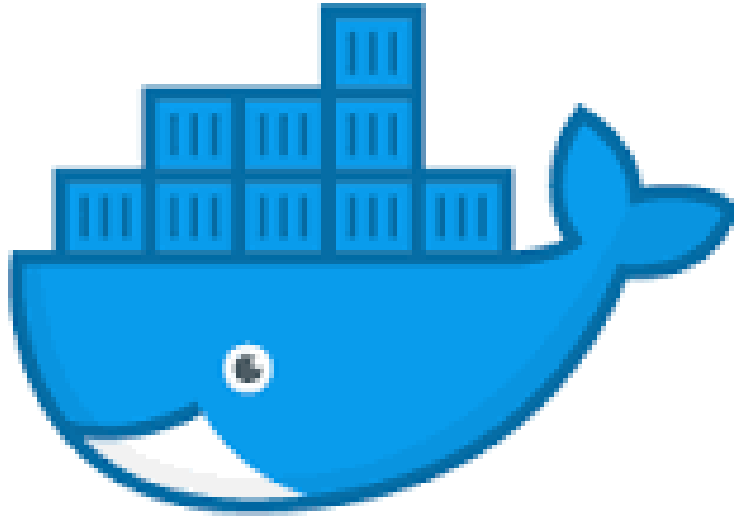


# Docker



# docker

- **Nombre:** Jesús Eduardo Jiménez Covarrubias
- **Materia:** Computación Tolerante a Fallas
- **Profesor:** Michel Emanuel López Franco
- **NRC:** 179961
- **Sección:** D06
- **Ciclo:** 2025-A

## Instrucciones:

Tarea: Utiliza Docker para generar una aplicación y si es de microservicios mejor.

<https://www.docker.com/products/docker-desktop/>

Comandos usados:

En una terminal ejecuta los siguientes comandos

```
docker images
```

```
docker ps
```

```
docker pull alpine:3.18.4
```

```
docker run -it alpine:3.18.4 sh
```

```
apk update
```

```
apk upgrade
```

```
apk add curl
```

```
curl www.google.com
```

```
exit
```

```
docker pull nginx:1.23
```

```
docker pull nginx
```

```
docker run nginx:1.23
```

En otra terminal ejecuta los siguientes

```
docker ps
```

```
docker run -d nginx:1.23
```

Recuerda que 7250a9f1d7b9 es el id de mi contenedor a ti te aparecera otro numero

```
docker logs {id}
```

Ejemplo:

```
docker logs 7250a9f1d7b9
```

```
docker stop 7250a9f1d7b9
```

Exponiendo un puerto

```
docker run -d -p 9090:80 nginx:1.23
```

Abre tu navegador y pon localhost:9090 mostrara un mensaje de bienvenida el nginx

```
docker ps -a
```

```
docker start -i 1d057b59b0a5
```

Ponerle un nombre a tu contenedor por lo general asigna los nombres aleatoriamente

```
docker run --name mi-web-app -d -p 9090:80 nginx:1.23
```

```
mostrara ->CONTAINER ID  IMAGE      COMMAND
CREATED      STATUS    PORTS      NAMES
```

```
48ae1dbc21e7  nginx:1.23  "/docker-entrypoint...."  4 seconds ago
Up 2 seconds  0.0.0.0:9090->80/tcp  mi-web-app
```

Ejemplo Node abrir vscode y generar el archivo

```
src/server.js
```

```
const express = require('express');
```

```
const app = express ();
```

```
app.get('/', (req, res) =>{
```

```
res. send("Welcome to my awesome app!");
```

```
});
```

```
app. listen (3000, function ( ) {
```

```
console. log ("app listening on port 3000");
```

```
});
```

Generar el archivo

```
./package.json
```

```
{
```

```
"name": "my-app",
```

```
"version": "1.0",
```

```
"dependencies": {
```

```
"express": "4.18.2"
```

```
}
```

```
}
```

En la terminal de vs code

```
node src/server.js
```

```
-> app listening on port 3000
```

Generar el archivo

```
./Dockerfile
```

```
FROM node:19-alpine
```

```
COPY package.json /app/
```

```
COPY src /app/
```

```
WORKDIR /app
```

```
RUN npm install
```

```
CMD [ "node","server.js" ]
```

En la terminal pon

```
docker build -t node-app:1.0 .
```

```
-t or -tag
```

Sets a name and optionally a tag in the “name:tag” format

En una terminal ve la imagen generada

```
docker images
```

```
docker run -d -p 3000:3000 node-app:1.0
```

modificar server.js

```
app.get('/', (req, res) =>{
```

```
res.send("Welcome to my awesome app v2!");
```

```
});
```

detener el contenedor y ejecutar la nueva version

```
docker stop c6a9b5578e98
```

```
docker build -t node-app:1.0 .
```

```
docker run -d -p 3000:3000 node-app:1.0
```

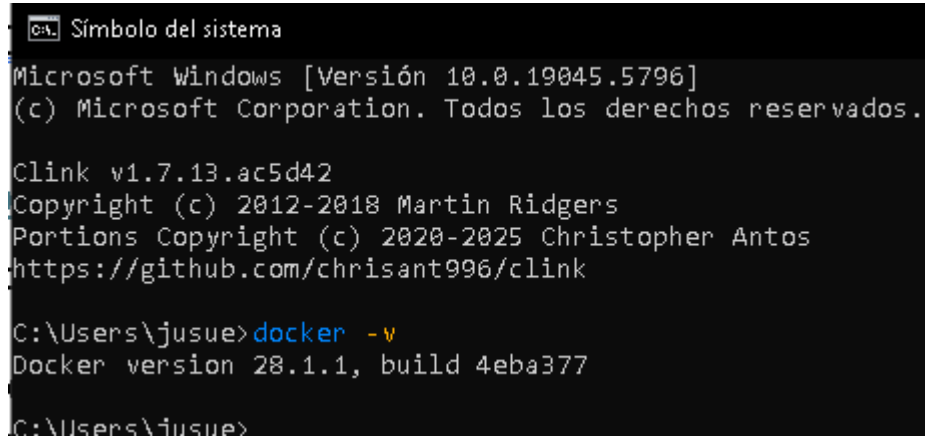
<https://github.com/michel-lopez-franco/IntroDockers>

El objetivo de esta actividad es realizar un ejemplo básico para poner en práctica la utilización de la herramienta Docker. Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos.

## **Desarrollo:**

Ejemplo básico utilizando Docker

Para el desarrollo de esta actividad, primero fue necesario realizar la instalación de Docker desde su sitio oficial, y para verificar que se instaló correctamente, desde el PowerShell se escribe la línea de comandos “docker -v” la cual nos mostrara la versión de Docker instalada.



```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.5796]
(c) Microsoft Corporation. Todos los derechos reservados.

Clink v1.7.13.ac5d42
Copyright (c) 2012-2018 Martin Ridgers
Portions Copyright (c) 2020-2025 Christopher Antos
https://github.com/chrisant996/clink

C:\Users\jusue>docker -v
Docker version 28.1.1, build 4eba377

C:\Users\jusue>
```

Ahora bien, para que Docker funcione correctamente también es necesario instalar el subsistema de Windows para Linux (WSL) en su versión 2, si todo esto se hizo correctamente, en la aplicación de Docker de nuestro computador debe abrirse sin ningún inconveniente.

Hablando a grandes rasgos de lo que hace el ejemplo, creará una aplicación Web en Python “Hola mundo” utilizando Flask framework y lo que va a hacer este contenedor es empaquetar la aplicación con todas las partes necesarias, por lo que, todas las bibliotecas y demás dependencias también se empaquetaran.

El primer paso para nuestra aplicación es generar un archivo Docker que tiene que ser llamado “Dockerfile”, y para generar el contenedor se necesitan tres cosas indispensables, la primera es el Dockerfile que ya se mencionó, la segunda es una imagen Docker y por último el contenedor en sí.

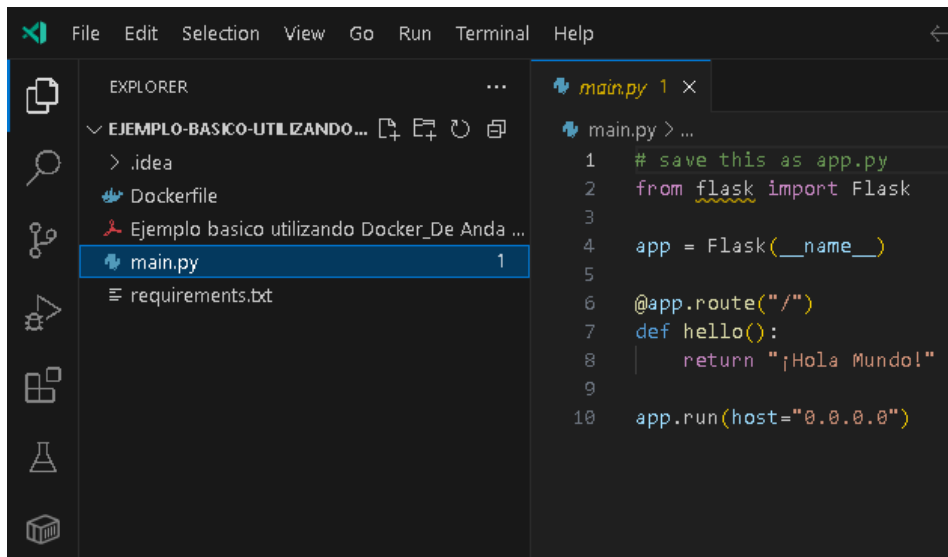
Lo que hace el Dockerfile es generar imágenes Docker, la imagen es una plantilla para ejecutar contenedores y el contenedor es el proceso en ejecución real donde tenemos nuestro proyecto empaquetado.

También, creamos un archivo llamado “requirements.txt” el cual contendrá las dependencias de nuestro archivo Python y también creamos el “main.py” que contendrá el código fuente.

La actividad pide la realización de un ejemplo básico y el objetivo principal de esta actividad es aprender a utilizar Docker. Por esta

razón el código de la página Web es un hola mundo utilizando el framework Flask de python, para instalarlo se ingresa la siguiente instrucción en la terminal de python, “pip install Flask”.

En la siguiente captura se muestra el código que generará el sitio web con el hola mundo.



```
File Edit Selection View Go Run Terminal Help
EXPLORER
EJEMPLO-BASICO-UTILIZANDO...
> .idea
Dockerfile
Ejemplo basico utilizando Docker_De Anda ...
main.py 1
requirements.txt
main.py 1 x
main.py > ...
1 # save this as app.py
2 from flask import Flask
3
4 app = Flask(__name__)
5
6 @app.route("/")
7 def hello():
8     return "¡Hola Mundo!"
9
10 app.run(host="0.0.0.0")
```

Si corremos este código nos genera el sitio en el localhost con la información correspondiente definida en el código.

