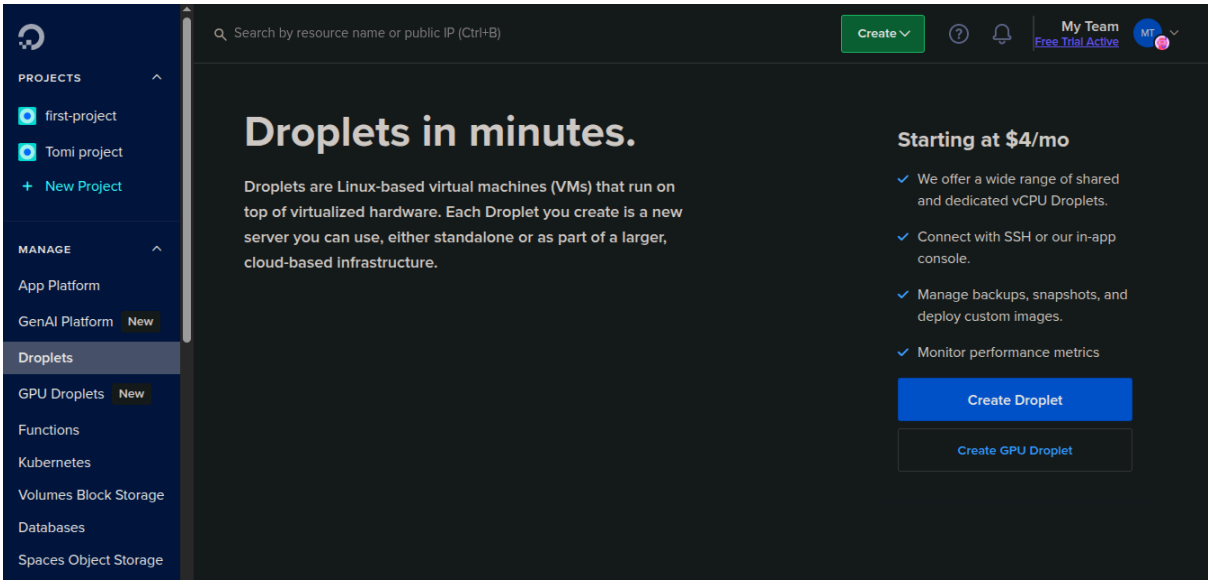
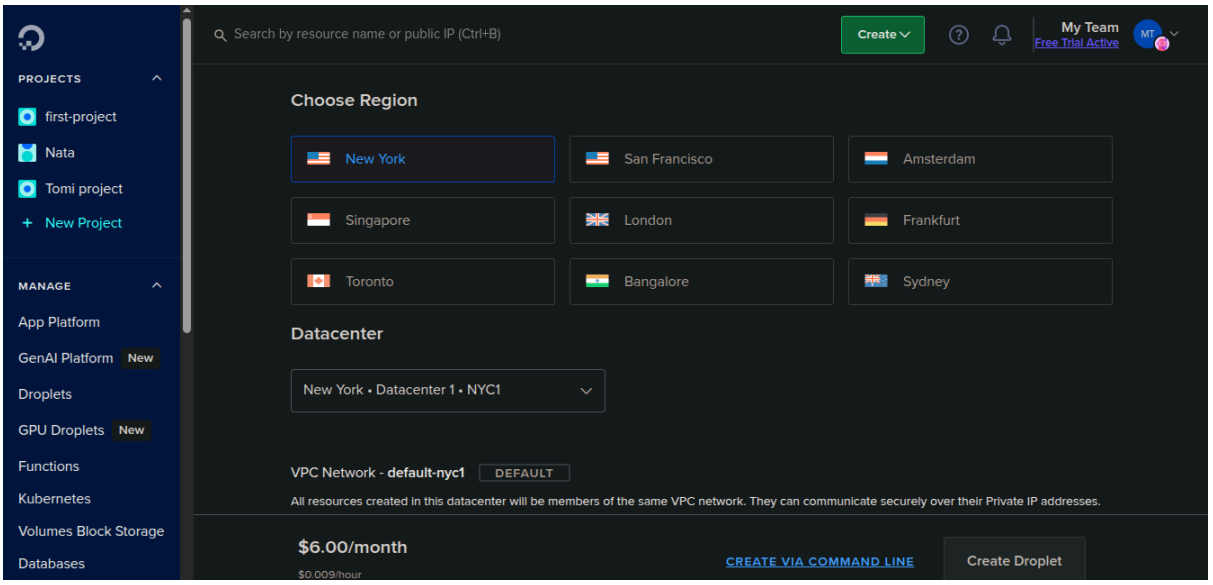
Digital Ocean:

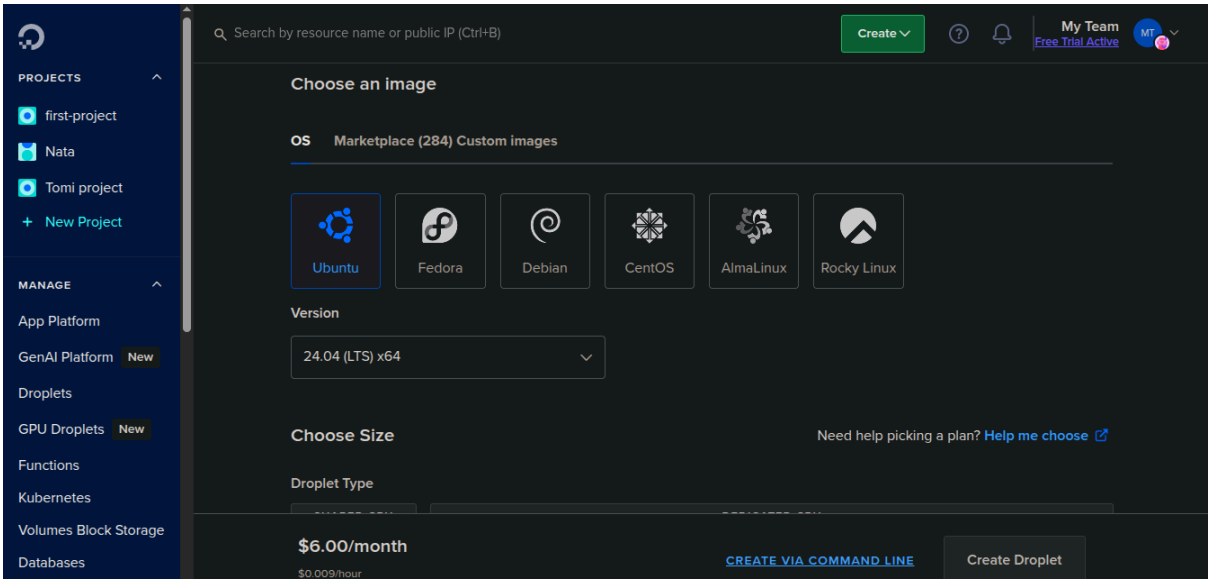
Para crear una VM con ubuntu server lo primero que debemos de hacer es ir a la sección "Droplets" del menú lateral izquierdo. En esta misma sección seleccionaremos el botón a la derecha que nos dice "Create Droplet”.



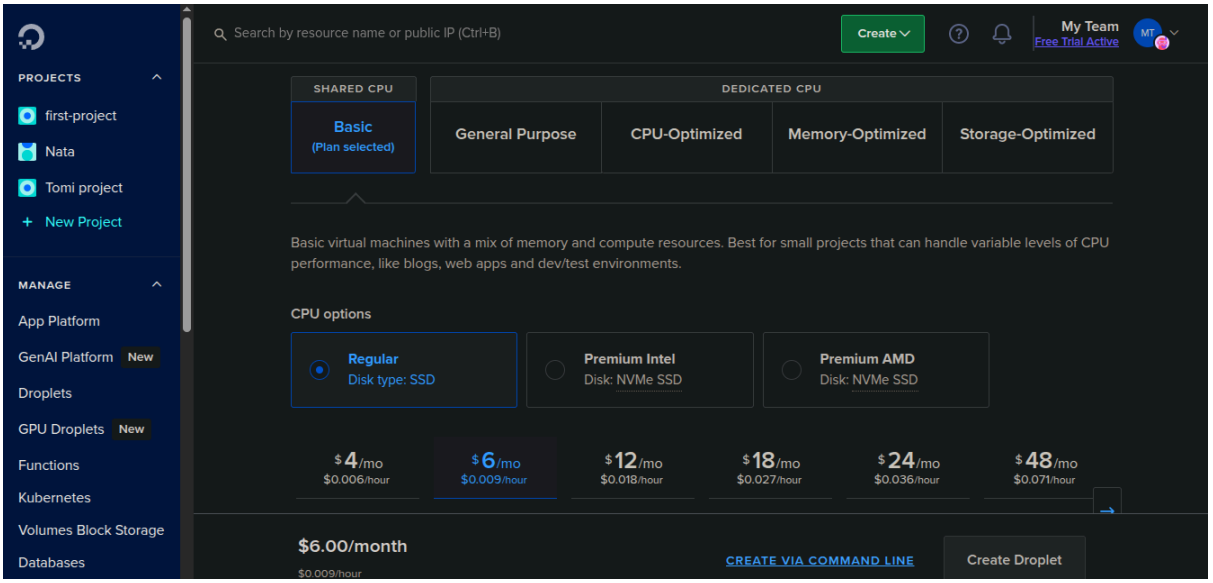
Seleccionamos la región y un Datacenter, en este caso existen 3 y seleccione Datacenter 1.



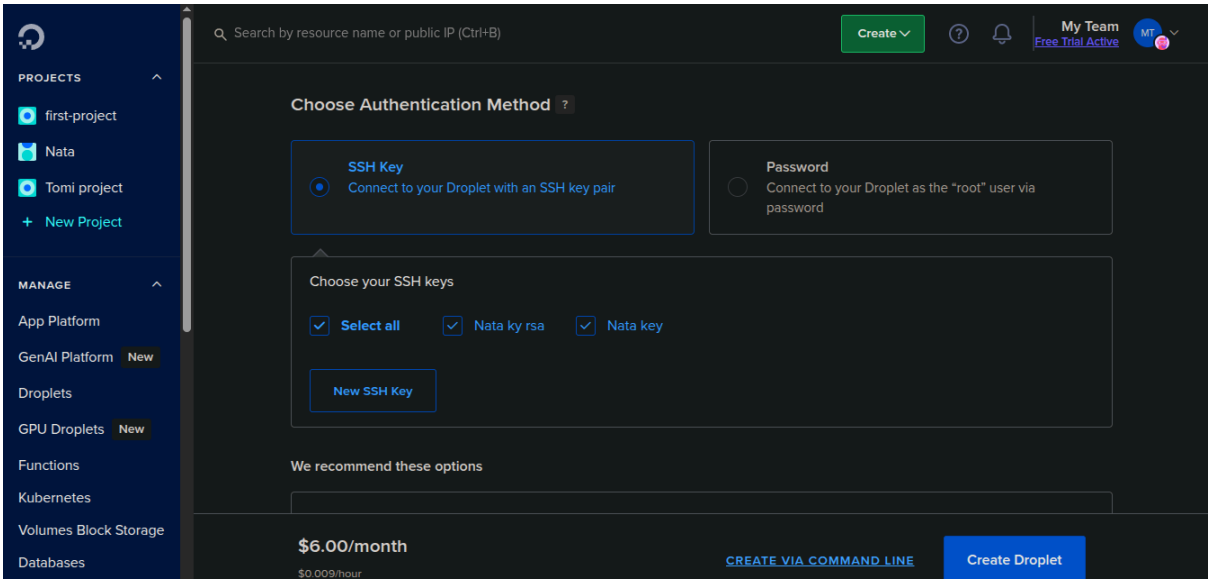
Elegimos la imagen del Sistema Operativo y la versión del mismo, en mi caso elegir una Ubuntu LTS para mejor funcionamiento y mitigar posibles errores.



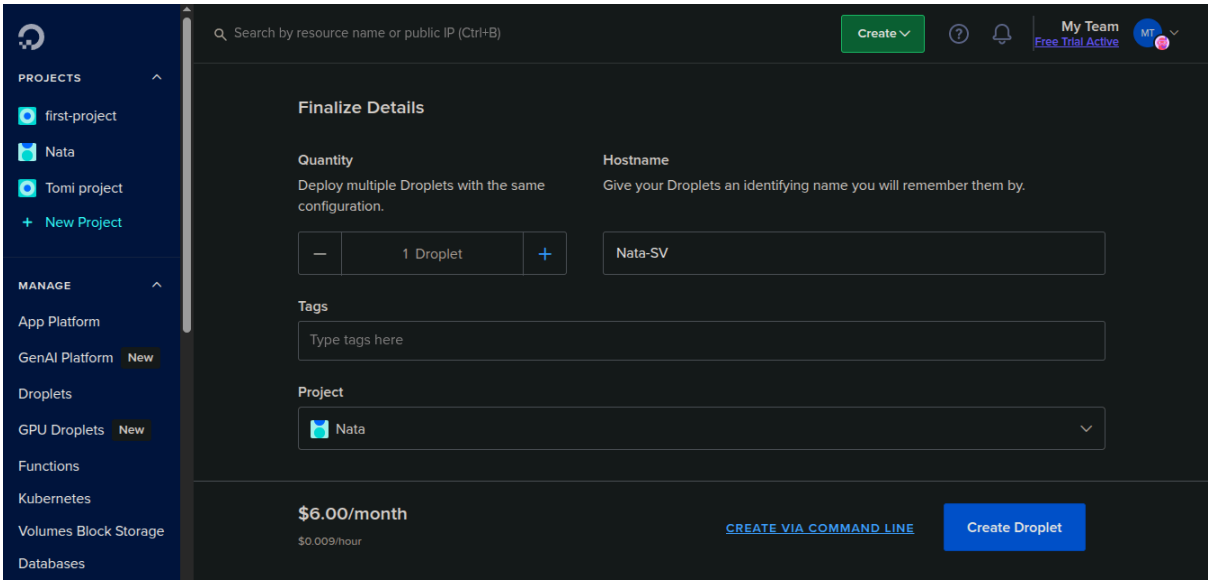
Seleccionamos el plan y la CPU Options, donde especifica la cantidad de RAM y núcleos de procesador nos van a otorgar.



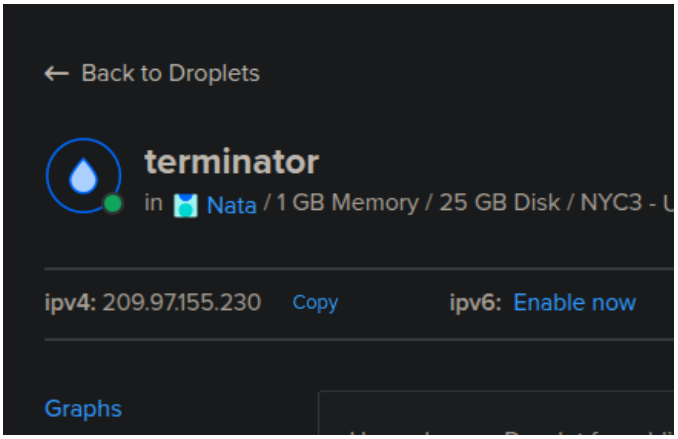
Lo que sigue es configurar el método de Autenticación, en mi caso configure una SSH Key que ya previamente tenía generada en mi Ubuntu /home/.ssh



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



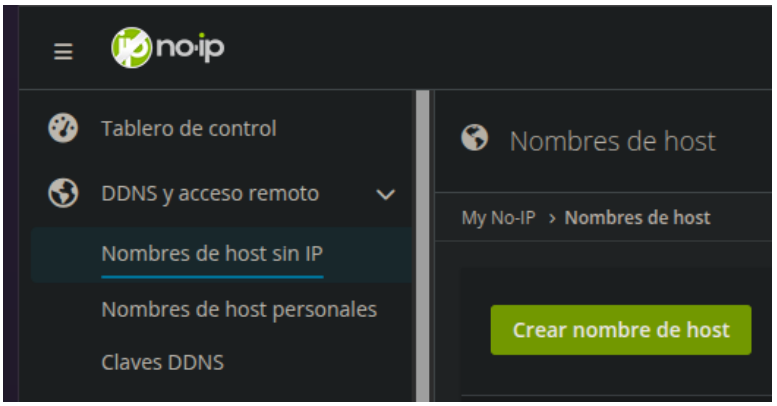
Vamos a copiar la IP del mismo para posteriormente usarla en No-IP



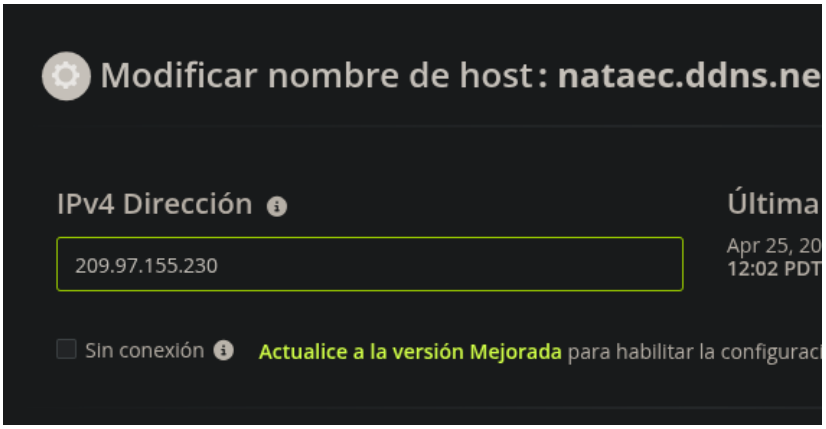
No-IP:

Vamos a crear una cuenta en la web.

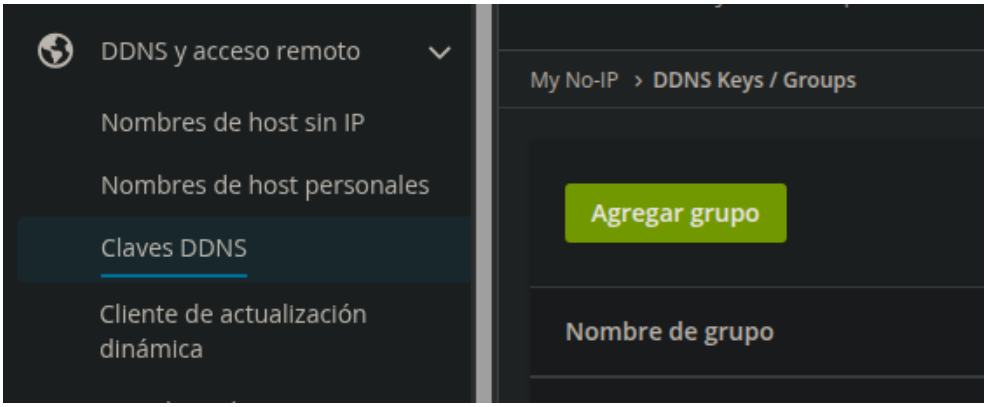
Nos vamos a ir al menú lateral izquierdo y seleccionamos “No-IP Hostnames” y creamos uno.



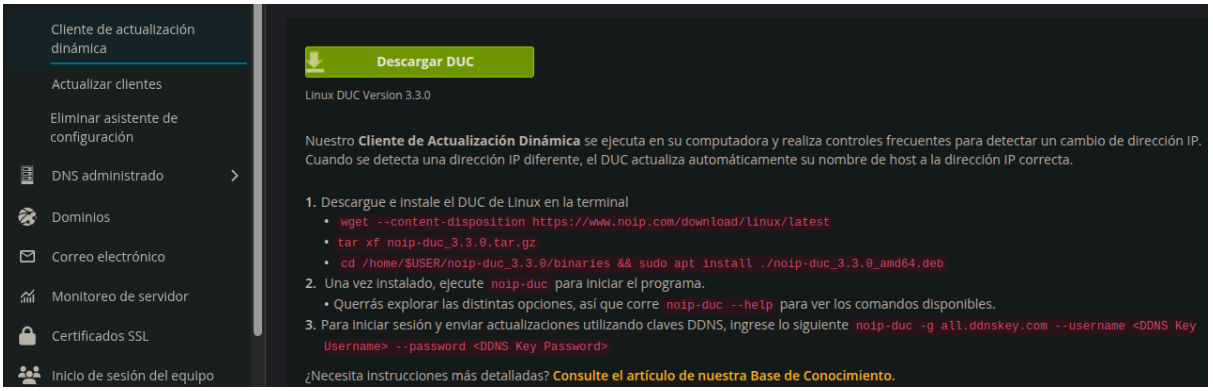
Donde se solicita la IP, pegamos la del Droplet y la actualizamos.



Luego iremos a “DDNS Key” y la creamos. Lo importante aquí es no perder estos datos, ya que la password será la única vez que la podremos copiar.



Por último, iremos a “Dynamic Update Client” para seguir los pasos que nos detalla para la descarga del cliente DUC y asociar nuestro DNS a un dispositivo, en este caso el Droplet.



El comando final
‘noip-duc -g all.ddnskey.com --username <DDNS Key Username> --password <DDNS Key Password>’

Tendrá que reemplazarse ‘<DDNS Key Username>’ por el usuario brindado al crear la DDNS Key, lo mismo con ‘<DDNS Key Password>’

Muy importante detallar que este proceso está automatizado en el script ‘init.sh’.

Script:

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.

- my-app-main
- noip-duc_3.3.0
- initduc.txt

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 Nextcloud 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


  nextcloud:
    image: nextcloud:latest
    restart: always
    expose:
      - "80"
    volumes:
      - nextcloud_data:/var/www/html/data
      - nextcloud_config:/var/www/html/config
      - nextcloud_apps:/var/www/html/apps
    environment:
      - VIRTUAL_HOST=nataec.ddns.net
      - LETSENCRYPT_HOST=nataec.ddns.net
      - LETSENCRYPT_EMAIL=canteronatanael8@gmail.com
      - MYSQL_HOST=nextcloud_db
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud
      - MYSQL_PASSWORD=nextcloud
      - MYSQL_ROOT_PASSWORD=root
    depends_on:
      - nginx-proxy
      - letsencrypt
      - nextcloud_db

  nextcloud_db:
    image: mariadb:10.6
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=root
      - MYSQL_DATABASE=nextcloud
      - MYSQL_USER=nextcloud
      - MYSQL_PASSWORD=nextcloud
    volumes:
      - nextcloud_db:/var/lib/mysql

volumes:
  certs:
  html:
  vhostd:
  acme:
  nextcloud_data:
  nextcloud_config:
  nextcloud_apps:
  nextcloud_db:
```