**TAREA 1 – EJECUTAR CONTENEDORES SIMPLES**

Aprender a usar contenedores para diferentes casos de uso.

**Contenedor de una sola tarea**

Ejemplo: Ejecutar hostname en Alpine Linux.

Comando: docker container run alpine hostname

**Contenedor interactivo**

Ejemplo: Acceder a una terminal en un contenedor Ubuntu.

Comando: docker container run -it --rm ubuntu bash

Permite ejecutar comandos dentro del contenedor (ls, cat /etc/issue).

**Contenedor en segundo plano**

Ejemplo: Ejecutar MySQL como servicio.

Comando: docker container run --detach --name mydb -e MYSQL\_ROOT\_PASSWORD=123 mysql

Usamos docker container logs y docker exec para interactuar con él.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

**TAREA 2 – CREAR Y EJECUTAR UNA APLICACIÓN PERSONALIZADA**

Empaquetar una app web (NGINX) en una imagen Docker.

**Construir la imagen**

Se usa un Dockerfile para copiar archivos HTML y configurar NGINX.

Comando: docker image build --tag mi-app:1.0 .

**Ejecutar el contenedor**

Publicar el puerto 80 del host hacia el contenedor: docker container run -d -p 80:80 --name web-app mi-app:1.0

**Verificar la app**

Acceder a http://localhost para ver el sitio web.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

**TAREA 3 – MODIFICAR UNA APP EN EJECUCIÓN Y DISTRIBUIRLA**

Actualizar la app sin reconstruir la imagen y subirla a Docker Hub.

**Modificación en caliente**

Usar *bind mounts* para sincronizar cambios locales: docker run -d -p 80:80 -v $(pwd):/usr/share/nginx/html --name web-app mi-app:1.0

Cambios en index.html se reflejan al instante.

**Crear una nueva versión**

Reconstruir la imagen con los cambios: docker image build --tag mi-app:2.0 .

**Subir a Docker Hub**

Compartir la imagen públicamente: docker image push mi-docker-id/mi-app:2.0