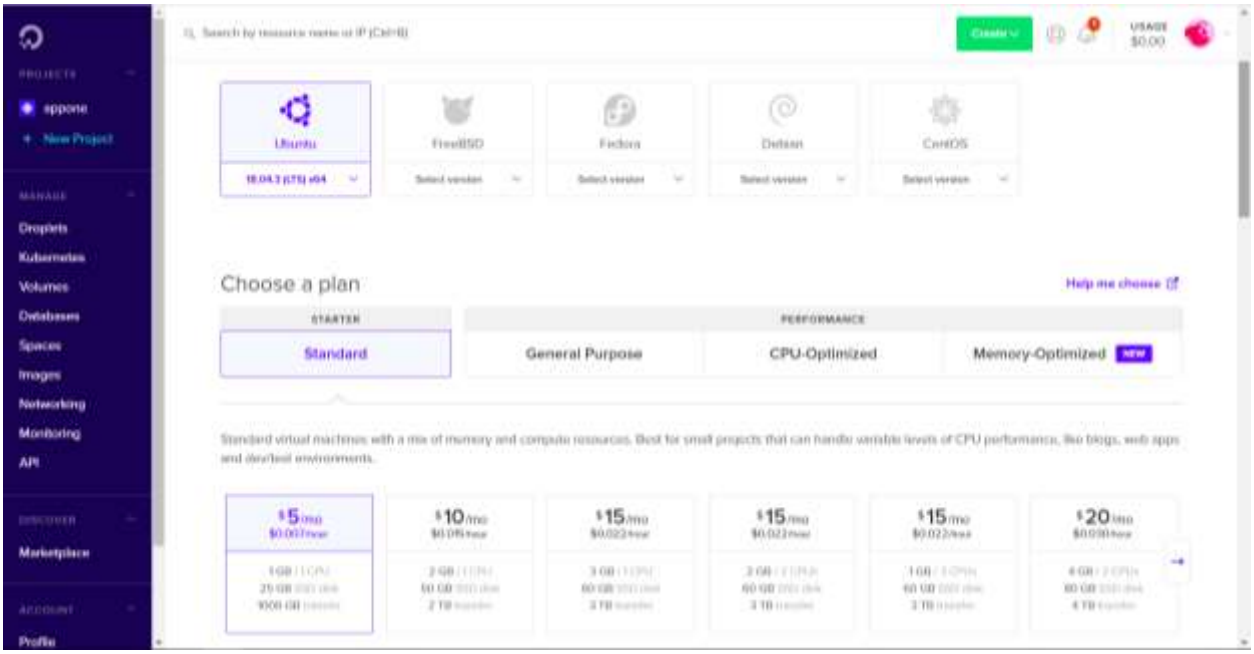
Configurar servidor VPS

Vamos a publicar nuestro proyecto en un vps, un vps es un servidor virtual privado.

Para este ejemplo vamos a utilizar digitalocean.com, pero tu puedes adquirir tu VPS en cual otro proveedor. La configuración del vps para desplegar nuestro proyecto laravel va a ser la misma sin importar el proveedor de vps.

Vamos primero a registrarnos en digitalocean, y crear nuestro vps, al parecer al registrarse nos están regalando 50 dólares de crédito.

Vamos a crear un nuevo servidor en este caso se llama droplet es una máquina virtual, a la que tendrás acceso por SSH (vía consola).



Ahora debemos tener las credenciales para acceder a nuestro servidor vps a traves de ssh. En digitaOcean las credenciales se envían al correo.

Para acceder podemos utilizar VNC viewer.

```
ssh root@ 157.245.175.1
```

La clave 42d3cfb84a9d5092f779da2664

Ingresamos nuevamente 42d3cfb84a9d5092f779da2664.

Ahora si ingresamos una nueva clave.

Ahora actualizamos el registro de dependencias.

```
sudo apt-get update
```

Como vamos a utilizar git vamos a instalarlo.

```
sudo apt-get install
```

```
git
```
Ahora Podemos ver la versión de git.

```
git –version
```

Vamos agregar un nuevo usuario en este servidor para no utilizar el usuario root.

```
adduser jcarlosad7
```

Ingresamos la contraseña (dos veces) ahora ingresamos los datos del usuario

Vamos agregar a este usuario al grupo de los super usuarios.

```
usermod -aG sudo jcarlosad7
```

Ahora probamos acceder con ese usuario

```
ssh jcarlosad7@157.245.175.1
```

Instalar nginx

NGINX "engine-ex" es un famoso software de servidor web de código abierto. En su versión inicial, funcionaba en servidores web HTTP. Sin embargo, hoy en día también sirve como proxy inverso, balanceador de carga HTTP y proxy de correo electrónico para IMAP, POP3 y SMTP.

Debido a su excelente capacidad para manejar muchas conexiones y a su velocidad, muchos sitios web de alto tráfico usan el servicio de NGINX. Algunos de estos gigantes del internet son Google, Netflix, Adobe, Cloudflare, WordPress.com y muchos más.

Accedemos a nuestra maquina virtual y para instalar nginx ingresamos el comando.

```
sudo apt-get install nginx
```

Una vez que el proceso termine accedemos al ip y tendremos funcionando nuestro servidor web.

Vamos ahora a verificar el status del firewall.

```
sudo ufw status
Status: inactive
```

Vemos que esta inactivo, vamos habilitarlo.

```
sudo ufw enable
```

Vemos nuevamente el status

```
sudo ufw status
```

Ahora nos aparece activo.

Si ahora accedemos nuevamente a la ip con el navegador tendremos un error. Vamos a solucionar agregamos permisos de nginx al firewall.

```
sudo ufw allow 'Nginx HTTP'
```

Ahora se ha agregado la regla en el firewall, si vamos nuevamente al navegador veremos que ya tenemos acceso.

Como hemos habilitado el firewall vamos a ingresar el siguiente comando para que el firewall permita las conexiones ssh, es decir que el firewall permita las conexiones remotas ssh como lo estamos haciendo hasta ahora.

```
sudo ufw allow ssh
```

Instalar y configurar mysql

En el desarrollo del sistema hemos utilizado mysql como gestor de base de datos, aquí en producción vamos a utilizar también mysql. Debemos entonces instalar mysql en nuestro servidor.

Para instalar mysql vamos a utilizar el siguiente comando.

```
sudo apt-get install mysql-server
```

Vamos acceder a nuestro gestor de base de datos mysql

```
sudo mysql
```

Ahora vamos a indicar una contraseña a nuestro usuario root de mysql, ya que no es para nada recomendable dejarlo sin contraseña.

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'administrador';
```

Con este comando estamos alterando al usuario root que está dentro de localhost para identificar con el password nativo y le indicamos el password.

Ahora refrescamos los privilegios.

```
FLUSH PRIVILEGES;
```

Ahora si vamos a salir del gestor de base de datos y vamos acceder nuevamente.

```
Exit
```

Y tratamos de acceder ahora con la clave

```
mysql -u root -p
```

Ahora si vamos a crear la base de datos que utilizaremos en nuestro proyecto de laravel.

```
create database dbsistema;
```

Ahora es recomendable crear un usuario para la administración de solo esa base de datos creada. Vamos a ejecutar el siguiente comando.

```
create user 'userdb'@'localhost' IDENTIFIED BY 'userdb';
```

Estamos creando el usuario userdb que corresponde a localhost y esta identificado con la contraseña userdb.

Vamos ahora asignarle permisos para trabajar con la base de datos dbsistema, ejecutamos el siguiente comando.

```
grant all on dbsistema.* to 'userdb'@'localhost';
```

Y por último refrescamos los privilegios.

```
flush privileges;
```

Ahora si vamos acceder con el usuario userdb.

```
mysql -u userdb -p
```

Vamos a revisar las bases de datos a la que tiene acceso.

```
show databases;
```

Ahora si revisamos que tiene acceso a la base de datos dbsistema, es decir puedo ejecutar comandos mysql en la base de datos dbsistema.

Instalando php y sus dependencias

Vamos ahora a instalar algunas dependencias necesarias para poder trabajar con PHP.

Vamos a instalar php en sus versiones actuales usando ppa.

```
sudo add-apt-repository ppa:ondrej/php
```

PPA (Perosonal Package Archive) Archivos de Paquetes Personales permite a los desarrolladores distribuir software y sus respectivas actualizaciones en forma directa con los usuarios de Ubuntu sin tener que esperar que se actualicen los propios repositorios de Ubuntu.

PHP 7.3 es la última versión estable de PHP. Gracias a Ondřej Surý por mantener el PPA de la mayoría de las populares versiones de PHP en launchpad.

Vamos actualizar primero.

```
Sudo apt-get update
```

Vamos a instalar también las siguientes dependencias.

```
sudo apt-get install php7.3-cli php7.3-fpm php7.3-curl php7.3-gd php7.3-mysql php7.3-mbstring php7.3-xml zip unzip
```

Vamos a comprobar si ya tenemos instalado php, ejecutamos el siguiente comando

```
php -v
```

Vamos a instalar también composer.

```
sudo apt-get install composer
```

Configurar nginx

Vamos ahora a configurar nginx para que pueda ejecutar archivos php. Vamos primero a ver el estado de nuestro servicio fpm.

```
sudo service php7.2-fpm status
```

El procesador FastCGI es el que nos va a permitir poder ejecutar php utilizando nginx.

Vamos a ver los procesos involucrados al momento de trabajar con php. Utilizamos el siguiente comando.

```
sudo ps aux | grep php
```

Ahora vamos acceder al siguiente archivo utilizando cat. cat es leer datos de archivos y mostrar sus contenidos.

```
cat include=/etc/php/7.2/fpm/pool.d/*.conf
```

Ahora en la última línea vemos que tenemos el include.

```
include=/etc/php/7.2/fpm/pool.d/*.conf
```

Vamos a copiar la siguiente ruta.

```
/etc/php/7.2/fpm/pool.d/*
```

Y vamos a ver el listado de archivos del directorio de esa ruta

```
ls /etc/php/7.2/fpm/pool.d/
```

Nos aparece

```
www.conf
```

Vamos a visualizar ese archivo de configuración de php utilizando cat.

```
cat /etc/php/7.2/fpm/pool.d/www.conf
```

Pero para encontrar mejor la propiedad a modificar vamos a utilizar el editor nano.

```
Sudo nano /etc/php/7.2/fpm/pool.d/www.conf
```

Vamos a buscar la propiedad listen.

Control + w y escribimos listen y para repetir la búsqueda **Alt + w**.

Copiamos esta ruta.

```
/run/php/php7.2-fpm.sock
```
de

```
listen = /run/php/php7.2-fpm.sock
```
Vamos ahora a editar el archivo de host virtual.

```
sudo nano /etc/nginx/sites-available/default
```

Vamos a descomentar esto, que quede de la siguiente manera.

```
location ~ \.php$ {
        include snippets/fastcgi-php.conf;
    #
    #   # With php-fpm (or other unix sockets):
        fastcgi_pass unix:/var/run/php/php7.0-fpm.sock;
    #   # With php-cgi (or other tcp sockets):
    #   fastcgi_pass 127.0.0.1:9000;
    }
```

Y vamos a modificar también por la ruta que copiamos.

```
location ~ \.php$ {
        include snippets/fastcgi-php.conf;
    #
    #   # With php-fpm (or other unix sockets):
        fastcgi_pass unix:/run/php/php7.2-fpm.sock;
    #   # With php-cgi (or other tcp sockets):
    #   fastcgi_pass 127.0.0.1:9000;
    }
```

Control + x para salir y enter para guardar los cambios.

Como hemos realizado cambios vamos actualizar nuestro servidor nginx.

```
sudo service nginx reload
```
Vamos a reiniciar el servicio fpm

```
sudo service php7.2-fpm restart
```
Si revisamos en nuestro navegador debe seguir funcionando nginx.

Los archivos que se muestran en el servidor nginx se almacenan en la carpeta var/www/html, vamos a revisar los archivos

```
ls /var/www/html
```

Ese html es el index que se esta mostrando cuando accedemos a nuestro servidor web con el navegador.

Vamos a eliminar ese archivo que nos aparece. Primero accedemos al directorio.

```
cd /var/www/html
```

Ahora eliminamos el archivo.

```
sudo rm index.nginx-debian.html
```

Revisamos con ls y ya no lo tenemos, si vemos en el navegador tenemos un error.

Vamos a crear un directorio

```
sudo mkdir public
```

vamos acceder a esa carpeta

```
cd public
```

Vamos a crear dentro un archivo php para que se visualice, recordemos que el directorio público de un proyecto de laravel se encuentra en la carpeta public, así que vamos ya a configurar nuestro archivo index dentro de este directorio.

```
sudo touch index.php
```

Ahora editamos ese archivo.

```
sudo nano index.php
```

```
  ?php
echo phpinfo();
?>
```

Si actualizamos nuestro navegador esto no va a funcionar, debemos editar nuevamente nuestro archivo de host virtual para que acepte como archivo de inicio un archivo index.php.

```
sudo nano /etc/nginx/sites-available/default
```

```
root /var/www/html/public;

        # Add index.php to the list if you are using PHP
index index.php index.html index.htm index.nginx-debian.html;
```

Como hemos realizado cambios vamos actualizar nuestro servidor nginx.

```
sudo service nginx reload
```

Vamos a reiniciar el servicio fpm

```
sudo service php7.2-fpm restart
```

Ahora revisamos el navegador y vemos que ya se muestra el resultado de la función phpinfo, de esta manera ya tenemos configurado nginx para que pueda ejecutar archivo php y ya tenemos configurado el directorio raíz para ejecutar nuestro proyecto laravel.

Subir Proyecto a repositorio

Para subir nuestro proyecto a nuestro servidor primero lo vamos a subir en un repositorio, podemos utilizar cualquier repositorio como gitlab, github, voy hacerlo utilizando github. Vamos a crear un nuevo repositorio.

```
echo "# adblog" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:jcarlosad7/adblog.git
git push -u origin master
```

Ahora si vamos a abrir nuestra terminal y creamos el nuevo repositorio en el proyecto ejecutamos un git init

```
git init
```

Ahora ejecutamos para agregar el remoto a este repositorio

```
git remote add origin git@github.com:jcarlosad7/adblog.git
```

Ahora agregamos un commit

```
Git add .
Git commit -m “first commit”
```

Ahora si subimos nuestro Proyecto a la rama master

```
git push -u origin master
```

Ahora ya tenemos nuestro repositorio publicado de manera correcta.

Configurar proyecto laravel en el servidor

Para subir ahora nuestro proyecto vamos a ubicarse en nuestro directorio /var/www/html y vamos a subir ahí nuestro proyecto.

Hacemos un ls y vemos que tenemos ahí nuestro directorio public, vamos a eliminar ese directorio que hicimos el video pasado.

```
sudo rm -r public
```

Vamos a clonar nuestro repositorio. Le indico con el punto que me lo clone en este directorio.

```
sudo git clone https://github.com/jcarlosad7/laravel19.git .
```

Revisamos la lista de archivo con ls -al. Vemos que los archivos pertenecen al usuario root, vamos asignar como propietario al usuario jcarlosad7.

```
sudo chown -R $USER:$USER /var/www/html
```

Ahora ejecutamos ls -al vemos que pertenecen al usuario jcarlosad7.
Ahora vamos a ejecutar composer install para instalar todas las librerías necesarias.
Si tenemos un error debemos solucionarlo.

Vamos ahora a crear un archivo .env, podemos copiar el archivo .en.example

```
cp .env.example .env
```

Vamos a ver el contenido de este archivo .env

```
cat .env
```

Y vemos que debemos generar la clave

```
php artisan key:generate
```

Ahora vemos la clave

```
cat .env
```

Ahora editamos el .env

```
sudo nano .env
```

```
Vamos a poner nuestro proyecto en producción
APP_ENV=production
```

Vamos a dejar nuestro debug para ver algún error mientras desplegamos nuestro proyecto, después ya lo deshabilitaremos.
En la app_url vamos a poner la ip.

```
APP_URL=http://157.245.175.1/
```

Cambiamos tambien las credenciales de acceso a la base de datos.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=dbsistema
DB_USERNAME=userdb
DB_PASSWORD=userdb
```

Debemos poner también las credenciales de envio de correo.

Ahora si accedemos http://157.245.175.1/
Vemos que tenemos un error porque no le hemos dado permisos a la carpeta storage.
Vemos el propietario de la carpeta storage ls-al
Vamos asignar los permisos con la siguiente.

```
sudo chown -R www-data:www-data storage/
```

Si ahora regresamos vemos que ya estamos accediendo, pero no tenemos la tabla entradas. Voy acceder al /login y vemos que tenemos un error 404. Y esto porque no le hemos asignado los permisos a nginx.
Vamos a modificar ya que todas las url están redireccionando al 404

```
sudo nano /etc/nginx/sites-available/default
```

```
location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        #try_files $uri $uri/ =404;
        try_files $uri $uri/ /index.php$is_args$args;
    }
```

De esta manera redirigemos teniendo en cuenta los argumentos de la url.
Ahora reiniciamos
Como hemos realizado cambios vamos actualizar nuestro servidor nginx.

```
sudo service nginx reload
```

Ahora si ya funcionan todas las rutas.

Vamos a ejecutar las migraciones.

```
php artisan migrate
```

Y si revisamos ahora ya tenemos nuestro proyecto de laravel funcionando correctamente.
Vamos a registrarnos.