1. GLPI es una aplicación de gestión de hardware y software que utiliza una base de datos MariaDB. Sigue los pasos indicados para crear un contenedor para la base de datos de MariaDB y un contenedor de GLPI.

```
docker pull diouxx/glpi

docker network create --subnet=172.18.0.0/16 red_glpi

docker run --name mariadb --net red_glpi --ip 172.18.0.100 -e
MARIADB_ROOT_PASSWORD=diouxx -e MARIADB_DATABASE=glpidb -e
MARIADB_USER=glpi_user -e MARIADB_PASSWORD=glpi -d mariadb:10.7

docker run --name glpi --net red_glpi -p 8080:80 -d diouxx/glpi
```

2. Muestra el detalle de la red red_glpi.

3. Responde a las siguientes preguntas:
   a. ¿Cómo se llaman los contenedores que has creado?
   b. ¿Qué tipo de red se ha creado?
   c. ¿Se podrán comunicar entre ellos?
   d. ¿Qué IP se ha asignado a cada contenedor?
   e. ¿Las IPs se han asignado de forma manual o automática?
   f. ¿Para qué se usa la opción -e?
   g. ¿A qué puerto hay que conectarse para acceder al servidor de GLPI?

4. Accede a la dirección http://localhost:8080 e incluye capturas de todos los pasos que has tenido que seguir hasta tener el servidor disponible. Fíjate bien en cómo se ha creado el contenedor de la base de datos (**mariadb**) para extraer los datos que te van a solicitar.

5. Abre la herramienta GLPI e incluye algunas capturas de su funcionamiento.

6. Lista las imágenes.

7. Lista los contenedores.

8. Inicia el contenedor de Apache httpd.

9. Accede al shell de este contenedor y ejecuta el siguiente comando sustituyendo *nombre* por tu nombre:

   ```
   echo "<html><body><h1>Servidor 1 de nombre</h1></body></html>" > htdocs/index.html
   ```

   A continuación, sal de este shell con el comando `exit`.

10. Accede a tu servidor web e incluye una captura de la página de inicio.

11. Crea otro contenedor de Apache httpd llamado **servidor_web_2** (tendrá que escuchar en un puerto distinto).

12. Accede al shell de este contenedor y ejecuta el siguiente comando sustituyendo *nombre* por tu nombre:

    ```
    echo "<html><body><h1>Servidor 2 de nombre</h1></body></html>" > htdocs/index.html
    ```

    A continuación, sal de este shell con el comando `exit`.

13. Accede a este segundo servidor web e incluye una captura de la página de inicio.

14. Crea otro contenedor de Apache httpd llamado **servidor_web_3** sin mapear el puerto 80 y usando el controlador de red **host**.

15. Accede al shell de este contenedor y ejecuta el siguiente comando sustituyendo *nombre* por tu nombre:

    ```
    echo "<html><body><h1>Servidor 3 de nombre</h1></body></html>" > htdocs/index.html
    ```

    A continuación, sal de este shell con el comando `exit`.

16. Muestra las redes disponibles.

17. Muestra el detalle de las redes **bridge** y **host**.

18. ¿Qué dirección IP tienen los servidores 1 y 2? ¿Y cuál tiene el servidor 3?

19. Comprueba que se puede acceder a los dos servidores Web que están en la red **bridge** directamente con la IP que se les ha asignado. Por tanto, hay una red entre el host y los dos contenedores y se pueden comunicar entre ellos.

20. ¿Qué diferencia hay con el servidor 3?

21. Has de comprobar ahora el acceso a estos servidores "desde Internet" por lo que te conectarás desde tu equipo. Para ello has de averiguar la dirección IP que tiene la máquina de Ubuntu (por ejemplo, en una ventana de Terminal, con el comando `ip a`).

22. Busca en Docker Hub cómo crear un contenedor de MySQL y créalo añadiendo las opciones `-it`.

23. Accede al shell del contenedor de MySQL

24. Ejecuta el siguiente comando del cliente de MySQL para conectar:

    `mysql --user=root --password=contaseña` (indica la contraseña que has configurado al crear el contenedor)

25. Una vez dentro del shell de SQL, ejecuta por ejemplo el comando: `show databases;`

26. Para los contenedores de glpi y de Apache httpd.

27. Elimina los contenedores de Apache httpd.

28. Lista los contenedores.

29. Lista las imágenes.

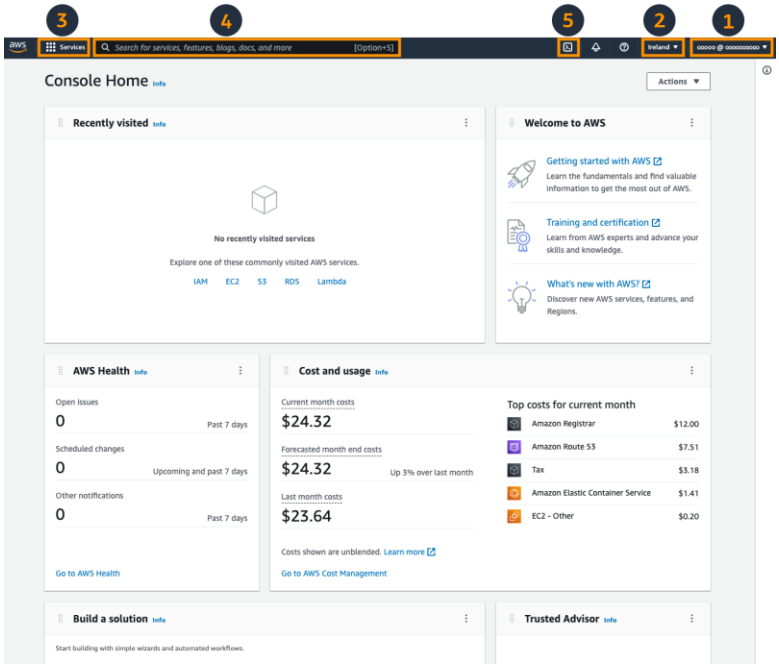
30. Elimina la imagen de Apache httpd.

31. Busca 2 proveedores de IaaS y 2 de PaaS. Explica en detalle qué servicios ofrecen, el precio y cómo es la consola de administración para el cliente. Indica las referencias consultadas.

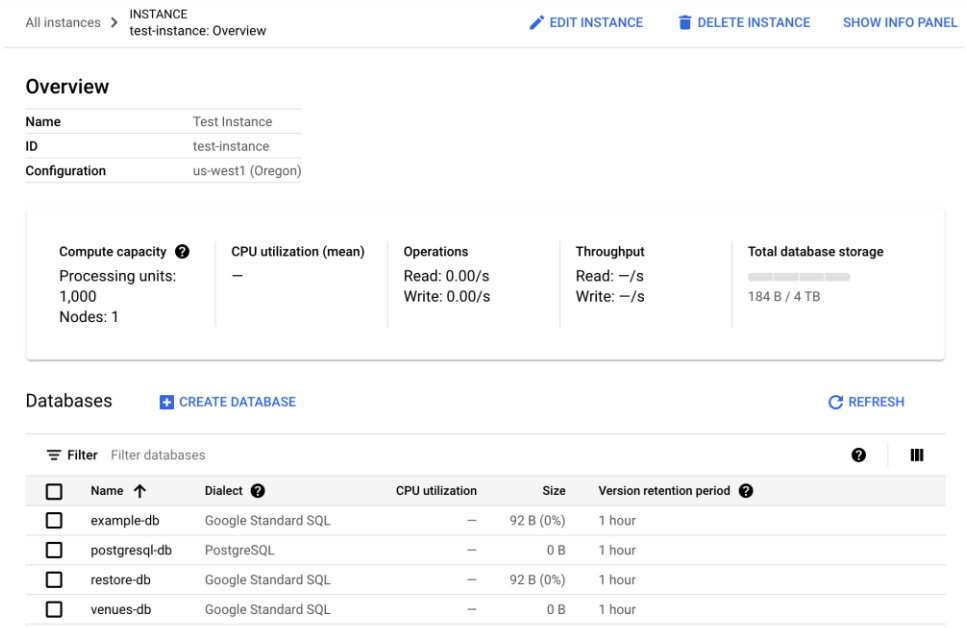
    Aquí tienes un resumen breve de dos proveedores de IaaS y dos de PaaS:

    Proveedores de IaaS:

    1. Amazon Web Services (AWS): Ofrece máquinas virtuales (EC2), almacenamiento (S3) y redes virtuales. Su consola permite monitoreo y gestión detallada. El costo de EC2 inicia en $0.0116 por hora, con opciones de capa gratuita.



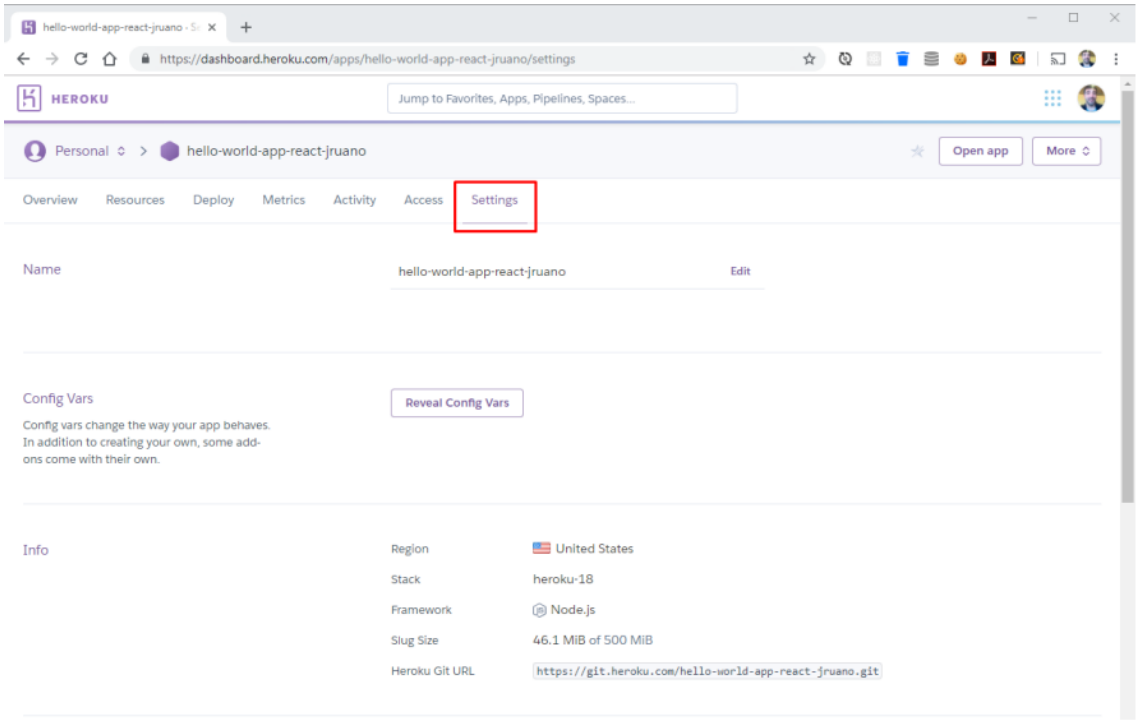
2. Google Cloud Platform (GCP): Incluye computación (Compute Engine) y almacenamiento en la nube. Su consola es intuitiva, y los precios empiezan en $0.0104 por hora con descuentos por uso sostenido.



Proveedores de PaaS:

1. Heroku: Plataforma para desarrollar en múltiples lenguajes, usando contenedores Dynos. Su consola es sencilla y ofrece planes desde $7 al mes.

SOLO ENCONTRE ESTO:



2. AWS Elastic Beanstalk: Soporta aplicaciones en Java, Python, y más, con monitoreo y escalado automático. No tiene costo adicional, solo el de los recursos subyacentes.