DOCUMENTO ESPECIFICACIÓN TÉCNICA KUBERNETES – PROYECTO MIGRACIÓN SICOF

Fábrica de software – ADA S.A.S.

**REVISIÓN Y CONTROL DE CAMBIOS**

**Revisión y Versiones**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Autor | Versión | Descripción | | Cargo | Fecha |
| Jhonn Edison Gomez Melo | 1.0 | Documento | Desarrollador | | 16/08/2024 |

**Control de Cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Autor | Versión | Descripción del cambio |
| 16/08/2024 | Jhonn Edison Gomez Melo | 1.0 | Liberación Documento |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

TABLA DE CONTENIDO

Contenido

[**RESUMEN** 4](#_Toc174696997)

[**INTRODUCCIÓN** 5](#_Toc174696998)

[**INTRODUCCIÓN A LAS URLS O VARIABLES** 5](#_Toc174696999)

[**¿QUE ES Y PARA QUE SIRVE KUBERNETES?** 5](#_Toc174697000)

[**ARQUITECTURA DE KUBERNETES** 8](#_Toc174697001)

[**REDES Y SERVICIOS EN KUBERNETES** 9](#_Toc174697002)

[**ALMACENAMIENTO EN KUBERNETES** 10](#_Toc174697003)

[**SEGURIDAD EN KUBERNETES** 11](#_Toc174697004)

[**ESCALABILIDAD Y ALTA DISPONIBILIDAD** 12](#_Toc174697005)

[**OBSERVABILIDAD Y MONITOREO** 13](#_Toc174697006)

[**OPERADORES EN KUBERNETES** 13](#_Toc174697007)

[**CASOS DE USO AVANZADOS** 14](#_Toc174697008)

[**CONSIDERACIONES SOBRE EL RENDIMIENTO** 14](#_Toc174697009)

[**¿QUE ES Y PARA QUE SIRVE KUBECTL?** 15](#_Toc174697010)

[**INSTALACIÓN DE KUBECTL** 16](#_Toc174697011)

[**CONCLUSIONES** 17](#_Toc174697012)

# **RESUMEN**

Este documento técnico ofrece una visión integral de Kubernetes, una plataforma de orquestación de contenedores que se ha convertido en una herramienta esencial para el despliegue, la gestión y la escalabilidad de aplicaciones en entornos distribuidos.

A lo largo del contenido, se desglosan los componentes clave de la arquitectura de Kubernetes, se exploran las redes y servicios que facilitan la comunicación entre contenedores, y se detallan las opciones de almacenamiento disponibles.

Además, se abordan las prácticas de seguridad, las capacidades de escalabilidad y alta disponibilidad, y las herramientas de observabilidad y monitoreo que aseguran el rendimiento óptimo de las aplicaciones. También se presenta un análisis de los operadores en Kubernetes, junto con casos de uso avanzados y consideraciones sobre el rendimiento. Por último, se introduce kubectl, la herramienta de línea de comandos para interactuar con Kubernetes, describiendo su instalación y usos principales.

# **INTRODUCCIÓN**

En el mundo actual de la informática en la nube, Kubernetes se ha posicionado como un estándar para la gestión de contenedores, proporcionando a las organizaciones la capacidad de ejecutar aplicaciones a gran escala de manera eficiente.

Este documento tiene como objetivo ofrecer una guía completa sobre Kubernetes, dirigida a profesionales que buscan profundizar en su comprensión y aplicación. A través de una serie de secciones detalladas, se cubrirán los aspectos fundamentales y avanzados de Kubernetes, desde su arquitectura y funcionamiento básico hasta su integración con herramientas de observabilidad y prácticas de seguridad.

Además, se analizarán los operadores que permiten extender la funcionalidad de Kubernetes y se examinarán los desafíos y soluciones para mantener el rendimiento en clústeres grandes y complejos. Finalmente, se discutirá kubectl, la interfaz de línea de comandos que permite a los administradores interactuar con Kubernetes, proporcionando un enfoque práctico para su instalación y uso.

## **INTRODUCCIÓN A LAS URLS O VARIABLES**

En este documento, todas las URLs o variables utilizadas están almacenadas en la Wiki, la se encuentra en la siguiente ruta:

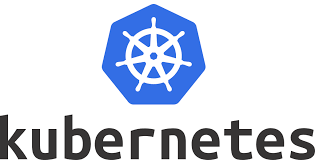
**Ruta de variables de Configuración:**

[**URL\_WIKI**/doku.php?id=ada:howto:sicoferp:factory:new-migracion-sicoferp:rutas\_de\_conexiones](http://10.1.20.89/doku.php?id=ada:howto:sicoferp:factory:new-migracion-sicoferp:rutas_de_conexiones)

A continuación, se listan las variables utilizadas en este documento para referirse a las URLs o variables, cabe resaltar que estas pueden cambiar dependiendo del ambiente en el que se está utilizando el enfoque:

* **URL\_SERVER\_DEV:** URL para el servidor de Desarrollo.
* **URL\_WIKI:** URL del repositorio Wiki

# **¿QUE ES Y PARA QUE SIRVE KUBERNETES?**



Kubernetes es una plataforma de orquestación de contenedores de código abierto que automatiza muchas de las tareas manuales involucradas en la implementación, gestión, escalado y operación de aplicaciones en contenedores.

Kubernetes, a menudo abreviado como K8s, fue desarrollado por Google y luego donado a la Cloud Native Computing Foundation (CNCF). Kubernetes proporciona una manera de gestionar aplicaciones en contenedores en un entorno distribuido.

Un contenedor es una unidad estándar de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de manera rápida y confiable en diferentes entornos informáticos. Kubernetes es responsable de coordinar estas unidades, asegurando que funcionen correctamente y estén disponibles para los usuarios.

**PRINCIPALES COMPONENTES DE KUBERNETES**

* **Nodo (Node)**: Un nodo es una máquina en el clúster de Kubernetes, que puede ser una máquina física o una máquina virtual. Cada nodo se ejecuta en al menos un contenedor y se gestiona por el plano de control de Kubernetes.
* **Pod**: La unidad básica de Kubernetes. Un pod es un grupo de uno o más contenedores que comparten almacenamiento y red, y se especifican cómo deben ejecutarse. Los contenedores dentro de un pod están destinados a ser gestionados como una sola entidad.
* **ReplicaSet**: Asegura que un número especificado de réplicas de un pod se están ejecutando en todo momento.
* **Deployment**: Proporciona actualizaciones declarativas para pods y ReplicaSets. Un Deployment declara el estado deseado para tu aplicación y Kubernetes se encarga de hacer los cambios necesarios para alcanzar ese estado.
* **Service**: Abstrae un conjunto lógico de pods y una política para acceder a ellos. Kubernetes proporciona descubrimiento de servicios y balanceo de carga.
* **ConfigMap y Secret**: ConfigMap es una API de Kubernetes que permite separar la configuración del contenedor de la imagen del contenedor. Secret es similar a ConfigMap, pero se utiliza para almacenar información sensible.
* **Namespace**: Proporciona un mecanismo para dividir recursos en un clúster en varios espacios con nombres (namespaces).

**¿PARA QUÉ SIRVE KUBERNETES?**

* **Orquestación de Contenedores**: Gestiona el ciclo de vida de contenedores, desplegándolos y asegurando que estén siempre en el estado deseado.
* **Escalabilidad**: Escala automáticamente las aplicaciones en función de la demanda. Puede agregar o quitar instancias de contenedores (pods) en función de las necesidades.
* **Alta Disponibilidad**: Kubernetes monitorea la salud de los contenedores y nodos. Si un contenedor o nodo falla, Kubernetes puede reiniciar o reemplazar el contenedor o redistribuir la carga a otros nodos disponibles.
* **Implementación y Rollback Automatizados**: Facilita las implementaciones y actualizaciones continuas de aplicaciones, permitiendo deshacer cambios en caso de problemas (rollbacks).
* **Balanceo de Carga y Distribución de Tráfico**: Kubernetes puede distribuir automáticamente el tráfico de red entrante a los pods adecuados, asegurando una distribución equitativa y eficiente del tráfico.
* **Gestión de Configuración y Secrets**: Administra la configuración de la aplicación y los secretos de manera segura y eficiente.
* **Planificación y Optimización de Recursos**: Kubernetes optimiza el uso de los recursos del clúster (CPU, memoria, almacenamiento) asegurando que las aplicaciones se ejecuten de manera eficiente.
* **Recuperación ante Fallos**: Detecta y mitiga fallos en los contenedores y nodos, reiniciando contenedores fallidos y redistribuyendo cargas de trabajo automáticamente.

**BENEFICIOS DE USAR KUBERNETES**

* **Portabilidad**: Permite ejecutar aplicaciones de manera consistente en diferentes entornos (local, desarrollo, producción, nubes públicas, nubes privadas).
* **Flexibilidad y Agilidad**: Facilita el desarrollo ágil y las prácticas de DevOps, mejorando la eficiencia en el ciclo de desarrollo de software.
* **Ahorro de Costos**: Optimiza el uso de recursos, lo que puede reducir los costos operativos en entornos de nube y en infraestructuras locales.

Es decir, Kubernetes es una poderosa herramienta para gestionar aplicaciones en contenedores de manera escalable, eficiente y automatizada, permitiendo a las organizaciones enfocarse más en el desarrollo de software y menos en la gestión de infraestructura.

## **ARQUITECTURA DE KUBERNETES**

**Plano de Control (Control Plane)**

El plano de control es la parte central de un clúster de Kubernetes. Es responsable de gestionar el estado del clúster, desde programar los pods en los nodos hasta mantener la configuración deseada del sistema.

* **kube-apiserver:** Es el punto de entrada para todas las operaciones administrativas en el clúster. Maneja las solicitudes REST y actualiza el estado del clúster en etcd, que es la base de datos de almacenamiento clave-valor de Kubernetes.
  + **Ejemplo:** Cuando un usuario crea un deployment usando kubectl, esta solicitud es manejada por el kube-apiserver.
* **etcd:** Es un almacén de datos distribuido y altamente disponible que Kubernetes usa para almacenar todos los datos del clúster.
  + **Ejemplo:** Almacenamiento de configuraciones de despliegue, secretos, y detalles del estado de los pods y nodos.
* **kube-scheduler:** Este componente se encarga de asignar pods no programados a nodos disponibles en el clúster.
  + **Ejemplo:** Si hay un nuevo pod que necesita ser ejecutado, el kube-scheduler decide en qué nodo se debe ejecutar, basándose en factores como el uso de recursos.
* **kube-controller-manager:** Ejecuta los controladores de Kubernetes, como el controlador de replicación, que asegura que el número deseado de pods esté siempre en ejecución.
  + **Ejemplo:** Si un pod falla, el controlador de replicación lo reemplaza automáticamente.

**Nodos y Agentes**

Los nodos son las máquinas (físicas o virtuales) en las que se ejecutan los contenedores gestionados por Kubernetes.

* **kubelet:** Es el agente que se ejecuta en cada nodo. Se asegura de que los contenedores descritos en los Pods estén funcionando correctamente.
  + **Ejemplo:** Si se describe un pod con un contenedor de Nginx, el kubelet en el nodo correspondiente inicia y supervisa ese contenedor.
* **kube-proxy:** Gestiona las reglas de red en los nodos, permitiendo la comunicación entre los diferentes servicios en un clúster.
  + **Ejemplo:** Permite que un servicio de backend se comunique con el frontend a través de las IPs internas del clúster.

## **REDES Y SERVICIOS EN KUBERNETES**



**Modelo de Redes**

Kubernetes asume un modelo de red plano en el que todos los pods pueden comunicarse entre sí, independientemente del nodo en el que se encuentren.

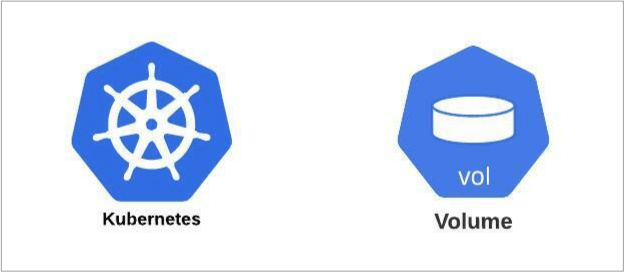
* **Overlay Networks:** Tecnologías como *Flannel* o *Calico* se utilizan para crear una red de superposición que conecta los pods a través de una red virtual.
  + **Ejemplo:** Calico puede usar IPs BGP (Border Gateway Protocol) para enrutar el tráfico entre nodos, proporcionando un mayor control sobre las políticas de red.

**Servicios y Endpoints**

Un servicio en Kubernetes es una abstracción que define un conjunto lógico de pods y una política para acceder a ellos.

* **ClusterIP:** Proporciona un IP interna para el servicio, accesible solo dentro del clúster.
  + **Ejemplo:** Un servicio backend podría estar expuesto a través de un ClusterIP, permitiendo que otros servicios internos se conecten a él.
* **NodePort:** Expone un servicio en un puerto específico en todos los nodos del clúster, permitiendo el acceso externo al clúster.
  + **Ejemplo:** Exponer un servicio de base de datos a través de un puerto específico para ser accedido desde fuera del clúster.

## **ALMACENAMIENTO EN KUBERNETES**



**Volumes y Persistent Volumes**

Kubernetes gestiona el almacenamiento mediante volúmenes que pueden ser utilizados por los contenedores para persistir datos.

* **Persistent Volume (PV):** Un recurso de almacenamiento en el clúster que puede ser reclamado por un pod.
  + **Ejemplo:** Un PV puede estar respaldado por un NFS (Network File System) o un almacenamiento en la nube como EBS en AWS.
* **Persistent Volume Claim (PVC):** Una solicitud para un PV que es vinculada a un pod.
  + **Ejemplo:** Un pod que necesita almacenar datos en un disco persistente puede usar un PVC para solicitar un PV con el tamaño requerido.

**Storage Classes**

Las Storage Classes permiten la provisión dinámica de volúmenes con diferentes características, como rendimiento o tipo de almacenamiento.

* **Ejemplo:** Se puede definir una Storage Class para SSDs de alto rendimiento en la nube y otra para almacenamiento en disco magnético más económico. Los pods pueden entonces solicitar volúmenes específicos basados en estas clases.

## **SEGURIDAD EN KUBERNETES**

**Autenticación y Autorización**

Kubernetes implementa mecanismos de seguridad robustos para controlar quién puede acceder al clúster y qué operaciones pueden realizar.

* **RBAC (Role-Based Access Control):** Permite la asignación de permisos a usuarios y grupos basados en roles.
  + **Ejemplo:** Un usuario con un rol de administrador puede tener acceso completo al clúster, mientras que un usuario con un rol de desarrollador puede estar limitado a ciertos espacios de nombres (namespaces).

**Políticas de Seguridad de Pods (Pod Security Policies)**

Permiten definir qué características de seguridad deben aplicarse a los pods.

* **Ejemplo:** Se puede crear una política que prohíba a los pods correr como root, o que requiera el uso de espacios de nombres de usuario para aislamiento adicional.

## **ESCALABILIDAD Y ALTA DISPONIBILIDAD**

**Horizontal Pod Autoscaler (HPA)**

Kubernetes puede escalar automáticamente el número de réplicas de un pod basado en métricas observadas como el uso de CPU.

* **Ejemplo:** Si una aplicación web experimenta un aumento en el tráfico, HPA puede aumentar automáticamente el número de réplicas para manejar la carga adicional.

**Cluster Autoscaler**

Automáticamente ajusta el tamaño del clúster agregando o eliminando nodos en función de la demanda de recursos.

* **Ejemplo:** Si la carga de trabajo del clúster disminuye durante la noche, el Cluster Autoscaler puede reducir el número de nodos para ahorrar costos.

## **OBSERVABILIDAD Y MONITOREO**



**Prometheus y Grafana**

Prometheus es una herramienta de monitoreo que recoge métricas del clúster, mientras que Grafana proporciona dashboards visuales.

* **Ejemplo:** Se puede usar Prometheus para monitorear el uso de recursos de los pods y nodos, y Grafana para visualizar estas métricas en tiempo real, ayudando a identificar problemas de rendimiento.

**Logging con Elasticsearch, Fluentd, y Kibana (EFK)**

El stack EFK proporciona una solución completa para el manejo de logs en Kubernetes.

* **Ejemplo:** Fluentd recopila los logs de los contenedores, Elasticsearch los almacena, y Kibana permite su análisis y visualización.

## **OPERADORES EN KUBERNETES**

**Definición de Operadores**

Los operadores son extensiones de Kubernetes que permiten gestionar aplicaciones complejas a través de la API de Kubernetes.

* **Ejemplo:** El *MySQL Operator* puede manejar la instalación, configuración, y escalabilidad de una base de datos MySQL dentro del clúster, incluyendo la realización de copias de seguridad y recuperación.

**Ejemplos de Operadores**

Algunos operadores populares incluyen:

* **Prometheus Operator:** Simplifica el despliegue y la gestión de Prometheus para el monitoreo en un clúster de Kubernetes.
* **Cert-Manager Operator:** Automáticamente gestiona certificados TLS para aplicaciones en Kubernetes.

## **CASOS DE USO AVANZADOS**

**Despliegue de Microservicios**

Kubernetes facilita la gestión de aplicaciones compuestas por microservicios, donde cada servicio puede ser desplegado, escalado y actualizado independientemente.

* **Ejemplo:** Una aplicación web puede estar compuesta por un frontend, un backend, y un servicio de base de datos, cada uno de los cuales se gestiona como un microservicio independiente en el clúster.

**Integración Continua y Despliegue Continuo (CI/CD)**

Kubernetes puede integrarse con herramientas CI/CD para automatizar el proceso de construcción, prueba y despliegue de aplicaciones.

* **Ejemplo:** Jenkins puede ser configurado para construir imágenes Docker, realizar pruebas, y desplegar estas imágenes en un clúster de Kubernetes, todo como parte de un pipeline automatizado.

## **CONSIDERACIONES SOBRE EL RENDIMIENTO**

**Optimización de Recursos**

Para maximizar la eficiencia en Kubernetes, es crucial ajustar correctamente los recursos de los pods.

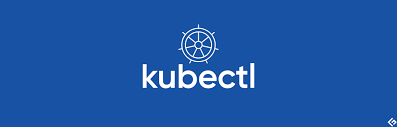
* **Ejemplo:** Configurar requests y limits para CPU y memoria en los pods asegura que cada contenedor tenga los recursos necesarios sin desperdiciar capacidad, lo que mejora la eficiencia del clúster.

**Mejora de la Latencia de Red**

Reducir la latencia en la red del clúster puede ser vital para aplicaciones sensibles al tiempo.

* **Ejemplo:** Implementar un *Service Mesh* como Istio para gestionar la comunicación entre microservicios y optimizar el enrutamiento del tráfico, reduciendo así la latencia.

# **¿QUE ES Y PARA QUE SIRVE KUBECTL?**



kubectl es una herramienta de línea de comandos para interactuar con Kubernetes, una plataforma de orquestación de contenedores. Kubernetes se encarga de automatizar la implementación, el escalado y las operaciones de aplicaciones en contenedores.

**Funciones y Uso de kubectl**

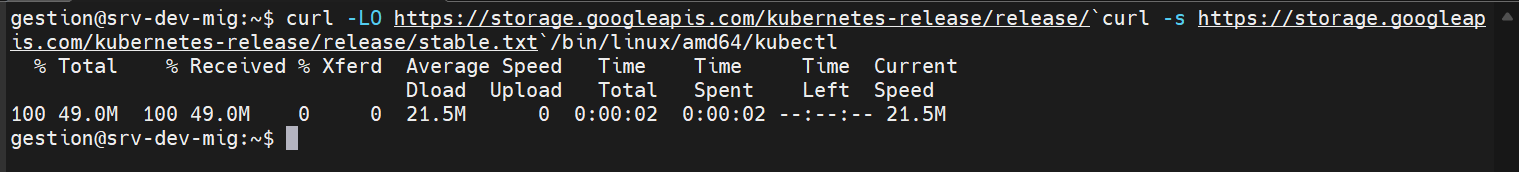
* **Gestión de Clústeres Kubernetes**: kubectl permite a los usuarios interactuar con los clústeres Kubernetes para administrar aplicaciones, despliegues y recursos. Con kubectl, puedes realizar acciones como crear, actualizar y eliminar aplicaciones en el clúster.
* **Implementación de Aplicaciones**: Se puede usar kubectl para implementar nuevas aplicaciones en un clúster Kubernetes. Esto incluye la creación de Pods, Deployments, Services y otros recursos necesarios para ejecutar aplicaciones.
* **Visualización y Monitorización:** kubectl proporciona comandos para ver el estado de los recursos dentro de un clúster. Puedes verificar el estado de los Pods, los logs de los contenedores y el estado general del clúster.
* **Escalado de Aplicaciones:** Se puede escalar las aplicaciones hacia arriba o hacia abajo, ajustando el número de réplicas de un Deployment.
* **Actualización y Rolling Updates:** kubectl permite actualizar aplicaciones sin tiempo de inactividad a través de actualizaciones continuas.
* **Gestión de Configuraciones y Secretos:** Permite gestionar configuraciones y secretos utilizados por las aplicaciones para almacenar configuraciones sensibles como contraseñas y claves API.
* **Acceso a la Consola de los Contenedores:** Facilita el obtener acceso a la consola de un contenedor en ejecución para realizar tareas de diagnóstico o depuración.

Es decir, kubectl es una herramienta esencial para gestionar y operar aplicaciones en un clúster Kubernetes. Facilita la administración de recursos, la implementación de aplicaciones y la realización de tareas de mantenimiento y diagnóstico.

### **INSTALACIÓN DE KUBECTL**

Para realizar la instalación de kubectl en el servidor de desarrollo(**URL\_SERVER\_DEV**), la instalación se realiza de la siguiente manera:

* **Desacargar kubectl en el servidor:** Se ejecuta el siguiente comando para descargar kubectl



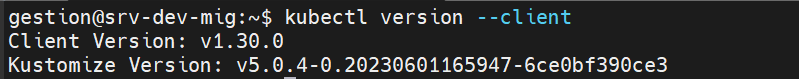
* **Dar permisos de ejecución:** Se dan permisos de ejecución con el siguiente comando



* **Mover el archivo a los binarios del sistema:** Se mueve el archivo instalado a los binarios del sistema con el siguiente comando



* **Validar la instalación de kubectl:** Se realiza la validación de la instalación de kubectl



# **CONCLUSIONES**

Kubernetes ha emergido como la solución líder en la orquestación de contenedores, proporcionando un conjunto robusto de herramientas y funcionalidades que permiten a las organizaciones desplegar, gestionar y escalar aplicaciones con una eficiencia sin precedentes. A lo largo de este documento, hemos explorado en profundidad los diversos componentes y características que hacen de Kubernetes una plataforma indispensable en el panorama tecnológico actual.

La **arquitectura de Kubernetes** destaca por su diseño modular, donde el plano de control y los nodos trabajan en conjunto para mantener el estado deseado del sistema. Esta estructura facilita la gestión y supervisión de los recursos, garantizando que las aplicaciones se ejecuten de manera óptima y resiliente.

En cuanto a las **redes y servicios**, Kubernetes ofrece mecanismos sofisticados para asegurar una comunicación fluida y segura entre los distintos componentes de una aplicación distribuida. La abstracción de servicios y el manejo eficiente de las redes internas permiten una escalabilidad horizontal que se adapta dinámicamente a las necesidades de carga.

El manejo del **almacenamiento** en Kubernetes, a través de volúmenes persistentes y clases de almacenamiento, asegura que los datos de las aplicaciones se gestionen de forma segura y persistente, independientemente de la naturaleza efímera de los contenedores. Esto es esencial para aplicaciones que requieren una gestión cuidadosa de datos, como bases de datos y sistemas de archivos compartidos.

La **seguridad** se aborda de manera integral, ofreciendo múltiples capas de protección que incluyen autenticación, autorización y políticas de seguridad para pods. Estas características aseguran que solo el personal y los procesos autorizados puedan interactuar con los recursos críticos del clúster.

Las capacidades de **escalabilidad y alta disponibilidad** de Kubernetes permiten que las aplicaciones manejen cargas variables y aseguran la continuidad del servicio incluso en escenarios de falla. Herramientas como el escalado automático de pods y la administración eficiente de nodos garantizan que los sistemas puedan adaptarse rápidamente a las demandas cambiantes.

La **observabilidad y monitoreo** son fundamentales para mantener la salud y el rendimiento del clúster. Integraciones con herramientas como Prometheus y Grafana ofrecen una visibilidad completa del estado del sistema, permitiendo una detección y resolución proactiva de problemas.

Los **operadores** en Kubernetes amplían aún más las capacidades de la plataforma, permitiendo la automatización de tareas complejas y la gestión de aplicaciones de estado de manera declarativa. Esta flexibilidad facilita la implementación de soluciones personalizadas que se alinean con las necesidades específicas de cada organización.

Además, los **casos de uso avanzados** demuestran la versatilidad de Kubernetes para abordar desafíos modernos en el desarrollo y despliegue de aplicaciones, desde la implementación de microservicios hasta la integración y el despliegue continuos (CI/CD).

Las **consideraciones sobre el rendimiento** resaltan la importancia de una planificación y configuración cuidadosas para maximizar la eficiencia y la eficacia de los recursos disponibles, asegurando que las aplicaciones ofrezcan un rendimiento óptimo en todo momento.

Finalmente, la herramienta de línea de comandos **kubectl** se presenta como una interfaz esencial para interactuar con el clúster de Kubernetes, ofreciendo comandos intuitivos y potentes que facilitan la gestión diaria de las aplicaciones y los recursos.

En conjunto, Kubernetes representa una revolución en la manera en que las aplicaciones modernas se desarrollan, despliegan y gestionan. Su adopción permite a las organizaciones alcanzar niveles superiores de agilidad, resiliencia y eficiencia, posicionándolas favorablemente en un entorno tecnológico en constante evolución. A medida que la comunidad continúa innovando y expandiendo las capacidades de Kubernetes, es esencial que las organizaciones se mantengan informadas y adopten prácticas recomendadas para aprovechar al máximo esta poderosa plataforma.