



Nombre: Eduardo Quetzal Delgado Pimentel

Código: 217239716

Fecha: 26/04/2024

Materia: Computación tolerante a fallas

Kubernetes

Introducción:

¿Qué es Kubernetes?

1. Escalabilidad automática: Permite escalar automáticamente los pods hacia arriba o hacia abajo según la carga de trabajo.
2. Balanceo de carga: Distribuye el tráfico de red entre los pods de manera equitativa para mejorar el rendimiento y la disponibilidad.
3. Auto recuperación: Detecta y reemplaza automáticamente los contenedores o pods que fallan.
4. Actualizaciones sin tiempo de inactividad: Permite actualizar las aplicaciones sin interrumpir el servicio para los usuarios finales.
5. Almacenamiento orquestado: Administra el almacenamiento adjunto a los contenedores y los pods de manera dinámica.
6. Configuración declarativa: Define el estado deseado del sistema en archivos de configuración, permitiendo que Kubernetes se encargue de mantener el estado actual alineado con el estado deseado.

En resumen, Kubernetes simplifica y automatiza el despliegue y la gestión de aplicaciones en entornos contenerizados, permitiendo a los equipos de desarrollo y operaciones enfocarse en la creación de software sin preocuparse por la infraestructura subyacente.

¿Qué es Ingress?

En Kubernetes, Ingress es un recurso que gestiona el acceso externo a los servicios dentro del clúster de Kubernetes. Permite exponer servicios HTTP y HTTPS bajo un único punto de entrada para el tráfico externo, facilitando la administración del enrutamiento del tráfico web hacia los servicios específicos que se ejecutan dentro del clúster.

Básicamente, Ingress actúa como un controlador de tráfico que dirige las solicitudes entrantes a los servicios adecuados según reglas de enrutamiento definidas. Estas reglas pueden incluir la asignación de nombres de dominio, la configuración de rutas y la definición de reglas de redireccionamiento, entre otras opciones.

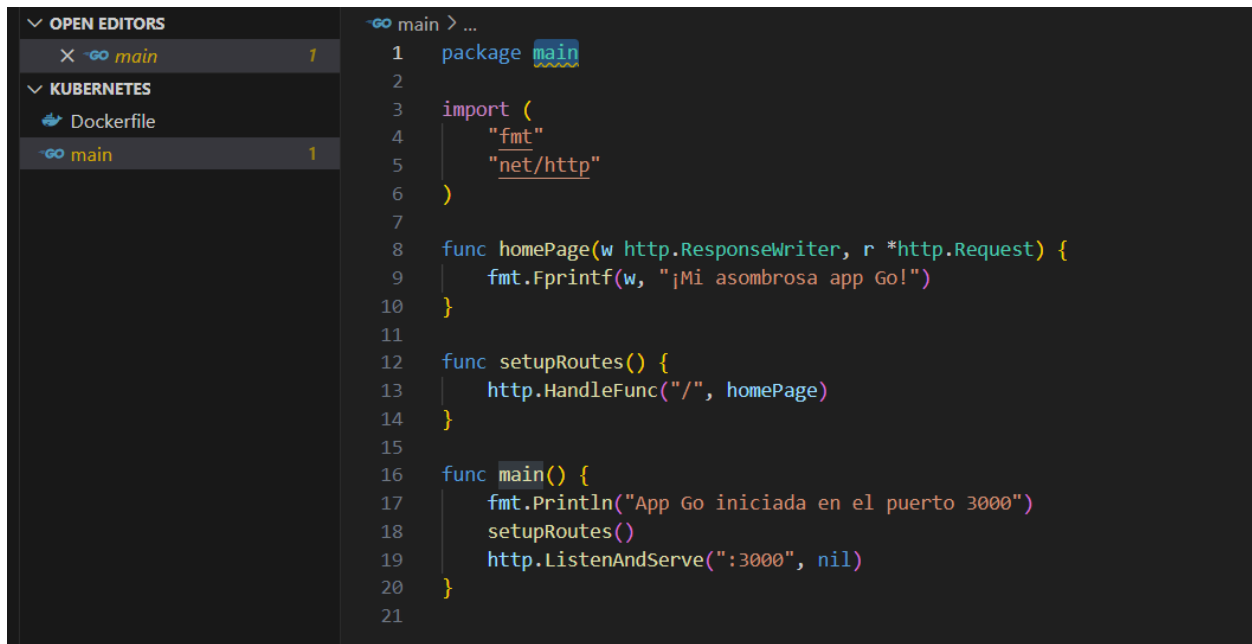
¿Qué es un LoadBalancer?

En el contexto de Kubernetes y otros entornos de infraestructura en la nube, un Load Balancer (balanceador de carga) es un componente que distribuye el tráfico de red entrante entre múltiples instancias de servidores, máquinas virtuales o contenedores para mejorar el rendimiento, la escalabilidad y la disponibilidad de las aplicaciones.

El Load Balancer actúa como un intermediario entre los clientes que envían solicitudes y los servidores o servicios que manejan esas solicitudes. Distribuye la carga de manera equitativa o según ciertos algoritmos predefinidos entre los recursos disponibles, lo que ayuda a evitar la sobrecarga de cualquier recurso individual y garantiza una mejor utilización de los recursos disponibles.

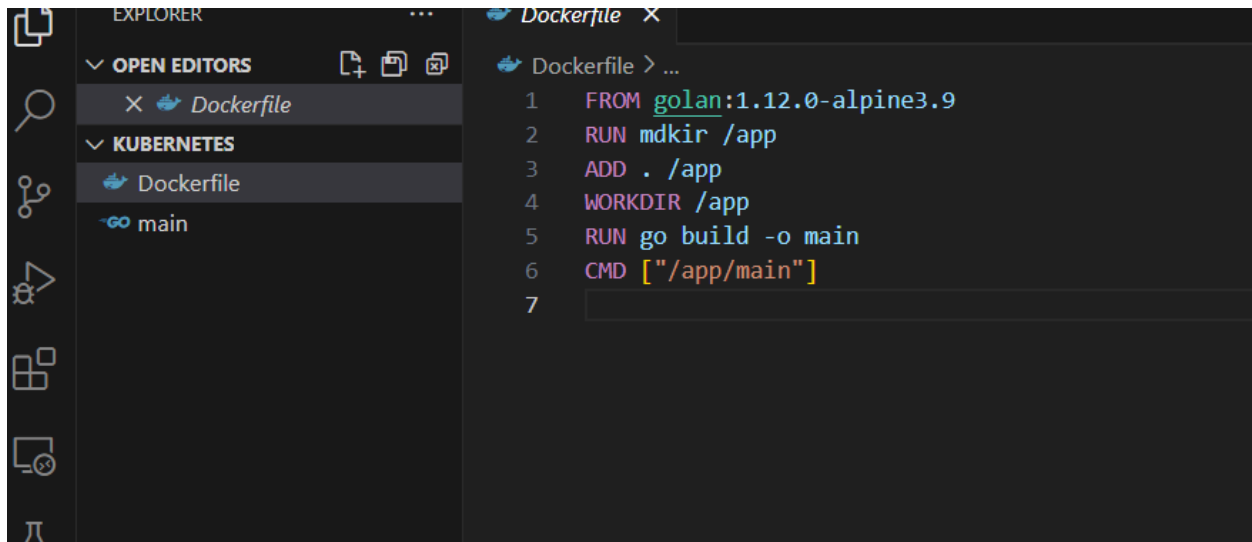
Desarrollo:

Aquí desarrollo el script de go como se ve en el video, aunque tengo varios problemas al momento de correrlo, he buscado la solución, pero parece ser algo de la configuración de mi visual, pero el script funciona, pero simplemente no hace lo que debería.



```
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6 )
7
8 func homePage(w http.ResponseWriter, r *http.Request) {
9     fmt.Fprintf(w, "¡Mi asombrosa app Go!")
10 }
11
12 func setupRoutes() {
13     http.HandleFunc("/", homePage)
14 }
15
16 func main() {
17     fmt.Println("App Go iniciada en el puerto 3000")
18     setupRoutes()
19     http.ListenAndServe(":3000", nil)
20 }
21
```

Aquí se ve el dockerfile, obviamente si no funciona el archivo go, este no hará nada ya que se conecta con el archivo anterior.

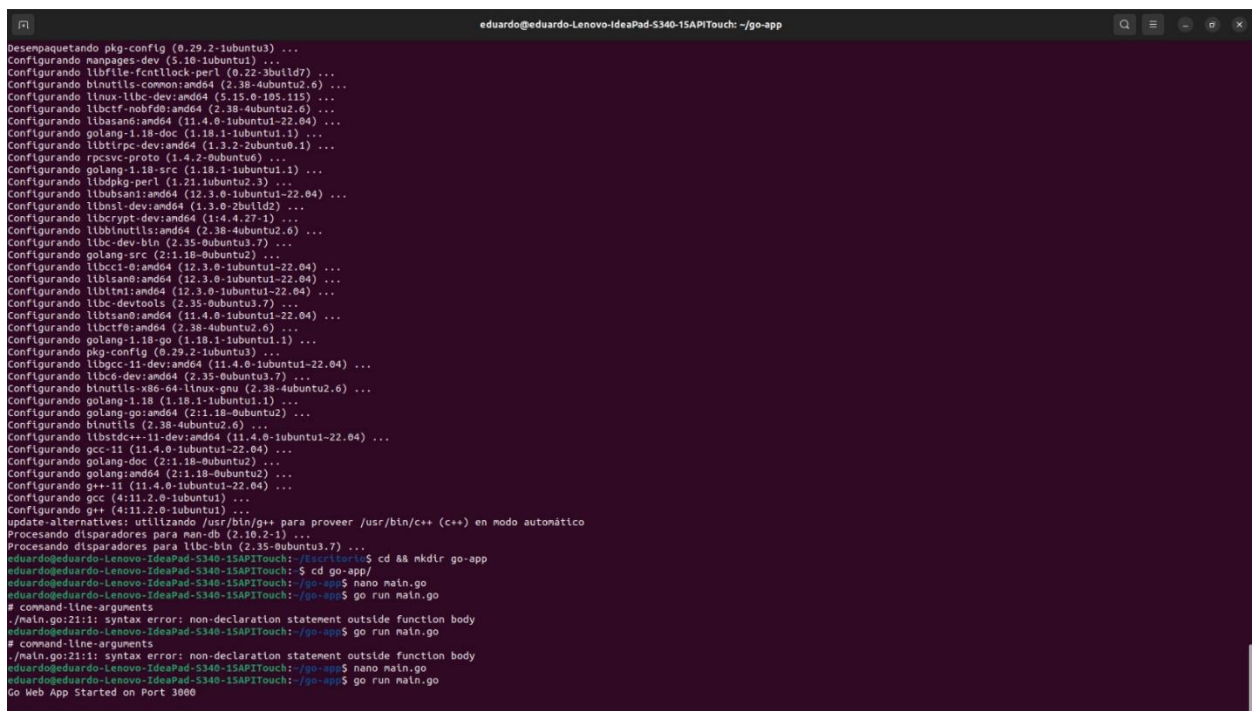


```
EXPLORER
  OPEN EDITORS
    Dockerfile
  KUBERNETES
    Dockerfile
    main
  Dockerfile
  main

Dockerfile
1 FROM golang:1.12.0-alpine3.9
2 RUN mkdir /app
3 ADD . /app
4 WORKDIR /app
5 RUN go build -o main
6 CMD ["/app/main"]
7
```

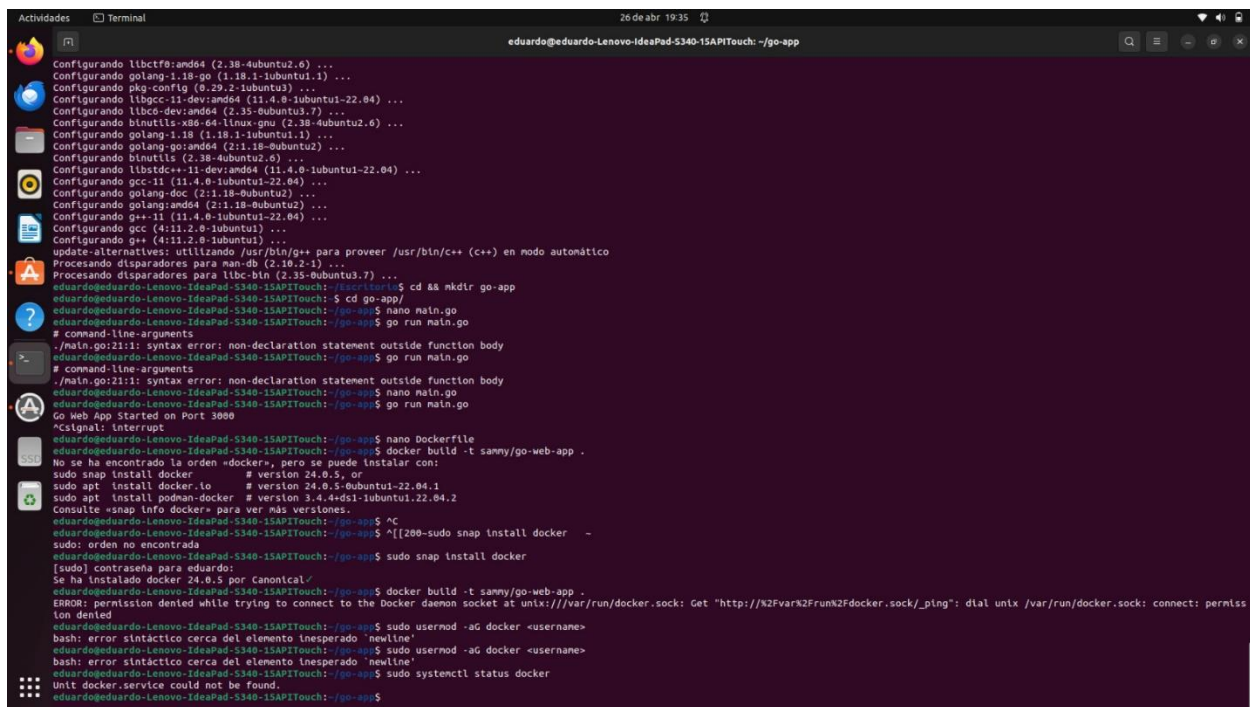
Todo lo realizado fue en Windows.

Después de no poder realizar el ejemplo lo intente en Linux.



```
eduardo@eduardo-Lenovo-IdeaPad-S340-15APITouch: ~/go-app
Desempaquetando pkg-config (0.29.2-1ubuntu3) ...
Configurando nanopages-dev (5.10-1ubuntu2) ...
Configurando libfile-fcntllock-perl (0.22-3build7) ...
Configurando binutils-common:amd64 (2.38-4ubuntu2.6) ...
Configurando linux-libc-dev:amd64 (5.15.0-105.115) ...
Configurando libctf-nobfd:amd64 (2.38-4ubuntu2.6) ...
Configurando libasan:amd64 (11.4.0-1ubuntu1-22.04) ...
Configurando golang-1.18-doc (1.18.1-1ubuntu1.1) ...
Configurando libtirpc-dev:amd64 (1.3.2-2ubuntu0.1) ...
Configurando rpsvc-proto (1.4.2-0ubuntu6) ...
Configurando golang-1.18-src (1.18.1-1ubuntu1.1) ...
Configurando libdpkg-perl (1.21.1ubuntu2.3) ...
Configurando libubsan1:amd64 (12.3.0-1ubuntu1-22.04) ...
Configurando libisl-dev:amd64 (1:0.24-1ubuntu1) ...
Configurando libcrypt-dev:amd64 (1:4.4.27-1) ...
Configurando libbinutils:amd64 (2.38-4ubuntu2.6) ...
Configurando libc-dev-bin (2.35-0ubuntu3.7) ...
Configurando golang-src (2:1.18-0ubuntu2) ...
Configurando libcc1-0:amd64 (12.3.0-1ubuntu1-22.04) ...
Configurando liblsan:amd64 (12.3.0-1ubuntu1-22.04) ...
Configurando libitm1:amd64 (12.3.0-1ubuntu1-22.04) ...
Configurando libc-devtools (2.35-0ubuntu3.7) ...
Configurando libtsan:amd64 (11.4.0-1ubuntu1-22.04) ...
Configurando libctf0:amd64 (2.38-4ubuntu2.6) ...
Configurando golang-1.18-go (1.18.1-1ubuntu1.1) ...
Configurando pkg-config (0.29.2-1ubuntu3) ...
Configurando libgcc-11-dev:amd64 (11.4.0-1ubuntu1-22.04) ...
Configurando libc6-dev:amd64 (2.35-0ubuntu3.7) ...
Configurando binutils-x86_64-linux-gnu (2.38-4ubuntu2.6) ...
Configurando golang-1.18 (1.18.1-1ubuntu1.1) ...
Configurando golang-go:amd64 (2:1.18-0ubuntu2) ...
Configurando binutils (2.38-4ubuntu2.6) ...
Configurando libstdc++-11-dev:amd64 (11.4.0-1ubuntu1-22.04) ...
Configurando gcc-11 (11.4.0-1ubuntu1-22.04) ...
Configurando golang-doc (2:1.18-0ubuntu2) ...
Configurando golang:amd64 (2:1.18-0ubuntu2) ...
Configurando g++-11 (11.4.0-1ubuntu1-22.04) ...
Configurando gcc (4:11.2.0-1ubuntu1) ...
Configurando g++ (4:11.2.0-1ubuntu1) ...
update-alternatives: utilizando /usr/bin/g++ para proveer /usr/bin/c++ (c++) en modo automático
Procesando disparadores para man-db (2.10.2-1) ...
Procesando disparadores para libc-bin (2.35-0ubuntu3.7) ...
eduardo@eduardo-Lenovo-IdeaPad-S340-15APITouch: ~/go-app$ cd && mkdir go-app
eduardo@eduardo-Lenovo-IdeaPad-S340-15APITouch: ~/go-app$ cd go-app/
eduardo@eduardo-Lenovo-IdeaPad-S340-15APITouch: ~/go-app$ nano main.go
eduardo@eduardo-Lenovo-IdeaPad-S340-15APITouch: ~/go-app$ go run main.go
# command-line-arguments
./main.go:21:1: syntax error: non-declaration statement outside function body
eduardo@eduardo-Lenovo-IdeaPad-S340-15APITouch: ~/go-app$ go run main.go
# command-line-arguments
./main.go:21:1: syntax error: non-declaration statement outside function body
eduardo@eduardo-Lenovo-IdeaPad-S340-15APITouch: ~/go-app$ nano main.go
eduardo@eduardo-Lenovo-IdeaPad-S340-15APITouch: ~/go-app$ go run main.go
Go Web App Started on Port 3000
```

Aquí si se pudo correr el script de go sin problemas, como se puede ver en la terminal.



```
Configurando libctf0:amd64 (2.38-4ubuntu2.6) ...
Configurando golang-1.18-go (1:1.18.1-1ubuntu1.1) ...
Configurando pkg-config (0.29.2-1ubuntu3) ...
Configurando libgcc-11-dev:amd64 (11.4.0-1ubuntu1-22.04) ...
Configurando libcc-dev:amd64 (2.35-4ubuntu3.7) ...
Configurando binutils-x86_64-linux-gnu (2.38-4ubuntu2.6) ...
Configurando golang-1.18 (1:1.18.1-1ubuntu1.1) ...
Configurando golang-go:amd64 (2:1.18-4ubuntu2) ...
Configurando binutils (2.38-4ubuntu2.6) ...
Configurando libstdc++-11-dev:amd64 (11.4.0-1ubuntu1-22.04) ...
Configurando gcc-11 (11.4.0-1ubuntu1-22.04) ...
Configurando golang-doc (2:1.18-4ubuntu2) ...
Configurando golang:amd64 (2:1.18-4ubuntu2) ...
Configurando g++-11 (11.4.0-1ubuntu1-22.04) ...
Configurando gcc (4:11.2.0-1ubuntu1) ...
Configurando g++ (4:11.2.0-1ubuntu1) ...
update-alternatives: utilizando /usr/bin/g++ para proveer /usr/bin/c++ (c++) en modo automatico
Procesando disparadores para man-db (2.10.2-1) ...
Procesando disparadores para libc-bin (2.35-0ubuntu3.7) ...
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ cd && mkdir go-app
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ cd go-app/
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ nano main.go
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ go run main.go
# command-line-arguments
./main.go:21:1: syntax error: non-declaration statement outside function body
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ go run main.go
# command-line-arguments
./main.go:21:1: syntax error: non-declaration statement outside function body
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ nano main.go
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ go run main.go
Go Web App Started on Port 3000
^Csignal: interrupt
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ nano Dockerfile
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ docker build -t sammy/go-web-app .
No se ha encontrado la orden «docker», pero se puede instalar con:
sudo snap install docker # version 24.0.5, or
sudo apt install docker.io # version 24.0.5-0ubuntu1-22.04.1
sudo apt install podman-docker # version 3.4.4+ds1-1ubuntu1-22.04.2
Consulte «snap info docker» para ver mas versiones.
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ ^C
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ [[200-sudo snap install docker -
sudo: orden no encontrada
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ sudo snap install docker
[sudo] contraseña para eduardo:
Se ha instalado docker 24.0.5 por Canonical
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ docker build -t sammy/go-web-app .
ERROR: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock%2Fping": dial unix /var/run/docker.sock: connect: permission denied
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ sudo usernod -aG docker <username>
bash: error sintactico cerca del elemento inesperado 'newline'
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ sudo usernod -aG docker <username>
bash: error sintactico cerca del elemento inesperado 'newline'
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$ sudo systemctl status docker
Unit docker.service could not be found.
eduardo@eduardo-Lenovo-IdeaPad-5340-15APITouch:~/go-app$
```

Aquí al momento de correr el dockerfile, que si funcionaba en Windows ahora aquí el comando no funcionaba, lo intente de varias maneras, pero simplemente no, así que no se pudo realizar bien la actividad.

Conclusión:

La actividad de intentar ejecutar y desplegar un script de Go en Kubernetes, primero en Windows y luego en Linux, proporcionó una experiencia valiosa para comprender las complejidades y desafíos de trabajar con entornos de desarrollo y contenedores en diferentes sistemas operativos. Aquí hay algunas conclusiones clave:

1. Diversidad de plataformas: La variedad de sistemas operativos en los que se pueden ejecutar aplicaciones y contenedores introduce la necesidad de comprender las diferencias y consideraciones específicas de cada plataforma. Lo que funciona en una puede no funcionar directamente en otra debido a diferencias en la configuración, herramientas disponibles y comportamiento del sistema.
2. Configuración y entorno de desarrollo: Los problemas encontrados pueden estar relacionados con la configuración del entorno de desarrollo, como variables de entorno, rutas de archivos o configuración de red. Es importante revisar detenidamente la configuración y asegurarse de que esté alineada con los requisitos del sistema y las herramientas utilizadas.
3. Compatibilidad de herramientas: Al trabajar con diferentes sistemas operativos, es crucial asegurarse de que las herramientas utilizadas sean compatibles con la plataforma en la que se ejecutan. Algunas herramientas pueden tener

características específicas de la plataforma o requerir ajustes adicionales para funcionar correctamente en entornos específicos.

4. Resolución de problemas: Enfrentar desafíos técnicos y buscar soluciones es parte integral del proceso de desarrollo de software. La experiencia de enfrentar y resolver problemas en múltiples plataformas ayuda a desarrollar habilidades de resolución de problemas y a comprender mejor el funcionamiento de los sistemas.
5. Flexibilidad y adaptabilidad: La capacidad para adaptarse a diferentes entornos y superar obstáculos es esencial en el desarrollo de software. Aprender a trabajar en diferentes plataformas y enfrentar desafíos diversos contribuye a mejorar la flexibilidad y la adaptabilidad como desarrollador.

En resumen, aunque la actividad presentó desafíos significativos al intentar ejecutar el mismo script en diferentes sistemas operativos, proporcionó una valiosa oportunidad para aprender sobre la configuración, compatibilidad de herramientas y resolución de problemas en entornos variados. Estas experiencias contribuyen al crecimiento profesional y la capacidad para abordar desafíos futuros de manera más efectiva.