Contenido

[INTRODUCCIÓN (APLICA PARA AMBOS CASOS) 1](#_Toc166001227)

[COMO HACER EL INGRESS-NGINX DENTRO DE UN POD 6](#_Toc166001228)

[COMO HACER EL INGRESS-NGINX FUERA DE UN POD 12](#_Toc166001229)

# INTRODUCCIÓN (APLICA PARA AMBOS CASOS)

Primero hice el despliegue común y corriente



Como se puede ver, el frontend funciona en el puerto 30102.

Primero, se revisa que se tenga un cluster activo corriendo. Para esto, se usa el comando “kubectl get nodes” y vemos que está el cluster “docker-desktop” corriendo bien

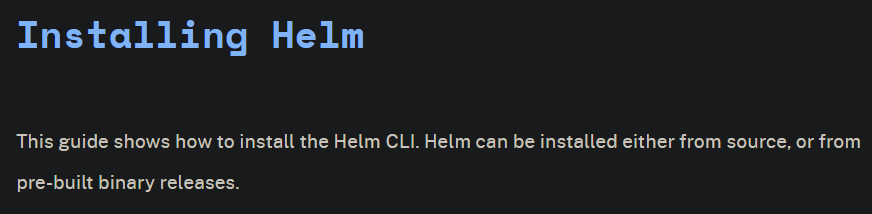


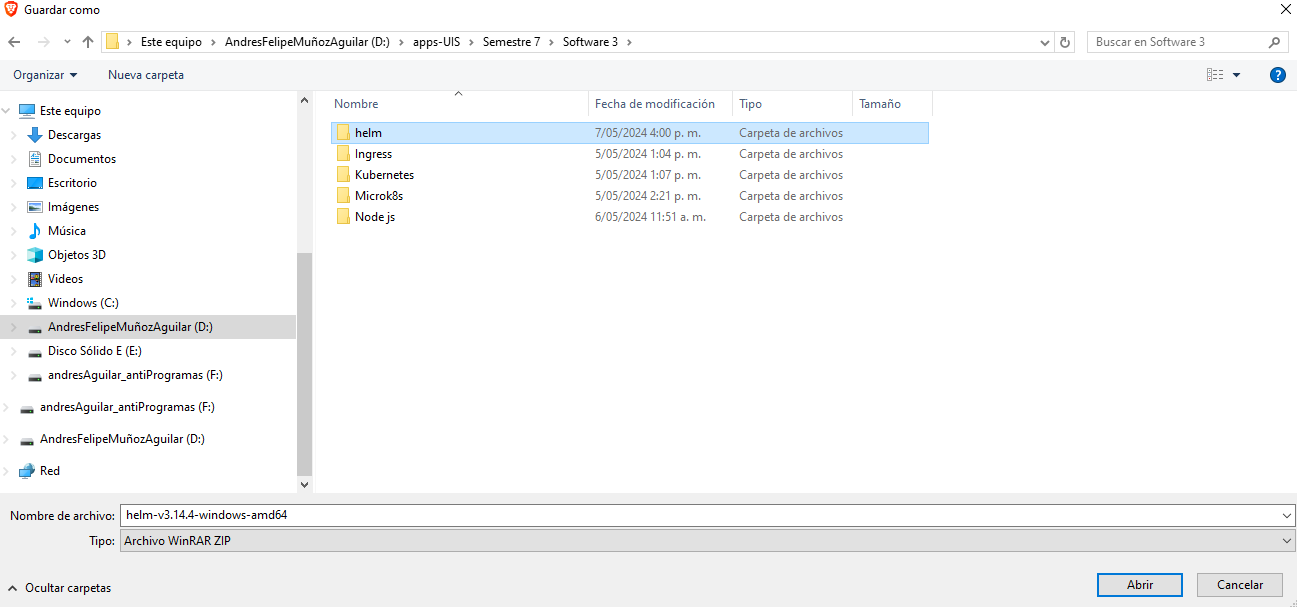


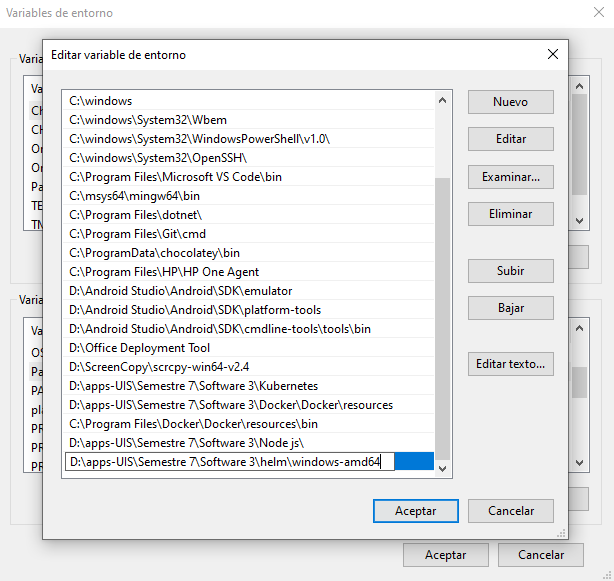
Hay que recordar que es en este cluster dentro del cual corren todas los pods (contenedores) y servicios de la aplicación de bookstore-review.

Luego se instalan 3 cosas: curl, kubectl y Helm.

Aquí instalo Helm

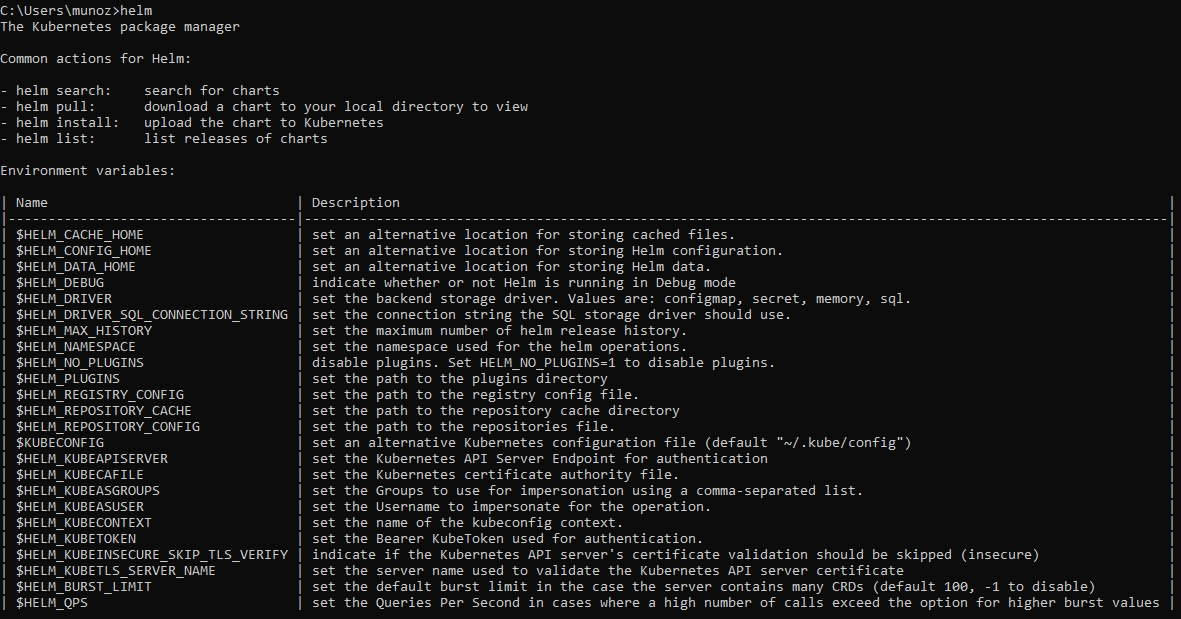






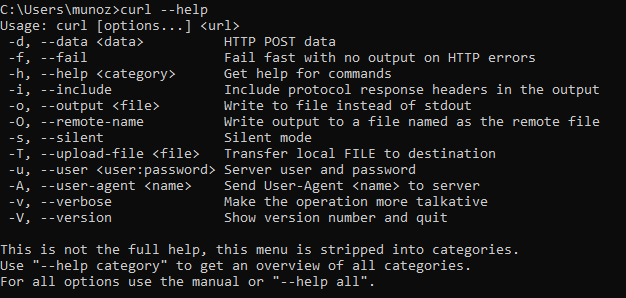
En la imagen anterior estoy añadiendo la variable de entorno

Y, finalmente, en la siguiente imagen compruebo que funcione



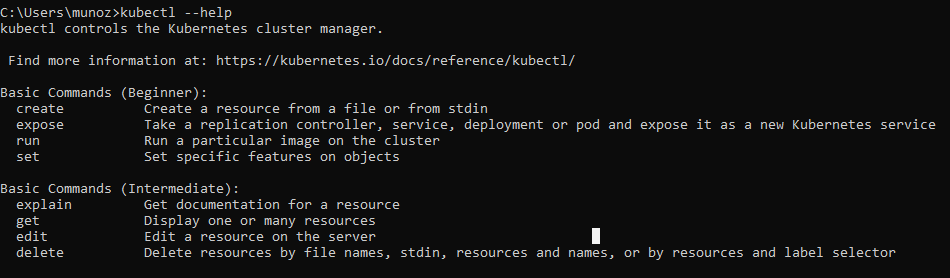
Bien, funciona bien

Ahora, a instalar curl… Oops… Se me olvidó que y a lo tengo en el PC (gracias papa Bill)



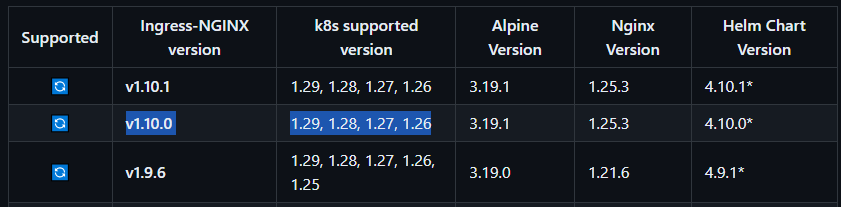
En la imagen anterior reviso que funcione bien el curl y, efectivamente, ta bien

Y ahora reviso que funcione bien el kubectl (que también ya lo tenia instalado)

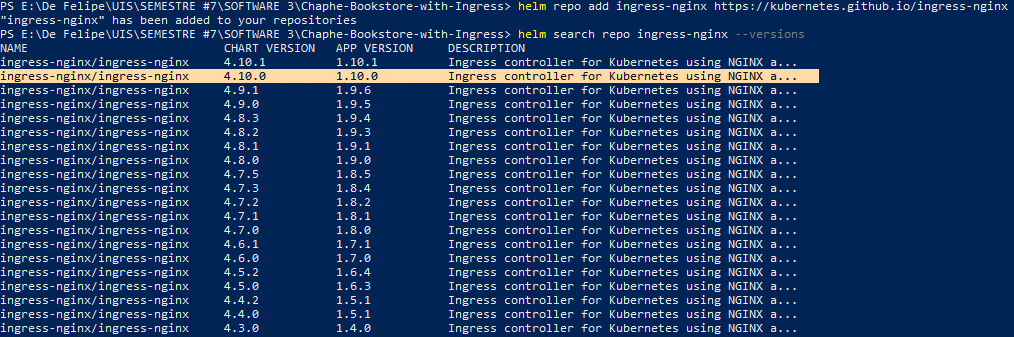


Efectivamente, todo bien

Ahora, instalamos el ingress-nginx. En mi caso, como uso Docker desktop 4.29.0, yo tengo la versión 1.29.2 de kubernetes. Por tanto, como necesito que el nginx-ingress sea compatible con mi versión de kubernetes, voy a usar la versión 1.10.0 de nginx-ingress

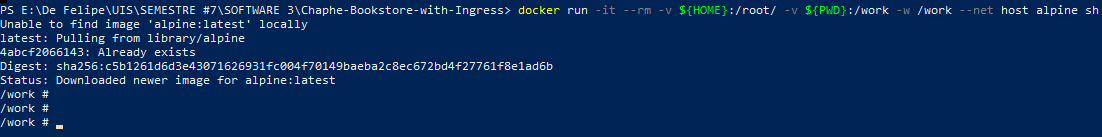


Entonces, corro el comando para añadir ingress-nginx a mis repositorios “helm search repo ingress-nginx --versions” y, luego, busco la versión de ingress-nginx que necesito. Esto lo hago en la siguiente imagen

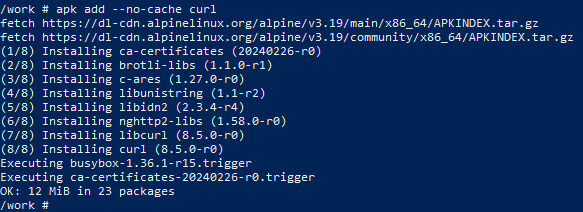


# COMO HACER EL INGRESS-NGINX DENTRO DE UN POD

Ahora, creamos un contenedor de alpine, para instalar estas cosas dentro del contenedor de alpine y que, desde aquí, se hagan las cosas de nginx

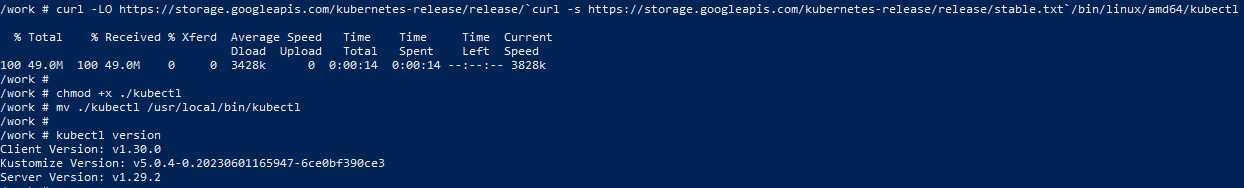


Ahí está, ahora, todo lo que instalamos en el PC, se debe instalar también en el contenedor de Alpine Linux

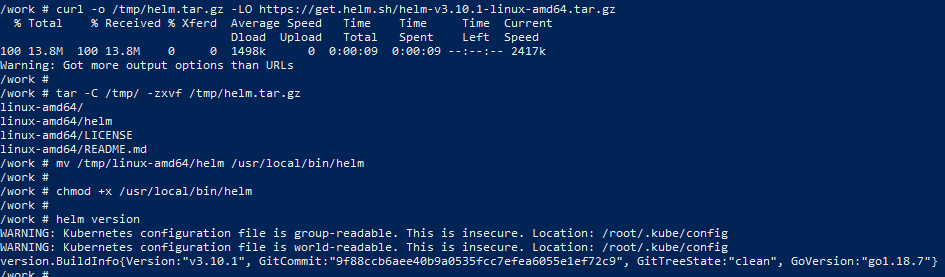


En la anterior imagen instalamos el curl en el Alpine Linux

Ahora, hacemos lo mismo con el kubectl. En la siguiente imagen se muestra la descarga e instalación de kubectl dentro del alpine en el contenedor



Ahora, instalamos el Helm dentro del contenedor de Alpine. Esto se hace en la siguiente imagen



Ahora, vemos si el Alpine percibe el contenedor que tiene. Esto se hace en la siguiente imagen.



Efectivamente, lo hace.

Ahora, vemos la versión del cluster que hay dentro del Alpine



Es 1.29.2, por lo tanto, instalamos el ingress v1.10.0 que soporta versiones de k8s 1.26, 1.27, 1.28 y 1.29.

Ahora, creamos 2 variables de entorno dentro del Alpine

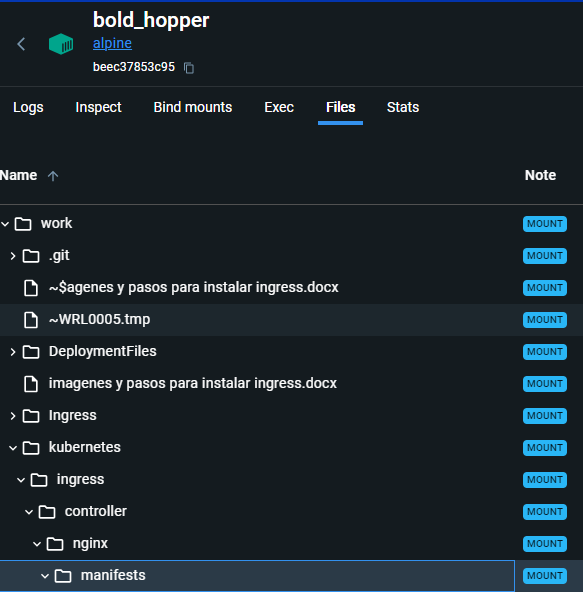


Eso es para poder descargar la versión correcta de nginx-ingress.

Ahora, creamos un directorio de manifiesto dentro de los archivos del alpine. Eso lo hacemos con el siguiente comando de la siguiente imagen:

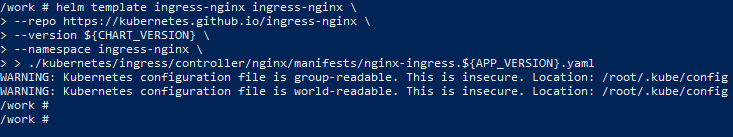


Ahora, en la siguiente imagen, nos vamos a Docker para revisar si efectivamente se creó ese directorio.

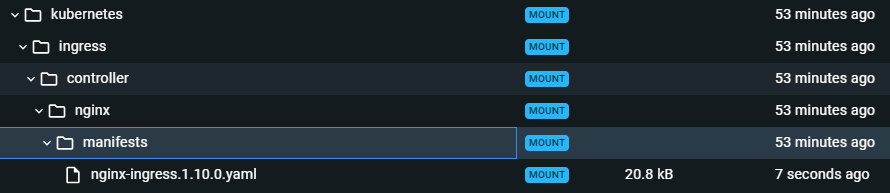


Yep… ahí está el directorio manifest.

Ahora, dentro de ese directorio vamos a crear un archivo YAML para el manejo de versiones del nginx-ingress. Para ello, usamos un template que va a llevar por dento el repositorio de donde se descarga el nginx-ingress, la versión del chart que, para este caso, es la versión 4.10.0 del chart y, finalmente, el namespace en el que actuará este nginx ingress. Finalmente, el ultimo comando es para hacer una salida a un archivo de nombre nginx-ingress.1.10.0.yaml. Esto es así porque app\_version fue una variable de entorno que declaramos hace unas 3 imágenes atrás

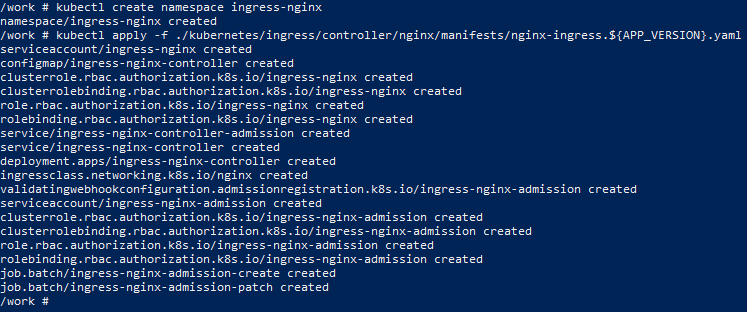


Ahora, en la siguiente imagen revisamos en Docker desktop que esté el archivo.

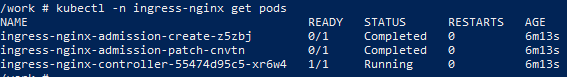


Efectivamente. Todo bien

Ahora, en la siguiente imagen se hace el deployment del nginx. Primero, se crea el namespace y luego se hace el apply del archivo YAML que creamos en el paso anterior



Ahora, en la siguiente imagen, se revisa que estén corriendo los pods necesarios para el nginx-ingress



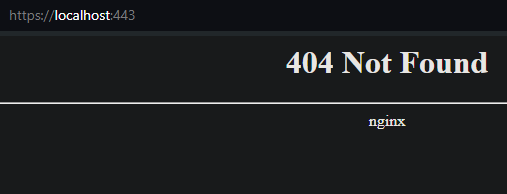
Este ingress puede ser alcanzado y testeado si, primero, revisamos sus servicios



Vemos que el servicio de tipo clusterIp tiene el puerto 443. Así que, para poder revisar si el nginx está funcionando, tenemos que ingresar al puerto 443. Pero… Como el nginx-ingress está dentro del contenedor alpine que creamos, primero, tenemos que hacer un forward desde el puerto 443 hacia el puerto 443. Esto se puede ver en la siguiente imagen



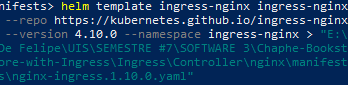
Ahora, en la siguiente imagen, nos vamos al navegador y vemos el puerto 443



Vemos que responde bien

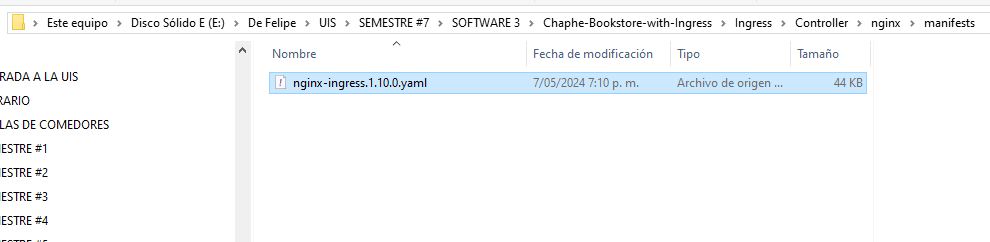
# COMO HACER EL INGRESS-NGINX FUERA DE UN POD

Siguiendo con el tutorial. Lo que hacemos ahora es crear, a través de consola, una archivo YAML que permita hacer un control de versiones si, alguna vez, se llega a hacer un reversionamiento del ingress-nginx.



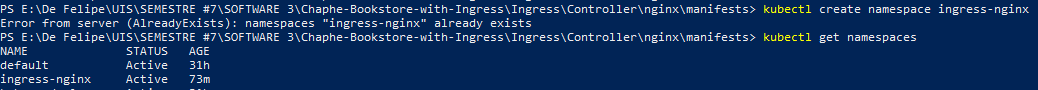
En este comando se ven 3 cosas: Primero, el Helm template que es la orden que crea el archivo yaml; la flag “-- repo” que indica de que repositorio se saca el ingress-nginx; la flag “-- version” que es la versión que se está usando actualmente en el ingress-nginx; la flag “-- namespace” que indica el namespace donde trabajará el ingress-nginx y, finalmente, la orden “> “path/to/directory””. Esta última orden lo que dice es que haga una salida en forma de archivo yaml al directorio "E:\De Felipe\UIS\SEMESTRE #7\SOFTWARE 3\Chaphe-Bookstore-with-Ingress\Ingress\Controller\nginx\manifests\". Es decir, dentro de esa carpeta cree el archivo “nginx-ingress.1.10.0.yaml” con las especificaciones que le di.

Ahora revisamos que ahí esté el archivo



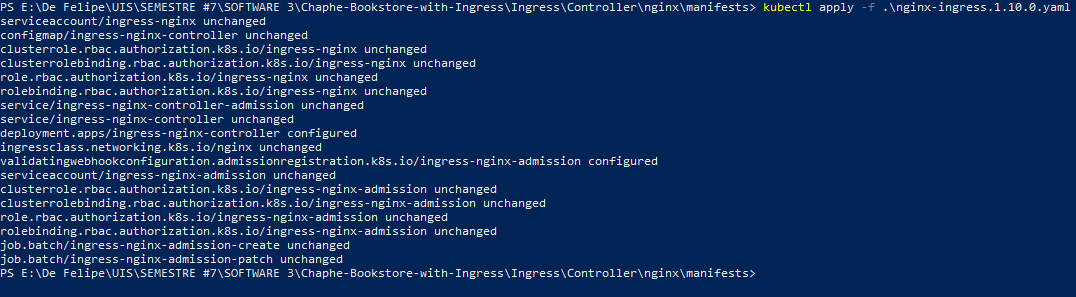
Vemos que sí está ahí creado.

Ahora, en la siguiente imagen, creamos el namespace donde va a correr el ingress-nginx

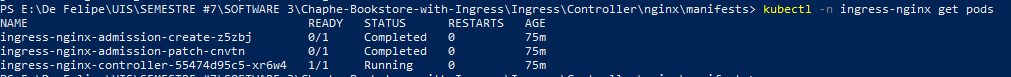


Todo bien

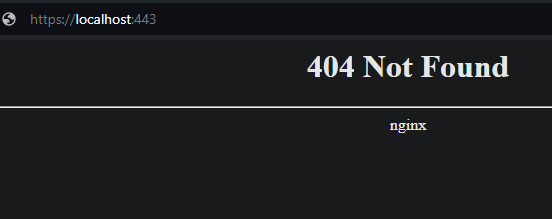
Ahora, lo que haremos será hacer el despliegue del ingress-nginx. Para esto, simplemente hacemos apply al archivo yaml que acabamos de crear



Y revisamos que, en este namespace (ingress-nginx), se hayan creado correctamente los pods que serán usados para el ingress-nginx

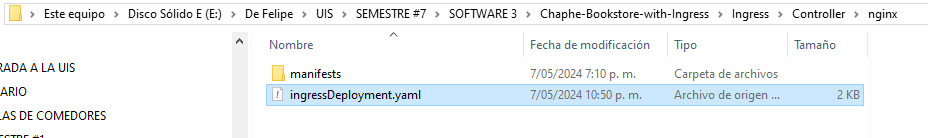


Vemos que todo bien.



Finalmente, revisamos que el servicio esté activo y todo bien y vemos que, efectivamente, ahí está corriendo el nginx desde el PC natural, sin necesidad de correr dentro de un pod de Alpine.

Ahora, en la siguiente ruta creamos un nuevo archivo YAML que corresponde al deployment del ingress con rutas basadas en “path”



Dentro de ese YAML se encuentra el siguiente código:

*apiVersion: networking.k8s.io/v1*

*kind: Ingress*

*metadata:*

*name: servicio-ingress-catalog*

*spec:*

*ingressClassName: nginx*

*rules:*

*- host: chaphe-bookstore.catalog.com*

*http:*

*paths:*

*- path: /*

*pathType: Prefix*

*backend:*

*service:*

*name: frontend-catalog-service*

*port:*

*number: 80*

*---*

*apiVersion: networking.k8s.io/v1*

*kind: Ingress*

*metadata:*

*name: servicio-ingress-reviews*

*spec:*

*ingressClassName: nginx*

*rules:*

*- host: chaphe-bookstore.reviews.com*

*http:*

*paths:*

*- path: /*

*pathType: Prefix*

*backend:*

*service:*

*name: frontend-reviews-service*

*port:*

*number: 80*

*---*

*apiVersion: networking.k8s.io/v1*

*kind: Ingress*

*metadata:*

*name: servicio-ingress-store*

*spec:*

*ingressClassName: nginx*

*rules:*

*- host: chaphe-bookstore.store.com*

*http:*

*paths:*

*- path: /*

*pathType: Prefix*

*backend:*

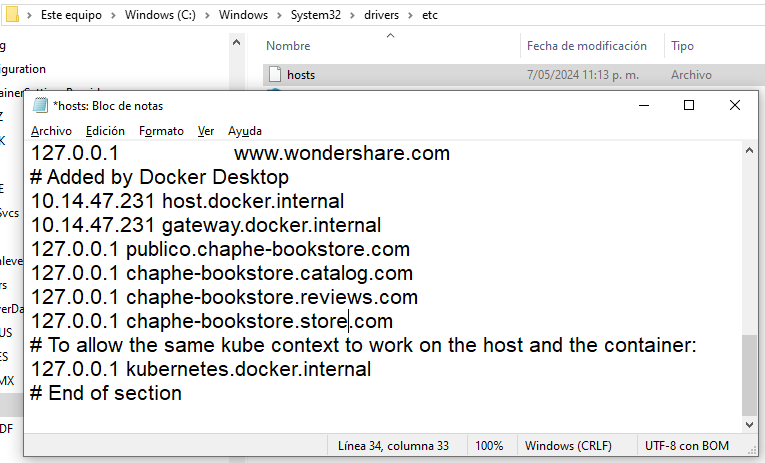
*service:*

*name: frontend-store-service*

*port:*

*number: 80*

Luego, añadimos la dirección DNS al archivo “hosts” de la ruta “Windows/System32/drivers/etc” para que se reconozca el Domain Name al escribirlo en el navegador

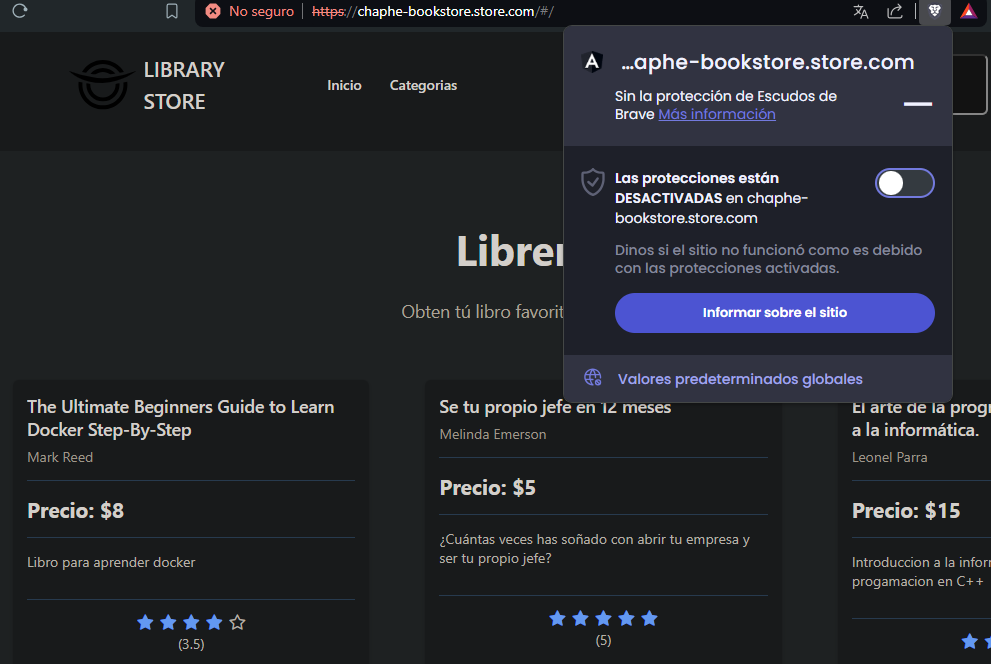


En este caso, hice que redirigiera al localhost.

Y, tras hacer el deployment, este es el comportamiento



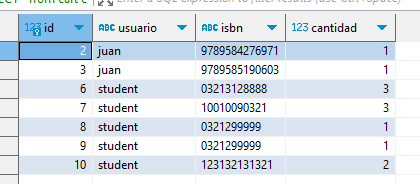
Todo bien, pero, no me cargan los libros… Esto es porque BRAVE me bloquea las extensiones automaticamente, pero, si le desactivo estas extensiones, se cargan los libros como se muestra en la imagen siguiente



Y hasta se pueden añadir productos al carrito y todo



Y esto se ve reflejado en la base de datos



Ya con esto, sólo falta traducirlo a unas diapos