# MLOps (Machine Learning Operations)

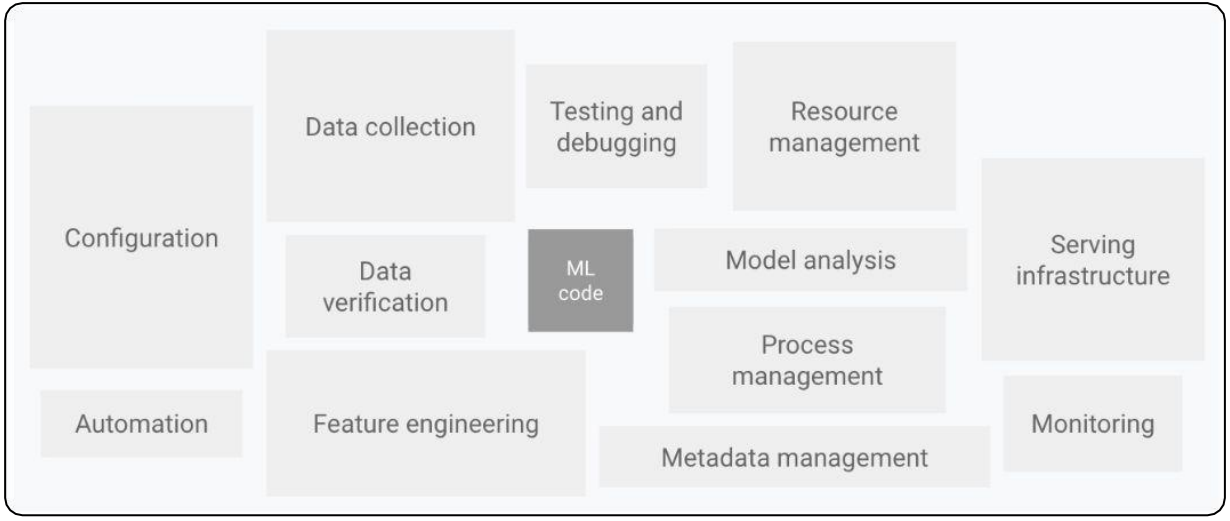
**Contexto:** Vivimos en una era de **datos masivos**, **procesamiento asequible** y **avances rápidos en ML**, lo que impulsa a las empresas a desarrollar modelos predictivos para generar valor comercial.

**¿Qué es MLOps?**
Un enfoque que **integra principios de DevOps en ML**, buscando la automatización y supervisión en todas las fases del ciclo de vida de ML, desde el desarrollo hasta la operación.

**Desafíos:** Más allá de desarrollar un modelo con buen rendimiento, el reto está en **crear y mantener sistemas de ML** que **operen eficientemente en producción.**

Un sistema de ML real **abarca mucho más que solo el código ML.** Los componentes críticos que lo rodeanincluyen configuración, automatización, gestión de datos, pruebas, y mucho más.

Para manejar la complejidad de estos sistemas, se aplican principios de DevOps adaptados al ML, conocidos como MLOps.
Esto incluye prácticas esenciales como: **CI (Integración Continua)** **CD (Despliegue Continuo)** **CT (Entrenamiento Continuo)**.



# MLOps (Machine Learning Operations)

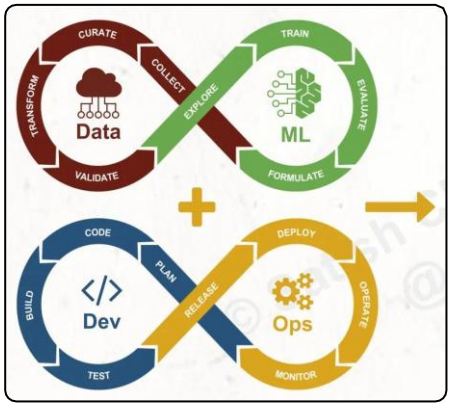
**DevOps en Software Tradicional:** Enfatiza la integración continua (CI) y la entrega continua (CD) para acelerar el desarrollo, mejorar la implementación, y garantizar actualizaciones confiables.

**Diferencias Clave:**

**Integración Continua (CI):** En ML, implica la validación de código, datos, esquemas, y modelos.

**Entrega Continua (CD):** En ML, abarca el despliegue automatizado de sistemas completos de entrenamiento y predicción.

**Entrenamiento Continuo (CT):** Aspecto único de MLOps que se enfoca en el reentrenamiento y despliegue automáticos de modelos.




**MLOps en Sistemas de ML**

Aplica prácticas de DevOps adaptadas a las peculiaridades de los sistemas de ML, incluyendo:

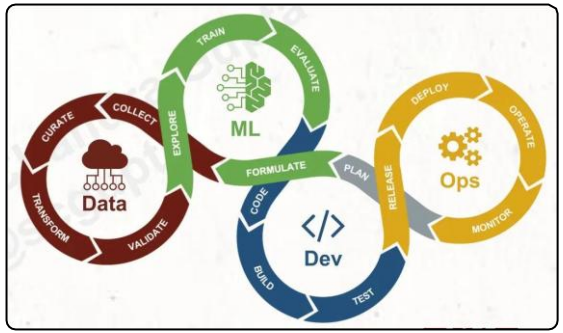
**Equipos Multidisciplinarios:** Integración de científicos de datos y expertos en ML que pueden no tener experiencia en ingeniería de software a nivel de producción.

**Desarrollo Experimental:** Proceso iterativo y experimental para explorar diferentes funciones, algoritmos, y técnicas de modelado, enfatizando la importancia de la reproducibilidad y la reutilización del código.

**Pruebas Complejas:** Validación de datos y modelos más allá de las pruebas de integración y unidad típicas en software convencional.

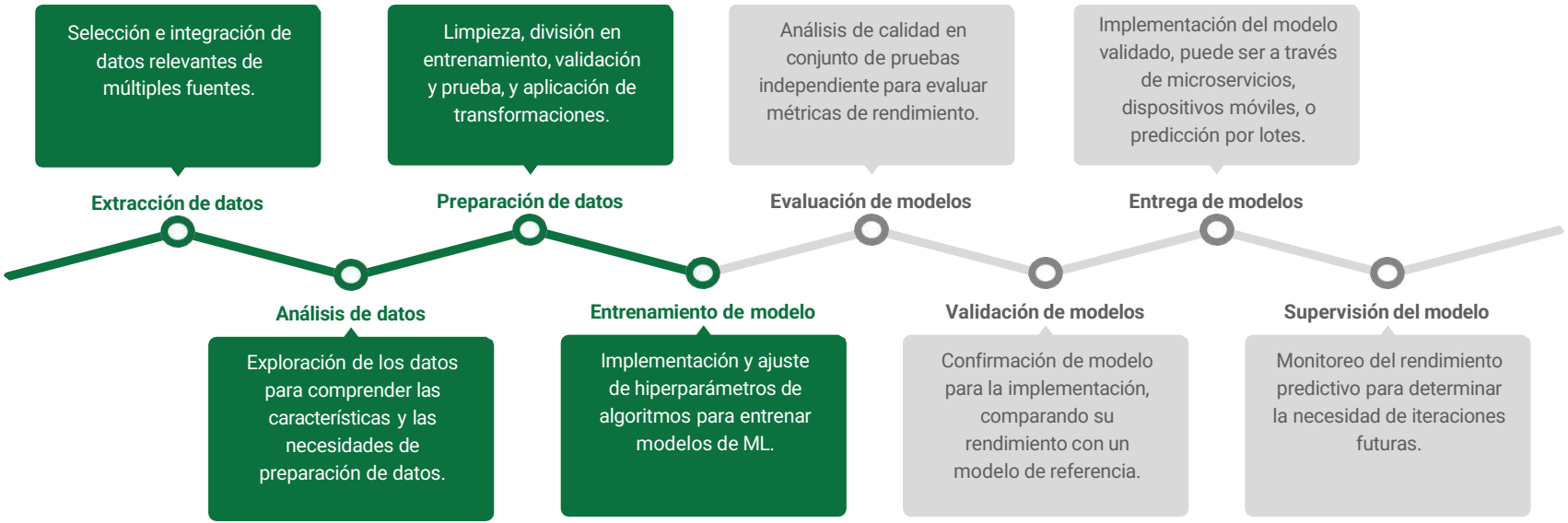
**Implementación Desafiante:** Necesidad de automatizar completamente las canalizaciones de entrenamiento y despliegue de modelos, aumentando la complejidad.

**Mantenimiento en Producción:** Supervisión continua del rendimiento del modelo y la calidad de los datos para manejar el decaimiento del modelo.



# MLOps (Machine Learning Operations)

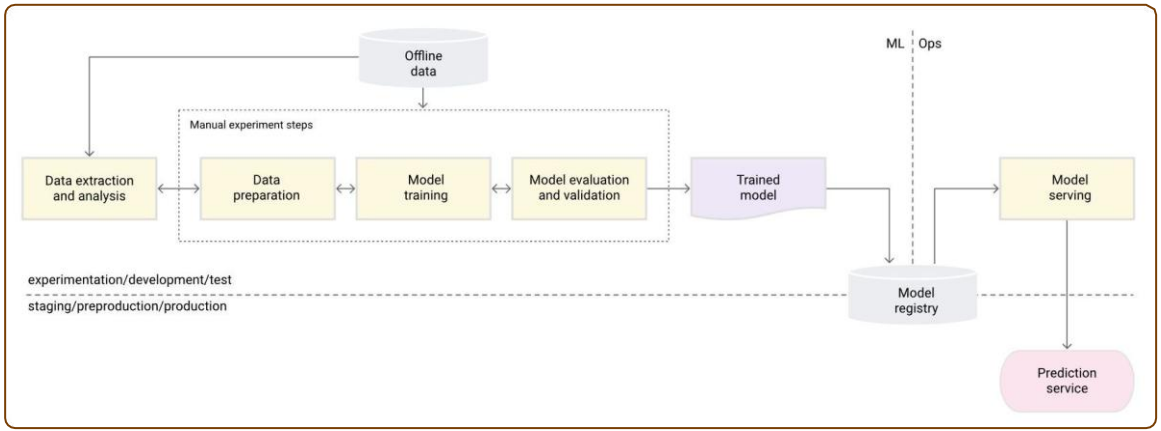
En cualquier proyecto de ML, **después de definir el caso de uso empresarial y establecer los criterios de éxito**, el proceso de entrega de un modelo de ML a la producción implica los siguientes pasos.



# MLOps (Machine Learning Operations)

## MLOps Nivel 0 - Proceso Manual

MLOps nivel 0 se caracteriza por un **proceso completamente manual** para desarrollar e implementar modelos de machine learning, representando el **nivel de madurez básico.**




**Foco en la Implementación de Servicio de Predicción:** El énfasis está en desplegar modelos como servicios de predicción, sin establecer prácticas de CI/CD.

**Falta de Supervisión Activa:** La ausencia de monitoreo continuo del modelo en producción puede ocasionar que la degradación del rendimiento pase desapercibida.

## Características

**Proceso Manual Interactivo:** Las tareas de análisis y validación se realizan a mano usando código en notebooks, lo que implica un proceso interactivo y basado en scripts.

**Desconexión entre Desarrollo y Operaciones:** Existe una separación clara entre los equipos que crean modelos y los que los implementan, con una transferencia manual de los modelos para su despliegue.

**Iteraciones de Versiones Poco Frecuentes:** Los modelos se actualizan o reentrenan raramente, resultando en la implementación de nuevas versiones solo ocasionalmente.

**Ausencia de CI/CD:** No se practica la integración ni la entrega continua, principalmente actualizpaocriones y camlbaios. baja frecuencia de

# MLOps (Machine Learning Operations)

## MLOps Nivel 0 - Proceso Manual

El nivel 0, común en **empresas que inician con ML**, presenta retos significativos, como la falta de adaptación de los modelos a los cambios en el entorno y los datos, lo que a menudo lleva a un rendimiento deficiente en producción.

### Pasos para Mantener la Exactitud del Modelo

**Supervisión Activa:** Monitorear constantemente la calidad y el rendimiento del modelo en producción para identificar y corregir la degradación y otros problemas.

**Reentrenamiento Frecuente:** Actualizar los modelos con los datos más recientes para reflejar patrones emergentes y cambios en las tendencias.

**Experimentación Continua:** Probar continuamente nuevas implementaciones, técnicas de ingeniería de atributos, arquitecturas de modelos y ajustes de hiperparámetros para mejorar la exactitud.

### MLOps para Mejora Continua

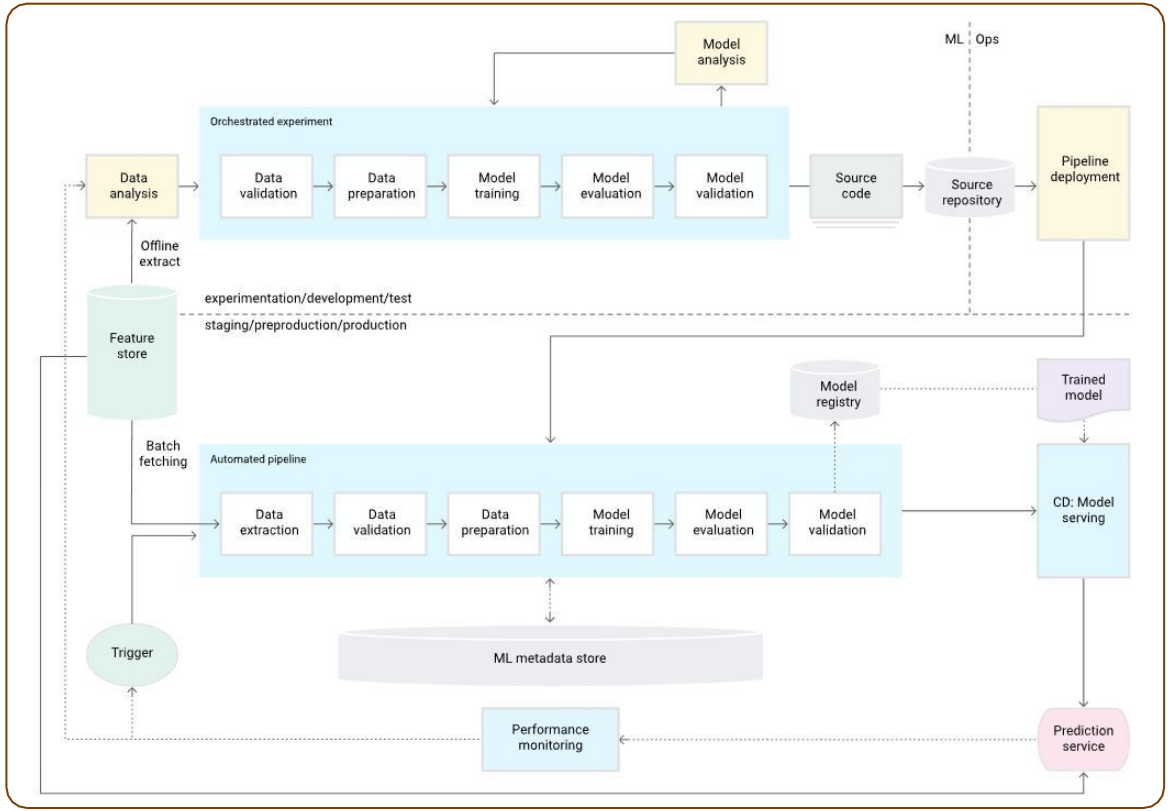
**Implementación de CI/CD y CT:** Adoptar prácticas de MLOps para establecer procesos automáticos de integración continua (CI), entrega continua (CD) y entrenamiento continuo (CT), permitiendo la rápida prueba, compilación, y despliegue de modelos.

**Canalización de Entrenamiento Automatizada:** Crear una canalización automatizada de entrenamiento de ML para facilitar el reentrenamiento y la implementación de modelos con eficiencia y precisión.

# MLOps (Machine Learning Operations)

**Nivel 1 de MLOps: Automatización de la canalización de ML**

El objetivo del nivel 1 es realizar un **entrenamiento continuo** del modelo mediante la automatización de la canalización de ML, lo que te permite lograr una **entrega continua** del servicio de predicción del modelo.
Para automatizar el proceso de uso de datos nuevos a fin de volver a entrenar los modelos en la producción, debes **ingresar los datos automatizados** y los pasos de **validación de modelos** en la canalización, y los **activadores de canalización** y la **administración de metadatos**.



# MLOps (Machine Learning Operations)

**Nivel 1 de MLOps: Automatización de la canalización de ML**

## Características

**Experimentación Rápida:** Los pasos experimentales son organizados y automatizados, facilitando iteraciones rápidas y una transición eficiente hacia la producción.

**Entrenamiento Continuo en Producción (CT):** Automatización del reentrenamiento de modelos con datos recientes en producción, utilizando activadores de canalización.

**Simetría Experimental-Operacional:** Unificación de las canalizaciones entre desarrollo y producción, clave para la práctica de MLOps y para asegurar consistencia y eficiencia operativa.

**Implementación de Canalización Completa:** Avance del Nivel 0, implementando no solo el modelo sino toda la canalización de entrenamiento de manera automática y recurrente para mantener los servicios de predicción actualizados.

**Entrega Continua de Modelos:** Implementación automatizada de modelos como servicios de predicción, utilizando datos actualizados, facilitando la entrega continua de servicios de predicción.

**Código Modular para Componentes y Canalizaciones:** Modularización del código fuente para reutilización y acoplamiento fácil en canalizaciones, con componentes organizados en contenedores.

# MLOps (Machine Learning Operations)

**Nivel 1 de MLOps: Automatización de la canalización de ML**

**Componentes Clave** para el **Entrenamiento Continuo** en MLOps Nivel 1

Procesos automáticos esenciales para asegurar la calidad y relevancia de los modelos entrenados con datos nuevos.
**Validación de Datos:** Detecta sesgos y anomalías en los datos antes del entrenamiento, evaluando si los datos cumplen con el esquema esperado y si reflejan cambios significativos que requieren reentrenamiento.
**Validación de Modelos:** Evalúa la calidad predictiva y la coherencia del rendimiento del modelo en diversos segmentos de datos antes de su despliegue en producción.

**Feature Store:** Repositorio centralizado para gestionar y reutilizar características de datos, facilitando la consistencia entre entrenamiento y predicción, y apoyando tanto el entrenamiento continuo como las predicciones en tiempo real.

**Administración de Metadatos:** Registro detallado de la ejecución de canalizaciones para facilitar el linaje, la reproducibilidad, y la depuración de modelos. Incluye información sobre versiones de canalización, argumentos de parámetros, artefactos producidos, y métricas de evaluación.

**Activadores de Canalización:** Mecanismos para iniciar el reentrenamiento automático de modelos basado en diversas condiciones, como:

- A Demanda: Ejecución manual según sea necesario.
- Programación Regular: Basada en la disponibilidad sistemática de datos nuevos.
- Disponibilidad de Datos Nuevos: Reentrenamiento desencadenado por la llegada de nuevos datos.
- Deterioro del Rendimiento del Modelo: Iniciado por una caída notable en el rendimiento del modelo.
- Desvío de Conceptos: Cambios significativos en las distribuciones de datos que indican la necesidad de reentrenamiento.

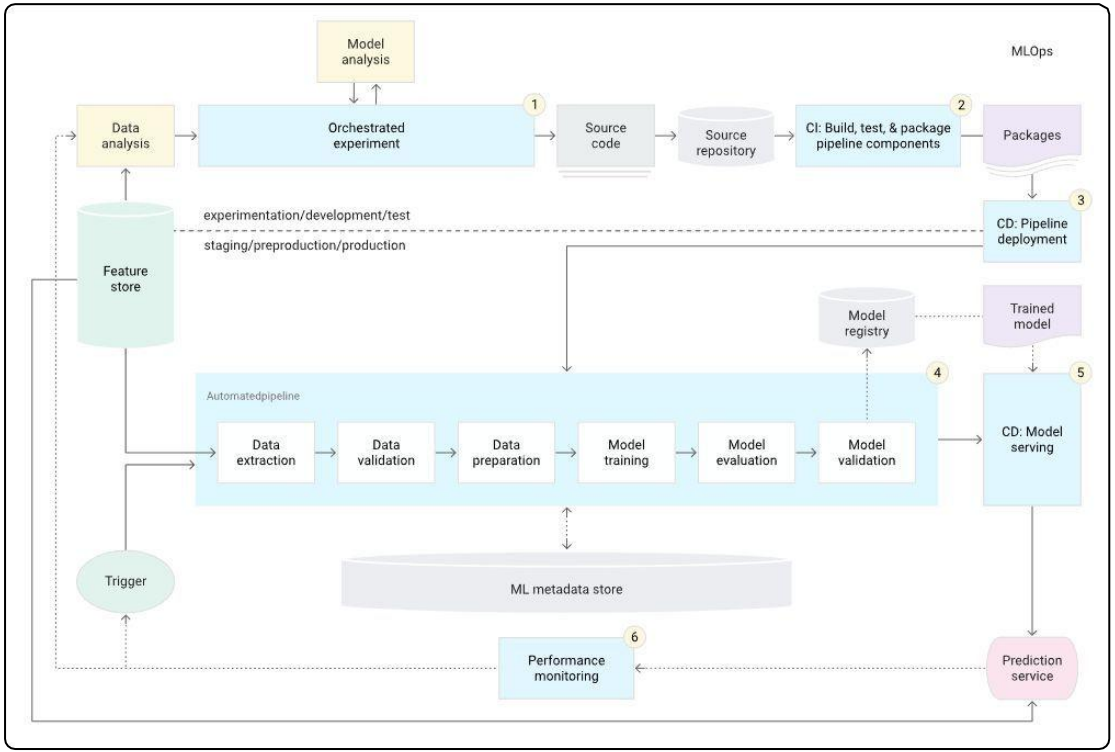
# MLOps (Machine Learning Operations)

**Nivel 2 de MLOps: Automatización de la canalización de CI/CD**

Un sistema de **CI/CD robusto y automatizado** es crucial para la actualización rápida y fiable de canalizaciones de ML en producción.

Facilita a los científicos de datos la **experimentación y aplicación de nuevas ideas** en **ingeniería de atributos**, **arquitecturas de modelos** y **ajustes de hiperparámetros**.

Este enfoque permite la **compilación, evaluación y despliegue automáticos** de nuevos componentes de la canalización en el entorno de producción, asegurando la eficiencia y agilidad en el desarrollo y operación de modelos de ML.



# MLOps (Machine Learning Operations)

**Nivel 2 de MLOps: Automatización de la canalización de CI/CD**

## Características

**Desarrollo y Experimentación:** Inicio con la prueba iterativa de algoritmos y modelos nuevos, cuyos resultados se organizan y envían a un repositorio de código.

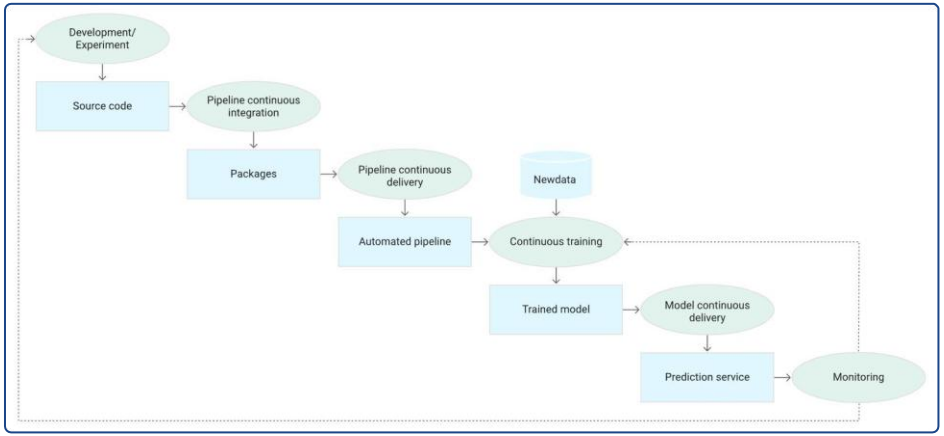
**Integración Continua (CI) de la Canalización:** Compilación del código fuente y ejecución de pruebas para generar componentes de canalización, que incluyen paquetes, ejecutables y artefactos.

**Entrega Continua (CD) de la Canalización:** Despliegue de los artefactos de CI en el entorno de destino, resultando en la implementación de una canalización actualizada con el modelo nuevo.

**Activación Automática:** Ejecución automática de la canalización en producción, ya sea por programación o activadores específicos, produciendo un modelo entrenado para el registro de modelos.

**Entrega Continua de Modelos:** Implementación del modelo entrenado como un servicio de predicción, finalizando con el despliegue del servicio de predicción del modelo.

**Supervisión:** Monitoreo del rendimiento del modelo en producción para identificar la necesidad de reentrenamientos o ajustes basados en el análisis de datos activos.



# MLOps (Machine Learning Operations)

**Nivel 2 de MLOps: Automatización de la canalización de CI/CD**

## Integración Continua (CI)

**Compilación y Empaquetado:** Al confirmarse nuevo código, la canalización y sus componentes se compilan, evalúan y empaquetan automáticamente, incluyendo la creación de paquetes, imágenes de contenedores y ejecutables.

### Pruebas de CI Incluyen:

**Pruebas de Unidad para Feature Engineering:** Verificar la lógica detrás de la creación y transformación de características.
**Pruebas de Unidad para Métodos del Modelo:** Evaluar funciones específicas del modelo, como la correcta implementación de codificación one-hot para datos categóricos.
**Convergencia del Entrenamiento de Modelos:** Confirmar que la pérdida del modelo disminuye adecuadamente a lo largo de las iteraciones sin sobreajustarse.
**Validación de Valores del Modelo:** Asegurar que el entrenamiento del modelo no resulte en valores NaN por errores como división por cero.
**Producción de Artefactos Esperados:** Comprobar que cada componente de la canalización genera los artefactos previstos.
**Integración de Componentes de la Canalización:** Evaluar la correcta interacción y funcionamiento conjunto de los distintos componentes de la canalización.

# MLOps (Machine Learning Operations)

**Nivel 2 de MLOps: Automatización de la canalización de CI/CD**

**Entrega Contínua (CD)**

**Automatizar el despliegue** de nuevas canalizaciones en el entorno de destino, proporcionando **servicios de predicción con modelos recién entrenados**.

## Consideraciones Clave para la Entrega Continua

**Compatibilidad del Modelo con Infraestructura:** Asegurar que los requisitos del modelo, como paquetes, memoria y recursos de procesamiento, estén disponibles en el entorno de destino.

**Validación de la API de Servicio:** Probar la API con datos de entrada esperados para verificar las respuestas correctas del servicio de predicción.

**Evaluación del Rendimiento del Servicio de Predicción:** Realizar pruebas de carga para medir métricas críticas, incluyendo consultas por segundo (QPS) y latencia.

**Validación de Datos para Reentrenamiento o Predicción por Lotes:** Verificar la adecuación y calidad de los datos utilizados.

**Cumplimiento de Objetivos de Rendimiento Predictivo:** Asegurar que los modelos satisfacen los estándares de rendimiento antes de su implementación.

## Estrategias de Implementación

**Automatizada en Entorno de Prueba:** Despliegue activado automáticamente con el envío de código a la rama de desarrollo.

**Semiatutomatizada en Entorno de Preproducción:** Despliegue activado tras la fusión del código en la rama principal y aprobación de los revisores.

**Manual en Entorno de Producción:** Implementación cuidadosa en producción tras pruebas exitosas en entornos anteriores.

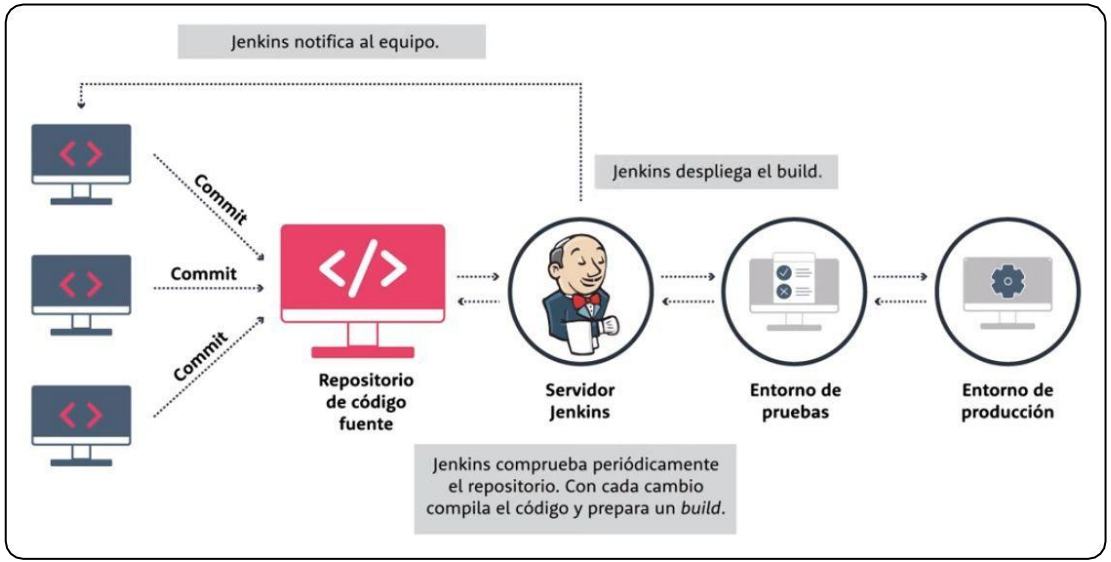
# CI/CD con Jenkins para el Desarrollo Local

**Jenkins** es un **servidor de automatización de código abierto** que se usa ampliamente para **automatizar partes del desarrollo de software** relacionadas con la construcción, prueba e implementación, facilitando la integración y entrega continuas (CI/CD).

En el contexto de **MLOps**, que se centra en **automatizar y mejorar el ciclo de vida de los modelos de aprendizaje automático**, Jenkins permite lo siguiente:

Jenkins automatiza el **proceso de integración continua (CI) para modelos de aprendizaje automático (ML)** mediante la gestión de cambios de código de múltiples contribuyentes en un único proyecto de software.

Jenkins automatiza la **entrega continua (CD) de modelos de ML**, garantizando su implementación confiable en diferentes entornos (desarrollo, puesta en escena, producción) con las configuraciones adecuadas.



# CI/CD con Jenkins para el Desarrollo Local

Instalar Jenkins en máquina local

Utilizar la versión de LTS (Long-Term-Support) de Jenkins como la imagen base

Preparar un entorno de Jenkins que pueda trabajar eficientemente con Docker

Instalar AWS CLI para interactuar con los Servicios Web de Amazon

Instalar el instalador de paquetes de Python3, pip

Instalar GnuPG para la gestión de claves y operaciones de seguridad

Añadir el usuario de jenkins al grupo docker

Dockerfile

```
FROM jenkins/jenkins:lts
USER root

RUN /usr/local/bin/install-plugins.sh
   docker-workflow:1.26

RUN apt-get update -qq \
    && apt-get install -qqy apt-transport-https ca-certificates curl
gnupg2 software-properties-common
RUN curl -fsSL https://download.docker.com/linux/debian/gpg|
apt-key add -

RUN add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/debia\n
   $(lsb_release -cs) \
   stable"

RUN apt-get update -qq \
    && apt-get install -y docker-ce docker-ce-cli containerd.io

RUN apt install awscli -y
RUN apt-get install python3-pip -y
RUN apt-get install -y gnupg

RUN usermod -aG docker jenkins
```

plugins.txt U ● Dock

Jenkins_CI-CD > Installation >

```
1  docker-workflow
2  git:latest
```

# CI/CD con Jenkins para el Desarrollo Local

## Instalar Jenkins en máquina local

Definir la **versión de la sintaxis de Docker Compose** que se está utilizando.

Dentro de **services**, se define un servicio llamado jenkins. En Docker Compose, un servicio es básicamente un contenedor en ejecución que forma parte de la aplicación.

**image:** jenkins-img indica que el servicio debe usar una imagen llamada jenkins-img.
**container_name:** jenkins-container asigna un nombre específico al contenedor que se va a crear a partir de este servicio.
**build:** ./ instruye a Docker Compose a construir la imagen del servicio usando el Dockerfile ubicado en el directorio actual (./)

docker-compose.yml

```
version: "3.0"
services:

  jenkins:
    image: jenkins-img
    container_name: jenkins-container
    build: ./

    volumes:
     - /var/run/docker.sock:/var/run/docker.sock
     - ./jenkins_home:/var/jenkins_home
       ports:

     - "8081:8080"
```

Los **volúmenes** permiten compartir archivos entre el contenedor y el sistema anfitrión o entre contenedores. En este caso, se montan dos volúmenes:
**/var/run/docker.sock:/var/run/docker.sock** monta el socket de Docker del anfitrión dentro del contenedor. Esto permite que el contenedor de Jenkins controle el **daemon de Docker** en el anfitrión, lo cual es necesario para que Jenkins pueda crear y gestionar contenedores Docker.
**./jenkins_home:/var/jenkins_home** monta un directorio llamado jenkins_home en el directorio actual del sistema anfitrión al directorio /var/jenkins_home dentro del contenedor. Esto es utilizado para **almacenar los datos de Jenkins** (como configuraciones, trabajos, y plugins) de manera persistente, permitiendo que los datos sobrevivan reinicios del contenedor.

# CI/CD con Jenkins para el Desarrollo Local

Instalar Jenkins en máquina local

**"8081:8080"** indica que el puerto 8080 del contenedor (donde Jenkins escucha por defecto) se mapea al puerto 8081 del sistema anfitrión. Esto permite acceder a la interfaz web de Jenkins navegando a http://localhost:8081 en el navegador del sistema anfitrión.

El comando **docker-compose up** se utiliza para iniciar y ejecutar toda la aplicación definida por un archivo docker-compose.yml.

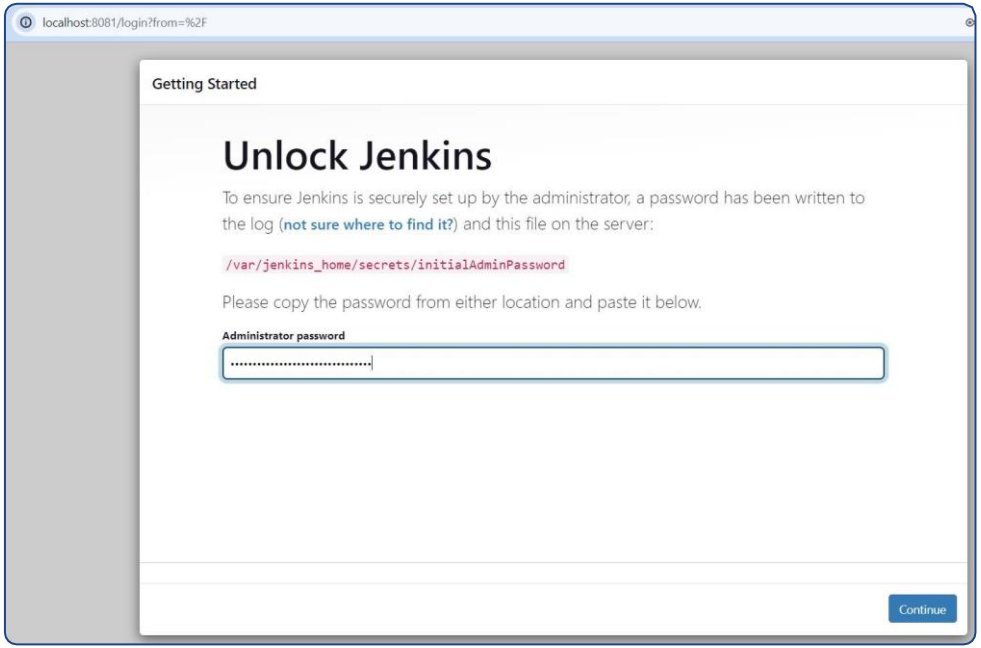
docker-compose.yml

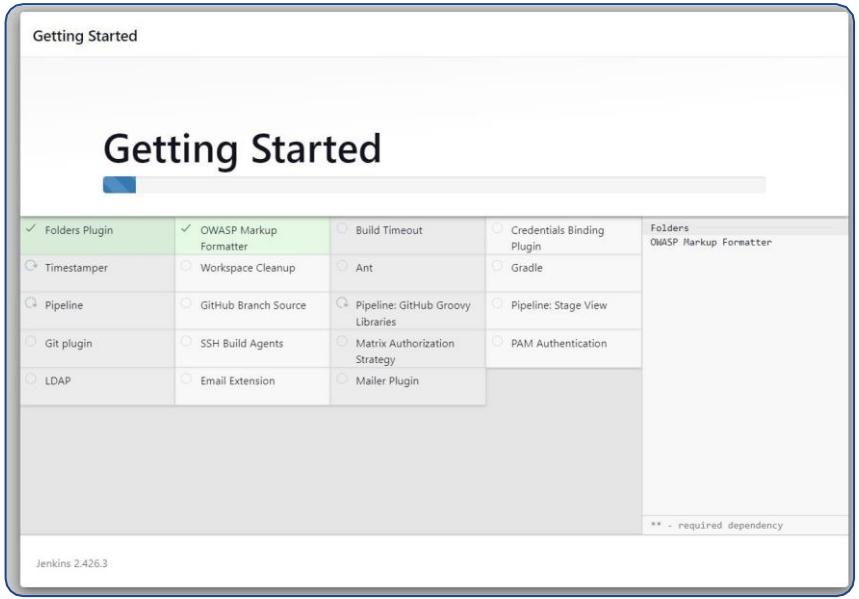
```yaml
version: "3.0"
services:

  jenkins:
    image: jenkins-img
    container_name: jenkins-container
    build: ./

    volumes:
     - /var/run/docker.sock:/var/run/docker.sock
     - ./jenkins_home:/var/jenkins_home
       ports:

     - "8081:8080"
```

```
PS D:\DMC\Jenkins_CI-CD\Installation> docker-compose up
[+] Running 1/1
 ! jenkins Warning
[+] Building 8.9s (16/16) FINISHED
 => [jenkins internal] load .dockerignore
 => => transferring context: 2B
 => [jenkins internal] load build definition from Dockerfile
 => => transferring dockerfile: 1.99kB
 => [jenkins internal] load metadata for docker.io/jenkins/je
 => [jenkins internal] load build context
 => => transferring context: 32B
 => [jenkins  1/11] FROM docker.io/jenkins/jenkins:lts
```

# CI/CD con Jenkins para el Desarrollo Local

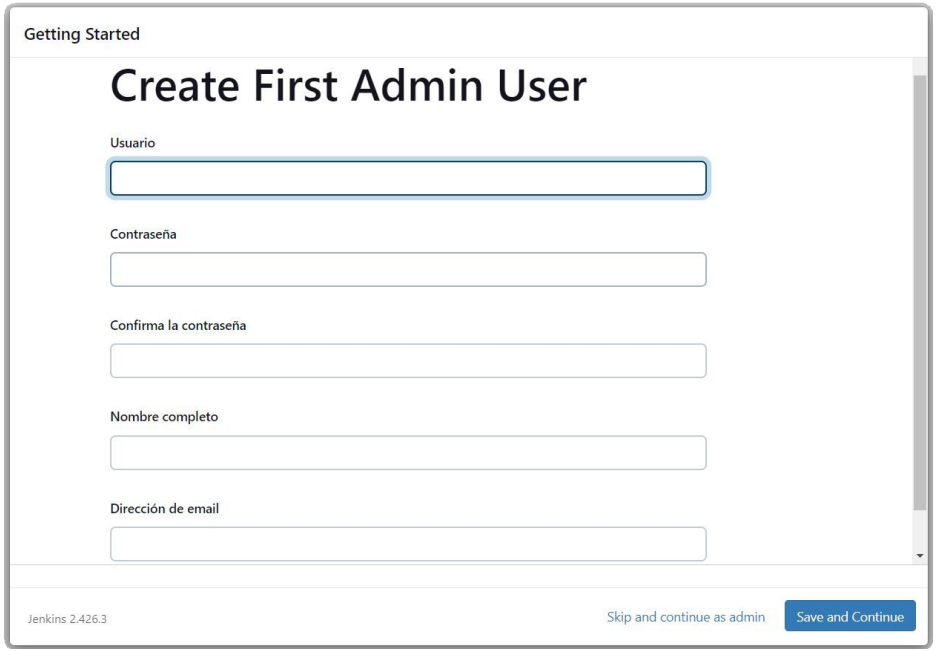
Instalar Jenkins en máquina local

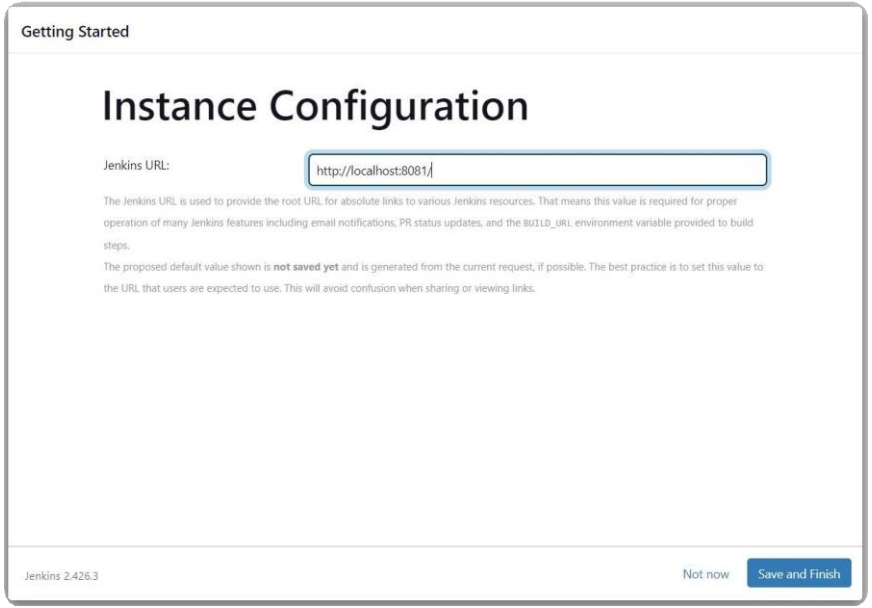


# CI/CD con Jenkins para el Desarrollo Local

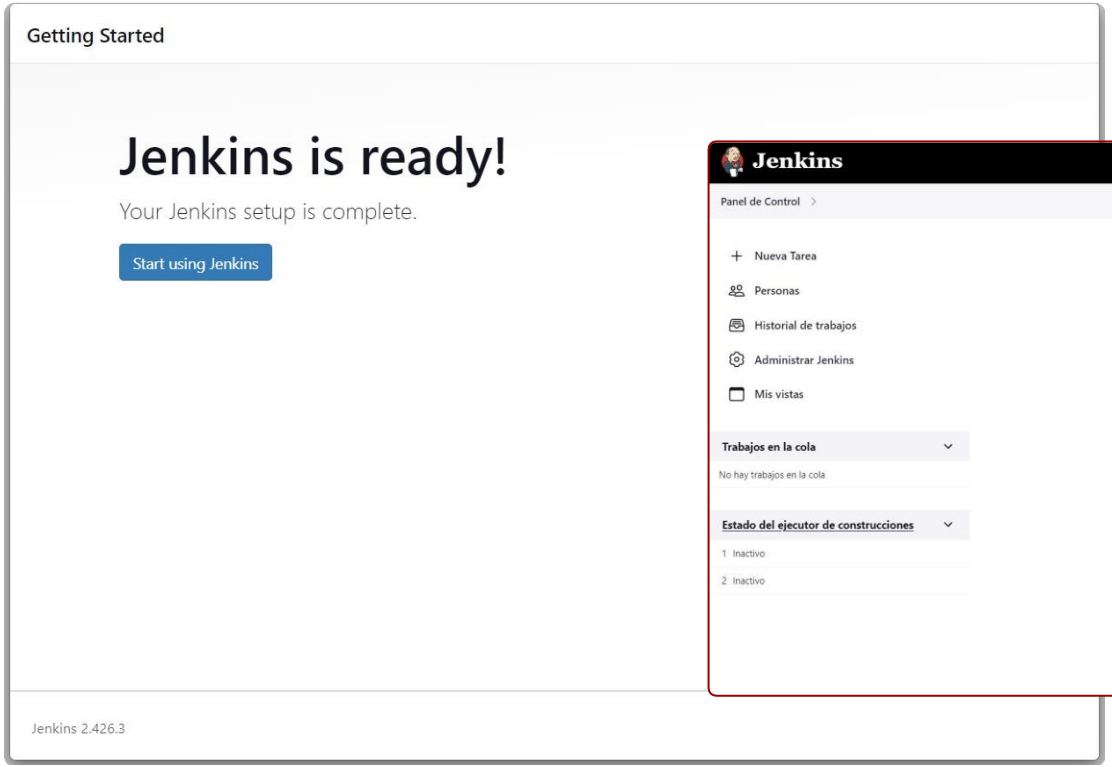
Instalar Jenkins en máquina local

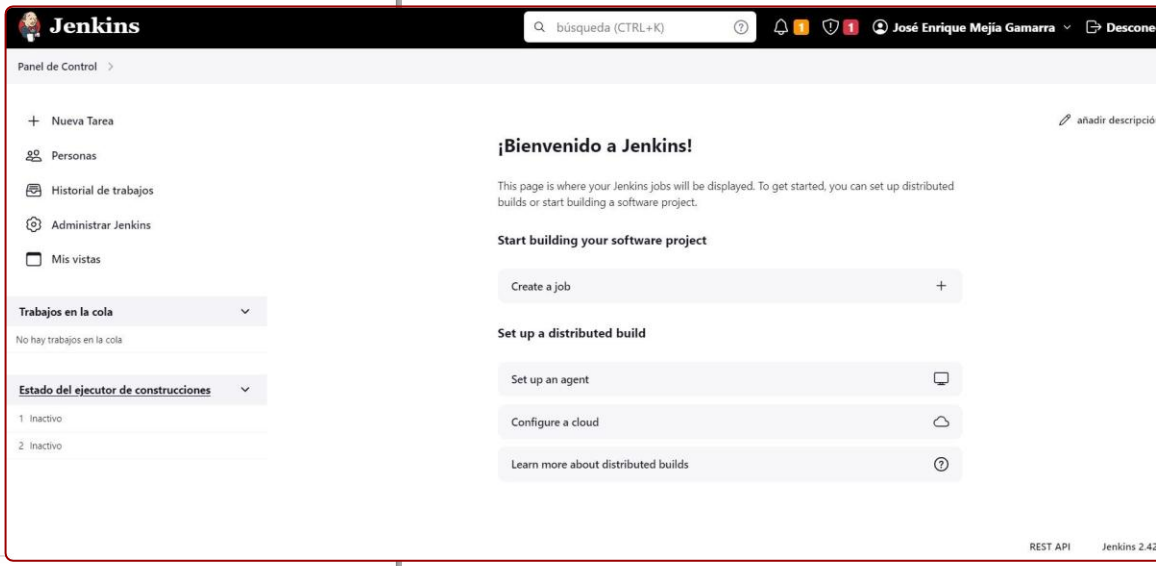


# CI/CD con Jenkins para el Desarrollo Local

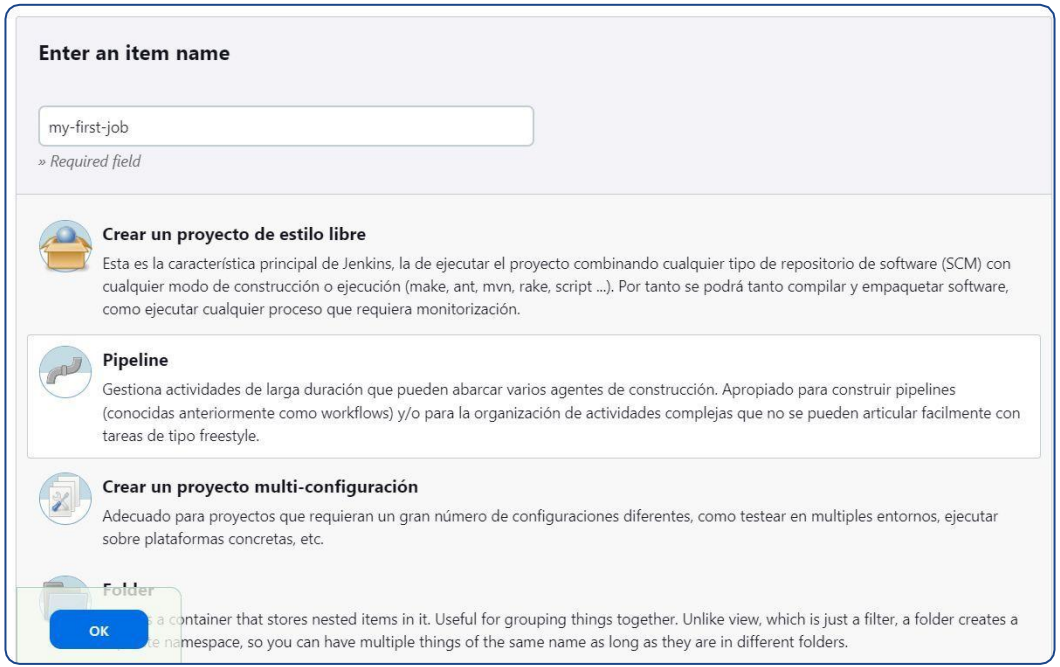
Instalar Jenkins en máquina local



# CI/CD con Jenkins para el Desarrollo Local

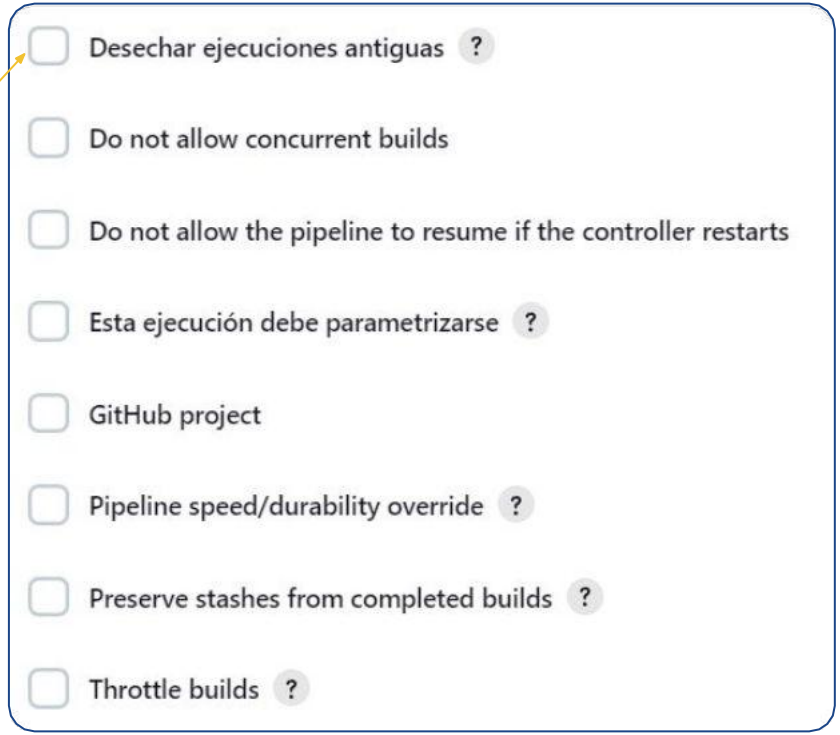
Crear Job - Pipeline



# CI/CD con Jenkins para el Desarrollo Local

Crear Job - Pipeline

Esta opción permite a Jenkins **administrar automáticamente el historial de ejecuciones** del pipeline, eliminando las más antiguas según ciertos criterios configurados.

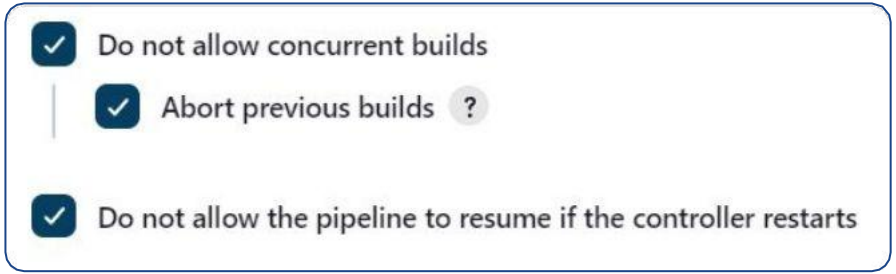



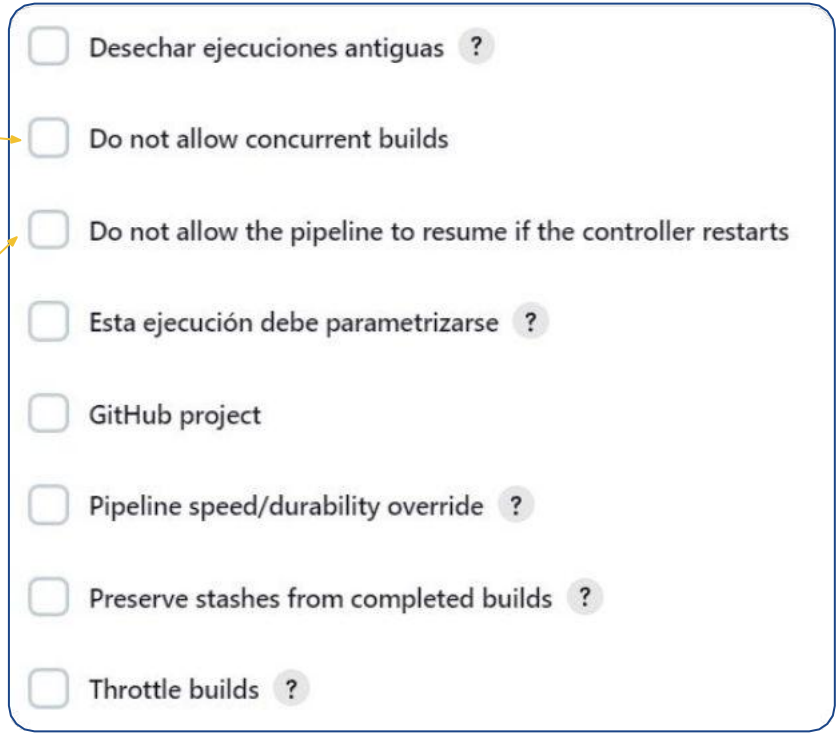
# CI/CD con Jenkins para el Desarrollo Local

Crear Job - Pipeline

Si esta opción está activada, **previene que se ejecuten múltiples** instancias del mismo pipeline al mismo tiempo.

**Impide que un pipeline** que estaba en ejecución al momento de reiniciar el controlador (Jenkins) **se reanude automáticamente.**

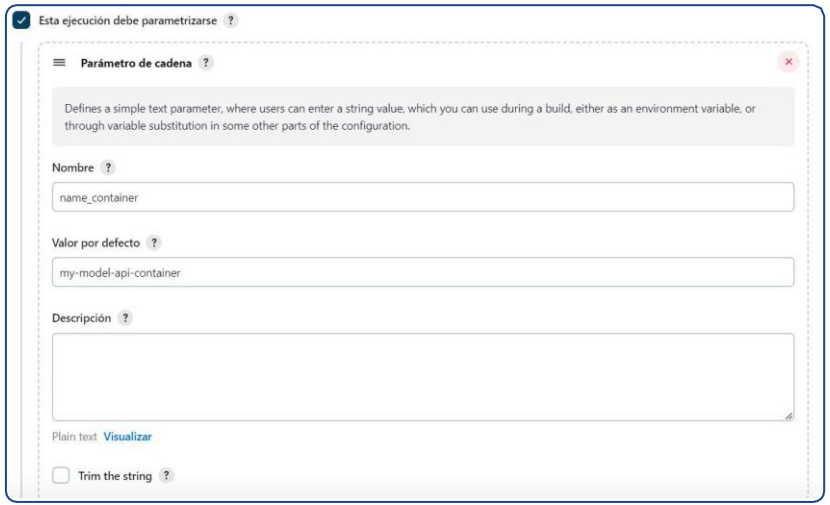


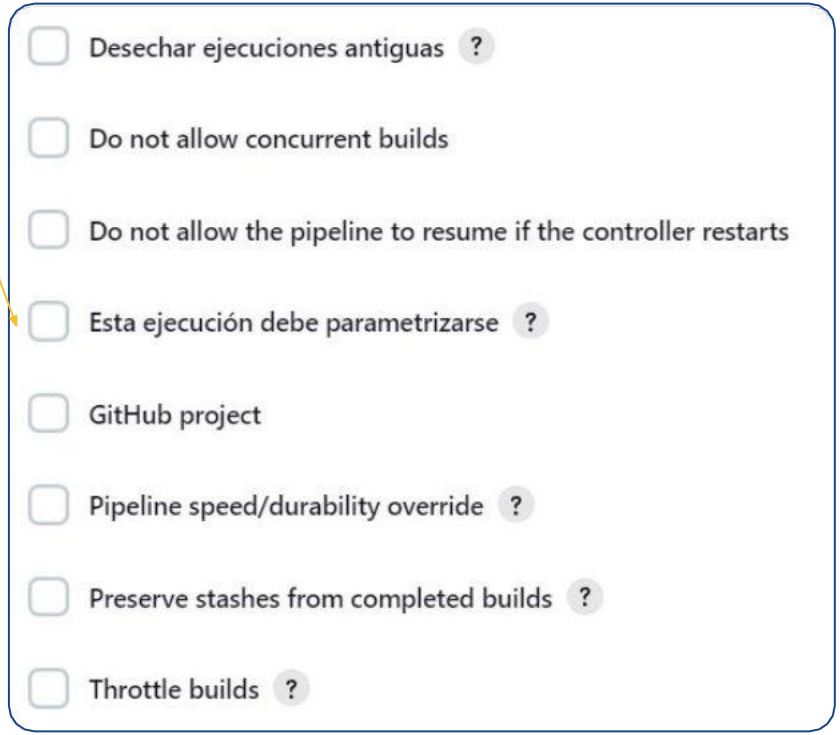



# CI/CD con Jenkins para el Desarrollo Local

Crear Job - Pipeline

Permite **definir parámetros para el pipeline**, los cuales pueden ser utilizados en la ejecución para modificar el comportamiento del pipeline basado en valores de entrada proporcionados al iniciar la construcción.



# CI/CD con Jenkins para el Desarrollo Local

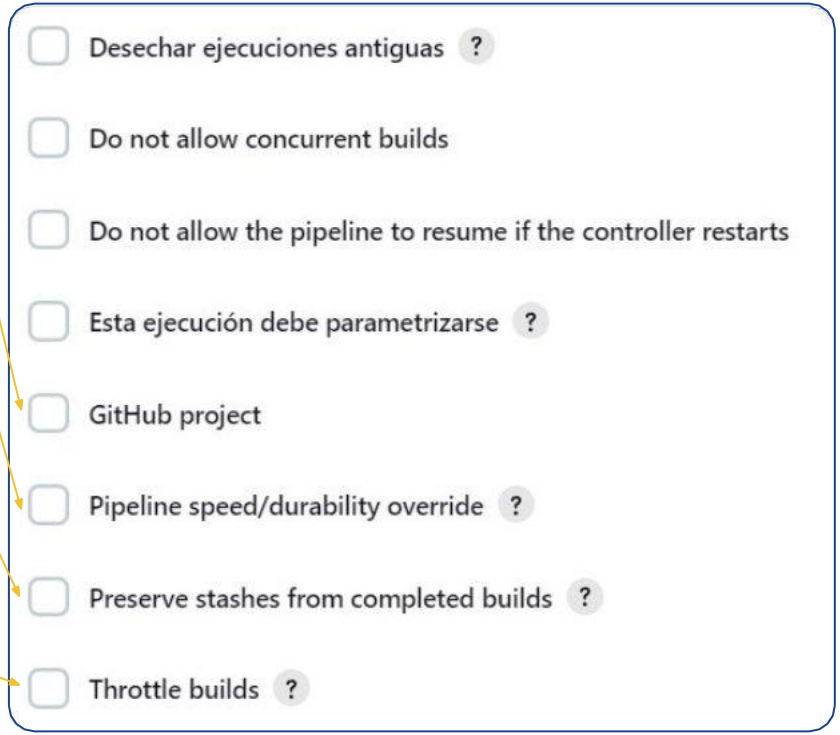
Crear Job - Pipeline

Esta opción permite **asociar el pipeline con un proyecto** específico en GitHub

Permite ajustar la configuración de **durabilidad del pipeline, equilibrando entre el rendimiento y la capacidad** de sobrevivir a reinicios inesperados.

Esta configuración permite conservar los **stashes** creados durante una ejecución del pipeline para que puedan ser **usados en ejecuciones futuras**. Los stashes son **colecciones temporales de archivos** que pueden ser guardados y restaurados en diferentes etapas del pipeline.

Permite **limitar el número de ejecuciones del pipeline** durante un período de tiempo determinado o asegurar que solo un cierto número de ejecuciones puedan correr simultáneamente dentro de una categoría de pipelines.

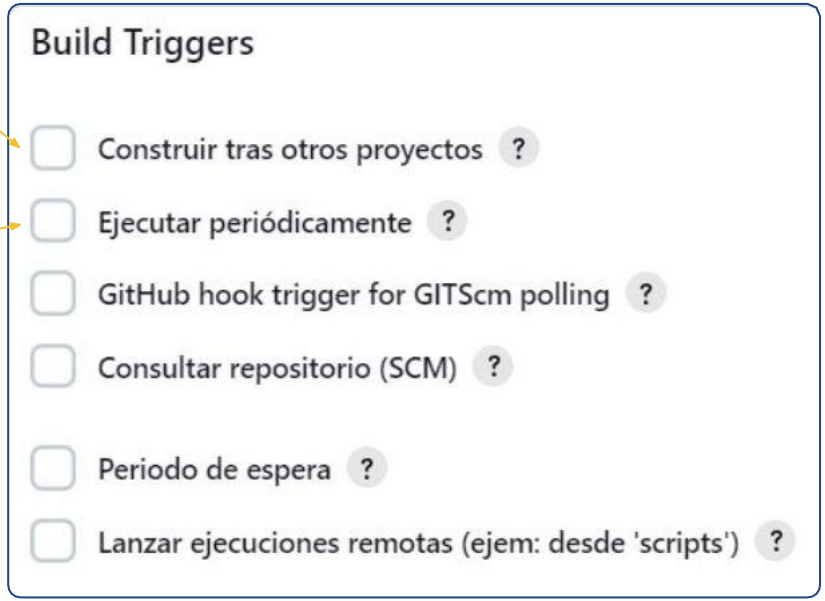


# CI/CD con Jenkins para el Desarrollo Local

Crear Job - Pipeline

Esta opción permite configurar el pipeline para que se **ejecute automáticamente después de que otro proyecto** o pipeline específico se haya construido.

Permite configurar el pipeline para que **se ejecute en intervalos regulares**, especificados mediante una sintaxis cron.

# CI/CD con Jenkins para el Desarrollo Local

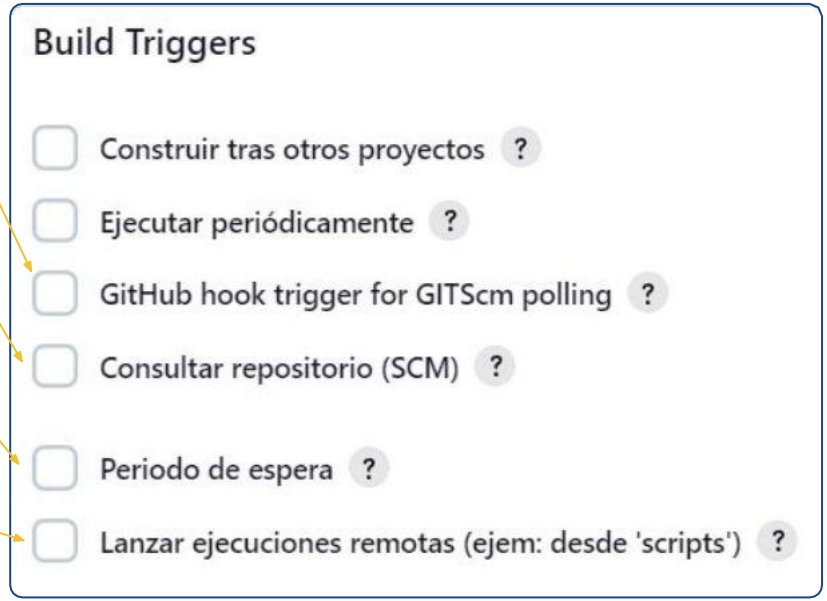
Crear Job - Pipeline

Este disparador es utilizado para iniciar automáticamente el pipeline cuando se **detectan cambios en un repositorio de GitHub.**

Con esta opción, Jenkins **revisará periódicamente el repositorio de control de versiones** (SCM) para detectar cambios.

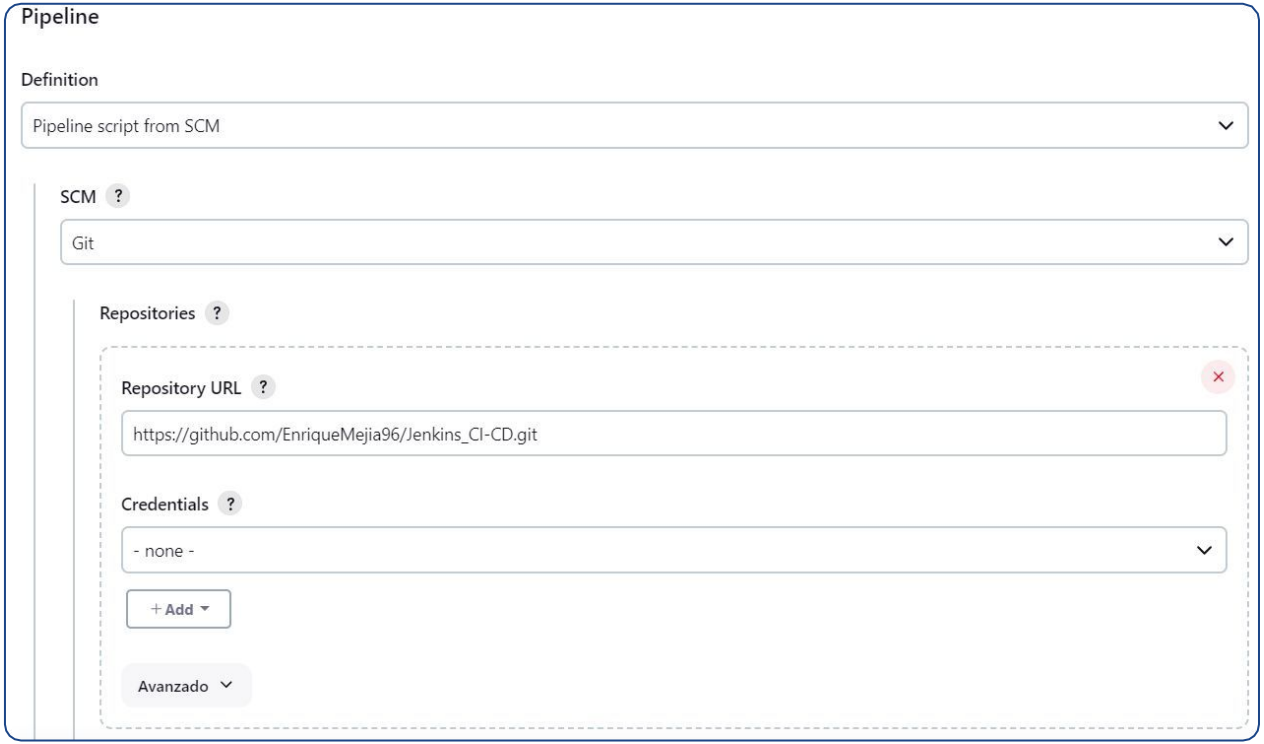
Este disparador le permite a Jenkins **esperar un cierto tiempo después de que se detectan cambios** en el SCM antes de iniciar la construcción.

Esta opción permite iniciar la construcción de Jenkins de **manera remota a través de una llamada HTTP GET o POST** a una URL específica de Jenkins.



# CI/CD con Jenkins para el Desarrollo Local

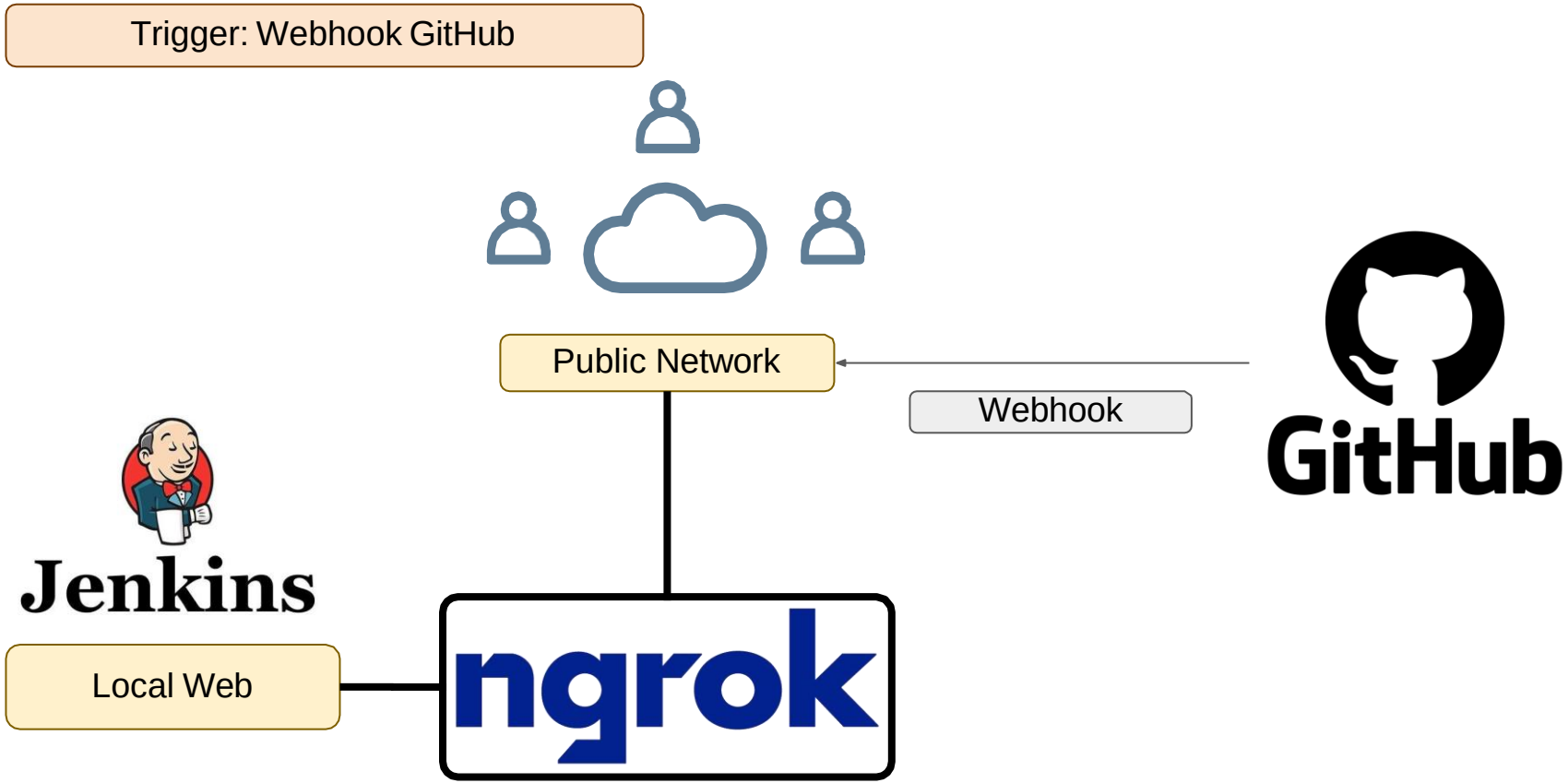
Crear Job - Pipeline



# CI/CD con Jenkins para el Desarrollo Local

Crear Job - Pipeline

# CI/CD con Jenkins para el Desarrollo Local

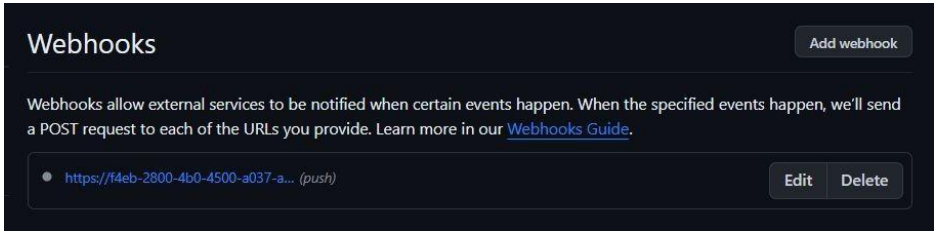


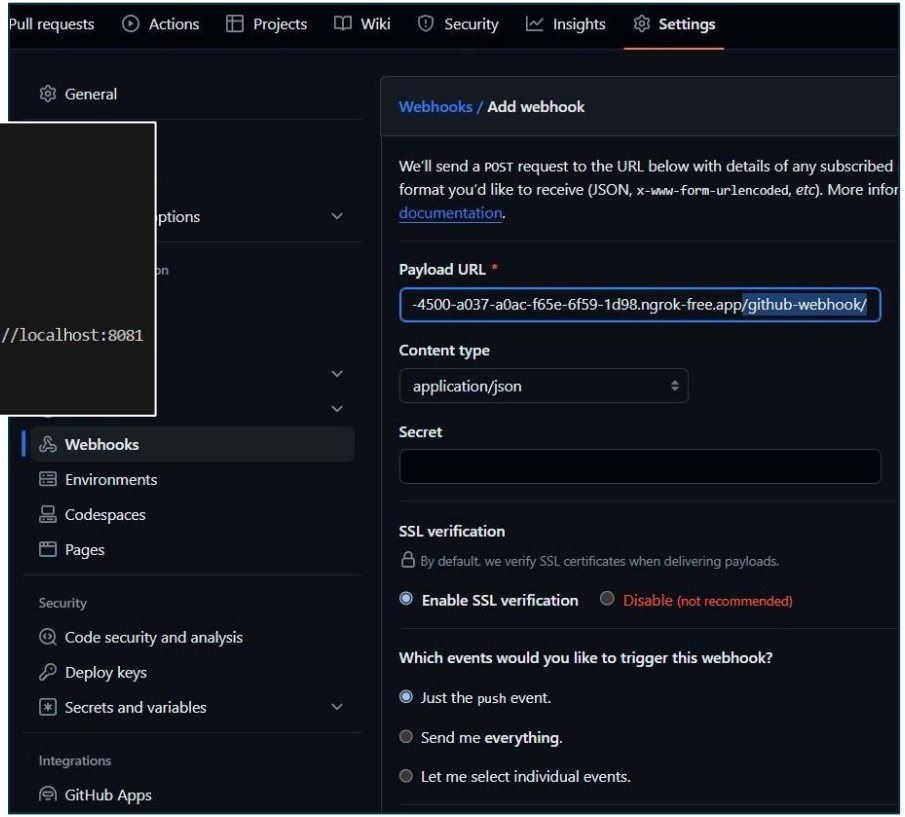
# CI/CD con Jenkins para el Desarrollo Local

Trigger: Webhook GitHub

```
ngrok http 8081
```

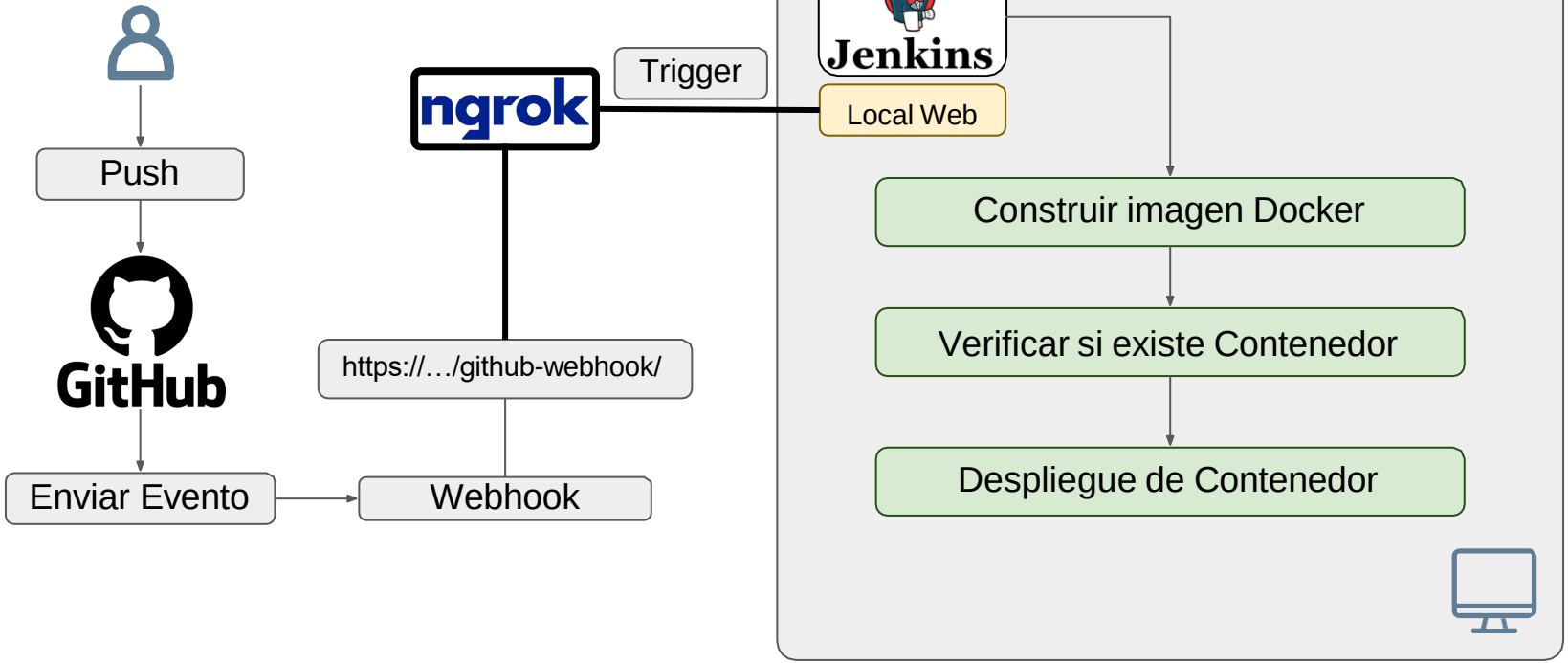
```
ngrok

Build better APIs with ngrok. Early access: ngrok.com/early-access

Session Status                online
Account                       enriquemejiagamarra@gmail.com (Plan: Free)
Version                       3.5.0
Region                        United States (us)
Latency                       172ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://f4eb-2800-4b0-4500-a037-a0ac-f65e-6f59-1d98.ngrok-free.app -> http://localhost:8081

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```



# CI/CD con Jenkins para el Desarrollo Local

## ¿Cómo funciona?



# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

## Paso 0: Guardar imagen en AWS ECR

### 0.0. Crear imagen local [Docker Desktop]

```
docker build -t my-predict-api:latest .
```

### 0.1. Iniciar sesión en AWS ECR

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin
533267339745.dkr.ecr.us-east-1.amazonaws.com
```

### 0.2. Crear repositorio en AWS ECR

```
aws ecr create-repository --repository-name my-predict-api --region us-east-1
```

### 0.3. Etiquetar imagen docker

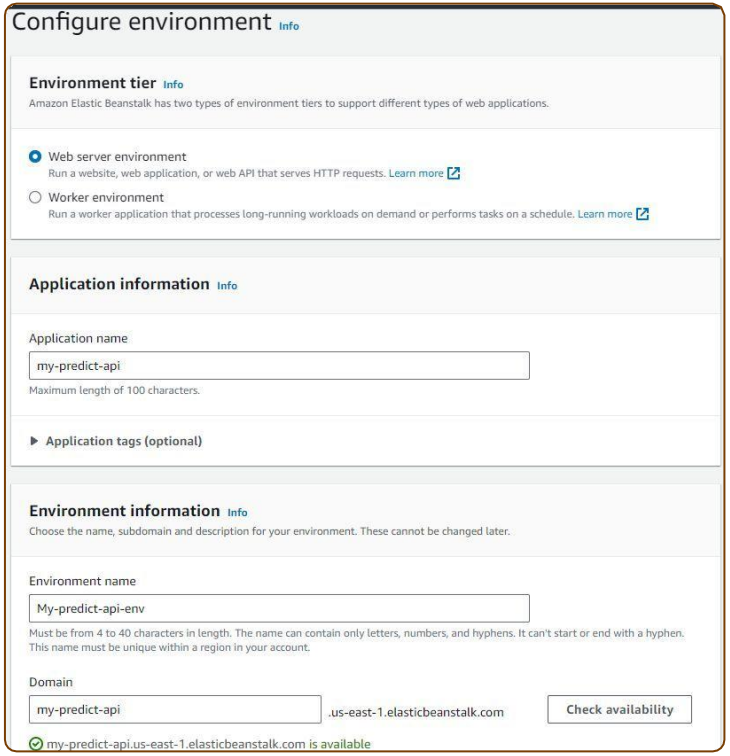
```
docker tag my-predict-api:latest 533267339745.dkr.ecr.us-east-1.amazonaws.com/my-predict-api:latest
```

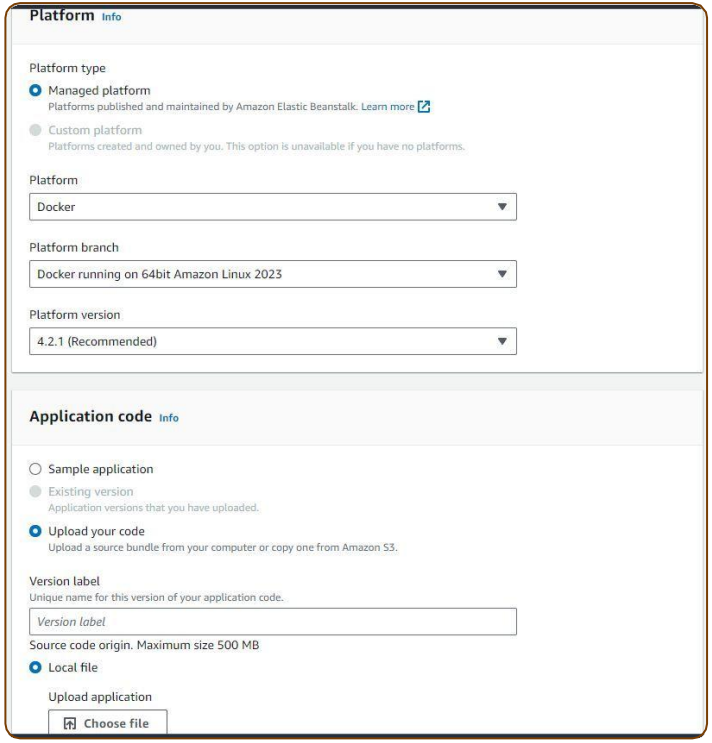
### 0.4. Hacer push a la imagen docker -> AWS ECR

```
docker push 533267339745.dkr.ecr.us-east-1.amazonaws.com/my-predict-api:latest
```

# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

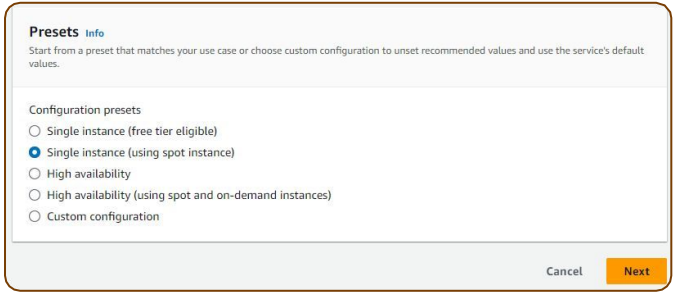
## Paso 1: Crear entorno en AWS Elastic Beanstalk

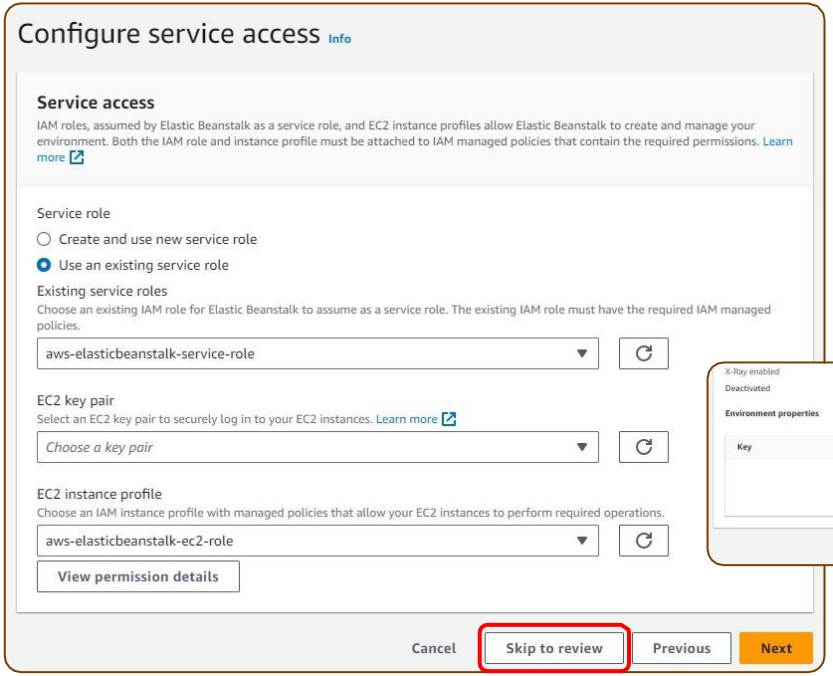



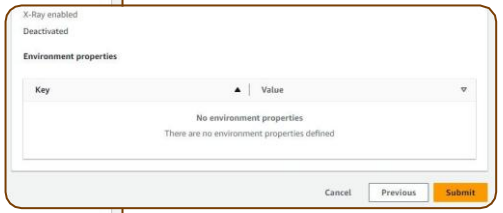



# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

## Paso 1: Crear entorno en AWS Elastic Beanstalk

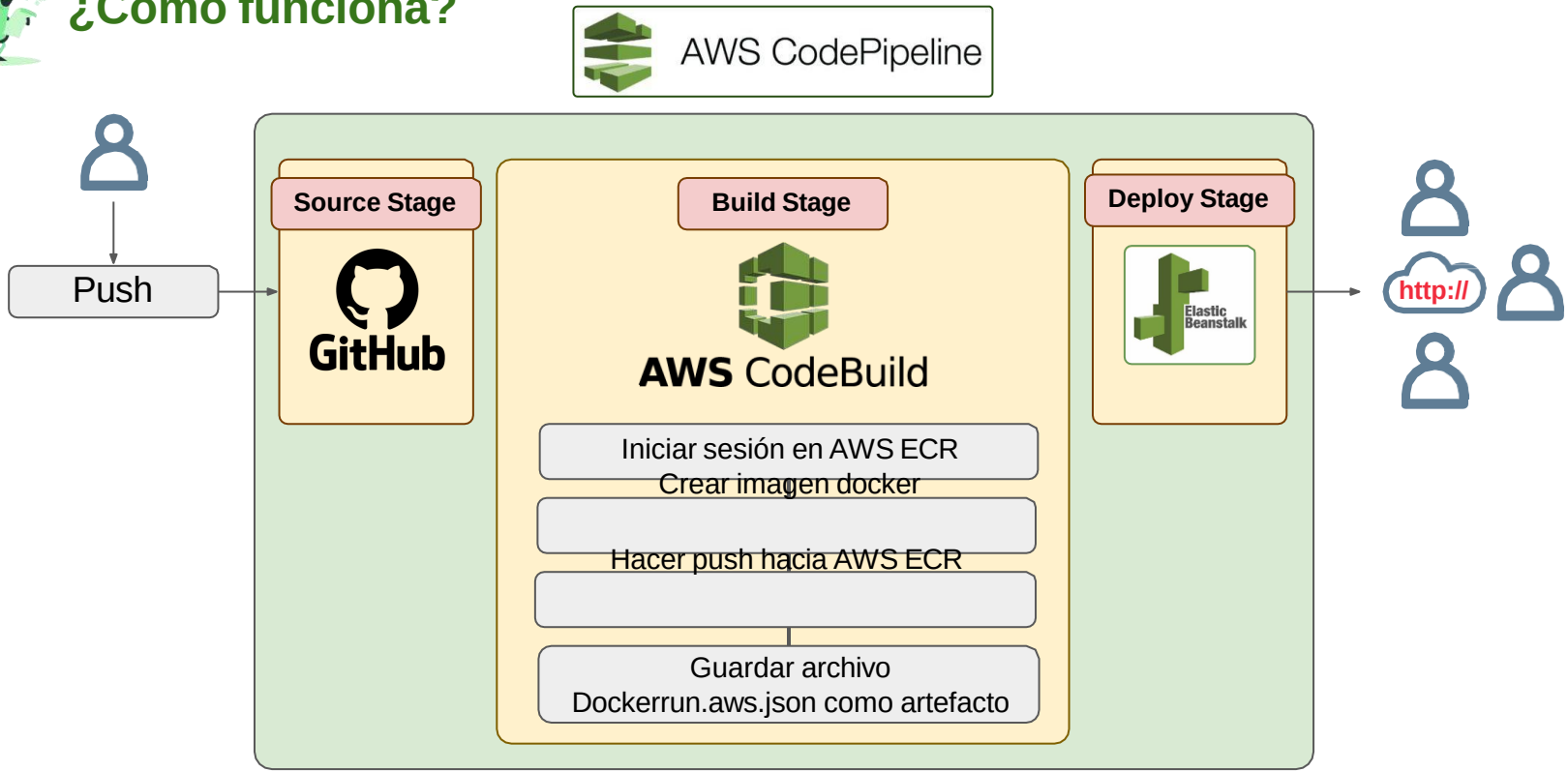



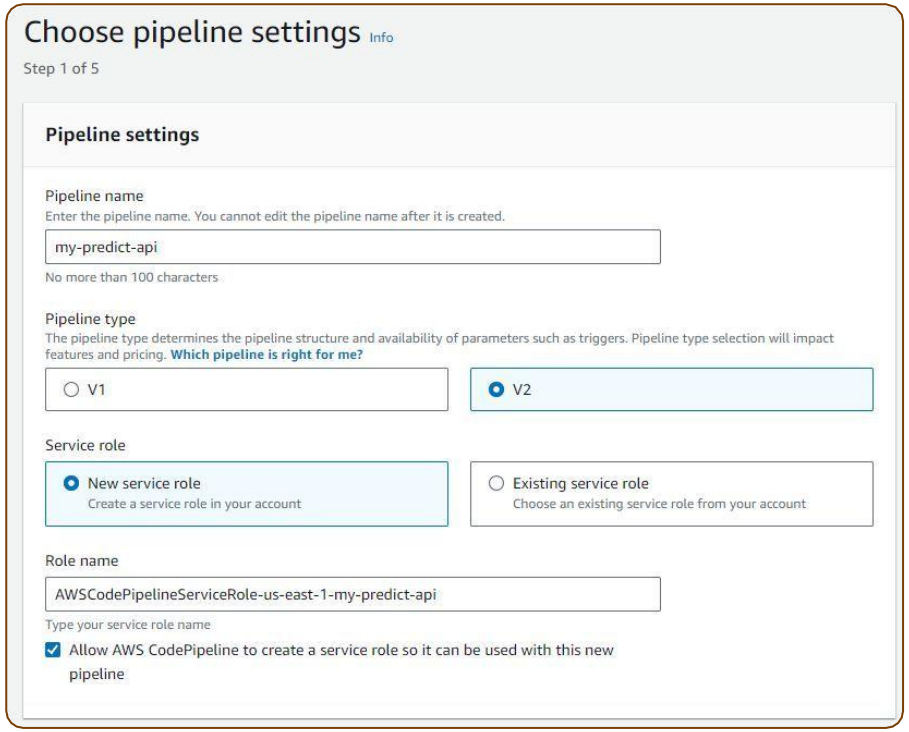
# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

## ¿Cómo funciona?



# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

## Paso 2: Crear pipeline en AWS CodePipeline



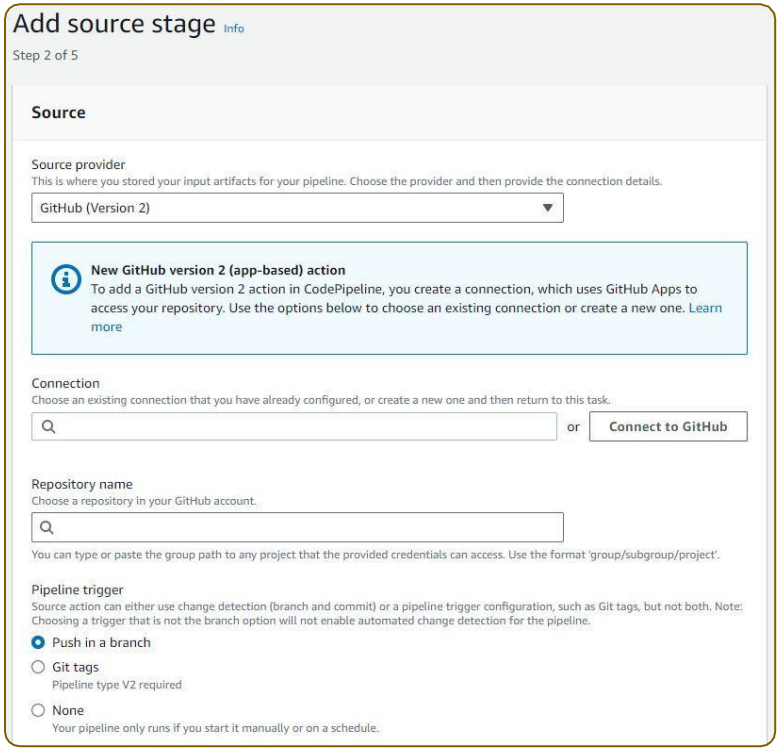

**AWS CodePipeline ofrece diferentes versiones (V1, V2)**

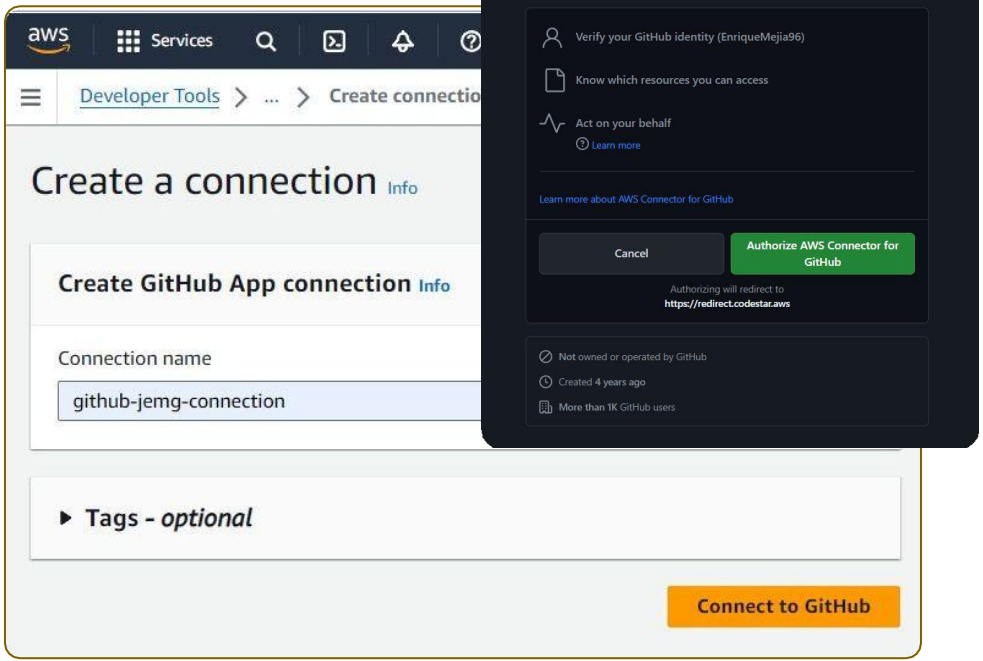
**V1:** la versión original de CodePipeline, que puede carecer de algunas funciones más nuevas, pero es familiar y **estable para las implementaciones existentes**.
**V2:** una versión más nueva que podría ofrecer **capacidades mejoradas**, rendimiento mejorado o un modelo de **precios más flexible**. Sin embargo, las funciones específicas disponibles en V1 pueden ser diferentes o no estar disponibles en V2.

Define el **rol de IAM** (Administración de identidad y acceso) que asume CodePipeline al **acceder a los recursos de AWS** en su nombre. Este rol necesita permisos suficientes para acceder a los recursos utilizados en su canalización

# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

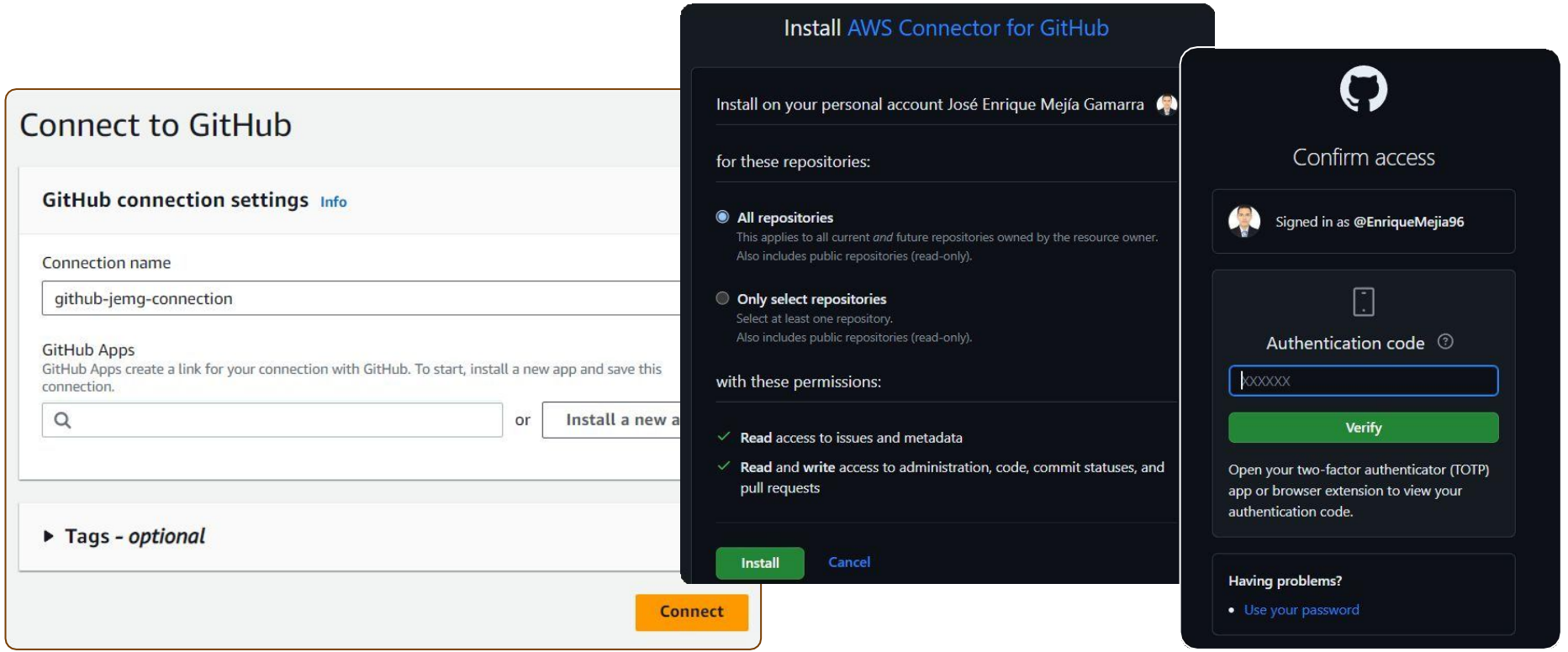
## Paso 3: AWS CodePipeline - Add Source Stage



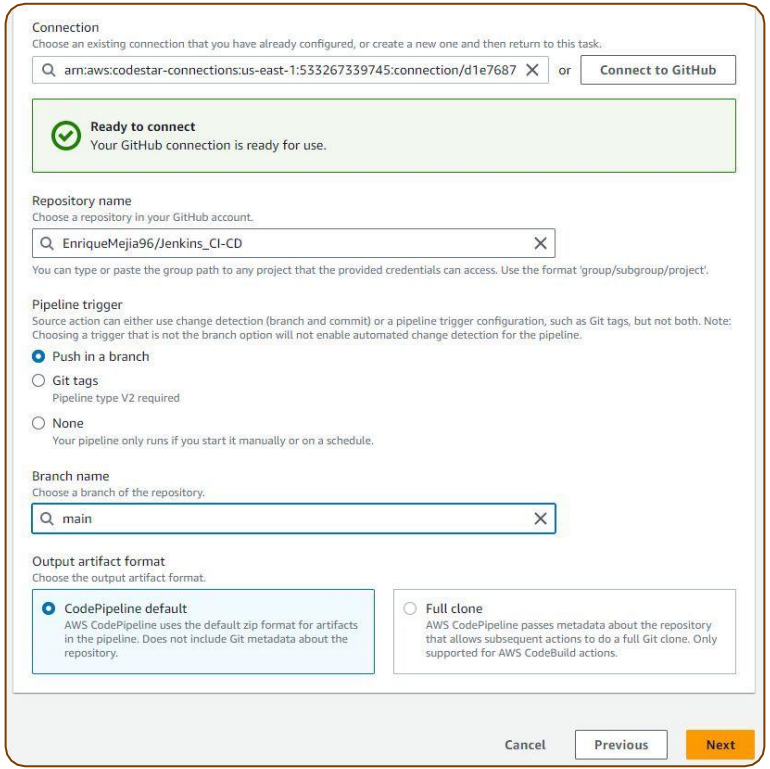
# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

## Paso 3: AWS CodePipeline - Add Source Stage



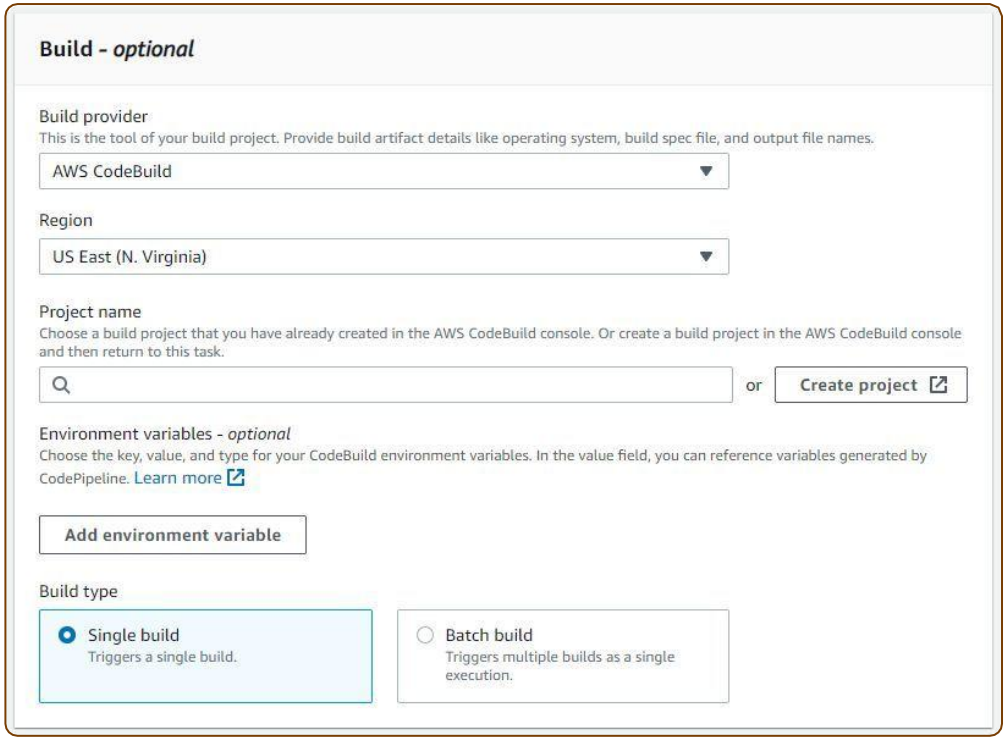
# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

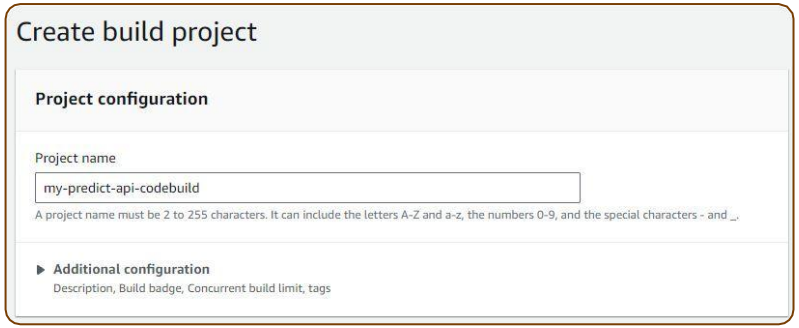
## Paso 3: AWS CodePipeline - Add Source Stage



# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

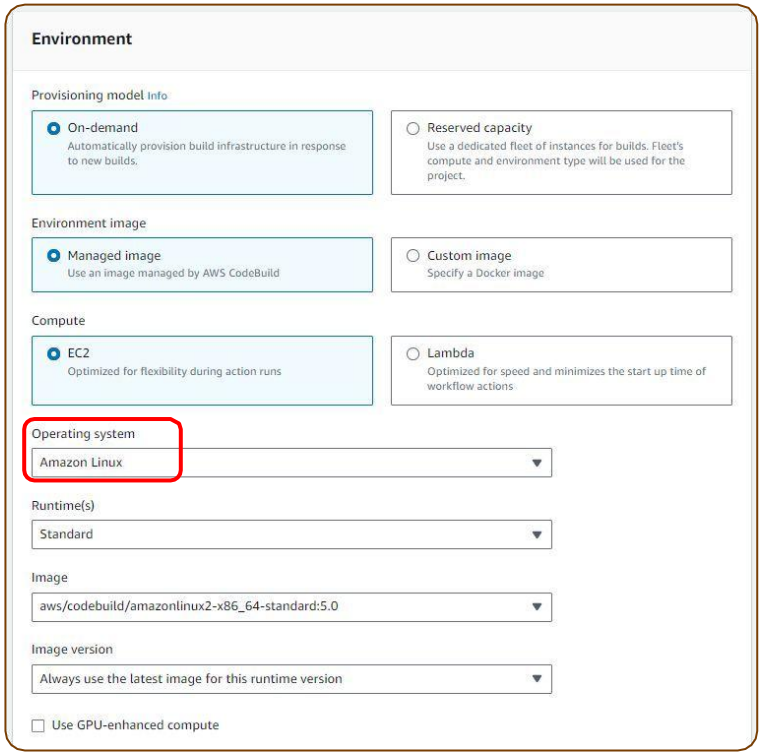
## Paso 3: AWS CodePipeline - Add Build Stage

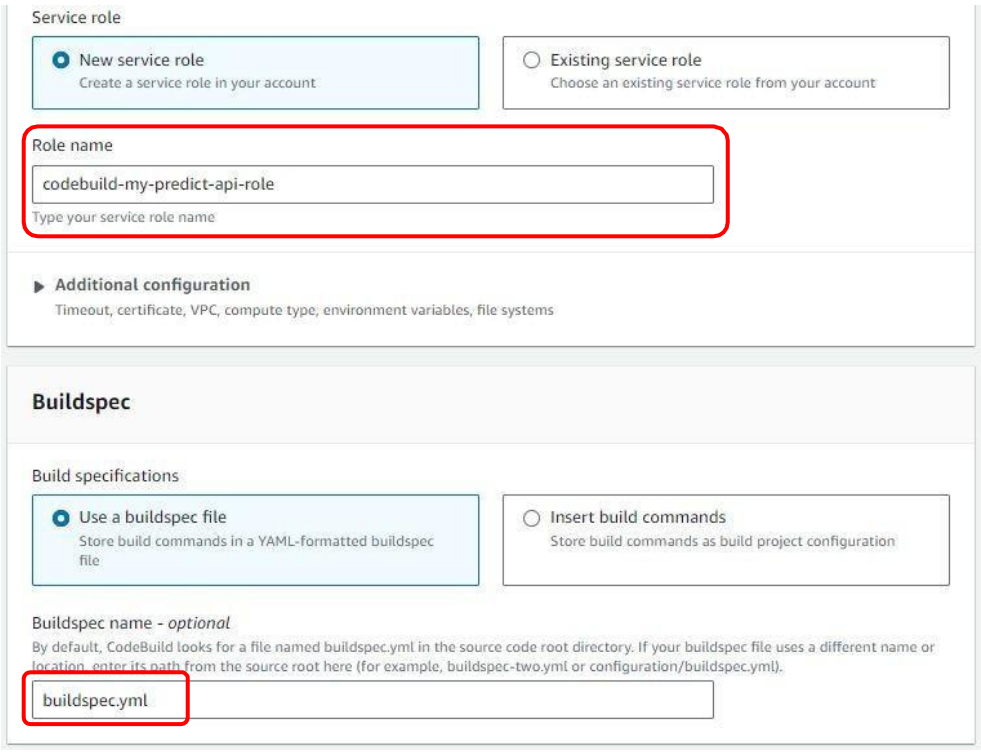


# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

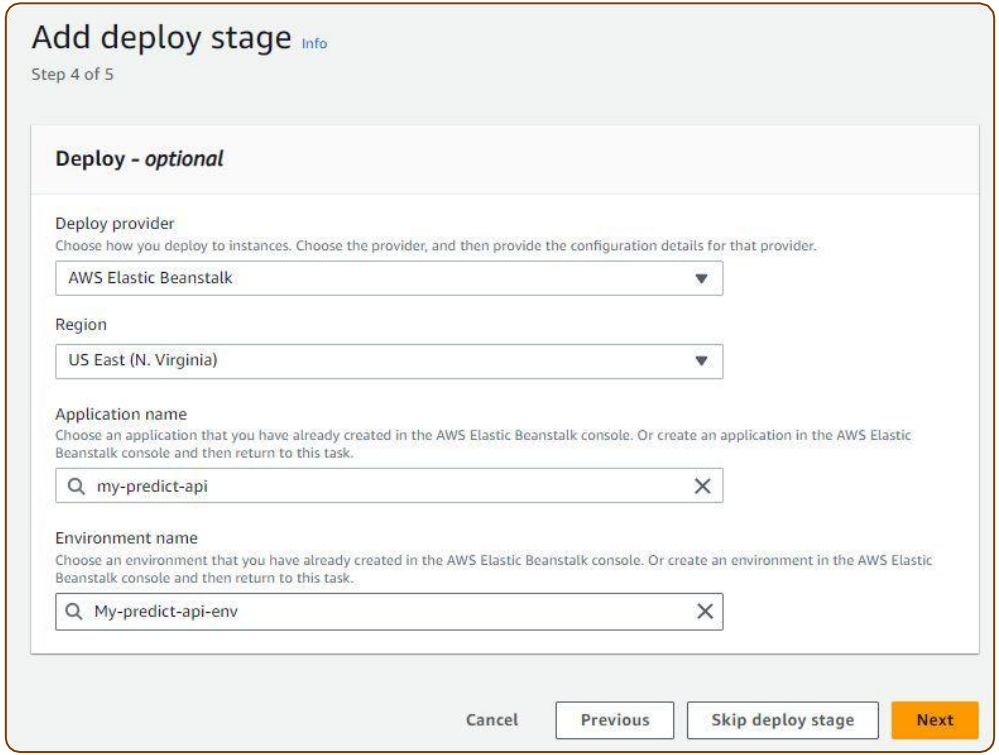
## Paso 3: AWS CodePipeline - Add Build Stage



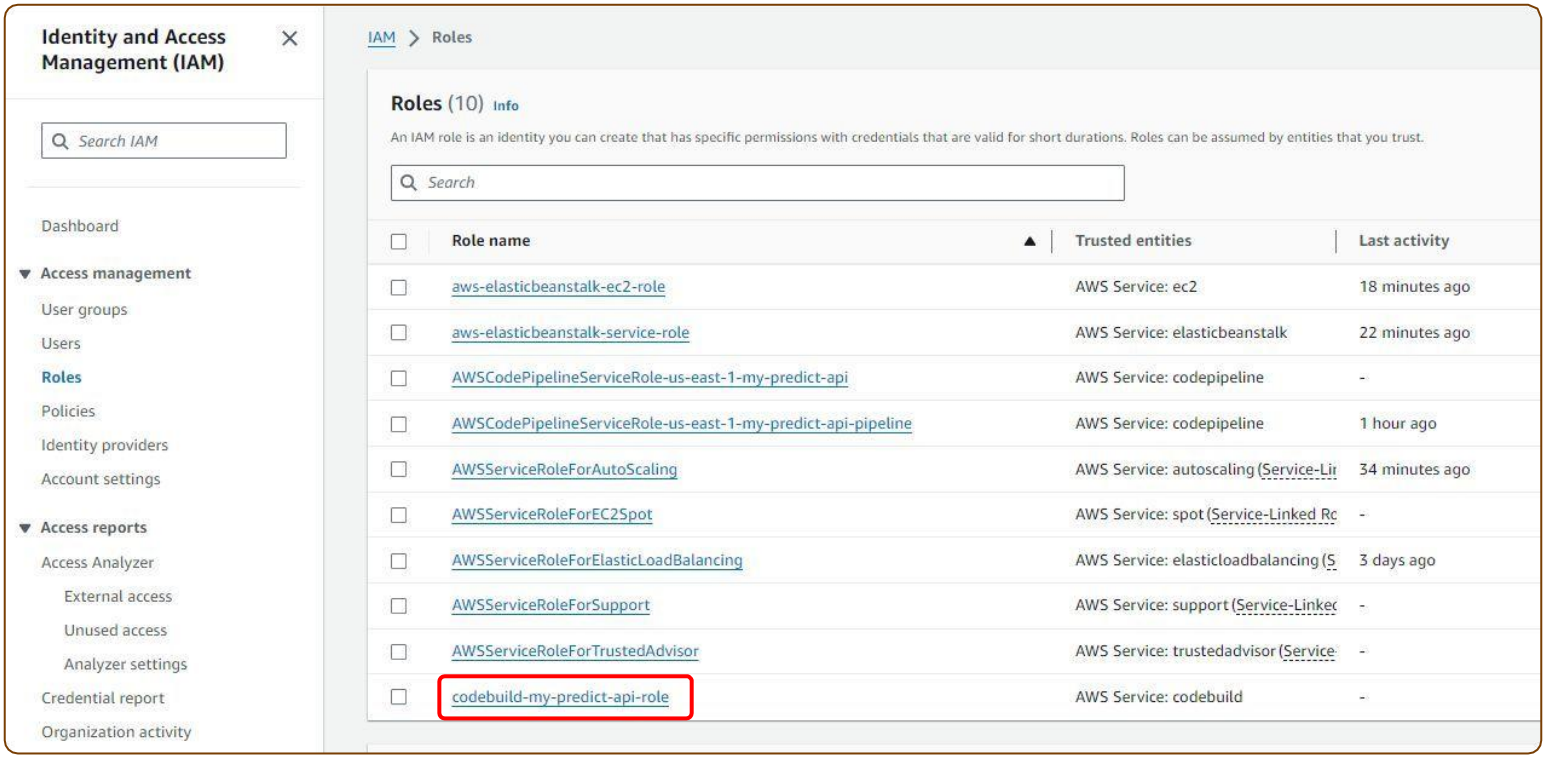
# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

## Paso 3: AWS CodePipeline - Add Deploy Stage



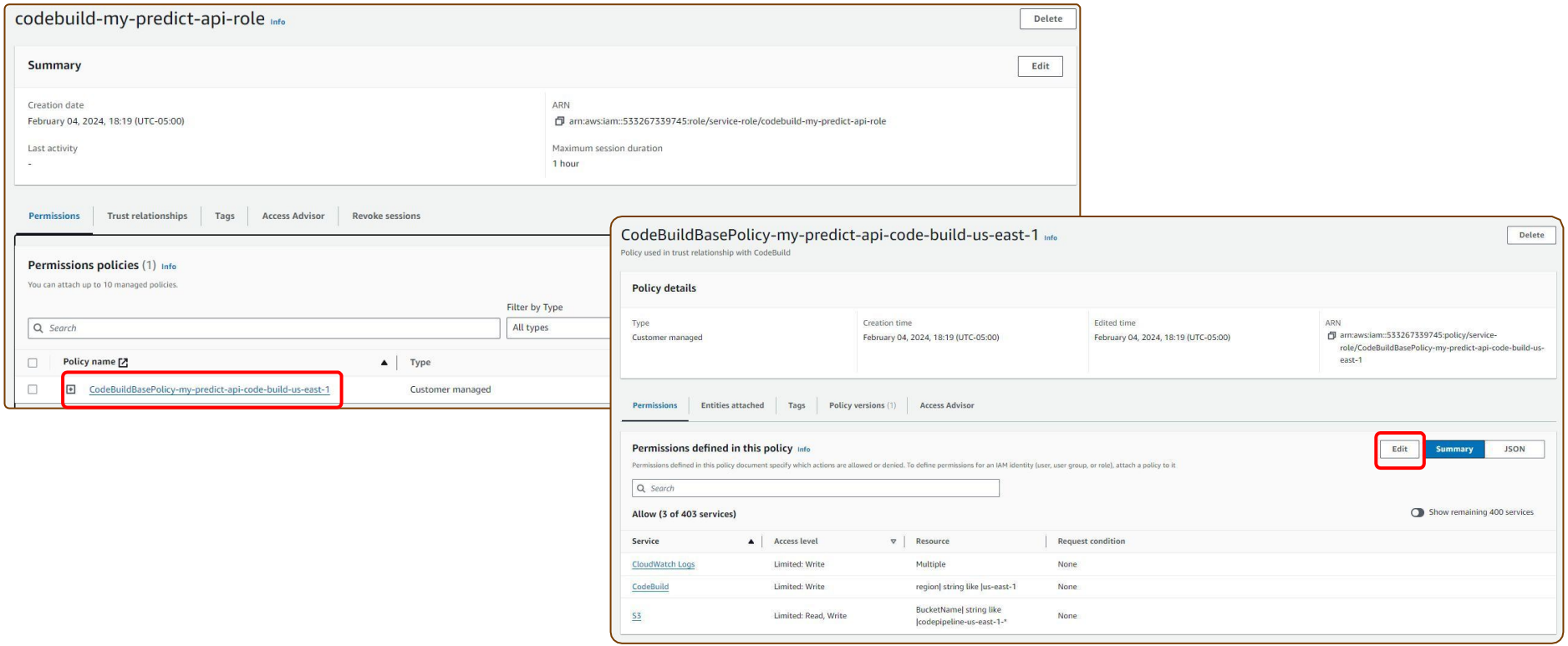
# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

## Paso 4: Agregar permisos ECR a Service Role - IAM



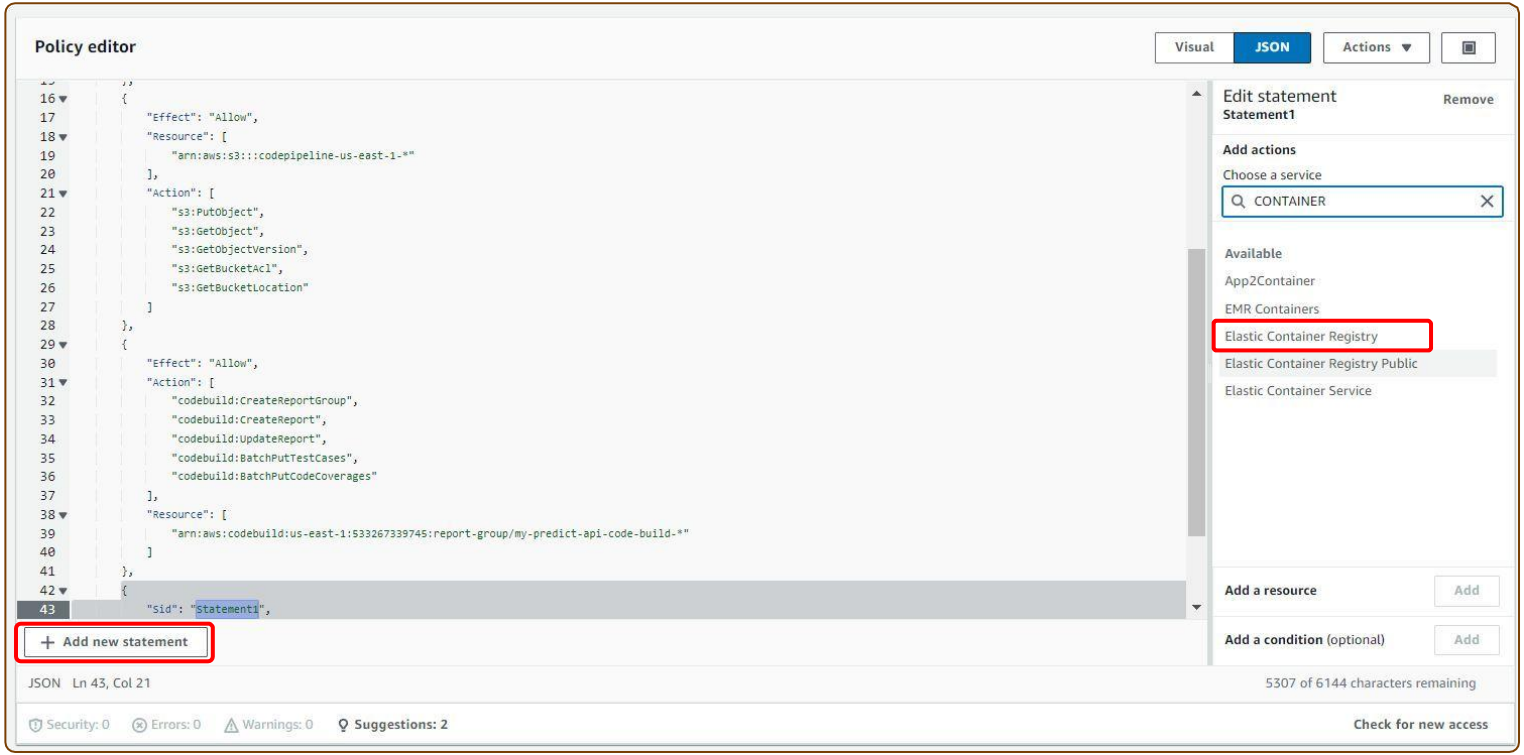
# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

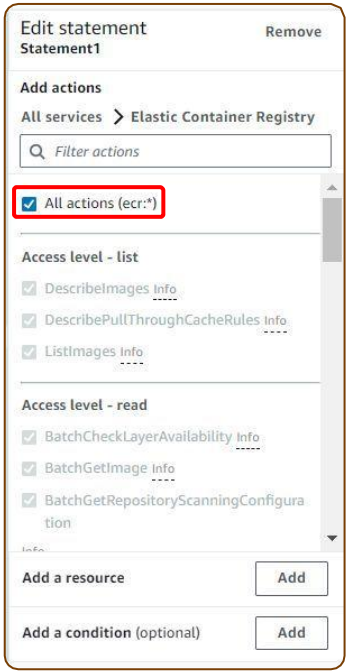
## Paso 4: Agregar permisos ECR a Service Role - IAM



# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

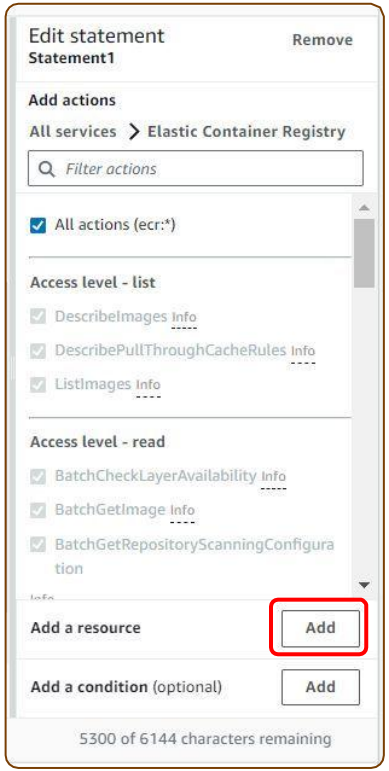
## Paso 4: Agregar permisos ECR a Service Role - IAM

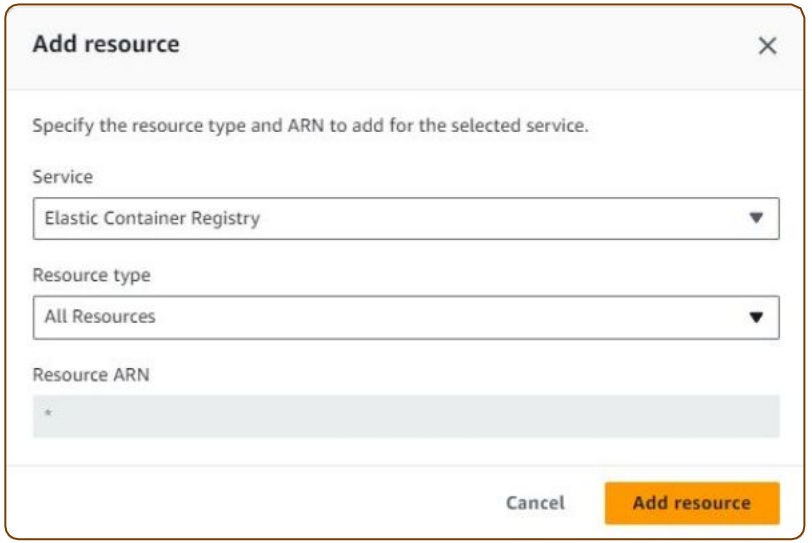


# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

## Paso 4: Agregar permisos ECR a Service Role - IAM



# CI/CD con AWS CodePipeline + AWS Elastic Beanstalk

## Paso 5: Probar Pipeline - CI/CD

buildspec.yml | Dockerrun.aws.json | app.py | Dockerfile

Jenkins_CI-CD > ml-project > app.py > home

You, 5 minutes ago | 1 author (You)

```python
# Importamos las bibliotecas necesarias
import uvicorn
from fastapi import FastAPI, File, UploadFile, Depends
import pandas as pd
from pydantic import BaseModel
import tempfile  # Biblioteca para crear archivos temporales
import shutil  # Biblioteca para copiar archivos
import joblib

# Crear una instancia de la aplicación FastAPI
app = FastAPI()

# Definir un endpoint para la raíz con método GET
@app.get("/")
def home():
    # Retorna un simple mensaje de texto
    return 'Hola mundo: Model API'      You, 5 minutes ago • up

# Definir un endpoint para manejar la subida de archivos Excel
@app.post("/upload-excel")
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | GITLENS

```
PS D:\DMC\Jenkins_CI-CD> git add .
PS D:\DMC\Jenkins_CI-CD> git commit -m 'update message'
[main da1685d] update message
 2 files changed, 1 insertion(+), 1 deletion(-)
 rename ml-project/buildspec.yml => buildspec.yml (100%)
PS D:\DMC\Jenkins_CI-CD> git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 497 bytes | 497.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/EnriqueMejia96/Jenkins_CI-CD.git
   e23cba7..da1685d  main -> main
```