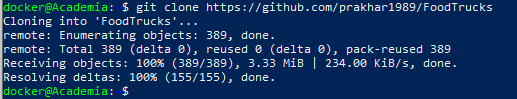
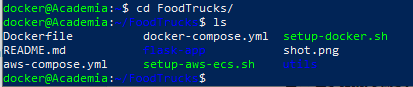
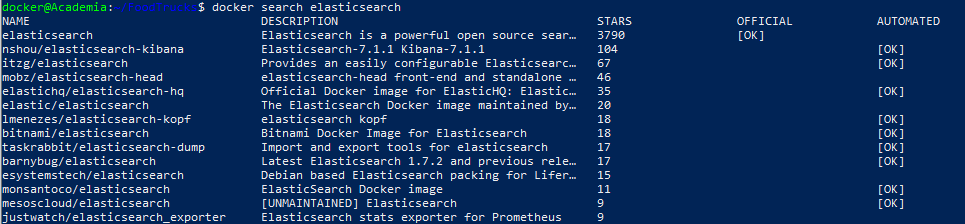
Practica Docker

1.- Clonar el repositorio

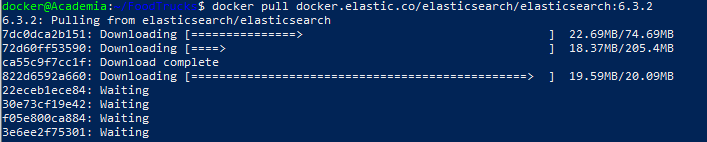


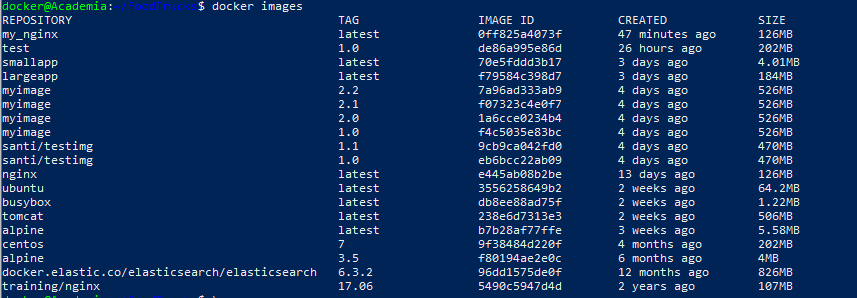


2.- Buscar si existe en el Docker Hub el elasticsearch



3.- Hacer pull de la imagen elastic.co/elasticsearch/elasticsearch:6.3.2



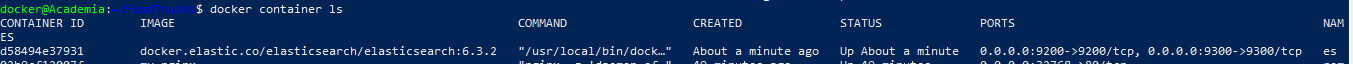


4.- Correr un contenedor especificando los puertos y las configuraciones de las variables de entorno para configurar el Elasticsearch cluster para correr en un solo nodo. El contenedor tendrá el nombre de *es* para hacer más los próximos pasos. El comando es el siguiente:

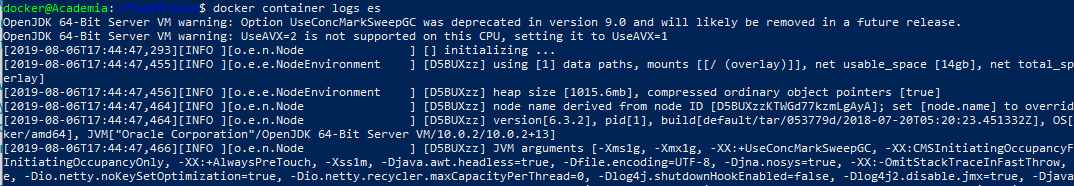
*docker run -d --name es -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" docker.elastic.co/elasticsearch/elasticsearch:6.3.2*



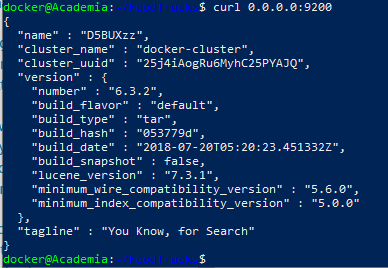
5.- Listar el contenedor



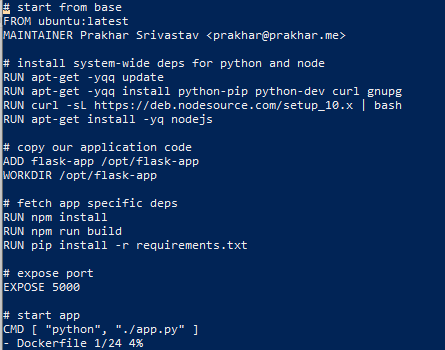
6.- Visualizar los logs que están en el nuevo contenedor *es*



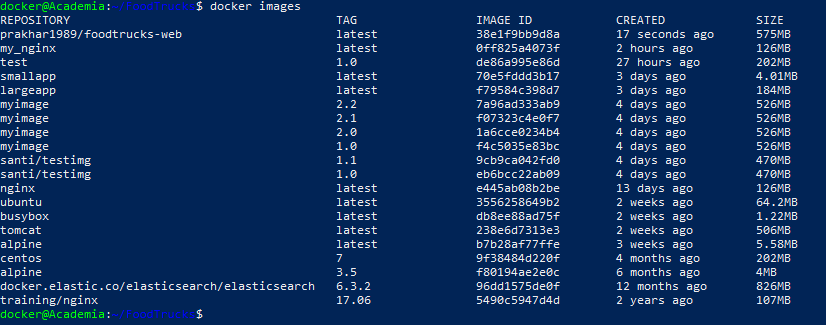
7.- Ver si se realizó el mapeo al puerto 9200 con el comando *curl*



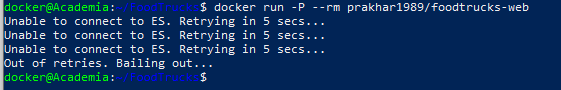
8.- Utilizar el Dockerfile que existe en el repositorio clonado



9.- Crear una imagen en base a ese Dockerfile



10.- Levantar la app corriendo un nuevo contenedor con la imagen creada



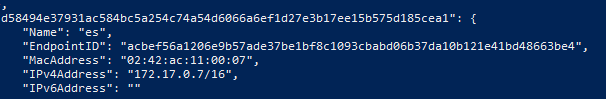
No es posible levantar el contenedor ya que este requiere establecer una conexión con otro, pero esta conexión no se ha establecido.

11.-Ver el puerto del contenedor *es*

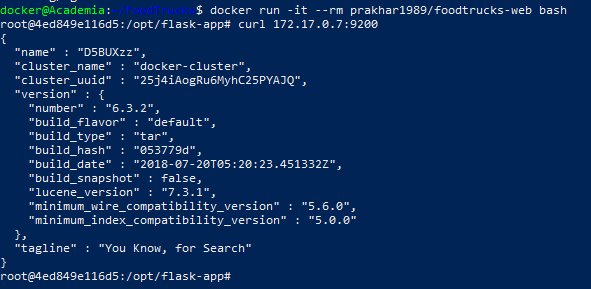
Con el comando *docker ps* podemos ver el puerto del contenedor



12.- Inspeccionar la red Bridge para ver los contenedores que forman parte de esta red y nos muestra la siguiente información acerca del contenedor *es*

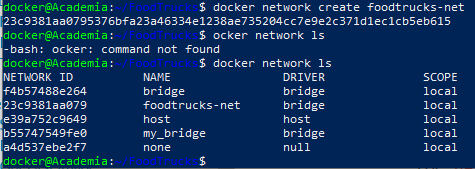


13.- Correr un contenedor con la imagen y ver si funciona el flask por medio de la IP utilizando el comando *curl*. Puede observarse que aquí si se puedo establecer la conexión con el contenedor *es* por medio de su IP y el puerto.

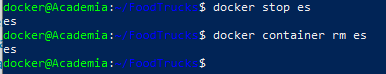


Un problema que surge es que no hay forma de crear una conexión indicando que es con el contenedor llamado *es* y no por su IP, para esto se requiere crear una nueva network para poder establecer conexión por medio de los nombres.

14.- Creando la nueva network

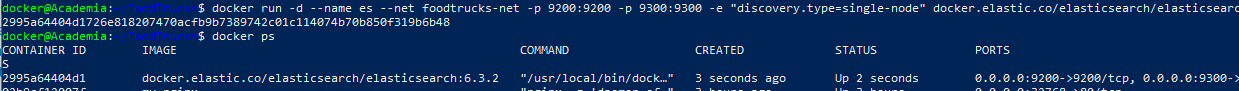


15.- Ahora para se requiere crear los contenedores y conectarlos a la nueva network creada, pero primero se requiere detener el contenedor *es* y borrarlo.

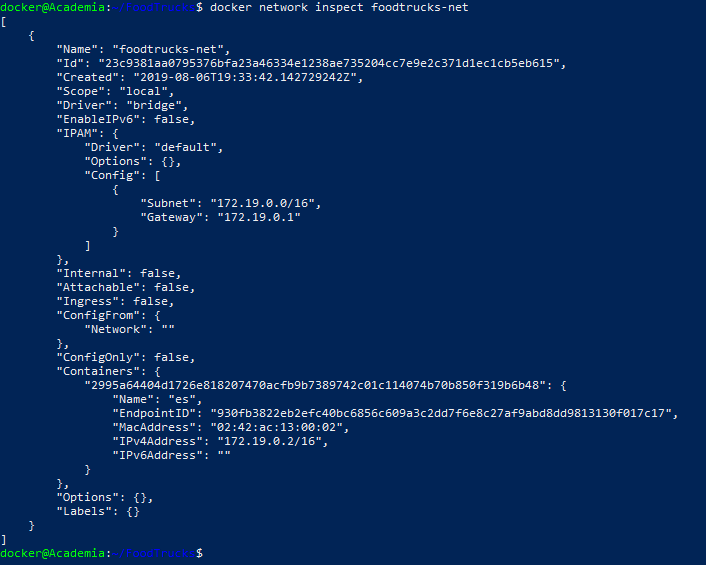


16.- Crear un nuevo contenedor y que esté conectado a la nueva red.

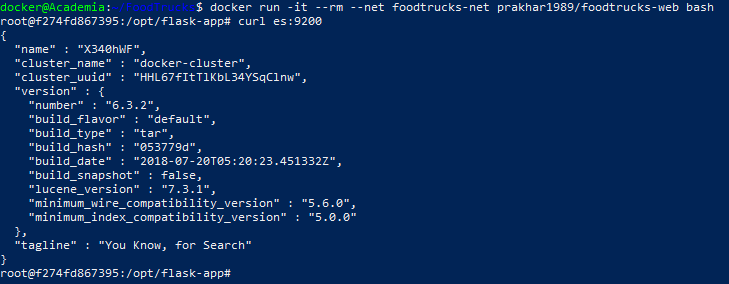
*docker run -d --name es --net foodtrucks-net -p 9200:9200 -p 9300:9300 -e "discovery.type=single-node" docker.elastic.co/elasticsearch/elasticsearch:6.3.2*



17.- Para verificar que el contenedor si se encuentra dentro de la nueva network inspeccionamos esa red y debería encontrarse el contenedor.



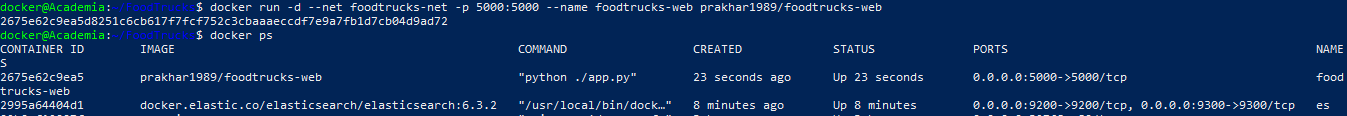
18.- Ahora se creará un nuevo contenedor que forme parte de esa red y ver si la conexión por medio del *curl* se puede realizar con el nombre en lugar de la IP.



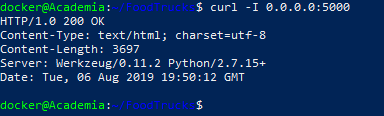
Ahora puede establecerse la comunicación sin la necesidad de escribir la dirección IP, con solo saber el nombre del contenedor la conexión será posible gracias a que estamos utilizando una network creada y no una default de docker.

19.- Crear el flask container que se va a utilizar

*docker run -d --net foodtrucks-net -p 5000:5000 --name foodtrucks-web prakhar1989/foodtrucks-web*

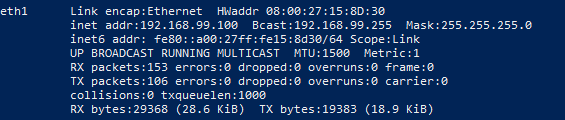


20.- La aplicación ya está levantada, primero se puede hacer la prueba utilizando el comando *curl*



21.- Para corroborar el funcionamiento se puede visualizar en el navegador, para hacer esto se requiere la IP de la docker machine y en el navegador se escribe la dirección IP y el puerto

Con el comando ifconfig encontramos la dirección IP



Ya encontramos la IP ahora en el navegador se escribe y concatenándole el puerto

*192.168.99.100:5000*

Resultado:

