

CI/CD: deploy automático con GitHub Actions

Hacés push, y tu sitio se actualiza solo. Eso es CI/CD.

QUÉ ES CI/CD

CONCEPTO CLAVE

CI = Continuous Integration — cada vez que pushés código, se construye y testea automáticamente.

CD = Continuous Deployment — si todo pasa, se despliega automáticamente a producción.

ANALOGÍA

Imaginá una cinta transportadora en una fábrica. Vos ponés el código al inicio de la cinta (git push), y al final sale tu app corriendo en la nube. Todo lo del medio es automático: construir, testear, desplegar. Esa cinta se llama **pipeline**.

SIN CI/CD

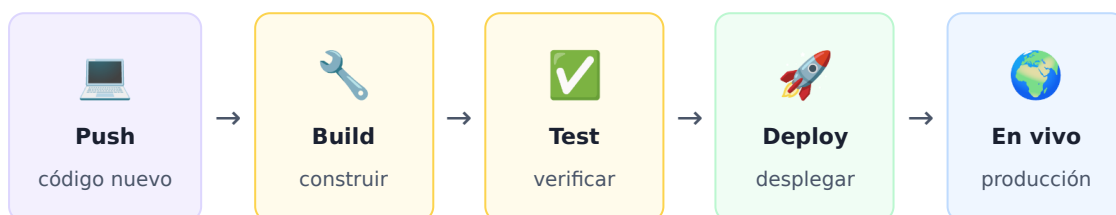
Cambiás el código, te conectás a la EC2 por SSH, copiás archivos a mano, reiniciás el servicio. Te olvidaste algo? Volvemos a empezar.

CON CI/CD

`git push` y listo. El pipeline se encarga de todo. Siempre igual, sin errores humanos, en cada push.

QUÉ ES UN PIPELINE

Un **pipeline** es una serie de pasos automáticos que se ejecutan en orden:



Si algún paso falla, el pipeline se detiene y te avisa. Así nunca llega código roto a producción.

GITHUB ACTIONS: TU PIPELINE GRATIS

GitHub Actions es la herramienta de CI/CD de GitHub. Creás un archivo `.yaml` en tu repo y GitHub ejecuta los pasos cada vez que pushés.

VOCABULARIO

- **Workflow** — el archivo `.yaml` que define tu pipeline
- **Trigger** — qué lo dispara (ejemplo: `on: push`)
- **Job** — un grupo de pasos que corren en una máquina virtual
- **Step** — un paso individual (ejemplo: "subir archivos a S3")
- **Secret** — datos sensibles (passwords, API keys) guardados en GitHub

El archivo va en `.github/workflows/deploy.yaml` — GitHub lo detecta automáticamente.

NUESTRO PIPELINE: PUSH → S3

Vamos a crear un pipeline que despliega el frontend a un **bucket S3** configurado como sitio web:

```
name: Deploy Frontend to S3

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      name: Checkout code
      uses: actions/checkout@v4

      name: Configure AWS credentials
      uses: aws-actions/configure-aws-credentials@v4
      with:
        aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
        aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
        aws-region: us-east-1

      name: Deploy to S3
      run: aws s3 sync ./frontend/ s3://${ secrets.S3_BUCKET_NAME } --delete
```

Son **3 pasos**:

- **Checkout** — baja el código del repo
- **Configure AWS** — se autentica con tus credenciales (guardadas como Secrets)
- **Deploy to S3** — sube los archivos del frontend al bucket

SECRETS: CREDENCIALES SEGURAS

Las credenciales de AWS **nunca van en el código**. Se guardan como **Secrets** en GitHub:

LOS 3 SECRETS QUE NECESITAMOS

- `AWS_ACCESS_KEY_ID` — tu access key de AWS
- `AWS_SECRET_ACCESS_KEY` — tu secret key de AWS
- `S3_BUCKET_NAME` — el nombre del bucket (lo creás con Terraform)

Se configuran en: **Repo** → **Settings** → **Secrets and variables** → **Actions** → **New repository secret**

SEGURIDAD

Los Secrets están encriptados. GitHub no los muestra en los logs y solo los puede ver el dueño del repo. Es la forma correcta de manejar credenciales.

EL BUCKET: CREADO CON TERRAFORM

El bucket S3 con **Static Website Hosting** lo creás con Terraform:

```
resource "aws_s3_bucket" "frontend" {
  bucket = "sesion4-tunombre-frontend"
}

# Activar hosting de sitio web
resource "aws_s3_bucket_website_configuration" "frontend" {
  bucket = aws_s3_bucket.frontend.id
  index_document  suffix "index.html"
}

# Permitir acceso público
resource "aws_s3_bucket_policy" "frontend" {
```

Después del `terraform apply`, tenés tu URL:

```
website_url = "http://sesion4-tunombre-frontend.s3-website-us-east-1.amazonaws.com"
```

EL FLUJO COMPLETO



Una vez configurado, cada `git push` actualiza tu sitio automáticamente. Cambiás el HTML, pushés, y en 30 segundos está en vivo.

LA EVOLUCIÓN COMPLETA DEL CURSO



Sesión 2: hiciste todo a mano. Sesión 3: empaquetaste en contenedores. Sesión 4: automatizaste la infra con Terraform y el deploy con CI/CD. El ciclo completo: **código → infraestructura → deploy automático**.

EN LA VIDA REAL

Empresas como Netflix, Spotify y MercadoLibre usan pipelines de CI/CD para desplegar cientos de veces al día. Lo que acabamos de hacer con GitHub Actions + S3 es la versión mínima de lo que ellos usan a gran escala.