**Set Up and Deploy the Flask Backend**

1. Arrancar Docker desktop
2. Descargar e instalar MINICUBE -- <https://minikube.sigs.k8s.io/docs/start/>
   1. Abrir Windows Powershell como administrador
   2. New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force

Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/latest/download/minikube-windows-amd64.exe' -UseBasicParsing

* 1. $oldPath = [Environment]::GetEnvironmentVariable('Path', [EnvironmentVariableTarget]::Machine)

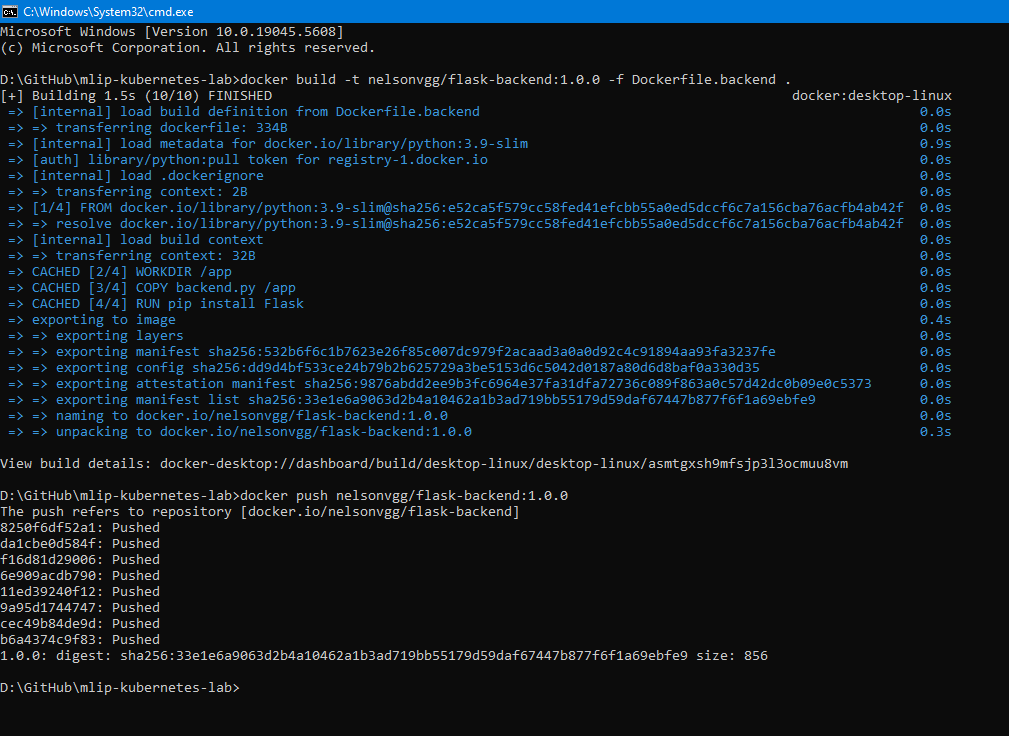
if ($oldPath.Split(';') -inotcontains 'C:\minikube'){

[Environment]::SetEnvironmentVariable('Path', $('{0};C:\minikube' -f $oldPath), [EnvironmentVariableTarget]::Machine)

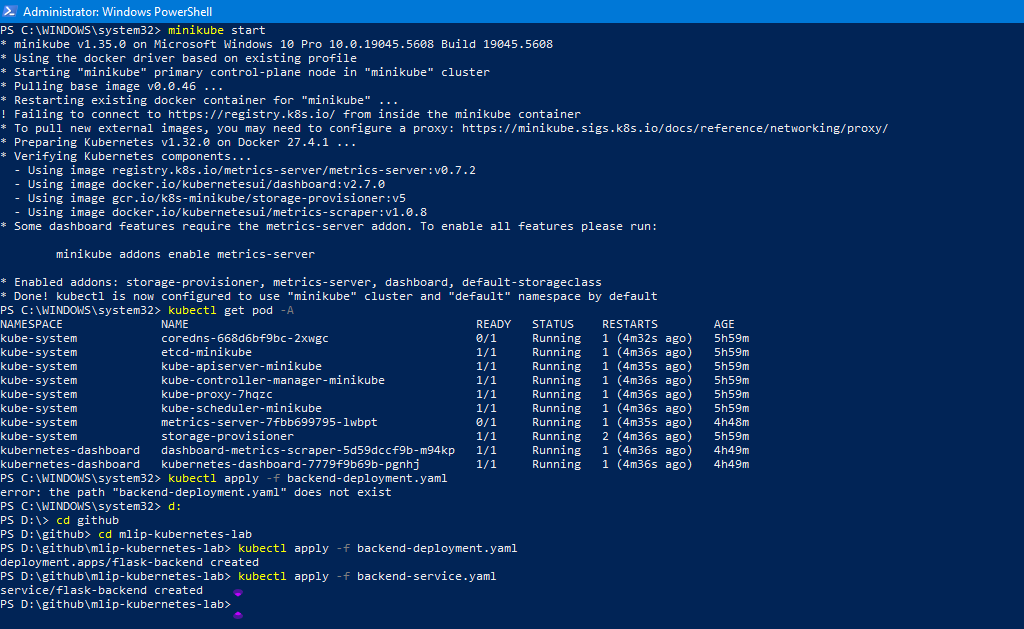
}

1. Arrancar minicube – Windows Powershell 🡪 minikube start
   1. Probar esta funcionando – Windows PowerShell 🡪 kubectl get pod -A

1. Modificar archivo “backend.py” 🡪 defines a simple backend web application using the Flask framework in Python
2. Crear y subir imagen docker usando archivo “Dockerfile.backend” 🡪 script containing a series of instructions to build a container image. Used to create a Docker image for a Python-based Flask application. Correr desde CMD en el directorio donde se encuentra el archivo “Dockerfile.backend”
   1. docker build -t ***nelsonvgg/flask-backend:1.0.0*** -f Dockerfile.backend .
   2. docker push ***nelsonvgg/flask-backend:1.0.0***

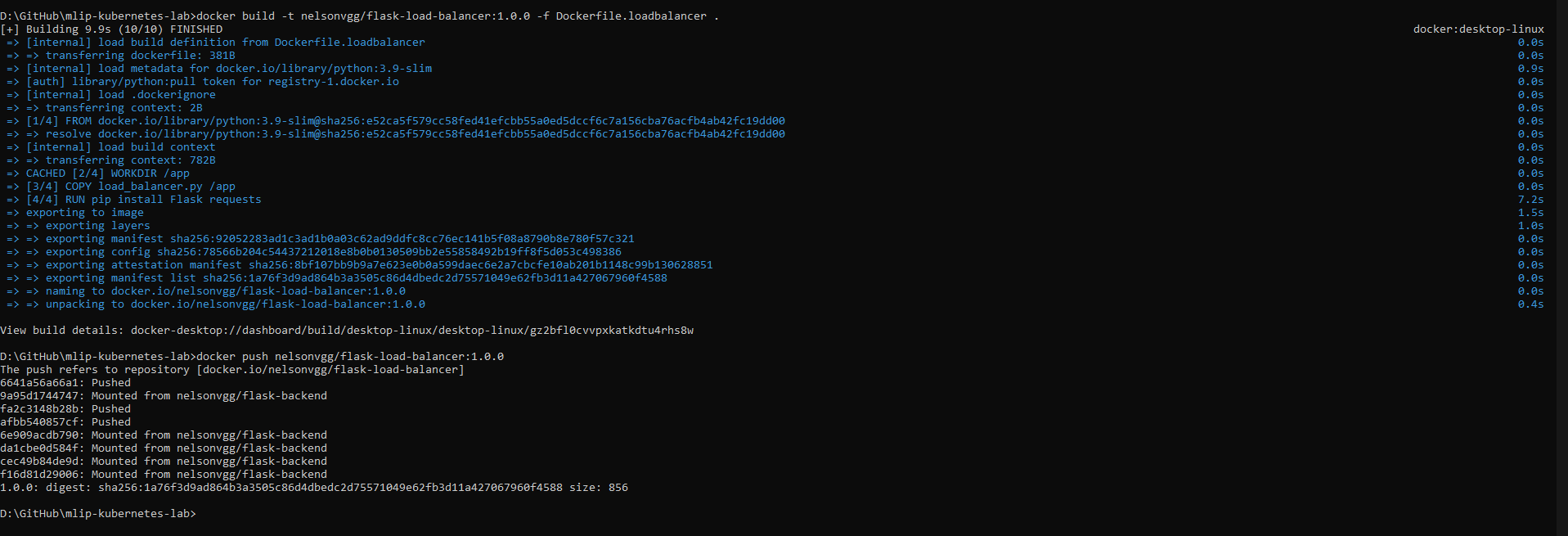


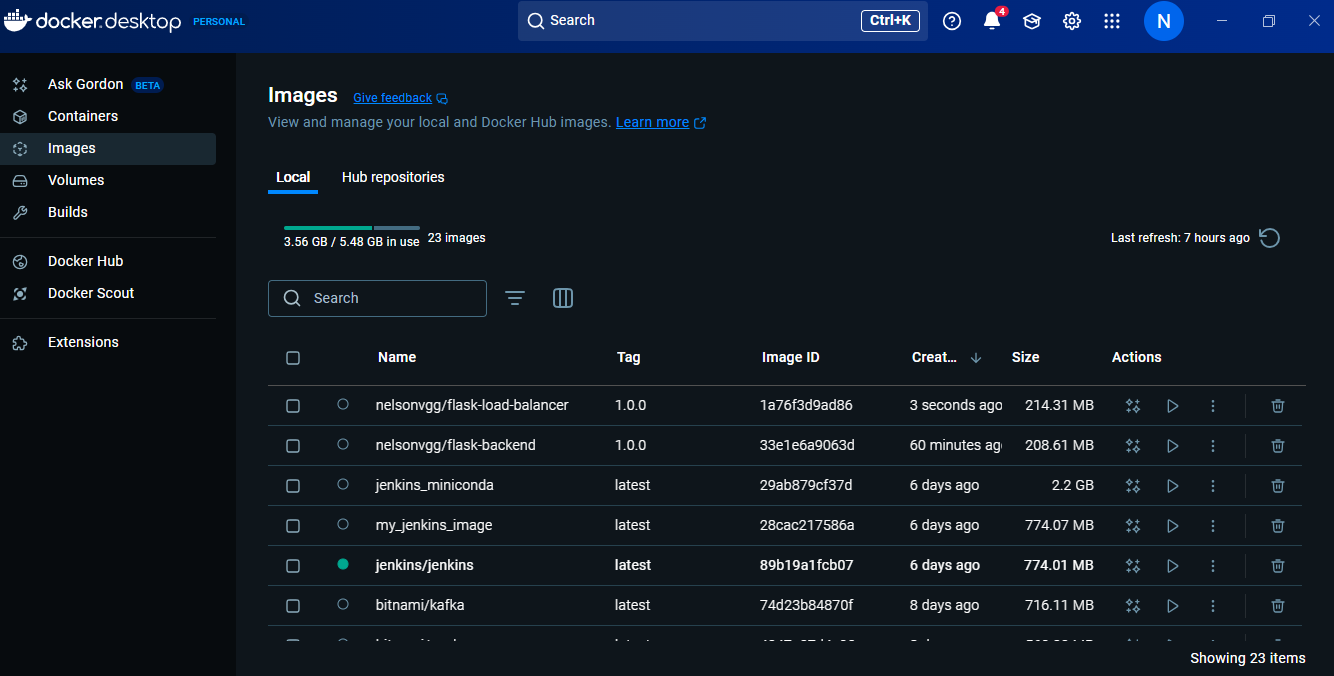
1. Configurar despliegue y servicio
2. Modificar archivo “backend-deployment.yaml” 🡪  Kubernetes Deployment YAML file
3. Modificar archivo “backend-service.yaml” 🡪 Kubernetes Service configuration YAML file
4. Desplegar el backend, con Windows Powershell, ubicados en la carpeta que contiene los archivos “backend-deployment.yaml” y “backend-service.yaml”
   1. kubectl apply -f backend-deployment.yaml
   2. kubectl apply -f backend-service.yaml



**Step 2: Set Up the Python Load Balancer**

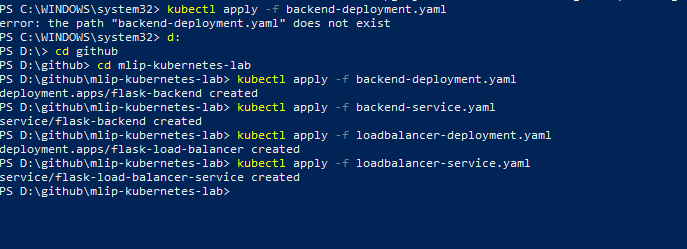
1. Modificar archivo “load\_balancer.py” 🡪 defines a **load balancer** using Flask framework in Python. The load balancer is designed to distribute incoming HTTP requests across multiple backend servers in a **round-robin** fashion. This approach ensures that requests are evenly distributed among the available servers, improving scalability and reliability.
2. Crear y subir imagen docker usando archivo “Dockerfile.loadbalancer” 🡪 script containing a series of instructions to build a container image.  Used to create a Docker image for a Python-based load balancer application. Correr desde CMD en el directorio donde se encuentra el archivo “Dockerfile.loadbalancer”
   1. docker build -t ***nelsonvgg/flask-load-balancer:1.0.0*** -f Dockerfile.loadbalancer .
   2. docker push ***nelsonvgg/flask-load-balancer:1.0.0***





**Step 3: Deploy the Load Balancer on Kubernetes**

1. Configurar despliegue y servicio FLASK-LOAD-BALANCER
2. Modificar archivo “loadbalancer-deployment.yaml” 🡪  Kubernetes Deployment YAML file
3. Modificar archivo “loadbalancer-service.yaml” 🡪 Kubernetes Service configuration YAML file
4. Desplegar el LOAD BALANCER, con Windows Powershell, ubicados en la carpeta que contiene los archivos “loadbalancer-deployment.yaml” y “loadbalancer-service.yaml”
   1. kubectl apply -f loadbalancer-deployment.yaml
   2. kubectl apply -f loadbalancer-service.yaml

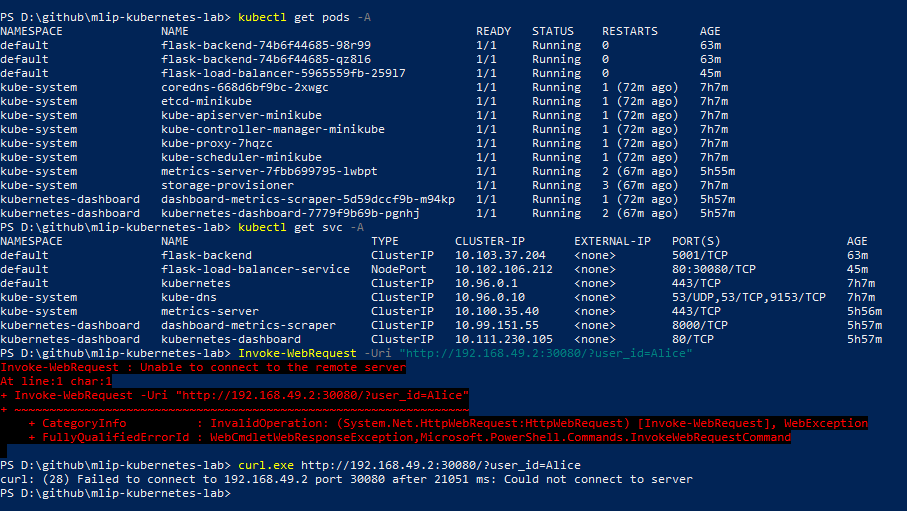


**Step 4: Access the Load Balancer and Test Traffic**

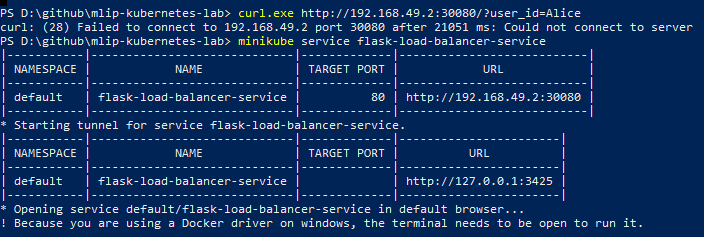
1. Obtener la IP de minicube. Desde Windows PowerShell
   1. minikube ip
      1. En este caso, la respuesta fue: 192.168.49.2
   2. Acceder a LOAD BALANCER usando “curl”, donde se usa la ip de minicube con el siguiente formato 🡪 curl "http://<minikube-ip>:NodePort/?user\_id=Alice"
      1. curl <http://192.168.49.2:30080/?user_id=Alice>
      2. curl <http://192.168.49.2:30080/?user_id=Nelson>
      3. curl <http://192.168.49.2:30080/?user_id=Juan>
      4. curl <http://192.168.49.2:30080/?user_id=Ana>

Para conocer NodePort

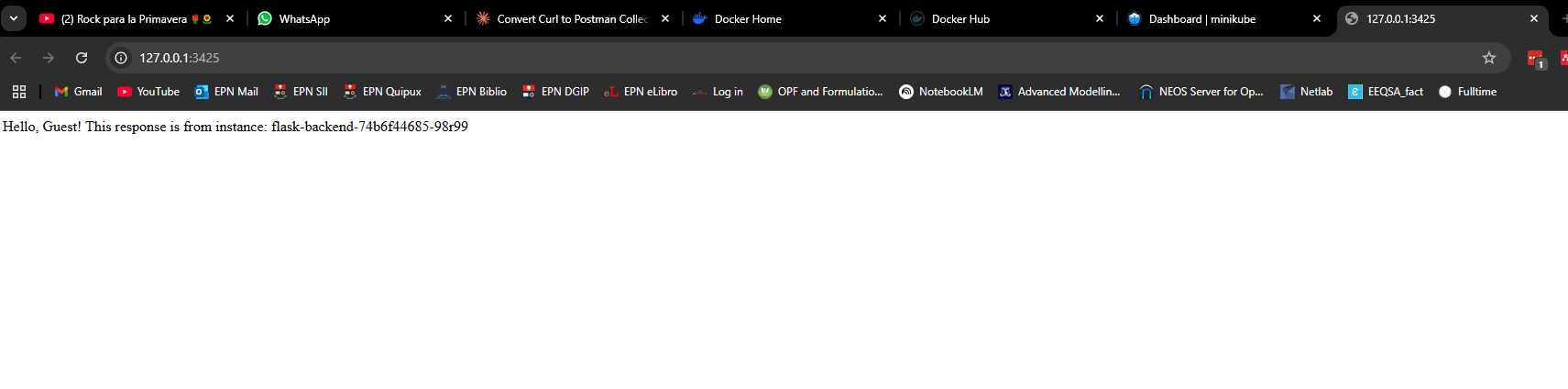
* kubectl get svc -A



1. Usar “minicube service” si NodePort no funciona
   1. Crear un túnel al “load balancer service” usando Windows Powershell
      1. minikube service flask-load-balancer-service



En navegador web



Como resultado se obtiene una URL del tipo [http://127.0.0.1:<some-port>](http://127.0.0.1:%3csome-port%3e) que puede ser usada para hacer pruebas al LOAD BALANCER

* 1. Enviar múltiples solicitudes y ver como cada solicitud es servida por una instancia/máquina diferente
     1. <http://127.0.0.1:3425/?user_id=Alice>
     2. <http://127.0.0.1:3425/?user_id=Nelson>
     3. <http://127.0.0.1:3425/?user_id=Juan>
     4. <http://127.0.0.1:3425/?user_id=Ana>

1. Arrancar el MINIKUBE DASHBOARD para monitorear el estado de los pods, servicios y despliegues
   1. En Windows PowerShell 🡪 minikube dashboard

**Step 5: Load Testing with Apache Benchmark (AB)**

1. Usar WSL (Windows Subsystem for Linux) – Abrir la terminal de WSL. En este caso, después de pruebas se determina que es Linux Alpine
2. Instalar SUDO – En terminal WSL
   1. su -
   2. apk update
   3. apk add sudo
3. Instalar Apache Benchmark (AB) – En terminal WSL
   1. sudo apk update
   2. sudo apk add apache2-utils
4. Hacer pruebas de carga 🡪 En terminal WSL
   1. ab -n 100 -c 10 <http://127.0.0.1:3425/?user_id=Alice>

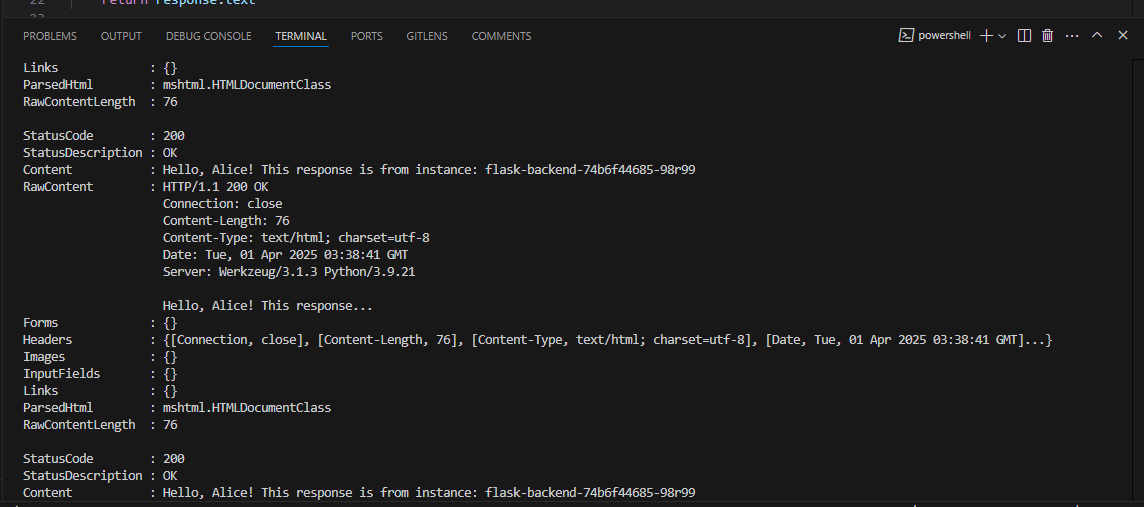
NO FUNCIONA

1. Usar CURL 🡪 Desde Windows Powershell (en Visual Studio Code)
   1. for ($i=0; $i -lt 100; $i++) {

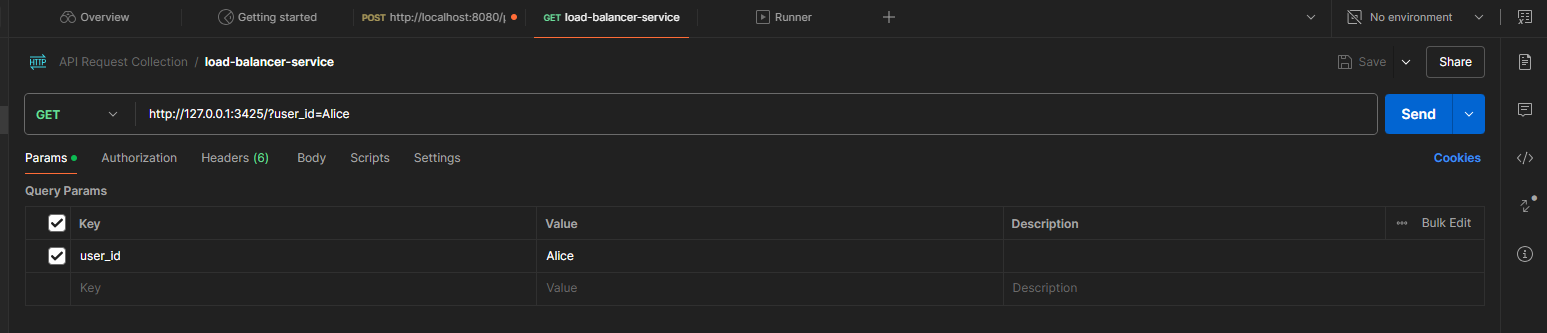
curl "http://127.0.0.1:<some-port>/?user\_id=Alice"

}

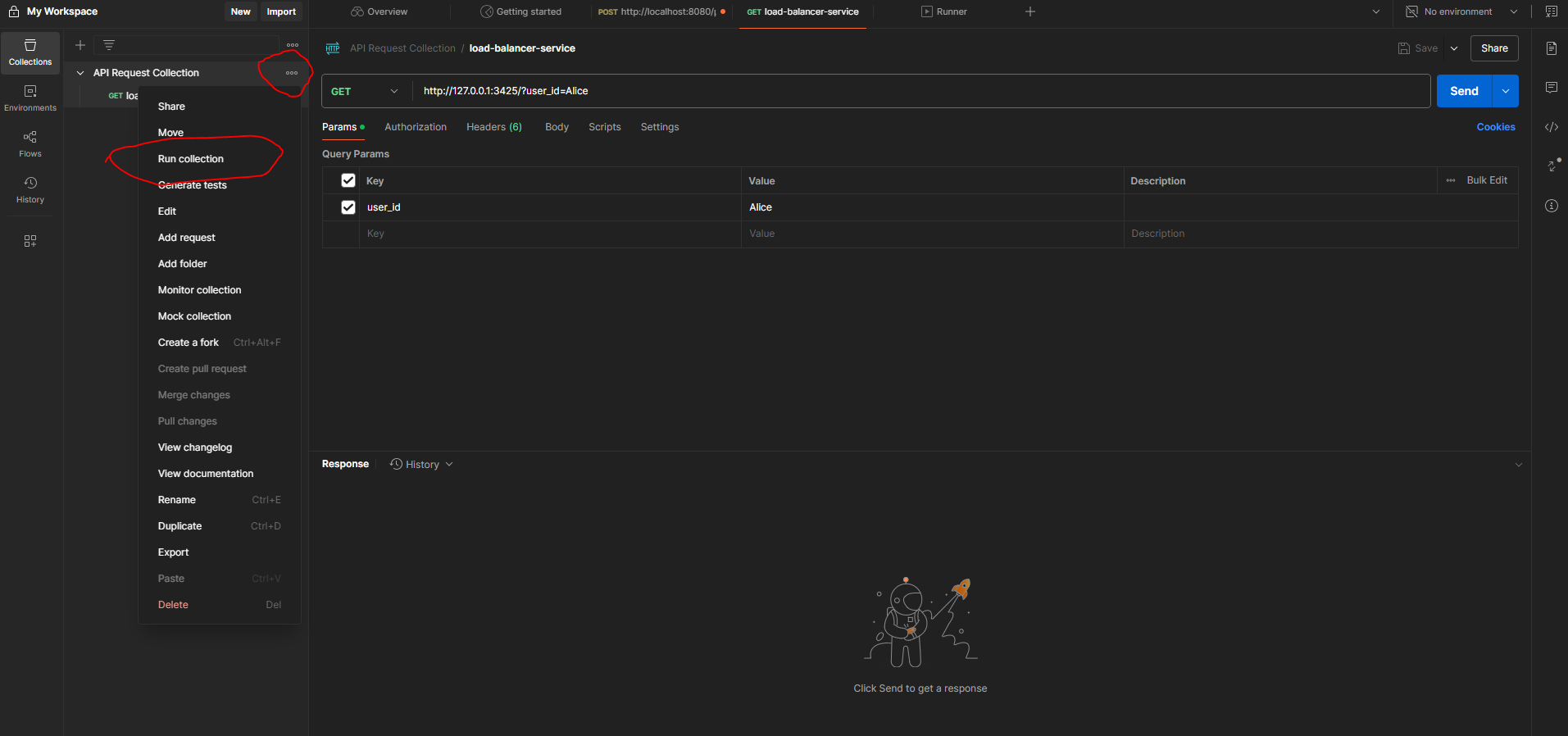
En el caso de usar VSC



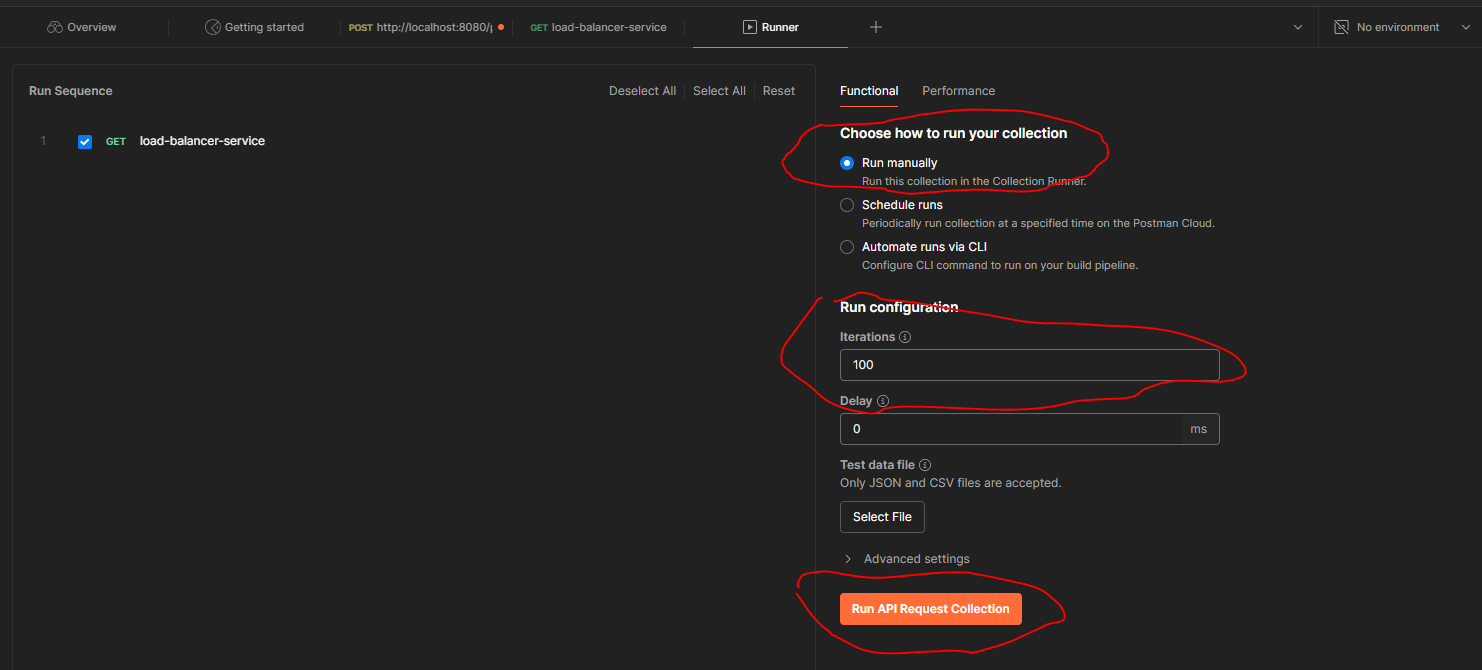
1. Usar Postman 🡪
   1. Crear un nuevo GET con dirección IP:Puerto/?user\_id, posteriormente grabar



* 1. Usar Collections Runner 🡪



* 1. Poner el contador de iteraciones a 100 y correr



1. Verificar logs de flask-backend
   1. kubectl logs -l app=flask-backend -f
2. Verificar logs de load-balancer
   1. kubectl logs -l app=flask-load-balancer -f