Manual de instalación y despliegue

Alvaro Garcia Sanabria

1 Instalación del sistema operativo anfitrión

Dado que la instalación del sistema operativo elegido en el diseño, Ubuntu Linux 22.04 LTS, está muy guiada por asistente y el proceso debiera ser *más que familiar* para cualquier estudiante o profesional de la disciplina informática por, entre otros, su popularidad y similitud con el de la distribución de Linux Debian, se procede a obviar aquí la descripción de tal proceso por insustancial para el público objetivo de este manual, indicado antes, además de potencial estorbo por las posibles políticas contradictorias en los entornos donde se aplicara, que convertirían una guía de instalación del sistema operativo en intranscendente.

No obstante, como ya se indicó en el cuerpo de esta Memoria, se vuelve a recordar la posible conveniencia del uso de LVM para asistir en la gestión del almacenamiento ya que retira limitaciones físicas tanto de los dispositivos como de sus particiones, incidiendo en que, de desearse su uso, debe activarse durante el particionado.

2 Instalación de software en el host anfitrión

Para la instalación de la solución aportada, basada en la virtualización mediante contenedores, debe instalarse primero Docker en el host anfitrión para usar los contenedores descritos en la Memoria, además de wget y del cliente de PostgreSQL. También es deseable disponer de una instalación de Git para copiar el repositorio que aloja los archivos que permiten replicar el entorno presentado.

2.1 Docker

La instalación de Docker requiere de unos pocos comandos, que se transcriben a continuación desde su documentación oficial:

```
$ sudo apt-get update && sudo apt-get dist-upgrade -y
$ sudo apt-get install -y ca-certificates curl
$ sudo install -m 0755 -d /etc/apt/keyrings
$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
    -o /etc/apt/keyrings/docker.asc
$ sudo chmod a+r /etc/apt/keyrings/docker.asc
$ echo \
    "deb [arch=$(dpkg --print-architecture) \
    signed-by=/etc/apt/keyrings/docker.asc] \
    https://download.docker.com/linux/ubuntu \
    $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
    sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
$ sudo apt-get update
$ sudo apt-get install -y \
    docker-ce docker-ce-cli containerd.io \
    docker-buildx-plugin docker-compose-plugin
```

2.1.1 Verificación de instalación y funcionamiento

Siendo recomendable probar la instalación, el fabricante emplaza a ejecutar el siguiente comando, que descarga y ejecuta un contenedor de prueba:

```
$ sudo docker run hello-world
```

2.1.2 Administración de Docker sin escalada de privilegios

Por defecto, Docker no permite que usuarios distintos de `root` usen el comando `docker` ya que este permite gestionar los contenedores. Pero, de desearse que los usuarios puedan gestionarlos sin que escalen privilegios, se puede crear el grupo de usuarios homónimo y añadirlos al mismo:

```
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
```

De estos comandos, el primero crea el grupo de usuarios descrito y el segundo añade el usuario que se indique al grupo, *sustituyendo* `$USER` *por su nombre y teniéndose que usar una vez por usuario, realizando la sustitución correspondiente en cada caso.* Si el usuario incluido al grupo `docker` tuviera alguna sesión abierta, necesitará cerrarla y volver a iniciarla para que se le apliquen los cambios.

> *Para el resto de este manual, se presumirá que este paso se ha realizado para el usuario que se use, por lo que se omitirá el uso de sudo al usar docker.*

2.2 Wget

Usar Wget es *necesario* para ejecutar el *script* `shared/fetch.sh`, que actualiza los datos de OpenStreetMap disponibles localmente y usados por el sistema. Su instalación es relativamente trivial mediante paquetería:

```
$ sudo apt-get update && sudo apt-get dist-upgrade -y
$ sudo apt-get install -y wget
```

2.3 Cliente de PostgreSQL

Disponer del cliente de PostgreSQL es *necesario* para ejecutar satisfactoriamente un *script* que crea las bases de datos y usuarios necesarios. Su instalación es factible mediante paquetería, aunque es *extremadamente* recomendado no recurrir a los orígenes por defecto sino añadir antes los de su fabricante como otra fuente de paquetes para disponer de la versión más actualizada:

```
$ sudo apt-get update && sudo apt-get dist-upgrade -y
$ sudo install -m 0755 -d /etc/apt/keyrings
$ sudo apt-get install -y ca-certificates wget
$ wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | \
    gpg --dearmor | sudo tee /etc/apt/keyrings/pgdg.gpg > /dev/null
$ sudo chmod a+r /etc/apt/keyrings/pgdg.gpg
$ sudo sh -c 'echo "deb [signed-by=/etc/apt/keyrings/pgdg.gpg] \
    https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > \
    /etc/apt/sources.list.d/pgdg.list'
$ sudo apt-get update
$ sudo apt-get install -y postgresql-client
```

2.4 Git

Usar Git es *recomendado* para obtener la *imprescindible copia del repositorio de este TFG*, donde están los archivos que definen los contenedores referidos en la Memoria y su configuración. Su instalación es relativamente trivial mediante paquetería:

```
$ sudo apt-get update && sudo apt-get dist-upgrade -y
$ sudo apt-get install -y git
```

2.5 Multiplexor de la terminal

Usar un multiplexor de la terminal es *completamente opcional, pero extremadamente recomendado* ya que su uso hace que, de trabajar localmente con el equipo donde se emplee, se pueda abandonar físicamente con garantías mínimas de no manipulación por parte de terceros; o que, de trabajar usando alguna conexión remota, no se pierda trabajo si la conexión se interrumpe. Esto es muy deseable dado que algunos de los pasos que se describen en este anexo pueden llegar a necesitar varias horas para completarse. Algunas opciones son `screen` o `tmux`, disponibles mediante paquetería:

```
$ sudo apt-get update && sudo apt-get dist-upgrade -y
$ sudo apt-get install -y screen tmux
```

3 Creación y/o captura de configuraciones e información

Instalado el software necesario, procede recabar copia del repositorio y de la información que los diversos programas necesitan para operar.

3.1 Obtención de copia del repositorio

De haber instalado Git como se ha indicado anteriormente, procurarse copia del repositorio es tan simple como clonarlo[1]:

```
$ git clone https://github.com/alvarogarcia23/tfg-prototipo
```

> *En lo sucesivo se supondrá, para todos los comandos indicados en este manual, que se ejecutan desde la raíz de la copia del repositorio asociado a este TFG.*

3.2 Obtención de datos de OpenStreetMap

En el repositorio anejo a la Memoria se incluye un *script* que automáticamente descarga, verifica y coloca adecuadamente la información geográfica necesaria, creada desde OSM (OpenStreetMap).

[1] Por defecto, la copia se almacenará en una subcarpeta con el mismo nombre del repositorio en la ruta actual, pero se puede especificar el nombre de la carpeta local y/o la ruta relativa o absoluta donde Git ha de depositar la copia solicitada si se le indica, indicándosele en el mismo comando al final de este.

Si se han seguido los pasos descritos hasta ahora, para usar el mencionado *script* basta con ejecutar los siguientes comandos:

```
$ chmod 755 fetch.sh
$ ./fetch.sh
```

Debe prestarse especial atención a las tres últimas líneas que se devuelvan por pantalla, que deberán indicar "OK" para los tres (3) archivos que el script verifica. En otro caso deberán borrarse los archivos correspondientes de la carpeta `/shared/osm/` y reintentarlo pasados unos minutos, u obtener, verificar y colocar en la ruta señalada los archivos en formato PBF con los datos de OSM pertinentes, cuidando de darles los nombres que se relacionan a continuación:

Región	Nombre del archivo
Castilla y León	`castilla-y-leon-latest.osm.pbf`
España	`spain-latest.osm.pbf`
Portugal	`portugal-latest.osm.pbf`

Table 1: Nombres de archivo a usar para las copias locales de datos de OSM.

4 Preparación de contenedores

Recabados los archivos y datos necesarios y/o convenientes, se ha de preparar las imágenes de contenedores, lo que se debe hacer mediante compilación de las imágenes a usar.

4.1 Identificación previa ante proveedores online

La mayoría de imágenes usadas son o se basan en otras publicadas en línea usando servicios de Docker (Docker Hub) o GitHub (GitHub Packages) que, para protegerse de abusos, *pueden requerir identificación previa, aunque de no satisfacerse la descarga puede fallar sin mención a tal requisito.* Por ello, se recomienda identificar el servicio-cliente `docker` desplegado ante ambos Docker y GitHub antes de continuar.

4.1.1 Identificación ante Docker Hub

La identificación ante los servidores en línea de Docker se reduce a registrarse en la página web de este[2], ejecutar el comando que se indica a continuación y

[2] https://hub.docker.com/signup

completar los pasos e instrucciones que se indiquen por pantalla, aportando el usuario y contraseña de la cuenta creada:

```
$ docker login
```

4.1.2 Identificación ante GitHub Packages

La identificación del servicio-cliente `docker` desplegado ante GitHub es relativamente más complejo que la descrita en el apartado anterior, ya que este otro proveedor obliga a todos sus usuarios a obtener con carácter previo un *token* que autorice a acceder a la API de este.

Los pasos para conseguir el mencionado *token* son:

- Crear la correspondiente cuenta de usuario de GitHub[3] si no se dispone de una y confirmar la dirección de correo.
- Habiendo accedido a la cuenta de GitHub, seleccionar la foto de perfil de nuestra cuenta en la esquina derecha del encabezado de la página y seleccionar a la opción *"Settings"*.
- En la nueva página, seleccionar la opción *"Developer settings"* al final del menú lateral izquierdo.
- En la nueva página, desplegar la opción *"Personal access tokens"* y seleccionar la opción *"Fine-grained tokens"*.
- En la nueva página, seleccionar el botón *"Generate new token"*, en la parte superior derecha debajo del encabezado.
- Establecer un nombre (*"Token name"*) y una fecha de caducidad (*"Expiration"*) adecuados para el *token* a crear.
- Fijar y/o confirmar que el propietario del token (*"Resource owner"*) sea el correcto, por lo general la propia cuenta personal y no alguna organización de la que se forme parte.
- Establecer el acceso a repositorios (*"Repository access"*) en, al menos, acceso público de solo lectura (*"Public Repositories (read-only)"*).
- Confirmar la creación del *token* seleccionando la opción correspondiente ("*Generate token*") al final del formulario.

[3] https://github.com/signup

- Una vez completado el proceso, *se mostrará el token creado exclusivamente una vez*, permitiendo copiarlo para usarlo con la opción correspondiente, *que de necesitarse se podrá regenerar, invalidando el anterior token asignado.*

Completado el proceso y disponiendo del *token*, la identificación ante GitHub simplemente requiere ejecutar el siguiente comando en una terminal, disponiendo usuario y contraseña como se indica más adelante:

```
$ docker login ghcr.io
```

> *La combinación de usuario y contraseña a entregar en la terminal para esta identificación es el del token recabado, no el del usuario de GitHub, implicando que como usuario se debe usar el nombre de la entidad designada como propietaria del token; y como contraseña el propio token, tal y como se haya copiado al crearlo (incluyendo, por ejemplo, espacios si los hubiera).*

4.2 Compilación de imágenes

Los contenedores usados han sido modificados expresamente para este TFG partiendo de bases previamente publicadas, por lo que las imágenes de estos no pueden ser descargadas, sino que deben ser compiladas.

Habiendo realizado las identificaciones previas en `docker` indicadas antes, se pueden dejar todos los esfuerzos necesarios para crearlas a los automatismos diseñados ejecutando el siguiente comando[4]:

```
$ docker compose build mapserver postgis valhalla nominatim webapp
```

5 Preparación de bases de datos

El gestor de base de datos usado, PostgreSQL, por defecto no incluye ninguno de los usuarios ni de las bases de datos necesarias para un uso adecuado. Solucionarlo requiere que PostgreSQL esté en ejecución, para lo que debe ejecutarse primero el siguiente comando:

```
$ docker compose up -d postgis
```

[4]Dado que Docker emplea un sistema caché para compilaciones de imágenes locales, sólo se rehacen los pasos necesarios en base a cambios en instrucciones o archivos locales, por lo que *de quererse forzar la regeneración completa sin emplear tal caché debe añadirse el modificador* --no-cache al comando.

Una vez en ejecución, se ha dispuesto otro *script* que, ejecutado desde la máquina anfitriona, creará los usuarios y bases de datos necesarias mediante el uso del cliente de PostgreSQL instalado antes:

```
$ chmod 755 ./postgis/sql_skel_create.sh
$ ./postgis/sql_skel_create.sh
```

6 Poblado de bases de datos

Como con todo sistema que emplea bases de datos, las bases de datos deben contener los datos con los que se trabajará, por lo que se ha de incorporar los datos a estas, esto es, su poblado.

6.1 Nominatim

La base de datos que emplea Nominatim no necesita de un procedimiento especial de poblado ya que este es controlado desde el contenedor de este software. No obstante, se recomienda dejarle hacer esta operación en ausencia de ejecución de otros componentes, ya que así se podrá aprovechar los recursos disponibles con mayor eficiencia y tardar menos tiempo en completarse. Para iniciarla basta con ejecutar el siguiente comando:

```
$ docker compose up nominatim
```

Téngase en cuenta que esta operación puede tardar más de una hora, dependiendo del hardware disponible, y que este comando bloqueará la terminal ya que mostrará el registro de la propia aplicación por pantalla, en la que deberá esperarse hasta que se muestre el siguiente mensaje:

```
--> Nominatim is ready to accept requests
```

Una vez aparezca, el proceso ha concluido, y ya se puede parar el servicio[5] mediante la combinación de teclas `Ctrl + C`.

6.2 Imposm (MapServer)

La base de datos usada por MapServer si requiere de un procedimiento especial para su poblado, en el que se ha de usar a Imposm.

[5] También puede pararse anticipadamente usando la misma combinación de teclas, pero el progreso alcanzado se perderá, comenzando de nuevo desde cero cuando se inicie de nuevo.

Primero, se ha de crear una caché en base a los archivos PBF con datos de OSM antes descargados en el paso 3.2 para después incorporar toda la información disponible a la base de datos:

```
$ ./imposm3/releases/imposm-0.11.1-linux-x86-64/imposm import \
    -mapping ./basemaps/imposm3-mapping.json \
    -read ./shared/osm/spain-latest.osm.pbf \
    -srid 3857 -cachedir ./imposm3-cache/ -diff -overwritecache
$ ./imposm3/releases/imposm-0.11.1-linux-x86-64/imposm import \
    -mapping ./basemaps/imposm3-mapping.json \
    -read ./shared/osm/portugal-latest.osm.pbf \
    -srid 3857 -cachedir ./imposm3-cache/ -diff -appendcache
```

En caso de quererse incorporar más zonas, basta con repetir el último comando cambiando el archivo `./shared/osm/portugal-latest.osm.pbf` por uno de los que se deseen incorporar, repitiéndolo con cada archivo.

Una vez generada la caché con la información de todas las zonas a incorporar, queda transcribirla a la base de datos:

```
$ ./imposm3/releases/imposm-0.11.1-linux-x86-64/imposm import \
    -mapping ./basemaps/imposm3-mapping.json \
    -cachedir ./imposm3-cache/ \
    -connection postgis://mapserver_imposm:imposm@localhost/mapserver_osm \
    -srid 3857 -diff -write -optimize -deployproduction
```

> *Si hubiesen cambiado los parámetros de conexión a la base de datos (nombre de usuario, contraseña, host o nombre de la base de datos) deberá modificarse la URL asociada al argumento -connection del comando anterior con los correctos.*

Completado este último comando, la información geográfica de OSM estará disponible en la base de datos para su uso por MapServer.

7 Generación de caché de MapServer con MapCache

Llegado este punto, sólo queda un último paso para tener todo preparado: llenar de contenido la caché de MapServer, gestionada por MapCache.

7.1 Generación de la copia caché

Para generar la copia caché que MapCache usará para responder a las solicitudes que se le realicen se debe arrancar el contenedor que alberga a PostgreSQL, así como el de MapServer y MapCache, y usar la utilidad `mapcache_seed` incluida en el segundo contenedor, lo que se consigue al ejecutar lo siguiente:

```
$ docker compose up -d mapserver postgis
$ docker exec -it mapserver /mapcache/mapcache_seed_cache.sh
```

8 Arranque de servicios

Por último, han de arrancarse los contenedores de los distintos servicios:

```
$ docker compose up -d mapserver postgis nominatim valhalla webapp
```

Una vez arrancados los servicios, se puede acceder a ellos en los siguientes puertos del host anfitrión:

Servicio	Puerto
Interfaz web	80
API de MapServer y MapCache	8082
API de rutas (Valhalla)	8002
API de búsqueda de lugares (Nominatim)	8080
Acceso a PostgreSQL	5432

Table 2: Puertos dispuestos para acceder a cada servicio.

Se ha de incidir en que, una vez arrancados los contenedores de los servicios con el comando anterior, estos continuarán ejecutándose indefinidamente, siendo reiniciados automaticamente por el servicio docker local de interrumpirse. No obstante, puede haber situaciones ante las que deba realizarse una parada controlada de los servicios desplegados, o bien detener el reinicio automático de los mismos, para lo que se ha de ejecutar el siguiente comando[6]:

```
$ docker compose stop mapserver postgis nominatim valhalla webapp
```

Una vez se desee reanudar los servicios, basta con repetir de nuevo el comando al principio de este apartado.

[6]Se pueden limitar los contenedores a parar retirando del comando aquellos que se desee continúen en ejecución.