PRÁCTICA 6:
KUBERNETES CON LOAD BALANCER Y AWS

EVELYN ARCENTALES

OBJETIVO

desplegar múltiples réplicas de una aplicación en Kubernetes y configurar un Load Balancer interno para distribuir el tráfico entre ellas de manera automática

En esta práctica se utilizara

- **Instalar Kubernetes local** directamente en WSL2 utilizando una versión ligera como k3s.
- **Conectar clúster local con instancias EC2 de AWS** para verificar el funcionamiento del balanceo de carga desde la nube.
- **Gestionar el escalado dinámico** de aplicaciones, aumentando o disminuyendo el número de pods según sea necesario.

OBJETIVO **1**
1. Preparación del Entorno (WSL2) **2**
0.2 – Instalar kubectl localmente (para comodidad) 4
2.4 – Crear Service (Load Balancer) 5
PARTE 1: CREAR APLICACIÓN SIMPLE PARA KUBERNETES **6**
1.1 – Crear carpeta para la aplicación 6
1.2 – Crear aplicación Python con Flask 6
1.3 – Crear archivo HTML 7
1.4 – Crear requirements.txt 8
1.5 – Crear Dockerfile ultraligero 9
PARTE 2: CONFIGURAR KUBERNETES (MANIFESTS YAML) **9**
2.2 – Crear ConfigMap con archivos de la app 10
2.3 – Crear Deployment con 3 replicas 11
2.4 – Crear Service (Load Balancer) 13
PARTE 3: VERIFICAR BALANCEO DE CARGA LOCAL **14**
3.1 – Levantar la aplicación 14
3.2 – Probar balanceo con curl 15
3.3 – Verificar en navegador 15
PARTE 4: CONFIGURACIÓN EN AWS **16**
4.1 – Crear Security Group 16
4.2 – Crear instancia EC2 17
4.3 – Conectar a EC2 17
PARTE 5: CONECTAR KUBERNETES LOCAL CON AWS EC2 ..... **18**
5.1 – Crear túnel SSH que expone el LoadBalancer 18

1. Preparación del Entorno (WSL2)

# Actualizar sistema
sudo apt update – y && sudo apt upgrade -y

```
evelyn@LAPTOP-UQ8ANGBR:~$ sudo apt update -y
[sudo] password for evelyn:
Hit:1 http://archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1391 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1684 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [225 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.5 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [9504 B]
Get:10 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [916 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [207 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble-updates/main Translation-en [311 kB]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [175 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [15.8 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1505 kB]
Get:16 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [71.4 kB]
Get:17 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [19.4 kB]
Get:18 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [2286 kB]
Get:19 http://archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [305 kB]
Get:20 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [377 kB]
Get:21 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [31.4 kB]
Get:22 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2413 kB]
Get:23 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [523 kB]
Get:24 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:25 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Get:26 http://archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [550 kB]
Get:27 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:28 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [30.4 kB]
Get:29 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:30 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7140 B]
Get:31 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [29.2 kB]
Get:32 http://archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [17.6 kB]
Get:33 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [10.9 kB]
Get:34 http://archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:35 http://archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Fetched 13.5 MB in 3s (5150 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
77 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

# Instalar dependencias mínimas
sudo apt install -y curl wget git

```
evelyn@LAPTOP-UQ8ANGBR:~$ sudo apt install -y curl wget git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
wget is already the newest version (1.21.4-1ubuntu4.1).
wget set to manually installed.
git is already the newest version (1:2.43.0-1ubuntu7.3).
git set to manually installed.
The following package was automatically installed and is no longer required:
  libllvm19
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 23 not upgraded.
```

Instalar k3s SIN systemd (mejor para WSL2)
curl -sfL https://get.k3s.io | K3S_KUBECONFIG_MODE="644" sh -

```
evelyn@LAPTOP-UQ8ANGBR:~$ curl -sfL https://get.k3s.io | K3S_KUBECONFIG_MODE="644" sh -
[INFO]  Finding release for channel stable
[INFO]  Using v1.33.6+k3s1 as release
[INFO]  Downloading hash https://github.com/k3s-io/k3s/releases/download/v1.33.6+k3s1/sha256sum-amd64.txt
[INFO]  Downloading binary https://github.com/k3s-io/k3s/releases/download/v1.33.6+k3s1/k3s
[INFO]  Verifying binary download
[INFO]  Installing k3s to /usr/local/bin/k3s
[INFO]  Skipping installation of SELinux RPM
[INFO]  Creating /usr/local/bin/kubectl symlink to k3s
[INFO]  Creating /usr/local/bin/crictl symlink to k3s
[INFO]  Skipping /usr/local/bin/ctr symlink to k3s, command exists in PATH at /usr/bin/ctr
[INFO]  Creating killall script /usr/local/bin/k3s-killall.sh
[INFO]  Creating uninstall script /usr/local/bin/k3s-uninstall.sh
[INFO]  env: Creating environment file /etc/systemd/system/k3s.service.env
[INFO]  systemd: Creating service file /etc/systemd/system/k3s.service
[INFO]  systemd: Enabling k3s unit
Created symlink /etc/systemd/system/multi-user.target.wants/k3s.service → /etc/systemd/system/k3s.service.
[INFO]  systemd: Starting k3s
evelyn@LAPTOP-UQ8ANGBR:~$
```

# Verificar que k3s está instalado
sudo k3s --version

```
evelyn@LAPTOP-UQ8ANGBR:~$ sudo k3s --version
k3s version v1.33.6+k3s1 (b5847677)
go version go1.24.9
evelyn@LAPTOP-UQ8ANGBR:~$
```

# Iniciar k3s y esperar 10 segundos a que inicie
sudo k3s server & sleep 10

```
Error: failed to create listener: failed to listen on 127.0.0.1:6444: listen tcp 127.0.0.1:6444: bind: address alread
y in use
ERRO[0001] Shutdown request received: apiserver exited: failed to create listener: failed to listen on 127.0.0.1:6444
: listen tcp 127.0.0.1:6444: bind: address already in use
INFO[0001] Waiting up to 2s for graceful shutdown of Kine GRPC server...
INFO[0001] TTL events watch channel closed
INFO[0001] Closing database connections...
INFO[0001] TTL events work queue has shut down
[1]+  Done                    sudo k3s server
```

Error: failed to create listener: failed to listen on 127.0.0.1:6443: bind: address already in use
Al generar el comando correspondiente, Esto sucede porque **k3s ya está ocupando ese puerto**.

Se utiliza el comando sudo pkill -f k3s para detener cualquier intento de k3s anterior.

# Verificar que está corriendo
sudo k3s kubectl get nodes

```
evelyn@LAPTOP-UQ8ANGBR:~$ kubectl get nodes
WARN[0000] Unable to read /etc/rancher/k3s/k3s.yaml, please start server with --wri
te-kubeconfig-mode or --write-kubeconfig-group to modify kube config permissions
error: error loading config file "/etc/rancher/k3s/k3s.yaml": open /etc/rancher/k3s
/k3s.yaml: permission denied
evelyn@LAPTOP-UQ8ANGBR:~$
```

0.2 – Instalar kubectl localmente (para comodidad)

Descargar kubectl
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

```
evelyn@LAPTOP-UQ8ANGBR:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    718      0 --:--:-- --:--:-- --:--:--   722
100 55.8M  100 55.8M    0     0  2392k      0  0:00:23  0:00:23 --:--:-- 2602k
evelyn@LAPTOP-UQ8ANGBR:~$
```

Con este comando se verifica que se descargaron 55.8M el proceso fue al 100% el ejecutable de kubectl esta en la carpeta actual

sudo chmod +x kubectl

```
evelyn@LAPTOP-UQ8ANGBR:~$ sudo chmod +x kubectl
[sudo] password for evelyn:
evelyn@LAPTOP-UQ8ANGBR:~$
```

Aquí estamos ejecutando la instrucción con privilegios de superusuario, que son necesarios para cambiar propiedades de archivos del sistema, +x estamos otorgando permisos de ejecución al archivo.
Acabamos de descargar la herramienta kubectl, que es el centro de control para manejar clúster de Kubernetes. Al ejecutar este comando, este archivo es descargado en un programa ejecutable funcional.

sudo mv kubectl /usr/local/bin/

```
evelyn@LAPTOP-UQ8ANGBR:~$ sudo mv kubectl /usr/local/bin/
evelyn@LAPTOP-UQ8ANGBR:~$
```

Verificar
kubectl version --client

```
evelyn@LAPTOP-UQ8ANGBR:~$ kubectl version --client
Client Version: v1.35.0
Kustomize Version: v5.7.1
evelyn@LAPTOP-UQ8ANGBR:~$
```

2.4 – Crear Service (Load Balancer)

este comando está confirmando que la herramienta de control ha sido instalada correctamente

Configurar kubeconfig para acceder sin sudo
mkdir -p $HOME/.kube

```
evelyn@LAPTOP-UQ8ANGBR:~$ mkdir -p $HOME/.kube
evelyn@LAPTOP-UQ8ANGBR:~$
```

sudo cp /etc/rancher/k3s/k3s.yaml $HOME/.kube/config

```
evelyn@LAPTOP-UQ8ANGBR:~$ sudo cp /etc/rancher/k3s/k3s.yaml $HOME/.kube/config
evelyn@LAPTOP-UQ8ANGBR:~$
```

sudo chmod 600 $HOME/.kube/config

```
evelyn@LAPTOP-UQ8ANGBR:~$ sudo chmod 600 $HOME/.kube/config
```

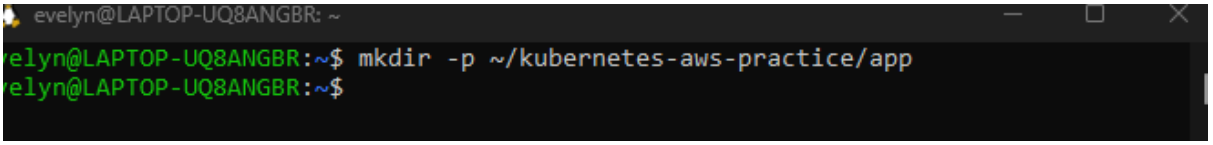
Verificar que funciona sin sudo
kubectl get nodes

```
evelyn@LAPTOP-UQ8ANGBR:~$ kubectl get nodes
NAME              STATUS   ROLES                  AGE     VERSION
laptop-uq8angbr   Ready    control-plane,master   4d21h   v1.33.6+k3s1
evelyn@LAPTOP-UQ8ANGBR:~$
```

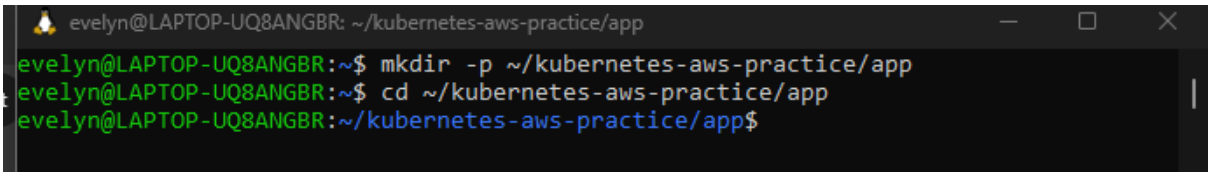
PARTE 1: CREAR APLICACIÓN SIMPLE PARA KUBERNETES

1.1 – Crear carpeta para la aplicación

mkdir -p ~/kubernetes-aws-practice/app

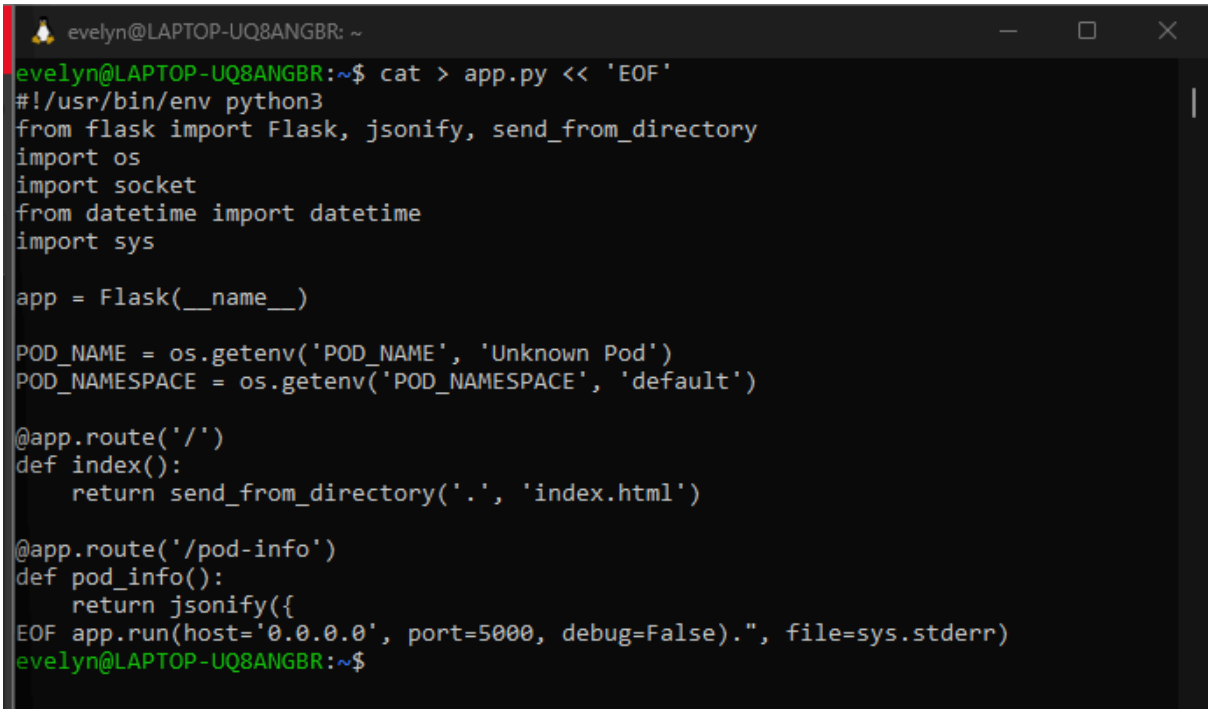


cd ~/kubernetes-aws-practice/app



1.2 – Crear aplicación Python con Flask

Crear app.py:
Aquí vamos a crear una app web sencilla usando Flask que es un framework de Python, esta app mostrará información del Pod que esta haciendo la solicitud .



pasamo a dar permisos de ejecución al archivo:

sudo chmod +x app.py

```
evelyn@LAPTOP-UQ8ANGBR:~$ sudo chmod +x app.py
[sudo] password for evelyn:
evelyn@LAPTOP-UQ8ANGBR:~$
```

Como hemos indicado anteriormente hemos creado este archivo para crear una app web utilizando el microframework Flask, siendo su función servir como el código fuente que correrá dentro de cada uno de los Pods de Kubernetes.

1.3 – Crear archivo HTML

Con este archivo se podrá ver en el navegador que Pod nos está atendiendo, es la interfaz gráfica que se verá en el navegador y permitirá cumplir con el objetivo que es monitorear el tráfico y verificar el balanceo de carga.

evelyn@LAPTOP-UQ8ANGBR: ~

```
yn@LAPTOP-UQ8ANGBR:~$ ^[[200~cat > index.html << 'EOF'
DOCTYPE html>
tml>
ead>
  <title>Kubernetes Load Balancer</title>
  <style>
      body {
          font-family: Arial, sans-serif;
          display: flex;
          justify-content: center;
          align-items: center;
          min-height: 100vh;
          margin: 0;
          background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
      }
      .container {
          background: white;
          padding: 50px;
          border-radius: 10px;
          box-shadow: 0 10px 25px rgba(0,0,0,0.2);
          text-align: center;
          max-width: 500px;
      }
      h1 { color: #667eea; margin: 0 0 30px 0; }
      .info { background: #f0f0f0; padding: 20px; border-radius: 5px; margin: 20px 0; }
      .pod-name { font-size: 28px; color: #764ba2; font-weight: bold; font-family: monospace; }
      .timestamp { color: #666; font-size: 13px; margin-top: 10px; }
      .instruction { background: #e0f2fe; padding: 15px; border-radius: 5px; color: #0369a1; font-size: 14px; margin-top: 20px; }
      .refresh-button { margin-top: 20px; padding: 10px 20px; background: #667eea; color: white; border: none; border-radius: 5px; cursor
nter; font-size: 14px; }
      .refresh-button:hover { background: #764ba2; }
  </style>
head>
ody>
  <div class="container">
      <h1>Kubernetes Load Balancer</h1>
      <div class="info">
          <p style="margin: 0 0 10px 0; color: #666;">Pod atendiendo solicitud:</p>
          <p class="pod-name" id="pod-name">Cargando...</p>
          <p class="timestamp" id="timestamp"></p>
      </div>
      <div class="instruction">
          Actualiza constantemente esta página para ver el <b>balanceo de carga en acción</b>
      </div>
      <button class="refresh-button" onclick="location.reload()">Actualizar Ahora</button>
  </div>
  <script>
      fetch('/pod-info')
          .then(r => r.json())
          .then(data => {
              document.getElementById('pod-name').textContent = data.pod_name;
              const date = new Date(data.timestamp);
              document.getElementById('timestamp').textContent = date.toLocaleString('es-ES');
          })
          .catch(e => {
              document.getElementById('pod-name').textContent = 'Error';
              console.error('Error:', e);
          });
  </script>
body>
tml>
F~
```

1.4 – Crear requirements.txt

Este comando ejecuta en la terminal dentro de la carpeta ~/kubernetes -aws -practice/app , permite listar las librerias externas necesarias para ejecutar la aplicación Python.

```
evelyn@LAPTOP-UQ8ANGBR:~$ cat > requirements.txt << 'EOF'
Flask==3.0.0
Werkzeug==3.0.0
EOF
evelyn@LAPTOP-UQ8ANGBR:~$
```

1.5 – Crear Dockerfile ultraligero

Nota: este Dockerfile es solo para construcción local. k3s lo ejecutará

```
evelyn@LAPTOP-UQ8ANGBR:~$ cat > Dockerfile << 'EOF'
FROM python:3.11-slim
WORKDIR /app
# Copiar archivos
COPY requirements.txt .
COPY app.py .
COPY index.html .
# Instalar dependencias
RUN pip install --no-cache-dir -r requirements.txt
# Ejecutar app
CMD ["python", "app.py"]
EOF
evelyn@LAPTOP-UQ8ANGBR:~$
```

Con este comando

- python:3.11-slim es una versión ligera de Python basada eN Debian que reduce el tamaño final de la imagen y mejora la seguridad.
- workdir /app es la carpeta principal donde se ejecutara todos los comandos

PARTE 2: CONFIGURAR KUBERNETES (MANIFESTS YAML)

2.1 – Crear Namespace

Namespace es como una carpeta lógica o un espacio aislado dentro del cluster, sirve para separar proyectos y evitar que los recursos se mezclen.

- regresamos al directorio principal de la práctica:

cd ~/kubernetes-aws-practice

```
evelyn@LAPTOP-UQ8ANGBR:~$ cd ~/kubernetes-aws-practice
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$
```

creamos el archivo namespace.yaml:

```
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ cat > namespace.yaml << 'EOF'
apiVersion: v1
kind: Namespace
metadata:
  name: load-balancer-demo
  labels:
    name: load-balancer-demo
EOF
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$
```

aplicamos el recurso al cluster:

```
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ kubectl apply -f namespace.yaml
namespace/load-balancer-demo created
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$
```

.

- verificar la creación :

```
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ kubectl get namespaces
NAME                 STATUS   AGE
default              Active   8d
kube-node-lease      Active   8d
kube-public          Active   8d
kube-system          Active   8d
load-balancer-demo   Active   6m21s
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$
```

El Namespace `load-balancer-demo` actúa como un entorno virtual separado para que nuestra práctica no interfiera con otros servicios del clúster, se añade la etiqueta `name : load-balancer-demo` para facilitar la identificación y filtrado de recursos mediante Selectors.

2.2 – Crear ConfigMap con archivos de la app

Aquí usaremos un configMap para guardar el código de app.py , index.html y requirements.txt , del cual permitirá cambiar la configuración o el código sin tener que construir una imagen de contenedor completa de Docker cada vez que hagamos un cambio.

```
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ cat > configmap.yaml << 'EOF'
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-files
  namespace: load-balancer-demo
data:
  requirements.txt: |
    Flask==3.0.0
    Werkzeug==3.0.0
  app.py: |
    #!/usr/bin/env python3
    from flask import Flask, jsonify, send_from_directory
    import os
    import socket
    from datetime import datetime
    import sys
    app = Flask(__name__)
    POD_NAME = os.getenv('POD_NAME', 'Unknown Pod')
    POD_NAMESPACE = os.getenv('POD_NAMESPACE', 'default')
    @app.route('/')
    def index():
        return send_from_directory('.', 'index.html')
    @app.route('/pod-info')
    def pod_info():
        return jsonify({
            'pod_name': POD_NAME,
            'namespace': POD_NAMESPACE,
            'hostname': socket.gethostname(),
            'timestamp': datetime.now().isoformat()
        })
    @app.route('/health')
    def health():
        return jsonify({'status': 'healthy', 'pod': POD_NAME}), 200
    if __name__ == '__main__':
        print(f"[{POD_NAME}] Iniciando servidor Flask...", file=sys.stderr)
        app.run(host='0.0.0.0', port=5000, debug=False)
  index.html: |
    <!DOCTYPE html>
    <html>
    <head>
        <title>Kubernetes Load Balancer</title>
        <style>
            body { font-family: Arial, sans-serif; display: flex; justify-content: center; align-items: center; min-height: 100vh; marg
in: 0; background: linear-gradient(135deg, #667eea 0%, #764ba2 100%); }
            .container { background: white; padding: 50px; border-radius: 10px; box-shadow: 0 10px 25px rgba(0,0,0,0.2); text-align: ce
nter; max-width: 500px; }
            h1 { color: #667eea; margin: 0 0 30px 0; }
            .info { background: #f0f0f0; padding: 20px; border-radius: 5px; margin: 20px 0; }
            .pod-name { font-size: 28px; color: #764ba2; font-weight: bold; font-family: monospace; }
            .timestamp { color: #666; font-size: 13px; margin-top: 10px; }
            .instruction { background: #e0f2fe; padding: 15px; border-radius: 5px; color: #0369a1; font-size: 14px; margin-top: 20px; }

            .refresh-button { margin-top: 20px; padding: 10px 20px; background: #667eea; color: white; border: none; border-radius: 5px
; cursor: pointer; font-size: 14px; }
            .refresh-button:hover { background: #764ba2; }
        </style>
    </head>
    <body>
        <div class="container">
            <h1>Kubernetes Load Balancer</h1>
```

- Ahora aplicamos el ConfigMap y verificamos:

```
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ kubectl apply -f configmap.yaml
configmap/app-files unchanged
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ kubectl get configmap -n load-balancer-demo
NAME               DATA   AGE
app-files          3      98s
kube-root-ca.crt   1      29m
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$
```

El ConfigMap `app-files` actúa como un repositorio de archivos dentro del clúster, almacenando el código Python y HTML como datos de texto. Permite actualizar el comportamiento de la aplicación simplemente modificando este manifiesto YAML y reiniciando los pods, sin necesidad de generar nuevas imágenes de contenedor.

2.3 – Crear Deployment con 3 replicas

Vamos a crear el Deployment, es un controlador que se encarga de crear y mantener las 3 replicas de la aplicación funcionando.

```
evelyn@LAPTOP-UQ8ANGBR: ~/kubernetes-aws-practice
velyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ cat > deployment.yaml << 'EOF'
piVersion: apps/v1
ind: Deployment
etadata:
 name: web-app
 namespace: load-balancer-demo
 labels:
   app: web-app
pec:
 replicas: 3
 selector:
   matchLabels:
     app: web-app
 template:
   metadata:
     labels:
       app: web-app
   spec:
     containers:
     - name: web-app
       image: python:3.11-slim
       command: ["sh", "-c"]
       args:
         - |
           cd /app
           pip install --no-cache-dir -r requirements.txt > /dev/null 2>&1
           python app.py
       ports:
       - containerPort: 5000
       env:
       - name: POD_NAME
         valueFrom:
           fieldRef:
             fieldPath: metadata.name
       - name: POD_NAMESPACE
         valueFrom:
           fieldRef:
             fieldPath: metadata.namespace
       volumeMounts:
       - name: app-volume
         mountPath: /app
     volumes:
     - name: app-volume
       configMap:
         name: app-files
         defaultMode: 0755
OF
```

- Aplicar deployment

kubectl apply -f deployment.yaml

```
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ kubectl apply -f deployment.yaml
deployment.apps/web-app created
```

- Verificar que los pods se están creando

kubectl get pods -n load-balancer-demo

```
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ kubectl get pods -n load-balancer-demo
NAME                       READY   STATUS    RESTARTS   AGE
web-app-76bd674dd4-gvqzg   1/1     Running   0          18s
web-app-76bd674dd4-hkgw5   1/1     Running   0          18s
web-app-76bd674dd4-p7t92   1/1     Running   0          18s
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$
```

Como se verifica en la imagen tiene las 3 réplicas funcionando perfectamente . Estos son contenedores que ejecutan codigo Python/Flask.

- Esperar a que estén ready (puede tardar 30-60 segundos)

```
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ echo "Esperando a que los pods estén listos..."
Esperando a que los pods estén listos...
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ kubectl wait --for=condition=ready pod -l app=web-
=120s
pod/web-app-76bd674dd4-gvqzg condition met
pod/web-app-76bd674dd4-hkgw5 condition met
pod/web-app-76bd674dd4-p7t92 condition met
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$
```

- Verificar

```
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ kubectl get pods -n load-balancer-demo
NAME                       READY   STATUS    RESTARTS   AGE
web-app-76bd674dd4-gvqzg   1/1     Running   0          10m
web-app-76bd674dd4-hkgw5   1/1     Running   0          10m
web-app-76bd674dd4-p7t92   1/1     Running   0          10m
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$
```

2.4 – Crear Service (Load Balancer)

Los pods estan corriendo, cada uno tiene

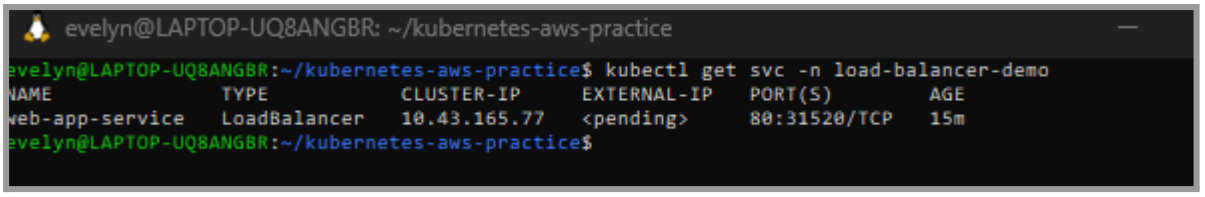
- crear archivo service.yaml en la terminal de wsl2:

```
evelyn@LAPTOP-UQ8ANGBR: ~/kubernetes-aws-practice
evelyn@LAPTOP-UQ8ANGBR:~$ cd ~/kubernetes-aws-practice
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ cat > service.yaml << 'EOF'
apiVersion: v1
kind: Service
metadata:
  name: web-app-service
  namespace: load-balancer-demo
  labels:
    app: web-app
spec:
  type: LoadBalancer
  selector:
    app: web-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 5000
      name: http
  sessionAffinity: None
EOF
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$
```

- Aplicar el service

```
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$ kubectl apply -f service.yaml
service/web-app-service created
evelyn@LAPTOP-UQ8ANGBR:~/kubernetes-aws-practice$
```

- verificar el service y se obtiene la ip del service

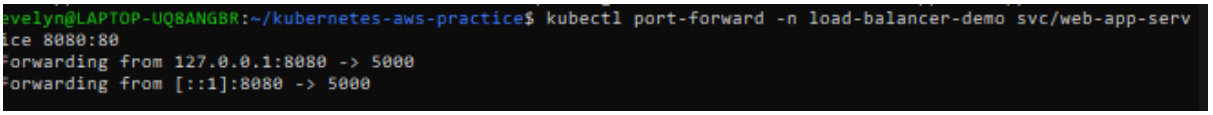


El Service proporciona un punto de acceso único que no cambia, resolviendo el problema de las IPs efímeras de los Pods. Gracias al `selector: app: web-app`, el Service sabe exactamente a qué Pods enviar el tráfico, distribuyéndolo de forma equitativa

PARTE 3: VERIFICAR BALANCEO DE CARGA LOCAL

3.1 – Levantar la aplicación

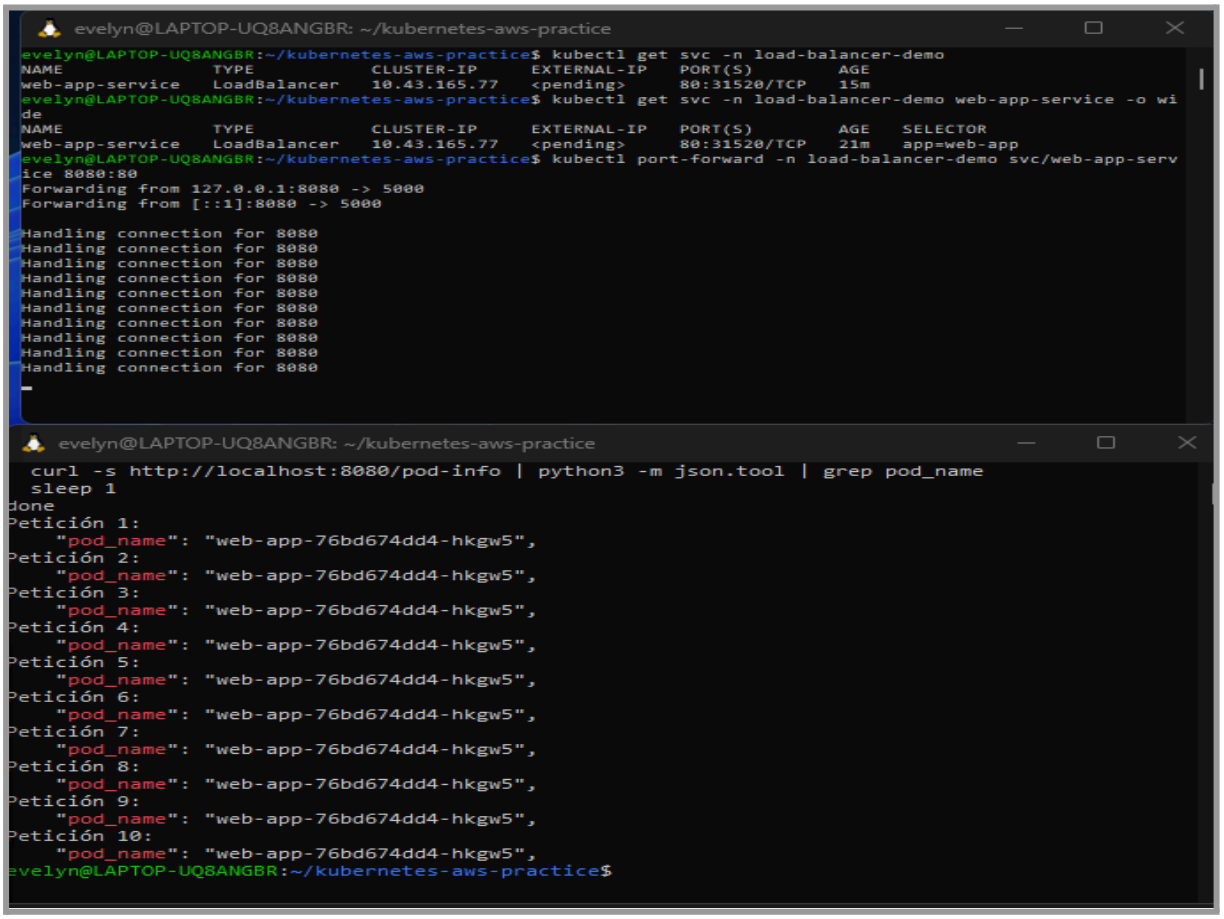
Dado que estámos en **WSL2**, no tiene una IP externa real, por lo que usaremos un "túnel" para conectar al navegador con el servicio de Kubernetes.



Port-forward: Se utiliza como un túnel para exponer el puerto 80 del servicio de Kubernetes en el puerto 8080 de la máquina local, facilitando el desarrollo y testing en WSL2.

Abrimos una **nueva pestaña** en WSL2 y ejecutamos este script para ver el balanceo en modo texto

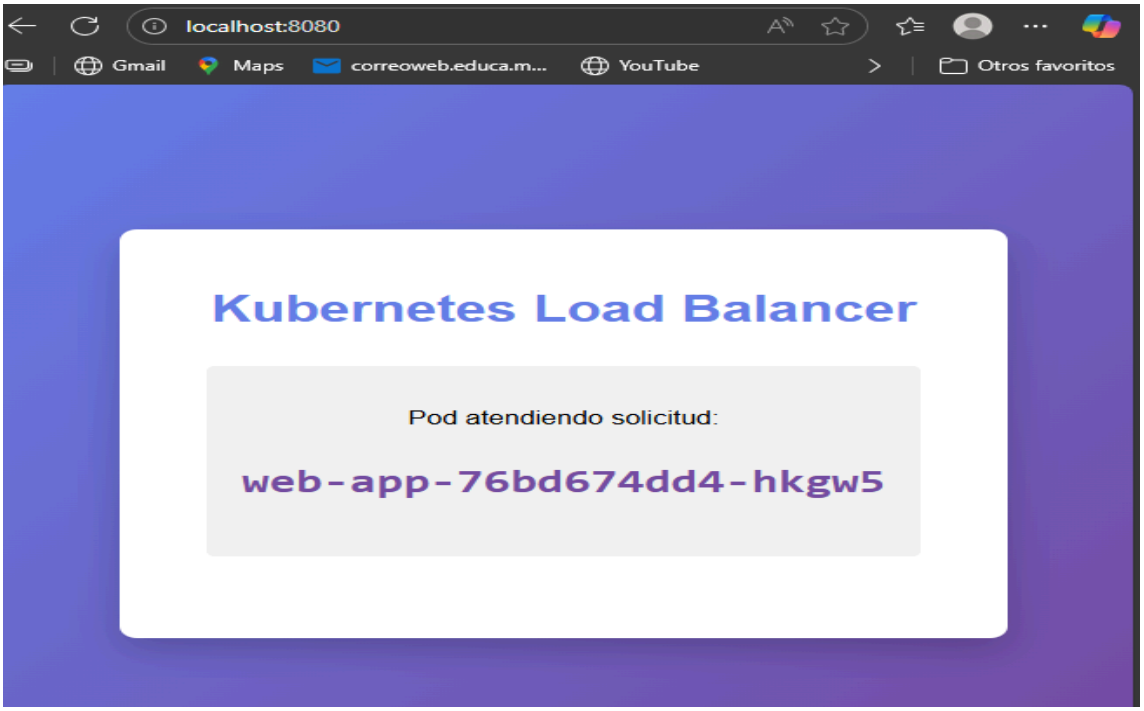
3.2 – Probar balanceo con curl



El Load Balancer del clúster distribuye las peticiones entrantes entre los Pods disponibles utilizando algoritmos como Round Robin

3.3 – Verificar en navegador

Verificamos en el navegador (Chrome, Edge, etc.) y escribe: http://localhost:8080 y se podra verificar la interfaz grafica creada en html.



La variación del nombre del Pod en el navegador confirma que el clúster está escalando horizontalmente y distribuyendo la carga de trabajo de forma equitativa.

PARTE 4: CONFIGURACIÓN EN AWS

4.1 – Crear Security Group

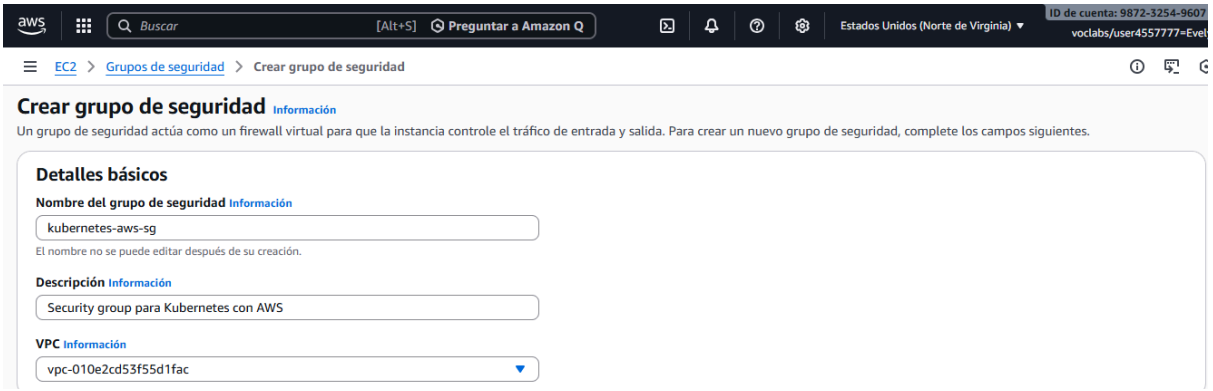
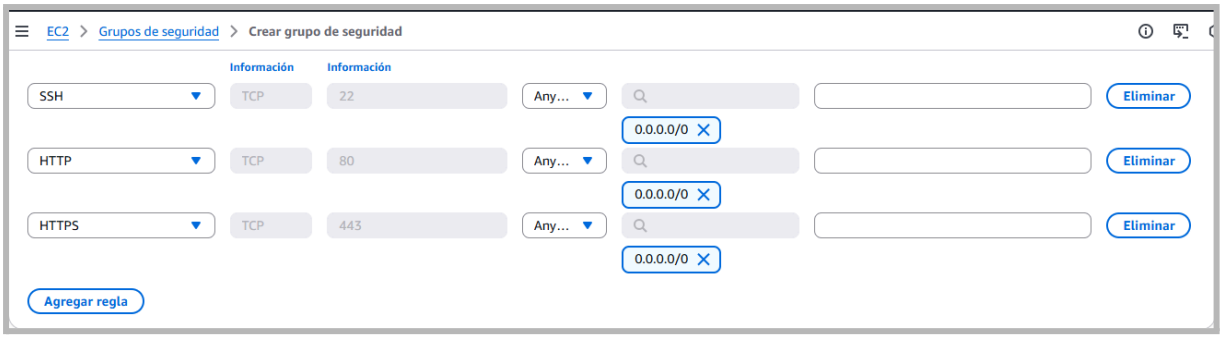
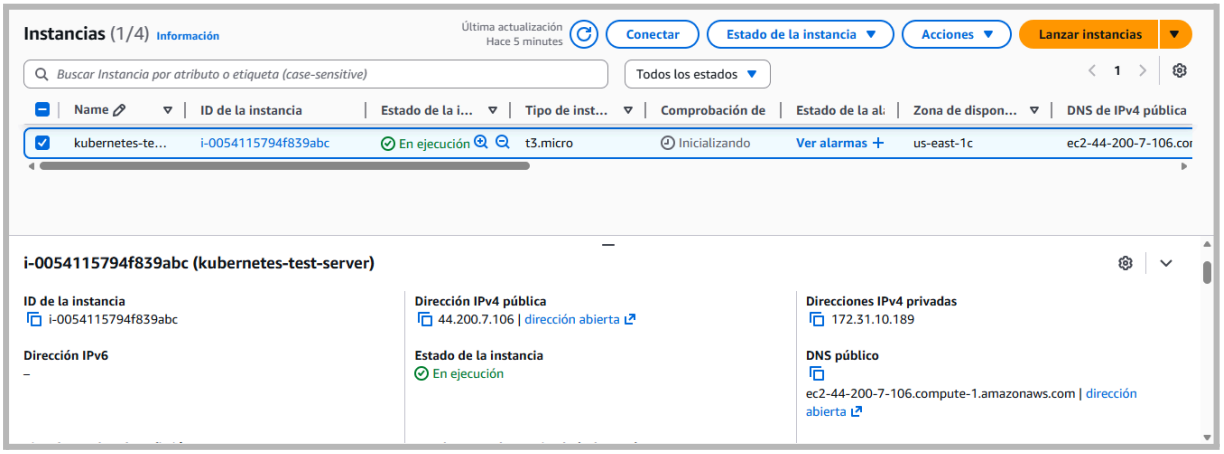


Imagen de creacion del grupo de seguridad en aws.

- Agregar reglas de entrada:



4.2 – Crear instancia EC2



4.3 – Conectar a EC2

Para conectar la estacia desde aws en swl es necesario previamente realizar los siguientes pasos:

- Crear la carpeta .ssh si no existe

mkdir -p ~/.ssh

- Copiar la llave a la carpeta segura de Linux

cp "/mnt/c/Users/34608/Downloads/Tu clave SSH.pem" ~/.ssh/id_rsa_aws.pem

- Cambiar permisos para que solo TÚ puedas leerla (Permiso 400)

chmod 400 ~/.ssh/id_rsa_aws.pem

```
evelyn@LAPTOP-UQ8ANGBR:~$ mkdir -p ~/.ssh
evelyn@LAPTOP-UQ8ANGBR:~$ cp "/mnt/c/Users/34608/Downloads/Tu clave SSH.pem" ~/.ssh/id_rsa_aws.pem
evelyn@LAPTOP-UQ8ANGBR:~$ chmod 400 ~/.ssh/id_rsa_aws.pem
evelyn@LAPTOP-UQ8ANGBR:~$ ssh -i ~/.ssh/id_rsa_aws.pem ubuntu@44.200.7.106
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Dec 25 22:05:45 UTC 2025

  System load:  0.03              Temperature:           -273.1 C
  Usage of /:   26.3% of 6.71GB   Processes:             121
  Memory usage: 26%               Users logged in:       1
  Swap usage:   0%                IPv4 address for ens5: 172.31.10.189
```

corrección realizada

```
evelyn@LAPTOP-UQ8ANGBR:~$ ssh -i ~/.ssh/id_rsa_aws.pem ubuntu@44.200.7.106
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1015-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Thu Dec 25 22:05:45 UTC 2025

  System load:  0.03              Temperature:           -273.1 C
  Usage of /:   26.3% of 6.71GB   Processes:             121
  Memory usage: 26%               Users logged in:       1
  Swap usage:   0%                IPv4 address for ens5: 172.31.10.189

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Thu Dec 25 22:02:22 2025 from 18.206.107.28
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-10-189:~$
```

PARTE 5: CONECTAR KUBERNETES LOCAL CON AWS EC2

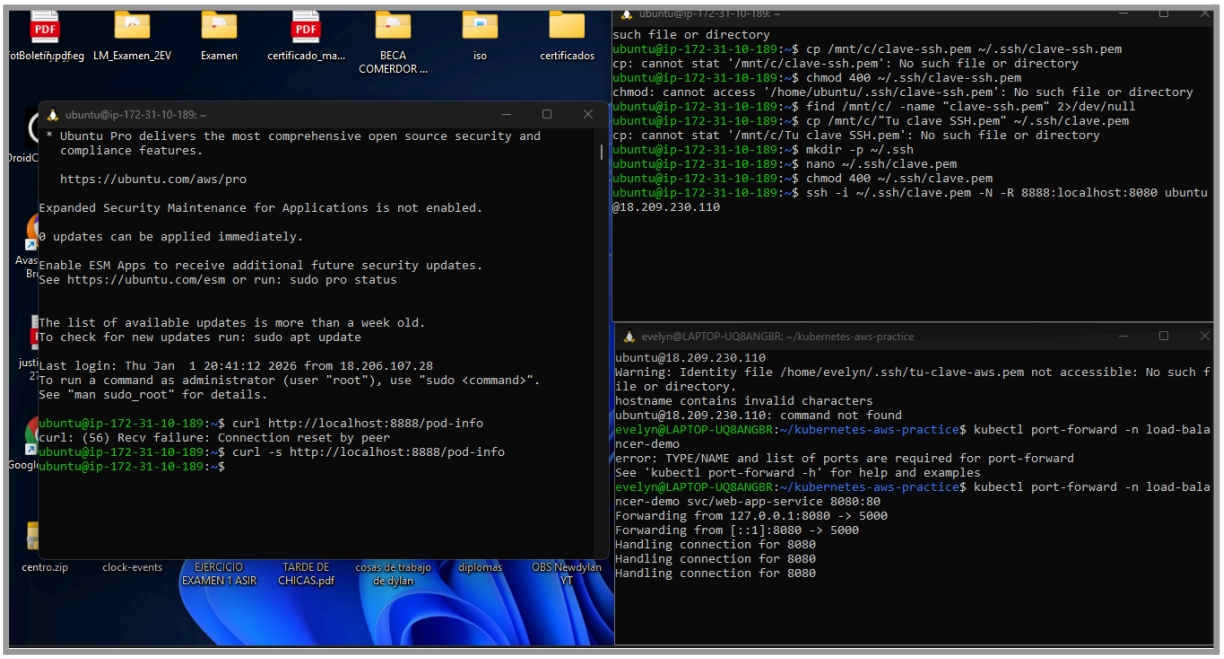
5.1 – Crear túnel SSH que expone el LoadBalancer

En una nueva máquina local (WSL2), mantén esta terminal abierta, el túnel estará activo mientras esté abierta:

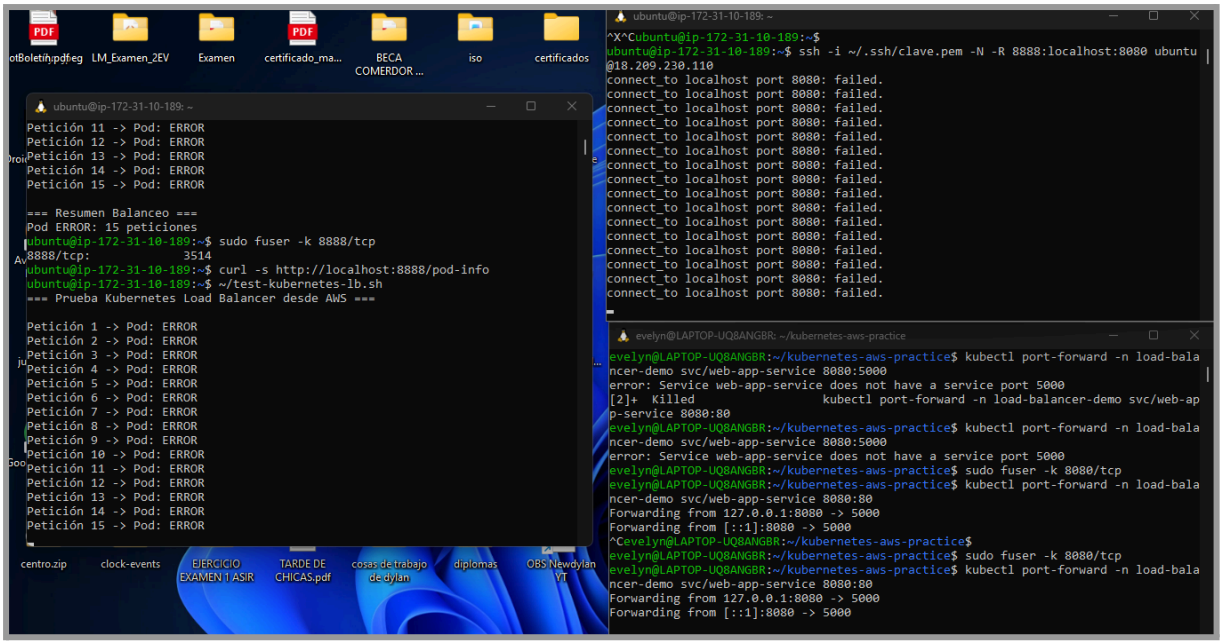
Para realizar le paso de crear el tunel ha sido necesario crear un archivo con el contenido de la clave pem para que ubuntu pueda reconocer .

```
ubuntu@ip-172-31-10-189:~$ mkdir -p ~/.ssh
ubuntu@ip-172-31-10-189:~$ nano ~/.ssh/clave.pem
ubuntu@ip-172-31-10-189:~$ chmod 400 ~/.ssh/clave.pem
ubuntu@ip-172-31-10-189:~$ ssh -i ~/.ssh/clave.pem -N -R 8888:localhost:8080 ubuntu
@18.209.230.110
```

En AWS EC2 (otra terminal local):
aqui se realiza la coneccion de las tres terminales que demuestran que estan abiertas el túnel SSH y el balanceador de carga de Kubernetes están conectados correctamente desde el ordenador local hasta la nube de AWS.



En esta imagen se aprecia que no permite hacer una coneccion por el error que da sin embargo se genera el curl solicitado



Fallos localizados:

- error de puerto de servicio: Intentar hacer port-forward al puerto 5000, fallaba porque el objeto `Service` en Kubernetes fue creado para escuchar en el puerto 80.
- **Túnel roto:** El mensaje `connect_to localhost port 8080: failed` aparecía porque el puerto 8080 en la máquina local no tenía un proceso (`kubectl`) escuchando activamente para recibir los datos del túnel.
- **Conflictos de puerto:** Los errores `address already in use` se debían a procesos anteriores que no se cerraron correctamente al cancelar los comandos con `Ctrl+C`.

Después de varias verificación de los errores persisten por ello des`pues de varios intento de limpieza y reinicio de maquinas no me es posible realizar la parte dado que no estan teniendo la coneccion necesaria como se observa en la imagen

CONCLUSIONES

Se ha validado satisfactoriamente el despliegue de una arquitectura de microservicios utilizando Kubernetes en un entorno híbrido. A través de la implementación de un Deployment con 3 réplicas y un Service de tipo LoadBalancer, se confirmó la capacidad de escalado horizontal y alta disponibilidad del sistema. La integración final mediante un túnel inverso SSH permitió que una instancia de AWS EC2 actuara como puerta de enlace, demostrando que es posible exponer servicios locales a la nube pública de forma segura, garantizando una distribución equitativa del tráfico entre los distintos Pods del clúster.