Guía Práctica de Kubernetes para Estudiantes con Docker y Flask en Windows

Sección 1: Introducción a Kubernetes

1.1 ¿Qué es Kubernetes?

Kubernetes, a menudo abreviado como K8s (contando las ocho letras entre la 'K' y la 's'), es una plataforma de código abierto, portátil y extensible diseñada para automatizar la implementación, el escalado y la gestión de aplicaciones en contenedores.[1] Originario de Google, que lo liberó como código abierto en 2014, Kubernetes combina más de 15 años de experiencia de Google en la ejecución de cargas de trabajo de producción a escala con las mejores ideas y prácticas de la comunidad.[2] Su nombre proviene del griego y significa "timonel" o "piloto", reflejando su función de dirigir y gestionar contenedores.[2]
El propósito fundamental de Kubernetes es proporcionar un marco robusto para ejecutar sistemas distribuidos de manera resiliente.[2] Facilita tanto la configuración declarativa como la automatización, permitiendo a los desarrolladores y administradores de sistemas gestionar aplicaciones empaquetadas en contenedores de forma eficiente y fiable.[2] Kubernetes se ha convertido en el estándar de facto para la orquestación de contenedores, respaldado por un ecosistema grande y en rápido crecimiento, con servicios, soporte y herramientas ampliamente disponibles.[2]

1.2 ¿Por qué Kubernetes? (Comparación con Docker)

Si bien Docker es una herramienta excelente para crear y ejecutar contenedores individuales, proporcionando un entorno estandarizado para las aplicaciones [4], Kubernetes aborda los desafíos que surgen al gestionar estas aplicaciones en contenedores a escala en entornos de producción.[2] Docker, por sí solo, se centra en el ciclo de vida de un contenedor en una única máquina.[5] Sin embargo, las aplicaciones modernas, a menudo compuestas por microservicios distribuidos, requieren una gestión más sofisticada que abarque múltiples contenedores y múltiples máquinas.[3]
Aquí es donde Kubernetes demuestra su utilidad superior en comparación con el uso exclusivo de Docker:

- **Orquestación:** Kubernetes automatiza la coordinación de contenedores a través de un clúster de máquinas.[1] Gestiona el despliegue, el escalado (automático o manual) y el ciclo de vida completo de las aplicaciones contenedorizadas.[2] Si un contenedor falla, Kubernetes puede reiniciarlo automáticamente.[2] Docker Swarm, la herramienta de

orquestación nativa de Docker, ofrece funcionalidades similares pero es generalmente considerada menos robusta y flexible que Kubernetes para despliegues complejos y a gran escala.[4]

- **Alta Disponibilidad y Auto-reparación:** Kubernetes está diseñado para la resiliencia.[2] Puede detectar y reemplazar automáticamente contenedores o nodos que fallen, redistribuir el trabajo a nodos saludables y asegurar que la aplicación siga funcionando incluso si partes de la infraestructura fallan.[2] Docker por sí solo no ofrece estas capacidades de auto-reparación distribuida.[2]
- **Escalabilidad:** Kubernetes permite escalar aplicaciones horizontalmente (añadiendo o quitando instancias de contenedores/pods) de forma automática o manual, basándose en la demanda o métricas como el uso de CPU.[2] Esto asegura que la aplicación pueda manejar cargas variables de manera eficiente.[7]
- **Descubrimiento de Servicios y Balanceo de Carga:** Kubernetes proporciona mecanismos integrados para que los contenedores se descubran entre sí y expone los contenedores a través de un nombre DNS o una dirección IP estable. También puede balancear la carga de red entre múltiples instancias de un contenedor.[2] Docker ofrece capacidades de red básicas, pero el descubrimiento y balanceo en entornos multi-host es más complejo sin una herramienta de orquestación.[2]
- **Despliegues y Rollbacks Automatizados:** Kubernetes facilita la realización de actualizaciones de aplicaciones (rollouts) y la reversión a versiones anteriores (rollbacks) de forma controlada y automatizada, a menudo sin tiempo de inactividad.[2] Permite definir el estado deseado y Kubernetes trabaja para alcanzarlo gradualmente.[2]
- **Gestión de Almacenamiento, Secretos y Configuración:** Kubernetes permite orquestar el almacenamiento, montando automáticamente sistemas de almacenamiento persistente (locales, en red o en la nube) para los contenedores.[2] También ofrece objetos dedicados (Secrets y ConfigMaps) para gestionar información sensible y configuraciones de aplicación de forma segura y desacoplada de las imágenes de contenedor.[2]

En resumen, Docker y Kubernetes no son competidores directos, sino tecnologías complementarias.[4] Docker se centra en la creación y ejecución de contenedores, mientras que Kubernetes se centra en la gestión y orquestación de esos contenedores a escala en un entorno distribuido.[3] Para aplicaciones complejas y entornos de producción, Kubernetes proporciona las capacidades de automatización, resiliencia y escalabilidad que Docker por sí solo no ofrece.[2]

1.3 Conceptos Fundamentales de Kubernetes

Para entender cómo funciona Kubernetes, es esencial familiarizarse con sus componentes y abstracciones principales [10]:

- **Cluster:** Un clúster de Kubernetes es el conjunto completo de máquinas (nodos) que ejecutan las aplicaciones contenedorizadas gestionadas por Kubernetes.[1] Consiste en al menos un nodo de control (control plane) y uno o más nodos trabajadores.

- **Nodo (Node):** Un nodo es una máquina trabajadora en Kubernetes, que puede ser una máquina virtual (VM) o física.[1] Cada nodo ejecuta los componentes necesarios para correr Pods y es gestionado por el plano de control. Los componentes clave en un nodo incluyen el kubelet (agente que se comunica con el plano de control y gestiona los contenedores en el nodo), kube-proxy (mantiene las reglas de red en los nodos) y un runtime de contenedores (como Docker, containerd, etc.) que se encarga de ejecutar los contenedores.[1]
- **Plano de Control (Control Plane):** Es el "cerebro" del clúster. Proporciona los servicios centrales de Kubernetes y la orquestación de las cargas de trabajo.[1] Sus componentes principales, que pueden ejecutarse en uno o varios nodos (nodos maestros), incluyen [1]:
    - kube-apiserver: Expone la API de Kubernetes. Es el frontend del plano de control.
    - etcd: Almacén clave-valor distribuido y consistente que guarda toda la configuración y el estado del clúster.
    - kube-scheduler: Observa los Pods recién creados sin nodo asignado y selecciona un nodo para que se ejecuten en él.
    - kube-controller-manager: Ejecuta procesos controladores que regulan el estado del clúster (por ejemplo, detectar nodos caídos, mantener el número correcto de réplicas).
    - cloud-controller-manager (opcional): Integra con la infraestructura de nube subyacente (si aplica).
- **Pod:** Es la unidad de computación más pequeña y básica que se puede crear y gestionar en Kubernetes.[2] Un Pod representa una única instancia de una aplicación en ejecución y puede contener uno o más contenedores que están estrechamente acoplados.[2] Los contenedores dentro de un mismo Pod comparten el mismo contexto de red (dirección IP, espacio de puertos) y pueden compartir volúmenes de almacenamiento.[10] Los Pods son efímeros por naturaleza; si un Pod falla, Kubernetes puede crear uno nuevo para reemplazarlo (normalmente gestionado por un Deployment).[10]
- **Servicio (Service):** Un Servicio es una abstracción que define un conjunto lógico de Pods y una política para acceder a ellos.[2] Proporciona una dirección IP estable y un nombre DNS único para un conjunto de Pods, permitiendo que otros componentes (dentro o fuera del clúster) se comuniquen con la aplicación sin necesidad de conocer las direcciones IP individuales y cambiantes de los Pods.[2] Los Servicios utilizan etiquetas (labels) y selectores (selectors) para determinar a qué Pods dirigir el tráfico.[14] Existen diferentes tipos de Servicios (ClusterIP, NodePort, LoadBalancer, ExternalName) para diferentes escenarios de exposición.[14]
- **Deployment:** Un Deployment es un objeto de nivel superior que gestiona el despliegue y escalado de un conjunto de Pods idénticos a través de ReplicaSets.[2] Permite describir el estado deseado de la aplicación (por ejemplo, qué imagen de contenedor usar y cuántas réplicas ejecutar) de forma declarativa.[2] El controlador de Deployment se encarga de crear los Pods necesarios y de gestionar las actualizaciones (por ejemplo,

rolling updates) y rollbacks a versiones anteriores de manera controlada.[2] Es la forma más común de desplegar aplicaciones sin estado en Kubernetes.[10]

Comprender estos conceptos es fundamental para empezar a trabajar con Kubernetes y desplegar aplicaciones de manera efectiva.

Sección 2: Configuración de un Entorno Local de Kubernetes en Windows

Para experimentar y aprender Kubernetes, no es necesario un clúster de producción completo. Se puede configurar un entorno local en una máquina con Windows. Existen varias herramientas para lograr esto, siendo las más populares Minikube y la funcionalidad integrada en Docker Desktop.

2.1 Prerrequisitos

Antes de instalar un entorno local de Kubernetes, es necesario asegurarse de que el sistema Windows cumple ciertos requisitos mínimos y tiene un hipervisor compatible instalado:

- **Recursos del Sistema:**
  - **CPU:** Se recomiendan 2 CPUs o más.[18]
  - **RAM:** Se necesita un mínimo de 2 GB de memoria libre.[18] Algunas configuraciones o herramientas pueden requerir más (hasta 8GB o más).[19]
  - **Espacio en Disco:** Se requieren al menos 20 GB de espacio libre en disco.[18]
- **Soporte de Virtualización:** La virtualización debe estar habilitada en la BIOS/UEFI del sistema. Se puede verificar ejecutando systeminfo en el símbolo del sistema y buscando los requisitos de Hyper-V.[21]
- **Hipervisor:** Se necesita un hipervisor o un gestor de contenedores/VM. Las opciones comunes para Windows son [21]:
  - **Hyper-V:** El hipervisor nativo de Microsoft, disponible en ediciones Pro, Enterprise o Education de Windows 10/11.[19] Debe estar habilitado a través de "Activar o desactivar las características de Windows".[19]
  - **VirtualBox:** Un hipervisor de código abierto popular que funciona en todas las ediciones de Windows.[21] Debe descargarse e instalarse por separado.[22]
  - **Docker Desktop:** Aunque no es un hipervisor tradicional, Docker Desktop (especialmente con el backend WSL 2) puede gestionar máquinas virtuales ligeras o contenedores para ejecutar Kubernetes.[22]
- **Conexión a Internet:** Necesaria para descargar Minikube, imágenes de Kubernetes y otros componentes.[18]
- **kubectl:** La herramienta de línea de comandos de Kubernetes para interactuar con el clúster. Aunque Minikube puede descargar su propia versión [26], y Docker Desktop lo incluye [25], a menudo es útil instalarlo por separado siguiendo las instrucciones oficiales.[21] Se puede instalar en Windows usando Chocolatey (choco install kubernetes-cli) [23] o descargando el binario.[22]

2.2 Método 1: Minikube

Minikube es una herramienta popular que ejecuta un clúster de Kubernetes de un solo nodo dentro de una máquina virtual (VM) o un contenedor en el equipo local.[21] Es ideal para desarrollo, pruebas y aprendizaje.[18]

Instalación:

Existen varias formas de instalar Minikube en Windows:

1. **Usando Chocolatey (Recomendado):**
    - Abrir PowerShell como Administrador.
    - Si no se tiene Chocolatey, instalarlo primero.[23]
    - Ejecutar: choco install minikube.[19]
    - Cerrar y reabrir la sesión de PowerShell/terminal para que Minikube esté en el PATH.[21]
2. **Usando Winget (Windows Package Manager):**
    - Abrir PowerShell.
    - Ejecutar: winget install minikube.[22]
3. **Descarga Directa (Instalador.exe):**
    - Descargar minikube-installer.exe desde la página de lanzamientos de Minikube en GitHub.[21]
    - Ejecutar el instalador.[21]
4. **Descarga Directa (Binario.exe):**
    - Descargar el archivo minikube-windows-amd64.exe desde la página de lanzamientos.[18]
    - Renombrar el archivo a minikube.exe.[18]
    - Añadir el directorio que contiene minikube.exe al PATH del sistema.[18]

Inicio del Clúster:

Minikube necesita un "driver" para crear el entorno de Kubernetes (VM o contenedor). Se debe especificar el driver deseado al iniciar.

- **Con VirtualBox:**
    - Asegurarse de que VirtualBox esté instalado.[22]
    - Abrir un terminal (PowerShell o Símbolo del sistema) **como Administrador**.[26]
    - Ejecutar: minikube start --driver=virtualbox.[18]
    - Se pueden añadir opciones para configurar recursos, por ejemplo: minikube start --driver=virtualbox --cpus=2 --memory=4096m --disk-size=30g.[22]
- **Con Hyper-V:**
    - Asegurarse de que Hyper-V esté habilitado.[24]
    - Abrir un terminal (PowerShell o Símbolo del sistema) **como Administrador**.[26]
    - Se necesita un conmutador virtual (Virtual Switch) en Hyper-V para que Minikube tenga conectividad de red. Si no existe uno adecuado (preferiblemente de tipo Externo para acceso a internet), Minikube puede intentar crear uno, o se puede especificar uno existente.[23] Es recomendable crear un conmutador Externo

previamente en el Administrador de Hyper-V.
    - Ejecutar: minikube start --driver=hyperv --hyperv-virtual-switch="<NombreDelVirtualSwitch>".[23] Reemplazar <NombreDelVirtualSwitch> con el nombre del conmutador virtual elegido. Minikube intentará encontrar uno si no se especifica.[24]
    - Minikube descargará la ISO necesaria y creará la VM en Hyper-V.[23]

Verificación:
Una vez que minikube start finalice:

1. **Comprobar el estado:** Ejecutar minikube status.[18] La salida debería indicar que los componentes host, kubelet y apiserver están Running.
2. **Interactuar con kubectl:**
    - Usar el kubectl integrado de Minikube: minikube kubectl -- get nodes o minikube kubectl -- get po -A.[26]
    - Si kubectl está instalado por separado y configurado, usarlo directamente: kubectl get nodes.[23] La salida debería mostrar el nodo de Minikube en estado Ready.
3. **Abrir el Dashboard (Opcional):** Ejecutar minikube dashboard.[18] Esto abrirá el panel de control web de Kubernetes en el navegador, proporcionando una vista gráfica del clúster.

2.3 Método 2: Kubernetes en Docker Desktop

Docker Desktop para Windows ofrece una opción integrada para ejecutar un clúster de Kubernetes de un solo nodo (o multi-nodo con la opción kind) directamente gestionado por Docker.[25] Es una opción conveniente si ya se utiliza Docker Desktop.[32]

Habilitación:

1. Asegurarse de que Docker Desktop esté instalado y funcionando.[27]
2. Abrir el **Dashboard** de Docker Desktop.
3. Ir a **Settings** (icono de engranaje).
4. Seleccionar la pestaña **Kubernetes**.[25]
5. Marcar la casilla **Enable Kubernetes**.[25]
6. (Opcional, versiones recientes) Elegir el método de aprovisionamiento del clúster (kubeadm o kind).[25] kind es más moderno y permite multi-nodo, pero requiere inicio de sesión y containerd.[25]
7. Hacer clic en **Apply & Restart**.[25] Docker Desktop descargará las imágenes necesarias y iniciará el clúster de Kubernetes en segundo plano. Esto puede tardar unos minutos dependiendo de la conexión a internet.[27]

Verificación:

1. **Estado en Docker Desktop:** El pie de página del Dashboard de Docker Desktop y el menú de Docker indicarán que Kubernetes está running.[25]
2. **Usar kubectl:** Docker Desktop instala kubectl.exe (normalmente en C:\Program Files\Docker\Docker\Resources\bin\) y configura el contexto automáticamente.[25] Abrir

PowerShell o Símbolo del sistema.

- Verificar la versión: kubectl version.[25] Debería mostrar las versiones del cliente y del servidor.
- Verificar el contexto actual: kubectl config current-context.[25] Debería ser docker-desktop. Si no lo es, cambiarlo con kubectl config use-context docker-desktop.[25]
- Listar los nodos: kubectl get nodes.[25] Debería mostrar un nodo llamado docker-desktop en estado Ready.

2.4 Eligiendo entre Minikube y Docker Desktop

Tanto Minikube como Docker Desktop Kubernetes son opciones viables para un entorno de desarrollo local en Windows. La elección depende de las necesidades específicas:

Característica	Minikube	Docker Desktop Kubernetes
Facilidad de Uso	Requiere instalación y configuración de driver separadas. CLI-céntrico.	Integrado en Docker Desktop. Se activa con un clic en la GUI.
Flexibilidad	Alta. Permite elegir versión de K8s [34], múltiples drivers (Hyper-V, VirtualBox, Docker, etc.) [21], configuración avanzada vía CLI.[34] Soporta multi-nodo.[22]	Baja. Versión de K8s fija a la de Docker Desktop.[32] Menos opciones de configuración directa.[34] Multi-nodo con 'kind' (reciente).[25]
Recursos	Puede consumir más recursos (VM completa) [37], pero configurable.[34]	Potencialmente más ligero (depende del backend, WSL2 vs Hyper-V) [34], pero la configuración de recursos es vía GUI.[34]
Integración Docker	Requiere pasos adicionales para usar el daemon de Docker del host (ej. minikube docker-env) o usa su propio Docker interno.	Integración directa. Las imágenes construidas localmente están disponibles inmediatamente.[38]
Add-ons	Comandos CLI para habilitar add-ons comunes (dashboard, ingress).[26]	Requiere aplicación manual de manifiestos para add-ons.
Comunidad/Soporte	Proyecto oficial de Kubernetes SIG.[36] Amplia comunidad y documentación.[36]	Soportado por Docker Inc. Buena documentación e integración.[30]
Estabilidad	Generalmente estable, pero depende del driver.	Generalmente estable, integrado en el ecosistema

		Docker.

Recomendación:

- Para usuarios que **ya usan Docker Desktop** y solo necesitan un clúster K8s local básico para probar despliegues, la opción integrada es la más **sencilla y rápida** de empezar.[32]
- Para usuarios que necesitan **flexibilidad** para probar **diferentes versiones de Kubernetes**, usar **drivers específicos** (como VirtualBox si Hyper-V no está disponible), configurar **recursos detalladamente vía CLI**, o buscan una experiencia más cercana a un clúster estándar, **Minikube** es una mejor opción.[32]

Ambas herramientas pueden coexistir en el mismo sistema si es necesario [32], simplemente se gestionan los contextos de kubectl para apuntar al clúster deseado.

Sección 3: Interacción con Kubernetes: kubectl Esenciales

Una vez que se tiene un clúster de Kubernetes local funcionando (ya sea con Minikube o Docker Desktop), la herramienta principal para interactuar con él es kubectl.

3.1 Introducción a kubectl

kubectl es la interfaz de línea de comandos (CLI) oficial para Kubernetes.[39] Permite ejecutar comandos contra clústeres de Kubernetes para desplegar aplicaciones, inspeccionar y gestionar recursos del clúster, ver logs y mucho más.[39] Se comunica con el plano de control del clúster (específicamente, el kube-apiserver) utilizando la API de Kubernetes.[39]
kubectl busca un archivo de configuración llamado config en el directorio $HOME/.kube para obtener la información de conexión al clúster (endpoint de la API, credenciales, contexto actual).[39] Tanto Minikube como Docker Desktop configuran automáticamente este archivo o su propio contexto al iniciarse.[25]
La sintaxis general de un comando kubectl es [39]:
kubectl [comando][flags]

- comando: La acción a realizar (ej. get, describe, create, apply, delete, logs, exec).
- TIPO_RECURSO: El tipo de recurso sobre el que actuar (ej. pods, nodes, services, deployments). Es insensible a mayúsculas/minúsculas y a menudo tiene abreviaturas (ej. po para pods, svc para services, deploy para deployments).[43]
- NOMBRE_RECURSO (opcional): El nombre específico del recurso. Si se omite, el comando suele actuar sobre todos los recursos del tipo especificado en el contexto actual.
- flags (opcional): Opciones adicionales para modificar el comportamiento del comando (ej. -n <namespace> para especificar un namespace, -o <formato> para cambiar el formato de salida, -l <selector> para filtrar por etiquetas).

3.2 Comandos Esenciales (get)

El comando get se utiliza para listar y obtener información básica sobre uno o más recursos.

- **kubectl cluster-info**
  - **Propósito:** Muestra información básica sobre los endpoints del plano de control y servicios clave del clúster.[40] Es útil para verificar la conectividad básica con el clúster.[45]
  - **Ejemplo de Salida:**
    Kubernetes control plane is running at https://192.168.49.2:8443
    CoreDNS is running at https://192.168.49.2:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

  - **Interpretación:** Indica la URL del servidor API principal y la ubicación del servicio DNS del clúster (CoreDNS en este caso), esencial para la resolución de nombres de servicios internos.[44] Si el comando falla, sugiere un problema de conectividad o configuración de kubectl.[45]
- **kubectl get nodes**
  - **Propósito:** Lista todos los nodos (máquinas trabajadoras) en el clúster.[40]
  - **Ejemplo de Salida:**
    NAME STATUS ROLES AGE VERSION
    minikube Ready control-plane 15m v1.29.1

  - **Interpretación:** Muestra el nombre de cada nodo, su estado (Ready significa saludable y operativo, NotReady indica un problema), los roles asignados (ej. control-plane o <none> para worker), cuánto tiempo lleva activo (AGE) y la versión de Kubernetes que ejecuta.[40] En Minikube o Docker Desktop K8s de un solo nodo, se verá un único nodo que a menudo tiene el rol control-plane (o master) pero también actúa como worker.[44]
- **kubectl get pods**
  - **Propósito:** Lista todos los pods en el namespace actual.[40] Se puede usar -n <namespace> para ver pods en otro namespace o -A (o --all-namespaces) para verlos en todos los namespaces.[50]
  - **Ejemplo de Salida:**
    NAME READY STATUS RESTARTS AGE
    mi-app-deployment-7b7d7f467b-xqrwv 1/1 Running 0 5m

  - **Interpretación:** Muestra el nombre del pod, cuántos contenedores están listos (READY, ej. 1/1), el estado actual del pod (Pending, Running, Succeeded, Failed, CrashLoopBackOff), el número de reinicios (RESTARTS, un número alto puede indicar problemas) y la edad del pod (AGE).[40]
- **kubectl get services** (o svc)
  - **Propósito:** Lista todos los servicios en el namespace actual.[40]
  - **Ejemplo de Salida:**

```
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP        20m
mi-servicio  NodePort    10.108.144.14  <none>        80:31567/TCP   8m
```

    - **Interpretación:** Muestra el nombre del servicio, su tipo (ClusterIP, NodePort, LoadBalancer), la IP interna del clúster (CLUSTER-IP), la IP externa (si aplica, <none> o <pending> si no está asignada), los puertos expuestos (PORT(S), ej. 80:31567/TCP significa que el servicio escucha en el puerto 80 internamente y en el puerto 31567 en los nodos) y la edad del servicio.[40]
- **kubectl get deployments** (o deploy)
    - **Propósito:** Lista todos los deployments en el namespace actual.[17]
    - **Ejemplo de Salida:**
      ```
      NAME              READY   UP-TO-DATE   AVAILABLE   AGE
      mi-app-deployment 1/1     1            1           12m
      ```
    - **Interpretación:** Muestra el nombre del deployment, cuántas réplicas están listas (READY, formato actual/deseado), cuántas están actualizadas a la última versión (UP-TO-DATE), cuántas están disponibles para servir tráfico (AVAILABLE) y la edad del deployment.[17]

3.3 Obtención de Información Detallada (describe)

El comando describe proporciona información mucho más detallada sobre un recurso específico, lo cual es muy útil para la depuración y el diagnóstico.

- **kubectl describe node <nombre-nodo>**
    - **Propósito:** Muestra detalles exhaustivos sobre un nodo específico, incluyendo su estado, condiciones (presión de disco/memoria), capacidad y recursos asignados, información del sistema (versión de Kubelet, runtime), y eventos recientes relacionados con el nodo.[42]
    - **Interpretación:** Buscar secciones clave como Conditions para ver el estado de salud detallado y Events para encontrar mensajes de error o advertencias recientes que puedan explicar por qué un nodo está NotReady o tiene problemas. La sección Allocated resources muestra cuánta CPU y memoria están usando los pods en ese nodo.[47]
- **kubectl describe pod <nombre-pod>**
    - **Propósito:** Muestra detalles exhaustivos sobre un pod específico, incluyendo su estado, IP, nodo asignado, información sobre sus contenedores (imagen, estado, reinicios, puertos, montajes de volúmenes, variables de entorno), condiciones y eventos recientes.[47]
    - **Interpretación:** La sección Events es a menudo la más útil para diagnosticar por qué un pod no se inicia (Pending), está fallando (CrashLoopBackOff, Error) o no está listo (READY 0/1).[47] Muestra eventos como la asignación a un nodo, el pull de imágenes, la creación y el inicio de contenedores, y cualquier error encontrado.

La sección Containers muestra el estado de cada contenedor individualmente. Estos comandos get y describe son fundamentales para observar y comprender el estado del clúster y de las aplicaciones que se ejecutan en él.

Sección 4: Despliegue de la Primera Aplicación (Enfoque Imperativo)

Ahora que se tiene un clúster local y se conocen los comandos básicos de kubectl, se puede desplegar la primera aplicación. Se comenzará con el enfoque *imperativo*, que implica dar órdenes directas a Kubernetes a través de comandos kubectl.

4.1 Enfoque Imperativo vs. Declarativo

Existen dos formas principales de gestionar los recursos en Kubernetes [53]:

- **Imperativo:** Se le dice a Kubernetes *qué hacer* paso a paso mediante comandos directos como kubectl create, kubectl expose, kubectl scale.[53] Es rápido para tareas sencillas, experimentación o aprendizaje.[54] Sin embargo, no es fácilmente reproducible, versionable ni auditable, lo que lo hace menos ideal para entornos de producción.[53]
- **Declarativo:** Se le dice a Kubernetes *cuál es el estado deseado* del sistema describiéndolo en archivos de manifiesto (generalmente YAML) y luego se usa kubectl apply -f <archivo.yaml>.[53] Kubernetes se encarga de alcanzar y mantener ese estado deseado.[53] Este enfoque es el preferido para la mayoría de los casos de uso, especialmente en producción, ya que los archivos YAML pueden versionarse en Git (GitOps), son reutilizables y proporcionan un registro claro de la configuración.[54]

En esta sección, se usará el enfoque imperativo para desplegar una aplicación Nginx simple. En la siguiente sección, se hará lo mismo pero usando el enfoque declarativo con archivos YAML.

4.2 Creación de un Deployment (kubectl create deployment)

Un Deployment gestiona la creación y el estado de los Pods de una aplicación.[17] Se puede crear un Deployment imperativamente usando kubectl create deployment.

- **Comando:** kubectl create deployment <nombre-deployment> --image=<imagen-contenedor>.[58]
- **Ejemplo con Nginx:** Para crear un Deployment llamado nginx-imperative que ejecute la imagen oficial de Nginx desde Docker Hub:
  Bash
  kubectl create deployment nginx-imperative --image=nginx
  Por defecto, esto crea un Deployment que gestionará 1 réplica (1 Pod).[58]
- **Verificación:**
  - kubectl get deployments: Debería mostrar nginx-imperative con READY 1/1.
  - kubectl get pods: Debería mostrar un pod con un nombre como nginx-imperative-xxxxxxxxxx-yyyyy en estado Running.

4.3 Exposición del Deployment (kubectl expose deployment)

Para acceder a la aplicación Nginx desde fuera del clúster, se necesita exponer el Deployment a través de un Servicio. Se usará el tipo NodePort para hacerlo accesible a través de un puerto en la IP del nodo local.[61]

- **Comando:** kubectl expose deployment <nombre-deployment> --type=NodePort --port=<puerto-servicio> --target-port=<puerto-contenedor>.[61]
    - --type=NodePort: Especifica el tipo de servicio.
    - --port: El puerto en el que el Servicio escuchará *dentro* del clúster.
    - --target-port: El puerto en el que el contenedor Nginx está escuchando (normalmente el puerto 80 para Nginx).
- **Ejemplo exponiendo Nginx:**
  Bash
  kubectl expose deployment nginx-imperative --type=NodePort --port=80 --target-port=80
  Esto crea un Servicio llamado nginx-imperative (toma el nombre del Deployment por defecto).[61]
- **Verificación:**
    - kubectl get services (o svc): Debería mostrar el servicio nginx-imperative de tipo NodePort. Anotar el puerto asignado en la columna PORT(S) (ej. 80:3xxxx/TCP).

4.4 Acceso a la Aplicación

Ahora que el servicio está expuesto, se puede acceder a Nginx. El método varía ligeramente entre Minikube y Docker Desktop.

- **Método 1: Usando minikube service (Solo para Minikube)**
    - Ejecutar: minikube service nginx-imperative --url.[62]
    - Esto abrirá una URL en el navegador (o la imprimirá en la consola) que apunta directamente a la aplicación Nginx a través de un túnel o la IP del nodo.[62] Si se usa el driver Docker, este comando creará un túnel y la terminal deberá permanecer abierta.[62]
- **Método 2: Usando la IP del Nodo y el NodePort (Minikube con drivers VM, o clústeres remotos)**
    1. Obtener la IP del nodo Minikube: minikube ip.[62]
    2. Obtener el NodePort asignado al servicio: kubectl get svc nginx-imperative. Buscar el puerto alto (ej. 3xxxx) en la columna PORT(S).
    3. Acceder en el navegador: http://<minikube-ip>:<nodeport>.[62]
- **Método 3: Usando kubectl port-forward (Funciona en ambos, recomendado para Docker Desktop)**
    - Este método crea un túnel seguro desde la máquina local directamente al servicio o pod, sin depender del NodePort.[65]
    - Ejecutar en una terminal separada: kubectl port-forward service/nginx-imperative <puerto-local>:<puerto-servicio>

        - Ejemplo: kubectl port-forward service/nginx-imperative 8080:80.[69]
    - Acceder en el navegador: http://localhost:8080.[65]
    - La terminal donde se ejecuta port-forward debe permanecer abierta.[68] Presionar Ctrl+C para detener el reenvío.

El enfoque imperativo permite poner en marcha rápidamente una aplicación, pero para una gestión más robusta y mantenible, se prefiere el enfoque declarativo, que se explorará a continuación.

Sección 5: Gestión Declarativa de Aplicaciones con YAML

Aunque los comandos imperativos son útiles para tareas rápidas, el enfoque declarativo mediante archivos YAML es el método estándar y recomendado para gestionar recursos en Kubernetes, especialmente en entornos colaborativos o de producción.[53]

5.1 Importancia de la Configuración Declarativa

La gestión declarativa consiste en definir el *estado deseado* de los recursos en archivos de configuración (manifiestos), en lugar de ejecutar una secuencia de comandos para alcanzar ese estado.[53] Kubernetes se encarga entonces de conciliar continuamente el estado *actual* del clúster con el estado *deseado* definido en los manifiestos.[13]

Ventajas del enfoque declarativo:

- **Control de Versiones (GitOps):** Los archivos YAML pueden (y deben) almacenarse en sistemas de control de versiones como Git. Esto permite rastrear cambios, colaborar en la configuración, revisar modificaciones y revertir a estados anteriores fácilmente.[54]
- **Reproducibilidad y Consistencia:** Aplicar el mismo archivo YAML siempre resultará en el mismo estado deseado en cualquier clúster compatible, asegurando despliegues consistentes entre entornos (desarrollo, staging, producción).[54]
- **Auditabilidad:** Los archivos YAML sirven como documentación clara y auditable de la configuración del clúster.[54]
- **Gestión de Cambios Complejos:** Las actualizaciones y cambios en la configuración se gestionan modificando el archivo YAML y volviendo a aplicar (kubectl apply), lo que permite a Kubernetes manejar las transiciones de estado de forma inteligente (por ejemplo, mediante rolling updates).[56]
- **Infraestructura como Código (IaC):** Trata la configuración de la infraestructura y las aplicaciones como código, aplicando las mejores prácticas de desarrollo de software a la gestión de operaciones.

5.2 Anatomía de un Deployment YAML

Un archivo YAML de Deployment define cómo se debe ejecutar y gestionar un conjunto de Pods idénticos. A continuación se muestra una estructura básica con explicaciones [17]:

YAML

```
# deployment-nginx.yaml
apiVersion: apps/v1       # Versión de la API para Deployments
kind: Deployment          # Tipo de objeto Kubernetes
metadata:
  name: nginx-yaml-deployment # Nombre único para este Deployment
  labels:
    app: nginx-yaml       # Etiqueta para identificar/agrupar este Deployment
spec:                     # Especificación del estado deseado
  replicas: 2             # Número deseado de Pods
  selector:               # Cómo el Deployment encuentra los Pods que gestiona
    matchLabels:
      app: nginx-yaml     # Debe coincidir con las etiquetas de la plantilla de Pod
  template:               # Plantilla para crear los Pods
    metadata:
      labels:             # Etiquetas que se aplicarán a cada Pod creado
        app: nginx-yaml   # Esta etiqueta debe coincidir con spec.selector.matchLabels
    spec:                 # Especificación del Pod
      containers:         # Lista de contenedores a ejecutar en el Pod
      - name: nginx-container  # Nombre del contenedor dentro del Pod
        image: nginx:latest    # Imagen Docker a utilizar (ej. nginx con tag 'latest')
        ports:
        - containerPort: 80    # Puerto que expone el contenedor
        # Aquí se pueden añadir más configuraciones: resources, env, volumeMounts, probes,
etc.
```

- **apiVersion y kind:** Identifican el tipo de objeto y la versión de la API utilizada.[17]
- **metadata:** Contiene información sobre el objeto, como su name (obligatorio y único en el namespace) y labels (pares clave-valor para organización y selección).[17]
- **spec:** Describe el estado deseado del Deployment.[17]
    - **replicas:** Número de Pods idénticos que se desean ejecutar.[17]
    - **selector:** Define cómo el Deployment identifica los Pods que debe gestionar. matchLabels especifica las etiquetas que deben tener los Pods.[17] **Es crucial que estas etiquetas coincidan con las definidas en template.metadata.labels**.[17]
    - **template:** Es la plantilla utilizada para crear cada Pod. Contiene su propia metadata (con labels para los Pods) y spec (la especificación del Pod).[17]
        - **template.spec.containers:** Define los contenedores que se ejecutarán dentro de cada Pod, incluyendo su name, image y ports.[17]

5.3 Anatomía de un Service YAML (NodePort)

Un archivo YAML de Service define cómo exponer un conjunto de Pods (normalmente gestionados por un Deployment) como un servicio de red. A continuación se muestra una estructura básica para un Service de tipo NodePort [76]:

YAML

```
# service-nginx.yaml
apiVersion: v1            # Versión de la API para Services
kind: Service             # Tipo de objeto Kubernetes
metadata:
  name: nginx-yaml-service  # Nombre único para este Service
spec:                     # Especificación del estado deseado
  type: NodePort          # Tipo de Service: expone en IP de nodo + puerto estático
  selector:               # Cómo el Service encuentra los Pods a los que enrutar tráfico
    app: nginx-yaml       # Debe coincidir con las etiquetas de los Pods del Deployment
  ports:                  # Lista de puertos que expone el Service
    - protocol: TCP       # Protocolo (TCP es default)
      port: 80            # Puerto en la IP interna del Service (ClusterIP)
      targetPort: 80      # Puerto en los Pods al que se reenvía el tráfico
      # nodePort: 30080    # Opcional: Puerto estático en los nodos (rango 30000-32767)
                           # Si se omite, Kubernetes asigna uno automáticamente.
```

- **apiVersion y kind:** Identifican el objeto Service.[76]
- **metadata.name:** Nombre del Service, usado para DNS interno.[76]
- **spec:** Describe el Service.[76]
    - **type: NodePort:** Define el tipo de exposición. NodePort hace que el servicio sea accesible en un puerto específico en *cada* nodo del clúster, además de asignarle un ClusterIP interno.[76]
    - **selector:** Clave para conectar el Service con los Pods. Debe contener etiquetas que coincidan con las etiquetas de los Pods que se quieren exponer (los gestionados por el Deployment nginx-yaml-deployment en este caso).[76]
    - **ports:** Define el mapeo de puertos.[76]
        - protocol: Protocolo de red (TCP, UDP, SCTP).
        - port: Puerto en el que el Service escucha dentro del clúster (en su ClusterIP).
        - targetPort: Puerto en los contenedores de los Pods al que se dirige el tráfico.
        - nodePort (solo para NodePort y LoadBalancer): Puerto estático (entre 30000-32767 por defecto) en el que el servicio es accesible en la IP de cada nodo. Es opcional; si no se especifica, Kubernetes asigna uno.[76]

5.4 Aplicación y Eliminación de Recursos vía YAML (kubectl apply -f,

kubectl delete -f)

- **kubectl apply -f <archivo.yaml>:**
  - Este es el comando fundamental para la gestión declarativa.[53]
  - Lee la definición del recurso del archivo YAML especificado y la aplica al clúster.
  - **Idempotente:** Si el recurso definido en el archivo no existe, lo crea. Si ya existe, lo actualiza para que coincida con la definición del archivo.[73] Kubernetes calcula e implementa solo los cambios necesarios.
  - Es el comando preferido para flujos de trabajo declarativos y GitOps porque maneja tanto la creación inicial como las actualizaciones posteriores de manera consistente.[73]
  - Ejemplo de salida al crear: deployment.apps/nginx-yaml-deployment created.[81]
  - Ejemplo de salida al actualizar: deployment.apps/nginx-yaml-deployment configured.[81]
- **kubectl create -f <archivo.yaml>:**
  - También crea recursos a partir de un archivo YAML.[81]
  - **No Idempotente:** Solo funciona si el recurso no existe previamente. Si el recurso ya existe, devuelve un error.[73]
  - Generalmente se usa menos en flujos declarativos, ya que no maneja actualizaciones. Puede ser útil para crear recursos que no se planea modificar a través de apply.
- **kubectl delete -f <archivo.yaml>:**
  - Elimina los recursos definidos en el archivo YAML especificado.[86]
  - Es una forma conveniente de limpiar los recursos creados a partir de un manifiesto.

Manos a la Obra (Declarativo):

1. Guardar el YAML del Deployment (Sección 5.2) como deployment-nginx.yaml.
2. Guardar el YAML del Service (Sección 5.3) como service-nginx.yaml.
3. **Aplicar el Deployment:**
   Bash
   kubectl apply -f deployment-nginx.yaml
   Verificar: kubectl get deployments, kubectl get pods -l app=nginx-yaml.
4. **Aplicar el Service:**
   Bash
   kubectl apply -f service-nginx.yaml
   Verificar: kubectl get services nginx-yaml-service. Anotar el NodePort si no se especificó uno.
5. **Acceder al Servicio:** Usar los métodos de la Sección 4.4 (ej. minikube service nginx-yaml-service --url o kubectl port-forward service/nginx-yaml-service 8080:80).
6. **Limpiar:**
   Bash
   kubectl delete -f service-nginx.yaml

kubectl delete -f deployment-nginx.yaml
Verificar que los recursos se hayan eliminado: kubectl get services,deployments,pods -l app=nginx-yaml.

El uso de kubectl apply -f es central para la filosofía declarativa de Kubernetes. Permite definir toda la infraestructura y configuración de la aplicación en archivos que pueden ser versionados y revisados, proporcionando un método robusto y repetible para gestionar el ciclo de vida de las aplicaciones. Este enfoque contrasta con el imperativo, que aunque útil para tareas rápidas, carece de la trazabilidad y consistencia necesarias para entornos complejos o de producción.

Sección 6: Despliegue de Su Aplicación Python Flask

Ahora se aplicarán los conceptos aprendidos para desplegar una aplicación Flask previamente contenedorizada en el clúster local de Kubernetes. Se asume que los estudiantes ya tienen conocimientos de Docker y han creado una imagen Docker para su aplicación Flask.

6.1 Preparación de la Aplicación Flask (Recapitulación de Contenerización)

Antes de desplegar en Kubernetes, la aplicación Flask debe estar empaquetada como una imagen de contenedor y disponible en un registro.

1. **Código de la Aplicación Flask:** Se necesita una aplicación Flask funcional. Por ejemplo, un simple app.py y un requirements.txt.
2. **Dockerfile:** Es necesario un Dockerfile para construir la imagen. Un ejemplo básico podría ser [66]:
   Dockerfile
   ```
   # Usar una imagen base de Python
   FROM python:3.9-slim

   # Establecer directorio de trabajo
   WORKDIR /app

   # Copiar archivos de requisitos e instalar dependencias
   COPY requirements.txt requirements.txt
   RUN pip install --no-cache-dir -r requirements.txt

   # Copiar el resto del código de la aplicación
   COPY..

   # Exponer el puerto en el que Flask se ejecutará (ej. 5000)
   EXPOSE 5000

   # Comando para ejecutar la aplicación (usando Flask o Gunicorn)
   # Asegurarse de que escucha en 0.0.0.0 para ser accesible desde fuera del
   ```

contenedor
CMD ["flask", "run", "--host=0.0.0.0", "--port=5000"]
Alternativa con Gunicorn: CMD ["gunicorn", "-b", "0.0.0.0:5000", "app:app"]
Es fundamental que el comando CMD ejecute el servidor Flask (o un servidor WSGI como Gunicorn) escuchando en 0.0.0.0 para que sea accesible desde la red del Pod, no solo desde localhost dentro del contenedor.[92] El puerto expuesto con EXPOSE (ej. 5000) debe coincidir con el puerto en el que la aplicación escucha.

3. **Construir la Imagen Docker:** Navegar al directorio que contiene el Dockerfile y ejecutar:
   Bash
   docker build -t <tu-usuario-dockerhub>/mi-flask-app:v1.
   Reemplazar <tu-usuario-dockerhub> con el nombre de usuario real y v1 con una etiqueta de versión.[92]
4. **Empujar la Imagen a un Registro:** Kubernetes necesita poder descargar la imagen. El clúster local (Minikube/Docker Desktop) generalmente no tiene acceso directo a las imágenes locales del host a menos que se tomen medidas específicas (como minikube image load [93] o usar el daemon de Docker Desktop). La forma estándar es empujar la imagen a un registro de contenedores como Docker Hub, Google Container Registry (GCR), etc..[66]
   Bash
   # Iniciar sesión en Docker Hub (si es necesario)
   # docker login

   # Empujar la imagen
   docker push <tu-usuario-dockerhub>/mi-flask-app:v1
   [91]

El hecho de que Kubernetes opere sobre imágenes de contenedor existentes subraya una separación clave de responsabilidades: el desarrollo y empaquetado de la aplicación (Docker) es un paso previo y distinto a la orquestación y gestión del despliegue (Kubernetes). Tener la imagen en un registro accesible es, por tanto, un prerrequisito indispensable.

6.2 Creación del Deployment YAML para Flask

Ahora se creará el archivo flask-deployment.yaml basado en el ejemplo de Nginx de la Sección 5.2, adaptándolo para la aplicación Flask.

YAML

```
# flask-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: flask-app-deployment  # Nombre del Deployment
  labels:
    app: flask-app          # Etiqueta para el Deployment
spec:
  replicas: 2               # Iniciar con 2 réplicas, por ejemplo
  selector:
    matchLabels:
      app: flask-app        # Selector para encontrar los Pods
  template:
    metadata:
      labels:
        app: flask-app      # Etiqueta para los Pods (debe coincidir con el selector)
    spec:
      containers:
      - name: flask-container # Nombre del contenedor
        # ¡IMPORTANTE! Usar la imagen Flask empujada al registro
        image: <tu-usuario-dockerhub>/mi-flask-app:v1
        ports:
        - containerPort: 5000   # Puerto que la aplicación Flask escucha (debe coincidir con Dockerfile y CMD)
```

91

Puntos Clave:

- Se han cambiado los nombres y etiquetas (metadata.name, labels, selector, template.metadata.labels) para reflejar que es una aplicación Flask (ej. flask-app).
- El campo spec.template.spec.containers.image **debe** apuntar a la imagen Flask que se construyó y empujó al registro en el paso anterior.
- El campo spec.template.spec.containers.ports.containerPort **debe** coincidir con el puerto en el que la aplicación Flask está escuchando dentro del contenedor (ej. 5000, según el Dockerfile y el comando CMD).

6.3 Creación del Service YAML para Flask

A continuación, se creará el archivo flask-service.yaml para exponer el Deployment Flask, usando NodePort para acceso externo simple.

YAML

```
# flask-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: flask-service      # Nombre del Service
```

```
spec:
  type: NodePort          # Tipo de Service para acceso externo simple
  selector:
    app: flask-app        # ¡IMPORTANTE! Debe coincidir con las etiquetas de los Pods del Deployment
  ports:
    - protocol: TCP
      port: 80            # Puerto en el ClusterIP del Service (puede ser 80, 5000, u otro)
      targetPort: 5000    # Puerto en los contenedores Flask (debe coincidir con containerPort)
      # nodePort: 30001   # Opcional: especificar un NodePort fijo
```

91

Puntos Clave:

- El spec.selector **debe** coincidir exactamente con las etiquetas de los Pods definidos en flask-deployment.yaml (en este caso, app: flask-app). Así es como el Servicio sabe a qué Pods enviar el tráfico.
- El targetPort (5000) **debe** coincidir con el containerPort que la aplicación Flask expone.
- El port (80) es el puerto en el que el Servicio escuchará dentro del clúster. Se puede elegir un puerto estándar como 80, o el mismo que el targetPort (5000), u otro. Las otras aplicaciones dentro del clúster se conectarán a este puerto.
- Se mantiene type: NodePort para simplificar el acceso externo en el entorno local.

6.4 Despliegue de la Aplicación Flask (kubectl apply)

Con los archivos YAML listos, se aplican al clúster:

Bash

```
kubectl apply -f flask-deployment.yaml
kubectl apply -f flask-service.yaml
```

92

Verificación:

- Comprobar que el Deployment se ha creado y las réplicas están listas:
  Bash
  kubectl get deployment flask-app-deployment
  kubectl get pods -l app=flask-app
  Se deberían ver 2 pods (o el número de réplicas especificado) en estado Running.
- Comprobar que el Service se ha creado y tiene un NodePort asignado:
  Bash
  kubectl get service flask-service

- Si los pods no se inician correctamente (ej. estado ImagePullBackOff o CrashLoopBackOff), revisar los logs del pod:
  Bash
  kubectl logs <nombre-del-pod-flask>
  [43] Y también los eventos del pod para más detalles:
  Bash
  kubectl describe pod <nombre-del-pod-flask>

6.5 Acceso a la Aplicación Flask (minikube service / kubectl port-forward)

Se utilizan los mismos métodos que en la Sección 4.4, pero aplicados al servicio Flask:

- **Método 1 (Minikube):**
  Bash
  minikube service flask-service --url
  [62] Acceder a la URL proporcionada en el navegador. Recordar mantener la terminal abierta si se usa el driver Docker.
- **Método 2 (Docker Desktop / Fiable):**
  Bash
  kubectl port-forward service/flask-service 8080:80
  (Se usa 8080 como puerto local y 80 como el port del servicio definido en flask-service.yaml. Si se usó un port diferente en el YAML, ajustar el comando). [65]
  Acceder a http://localhost:8080 en el navegador.

El uso de kubectl port-forward es especialmente práctico durante el desarrollo. Permite una conexión directa desde la máquina local al servicio que se ejecuta dentro del clúster, evitando las complejidades de la configuración de red externa (NodePort, LoadBalancer) y facilitando la prueba y depuración rápidas de la aplicación desplegada.[67] Es una forma segura y temporal de interactuar con la aplicación como si se estuviera ejecutando localmente.[68]

Sección 7: Gestión del Ciclo de Vida de la Aplicación

Una vez desplegada la aplicación, Kubernetes facilita la gestión de su ciclo de vida, incluyendo el escalado para manejar la carga y las actualizaciones para introducir nuevas versiones.

7.1 Escalado de la Aplicación (kubectl scale deployment)

El escalado se refiere a ajustar el número de instancias (Pods) de la aplicación que se ejecutan simultáneamente.[100] Esto es crucial para adaptarse a las variaciones de carga y para garantizar la alta disponibilidad.[100]

- **Comando Imperativo:** Se utiliza el comando kubectl scale para cambiar el número de réplicas de un Deployment existente.[103]

Bash
kubectl scale deployment <nombre-deployment> --replicas=<nuevo-numero>

[100]

- <nombre-deployment>: El nombre del Deployment a escalar (ej. flask-app-deployment).
- <nuevo-numero>: El número deseado de réplicas.

- **Manos a la Obra (Escalado):**
  1. **Verificar estado actual:**
     Bash
     kubectl get deployment flask-app-deployment
     kubectl get pods -l app=flask-app
     Anotar el número actual de réplicas listas.
  2. **Escalar hacia arriba (Scale Out):** Aumentar el número de réplicas, por ejemplo, a 3.
     Bash
     kubectl scale deployment flask-app-deployment --replicas=3

  3. **Verificar el escalado:** Esperar unos segundos y volver a ejecutar:
     Bash
     kubectl get deployment flask-app-deployment
     kubectl get pods -l app=flask-app
     Se debería ver READY 3/3 en el deployment y 3 pods en estado Running. Kubernetes ha creado automáticamente los nuevos pods necesarios.[103]
  4. **Escalar hacia abajo (Scale In):** Reducir el número de réplicas, por ejemplo, a 1.
     Bash
     kubectl scale deployment flask-app-deployment --replicas=1

  5. **Verificar de nuevo:**
     Bash
     kubectl get deployment flask-app-deployment
     kubectl get pods -l app=flask-app
     Se debería ver READY 1/1 y solo 1 pod. Kubernetes ha terminado gradualmente los pods sobrantes.[103]
  6. **Escalar a Cero (Opcional):** Se puede escalar un deployment a 0 réplicas para detener temporalmente la aplicación sin eliminar su configuración. Esto es útil para mantenimiento o para ahorrar recursos en entornos de no producción.[100]
     Bash
     kubectl scale deployment flask-app-deployment --replicas=0
     Verificar que no hay pods (No resources found). Para reiniciar la aplicación, simplemente escalar de nuevo a un número mayor que 0.

El escalado es una de las capacidades clave de Kubernetes, permitiendo que las aplicaciones

se adapten dinámicamente a las necesidades. Aunque aquí se realiza manualmente con kubectl scale, en entornos reales esto se suele automatizar con el Horizontal Pod Autoscaler (HPA).[102]

7.2 Actualización de la Aplicación con Rolling Updates

Cuando se necesita desplegar una nueva versión del código de la aplicación (por ejemplo, con correcciones de errores o nuevas características), Kubernetes utiliza por defecto la estrategia de **Rolling Update** (actualización continua) para los Deployments.[17]

- **¿Cómo Funcionan las Rolling Updates?**
  - El objetivo es actualizar la aplicación a una nueva versión **sin tiempo de inactividad** (zero downtime).[17]
  - El proceso se desencadena cuando se modifica la plantilla del Pod dentro de la especificación del Deployment (el campo spec.template), típicamente cambiando la etiqueta de la imagen del contenedor (spec.template.spec.containers.image).[17]
  - Al aplicar el cambio (usando kubectl apply -f <archivo-deployment.yaml>), el controlador de Deployment detecta la diferencia.[105]
  - Crea un **nuevo ReplicaSet** con la plantilla actualizada, pero inicialmente con 0 réplicas.[17]
  - Luego, **incrementa gradualmente** el número de réplicas del nuevo ReplicaSet mientras **reduce gradualmente** el número de réplicas del antiguo ReplicaSet.[105]
  - Kubernetes se asegura de que los nuevos Pods estén listos (Ready) antes de terminar los antiguos.[105]
  - Si el Deployment está expuesto por un Service, el Service automáticamente ajusta los endpoints para enviar tráfico solo a los Pods listos (tanto antiguos como nuevos durante la transición).[105]
- **Controlando la Actualización: maxSurge y maxUnavailable**
  - Estos parámetros, definidos en spec.strategy.rollingUpdate dentro del YAML del Deployment, controlan la velocidad y la disponibilidad durante la actualización.[105]
  - maxUnavailable: Número o porcentaje máximo de Pods (del total deseado) que pueden estar no disponibles durante la actualización. Por defecto suele ser 25% o 1. Un valor más bajo garantiza mayor disponibilidad pero ralentiza la actualización.[105]
  - maxSurge: Número o porcentaje máximo de Pods (por encima del total deseado) que se pueden crear adicionalmente durante la actualización. Por defecto suele ser 25% o 1. Permite que los nuevos Pods se inicien antes de que los antiguos se terminen, acelerando la actualización pero consumiendo temporalmente más recursos.[105]
- **Manos a la Obra (Rolling Update):**
  1. **Preparar Nueva Versión:** Construir y empujar una nueva versión de la imagen Docker de la aplicación Flask (ej. <tu-usuario-dockerhub>/mi-flask-app:v2). Asegurarse de que esta versión tenga algún cambio visible para poder verificar la

actualización.

2. **Actualizar el Deployment YAML:** Editar el archivo flask-deployment.yaml. Cambiar el valor del campo spec.template.spec.containers.image a la nueva etiqueta de imagen (ej. <tu-usuario-dockerhub>/mi-flask-app:v2).
3. **Aplicar el Cambio:**
   Bash
   kubectl apply -f flask-deployment.yaml
   La salida indicará deployment.apps/flask-app-deployment configured.
4. **Monitorizar el Despliegue:** Observar el proceso de actualización en tiempo real.
   - Comprobar el estado del rollout:
     Bash
     kubectl rollout status deployment/flask-app-deployment
     [40] Esperar hasta que indique "successfully rolled out".
   - Observar los Pods (usar la opción -w para ver cambios en tiempo real):
     Bash
     kubectl get pods -l app=flask-app -w
     Se verá cómo los Pods antiguos entran en estado Terminating y se crean nuevos Pods con nombres diferentes que eventualmente alcanzan el estado Running.
5. **Verificar la Actualización:**
   - Acceder a la aplicación Flask de nuevo usando minikube service o kubectl port-forward (como en la Sección 6.5). Se debería ver el contenido o comportamiento de la versión v2.
   - Inspeccionar uno de los nuevos Pods para confirmar la versión de la imagen:
     Bash
     # Obtener el nombre de un nuevo pod
     NEW_POD_NAME=$(kubectl get pods -l app=flask-app -o jsonpath='{.items.metadata.name}')
     # Describir el pod y buscar la imagen
     kubectl describe pod $NEW_POD_NAME | grep Image:
     [105] Debería mostrar la imagen con la etiqueta :v2.

- **Rollbacks:** Si algo sale mal con la nueva versión (ej. los nuevos pods entran en CrashLoopBackOff), Kubernetes detendrá el rollout (si progressDeadlineSeconds se excede o las readiness probes fallan). Se puede revertir manualmente a la versión anterior estable usando:
  Bash
  kubectl rollout undo deployment/flask-app-deployment

  También se puede volver a una revisión específica usando kubectl rollout history y kubectl rollout undo deployment/<nombre> --to-revision=<numero>.

La gestión de actualizaciones mediante rolling updates declarativas es una práctica estándar

en Kubernetes. Al definir el estado deseado en el archivo YAML y permitir que Kubernetes gestione la transición, se obtienen despliegues más seguros, predecibles y con mínima interrupción para los usuarios, además de la capacidad incorporada de revertir cambios si es necesario.

Sección 8: Limpieza del Entorno

Una vez finalizada la práctica o el desarrollo, es importante limpiar los recursos creados en el clúster de Kubernetes y, opcionalmente, detener o eliminar el propio clúster local para liberar recursos del sistema.

8.1 Eliminación de Recursos de Kubernetes (Deployments, Services)

Los Deployments y Services creados consumen recursos dentro del clúster. Es una buena práctica eliminarlos cuando ya no se necesiten.
Existen varias formas de eliminar recursos:

- **Método 1: Usando los archivos YAML (Recomendado si se usó apply -f)**
    - Si se crearon los recursos usando kubectl apply -f <archivo.yaml>, se pueden eliminar usando kubectl delete -f <archivo.yaml>.[86] Kubernetes utiliza la información del archivo para identificar y eliminar los objetos correspondientes.
    - **Orden:** Generalmente es buena idea eliminar primero los Services que exponen los Deployments, y luego los Deployments mismos (que a su vez eliminarán los ReplicaSets y Pods asociados).

  Bash
  kubectl delete -f flask-service.yaml
  kubectl delete -f flask-deployment.yaml
  # O si se usaron los archivos de Nginx YAML:
  # kubectl delete -f service-nginx.yaml
  # kubectl delete -f deployment-nginx.yaml

- **Método 2: Por tipo y nombre (Útil para recursos creados imperativamente)**
    - Se puede eliminar un recurso específico especificando su tipo y nombre.[87]
      Bash
      kubectl delete service <nombre-servicio>
      kubectl delete deployment <nombre-deployment>

    - Ejemplos:

  Bash
  kubectl delete service flask-service
  kubectl delete deployment flask-app-deployment
  # O para los recursos imperativos de Nginx:
  # kubectl delete service nginx-imperative
  # kubectl delete deployment nginx-imperative
  Se pueden eliminar múltiples recursos en un solo comando: kubectl delete service/flask-service deployment/flask-app-deployment.[111]
- **Método 3: Por etiqueta (Usar con precaución)**

    - Se pueden eliminar todos los recursos que coincidan con una etiqueta específica usando el flag -l.

  Bash
  kubectl delete service -l app=flask-app
  kubectl delete deployment -l app=flask-app
  Esto eliminará *todos* los servicios y deployments con esa etiqueta en el namespace actual. Hay que ser cuidadoso para no eliminar recursos no deseados.
- **Método 4: Todos los recursos en un Namespace (Usar con extrema precaución)**
    - El comando kubectl delete all --all -n <namespace> elimina *todos* los recursos (pods, services, deployments, etc.) dentro del namespace especificado.[87] **No se recomienda** para este tutorial, ya que podría eliminar recursos inesperados, especialmente si se trabaja en el namespace default. Es mejor eliminar los recursos específicos creados.

Manos a la Obra (Limpieza de Recursos):

1. Eliminar los recursos de la aplicación Flask (usando el método 1 o 2):
   Bash
   # Usando nombres
   kubectl delete service flask-service
   kubectl delete deployment flask-app-deployment
   # O usando archivos YAML
   # kubectl delete -f flask-service.yaml
   # kubectl delete -f flask-deployment.yaml

2. Eliminar los recursos de Nginx si se crearon (imperativa o declarativamente):
   Bash
   # Ejemplo para nombres imperativos
   kubectl delete service nginx-imperative
   kubectl delete deployment nginx-imperative
   # Ejemplo para nombres declarativos
   # kubectl delete service nginx-yaml-service
   # kubectl delete deployment nginx-yaml-deployment

3. **Verificar la eliminación:** Comprobar que los recursos ya no existen.
   Bash
   kubectl get services,deployments,pods -l app=flask-app
   kubectl get services,deployments,pods -l app=nginx-imperative
   kubectl get services,deployments,pods -l app=nginx-yaml
   La salida debería ser No resources found.

8.2 Detención y Eliminación del Clúster Local

Una vez eliminados los recursos dentro del clúster, se puede detener o eliminar completamente el clúster local para liberar los recursos de la máquina host (CPU, RAM, disco).

- **Para Minikube:**
    - **Detener (minikube stop):** Detiene la máquina virtual o contenedor de Minikube, pero conserva todos los datos y la configuración del clúster.[20] Es útil si se planea reanudar el trabajo más tarde. El clúster se puede reiniciar con minikube start.
      Bash
      minikube stop

    - **Eliminar (minikube delete):** Elimina permanentemente la máquina virtual o contenedor del clúster y todos sus archivos asociados.[113] Esto libera todo el espacio en disco y los recursos. Para volver a usar Minikube, habrá que ejecutar minikube start de nuevo, lo que creará un clúster nuevo y vacío.
      Bash
      minikube delete
      Para eliminar todos los perfiles de Minikube (si se han creado múltiples clústeres con -p):
      Bash
      minikube delete --all
      [115] Para una limpieza más profunda que elimina también el directorio .minikube del home del usuario:
      Bash
      minikube delete --all --purge
      [116] (Usar con precaución).
- **Para Docker Desktop:**
    - **Deshabilitar Kubernetes:** Es la forma principal de detener el clúster gestionado por Docker Desktop.
        1. Ir a Docker Desktop **Settings** -> **Kubernetes**.
        2. Desmarcar la casilla **Enable Kubernetes**.
        3. Hacer clic en **Apply & Restart**.[25] Esto detiene y elimina los contenedores de Kubernetes que Docker Desktop estaba ejecutando, y también elimina el comando kubectl que instaló.[25] Los datos de las aplicaciones desplegadas (si usaban volúmenes persistentes gestionados por K8s) también se perderán.
    - **Resetear Clúster Kubernetes:** En **Settings** -> **Kubernetes**, también existe una opción para **Reset Kubernetes Cluster**.[25] Esto elimina todas las cargas de trabajo y configuraciones del clúster, devolviéndolo a un estado limpio inicial, pero manteniendo Kubernetes habilitado. Es útil para solucionar problemas sin tener que deshabilitar y volver a habilitar Kubernetes por completo.

Manos a la Obra (Limpieza del Clúster):

- Si se usó Minikube, ejecutar minikube stop (para pausar) o minikube delete (para eliminar).
- Si se usó Docker Desktop, ir a Settings y deshabilitar Kubernetes.

Es importante entender la diferencia entre detener y eliminar. Detener (minikube stop o

simplemente cerrar Docker Desktop sin deshabilitar K8s) preserva el estado del clúster para uso futuro, consumiendo espacio en disco pero no necesariamente CPU/RAM activa. Eliminar (minikube delete o deshabilitar K8s en Docker Desktop) elimina completamente el clúster y su estado, liberando todos los recursos asociados.

Sección 9: ¿Hacia Dónde Ir Ahora?

Se han cubierto los fundamentos de Kubernetes, desde la configuración de un entorno local en Windows hasta el despliegue y la gestión básica de aplicaciones usando kubectl y archivos YAML. Este es un excelente punto de partida, pero Kubernetes es una plataforma vasta con muchas más capacidades.

9.1 Breve Introducción a Conceptos Avanzados

A medida que se profundiza en Kubernetes, se encontrarán conceptos adicionales que son cruciales para construir aplicaciones más complejas y robustas:

- **Namespaces (Espacios de Nombres):**
  - **Propósito:** Proporcionan una forma de dividir los recursos del clúster en entornos virtuales lógicamente aislados.[119] Son útiles para organizar recursos por proyecto, equipo o entorno (ej. desarrollo, producción) dentro del mismo clúster físico.[119]
  - **Funcionamiento:** Los nombres de los recursos (como Pods, Services, Deployments) deben ser únicos *dentro* de un namespace, pero pueden repetirse *entre* namespaces.[119] Permiten aplicar políticas de control de acceso (RBAC) y cuotas de recursos por namespace.[119]
  - **Namespaces por Defecto:** Kubernetes viene con namespaces como default (donde se crean los recursos si no se especifica otro), kube-system (para componentes del sistema K8s) y kube-public (para recursos públicamente legibles).[119] Se recomienda no usar default para aplicaciones reales y crear namespaces específicos.[119]
- **ConfigMaps:**
  - **Propósito:** Permiten desacoplar los datos de configuración no sensibles (pares clave-valor) del código de la aplicación o de las imágenes de contenedor.[123] Esto facilita la gestión de la configuración en diferentes entornos.[124]
  - **Funcionamiento:** Almacenan datos como cadenas de texto.[124] Los Pods pueden consumir estos datos como variables de entorno, argumentos de línea de comandos o montándolos como archivos de configuración en un volumen.[123]
  - **Limitación:** No son para datos sensibles y tienen un límite de tamaño (aproximadamente 1 MiB).[124]
- **Secrets:**
  - **Propósito:** Diseñados específicamente para almacenar y gestionar información sensible como contraseñas, tokens OAuth, claves SSH, certificados TLS, etc..[127] Al igual que los ConfigMaps, permiten separar datos confidenciales del código de la aplicación.[129]

    - **Funcionamiento:** Almacenan datos como pares clave-valor. Los Pods pueden consumirlos como variables de entorno o montándolos como archivos en volúmenes.[128]
    - **Seguridad:** Por defecto, los datos en los Secrets se almacenan en etcd codificados en Base64, **no cifrados**.[127] Cualquiera con acceso a la API o a etcd puede leerlos. Es crucial habilitar el cifrado en reposo (Encryption at Rest) y usar RBAC para restringir el acceso a los Secrets en entornos de producción.[129] Para una gestión de secretos más robusta, a menudo se integran herramientas externas como HashiCorp Vault o gestores de secretos de proveedores de nube.[127]

9.2 Recursos para Seguir Aprendiendo

El ecosistema de Kubernetes es amplio y está en constante evolución. Aquí hay algunos recursos para continuar el aprendizaje:

- **Documentación Oficial de Kubernetes:** El sitio kubernetes.io/docs es la fuente de información más completa y actualizada. Incluye tutoriales, guías conceptuales, tareas y referencias de API y kubectl.[49]
- **Tutoriales de Kubernetes:** Buscar tutoriales específicos sobre temas como:
    - **Ingress:** Para una gestión más avanzada del tráfico entrante hacia los Services.
    - **PersistentVolumes (PV) y PersistentVolumeClaims (PVC):** Para gestionar almacenamiento persistente para aplicaciones con estado.
    - **StatefulSets:** Para gestionar aplicaciones con estado que requieren identidades de red estables y almacenamiento persistente único por instancia.
    - **Helm:** El gestor de paquetes para Kubernetes, que simplifica la definición, instalación y actualización de aplicaciones complejas.
- **Comunidad Kubernetes:** Participar en foros, grupos de usuarios o contribuir al proyecto.
- **Cursos y Certificaciones:** Considerar cursos en línea (ej. KodeKloud, A Cloud Guru) o certificaciones como Certified Kubernetes Administrator (CKA) o Certified Kubernetes Application Developer (CKAD).

9.3 Conclusión

Este documento ha proporcionado una guía práctica para comenzar con Kubernetes en un entorno Windows, asumiendo conocimientos previos de Docker y Flask. Se ha cubierto la instalación de un clúster local (Minikube y Docker Desktop), el uso de comandos kubectl esenciales para la inspección y gestión, el despliegue de aplicaciones mediante enfoques imperativos y declarativos (YAML), y la gestión básica del ciclo de vida de las aplicaciones (escalado y actualizaciones).

Se ha enfatizado la importancia del enfoque declarativo con archivos YAML para una gestión robusta y versionable, y se han proporcionado ejemplos prácticos para desplegar tanto una aplicación Nginx simple como una aplicación Flask personalizada. Finalmente, se introdujeron brevemente conceptos más avanzados como Namespaces, ConfigMaps y Secrets, que son

los siguientes pasos lógicos en el viaje de aprendizaje de Kubernetes.
La orquestación de contenedores con Kubernetes abre un mundo de posibilidades para construir, desplegar y escalar aplicaciones modernas de manera eficiente y resiliente. Se anima a los estudiantes a seguir explorando y experimentando con las capacidades de esta poderosa plataforma.

Obras citadas

1. Core concepts for Azure Kubernetes Service (AKS) - Learn Microsoft, fecha de acceso: mayo 1, 2025, https://learn.microsoft.com/en-us/azure/aks/core-aks-concepts
2. Overview | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/concepts/overview/
3. Kubernetes vs Docker: What's the difference? - Dynatrace, fecha de acceso: mayo 1, 2025, https://www.dynatrace.com/news/blog/kubernetes-vs-docker/
4. What Is the Difference Between Dockers and Kubernetes? - Palo Alto Networks, fecha de acceso: mayo 1, 2025, https://www.paloaltonetworks.com/cyberpedia/kubernetes-docker
5. Kubernetes Vs. Docker Vs. OpenShift: A 2025 Shootout - CloudZero, fecha de acceso: mayo 1, 2025, https://www.cloudzero.com/blog/kubernetes-vs-docker/
6. Kubernetes vs Docker - Difference Between Container Technologies ..., fecha de acceso: mayo 1, 2025, https://aws.amazon.com/compare/the-difference-between-kubernetes-and-docker/
7. Kubernetes vs Docker: 5 Differences You Should Know - K21Academy, fecha de acceso: mayo 1, 2025, https://k21academy.com/docker-kubernetes/kubernetes-vs-docker/
8. Kubernetes vs. Docker | Atlassian, fecha de acceso: mayo 1, 2025, https://www.atlassian.com/microservices/microservices-architecture/kubernetes-vs-docker
9. Learn Kubernetes Basics, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/tutorials/kubernetes-basics/
10. Kubernetes: Core Concepts | CNCF, fecha de acceso: mayo 1, 2025, https://www.cncf.io/blog/2019/05/10/kubernetes-core-concepts/
11. Kubernetes Tutorial for Beginners: Basic Concepts - Spacelift, fecha de acceso: mayo 1, 2025, https://spacelift.io/blog/kubernetes-tutorial
12. Docker Swarm vs Kubernetes: how to choose a container orchestration tool - CircleCI, fecha de acceso: mayo 1, 2025, https://circleci.com/blog/docker-swarm-vs-kubernetes/
13. Kubernetes concepts - Amazon EKS - AWS Documentation, fecha de acceso: mayo 1, 2025, https://docs.aws.amazon.com/eks/latest/userguide/kubernetes-concepts.html
14. Kubernetes Core Concepts - Services - Learn Code Online, fecha de acceso: mayo 1, 2025, https://blog.learncodeonline.in/kubernetes-core-concepts-services
15. Kubernetes vs Docker - A Detailed Comparison - Refine, fecha de acceso: mayo

1, 2025, https://refine.dev/blog/kubernetes-vs-docker/
16. Concepts - Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/concepts/
17. Deployments | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
18. Civo Academy - Setting up a local cluster using Minikube - Civo.com, fecha de acceso: mayo 1, 2025, https://www.civo.com/academy/kubernetes-setup/install-minikube
19. Getting started with Docker and Kubernetes on Windows 10 - Learnk8s, fecha de acceso: mayo 1, 2025, https://learnk8s.io/installing-docker-kubernetes-windows
20. Developing for Kubernetes with minikube - GitLab Docs, fecha de acceso: mayo 1, 2025, https://docs.gitlab.com/charts/development/minikube/
21. Install Minikube - Kubernetes, fecha de acceso: mayo 1, 2025, https://k8s-docs.netlify.app/en/docs/tasks/tools/install-minikube/
22. Install Minikube on Windows for Kubernetes: Quick Setup Guide - Ambassador Labs, fecha de acceso: mayo 1, 2025, https://www.getambassador.io/blog/kubernetes-with-minikube-on-windows-home
23. How to Install Kubernetes MiniKube - Altaro, fecha de acceso: mayo 1, 2025, https://www.altaro.com/msp-dojo/install-kubernetes-minikube/
24. hyperv - Minikube - Kubernetes, fecha de acceso: mayo 1, 2025, https://minikube.sigs.k8s.io/docs/drivers/hyperv/
25. Deploy on Kubernetes | Docker Docs, fecha de acceso: mayo 1, 2025, https://docs.docker.com/desktop/features/kubernetes/
26. minikube start | minikube, fecha de acceso: mayo 1, 2025, https://minikube.sigs.k8s.io/docs/start/
27. Getting Started with Kubernetes on Docker Desktop, fecha de acceso: mayo 1, 2025, https://birthday.play-with-docker.com/kubernetes-docker-desktop/
28. Choosing the Right Tool for Your Local Kubernetes Development Environment, fecha de acceso: mayo 1, 2025, https://everythingdevops.dev/choosing-the-right-tool-for-your-local-kubernetes-development-environment/
29. Hello Minikube - Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/tutorials/hello-minikube/
30. How to Set Up a Kubernetes Cluster on Docker Desktop, fecha de acceso: mayo 1, 2025, https://www.docker.com/blog/how-to-set-up-a-kubernetes-cluster-on-docker-desktop/
31. How to install Docker Desktop and enable Kubernetes support - TechRepublic, fecha de acceso: mayo 1, 2025, https://www.techrepublic.com/article/how-to-install-docker-desktop-kubernetes-support/
32. Minikube vs Docker Desktop Kubernetes? - General Discussions, fecha de acceso: mayo 1, 2025, https://discuss.kubernetes.io/t/minikube-vs-docker-desktop-kubernetes/23629

33. Install Docker Desktop on Windows, fecha de acceso: mayo 1, 2025, https://docs.docker.com/desktop/setup/install/windows-install/
34. Minikube vs Kubernetes in Docker for Windows - Stack Overflow, fecha de acceso: mayo 1, 2025, https://stackoverflow.com/questions/51209870/minikube-vs-kubernetes-in-docker-for-windows
35. Minikube vs. Docker Desktop for Local Development - Adib Saikali, fecha de acceso: mayo 1, 2025, https://adibsaikali.com/2020/07/25/minikube-vs-docker-desktop-for-local-development/
36. Minikube vs. k3d vs. kind vs. Getdeck - blueshoe, fecha de acceso: mayo 1, 2025, https://www.blueshoe.io/blog/minikube-vs-k3d-vs-kind-vs-getdeck-beiboot/
37. Goodbye Docker Desktop, Hello Minikube! : r/kubernetes - Reddit, fecha de acceso: mayo 1, 2025, https://www.reddit.com/r/kubernetes/comments/pjlt52/goodbye_docker_desktop_hello_minikube/
38. Local Kubernetes Clusters: A Comparison for Local Development and CI | Senacor Blog, fecha de acceso: mayo 1, 2025, https://senacor.blog/local-kubernetes-clusters-a-comparison-for-local-development-and-ci/
39. Command line tool (kubectl) - Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/reference/kubectl/
40. Kubectl Cheat Sheet - 15 Kubernetes Commands & Objects - Spacelift, fecha de acceso: mayo 1, 2025, https://spacelift.io/blog/kubernetes-cheat-sheet
41. Using kubectl to Create a Deployment - Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/tutorials/kubernetes-basics/deploy-app/deploy-intro/
42. Kubernetes-Kubectl-CLI-Cheat-Sheet.pdf - Altoros, fecha de acceso: mayo 1, 2025, https://www.altoros.com/wp-content/uploads/pdf/Kubernetes-Kubectl-CLI-Cheat-Sheet.pdf
43. Kubectl Cheat Sheet: 12 Kubectl Commands with Examples - Middleware Observability, fecha de acceso: mayo 1, 2025, https://middleware.io/blog/kubectl-cheat-sheet/
44. Display Kubernetes Cluster Information - LabEx, fecha de acceso: mayo 1, 2025, https://labex.io/tutorials/kubernetes-kubernetes-cluster-information-8426
45. Kubectl Demystified: Mastering the `kubectl cluster-info` Command - DEV Community, fecha de acceso: mayo 1, 2025, https://dev.to/naveens16/kubectl-demystified-mastering-the-kubectl-cluster-info-command-4m70
46. kubectl cluster-info | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/reference/kubectl/generated/kubectl_cluster-info/
47. kubectl Quick Reference | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/reference/kubectl/cheatsheet/
48. What is Kubectl Get Nodes and How to Use It - Spacelift, fecha de acceso: mayo

1, 2025, https://spacelift.io/blog/kubectl-get-nodes
49. Kubectl Cheat Sheet - Kubernetes Commands (Basic to Advanced) - StrongDM, fecha de acceso: mayo 1, 2025, https://www.strongdm.com/blog/kubernetes-cheat-sheet
50. kubectl Cheat Sheet - Kubernetes, fecha de acceso: mayo 1, 2025, https://jamesdefabia.github.io/docs/user-guide/kubectl-cheatsheet/
51. kubectl - Kubernetes API - Get Pods on Specific Nodes - Stack Overflow, fecha de acceso: mayo 1, 2025, https://stackoverflow.com/questions/39231880/kubernetes-api-get-pods-on-specific-nodes
52. kubectl Cheat Sheet - Kubernetes - Confluence, fecha de acceso: mayo 1, 2025, https://confluence.appstate.edu/display/K8S/kubectl+Cheat+Sheet
53. Imperative vs. declarative Kubernetes commands: What's the difference? - The Server Side, fecha de acceso: mayo 1, 2025, https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Imperative-vs-declarative-Kubernetes-commands-Whats-the-difference
54. Understanding Declarative vs. Imperative Approaches in Kubernetes - Shashank's blog, fecha de acceso: mayo 1, 2025, https://jamy.hashnode.dev/understanding-declarative-vs-imperative-approaches-in-kubernetes
55. Kubernetes - What exactly is imperative Vs Declarative - Stack Overflow, fecha de acceso: mayo 1, 2025, https://stackoverflow.com/questions/60242051/kubernetes-what-exactly-is-imperative-vs-declarative
56. K8s 1.17 - Imperative vs Declarative - Steven McGown, fecha de acceso: mayo 1, 2025, https://smcgown.com/blog/kubernetes/1-17-imperative-vs-declarative/
57. Imperative vs. Declarative in Kubernetes - KubeOps, fecha de acceso: mayo 1, 2025, https://kubeops.net/blog/imperative-vs-declarative
58. kubectl create deployment | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/reference/kubectl/generated/kubectl_create/kubectl_create_deployment/
59. Kubernetes Imperative Commands Every Engineer Should Learn, fecha de acceso: mayo 1, 2025, https://luispreciado.blog/posts/kubernetes/core-concepts/imperative-commands
60. Kubectl imperative command for deployment - Stack Overflow, fecha de acceso: mayo 1, 2025, https://stackoverflow.com/questions/61113942/kubectl-imperative-command-for-deployment
61. kubectl expose | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/reference/kubectl/generated/kubectl_expose/
62. Accessing apps | minikube, fecha de acceso: mayo 1, 2025, https://minikube.sigs.k8s.io/docs/handbook/accessing/
63. How to access NodePort in Minikube with docker driver? - Stack Overflow, fecha de acceso: mayo 1, 2025, https://stackoverflow.com/questions/62539604/how-to-access-nodeport-in-mini

kube-with-docker-driver
64. Module 4 - Expose your app publicly - Minikube - Kubernetes, fecha de acceso: mayo 1, 2025, https://minikube.sigs.k8s.io/docs/tutorials/kubernetes_101/module4/
65. Solved: Minikube nodeport service and port forwarding - Google Cloud Community, fecha de acceso: mayo 1, 2025, https://www.googlecloudcommunity.com/gc/Developer-Tools/Minikube-nodeport-service-and-port-forwarding/m-p/615566
66. Build and deploy a Flask application - HSRN Kubernetes Cluster - NYU, fecha de acceso: mayo 1, 2025, https://k8s-docs.hsrn.nyu.edu/flask-service/
67. How to Use Kubectl Port-forward in Kubernetes Applications - Spacelift, fecha de acceso: mayo 1, 2025, https://spacelift.io/blog/kubectl-port-forward
68. Kubernetes Port Forwarding: A Practical Guide - Plural.sh, fecha de acceso: mayo 1, 2025, https://www.plural.sh/blog/kubernetes-port-forward-guide/
69. kubectl port-forward: Kubernetes Port Forwarding Guide - phoenixNAP, fecha de acceso: mayo 1, 2025, https://phoenixnap.com/kb/kubectl-port-forward
70. kubectl port-forward | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/reference/kubectl/generated/kubectl_port-forward/
71. How to Use Kubectl Port-Forward to Create a Connection & More - Komodor, fecha de acceso: mayo 1, 2025, https://komodor.com/learn/kubectl-port-forwarding-how-it-works-use-cases-examples/
72. How to Create a Kubernetes Deployment using YAML - Mirantis, fecha de acceso: mayo 1, 2025, https://www.mirantis.com/blog/introduction-to-yaml-creating-a-kubernetes-deployment/
73. kubectl apply: Syntax, Examples, kubectl apply vs. create/replace - Komodor, fecha de acceso: mayo 1, 2025, https://komodor.com/learn/kubectl-apply-syntax-examples-and-kubectl-apply-vs-create-vs-replace/
74. Kubernetes Deployment YAML File with Examples - Spacelift, fecha de acceso: mayo 1, 2025, https://spacelift.io/blog/kubernetes-deployment-yaml
75. Complete Kubernetes Deployment Template Guide (With Example YAML) | Zeet.co, fecha de acceso: mayo 1, 2025, https://zeet.co/blog/kubernetes-deployment-template
76. Service | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/concepts/services-networking/service/
77. Understand Kubernetes Services | GKE networking - Google Cloud, fecha de acceso: mayo 1, 2025, https://cloud.google.com/kubernetes-engine/docs/concepts/service
78. Service - Kubernetes examples, fecha de acceso: mayo 1, 2025, https://k8s-examples.container-solutions.com/examples/Service/Service.html
79. Kubernetes – NodePort Service - GeeksforGeeks, fecha de acceso: mayo 1, 2025, https://www.geeksforgeeks.org/kubernetes-nodeport-service/
80. How to expose nginx on public Ip using NodePort service in Kubernetes? - Stack

Overflow, fecha de acceso: mayo 1, 2025, https://stackoverflow.com/questions/48857092/how-to-expose-nginx-on-public-ip-using-nodeport-service-in-kubernetes
81. Kubectl Apply vs. Kubectl Create - What's the Difference? - Spacelift, fecha de acceso: mayo 1, 2025, https://spacelift.io/blog/kubectl-apply-vs-create
82. Kubectl Apply vs. Create: Understanding the Difference - Devtron, fecha de acceso: mayo 1, 2025, https://devtron.ai/blog/kubectl-apply-vs-create/
83. kubectl apply | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/reference/kubectl/generated/kubectl_apply/
84. What is the difference between kubectl apply and replace?? : r/kubernetes - Reddit, fecha de acceso: mayo 1, 2025, https://www.reddit.com/r/kubernetes/comments/z8pvsm/what_is_the_difference_between_kubectl_apply_and/
85. Kubectl Apply Vs Create - Startup House, fecha de acceso: mayo 1, 2025, https://startup-house.com/glossary/kubectl-apply-vs-create
86. kubectl delete | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/reference/kubectl/generated/kubectl_delete/
87. Kubernetes – Kubectl Delete - GeeksforGeeks, fecha de acceso: mayo 1, 2025, https://www.geeksforgeeks.org/kubernetes-kubectl-delete/
88. Kubernetes - Delete Resources - Learning-Ocean, fecha de acceso: mayo 1, 2025, https://learning-ocean.com/tutorials/kubernetes/kubernetes-delete-resources/
89. How to Use Kubectl Delete Deployment in Kubernetes - Spacelift, fecha de acceso: mayo 1, 2025, https://spacelift.io/blog/kubectl-delete-deployment
90. Remove deployment and service - IBM, fecha de acceso: mayo 1, 2025, https://www.ibm.com/docs/en/zoscp/1.1.0?topic=kubernetes-8-remove-deployment-service
91. How to Deploy a Flask Application to Kubernetes - Aahil's Blog, fecha de acceso: mayo 1, 2025, https://aahil13.hashnode.dev/how-to-deploy-a-flask-application-to-kubernetes
92. How to Deploy Flask App on Kubernetes? | GeeksforGeeks, fecha de acceso: mayo 1, 2025, https://www.geeksforgeeks.org/how-to-deploy-flask-app-on-kubernetes/
93. Deploying flask app to Kubernetes using Minikube - DEV Community, fecha de acceso: mayo 1, 2025, https://dev.to/joykingoo/deploying-flask-app-to-kubernetes-using-minikube-6jl
94. Learning Project - Deploy Flask App With MySQL on Kubernetes - Reddit, fecha de acceso: mayo 1, 2025, https://www.reddit.com/r/kubernetes/comments/1iu92sy/learning_project_deploy_flask_app_with_mysql_on/
95. ginomempin/sample-flask-with-kubernetes - GitHub, fecha de acceso: mayo 1, 2025, https://github.com/ginomempin/sample-flask-with-kubernetes
96. Deploy a Python To-Do App with Flask, Kubernetes, and CockroachDB Cloud, fecha de acceso: mayo 1, 2025, https://www.cockroachlabs.com/docs/cockroachcloud/deploy-a-python-to-do-a

pp-with-flask-kubernetes-and-cockroachcloud
97. noahgift/kubernetes-hello-world-python-flask - GitHub, fecha de acceso: mayo 1, 2025, https://github.com/noahgift/kubernetes-hello-world-python-flask
98. Deploying Flask Applications in Kubernetes (Digitalocean) | Bhavani's Digital Garden, fecha de acceso: mayo 1, 2025, https://www.bhavaniravi.com/devops/kubernetes-101-deploy-apps-in-kubernetes
99. Using a Service to Expose Your App - Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/tutorials/kubernetes-basics/expose/expose-intro/
100. Kubectl Scale Deployment: Examples & Best Practices - Groundcover, fecha de acceso: mayo 1, 2025, https://www.groundcover.com/blog/kubectl-scale
101. How to Use Kubectl Scale on Deployment - KodeKloud, fecha de acceso: mayo 1, 2025, https://kodekloud.com/blog/kubectl-scale/
102. Mastering kubectl Scale Deployment: A Comprehensive Guide for Developers - StackState, fecha de acceso: mayo 1, 2025, https://www.stackstate.com/blog/mastering-kubectl-scale-deployment-a-comprehensive-guide-for-developers/
103. kubectl scale | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/reference/kubectl/generated/kubectl_scale/
104. Kubectl Scale Deployment to 0 - stormforge.io, fecha de acceso: mayo 1, 2025, https://stormforge.io/kubernetes-autoscaling/kubectl-scale-deployment-to-0/
105. Performing a Rolling Update | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/tutorials/kubernetes-basics/update/update-intro/
106. 8 Kubernetes Deployment Strategies - Spot.io, fecha de acceso: mayo 1, 2025, https://spot.io/resources/kubernetes-autoscaling/8-kubernetes-deployment-strategies/
107. Kubernetes Rolling Deployment: A Practical Guide - Codefresh, fecha de acceso: mayo 1, 2025, https://codefresh.io/learn/software-deployment/kubernetes-rolloing-deployment-a-practical-guide/
108. Rolling Updates and Rollbacks in Kubernetes: Managing Application Updates, fecha de acceso: mayo 1, 2025, https://www.geeksforgeeks.org/rolling-updates-and-rollbacks-in-kubernetes-managing-application-updates/
109. Complete Guide On Kubernetes Update Deployment: What Is It & How To Use It | Zeet.co, fecha de acceso: mayo 1, 2025, https://zeet.co/blog/kubernetes-update-deployment
110. How to Use Kubectl Delete Deployment (With Examples) - KodeKloud, fecha de acceso: mayo 1, 2025, https://kodekloud.com/blog/kubectl-delete-deployment/
111. How to Delete Kubernetes Service - kubectl - Stack Overflow, fecha de acceso: mayo 1, 2025, https://stackoverflow.com/questions/47693911/how-to-delete-kubernetes-service

112. Deleting a Service or Deployment - Oracle Help Center, fecha de acceso: mayo 1, 2025, https://docs.oracle.com/en/operating-systems/olcne/2.0/kubernetes/kubectl_delete_task.html
113. Stopping and deleting minikube - O'Reilly Media, fecha de acceso: mayo 1, 2025, https://www.oreilly.com/library/view/blockchain-for-enterprise/9781788479745/a65ecea0-944d-4db8-a285-fe9940bed8cc.xhtml
114. stop | minikube, fecha de acceso: mayo 1, 2025, https://minikube.sigs.k8s.io/docs/commands/stop/
115. Basic controls - Minikube - Kubernetes, fecha de acceso: mayo 1, 2025, https://minikube.sigs.k8s.io/docs/handbook/controls/
116. delete | minikube, fecha de acceso: mayo 1, 2025, https://minikube.sigs.k8s.io/docs/commands/delete/
117. How to completely remove Kubernetes after disabling it from Docker Desktop?, fecha de acceso: mayo 1, 2025, https://stackoverflow.com/questions/76207853/how-to-completely-remove-kubernetes-after-disabling-it-from-docker-desktop
118. How do I turn it off? : r/kubernetes - Reddit, fecha de acceso: mayo 1, 2025, https://www.reddit.com/r/kubernetes/comments/17gbvds/how_do_i_turn_it_off/
119. Namespaces | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/
120. What is a Namespace in Kubernetes? - Veeam, fecha de acceso: mayo 1, 2025, https://www.veeam.com/glossary/kubernetes-namespace.html
121. Kubernetes best practices: Specifying Namespaces in YAML | Google Cloud Blog, fecha de acceso: mayo 1, 2025, https://cloud.google.com/blog/products/containers-kubernetes/kubernetes-best-practices-organizing-with-namespaces
122. Namespaces Walkthrough - Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/tasks/administer-cluster/namespaces-walkthrough/
123. Configure a Pod to Use a ConfigMap - Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/
124. ConfigMaps | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/concepts/configuration/configmap/
125. In-Depth Guide to Kubernetes ConfigMap & Secret Management Strategies, fecha de acceso: mayo 1, 2025, https://www.getambassador.io/blog/kubernetes-configurations-secrets-configmaps
126. What is Kubernetes ConfigMap? - Sysdig, fecha de acceso: mayo 1, 2025, https://sysdig.com/learn-cloud-native/what-is-kubernetes-configmap/
127. Kubernetes External Secrets: Tutorial & Instructions - Kubecost, fecha de acceso: mayo 1, 2025,

https://www.kubecost.com/kubernetes-devops-tools/kubernetes-external-secrets/
128. Kubernetes Secrets in KubeSphere, fecha de acceso: mayo 1, 2025, https://kubesphere.io/docs/v3.4/project-user-guide/configuration/secrets/
129. Secrets | Kubernetes, fecha de acceso: mayo 1, 2025, https://kubernetes.io/docs/concepts/configuration/secret/