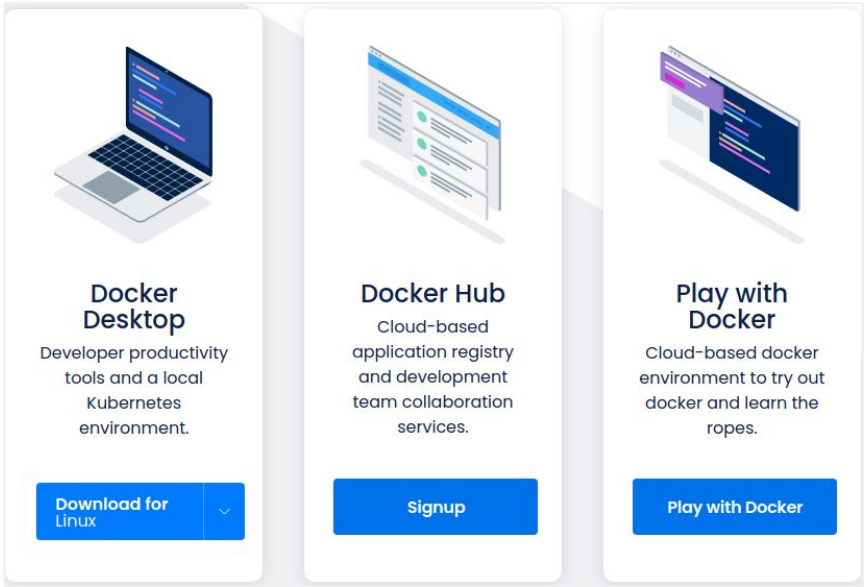
Big Data con Hadoop y Spark

Módulo 01 - Resolución del desafío

Consideraciones

1. Es necesario tener instalado **Docker.**
2. Registrarnos en Docker Hub. https://hub.docker.com/
3. Al ejecutar las instrucciones, anteponer **“sudo”**.



Resolución del ejercicio 1

1. Ejecutar en la consola el contenedor “hello-world” del Docker-Hub y luego verificar si está ejecutando:
    a. **`$ docker run hello-world`** (corro el contenedor hello-world)
    b. **`$ docker ps`** (muestra los contenedores activos)

2. Ejecutar una inspección de un contenedor específico
    c. **`$ docker ps -a`** (muestra todos los contenedores)
    d. **`$ docker inspect <container ID>`** (muestra el detalle completo de un contenedor)
    e. **`$ docker inspect <name>`** (igual que el anterior pero invocado con el nombre)

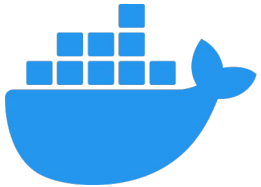
3. Ejecutar el contenedor “hello-world” asignándole un nombre distinto.

    a. **`$ docker run --name hola-mundo dp hello-world`** (le asigno un nombre custom “hola-mundo”)

4. **`$ docker rename hola-mundo hola-a-todos`** (cambio el nombre de hola-mundo a hola-a-todos)

5. **`$ docker rm <ID o nombre>`** (borro un contenedor)

6. **`$ docker container prune`** (borro todos lo contenedores que estén parados)



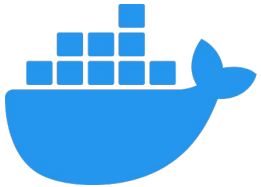
7. Explorar Docker Hub y probar ejecutar alguna de las imágenes.
8. Ejecutar la imagen **“ubuntu”**:
    a. **`$ docker run ubuntu`** (corre un ubuntu pero lo deja apagado)
    b. **`$ docker run -it ubuntu`** (lo corre y entro al shell de ubuntu)
    `-i:` interactivo
    `-t:` abre la consola

    `$ cat /etc/lsb-release` (veo la versión de Linux)
9. Ejecutar la imagen **“nginx”** y probar los comandos **“stop”** y **“rm”:**
    c. **`$ docker run -d --name proxy nginx`** (corro un nginx)
    d. **`$ docker stop proxy`** (apaga el contenedor)
    e. **`$ docker rm proxy`** (borro el contenedor)
    f. **`$ docker rm -f <contenedor>`** (lo para y lo borra)

Ejercicio 2: Docker

Ejecutar **nginx** exponiendo el puerto 8080 de mi máquina.

1. Exponer contenedores:
   `$ docker run -d --name proxy -p 8081:80 nginx`
   (corro un nginx y expongo el puerto 80 del contenedor en el puerto 8080 de mi máquina).
2. **`localhost:8081`** (desde mi navegador compruebo que funcione).



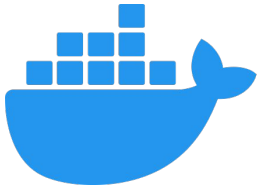
3. Ejecutar el comando **“logs”** para ver los logs del contenedor de **nginx**:

   `$ docker logs proxy` (veo los logs).

4. **`$ docker logs -f proxy`** (hago un follow del log).

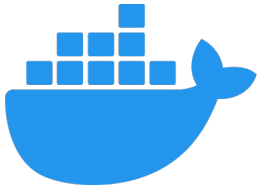
5. Ejecutar comando **“logs –tail”** para ver las últimas N entradas de log.

   `$ docker logs --tail 10 -f proxy` (veo y sigo solo las 10 últimas entradas del log).



Ejercicio 3: Bind Mounts

1. Ejecutar la imagen **“mongodb”** y asociarla con un directorio en mi máquina:
   a. **`$ mkdir dockerdata`** (creo un directorio en mi máquina).
   b. **`$ docker run -d –-name mongodb -v <path de mi maquina>:<path dentro del contenedor(/data/db)> mongo`**
   (corro un contenedor de mongo y creo un bind mount).



Ejecutar el comando **“exec”** para introducirse en el shell de un contenedor:

2. **`$ docker ps`** (veo los contenedores activos)
3. **`$ docker exec -it mongodb bash`** (entro al bash del contenedor)
4. Ejecutar los siguientes comandos:
    a. **`$ mongo`** (me conecto a la base de datos)
    b. **`show dbs`** (listo las bases de datos)
    c. **`use prueba`** (creo la base “prueba”)
    d. **`db.prueba.insert({‘color’: ’azul’})`** (inserto un nuevo dato)
    e. **`db.prueba.find()`** (veo el dato que cargué)
    f. Revisar el contenido del directorio creado.
    g. Volver a ejecutar **del** y verificar que el dato insertado en una ejecución previa ya se pueda ver, debido a que la nueva ejecución levanta lo datos ligados mediante Bind.

¡Terminaste el módulo!

Estás listo para rendir el examen