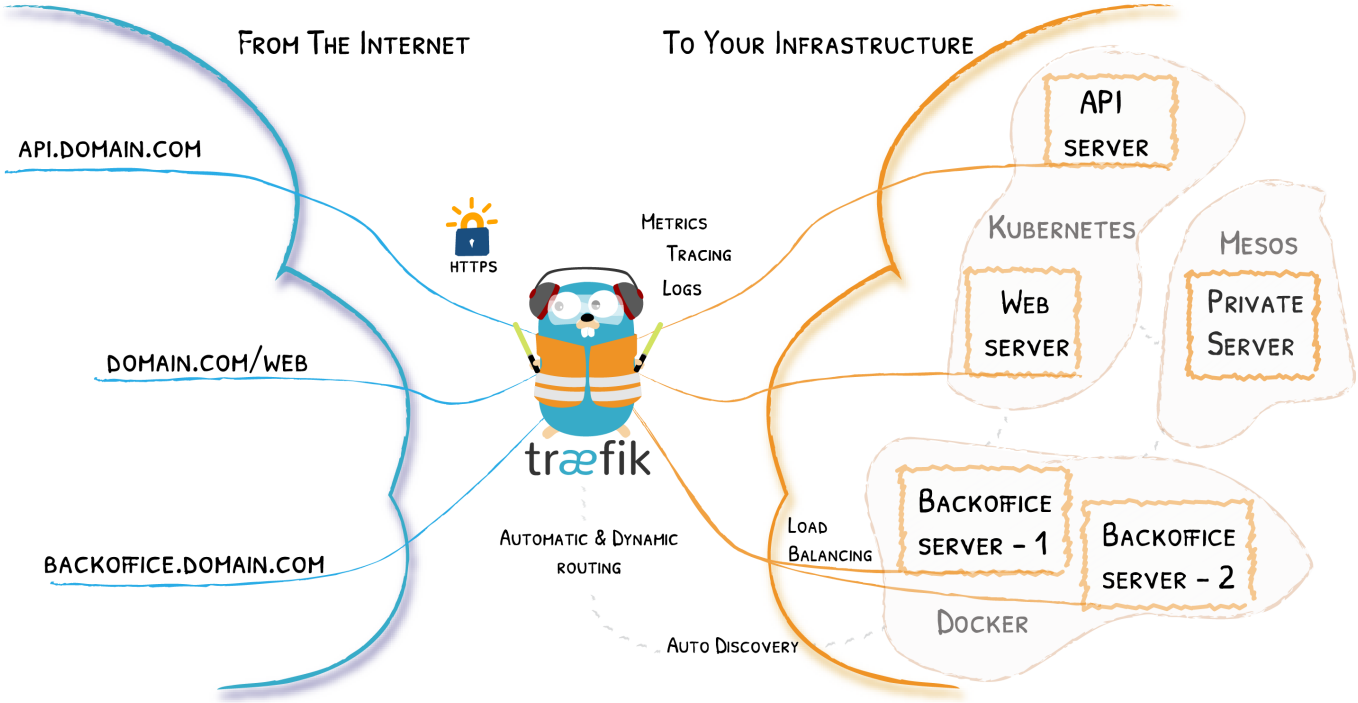
Actividad 6 Træfik

En esta práctica observaremos sus principales características de Træfik.



3.1. *Træfik* como proxy inverso

Vamos a lanzar varios servicios que serán accesibles a través de *Træfik*, un proxy inverso que crea los caminos de las webs de forma automática y dinámica, definidos con etiquetas en otros ficheros `docker-compose`. En este caso usaremos *Træfik* 1.7 (una versión ya antigua pero más clara).

Para arrancar, entramos en el directorio `traefik`:

```
docker network create traefik
SERV='localhost' docker-compose -f traefik.yml up -d --scale whoami=2
SERV='localhost' docker-compose -f docker-compose-wp.yml up -d
SERV='localhost' docker-compose -f docker-compose-nextcloud.yml up -d
curl -H Host:nextcloud.localhost http://nextcloud.localhost/ # si no accedemos por web
```

El fichero `traefik.yml` arranca el propio contenedor `traefik`, dos instancias de `whoami` que hacen balanceo de carga (responden alternativamente) con `--scale whoami=2`, y también arranca `Portainer`, un gestor gráfico de los contenedores.

El panel de control de *Træfik* es accesible en `http://localhost/traefik/`, el panel de información de *Portainer* en `http://portainer.localhost/` y las páginas de *whoami* que cargan alternativamente las de los dos contenedores en `http://localhost/whoami`

```yaml
# SERV='localhost' docker-compose -f traefik.yml up -d --scale whoami=2
# ver https://lsi.vc.ehu.eus/pablogn/docencia/ISO/Act9%20Docker/
# ver http://lsi.vc.ehu.eus/pablogn/docencia/AS/Act3%20Traefik/

version: '3.3'

services:
  traefik:
    image: traefik:v1.7-alpine
    container_name: "traefik"
    command: --api --docker --docker.domain=${SERV} --logLevel=DEBUG
    ports:
      - "80:80"
      - "8080:8080"
    labels:
      - "traefik.docker.network=frontend"
      - "traefik.enable=true"
      - "traefik.frontend.rule=Host:${SERV}; PathPrefixStrip:/traefik"
      - "traefik.port=8080"
      - "traefik.protocol=http"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - /dev/null:/traefik.toml
    networks:
      - "traefik"

  whoami:
    image: containous/whoami
    labels:
      - "traefik.enable=true"
      - "traefik.backend=whoami"
      - "traefik.frontend.rule=PathPrefixStrip: /whoami"
      - "traefik.http.routers.whoami.rule=Host(${SERV})"
      - "traefik.http.routers.whoami.entrypoints=web"
    networks:
      - "traefik"

  portainer: # https://www.smarthomebeginner.com/traefik-2-docker-tutorial/#
      Portainer_with_Traefik_2_and_OAuth
    image: portainer/portainer
    container_name: portainer
    restart: unless-stopped
    command: -H unix:///var/run/docker.sock
    networks:
      - traefik
      - default
    labels:
      - "traefik.enable=true"
      - "traefik.protocol=http"
      - "traefik.docker.network=frontend"
      - "traefik.frontend.rule=Host:portainer.${SERV}"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data

    security_opt:
      - no-new-privileges:true

volumes:
  portainer_data:

networks:
  traefik:
    external:
      name: traefik
```

Fichero 3.1: `traefik.yml`

El `docker-compose-wp.yml` lanza un contenedor Wordpress (o más réplicas) apoyado en un contenedor que gestiona la base de datos MySQL en la red `wp_back` y accesible en el camino `http://wp.localhost/`

```
# docker-compose up -d --scale wordpress=3 --scale db=2

version: '3'

services:
   wpdb:
     image: mysql:5.7
     volumes:
       - wpdb:/var/lib/mysql
     restart: unless-stopped
     labels:
       - traefik.enable=false
     networks:
       - wp_back
     environment:
       MYSQL_ROOT_PASSWORD: wordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - wpdb
     image: wordpress:latest
     labels:
      - "traefik.enable=true"
      - "traefik.port=80"
      - "traefik.backend=wordpress"
      - "traefik.frontend.rule=Host:wp.${SERV}"
  #    - "traefik.frontend.rule=Host:${SERV}; PathPrefix:/wp"
  #    - "traefik.frontend.rule=PathPrefixStrip:/wp"
      - "traefik.docker.network=traefik"
     restart: unless-stopped
 #    ports:
 #      - "80:80"
     networks:
      - traefik
      - wp_back
     environment:
       WORDPRESS_DB_HOST: wpdb:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
     volumes:
       - wp:/var/www/html
     deploy:
       mode: replicated
       replicas: 2
       restart_policy:
         condition: always
       labels:
         APP: WORDPRESS

volumes:
    wpdb:
    wp:

networks:
  wp_back:
    driver: bridge
  traefik:
    external:
      name: traefik

# docker-compose scale wordpress=3
```

Fichero 3.2: `docker-compose-wp.yml`

El `docker-compose-nextcloud.yml` lanza un contenedor con la nube privada, apoyado en un contenedor que gestiona otra base de datos MariaDB (que también usa el mismo puerto pero está en una red distinta `nc_back`) y accesible en el camino `http://nextcloud.localhost/`

```
### https://github.com/nextcloud/docker
## https://blog.ssdnodes.com/blog/installing-nextcloud-docker/
# https://docs.nextcloud.com/server/16/admin_manual/configuration_server/
    config_sample_php_parameters.html

# SERV='localhost' docker-compose -f docker-compose-nextcloud.yml up -d
# SERV='localhost' docker-compose -f docker-compose-nextcloud.yml ps
# SERV='localhost' docker-compose -f docker-compose-nextcloud.yml logs nextcloud
# SERV='localhost' docker-compose -f docker-compose-nextcloud.yml exec nextcloud bash

version: '2'

volumes:
  nextcloud:
  ncdb:

services:
  ncdb:
    image: mariadb
    container_name: db
    command: --transaction-isolation=READ-COMMITTED --binlog-format=ROW
    restart: unless-stopped
    labels:
       - traefik.enable=false
       - traefik.docker.network=traefik
    volumes:
      - ncdb:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=Contrasenna
      - MYSQL_PASSWORD=Contrasenna
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud
    networks:
       - nc_back

  nextcloud:
    image: nextcloud
    container_name: nextcloud
    depends_on:
      - ncdb
    volumes:
      - nextcloud:/var/www/html
    labels:
      - "traefik.enable=true"
      - "traefik.port=80"
      - "traefik.backend=nextcloud"
#       - "traefik.frontend.rule=Host:${SERV}; PathPrefixStrip:/nextcloud"
      - "traefik.frontend.rule=Host:nextcloud.${SERV}"
      - "traefik.docker.network=traefik"
    environment:
      - NEXTCLOUD_ADMIN_USER=Pablo
      - NEXTCLOUD_ADMIN_PASSWORD=Contrasenna
      - SMTP_HOST=smtp.ehu.eus
      - LC_ALL=C.UTF-8
      - TZ=Europe/Madrid
    restart: unless-stopped
    networks:
      - traefik
      - nc_back
networks:
  nc_back:
    driver: bridge
  traefik:
    external:
      name: traefik
```

Fichero 3.3: `docker-compose-nextcloud.yml`

Podemos ver la actividad de uno de los contenedores que forman el servicio:

```
SERV='localhost' docker-compose -f docker-compose-nextcloud.yml logs nextcloud
```

En esta captura vemos qué se ejecuta en el contenedor de Nextcloud:

```
$ SERV='localhost' docker-compose -f docker-compose-nextcloud.yml exec nextcloud bash
root@4134890375cd:/var/www/html# ps aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.4 280804 36112 ?        Ss   17:13   0:00 apache2 -DFOREGROUND
www-data    112  0.0  0.3 281568 28032 ?        S    17:13   0:00 apache2 -DFOREGROUND
www-data    113  0.0  0.1 280836  9156 ?        S    17:13   0:00 apache2 -DFOREGROUND
www-data    114  0.0  0.1 280836  9156 ?        S    17:13   0:00 apache2 -DFOREGROUND
www-data    115  0.0  0.1 280836  9156 ?        S    17:13   0:00 apache2 -DFOREGROUND
www-data    116  0.0  0.1 280836  9156 ?        S    17:13   0:00 apache2 -DFOREGROUND
www-data    117  0.0  0.1 280836  9160 ?        S    17:14   0:00 apache2 -DFOREGROUND
root        118 13.3  0.0   5620  3540 pts/0    Ss   18:07   0:00 bash
root        124  0.0  0.0   9392  3108 pts/0    R+   18:07   0:00 ps aux
root@4134890375cd:/var/www/html#
```

A continuación, el resultado de `history` después de hacer el laboratorio:

```
1937  ls
1938  cd Traefik/
1939  ls
1943  docker network create traefik
1944  SERV='localhost' docker-compose -f traefik.yml up -d --scale whoami=2
1945  docker-compose -f traefik.yml ps
1946  curl localhost/whoami
1947  curl localhost/whoami1947  docker-compose -f traefik.yml ps
1948  ls
1950  SERV='localhost' docker-compose -f docker-compose-wp.yml up -d
1953  curl -H Host:portainer.localhost http://127.0.0.1
1954  curl -H Host:wp.localhost http://wp.localhost/
1955  docker-compose -f docker-compose-wp.yml logs wordpress
1956  SERV='localhost' docker-compose -f docker-compose-nextcloud.yml up -d
1958  docker-compose -f docker-compose-nextcloud.yml logs nextcloud
1959  docker-compose -f docker-compose-nextcloud.yml logs db
1965  curl -H Host:nextcloud.localhost http://nextcloud.localhost/
1966  docker ps -a
1967  SERV='localhost' docker-compose -f docker-compose-nextcloud.yml logs nextcloud
1968  SERV='localhost' docker-compose -f docker-compose-nextcloud.yml exec netxcloud bash
1973  docker-compose -f traefik.yml down
1976  docker-compose -f docker-compose-nextcloud.yml down
1977  docker-compose -f docker-compose-wp.yml down
1979  docker volume prune
1980  docker volume ls
1983  docker ps -a
1984  docker network ls
1985  docker network rm traefik
```

Fichero 3.4: `history`

3.2. Vagrant con `docker-compose`

Vamos a lanzar exactamente lo mismo, pero desde Vagrant.

Para el siguiente sistema no es necesario librar puertos, ya que por si acaso están usándose puertos más allá del 1024 para no necesitar permisos de `root`. Sin embargo, conviene borrar los contenedores actuales para mantener limpio el sistema, como se hace a partir de la línea 1973 del `history`.

Se puede combinar el uso de Máquinas Virtuales automatizadas con Vagrant para lanzar diferentes grupos de contenedores con `docker-compose` como se ve en el ejemplo de la carpeta *Vagrant*.

En el `Vagrantfile` se define un programa `bash` (llamados `scripts`) que instalará el software `docker-compose`, creará una red definida por software (SDN) llamada `traefik` y lanzará los tres ficheros `docker-compose`.

Después de configurar Virtualbox se define la máquina virtual con la imagen Bionic, IP por DHCP, abre puertos, *provisiona* (instala) Docker, copia los tres ficheros `docker-compose` del apartado anterior y ejecuta el `script` (en el último comentario del fichero está la alternativa sin script, usando la instrucción `inline`).

Tras ejecutar el `vagrant up`, como comprobación deberíamos acceder desde la línea de comandos con esta instruccióna Nextcloud y a Træfik:

```
curl -H Host:nextcloud.mv http://nextcloud.localhost:8000/
curl -H Host:mv http://nextcloud.localhost:8000/traefik/
```

Quedaría configurar Landrush (con `vagrant plugin install landrush`) para crear dominios dinámicos para las máquinas virtuales (necesita `root`), y habilitarlo en el `Vagrantfile` en la línea:

```
m.landrush.enabled = true # vagrant plugin install landrush
```

Éste es el `Vagrantfile`:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# This script to install docker-compose will get executed after we have provisioned the box
$script = <<-SCRIPT
curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-`uname -s`-`uname
    -m` -o /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
# docker-compose up -d
docker network create traefik
SERV='mv' docker-compose -f traefik.yml up -d --scale whoami=2
SERV='mv' docker-compose -f docker-compose-wp.yml up -d # --scale wordpress=3 --scale db=2
SERV='mv' docker-compose -f docker-compose-nextcloud.yml up -d
SCRIPT

Vagrant.configure("2") do |config|
    config.vm.provider :virtualbox do |v|
        # On VirtualBox, we don't have guest additions or a functional vboxsf
        # in CoreOS, so tell Vagrant that so it can be smarter.
        v.check_guest_additions = false
        v.functional_vboxsf     = false
  v.customize ["modifyvm", :id, "--audio", "none"]
   end
end

Vagrant.configure("2") do |config|
    config.vm.define vm_name = "mv" do |m|
        m.vm.box = "ubuntu/bionic64"
 #       m.landrush.enabled = true #vagrant plugin install landrush
#        m.vm.hostname = "myhost.vagrant.test"
 #       m.cache.scope = :box   # vagrant plugin install vagrant-cachier
        m.vm.hostname = vm_name
        ip = "172.21.12.66"
        m.vm.network :private_network, ip: ip
      # m.vm.network "private_network", type: "dhcp"
        m.vm.network "forwarded_port", guest: 80, host: 8000 # <1024 necesita root
```

```
  #       m.vm.network "forwarded_port", guest: 8080, host: 8880
        m.vm.provision "docker"
        m.vm.provision "file", source: "./docker-compose-wp.yml", destination: "./docker-compose-wp.
            yml"
        m.vm.provision "file", source: "./docker-compose-nextcloud.yml", destination: "./docker-
            compose-nextcloud.yml"
        m.vm.provision "file", source: "./traefik.yml", destination: "./traefik.yml"
        m.vm.provision "shell", inline: $script
#         m.vm.provision "shell",
#inline: "DEBIAN_FRONTEND=noninteractive SERV='mv' docker-compose -f traefik.yml up -d --scale
    whoami=2 && SERV='mv' docker-compose -f docker-compose-wp.yml -d && SERV='mv' docker-compose -f
    docker-compose-nextcloud.yml -d
    end
end
```

Fichero 3.5: `Vagrantfile`