Pre-requisitos

1. Para esta sesión va a requerir una cuenta de Amazon Web Services - AWS. Para esto hay varias opciones:

   a) Utilizar una cuenta propia ya creada.

   b) Crear una cuenta nueva. En este caso recuerde que requiere ingresar datos de una tarjeta de crédito. Usaremos recursos disponibles en la capa gratuita (https://aws.amazon.com/free/), pero es posible que se generen algunos costos menores.

   c) Usar la cuenta de AWS Academy enviada a su correo por el instructor. En este caso emplee el Learner Lab.

2. Ingrese a su cuenta y familiarícese con la consola.

3. Asegúrese de que en la esquina superior derecha aparezca la región *N. Virginia*.

4. **Nota:** defina un nombre para su grupo, defínalo claramente en **reporte** y use este nombre como parte inicial de **todos los recursos que cree**.

5. **Nota 2:** la entrega de este taller consiste en un **reporte** y unos **archivos de soporte**. Cree el archivo de su **reporte** como un documento de texto en el que pueda fácilmente incorporar capturas de pantalla, textos y similares. Puede ser un archivo de word, libre office, markdown, entre otros.

1. Instale Docker y lance su primer contenedor de prueba

1. En la consola de EC2 lance una instancia t2.micro, **Ubuntu server** con la configuración estándar.

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 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. Elimine versiones anteriores de Docker (esto genera un error si no hay versiones anteriores, en cuyo caso puede continuar también)
   `sudo apt-get remove docker docker-engine docker.io containerd runc`

4. Actualice el índice de paquetes
   `sudo apt-get update`

5. Instale dependencias

```
sudo apt-get install \
ca-certificates \
curl \
gnupg \
lsb-release
```

6. Agregue la llave GPG de Docker

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
    --dearmor -o /etc/apt/keyrings/docker.gpg
```

7. Agregue el repositorio a su sistema

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

8. Actualice nuevamente el índice de paquetes

```
sudo apt-get update
```

9. Instale Docker Engine, containerd, y Docker Compose

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

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

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

11. Copie la salida en pantalla en su **reporte**.

2. Lance una aplicación sencilla en Python en un Docker

1. Localmente descargue los archivos `Dockerfile` y `hello.py`, que acompañan este taller.

2. Abra el archivo `Dockerfile` y describa su contenido en su **reporte**. 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.

3. Copie estos archivos a su máquina remota. Suponiendo que tanto estos archivos como su llave .pem se encuentran en la misma carpeta, navegue a esta carpeta y emita el comando

   ```
   scp -i key.pem Dockerfile ubuntuIP:/home/ubuntu
   ```

   Repita para hello.py.

4. Conéctese nuevamente a la máquina y verifique que los dos archivos se encuentran disponibles en la misma carpeta.

5. Desde esta carpeta construya (build) la imagen con el comando

   ```
   sudo docker build -t test:latest .
   ```

   Aquí *test* es el nombre del contenedor, *latest* es la versión y el punto . define que el folder de contexto para la ejecución es la carpeta actual. Esto implica que tanto el Dockerfile que define al contenedor como el archivo hello.py que se copia al contenedor deben estar en la misma carpeta, desde donde se ejecuta la instrucción anterior. Una vez se haya ejecutado la construcción del contenedor, tome un pantallazo e **inclúyalo** en su reporte.

6. Ahora ejecute el contenedor con el comando

   ```
   sudo docker run -p 8000:8000 test:latest
   ```

   Note que además de indicar el nombre y versión del contenedor a ejecutar, se enlazan el puerto 8000 del contenedor con el mismo de la máquina, para así hacer disponible lo que se exponga a través de este puerto a la máquina.

7. Abra el puerto 8000 en su instancia y explore en el navegador la aplicación lanzada.

8. Por Slack envíe la URL completa para acceder a la aplicación desplegada (IP:puerto). En su **reporte** incluya un pantallazo del navegador con la aplicación funcionando.

9. En la terminal de la máquina detenga el proceso con CTRL+C. Si tiene dificultades conéctese a la máquina con otra terminal, liste los contenedores disponibles con

   ```
   sudo docker ps -a
   ```

   Confirme el ID del contenedor que está corriendo y deténgalo con

```
sudo docker container stop ID
```

Verifique que el proceso de la aplicación haya terminado en la terminal donde la ejecutó.

3. Lanzando otros contenedores

1. Lance un nuevo contenedor y conéctese a la terminal interactiva del mismo usando bash con el comando

   ```
   sudo docker run -it ubuntu bash
   ```

   Copie la salida en pantalla en su **reporte**.

2. Note que en este caso ha quedado conectado a la terminal del contenedor. En la terminal del contenedor verifique la estructura de archivos con `ls`. En su **reporte** incluya las carpetas disponibles en el directorio raíz.

3. Salga de la terminal del contenedor con `exit`

4. Liste todos los contenedores disponibles con el comando

   ```
   sudo docker ps -a
   ```

   Incluya un pantallazo con el listado y sus características en su **reporte**.

5. En la VM clone el repositorio de prueba

   ```
   git clone https://github.com/dockersamples/node-bulletin-board
   ```

6. Navegue al directorio del repositorio

   ```
   cd node-bulletin-board/bulletin-board-app
   ```

7. Liste todos los archivos en esta carpeta en su **reporte**.

8. Explore el Dockerfile, ¿qué información encuentra allí? Explique los comandos en su **reporte**.

9. Explore el package.json, ¿qué información encuentra allí? Inclúyala en su **reporte**.

10. Construya la imagen

    ```
    sudo docker build --tag bulletinboard:1.0 .
    ```

11. Ejecute el contenedor con nombre bb y publíquelo en el puerto 8000

```
sudo docker run --publish 8000:8080 --detach --name bb
    bulletinboard:1.0
```

12. Liste todos los contenedores disponibles con el comando

    ```
    sudo docker ps -a
    ```

    Incluya un pantallazo con el listado y sus características en su **reporte**.

13. Abra el puerto 8000 en su instancia y explore en el navegador la aplicación lanzada. En su **reporte** explique por qué en este caso se publica en los puertos 8000:8080 y antes se hacía en 8000:8000.

14. Por Slack envíe la URL completa para acceder a la aplicación desplegada (IP+puerto).

4. Más operaciones en Docker

1. Liste las imágenes locales

   ```
   sudo docker image ls
   ```

   Incluya el resultado en su **reporte**.

2. Liste los contenedores locales en ejecución.

   ```
   sudo docker container ls
   ```

   Incluya el resultado en su **reporte**.

3. Liste todos los contenedores locales.

   ```
   sudo docker container ls -a
   ```

   Incluya el resultado en su **reporte**. Describa las diferencias entre imágenes, contenedores y contenedores en ejecucións.

4. Intente eliminar la imagen hello-world

   ```
   sudo docker image rm hello-world
   ```

   Incluya el resultado en su **reporte**. ¿Por qué no es posible eliminar la imagen?

5. Elimine imagen hello-world

   ```
   sudo docker image rm --force hello-world
   ```

   Incluya el resultado en su **reporte**. Verifique que la imagen (y el contenedor asociado) se hayan eliminado. Incluya un pantallazo de los comandos y la salida en su **reporte**.

6. Traiga la imagen de hello-world del registro

   ```
   sudo docker pull hello-world
   ```

   Incluya el resultado y su interpretación en su **reporte**. Verifique que la imagen se encuentre localmente.

7. Ejecute un contenedor con nombre holamundo a partir de la imagen hello-world

   ```
   sudo docker container run --name holamundo hello-world
   ```

   Note que al ejecutar retorna inmediatamente a la terminal de la instancia.

8. Traiga la imagen del servidor web apache (httpd) del registro

   ```
   sudo docker pull httpd
   ```

   Incluya el resultado y su interpretación en su **reporte**. Verifique que la imagen se encuentre localmente.

9. Verifique que la imagen se encuentre localmente

   ```
   sudo docker images
   ```

   Incluya un pantallazo del resultado en su **reporte**.

10. Usando esta imagen lance un contenedor con nombre docker-apache y acople el puerto 80 de la máquina con el puerto 80 del contenedor

    ```
    sudo docker run -d --name docker-apache -p 80:80 -d httpd
    ```

    El servidor debe estar corriendo por el puerto 80, luego debe poder accederlo desde el navegador solamente con la IP. Incluya un pantallazo del resultado en su **reporte**.

11. Por Slack comparta la IP de su máquina y contenedor con la página web corriendo.

12. Pruebe ahora a detener el contenedor con el comando

    ```
    sudo docker container stop docker-apache
    ```

    Verifique en su navegador que el servicio se ha detenido. Tome un pantallazo para su **reporte**.

13. Reinicie nuevamente contenedor con el comando

    ```
    sudo docker container start docker-apache
    ```

    Verifique en su navegador que el servicio ha regresado. Tome un pantallazo para su **reporte**.

14. Para ver los últimos 10 registros del log de su contenedor ejecute el comando
    `sudo docker container logs --tail 10 docker-apache`
    Incluya el resultado en su **reporte**.

15. Cree una cuenta en Docker Hub `https://hub.docker.com` y cree un repositorio público, por ejemplo `test-repo`.

16. Desde la terminal de su instancia loguéese en su cuenta usando el comando

```
sudo docker login
```

17. Taguee la imagen httpd para subirla al repositorio

```
sudo docker tag httpd usuario/repositorio:httpd
```

18. Publique su imagen para subirla al repositorio

```
sudo docker push usuario/repositorio:httpd
```

19. Envíe por Slack e incluya en su **reporte** el enlace a su repositorio con la imagen.