Introducción:

El proyecto tiene como objetivo principal la personalización y el despliegue de una aplicación básica en React utilizando Kubernetes. A través de este proyecto, aplicaremos los conceptos aprendidos hasta ahora y pondremos en práctica habilidades relacionadas con el control de versiones, personalización de aplicaciones, despliegue en un cluster de Kubernetes y documentación de referencia.

Nuestro proyecto se basa en un archivo ZIP que contiene un proyecto básico en React. Utilizaremos este proyecto como punto de partida y lo personalizaremos para adaptarlo a nuestras necesidades. Esto implica modificar elementos de la aplicación, como el texto y otros elementos visuales, para que reflejen nuestra identidad y estilo.

Luego, desplegaremos la aplicación personalizada en un cluster de Kubernetes utilizando el Google Kubernetes Engine. Este entorno nos permitirá gestionar y escalar la aplicación de manera eficiente y confiable. Utilizaremos los conocimientos adquiridos durante el curso y nos apoyaremos en el laboratorio 4 como referencia.

Además del despliegue de la aplicación, crearemos un repositorio en GitHub para controlar el código fuente y el historial de versiones de la aplicación. Esto nos permitirá realizar un seguimiento de los cambios realizados y colaborar de manera efectiva con otros desarrolladores si es necesario.

Finalmente, documentaremos el proyecto en un informe de referencia. Este informe contendrá diagramas explicativos del flujo de la aplicación y los componentes utilizados, así como también incluirá información sobre los servicios en la nube empleados en el despliegue de la aplicación. También proporcionaremos los pasos detallados para desplegar la aplicación, lo cual servirá como guía para futuros desarrollos y despliegues similares.

El proyecto tiene como objetivo fortalecer nuestros conocimientos en el uso de herramientas de control de versiones, personalización de aplicaciones, despliegue en Kubernetes y documentación de proyectos. Al finalizar, obtendremos una aplicación personalizada y desplegada en un entorno de Kubernetes, así como también un informe completo que servirá como referencia para el proyecto y futuros trabajos similares.

Flujo de la Aplicación:

El flujo de la aplicación sigue una estructura cliente-servidor, donde el cliente interactúa con la interfaz de usuario y realiza solicitudes al servidor para obtener respuestas o realizar acciones específicas. A continuación, se describe el flujo paso a paso:

**Interacción del Usuario:**

* El usuario accede a la aplicación a través de un navegador web.
* La aplicación muestra una interfaz de usuario donde el usuario puede interactuar y realizar acciones.

**Renderizado Inicial:**

* Cuando el usuario accede a la aplicación, se carga la página inicial.
* El navegador solicita los recursos necesarios al servidor, como archivos HTML, CSS y JavaScript.
* El servidor envía los recursos solicitados al navegador.

**Interacción del Usuario:**

* El usuario interactúa con la interfaz de usuario, como hacer clic en botones, ingresar datos en formularios, etc.
* Estas interacciones generan eventos en el navegador.

**Manejo de Eventos:**

* El JavaScript en el cliente maneja los eventos generados por la interacción del usuario.
* Por ejemplo, al hacer clic en un botón, se activa un evento de clic.

**Solicitud al Servidor:**

* Cuando ocurre un evento que requiere una interacción con el servidor, el cliente realiza una solicitud al servidor.
* La solicitud se realiza a través de una API RESTful u otro mecanismo de comunicación.

**Procesamiento en el Servidor:**

* El servidor recibe la solicitud del cliente y procesa la información.
* Puede involucrar acciones como acceder a una base de datos, realizar cálculos o interactuar con otros servicios.

**Respuesta del Servidor:**

* El servidor genera una respuesta a la solicitud del cliente.
* La respuesta puede ser en forma de datos, como JSON, o una actualización en la interfaz de usuario.

**Actualización de la Interfaz de Usuario:**

* El cliente recibe la respuesta del servidor y actualiza la interfaz de usuario en consecuencia.
* Puede incluir cambios en el contenido, estado o apariencia de la interfaz.

**Ciclo Continuo:**

* El flujo de interacción del usuario y comunicación con el servidor continúa mientras el usuario utiliza la aplicación.
* Los eventos generados por el usuario, las solicitudes al servidor y las respuestas se repiten según sea necesario.

Diagrama de Flujo (ejemplo visual):



Este flujo ilustra el ciclo continuo de interacción entre el usuario, el cliente y el servidor. Cada interacción del usuario desencadena eventos que se procesan en el cliente y se comunican al servidor para obtener respuestas o realizar acciones específicas.

**Componentes utilizados en la aplicación:**

React: es una biblioteca de JavaScript utilizada para construir interfaces de usuario interactivas y reactivas, funciona utilizando componentes reutilizables que representan diferentes partes de la interfaz de usuario, permite crear interfaces de usuario dinámicas y de alto rendimiento al utilizar un enfoque basado en componentes y la manipulación eficiente del DOM.

Node.js: es un entorno de tiempo de ejecución de JavaScript del lado del servidor, permite ejecutar JavaScript en el servidor y construir aplicaciones web escalables y de alto rendimiento, utiliza un modelo de E/S sin bloqueo y basado en eventos, lo que lo hace adecuado para aplicaciones en tiempo real y de alta concurrencia.

Kubernetes: es una plataforma de orquestación de contenedores de código abierto, permite gestionar y desplegar aplicaciones en contenedores de manera eficiente y confiable, Kubernetes se encarga de la administración de los recursos, la escalabilidad, la recuperación ante fallas y la gestión del tráfico de las aplicaciones desplegadas.

**Interacción entre los componentes:**

React y Node.js:

React se utiliza para construir la interfaz de usuario en el lado del cliente.

Node.js se utiliza en el servidor para manejar las solicitudes del cliente y proporcionar los datos necesarios para la interfaz de usuario.

React y Node.js interactúan a través de solicitudes HTTP o API RESTful para intercambiar datos y mantener la interfaz de usuario actualizada.

**Kubernetes y la aplicación:**

Kubernetes se utiliza para orquestar y desplegar la aplicación en un entorno de contenedores.

La aplicación React y el servidor Node.js se empaquetan en contenedores y se despliegan en un clúster de Kubernetes.

Kubernetes se encarga de la administración de los contenedores, la gestión de la escala, la alta disponibilidad y la distribución del tráfico a través de los diferentes componentes de la aplicación.

En resumen, React se encarga de construir la interfaz de usuario en el lado del cliente, Node.js maneja las solicitudes del cliente y proporciona los datos necesarios, y Kubernetes se encarga de la orquestación y el despliegue de la aplicación en un entorno de contenedores. Estos componentes trabajan juntos para proporcionar una aplicación web interactiva y escalable.

Diagrama que visualiza los servicios de computación en la nube utilizados, centrándose en Google Kubernetes Engine (GKE):



En este diagrama, la aplicación React se empaqueta en un contenedor Docker. Luego, se despliega en Google Kubernetes Engine (GKE), que es un servicio de orquestación de contenedores proporcionado por Google Cloud Platform.

GKE utiliza un clúster de Kubernetes, que consiste en un grupo de nodos. Estos nodos son instancias de máquinas virtuales (VMs) que se crean y administran automáticamente por GKE.

Finalmente, los nodos GKE se ejecutan en recursos de computación en la nube, que pueden ser máquinas virtuales (VMs) escalables y administradas por el proveedor de la nube.

En resumen, la aplicación React se empaqueta en un contenedor Docker y se despliega en GKE. GKE se encarga de la orquestación y administración de los contenedores en un clúster de Kubernetes, que se ejecuta en nodos GKE que a su vez se ejecutan en recursos de computación en la nube. Esto proporciona una infraestructura escalable y confiable para ejecutar la aplicación en la nube.

**URL de la aplicación desplegada en el cluster de Kubernetes:**

**URL del repositorio en GitHub donde se encuentra el código fuente de la aplicación.:**

[**https://github.com/Edd03052601/ProyCloudComputing/releases/tag/ProyectoCloudComputing\_V1**](https://github.com/Edd03052601/ProyCloudComputing/releases/tag/ProyectoCloudComputing_V1)

A continuación, se proporcionan los pasos detallados para desplegar la aplicación en un clúster de Google Kubernetes Engine (GKE). Asegúrate de haber configurado previamente un proyecto en Google Cloud Platform y tener acceso a GKE:

1. Configuración inicial:

* Crea un nuevo proyecto en Google Cloud Platform.
* Instala el SDK de Google Cloud en tu máquina local y configura las credenciales de tu proyecto utilizando el comando gcloud auth login.

1. Creación del clúster de GKE:

* Abre una terminal y asegúrate de estar autenticado con el SDK de Google Cloud.

Crea un clúster de GKE con el siguiente comando:

gcloud container clusters create nombre-cluster --num-nodes=2 --zone=zona-elegida

Reemplaza nombre-cluster con el nombre que desees para tu clúster y zona-elegida con la zona de tu preferencia.

1. Configuración de kubectl:

* Instala y configura kubectl, la herramienta de línea de comandos para interactuar con clústeres de Kubernetes.

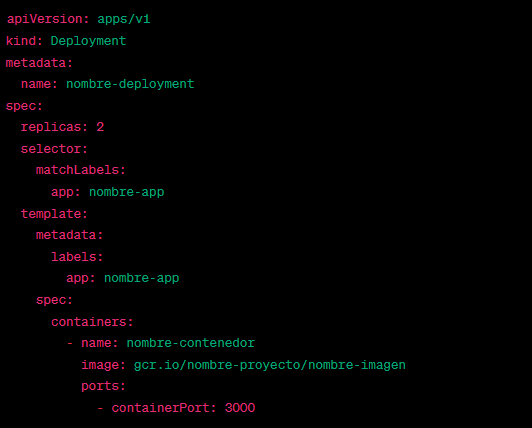
Ejecuta el siguiente comando para obtener las credenciales de tu clúster de GKE y configurar kubectl:

gcloud container clusters get-credentials nombre-cluster --zone=zona-elegida

Asegúrate de reemplazar nombre-cluster y zona-elegida con los valores correspondientes.

1. Creación de archivos de configuración:

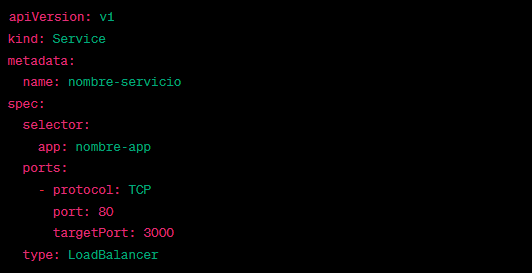
* Crea un archivo deployment.yaml con la siguiente configuración:



Reemplaza nombre-deployment, nombre-app, nombre-contenedor, nombre-proyecto y nombre-imagen con los valores correspondientes. Este archivo define un despliegue de Kubernetes para la aplicación React.

1. Creación del servicio:

* Crea un archivo service.yaml con la siguiente configuración:



Reemplaza nombre-servicio y nombre-app con los valores correspondientes. Este archivo define un servicio de Kubernetes para enrutar el tráfico hacia los pods del despliegue.

1. Despliegue de la aplicación:

* En la terminal, ejecuta los siguientes comandos para desplegar la aplicación:

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

Esto creará los recursos de despliegue y servicio en tu clúster de GKE.

1. Verificación del despliegue:

* Ejecuta el comando kubectl get services para obtener la dirección IP externa del servicio. Espera hasta que la columna "EXTERNAL-IP" muestre una dirección IP válida.
* Accede a la aplicación utilizando la dirección IP externa en tu navegador web. Deberías poder ver la aplicación React desplegada correctamente.

¡Listo! Has desplegado la aplicación en GKE utilizando Kubernetes. Asegúrate de personalizar y ajustar los nombres y configuraciones según tus necesidades específicas.