

2021

Práctica Final Docker

MARKEL ORALLO NOGUEIRA

¿Sabeis lo que es Elasticsearch? ¿Podéis añadir una breve descripción?

Elasticsearch es una base de datos no relacional basada en lucene. Forma parte de la pila ElasticStack y a dia de hoy es muy popular en el mercado. Su principal potencial es la búsqueda de texto en los documentos insertados en ella.

\$docker search elasticsearch

- ¿Qué has obtenido?

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
elasticsearch	Elasticsearch is a powerful open source search and analytics engine for all your text and structured data on any cloud. It is a distributed, multitenant-capable, schema-free document database that makes it easy to store, search, and analyze all kinds of unstructured and semi-structured data.	5238	[OK]	
nshou/elasticsearch-kibana	Elasticsearch-7.12.1 Kibana-7.12.1	131		[OK]
mobz/elasticsearch-head	elasticsearch-head front-end and standalone application	80		
elasticsearchhq/elasticsearch-hq	Official Docker image for ElasticHQ: Elasticsearch monitoring and management application	76		[OK]
itzg/elasticsearch	Provides an easily configurable Elasticsearch Docker image	71		[OK]
elastic/elasticsearch	The Elasticsearch Docker image maintained by the official Elasticsearch team	54		
taskrabbit/elasticsearch-dump	Import and export tools for elasticsearch	27		[OK]
imenezes/elasticsearch-kopf	elasticsearch kopf	18		[OK]
barnybug/elasticsearch	Latest Elasticsearch 1.7.2 and previous releases	17		[OK]
justwatch/elasticsearch_exporter	Elasticsearch stats exporter for Prometheus	17		
blacktop/elasticsearch	Alpine Linux based Elasticsearch Docker Image	16		[OK]
esystemstech/elasticsearch	Debian based Elasticsearch packing for LibreOffice	15		
monsantoco/elasticsearch	ElasticSearch Docker image	11		[OK]
mesoscloud/elasticsearch	[UNMAINTAINED] Elasticsearch	9		[OK]
centerforopenscience/elasticsearch	Elasticsearch	4		[OK]
dtagdevsec/elasticsearch	elasticsearch	4		[OK]
barchart/elasticsearch-aws	Elasticsearch AWS node	3		
thingswise/elasticsearch	Elasticsearch + etcd2 peer discovery	1		[OK]
away/elasticsearch-docker-beat	"Beat" extension to read logs of containers	1		[OK]
jetstack/elasticsearch-pet	An.elasticsearch image for kubernetes PetSets	1		[OK]
dsteinkopf/elasticsearch-ingest-attachment	elasticsearch + ingest-attachment to be used	1		[OK]
phenompeople/elasticsearch	Elasticsearch is a powerful open source search and analytics engine for all your text and structured data on any cloud.	1		[OK]
kuzzleio/elasticsearch	Elasticsearch container based on Alpine Linux	1		[OK]
travix/elasticsearch-kubernetes	To run ElasticSearch in kubernetes and export metrics	0		[OK]
wreulicke/elasticsearch	elasticsearch	0		[OK]

Si ejecutamos docker search elasticsearch encontramos todas las imágenes de que en su nombre tienen la cadena "elasticsearch". La primera de ellas es la imagen oficial de elasticsearch.

Añade los pantallazos para verificar que el contenedor está en marcha y los logs para observar que el servicio Elasticsearch está en marcha.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	on branch main
0af75d137b4d	docker.elastic.co/elasticsearch/elasticsearch:6.3.2	"/usr/local/bin/docker-entrypoint.sh elasticsearch"	14 hours ago	Up 14 hours	0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 0.0.0.0:9300->9300/tcp, :::9300->9300/tcp	es

¿Qué es curl?

¿Qué respuesta has obtenido?

Curl es un interprete de comandos para la transferencia de archivos(HTTP,FTP...). En este caso lo usamos para realizar una petición GET a la url <http://localhost:9200>

```
/ markel in MACC/1_cuarto/ > curl localhost:9200
{
  "name" : "Auo33wA",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "jkq-5eHJST-y1VktFDAbiQ",
  "version" : {
    "number" : "6.3.2",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "053779d",
    "build_date" : "2018-07-20T05:20:23.451332Z",
    "build_snapshot" : false,
    "lucene_version" : "7.3.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

En la respuesta vemos información sobre el servicio de elasticsearch y su configuración. Podemos ver:

- Nombre del nodo
- Nombre del cluster
- UUID del cluster
- Version y build de elasticsearch
- Version de lucene
- Otros

```
/ markel in MACC/1_cuarto/ > docker logs es
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release.
[2021-09-30T08:02:15,402][INFO ][o.e.n.Node           ] [] initializing ...
[2021-09-30T08:02:15,457][INFO ][o.e.e.NodeEnvironment ] [Auo33wA] using [1] data paths, mounts [[/ (overlay)]], net usable_space [50.1gb], net
total_space [58.4gb], types [overlay]
[2021-09-30T08:02:15,458][INFO ][o.e.e.NodeEnvironment ] [Auo33wA] heap size [989.8mb], compressed ordinary object pointers [true]
[2021-09-30T08:02:15,459][INFO ][o.e.n.Node           ] [Auo33wA] node name derived from node ID [Auo33wAFRqSY_fczMKFDU]; set [node.name] to o
verride
[2021-09-30T08:02:15,459][INFO ][o.e.n.Node           ] [Auo33wA] version[6.3.2], pid[1], build[default/tar/053779d/2018-07-20T05:20:23.451332Z]
[2021-09-30T08:02:15,459][INFO ][o.e.n.Node           ] [Auo33wA] JVM arguments [-Xms1g, -Xmx1g, -XX:+UseConcMarkSweepGC, -XX:CMSInitiatingOccu
pancyFraction=75, -XX:+UseCMSInitiatingOccupancyOnly, -XX:+AlwaysPreTouch, -Xss1m, -Djava.awt.headless=true, -Dfile.encoding=UTF-8, -Djava.nosys=tru
e, -XX:-OmitStackTraceInFastThrow, -Dio.netty.noUnsafe=true, -Dio.netty.noKeySetOptimization=true, -Dio.netty.recycler.maxCapacityPerThread=0, -Dic
g4j.shutdownHookEnabled=false, -Dlog4j2.disable.jmx=true, -Djava.io.tmpdir=/tmp/elasticsearch.5HEsqHKz, -XX:+HeapDumpOnOutOfMemoryError, -XX:HeapDu
mpPath=data, -XX:ErrorFile=logs/hs_err_pid%p.log, -Xlog:gc*,gc+age=trace,safepoint:file=logs/gc.log:utctime,pid,tags:filecount=32,filesize=64m, -Dj
ava.locale.providers=COMPAT, -XX:UseAVX=2, -Des.cgroups.hierarchy.override=/, -Des.path.home=/usr/share/elasticsearch, -Des.path.conf=/usr/share/el
asticsearch/config, -Des.distribution.flavor=default, -Des.distribution.type=tar]
[2021-09-30T08:02:17,104][INFO ][o.e.p.PluginsService   ] [Auo33wA] loaded module [aggs-matrix-stats]
[2021-09-30T08:02:17,104][INFO ][o.e.p.PluginsService   ] [Auo33wA] loaded module [analysis-common]
```

- **Completa el Dockerfile y escribe la versión final**
- **Comenta en cada línea lo que se ejecuta y para qué**

```
# Empezamos a construir nuestra imagen desde la imagen ubuntu:latest
FROM ubuntu:latest

# Ponemos una etiqueta para saber quien mantiene la imagen creada
LABEL maintainer="markel.orallo@alumni.mondragon.edu"

# Instalamos las dependencias de Python y node js para que la aplicación puede correr
# dentro del contenedor
RUN apt-get -yqq update
RUN apt-get -yqq install python3-pip python3-dev curl gnupg
RUN curl -sL https://deb.nodesource.com/setup_10.x | bash
RUN apt-get install -yq nodejs

# Copiamos el código de la aplicación dentro del contenedor
ADD flask-app /opt/flask-app
WORKDIR /opt/flask-app

# Instalamos las dependencias de la aplicación a través del fichero requirement.txt y
# hacemos el build de la aplicación.
RUN npm install
RUN npm run build
RUN pip3 install -r requirements.txt

# Actualizamos la versión de la librería requests
RUN pip3 install --upgrade requests

# Documentamos el Puerto en el que funciona la aplicación
EXPOSE 5000

# Iniciamos la aplicación
CMD [ "python3", "app.py" ]
```

- Añade pantallazos con las evidencias

```
[markel in PF_Docker/P0_DockerPraktikak_2021-main] > docker build -t markel149/imagenflask .
[+] Building 4.2s (17/17) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 120B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [auth] library/ubuntu:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 6.95kB
=> [ 1/11] FROM docker.io/library/ubuntu:latest@sha256:a0d9e826ab87bd665fcf640598a871b748b4b70a01a4f3d174d4fb02adad07a9
=> CACHED [ 2/11] RUN apt-get -yqq update
=> CACHED [ 3/11] RUN apt-get -yqq install python3-pip python3-dev curl gnupg
=> CACHED [ 4/11] RUN curl -SL https://deb.nodesource.com/setup_10.x | bash
=> CACHED [ 5/11] RUN apt-get install -y nodejs
=> CACHED [ 6/11] ADD flask-app /opt/flask-app
=> CACHED [ 7/11] WORKDIR /opt/flask-app
=> CACHED [ 8/11] RUN npm install
=> CACHED [ 9/11] RUN npm run build
=> CACHED [10/11] RUN pip3 install -r requirements.txt
=> CACHED [11/11] RUN pip3 install --upgrade requests
=> exporting to image
=> => exporting layers
=> => writing image sha256:505c9c58ecad1564ad4ca39be5de7bbd410f57f8d4d96ab5ca780d956111d91e
=> => naming to docker.io/markel149/imagenflask

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

```
| markel in PF_Docker/PO_DockerPraktikak_2021-main/ > docker push markel149/imagenflask
on branch main
Using default tag: latest
The push refers to repository [docker.io/markel149/imagenflask]
33114e72e532: Pushed
c210c1c26b94: Pushed
2e8554b2cecc: Pushed
8e7fac325aa8: Pushed
5f70bf18a086: Mounted from dockersamples/static-site
af49d139c6e6: Pushed
f2cd8570b28a: Pushed
d712f1728f28: Pushed
826dd4b9a11c: Pushed
87eed1230d52: Pushed
da55b45d310b: Mounted from library/ubuntu
latest: digest: sha256:85d81cd2a08c2dfb08e82248f55815c09898078d1b75876e2f884e9954e47e8a size: 2636
```

```
[markel in PF_Docker/PO_DockerPraktikak_2021-main/ > docker run -P --rm markel149/imagenflask
on branch main
Unable to connect to ES. Retrying in 5 secs...
Unable to connect to ES. Retrying in 5 secs...
Unable to connect to ES. Retrying in 5 secs...
Out of retries. Bailing out...
app.py:28: DeprecationWarning: Using positional arguments for APIs is deprecated and will be disabled in 8.0.0. Instead use only keyword arguments for all
APIs. See https://github.com/elastic/elasticsearch-py/issues/1698 for more information
  status = es.indices.exists(index)
```

- ¿Has conseguido ponerlo en marcha?

No, la imagen reporta un error de que no es capaz de conectarse a la base de datos de elasticsearch.

- ¿Está la red de los dos contenedores configurado de manera adecuada?

No, al usar la configuración por defecto de red de docker, la aplicación no es capaz de resolver el nombre de dns de “es”, por lo tanto no puede conectarse a la base de datos.

Docker network

¿Qué obtienes? ¿En qué IP y puerto está ejecutándose el contenedor Elasticsearch (ES)?

Elasticsearch se ejecuta en los puertos 9200 y 9300. Escucha en la dirección 0.0.0.0, esto es, en todas las interfaces del contenedor.

La IP del contenedor es 172.17.0.2 .

```
EndpointID: "e0ae1f71fc07c  
Gateway": "172.17.0.1",  
GlobalIPv6Address": "",  
GlobalIPv6PrefixLen": 0,  
IPAddress": "172.17.0.2",  
IPPrefixLen": 16
```

Mira las redes existentes y haz un inspect de la red bridge. ¿Puedes ver si tiene el contenedor “es” conectado? ¿Cuál es su IP?

```
markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker network ls  
NETWORK ID      NAME      DRIVER      SCOPE  
6f3e2f8b6a8c    bridge    bridge      local  
c44eabc6d42e    host      host      local  
5ad364c73089    none      null      local
```

```
markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker inspect bridge  
[  
  {  
    "Name": "bridge",  
    "Id": "6f3e2f8b6a8c8cf6ebcf169ff29cae1a1c7d929fa455c658e961c9981e8f0d81",  
    "Created": "2021-10-10T20:07:02.369384483Z",  
    "Scope": "local",  
    "Driver": "bridge",  
    "EnableIPv6": false,  
    "IPAM": {  
      "Driver": "default",  
      "Options": null,  
      "Config": [  
        {  
          "Subnet": "172.17.0.0/16",  
          "Gateway": "172.17.0.1"  
        }  
      ]  
    },  
    "Internal": false,  
    "Attachable": false,  
    "Ingress": false,  
    "ConfigFrom": {  
      "Network": ""  
    },  
    "ConfigOnly": false,  
    "Containers": {  
      "0af75d137b4d05b517b6c64a67b34cc5c3934977ae74720e4edd27fa4fa8ca53": {  
        "Name": "es",  
        "EndpointID": "e0ae1f71fc0709fff4406ae1f82f5d3aab2d9e6b00f35adc997305c9c04f7f3",  
        "MacAddress": "02:42:ac:11:00:02",  
        "IPv4Address": "172.17.0.2/16",  
        "IPv6Address": ""  
      }  
    },  
    "Options": {  
      "com.docker.network.bridge.default_bridge": "true",  
      "com.docker.network.bridge.enable_icc": "true",  
      "com.docker.network.bridge.enable_ip_masquerade": "true",  
      "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",  
      "com.docker.network.bridge.name": "docker0",  
      "com.docker.network.driver.mtu": "1500"  
    },  
    "Labels": {}  
  }  
]
```

La IP del contenedor de elasticsearch es : 172.17.0.2

Pon en marcha el contenedor de la aplicación Flask con el siguiente comando:

```
docker run -it --rm imagenflask bash
```

```
✓ markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker run --rm -it markel149/imagenflask /bin/bash
root@27e6455899ec:/opt/flask-app#
```

Desde el terminal del contenedor, intenta conectar con el contenedor Flask. ¿Responde? Añade pantallazos de la evidencia.

```
root@27e6455899ec:/opt/flask-app# curl 172.17.0.2:9200
{
  "name" : "4k4SB65",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "J50LX89hQQuD6dHoX2Dgrw",
  "version" : {
    "number" : "6.3.2",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "053779d",
    "build_date" : "2018-07-20T05:20:23.451332Z",
    "build_snapshot" : false,
    "lucene_version" : "7.3.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Si, el contenedor de elasticsearch responde si hacemos un curl a su IP.

- **¿Qué proceso tenemos que hacer para esto?**

Para que el contenedor con la aplicación de flask resuelva el contenedor de elasticsearch tenemos que conectarlos a una red que no sea la de por defecto de docker. De esta manera los contenedores conectados a esa red se podrán resolver a través de DNS.

- Define los pasos e implementa. Añade las evidencias mediante pantallazos.

1. Crear una red diferente a la de por defecto de docker
2. Conectar los dos contenedores a la nueva red.

```
✓ markel in P0_DockerPraktikak_2021-main/flask-app/ > docker network create final_docker
1dc045b9b36a079da76d514a7d003f0260bab5c0d620fc6941c28e331c7dd1
✓ markel in P0_DockerPraktikak_2021-main/flask-app/ > docker network ls
NETWORK ID      NAME        DRIVER      SCOPE
6f3e2f8b6a8c    bridge      bridge      local
1dc045b9b36a    final_docker  bridge      local
c44eabc6d42e    host        host        local
5ad364c73089    none        null       local
```

```
x markel in P0_DockerPraktikak_2021-main/flask-app/ > docker run -d --net final_docker --name es -p 9200:9200 -p 9300:9300 -e "discovery-type=single-node"
docker.elastic.co/elasticsearch/elasticsearch:6.3.2
ae8ea7aea27c093bd3367ad348f689e46823be5d649d723e23ba5ffe15a861ca
```

```
x markel in P0_DockerPraktikak_2021-main/flask-app/ > docker run --rm --net final_docker --name flask markel149/imagenflask
on branch main
Index not found...
Loading data in elasticsearch ...
Total trucks loaded: 629
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
app.py:28: DeprecationWarning: Using positional arguments for APIs is deprecated and will be disabled in 8.0.0. Instead use only keyword arguments for all APIs. See https://github.com/elastic/elasticsearch-py/issues/1698 for more information
status = es.indices.exists(index)
app.py:19: DeprecationWarning: The 'body' parameter is deprecated for the 'index' API and will be removed in 8.0.0. Instead use the 'document' parameter. See https://github.com/elastic/elasticsearch-py/issues/1698 for more information
res = es.index(index="sfdata", doc_type="truck", id=id, body=truck)
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://172.18.0.3:5000/ (Press CTRL+C to quit)
```

¿Ahora son capaces de descubrirse? Pon en marcha ambos contenedores y a ver si esta vez la aplicación Flask arranca sin problemas.

Lanza el contenedor Flask, pero primero entrando en su terminal

- ¿Responde? ¿Es capaz de resolver el nombre del contenedor?

```
✓ markel in P0_DockerPraktikak_2021-main/flask-app/ > docker run --rm --net final_docker --name flask -it markel149/imagenflask /bin/bash on branch main
root@87d7f55d97ba:/opt/flask-app# curl es:9200
{
  "name" : "hBRIZhC",
  "cluster_name" : "docker-cluster",
  "cluster_uid" : "3ItfZRGVsChZ0REKoUwJw",
  "version" : {
    "number" : "6.3.2",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "0533779d",
    "build_date" : "2018-07-20T05:20:23.451332Z",
    "build_snapshot" : false,
    "lucene_version" : "7.3.1",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
```

Si ahora es capaz de resolver el contenedor de elasticsearch.

- **Desde el directorio de trabajo, lanza la aplicación Python:**

```
#python3 app.py
```

```
root@87d7f5d9ba:/opt/flask-app# python3 app.py
app.py:28: DeprecationWarning: Using positional arguments for APIs is deprecated and will be disabled in 8.0.0. Instead use only keyword arguments for all APIs. See https://github.com/elastic/elasticsearch-py/issues/1698 for more information
  status = es.indices.exists(index)
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://172.18.0.3:5000/ (Press CTRL+C to quit)
```

- **¿Responde? ¿Se comunica con el contenedor de “es”?**

Si, ahora se conecta con el contenedor de elasticsearch.

- **Sal del contenedor Flask y páralo. Pon en marcha de nuevo pero esta vez sin entrar en el terminal:**

- \$ docker run -d --net **nombrered** -p 5000:5000 --name foodtrucks-web **Imagenflask**

```
markel in P0_DockerPraktikak_2021-main/flask-app/ > docker run --rm --net final_docker -p 5000:500 --name flask -d markel149/imagenflask
63d764568221201e990b76c7e7bab91e73beee4dd950b8d426e7bcd888cfb9d2
```

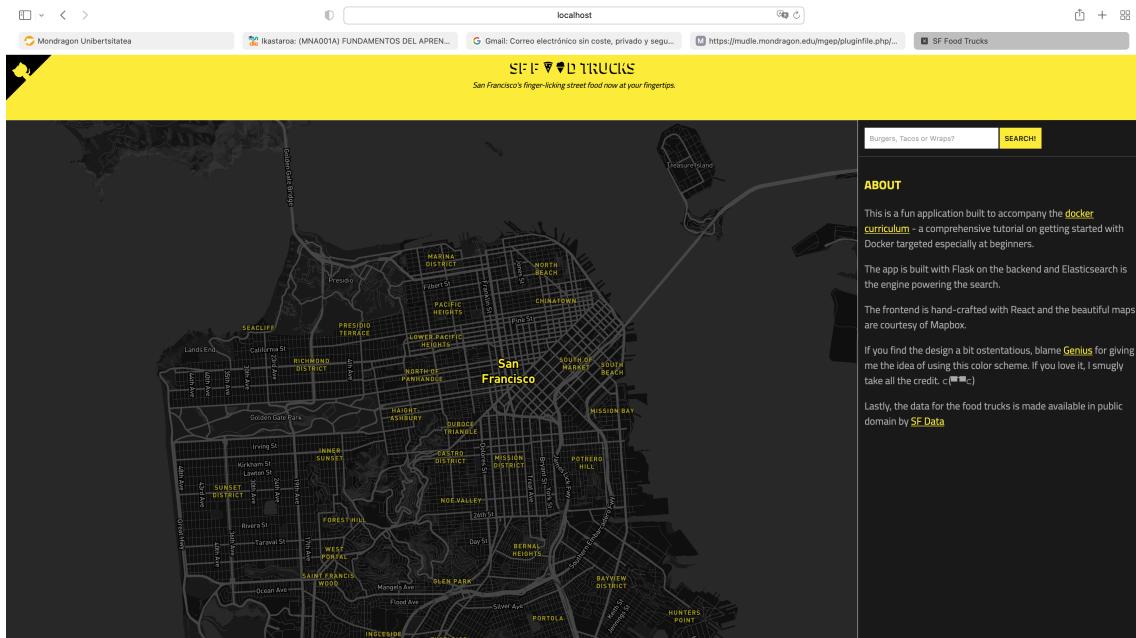
- **Comprueba que ambos contenedores están activos.**

on branch main						
CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
63d764568221	markel149/imagenflask	flask	"python3 app.py"	20 seconds ago	Up 19 seconds	5000/tcp, 0.0.0.0:5000->5000
de8ea7ae27c	docker.elastic.co/elasticsearch/elasticsearch:6.3.2	es	"/usr/local/bin/dock..."	11 minutes ago	Up 11 minutes	0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 0.0.0.0:9300->9300/tcp, :::9300->9300/tcp

- **Conecta a la aplicación Flask mediante el comando curl desde línea de comandos.**

```
markel in P0_DockerPraktikak_2021-main/flask-app/ > curl localhost:5000
<!DOCTYPE html>
<html>
<head>
  <meta charset='utf-8' />
  <title>SF Food Trucks</title>
  <meta name='viewport' content='initial-scale=1,maximum-scale=1,user-scalable=no' />
  <link href='https://fonts.googleapis.com/css?family=Titillium+Web:400,700' rel='stylesheet' type='text/css'>
  <script src='https://api.mapbox.com/mapbox-gl-js/v1.9.1/mapbox-gl.js'></script>
  <link href='https://api.mapbox.com/mapbox-gl-js/v1.9.1/mapbox-gl.css' rel='stylesheet' />
  <link href='/static/styles/main.css' rel='stylesheet' />
  <link rel='stylesheet' href='http://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css' />
  <link rel="apple-touch-icon" sizes="57x57" href="/static/icons/apple-icon-57x57.png">
  <link rel="apple-touch-icon" sizes="60x60" href="/static/icons/apple-icon-60x60.png">
  <link rel="apple-touch-icon" sizes="72x72" href="/static/icons/apple-icon-72x72.png">
  <link rel="apple-touch-icon" sizes="76x76" href="/static/icons/apple-icon-76x76.png">
  <link rel="apple-touch-icon" sizes="114x114" href="/static/icons/apple-icon-114x114.png">
  <link rel="apple-touch-icon" sizes="120x120" href="/static/icons/apple-icon-120x120.png">
  <link rel="apple-touch-icon" sizes="144x144" href="/static/icons/apple-icon-144x144.png">
  <link rel="apple-touch-icon" sizes="152x152" href="/static/icons/apple-icon-152x152.png">
```

- Abre un navegador y conecta a la ip local del host (127.0.0.1) y en el puerto de la aplicación 5000). ¿Qué has obtenido?



Docker compose

Completa el docker-compose.yml con los datos que faltan y explica lo que se está haciendo en cada paso.

```
version: "3"
services:
  #Lanzamos el contenedor de elasticsearch
  es:
    #Definimos la imagen que vamos a usar para el contenedor de elasticsearch
    image: docker.elastic.co/elasticsearch/elasticsearch:6.3.2
    #Definimos el nombre que tendrá el contenedor
    container_name: es
    #A través de una variable de entorno definimos que el cluster de elasticsearch será de
    #un único nodo.
    environment:
      - discovery.type=single-node
    # Definimos los puertos en los que escucha la base de datos de elasticsearch
    ports:
      - 9200:9200/tcp
      - 9300:9300/tcp
    # Creamos un volumen para guardar la información que guarda elasticsearch de
    #manera persistente
    volumes:
```

```

- esdata1:/usr/share/elasticsearch/data
#Lanzamos la aplicación de flask
web:
# Definimos la imagen que va a usar el contenedor de la aplicación de flask
image: markel149/imagenflask
# Definimos el comando que se ejecutará al levantar el contenedot
command: python3 app.py
# Definimos que este contenedor depende del contenedor de elasticsearch. De esta
manera hasta que elasticsearch no este levantado no se ejecutara este contenedor.
depends_on:
- es
# Definimos el Puerto donde escuchara nuestra aplicación de flask
ports:
- 5000:5000/tcp
# Definimos el volume para que cuando se levante el contenedor de flask todo el
código se monte dentro del contenedor.
volumes:
- ./flask-app:/opt/flask-app
# Definimos el volumen que usara la aplicación de elasticsearch.
volumes:
esdata1:
driver: local

```

Asegurate que los contenedores puestos en marcha en los pasos anteriores están ahora parados.

Lanza el docker-compose.yml y presenta las evidencias de que ambos contenedores están activos y la aplicación sigue funcionando.

```

markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker compose up -d
[+] Running 4/4
  ● Network po_dockerpraktikak_2021-main_default   Created                               0.0s
  ● Volume "po_dockerpraktikak_2021-main_esdata1"    Created                               0.0s
  ● Container es                                     Started                               0.5s
  ● Container po_dockerpraktikak_2021-main_web_1   Started   [2.0s]
markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
NAMES              COMMAND             CREATED           STATUS           PORTS
a3c2c66c178b        markel149/imagenflask      "python3 app.py"   5 seconds ago     Up 3 seconds      0.0.0.0:5000->5000/tcp, :::5000->5000/tcp
527e1a9fb41a        docker.elastic.co/elasticsearch/elasticsearch:6.3.2   "/usr/local/bin/dock..."  5 seconds ago     Up 4 seconds      0.0.0.0:9200->9200/tcp, :::9200->9200/tcp, 0.0.0.0:9300->9300/tcp, :::9300->9300/tcp
es

```

Borra toda la infraestructura (docker-compose down –v).

```

markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker compose down -v
[+] Running 4/4
  ● Container po_dockerpraktikak_2021-main_web_1   Removed
  ● Container es                                     Removed
  ● Network po_dockerpraktikak_2021-main_default   Removed
  ● Volume po_dockerpraktikak_2021-main_esdata1    Removed
markel in PF_Docker/P0_DockerPraktikak_2021-main/ >

```

Borra también la red antes creada para unir ambos contenedores.

```

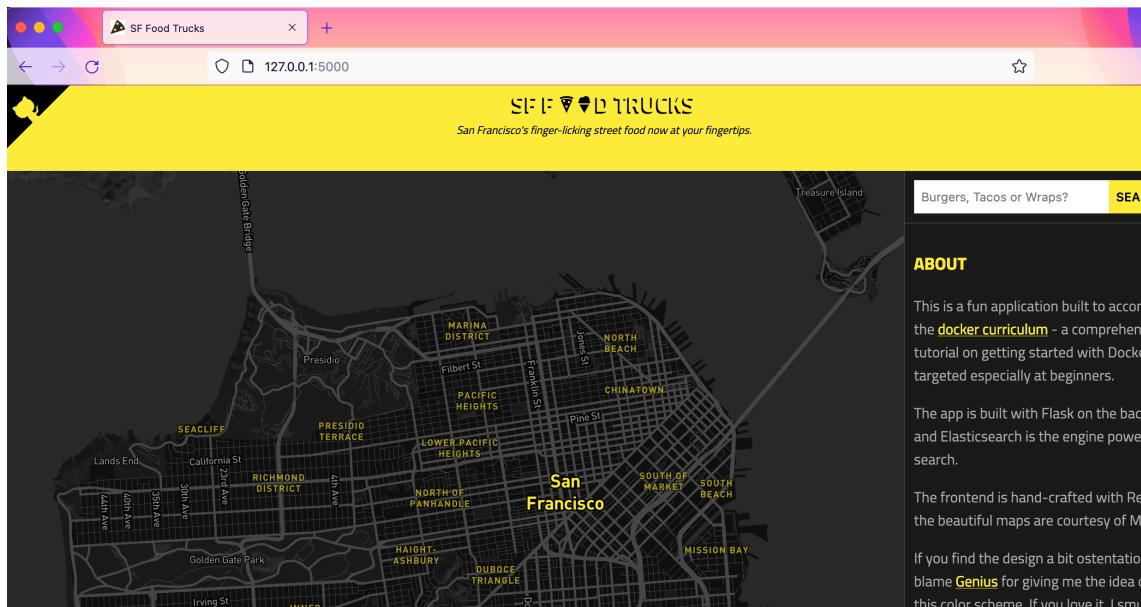
markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker network ls
NETWORK ID        NAME      DRIVER      SCOPE
6f3e2f8b6a8c      bridge    bridge      local
c44edbc6d42e      host      host      local
5ad364c73089      none      null      local
markel in PF_Docker/P0_DockerPraktikak_2021-main/ >

```

Nos aseguramos que la red que hemos creado esta borrada.

Lanza de nuevo el docker-compose y comprueba que todo funciona (añade evidencias).

```
✓ markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
58628e4860a9      markel149/imagenflask   "python3 app.py"    21 seconds ago     Up 19 seconds      0.0.0.0:5000->5000/tcp, ::1:5000->5000/tcp
f48f3c05cbae      docker.elastic.co/elasticsearch/elasticsearch:6.3.2   "/usr/local/bin/dock..."  21 seconds ago     Up 20 seconds      0.0.0.0:9200->9200/tcp, ::1:9200->9200/tcp
./markel in PF_Docker/P0_DockerPraktikak_2021-main/ > curl localhost:5000
<!DOCTYPE html>
<html>
<head>
<meta charset='utf-8' />
<title>SF Food Trucks</title>
<meta name='viewport' content='initial-scale=1,maximum-scale=1,user-scalable=no' />
<link href='https://fonts.googleapis.com/css?family=Titillium+Web:400,700' rel='stylesheet' type='text/css'>
<script src='https://api.mapbox.com/mapbox-gl-js/v1.9.1/mapbox-gl.js'></script>
<link href='https://api.mapbox.com/mapbox-gl-js/v1.9.1/mapbox-gl.css' rel='stylesheet' />
<link href='/static/styles/main.css' rel='stylesheet' />
<link rel='stylesheet' href='http://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css' />
<link rel='apple-touch-icon' sizes='57x57' href='/static/icons/apple-icon-57x57.png'>
<link rel='apple-touch-icon' sizes='60x60' href='/static/icons/apple-icon-60x60.png'>
<link rel='apple-touch-icon' sizes='72x72' href='/static/icons/apple-icon-72x72.png'>
```



Chequea las redes. ¿Hay alguna nueva? ¿Pero hemos indicado algo sobre las redes en el docker-compose.yml? ¿Qué ha pasado al lanzar el Compose? ¿Qué conclusiones sacas?

```
✓ markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker network ls
NETWORK ID          NAME                DRIVER      SCOPE
6f3e2f8b6a8c        bridge              bridge      local
c44eabc6d42e        host                host       local
5ad364c73089        none                null       local
8b29a285cd83        po_dockerpraktikak_2021-main_default  bridge      local
✓ markel in PF_Docker/P0_DockerPraktikak_2021-main/ >
```

Aunque no indiquemos nada en el docker compose, cuando lanzamos varios contenedores con docker-compose se crea una red nueva para dichos contenedores. Podemos comprobarlo cuando lanzamos el comando docker-compose up -d, si nos fijamos en el output vemos que se crea una red:

```
x markel in PF_Docker/P0_DockerPraktikak_2021-main/ > vim docker-compose.yml
✓ markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker compose up -d
[+] Running 4/4
  :: Network po_dockerpraktikak_2021-main_default   Created
```

Development workflow

Hacemos un curl a la ruta /hello y vemos que no obtenemos respuesta porque esta ruta no está definida en la aplicación de flask.

```
✓ markel in PF_Docker/P0_DockerPraktikak_2021-main/ > curl localhost:5000/hello
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.</p>
```

- **Prueba a ejecutar \$ curl 127.0.0.1:5000/debug ¿Qué devuelve?**

```
✓ markel in PF_Docker/P0_DockerPraktikak_2021-main/ > curl localhost:5000/debug
{"msg": "yellow open sfdata odWLCG9fQyyLJoZAjNOb4Q 5 1 629 0 1.2mb 1.2mb\n", "status": "success"}
```

Como la ruta /debug si está definida en nuestro app.py obtenemos la respuesta definida por la aplicación.

- **En este contexto, ¿cómo podemos añadir la ruta hello? Pues cambiando el código en app.py.**

Si cambiamos nuestro código en la aplicación app.py y añadimos el de la práctica podremos obtener una respuesta llamando a la ruta /hello.

```
7 @app.route('/')
6 def index():
5     return render_template('index.html')
4
3 @app.route('/hello')
2 def hello():
1     return "hello world!"
0
1 @app.route('/debug')
2 def test_es():
3     resp = []
4     try:
5         msg = es.cat.indices()
6         resp["msg"] = msg
```

Una vez añadido guardamos los cambios.

- **Abre app.py con tu editor favorito y realiza este cambio: Ahora prueba de nuevo**

\$ curl 127.0.0.1:5000/hello

- **¿Qué devuelve? ¿Se ha solucionado?**

Sigue devolviendo que no encuentra esa ruta, ya que el contenedor ni la aplicación se han reiniciado para cargar el nuevo código que hemos añadido.

- **¿Dónde has hecho el cambio? ¿En el host machine?**

Si, en el host machine

- **¿El contenedor qué app.py está ejecutando? ¿Se ha actualizado?**

Si, se esta ejecutando pero no se ha actualizado.

- **Comprueba entrando en el contenedor ejecutando el comando bash**

\$docker-compose run web bash
Edita app.py y comprueba que version es.

```
x markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker-compose run web bash
Creating po_dockerpraktikak_2021-main_web_run ... done
root@7e56cccd99578:/opt/flask-app# cat app.py | grep hello
@app.route('/hello')
def hello():
    return "hello world!"
root@7e56cccd99578:/opt/flask-app#
```

Como podemos ver, si nos conectamos al contenedor y miramos si el fichero app.py esta actualizado este contiene el código que hemos añadido en el host. Esto se debe a que el código lo hemos montado dentro del contenedor a través de un volumen, de esta manera, todo lo que modifiquemos en el host se modificará en el contenedor y viceversa.

Aunque hayamos realizado los cambios, la aplicación no se ha reiniciado en ningún momento, por lo tanto el código está presente pero aun no lo está corriendo debido a que el contenedor ni la aplicación se han reiniciado.

```
5      volumes:
6        - esdata1:/usr/share/elasticsearch/data
7
8  web:
9    image: markel149/imagenflask
10   command: python3 app.py
11   environment:
12     - DEBUG=true
13
14 depends_on:
```

- Para los contenedores mediante el comando docker-compose

```
✓ markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker compose down -v
[+] Running 6/4
  :: Container po_dockerpraktikak_2021-main_web_1           Removed
  :: Container po_dockerpraktikak_2021-main_web_run_7ea73e489a67 Removed
  :: Container po_dockerpraktikak_2021-main_web_run_dc16a1232540 Removed
  :: Container es                                         Removed
  :: Volume po_dockerpraktikak_2021-main_esdata1           Removed
  :: Network po_dockerpraktikak_2021-main_default          Removed
```

- Lanza de nuevo

```
✓ markel in PF_Docker/P0_DockerPraktikak_2021-main/ > docker-compose up -d
Creating network "po_dockerpraktikak_2021-main_default" with the default driver
Creating volume "po_dockerpraktikak_2021-main_esdata1" with local driver
Creating es ... done
Creating po_dockerpraktikak_2021-main_web_1 ... done
✓ markel in PF_Docker/P0_DockerPraktikak_2021-main/ >
```

- Ahora actualiza de nuevo el fichero app.py añadiendo la nueva ruta (/hello) y prueba accediendo a esta ruta. ¿Qué obtienes?

```
✓ markel in PF_Docker/P0_DockerPraktikak_2021-main/ > curl localhost:5000/hello
hello world!%
✓ markel in PF_Docker/P0_DockerPraktikak_2021-main/ >
```

Ahora una vez reiniciado el contenedor vemos que el código se a cargado correctamente en la aplicación y que esta nos responde de la manera esperada.