Uso de containers Docker

Preparar el entorno de trabajo

Instalar Docker según la bitácora en el sitio de la cátedra.

Comandos básicos de Docker

Ejecute los siguientes comando en una consola y explique que es lo que hacen usando la documentación de Docker:

Obtener imagenes de Docker Hub

```
docker pull ubuntu:14.04
docker pull nginx:1.9
```

Listando las imagenes

```
docker images
```

Ejecutando un shell dentro de un container Ubuntu 14.04

```
docker run -it ubuntu:14.04 bash
```

Para salir de la consola Presionar CTRL+p y luego CTRL+q

Para ver los containers en ejecución

```
docker ps
```

Para ver todos los containers (detenidos y en ejecución)

```
docker ps -a
```

Para reconectarse a la consola de un container:

```
docker attach 6e7a799d8965
```

Presionar la tecla <enter>

Donde 6e7a799d8965 es el ID del container al que queremos conectarnos... se pueden usar sólo los primeros N caracteres del ID (siempre que no se repitan en otro container).

También se puede usar el nombre del container (que aparece en la columna NAMES de docker ps) para el caso del container 6e7a799d8965, su nombre (autogenerado) es sad_einstein

Ejecutar un comando en un container que está corriendo:

```
docker exec sad_einstein ls
```

Si necesitamos ejecutar un comando interactivo (que requiere una consola):

```
docker exec -it sad_einstein top
```

Para pausar/reanudar todos los procesos de un container:

```
docker pause sad_einstein
docker unpause sad_einstein
```

Para parar/iniciar/reiniciar un container:

```
docker stop sad_einstein
docker start sad_einstein
docker restart sad_einstein
```

Para terminar un container (de manera incondicional):

```
docker kill sad_einstein
```

Para ver los logs de un container:

```
docker logs sad_einstein
```

Para ver las estadísticas de uso de recursos de un container:

```
docker stats sad_einstein
```

Para ver el top de los procesos de un container:

```
docker top sad_einstein
```

Eliminar un container:

```
docker rm sad_einstein
```

Manejo de imagenes:

Obtener/enviar imagenes desde/hacia Docker Hub:

```
docker pull linode/lamp
docker push mi_user/mi_imagen
```

Eliminar una imagen:

```
docker rmi linode/lamp
```

Ahora creamos un container que ejecute un shell dentro de una imagen con un servidor Nginx:

```
docker run -p 8080:80 -it nginx:1.9 bash
```

Luego iniciemos el servicio nginx

```
service nginx start
```

La alternativa es crear un container Nginx en modo dettached, con el puerto 80 expuesto y estableciéndole como nombre "nginx1":

```
docker run --name nginx1 -d -i -p 8080:80 nginx:1.9
```

Para conectarnos con un shell deberemos usar el comando:

```
docker exec -it nginx1 /bin/bash
```

Para ver más opciones de la imagen Nginx de docker ver aquí.

Crearemos unos directorios para probar la ejecución de varios containers

```
mkdir nginx
cd nginx/
mkdir sitio1
mkdir sitio2
mkdir sitio3
echo "Sitio 1" > sitio1/index.html
echo "Sitio 2" > sitio2/index.html
echo "Sitio 3" > sitio3/index.html
```

Ejecutamos tres containers apuntando a los directorios recién creados.

```
docker run --name sitio1 -p 8081:80 -v
/home/usuario/nginx/sitio1/:/usr/share/nginx/html:ro -d nginx:alpine

docker run --name sitio2 -p 8082:80 -v
/home/usuario/nginx/sitio2/:/usr/share/nginx/html:ro -d nginx:alpine

docker run --name sitio3 -p 8083:80 -v
/home/usuario/nginx/sitio3/:/usr/share/nginx/html:ro -d nginx:alpine
```

Verificamos que estén corriendo los containers con:

```
docker ps
```

Verificamos que estén accesibles en:

http://localhost:8081

http://localhost:8082

http://localhost:8083

Para ver el consumo de recursos de los containers:

```
docker stats
```

Para ver otras imágenes oficiales consultar: https://hub.docker.com/explore/