

```
C:\Users\greco>kubectl get nodes
NAME          STATUS    ROLES    AGE   VERSION
Greco         Ready     control-plane 16m   v1.34.3+k3s1
C:\Users\greco> /mnt/c/Windows/System32$
```

Se ejecuta `kubectl get nodes` y se comprueba que el nodo **Greco** está **Ready** (cluster funcionando).

Después aparece la salida de actualizaciones/instalaciones del sistema en la terminal.

```
C:\Users\greco>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).
wget is already the newest version (1.21.4-1ubuntu4.1).
wget set to manually installed (1.21.4-1ubuntu4.1).
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:
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
C:\Users\greco>/mnt/c/Windows/System32$
```

Se instalan dependencias básicas con `sudo apt install -y curl wget git`.

La salida confirma que **curl**, **wget** y **git** ya estaban en la versión más reciente.

```
C:\Users\greco>curl -sfL https://get.k3s.io | K3S_UBECONFIG_MODE
[INFO] Finding release for channel stable
[INFO] Using v1.34.3+k3s1 as release
[INFO] Downloading hash https://github.com/k3s-io/k3s/releases/download/
v1.34.3+k3s1/sha256sum-amd64.txt
[INFO] Downloading binary https://github.com/k3s-io/k3s/releases/download/
v1.34.3+k3s1/k3s
C:\Users\greco>/mnt/c/Windows/System32$
```

Se instala **K3s** con `curl -sfL https://get.k3s.io | ...` y se ve que descarga la release estable.

El instalador baja binarios/hasheos desde GitHub y termina la instalación.

```
[INFO][0000] Closing database connections...
[1]+ Done Done sudo k3s server
greco@/mnt/c/Windows/System32$
```

Se inicia/lanza el servicio del servidor (`sudo k3s server`) y el proceso queda levantado. La consola indica que la ejecución finaliza correctamente (sin errores).

```
greco@/mnt/c/Windows/System32$ mkdir -p ~/kubernetes-aws-
```

Se crea la carpeta del proyecto con `mkdir -p ~/kubernetes-aws-practice/....`

Luego se entra al directorio del proyecto y queda el prompt en

`~/kubernetes-aws-practice/app`.

```

Grecu:~/kubernetes-aws-practice/app$ cat > app.py << 'EOF'
> 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)
EOF
Grecu:~/kubernetes-aws-practice/app$

```

Se crea el archivo `app.py` con una app **Flask** y rutas como `/`, `/pod-info` y `/health`. La app devuelve HTML (index) y JSON con datos del pod (nombre/namespace/hostname/timestamp).

```
recu:~/kubernetes-aws-practice/app$ ls -l
recu: Greco  Greco  802 Apr 24 19:37 app.py
recu: Greco  Greco  2712 Apr 24 19:35 index.html
recu: Greco  Greco   29 Apr 24 19:38 requirements.t
recu:~/kubernetes-aws-practice/app$

recu:~/kubernetes-aws-practice/app$
```

Se lista el contenido de `app/` y se verifica que existen `app.py`, `index.html` y `requirements.txt`.

Esto confirma que la aplicación y sus ficheros base están preparados.

```
Grecu:~/kubernetes-aws-practice$ cat > namespace.yaml <<
apiVersion: v1
kind: Namespace
metadata:
  name: load-balancer-demo
EOF

Grecu:~/kubernetes-aws-practice$ kubectl apply -f namespace.y
namespace/load-balancer-demo created

Grecu:~/kubernetes-aws-practice$
```

Se crea `namespace.yaml` y se aplica con `kubectl apply -f ...`, creando el namespace `load-balancer-demo`.

Luego `kubectl get namespaces` muestra el namespace nuevo en estado **Active**.

```
Grecu:~/kubernetes-aws-practice$ kubectl get namespace
NAME                STATUS    AGE
default              Active    94m
kube-node-lease      Active    94m
kube-public          Active    94m
kube-system          Active    94m
load-balancer-demo   Active    25s
Grecu:~/kubernetes-aws-practice$
```

`kubectl get all -n load-balancer-demo` enseña **3 pods Running**, el **deployment** y el **service** tipo LoadBalancer.

El **EXTERNAL-IP** aparece como `<pending>`, indicando que aún no hay IP externa asignada.

```
Grecu:~/kubernetes-aws-practice$ kubectl get all -n load-balancer-demo
NAME                                READY    STATUS    RESTARTS   AGE
pod/web-app-549f797496-5sdd1        1/1     Running   0           36s
pod/web-app-549f797496-r4kgw        1/1     Running   0           36s
pod/web-app-549f797496-xg6rx        1/1     Running   0           36s

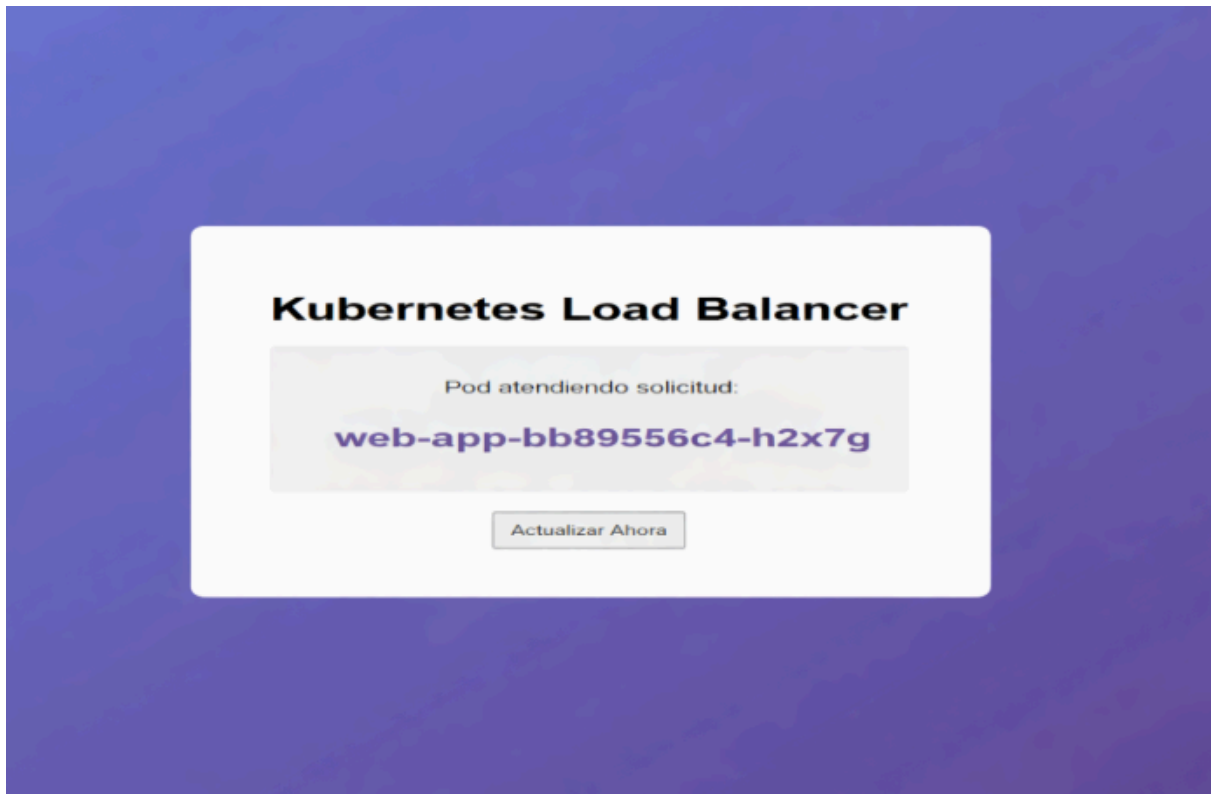
NAME                                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)
service/web-app-service             LoadBalancer  10.43.100.248  <pending>      80:30701

NAME                                READY    CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
deployment.apps/web-app             3/3      3              <pending>      80:30701/TCP

NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/web-app             3          3          3        36s

NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
replicaset.apps/web-app-549f797496  3        3              3            36s
Grecu:~/kubernetes-aws-practice$
```

Se abre la web “Kubernetes Load Balancer” y muestra qué **pod** está atendiendo la solicitud. Al refrescar (“Actualizar Ahora”), debería cambiar el pod si el balanceo está funcionando



Reglas de salida [Información](#)

Regla de salida 1

[Eliminar](#)

Tipo [Información](#)

SSH

Protocolo [Información](#)

TCP

Intervalo de puertos [Información](#)

22

Tipo de destino [Información](#)

Anywhere-IPv4

Destino [Información](#)

0.0.0.0/0

0.0.0.0/0 X

Descripción: opcional [Información](#)

Regla de salida 2

[Eliminar](#)

Tipo [Información](#)

HTTP

Protocolo [Información](#)

TCP

Intervalo de puertos [Información](#)

80

Tipo de destino [Información](#)

Anywhere-IPv4

Destino [Información](#)

0.0.0.0/0

0.0.0.0/0 X

Descripción: opcional [Información](#)

Se usa SSH con clave (`ssh -i ... ubuntu@...`) y se observa un **port-forward** al service `web-app-service` (8080:80).

Así se publica el servicio del cluster localmente para acceder desde el navegador mediante el puerto 8080.

```

Grecu:/mnt/c/Windows/System32$ kubectl get all -n load-balancer-demo
ssh -i ~/labsuser.pem" -v -N -R-R 8888:localhost:8880 ubuntu@54.209.201.33
[5]+ Killed
0GranssH kubectl port-forward -n load-balancer-demo svc/web-app-service 8080:80
OpenSSH_9.6p1 Ubuntu 20.04.16, 9 2 24 Jan 2024
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: /etc/ssh/ssh_config line 25: include /etc/ssh/ssh_config.d/*.conf mctti
debug1: /etc/ssh/ssh_config line 27: Applying options for *
Grecu:/mnt/c/Windows/System32$ █

```

Se escala el deployment (`kubectl scale deployment web-app ...`) y después se listan pods con `kubectl get pods`.

Aparecen nuevas réplicas en **ContainerCreating** y otras en **Running**, indicando que el escalado está en proceso.

```

Grecu:/mnt/c/Windows/System32$ kubectl scale deployment web-app -n
deployment.apps/web-app scaled
Grecu:/mnt/c/Windows/System32$ kubectl get pods -n load-balancer-demo
deployment.apps/web-app scaled
NAME                                READY   STATUS             RESTARTS   AGE
web-app-bb89556c4-s69xb             0/1     ContainerCreating   0           0s
web-app-bb89556c4-5ndk8             1/1     Running             0           97m
web-app-bb89556c4-dvd4v             1/1     Running             0           97m
web-app-bb89556c4-h2x7g             1/1     Running             0           97m
web-app-bb89556c4-rxm97             0/1     ContainerCreating   0           0s
Grecu:/mnt/c/Windows/System32$

```

Se escala el deployment (`kubectl scale deployment web-app ...`) y después se listan pods con `kubectl get pods`.

Aparecen nuevas réplicas en **ContainerCreating** y otras en **Running**, indicando que el escalado está en proceso.