

# PRACTICA CREATIVA 2

ÁLVARO GARCÍA RUIZ-ESCRIBANO  
CARMEN VAZQUEZ PEREZ DE LA CRUZ  
ALEJANDRO ARRÚÑADA MORALES

## Índice

1. Objetivos de la práctica.....	2
2. Despliegue de la aplicación en máquina virtual pesada.....	3
3. Despliegue de una aplicación monolítica usando Docker.....	5
4. Segmentación de una aplicación monolítica en microservicios utilizando docker-compose.....	6
5. Despliegue de una aplicación basada en microservicios utilizando kubernetes.....	10
6. Comparaciones.....	14
7. Anexo.....	16

## **1. Objetivos de la práctica**

El objetivo es la creación de distintos escenarios para el despliegue de una aplicación robusta y escalable utilizando las distintas tecnologías aprendidas en la asignatura.

La práctica se ha estructurado en cuatro bloques fundamentales:

- 1. Despliegue de una Aplicación Monolítica en una Máquina Virtual en Google Cloud y/o en el Escenario de la Práctica Creativa 1:**

En esta fase, se aborda la implementación de una aplicación monolítica en una máquina virtual, aprovechando las capacidades de Google Cloud y/o el entorno definido en la práctica creativa previa. Este primer paso sienta las bases para comprender los fundamentos del despliegue en la nube.

- 2. Despliegue de una Aplicación Monolítica Utilizando Docker:**

La segunda etapa se enfoca en la contenerización de la aplicación monolítica mediante Docker. Se explora la creación de un entorno aislado y reproducible que facilita la gestión y distribución eficiente de la aplicación. Esta fase permite comprender la importancia de la virtualización a nivel de contenedores y sus beneficios en términos de portabilidad.

- 3. Segmentación de una Aplicación Monolítica en Microservicios Utilizando Docker-Compose:**

El tercer bloque se adentra en la transformación de la aplicación monolítica en una arquitectura basada en microservicios, haciendo uso de Docker-Compose. Aquí, se divide la aplicación en componentes independientes y autónomos, lo que facilita el despliegue y escalabilidad individual de cada microservicio. Esta fase profundiza en la modularidad y la flexibilidad inherente a las arquitecturas de microservicios.

- 4. Despliegue de una Aplicación Basada en Microservicios Utilizando Kubernetes**

La última fase se centra en el manejo de los microservicios mediante Kubernetes. Se explora la gestión automatizada de contenedores, escalabilidad dinámica y alta disponibilidad. La implementación en Kubernetes permite una administración eficiente de los microservicios en un entorno de producción, comprendiendo así la importancia de la orquestación en arquitecturas distribuidas.

## 2. Despliegue de la aplicación en máquina virtual pesada

El despliegue de la aplicación se presenta con dos opciones distintas: una en una máquina virtual pesada en la infraestructura de Google Cloud y otra en el entorno de la práctica creativa 1. Hemos elegido instalarla en una máquina virtual alojada en Google Cloud, ya que refleja el entorno de trabajo habitual en la industria y ofrece una implementación más sencilla.

El proceso comienza creando la máquina virtual en Google Cloud. Para evitar problemas con las dependencias, optamos por la imagen de disco "Debian GNU/Linux 10 (buster)".

### Disco de arranque ?

Nombre	pc2-pesada
Tipo	Disco persistente estándar nuevo
Tamaño	10 GB
Tipo de licencia ?	Gratis
Imagen	 Debian GNU/Linux 10 (buster)

Una vez creada, configuramos una regla de firewall en el proyecto para habilitar el acceso al puerto 9080.

<a href="#">habilitar-puerto-9080</a>	Entrada	Aplicar a todas	Intervalos de IP:	tcp:9080	Permitir	1000	<a href="#">default</a>
---------------------------------------	---------	-----------------	-------------------	----------	----------	------	-------------------------

Posteriormente, accedemos a la consola de la máquina virtual mediante una conexión ssh y cargamos el archivo mvpesada.py. No hará falta realizar ninguna acción para la instalación de la aplicación ni para su ejecución, puesto que todo lo necesario a realizar está automatizado en el script. En este momento ya podemos ejecutar el script con el comando.

```
agruizesc@pc2-pesada:~$ python3 mvpesada.py
```

Este script, el cual se encuentra detallado en el anexo 1, funciona resumidamente de la siguiente manera.

Por funciones:

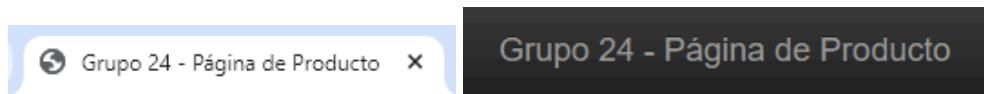
Instalar herramientas: esta función instala y actualiza todas las herramientas necesarias (python, git y pip)

Clonar repositorio: esta función clona el repositorio de github donde se encuentra la aplicación en el directorio de trabajo.

Modificar requirements: esta función elimina de requirements.txt la versión específica de requests. Esto lo hacemos debido a que si no salta un error con las versiones de alguna dependencia.

Instalar dependencias: esta función instala las dependencias especificadas en requirements.txt.

Modificar código: esta función modifica el título de la aplicación y de la pestaña de la aplicación para poner el número de nuestro grupo.



Dirección app: esta función se encarga de conseguir la dirección pública de nuestra mv y saca por pantalla la dirección de la aplicación.

```
La aplicacion va a estar accesible desde http://34.38.187.172:9080/productpage
```

Ejecutar aplicación: esta función arranca la aplicación mediante un script al cual le pasamos como parámetro el puerto en el que queremos que la aplicación reciba las peticiones (9080).

Así es como se ve la aplicación:



The screenshot shows a web application interface for a book. At the top, it says "The Comedy of Errors". Below that, there is a summary: "Summary: Wikipedia Summary: The Comedy of Errors is one of William Shakespeare's early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play." There are two tabs at the top right: "Book Details" and "Book Reviews". Under "Book Details", there is a table of information:

Type:	paperback
Pages:	65
Publisher:	Courier Corporation
Language:	English
ISBN-10:	9780486424613
ISBN-13:	9780486424613

### 3. Despliegue de una aplicación monolítica usando Docker

Para realizar el despliegue com Docker hemos creado una máquina virtual en Google Cloud idéntica a la del paso anterior, con la imagen base "Debian GNU/Linux 10 (bust)" y el puerto 9080 habilitado, para que funcione correctamente.

Una vez arrancada la máquina virtual hay que actualizar el sistema y descargar Docker ejecutando los siguientes comandos.

```
carmenvazquez2002@instance-1:~$ sudo apt update
```

```
carmenvazquez2002@instance-1:~$ sudo apt upgrade -y
```

```
carmenvazquez2002@instance-1:~$ sudo apt install docker.io
```

Cargamos a la máquina el correspondiente Dockerfile

```
FROM python:3.7.7-slim

RUN apt-get update && apt-get install -y git

RUN git clone https://github.com/CDPS-ETSIT/practica_creativa2.git

RUN pip3 install -r practica_creativa2/bookinfo/src/productpage/requirements.txt

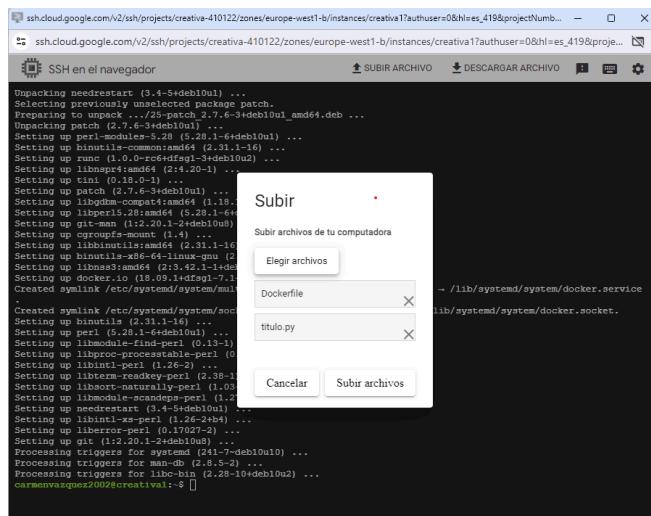
ENV env_GRUPO_NUMERO $GRUPO_NUMERO

COPY titulo.py .

EXPOSE 9080

CMD ["sh", "-c", "python3 titulo.py $GRUPO_NUMERO && python3 /practica_creativa2/bookinfo/src/productpage/productpage.py 9080"]
```

Así como un script de python que automatiza el cambio del título de la aplicación, el cual se ejecuta con el Dockerfile.



Para construir la imagen ejecutamos el comando que se nos ha indicado

```
carmenvazquez2002@creativa1:~$ sudo docker build -t grupo24/productpage --build-arg GROUPO_NUMERO=24 .
```

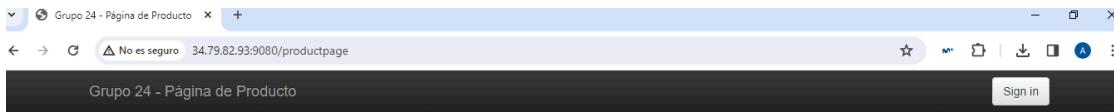
Observamos que se ha construido correctamente con el nombre indicado.

```
Successfully built 6fa827625d38
Successfully tagged grupo24/productpage:latest
```

Si ejecutamos el comando para ponerlo en marcha y luego utilizamos docker ps observamos está corriendo perfectamente.

```
carmenvazquez2002@creativa1:~$ sudo docker run --name g24-product-page -p 9080:9080 -e GROUPO_NUMERO=24 -d grupo24/productpage
c1d0e09540d9dc349dd8a22d55a2fb6d472d6c99ff9ac61ee498656d0154737
carmenvazquez2002@creativa1:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
c1d0e09540d9        grupo24/productpage   "sh -c 'python3 titu..."   About a minute ago   Up About a minute   0.0.0.0:9080->9080/tcp   g24-product-page
```

La aplicación estará accesible en la dirección <ip-publica>:9080/productpage



## 4. Segmentación de una aplicación monolítica en microservicios utilizando docker-compose

Continuaremos utilizando Google Cloud para alojar una máquina virtual donde trabajaremos con docker-compose. El escenario será idéntico, distribución Debian 10 y el puerto 9080 accesible.

Para comenzar, instalaremos docker y docker-compose en el entorno mediante los siguientes comandos:

```
agruizesc@pc2-docker-compose:~$ sudo apt update  
agruizesc@pc2-docker-compose:~$ sudo apt upgrade  
agruizesc@pc2-docker-compose:~$ sudo apt install docker.io  
agruizesc@pc2-docker-compose:~$ sudo apt install docker-compose
```

Luego, clonamos el repositorio de la aplicación:

```
agruizesc@pc2-docker-compose:~$ git clone https://github.com/CDPS-ETSIT/practica_creativa2.git
```

En este punto, subimos todos los archivos que hemos codificado: los Dockerfiles (productpage, details, ratings y reviews), los archivos docker-compose.yml (original, v1, v2 y v3), y el script de Python `control-version-reviews.py`. Todos estos archivos se encuentran detallados en el anexo 3.

Este último archivo automatiza el proceso de cambio entre versiones de reviews. Introduciendo en la consola la versión actual y la versión a la que se quiere cambiar, ajusta el archivo Dockerfile-reviews con las variables de entorno que caracterizan cada versión. También cambia entre un archivo docker-compose sin el contenedor ratings para la versión 1 o uno para las versiones 2 y 3, donde se añade este contenedor y su enlace desde el contenedor reviews. Cada versión de docker-compose.yml sirve para crear la imagen correspondiente a su versión de reviews.

Aunque para reviews se nos dice “Construir la imagen utilizando el fichero Dockerfile alojado en el directorio src/reviews/reviews-wlpcfg” hemos optado por crear otro dockerfile para reviews, aunque tiene el mismo contenido que el mencionado, esto lo hacemos así por tener organizados los dockerfiles en un mismo directorio.

Las aplicaciones en Java deben compilarse antes de ejecutarlas, como es el caso de reviews. Para hacerlo, accedemos a la ruta `practica\_creativa2/bookinfo/src/reviews/` y ejecutamos el comando:

```
agruizesc@pc2-docker-compose:~/practica_creativa2/bookinfo/src/reviews$ sudo docker run --rm -u root -v "$(pwd)"  
:/home/gradle/project -w /home/gradle/project gradle:4.8.1 gradle clean build
```

Una vez hecho esto, procedemos a arrancar los contenedores. Docker-compose, a diferencia de Docker, permite levantar los contenedores mediante la etiqueta “--build”, de forma que se encarga de generar las imágenes sin necesidad de la creación individual y manual de cada contenedor.

```
agruizesc@pc2-docker-compose:~$ sudo docker-compose up --build
```

Cada vez que deseemos cambiar de versión, primero ejecutaremos el script de Python, indicando la versión actual y a la que queremos cambiar (por defecto, comenzamos en la versión 1) y, luego, volveremos a generar las imágenes y a levantar los contenedores.

```
agruiZesc@pc2-docker-compose:~$ python3 control-version-reviews.py
Ingrese la versión actual (v1, v2 o v3): v1
Ingrese la versión deseada (v1, v2 o v3): v2
agruiZesc@pc2-docker-compose:~$ sudo docker-compose up --build
```

Este procedimiento permitirá cambiar entre las versiones de 'reviews' de manera controlada y eficiente.

La aplicación estará accesible en la dirección <https://<ip-publica>:9080/productpage>

The Comedy of Errors

Summary: Wikipedia Summary: The Comedy of Errors is one of William Shakespeare's early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

**Book Details**

Type: paperback  
Pages: 65  
Publisher: Courier Corporation  
Language: English  
ISBN-10: 0486424618  
ISBN-13: 9780486424613

**Book Reviews**

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!  
— Reviewer1

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.  
— Reviewer2

## V2

The Comedy of Errors

Summary: Wikipedia Summary: The Comedy of Errors is one of William Shakespeare's early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

**Book Details**

Type: paperback  
Pages: 65  
Publisher: Courier Corporation  
Language: English  
ISBN-10: 0486424618  
ISBN-13: 9780486424613

**Book Reviews**

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!  
— Reviewer1  
★★★★★

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.  
— Reviewer2  
★★★★★

## V3

The Comedy of Errors

Summary: Wikipedia Summary: The Comedy of Errors is one of William Shakespeare's early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

**Book Details**

Type: paperback  
Pages: 65  
Publisher: Courier Corporation  
Language: English  
ISBN-10: 0486424618  
ISBN-13: 9780486424613

**Book Reviews**

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!  
— Reviewer1  
★★★★★

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.  
— Reviewer2  
★★★★★

## Diferencias entre Docker (un solo contenedor) y Docker-Compose (multicontenedor)

Docker y Docker-Compose ofrecen enfoques distintos para la gestión de contenedores. Mientras que Docker se centra en la administración de un solo contenedor, Docker-Compose está diseñado para coordinar y gestionar múltiples contenedores como una aplicación única. En el caso de Docker, se utilizan comandos individuales para construir, ejecutar y gestionar un contenedor, con la configuración definida principalmente en un archivo llamado Dockerfile. Por otro lado, Docker-Compose utiliza un archivo YAML estructurado (docker-compose.yml) para definir la configuración de servicios, redes y volúmenes, permitiendo la gestión simultánea de

varios contenedores con un solo comando. Las ventajas de Docker-Compose incluyen su capacidad para gestionar de manera sencilla aplicaciones compuestas por múltiples microservicios, gracias a su definición declarativa y facilidad para escalar. Sin embargo, suelen presentar cierta complejidad adicional y requerir configuración específica respecto a docker. Por otro lado, Docker es más simple y eficiente para proyectos pequeños y aislados, pero puede ser menos adecuado para aplicaciones complejas con múltiples servicios interconectados. Por lo que la elección entre ambas herramientas dependerá de la complejidad y los requisitos específicos de la aplicación en cuestión.

## 5. Despliegue de una aplicación basada en microservicios utilizando kubernetes

Lo primero de todo es subir las imágenes de los contenedores creados en el apartado anterior a Docker Hub. Para ello iniciaremos sesión con docker login en el escenario de docker-compose. Solucionamos un error al loguear con el siguiente foro:

<https://stackoverflow.com/questions/51222996/docker-login-fails-on-a-server-with-no-x11-installed>

Una vez logueados procedemos a subir las imágenes de los contenedores, para ello lo primero cambiamos el nombre de las imágenes para subirlo con nuestro usuario al repositorio que hemos creado con el nombre del microservicio en el tag. Una vez hecho esto las subimos con el comando push.

Ejemplo con details:

```
agruizesc@pc2-docker-compose:~$ sudo docker tag grupo24/details:latest alvarogarci/practica-creativa2:details  
agruizesc@pc2-docker-compose:~$ sudo docker push alvarogarci/practica-creativa2:details
```

Esto lo haremos para cada microservicio (productpage, details, rating y reviews (v1, v2 y v3). Para las distintas imágenes de reviews iremos cambiando entre versiones para subir la imagen correspondiente a cada versión.

El repositorio queda de la siguiente forma:

<p>TAG <a href="#">reviews-v3</a> Last pushed a minute ago by <a href="#">alvarogarci</a></p> <p>DIGEST <a href="#">5de2d76c915</a></p>	OS/ARCH linux/amd64	LAST PULL ---	COMPRESSED SIZE 398.38 MB
<p>TAG <a href="#">reviews-v2</a> Last pushed 3 minutes ago by <a href="#">alvarogarci</a></p> <p>DIGEST <a href="#">8327a4125799</a></p>	OS/ARCH linux/amd64	LAST PULL ---	COMPRESSED SIZE 398.38 MB
<p>TAG <a href="#">ratings</a> Last pushed 5 minutes ago by <a href="#">alvarogarci</a></p> <p>DIGEST <a href="#">ef9025795752</a></p>	OS/ARCH linux/amd64	LAST PULL ---	COMPRESSED SIZE 49.41 MB
<p>TAG <a href="#">reviews-v1</a> Last pushed 28 minutes ago by <a href="#">alvarogarci</a></p> <p>DIGEST <a href="#">b058607f5b71</a></p>	OS/ARCH linux/amd64	LAST PULL ---	COMPRESSED SIZE 398.38 MB
<p>TAG <a href="#">details</a> Last pushed 31 minutes ago by <a href="#">alvarogarci</a></p> <p>DIGEST <a href="#">18e9a82b7bb9</a></p>	OS/ARCH linux/amd64	LAST PULL ---	COMPRESSED SIZE 59.59 MB
<p>TAG <a href="#">productpage</a> Last pushed 33 minutes ago by <a href="#">alvarogarci</a></p> <p>DIGEST <a href="#">d244badde3a0</a></p>	OS/ARCH linux/amd64	LAST PULL ---	COMPRESSED SIZE 74 MB

Ahora crearemos el cluster de kubernetes mediante la herramienta GKE de Google Cloud. Elegimos un total de 5 nodos y deshabilitamos la opción de autoescalado.

### Tamaño

Cantidad de nodos \*

5

El rango de direcciones del pod limita el tamaño máximo del clúster. [Más información](#)

Habilitar el escalador automático de clústeres

Cluster autoscaler automatically creates or deletes nodes based on workload needs. [Learn more](#)

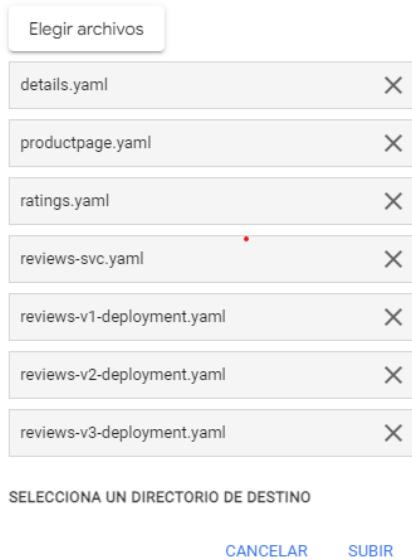
Al crear el cluster con 5 nodos se crearán automáticamente 5 instancias de vm correspondientes a estos nodos.

Ahora nos conectamos al cluster mediante la cloud shell.

Al igual que se hacía en docker compose con los archivos .yml, en kubernetes usaremos archivos .yaml para el despliegue. Para la creación de estos archivos usaremos de base el ejemplo que se nos proporciona.

Procedemos a subir todos los archivos necesarios (estos archivos se encuentran detallados en el anexo 4):

## Subir



Disponemos de un archivo de configuración .yaml para cada servicio, a excepción de reviews, que dispondremos de uno “base” y de uno para cada versión..

Ahora desplegamos los archivos de configuración, para una primera implementación de la versión uno de reviews, por lo que desplegamos productpage, details, ratings (aunque no se use para esta versión) y reviews (el “base” svc y el v1).

```
agruizesc@cloudshell:~ (gruponcio-tuentifor)$ kubectl apply -f /home/agruizesc/productpage.yaml
agruizesc@cloudshell:~ (gruponcio-tuentifor)$ kubectl apply -f /home/agruizesc/details.yaml
agruizesc@cloudshell:~ (gruponcio-tuentifor)$ kubectl apply -f /home/agruizesc/reviews-svc.yaml
agruizesc@cloudshell:~ (gruponcio-tuentifor)$ kubectl apply -f /home/agruizesc/ratings.yaml
agruizesc@cloudshell:~ (gruponcio-tuentifor)$ kubectl apply -f /home/agruizesc/reviews-v1-deployment.yaml
```

Para mostrar el estado de los pods usamos:

NAME	READY	STATUS	RESTARTS	AGE
details-v1-6b4c6d785c-6mfnm	1/1	Running	0	24m
details-v1-6b4c6d785c-g2nk5	1/1	Running	0	24m
details-v1-6b4c6d785c-n5v8j	1/1	Running	0	24m
productpage-v1-c949b47-twblod	1/1	Running	0	24m
ratings-v1-d78665b5c-4jzcx	1/1	Running	0	7m54s
ratings-v1-d78665b5c-6dqft	1/1	Running	0	7m54s
reviews-v1-694d6c46dc-548zj	1/1	Running	0	23m

El motivo por el que salen 3 pods para details y 2 para ratings es el factor de replicación, el cual es tres para details y dos para ratings según indica el enunciado.

Sacamos por pantalla la ip externa de la aplicación:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT (S)	AGE
productpage-acceso	LoadBalancer	34.118.230.141	34.175.63.228	9080:31738/TCP	21m

La aplicación ya está operativa, para acceder a ella iremos a la dirección  
<ip-externa>:9080/productpage



The Comedy of Errors

Summary: [Wikipedia Summary](#): The Comedy of Errors is one of William Shakespeare's early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

Book Details		Book Reviews
Type:	paperback	An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!
Pages:	65	— Reviewer1
Publisher:	Courier Corporation	Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.
Language:	English	— Reviewer2
ISBN-10:	0486424618	
ISBN-13:	9780486424613	

Para cambiar entre versiones primero desplegamos los archivos de configuración de las versiones de reviews faltantes y actualizamos el service para que seleccione la versión que deseamos que esté activa en ese momento. Por ejemplo, si queremos cambiar a v3 ejecutaremos el comando:

```
agruizesc@cloudshell:~ (gruponcio-tuentifor)$ kubectl patch service reviews -p '{"spec":{"selector":{"version":"v3"}}}'
```

Esto cambiará la selección del service y comenzará a enrutar el tráfico sólo hacia los pods que tengan la etiqueta version: v3.

**Incluya en la memoria de la práctica las diferencias que encuentra al crear los pods, así mismo la diferencia que ve para escalar esta última solución.**

Los pods en Kubernetes se manejan fácilmente, desde la creación hasta la eliminación. En nuestra práctica al configurar un clúster con 5 nodos, observamos automáticamente la creación de 5 instancias de estos pods.

Esto resalta la eficiencia de Kubernetes y su capacidad de escalabilidad horizontal. Con un simple comando, `kubectl scale deployments.apps/details-v1 --replicas=3` podemos replicar servicios, como "details", adaptándose en tiempo real a las necesidades del sistema..

La implementación de pods en Kubernetes se caracteriza por su simplicidad operativa y su capacidad de escalabilidad eficiente, destacando la agilidad y flexibilidad que aporta a la gestión de recursos en un entorno de contenedores.

## 6. Comparaciones

	Puntos débiles:	Soluciones:
<b>Despliegue de una aplicación monolítica en una máquina virtual en Google Cloud</b>	Falta de escalabilidad horizontal: La aplicación monolítica desplegada en una máquina virtual puede enfrentar dificultades para escalar horizontalmente, lo que limita su capacidad para manejar un aumento significativo en la carga de trabajo.	Implementar balanceo de carga: Introducir un平衡器 que distribuya el tráfico entre múltiples instancias de la aplicación para mejorar la escalabilidad horizontal.  Utilizar servicios gestionados: En lugar de depender de una máquina virtual única, considerar el uso de servicios gestionados de Google Cloud que ofrecen escalabilidad automática según la demanda.

<b>Aplicación monolítica usando Docker</b>	<p>Limitaciones en la escalabilidad y distribución: Docker, por sí solo, puede tener limitaciones en la gestión de la escalabilidad y distribución de la aplicación monolítica.</p>	<p>Orquestación con Docker Swarm o Kubernetes: Implementar herramientas de orquestación como Docker Swarm o Kubernetes para facilitar la gestión de contenedores, escalabilidad y distribución de la aplicación.</p> <p>Uso de Kubernetes para mejorar la orquestación: Kubernetes ofrece características avanzadas de orquestación, como la autoescalabilidad y la distribución equitativa de la carga, abordando las limitaciones de Docker por sí solo.</p>
<b>Segmentación de una aplicación monolítica en microservicios utilizando Docker-Compose:</b>	<p>Acoplamiento entre microservicios: Puede surgir un acoplamiento entre microservicios, lo que dificulta la independencia y escalabilidad de cada componente.</p>	<p>Diseño de interfaces claras: Definir interfaces bien definidas y mantener una comunicación clara entre los microservicios para minimizar el acoplamiento.</p> <p>Utilización de tecnologías de mensajería: Introducir tecnologías de mensajería o colas de eventos para facilitar la comunicación asíncrona y reducir el acoplamiento directo entre microservicios.</p>
<b>Despliegue de una aplicación basada en microservicios utilizando Kubernetes:</b>	<p>Complejidad de gestión: La administración de un clúster de Kubernetes y la definición de archivos de despliegue individuales pueden volverse complejas.</p>	<p>Implementación de Helm Charts: Utilizar Helm Charts para gestionar y simplificar el despliegue de aplicaciones en Kubernetes, proporcionando un paquete de configuración y definición de recursos.</p> <p>Automatización con GitOps: Implementar GitOps para automatizar la implementación y gestión continua de la infraestructura de Kubernetes, reduciendo la complejidad operativa.</p>

## 7. Anexo

### Anexo 1: Script mvpesada.py

```
1 import os
2 import subprocess
3 import json
4
5 def instalar_herramientas():
6     subprocess.run(["sudo", "apt", "update"], check=True)
7     subprocess.run(["sudo", "apt", "install", "--only-upgrade", "python3"], check=True)
8     subprocess.run(["sudo", "apt", "install", "git"], check=True)
9     subprocess.run(["sudo", "apt", "install", "python3-pip"], check=True)
10
11 def clonar_repositorio():
12     subprocess.run(["git", "clone", "https://github.com/CDPS-ETSIT/practica_creativa2.git"], check=True)
13
14 def modificar_requirements():
15     with open("practica_creativa2/bookinfo/src/productpage/requirements.txt", "r") as file:
16         lines = file.readlines()
17     with open("practica_creativa2/bookinfo/src/productpage/requirements.txt", "w") as file:
18         for line in lines:
19             if line.strip().startswith("requests=="):
20                 file.write("requests\n") # Eliminar la versión específica de requests
21             else:
22                 file.write(line)
23
24 def instalar_dependencias():
25     os.chdir('practica_creativa2/bookinfo/src/productpage')
26     subprocess.run(["pip3", "install", "-r", "requirements.txt"], check=True)
27
28 def modificar_codigo(numero_grupo):
29     with open('templates/productpage.html', 'r') as archivo:
30         contenido = archivo.read()
31
32     # Reemplazar titulo
33     nuevo_contenido = contenido.replace("BookInfo Sample", f"Grupo {numero_grupo} - Página de Producto")
34
35     # Reemplazar titulo pestaña
36     nuevo_contenido = nuevo_contenido.replace("Simple Bookstore App", f"Grupo {numero_grupo} - Página de Producto")
37
38     with open('templates/productpage.html', 'w') as archivo:
39         archivo.write(nuevo_contenido)
40
41 def direccion_app(puerto):
42     # Obtener la IP pública usando httpbin.org
43     ip_publica = subprocess.check_output(["curl", "-s", "http://httpbin.org/ip"]).decode("utf-8")
44     ip_json = json.loads(ip_publica)
45     ip_direccion = ip_json["origin"]
46     print(f"La aplicación va a estar accesible desde http://[{ip_direccion}]:{puerto}/productpage")
47
48 def ejecutar_aplicacion(puerto):
49     direccion_app(puerto)
50     subprocess.run(["python3", "productpage_monolith.py", f"{puerto}"], check=True)
51
52 def main():
53
54     # Establecer variable de entorno
55     os.environ['GRUPO_NUMERO'] = '24'
56
57     # Obtener el número de grupo desde la variable de entorno
58     numero_grupo = os.getenv('GRUPO_NUMERO')
59
60     puerto = 9080 # Puerto deseado para recibir peticiones
```

```

61     instalar_herramientas()
62     clonar_repositorio()
63     modificar_requirements()
64     instalar_dependencias()
65     modificar_codigo(numero_grupo)
66     ejecutar_aplicacion(puerto)
67
68
69 if __name__ == "__main__":
70     main()

```

## Anexo 2 Archivo parte 2

```

titulo.py ×
C: > Users > Alvaro > Desktop > Practica Creativa 2 > Docker > titulo.py > ...
1 import sys
2
3 def modificar_codigo(numero_grupo):
4
5     archivo_path = '/practica_creativa2/bookinfo/src/productpage/templates/productpage.html'
6
7     with open(archivo_path, 'r') as archivo:
8         contenido = archivo.read()
9
10    # Reemplazar título
11    nuevo_contenido = contenido.replace("BookInfo Sample", f"Grupo {numero_grupo} - Página de Producto")
12
13    # Reemplazar título pestaña
14    nuevo_contenido = nuevo_contenido.replace("Simple Bookstore App", f"Grupo {numero_grupo} - Página de Producto")
15
16    with open(archivo_path, 'w') as archivo:
17        archivo.write(nuevo_contenido)
18
19
20 if __name__ == "__main__":
21
22    if len(sys.argv) != 2:
23        print("Uso: python3 modificar_codigo.py <numero_grupo>")
24        sys.exit(1)
25
26
27 numero_grupo = sys.argv[1]
28
29 modificar_codigo(numero_grupo)
30

```

```

Dockerfile ×
C: > Users > Alvaro > Desktop > Practica Creativa 2 > Docker > Dockerfile
1 FROM python:3.7.7-slim
2
3 RUN apt-get update && apt-get install -y git
4
5 RUN git clone https://github.com/CDPS-ETSIT/practica_creativa2.git
6
7 RUN pip3 install -r practica_creativa2/bookinfo/src/productpage/requirements.txt
8
9 ARG GRUPO_NUMERO
10
11 ENV env_GRUPO_NUMERO $GRUPO_NUMERO
12
13 COPY titulo.py .
14
15 EXPOSE 9080
16
17 CMD ["sh", "-c", "python3 titulo.py $GRUPO_NUMERO && python3 /practica_creativa2/bookinfo/src/productpage/productpage.py 9080"]
18
19

```

## Anexo 3: Archivos parte 3

```
↳ Dockerfile-productpage X
↳ Dockerfile-productpage
1  FROM python:3.7.7-slim
2
3  ADD /practica_creativa2/bookinfo/src/productpage ./
4
5  EXPOSE 9080
6
7  RUN pip install -r requirements.txt
8
9  CMD ["python", "productpage.py", "9080"]
10
```

```
↳ Dockerfile-reviews X
↳ Dockerfile-reviews
1  FROM websphere-liberty:20.0.0.6-full-java8-ibmjava
2
3  ENV SERVERDIRNAME reviews
4
5  COPY /practica_creativa2/bookinfo/src/reviews/reviews-wlpcfg/servers/LibertyProjectServer /opt/ibm/wlp/usr/servers/defaultServer/
6
7  RUN /opt/ibm/wlp/bin/installUtility install --acceptLicense /opt/ibm/wlp/usr/servers/defaultServer/server.xml && \
8      chmod -R g=rwx /opt/ibm/wlp/output/defaultServer/
9
10 ARG service_version
11 ARG enable_ratings
12 ARG star_color
13 ENV SERVICE_VERSION ${service_version:-v1}
14 ENV ENABLE_RATINGS ${enable_ratings:-false}
15 ENV STAR_COLOR ${star_color:-black}
16
17 CMD ["/opt/ibm/wlp/bin/server", "run", "defaultServer"]
18
```

```
↳ Dockerfile-ratings X
↳ Dockerfile-ratings
1  FROM node:12.18.1-slim
2
3  WORKDIR /opt/microservices/
4
5  COPY /practica_creativa2/bookinfo/src/ratings/package.json ./
6  COPY /practica_creativa2/bookinfo/src/ratings/ratings.js ./
7
8  ENV SERVICE_VERSION v1
9
10 RUN npm install
11
12 EXPOSE 9080
13
14 CMD ["node", "ratings.js", "9080"]
15
```

```
↳ Dockerfile-details ✘
↳ Dockerfile-details
1  FROM ruby:2.7.1-slim
2
3  WORKDIR /opt/microservices
4
5  COPY /practica_creativa2/bookinfo/src/details/details.rb ./
6
7  ENV SERVICE_VERSION v1
8  ENV ENABLE_EXTERNAL_BOOK_SERVICE true
9
10 EXPOSE 9080
11
12 CMD ["ruby", "details.rb", "9080"]
13
```

```
↳ docker-compose.yml ✘
↳ docker-compose.yml
1  version: '3'
2
3  services:
4    productpage:
5      build:
6        context: .
7        dockerfile: Dockerfile-productpage
8        image: grupo24/productpage
9        restart: always
10       ports:
11         - 9080:9080
12       links:
13         - details
14         - reviews
15
16    details:
17      build:
18        context: .
19        dockerfile: Dockerfile-details
20        image: grupo24/details
21        restart: always
22        expose:
23          - 9080
24
25    reviews:
26      build:
27        context: .
28        dockerfile: Dockerfile-reviews
29        image: grupo24/reviews
30        restart: always
31        expose:
32          - 9080
```

```
↳ docker-compose-v1.yml ✘
  ↵ docker-compose-v1.yml
  1   version: '3'
  2
  3   services:
  4     productpage:
  5       build:
  6         context: .
  7         dockerfile: Dockerfile-productpage
  8         image: grupo24/productpage
  9         restart: always
 10        ports:
 11          - 9080:9080
 12        links:
 13          - details
 14          - reviews
 15
 16     details:
 17       build:
 18         context: .
 19         dockerfile: Dockerfile-details
 20         image: grupo24/details
 21         restart: always
 22         expose:
 23           - 9080
 24
 25     reviews:
 26       build:
 27         context: .
 28         dockerfile: Dockerfile-reviews
 29         image: grupo24/reviews
 30         restart: always
 31         expose:
 32           - 9080
```

```
↳ docker-compose-v2.yml ×
↳ docker-compose-v2.yml
 1  version: '3'
 2
 3  services:
 4    productpage:
 5      build:
 6        context: .
 7        dockerfile: Dockerfile-productpage
 8        image: grupo24/productpage
 9        restart: always
10      ports:
11        - 9080:9080
12      links:
13        - details
14        - reviews
15
16    details:
17      build:
18        context: .
19        dockerfile: Dockerfile-details
20        image: grupo24/details
21        restart: always
22      expose:
23        - 9080
24
25    reviews:
26      build:
27        context: .
28        dockerfile: Dockerfile-reviews
29        image: grupo24/reviews
30        restart: always
31      expose:
32        - 9080
33      links:
34        - ratings
35
36    ratings:
37      build:
38        context: .
39        dockerfile: Dockerfile-ratings
40        image: grupo24/ratings
41        restart: always
42      expose:
43        - 9080
```

```
35
36    ratings:
37      build:
38        context: .
39        dockerfile: Dockerfile-ratings
40        image: grupo24/ratings
41        restart: always
42      expose:
43        - 9080
```

(el mismo que docker-compose-v3.yml)

```

control-version-reviews.py ×
control-version-reviews.py > ⌂ copiar_archivo_compose
1  import fileinput
2  import subprocess
3
4  def actualizar_dockerfile(version_actual, version_deseada):
5      valores_version = {
6          'v1': {
7              'SERVICE_VERSION': '${service_version:-v1}',
8              'ENABLE_RATINGS': '${enable_ratings:-false}',
9              'STAR_COLOR': '${star_color:-black}'
10         },
11         'v2': {
12             'SERVICE_VERSION': '${service_version:-v2}',
13             'ENABLE_RATINGS': '${enable_ratings:-true}',
14             'STAR_COLOR': '${star_color:-black}'
15         },
16         'v3': {
17             'SERVICE_VERSION': '${service_version:-v3}',
18             'ENABLE_RATINGS': '${enable_ratings:-true}',
19             'STAR_COLOR': '${star_color:-red}'
20         }
21     }
22
23     if version_actual == version_deseada:
24         print(f'La versión actual y la versión deseada son las mismas ({version_actual}). No se realiza ningún cambio.')
25         return
26
27     with fileinput.FileInput('Dockerfile-reviews', inplace=True) as archivo:
28         for linea in archivo:
29             for variable, valor in valores_version[version_deseada].items():
30                 linea = linea.replace(f'${variable} ${valores_version[version_actual][variable]}',
31                                     f'${variable} ${valor}')
32         print(linea, end='')
33

```

```

34  def copiar_archivo_compose(version):
35      if version == 'v1':
36          with open('docker-compose-v1.yml', 'r') as file:
37              contenido = file.read()
38
39          with open('docker-compose.yml', 'w') as file:
40              file.write(contenido)
41      elif version == 'v2':
42          with open('docker-compose-v2.yml', 'r') as file:
43              contenido = file.read()
44
45          with open('docker-compose.yml', 'w') as file:
46              file.write(contenido)
47      elif version == 'v3':
48          with open('docker-compose-v3.yml', 'r') as file:
49              contenido = file.read()
50
51          with open('docker-compose.yml', 'w') as file:
52              file.write(contenido)
53      else:
54          print("Versión no válida")
55
56      version_actual = input("Ingrese la versión actual (v1, v2 o v3): ")
57      version_deseada = input("Ingrese la versión deseada (v1, v2 o v3): ")
58
59      actualizar_dockerfile(version_actual, version_deseada)
60      copiar_archivo_compose(version_deseada)
61
62

```

## Anexo 4 Archivos parte 4

```
*** details.yaml X
C: > Users > Alvaro > Desktop > Practica Creativa 2 > Kubernetes > *** details.yaml
 1   apiVersion: v1
 2   kind: Service
 3   metadata:
 4     name: details
 5     labels:
 6       app: details
 7       service: details
 8   spec:
 9     ports:
10       - port: 9080
11         name: http
12       selector:
13         app: details
14   ---
15   apiVersion: apps/v1
16   kind: Deployment
17   metadata:
18     name: details-v1
19     labels:
20       app: details
21       version: v1
22   spec:
23     replicas: 3
24     selector:
25       matchLabels:
26         app: details
27         version: v1
28     template:
29       metadata:
30         labels:
31           app: details
32           version: v1
33     spec:
34       containers:
35         - name: details
36           image: alvarogarci/practica-creativa2:details
37           imagePullPolicy: IfNotPresent
38           ports:
39             - containerPort: 9080
40           securityContext:
41             runAsUser: 1000
42   ---
43
```

```
*** ratings.yaml X
C: > Users > Alvaro > Desktop > Practica Creativa 2 > Kubernetes > *** ratings.yaml
 1  apiVersion: v1
 2  kind: Service
 3  metadata:
 4    name: ratings
 5    labels:
 6      app: ratings
 7      service: ratings
 8  spec:
 9    ports:
10      - port: 9080
11        name: http
12        selector:
13          app: ratings
14    ---
15  apiVersion: apps/v1
16  kind: Deployment
17  metadata:
18    name: ratings-v1
19    labels:
20      app: ratings
21      version: v1
22  spec:
23    replicas: 2
24    selector:
25      matchLabels:
26        app: ratings
27        version: v1
28    template:
29      metadata:
30        labels:
31          app: ratings
32          version: v1
33      spec:
34        containers:
35          - name: ratings
36            image: alvarogarci/practica-creativa2:ratings
37            imagePullPolicy: IfNotPresent
38            ports:
39              - containerPort: 9080
40            securityContext:
41              runAsUser: 1000
42    ---
43
```

```
** productpage.yaml X
C: > Users > Alvaro > Desktop > Practica Creativa 2 > Kubernetes > ** productpage.yaml
14  ---
15  apiVersion: apps/v1
16  kind: Deployment
17  metadata:
18    name: productpage-v1
19    labels:
20      app: productpage
21      version: v1
22  spec:
23    replicas: 1
24    selector:
25      matchLabels:
26        app: productpage
27        version: v1
28    template:
29      metadata:
30        labels:
31          app: productpage
32          version: v1
33      spec:
34        containers:
35          - name: productpage
36            image: alvarogarci/practica-creativa2:productpage
37            imagePullPolicy: IfNotPresent
38            ports:
39              - containerPort: 9080
40            securityContext:
41              runAsUser: 1000
42    ---
43  apiVersion: v1
44  kind: Service
45  metadata:
46    name: productpage-acceso
47  spec:
48    type: LoadBalancer
49    selector:
50      app: productpage
51    ports:
52      - name: http
53        port: 9080
54        targetPort: 9080
55    ---
```

```
** reviews-svc.yaml X
C: > Users > Alvaro > Desktop > Practica Creativa 2 > Kubernetes > ** reviews-svc.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: reviews
5    labels:
6      app: reviews
7      service: reviews
8  spec:
9    ports:
10      - port: 9080
11        name: http
12    selector:
13      app: reviews
14    ---
```

```
*** reviews-v1-deployment.yaml ***
C: > Users > Alvaro > Desktop > Practica Creativa 2 > Kubernetes > *** reviews-v1-deployment.yaml
 1   apiVersion: apps/v1
 2   kind: Deployment
 3   metadata:
 4     name: reviews-v1
 5     labels:
 6       app: reviews
 7       version: v1
 8   spec:
 9     replicas: 1
10     selector:
11       matchLabels:
12         app: reviews
13         version: v1
14     template:
15       metadata:
16         labels:
17           app: reviews
18           version: v1
19     spec:
20       containers:
21         - name: reviews
22           image: alvarogarci/practica-creativa2:reviews-v1
23           imagePullPolicy: IfNotPresent
24           env:
25             - name: LOG_DIR
26               value: "/tmp/logs"
27           ports:
28             - containerPort: 9080
29           volumeMounts:
30             - name: tmp
31               mountPath: /tmp
32             - name: wlp-output
33               mountPath: /opt/ibm/wlp/output
34           securityContext:
35             runAsUser: 1000
36           volumes:
37             - name: wlp-output
38               emptyDir: {}
39             - name: tmp
40               emptyDir: {}
41
```

```
*** reviews-v2-deployment.yaml ***
C: > Users > Alvaro > Desktop > Practica Creativa 2 > Kubernetes > *** reviews-v2-deployment.yaml
 1   apiVersion: apps/v1
 2   kind: Deployment
 3   metadata:
 4     name: reviews-v2
 5     labels:
 6       app: reviews
 7       version: v2
 8   spec:
 9     replicas: 1
10     selector:
11       matchLabels:
12         app: reviews
13         version: v2
14     template:
15       metadata:
16         labels:
17           app: reviews
18           version: v2
19     spec:
20       containers:
21         - name: reviews
22           image: alvarogarci/practica-creativa2:reviews-v2
23           imagePullPolicy: IfNotPresent
24           env:
25             - name: LOG_DIR
26               value: "/tmp/logs"
27           ports:
28             - containerPort: 9080
29           volumeMounts:
30             - name: tmp
31               mountPath: /tmp
32             - name: wlp-output
33               mountPath: /opt/ibm/wlp/output
34           securityContext:
35             runAsUser: 1000
36           volumes:
37             - name: wlp-output
38               emptyDir: {}
39             - name: tmp
40               emptyDir: {}
41   ---
42
```

```
⌘ reviews-v3-deployment.yaml ✘
C: > Users > Alvaro > Desktop > Practica Creativa 2 > Kubernetes > ⌘ reviews-v3-deployment.yaml
  1   apiVersion: apps/v1
  2     kind: Deployment
  3   metadata:
  4     name: reviews-v3
  5   labels:
  6     app: reviews
  7     version: v3
  8   spec:
  9     replicas: 1
10   selector:
11     matchLabels:
12       app: reviews
13       version: v3
14   template:
15     metadata:
16       labels:
17         app: reviews
18         version: v3
19     spec:
20       containers:
21         - name: reviews
22           image: alvarogarci/practica-creativa2:reviews-v3
23           imagePullPolicy: IfNotPresent
24           env:
25             - name: LOG_DIR
26               value: "/tmp/logs"
27             ports:
28               - containerPort: 9080
29             volumeMounts:
30               - name: tmp
31                 mountPath: /tmp
32               - name: wlp-output
33                 mountPath: /opt/ibm/wlp/output
34             securityContext:
35               runAsUser: 1000
36             volumes:
37               - name: wlp-output
38                 emptyDir: {}
39               - name: tmp
40                 emptyDir: {}
41   ---
42
```