Universidad de Guadalajara
Sistema de Educación Media Superior
Centro Universitario de Ciencias Exactas e Ingenierías

Ejemplo utilizando Docker

Materia: Computación Tolerante a Fallas

D06 2023 B

Alumno: Esquivel Barbosa Diego Humberto

Código: 211401635

Carrera: INCO

Fecha: 30/10/2023

Introducción

Docker se ha convertido en una tecnología fundamental en el mundo del desarrollo de software y la gestión de aplicaciones. Permite empaquetar aplicaciones y sus dependencias en contenedores, lo que facilita la implementación, distribución y escalabilidad de software en una variedad de entornos. Esta tarea guía a través de la experiencia de trabajar con Docker para crear y gestionar contenedores, además de implementar una aplicación Node.js en un entorno de contenedorizado.

Esta tarea te guía a través del uso de Docker para crear y gestionar contenedores y aplicaciones. Al final de la actividad, deberías haber obtenido los siguientes resultados:

Verificación de Docker: Debes haber verificado si Docker está instalado en tu sistema. Si no lo tenías instalado, deberías haber descargado Docker Desktop desde https://www.docker.com/products/docker-desktop/ e instalado.

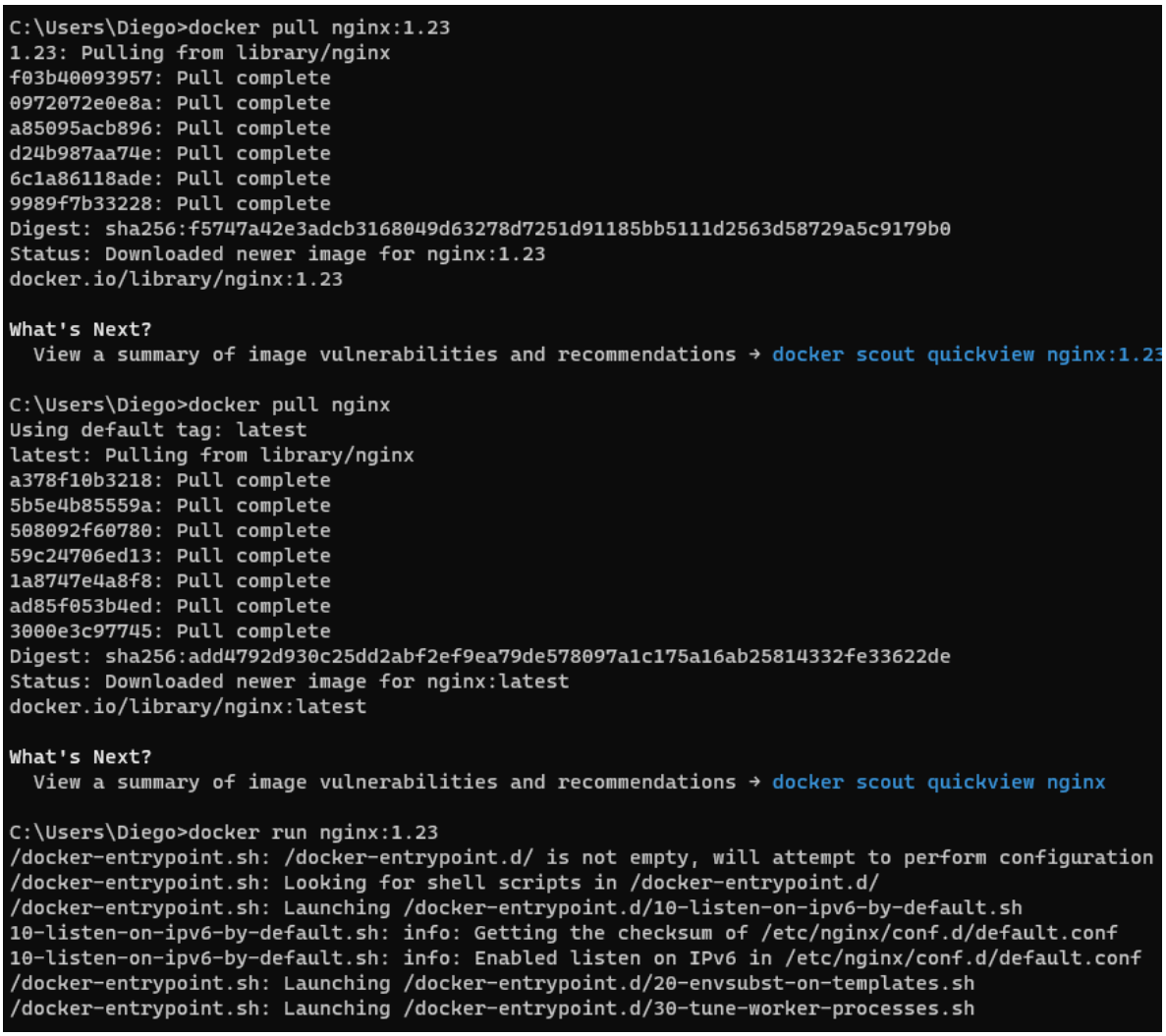
```
C:\Users\Diego>docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE

C:\Users\Diego>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

C:\Users\Diego>docker pull alpine:3.18.4
3.18.4: Pulling from library/alpine
96526aa774ef: Pull complete
Digest: sha256:eece025e432126ce23f223450a0326fbebde39cdf496a85d8c016293fc851978
Status: Downloaded newer image for alpine:3.18.4
docker.io/library/alpine:3.18.4

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview alpine:3.18.4

C:\Users\Diego>docker run -it alpine:3.18.4 sh
/ # apk update
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.18/community/x86_64/APKINDEX.tar.gz
v3.18.4-129-gdb5b5ce6661 [https://dl-cdn.alpinelinux.org/alpine/v3.18/main]
v3.18.4-130-gea49896cd30 [https://dl-cdn.alpinelinux.org/alpine/v3.18/community]
OK: 20073 distinct packages available
/ # apk upgrade
(1/7) Upgrading musl (1.2.4-r1 -> 1.2.4-r2)
(2/7) Upgrading busybox (1.36.1-r2 -> 1.36.1-r4)
Executing busybox-1.36.1-r4.post-upgrade
(3/7) Upgrading busybox-binsh (1.36.1-r2 -> 1.36.1-r4)
(4/7) Upgrading libcrypto3 (3.1.3-r0 -> 3.1.4-r0)
(5/7) Upgrading libssl3 (3.1.3-r0 -> 3.1.4-r0)
(6/7) Upgrading ssl_client (1.36.1-r2 -> 1.36.1-r4)
(7/7) Upgrading musl-utils (1.2.4-r1 -> 1.2.4-r2)
Executing busybox-1.36.1-r4.trigger
OK: 7 MiB in 15 packages
/ # apk add curl
(1/7) Installing ca-certificates (20230506-r0)
(2/7) Installing brotli-libs (1.0.9-r14)
(3/7) Installing libunistring (1.1-r1)
(4/7) Installing libidn2 (2.3.4-r1)
(5/7) Installing nghttp2-libs (1.57.0-r0)
(6/7) Installing libcurl (8.4.0-r0)
(7/7) Installing curl (8.4.0-r0)
Executing busybox-1.36.1-r4.trigger
Executing ca-certificates-20230506-r0.trigger
OK: 12 MiB in 22 packages
```



Imágenes y contenedores de Docker: Aprendiste a usar comandos como docker images y docker ps para verificar las imágenes y contenedores de Docker en tu sistema. Esto te permite ver qué imágenes tienes y qué contenedores están en ejecución.

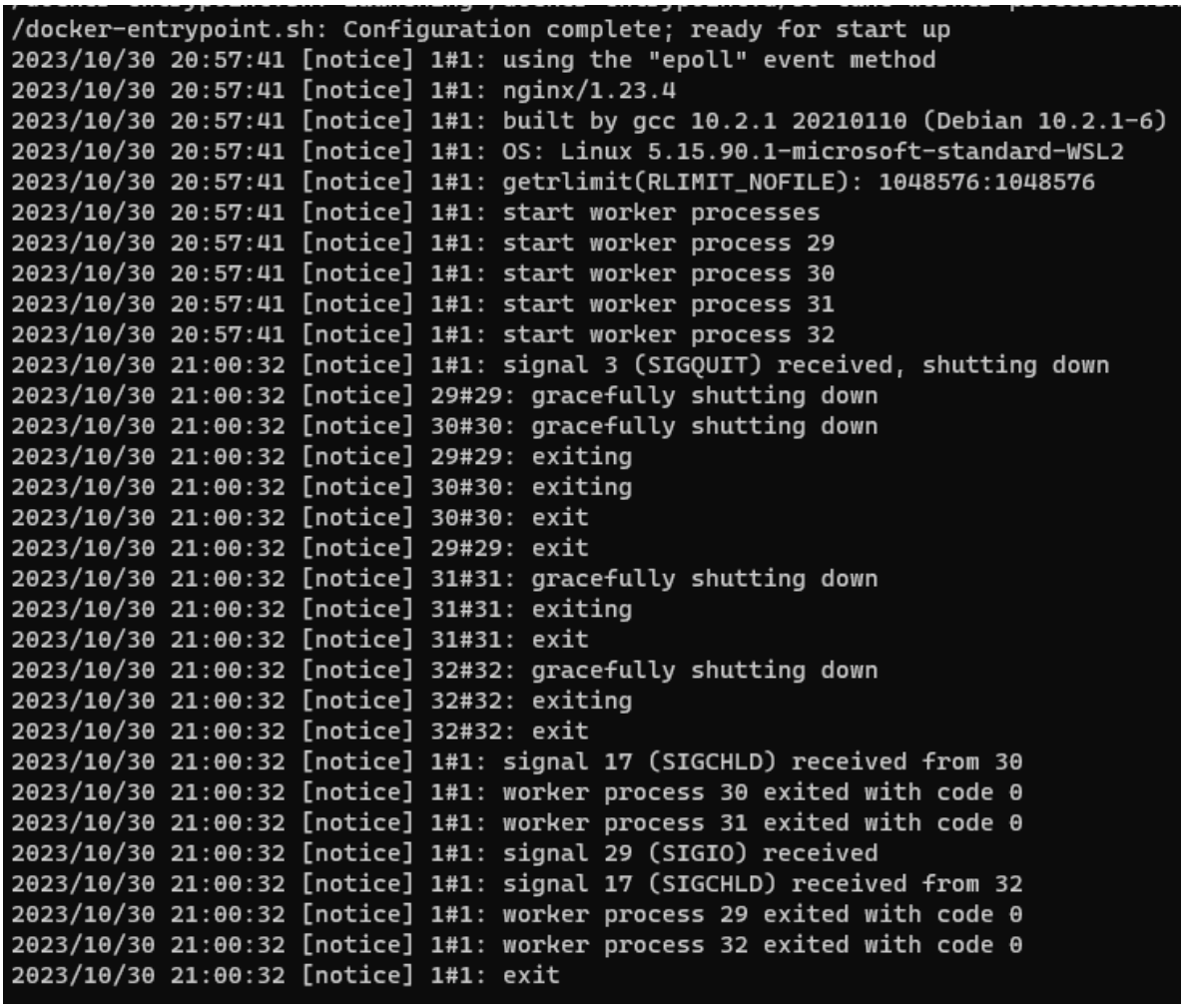
Usar Alpine Linux: Ejecutaste un contenedor interactivo de Alpine Linux, actualizaste el sistema dentro del contenedor, instalaste la herramienta curl, y realizaste una solicitud web para verificar la conectividad de red. Esto te ayudó a familiarizarte con el uso de contenedores y la ejecución de comandos en ellos.

Uso de Nginx: Descargaste y ejecutaste contenedores de Nginx para servir páginas web. Pudiste ejecutar un contenedor en segundo plano (modo demonio) y exponer un puerto, lo que te permitió acceder a la página de bienvenida de Nginx a través de un navegador web.

Creación de una aplicación Node.js: Creaste una aplicación Node.js simple utilizando Express. Esto implicó escribir código para un servidor web que muestra un mensaje de bienvenida en el puerto 3000.

Creación de un Dockerfile: Creaste un archivo llamado Dockerfile que contiene instrucciones para construir una imagen de Docker para tu aplicación Node.js. En este archivo, copiaste los archivos necesarios, instalaste las dependencias y especificaste el comando para ejecutar la aplicación.

Construcción y ejecución de la imagen: Usando Docker, construiste una imagen de tu aplicación Node.js a partir del Dockerfile. Posteriormente, ejecutaste un contenedor basado en esta imagen y expusiste el puerto 3000 de la aplicación para que sea accesible desde tu máquina local.




```
C:\Users\Diego>docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED              STATUS              PORTS     NAMES
90ec69f4eaaa   nginx:1.23   "/docker-entrypoint.…"   About a minute ago   Up About a minute   80/tcp    confident_brattain

C:\Users\Diego>docker run -d nginx:1.23
7cae402b34414c5febc9789409795429a1ec5abc8df4c32ba5dfc73fdaa01d96

C:\Users\Diego>docker logs 90ec69f4eaaa
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/10/30 20:57:41 [notice] 1#1: using the "epoll" event method
2023/10/30 20:57:41 [notice] 1#1: nginx/1.23.4
2023/10/30 20:57:41 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/10/30 20:57:41 [notice] 1#1: OS: Linux 5.15.90.1-microsoft-standard-WSL2
2023/10/30 20:57:41 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/10/30 20:57:41 [notice] 1#1: start worker processes
2023/10/30 20:57:41 [notice] 1#1: start worker process 29
2023/10/30 20:57:41 [notice] 1#1: start worker process 30
2023/10/30 20:57:41 [notice] 1#1: start worker process 31
2023/10/30 20:57:41 [notice] 1#1: start worker process 32

C:\Users\Diego>docker stop 90ec69f4eaaa
90ec69f4eaaa

C:\Users\Diego>docker run -d -p 9090:80 nginx:1.23
70d31bc05c4a44b0357dc90a161172d961f91af7de34863030c3f5f2d364a595

C:\Users\Diego>docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                          PORTS                  NAMES
70d31bc05c4a   nginx:1.23     "/docker-entrypoint.…"   47 seconds ago   Up 46 seconds                   0.0.0.0:9090->80/tcp   pensive_he
llman
7cae402b3441   nginx:1.23     "/docker-entrypoint.…"   2 minutes ago    Up 2 minutes                    80/tcp                 pensive_mc
carthy
90ec69f4eaaa   nginx:1.23     "/docker-entrypoint.…"   4 minutes ago    Exited (0) About a minute ago                          confident_
brattain
384d2c9dc701   alpine:3.18.4  "sh"                     6 minutes ago    Exited (0) 5 minutes ago                               ecstatic_c
hebyshev
```

```
C:\Users\Diego>docker start -i 1d057b59b0a5
Error response from daemon: No such container: 1d057b59b0a5

C:\Users\Diego>docker run --name mi-web-app -d -p 9090:80 nginx:1.23
5fce5958629c1499dd5cb0f73edbb63e7dde8b553b8da6c5d3ac3f6cdb625c20
docker: Error response from daemon: driver failed programming external connectivity on endpoint mi-web-app (7c97c03f0e5d4f39fae70fda29cdef
cb04fc53ba34915ee4240caefd37d9b510): Bind for 0.0.0.0:9090 failed: port is already allocated.

C:\Users\Diego>docker run --name mi-web-app -d -p 9090:80 nginx:1.23
docker: Error response from daemon: Conflict. The container name "/mi-web-app" is already in use by container "5fce5958629c1499dd5cb0f73ed
bb63e7dde8b553b8da6c5d3ac3f6cdb625c20". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.

C:\Users\Diego>mostrara ->CONTAINER ID   IMAGE        COMMAND                  CREATED         STATUS         PORTS                  NAMES

"mostrara" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Users\Diego>48ae1dbc21e7   nginx:1.23   "/docker-entrypoint.…"   4 seconds ago   Up 2 seconds   0.0.0.0:9090->80/tcp   mi-web-app
El sistema no puede encontrar la ruta especificada.

C:\Users\Diego>
```

server.js 7

C: > Users > Diego > Desktop > server.js > ...

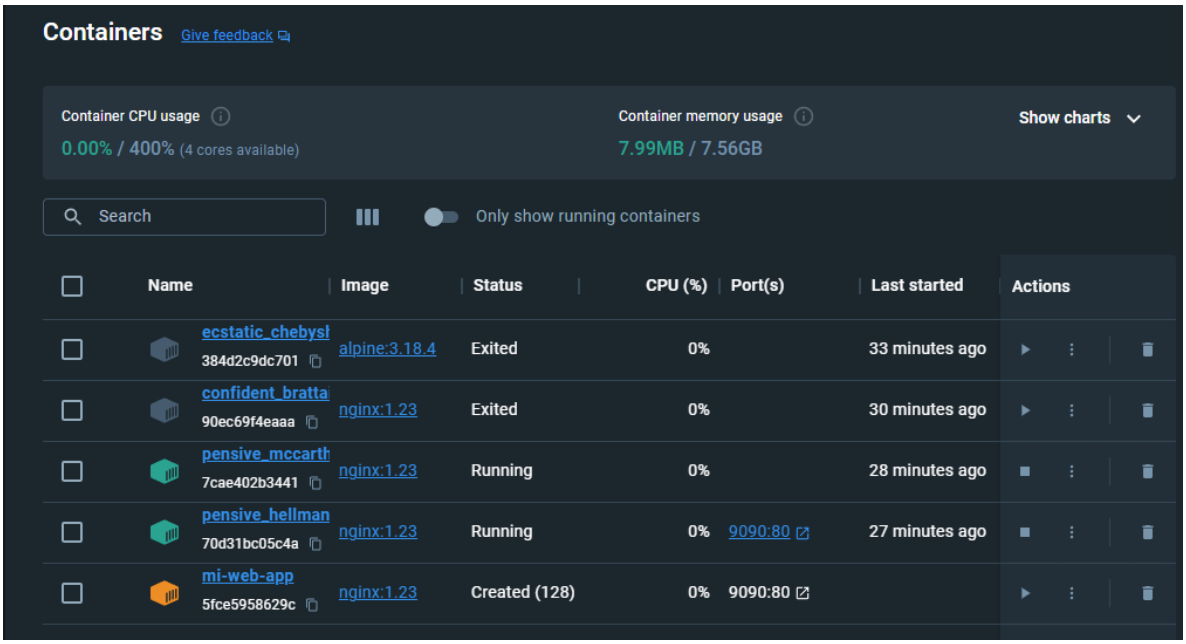
```
1   const express = require('express');
2   const app = express ();
3
4   app.get('/', (req, res) =>{
5   res. send("Welcome to my awesome app!");
6   });
7
8   app. listen (3000, function ( ) {
9   console. log ("app listening on port 3000");
10  });
11  Generar el archivo
12
13  ./package.json
14
15  {
16  "name": "my-app",
17  "version": "1.0",
18  "dependencies": {
19  "express": "4.18.2"
20  }
21  }
```

PROBLEMAS 7 SALIDA CONSOLA DE DEPURACIÓN PUERTOS POLYGLOT NOTEBOOK TERMINAL powershell

```
PS C:\Users\Diego> node src/server.js
node:internal/modules/cjs/loader:1080
  throw err;
  ^

Error: Cannot find module 'C:\Users\Diego\src\server.js'
    at Module._resolveFilename (node:internal/modules/cjs/loader:1077:15)
    at Module._load (node:internal/modules/cjs/loader:922:27)
    at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:86:12)
    at node:internal/main/run_main_module:23:47 {
  code: 'MODULE_NOT_FOUND',
  requireStack: []
}

Node.js v18.18.0
PS C:\Users\Diego> app listening on port 3000
```



Modificación y actualización: Hiciste cambios en el código de la aplicación Node.js, y luego volviste a construir la imagen de Docker. Esto te permitió detener el contenedor existente, ejecutar una nueva versión y verificar los cambios.

En resumen, al final de la tarea, deberías haber obtenido experiencia en el uso de Docker para administrar contenedores, crear imágenes personalizadas y ejecutar aplicaciones dentro de contenedores. Además, habrás practicado la creación y modificación de aplicaciones Node.js junto con el uso de Docker para construir y ejecutar estas aplicaciones. Estas habilidades son valiosas para el desarrollo y la gestión de aplicaciones en entornos de contenedores.

Conclusión

La utilización de Docker para administrar contenedores y aplicaciones brinda una mayor flexibilidad y eficiencia en el desarrollo y la implementación de software. A través de esta tarea, hemos explorado conceptos clave como la creación de imágenes personalizadas con Docker, la gestión de aplicaciones Node.js y la exposición de servicios a través de contenedores. Hemos demostrado cómo Docker simplifica el proceso de desarrollo y despliegue, facilitando la creación y administración de aplicaciones en entornos de contenedores. La habilidad para modificar, construir y ejecutar aplicaciones en contenedores brinda un marco sólido para el desarrollo de aplicaciones escalables y resistentes.

Bibliografía

Docker Desktop. (n.d.). Docker. https://www.docker.com/products/docker-desktop/

Alpine Linux. (n.d.). Alpine Linux. https://alpinelinux.org/

Node.js. (n.d.). Node.js. https://nodejs.org/

Express - Node.js web application framework. (n.d.). Express.js. https://expressjs.com/

Nginx. (n.d.). Nginx. https://www.nginx.com/

Merkle, S. (2018). Docker Containers: Build and Deploy with Kubernetes, Flannel, Cockpit, and Atomic. Apress.

Lebkov, R., & Prikshet, P. (2018). Docker: Up and Running: Shipping Reliable Containers in Production. O'Reilly Media.

Boyd, C., & Johnston, J. (2016). Docker in Action. Manning Publications.