REFERENCIA RÁPIDA DE KUBERNETES

kubectl, pods, deployments, services, configs, depuración

Fundamentos de kubectl

Información del clúster

```
kubectl cluster-info
kubectl get nodes
kubectl config current-context
kubectl config use-context my-cluster
```

Comandos esenciales

`kubectl get <resource>`	Listar recursos
`kubectl describe <resource> <name>`	Información detallada del recurso
`kubectl create -f file.yaml`	Crear recurso desde archivo
`kubectl apply -f file.yaml`	Crear o actualizar recurso
`kubectl delete -f file.yaml`	Eliminar recurso desde archivo
`kubectl edit <resource> <name>`	Editar recurso en el lugar
`kubectl api-resources`	Listar todos los tipos de recurso

Formatos de salida

`-o wide`	Columnas adicionales (IP, nodo)
`-o yaml`	Salida YAML completa
`-o json`	Salida JSON completa
`-o jsonpath='{.spec}'`	Extraer campos específicos
`--sort-by=.metadata.name`	Ordenar salida por campo

Pods

Operaciones con pods

```
kubectl get pods
kubectl get pods -A                 # all namespaces
kubectl run nginx --image=nginx     # quick pod
kubectl delete pod nginx
```

YAML de pod

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp
  labels: { app: myapp }
spec:
  containers:
    - name: app
      image: nginx:1.27
      ports:
        - containerPort: 80
```

Valores de estado del pod

`Running`	Todos los contenedores iniciados
`Pending`	Esperando planificación o descarga de imagen
`CrashLoopBackOff`	El contenedor sigue fallando y reiniciándose
`ImagePullBackOff`	No se puede descargar la imagen del contenedor
`Completed`	Ejecutado hasta completarse (Jobs)

Deployments

YAML de deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels: { app: web }
  template:
    metadata:
      labels: { app: web }
    spec:
      containers:
        - name: web
          image: nginx:1.27
          ports:
            - containerPort: 80
```

Comandos de deployment

`kubectl get deploy`	Listar deployments
`kubectl scale deploy web --replicas=5`	Escalar réplicas
`kubectl set image deploy/web web=nginx:1.28`	Actualizar imagen (rolling)
`kubectl rollout status deploy/web`	Ver progreso del rollout
`kubectl rollout undo deploy/web`	Revertir a la revisión anterior
`kubectl rollout history deploy/web`	Ver historial de revisiones

Services

Tipos de service

`ClusterIP`	Solo interno (por defecto)
`NodePort`	Exponer en la IP de cada nodo en un puerto estático
`LoadBalancer`	Balanceador de carga externo (cloud)
`ExternalName`	Alias DNS a servicio externo

YAML de service

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

Exponer rápidamente

```
kubectl expose deploy web --port=80 --type=ClusterIP
kubectl expose deploy web --port=80 --type=NodePort
kubectl get svc
```

ConfigMaps y Secrets

ConfigMap

```
kubectl create configmap app-cfg \
  --from-literal=DB_HOST=db.example.com \
  --from-file=config.ini
```

Secret

```
kubectl create secret generic db-creds \
  --from-literal=username=admin \
  --from-literal=password=s3cret
```

Usar en pods

```
# As environment variables
envFrom:
  - configMapRef: { name: app-cfg }
  - secretRef:    { name: db-creds }

# As volume mount
volumes:
  - name: cfg
    configMap: { name: app-cfg }
```

Comandos

`kubectl get cm`	Listar ConfigMaps
`kubectl get secret`	Listar Secrets
`kubectl describe cm app-cfg`	Mostrar datos del ConfigMap
`kubectl get secret db-creds -o yaml`	Mostrar Secret (codificado en base64)

Namespaces

Comandos de namespace

`kubectl get ns`	Listar namespaces
`kubectl create ns staging`	Crear namespace
`kubectl delete ns staging`	Eliminar namespace y todos sus recursos
`kubectl get pods -n staging`	Listar pods en el namespace
`kubectl get pods -A`	Listar pods en todos los namespaces

Establecer namespace por defecto

```
kubectl config set-context --current \
  --namespace=staging
```

Volúmenes

PersistentVolumeClaim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: data-pvc
spec:
  accessModes: [ReadWriteOnce]
  resources:
    requests: { storage: 10Gi }
```

Montar en pod

```
volumes:
  - name: data
    persistentVolumeClaim:
      claimName: data-pvc
containers:
  - volumeMounts:
      - name: data
        mountPath: /app/data
```

Tipos de volumen

`emptyDir`	Directorio temporal, eliminado con el pod
`hostPath`	Montar ruta del sistema de archivos del host
`persistentVolumeClaim`	Almacenamiento persistente (PVC)
`configMap`	Montar ConfigMap como archivos
`secret`	Montar Secret como archivos

Ingress

YAML de Ingress

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: web-ingress
spec:
  rules:
    - host: app.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web-svc
                port: { number: 80 }
```

Notas sobre Ingress

`Ingress Controller`	Obligatorio (nginx-ingress, traefik, etc.)
`pathType: Prefix`	Coincidir con el prefijo de la URL
`pathType: Exact`	Coincidir con la ruta URL exacta
`TLS`	Agregar sección `tls:` con nombre del secret

Depuración

Comandos de diagnóstico

`kubectl logs <pod>`	stdout/ stderr del contenedor
`kubectl logs <pod> -c <ctr>`	Logs de un contenedor específico
`kubectl logs <pod> --previous`	Logs del contenedor que falló
`kubectl describe pod <pod>`	Eventos, condiciones, estado
`kubectl exec -it <pod> -- sh`	Shell dentro del contenedor
`kubectl port-forward <pod> 8080:80`	Reenviar puerto local al pod
`kubectl top pods`	Uso de CPU/ memoria (metrics-server)
`kubectl get events --sort-by=.lastTimestamp`	Línea de tiempo de eventos del clúster

Pod de depuración

```
kubectl run debug --rm -it --image=busybox -- sh
# or attach ephemeral container
kubectl debug -it <pod> --image=busybox
```

Patrones comunes

Etiquetas y selectores

```
kubectl get pods -l app=web
kubectl get pods -l 'env in (prod,staging)'
kubectl label pod myapp env=prod
```

Límites de recursos

```
resources:
  requests: { cpu: 100m, memory: 128Mi }
  limits:   { cpu: 500m, memory: 256Mi }
```

Liveness y Readiness

```
livenessProbe:
  httpGet: { path: /healthz, port: 8080 }
  initialDelaySeconds: 5
  periodSeconds: 10
readinessProbe:
  httpGet: { path: /ready, port: 8080 }
```

Recetas rápidas

`Dry run`	`kubectl apply -f file.yaml --dry-run=client`
`Generate YAML`	`kubectl create deploy web --image=nginx --dry-run=client -o yaml`
`Watch`	`kubectl get pods -w`
`Copy files`	`kubectl cp file.txt pod:/tmp/`
`Restart deploy`	`kubectl rollout restart deploy/web`