

Clase 3: Despliegue Continuo

Módulo 8: Fundamentos de Integración Continua

Apoyado por:

CORFO

Clase de hoy

01

CD Conceptos Clave

Despliegue Continuo.
Operaciones

02

Deploy to Render



2A

CD Conceptos Clave

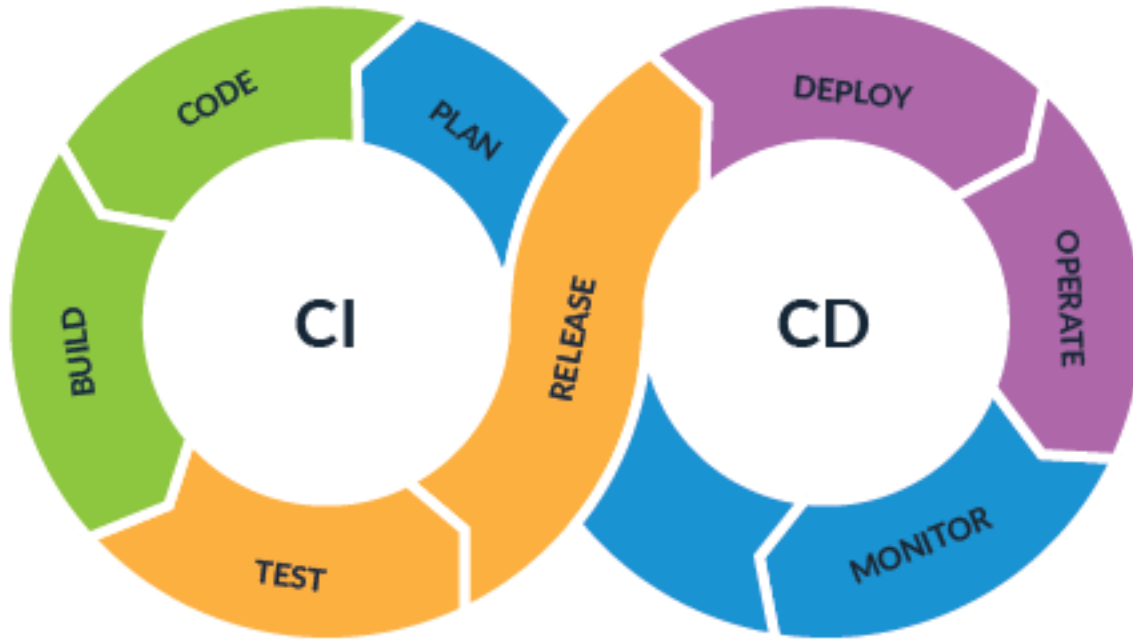
Bloque A

Qué veremos en bloque A

Conceptos Clave

- Definición y diferencias entre Despliegue Continuo y Entrega Continua.
- Compilación y despliegue
- Operaciones
- Ejemplo práctico

Entrega y Despliegue Continuo



Entrega y Despliegue Continuo [2]

- **Entrega Continua (Continuous Delivery):**
 - Amplía la Integración Continua.
 - Permite desplegar en cualquier momento, de forma confiable.
 - Automatiza todo el proceso hasta antes del despliegue (que es manual).
 - Asegura que cada cambio pase por pruebas y validaciones robustas.
- **Despliegue Continuo (Continuous Deployment):**
 - Automatiza también el despliegue a producción.
 - Cada cambio aprobado se publica inmediatamente a los usuarios.
 - Ideal cuando se tiene alta confianza en los tests automáticos.

Compilación y despliegue

- En un entorno DevOps, la compilación y despliegue de aplicaciones son parte del proceso automatizado y continuo que permite la entrega rápida y confiable de software.
- La **compilación** del código es el primer paso crítico después de que los desarrolladores han integrado sus cambios en el repositorio. Este proceso convierte el código fuente en binarios ejecutables o en otros artefactos utilizables.
- El CI/CD se encarga de ejecutar la herramienta de compilación como Maven/Gradle en el caso de Java o npm en el caso de Javascript. Los artefactos generados deben ser almacenados para ser reutilizados en pasos posteriores.

Compilación y despliegue [3]

- El **despliegue** es el proceso de tomar el código compilado y los artefactos asociados y ponerlos en un entorno de ejecución:
 - servidor de pruebas
 - un entorno de staging
 - Producción
- Para esto el CI/CD puede utilizar herramientas como docker y kubernetes para el despliegue de containers u otro pipelines específicos de acuerdo a la infraestructura y lenguajes utilizados.

Operaciones

- La **Infraestructura** y las **Operaciones** juegan un papel crucial para asegurar que el ciclo de desarrollo y entrega de software sea ágil, eficiente y confiable.
- Con las prácticas de CI/CD la infraestructura subyacente debe ser igualmente dinámica y adaptable.
- Tradicionalmente, las operaciones de infraestructura han sido vistas como un proceso separado y lento, caracterizado por la configuración manual de servidores, redes y almacenamiento.
- Con DevOps sin embargo se promueve la colaboración estrecha entre desarrolladores y equipos de operaciones, fomentando una cultura donde la infraestructura es tratada con el mismo rigor que el código de aplicación.

Operaciones [2]

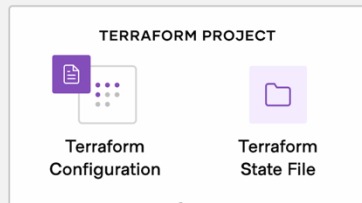
- El concepto de **Infraestructura como Código** (IaC) permite a los equipos definir y gestionar la infraestructura mediante archivos de configuración versionados, probados y reutilizados de manera similar al código de software.
- Esto mejora la velocidad y la eficiencia en la creación de entornos, a la vez que reduce los errores humanos y aumenta la consistencia y reproducibilidad de los entornos de desarrollo, pruebas y producción.



Operaciones [3]

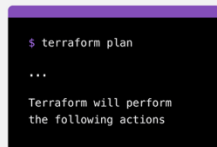
Write

Define infrastructure in configuration files



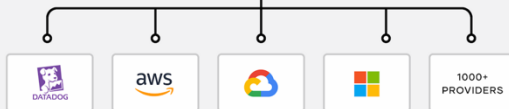
Plan

Review the changes
Terraform will make to
your infrastructure



Apply

Terraform provisions
your infrastructure and
updates the state file.



```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }

  required_version = ">= 1.2.0"
}

provider "aws" {
  region = "us-west-2"
}

resource "aws_instance" "app_server" {
  ami           = "ami-830c94e3"
  instance_type = "t2.micro"

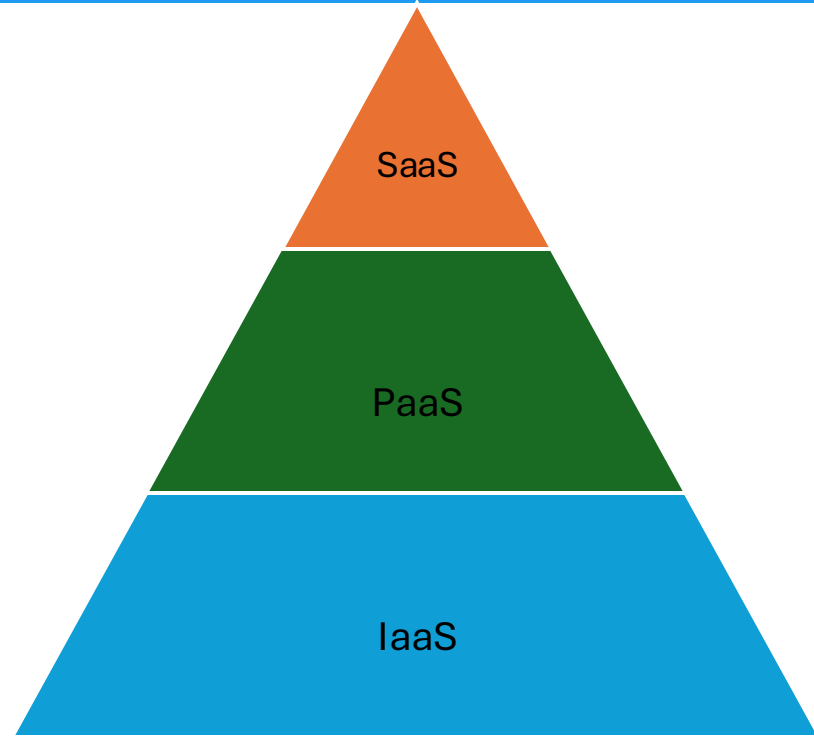
  tags = {
    Name = "ExampleAppServerInstance"
  }
}
```

Operaciones [4]

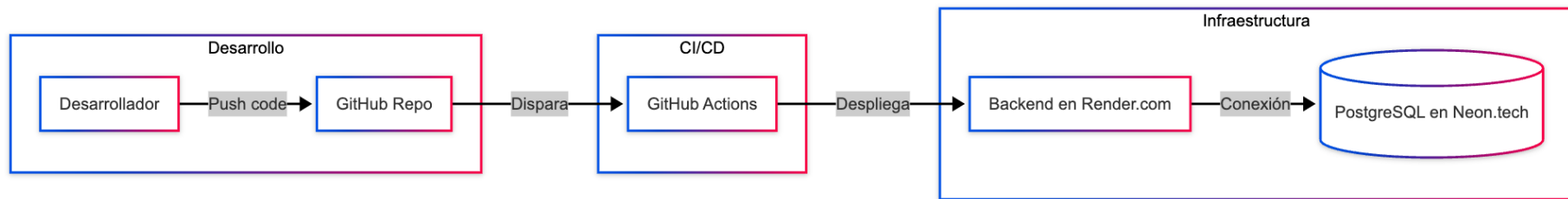
Característica	Local	Cloud
Control	Control total	Control compartido, según proveedor y modelo
Escalabilidad	Limitada, depende de disponibilidad	Alta, recursos escalables y bajo demanda
Costo Inicial	Altos, hardware y software	Bajos, pago por uso
Mantenimiento	Responsabilidad interna	Gestionado por proveedor
Seguridad	Gestionada internamente, mayor control.	Control compartido, menor control interno.

Operaciones [5]

- La nube ofrece varios **modelos de servicio** que permiten a las organizaciones elegir el nivel de control, flexibilidad y gestión que desean sobre su infraestructura tecnológica.
- Estos modelos incluyen:
 - Infraestructura como Servicio IaaS
 - Plataforma como Servicio PaaS
 - Software como Servicio SaaS




Ejemplo Practico / Arquitectura*



*draft


Ejemplo Practico / Arquitectura [2]


 **Render**

[Contact](#) [Sign In](#) [Get Started](#)






Your fastest path to production


Build, deploy, and scale your apps with unparalleled ease – from your first user to your billionth

[Get Started for Free](#) 

[Contact Sales](#) 

Trusted by over two million developers and teams

 Watershed  Bridge  BLACKBOX.AI  Felt = Equals 



Hobby

For personal projects and small-scale applications.

\$0 USD

per user/month plus compute costs*

[Start deploying](#)

Deploy full-stack apps in minutes



Fully-managed datastores



Custom domains



Global CDN & regional hosting

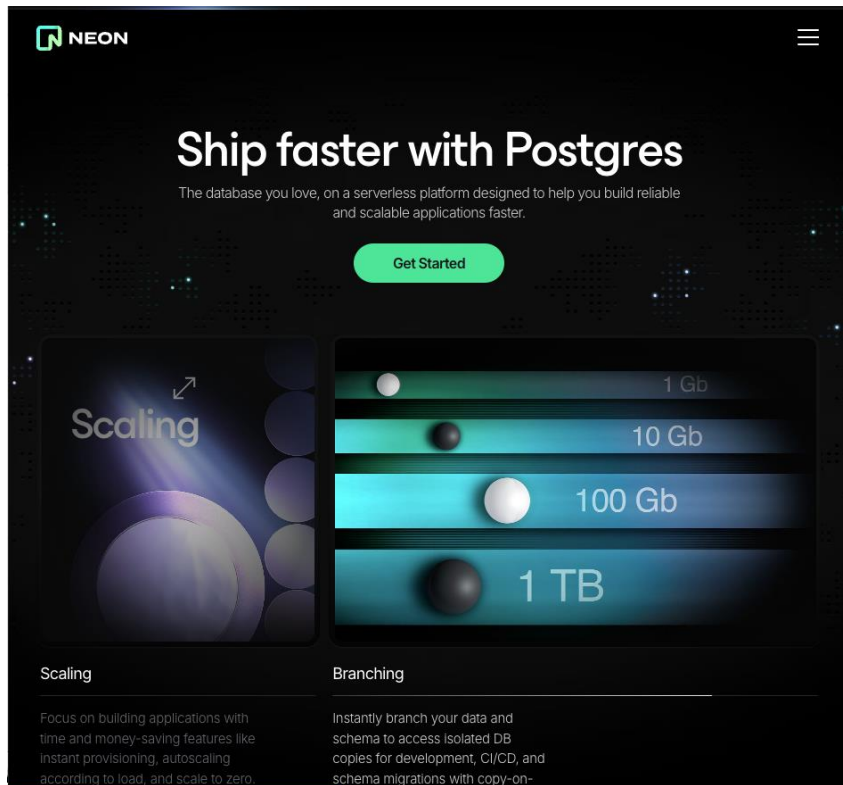


Get security out of the box



Email support

Ejemplo Practico / Arquitectura [3]



The image shows the Neon PostgreSQL landing page. It features a dark background with a grid of dots. The Neon logo is in the top left. The main headline is 'Ship faster with Postgres', followed by a sub-headline: 'The database you love, on a serverless platform designed to help you build reliable and scalable applications faster.' A green 'Get Started' button is centered below the text. On the left, there's a 'Scaling' section with a graphic of a sphere and an upward arrow. On the right, there's a 'Branching' section with a graphic of a sphere and a branching diagram. Below these, there are two columns: 'Scaling' and 'Branching'. The 'Scaling' column has a graphic of a sphere and an upward arrow. The 'Branching' column has a graphic of a sphere and a branching diagram. Below these, there are two columns: 'Scaling' and 'Branching'. The 'Scaling' column has a graphic of a sphere and an upward arrow. The 'Branching' column has a graphic of a sphere and a branching diagram.

NEON

Ship faster with Postgres

The database you love, on a serverless platform designed to help you build reliable and scalable applications faster.

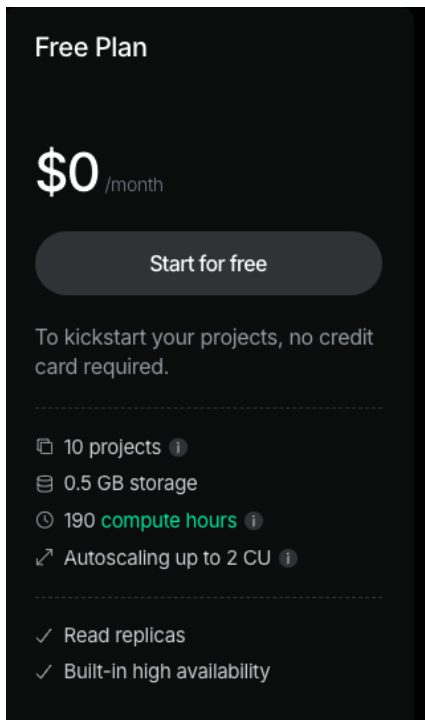
[Get Started](#)

Scaling

Focus on building applications with time and money-saving features like instant provisioning, autoscaling according to load, and scale to zero.

Branching

Instantly branch your data and schema to access isolated DB copies for development, CI/CD, and schema migrations with copy-on-



The image shows the 'Free Plan' details for Neon. It features a dark background with a grid of dots. The 'Free Plan' title is at the top. Below it, the price is '\$0 /month'. A dark grey 'Start for free' button is centered. Below the button, there's a paragraph: 'To kickstart your projects, no credit card required.' Below this, there's a list of features: '10 projects', '0.5 GB storage', '190 compute hours', and 'Autoscaling up to 2 CU'. At the bottom, there are two checkmarks: 'Read replicas' and 'Built-in high availability'.

Free Plan

\$0 /month

[Start for free](#)

To kickstart your projects, no credit card required.

- 10 projects
- 0.5 GB storage
- 190 **compute hours**
- Autoscaling up to 2 CU

- ✓ Read replicas
- ✓ Built-in high availability

Paso a Paso

- Revisemos la guía paso a paso



Trabajo grupal – Bloque A

Paralelo 2

G1	G2	G3	G4
Nicolas Mardones	Víctor Meza Herrera	Daniela Méndez Gándara	Claudia Blanco
Manuel Denis	Estefania Manriquez	Álvaro Pérez	Ariel Inostroza
Bryan Castillo	Patricio Vera	Pedro Nahum	Héctor Aguayo
GERALDY SUAREZ	Oscar Torres	Javier Gajardo	Ruben Sanhueza Ramirez
Scarlett Espinoza	Braulio Quiroz	Angela Proboste Neira	Félix González
Ulises Campodónico	Yerko Gallardo	Nicolás Guzmán	Ariel Mora
Carol Leiva	Rodrigo Araya		
G5	G6	G7	G8
Fabian Díaz	Camila Oyarzún	Mayerlyn Rodriguez	Daniela Porto
Natalia Rivera	Stefanya Pulgar	Sebastian Vega	Cristian Chavez Jara
Juan Salinas	Carlos Emilio Azócar Riquelme	Efrain Duarte Campos	Juan Rodrigo Vega
Rodrigo Pastén Cortés	Nicolas Rojas	Bianel Bianchini	Rodolfo Cantillana
Flavio Jara R.	luis.paillan.cnc@gmail.com	Bastián Gamboa Labbé	Abraham Ruiz
Daniel García	Cristóbal Gajardo	Pablo Uribe	Rodrigo Álvarez

Trabajo Grupal Bloque A

1. Configure aplicación DRF

1. Haga un fork del repositorio <https://github.com/rarce/django-drf-cicd-example>
2. Cree una cuenta en <https://render.com>
3. Cree una cuenta en <https://neon.tech/>
4. Baje primero el proyecto de manera local y pruebe que funcione:
 1. Cree el ambiente virtual e instale dependencias
 2. Conectese a su bd en neon.tech, para esto debe configurar todas las variables del .env (a partir del .env.example) **NOTA: esto es por simplicidad no debemos mezclar ambientes**
 3. Ejecute el servidor, como usamos whitenoise y desactivamos los assets debe primero generarlos (mire el archivo build.sh)



Break!



2B

Despliegue en Render

Bloque B

Qué veremos en Bloque B

- Desplegar la aplicación DRF en render.com



Trabajo grupal – Bloque B

Trabajo Grupal Bloque B

1. Desplegar aplicación DRF en render.com

1. Configure un nuevo Proyecto en render.com
2. Conecte este Proyecto con su repositorio en Github
3. Configure todas las variables en su deployment (mire la guía paso a paso)
4. ¡Revise su aplicación publicada!

Paralelo 2

G1	G2	G3	G4
Nicolas Mardones	Víctor Meza Herrera	Daniela Méndez Gándara	Claudia Blanco
Manuel Denis	Estefania Manriquez	Álvaro Pérez	Ariel Inostroza
Bryan Castillo	Patricio Vera	Pedro Nahum	Héctor Aguayo
GERALDY SUAREZ	Oscar Torres	Javier Gajardo	Ruben Sanhueza Ramirez
Scarlett Espinoza	Braulio Quiroz	Angela Proboste Neira	Félix González
Ulises Campodónico	Yerko Gallardo	Nicolás Guzmán	Ariel Mora
Carol Leiva	Rodrigo Araya		
G5	G6	G7	G8
Fabian Díaz	Camila Oyarzún	Mayerlyn Rodriguez	Daniela Porto
Natalia Rivera	Stefanya Pulgar	Sebastian Vega	Cristian Chavez Jara
Juan Salinas	Carlos Emilio Azócar Riquelme	Efrain Duarte Campos	Juan Rodrigo Vega
Rodrigo Pastén Cortés	Nicolas Rojas	Bianel Bianchini	Rodolfo Cantillana
Flavio Jara R.	luis.paillan.cnc@gmail.com	Bastián Gamboa Labbé	Abraham Ruiz
Daniel García	Cristóbal Gajardo	Pablo Uribe	Rodrigo Álvarez

Referencias

- **CI/CD**
 - <https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment>
- **What is continuos deployment?**
 - <https://www.ibm.com/think/topics/continuous-deployment>
- **Deploy Django on Render.com**
 - <https://render.com/docs/deploy-django>
- **Conectar Django Apps to Neon**
 - <https://neon.tech/docs/guides/django>

¿Preguntas?

¡Hemos llegado al final de la clase!

