

Creando Mi Portafolio

Redactado por:

Guillermo Garcia Canul

En esta guía veremos los pasos básicos para generar nuestro propio portafolio y como poder darle un host a este mismo haciendo uso de herramientas cómo:

1. Digital Ocean
2. Name Cheap

Es importante mencionar que aquí solamente veremos con profundidad el proceso para poder hacer el despliegue de nuestra aplicación, en términos de programación no vamos a profundizar, pues esta guía busca ser una base para aplicar nuestros conocimientos, sin embargo siéntete libre para tomar inspiración de mi propio portafolio, es público.

Mi Portafolio

Evita Plagios

Aunque el portafolio es público, recuerda que cada uno de nosotros tiene sus propios proyectos, y este portafolio únicamente es una base para que puedas crear el tuyo.

Una vez que hemos hablado de manera general, es importante que sepamos con que framework u herramienta vamos a desarrollar nuestro portafolio, en mi caso he seleccionado hacerlo con Laravel.

Este contiene 2 vistas principales, la primera que es la vista de inicio y una secundaria que es para un formulario de contacto.

Creando Mi Droplet En Digital Ocean











Vamos a empezar viendo cómo vamos a crear nuestro droplet, el cuál nos va a servir de manera directa para poder darle el host a nuestra aplicación.

Create Droplets

[Learn](#)

Droplets are virtual machines that anyone can setup in seconds. You can use droplets, either standalone or as part of a larger, cloud based infrastructure.

Choose Region

 New York	 San Francisco	 Amsterdam
 Singapore	 London	 Frankfurt
 Toronto	 Bangalore	 Sydney
 Atlanta		

Datacenter

San Francisco • Datacenter 2 • SFO2

💡 **Tip:** Select the datacenter closest to you or your users

[Dismiss](#)

Avoid any potential latency by selecting a region closest to you - a region is a geographic area where we have one or more datacenters.

VPC Network - default-sfo2

DEFAULT

All resources created in this datacenter will be members of the same VPC network. They can communicate securely over their Private IP addresses.

Choose an image

OS Marketplace (282) Custom images

 Ubuntu	 Fedora	 Debian	 CentOS	 AlmaLinux	 Rocky Linux
---	---	---	---	--	--

Version

25.04 x64

IMG_1.1_Region_And_Image

Seleccionando Región e Imagen

En la pantalla de creación del **Droplet**, DigitalOcean te pedirá:

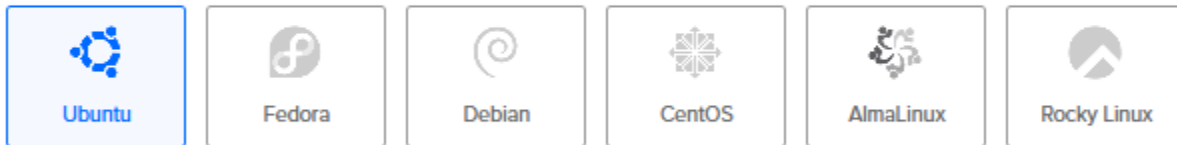
1. **Choose Region** → selecciona el datacenter más cercano a ti o a tus usuarios.
En mi caso he seleccionado **San Francisco**.
Esto reduce la latencia al momento de acceder a tu aplicación.
2. **Choose an Image (Sistema Operativo)** → para Laravel lo más recomendado es **Ubuntu LTS**.
Ejemplo: Ubuntu 22.04 x64 .

 **Consideraciones**

Evita usar versiones demasiado nuevas (como 25.04) ya que pueden tener menos soporte en tutoriales o paquetes

Choose an image

OS Marketplace (282) Custom images



Version

24.04 (LTS) x64

IMG_1.2_Choose_Region_Imagen

Seleccionando Plan

El siguiente paso será elegir el plan de recursos para tu Droplet:

- **Basic Plan** → es más que suficiente para un portafolio.
- Recomendado: **1 GB RAM, 1 vCPU, 25 GB SSD**.
- Si tu aplicación crece, puedes escalar en cualquier momento.

Choose Size

Need help picking a plan? [Help me choose](#)

Droplet Type

SHARED CPU	DEDICATED CPU			
Basic (Plan selected)	General Purpose	CPU-Optimized	Memory-Optimized	Storage-Optimized

Basic virtual machines with a mix of memory and compute resources. Best for small projects that can handle variable levels of CPU performance, like blogs, web apps and dev/test environments.

CPU options

☒ Regular
Disk type: SSD

☐ Premium Intel
Disk: NVMe SSD

☐ Premium AMD
Disk: NVMe SSD

\$6/mo \$0.009/hour	\$12/mo \$0.018/hour	\$18/mo \$0.027/hour	\$24/mo \$0.036/hour	\$48/mo \$0.071/hour	\$96/mo \$0.143/hour
1 GB / 1 CPU 25 GB SSD Disk 1000 GB transfer	2 GB / 1 CPU 50 GB SSD Disk 2 TB transfer	2 GB / 2 CPUs 60 GB SSD Disk 3 TB transfer	4 GB / 2 CPUs 80 GB SSD Disk 4 TB transfer	8 GB / 4 CPUs 160 GB SSD Disk 5 TB transfer	16 GB / 8 CPUs 320 GB SSD Disk 6 TB transfer



[Show all plans](#)

IMG_1.3_Choose_Plan

Servicios Adicionales

En caso de que nuestro portafolio contenga una base de datos u contenga lógica de guardado dentro de la aplicación podremos seleccionar si queremos agregar volúmenes o hacer respaldos de manera periódica, en mi caso tengo contratado un servicio de correo, por lo que no necesito guardar información en la aplicación, pues gracias al formulario todo se manda a un correo especial.

Additional Storage

Add Volume

Need more disk space? Add a volume with no manual setup.
Block storage volumes add extra disk space. We automatically format and mount your volume so it's available as soon as your Droplet is, and you can move volumes seamlessly between Droplets at any time. Think of it like a flash drive for your VM.

Backups

☐ **Enable automated backup plan**
Automatically take backups at the time you specify

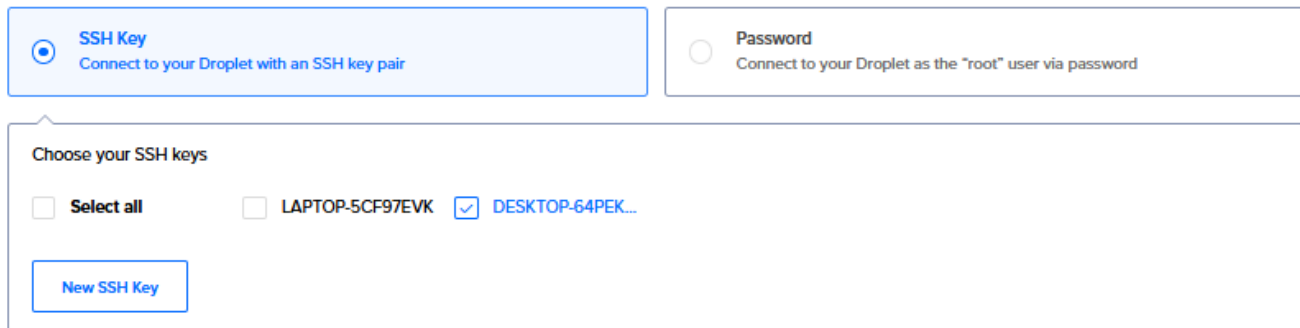
IMG_1.4_Additional_Services

Autenticación

En este paso definimos **cómo accederemos al servidor** una vez creado el Droplet. DigitalOcean ofrece dos opciones principales:

1. **Contraseña**
2. **SSH Keys**

Choose Authentication Method ?



The screenshot shows the 'Choose Authentication Method' section of the DigitalOcean console. It features two radio button options: 'SSH Key' (selected) and 'Password'. Below the 'SSH Key' option, there is a section titled 'Choose your SSH keys' containing three checkboxes: 'Select all', 'LAPTOP-5CF97EVK', and 'DESKTOP-64PEK...' (which is checked). A 'New SSH Key' button is located at the bottom left of this section.

IMG_1.5_Choose_Authentication

En mi caso he decidido generar la llave de seguridad desde mi máquina local, pues es más seguro y así solamente yo que tengo mi IP soy capaz de acceder a este servidor.

⚠ Generando mi SSH

Si queremos generar nuestra llave bastará con abrir la terminal en modo administrador y lanzar el comando `ssh-keygen -t ed25519 -C "tu_correo@example.com"`, esto nos genera dos llaves, una privada y una pública, la segunda la subiremos a DO y la primera la vamos a almacenar

Detalles Finales

Al finalizar de configurar la información básica de nuestro Droplet podremos seleccionar la cantidad de droplets que queremos desplegar, el nombre de identificación, la etiqueta y el proyecto al que pertenecerá/pertenecieran.

Finalize Details

Quantity

Deploy multiple Droplets with the same configuration.

—	1 Droplet	+
---	-----------	---

Hostname

Give your Droplets an identifying name you will remember them by.

Tags

Portfolio

Type tags here

This project has been selected
as you only have one project

Project

 first-project

▼

IMG_1.6_Finalize_Details

Conectándonos A Nuestro Droplet

Ahora para que podamos hacer la conexión a nuestro Droplet, bastará con que hagamos:

```
ssh root@ip_del_droplet
```

Si nos conectamos de manera correcta veremos que nos aparecerá información de nuestro droplet y a la vez el acceso al root del servidor

```
Expanded Security Maintenance for Applications is not enabled.

26 updates can be applied immediately.
21 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@My-Portfolio:~#
```

IMG_2_Root

Configurando Nuestro Servidor

Una vez que estemos dentro de nuestro droplet lo que vamos a hacer será actualizar los paquetes del servidor, para eso haremos uso del comando:

```
apt update
apt upgrade -y
```

Con estos comandos actualizamos paquetes de seguridad y librerías.

```
Pending kernel upgrade!
Running kernel version:
 6.8.0-71-generic
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.8.0-79-generic.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...
systemctl restart packagekit.service

Service restarts being deferred:
systemctl restart unattended-upgrades.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@My-Portfolio:~#
IMG_3_Config_Our_Server
```

Cómo siguiente paso deberemos instalar nuestras dependencias básicas, esto lo haremos incluso antes de instalar Laravel, php y Composer, para eso bastará con que hagamos

```
apt install -y git unzip curl software-properties-common
```

```
root@My-Portfolio:~# apt install -y git unzip curl software-properties-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.3).
git set to manually installed.
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
software-properties-common is already the newest version (0.99.49.3).
software-properties-common set to manually installed.
IMG_3.1_Installing_Common_Config
```

Instalando PHP

Una vez que cumplimos con estos requisitos, podremos instalar PHP y las extensiones necesarias, para la versión actual **LARAVEL 12** Requerimos PHP >= 8.1, por lo que deberemos lanzar los comandos:

```
add-apt-repository ppa:ondrej/php -y
apt update
apt install -y php8.2 php8.2-cli php8.2-common php8.2-mysql php8.2-xml php8.2-
bcmath php8.2-mbstring php8.2-curl php8.2-fpm
```

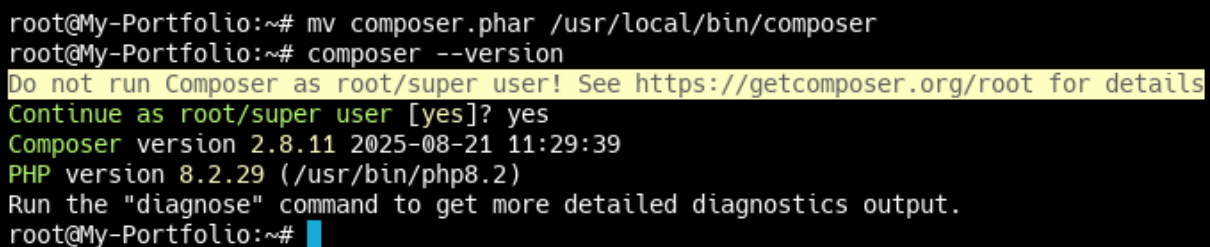
🔥 Consideraciones

A la fecha actual 03-09-2025 es la versión más estable, sin embargo si lees esto puede que hayan versionas LTS más nuevas de PHP y una versión superior de Laravel, puedes instalarla en lugar del 8.2

Instalando Composer

Otro de los requerimientos básicos para correr nuestro proyecto es instalar Composer, pieza clave para levantar nuestro servicio de Laravel, por lo que en este caso los comandos que estaremos usando serán:

```
curl -sS https://getcomposer.org/installer | php
mv composer.phar /usr/local/bin/composer
composer --version
```



```
root@My-Portfolio:~# mv composer.phar /usr/local/bin/composer
root@My-Portfolio:~# composer --version
Do not run Composer as root/super user! See https://getcomposer.org/root for details
Continue as root/super user [yes]? yes
Composer version 2.8.11 2025-08-21 11:29:39
PHP version 8.2.29 (/usr/bin/php8.2)
Run the "diagnose" command to get more detailed diagnostics output.
root@My-Portfolio:~#
```

IMG_3.2_Installing_Composer

Instalando Nginx

Ahora los siguientes pasos serán determinantes para asegurarnos que nuestra aplicación sea visible en internet, deberemos instalar Nginx para el servidor y Certbot para los certificados https.

En el caso de Nginx los comandos que estaremos empleando serán los siguientes:


```
apt install -y nginx
systemctl enable nginx
systemctl start nginx
```

En Caso De Error

⚡ Posibles errores

En caso de que veamos un error de ocupación de nuestro puerto 80, lo más probable sea por que el servidor de nuestro droplet está usando Apache, en ese caso deberemos desactivar ese servicio para poder usar Nginx

```
root@My-Portfolio:~# systemctl start nginx
Job for nginx.service failed because the control process exited with error code.
See "systemctl status nginx.service" and "journalctl -xeu nginx.service" for details.
root@My-Portfolio:~# systemctl status nginx.service
* nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: failed (Result: exit-code) since Wed 2025-09-03 19:55:44 UTC; 32s ago
     Docs: man:nginx(8)
  Process: 19787 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 19789 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=1/FAILURE)
     CPU: 19ms

Sep 03 19:55:42 My-Portfolio nginx[19789]: nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
Sep 03 19:55:43 My-Portfolio nginx[19789]: nginx: [emerg] bind() to [::]:80 failed (98: Address already in use)
Sep 03 19:55:43 My-Portfolio nginx[19789]: nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
Sep 03 19:55:43 My-Portfolio nginx[19789]: nginx: [emerg] bind() to [::]:80 failed (98: Address already in use)
Sep 03 19:55:43 My-Portfolio nginx[19789]: nginx: [emerg] bind() to 0.0.0.0:80 failed (98: Address already in use)
Sep 03 19:55:43 My-Portfolio nginx[19789]: nginx: [emerg] bind() to [::]:80 failed (98: Address already in use)
Sep 03 19:55:44 My-Portfolio nginx[19789]: nginx: [emerg] still could not bind()
Sep 03 19:55:44 My-Portfolio systemd[1]: nginx.service: Control process exited, code=exited, status=1/FAILURE
Sep 03 19:55:44 My-Portfolio systemd[1]: nginx.service: Failed with result 'exit-code'.
Sep 03 19:55:44 My-Portfolio systemd[1]: Failed to start nginx.service - A high performance web server and a reverse proxy server.
```

IMG_3.3_Possible_Errors

Podremos solucionar esto deteniendo y dejando desactivado el servicio de apache, mediante el uso de los siguientes comandos:

```
systemctl stop apache2
systemctl disable apache2
```

Una vez que estemos seguros que lo hemos eliminado podremos intentar nuevamente:

```
systemctl start nginx
```

Y si todo ha salido bien, veremos que nuestro servicio está corriendo al usar el comando:

```
systemctl status nginx.service
```

```

root@My-Portfolio:~# lsof -t :80
COMMAND  PID    USER  FD  TYPE DEVICE SIZE/OFF NODE NAME
apache2 19304   root   4u  IPv6 42684    0t0  TCP *:http (LISTEN)
apache2 19307 www-data 4u  IPv6 42684    0t0  TCP *:http (LISTEN)
apache2 19309 www-data 4u  IPv6 42684    0t0  TCP *:http (LISTEN)
apache2 19310 www-data 4u  IPv6 42684    0t0  TCP *:http (LISTEN)
apache2 19311 www-data 4u  IPv6 42684    0t0  TCP *:http (LISTEN)
apache2 19312 www-data 4u  IPv6 42684    0t0  TCP *:http (LISTEN)
apache2 19361 www-data 4u  IPv6 42684    0t0  TCP *:http (LISTEN)
root@My-Portfolio:~# systemctl stop apache2
root@My-Portfolio:~# systemctl disable apache2
Synchronizing state of apache2.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install disable apache2
Removed "/etc/systemd/system/multi-user.target.wants/apache2.service".
root@My-Portfolio:~# systemctl start nginx
root@My-Portfolio:~# system status nginx.service
Command 'system' not found, did you mean:
  command 'systemd' from deb systemd (255.4-1ubuntu8.10)
  command 'system3' from deb simh (3.8.1-6.1)
Try: apt install <deb name>
root@My-Portfolio:~# systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Wed 2025-09-03 20:02:42 UTC; 32s ago
     Docs: man:nginx(8)
   Process: 19940 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 19942 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Main PID: 19943 (nginx)
    Tasks: 2 (limit: 1110)
   Memory: 1.7M (peak: 1.9M)
      CPU: 24ms
   CGroup: /system.slice/nginx.service
           └─19943 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─19944 "nginx: worker process"

```

IMG_3.4_Solution

Instalando Certbot

El siguiente paso será activar el certificado https para nuestra página aprovechando el uso de una herramienta Let's Encrypt cómo Certbot, una vez dicho esto podremos usar el comando:

```
apt install -y certbot python3-certbot-nginx
```

Configurando Nuestro Sitio Con Nginx

Ya hemos hecho la instalación de todas las herramientas necesarias, ahora debemos crear un archivo personalizado para configurar nuestro proyecto, en este caso he elegido llamar mi carpeta cómo **portfolio**, lo que sigue es movernos a raíz y desde ahí navegar a la ruta de Nginx y crear dicha carpeta:

```

cd ../../
cd etc/nginx/sites-available
mkdir portfolio

```

Una vez que hayamos hecho esto, podremos proseguir con:

```
nano portfolio
```

Esto nos habilitará el editor de nano, por lo que podremos escribir la configuración base de nuestro sitio, puedes usar algo cómo la siguiente estructura:

```
server {
    listen 80;
    server_name midominio.com www.midominio.com;

    root /var/www/portfolio/public;

    index index.php index.html;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php8.2-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }
}
```

Para este punto será necesario que reemplacemos `server_name midominio.com www.midominio.com;`, por nuestra dirección de dominio real.

Creando El Enlace Simbólico En sites-enabled

Para activar nuestra configuración de Nginx deberemos hacer:

```
ln -s /etc/nginx/sites-available/portfolio-config /etc/nginx/sites-enabled/
```

Ahora vamos a probar la configuración:

```
nginx -t
```

Y si todo está en Ok, deberemos reiniciar Nginx.

```
root@My-Portfolio:/etc/nginx/sites-enabled# ln -s /etc/nginx/sites-available/portfolio-config /etc/nginx/sites-enabled/
root@My-Portfolio:/etc/nginx/sites-enabled# ls
default portfolio-config
root@My-Portfolio:/etc/nginx/sites-enabled# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@My-Portfolio:/etc/nginx/sites-enabled#
```

IMG_4_Nginx_Config_Enabled

Reiniciamos con:

```
systemctl reload nginx
```

Configurando Nuestros DNS


Ya hemos hecho la configuración del Droplet, dentro y fuera de Digital Ocean, lo que sigue ahora es configurar los DNS desde Name Cheap hasta DO, para que podamos usar el servicio de VPS.

En este punto es indispensable que tengamos la dirección IP de nuestro Droplet, pues vamos a emplear esta para poder hacer la configuración de nuestro DNS

En nuestro servidor de DNS (NameCheap para este caso) vamos a agregar dos tipos de dns.

1. @
2. www

Estos nos van a servir para indicar que esperamos sean accesibles a partir de todos los lugares que contengan un www, por lo que estamos dejando el puerto abierto a cualquier punto de internet, para hacer la configuración de estos dos, lo que debemos hacer es agregar ambos registros del tipo A, en el host colocar en el primero el @ y el segundo www, para finalizar colocando el valor de nuestra IP generada por el Droplet.

<input type="checkbox"/> Type	Host	Value	TTL	
<input type="checkbox"/> A Record	@		Automatic	
<input type="checkbox"/> A Record	www		Automatic	
 ADD NEW RECORD				

IMG_5_Configuration_For_Records

De esta manera habremos hecho el enlace desde nuestro dominio a nuestro droplet y en el transcurso de 24-48 hrs deberíamos ser capaces de ver la propagación de nuestro dominio, podemos visualizar si este ya ha concluido entrando a la siguiente dirección




Comprobar propagación

Una vez que se haya hecho de manera correcta la propagación, podremos ver en qué servidores ya se encuentra accesible nuestro propio dominio





































DNS CHECK

A

Search

   CD Flag

Refresh: 20 sec.

 San Francisco CA, United States OpenDNS 	178.128.184.248 	
 Mountain View CA, United States Google 	178.128.184.248 	
 Berkeley, US Quad9 	178.128.184.248 	
 New York, United States Oracle Corporation 	178.128.184.248 	
 Ashburn, United States NeuStar 	178.128.184.248 	
 San Jose, United States Corporate West Computer Systems 	178.128.184.248 	
 Virginia, United States VeriSign Global Registry Services 	178.128.184.248 	
 Burnaby, Canada Fortinet Inc 	178.128.184.248 	
 Yekaterinburg, Russian Federation Skydns 	178.128.184.248 	

IMG_5.1_DNS_Check

Activando El HTTPS Con CertBot

Ahora que hemos hecho la propagación y hemos configurado e instalado la paquetería necesaria para hacer correr la aplicación es necesario que activemos el Let's Encrypt que habíamos mencionado con anterioridad, por lo que tendremos que generar los certificados para nuestro dominio, haciendo uso del comando:

```
certbot --nginx -d thinkguille.space -d www.thinkguille.space
```

⚠ Reemplazar Tu Dominio

En este caso hemos puesto el dominio de ejemplo para esta guía, sin embargo, ahora deberás reemplazar ese dominio por el que tú tengas

Si todo ha salido bien deberíamos ser capaces de ver un mensaje que diga **"Successfully deployed certificate"**

```
Deploying certificate
Successfully deployed certificate for thinkguille.space to /etc/nginx/sites-enabled/portfolio-config
Successfully deployed certificate for www.thinkguille.space to /etc/nginx/sites-enabled/portfolio-config
Congratulations! You have successfully enabled HTTPS on https://thinkguille.space and https://www.thinkguille.space
```

```
-----
If you like Certbot, please consider supporting our work by:
* Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
* Donating to EFF: https://eff.org/donate-le
-----
```

```
root@My-Portfolio:~#
```

IMG_6_Deployed_Certificate

Este certificado cuenta con un tiempo de expiración, por lo que si queremos mantener nuestro dominio verificado con el Let's Encrypt debemos hacer que CertBot ejecute un cronjob de manera que pueda estar dando una renovación cada 90 días. Para eso ejecutaremos el siguiente comando:

```
systemctl list-timers | grep certbot
```

Y para probar que se haga dicha renovación usaremos el comando:

```
certbot renew --dry-run
```

Con esto deberemos ver algo parecido a la imagen 6.1_Renew

```
root@My-Portfolio:~# systemctl list-timers | grep certbot
Tue 2025-09-09 08:10:02 UTC    17h Mon 2025-09-08 12:08:00 UTC    2h 52min ago certbot.timer
root@My-Portfolio:~# certbot renew --dry-run
Saving debug log to /var/log/letsencrypt/letsencrypt.log

-----
Processing /etc/letsencrypt/renewal/thinkguille.space.conf
-----
Account registered.
Simulating renewal of an existing certificate for thinkguille.space and www.thinkguille.space

-----
Congratulations, all simulated renewals succeeded:
  /etc/letsencrypt/live/thinkguille.space/fullchain.pem (success)
-----
root@My-Portfolio:~#
```

IMG_6.1_Renew

Desplegar Nuestra Aplicación

En los pasos anteriores dejamos todo en bandeja de plata para que podamos subir nuestra aplicación (Laravel para este ejemplo), dependiendo del Framework o la tecnología que estemos empleando será la configuración que deberemos hacer dentro de nuestro proyecto.

En este punto tenemos dos opciones:

1. Descargar un repositorio que tengamos creado o uno guía
2. Subir la app directamente desde nuestra máquina local

En este caso específico veremos el repositorio (Suponiendo que hayas creado tu propio portfolio de manera versionada), por lo que dentro de nuestra terminal vamos a ejecutar nuestro servidor y haremos el movimiento dentro de:

```
cd var/www
```

Una vez que estemos dentro de esta carpeta clonaremos nuestro repositorio haciendo uso del comando:

```
git clone https://github.com/my_user/my_repo.git portfolio
```

Instalamos las dependencias que sean necesarias:

```
composer install --no-dev --optimize-autoloader
```

Configuramos nuestro entorno:

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

```
nano .env
```

Dentro del .env vamos a hacer los siguientes cambios:

```
APP_NAME=Laravel
APP_ENV=production
APP_KEY=
APP_DEBUG=false
APP_URL=https://thinkguille.space
```

IMG_7_Update_.env

Ahora generamos la llave de nuestra aplicación:

```
php artisan key:generate
```

Por último nos aseguramos de dejar nuestro .env limpio de cualquier indicación de base de datos y podríamos ver de manera correcta nuestro portfolio en la web.

Y Si No Veo Los Estilos En Mi Portfolio?

En caso de que no esté funcionando tailwind dentro de nuestro portfolio de lado del servidor, será indispensable que hagamos una instalación adicional, dicha instalación será node, para eso corremos el comando:

```
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
```

Y posteriormente

```
sudo apt install -y nodejs
```

Una vez que hayamos descargado esto podemos movilizarnos nuevamente a nuestra carpeta y ejecutamos los comandos de instalación y generación de paquetes

```
cd /var/www/portfolio
npm install
npm run build
```

Con estos cambios deberíamos ser capaces de visualizar los estilos dentro de nuestro propio portfolio.



thinkguille.space



About me

Technical Project Lead con experiencia entregando apps cloud-native en Azure/AWS.

Lidero equipos, defino arquitecturas escalables e integro autenticación segura con Entra ID.

Fuerte base en REST APIs, microservicios, CI/CD y despliegues con contenedores.

Skills

- Leadership & Agile/Scrum
- Azure Container Apps Azure SQL Blob Storage

- Static Web Apps
- Control de usuarios y accesos
- Testing / QA
- Microsoft Power Platform
- Atención a usuarios y training

IMG_8_Styles

Automatizando El Deploy

Muchas veces nuestro portfolio irá cambiando con el tiempo, por lo cuál se convierte en una necesidad, la automatización del proceso de despliegue, cada que nosotros realicemos cambios y subamos estos mismos a nuestro repositorio.

Dependiendo de nuestro servicio es la manera en la que haremos nuestro hook, en este caso Digital Ocean nos ofrece funciones que podrían servirnos para hacer el auto-deploy, sin embargo si queremos simular o acercarnos más a un entorno real de producción, siempre será una mejor opción emplear GitHub Actions, por lo cuál en esta guía vamos a proceder a cómo hacer la configuración de este.

Primero debemos de crear dentro de la carpeta de nuestro portfolio dos subcarpetas

```
.github/workflows
```

Y dentro de estas mismas crearemos un archivo llamado ***deploy.yml***

Este debe contener algo similar a:

```
name: Deploy Portfolio

on:
  push:
    branches: ["main"]

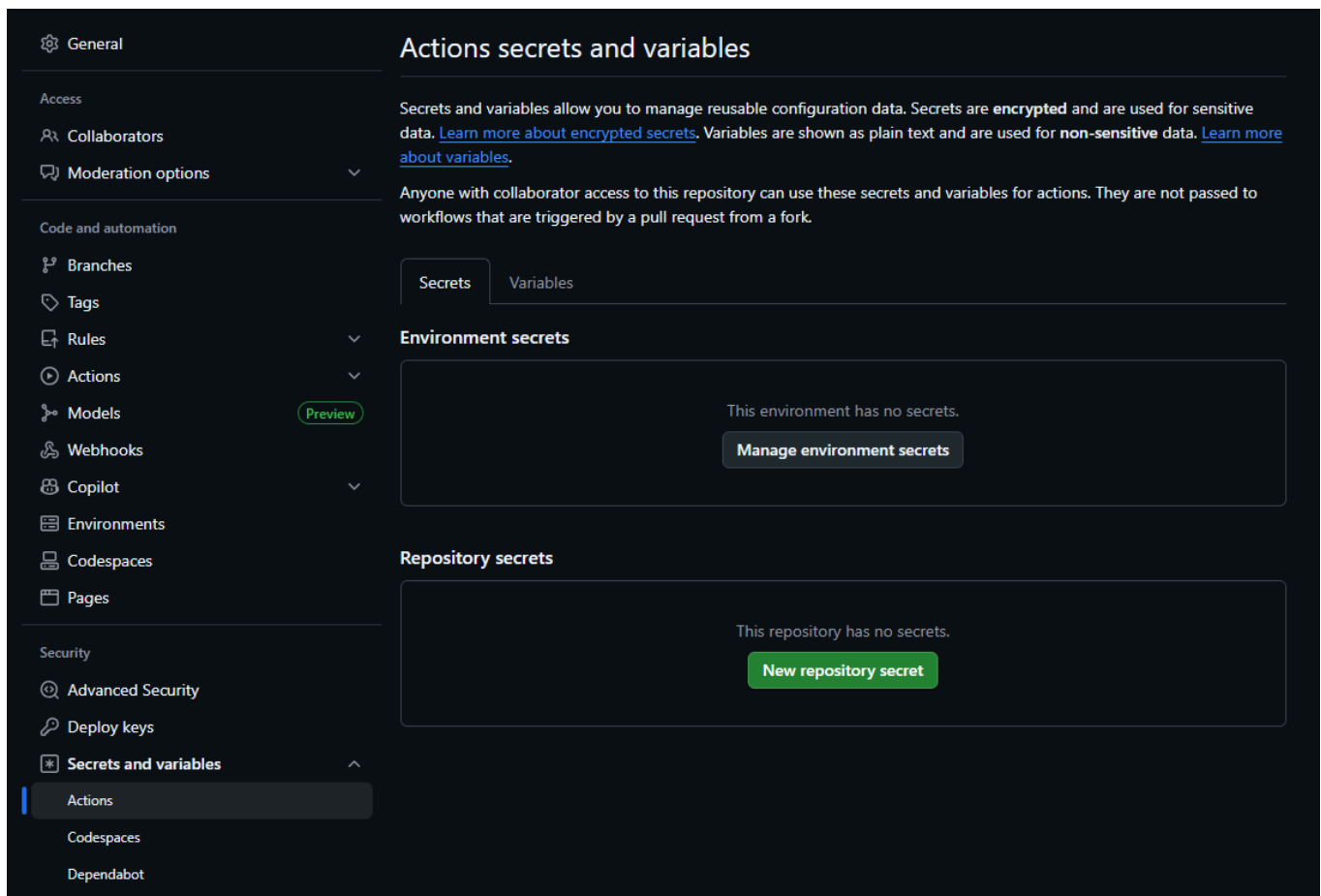
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
```

steps:

- name: Checkout repo
uses: actions/checkout@v3
- name: Instal PHP
uses: shivammathur/setup-php@v2
with:
php-version: '8.2'
- name: Instal Dependencys For Composer
run: composer install --no-dev --optimize-autoloader
- name: Instal Node.js
uses: actions/setup-node@v3
with:
node-version: '20'
- name: Compile assets
run: |
npm install
npm run build
- name: Deploy SSH
uses: appleboy/ssh-action@v0.1.10
with:
host: \${ secrets.DO_HOST }
username: \${ secrets.DO_USER }
key: \${ secrets.DO_SSH_KEY }
script: |
cd /var/www/portfolio
git pull origin main
composer install --no-dev --optimize-autoloader
npm install
npm run build
php artisan migrate --force
php artisan optimize

Debido a que este repositorio es público, es importante que no pongamos de manera directa las credenciales de nuestro servidor, por lo que hemos pasado como variables secretas aquellos puntos de ingreso sensibles.

Sin embargo para que nuestro archivo sepa que llaves son las que se van a emplear, usaremos el baúl que nos ofrece GitHub, si nos movemos directamente a Settings/Secrets And Variables/Actions podremos hacer el almacenaje de nuestras credenciales:



IMG_9_Secrets_And_Variables

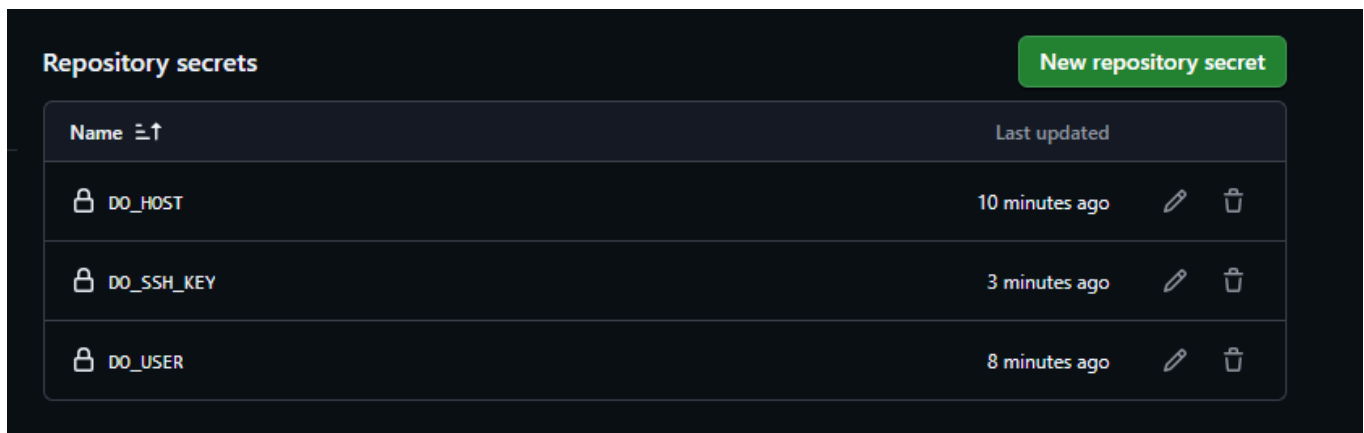
Aquí vamos a configurar 3 secretos:

- **DO_HOST** → 123.45.67.89 (ejemplo de tu IP pública).
- **DO_USER** → root .
- **DO_SSH_KEY** → pega tu llave privada completa en formato:

🔥 Sacar SSH

Si quieres saber cómo sacar tu SSH y dónde ese encuentra ubicada esta misma puedes hacer `ssh-keygen -t rsa -b 4096 -C "tu_correo@ejemplo.com"`

Una vez que hayas hecho la configuración de tus secretos podras verlos listados:



IMG_9.1_Created_Secrets

Con esto configurado, cada que se suban cambios a main, se hará el despliegue de forma automática en nuestro servidor.

Para Concluir

Con esta guía hemos recorrido todo el ciclo necesario para levantar y mantener en producción nuestro portafolio personal utilizando Laravel en un Droplet de DigitalOcean. Algunos de los temas que estuvimos viendo abarcan:

- Desde la creación del servidor y configuración básica (PHP, Composer, Nginx, Certbot).
- Pasando por el despliegue manual inicial y la resolución de problemas comunes (como la compilación de estilos con Tailwind y Vite).
- Hasta llegar a un entorno automatizado mediante **GitHub Actions**, que nos permite simular un flujo real de CI/CD como se utiliza en entornos empresariales.

De esta forma, cada cambio que realicemos en nuestro repositorio de GitHub se verá reflejado en el servidor de manera automática, asegurando que nuestro portafolio esté siempre actualizado y optimizado sin necesidad de pasos manuales.

Este proceso no solo nos puede servir como un ejemplo práctico para desplegar un portafolio, sino que también representa una base sólida para proyectos más complejos que requieran un flujo de integración y despliegue continuo.