

Despliegue y CI / CD con netlify

- **Nombre:** Ridel
- **Apellidos:** Saavedra Flores

índice

1. Introducción.....	3
1.1 ¿Qué es Netlify?.....	3
1.2 Objetivo.....	3
2. Instalación y configuración.....	3
3. Despliegue de una app con vue.js.....	4
.....	5
.....	8
4. CI/CD Pipeline con Netlify y github actions.....	8
4.1 Configurar el entorno.....	8
4.2 Configurar el github workflow.....	10
4.3 Testeando el Workflow.....	13

1. Introducción

1.1 ¿Qué es Netlify?

Netlify es una plataforma la cual nace de la necesidad de automatizar proyectos webs estáticos. Esta herramienta aúna en la tareas de integración continua y de despliegue de infraestructuras web en un flujo de ejecución único, es decir que todo se ejecuta en un único proceso

1.2 Objetivo

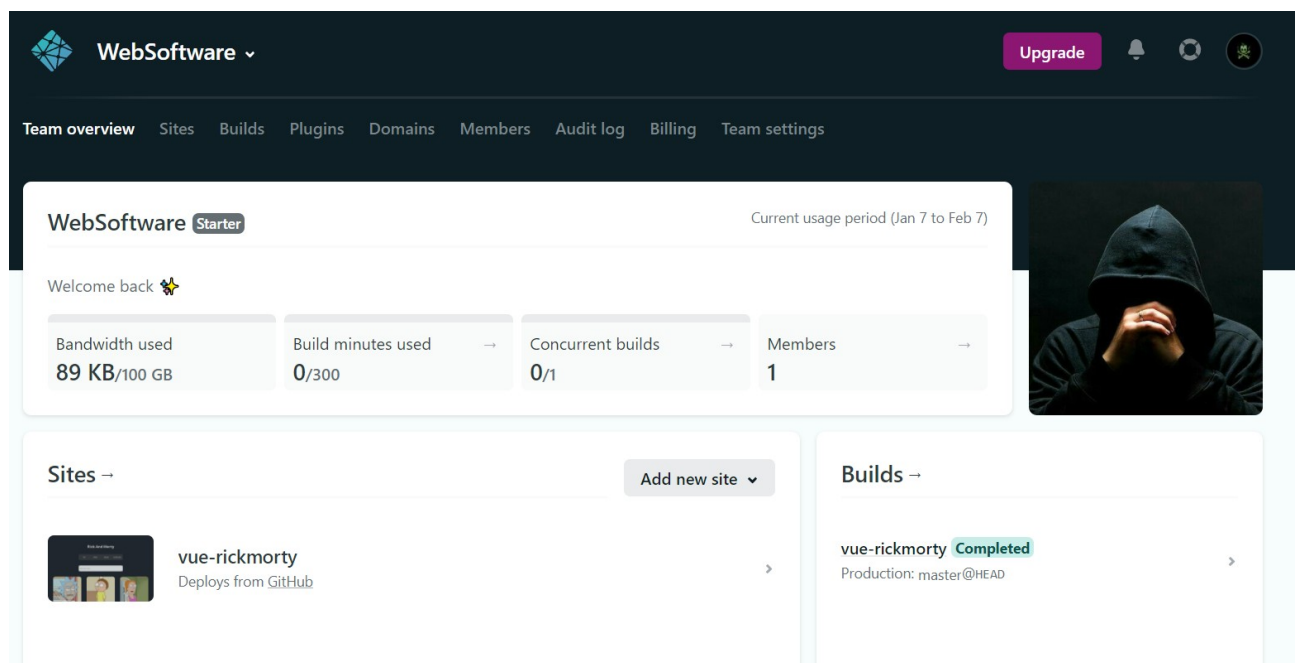
El objetivo de esta herramienta, al igual que de cualquier otra del mismo tipo, es automatizar el despliegue de aplicaciones con la implementación de test de integración continua

2. Instalación y configuración

Lo primero que tenemos que hacer es registrarnos, para ello vamos a ir a la siguiente dirección: <https://www.netlify.com/> y le daremos al botón de **Sign Up** y seguir los pasos de registro



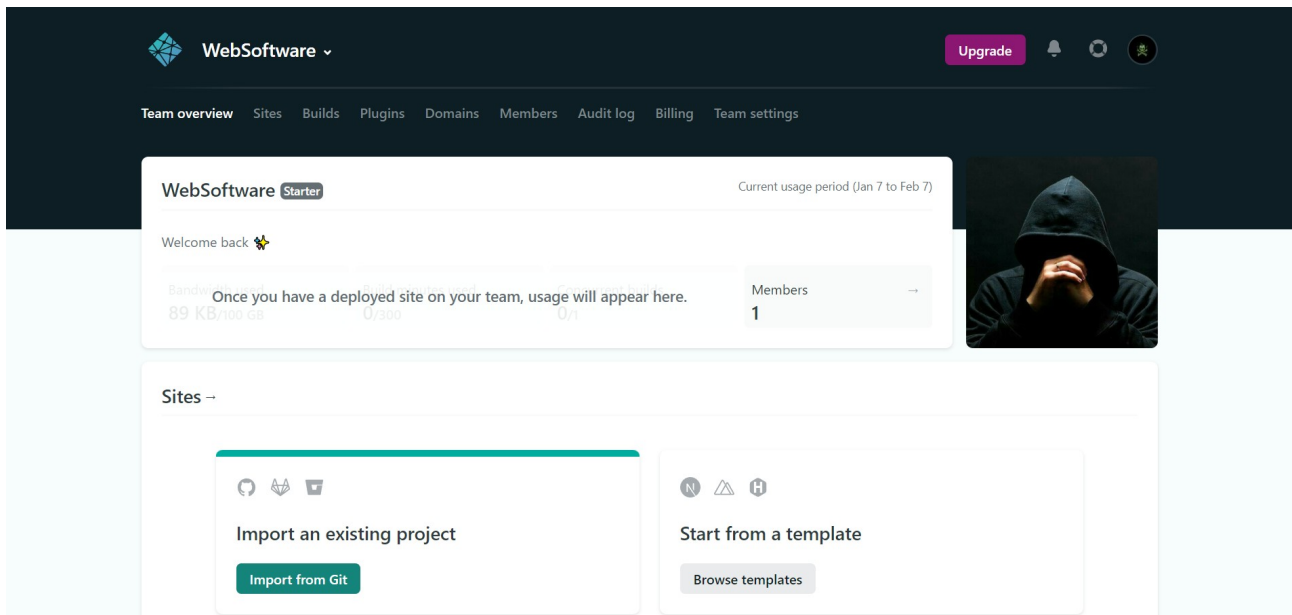
y ya podríamos acceder al dashboard



nos pedirá algunas configuraciones, como un nombre de equipo y algunas otras más

3. Despliegue de una app con vue.js

Una vez echo los pasos del apartado anterior, deberíamos tener un dashboard como el siguiente:



ahora, para seleccionar el repositorio, le daremos a **import from Git** y nos saldrá algo como lo siguiente:

Import an existing project from a Git repository

From zero to hero, three easy steps to get your site on Netlify.

1. Connect to Git provider
2. Pick a repository
3. Site settings, and deploy!

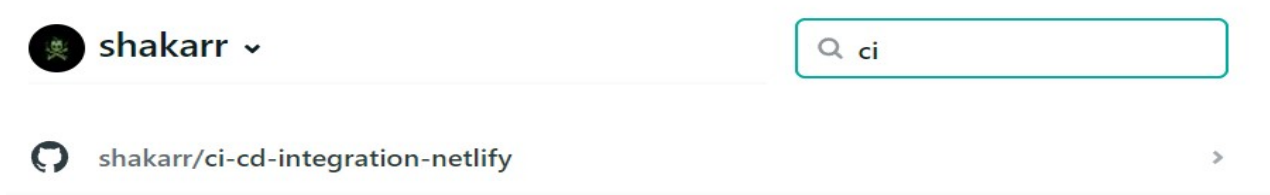
Connect to Git provider

Choose the Git provider where your site's source code is hosted. When you push to Git, we run your build tool of choice on our servers and deploy the result.

You can [unlock options for self-hosted GitHub/GitLab](#) by upgrading to the Business plan.



seleccionaremos GitHub en nuestro caso, y seleccionamos el repositorio que queremos desplegar:



Site settings for shakarr/ci-cd-integration-netlify

Get more control over how Netlify builds and deploys your site with these settings.

Owner

WebSoftware



Branch to deploy

master



Basic build settings

If you're using a static site generator or build tool, we'll need these settings to build your site.

[Learn more in the docs](#) ↗

Base directory



Build command

npm run build



Publish directory

build



luego seleccionaremos, el nombre del equipo en caso de haber más de uno, la rama en la cual vamos a desplegar, y el comando para construir la app, en esta caso: **yarn build** y le damos a deploy

Deploy site

nos saldrá algo como lo siguiente:

gracious-shaw-52a11e

- [Site deploy in progress](#)

Deploys from [GitHub](#). Created at 7:15 PM.



⚙ Site settings

⚙ Domain settings

y como podemos ver se ha desplegado correctamente

pedantic-einstein-0f8663

- <https://pedantic-einstein-0f8663.netlify.app>

Builds are stopped. Last published at 3:32 PM.



Activate builds

⚙ Site settings

⚙ Domain settings

si queremos ver el output de la consola solo tendríamos que hacer click **Production** y podríamos ver la consola

Deploy log

Preview



```
1 7:15:33 PM: Build ready to start
2 7:15:36 PM: build-image version: 122b31996ccaffd45d820a452d6227f8312110cc (focal)
3 7:15:36 PM: build-image tag: v4.5.3
4 7:15:36 PM: buildbot version: 0854df8549ceb2ae5c3f0bb7326040a5c2ced0c5
5 7:15:36 PM: Fetching cached dependencies
6 7:15:36 PM: Failed to fetch cache, continuing with build
7 7:15:36 PM: Starting to prepare the repo for build
8 7:15:36 PM: No cached dependencies found. Cloning fresh repo
9 7:15:36 PM: git clone https://github.com/shakarr/rick-morty-vue
10 7:15:37 PM: Preparing Git Reference refs/heads/master
11 7:15:37 PM: Parsing package.json dependencies
12 7:15:38 PM: Starting build script
13 7:15:38 PM: Installing dependencies
```

En caso de que quisiéramos cambiar el nombre de dominio por uno mas de nuestro agrado, solo tendríamos que ir a: **Domain Settings > Options > Edit site name**

pedantic-einstein-0f8663

- <https://pedantic-einstein-0f8663.netlify.app>

Builds are stopped. Last published at 3:32 PM.



Activate builds

⚙ Site settings

⚙ Domain settings

Domains

Use your own domain for your Netlify site for free

Custom domains

By default, your site is always accessible via a Netlify subdomain based on the site name. Custom domains allow you to access your site via one or more non-Netlify domain names.

- pedantic-einstein-0f8663.netlify.app
Default subdomain

Options ^

[Learn more about custom domains in the docs](#) ↗

Edit site name

Add custom domain

y nos saldrá un panel como el siguiente:

Change site name

The site name determines the default URL for your site. Only alphanumeric characters and hyphens are allowed.

Site name

pedantic-einstein-0f8663

<https://pedantic-einstein-0f8663.netlify.app>

Save

Cancel

y como vemos ya se ha cambiado el nombre de nuestra app

Settings for ci-cd-integration-netlify

ci-cd-integration-netlify.netlify.app

Deploys from [GitHub](#). Owned by [WebSoftware](#).

Last update at 3:49 PM (a few seconds ago)



4. CI/CD Pipeline con Netlify y github actions

4.1 Configurar el entorno

Como ya tenemos la app desplegada, vamos a hacer lo siguiente, nos vamos a ir a opciones y vamos a deshabilitar la construcción automática

Build settings

Repository:

[Link to a different repository →](#)

Base directory:

For monorepos or sites built from a subdirectory of a repository, the directory to change to before starting a build.

Build command:

Publish directory:

Deploy log visibility:

☒ **Public logs**
Anyone with a deploy detail URL will be able to access the logs.

☐ **Private logs**
Only site administrators will be able to access the logs.

Builds:

☐ **Activate builds**
Netlify will build your site according to your continuous deployment settings when you push to your Git provider.

☒ **Stop builds**
Netlify will never build your site. You can build locally via the CLI and then publish new deploys manually via the CLI or the API.

⚠ Your site won't be updated by Netlify until you activate builds.

lo siguiente que vamos a hacer es generar un token de netlify y añadirlo a github, para ello, en netlify no iremos a **User Settings > Applications > New access token**

Personal access tokens

Create personal access tokens for use in shell scripts and API access.

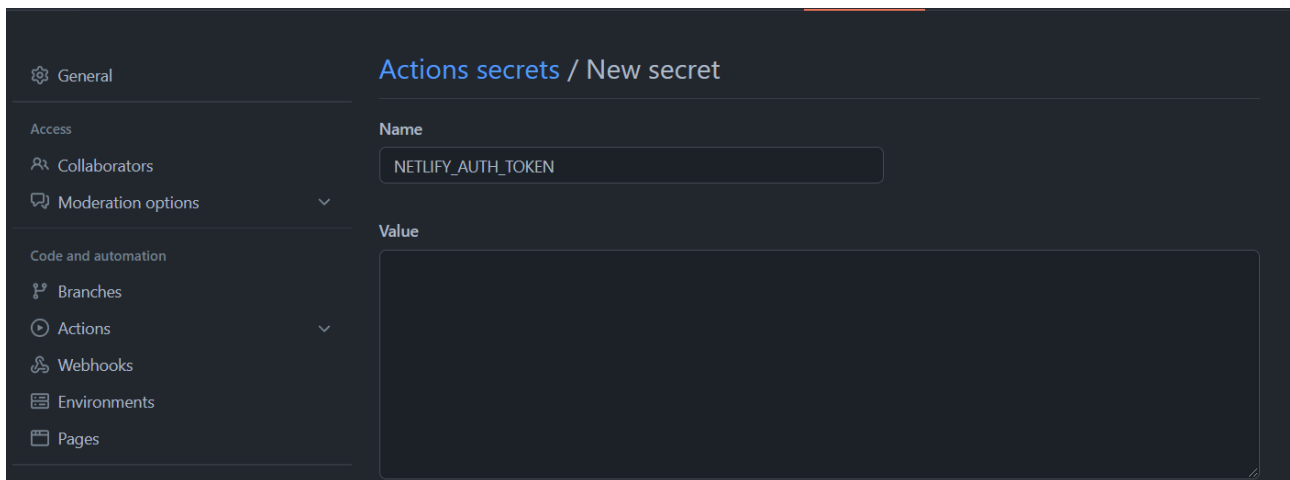
ci-cd-integration-netlify

Created at 4:02 PM (a few seconds ago)

Options ▾

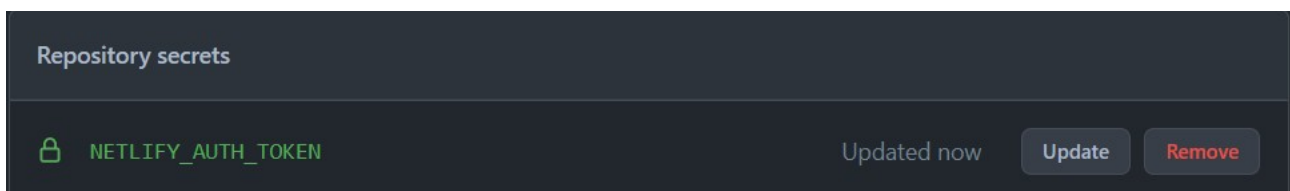
New access token

ahora en github no iremos a **Settings > Secrets > Actions > New Secret**



The screenshot shows the GitHub 'Actions secrets / New secret' page. On the left is a sidebar with navigation links: General, Access, Collaborators, Moderation options, Code and automation, Branches, Actions, Webhooks, Environments, and Pages. The main area has a 'Name' field containing 'NETLIFY_AUTH_TOKEN' and a large 'Value' text area below it.

y pondremos el token en el value y ya lo tendríamos guardado






The screenshot shows the 'Repository secrets' section in GitHub. It displays a single secret named 'NETLIFY_AUTH_TOKEN' with a green lock icon. To the right of the secret name, it says 'Updated now' and there are 'Update' and 'Remove' buttons.

ahora vamos a guardar el id de la app como un secreto también, para ello no iremos a netlify y luego a: **Settings > Site information > app id**

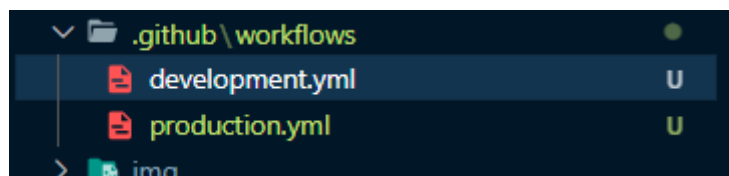
Site name:	ci-cd-integration-netlify
Owner:	WebSoftware
API ID:	f0297721-fd38-4159-95f8-d4853c3fc034

y lo añadimos como hicimos con el token, y para un id de dev también

Repository secrets			
	DEV_NETLIFY_SITE_ID	Updated now	<button>Update</button> <button>Remove</button>
	NETLIFY_AUTH_TOKEN	Updated 22 minutes ago	<button>Update</button> <button>Remove</button>
	NETLIFY_SITE_ID	Updated 19 minutes ago	<button>Update</button> <button>Remove</button>

4.2 Configurar el github workflow

Dentro de la carpeta **.git** vamos a crear un directorio llamado **workflow** para ello hacemos un **mkdir .github/workflows**



ahora dentro de la carpeta vamos a crear el archivo **development.yml**:

```
.git > workflows > development.yml
1  # Name of workflow
2  name: Development workflow
3
4  # When workflow is triggered
5  on:
6    pull_request:
7      branches:
8        - dev
9
10 # Jobs to carry out
11 jobs:
12   deploy:
13     # Operating system to run job on
14     runs-on: ubuntu-latest
15
16     # Steps in job
17     steps:
18       # Get code from repo
19       - name: Checkout code
20         uses: actions/checkout@v1
21       # Install NodeJS
22       - name: Use Node.js 12.x
23         uses: actions/setup-node@v1
24         with:
25           node-version: 12.x
26       # Run npm install and build on our code
27       - run: npm install
28       - run: npm run build --if-present
29       # Deploy to Netlify using our dev secrets
30       - name: Deploy to netlify
31         uses: netlify/actions/cli@master
32         env:
33           NETLIFY_AUTH_TOKEN: ${ secrets.NETLIFY_AUTH_TOKEN }
34           NETLIFY_SITE_ID: ${ secrets.DEV_NETLIFY_SITE_ID }
35         with:
36           args: deploy --dir=build --prod
37           secrets: ["DEV_NETLIFY_AUTH_TOKEN", "NETLIFY_SITE_ID"]
```

y luego tendremos también el archivo **production.yml**:


```
.github > workflows > production.yml
1  # Name of workflow
2  name: Production workflow
3
4  # When workflow is triggered
5  on:
6    push:
7      tags:
8        - "v*"
9
10 # Jobs to carry out
11 jobs:
12   deploy:
13     # Operating system to run job on
14     runs-on: ubuntu-latest
15     # Steps in job
16     steps:
17       # Get code from repo
18       - name: Checkout code
19         uses: actions/checkout@v1
20       # Install NodeJS
21       - name: Use Node.js 12.x
22         uses: actions/setup-node@v1
23         with:
24           node-version: 12.x
25       # Run npm install and build on our code
26       - run: npm install
27       - run: npm run build --if-present
28       # Deploy to Netlify using our production secrets
29       - name: Deploy to netlify
30         uses: netlify/actions/cli@master
31         env:
32           NETLIFY_AUTH_TOKEN: ${ secrets.NETLIFY_AUTH_TOKEN }
33           NETLIFY_SITE_ID: ${ secrets.NETLIFY_SITE_ID }
34         with:
35           args: deploy --dir=build --prod
36           secrets: ["NETLIFY_AUTH_TOKEN", "NETLIFY_SITE_ID"]
```

ahora vamos a crear un tag para probar el que todo funciona correctamente, para ello haremos lo siguiente:


```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN

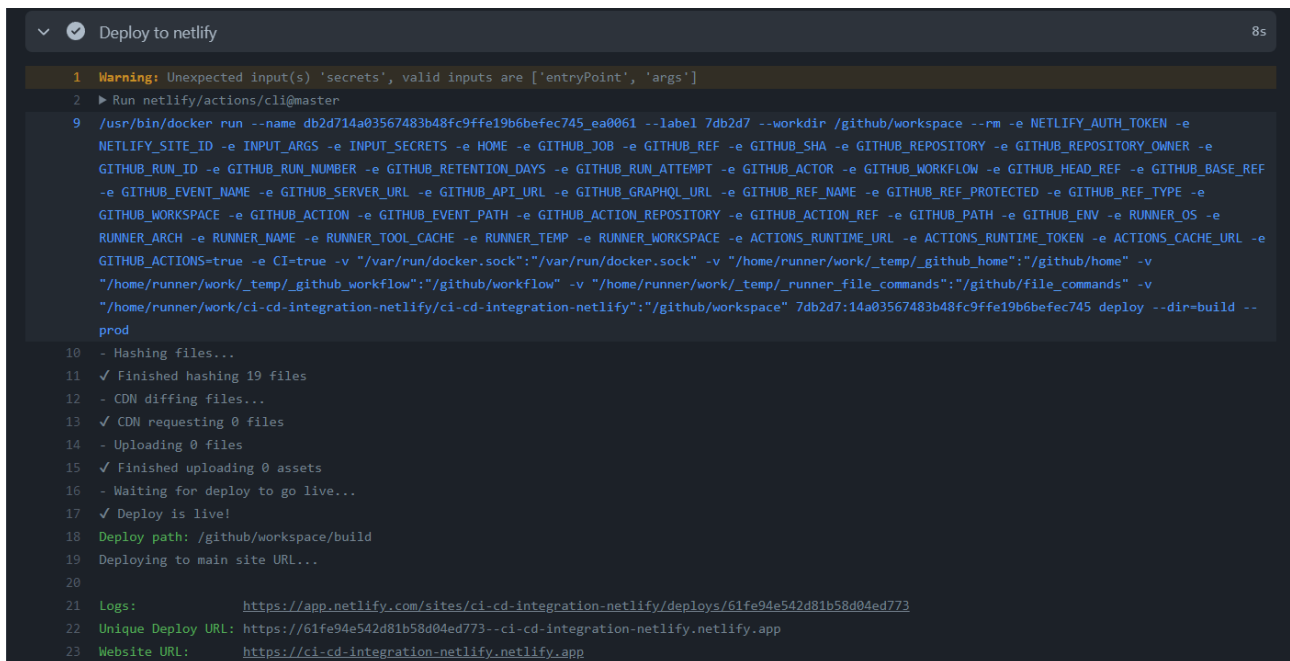
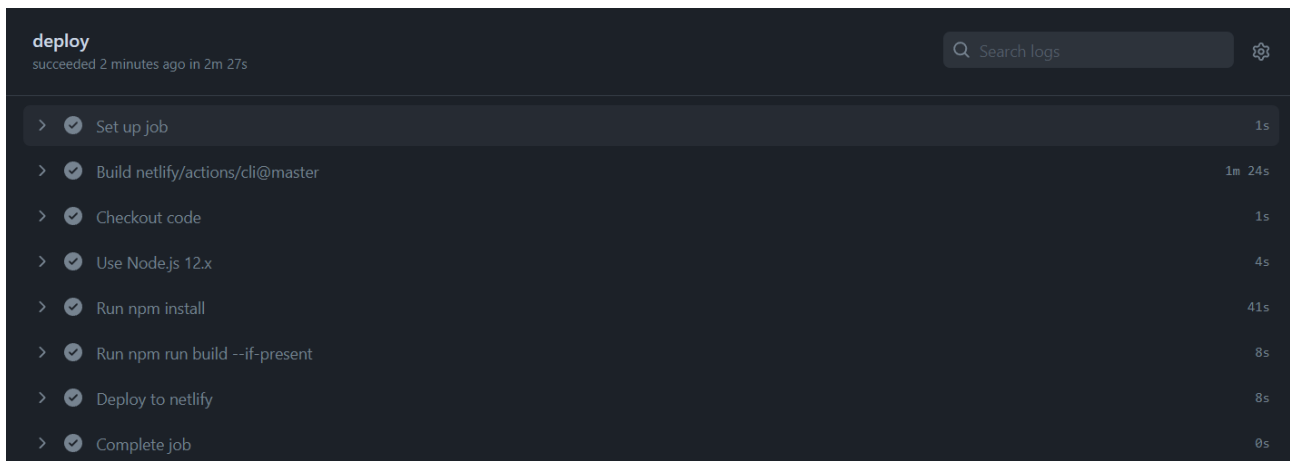
PS C:\Users\risaa\OneDrive\Escritorio\ci-cd-integration-netlify> git tag v0.1
PS C:\Users\risaa\OneDrive\Escritorio\ci-cd-integration-netlify> git push origin v0.1
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/shakarr/ci-cd-integration-netlify.git
 * [new tag]          v0.1 -> v0.1
PS C:\Users\risaa\OneDrive\Escritorio\ci-cd-integration-netlify> |
```

y como vemos todo ha ido correctamente, por lo que el archivo de **production.yml** funciona correctamente

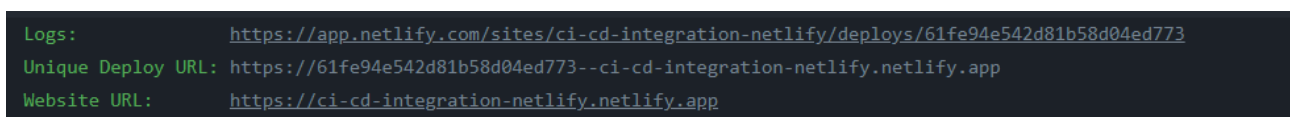
Triggered via push 5 minutes ago	Status	Total duration	Artifacts
 shakarr pushed -> 97ea7ff v0.1	Success	2m 39s	—

production.yml
on: push

 **deploy** 2m 27s

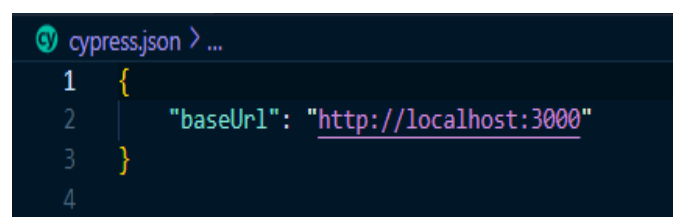
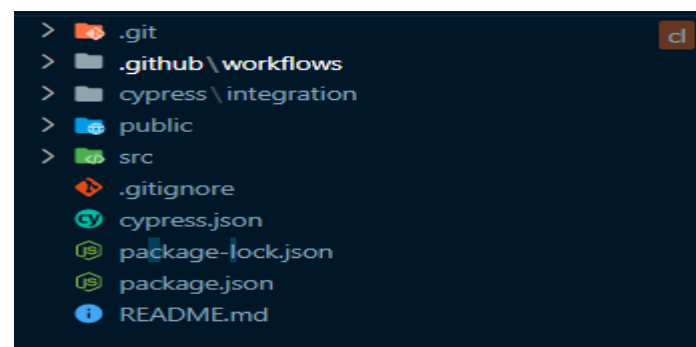


como podemos ver ha pasado el deploy y se ha desplegado en netlify



4.3 Testeando el Workflow

Lo primero que haremos será crear el archivo **cypress.js** en el root del proyecto



luego vamos a crear una carpeta llamada **cypress/integration** la cual contendrá tests, en este caso vamos a crear el siguiente:

```
init.spec.js X
cypress > integration > init.spec.js > ...
1 describe('Cypress', () => {
2   it('is working', () => {
3     expect(true).to.equal(true);
4   });
5
6   it('visits the app', () => {
7     cy.visit('/');
8   });
9 });
10
```

acto seguido, instalamos cypress, para ello hacemos un **npm install cypress**



```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN
PS C:\Users\risaa\OneDrive\Escritorio\ci-cd-integration-netlify> npm install cypress
[.....] - fetchMetadata: sill resolveWithNewModule date-fns@1.30.1 checking installable status
```





y crearemos el archivo **test.yml** dentro del directorio de **workflow** con el siguiente contenido:




```
1 # Name of workflow
2 name: Test workflow
3
4 # Trigger workflow on all pull requests
5 on:
6   pull_request:
7     branches:
8       - dev
9       - master
10
11 # Jobs to carry out
12 jobs:
13   test:
14     # Operating system to run job on
15     runs-on: ubuntu-latest
16     # Steps in job
17     steps:
18       # Get code from repo
19       - name: Checkout code
20         uses: actions/checkout@v1
21       # Install NodeJS
22       - name: Use Node.js 12.x
23         uses: actions/setup-node@v1
24       with:
25         node-version: 12.x
26       # Build the app using cypress
27       - name: Cypress run
28         uses: cypress-io/github-action@v1
29       with:
30         build: npm run build
31         start: npm start
32         wait-on: http://localhost:3000
33         browser: chrome
```

esto lo que hará es que cada vez que hagamos un pull, cypress abrirá la app en google, en el puerto **3000** y comprobara que todo funciona correctamente



test integration #1

 **Open** shakarr wants to merge 1 commit into `master` from `dev` 

 Conversation **0**  Commits **1**  Checks **0**  Files changed **1**



 **shakarr** commented now Owner  

No description provided.





  initial commit 1caaa75

Test workflow

test.yml

 Filter workflow runs 

1 workflow run Event ▾ Status ▾ Branch ▾ Actor ▾

 **test integration** dev  13 seconds ago  In progress 

Test workflow #1: Pull request #1 opened by shakarr

2 **passing** (687ms)

(Results)

Tests:	2
Passing:	2
Failing:	0
Pending:	0
Skipped:	0
Screenshots:	0
Video:	true
Duration:	0 seconds
Spec Ran:	init.spec.js

(Run Finished)

Spec	Tests	Passing	Failing	Pending	Skipped
✓ init.spec.js	687ms	2	2	-	-
✓ All specs passed!	687ms	2	2	-	-