Kubernetes_

Guía de atajos · kubectl y DevSecOps

```
user@dajomar:~$ kubectl get pods
```

Arquitectura

Control plane	Función
kube-apiserver	La puerta de entrada: todo pasa por su API.
etcd	Base de datos del clúster. ⚠ Guarda los Secrets.
scheduler	Decide en qué nodo corre cada pod.
controller-manager	Mantiene el estado deseado (réplicas, nodos).
Cada nodo	
kubelet	Agente que arranca y vigila los pods.
kube-proxy	Reglas de red para los Services.
runtime	Ejecuta los contenedores (containerd).

Igual que Terraform, Kubernetes es **declarativo**: le dices "quiero 3 réplicas" y él se encarga de que siempre haya 3.

Objetos principales

Pod	Unidad mínima: uno o más contenedores juntos.
Deployment	Mantiene N réplicas de un pod y hace rollouts.
StatefulSet	Pods con identidad y disco fijo (bases de datos).
DaemonSet	Un pod en cada nodo (agentes de logs, EDR).
Job / CronJob	Tareas que terminan / programadas.
Service	IP y DNS estables para un grupo de pods.
Ingress	Entrada HTTP/HTTPS desde fuera del clúster.
ConfigMap	Configuración no sensible.
Secret	Datos sensibles (⚠ solo en base64).
Namespace	Separa recursos por equipo o entorno.
PV / PVC	Disco persistente / solicitud de disco.

Contextos y namespaces

```
kubectl config get-contexts
kubectl config use-context prod-cluster
kubectl config current-context
kubectl get ns
kubectl config set-context --current \
  --namespace=mi-app
```

La config vive en `~/.kube/config`. ⚠ Contiene credenciales: trátala como una contraseña.

Pro Tip:
Antes de cualquier cambio corre `kubectl config current-context`. El error más caro de Kubernetes es borrar algo en **prod** creyendo que estabas en **dev**.

Consultar

```
kubectl get pods  # namespace actual
kubectl get pods -A  # todos los namespaces
kubectl get pods -o wide         # IP y nodo
kubectl get pods -w  # observar en vivo
kubectl get deploy,svc,ingress
kubectl get pod web-7d9f -o yaml
kubectl get pods -l app=web      # por etiqueta
kubectl describe pod web-7d9f    # eventos!
kubectl explain deployment.spec  # documentación
kubectl api-resources
```

Crear y aplicar

```
kubectl apply -f deploy.yaml
kubectl apply -f k8s/        # carpeta completa
kubectl diff -f deploy.yaml  # qué cambiaría
kubectl delete -f deploy.yaml
# generar YAML sin crear nada
kubectl create deployment web --image=nginx \
  --dry-run=client -o yaml > deploy.yaml
```

Deployments y rollouts

```
kubectl scale deploy web --replicas=3
kubectl set image deploy/web web=nginx:1.27
kubectl rollout status deploy/web
kubectl rollout history deploy/web
kubectl rollout undo deploy/web  # volver atrás
kubectl rollout restart deploy/web
```

Depurar

```
kubectl logs web-7d9f
kubectl logs -f deploy/web  # seguir en vivo
# antes del crash
kubectl logs web-7d9f --previous
# otro contenedor
kubectl logs web-7d9f -c sidecar
kubectl exec -it web-7d9f -- sh
kubectl port-forward svc/web 8080:80
kubectl get events --sort-by=.lastTimestamp
kubectl top pods  # requiere metrics-server
kubectl debug -it web-7d9f --image=busybox
```

Estado	Causa probable
Pending	Sin recursos en los nodos o PVC sin disco.
ImagePullBackOff	Imagen o tag incorrecto, o registry privado sin credenciales.
CrashLoopBackOff	La app arranca y se cae: mira `logs --previous`.
OOMKilled	Superó su límite de memoria.
CreateContainer ConfigError	Falta un ConfigMap o Secret referenciado.

Deployment seguro (YAML)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 2
  selector:
    matchLabels: { app: web }
  template:
    metadata:
      labels: { app: web }
    spec:
      automountServiceAccountToken: false
      securityContext:
        runAsNonRoot: true
        seccompProfile: { type: RuntimeDefault }
      containers:
        - name: web
          image: registry.io/web:1.4.2  # tag fijo
          ports: [{ containerPort: 8080 }]
          securityContext:
            allowPrivilegeEscalation: false
            readOnlyRootFilesystem: true
            capabilities: { drop: ["ALL"] }
          resources:
            requests: { cpu: 100m, memory: 128Mi }
            limits:   { memory: 256Mi }
          readinessProbe:
            httpGet: { path: /health, port: 8080 }
```

Services

ClusterIP	Solo dentro del clúster (por defecto).
NodePort	Abre un puerto (30000-32767) en cada nodo.
LoadBalancer	Balanceador externo de la nube.

```
kubectl expose deploy web --port=80 \
  --target-port=8080
```

Dentro del clúster se llega por DNS: `web.mi-app.svc.cluster.local` o solo `web` desde el mismo namespace.

ConfigMaps y Secrets

```
kubectl create configmap app-cfg \
  --from-literal=LOG_LEVEL=info
kubectl create secret generic db \
  --from-literal=password='S3cr3t!'
kubectl get secret db \
  -o jsonpath='{.data.password}' | base64 -d
```

⚠ **base64 no es cifrado.** Cualquiera con permiso de leer Secrets ve las contraseñas. Activa el cifrado de etcd y usa External Secrets, Sealed Secrets o un vault.

DevSecOps: RBAC

```
kubectl auth can-i delete pods
kubectl auth can-i --list \
  --as=system:serviceaccount:mi-app:default
kubectl get roles,rolebindings -n mi-app
kubectl get clusterrolebindings
```

Role/RoleBinding dan permisos en un namespace; **ClusterRole/ClusterRoleBinding**, en todo el clúster. Mínimo privilegio: nada de `cluster-admin` para apps ni personas en el día a día.

DevSecOps: aislar y endurecer

Pod Security Admission por namespace

```
kubectl label ns mi-app \
  pod-security.kubernetes.io/enforce=restricted
```

NetworkPolicy: bloquear todo por defecto

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
  namespace: mi-app
spec:
  podSelector: {}
  policyTypes: ["Ingress", "Egress"]
```

Después abres solo el tráfico necesario. Requiere un CNI compatible (Calico, Cilium).

Escaneo y auditoría

```
trivy image registry.io/web:1.4.2  # CVEs
trivy k8s --report summary         # clúster
kubescape scan                     # NSA / CIS
kube-bench run  # CIS en los nodos
checkov -d k8s/  # YAML antes de aplicar
```

Helm

```
helm repo add bitnami \
  https://charts.bitnami.com/bitnami
helm repo update
helm search repo nginx
helm upgrade --install web bitnami/nginx \
  -n web --create-namespace -f values.yaml
helm list -A
helm template web bitnami/nginx  # ver YAML
helm rollback web 1
helm uninstall web -n web
```

Laboratorio local

kind	Clúster dentro de Docker. Ideal para CI y pruebas.
minikube	Clúster local con addons fáciles.
k3s	Kubernetes liviano. Perfecto para un homelab.

```
kind create cluster --name lab
kind delete cluster --name lab
```

Pro Tip:
Agrega a tu `~/.bashrc`: `alias k=kubectl` y `source <(kubectl completion bash)`. Para cambiar de contexto y namespace rápido, instala `kubectx` y `kubens`.