Práctica 3.3: Despliegue de una aplicación React en Netlify y Vercel

1. Introducción

En la práctica anterior hemos visto cómo desplegar una aplicación de Node.js sobre un servidor Express en local (en nuestro propio servidor Debian). La práctica anterior podría asemejarse a las pruebas que realiza un desarrollador antes de pasar su aplicación al entorno de producción.

Ya sabemos que el *despliegue o deployment* como el proceso de mover nuestro código, típicamente de un sistema de control de versiones a una plataforma de hosting donde se aloja y es servida a los usuarios finales.

A la hora de desplegar la aplicación en producción, podría utilizarse el método de copiar los archivos al servidor concreto vía FTP (desuso), SSH u otros; y desplegarla para dejarla funcionando. No obstante, esta práctica se acerca más a la realidad ya que utilizaremos un repositorio de Github y una plataforma de PaaS (*Platform as a Service*) como Netlify para desplegar adecuadamente nuestra aplicación en producción.

1.1 ¿Qué es Github?

A pesar de que trataremos un poco más en profundidad Github en el siguiente tema, daremos una breve explicación aquí. GitHub es un servicio basado en la nube que aloja un sistema de control de versiones (VCS) llamado Git. Éste permite a los desarrolladores colaborar y realizar cambios en proyectos compartidos, a la vez que mantienen un seguimiento detallado de su progreso.

El control de versiones es un sistema que ayuda a rastrear y gestionar los cambios realizados en un archivo o conjunto de archivos. Utilizado principalmente por ingenieros de software para hacer un seguimiento de las modificaciones realizadas en el código fuente, el sistema de control de versiones les permite analizar todos los cambios y revertirlos sin repercusiones si se comete un error.

1.2 ¿Qué es Netlify?

Netlify es un proveedor de alojamiento en la nube que proporciona servicios de *backend* sin servidor (*serverless*[1]) para **sitios web estáticos**. Está diseñado para maximizar la productividad en el sentido de que permite a los desarrolladores (especialmente orientados al *frontend*), y a los ingenieros construir, probar y desplegar rápidamente sitios web/aplicaciones.

Funciona conectándose a un repositorio de GitHub, de donde extrae el código fuente. A continuación, ejecutará un proceso de construcción para prerenderizar las páginas de nuestro sitio web/aplicación en archivos estáticos.

Cuando hablamos de prerenderizar, es un paso del *build* en el que las páginas de tu aplicación se generan de antemano como archivos HTML estáticos, en lugar de esperar a que un servidor las construya dinámicamente cada vez que un usuario las visita. Durante el despliegue, una herramienta genera de forma anticipada esos archivos HTML completos (más los *assets* como CSS, JS e imágenes). Así, cuando un usuario visita la página, Netlify solo sirve un archivo ya listo desde su CDN, muy rápido y sin necesidad de computación en el servidor.

Esto trae varias ventajas:

- **Rendimiento**: las páginas se cargan mucho más rápido al venir de un archivo estático en la red de distribución de Netlify.
- **Seguridad**: no hay un *backend* que “ejecutar” en cada request, menos superficie de ataque.
- **SEO**: los buscadores reciben el HTML completo, no solo un esqueleto de JavaScript.
- **Costo**: servir archivos estáticos en CDN es más barato que mantener servidores corriendo.

Hay numerosas razones a favor de usar Netlify, aquí están algunas de ellas:

- Netlify hace que sea increíblemente sencillo desplegar un sitio web - de hecho, la forma más sencilla de lograrlo es utilizar GitHub, GitLab o Bitbucket para configurar el despliegue continuo.
- Netlify hace que sea súper fácil lanzar un sitio web con su solución de gestión de DNS incorporada.
- Podríamos desplegar fácilmente sólo una rama específica de nuestro proyecto Git - esto es útil para probar nuevas características que pueden o no llegar a la rama maestra/principal, o para determinar rápidamente cómo un PR (Pull Request) afectará a su sitio.

[1] *Serverless computing* es un modelo de ejecución de computación en la nube en el que el proveedor de los servicios en la nube destina por demanda recursos de las máquinas virtuales, cuidando de los servidores por sus clientes.

- Netlify te permite previsualizar cualquier despliegue que hagas o quieras hacer - esto te permite a ti y a tu equipo ver cómo se verán los cambios en producción sin tener que desplegarlos en tu sitio existente.
- Netlify proporciona una práctica función de envío de formularios que nos permite recoger información de los usuarios.

Tanto **Github**, como **Netlify**, como otras diferentes alternativas, pueden ser controlados desde el terminal de nuestro Linux, por lo que seguiremos el procedimiento de contectarnos vía SSH a nuestro Debian y realizar las operaciones por terminal.

2. Creación de nuestra aplicación

Tras loguearnos por SSH en nuestro Debian, nos crearemos un directorio para albergar la aplicación con el nombre que queramos. En ese directorio, crearemos los 3 archivos (dos .html y un .js), que conformarán nuestra sencilla aplicación de ejemplo:

head.html

```
<!DOCTYPE html>
<html>
<head>
        <title>Hola Mundo</title>
</head>
<body>
        <h1>Esta es la página principal</h1>
<p><a href="/tailPage">Ir a la siguiente página</a></p>
</body>
```

tail.html

```
<!DOCTYPE html>
<html>
<head>
        <title>Hola Mundo</title>
</head>
<body>
        <h1>FUNCIONA</h1>


</body>
```

aplicación.js

```
var http = require('http');
var fs = require('fs'); // para obtener los datos del archivo html
var port = process.env.PORT || 8080;


http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/html' });


    // req.url almacena el path o ruta de la URL
    var url = req.url;
    if (url === "/") {
// fs.readFile busca el archivo HTML
// el primer parámetro es el path al archivo HTML
// y el segundo es el callback de la función
// si el archivo no se encuentra, la función devuelve un error
// si el archivo se encuentra, el contenido del mismo se encuentra en pgres
        fs.readFile("head.html", function (err, pgres) {
            if (err)
                res.write("HEAD.HTML NOT FOUND");
            else {
                // Las siguientes 3 lineas
                // tienen la función de enviar el archivo html
                // y finalizar el proceso de respuesta
                res.writeHead(200, { 'Content-Type': 'text/html' });
                res.write(pgres);
                res.end();
            }
        });
    }
    else if (url === "/tailPage") {
        fs.readFile("tail.html", function (err, pgres) {
            if (err)
                res.write("TAIL.HTML NOT FOUND");
            else {
```

```
                res.writeHead(200, { 'Content-Type': 'text/html' });
                res.write(pgres);
                res.end();
            }
        });
    }

}).listen(port, function () {
    console.log("SERVER STARTED PORT: 8080");
});
```

Ahora, tal y como hacemos siempre a la hora de crear nuestra aplicación **Node.js**, con el fin de crear el archivo package.json, utilizaremos en el terminal el comando:

`npm init`

Podemos probar que nuestra aplicación funciona perfectamente en local:

`node aplicacion.js`

Y tras ello, debemos poder acceder, desde nuestra máquina anfitriona a `http://IP-maq-virtual:8080` (cuidado no tengas algún servicio abierto en ese puerto)

Ya con la aplicación creada y comprobada, podremos desplegarla en múltiples plataformas en la nube, como AWS, GCP, Azure, Digital Ocean, Heroku...

Para que nos funcione en la plataforma PaaS, en el archivo `package.json` que se nos ha creado al hacer el `npm init` debemos hacerle una modificación. En el bloque **`scripts`**, debemos borrar lo que haya dentro y dejar únicamente dentro de él:

`"start": "node aplicacion.js"`

De forma que el sitio donde la despleguemos sepa que comando utilizar para iniciar la aplicación tras desplegarla.

3. Aplicación para Netlify

Puesto que el interés en este módulo radica en el proceso de despliegue, suponiendo que la parte de desarrollo ya es abordada en otros módulos, vamos a utilizar una aplicación de ejemplo que nos ahorre tiempo para centrarnos en el despliegue.

Nos clonaremos este repositorio:

```
git clone https://github.com/StackAbuse/color-shades-generator
```

4. Proceso de despliegue en Netlify

Por mera curiosidad y ambición de aprendizaje, vamos a ver dos métodos de despliegue en Netlify:

- Despliegue manual desde el CLI de Netlify, es decir, desde el terminal, a partir de un directorio local de nuestra máquina.
- Despliegue desde un código publicado en uno de nuestros repositorios de Github

El primero nos permitirá conocer el CLI de Netlify y el segundo nos acercará más a una experiencia real de despliegue.

Vuestra primera tarea será registraros en Netlify con vuestro email y decirle que no cuando os pida enlazar con vuestra cuenta de Github (lo haremos más adelante).

4.1 Despliegue mediante CLI

Una vez registrados, debemos instalar el CLI de Netlify para ejecutar sus comandos desde el terminal:

```
npm install netlify-cli -g
```

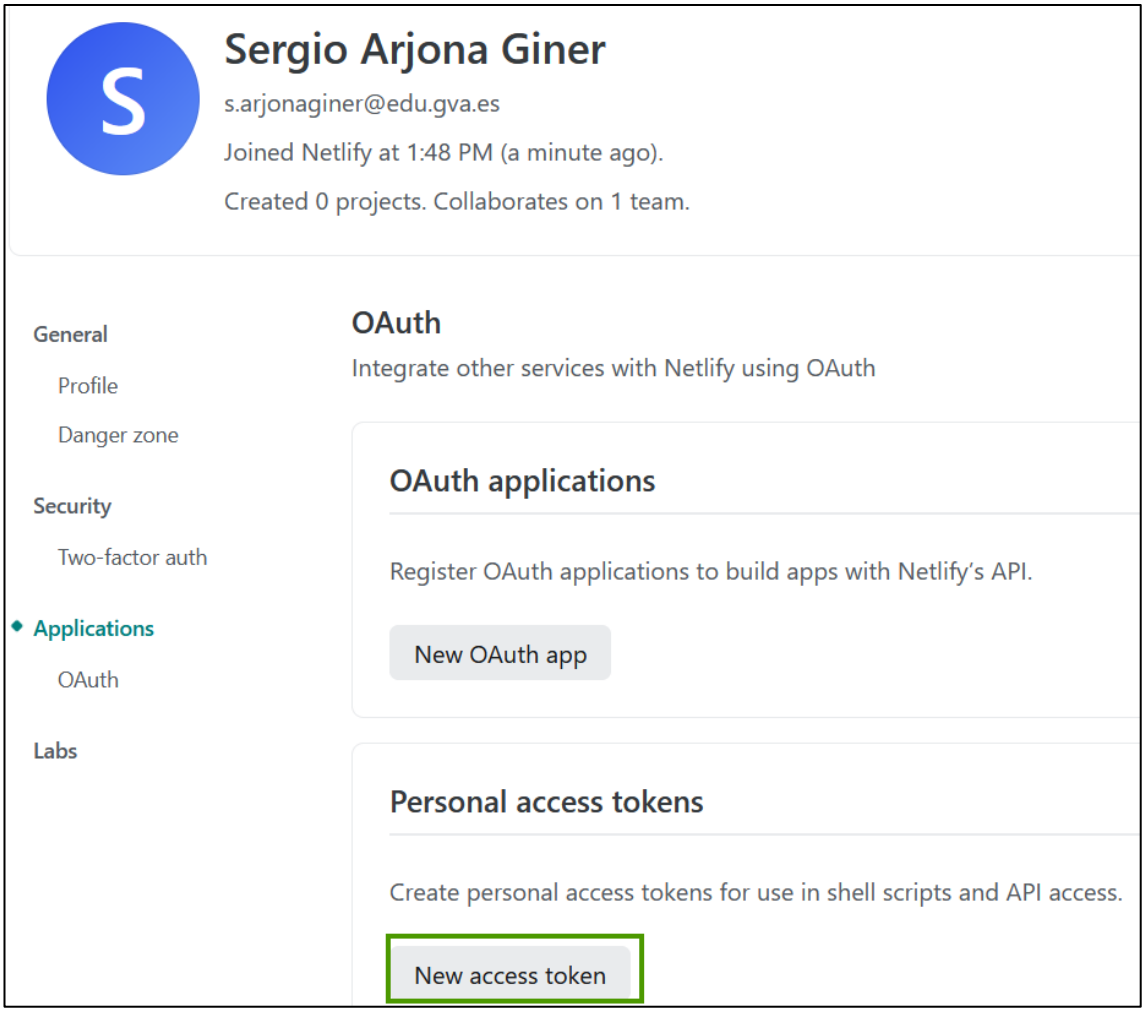
Está claro que para realizar acciones de deploy, Netlify nos solicitará una autenticación, esto se hace mediante el comando:

```
netlify login
```

El cual nos muestra una pantalla del navegador para que concedamos la autorización pertinente. Sin embargo, recordemos el problema de que estamos conectados por SSH a nuestro servidor y no tenemos la posibilidad del uso de un entorno gráfico.

En este caso, siguiendo las instrucciones de la documentación:

1 Generamos un token de acceso, que nos permitirá autenticarnos contra Netlify



2 Lo establecemos como variable de ambiente en nuestro servidor

```
sergio@Debian-DAW:~/color-shades-generator$ export NETLIFY_AUTH_TOKEN=nfp_
SQ
sergio@Debian-DAW:~/color-shades-generator$ echo $NETLIFY_AUTH_TOKEN
nfp_qa
```

3 Y nos logeamos

```
netlify login
```

Bueno, tenemos el código de nuestra aplicación, tenemos nuestra cuenta en Netlify y tenemos el CLI necesario para ejecutar comandos desde el terminal en esa cuenta... ¿Podemos proceder al despliegue sin mayores complicaciones?

La respuesta es **NO**, como buenos desarrolladores y en base a experiencias anteriores, ya sabéis que hay que hacer un *build* de la aplicación para, posteriormente, desplegarla. Vamos a ello.

En primer lugar, como sabemos, debemos instalar todas las dependencias que vienen indicadas en el archivo `package.json`:

```
npm install
```

Y cuando ya las tengamos instaladas podemos proceder a realizar el *build*:

```
npm run build
```

Esto nos creará una nueva carpeta llamada *build* que contendrá la aplicación que debemos desplegar. Y ya podemos hacer un *pre-deploy* de la aplicación:

```
netlify deploy
```

Nos hará algunas preguntas para el despliegue:

- Indicamos que queremos crear y configurar un nuevo site
- El *Team* lo dejamos por defecto
- Le indicamos el nombre que queremos emplear para la web (*nombre-practica3-4*) y el directorio a utilizar para el *deploy* (directorio `./build`).

```
sergio@Debian-DAW:~/prueba/color-shades-generator$ netlify deploy
This folder isn't linked to a project yet

To create and deploy in one go, use: netlify deploy --create-site <SITE_NAME>
? What would you like to do? +  Create & configure a new project
? Team: IES Doctor Balmis
? Project name (leave blank for a random name; you can change it later):
sergioarjona-practica3-3
 ›   Warning: sergioarjona-practica3-3.netlify.app already exists. Please try a different slug.
? Project name (leave blank for a random name; you can change it later):
sergio@Debian-DAW:~/prueba/color-shades-generator$ netlify deploy
This folder isn't linked to a project yet

To create and deploy in one go, use: netlify deploy --create-site <SITE_NAME>
? What would you like to do? +  Create & configure a new project
? Team: IES Doctor Balmis
? Project name (leave blank for a random name; you can change it later):
sergioarjona-practica3-3

Project Created

Admin URL:  https://app.netlify.com/projects/sergioarjona-practica3-3
URL:        https://sergioarjona-practica3-3.netlify.app
```

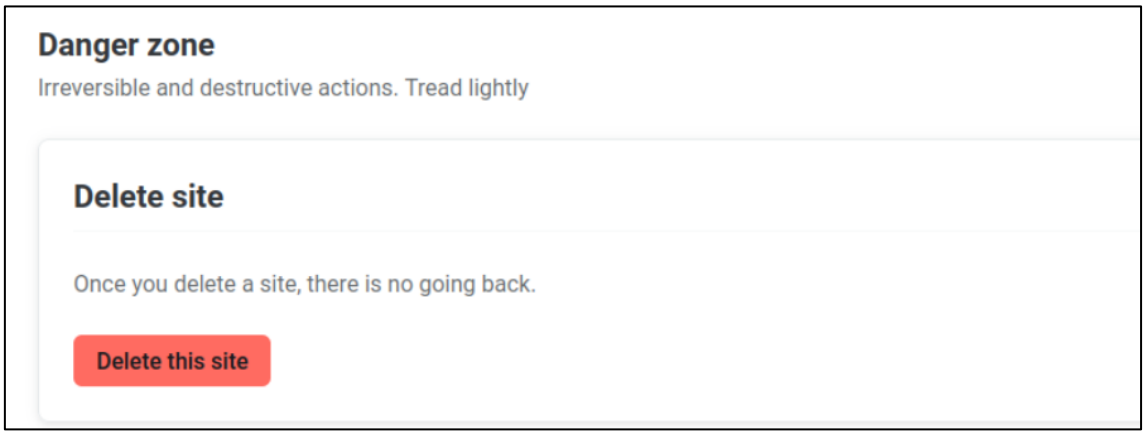
Y si nos indica que todo ha ido bien e incluso podemos ver el "borrador" (*Website Draft URL*) de la web que nos aporta, podemos pasarla a producción finalmente tal y como nos indica la misma salida del comando:

```
If everything looks good on your draft URL, deploy it to your main site URL
with the --prod flag.

netlify deploy --prod
```

Ahora comprueba que puedes acceder mediante la URL.

4.2 Despliegue mediante conexión con Github

En primer lugar, vamos a eliminar el site que hemos desplegado antes en Netlify para evitarnos cualquier problema y/o conflicto:



En segundo lugar, vamos a borrar el directorio donde se halla el repositorio clonado en el paso anterior para así poder empezar de 0:

```
rm -rf directorio_repositorio
```

Como queremos simular que hemos picado el código a man o en local y lo vamos a subir a Github por primera vez, nos descargaremos los fuentes en formato .zip sin que tenga ninguna referencia a Github:

```
wget https://github.com/StackAbuse/color-shades-generator/archive/refs/heads/main.zip
```

Creamos una carpeta nueva y descomprimimos dentro el zip:

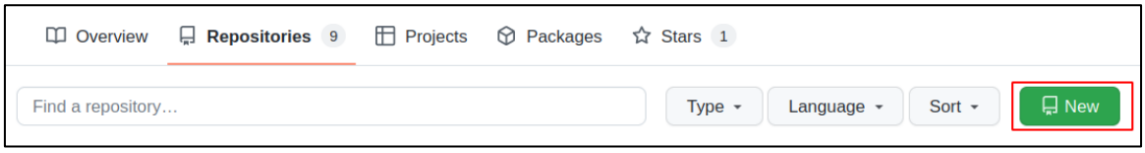
```
mkdir practica3.3
```

```
unzip main.zip -d practica3.4/
```

Entramos en la carpeta donde está el código:

```
cd practica3.4/color-shades-generator-main/
```

Ahora debemos crear un repositorio **completamente vacío** en Github que se llame practicaTresTres:



Y tras ello, volviendo al terminal a la carpeta donde estábamos, la iniciamos como repositorio, añadimos todo el contenido de la misma para el `commit`, hacemos el `commit` con el mensaje correspondiente y creamos la rama `main`:

```
$ git init
```

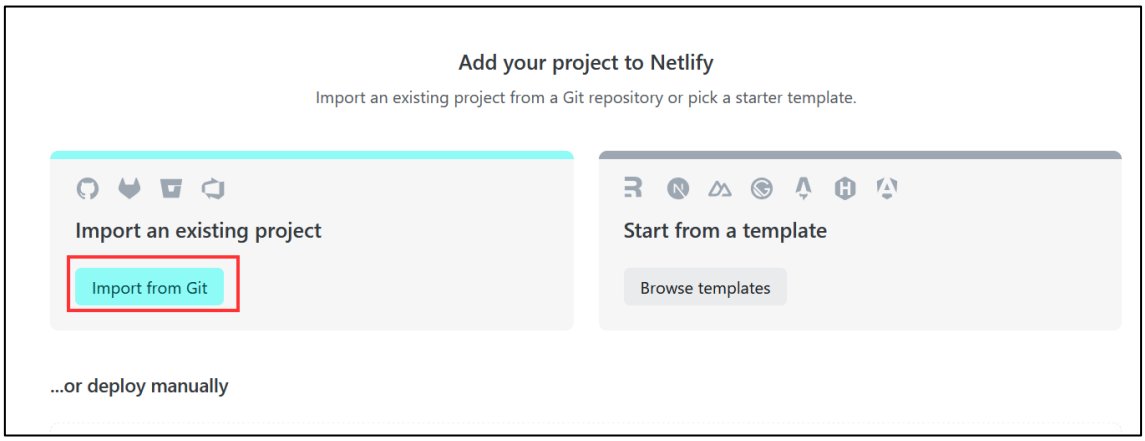
```
$ git add .
```

```
$ git commit -m "Subiendo el código..."
$ git branch -M main
```

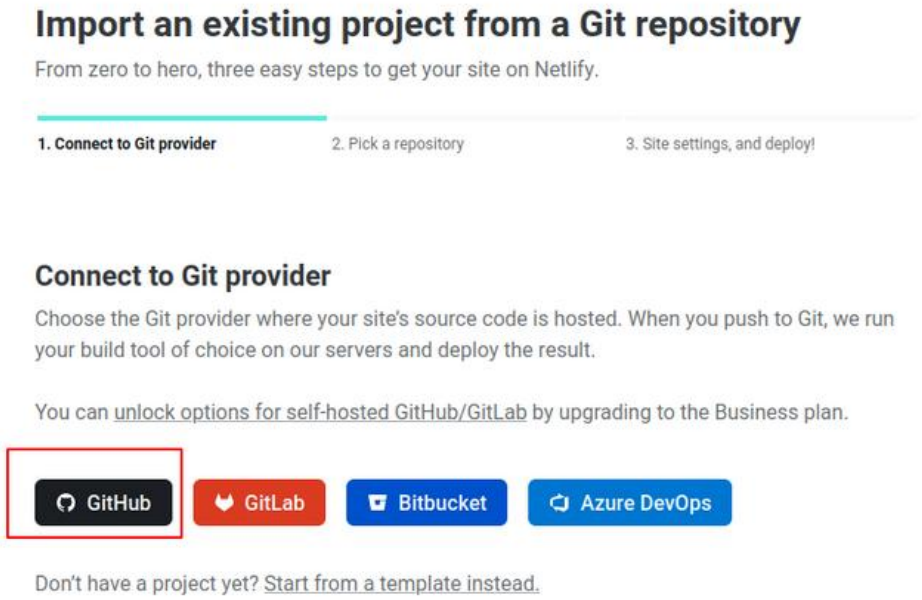
Y ahora sólo queda referenciar nuestra carpeta al repositorio recién creado en Github y hacer un `push` para subir todo el contenido del `commit` a él:

```
$ git remote add origin https://github.com/username/practicaTresCuatro.git
$ git push -u origin main
```

Ahora que ya tenemos subido el código a GitHub, de alguna manera debemos *enganchar* o enlazar nuestra cuenta de Github con la de Netlify para que éste último pueda traerse el código de allí, hacer el build y desplegarlo. Así pues, entramos en nuestro *dashboard* de Netlify y le damos a importar proyecto existente de git:

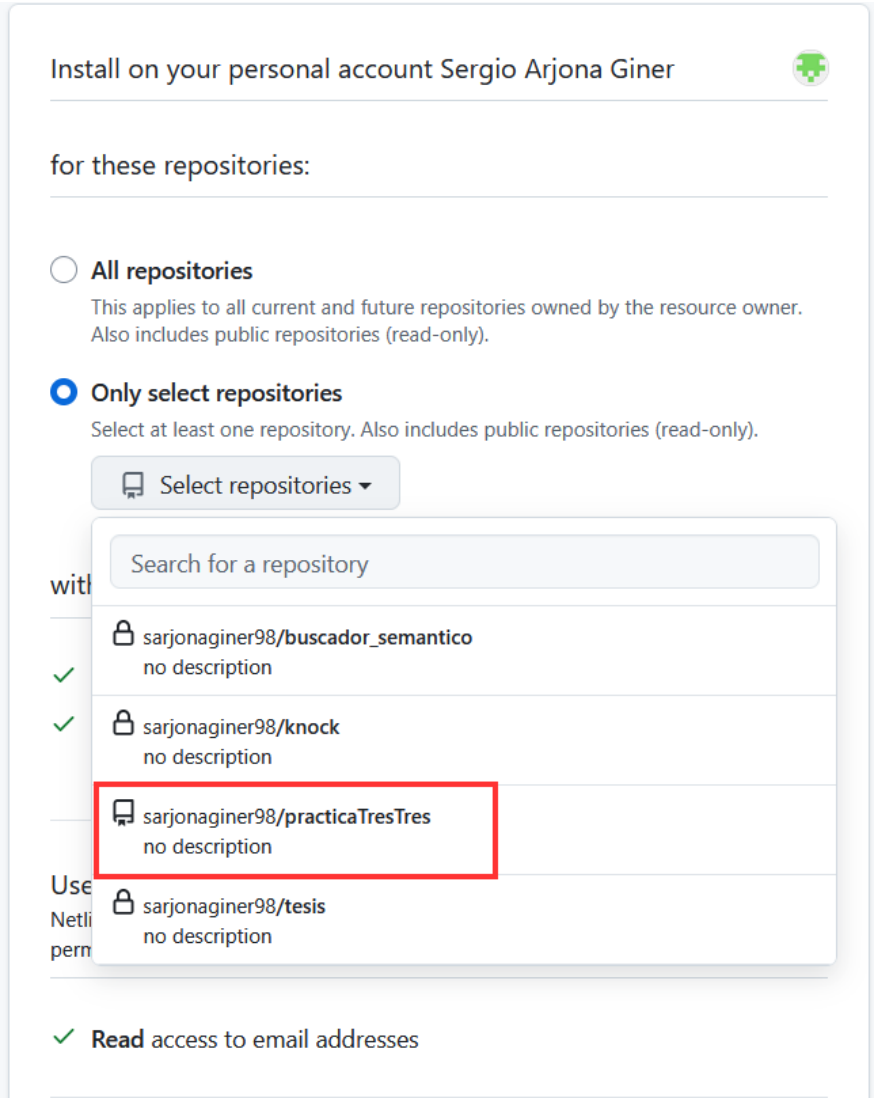


Le indicamos que concretamente de Github:

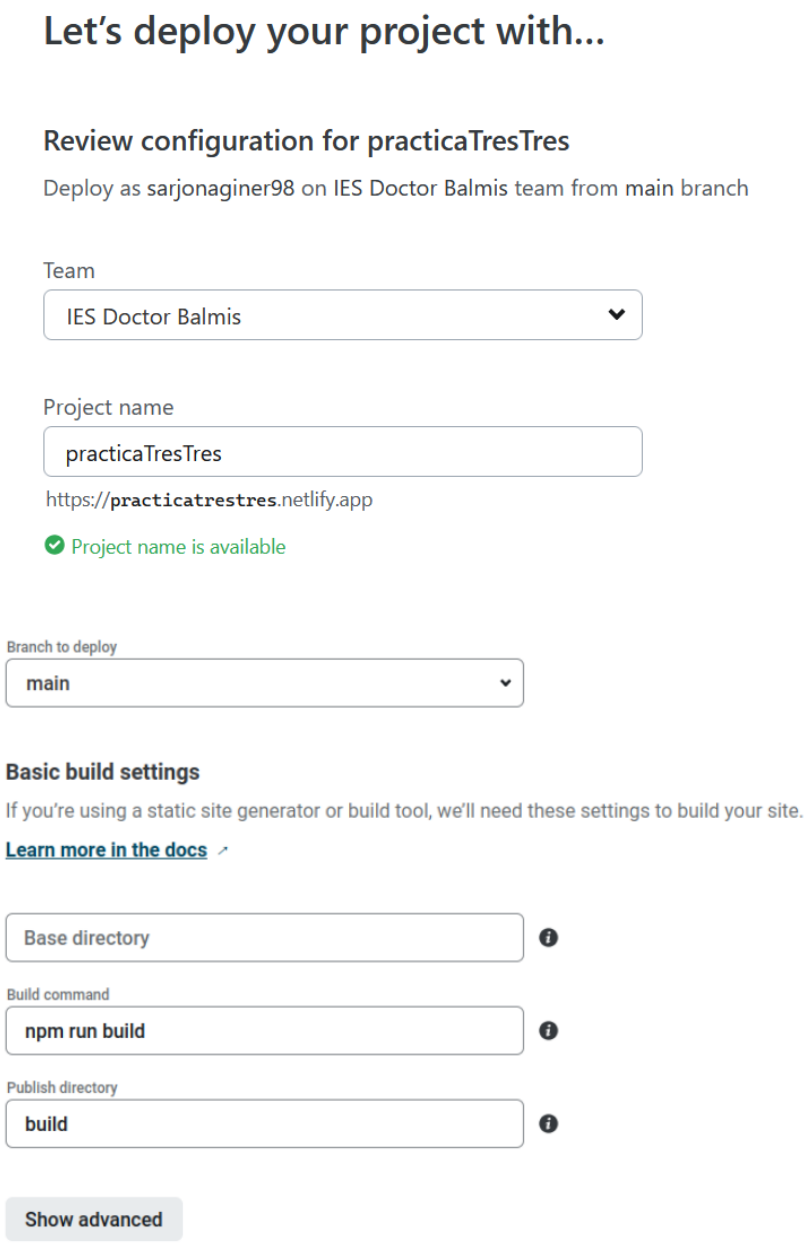


Y nos saltará una ventana pidiendo que autoricemos a Netlify a acceder a nuestros repositorios de Github.

Y luego le indicaremos que **NO acceda a TODOS** nuestros repositorios sino **sólo al repositorio que necesitamos**, que es donde tenemos el código de nuestra aplicación. Y ya quedará todo listo.



Y desplegamos la aplicación:



Netlify se encargará de hacer el *build* de forma automática tal y como hemos visto en la imagen de arriba, con el comando `npm run build`, publicando el contenido del directorio `build`.

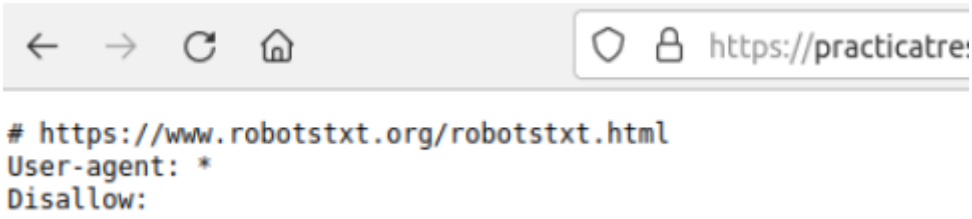
Tras el deploy, en "Site settings" podeís y debéis cambiar el nombre de la aplicación por **nombre-practica3-3**, donde *nombre* es vuestro nombre.

Lo que hemos conseguido de esta forma es que, cualquier cambio que hagamos en el proyecto y del que hagamos `commit` y `push` en Github, automáticamente genere un

nuevo despliegue en Netlify. Es el principio de lo que más adelante veremos como *despliegue continuo*.

Comprobemos que realmente es así:

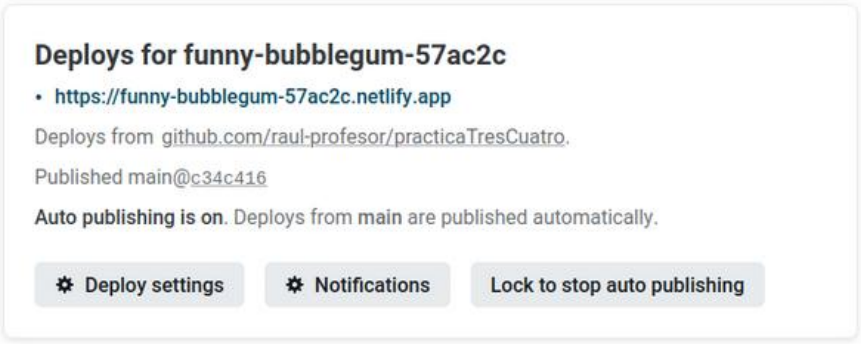
- Dentro de la carpeta *public* encontramos el archivo *robots.txt*, cuyo cometido es indicar a los rastreadores de los buscadores a qué URLs del sitio pueden acceder. A este archivo se puede acceder a través de la URL del site: **`https://dominio/robots.txt`**

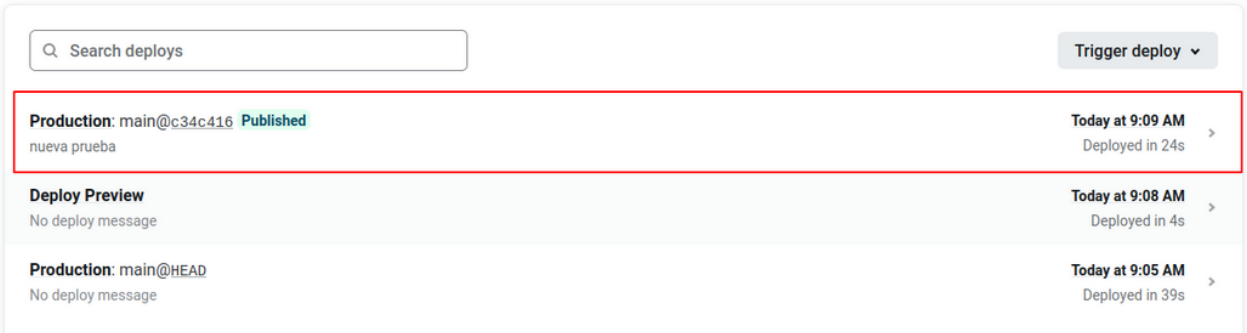


- Dentro de la carpeta *public*, utilizando el editor de texto que prefiráis en vuestro terminal, modificad el archivo `robots.txt` para que excluya un directorio que se llame **`nombre_apellido`**, utilizando obviamente vuestro nombre y apellido.

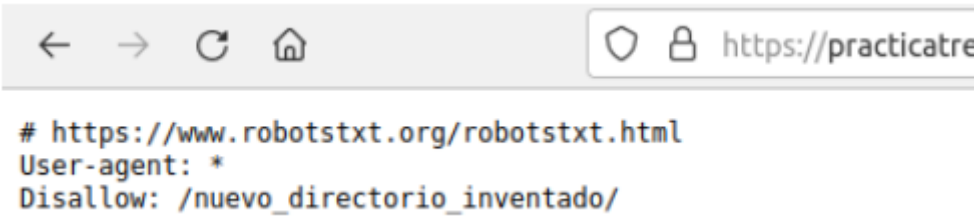
```
User-agent: *

Disallow: /nombre_y_apellido/
```

- Haz un nuevo *commit* y *push* (del caso anterior, recuerda el comando git previo para añadir los archivos a hacer *commit*)
- Comprueba en el *dashboard* de Netlify que se ha producido un nuevo *deploy* de la aplicación hace escasos segundos





- Accede a `https://url_de_la_aplicacion/robots.txt` y comprueba que, efectivamente, se ve reflejado el cambio:



TAREAS

1. Documenta la realización de toda esta práctica adecuadamente, con las explicaciones y justificaciones necesarias y las capturas de pantalla pertinentes.

2. Repite y replica este proceso de despliegue en Vercel, otra plataforma PaaS.

Recuerda grabar un vídeo en que se muestre el despliegue tanto por CLI como por github.