Kubernetes

Apuntes personales, guías prácticas y cosas que fui entendiendo mientras intento que todo funcione. Desde la instalación hasta los primeros despliegues, y lo que vaya necesitando en el camino.

Apuntes iniciales de Kubernetes

Estos apuntes no pretenden ser una guía completa, sino un registro personal mientras aprendo Kubernetes. Sirven para tener a mano los comandos básicos, los errores más comunes y las ideas que conviene no olvidar al empezar.

Fundamentos del comando `kubectl`

Sintaxis general:

```
kubectl [comando] [tipo_recurso] [nombre] [flags]
```

Ejemplo rápido:

```
kubectl get pods -n kube-system
```

Comandos esenciales

Comando	Descripción
`kubectl get pods`	Lista los pods en el namespace actual.
`kubectl get all`	Muestra todos los recursos del namespace.
`kubectl describe pod <nombre>`	Detalla estado, eventos y errores de un pod.
`kubectl logs <pod>`	Muestra los logs de un pod.
`kubectl exec -it <pod> -- /bin/bash`	Accede a una shell dentro del pod.
`kubectl apply -f <archivo.yaml>`	Crea o actualiza recursos de forma declarativa.
`kubectl delete -f <archivo.yaml>`	Elimina los recursos definidos en el archivo.
`kubectl create deployment nginx --image=nginx`	Crea un deployment de forma imperativa.
`kubectl get namespaces`	Lista todos los namespaces.
`kubectl create namespace <ns>`	Crea un nuevo namespace.

Comando	Descripción
`kubectl config get-contexts`	Muestra los contextos configurados.
`kubectl config use-context <contexto>`	Cambia entre contextos de clúster.
`kubectl scale deployment <nombre> --replicas=<n>`	Escala un deployment.
`kubectl set image deployment/<nombre> <contenedor>=<imagen>`	Actualiza la imagen de un contenedor.
`kubectl top pod` / `kubectl top node`	Muestra el uso de CPU y memoria.
`kubectl get events`	Lista los eventos recientes del clúster.
`kubectl cordon <nodo>`	Marca un nodo como no programable.
`kubectl drain <nodo>`	Desprograma los pods de un nodo.
`kubectl uncordon <nodo>`	Reactiva el nodo para nuevas cargas.

Consejos prácticos

- Usa `alias k=kubectl` para escribir menos.
- Añade `-n <namespace>` para trabajar fuera del namespace `default`.
- Prefiere `apply` frente a `create` o `run` en entornos reales.
- Al depurar: combina `describe`, `logs` y `events`.
- Antes de culpar al clúster, revisa tu `kubeconfig` y el contexto activo.

Flujo típico de trabajo

```
# 1. Ver qué hay
kubectl get all -n miapp

# 2. Revisar logs
kubectl logs <pod>

# 3. Entrar al contenedor
kubectl exec -it <pod> -- /bin/bash

# 4. Aplicar cambios
kubectl apply -f deployment.yaml

# 5. Ver resultados
kubectl describe deployment miapp
```

Lo que hay que tener presente...

1. **Curva de aprendizaje empinada** Kubernetes no es “Docker elevado”. Hay conceptos complejos (control plane, API server, etcd, scheduler, controllers). No esperes que todo tenga sentido de golpe.
2. **Manifestos YAML = fuente de errores humanos** Un indentado mal, un label selector incorrecto o un puerto mal mapeado pueden romper el sistema. Revisa siempre que tus `spec.selector.matchLabels` coincidan con `template.metadata.labels`.
3. **No usar `latest` en las imágenes** La etiqueta `latest` cambia sin aviso. Evítala en producción.
4. **Define probes (liveness / readiness)** Si no lo haces, tu aplicación puede parecer viva aunque no sirva tráfico correctamente.
5. **Requests / Limits son obligatorios** Define siempre límites de CPU y memoria para evitar que un pod consuma todo un nodo.
6. **Namespaces y contexto** Comprueba siempre `kubectl config current-context` y usa `-n <namespace>` para evitar confusiones.
7. **Seguridad desde el día cero**
   - Activa y configura RBAC con mínimos privilegios.
   - No expongas la API del clúster a internet.
   - Gestiona secretos con cifrado y control de acceso.
   - Define `securityContext` y `NetworkPolicies` para limitar privilegios y tráfico.
   - Activa auditorías y logs.
8. **Diseña para fallar** Kubernetes no elimina errores: los gestiona. Usa `readinessProbes`, *rolling updates* y *PodDisruptionBudgets*.
9. **Observabilidad no es opcional** Métricas, logs y alertas son esenciales para entender el estado del clúster.
10. **Versiones y compatibilidad** Las APIs cambian rápido. Mantén tus manifiestos y clúster actualizados.
11. **Segmentación lógica** Divide por entorno (dev, staging, prod) o por equipo. Evita permisos amplios o namespaces únicos.
12. **Infraestructura declarativa / GitOps** No gestiones todo a mano. Usa Terraform, Helm, ArgoCD o similares para mantener reproducibilidad y control.

> “ Referencia útil: Kubernetes kubectl Cheat Sheet (oficial)

Componentes y conceptos básicos de Kubernetes

Mapa rápido de las partes que componen un clúster y los objetos más habituales que se gestionan. No pretende explicar cada detalle, solo servir de referencia visual y orden mental.

Arquitectura general

Control Plane (el cerebro del clúster):

- **API Server** → punto de entrada de todas las peticiones (`kubectl`, dashboards, controladores).
- **etcd** → base de datos donde se guarda el estado del clúster.
- **Controller Manager** → vigila que el estado real coincida con el deseado (replicas, jobs, etc.).
- **Scheduler** → decide en qué nodo se ejecuta cada Pod.

Nodos de trabajo (Workers):

- **kubelet** → agente que ejecuta Pods y reporta su estado.
- **kube-proxy** → gestiona reglas de red y balanceo.
- **Container Runtime** → ejecuta los contenedores (containerd, CRI-O, Docker, etc.).

Recursos fundamentales

Tipo	Descripción rápida
Pod	Unidad mínima ejecutable: uno o más contenedores que comparten red y almacenamiento.
ReplicaSet	Asegura que haya N réplicas de un Pod. Suele gestionarse a través de un Deployment.
Deployment	Controla la creación y actualización de Pods. Ideal para aplicaciones sin estado.
StatefulSet	Igual que Deployment, pero mantiene identidad y almacenamiento por Pod.

Tipo	Descripción rápida
DaemonSet	Asegura que haya un Pod corriendo en cada nodo (útil para logs, monitorización).
Job / CronJob	Ejecutan tareas puntuales o programadas.

Almacenamiento

Recurso	Función
Volume	Espacio de almacenamiento montado en un Pod. Puede ser efímero o persistente.
PersistentVolume (PV)	Recurso del clúster que representa almacenamiento físico o lógico disponible.
PersistentVolumeClaim (PVC)	Petición de almacenamiento por parte de un usuario. Se vincula con un PV.
StorageClass	Define políticas de aprovisionamiento dinámico (por tipo, rendimiento, etc.).

Red y exposición de servicios

Recurso	Función
Service	IP estable y punto de acceso para un conjunto de Pods.
ClusterIP	Servicio interno (por defecto).
NodePort	Expone el servicio en un puerto del nodo.
LoadBalancer	Usa un balanceador externo (en nubes o setups con MetalLB).
Ingress	Reglas HTTP(S) que enrutan tráfico externo hacia los Services.
Ingress Controller	Implementa las reglas de Ingress (NGINX, Traefik, etc.).
NetworkPolicy	Controla el tráfico permitido entre Pods o namespaces.

Configuración y seguridad

Recurso	Función
ConfigMap	Almacena configuración no sensible.

Recurso	Función
Secret	Datos sensibles (contraseñas, tokens).
ServiceAccount	Identidad usada por procesos dentro del clúster.
Role / ClusterRole	Define permisos.
RoleBinding / ClusterRoleBinding	Asigna permisos a usuarios o cuentas de servicio.
SecurityContext	Define restricciones de privilegios a nivel de Pod o contenedor.
PodSecurity (PSA/PSS)	Políticas globales de seguridad de Pods.

Observabilidad y control

Recurso / componente	Función
Events	Registro de sucesos del clúster.
Metrics Server	Proporciona métricas de CPU/RAM (para `kubectl top`).
Logs	Salida de contenedores y del sistema.
Probes (liveness/readiness/startup)	Verifican salud y disponibilidad de los Pods.
PodDisruptionBudget (PDB)	Define cuántos Pods pueden estar fuera de servicio simultáneamente.

> “ Referencias: Kubernetes Concepts (docs oficiales) · DevOpsCube - Kubernetes Architecture

---

Analogía para no romperse la cabeza

Control Plane → el **jefe de la obra**: da órdenes, no toca nada. **API Server** → el **secretario**: recibe tus órdenes y las reparte. **etcd** → el **cuaderno** donde se apunta todo. **Scheduler** → el **encargado** que reparte tareas entre los obreros. **Controller Manager** → el **pesado** que revisa que todo siga como estaba planeado.

Nodos de trabajo → los **obreros**.

- **kubelet** → el **capataz** del nodo.
- **kube-proxy** → el **tío del tráfico** que pone conos en la red.

- **Container Runtime** → la **máquina** que realmente enciende las cosas.

Pod → una **caja de herramientas**. **ReplicaSet** → el **contador** de cajas. **Deployment** → el **planificador** de cuándo y cómo se cambian. **StatefulSet** → el que lleva etiquetas con nombres. **DaemonSet** → el **repartidor** que deja una caja en cada nodo. **Job / CronJob** → el que hace algo una vez o a horas fijas.

PV / PVC / StorageClass → el **almacén**, el **ticket de pedido** y el **tipo de almacén**. PV = almacén físico. PVC = papelito que dice “quiero un trozo de almacén”. StorageClass = tipo de almacén (rápido, barato, lento…).

Service → el **interfono** para hablar con Pods. **ClusterIP** → solo dentro del edificio. **NodePort** → el portero abre una puerta concreta. **LoadBalancer** → el **recepcionista elegante**. **Ingress** → el **telefonista** que enruta llamadas desde fuera. **Ingress Controller** → quien ejecuta las reglas del telefonista.

ConfigMap → el **post-it** con configuraciones. **Secret** → el **post-it en una caja fuerte**. **RBAC / Roles** → el **carné de acceso**. **SecurityContext** → el **contrato** de cada trabajador.

Probes → los **médicos** que revisan si los Pods respiran. **PDB** → el **seguro** que impide despedir a todos a la vez. **Events / Logs / Metrics** → los **chismes del grupo de WhatsApp** donde se entera todo el mundo.