nginx

¿Qué es?

Según la Wikipedia:

> **Nginx** (pronunciado en inglés «ényin-ex», / nd n- ks/) es un servidor web/proxy inverso ligero de alto rendimiento y un proxy para protocolos de correo electrónico (IMAP/POP3).
>
> Es software libre y de código abierto, licenciado bajo la Licencia BSD simplificada; también existe una versión comercial distribuida bajo el nombre de Nginx Plus. Es multiplataforma, por lo que corre en sistemas tipo Unix (GNU/Linux, BSD, Solaris, Mac OS X, etc.) y Windows.
>
> El sistema es usado por una larga lista de sitios web conocidos, como: WordPress, Netflix, Hulu, GitHub, Ohloh, SourceForge, TorrentReactor y partes de Facebook (como el servidor de descarga de archivos zip pesados).

Instalación

Ahora que ya sabemos que es un servidor web muy usado, vamos a instalarlo en nuestro equipo. Por ejemplo, siguiendo la documentación oficial en la siguiente dirección.

Parece complicado, ¿no?

Si la respuesta a la pregunta es afirmativa, verás porqué es tan estupendo docker. No te tienes que preocupar de instalar `Nginx`: lo hace él solo.

Sólo tienes que buscar la imagen en DockerHub

Y ejecutar la imagen llamada `nginx`, lo que hará que la descargue y la guarde en la caché de docker (las siguientes veces ya no la descarga)

```
docker run --rm -d -p 8080:80 --name web nginx
```

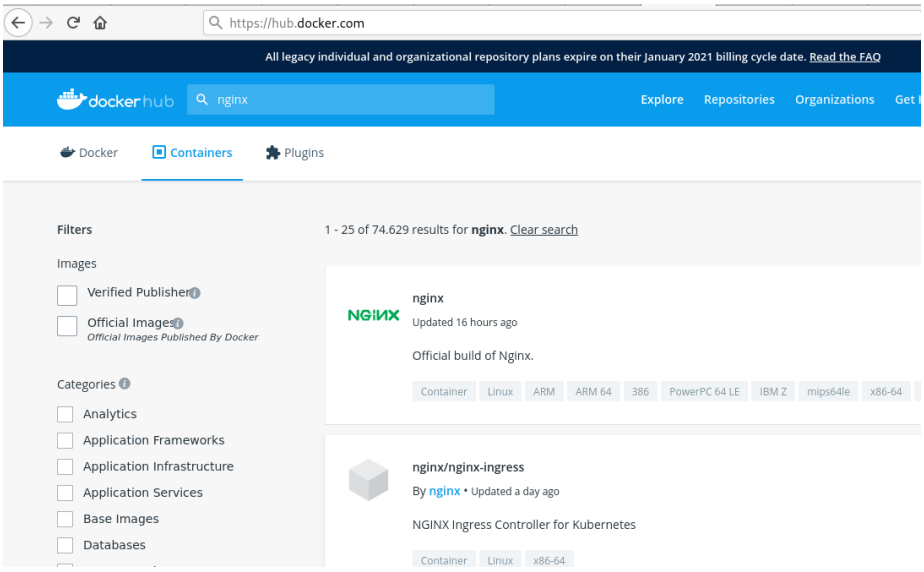

Figure 1: DockerHub

Con el comando anterior, comienza la ejecución del contenedor como un demonio (`-d`) y publica el puerto `8080` en la red del host. También llamó `web` al contenedor usando la opción `--name`.

Ya sólo tienes que visitar http://localhost:8080 y a funcionar!

Lo que vemos es la página por defecto de `Nginx`. Pero ahora vamos a servir una hecha por nosotros.

Primero paramos el contenedor, llamado `web`

```
docker stop web
```

Agregar HTML personalizado

De forma predeterminada, Nginx busca en el directorio `/usr/share/nginx/html` dentro del contenedor los archivos para servir. Necesitamos poner nuestros archivos html en este directorio. Una forma bastante sencilla de hacer esto es usar un volumen montado. Con volúmenes montados, podemos vincular un directorio en nuestra máquina local y asignar ese directorio a nuestro contenedor en ejecución.

Creamos una página html personalizada y luego la publicamos usando la imagen Nginx.

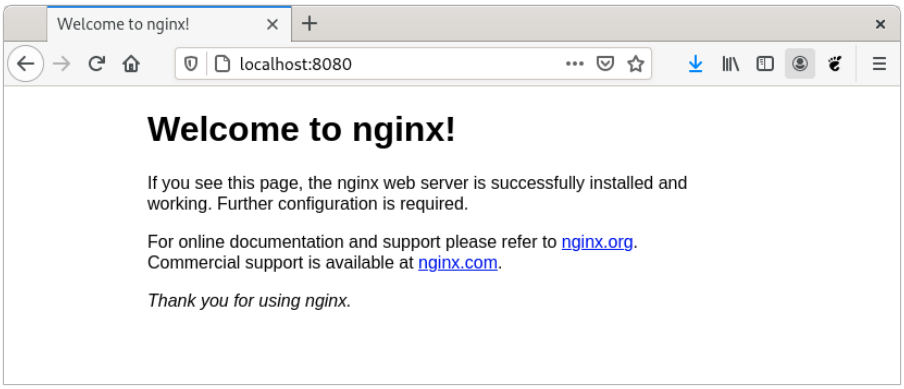

Figure 2: nginx

Crea un directorio en `Documentos` llamado `nginx` y dentro otro directorio llamado `site-content`. En este directorio, agrega un archivo `index.html` e introduce el siguiente html:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Docker Nginx</title>
</head>
<body>
  <h2>Hello desde Cibeseguridad</h2>
</body>
</html>
```

Ahora ejecuta el siguiente comando, que es el mismo que el anterior, pero ahora agregamos la marca `-v` para crear un volumen. Esto montará nuestro directorio local `~/Documentos/nginx/site-content` localmente en el contenedor en ejecución en: `/usr/share/nginx/html`

```
docker run --rm -d -p 8080:80 --name web -v ~/Documentos/nginx/site-content:/usr/share/nginx
```

Y este es el resultado

Crear una imagen personalizada

Los volúmenes son una excelente opción para ejecutar localmente y compartir archivos en un contenedor en ejecución. Pero, ¿qué pasa si queremos mover esta imagen y que nuestros archivos html se muevan con ella?

Figure 3: HTML Personalizado

Hay un par de opciones disponibles, pero una de las formas más portátiles y sencillas de hacer esto es copiar nuestros archivos html en la imagen creando una imagen personalizada.

Para crear una imagen personalizada, necesitaremos crear un `Dockerfile` y agregarle nuestros comandos.

En el mismo directorio, crea un archivo llamado Dockerfile y pega los siguientes comandos.

```
FROM nginx:latest
COPY ./site-content/index.html /usr/share/nginx/html/index.html
```

Comenzamos a construir nuestra imagen personalizada usando una imagen base. En la línea 1, puedes ver que hacemos esto usando el comando `FROM`. Esto extraerá la última imagen de `nginx` a nuestra máquina local y luego construirá nuestra imagen personalizada encima de ella.

A continuación, copiamos nuestro archivo `index.html` en el directorio /usr/share/nginx/html dentro del contenedor, sobrescribiendo el archivo `index.html` predeterminado proporcionado por nginx.

Notarás que no agregamos un `ENTRYPOINT` o un `CMD` a nuestro `Dockerfile`. Usaremos el `ENTRYPOINT` y el `CMD` subyacentes proporcionados por la imagen nginx.

Para construir nuestra imagen, ejecuta el siguiente comando:

```
docker build -t webserver .`
```

El comando de compilación le dirá a Docker que ejecute los comandos ubicados en nuestro Dockerfile. Verás una salida similar en tu terminal a la siguiente:

```
Sending build context to Docker daemon  3.072kB
Step 1/2 : FROM nginx:latest
latest: Pulling from library/nginx
bf5952930446: Pull complete
ba755a256dfe: Pull complete
c57dd87d0b93: Pull complete
d7fbf29df889: Pull complete
1f1070938ccd: Pull complete
Digest: sha256:36b74457bccb56fbf8b05f79c85569501b721d4db813b684391d63e02287c0b2
Status: Downloaded newer image for nginx:latest
 ---> 08393e824c32
Step 2/2 : COPY ./index.html /usr/share/nginx/html/index.html
 ---> fe915b3e4179
Successfully built fe915b3e4179
Successfully tagged webserver:latest
```

Figure 4: Build

Ahora podemos ejecutar nuestra imagen en un contenedor, pero esta vez no tenemos que crear un volumen para incluir nuestro html pues ya lo hacemos en el Dockerfile al copiar el archivo.

```
docker run --rm -d -p 8080:80 --name web webserver
```

Figure 5: Dockerfile

Basado en

https://www.docker.com/blog/how-to-use-the-official-nginx-docker-image/