

# ESTRATEGIAS PARA ESCALAR APLICACIONES WEB

LENGUAJES INTERPRETADOS EN EL CLIENTE

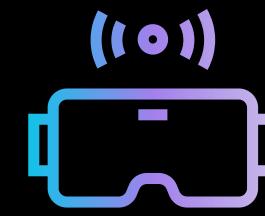


LINK DE  
GITHUB

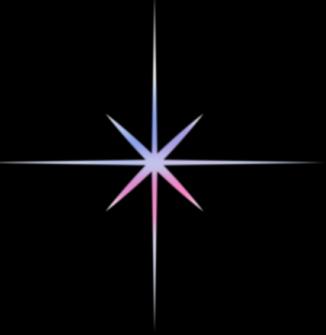


kubernetes



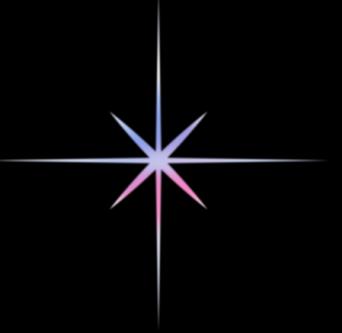


# Estrategias de Escalabilidad



Las aplicaciones web desarrolladas con Python, PHP o Node.js, requieren estrategias eficientes para garantizar rendimiento, disponibilidad y estabilidad a medida que aumenta la demanda de usuarios. Entre las principales estrategias de escalabilidad encontramos:





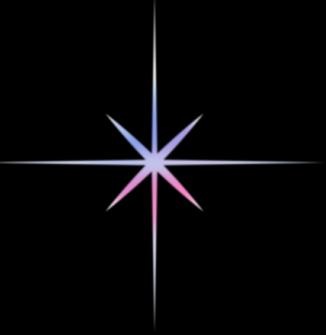
# ¿Qué es Horizontalización?

La horizontalización consiste en replicar instancias de una aplicación en múltiples servidores para distribuir la carga y mejorar la disponibilidad.

## Beneficios:

- Mayor tolerancia a fallos.
- Mejor rendimiento en aplicaciones con alto tráfico.
- Permite escalabilidad dinámica según la demanda.

# ¿Qué es Balanceo de Carga?



El balanceo de carga distribuye el tráfico entre múltiples instancias de una aplicación, evitando la sobrecarga de un solo servidor.

Técnicas comunes:

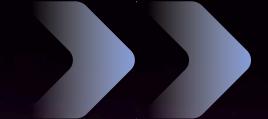
- Round Robin: Distribuye las solicitudes equitativamente entre servidores.
- Least Connections: Envía las solicitudes al servidor menos ocupado.
- IP Hashing: Asigna un usuario a un servidor específico según su IP.

Beneficios:

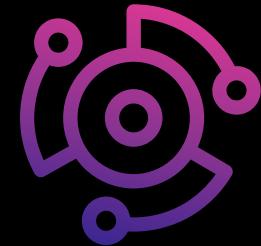
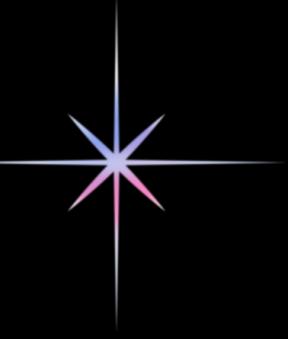
Asegura una distribución equitativa de la carga.

Reduce el tiempo de respuesta y mejora la experiencia del usuario.

Aumenta la disponibilidad y confiabilidad del sistema.



# Replicación y uso de Balanceo de Carga para mejorar el rendimiento



La replicación de instancias consiste en tener múltiples copias de una aplicación en diferentes servidores para distribuir las cargas y mejorar la disponibilidad. Los平衡adores de carga garantizan que las solicitudes del usuario se distribuyan equitativamente entre las instancias disponibles.

- Mejorando la Disponibilidad
- Mejorando el Rendimiento



# ¿Qué es Docker?

## Docker

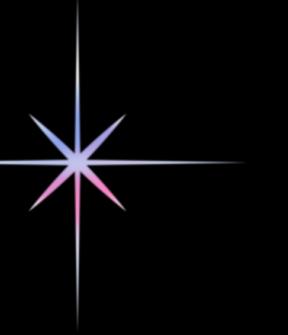
permite empaquetar aplicaciones y sus dependencias en contenedores ligeros y portables, facilitando la implementación en cualquier entorno.

## Impacto en la escalabilidad:

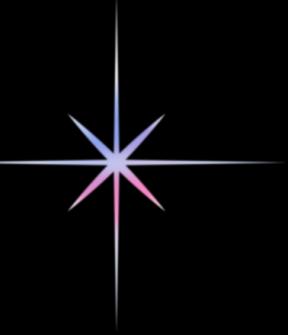
- Aislamiento: Cada contenedor opera de manera independiente, evitando conflictos entre aplicaciones.
- Portabilidad: Los contenedores pueden ejecutarse en cualquier sistema operativo con Docker.
- Eficiencia: Utilizan menos recursos que las máquinas virtuales.
- Escalabilidad: Permiten desplegar rápidamente nuevas instancias de una aplicación.

# Kubernetes

Kubernetes es una plataforma de orquestación de contenedores que ha ganado una gran adopción debido a su capacidad para gestionar el ciclo de vida completo de las aplicaciones en contenedores, incluyendo el despliegue, el escalado, la distribución de la carga y la gestión de fallos. Kubernetes es una herramienta extremadamente poderosa para gestionar aplicaciones en producción a gran escala, especialmente cuando se utilizan contenedores, como los creados con Docker. A continuación, se analizan los aspectos clave de Kubernetes que permiten una gestión eficiente del escalado.

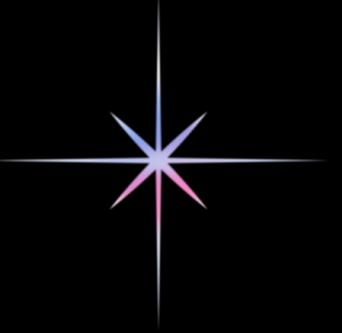


# Automatización del despliegue



Kubernetes facilita la implementación de aplicaciones mediante Deployments, que permiten definir cómo deben ejecutarse los contenedores.

- Se integra con herramientas de CI/CD, permitiendo que las actualizaciones ocurran automáticamente.
- Usa rolling updates para actualizar sin interrupciones y rollbacks para volver a una versión anterior si hay fallos.

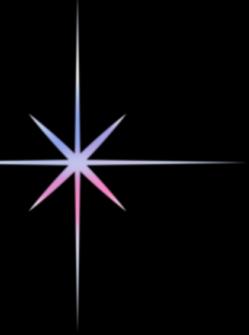


# Escalado Automático

Kubernetes ajusta los recursos de la aplicación según la demanda.

- Escalado Horizontal (HPA): Aumenta o reduce el número de contenedores según el tráfico.
- Escalado Vertical (VPA): Ajusta los recursos de CPU y memoria en cada contenedor.
- Escalado de Nodos: Agrega o elimina servidores del clúster para optimizar el uso de infraestructura.

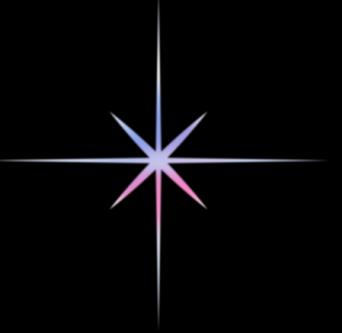
# Gestión de Fallos



Si un contenedor o servidor falla, Kubernetes lo detecta y lo soluciona automáticamente.

- Auto-reparación: Si un pod deja de funcionar, Kubernetes lo reinicia.
- Redistribución de Pods: Si un nodo falla, Kubernetes mueve los contenedores a otro nodo disponible.
- Monitoreo de salud: Usa sondas para comprobar si un contenedor está funcionando correctamente.





# Balanceo de Carga

Kubernetes distribuye el tráfico entre los diferentes contenedores para evitar sobrecargas.

- Servicios de Kubernetes: Garantizan que las solicitudes se dirijan a los contenedores disponibles.
- Integración con balanceadores externos: Compatible con herramientas como AWS ELB y Google Cloud Load Balancer.



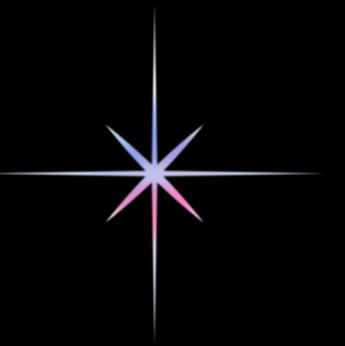
# Actualizaciones continuas y rollbacks



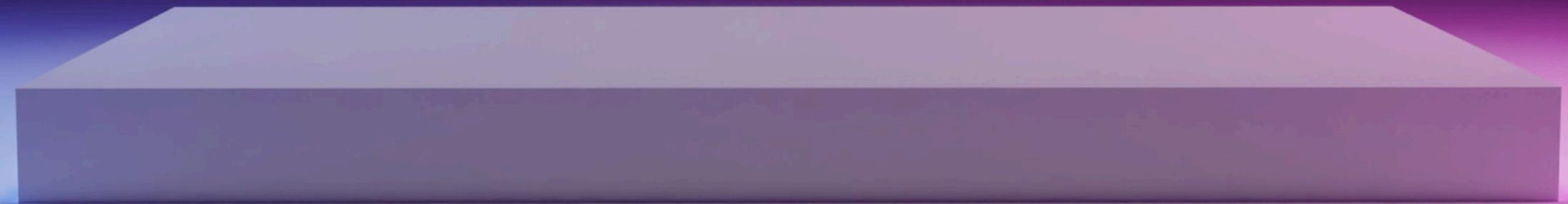
Kubernetes permite actualizar aplicaciones sin afectar su disponibilidad.

- Rolling updates: Actualiza progresivamente sin interrupciones.
- Rollback automático: Si una nueva versión falla, Kubernetes vuelve a la versión estable anterior.

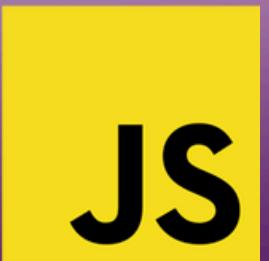


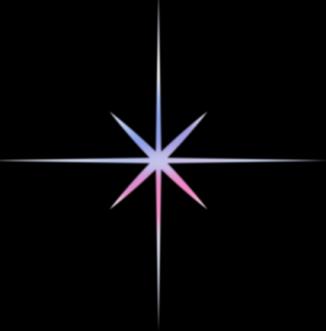


# ¡GRACIAS POR SU ATENCIÓN!



kubernetes





- **NEMESIS ALEJANDRA VALENCIA RIVERA VR211067**
- **JOSE LUIS QUINTANILLA LOPEZ QL210503**
- **GEOVANY ARTURO PINEDA FUENTES PF211251**
- **ALEXANDRA GUADALUPE PADILLA RAMÍREZ NR221019**
- **LUIS ANTONIO MENDEZ PARADA MP220885**



**kubernetes**

