job
spec:
  template:
    spec:
      containers:
        - name: task
          image: busybox
          command: ["sh", "-c", "echo Backup done > /backup/log.txt"]
      restartPolicy: Never
apiVersion: batch/v1
kind: CronJob
metadata:
  name: daily-job
spec:
  schedule: "0 2 * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: daily
              image: busybox
              command: ["sh", "-c", "echo Daily task >> /backup/log.txt"]
          restartPolicy: Never
```

restartPolicy: Never — job pod fail ho to re-run policy alag se handle karega.

9) ConfigMap & Secret - Configs and Sensitive Data

ConfigMap non-sensitive configuration rakhta hai; Secret sensitive data (base64 encoded) rakhta hai.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  APP_MODE: production
  LOG_LEVEL: info
apiVersion: v1
kind: Secret
metadata:
  name: db-secret
type: Opaque
```

```
data:
  username: YWRtaW4=
  password: cGFzc3dvcmQ=
```

Secret me values base64 encode hoti hain; kubectl create secret generic --from-literal=key=value bhi use hota hai.

10) Mini Project: Frontend + Backend (Minikube)

Short summary: Backend Node app on port 5000, Frontend nginx, Services ClusterIP, Ingress routes / and /api.

```
# Backend Deployment (inline Node)
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
  namespace: dev-ns
spec:
  replicas: 2
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
        - name: node-back
          image: node:16
          command: ["node","-e","require('http').createServer((req,res)=>res.end('OK')).listen(
---
# Backend Service
apiVersion: v1
kind: Service
metadata:
  name: backend-svc
  namespace: dev-ns
spec:
  selector:
    app: backend
  ports:
    - port: 5000
      targetPort: 5000
  type: ClusterIP
# Frontend Deployment (nginx)
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  namespace: dev-ns
spec:
  replicas: 2
  selector:
    matchLabels:
      app: frontend
```

```
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - containerPort: 80
---
# Frontend Service
apiVersion: v1
kind: Service
metadata:
  name: frontend-svc
  namespace: dev-ns
spec:
  selector:
    app: frontend
  ports:
    - port: 80
      targetPort: 80
  type: ClusterIP
```

Apply: kubectl apply -f . Create ingress and /etc/hosts mapping to test locally.

11) Bonus: Top kubectl Commands (Minikube)

```
# Get nodes and pods
kubectl get nodes
kubectl get pods -A

# Describe and logs
kubectl describe pod <pod-name>
kubectl logs <pod-name>

# Apply & delete files
kubectl apply -f file.yaml
kubectl delete -f file.yaml

# Exec into pod
kubectl exec -it <pod-name> -- sh

# Port-forward to local host
kubectl port-forward svc/frontend-svc 8080:80

# Scale deployment
kubectl scale deployment backend --replicas=4 -n dev-ns

# Get services and ingress
kubectl get svc -n dev-ns
kubectl get ingress -n dev-ns
```

Practice ye commands daily. Ye tumhara bread-and-butter honge jab cluster manage karoge.

12) Quick Cheatsheet

```
# Create namespace
kubectl create namespace dev-ns

# Apply file
kubectl apply -f file.yaml -n dev-ns

# Show pods in namespace
kubectl get pods -n dev-ns

# Delete pod
kubectl delete pod <pod-name> -n dev-ns
```

Ye short cheatsheet raat ko practice ke liye rakho. Baar baar chalane se haath set ho jayega.