Kubernetes en producción: objetos, seguridad y operación efectiva

W. J. Carrillo Sandoval
Carnet: 7690-21-3740
wcarrillos1@miumg.edu.gt

Resumen
Kubernetes abstrae la infraestructura mediante un plano de control que programa contenedores y automatiza su ciclo de vida. Este artículo explica objetos clave (Pods, Deployments, Services, Ingress, ConfigMaps/Secrets), controles de seguridad (Namespaces, RBAC, NetworkPolicies, PodSecurity) y prácticas operativas (probes, HPA, PDB, CSI) para lograr resiliencia y observabilidad.
Palabras clave: kubernetes, contenedores, RBAC, autoscaling, CSI

1. Arquitectura de K8s

El **API Server** expone la interfaz; el **Scheduler** decide ubicaciones de Pods; el **Controller Manager** reconcilia estado; **etcd** persiste el estado del clúster. En los nodos, *kubelet* y el *runtime* ejecutan contenedores.

2. Objetos y ciclo de vida

- **Pod**: unidad mínima de ejecución.
- **Deployment**: réplicas, *rolling updates* y *rollbacks*.
- **Service**: acceso estable (ClusterIP, NodePort, LoadBalancer).
- **Ingress**: enrutamiento HTTP/HTTPS por host o path.
- **ConfigMap/Secret**: configuración y credenciales fuera de la imagen.

3. Resiliencia y escalamiento

Readiness/liveness probes aíslan pods enfermos; *PodDisruptionBudget* protege disponibilidad; *requests/limits* previenen sobrecompromiso; *HPA* escala por CPU/memoria o métricas personalizadas (p.ej., latencia P95); *cluster autoscaler* ajusta nodos.

4. Seguridad en el clúster

Namespaces para segmentación; RBAC de mínimo privilegio; *NetworkPolicies* para tráfico este-oeste; *PodSecurity*, escaneo y firma de imágenes; rotación de secretos.

5. Estado y almacenamiento

StatefulSets y volúmenes persistentes *CSI* con *snapshots* y políticas de *backup/restore*. Considerar *storage classes* y *topology-aware scheduling*.

6. Observabilidad

Prometheus + Alertmanager (métricas), logs centralizados y OpenTelemetry (trazas). SLOs y *error budgets* para gobernar canarios y *rollbacks*.

7. Observaciones y comentarios

Un *tenant model* claro y *namespaces* por equipo/entorno, con plantillas base (ingress, limits, policies), reduce variación y riesgo.

8. Conclusiones

1. Modelar bien los objetos simplifica despliegues y recuperaciones.
2. Seguridad efectiva combina RBAC, NetworkPolicies y gestión de imágenes.
3. Probes + HPA + PDB elevan disponibilidad y calidad de servicio.

Bibliografía

1. The Kubernetes Authors. *Kubernetes Documentation*. `https://kubernetes.io/docs/`
2. Burns, B., Beda, J., Hightower, K. (2022). *Kubernetes: Up & Running* (3a ed.). O'Reilly.
3. OpenTelemetry. *Documentation*. `https://opentelemetry.io/docs/`