Después de crear el droplet, este te proporcionará una dirección IP.  
Puedes acceder a través de SSH desde tu **CMD**

Por defecto en los droplets el usuario es root, y el puerto inicial es el 22.

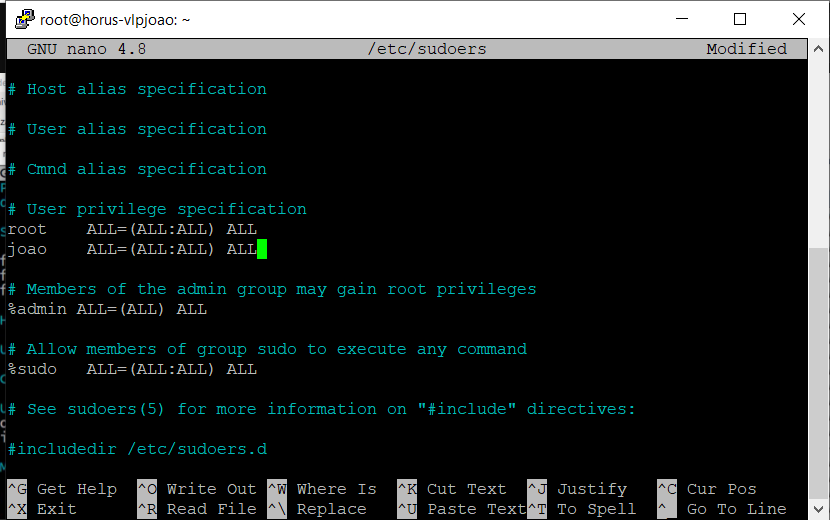
Cmd: ssh <user>@<ip>

Después de conectarte, te pedirá confirmación y la contraseña de ese droplet(maquina virtual)

**Opciones de la cmd:**

* -p: especificamos un puerto ej: ssh <user>@<ip> -p 22

# Manejo de usuarios

* Cambio de contraseña:
  + <passwd>: Con este commando podr[as cambiarle la contrasena al usuario actual
* Anadir usuario:
  + <adduser> usuario: Con esto anades un usuario, después te pedirá el password. Al crearsea automáticamente se le asignara un grupo.
* Otorgar privigelios: Modificaremos un archivo para asignarlos desde ahí.
  + Nano /etc/sudoers
  + 

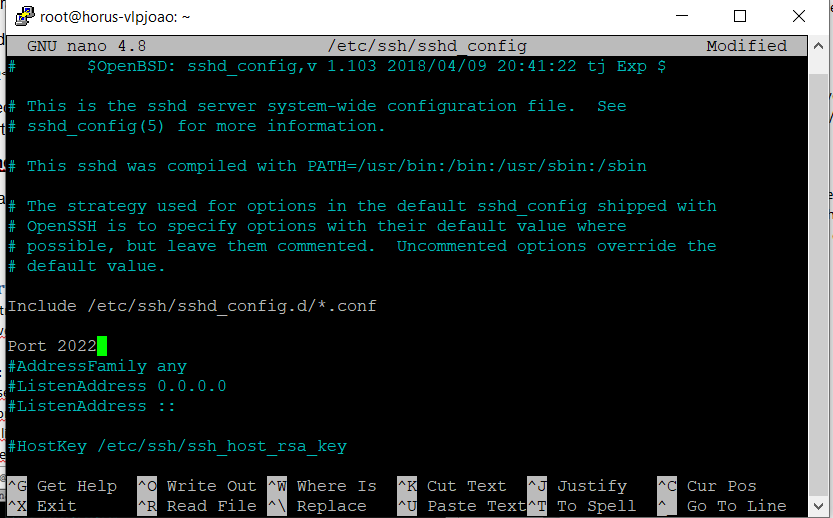
# Cambio de puerto SSH

Es recomendable cambiar el puerto por defecto del VPS ya que se puede acceder a este de forma fácil.

Modificar el sshd\_config

* Modificar el archivo SSH del sistema:
  + Nano /etc/ssh/sshd\_config

Descomentamos el código del puerto y ponemos el puerto que queramos



* Reseteamos el ssh
  + Service ssh restart (Ya que estamos en el usuario root, no hace falta usar SUDO)

# Configurar firewall

Los firewalls sirven para filtrar todo el tráfico de la red y protegerte de accesos no autorizados.

* Verificar el estado del firewall
  + Ufw status
* Activar los firewall
  + Ufw enable
    - Aquí decidimos si vamos a activarlo para todas las conexiones ssh actuales
* Habilitar un puerto
  + Ufw allow servicio/puerto
    - Ej: ufw allow 2022
* Enumerar los puertos:
  + Ufw status numbered
    - Enumera los puertos en orden
* Denegar un puerto
  + Ufw deny <puerto>
* Borrar una regla
  + Ufw delete <numero\_regla(se consigue con numbered)>
* Desactivar el firewall
  + Ufw disabled
* Resetear reglas
  + Ufw restart

# Clonacion del proyecto

En la carpeta del usuario del VPS, clonaremos el proyecto.

En el usuario root puede ser en la carpeta ROOT

EN caso de github, recordemos utilizar Perrsonal Access Tokens para poder acceder al repositorio. Esta PAT lo usaremos pcomo contraseña del usuario

<https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>

* Instalamos virtualenv
  + Sudo apt-get install virtualenv
* Creamos el entorno virtual
  + Virtualenv <nombre> -ppython3 (para que se cree con Python 3)
* Activar entorno virtual
  + Source <path/bin/>activate
    - EJ: source env/bin/actívate
* Instalamos las dependencias
  + Pip install requirements.txt
* Instalar los complementos de weasyprint
  + <https://doc.courtbouillon.org/weasyprint/stable/first_steps.html#ubuntu-20-04>

# Ejecutar proyecto

* Ejecutamos el proyecto en el VPS
  + python manage.py runserver 0:8000(<ip:puerto>)
    - definimos la ip y el puerto en el que se va a ejecutar

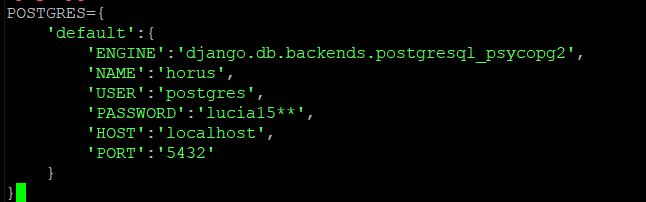
**Recuerda abrir el puerto en el que vas a ejecutar el sitio**

* Para acceder al sistema desde la IP del VPS colocamos la ip con el puerto
  + <http://192.241.138.220:8000/>

# Configuracion de PostgreSQL

https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-20-04

* Actualizamos el servidor
  + Sudo apt update
* Instalamos los paquetes de postgre
  + sudo apt install postgresql postgresql-contrib
* Accedemos a la cuenta postgres
  + sudo -u postgres psql
* Cambiamos la contraseña por seguridad
  + \password
* Cambiar zona horaria
  + ALTER ROLE postgres SET timezone TO ‘UTC’;
* Crear base de datos
  + Create database <nombre>
* Agregamos la configuracion al archive db.py para llamarlo en el settings
  + Locahost hace referencia a la ip del servidor



Ya después hacemos las migraciones y las cosas necesarias

Consejos:

* Es recomendable cambiar el formato de codicacion de caracteres de postgres a UTF8
* Es recomendable cambiar la zona horaria a la que tu desees, se recomienda solo UTc

# Creacion de un archivo BASH para correr servidor django

<https://bioinf.comav.upv.es/courses/unix/scripts_bash.html>

BASH es un lenguaje de comandos, que básicamente se programa en un fichero de texto que contiene un conjunto de comandos y ordenes que son interpretables por la SHELL

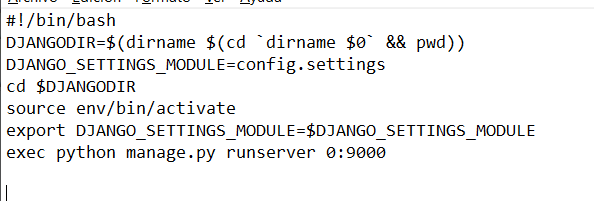
* Creamos una carpeta para los archivos del deploy
  + Mkdir deploy
* Creamos un archivo sh que contendrá los comandos para ejecutar el servidor
  + Nano server.sh
  + Para ejecutarlo usamos el ./<archivo>.sh

En la cabecera del archivo bash debemos escribir #!/bin/bash

Consejo:

* Aprender sobre los permisos en Linux para poder ejecutar los archivos
  + <https://www.solvetic.com/tutoriales/article/1458-entender-los-permisos-linux-chmod/>

Archivo final para corer el servidor de forma facil

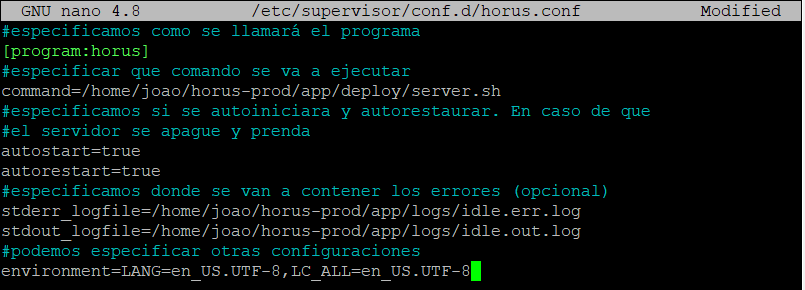


# Despliegue con Supervisor

Supervisor es un gestor de procesos de Linux. En pocas palabras será el que se encargará de mantener el servicio de Django corriendo en el servidor.

<https://www.digitalocean.com/community/tutorials/how-to-install-and-manage-supervisor-on-ubuntu-and-debian-vps>

* Instalamos supervisor
  + sudo apt install supervisor
* Creamos un archivo dentro de la carpeta de supervisor, recomendado usar el nombre de la aplicación
  + sudo nano /etc/supervisor/conf.d/horus.conf
* Especificamos la configuración



* Para ejecutar la tarea debemos ejecutar el siguiente comando
  + **Sudo supervisorctl reread**
    - Este sirve para verificar si un nuevo servicio se encuentra disponible
  + **Sudo supervisorctl update**
    - Para anadir esa tarea para que el supervisor lo administre. Tambien para que el servicio se actualice si se hizo un cambio en las configuraciones
  + **Sudo supervisorctl status**
    - Para saber si se esta ejecutando, te muestra todos los procesos
  + **Sudo supervisorctl stop <servicio>/all**
    - Para detener un servicio o todas
  + **Sudo supervisorctl start <servicio>**
    - Para iniciar un servicio
  + **Sudo supervisorctl restart <servicio>/all**
    - Para reiniciarlo, un servicio en especifico o todos
  + Si no quieres el servicio corra mas, solo borra el archivo .conf y updatea el supervisor.
* LOGS
  + En el archivo err.log es donde se guarda lo que se esta ejecutando en el servicio. Para ver en tiempo real lo que esta en ese archivo usa
    - * Tail –f err.log

# Configurar dominio y DNS

<https://linube.com/ayuda/articulo/193/tipos-de-registro-dns>

Verificamos el tipo de DNS que queremos, ya que cada uno cumple una función diferente

Al registrar la configuración del dominio, escogemos de que tipo será(verificar los tipos de registro DNS), si accederemos al sitio con www o poder usar @, la ip, y el TTL que no es mas que cada cuanto tiempo se reflejarán los cambios del sitio.

# 

# Configurando WebServer GUNICORN

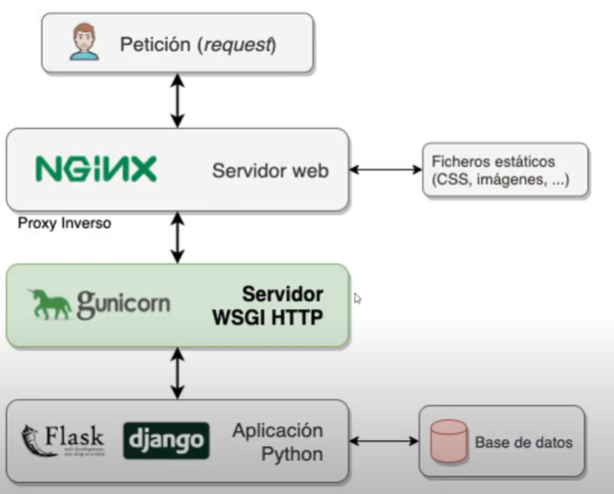
https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu-22-04

<https://docs.djangoproject.com/en/3.2/howto/deployment/wsgi/gunicorn/>

https://docs.gunicorn.org/en/latest/run.html

Los webserver se encargan de manejar todo el contenido estatico de la pagina web.

Esto lo hace de manera mas rápida cuando el cliente haga peticiones.

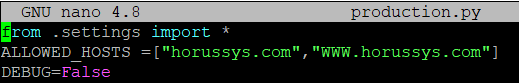


Ya que esto es lo recomendado, podemos tener en cuenta que tener el servidor corriendo con el servidor de django no es lo más recomendable. Ya que un webserver entrega de mejor manera los archivos estáticos.

Estos ofrecen una gestión de peticiones simultáneas más eficientes.

Para que exista una comunicación entre el webserver y la aplicación, debe existir un **SERVIDOR WSGI HTTP**, que permite conectar por medio de **sockets** la aplicación y el webserver

Pasos:

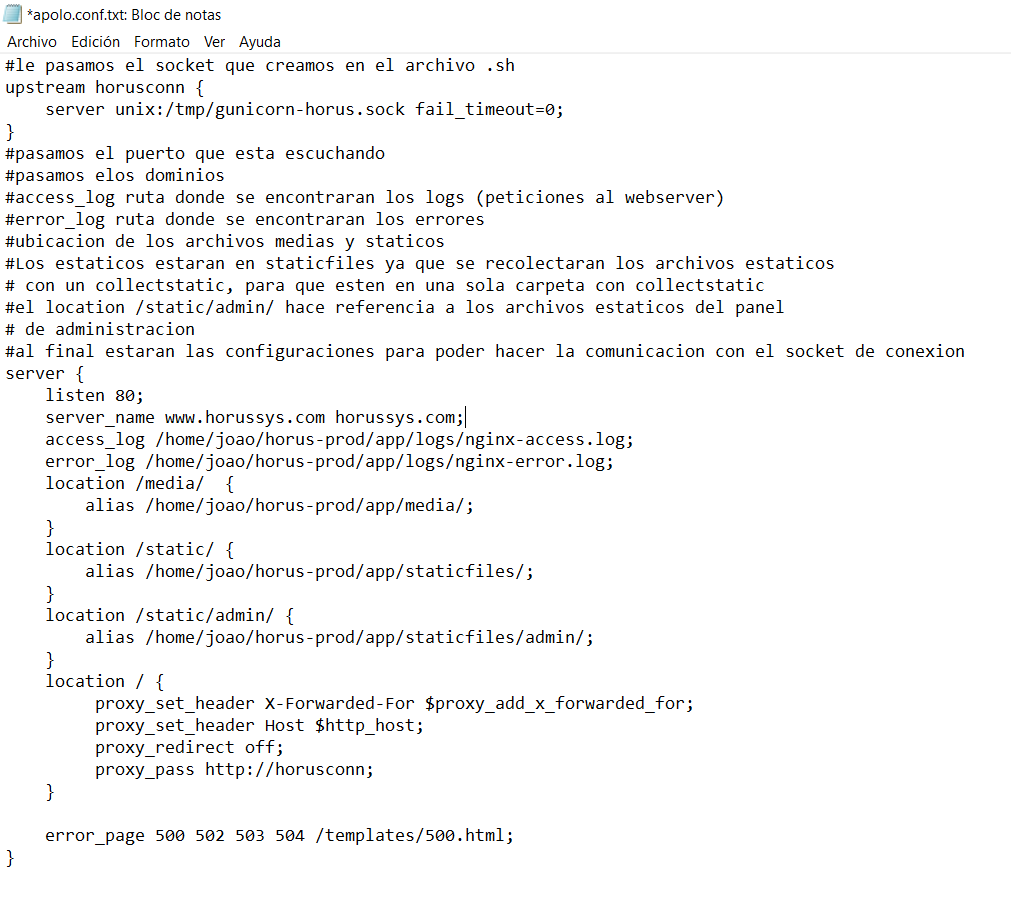
* Instalar GUNICORN
  + Pip install gunicorn
* Creamos un archivo llamado Production que contenga la configuración que necesita el servidor para no tener que modificar el WSGI.
  + 
  + En el WSGI en vez de usar config.settings pondrás config.production.
  + 
* Ejecutamos django con gunicorn con el archivo wsgi.
  + Gunicorn config.wsgi (para ejecutar esto debemos apagar el servicio de django)
* Crear archivo BASH para que el supervisor maneje el nginx automáticamente
  + Creamos un archivo gunicorn.sh en la carpeta deploy
  + Le pasamos la configuración necesaria para la ejecución
  + Le damos permisos al archivo
    - Sudo chmod 7777 gunicorn.sh
  + Probamos el archivo ejecutando, para ver si funciona correctamente
    - Bash gunicorn.sh

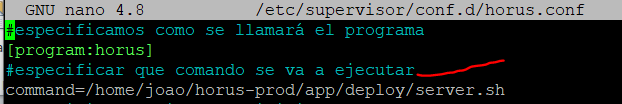
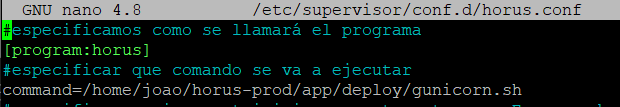
# Configuración del Webserver Nginx

http://nginx.org/en/docs/

<https://www.digitalocean.com/community/tutorials/como-instalar-nginx-en-ubuntu-18-04-es>

* Instalamos nginx
  + Sudo apt install nginx
* Configuramos reglas del firewall y anadimos nginx
  + Sudo ufw allow ‘Nginx HTTP’
* Verificamos si se esta ejecutando nginx
  + Systemctl status nginx
* Ejecutamos el collectstatic pero con la opción –link, esto para que no se copien los archivos sino que solo se cree una referencia simbólica a los archivos estáticos
  + Python manage.py collectstatic –link
    - Creamos esto ya que en la configuración del nginx hacemos referencia a los staticfiles
* Creamos el archivo .conf en una carpeta de nginx
  + sudo nano /etc/nginx/sites-available/horus.conf
  + Le pasamos la configuración necesaria



* Creamos una referencia (link) al archivo .conf en la carpeta enable de nginx
  + sudo ln -s /etc/nginx/sites-available/horus.conf /etc/nginx/sites-enabled/
    - Esto creara un archivo simbolico desde tal archivo hasta la carpeta que especifiques después
* Verificamos que todo este correcto con
  + Sudo nginx –t (para hacer test)
* Modificamos el archivo de supervisor
  + Anteriormente se ejecutaba el archivo server.sh, ahora se debe ejecutar gunicorn.sh
  + 
  + 
* Actualizamos supervisor
  + Sudo supervisorctl update
* Reiniciamos el Nginx
  + Sudo systemctl restart nginx

Ahora si el sistema estará apto para abrirlo. Podremos eliminar el puerto que usábamos para el servidor de django del firewall

# Instalacion de certificado SSL

<https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-20-04-es>

* Instalar cerbot (se encargara de obtener los cerfiticados)
  + sudo apt install certbot python3-certbot-nginx
* Verificamos la configuracion de nginx (horus.conf) en /etc/nginx/sites-available/ Para ver si tiene asignado un dominio en server\_name
  + Sudo nano /etc/nginx/sites-available/horus.conf
* Verificamos las reglas de seguridad(firewall) para nginx, que tenga nginx activado
  + Sudo ufw status
* Obtenemos el certificado SSL
  + sudo certbot --nginx -d example.com -d [www.example.com](http://www.example.com)
  + Esto te pedirá el correo si es primera vez para verificar que eres el dueño del dominio
  + Seleccionar la opción de REDIRECT, ya que así se elimina el trafico HTTP

Si quieres verificar la instalación del certificado busca el archivo .conf del proyecto de nginx ya que ahí se incluirán automáticamente las certificaciones SSL.

* Debemos habilitar el perfil Nginx Full y eliminar el perfil redundante Nginx HTTP
  + sudo ufw allow 'Nginx Full'
  + sudo ufw delete allow 'Nginx HTTP'

# Consejos

* Si no eres el usuario root, debes usar **SUDO** para los comandos que vayas a ejecutar, porque sinoi te dira que no tienes los permisos
* Crear usuarios nuevos en el vps, ya que no es recomendable usar el usuario root, yua que si se daña no habrá forma de recuperarlo, en cambio los usuarios secundarios se pueden eliminar y crear fácilmente.
* Si defines un puerto de ejecución eal momento de correr el servidor loca, debes abrir ese puerto en el firewall

DNS Checker para ver como se va desplegando tu sitio a través del mundo.

https://dnschecker.org/#A/horussys.com