Documentación del Proceso: Desarrollar y Desplegar una API Web con PHP en Docker y Kubernetes.

1. Crear y empaquetar la API en una imagen Docker

Paso 1: Preparar la aplicación PHP para Docker

1. **Archivos Necesarios:**
    - **index.php**: Archivo principal que contiene la lógica de la API.
    - **Dockerfile**: Archivo que define la construcción de la imagen Docker para la API.
    - **composer. json** (si es necesario): Si usas librerías como JWT, asegúrate de tener este archivo para gestionar dependencias.

Paso 2: Crear el archivo Dockerfile

El Dockerfile define el entorno de ejecución de la aplicación en un contenedor. El siguiente es un ejemplo de cómo crear un Dockerfile para tu aplicación PHP:

Dockerfile

{

Usamos una imagen oficial de PHP 8.2.12 con Apache

FROM php:8.2.12-apache

Instalar dependencias necesarias para PHP

RUN apt-get update && apt-get install -y libpng-dev libjpeg-dev libfreetype6-dev git unzip

Habilitar mod_rewrite para Apache (requerido por muchas aplicaciones PHP)

RUN a2enmod rewrite

Copiar los archivos de la aplicación al contenedor

COPY. /var/www/html/

Instalar Composer para gestionar dependencias

RUN curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer

WORKDIR /var/www/html

Instalar dependencias PHP (si usas librerías como JWT)

RUN composer install

Exponer el puerto 80 para que Apache pueda servir la aplicación

EXPOSE 80

}

Paso 3: Construir la imagen Docker

Una vez que tengas el Dockerfile, puedes construir la imagen Docker ejecutando el siguiente comando en el directorio donde se encuentra el Dockerfile:

Docker build -t api .

Paso 4: Ejecutar la aplicación en Docker

Para ejecutar la imagen que acabas de construir en un contenedor, usa el siguiente comando:

Docker run -p 8000:80 mi-aplicacion-php

Puedes verificar el funcionamiento accediendo a http://localhost:8000.

2. Configurar el clúster de Kubernetes para desplegar las réplicas de la API con balanceo de carga

Paso 1: Desplegar la aplicación en Kubernetes

Aplica el archivo de despliegue en tu clúster de Kubernetes ejecutando el siguiente comando:

```
kubectl apply -f k8s/deployment.yaml
```

Para verificar que las réplicas se han desplegado correctamente, ejecuta:

```
kubectl get pods
```

Paso 3: Verificar el balanceo de carga

Paso 2: Aplicar la configuración de autoscaling

Aplica el archivo de autoscaling al clúster de Kubernetes con el siguiente comando:

```
kubectl apply -f k8s/hpa.yaml
```

Para verificar el estado del escalado horizontal, puedes ejecutar:

```
kubectl get hpa
```

4. Balanceo de carga y seguridad

Cómo el balanceador de carga gestiona el tráfico

El **balanceador de carga** en Kubernetes distribuye las solicitudes de los usuarios entre las distintas réplicas de la API. Si se usa un servicio de tipo LoadBalancer, Kubernetes crea un balanceador de carga externo (en la nube) que asigna una IP pública y distribuye las solicitudes entre los pods.

1. **Gestión de tráfico entre réplicas**: Kubernetes asegura que las solicitudes se distribuyan equitativamente entre las réplicas para maximizar la disponibilidad y evitar sobrecargar un solo pod.
2. **Alta disponibilidad**: Si una réplica falla, el balanceador de carga redirige automáticamente el tráfico a las réplicas que están funcionando, lo que garantiza la disponibilidad continua de la API.

Seguridad en la API

Para garantizar el acceso seguro a los datos expuestos por la API, implementamos un sistema de **autenticación JWT**:

1. **Verificación del token JWT**: En cada solicitud a la API, se requiere un token JWT válido en el encabezado de autorización. Si el token no está presente o es inválido, la API devolverá un error 403 (No autorizado).
2. **Protección de endpoints sensibles**: Solo los usuarios autenticados con un token válido podrán acceder a los endpoints protegidos de la API. El código PHP en el archivo index.php se encarga de verificar este token antes de procesar cualquier solicitud.