Traefik

> **Aviso**: en este enunciado se usaba como TLD para los ejemplos `.local`. No es uno de los dominios recomendados para los *Private DNS Namespaces*. En su lugar debemos usar alguno de los recomendados:
>
> ```
> .intranet.
> .internal.
> .private.
> .corp.
> .home.
> .lan.
> ```
>
> Se ha modificado la práctica original para usar el TLD `.internal`

1. Introducción

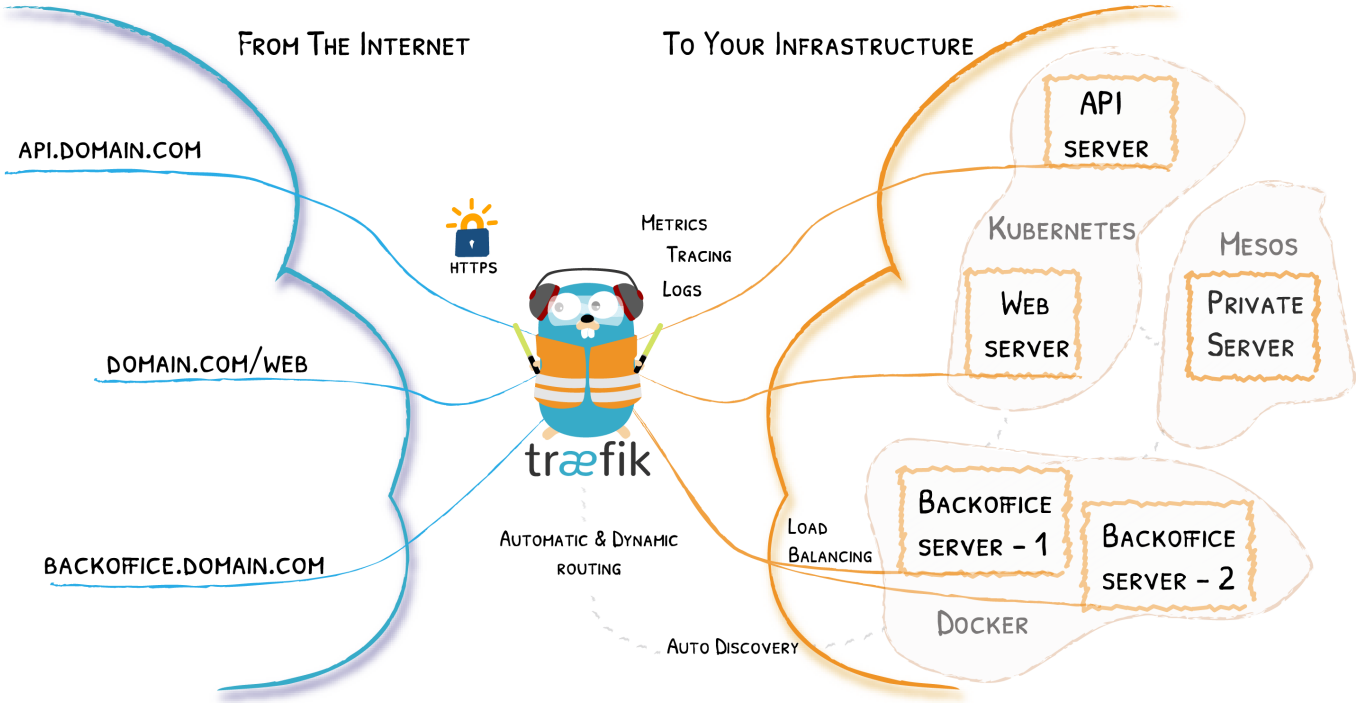
Un proxy inverso es un servidor que se sitúa delante de uno o varios servidores web, e intercepta las solicitudes de los clientes. Esto es diferente de un proxy de reenvío, en el que el proxy se sitúa delante de los clientes. Con un proxy inverso, cuando los clientes envían solicitudes al servidor de origen de un sitio web, el servidor de proxy inverso intercepta esas solicitudes en el perímetro de la red. El servidor proxy inverso enviará entonces las solicitudes al servidor de origen y recibirá las respuestas del servidor de origen.

La diferencia entre un proxy de reenvío y uno inverso es sutil pero importante. Una forma simplificada de resumirla sería decir que un proxy de reenvío se sitúa delante de un cliente y se asegura de que ningún servidor de origen se comunique nunca directamente con ese cliente específico. Por otro lado, un proxy inverso se sitúa delante de un servidor de origen y se asegura de que ningún cliente se comunique nunca directamente con ese servidor de origen.

1.1. Traefik como proxy inveso

> Nginx o Apache pueden cumplir perfectamente esta función, aunque hoy en día hay una nueva categoria de proxys reversos pensados para microservicios y también diseñados para “dialogar” con plataformas de contenedores como Docker, Kubernetes, etc. Este es el caso de Traefik.

Traefik is an open-source Edge Router that makes publishing your services a fun and easy experience. It receives requests on behalf of your system and finds out which components are responsible for handling them.



What sets Traefik apart, besides its many features, is that it automatically discovers the right configuration for your services. The magic happens when Traefik inspects your infrastructure, where it finds relevant information and discovers which service serves which request.

The main features include dynamic configuration, automatic service discovery, and support for multiple backends and protocols.

Traefik is based on the concept of EntryPoints, Routers, Middlewares and Services.

- EntryPoints: EntryPoints are the network entry points into Traefik. They define the port which will receive the packets, and whether to listen for TCP or UDP.
- Routers: A router is in charge of connecting incoming requests to the services that can handle them.
- Middlewares: Attached to the routers, middlewares can modify the requests or responses before they are sent to your service
- Services: Services are responsible for configuring how to reach the actual services that will eventually handle the incoming requests.

1.1.1. Edge Router¶

Traefik is an Edge Router, it means that it's the door to your platform, and that it intercepts and routes every incoming request: it knows all the logic and every rule that determine which services handle which requests (based on the path, the host, headers, etc.).

1.1.2. Auto Service Discovery¶

Where traditionally edge routers (or reverse proxies) need a configuration file that contains every possible route to your services, Traefik gets them from the services themselves.

Deploying your services, you attach information that tells Traefik the characteristics of the requests the services can handle.

2. Iniciamos Traefik

Comenzamos con el ejemplo que nos proporciona Traefik: https://doc.traefik.io/traefik/getting-started/quick-start/. el fichero inicial sólo contiene el servicio `reverse-proxy` que lanza Traefik

```
version: '3'

services:
  reverse-proxy:
    # The official v2 Traefik docker image
    image: traefik:v2.10
    # Enables the web UI and tells Traefik to listen to docker
    command:
      - --api.insecure=true
      - --providers.docker
      - --providers.docker.exposedbydefault=false
    ports:
      # The HTTP port
      - "80:80"
      # The Web UI (enabled by --api.insecure=true)
      - "8080:8080"
    volumes:
      # So that Traefik can listen to the Docker events
      - /var/run/docker.sock:/var/run/docker.sock
```

Iniciamos el servicio con el comando `docker-compose` especificando el servicio que queremos activar:

```
clases@vm:~/daweb/p05_traefik$ docker compose up -d reverse-proxy
[+] Running 2/2
...

0
clases@vm:~/daweb/p05_traefik$
```

Y podemos acceder al tablero (*dashboard*) de Traefik en la url `http://localhost:8080` o a la API (`/api/rawdata`) :

```

```

```
clases@vm:~/daweb/p05_traefik$ firefox http://localhost:8080/api/rawdata
clases@vm:~/daweb/p05_traefik$ firefox http://localhost:8080
```

3. Descubriendo servicios

3.1. Ejemplo inicial

La guía rápida nos propone añadir un servicio web de nombre *whoami* que muestra información del equipo sobre el que se ejecuta ese servicio. El bloque a añadir en `docker-compose.yml`. Hemos modificado levemente el fichero para usar el dominio `midominio.internal`:

```
  whoami:
    # A container that exposes an API to show its IP address
    image: traefik/whoami
    labels:
      - "traefik.enable=true" # necesario si  --
providers.docker.exposedbydefault=false
      - "traefik.http.routers.whoami.entrypoints=http" # default
      -
"traefik.http.routers.whoami.rule=Host(`whoami.midominio.internal`)"
```

Para activar dicho servicio ejecutamos

```
clases@vm:~/daweb/p05_traefik$ docker compose up -d whoami
[+] Running 1/0
 ✔ Container p05_traefik-whoami-1  Running
0.0s
clases@vm:~/daweb/p05_traefik$
```

Traefik descubre este nuevo servicio y lo añade a su configuración creando una ruta que enlaza la dirección `whoami.midominio.internal` a ese servicio. Para probarlo con `curl` debemos usar la opción `-H Host:whoami.midominio.internal`

```
clases@vm:~/daweb/p05_traefik$ curl -H "Host:whoami.midominio.internal"
localhost
Hostname: 5d2d95b8f9b3
IP: 127.0.0.1
IP: 172.19.0.3
...
clases@vm:~/daweb/p05_traefik$
```

Probamos el servicio con

- `curl`,
- el parámetro `-H "Host:whoami.midominio.internal"`
- y como destino `localhost`
  porque si intentamos acceder a la dirección whoami.midominio.com (en el navegador o haciendo ping) nos dará error ya que no existe ese dominio y no tiene ninguna IP asociada.

Para resolver la IP de un nombre de host se consulta primero al fichero `/etc/hosts` y luego a los servidores DNS. Si asociamos en ese fichero el nombre `whoami.midominio.internal` a la dirección local podremos usar ese nombre con el navegador, etc:

```
clases@vm:~/daweb/p05_traefik$ cat /etc/hosts
127.0.0.1       whoami.midominio.internal localhost
127.0.1.1       vm

...

clases@vm:~/daweb/p05_traefik$
```

Podemos probar tras modificar el fichero con `ping`

```
clases@vm:~/daweb/p05_traefik$ ping whoami.midominio.internal
PING whoami.midominio.internal (127.0.0.1) 56(84) bytes of data.
64 bytes from whoami.midominio.internal (127.0.0.1): icmp_seq=1 ttl=64
time=0.064 ms
64 bytes from whoami.midominio.internal (127.0.0.1): icmp_seq=2 ttl=64
time=0.088 ms
^C
--- whoami.midominio.internal ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1021ms
rtt min/avg/max/mdev = 0.064/0.076/0.088/0.012 ms
clases@vm:~/daweb/bloque_01_docker/p05_traefik$
clases@vm:~/daweb/p05_traefik$
```

Y ya podemos acceder a esa dirección desde el navegador web:

```
clases@vm:~/daweb/p05_traefik$ firefox whoami.midominio.internal &
[1] 9050
clases@vm:~/daweb/p05_traefik$
```

Vamos a añadir otras dos direcciones (que usaremos en otros apartados) al fichero `/etc/hosts`:

- web.midominio.internal y
- php.otrodominio.internal

```
```

```
clases@vm:~/daweb/p05_traefik$ cat /etc/hosts
127.0.0.1       php.otrodominio.internal web.midominio.internal
whoami.midominio.internal localhost
127.0.1.1       vm

....

clases@vm:~/daweb/p05_traefik$
```

3.2. Servicio nginx

Ahora vamos a añadir un servidor web nginx. El bloque a añadir en `docker-compose.yml` sería:

```
  nginx:
    image: nginx
    restart: unless-stopped
    volumes:
      - ./web:/usr/share/nginx/html
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.nginx.entrypoints=http"
      - "traefik.http.routers.nginx.rule=Host(`web.midominio.internal`)"
```

En este caso se asocia el host `web.midominio.internal` al servicio. Para activar dicho servicio ejecutamos

```
clases@vm:~/daweb/p05_traefik$ docker compose up -d nginx
[+] Running 1/1
 ✔ Container p05_traefik-nginx-1  Started
0.0s
clases@vm:~/daweb/p05_traefik$
```

Nuevamente Traefik descubre el servicio en tiempo real y lo añade a sus rutas. Ahora podemos usar directamente la dirección `web.midominio.internal` con curl o firefox

```
clases@vm:~/daweb/p05_traefik$ curl web.midominio.internal
<!DOCTYPE html>
....
<body>
    <p>Página inicio servidor nginx. </p>
</body>
</html>
clases@vm:~/daweb/p05_traefik$
```

3.3. Servidor php

Añadimos ahora un servidor web con PHP. En este caso el servidor mapea un directorio local al contenedor con el fichero `index.php` que muestra la IP y nombre del host.

El contenido del fichero `index.php` es

```
<?php echo 'Server IP: '.$_SERVER['SERVER_ADDR'].nl2br(" \n");?>
<?php echo 'Hostname: '.gethostname().nl2br(" \n");?>
```

Añadimos el bloque del nuevo servicio a `docker-compose.yml`. En este caso usaremos otro nombre de dominio

```
  phpserver:
    image: php:apache
    restart: unless-stopped
    volumes:
      - ./html:/var/www/html
    labels:
      - "traefik.enable=true"
      # - "traefik.http.routers.apache.entrypoints=http"
      - "traefik.http.routers.apache.rule=Host(`php.otrodominio.internal`)"
```

Y lanzamos ese servicio:

```
clases@vm:~/daweb/p05_traefik$ docker compose up -d phpserver
[+] Running 1/1
 ✔ Container p05_traefik-phpserver-1
Started                0.1s
clases@vm:~/daweb/p05_traefik$
```

Y probamos en consola y en navegador:

```
clases@vm:~/daweb/p05_traefik$ curl php.otrodominio.internal
Server IP: 172.20.0.5 <br />
Hostname: 93206dc48cea <br />
clases@vm:~/daweb/p05_traefik$ firefox php.otrodominio.internal
```

4. Escalamos el servicio: balanceo de carga

Podemos "*escalar*" este último servicio creando replicas del servicio apache usando la opción `--scale`:

```
clases@vm:~/daweb/p05_traefik$ docker compose up -d --scale phpserver=3
[+] Running 6/6
 ✔ Container p05_traefik-reverse-proxy-1
  Running                                 0.0s
 ✔ Container p05_traefik-nginx-1
Running                                   0.0s
 ✔ Container p05_traefik-whoami-1
 Running                                  0.0s
 ✔ Container p05_traefik-phpserver-1
 Running                                  0.0s
 ✔ Container p05_traefik-phpserver-3
 Started                                  0.0s
 ✔ Container p05_traefik-phpserver-2
 Started                                  0.1s
clases@vm:~/daweb/p05_traefik$
```

Si accedemos ahora al servicio podemos comrpobar cómo se va accediendo a los distintas replicas del servidor php ya que traefik hace balanceo de carga.

```
clases@vm:~/daweb/p05_traefik$ curl php.otrodominio.internal
Server IP: 172.20.0.5 <br />
Hostname: 93206dc48cea <br />
clases@vm:~/daweb/p05_traefik$ curl php.otrodominio.internal
Server IP: 172.20.0.6 <br />
Hostname: a23b9ac33c77 <br />
clases@vm:~/daweb/p05_traefik$ curl php.otrodominio.internal
Server IP: 172.20.0.7 <br />
Hostname: 8ecfc43015ab <br />
clases@vm:~/daweb/p05_traefik$ curl php.otrodominio.internal
Server IP: 172.20.0.5 <br />
Hostname: 93206dc48cea <br />
clases@vm:~/daweb/p05_traefik$
```

5. Ejercicio

Añadir a traefik un servicio wordpres en el dominio `miblog.example.internal`

6. Enlaces

- Quickstart de Traefik: https://doc.traefik.io/traefik/getting-started/quick-start/
- Traefik Docker Example and Introduction: https://www.middlewareinventory.com/blog/traefik-docker/?utm_source=pocket_reader

7. Anexo: Fichero `docker-compose.yml` completo

```
```

```
version: '3'

services:
  reverse-proxy:
    # The official v2 Traefik docker image
    image: traefik:v2.10
    # Enables the web UI and tells Traefik to listen to docker
    command:
      - --api.insecure=true
      - --providers.docker
      - --providers.docker.exposedbydefault=false
    ports:
      # The HTTP port
      - "80:80"
      # The Web UI (enabled by --api.insecure=true)
      - "8080:8080"
    volumes:
      # So that Traefik can listen to the Docker events
      - /var/run/docker.sock:/var/run/docker.sock

  whoami:
    # A container that exposes an API to show its IP address
    image: traefik/whoami
    labels:
      - "traefik.enable=true" # necesario si  --
providers.docker.exposedbydefault=false
      - "traefik.http.routers.whoami.entrypoints=http" # default
      -
"traefik.http.routers.whoami.rule=Host(`whoami.midominio.internal`)"

  nginx:
    image: nginx
    restart: unless-stopped
    volumes:
      - ./web:/usr/share/nginx/html
    labels:
      - "traefik.enable=true"
      - "traefik.http.routers.nginx.entrypoints=http"
      - "traefik.http.routers.nginx.rule=Host(`web.midominio.internal`)"

  phpserver:
    image: php:apache
    restart: unless-stopped
    volumes:
      - ./html:/var/www/html
    labels:
      - "traefik.enable=true"
      # - "traefik.http.routers.apache.entrypoints=http"
      - "traefik.http.routers.apache.rule=Host(`php.otrodominio.internal`)"
```

8. Dudas de clase

1. En la configuración inicial de esta práctica traefik expone todos los contenedores que se activen. Para evitarlo hay que añadir la opción `--providers.docker.exposedbydefault=false` en el arranque de traefik:

```
services:
  reverse-proxy:
    image: traefik:v2.10
    command:
      - --providers.docker.exposedbydefault=false
      - --api.insecure=true
      - --providers.docker
```

En este caso para activar un servicio y que traefik lo *descubra* es necesario añadir la etiqueta `traefik.enable=true` en el servicio:

```
  whoami:
    image: traefik/whoami
    labels:
      - "traefik.enable=true"
      -
"traefik.http.routers.whoami.rule=Host(`whoami.midominio.internal`)"
```