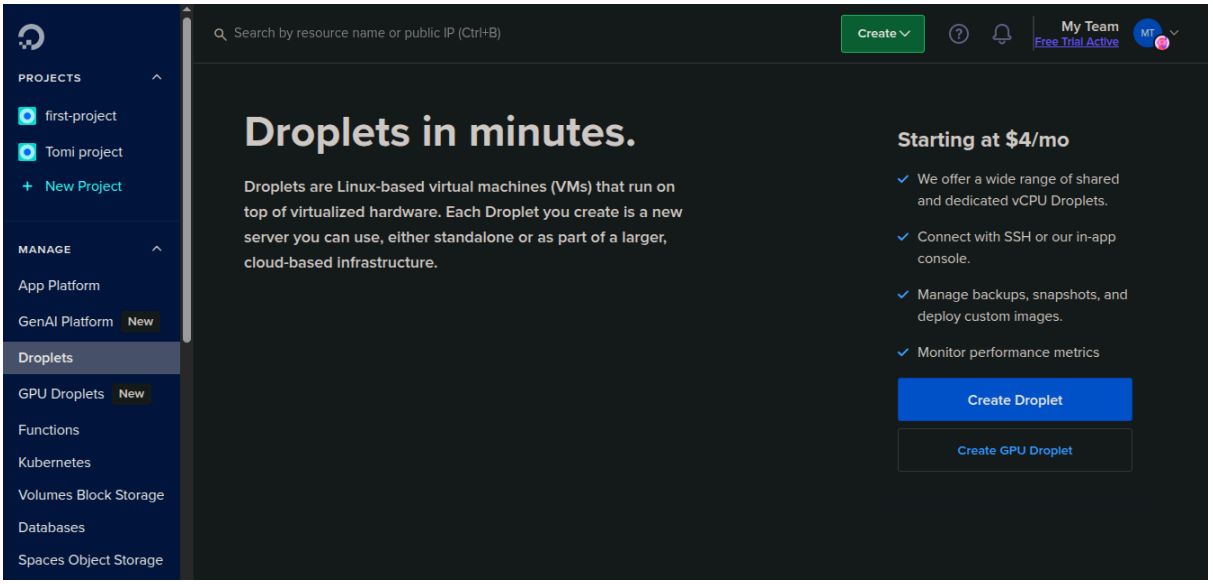
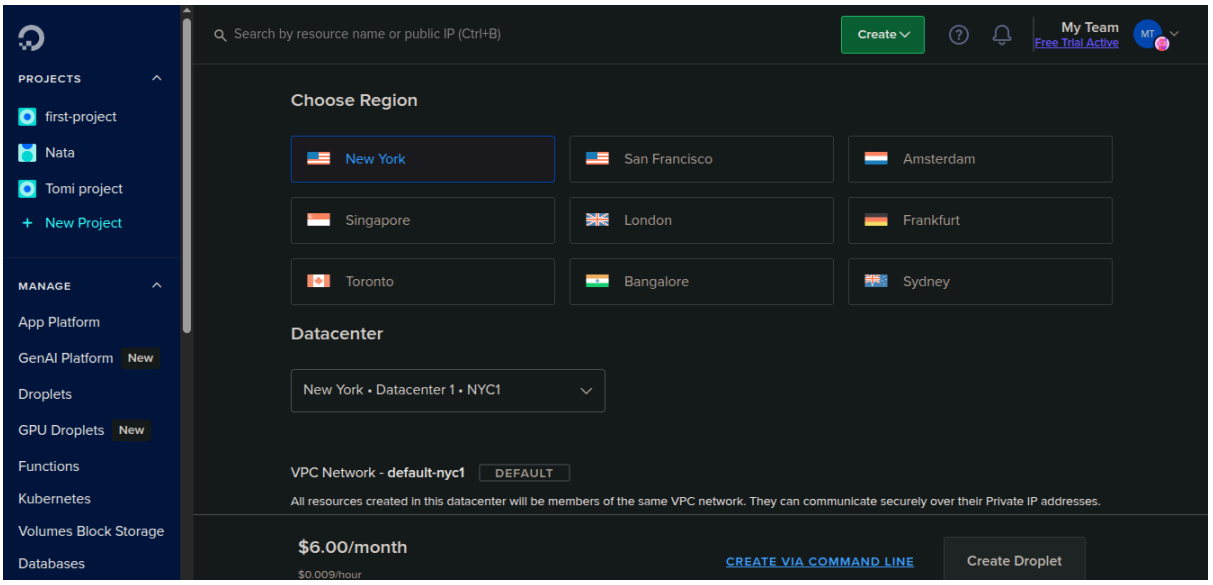
1. Levantar una VM con ubuntu server.

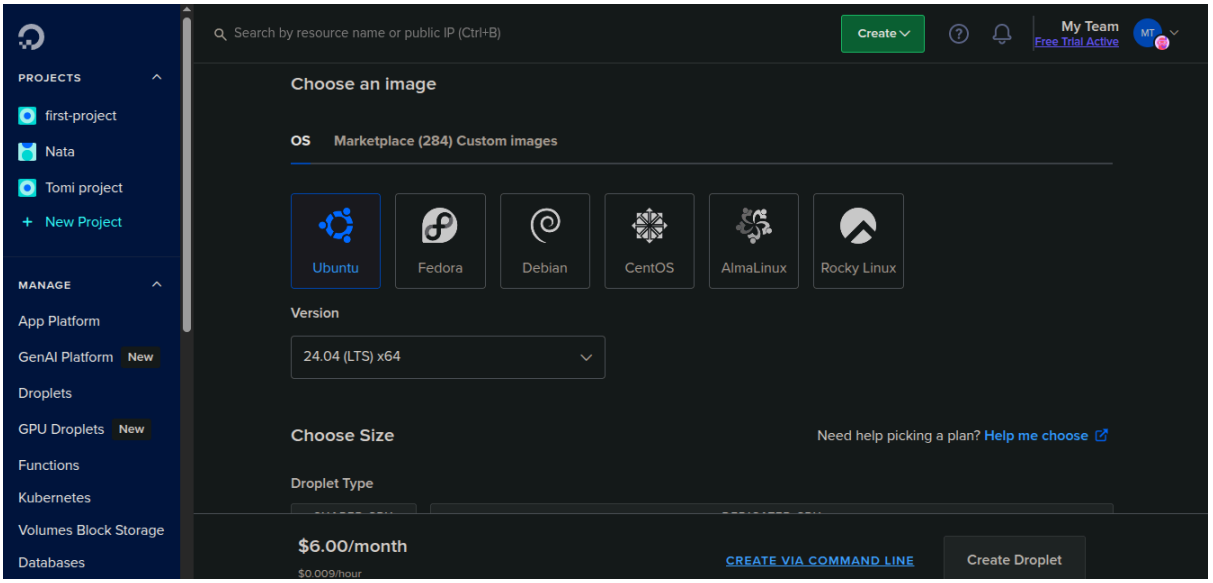
Para crear una VM con ubuntu server lo primero que debemos de hacer es ir a la sección "Droplets" del menú lateral izquierdo. En esta misma sección seleccionaremos el botón a la derecha que nos dice "Create Droplet".



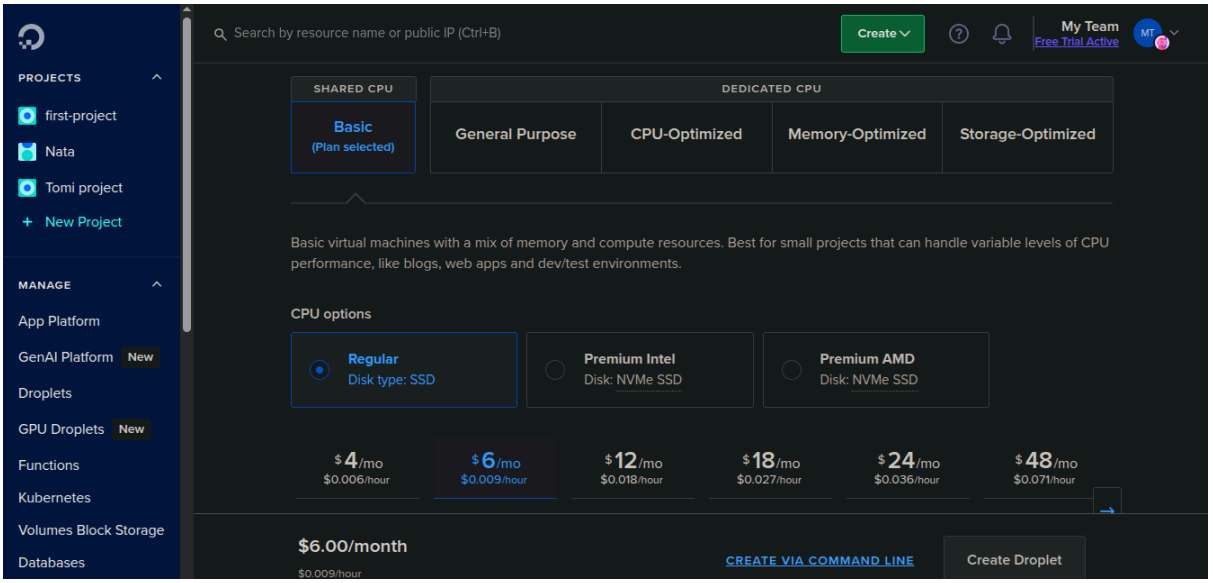
Seleccionamos la región y un Datacenter, en este caso existen 3 y seleccione Datacenter 1.



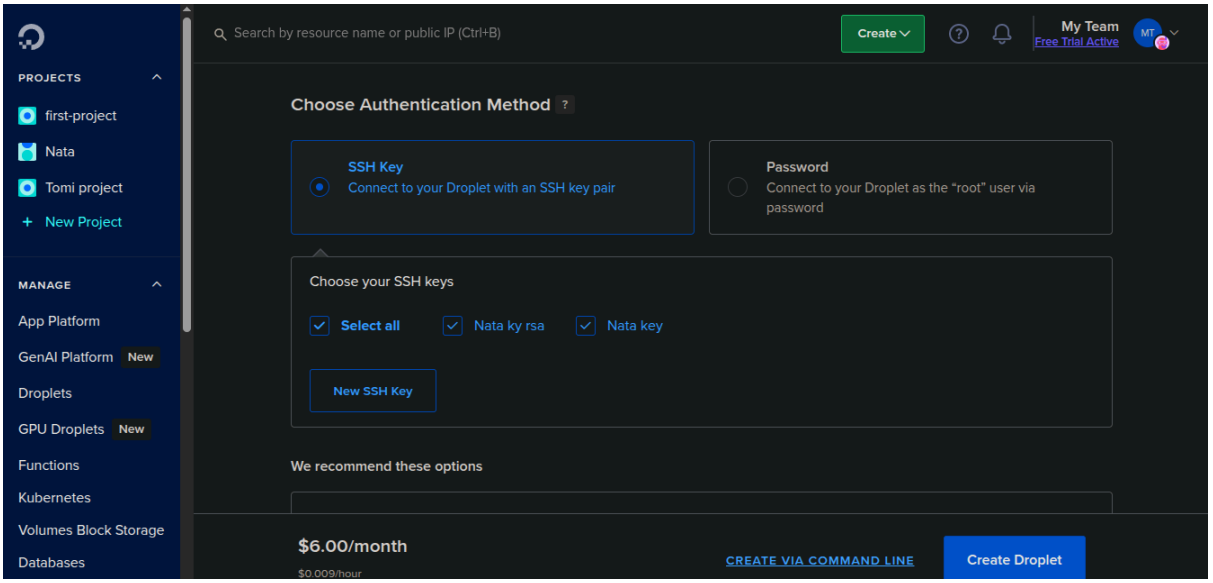
Elegimos la imagen del Sistema Operativo y la versión del mismo, en mi caso elegir una Ubuntu LTS para mejor funcionamiento y mitigar posibles errores.



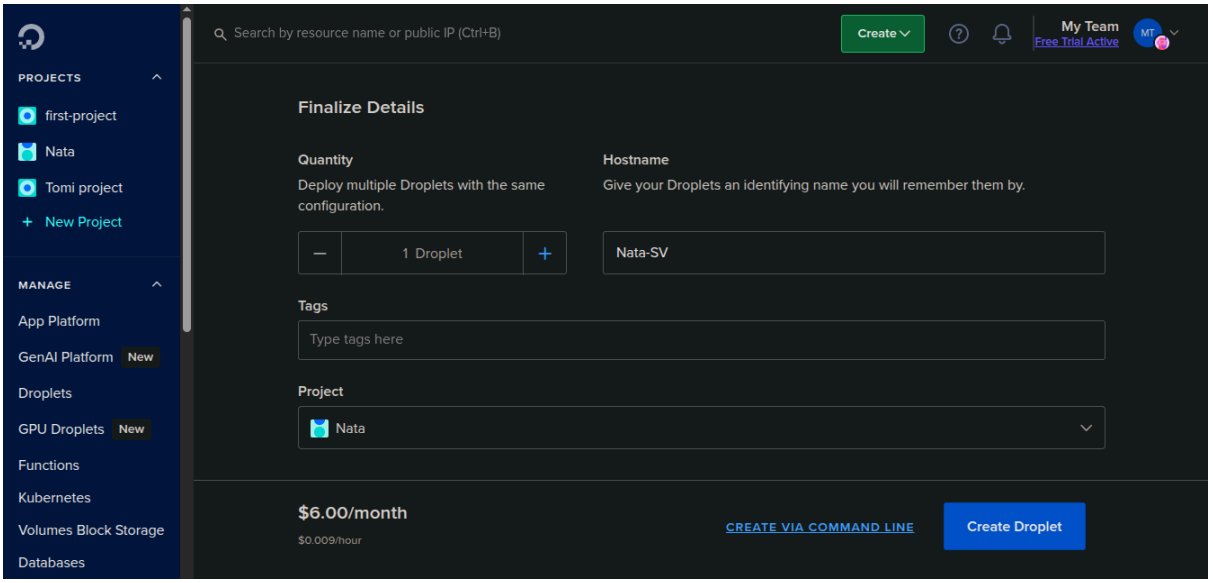
Seleccionamos el plan y la CPU Options, donde especifica la cantidad de RAM y núcleos de procesador nos van a otorgar.



Lo que sigue es configurar el método de Autenticación, en mi caso configure una SSH Key que ya previamente tenía generada en mi Ubuntu /home/.ssh



Por último, le asignamos un nombre al Droplet, lo asociamos a un proyecto y podemos crear la VM.



Para la creacion del Script setup.sh:

Click aqui para dirigirse al script

El script debe ejecutarse SOLO como usuario root.

Si lo ejecutamos la STDOUT será:

```
usermod: user 'webexperto' does not exist
info: Adding user `webexperto' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `webexperto' (1001) ...
warn: Waiting for lock to become available...
info: Adding new user `webexperto' (1001) with group `webexperto (1001)' ...
info: Creating home directory `/home/webexperto' ...
info: Copying files from `/etc/skel' ...
info: Adding new user `webexperto' to supplemental / extra groups `users' ...
info: Adding user `webexperto' to group `users' ...
info: Adding user `userssh' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `userssh' (1002) ...
info: Adding new user `userssh' (1002) with group `userssh (1002)' ...
info: Creating home directory `/home/userssh' ...
info: Copying files from `/etc/skel' ...
info: Adding new user `userssh' to supplemental / extra groups `users' ...
info: Adding user `userssh' to group `users' ...
Se pueden actualizar 55 paquetes. Ejecute «apt list --upgradable» para verlos.
The following upgrades have been deferred due to phasing:
  ubuntu-drivers-common
```

Al haber tantas actualizaciones y descargas es imposible hacer una captura solamente, por lo que pasaré el tail del script.

```
Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
 Test @ session #1: bash[886], login[784]
 Test @ session #3: bash[1431], sshd[1375]
 Test @ user manager service: systemd[874]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
info: Adding user `nginx' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `nginx' (1003) ...
info: Adding new user `nginx' (1003) with group `nginx (1003)' ...
info: Creating home directory `/home/nginx' ...
info: Copying files from `/etc/skel' ...
info: Adding new user `nginx' to supplemental / extra groups `users' ...
info: Adding user `nginx' to group `users' ...
root@test1:~#
```

Al reiniciar el Droplet ahora tenemos que loguearnos via ssh con el usuario permitido para la misma conexión, el “userssh”.

Cuando ingresemos a la home correspondiente del usuario, veremos 5 archivos.

- main.zip
- my-app-main
- noip-duc_3.3.0.tar.gz
- noip-duc_3.3.0

Esto se debe a que descargamos de mi propio GitHub el repo con todo el contenido necesario para runnear la web ‘main.zip’.
Además de eso nos va a descargar el comprimido para instalar el DUC (Dynamic Update Client) ‘noip-duc_3.3.0.tar.gz’ que nos solicita No-IP para generar la asociación del DNS con el Droplet y su IP.

Docker Compose

Ahora vamos a entrar en el proceso de creación y ejecución del docker-compose definido para este proyecto.

Primeramente tenemos que cambiar de usuario durante la sesión del actual “userssh”, por “nginx”. Este último es quien tiene los permisos del grupo docker, es decir es quien puede ejecutar los comandos del binario de ‘dockerd’.

Para la creación del compose nos basamos en un repositorio de GitHub de PeladoNerd. En el mismo tenemos como servicios un sitio web estático, un generador de certificados ssl y un servidor proxy.

Este archivo `compose.yml` configura un proxy inverso Nginx (`nginx-proxy`) que gestiona el tráfico HTTP y HTTPS. Utiliza un compañero (`letsencrypt`) para obtener y renovar automáticamente certificados SSL/TLS para tu dominio (`nataec.ddns.net`). El servicio `www` contiene la aplicación web real (en este caso, un servidor Nginx sirviendo contenido desde `./sitio1`) y está configurado para ser accesible a través del proxy utilizando las variables de entorno `VIRTUAL_HOST` y `LETSENCRYPT_HOST`. Las dependencias aseguran que los servicios se inicien en el orden correcto. Los volúmenes se utilizan para compartir la configuración de Nginx, los certificados SSL/TLS, los archivos del sitio web y el socket de Docker entre los contenedores y el host.

```
services:
  nginx-proxy:
    image: jwilder/nginx-proxy
    restart: always
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - /var/run/docker.sock:/tmp/docker.sock:ro
      - ./certs:/etc/nginx/certs:ro
      - ./vhostd:/etc/nginx/vhost.d
      - ./html:/usr/share/nginx/html
      - ./acme:/etc/acme.sh
    labels:
      - com.github.jrcs.letsencrypt_nginx_proxy_companion.nginx_proxy

  letsencrypt:
    image: jrcs/letsencrypt-nginx-proxy-companion
    restart: always
    environment:
      - NGINX_PROXY_CONTAINER=nginx-proxy
      - CREATE_DEFAULT_CERTIFICATE=true
      - DEBUG=true
    volumes:
      - ./certs:/etc/nginx/certs:rw
      - ./vhostd:/etc/nginx/vhost.d
      - ./html:/usr/share/nginx/html
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ./acme:/etc/acme.sh
    volumes_from:
      - nginx-proxy:rw

  www:
    image: nginx:1.28.0-alpine3.21
    restart: always
    expose:
      - "80"
    volumes:
      - ./sitio1:/usr/share/nginx/html:ro
    environment:
      - VIRTUAL_HOST=nataec.ddns.net
      - LETSENCRYPT_HOST=nataec.ddns.net
      - LETSENCRYPT_EMAIL=canteronatanael8@gmail.com
    depends_on:
      - nginx-proxy
      - letsencrypt

volumes:
  certs:
  html:
  vhostd:
  acme:
```

A continuacion detallamos cada parte del compose:

Sección Principal: `services`

Esta sección define los diferentes contenedores (servicios) que Docker Compose va a crear y gestionar. Cada entrada dentro de `services` representa un servicio individual.

- **`nginx-proxy:`**
    - **`image: jwilder/nginx-proxy`**: Especifica la imagen de Docker que se utilizará para este servicio. En este caso, se está utilizando la imagen `jwilder/nginx-proxy`, que es un proxy inverso de Nginx automatizado. Este proxy se configura automáticamente para reenviar el tráfico a otros contenedores basándose en variables de entorno.
    - **`restart: always`**: Indica que Docker debe intentar reiniciar este contenedor automáticamente si falla o se detiene. Esto asegura la disponibilidad del proxy.
    - **`ports:`**: Define las asignaciones de puertos entre el host (tu máquina) y el contenedor.
        - **`- "80:80"`**: Mapea el puerto 80 de tu máquina al puerto 80 del contenedor. Este es el puerto estándar para el tráfico HTTP.
        - **`- "443:443"`**: Mapea el puerto 443 de tu máquina al puerto 443 del contenedor. Este es el puerto estándar para el tráfico HTTPS.
    - **`volumes:`**: Define los puntos de montaje de volúmenes, que permiten compartir datos entre el host y el contenedor o entre contenedores.
        - `- /var/run/docker.sock:/tmp/docker.sock:ro`: Monta el socket de Docker de tu máquina dentro del contenedor en modo de solo lectura (`ro`). Esto permite al `nginx-proxy` escuchar los eventos de creación y eliminación de otros contenedores para reconfigurarse automáticamente.
        - `- ./certs:/etc/nginx/certs:ro`: Monta el directorio `./certs` de tu máquina dentro del directorio `/etc/nginx/certs` del contenedor en modo de solo lectura. Aquí se almacenarán los certificados SSL/TLS.
        - `- ./vhostd:/etc/nginx/vhost.d`: Monta el directorio `./vhostd` de tu máquina dentro del directorio `/etc/nginx/vhost.d` del contenedor. Este directorio se utiliza para configurar hosts virtuales adicionales para Nginx.
        - `- ./html:/usr/share/nginx/html`: Monta el directorio `./html` de tu máquina dentro del directorio `/usr/share/nginx/html` del contenedor. Este es el directorio raíz predeterminado para los archivos HTML que podría servir Nginx si no se define un host virtual específico.

        - `- ./acme:/etc/acme.sh`: Monta el directorio `./acme` de tu máquina dentro del directorio `/etc/acme.sh` del contenedor. Aquí se almacenan los datos relacionados con la obtención de certificados SSL/TLS utilizando ACME (como Let's Encrypt).
    - **`labels:`**: Asigna etiquetas al contenedor. En este caso, la etiqueta `com.github.jrcs.letsencrypt_nginx_proxy_companion.nginx_proxy` es utilizada por el contenedor `letsencrypt` para identificar este proxy.
- **`letsencrypt:`**

    - **`image: jrcs/letsencrypt-nginx-proxy-companion`**: Especifica la imagen de Docker para este servicio, que es un compañero del `nginx-proxy` para automatizar la obtención y renovación de certificados SSL/TLS utilizando Let's Encrypt.
    - **`restart: always`**: Al igual que con `nginx-proxy`, asegura que este contenedor se reinicie si falla.
    - **`environment:`**: Define variables de entorno que se pasarán al contenedor.
        - **`- NGINX_PROXY_CONTAINER=nginx-proxy`**: Indica a este contenedor el nombre del contenedor `nginx-proxy` con el que debe interactuar.
        - **`- CREATE_DEFAULT_CERTIFICATE=true`**: Le dice al companion que cree un certificado autofirmado por defecto si no se encuentra un certificado para un host virtual.
        - **`- DEBUG=true`**: Activa el modo de depuración para obtener más información en los logs.
    - **`volumes:`**: Define los puntos de montaje de volúmenes para este contenedor. Observa que algunos directorios se comparten con `nginx-proxy`.
        - `- ./certs:/etc/nginx/certs:rw`: Monta el directorio `./certs` de tu máquina dentro del contenedor en modo de lectura y escritura (`rw`). Esto permite al companion escribir los certificados obtenidos.
        - `- ./vhostd:/etc/nginx/vhost.d`: Monta el directorio `./vhostd`.
        - `- ./html:/usr/share/nginx/html`: Monta el directorio `./html`.
        - `- /var/run/docker.sock:/var/run/docker.sock:ro`: Monta el socket de Docker en modo de solo lectura.
        - `- ./acme:/etc/acme.sh`: Monta el directorio `./acme`.
    - **`volumes_from:`**: Comparte los volúmenes definidos en otro servicio.
        - **`- nginx-proxy:rw`**: Monta todos los volúmenes del servicio `nginx-proxy` en este contenedor con permisos de lectura y escritura. Esto es una forma concisa de compartir los directorios de configuración de Nginx y los archivos HTML.

- **`www:`**
    - **`image: nginx:1.28.0-alpine3.21`**: Especifica la imagen de Docker para este servicio, que es una imagen oficial de Nginx basada en Alpine Linux, una distribución ligera. La etiqueta `1.28.0-alpine3.21` indica una versión específica de Nginx.
    - **`restart: always`**: Asegura el reinicio automático en caso de fallo.
    - **`expose:`**: Declara los puertos en los que el contenedor escucha dentro de su propia red Docker. Estos puertos no se publican directamente en el host a menos que se especifiquen en la sección `ports` del servicio `nginx-proxy`.
        - **`- "80"`**: Indica que el contenedor Nginx dentro de su red Docker escucha en el puerto 80.
    - **`volumes:`**: Define los puntos de montaje de volúmenes para este contenedor.
        - **`- ./sitio1:/usr/share/nginx/html:ro`**: Monta el directorio `./sitio1` de tu máquina dentro del directorio `/usr/share/nginx/html` del contenedor en modo de solo lectura. Aquí se colocarán los archivos de tu sitio web.
    - **`environment:`**: Define variables de entorno para este contenedor. Estas variables son cruciales para la configuración automática del `nginx-proxy`.
        - **`- VIRTUAL_HOST=nataec.ddns.net`**: Esta variable le dice al `nginx-proxy` que este contenedor debe ser accesible a través del dominio `nataec.ddns.net`. El proxy utilizará esta información para configurar el enrutamiento del tráfico.
        - **`- LETSENCRYPT_HOST=nataec.ddns.net`**: Esta variable le indica al `letsencrypt` companion que solicite un certificado SSL/TLS para el dominio `nataec.ddns.net`.
        - **`- LETSENCRYPT_EMAIL=canteronatanael8@gmail.com`**: Proporciona una dirección de correo electrónico que Let's Encrypt puede usar para notificaciones sobre el certificado.
    - **`depends_on:`**: Define las dependencias entre los servicios. Docker Compose se asegurará de que los servicios listados aquí se inicien antes que este servicio.
        - **`- nginx-proxy`**: Asegura que el contenedor `nginx-proxy` esté en funcionamiento antes de iniciar el contenedor `www`.
        - **`- letsencrypt`**: Asegura que el contenedor `letsencrypt` esté en funcionamiento antes de iniciar el contenedor `www`. Esto es importante para que el proxy esté listo para manejar las solicitudes y, si es necesario, el companion pueda obtener el certificado inicial.

Sección: `volumes`

Esta sección define volúmenes con nombre que pueden ser utilizados por los servicios. Esto permite que los datos persistan incluso si los contenedores se eliminan. En este caso, se

definen volúmenes con nombre, aunque en la sección `services`, los volúmenes se montan directamente desde directorios del host (volúmenes de enlace).

- **`certs:`**: Define un volumen llamado `certs`.
- **`html:`**: Define un volumen llamado `html`.
- **`vhostd:`**: Define un volumen llamado `vhostd`.
- **`acme:`**: Define un volumen llamado `acme`.

En este `docker-compose.yml`, aunque se definen estos volúmenes con nombre, los servicios `nginx-proxy` y `letsencrypt` están utilizando principalmente **volúmenes de enlace** (bind mounts) que mapean directorios de tu sistema de archivos directamente dentro de los contenedores. La sección `volumes` aquí podría estar presente por convención o para ser utilizada en otros servicios que se pudieran agregar al `compose` en el futuro.

Por último tendremos que hacer un ‘curl -I nataec.ddns.net’ para recibir el status de redireccionamiento de http>https. Sino podemos visitar el sitio