

Modulo 8: Fundamentos de Integración Continua

Clase 3: Continuos Deployment

Crear un Pipeline CI en GitHub Actions para un Proyecto Python/Django con despliegue automatico a render.com y utilización de base de datos PostgreSQL en neon.tech.

Parte 1: Preparar tu proyecto

Nos basaremos en un proyecto de ejemplo disponible en el repositorio [django-drf-cicd-example](https://github.com/tu-usuario/django-drf-cicd-example). Para trabajar lo primero que haremos será hacer un fork del repositorio. Luego vamos a clonar el nuevo repositorio creado en nuestro equipo.

```
git clone https://github.com/tu-usuario/django-drf-cicd-example.git
```

Luego vamos a crear el ambiente virtual e instalar las dependencias del proyecto.

```
# cambiar a la carpeta del proyecto
cd django-drf-cicd-example
# crear y activar el ambiente virtual
python -m venv venv
# activar el ambiente virtual en windows
./venv/Scripts/activate
# para activar el ambiente virtual en linux/osx
source venv/bin/activate
# instalar las dependencias
pip install -r requirements.txt
```

Luego vamos a crear el archivo .env desde el archivo .env.example y agregar las variables de entorno.

```
# crear el archivo .env
cp .env.example .env
```

Requerimos las siguientes variables de entorno:

- PGDATABASE
- PGUSER
- PGPASSWORD
- PGHOST
- PGPORT
- SECRET_KEY
- PRODUCTION_HOST
- DEBUG

De momento solo podemos configurar las variables `DEBUG=true` y `SECRET_KEY`. Ya que para las otras variables necesitamos una cuenta en Render y Neon.

Vamos a probar el proyecto localmente para verificar que todo esté funcionando correctamente. Para esto vamos a modificar el archivo settings que se utiliza de modo que localmente utilizará un bd sqlite.

```
# correremos las migraciones
python manage.py migrate --settings=myproject.settings_dev
# iniciamos el servidor
python manage.py runserver --settings=myproject.settings_dev
```

Si todo está bien, deberíamos poder acceder a la aplicación en `http://localhost:8000`

Creacion de cuentas

La manera más simple de crear cuentas en Render y Neon es hacerlo con una cuenta de Google o GitHub.

- [Render](#)
- [Neon](#)

Creación de BD en Neon

Al ingresar a Neon por primera vez nos pedirá crear un proyecto. Podemos elegir el nombre del proyecto y la región, además de la ultima versión de Postgres.

/ Get started with Neon

Project name

talentofuturo

Postgres version

17

Cloud Service Provider



AWS



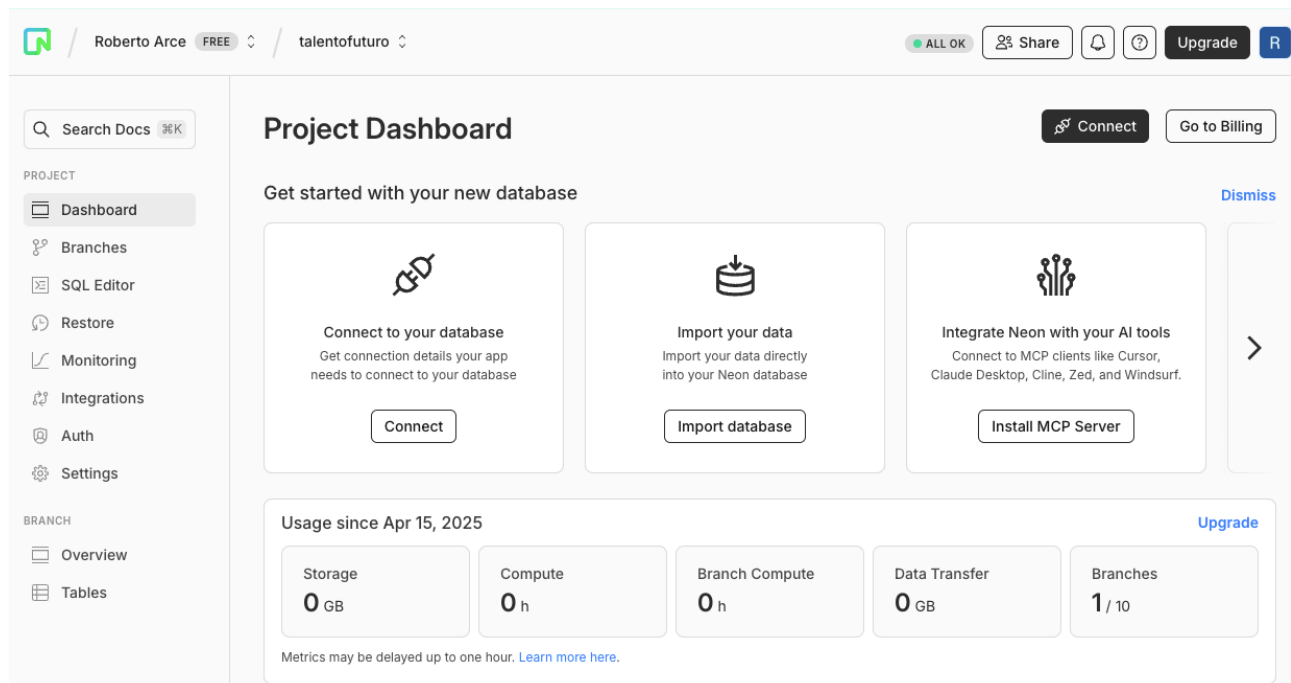
Azure

Region

AWS US East 1 (N. Virginia)

Select the region closest to your application.

Desde el dashboard de Neon podemos ver los detalles de nuestro proyecto, las tablas, queries, etc. Daremos click en "Connect" para obtener las credenciales de la base de datos.



The screenshot shows the Neon Project Dashboard for a project named 'talentofuturo'. The dashboard includes a sidebar with navigation options like Dashboard, Branches, SQL Editor, Restore, Monitoring, Integrations, Auth, and Settings. The main content area features a 'Project Dashboard' header with a 'Connect' button and a 'Go to Billing' link. Below this, there are three main action cards: 'Connect to your database', 'Import your data', and 'Integrate Neon with your AI tools'. At the bottom, there is a 'Usage since Apr 15, 2025' section showing metrics for Storage, Compute, Branch Compute, Data Transfer, and Branches.

Project Dashboard

Get started with your new database

Connect to your database
Get connection details your app needs to connect to your database
[Connect](#)

Import your data
Import your data directly into your Neon database
[Import database](#)

Integrate Neon with your AI tools
Connect to MCP clients like Cursor, Claude Desktop, Cline, Zed, and Windsurf.
[Install MCP Server](#)

Usage since Apr 15, 2025

Metric	Value
Storage	0 GB
Compute	0 h
Branch Compute	0 h
Data Transfer	0 GB
Branches	1 / 10

Metrics may be delayed up to one hour. [Learn more here.](#)

Connect to your database

Branch

main DEFAULT

Compute

Primary ACTIVE

Database

neondb

Role

neondb_owner

[Reset password](#)

Connection string

Connection pooling ⓘ

postgresql://neondb_owner:*****@ep-soft-moon-a4xzzsz0-pooler.us-east-1.aws.neon.tech/neondb?sslmode=require

Show password

Copy snippet

Your password is saved in a secure storage vault. [Learn more about connecting](#)

Close

Debemos guardar el string de conexión para utilizarlo en nuestro proyecto.

Neon nos proporciona un string de conexión para utilizarlo en nuestro proyecto. Este string se ve así:

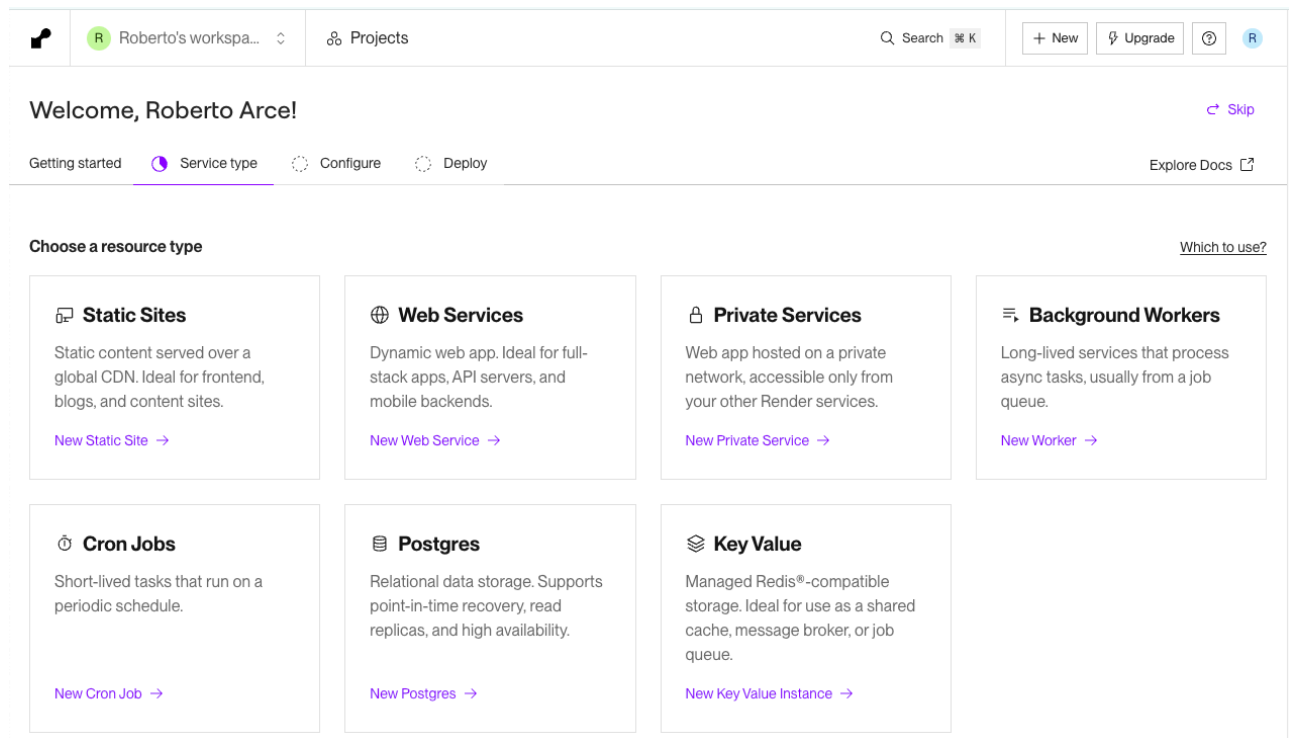
```
postgresql://user:password@host:port/database?sslmode=require
```

Desde este completaremos los valores de las variables de entorno de nuestro archivo .env. Por ejemplo:

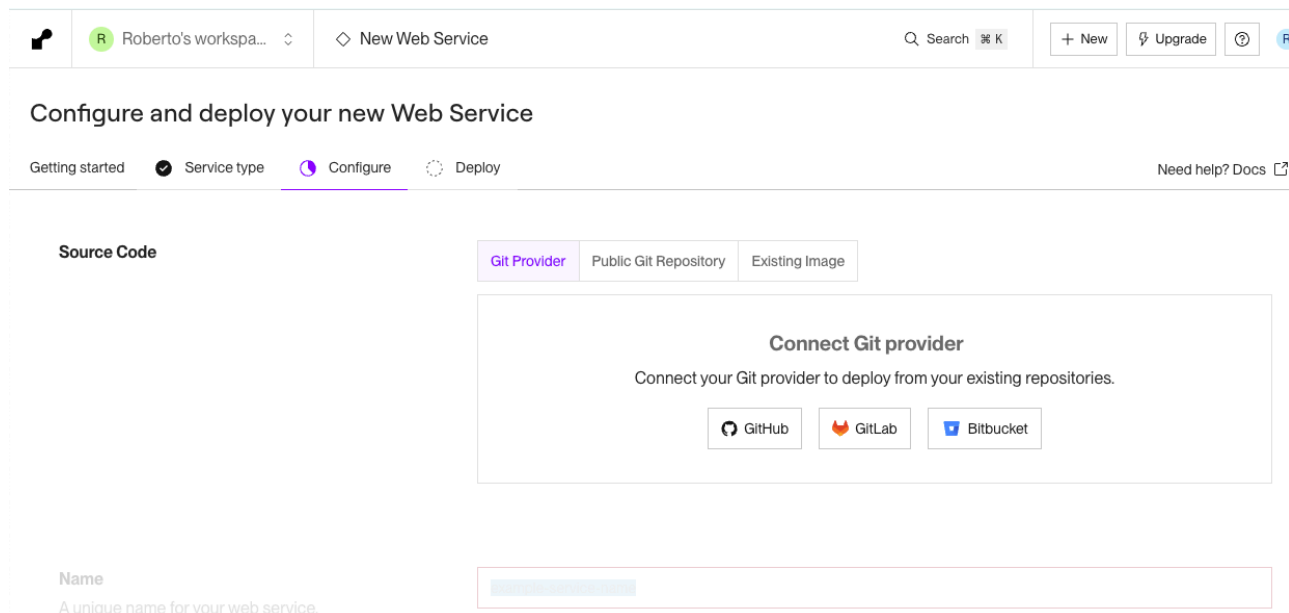
- `PGDATABASE` : database
- `PGUSER` : user
- `PGPASSWORD` : password
- `PGHOST` : host
- `PGPORT` : port

Creación de proyecto en Render

Tras crear una cuenta en Render, desde el dashboard podemos crear un nuevo proyecto, debemos crear uno de tipo "Web Services".



Conectar el proyecto con GitHub.



En github puedes entregar acceso especifico al repositorio que render necesita.

Permissions

✓ **Read** access to Dependabot alerts, administration, code, and metadata

✓ **Read and write** access to actions, checks, commit statuses, deployments, environments, issues, pull requests, repository hooks, and workflows


Repository access

☐ **All repositories**

This applies to all current and future repositories owned by the resource owner. Also includes public repositories (read-only).

☒ **Only select repositories**

Select at least one repository. Also includes public repositories (read-only).

 **Select repositories** ▾

Selected 1 repository.

 rarc/django-drf-cicd-example

×

Save

Cancel

Ahora debemos configurar el entorno de render.

Branch

The Git branch to build and deploy.

main

Region

Your services in the same [region](#) can communicate over a [private network](#).

Virginia (US East)

Root Directory Optional

If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a [monorepo](#).

e.g. `src`

Build Command

Render runs this command to build your app before each deploy.

```
$ pip install -r requirements.txt
```

Start Command

Render runs this command to start your app with each deploy.

```
$ gunicorn myproject.wsgi:application --bind 0.0.0.0:8000
```




Debemos asegurarnos de elegir lo siguiente:

- Lenguaje: Python
- Branch: main
- Región: US East
- Build Command: `./build.sh`
- Start Command:
`python -m gunicorn myproject.asgi:application -k uvicorn.workers.UvicornWorker`
- Instance Type: Free

Luego en variables de entorno debemos agregar las siguientes variables:

Environment Variables

Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more.](#)

PGDATABASE	
PGUSER	
PGPASSWORD	
PGHOST	
PGPORT	
SECRET_KEY	

- `SECRET_KEY` : clave secreta para el proyecto, podemos generar una con `python -c 'import secrets; print(secrets.token_hex(32))'`
- `PGDATABASE` valor desde neon
- `PGUSER` valor desde neon
- `PGPASSWORD` valor desde neon
- `PGHOST` valor desde neon
- `PGPORT` valor desde neon
- `PRODUCTION_HOST` lo dejaremos como example.com de momento ya que necesitamos que render nos asigne el dominio.

Una vez configurado todo, podemos hacer click en el botón "Deploy" para iniciar el despliegue.

Cuando el despliegue termine, podemos ver el dominio asignado en el dashboard de Render. Debemos editar nuevamente el entorno en Render, para esto vamos a Manage > Environment, y desde aquí damos en "Edit" en la sección "Environment Variables" y completamos el valor de `PRODUCTION_HOST` con el dominio asignado.