Tarea Corta - Observability

Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Bases de Datos II (IC 4302)
Primer Semestre 2023

1. **Objetivo General**
   - ➔ Implementar una solución de Observability para bases de datos SQL y NoSQL.

2. **Objetivos Específicos**
   - ➔ Instalar y configurar motores de bases de datos SQL y NoSQL mediante Kubernetes.
   - ➔ Instalar y configurar una solución de monitoreo y alertas utilizando Prometheus y Grafana.
   - ➔ Implementar pruebas de carga sobre bases de datos SQL y NoSQL mediante el uso de la herramienta Gatling.
   - ➔ Instrumentar una aplicación para exponer métricas mediante Prometheus.

3. **Datos Generales**
   - ➔ El valor de la tarea corta: 10%
   - ➔ La tarea debe ser implementada por grupos máximo de 5 personas.
   - ➔ La **fecha de entrega** es 17/03/2023 antes de las 11:59 pm.
   - ➔ Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo con el reglamento. La copia incluye código que se puede encontrar en Internet y que sea utilizado parcial o totalmente sin el debido reconocimiento al autor.
   - ➔ La revisión es realizada por el profesor asignado al curso, él mismo se reserva el derecho de solicitar una revisión virtual con los miembros del grupo para evacuar cualquier duda sobre la implementación.
   - ➔ Se espera que todos y todas las integrantes del grupo entiendan la implementación suministrada.
   - ➔ Se deben seguir buenas prácticas de programación. Por ejemplo, documentación interna y externa, estándares de código, diagramas de arquitectura, diagramas de flujo, pruebas unitarias son algunas de las buenas prácticas que se esperan de un estudiante de Ingeniería en Computación.
   - ➔ El lenguaje de implementación en caso de ser requerido y no especificado por el profesor debe ser seleccionado por cada grupo y debe estar debidamente documentado.
   - ➔ Toda documentación debe ser implementada en Markdown.
   - ➔ El email de entrega debe contener una copia del proyecto en formato tar.gz y un enlace al repositorio dónde se encuentra almacenado, debe seguir los lineamientos en el programa de curso.
   - ➔ Al no entregar documentación se obtiene una nota de 0.

4. **Descripción**

Debido a la explosión en el uso de microservicios en los años recientes, el desarrollo de aplicaciones más interactivas y tecnologías como Internet of Things, se han comenzado a generar grandes cantidades de datos de monitoreo, entre las cuales podemos encontrar métricas de sistemas, logs y traces, esta situación limita la posibilidad de que seres humanos y sistemas computacionales convencionales (bases de datos SQL) puedan extraer información valiosa que permita entre otras cosas entender y predecir el comportamiento del sistema con el fin de tomar las medidas pertinentes a tiempo para asegurar su máximo disponibilidad, evitando actuar de forma reactiva.

Con la generación de grandes cantidades de datos de series temporales, se han desarrollado bases de datos NoSQL llamadas Time Series Databases, una de las cuales es Prometheus, la cual ha evolucionado y se ha convertido en la solución OpenSource líder para monitoreo, la misma se integra con otras herramientas OpenSource como Kubernetes, Grafana y Thanos para formar un suite de aplicaciones que pueden competir con las soluciones SaaS lideres en el mercado como lo son Datadog, NewRelic y Dynatrace.

Para este trabajo, cada uno de los grupos deberá llevar a cabo las siguientes tareas para implementar una solución de monitoreo:

Instalación de motores de base de datos

Para el desarrollo de esta tarea cada uno de los grupos deberá elaborar Helm Charts que permitan la instalación y configuración de las siguientes herramientas:

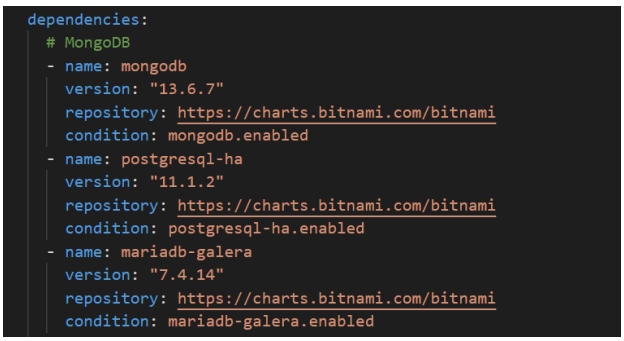
- Prometheus
  - https://bitnami.com/stack/prometheus-operator/helm
- Grafana
  - https://bitnami.com/stack/grafana-operator/helm
- MariaDB (versión OpenSource de MySQL)
  - https://bitnami.com/stack/mariadb/helm
  - Mínimo de 3 instancias con un primary y dos replicas.
- MariaDB-Galera
  - https://bitnami.com/stack/mariadb-galera/helm
  - Mínimo de 2 instancias.
- MongoDB
  - https://bitnami.com/stack/mongodb/helm
  - mínimo de 3 replicas
- Elasticsearch
  - https://www.elastic.co/guide/en/cloud-on-k8s/2.2/index.html
  - mínimo de 3 data nodes con un máster node

- PostgreSQL HA
    - https://bitnami.com/stack/postgresql-ha/helm
- PostgreSQL
    - https://bitnami.com/stack/postgresql/helm

Se recomienda la elaboración de los siguientes Helm Charts:

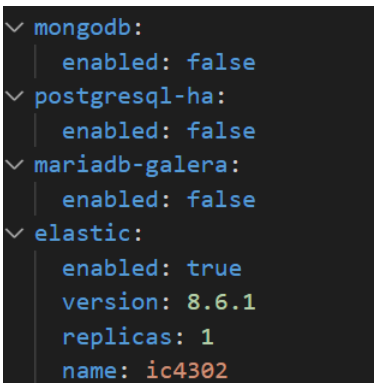
- **bootstrap**: Creara todos los namespaces e instalación de operadores.
- **databases**: Este contendrá las siguientes herramientas
    - MariaDB
    - MongoDB
    - Elasticsearch
    - PostgreSQL
    - MariaDB Galera
    - PostgreSQL HA

    El uso del motor de bases de datos deseado se puede controlar mediante dependencias y condiciones a nivel de Helm Charts, por ejemplo, después de ejecutar ***helm create databases***, se puede agregar el siguiente código al archivo **Chart.yaml**



(Imagen de ejemplo)

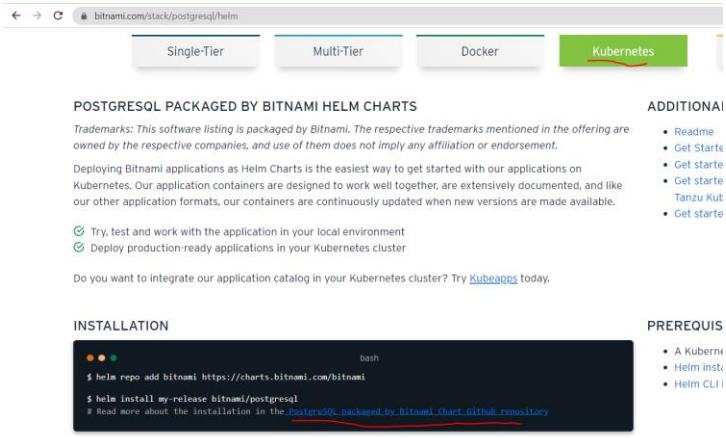
Y luego agregar en el archivo **values.yaml** lo siguiente:



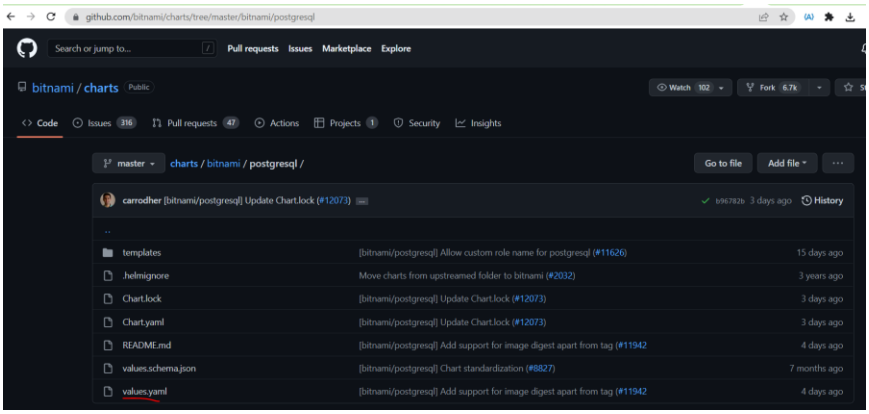
(Imagen de ejemplo)

- **monitoring-stack**: Contendrá las herramientas Prometheus, Grafana y Thanos en caso de que se desee usar.
- **app**: Contendrá los Helm Charts requeridos para la aplicación que se debe instrumentar para probar las bases de datos que no exponen una interfaz HTTP.
- **grafana-config**: Cargara dinámicamente los Dashboards y Datasources.

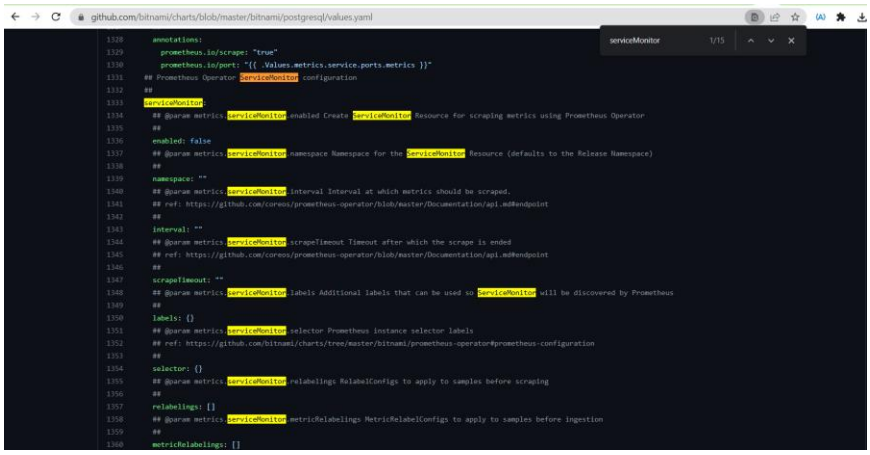
Es importante mencionar que cada uno de los paquetes de instalación proporcionados por Bitnami, se encuentran preparados para generar métricas de uso, las cuales pueden ser utilizadas por Prometheus de forma dinámica (auto-discovery), para habilitarlo basta con usar los parámetros correctos para el helm chart, por ejemplo en el caso de PostgreSQL, en la documentación y en la sección de instalación, se le debe dar clic al link *"# Read more about the installation in the PostgreSQL packaged by Bitnami Chart Github repository"*



Esto los llevara al repositorio de GitHub de este Helm Chart, https://github.com/bitnami/charts/tree/master/bitnami/postgresql, en el mismo se encuentra un archivo llamado values.yaml



En este archivo se encuentra todas las posibles configuraciones para el Helm Chart, basta con buscar *serviceMonitor,* para encontrar como habilitar la integración con Prometheus.



En el caso de Elasticsearch, los paquetes oficiales de ECK no contienen una integración con Prometheus, por esta razón es necesario la siguiente herramienta https://prometheus-community.github.io/helm-charts/

Configuración de Grafana

Las configuraciones de Grafana se cargaran automaticamente mediante los Helm Charts que se instalaran, cada grupo debera identificar cuales son los mejores dashboards para visualizar los datos, los mismos pueden ser encontrados en https://grafana.com/grafana/dashboards/.

El profesor brindara un ejemplo de configuración, el mismo muestra un ejemplo de como realizar esta tarea con Elasticsearch.

El siguiente es un ejemplo de como se vera un dashboard:

Generación de pruebas

Cada uno de los grupos, deberá generar pruebas de carga con Gatling, las mismas realizaran en los motores de bases de datos los siguientes tipos de pruebas:

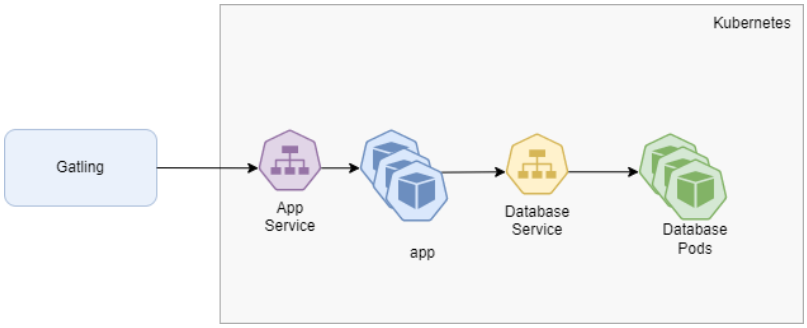
- Creación de registros/documentos.
- Borrado de registros/documentos.
- Actualización de registros/documentos.
- Búsquedas de registros/documentos.

Las mismas deberán correr por largos periodos de tiempo (al menos 30 minutos), para permitir generar gráficos de monitoreo donde se pueda observar el comportamiento de los diferentes motores de bases de datos, es importante observar los siguientes parámetros (no todas las bases de datos los exponen):

- Disco.
- Memoria.
- CPU.
- Network.
- File Descriptors.
- IOPS.
- Open Connections.
- Queries per second.
- Query response time.
- Thread Pools.
- File Descriptors.

La idea detrás de las pruebas de carga es medir el rendimiento de las bases de datos con una configuración especifica, traten de utilizar los mismos datos en diferentes motores, es importante mencionar que Gatling se ejecutara fuera de Kubernetes.

Gatling es una herramienta que trabaja con pruebas de carga sobre endpoints HTTP, no todas las bases de datos SQL y NoSQL exponen una interfaz para interactuar con este protocolo, por esta razón se deberá implementar una aplicación intermediaria, la misma se ilustra en la siguiente imagen:



La aplicación intermediaria, deberá estar implementada en Python con Flask, estará manejada por un Deployment de Kubernetes y se encuentra detrás de un servicio de tipo ClusterIP. Además, esta aplicación deberá estar instrumentada para exponer métricas con Prometheus, acá se puede encontrar un ejemplo, solo se deberá exponer el número de peticiones HTTP.

Documentación

La documentación debe incluir al menos:

- Guía de instalación y uso de la tarea: Se debe explicar en detalle como ejecutarla utilizarla.
- Configuración de las herramientas (haciendo énfasis en los valores utilizados para cada una).
- Pruebas de carga realizadas: se debe especificar el tipo de datos que se están almacenando (dataset), tipo de prueba (creación, borrado, actualización y búsqueda), parámetros utilizados (configuración de Gatling), resultados (apoyados por la información de monitoreo y gráficos obtenidos) y conclusiones de la prueba, se deben incluir al menos 5 pruebas por motor de búsqueda.
- Conclusiones y recomendaciones de la tarea corta.

6. **Entregables**
   - Documentación.
   - Proyecto del Gatling con todos los scripts requeridos.
   - Aplicación Intermediaria.
   - Helm Charts para la instalación de las herramientas requeridas por la tarea.

7. **Evaluación**

Funcionalidad / Requerimiento	Porcentaje
Helm Charts	25%
Pruebas Gatling	40%
Documentación	25%