

GTV

16/12/2022

—

Miguel Moreno Such
Carlos Molina Alcolea
César Verdú Motos

Índice

Creación de proyecto en Digital Ocean	2
Conectar proyecto de Digital Ocean con Laravel Forge	3
Cómo obtener un dominio para nuestro proyecto	5
Añadir nuevo sitio web en Forge	7
Instalación de PhpMyAdmin en Forge	9
Configuración de la aplicación en Forge	10
Habilitar Push to Deploy	10
Utilizar una rama prod para el despliegue de la aplicación	11
Preparación del fichero .env en Laravel Forge	11
Generación de un certificado SSL	12
Habilitación de notificaciones de deploy en Telegram	13
Despliegue	14
Tests automáticos con Github Actions	15


Creación de proyecto en Digital Ocean

Para el hosting hemos utilizado Digital Ocean porque se ajustaba a nuestras necesidades para el hosteo. Es un proveedor de servidores virtuales privados en la nube con una amplia variedad de herramientas para hacer nuestro hosting más sencillo.

Lo primero que tenemos que hacer después de registrarnos y verificar nuestra cuenta es crear un nuevo proyecto

- 1 Create Project
- 2 Move Resources

Create new project



Name your project

Enter name

gtv

✓

Add a description

Helpful for teams or differentiating between projects with similar names.

Enter description

Proyecto de Fin de Grado

Tell us what it's for

This will help us to provide a more relevant experience.

Class project / Educational purposes

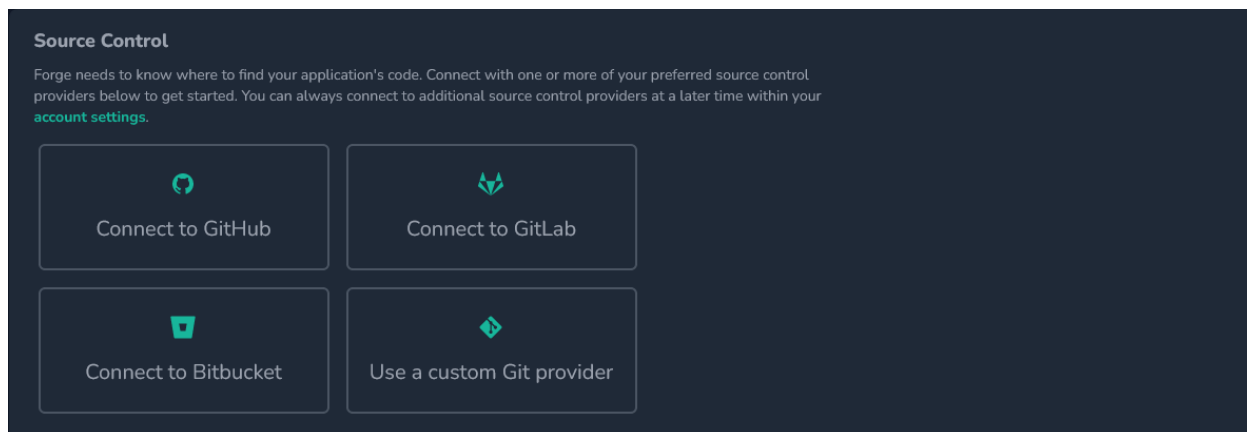
* ▼

Create Project

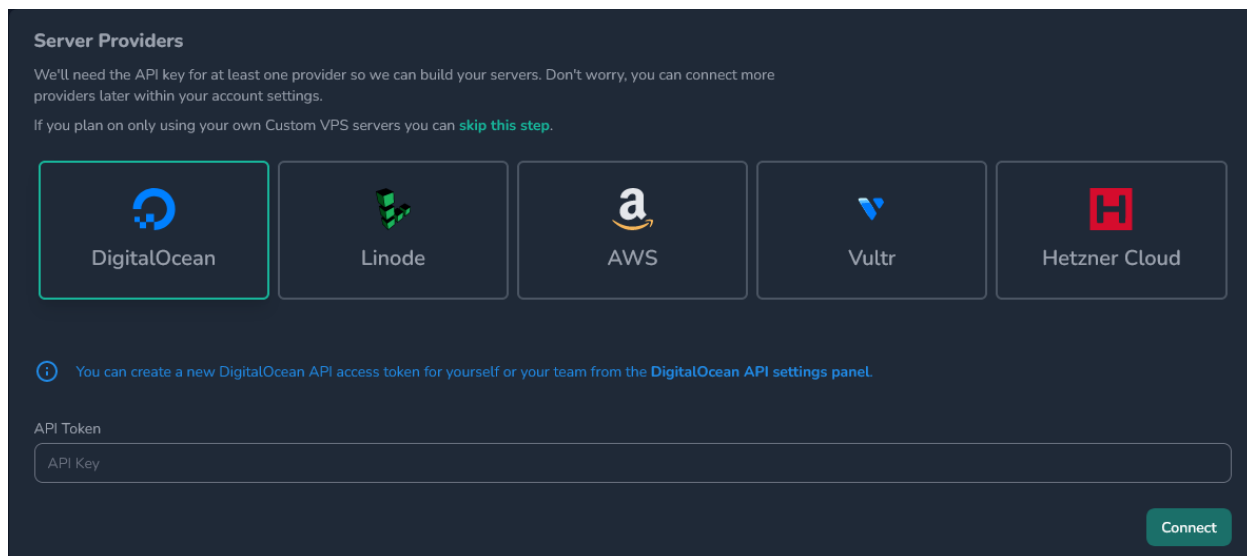
[Cancel](#)

Conectar proyecto de Digital Ocean con Laravel Forge

Una vez creado nuestro proyecto en Digital Ocean, nos dirigimos a Laravel Forge para desplegar nuestra aplicación. Tras registrarnos nos aparecerá una página en la que tenemos que especificar nuestra manera de controlar el código fuente



Y un proveedor de servidores, en nuestro caso Digital Ocean. En este último hay que introducir una API key para vincular ambos.



Esta API key se obtiene en la página de nuestro proyecto de Digital Ocean, accediendo a la sección *Applications & API* del panel de control del proyecto y generando una nueva API key

×

New personal access token

Token name

Enter token name

Gtv-token

✓

Expiration

Select token expiry

90 days

✓

Select scopes

☒ Read (default) ☒ Write (optional)

Read our [personal access token documentation](#) for more information on scopes.

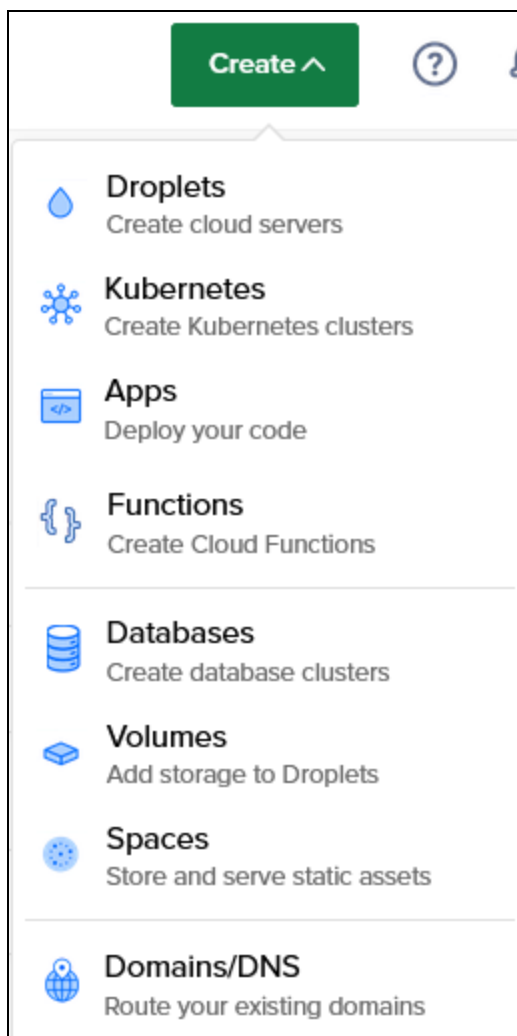
Generate Token

Una vez generado, lo copiamos y lo pegamos en el campo API token de la imagen anterior

Cómo obtener un dominio para nuestro proyecto

Para nuestro proyecto hemos decidido adquirir un dominio a través de la página namecheap.com, una vez adquirido nuestro dominio, desde el panel de control de la página podemos conectar el dominio de la página con la dirección IP del servidor de Digital Ocean.

Para obtener las direcciones a los DNS's de nuestro servidor nos dirigimos nuevamente a Digital Ocean y nos vamos al panel *Create* donde seleccionamos *Domain/DNS*



Añadimos el nuevo dominio y clickamos en *Add Domain*.

Add a domain

Enter a domain that you own below and start managing your DNS within your DigitalOcean account.

Invalid hostname
gtvtrabajofinalgrado.xyz

gtv

Add Domain

Ahora copiaremos las direcciones que hay en el apartado *Domain Records* y las usaremos en el panel de control de *NameCheap*

NS	gtvtrabajofinalgrado.xyz	<u>directs to ns1.digitalocean.com.</u>	1800	More ▾
NS	gtvtrabajofinalgrado.xyz	<u>directs to ns2.digitalocean.com.</u>	1800	More ▾
NS	gtvtrabajofinalgrado.xyz	<u>directs to ns3.digitalocean.com.</u>	1800	More ▾

Así es como quedaría:

NAMESERVERS

?

Custom DNS ▾

ns1.digitalocean.com.
ns2.digitalocean.com.
ns3.digitalocean.com.
+ ADD NAMESERVER

Ahora sólo nos queda añadir un par de registros más para que apunten a la IPv4 de nuestro server

A	www.gtvtrabajofinalgrado.xyz	directs to 188.166.21.32	3600	More ▾
A	gtvtrabajofinalgrado.xyz	directs to 188.166.21.32	3600	More ▾

Y ya tenemos acceso a nuestra página desde nuestro dominio.

Añadir nuevo sitio web en Forge

Con nuestro dominio funcionando y apuntando a la IP de nuestro servidor, ya podemos añadir el sitio web a Laravel Forge para que se despliegue la aplicación

New Site

Think of sites as representing each "domain" on your server. The "default" site is included with each freshly provisioned server; however, you should delete it and create a new site with a valid domain name when you are ready to launch your production site. If you need to host additional domains or sub-domains, you may add them here.

When using Website Isolation, Forge will automatically create an isolated PHP-FPM process for the given site.

Root Domain

gtvtrabajofingrado.xyz

Aliases

second-domain.com,third-domain.com

Project Type

General PHP / Laravel

Web Directory

/public

PHP Version

PHP 8.1

☐ Allow Wildcard Sub-Domains

☐ Use Website Isolation

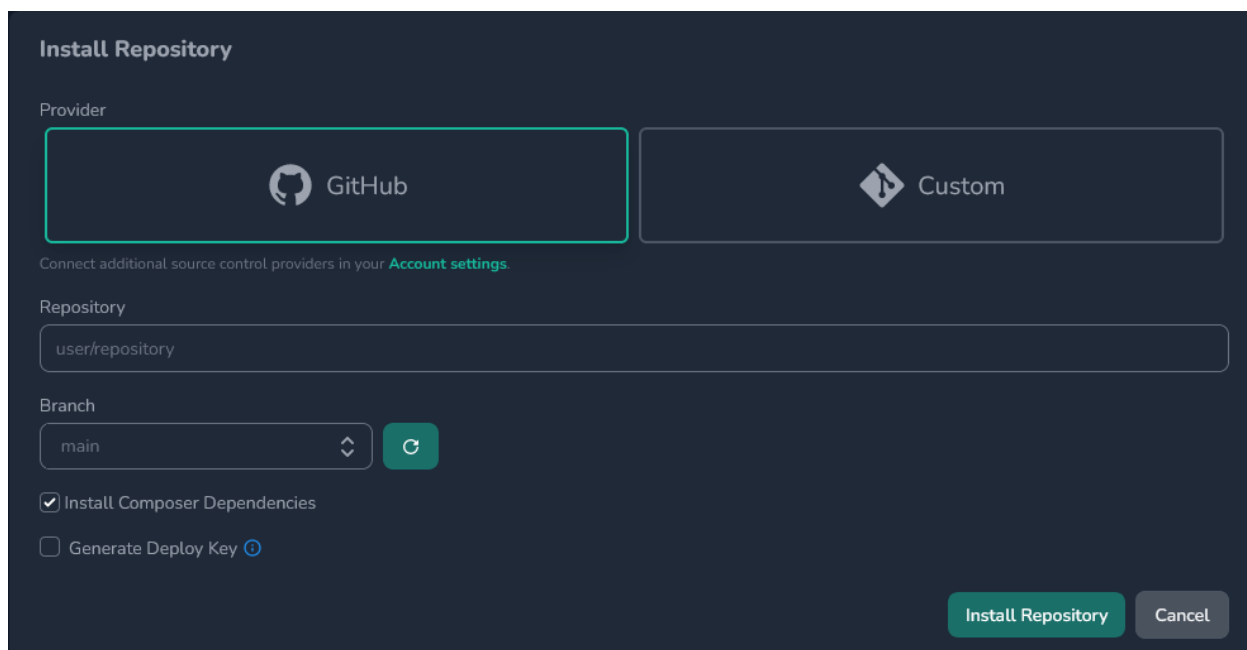
☒ Create Database

Database Name

gtv.db

Add

Una vez añadamos el sitio web, se iniciará el proceso de creación de éste y se nos mostrarán varias pantallas para terminar con la instalación, empezamos con:



Install Repository

Provider

GitHub Custom

Connect additional source control providers in your [Account settings](#).

Repository

user/repository

Branch

main

☒ Install Composer Dependencies

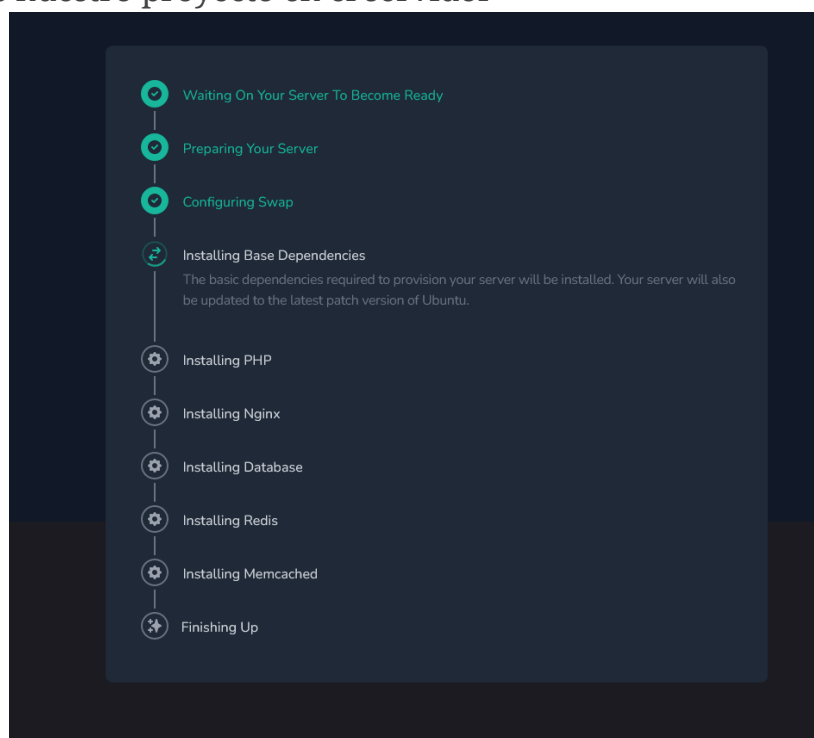
☐ Generate Deploy Key

Install Repository Cancel

Donde pondremos la dirección de nuestro repositorio y la rama que queremos desplegar, a parte de dos opciones que tenemos que marcar.

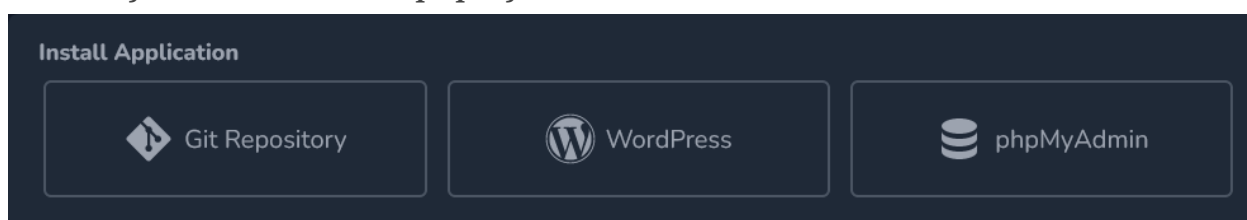
La segunda opción es importante a la hora de habilitar la funcionalidad *Push to Deploy*.

Una vez rellenados todos los campos, clickamos Install Repository y empezará la instalación de nuestro proyecto en el servidor



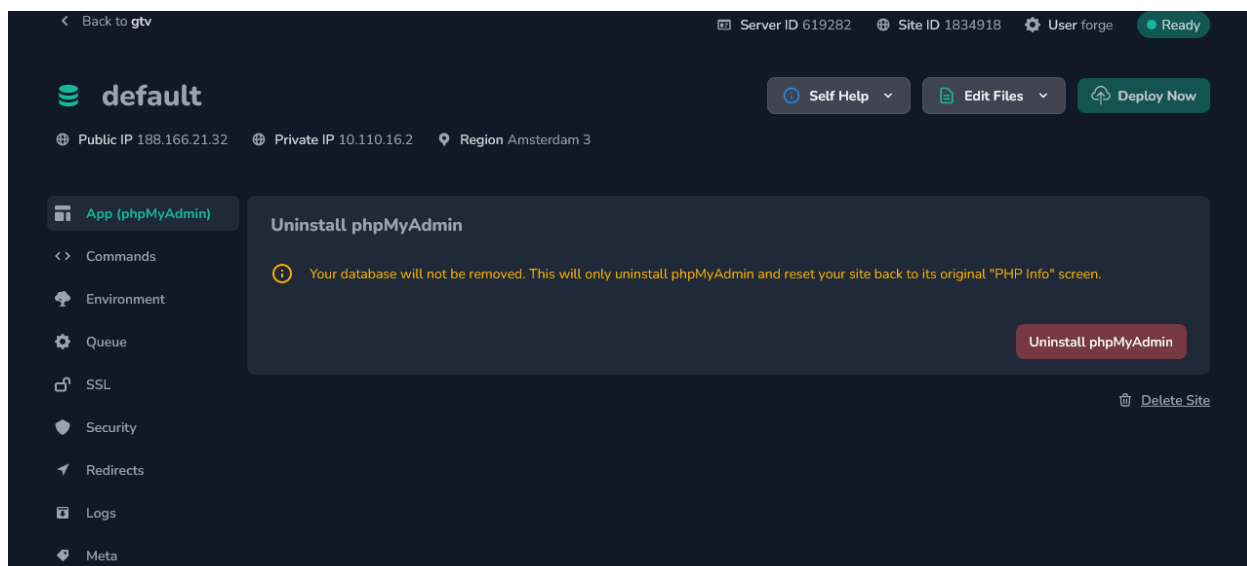
Instalación de PhpMyAdmin en Forge

Para la gestión de la base de datos del servidor hemos decidido utilizar PhpMyAdmin, para ello, nos dirigimos nuevamente al apartado *sites* de nuestro servidor en Forge, y repetimos el mismo proceso que para desplegar la aplicación, salvo que en este caso no pondremos nombre en el campo *root domain* del formulario de creación, lo que hará que se cree con el nombre default, y seleccionaremos phpmyadmin









Una vez seleccionado, le ponemos un nombre a la base de datos y creamos un usuario llamado forge y terminaremos con la instalación de phpmyadmin en el servidor.

Para acceder sólo hay que irse al nuevo sitio creado y clicar en el nombre



Configuración de la aplicación en Forge

Con todo preparado, ya podemos preparar la aplicación para su despliegue, para ello nos dirigimos al sitio que hemos creado anteriormente con el nombre del dominio

Active Sites			
Domain	App	Last Deployed	Isolated
default	 phpMyAdmin PHP 8.1	—	
  gtvtrabajofinalgrado.xyz	 Cma13/gtv (prod) PHP 8.1	✓ 16 dic 2022, 9:18	

Y desde ahí configuraremos todo

Habilitar Push to Deploy

Push to Deploy es una funcionalidad muy interesante que nos ofrece Laravel Forge, consiste en que cada vez que se realice un push al repositorio de nuestra aplicación, Forge automáticamente realizará un pull en el servidor y volverá a desplegar la aplicación sin que tengamos que intervenir, lo que facilita mucho el despliegue.


Para habilitarlo simplemente tenemos que seleccionar la siguiente opción en el panel de control del servidor:

Deployment

Quick deploy allows you to easily deploy your projects when you push to source control. When you push to this application's deployment branch, Forge will pull your latest code from source control and execute your deployment script.

[Enable Quick Deploy](#)[View Latest Deployment Log](#)

Y añadir al repositorio la Deploy Key creada anteriormente en la instalación de la aplicación



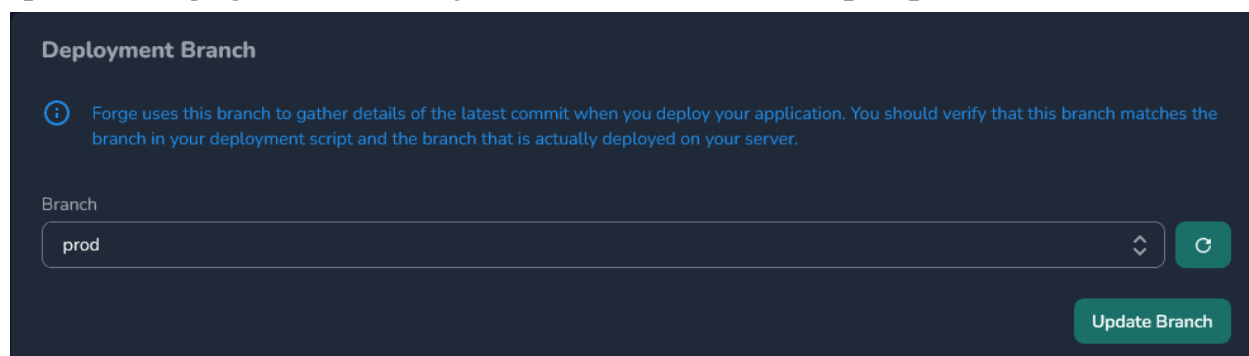
forge
SHA256:6Cr9NYCJd+OiVamemawNg8Lb8YtXOAdkIc6FT/Ygw1M
Added on Dec 13, 2022 by @miguelms99
Last used within the last week — Read/write

[Delete](#)

Utilizar una rama *prod* para el despliegue de la aplicación

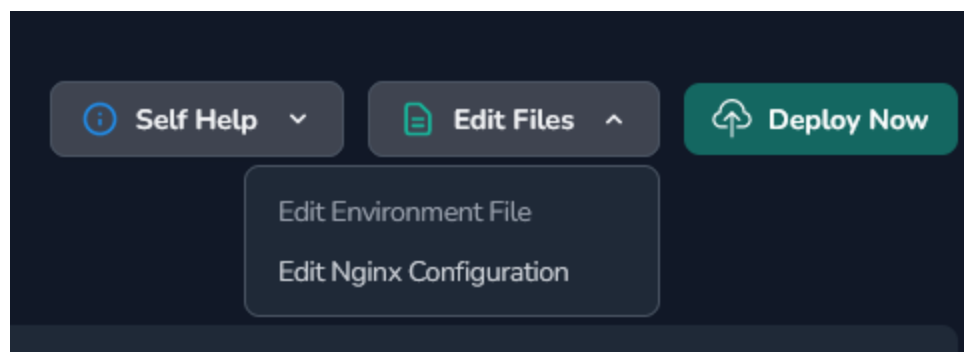
Para poder comprobar que todo esté bien antes de desplegar la aplicación, hemos creado una rama llamada *prod* en nuestro repositorio a la que mergearemos nuestra rama master cuando hayamos comprobado que los cambios realizados a la aplicación no generen ningún tipo de error.

Para usar la rama *prod* para los despliegues nos dirigimos al panel de control, al apartado *Deployment Branch* y seleccionamos la rama que queramos usar:

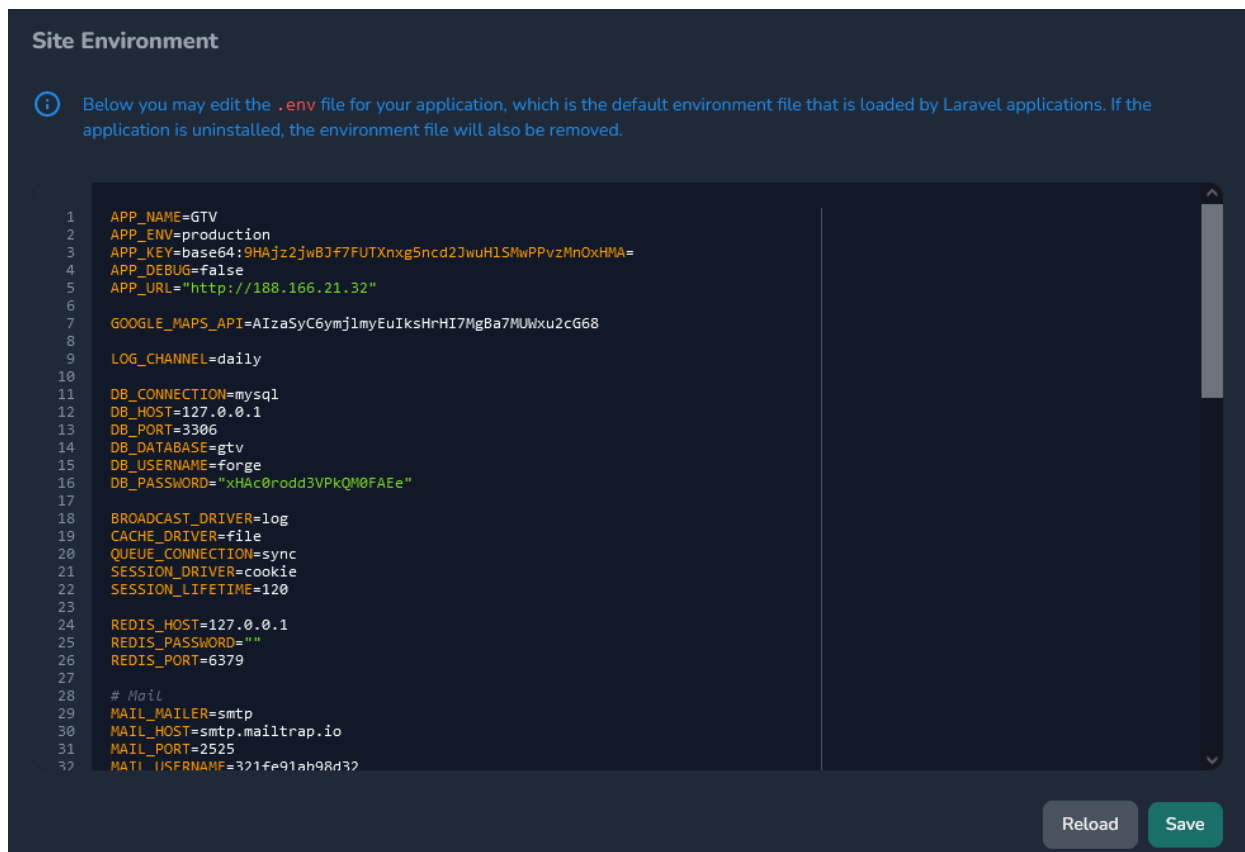


Preparación del fichero *.env* en Laravel Forge

Forge nos permite editar directamente el fichero de entorno de Laravel cómodamente desde el panel de control, sólo tenemos que irnos a los botones de acción y seleccionar:

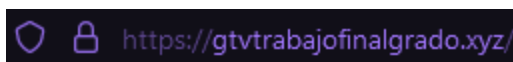
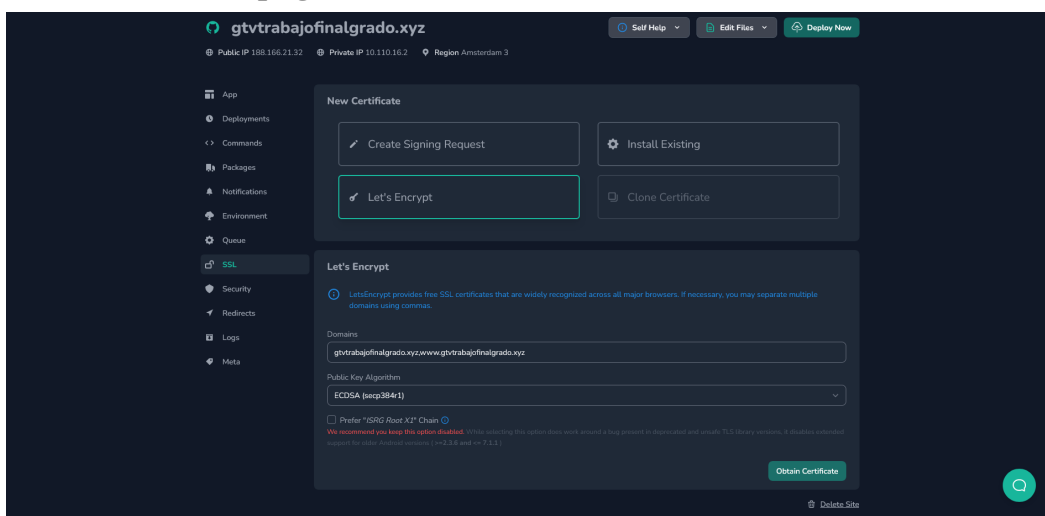


Esto nos abrirá la siguiente sección donde podremos editar el fichero:



Generación de un certificado SSL

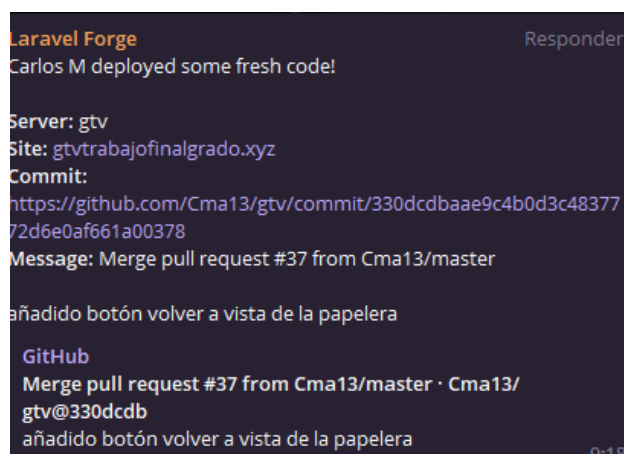
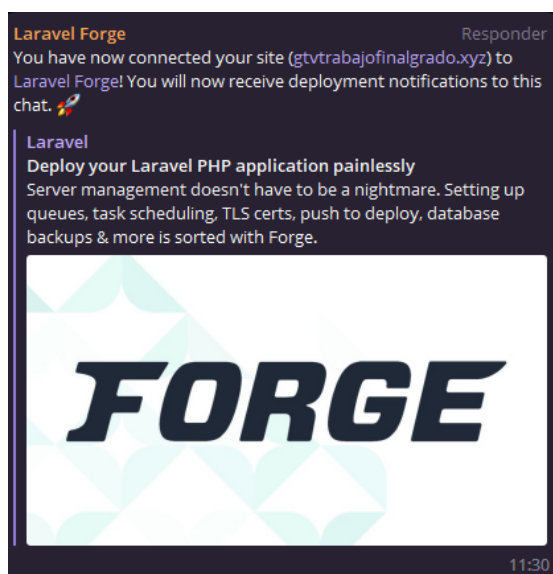
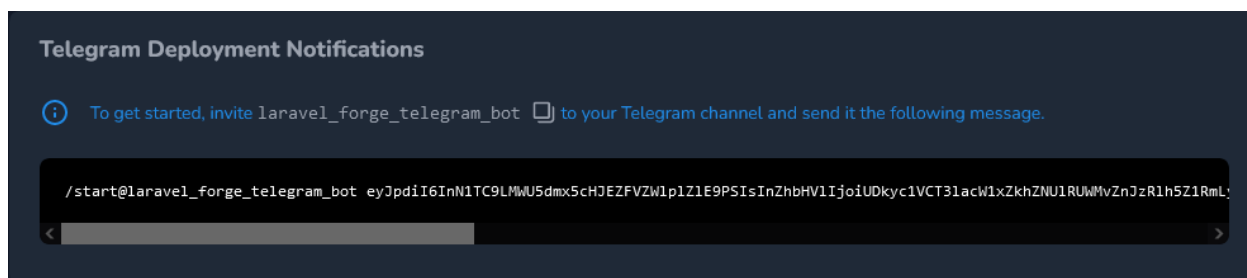
El proceso para generar un certificado SSL para nuestra página es muy sencillo, nos vamos al apartado SSL y simplemente seleccionamos *Let's Encrypt*, tras unos momentos nuestra página tendrá su certificado SSL



Habilitación de notificaciones de deploy en Telegram

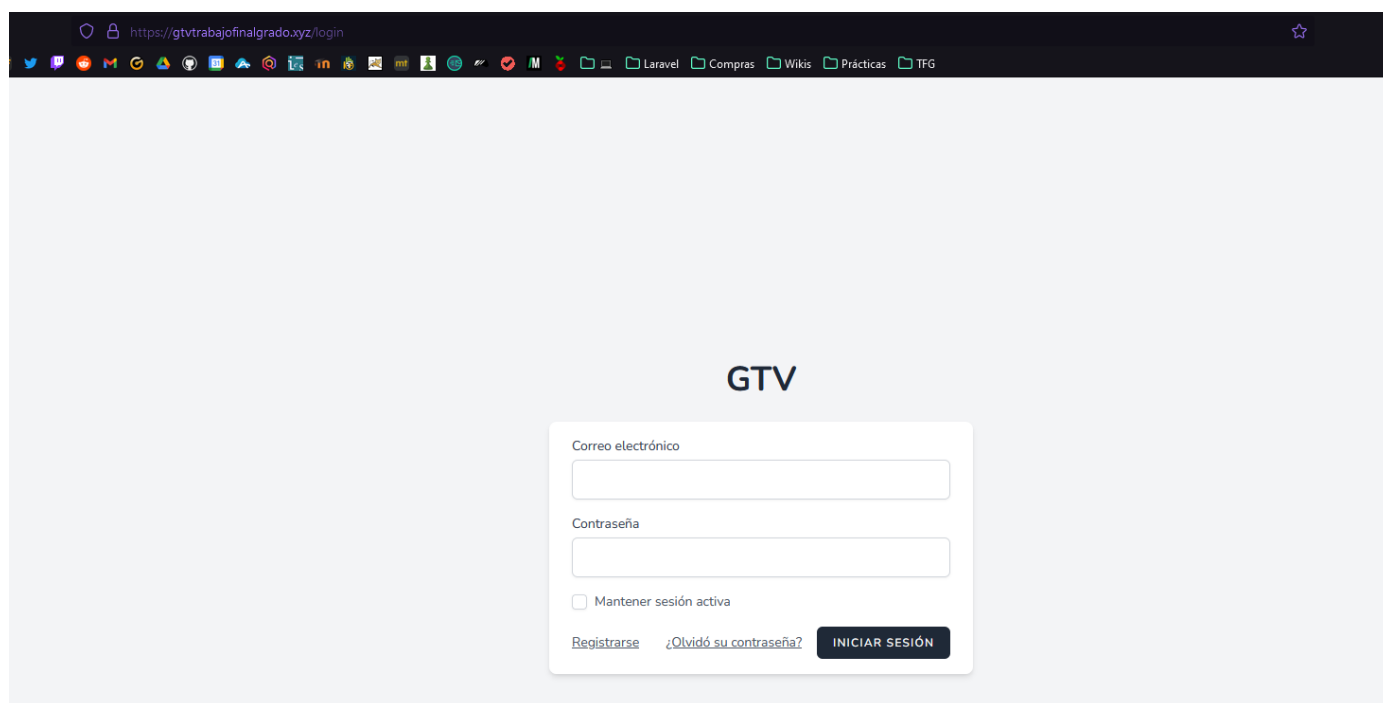
Forge nos ofrece la posibilidad de poder habilitar notificaciones que nos avisen de los despliegues de la aplicación en Slack, Discord, Telegram y por correo, nosotros hemos elegido la opción de Telegram.

Para ello nos vamos al apartado *Notifications* en el panel de control y bajamos hasta la sección *Telegram Deployment Notifications*, invitamos al bot a nuestro grupo, y copiamos el comando de telegram que se nos muestra



Despliegue

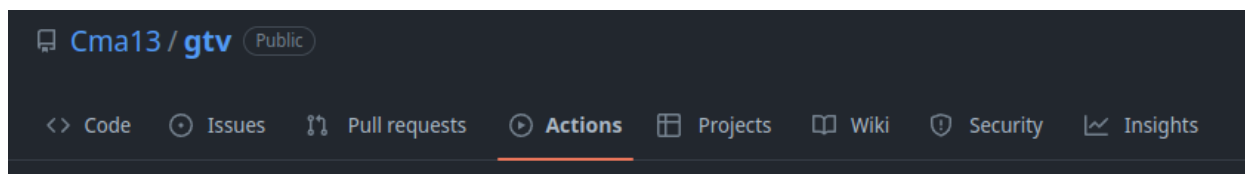
Una vez configurado todo, nos dirigimos a los botones de acción del panel de control y seleccionamos *Deploy now*, tras unos momentos y si todo ha ido bien, tendremos nuestra aplicación de Laravel desplegada correctamente en nuestro dominio



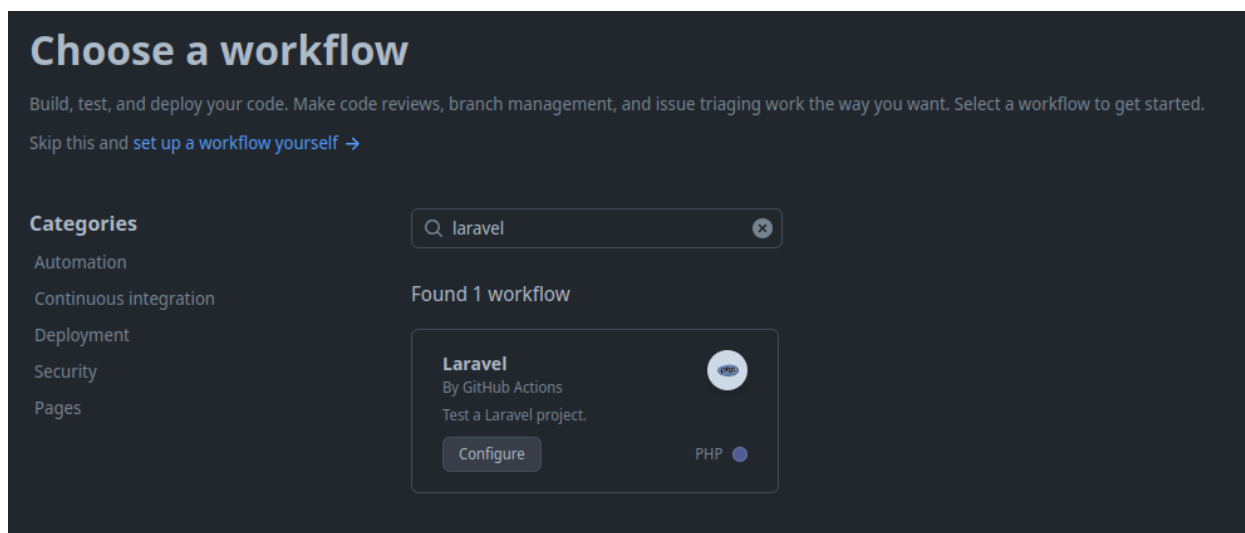
Tests automáticos con Github Actions

GitHub Actions es una plataforma de integración continua y despliegue continuo (CI/CD) que nos permite automatizar la compilación, las pruebas y el despliegue de nuestra aplicación. Nosotros hemos usado Github Actions para automatizar la ejecución de los tests. Los tests se ejecutan cada vez que se hace una pull request y cada vez que se hace un merge a la rama principal.

Para configurar esta automatización vamos a la pestaña Actions de nuestro repositorio:



Buscamos el workflow Laravel y hacemos click en configure:




En el siguiente paso configuramos la creación de un documento llamado `laravel.yml` en la carpeta `.github/workflows` de nuestro proyecto. Este documento define el comportamiento del workflow. Veamos el documento `laravel.yml`:



```
1  name: Laravel
2
3  on:
4    push:
5      branches: [ "master" ]
6    pull_request:
7      branches: [ "master" ]
8
9  jobs:
10   laravel-tests:
11
12     runs-on: ubuntu-latest
13
14     steps:
15     - uses: shivammathur/setup-php@15c43e89cdef867065b0213be354c2841860869e
16       with:
17         php-version: '8.1'
18     - uses: actions/checkout@v3
19     - name: Copy .env
20       run: php -r "file_exists('.env') || copy('.env.example', '.env');"
21     - name: Install Dependencies
22       run: composer install -q --no-ansi --no-interaction --no-scripts --no-progress --prefer-dist
23     - name: Generate key
24       run: php artisan key:generate
25     - name: Directory Permissions
26       run: chmod -R 777 storage bootstrap/cache
27     - name: Create Database
28       run: |
29         mkdir -p database
30         touch database/database.sqlite
31     - name: Execute tests (Unit and Feature tests) via PHPUnit
32       env:
33         DB_CONNECTION: sqlite
34         DB_DATABASE: database/database.sqlite
35       run: vendor/bin/phpunit
```

El documento es muy sencillo y fácil de modificar para que se adapte a nuestras necesidades. Algunos ejemplos de posibles modificaciones son:

- Cambiar el nombre de nuestra rama principal en las líneas 5 y 7
- Añadir la ejecución automática de tests para otras ramas
- Cambiar la versión de php en la línea 17
- Cambiar la base de datos en la línea 33



Una vez configurado el workflow podremos ver el resultado de ejecutar los tests cada vez que hacemos una pull request o un merge a master.




**Some checks haven't completed yet**

1 in progress check

[Hide all checks](#)

 **Laravel / laravel-tests (pull_request)** *In progress — This check has started...*


[Details](#)


**This branch has no conflicts with the base branch**

Merging can be performed automatically.

Merge pull request


or view [command line instructions](#).



**All checks have passed**

1 successful check


[Show all checks](#)


**This branch has no conflicts with the base branch**

Merging can be performed automatically.

Merge pull request



or view [command line instructions](#).




**All checks have failed**

1 failing check

[Hide all checks](#)

 **Laravel / laravel-tests (pull_request)** *Failing after 48s*

[Details](#)

**This branch has no conflicts with the base branch**

Merging can be performed automatically.

Merge pull request

or view [command line instructions](#).