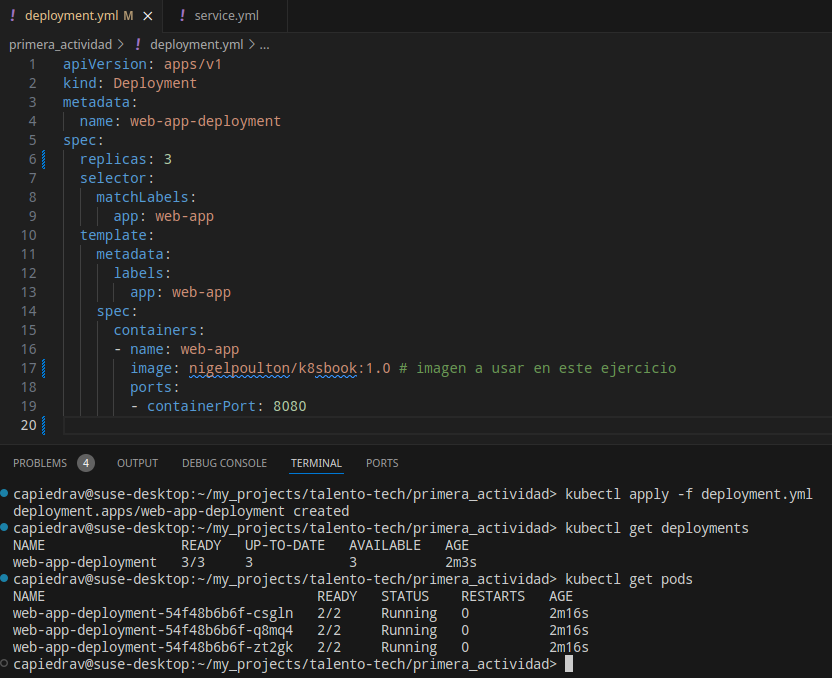
**Informe de Actividades TalentoTech – Arquitectura En la Nube**

**Elaborado por:** Carlos Andrés Piedrahita Velásquez – 8026998 – **Grupo 7**

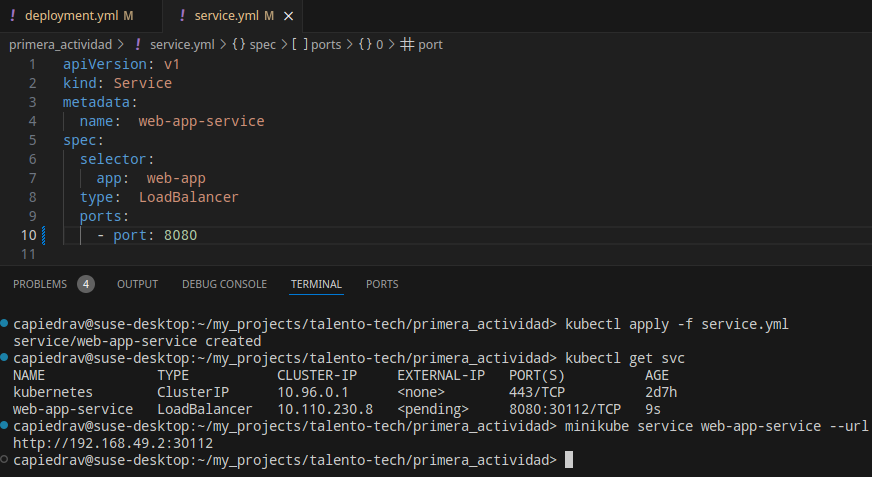
**Repositorio:** https://github.com/capiedrav/actividades-talentotech2

**Primera actividad – Despliegue de aplicación en Kubernetes**

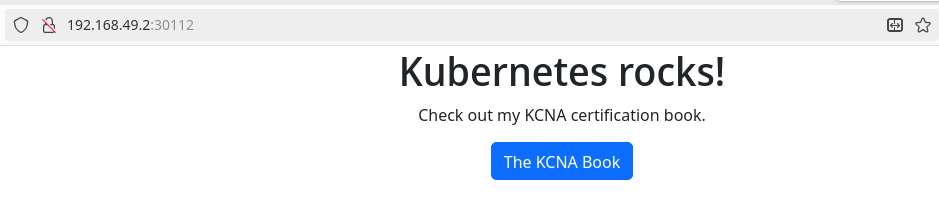
1. Aplicar manifiesto del despliegue y verificar que esté corriendo:

****

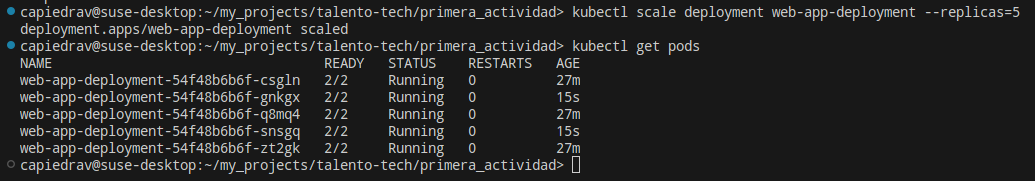
1. Aplicar manifiesto del servicio y verificar que esté corriendo:



1. Probar la aplicación:

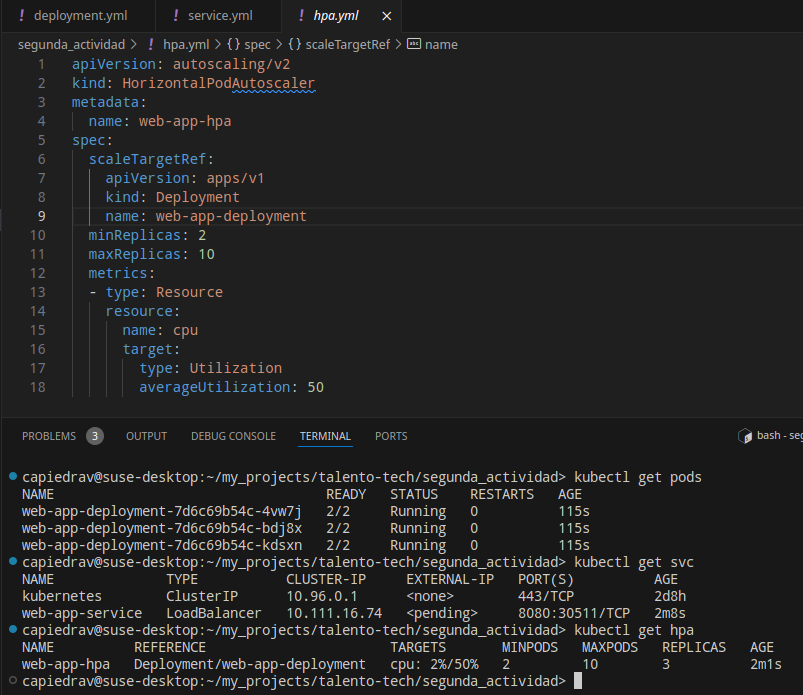
****

1. Escalar el despliegue:

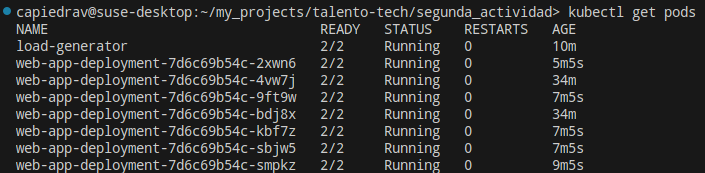


**Segunda actividad – Escalado en Kubernetes**

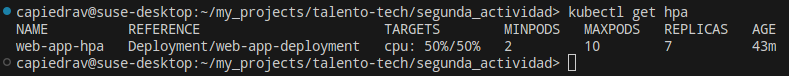
1. Aplicar manifiestos de deployment, servicio, hpa y verificar que estén funcionando:



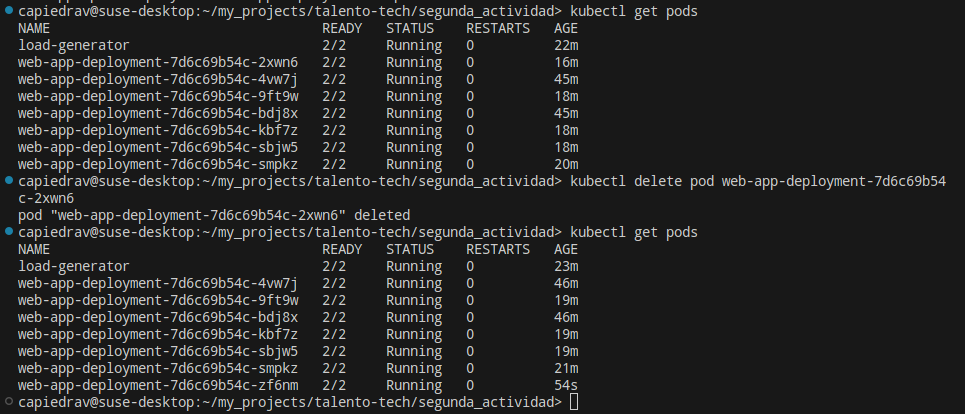
1. Generar tráfico para la aplicación para activar el escalado automático:



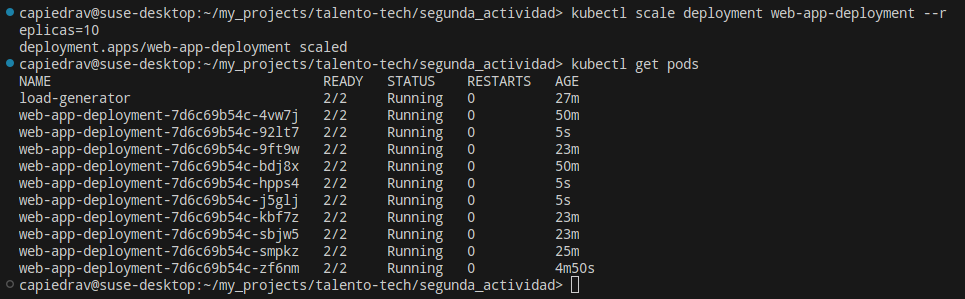
1. Verificar el estado del hpa:



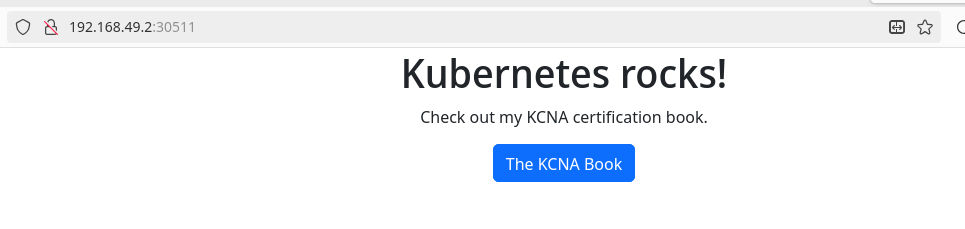
1. Eliminar un pod y verificar que se cree nuevamente:



1. Escalar manualmente el despliegue:

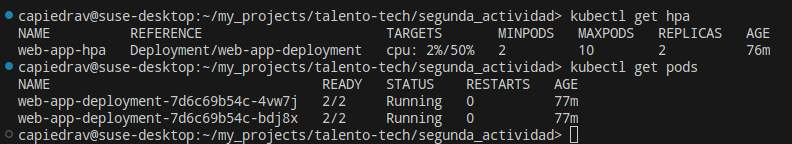


1. Acceder a la aplicación y verificar que sigue funcionando:



1. Después de que la carga de trabajo disminuye, verificar que los pods en exceso se eliminan:





**Tercera Actividad – Seguridad en Contenedores Docker**

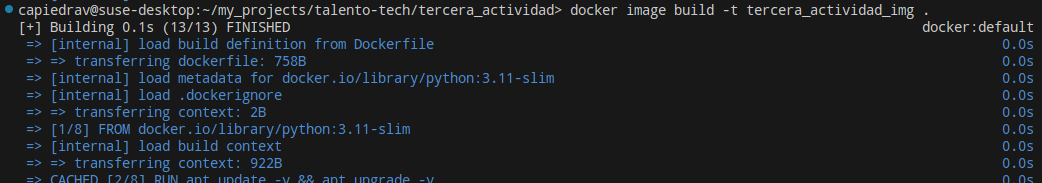
Actividad basada en el video [Docker Security Best Practices](https://www.youtube.com/watch?v=9xz9sBjZwXs)

En el video se propone como buenas prácticas de seguridad usar una imagen liviana, actualizar los paque

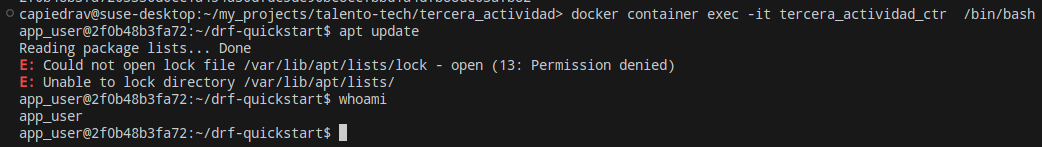
tes de la imagen, crear un usuario sin permisos de administrador y limitar la cantidad de recursos que puede usar el contenedor una vez esté ejecutandose.

1. Usar una imagen liviana, actualizar los paquetes de la imagen, crear un usuario sin permisos de administrador:





1. Verificar que el usuario no tiene permisos de administrador:



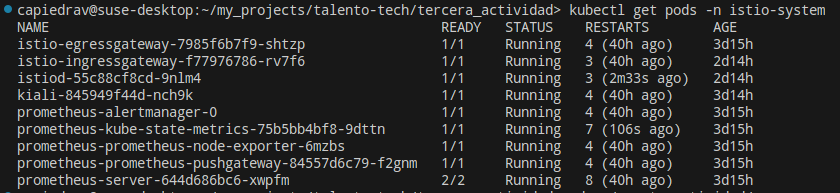
1. Limitar la cantidad de recursos que puede usar el contenedor una vez esté ejecutandose:



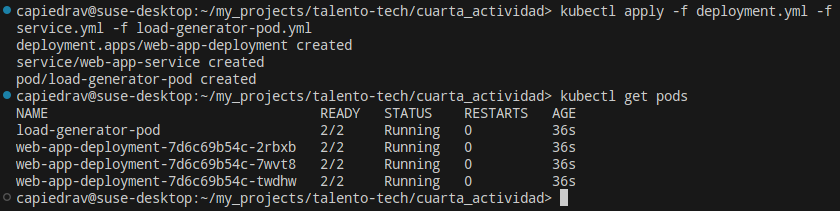


**Cuarta Actividad - Monitoreo de Aplicaciones en Kubernetes con Istio, Kiali y Prometheus**

1. Instale Istio, Kiali y Prometheus:



1. Lanzar aplicaciones:

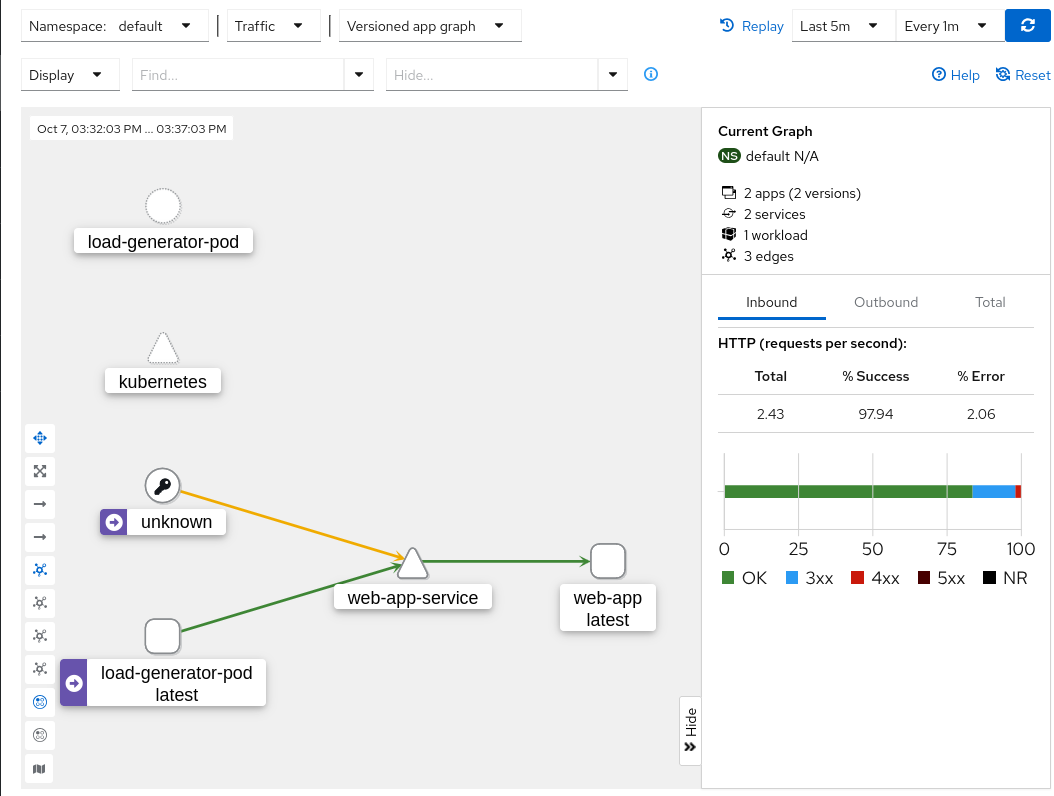


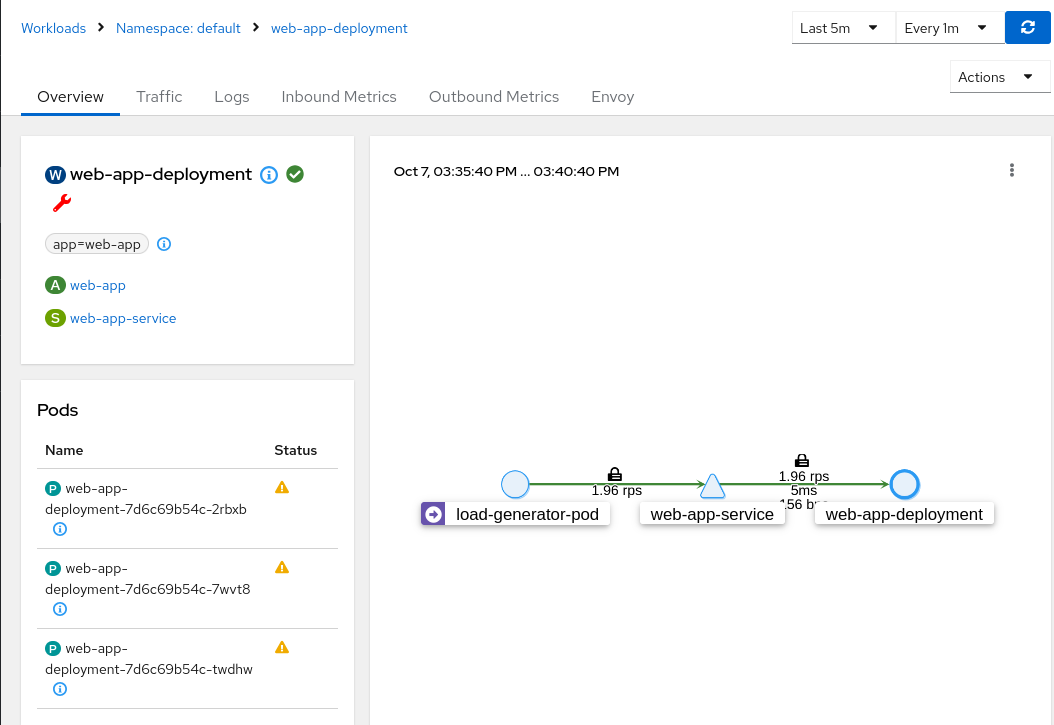
1. Lanzar el dashboard de Kiali:



**Continúa en la siguiente página**

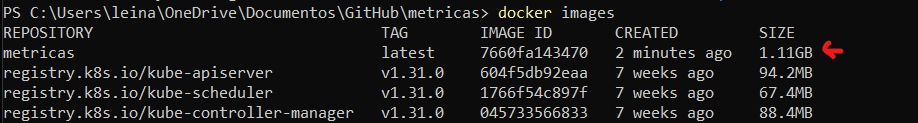
1. Monitorear el cluster desde Kiali:



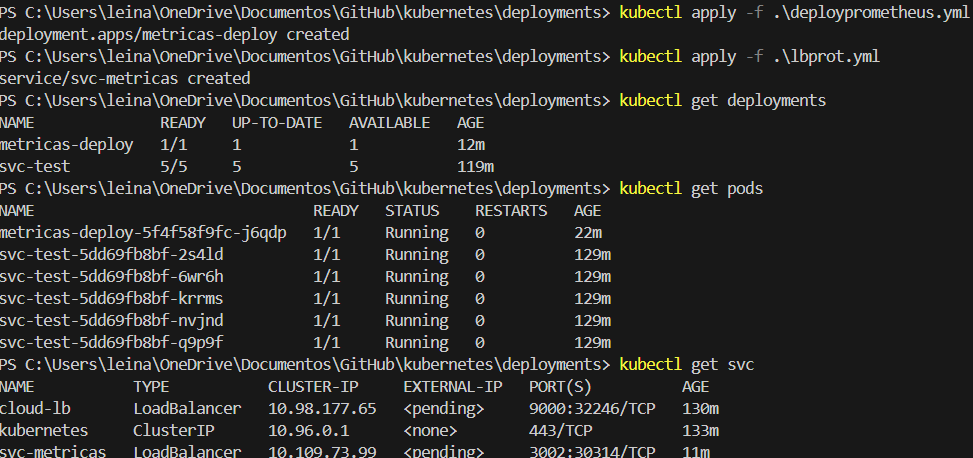


**Cuarta Actividad Parte 2 – Monitorización de un clúster Kubernetes mediante Prometheus y Grafana**

1. Configuración de la aplicación para pruebas con Prometheus y Grafana, primero se debe crear la imagen y subirla a Docker-hub



1. Ingresamos al folder de deployments compartido en kubernetes (osmergit) en github y cargamos los manifiestos:

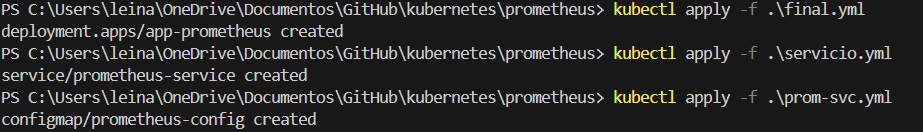


1. Realizamos la instalación de Helm, descargando y actualizando los repositorio:

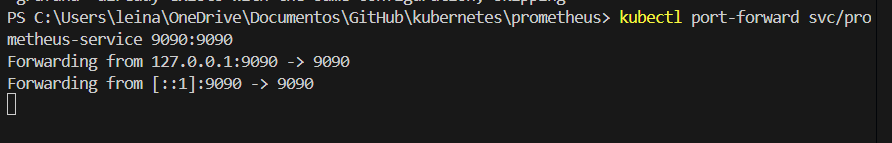
A screen shot of a computer

Description automatically generated

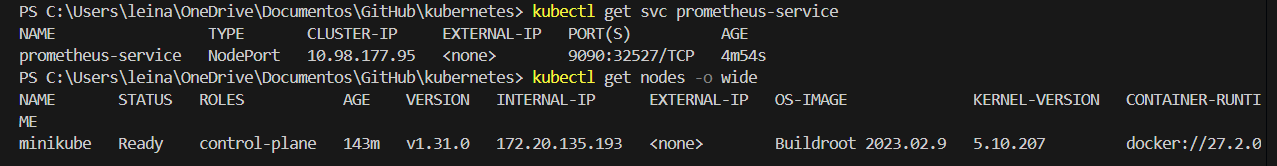
1. Ingresamos al folder de prometheus compartido en kubernetes (osmergit) en github y cargamos el deployment, el service y el configmap:



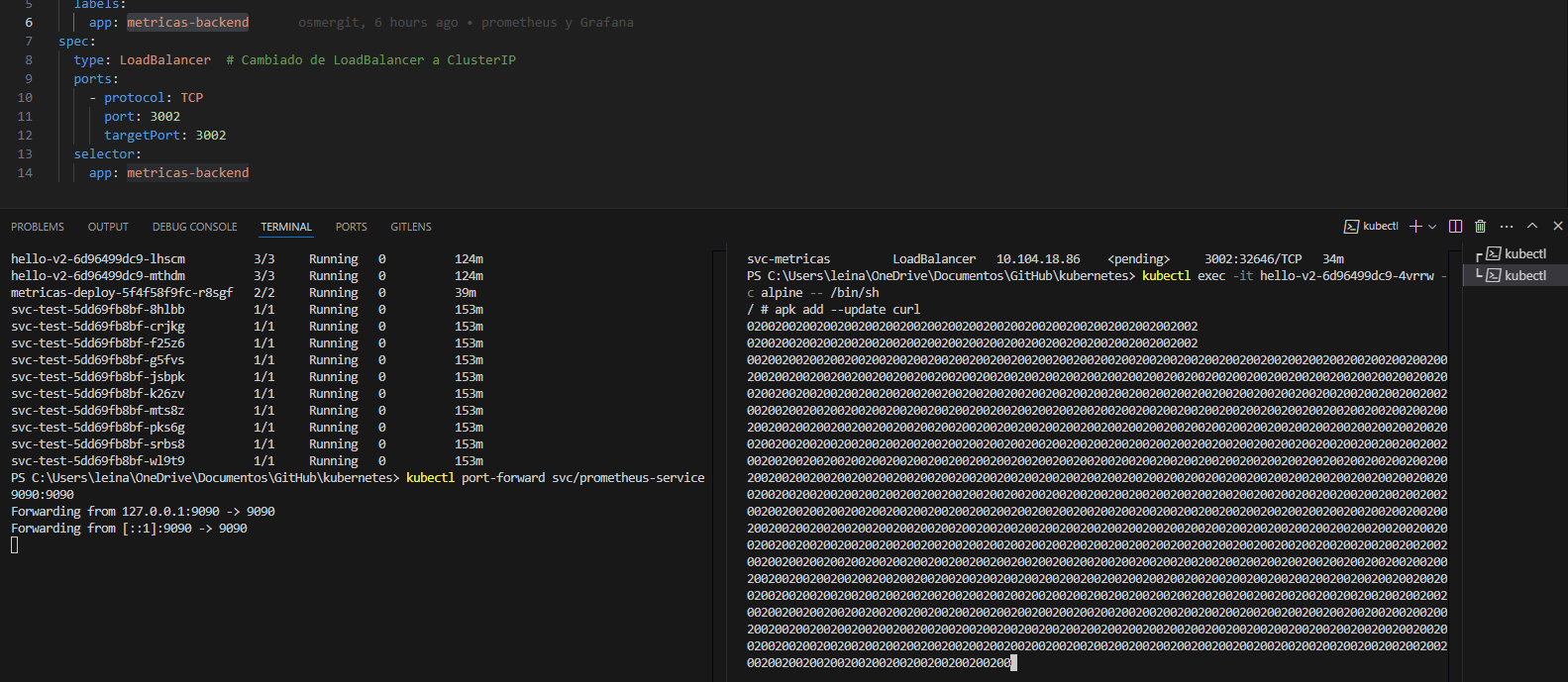
1. Lanzamos Prometheus mediante la ejecución de su servicio (port-forward):



1. Verificamos en que puerto se esta ejecutando prometheus internamente:



1. Con el objetivo de ver las métricas cargamos tráfico a la aplicación, en un pod especifico (hello-v2-…. Fue el elegido, según la documentación):



1. Después, una vez en Prometheus, podemos diferentes Querys, cómo se puede ver a continuación:

Métrica de peticiones http por método:

A screenshot of a computer

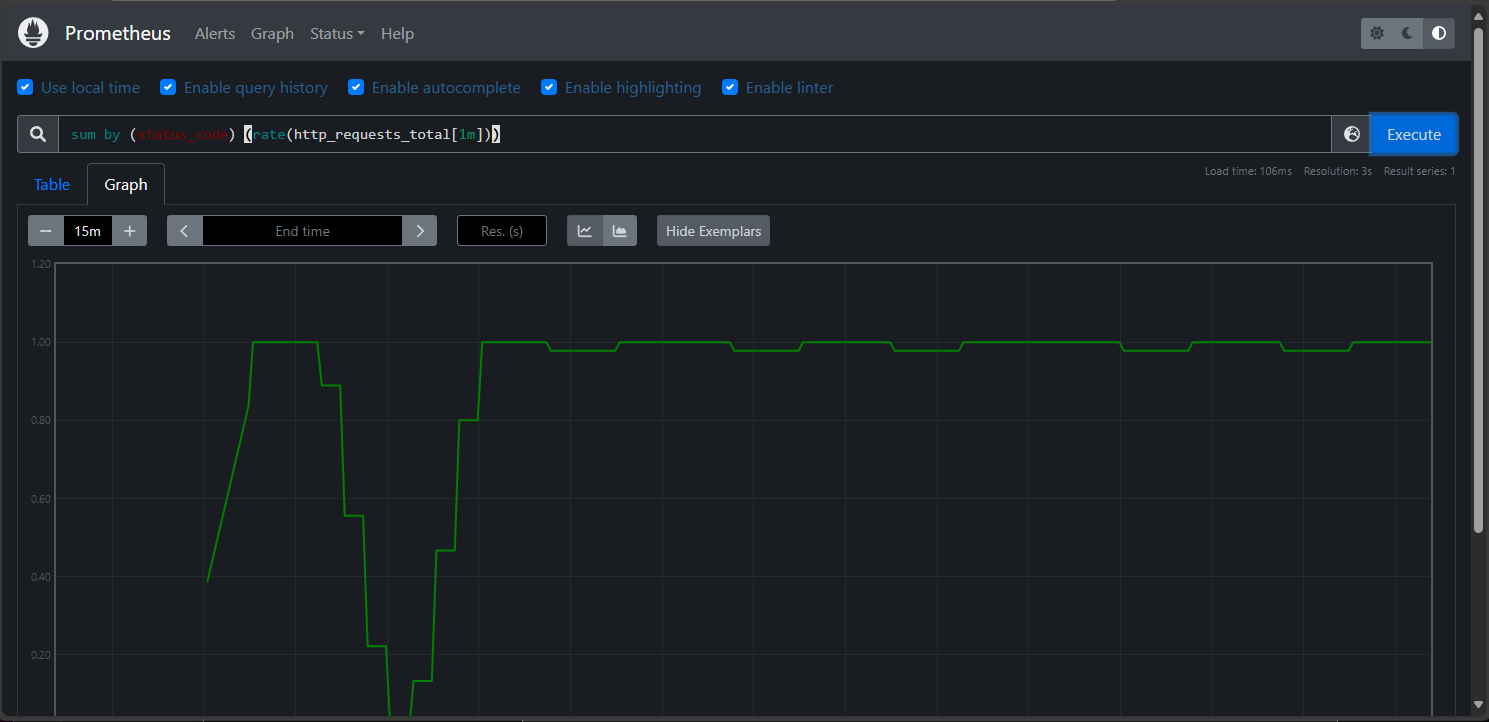
Description automatically generated

Métrica de procesos activos en cpu por segundos

A screenshot of a graph

Description automatically generated

Métrica de peticiones por código de estado:



Métrica de peticiones http por método y código de estado:

