GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

Ctra. Estació s/n. 46870 Ontinyent (Valencia)
Telf. 962919375. E-mail: 46006100@gva.es

Práctica docker -compose

Documentar correctamente y de forma clara el proceso de dockerización de una aplicación de servidor utilizando docker-compose, podéis utilizar la aplicación de la asignatura de servidor.

Así pues, partiendo de la misma rama main, crearéis una **nueva rama denominada main_docker_compose** que subiréis al repositorio remoto. Sobre esta y en la carpeta raíz de vuestro código, deberá existir un fichero docker-compose.yml que implementará lo siguiente:

- Servicio de **mongodb**
  - Utilizará la imagen docker oficial de mongo (link)
  - El contenedor asociado se denominará mongo_container
  - Será el primer servicio en arrancar
  - La primera tarea que hará nada más arrancar será crear las tablas necesarias (link) y realizar una restauración de datos (link) partiendo de un fichero dump (link) que previamente habréis generado y almacenado en una carpeta mongo de vuestro proyecto.
  - Todos los ficheros de configuración necesarios residirán en una carpeta mongo de nuestro repositorio

- Servicio de **backend**
  - Igual que el servicio anterior, utilizará una **multi-stage build** (al menos tendrá dos etapas) para generar la imagen de la parte backend de vuestro proyecto implementada con express. Partirá de una imagen node:19-alpine
  - No arrancará hasta que el servicio de mongodb no esté preparado completamente
  - El contenedor asociado se denominará backend_container
  - Ejecutará como primer comando nada más arrancar: `npm start`

- Servicio de **frontend**
  - Utilizará una **multi-stage build** (al menos tendrá dos etapas) para generar la imagen de vuestra parte implementada en angularjs o otro framework. Partirá de una imagen node:19-alpine
  - Arrancará tras el servicio de backend
  - El contenedor asociado se denominará frontend_container
  - Ejecutará como primer comando nada más arrancar: `npm start`

- Servicio **mongo-express**.
  - Nos permitirá administrar la base de datos mongo. Utilizará la imagen oficial de mongo-express (link)
  - El contenedor asociado se denominará adminMongo_container
  - Arrancará después del servicio de mongodb

- Servicio de **loadbalancer de nginx**
  - Nos permitirá implementar un sistema de balanceo de carga/proxy en nuestro sistema
  - Partirá de la imagen oficial de nginx

- Asociará un fichero de configuración de nginx (nginx.conf) que tendremos en la carpeta loadbalancer de nuestro repositorio con el mismo fichero de la carpeta /etc/nginx/ de la imagen. Este fichero nos permitirá implementar el balanceador de carga y su contenido será similar al adjuntado a esta tarea (realizando las modificaciones oportunas para adecuarlo a vuestras necesidades).
- Ejecutará como primer comando nada más arrancar: `nginx -g daemon off`

- Servicios de **monitorización** (opcionales)

    - Servicio Prometheus

      Prometheus es una aplicación que nos permite recoger métricas de una aplicación en tiempo real. Como veréis en el ejemplo de app.js, se incluye una dependencia en el código (prom-client) que permite crear contadores de peticiones que podemos asociar fácilmente a nuestros endpoints de manera que podemos saber cuántas veces se ha llamado a una función de nuestra api.

      En nuestro caso, el servicio de prometheus se encargará de arrancar en el puerto 9090 de nuestro host un contenedor (prometheus_practica) basado en la imagen prom/prometheus:v2.20.1. Para poder configurar correctamente este servicio, será necesario realizar además dos acciones:

        - Copiar el fichero adjunto prometheus.yml al directorio /etc/prometheus del contenedor
        - Ejecutar el comando --config.file=/etc/prometheus/prometheus.yml

    - Servicio Grafana

      Este servicio será el encargado de graficar todas las métricas creadas por el servicio de Prometheus. Por tanto, siempre arrancará tras el de prometheus. En nuestro caso, el servicio de grafana se encargará de arrancar en el puerto 3500 de nuestro host un contenedor (grafana_practica) basado en la imagen grafana/grafana:7.1.5 que, además, se caracterizará por:

        - Establecer las variables de entorno necesarias para:
            - Deshabilitar el login de acceso a Grafana
            - Permitir la autenticación anónima
            - Que el rol de autenticación anónima sea Admin
            - Que instale el plugin grafana-clock-panel 1.0.1

        - Dispondrá de un volumen nombrado (myGrafanaVol) que permitirá almacenar los cambios en el servicio ya que se asociará con el directorio /var/lib/grafana

      Además, para una correcta configuración de Grafana, será necesario realizar la copia del fichero adjunto datasources.yml al directorio del contenedor /etc/grafana/provisioning/datasources/.

Como ayuda aquí tenéis un ejemplo de proyecto realizado en años anteriores que se

integraba Grafana y prometheus con la aplicación desarrollada en el módulo de servidor:

- https://vicnx.github.io/Metrics_Prometheus_Grafana_Nodejs/

A tener en cuenta:

- Con toda seguridad necesitareis modificar a nivel de código las referencias a endpoints o url's que tengan que ver con la base de datos para que la aplicación funcione correctamente.
- Necesitareis que todos los servicios estén conectados en una misma red (practica_net). Por supuesto, cada uno de ellos gestionará unos puertos que, de no indicarse uno específico, podéis elegir libremente.
- Todas las variables de entorno que necesiten los servicios estarán en un fichero .env común del proyecto que cada servicio leerá.

Como respuesta a la actividad planteada, entregaréis el link a vuestro repositorio github. En el README.md de la rama creada deberéis documentar los principales pasos/cambios realizados para poder implementar la configuración pedida y los pasos necesarios para poder poner en marcha el proyecto.