Maestría
en Inteligencia
Analítica de Datos

Despliegue con contenedores: Docker

Instrucciones:

En este taller exploraremos el despliegue de soluciones analíticas usando contenedores Docker.

La entrega de este taller consiste en un reporte en formato de documento de texto, donde pueda incorporar fácilmente capturas de pantalla, textos y elementos similares. Puede utilizar formatos como Word, LibreOffice, Markdown, u otros.

Parte 1: despliegue con contenedores

En esta primera parte usaremos una máquina virtual para desplegar allí nuestra API empleando un contenedor Docker.

1. En la consola de EC2 lance una instancia t2.small, **Ubuntu server** con 20 GB de disco. **Incluya un pantallazo de la consola de AWS EC2 con la máquina en ejecución en su reporte. Su usuario de AWS y las IPs privada y pública deben estar visible en el pantallazo.**
2. Para conectarse a la instancia, en una terminal emita el comando

   ```
   ssh -i /path/to/llave.pem ubuntu@IP
   ```

   donde */path/to/* se refiere a la ubicación del archivo *llave.pem* que descargó, e IP es la dirección IP pública de la instancia EC2 que lanzó. Si prefiere, en la terminal puede navegar a la ubicación del archivo llave.pem y emitir el comando

   ```
   ssh -i llave.pem ubuntu@IP
   ```

   Note que usamos en este caso ubuntu en vez de ec2-user, pues éste es el usuario creado por defecto como administrador con sistema operativo Ubuntu server.
3. Ahora pasamos a instalar Docker en la máquina, para lo cual requerimos eliminar posible versiones anteriores y agregar el repositorio de la última versión estable de Docker.
4. En la máquina virtual, elimine versiones anteriores de Docker (esto genera un error si no hay versiones anteriores, en cuyo caso puede continuar sin problema)

   ```
   sudo apt-get remove docker docker-engine docker.io containerd runc
   ```

5. Actualice el índice de paquetes

```
sudo apt-get update
```

6. Instale dependencias para verificar certificados (ca-certificates), obtener objetos con su URL (curl) y administrar llaves PGP (gnupg)

```
sudo apt-get install ca-certificates curl gnupg
```

7. Agregue la llave de Docker

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -
    o /etc/apt/keyrings/docker.gpg
```

Aclaraciones: Por cuestiones de espacio en el documento, este comando quedó separado por el cambio de renglón. Es importante que tenga en cuenta que, luego de la instrucción '–dearmor' viene un guón '-' seguido de una 'o' sin ningún espacio en medio. Es decir, el fragmento de este comando debería tener la siguiente apariencia: '–dearmor -o /etc/apt/...'

```
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

8. Agregue el repositorio de Docker a su sistema para la instalación

```
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker
     .gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Aclaraciones: Por cuestiones de espacio en el documento, este comando quedó separado por el cambio de renglón. Es importante que tenga en cuenta que, el primer renglón del comando en este documento acaba con 'docker' y el segundo renglón comienza con ',gpg]', y estos en realidad deberían ir juntos sin ningún espacio de por medio. Esto quiere decir que cuando esté escribiendo el comando en su cmd, debería escribir 'docker.gpg] ...'.

9. Actualice nuevamente el índice de paquetes con este nuevo repositorio incluido

```
sudo apt-get update
```

10. Instale Docker Engine, containerd, y Docker Compose


```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-
    plugin docker-compose-plugin
```

11. Para verificar su instalación, descargue, construya y ejecute la imagen hello-world

    ```
    sudo docker run hello-world
    ```

    Copie y explique la salida en pantalla en su **reporte**.
12. Adjunto a este enunciado encontrará el archivo *docker-api-starter.zip*. Descomprima el archivo **localmente** en una carpeta, que llamaremos raíz, tal que allí quede la carpeta bankchurn-api y el archivo Dockerfile. En esta carpeta raíz cree un repositorio git, y conéctelo con un repositorio nuevo en GitHub. Al terminar debe tener en su carpeta raíz 2 carpetas, .git y bankchurn-api, así como el archivo Dockerfile.
13. Explore el archivo Dockerfile y en su reporte explique brevemente el paso a paso de la configuración definida en este archivo. Para esto tenga presente que
    - FROM determina la imagen base que se usa como sistema operativo y aplicaciones iniciales.
    - RUN ejecuta comandos al interior del contenedor.
    - COPY copia archivos del sistema hospedador (la máquina virtual) al contenedor.
    - ENV permite definir variables de entorno.
    - EXPOSE abre un puerto del contenedor.
    - CMD es el comando que se ejecuta al lanzar el contenedor.
14. Regrese a la terminal de su máquina virtual en EC2 y clone allí el repositorio. Ingrese a la carpeta del repositorio, donde debe encontrar la misma estructura de carpetas (.git, bankchurn-api) y el Dockerfile.
15. A partir del archivo Dockerfile construya la imagen que usará más adelante para lanzar contenedores

    ```
    sudo docker build -t bankchurn-api:latest .
    ```

    Note que el comando termina con un espacio y un punto. Esto indica que se usa la carpeta actual como base para construir la imagen.
16. Liste las imágenes de docker con el comando

    ```
    sudo docker images
    ```

    Debe contar con la imagen recién creada de bankchurn-api y la hello-world creada anteriormente. Incluya un pantallazo de la salida en su **reporte**. Su IP privada debe ser visible.
17. Ahora ejecute un contenedor usando la imagen creada con el comando

    ```
    sudo docker run -p 8001:8001 -it -e PORT=8001 bankchurn-api
    ```

Incluya un pantallazo de la salida de este comando en su **reporte**. La IP privada debe ser visible.

18. Note que aquí conectamos el puerto 8001 del contenedor con el puerto 8001 de la máquina virtual (-p), dejamos activa la terminal interactiva (-it) y además definimos una variable de entorno PORT con el valor 8001, que se requiere al ejecutar el contenedor (-e). Para clarificar esto revise el archivo run.sh en la carpeta bankchurn-api y note que allí se usa una variable de entorno PORT.
19. Vaya ahora a la consola de EC2, seleccione su máquina virtual y modifique el grupo de seguridad para permitir tráfico por el puerto 8001. Es decir, en el grupo de seguridad de la máquina edite las reglas de entrada y agregue una que permita tráfico por el puerto TCP 8001 desde cualquier IP (anywhere IPv4).
20. Copie la IP pública de su máquina y en un navegador local visite la página IP:8001. Allí debe aparecer la API de bankchurn en ejecución. Incluya un pantallazo del navegador en su **reporte**. La dirección (IP pública) debe ser visible.

Parte 2: despliegue con contenedores en Railway

Ahora queremos desplegar la misma API usando un contenedor Docker en Railway, el servicio PaaS de despliegue.

1. En esta parte emplearemos Railway App, un servicio para el despliegue de aplicaciones. Cree una cuenta en `https://railway.app/dashboard`
   Podrá acceder a un plan con 5 dólares de créditos, que deben ser suficientes para ésta y otras actividades del curso. Note que puede usar su cuenta de GitHub para crear la cuenta Railway.
2. Para emplear Railway necesitamos del manejador de paquetes npm. En la máquina virtual instale npm en su versión estándar para Ubuntu

   ```
   sudo apt install npm
   ```

3. Ahora queremos actualizar npm y node para contar con versiones recientes, necesarias para Railway. Para esto instale nvm

   ```
   curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.38.0/install.sh |
       bash
   ```

   Aclaraciones: Por cuestiones de espacio en el documento, este comando quedó separado por el cambio de renglón. Es importante que tenga en cuenta que, a pesar del cambio de renglón, la instrucción 'bash' está separa del símbolo '|' por un espacio. Es decir, el fragmento de esta instrucción luce así: '.../install.sh | bash'.

   ```
   source ~/.bashrc
   ```

4. Revise la versión de nvm y las versiones remotas disponibles de node

```
nvm --version

nvm ls-remote
```

5. Instale la última versión LTS (long-term support). de node En el momento de escribir esta guía, corresponde a la 20.9

   ```
   nvm install v20.9.0
   ```

6. Revise la versión de node y npm

   ```
   node --version

   npm --version
   ```

7. Ahora sí podemos instalar railway usando npm

   ```
   npm i -g @railway/cli
   ```

8. Revise la versión de railway

   ```
   railway --version
   ```

9. Loguéese a railway con el comando

   ```
   railway login --browserless
   ```

   el cual genera una URL. Copie esta URL en un navegador (Chrome, Firefox, etc.) donde tenga abierta su cuenta Railway. Verifique el acceso en el navegador y revise en la terminal de la máquina que quedó logueado.

10. En su navegador asegúrese de tener abiertas sus cuentas de GitHub y Railway.
11. Si ya ha usado Railway con Github, continúe al paso 13. Si es la primera vez usando Railway con Github, en el navegador vaya a su cuenta de Railway y enlace su cuenta con Github: click en New project, seleccione Deploy from GitHub repo, click en Configure GitHub App.
12. Seleccione la cuenta de Github donde instalará la aplicación Railway. Permita acceso a todos los repositorios. Confirme usando su método preferido de verificación.
13. Con su cuenta ya configurada y enlazada con GitHub, cree un nuevo proyecto vacío. Click en New project, seleccione Empty project.
14. En el proyecto creado agregue un servicio vacío: click en Add a service, seleccione Empty service.
15. Justo arriba del servicio le aparecerá una ventana para aplicar los cambios en el proyecto, que corresponden a la creación del servicio. Dé clic en 'Deploy'.
16. En el proyecto encontrará en la parte inferior una caja con un mensaje "Set up your project locally". Al hacer click allí encontrará un paso de conexión con la siguiente forma


```
railway link idproyecto
```

donde idproyecto es un código de identificación del proyecto. Copie este ID. Cierre la ventana de diálogo.

17. De regreso en la terminal de su máquina virtual, ingrese a la carpeta del repositorio, liste los contenidos de la carpeta

    ```
    ls -la
    ```

    y verifique que tiene las dos carpetas mencionadas anteriormente: .git, y bankchurn-api, así como el Dockerfile.
18. Desde esta carpeta enlace el proyecto y el servicio de Railway con el comando

    ```
    railway link proyectoid
    ```

    donde proyectoid es el ID de proyecto que copió de Railway.
19. Con el enlace realizado, estamos listos para lanzar la aplicación en Railway con el comando

    ```
    railway up --detach
    ```

    Tome un pantallazo de la salida de este comando en inclúyala en su reporte.
20. Regrese al sitio de Railway, en su proyecto y servicio. De click en la pestaña Deployments, allí verá que el despliegue se está realizando. De click en viewlogs e identifique que al principio se indique que se está usando el Dockerfile encontrado:

    ```
    Using detected Dockerfile
    ```

    Tome un pantallazo de los logs de despliegue con este mensaje para su reporte.
21. Cuando termine de desplegar, queda en estado Active. Regrese a la ventana anterior (click en la x de la esquina superior derecha). Click en la pestaña Settings. En la sección Networking y Public Networking, click en Generate Domain. Al generar el dominio **tome un pantallazo del dominio generado para su reporte**.
22. Note que se ha creado una URL. **Copie esa URL en su reporte.**
23. De click en la URL o cópiela y péguela en su navegador. Note que se despliega la API, como se había realizado anteriormente en un máquina virtual.
24. Click en here para ir a los docs. Expanda la ruta POST /api/v1/predict y **tome un pantallazo donde sea visible la URL y las rutas de la API para su reporte.**
25. Al terminar esta actividad puede terminar la ejecución del servicio en Railway. Seleccione el servicio y allí el menú Deployments. Seleccione el despliegue en verde (ejecución), click en los 3 puntos y seleccione Remove.