Tarea corta #1

> Integrantes:

- Max Richard Lee Chung - 2019185076
- Miguel Ku Liang - 2019061913
- Sebastian Campos Zuniga - 2016140230
- Dylan Stef Torres Walker - 2018135751

Guía de instalación

El programa necesita de la aplicación "Docker" y habilitar el servicio de "Kubernetes" para tener un clúster inicial y básico llamado "docker-desktop". La aplicación para controlar y monitorear el comportamiento del clúster de Docker se denomina "Lens".

Instalación de helm charts

Para el desarrollo de esta tarea corta, se van a implementar dos helm charts principales, databases y monitoring. Dentro del helm chart databases, se utilizó la base de datos PostgreSQL, MariaDB y MongoDB. Dentro del helm chart monitoring, se utilizó Grafana y Prometheus. Antes de instalar todos componentes, se debe de ubicar o crear una carpeta cualquiera para luego descargar las dependencias necesarias. Ya con la carpeta seleccionada, se ingresa a la consola de comando y realizar un "cd" a dicha carpeta y ejecutar el siguiente comando:

```
helm create batabases
helm create monitoring
```

Luego de ejecutar los comandos, se debe de eliminar el contenido de la carpeta "templates" de los archvios generados para evitar algún problema a la hora de instalar los recursos. En este punto, se debe de gestionar las dependencias de los helm charts de databases y monitoring dentro del archivo Chart.yaml al final del códgio ya proporcionado, el cual se debe de conocer las versión, enlace del repositorio y nombre para poder extraer y descargar como un archivo comprimido los recursos de cada uno. Los repositorios se deben de buscar en Google buscando, como por ejemplo, el repositorio de Bitnami. Al conocer el enlace del repositorio, se añade a la libería de helm charts y buscar el nombre del recurso al que se quiera utilizar u actualizar como se muestra en el siguiente bloque de código.

```
helm add repo <enlace proporcionado>    # Agregar a la librería de helm charts
helm update repo <enlace proporcionado> # Actualizar el repositorio
helm search repo <recurso a buscar>     # Buscar la información del recurso
```

Al conocer los datos específicos de los recuros a utilizar, se agregan las dependencias como se observa en el siguiente apartado. Además, se agregaron variables para habilitar o deshabilitar el recurso en específico, el cual se comentará más adelante cómo configurarlo.

```
# Chart.yaml de monitoring
dependencies:
- name: kube-prometheus
  version: "8.1.5"
  repository: "https://charts.bitnami.com/bitnami"
  condition: enableGrafana
- name: grafana
  version: "8.2.6"
  repository: "https://charts.bitnami.com/bitnami"
  condition: enablePrometheus
```

```
# Chart.yaml de databases
dependencies:
- name: postgresql
  version: "11.8.1"
  repository: "https://charts.bitnami.com/bitnami"
  condition: enablePostgreSQL
- name: mongodb
  version: "13.1.2"
  repository: "https://charts.bitnami.com/bitnami"
  condition: enableMongoDB
- name: mariadb
  version: "11.3.0"
  repository: "https://charts.bitnami.com/bitnami"
  condition: enableMariaDB
- name: elasticsearch
  version: "19.3.0"
  repository: "https://charts.bitnami.com/bitnami"
  condition: enableElasticsearch
- name: eck-operator
  version: "2.4.0"
  repository: "https://helm.elastic.co"
  condition: enableECKOperator
- name: prometheus-elasticsearch-exporter
  version: "4.14.0"
  repository: "https://prometheus-community.github.io/helm-charts"
  condition: enableElasticsearchExporter
```

Al tener listas las dependencias con los recursos deseados, se debe de hacer un "cd" a cada una de las carpetas generadas para poder ejecutar el siguiente comando, con el fin de "actualizar" las dependencias y que se descarguen los archivos comprimidos como se mencionó anteriormente e instalarlas.

```
cd databases                        # Ingresa a la carpeta
helm dependency update              # Actualiza las dependencias
cd ..                               # Se sale de la carpeta
helm install databases databases    # Se instala el helm chart
```

En caso contrario

```
cd monitoring
helm dependency update
cd ..
helm install monitoring monitoring
```

Sin embargo, si se instalan directamente, las bases de datos no tendrán las configuraciones que se desea, por lo que se modificó el archivo values.yaml para dicho aspecto. Para ello, se agregaron variables para habilitar o deshabilitar dicho recurso y habilitar las métricas del servicio de monitoreo para que Prometheus pueda observarlos y notificar en Grafana.

```
  enablePostgreSQL: true
  enableMariaDB: true
  enableMongoDB: true
  enableElasticsearch: true
  enableECKOperator: false
  enableElasticsearchExporter: true
```

La primera configuración de bases de datos es la de PostgreSQL, en donde únicamente se le agregó el nombre de la base de datos de la autentificación.

```
  postgresql:
    auth:
      database: "postgresql"
    metrics:
      enabled: true
      serviceMonitor:
        enabled: true
```

La segunda configuración de bases de datos es la de MariaDB, en donde se tuvo que crear un Secret (secreto) para almacenar las contraseñas de autentificación para poder acceder al sistema como se muestra a continuación cuya contraseña de administrador y usuario de las réplicas es "root" y "replicator" respectivamente y para ejecutarlo se corre el último comando.

```
apiVersion: v1
kind: Secret
metadata:
    name: mariadb-secret
type: Opaque
data:
  mariadb-root-password: cm9vdA==
  mariadb-replication-password: cmVwbGljYXRvcg==
kubectl apply -f <archivo.yaml>
```

Ya con el Secret ejecutado, se menciona que el Secret a utilizar es el creado y modificar la estructura de la base de datos para permitir el uso de réplicas primarias y secundarias. Como se puede observar, tenemos uso de una réplica primaria y 2 réplicas secundarias.

```
mariadb:
  image:
    pullSecrets: [mariadb-secret]
    debug: true
  architecture: replication
  auth:
    existingSecret: mariadb-secret
  primary:
    name: primary
  secondary:
    replicaCount: 2
  metrics:
    enabled: true
    serviceMonitor:
      enabled: true
```

Dentro de la configuración de la base de datos MongoDB, se cambió la arquitectura, al igual que MariaDB, a una de tipo replicación para poder crear tres réplicas. La autentificación se puede deshabilitar para facilitar el acceso a la base de datos sin contraseña alguna.

```
mongodb:
  architecture: replicaset
  useStatefulSet: true
  auth:
   enabled: false
  replicaCount: 3
  metrics:
   enabled: true
   serviceMonitor:
     enabled: true
```

Por último, la base de datos Elascticsearch se configuró los recursos del ordenador a utilizar por cada réplica (un máster, tres nodos de datos, un coordinador y un procesador de datos), en este caso, se optó por configurarlo a un consumo de 1Mb excepto el máster, el cual usa 2 Mb.

```
elasticsearch:
  # Configuracion del nodo master
  master:
    # Numero de replicas definidas para el master
    replicaCount: 1
    resources:
      requests:
        memory: "2Mi"
```

```
          cpu: "125m"
    # Configuracion del nodo de datos
    data:
      # Numero de replicas definidas para almacenar los datos
      replicaCount: 3
      resources:
        requests:
          memory: "1Mi"
          cpu: "125m"
    # Configuracion del nodo procesador de datos
    ingest:
      # Numero de replicas definidas para el procesador de datos
      replicaCount: 1
      resources:
        requests:
          memory: "1Mi"
          cpu: "125m"
    # Configuracion del nodo coordinador
    coordinating:
      # Numero de replicas definidas para el coordinador
      replicaCount: 1
      resources:
        requests:
          memory: "1Mi"
          cpu: "125m"
    metrics:  # Habilitar el servicio de monitoreo
      enabled: true
      serviceMonitor:
        enabled: true
```

No obstante, Elasticsearch necesita de un operador y exportador como se muestra a continuación.

```
# -- Cambios a los valores del operador de Elasticsearch
  eck-operator:
    nameOverride: "eck-operator"
    fullnameOverride: "eck-operator"
    managedNamespaces: []

# -- Cambios a los valores del exportador de datos de Elasticsearch
  prometheus-elasticsearch-exporter:
    serviceMonitor:
      enabled: true
```

Configuración de herramientas no automatizadas

Grafana

Configurar Grafana primero es necesario haber instalado el helm chart monitoring para tener un Pod funcional de grafana, en donde se selecciona para poder acceder al enlace con el puerto de Grafana. Al estar

en la página, se le solicita un usuario, en donde el usuario predeterminado es "admin" y la contraseña se adquiere dentro del Secret "monitoring-grafana-admin", el cual es generado automáticamente.

Una vez dentro de la aplicación, es necesario agregar a Prometheus para monitorear y alertar eventos. Dentro de la esquina inferior izquierda, se encuentra el botón de configuracíón con forma de engranaje, al darle click, se despliega la página y pestaña de "Data sources", en donde se da la opción de crear uno nuevo. La nueva ventana despliega todas las opciones de "Data Source", el cual se le da a la opción de Prometheus. Ya en la opción de ingresar los datos del nuevo recurso, es necesario conocer el enlace del Prometheus creado por el helm chart cuyo enlace es el nombre del "Service" (servicio) "monitoring-kube-prometheus-prometheus" seguido de :9090 (puerto predeterminado) y se agrega.

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

Por último, es necesario buscar un tablero "dashboard" para poder ver todas las métricas que Prometheus monitorea en Grafana, cuyo tablero se busca uno apropiado con los datos requeridos. Dicho tablero es preferible de la misma página de Grafana para poder importarlo con un ID en la sección de "Dashboards".

Conclusiones

El objetivo de esta tarea es adquir el conocimiento, y poder hacer uso de las tecnologías actuales del mercado. Para este trabajo se hizo uso de 4 bases de datos y se aprendió mediante el uso de Helm Charts para su instalación y configuración. Además, la configuración del los sistemas de monitoreo para ver el comportamiento en las diferentes bases de datos. Por último, se tuvo que hacer uso de Gatling para gener los datos para las pruebas de las bases de datos pero por motivos de tiempo no se logró dicho objetivo.

Referencias

- Repositorio
- Bitnami - MariaDB (2022) Github. Recuperdo de MariaDB
- Bitnami - MongoDb (2022) Github. Recuperado de MongoDB
- Bitnami - PostgresSQL (2022) Github. Recuperado de PostgreSQL
- Bitnami - Elasticsearch (2022) Github. Recuperado de Elasticsearch
- Elastic (2022). Elastic Cloud on Kubernetes [2.4]. Recuperado de Elasticsearch-operator
- Prometheus-community - Prometheus elasticsearch exporter (2022) Github. Recuperado de Elasticsearch-exporter