
Caché con Nginx para los servicios de imágenes de Docker

**Descripción**

# Introducción

Buenas colegas, ahora que estoy empezando con toda la migración de mi infraestructura a Docker Swarm me he encontrado que cada vez que levanto un servicio en un nodo diferente hay que esperar a que la imagen del servicio sea descargada de alguno de los registros de internet.

Pues ahora me di a la tarea de por qué no cachear todas esas peticiones de manera local en mi red, así eliminaba la necesidad de usar el canal de internet para volver a bajar un recurso que ya se encontraba local, amén de la cantidad de banda ancha que tengamos.

Existen varios programas para hacer caché de paquetería de repositorios que soportaban ser de intermediarios con los registros externos de imágenes de Docker, como Nexus Artifactory o el propio Registry de Docker, pero me encontraba con un detalle, cada vez que pedía una imagen, hasta que estos no lo tenían en su caché no lo enviaban al cliente, por consecuente, te quedabas como desesperado al no poder ver el progreso de descarga de las imágenes.

Por dicho detalle me decidí a usar a Nginx, que por defecto tiene soporte para realizar caché local a las peticiones, sería configurarlo para que funcionara como proxy reverso hacia algún servicio en internet, copiando localmente los blob de las imágenes.

# Por fin… manos a la obra

Aunque como tal la configuración de nginx no depende del Sistema Operativo, los comandos los basaré en Debian, pues es lo que actualmente uso.

Lo primero que tenemos que hacer es instalar nginx

```
$ apt install nginx
```

Ahora solo es configurarle el sitio. Un detalle en este apartado, aunque lo podremos configurar completamente en su fichero nginx.conf seguiré las pautas de Debian y crearé su sitio en /etc/nginx/sites-enabled/

Ahora borraremos el sitio que viene por defecto

```
$ rm /etc/nginx/sites-enabled/default
```

y creamos nuestro sitio /etc/nginx/sites-enabled/docker con algun editor de texto

```
$ nano /etc/nginx/sites-enabled/docker
```

le ponemos este contenido (que explicaré más adelante)

```
proxy_cache_path /var/cache/nginx levels=1:2 keys_zone=docker-proxy-cache:100m max_size=100g inactive=876
```

```
upstream registry-docker {
        server repo.docker.ir:5000;
        server havanacontainers.com:80;
}

server {
  listen          0.0.0.0:80;
  server_name     docker.mtz.desoft.cu;

  proxy_buffering         off;

  client_max_body_size 10G;

  access_log      /var/log/nginx/docker-access.log;
  error_log       /var/log/nginx/docker-error.log;

  proxy_intercept_errors off;
  proxy_send_timeout 120;
  proxy_read_timeout 300;

  set $no_cache 0;

  if ($request_uri ~* "/(manifests/)" )
  {
    set $no_cache 1;
  }

  location / {
    proxy_pass  http://registry-docker;

    proxy_cache_bypass $no_cache;  # Don't pull from cache based on $no_cache
    proxy_no_cache $no_cache; # Don't save to cache based on $no_cache

    proxy_cache_revalidate on;
    proxy_cache_valid 200 365d;
```

```
    proxy_cache_use_stale error timeout updating http_500 http_502 http_503 http_504;
    proxy_hide_header      Set-Cookie;
    proxy_ignore_headers Set-Cookie X-Accel-Expires Expires Cache-Control;
    proxy_buffering        on;
    proxy_cache docker-proxy-cache;

    proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504;

    ### Set headers ####
    proxy_set_header        Accept-Encoding   "";
    proxy_set_header        Host            $host;

    proxy_set_header        X-Forwarded-Proto $scheme;
    add_header              Front-End-Https   on;
    add_header X-Cache-Status $upstream_cache_status;

    proxy_redirect     off;
  }
}
```

Explicando algunos fragmentos:

```
proxy_cache_path /var/cache/nginx levels=1:2 keys_zone=docker-proxy-cache:100m max_size=100g inactive=876
```

Con esa línea, se configura la caché, se dice que estará en la carpeta /var/cache/nginx , que tendrá 2 niveles de carpetas y que ocupará un máximo de 100 Gb y que los datos dentro pueden estar inactivos por 8760 horas.

```
  set $no_cache 0;

  if ($request_uri ~* "/(manifests/)" )
  {
    set $no_cache 1;
  }
```

Aquí se establece una variable para definir si se cachea o no, en este caso es para no cachear las peticiones que contengan manifests debido que es la que contiene la información de las capas. Aquí no es aconsejable cachearlas porque sino, por ejemplo, una imagen de

tipo latest no se actualizaría nunca mientra dure la cache.

```
proxy_cache_bypass $no_cache;
proxy_no_cache $no_cache;
```

En dependencia de lo anterior, cachea o no.

# Finalizando

Ahora solo tenemos que reiniciar el servicio de nginx y tendremos nuestro servicio de caché de imágenes

```
$ systemctl restart nginx
```

## Configurando Docker

Ahora solo bastaría configurar Docker para que haga uso de este registro, aunque podemos descargar directemente de él:

```
$ docker pull cache_server:80/prom/prometheus
```

pero lo mejor es usarlo directamente, para eso vea el post Docker desde paises bloqueados en la última parte. Otro de los detalles es aumentar la seguridad del proxy usando SSL.

```
$ nano /etc/nginx/sites-enabled/docker
```

Y agregamos estas lineas

```
# <- your SSL Settings here!
        ssl_certificate /etc/ssl/server_crt.pem;
        ssl_certificate_key /etc/ssl/server_key.pem;
        ssl_trusted_certificate /etc/ssl/server_crt.pem;
        ssl_session_timeout 1d;
        ssl_session_cache shared:SSL:50m;
```

```
ssl_session_tickets off;
```

Espero les sirva. Esperen mas tutoriales mios.

**Categoría**

1. Como se hace
2. Docker
3. Redes
4. Servidores
5. Software Libre

**Fecha de creación**
agosto 2018
**Autor**
h3r3t1c