Proyecto 2 - Journal Search Platform (JSP)

Tecnológico de Costa Rica Escuela de Ingeniería en Computación Bases de Datos II (IC 4302) Segundo Semestre 2022

> *Gabriela Gutiérrez Valverde - 2019024089 Brayan Marín Quirós - 2019180348 David Achoy Yakimova - 2020053336*

Guía de Instalación y uso:

Para iniciar es necesario que se tenga instalado Docker, con Kubernetes activo y debemos asegurarnos que tenga la configuración con Linux, esto es solamente seguir las instrucciones de instalación. Y Helm, ya que esto es necesario para poder ejecutar los comandos. Además, utilizamos Lens para trabajar con más facilidad Kubernetes.

Ya con esto debemos abrir la consola de comandos. En el caso de Windows trabajamos con el PowerShell, como el proyecto ya está creado, no es necesario volver a ejecutar los comandos de "Helm create …". Por lo tanto, el siguiente paso es ejecutar los comandos de instalación de las bases de datos, estos son los siguientes:

Para ingresar a la carpeta del proyecto

```
cd "C:\...\Proyectos\Proyecto2"
```

Debemos realizar la instalación de los motores de monitoreo, con el siguiente comando:

```
helm install monitoring monitoring
```

Y una vez que tenemos el sistema de monitoreo debemos instalar las bases de datos con el comando:

```
helm install databases databases
```

Ya con esto contamos con la instalación de Prometheus, Grafana, RabbitMQ, Elastic Cloud on Kubernetes y MariaDB.

Ahora se puede utilizar la aplicación de Docker Desktop para ver los contenedores que se han creado. Y con la aplicación de Lens es posible consultar los pods, secrets, services, entre otros.

Lo siguiente es instalar la aplicación, es decir, el conjunto de archivos Python que se ejecutan como imágenes de Docker. Para lo que se utiliza el comando:

```
helm install application application
```

Al ejecutar este comando comienza a correr la aplicación con los pods del API, Loader, downloader, details downloader, jatsxml downloader.

Para el monitoreo:

Debemos ingresar a la interfaz de Grafana con estos credenciales:

USER	PASSWORD
admin	11111111

Al ingresar, en la configuración debemos presionar la opción **Add data source** y seleccionar la opción de **Prometheus**, debemos agregar el siguiente dato:

URL
http://monitoring-kube-prometheus-prometheus:9090/

Y presionar **Save & test**.

Para visualizar los datos se debe agregar Dashboards los cuales se pueden conseguir aquí. Se pueden agregar copiando el código y en el apartado de Dashboards en el Grafana local presionamos **New** y lo importamos, seleccionamos Prometheus como datasource, con esto podemos observar los datos deseados.

Aplicación y Conexión con Firebase

Para utilizar la aplicación móvil es necesario ingresar desde el sitio web de Thunkable y se puede localizar desde la búsqueda de proyectos ingresando el nombre de usuario "_proyectos.bases25_" o desde este enlace

Al ejecutar la aplicación se inicia en la pantalla de Login, pero primero se debe crear una cuenta para ingresar. La información de la cuenta se dirige automáticamente a Firebase que está siempre disponible. Luego se cuenta con un menú de opciones que hacen las llamadas a la API para ejecutar las tareas.

Para ingresar a Firebase en caso de querer revisar la información las credenciales son las siguientes:

USER	PASSWORD
proyectos.bases2	ProyBD2-2022

Pruebas

Para esta sección se diseñaron varias pruebas con el fin de corroborar el correcto funcionamiento de todos los módulos. Es importante destacar que las pruebas a continuación son creadas para cada módulo de manera individual, sin embargo, todas estas se encuentran de manera automatizada, por lo que con solo aplicar los pasos del módulo *loader* bastaría para comprobar el funcionamiento completo del programa.

Las pruebas se realizarán a los siguientes módulos de la aplicación:

Loader:

Pasos:

1. Insertar un job en la tabla jobs de MariaDB, este se procesará de manera automática a través de la transacción designada.
2. Una vez obtenido el job en el pod, este llamará al endpoint "https://api.biorxiv.org/covid19/0,"debe crear una entrada en la tabla jobs y otra en groups con los valores respectivos. Corroborar esta información en MariaDB.

Downloader:

Para corroborar el funcionamiento del downloader utilizarán valores “quemados” directamente en el código. Una vez realizados los pasos a continuación se debe repetir la prueba desde el loader pero ya con los valores sin “quemar”, es decir, con todo automatizado. Los valores quemados son los mensajes que se envían a la cola a través de RabbitMQ.

Pasos:

1. Al recibir el mensaje a través de los valores “quemados” o ya sea, de manera automatizada, el programa debe cambiar el **stage** a “downloader” y el status a “in progress”. Esta nueva información se agregará en un record en la tabla *history*, junto a los cambios respectivos en los demás atributos. Corroborar estos datos.
2. Corroborar los datos descargados en el índice groups en Elasticsearch.
3. Corroborar que la en la tabla *history* se haya actualizado el status, end y message.
4. Corroborar que el grupo haya actualizado su status a “completed”.

Details downloader:

Para corroborar el funcionamiento del downloader utilizarán valores “quemados” directamente en el código. Una vez realizados los pasos a continuación se debe repetir la prueba desde el loader pero ya con los valores sin “quemar”, es decir, con todo automatizado. Los valores quemados son los mensajes que se envían a la cola a través de RabbitMQ.

Pasos:

1. Al recibir el mensaje a través de los valores “quemados” o ya sea, de manera automatizada, el programa debe cambiar el **stage** a “details-downloader” y el status a “in progress”. Esta nueva información se agregará en un record en la tabla *history*, junto a los cambios respectivos en los demás atributos. Corroborar estos datos.
2. El pod debe descargar los detalles para documento y agregarlos en Elasticsearch. Realizar la consulta respectiva del grupo con el fin de corroborar esta información.
3. Corroborar que el grupo haya actualizado su status a “completed”.

Jatsxml Processor:

Para probar este componente se busca un archivo con campo jatsxml

Pasos:

1. Automatizada, el programa debe cambiar el **stage** a “a jatsxml-processor” y el status a “in progress”
2. El pod debe realizar la conversión y actualizar cada componente
3. Corroborar que el grupo haya actualizado su status a “completed”. Y lo remueve de elasticsearch

API:

Pasos:

1. Realizar un POST con la información de un job. Corroborar que esta información se ingresa en la tabla jobs de MariaDB
2. Realizar un GET de los artículos científicos y comprobar los datos.

Recomendaciones y Conclusiones

Recomendaciones

1. Se recomienda leer toda la documentación antes de ejecutar el programa.
2. En caso de querer modificar la configuración de los pods se debe hacer desde el chart correspondiente, esto debido a que se utilizan variables de entorno para su configuración.
3. Se recomienda conocer la librería de ElasticSearch, la cual se puede instalar con el siguiente comando:

   ```
   python -m pip install elasticsearch
   ```

4. Se recomienda conocer la librería de MariaDB, la cual se puede instalar con el siguiente comando:

   ```
   python -m pip install mariadb
   ```

5. Se recomienda conocer la librería de RabbitMQ, la cual se puede instalar con el siguiente comando:

   ```
   python -m pip install pika
   ```

6. Para realizar pruebas se recomienda hacerlo con los pods de la imagen de python, no solamente con una descarga de python ya que puede existir una diferencia de versiones y algunas funciones podrían no estar disponibles en la versión.
7. En el caso de utilizar el sistema operativo Linux, algunos comandos presentados en las instrucciones de este documento podrían variar, por lo que se recomienda buscar los comandos adecuados para el sistema operativo.
8. Es recomendable contar con una computadora con suficiente Memoria y CPU para que al levantar la aplicación y utilizarla tenga un rendimiento óptimo.
9. Se recomienda realizar una lectura al código fuente de la aplicación para familiarizarse con el uso de esta.
10. Se recomienda definir desde el inicio los nombres de todas las colas en las variables de entorno.

Conclusiones

Es importante destacar que para el momento que se crea esta documentación el programa se encuentra incompleto, por lo que muchas pruebas listadas anteriormente o módulos podrían estar o no en el producto final. Además que por el estado actual del proyecto, todavía no se han podido llevar a cabo las pruebas suficientes, así como tampoco se cuentan con todas las referencias bibliográficas. Se espera la comprensión del caso.

1. Se obtuvo un gran aprendizaje de las herramientas de Docker y Kubernetes, y la razón de su necesidad para el proyecto.
2. Se trabajó más con la herramienta de Helm para la automatización del proyecto, objetivo que no fue logrado para la entrega de Observability.
3. El trabajo y la documentación no se encuentran completos debido a la poca organización del grupo y a la gran cantidad de trabajos existentes para el final de semestre.
4. Se logró toda la instalación y configuración de las Bases de Datos, ElasticSearch y MariaDB para el trabajo del proyecto.

5. Se logró crear todas las colas correspondientes con la ayuda de la herramienta RabbitMQ.
6. Se obtienen exitosamente métricas de MariaDB y RabbitMQ.
7. El proyecto no se encuentra completo, pues faltan varios módulos por realizar. Se espera que para la revisión se puedan completar más módulos.
8. La curva de aprendizaje para completar los objetivos del proyecto fue un poco menor debido a la previa experiencia con el proyecto #1; sin embargo, la mala organización no permitió completar todas las tareas asignadas al momento de crear este documento.
9. Fue necesario contar con tiempo extra para poder lograr una completitud mayor a un 40% del proyecto.
10. Todos los aprendizajes fueron compartidos por el grupo de trabajo.

Referencias Bibliográficas

How to Connect Python Programs to MariaDB - referencia

Elastic Python Client - referencia

Configure ECK - referencia

Create Index API - referencia

MySQL: Get column name or alias from query - referencia

Python String Methods - referencia

Python Official Image - referencia