Llevando el ML a producción.

Arquitectura de Contenedores.

Paradigma

Qué vamos a ver.

1. Proyectos de Machine Learning.
2. ¿Qué es el MLOps?.
3. Diseño de Código.
4. Arquitectura de Contenedores.
5. Diseño de Pipelines.

Autores y contactos.

Ángel Delgado.

Machine Learning Engineer

adelgado@paradigmadigital.com

01.

Título presentación proyecto.

Proyectos de Machine Learning.

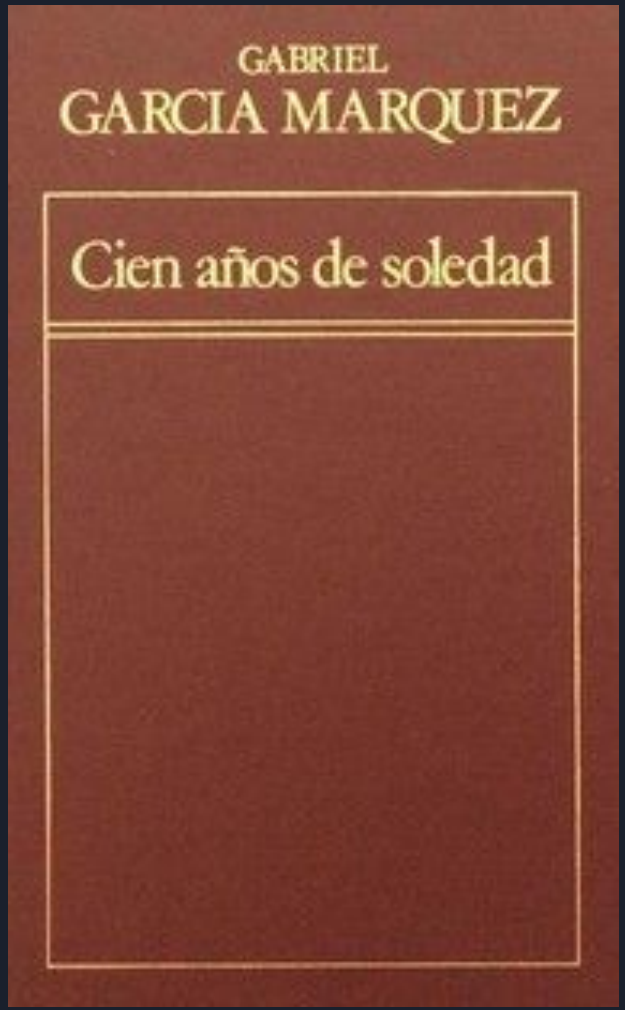
01.01

¿Qué es el Machine Learning?

¿Qué es Machine Learning?

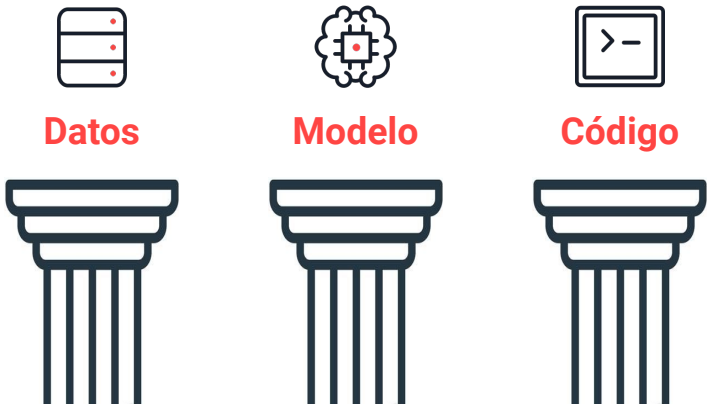
El Machine Learning es el área de la **Inteligencia Artificial**, que permite a las máquinas “aprender” a partir del procesamiento de datos.

Los procesos de cómputo que permiten realizar este aprendizaje se denominan **entrenamiento**, mientras que los que usan ese aprendizaje para inferir resultados, se denominan **predicción.**



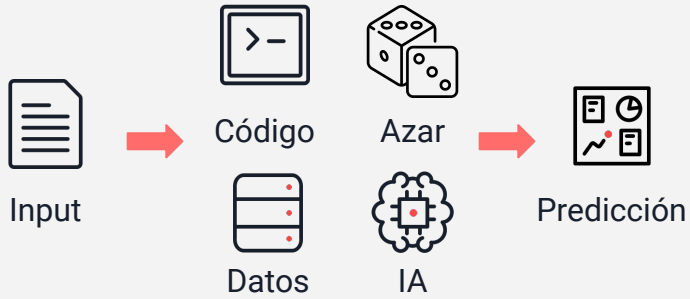
Componentes.

¿Qué es lo que caracteriza un proyecto de Machine Learning?



Desarrollo de Software

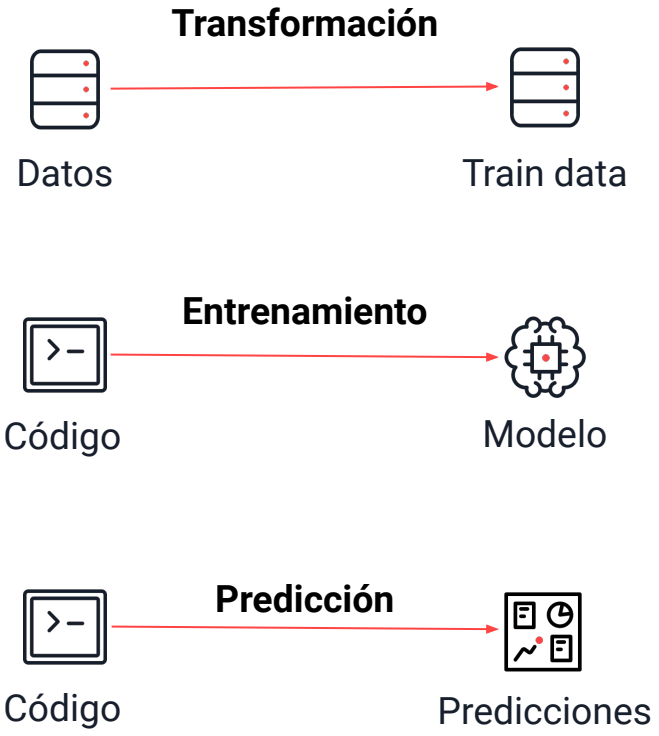
Desarrollo de Machine Learning



Procesos

Los componente que definen el Machine Learning se relacionan con una serie de procesos:

- **Transformación:** Proceso que prepara los datos.
- **Entrenamiento:** Proceso que genera un modelo.
- **Predicción:** Proceso que genera predicciones.



01.02

Título presentación proyecto. Título sección.

Dificultades.

Hay muchos más procesos!!!

Además de los procesos anteriores, es común que también existan **otros muchos procesos** que complementen a los anteriores. Existe gran diversidad de ellos, pero alguno son:

Testeo

FeatureStore

AutoML

Cross-Validation

A/B Testing

Explicabilidad

Problemas comunes.

- Diferentes **versiones** de procesos
- **Concordancia** entre código y modelos
- **Trazabilidad** entre datos y modelos
- Dificultad de **automatización**.

02.

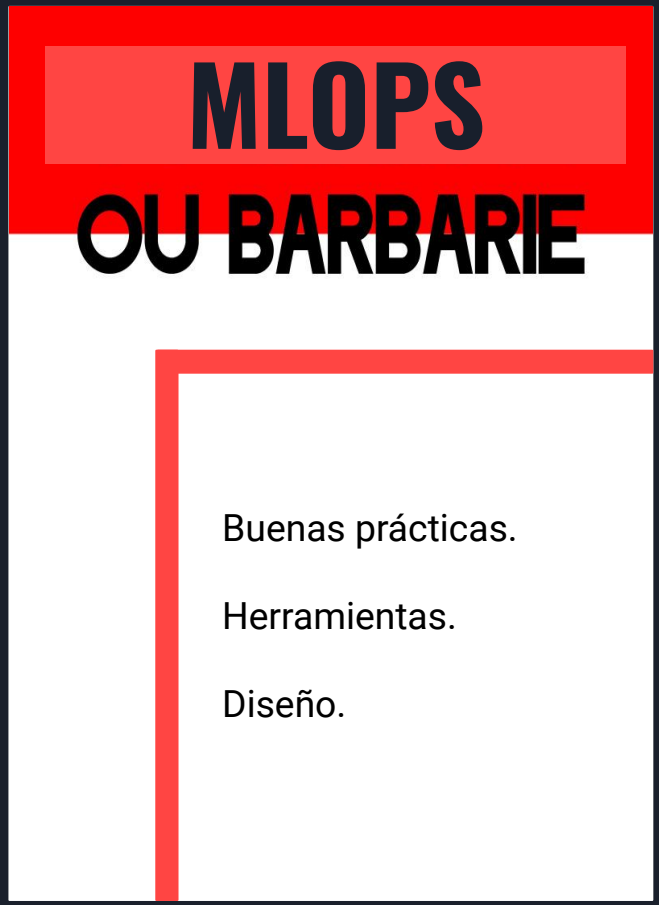
Título presentación proyecto.

¿Qué es el MLOps?

¿Qué es el MLOps?

MLOps es un conjunto de **buenas prácticas y herramientas** que nos permiten automatizar todos los procesos de un proyecto de ML, así como versionar y gobernar todas sus componentes.

¿He dicho ya que también son buenas prácticas?



02.01

Título presentación proyecto. Título sección.

Herramientas.

MLOps: Servicios y Productos.

Plataformas	Servicios	Frameworks	Productos
Airflow, Kubeflow, StepFunctions	Sagemaker, VertexAI, AzureML	mlflow, PyCaret	H2O.ai, DataRobot

- **Alto nivel de mantenimiento**
- **Bajo nivel de locking**

- **Bajo nivel de mantenimiento**
- **Alto nivel de locking**

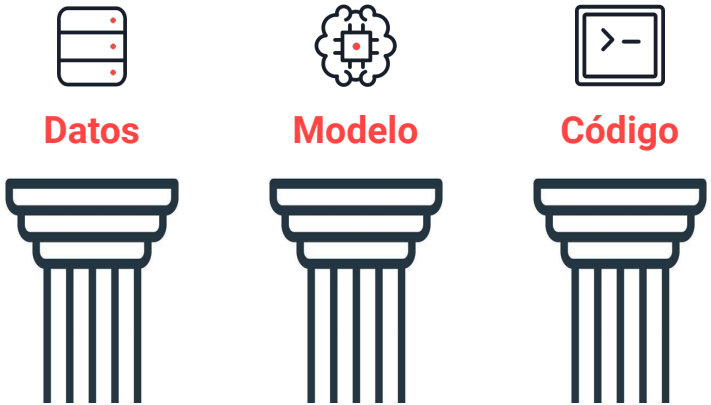
02.02

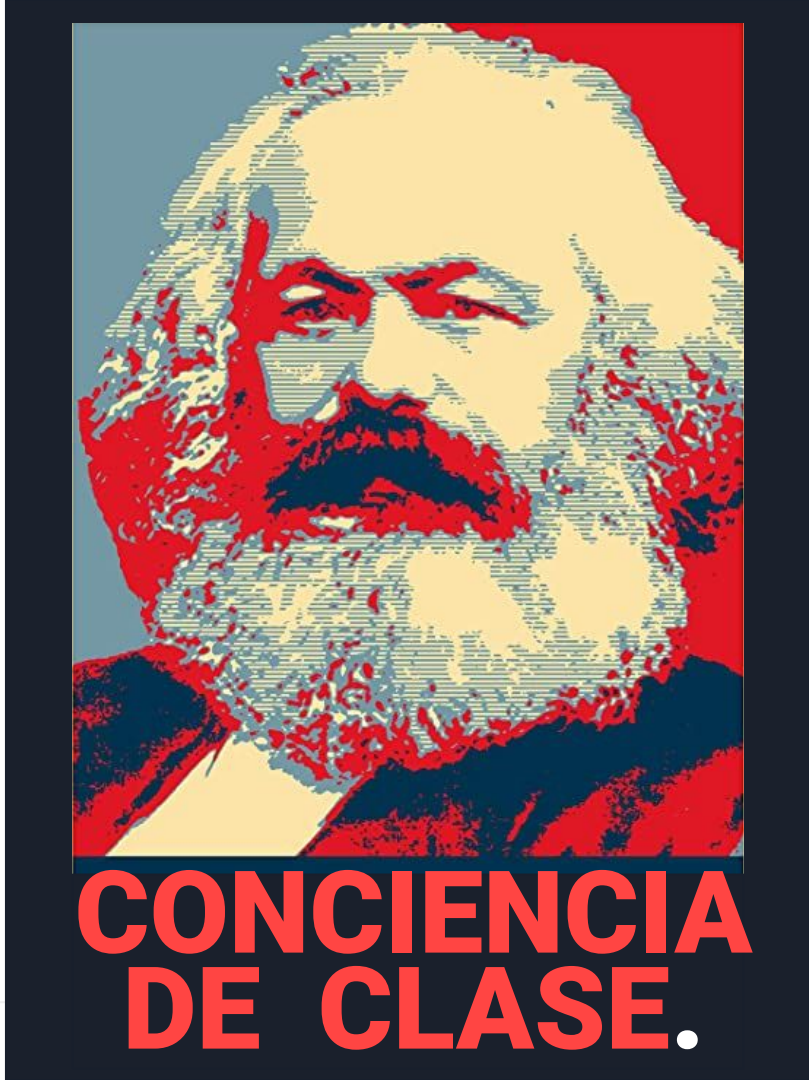
Título presentación proyecto. Título sección.

Buenas prácticas.

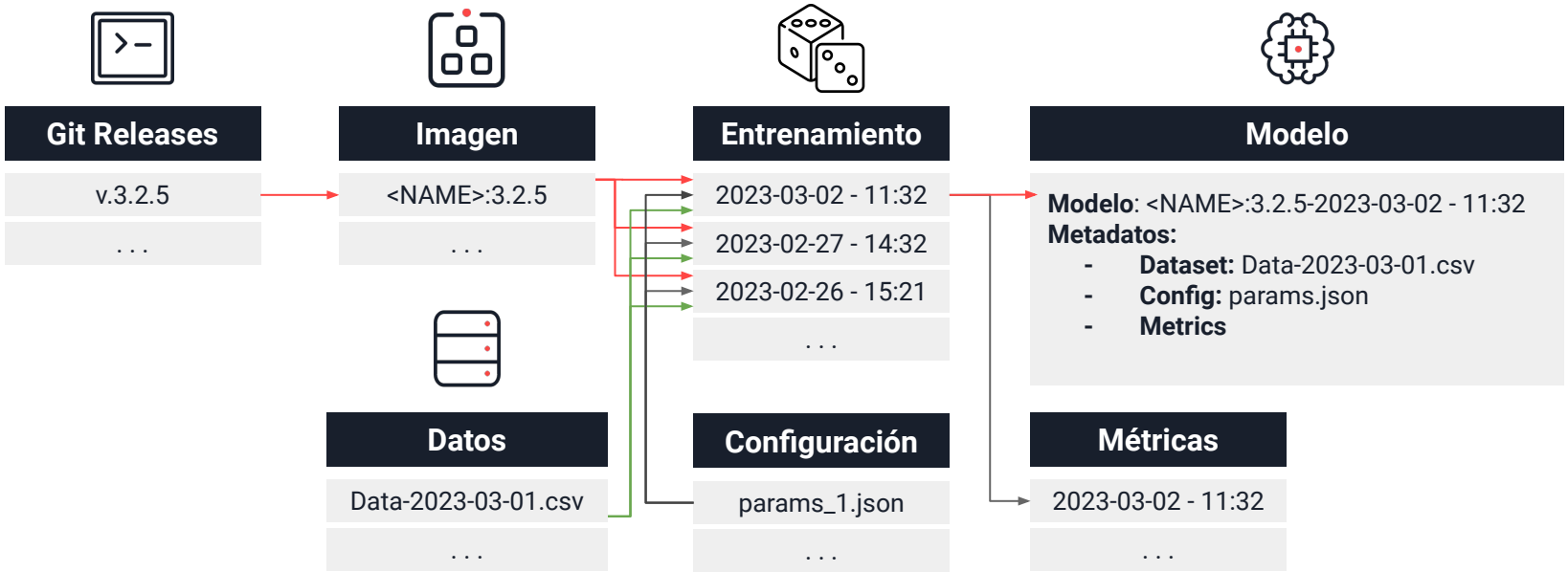
Buenas prácticas.

¿Cómo las aplicamos? Teniendo conciencia de los elementos que componen el Machine Learning.



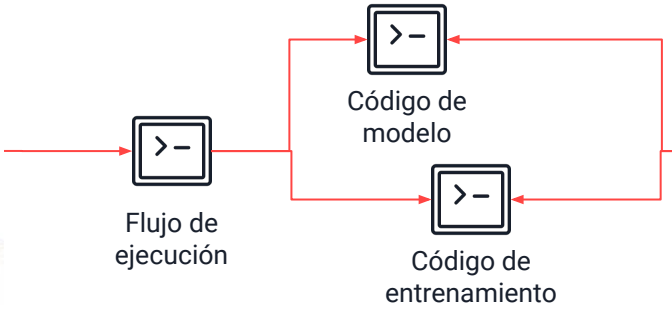


Naming y versionado.



Diseño de código: El telón de acero.

Separación entre la lógica de automatización de procesos y del código de Machine Learning.



Data Scientist

Desarrolla el **código** de Machine Learning sin preocuparse de los flujos.

Código de ML

ML Engineer

Desarrolla los **flujos** de Machine Learning sin preocuparse del algoritmo.

03.

Título presentación proyecto.

Desarrollo de código de ML.

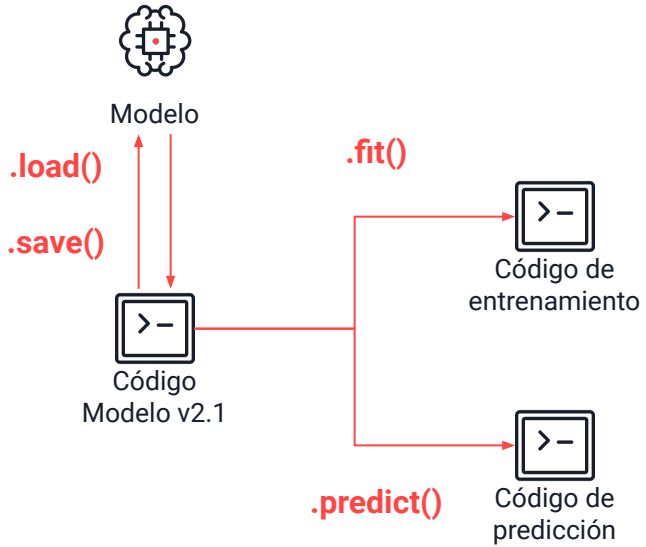
03.01

Título presentación proyecto. Título sección.

Diseño de código.

Interfaz de modelo.

- **Interfaz homogéneo** para todos los modelos
- **Separación de la lógica de ML:** código de modelo vs código de procesos.
- **Congruencia entre procesos:** entrenamiento, predicción, ...



Interfaz de modelo.

- **Interfaz homogéneo** para todos los modelos.
- **Separación de la lógica de ML:** código de modelo vs código de procesos.
- **Congruencia entre procesos:** entrenamiento, predicción, ...

```
import argparse
import pandas as pd
from .model_logic import ModelLogic


class ModelLogic():

    def __init__(self, **parameters):
        self.parameters = parameters

    def fit(self, X, y):
        ...
        return self

    def predict(self, X):
        ...
        return prediction

    def save_model(self, model_path):
        ...
        return self

    def load_model(self, model_path):
        ...
        return self
```

03.02

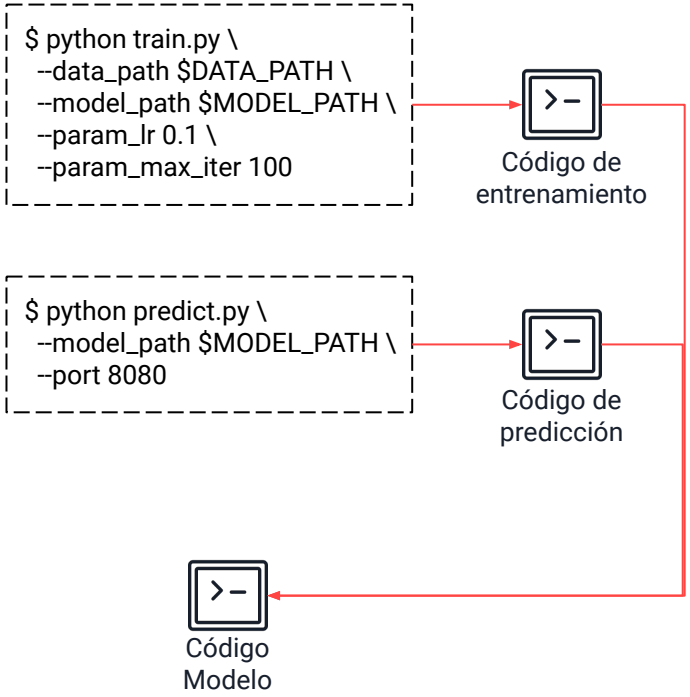
Título presentación proyecto. Título sección.

Diseño de comandos.

Diseño de comandos.

- **Parámetros de ejecución:** Para facilitar el proceso de experimentación es necesario que sea parametrizable.

- **Lógica de ejecución:** Lectura de datos, entrenamiento, guardado del modelo entrenado, lectura del modelo entrenado…



Diseño de comandos.

- **Parámetros de ejecución:** Para facilitar el proceso de experimentación es necesario que sea parametrizable.

- **Lógica de ejecución:** Lectura de datos, entrenamiento, guardado del modelo entrenado, lectura del modelo entrenado…

```
import argparse
import pandas as pd
from .model import ModelLogic


def train(args):
    df_train = pd.read_csv(args.data_path)
    X_train = df_train.drop(["Species"],axis=1)
    y_train = df_train["Species"]

    ModelLogic(learning_rate=args.lr
        ).fit(X_train, y_train
        )save(args.model_path)


if __name__ == "__main__" :

    parser = argparse.ArgumentParser()
    parser.add_argument("--lr", type=int)
    parser.add_argument("--data_path", type=str)
    parser.add_argument("--model_path", type=str)
    args = parser.parse_args()

    train(args)
```

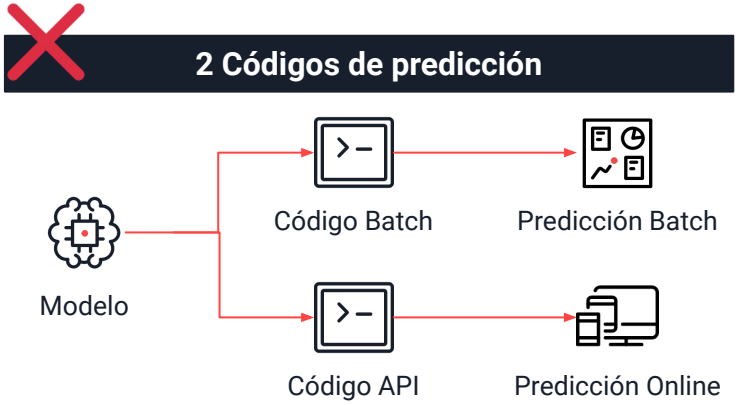
03.03

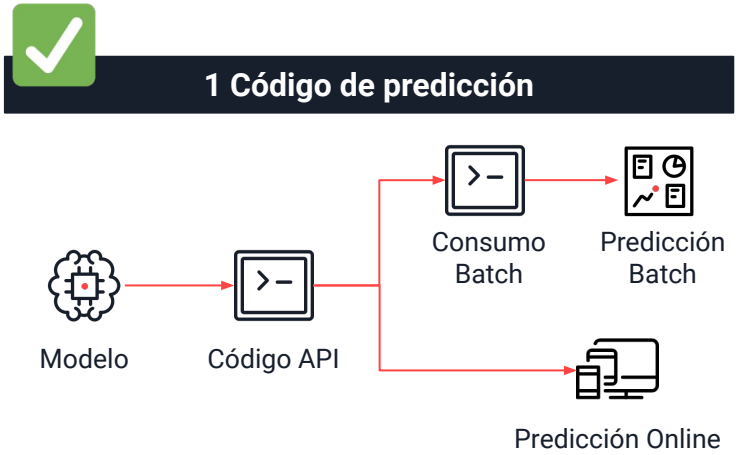
Título presentación proyecto. Título sección.

Servicio de predicción.

Servicio de predicción.

- **Predicción online vs batch:** Predicciones sobre un archivo o predicciones bajo demanda.
- **Unificación del proceso de predicción:** Para que no existan dos lógicas distintas de predicción.





Servicio de predicción.

- **Predicción online vs batch:** Predicciones sobre un archivo o predicciones bajo demanda.
- **Unificación del proceso de predicción:** Para que no existan dos lógicas distintas de predicción.

```python
import argparse
from fastapi import FastAPI
from model_logic import ModelLogic

def run(args):
    model = ModelLogic().load(args.model_path)
    app = FastAPI()

    @app.post("/predict")
    def predict(request: dict):
        predictions = []
        for instance in request["instances"]:
            pred = model.predict(instance)
            predictions.append(pred)

        return {"predictions": predictions}
    return app

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--port', type=int)
    parser.add_argument('--model_path', type=str)
    args = parser.parse_args()

    uvicorn.run(run(args), host='0.0.0.0', port=args.port)
```

03.04

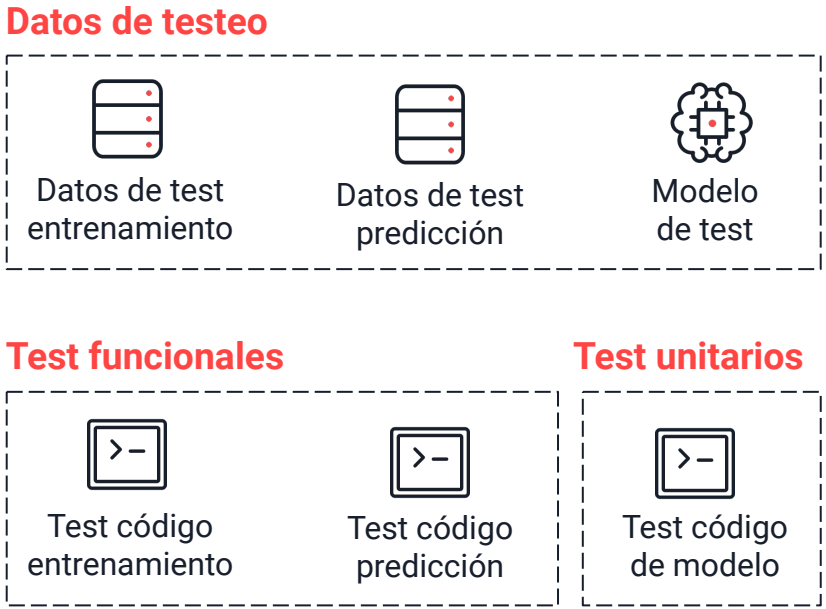
Título presentación proyecto. Título sección.

Testing de código.

Testing de código.

- **Test del modelo:** El modelo de Machine Learning se puede validar por medios de **test unitarios.**

- **Test de comandos:** Los diferentes comandos de ejecución se pueden validar por medios de **test funcionales.**



Testing de código.

- **Test del modelo:** El modelo de Machine Learning se puede validar por medios de **test unitarios.**

- **Test de comandos:** Los diferentes comandos de ejecución se pueden validar por medios de **test funcionales.**

```python
import os, tempfile, pytest
from src.model import ModelLogic


class TestModelLogic():


    def test_fit(self, X, y):
        model = ModelLogic().fit(X,y)


    def test_predict(self, X, y):
        y = ModelLogic().fit(X,y).predict(X)


    def test_save(self,):
        with tempfile.TemporaryDirectory() as tmp:
            ModelLogic().save(f'{tmp}/model.pkl')


    def test_load(self):
        with tempfile.TemporaryDirectory() as tmp:
            ModelLogic().load(f'{tmp}/model.pkl')
```

04.

Título presentación proyecto.

Arquitectura de contenedores.

04.01

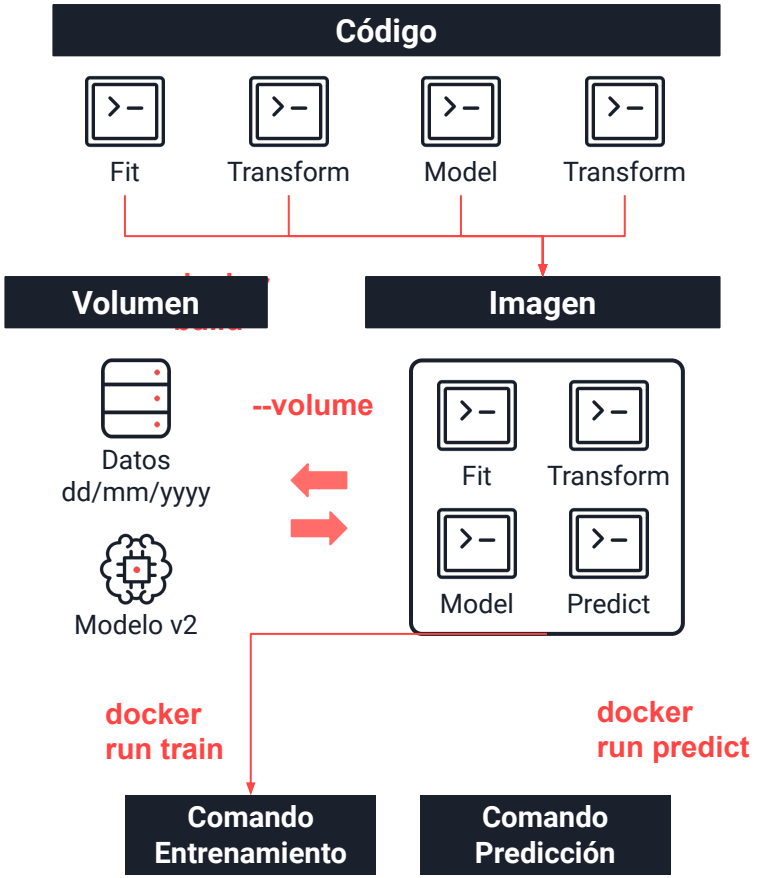
Título presentación proyecto. Título sección.

Docker.

Docker en Machine Learning.

Docker es un sistema de **virtualización** de software ligera. En un proyecto de Machine Learning es útil por lo siguiente:

- **Multiplataforma:** En cualquier SO.
- **Encapsulado:** Software autocontenido.
- **Comandos:** Permite ejecutar comandos.



04.03

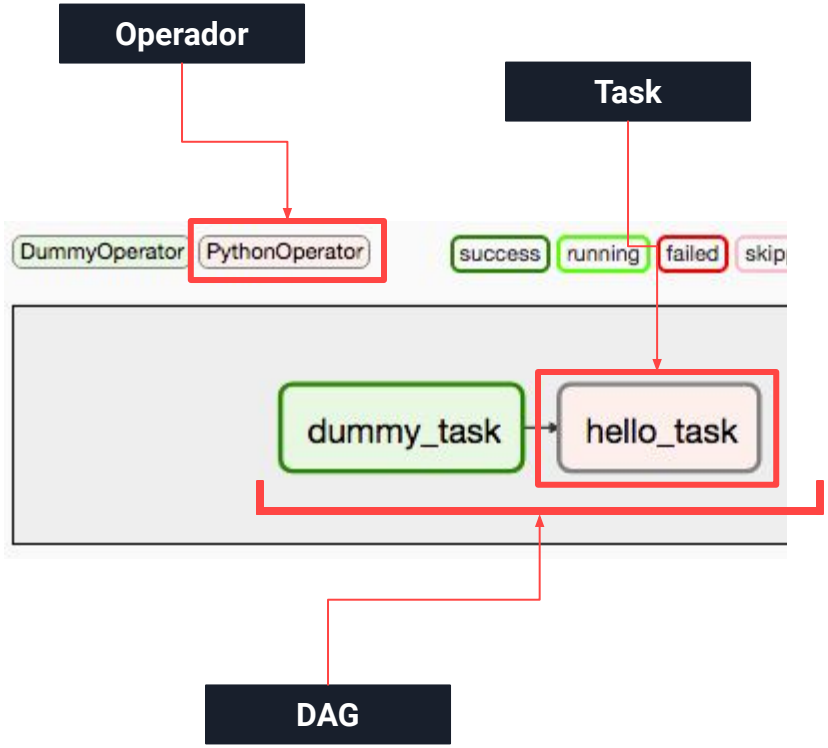
Título presentación proyecto. Título sección.

Airflow.

Airflow.

Airflow es un **orquestador** de procesos. Definir una secuencia de ejecuciones. Sus componentes son:

- **DAG:** Definición del flujo de ejecuciones.
- **Task:** Cada uno de los nodos del DAG.
- **Operator**: Arquetipos de tasks.



Airflow - DockerOperator.

Los componentes que tiene Airflow son los siguientes:

- **DAG:** Definición del flujo de ejecuciones.
- **Task:** Cada uno de los nodos de un DAG.
- **Operator**: Arquetipos de tasks.

DockerOperator: Task que ejecuta un comando Docker.

```
from docker.types import Mount
from plugins.operators import DockerOperator


t1 = DockerOperator(
    task_id="training_step",
    image="{{ params.image_name }}",
    auto_remove=True,
    entrypoint=[ "python", "train.py" ],
    mount_tmp_dir=True,
    mounts=[
        Mount(type="bind",
            source = input_dir
            target='/input/'),
        Mount(type="bind",
            source = target_dir
            target="/output/')
    ],
    environment={
        "LR" : "{{ params.param_lr }}",
        "DATA_PATH" : "/input/{{ params.dataset  }}",
        "MODEL_PATH" : "/output/{{ params.model }}"
    },
)
```

05.

Título presentación proyecto.

Desarrollo de Pipelines.

05.01

¿Qué es un Pipeline?

Pipelines de ML.

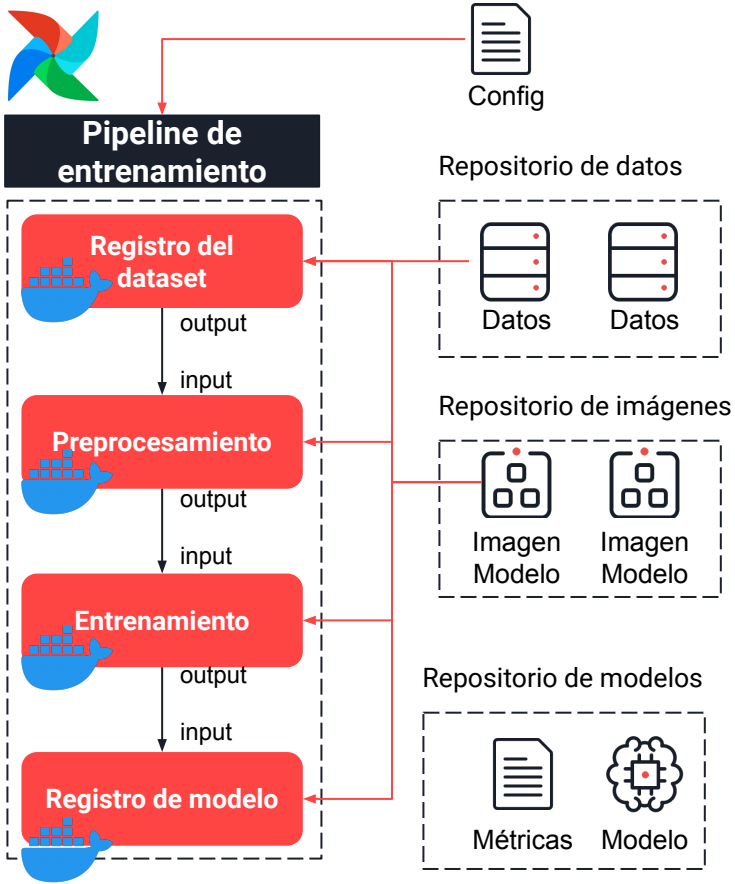
Los pipelines de sirven para definir las secuencia de procesos que componen una aplicación de Machine Learning:

- **Pipeline de entrenamiento**
  - Paso 1: Registro del Dataset.
  - Paso 2: Preprocesamiento.
  - Paso 3: Entrenamiento.
  - Paso 4: Registro de modelo.

Pipelines de ML con Airflow.

Los pipelines se pueden programar de muchas maneras. Una de ellas es por medio de DAGs de Airflow. La ventaja que ofrecen es:

- **Configuración:** Parametrización.
- **Experimentos:** Metadatos
- **Métricas:** Almacenar resultados



Pipelines de ML con Airflow.

Los pipelines se pueden programar de muchas maneras. Una de ellas es por medio de DAGs de Airflow. La ventaja que ofrecen es:

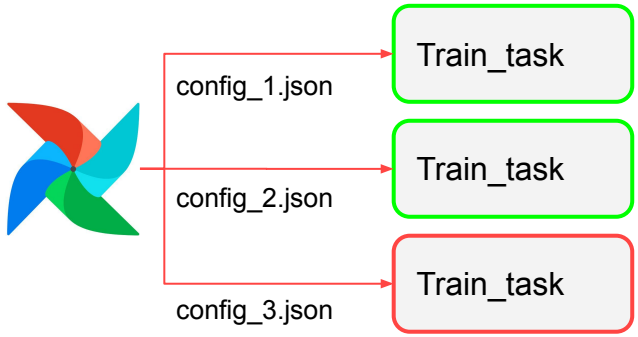
- **Configuración:** Parametrización.
- **Experimentos:** Metadatos
- **Métricas:** Almacenar resultados

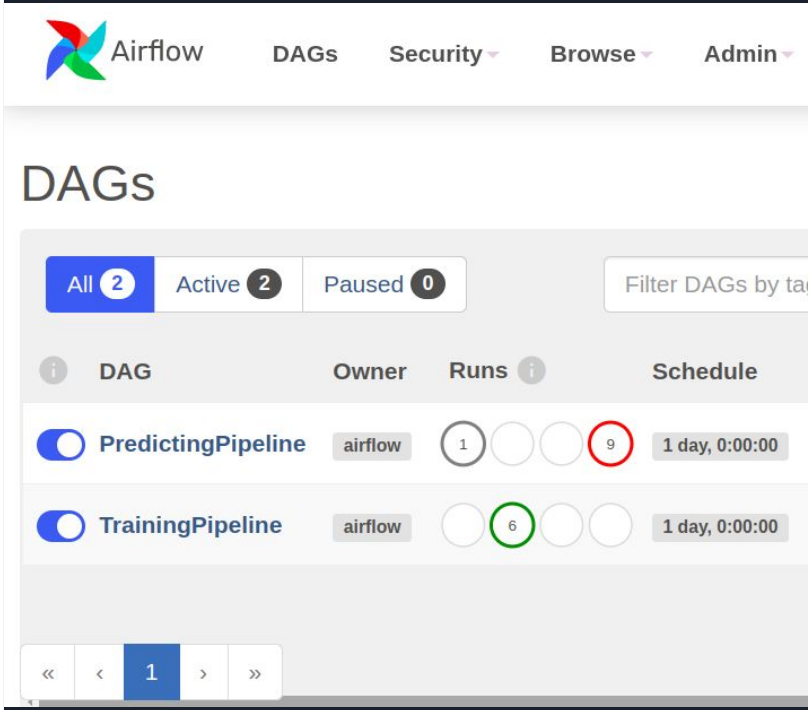
```
from airflow import DAG
from docker.types import Mount
from plugins.operators import RegisterModel
from plugins.operators import DockerOperator


params = { · · · }

with DAG( 'TrainingPipeline', params=params) as dag:

  t1 = DockerOperator(
    task_id="preprocess_step",
    entrypoint[ "python", "tranform.py" ],
    · · · )

  t2 = DockerOperator(
    task_id="training_step",
    entrypoint[ "python", "train.py" ],
    · · · )

  t3 = RegisterModel(
    task_id="register_step",
    · · · )

  t1 >> t2 >> t3
```

Experimento.

Se denomina experimento a cada una de las ejecuciones que se realizan, a la hora de registrar resultados y métricas.

Con Airflow podemos ejecutar múltiples experimentos simultáneamente y ver el estado de cada uno de ellos en cada momento.

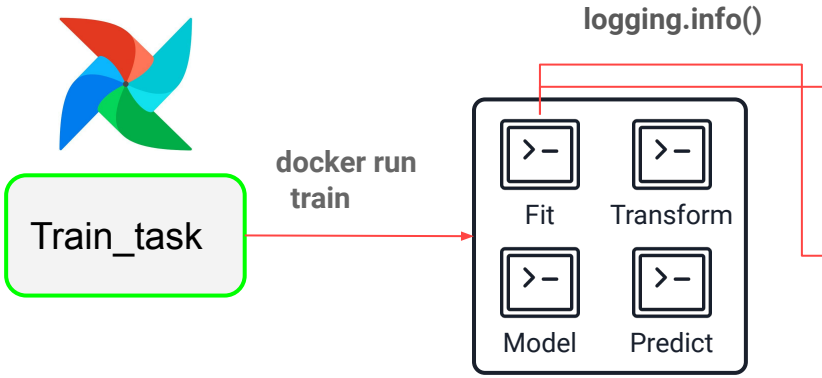




Paradigma

Monitorización.

Podemos ver los logs de cada ejecución de cada proceso docker por medio de los logs de Airflow asociados al task donde se ejecutó el proceso.



Airflow DAGs Security Browse Admin

```
[2023-04-03, 18:39:58 UTC] {taskinstance.py:1431} INFO - Expor
AIRFLOW_CTX_DAG_OWNER=***
AIRFLOW_CTX_DAG_ID=TrainingPipeline
AIRFLOW_CTX_TASK_ID=training_step
AIRFLOW_CTX_EXECUTION_DATE=2023-04-02T18:39:48.829574+00:00
AIRFLOW_CTX_DAG_RUN_ID=scheduled__2023-04-02T18:39:48.829574+0
[2023-04-03, 18:39:58 UTC] {docker.py:236} INFO - Starting doc
[2023-04-03, 18:39:58 UTC] {docker.py:247} WARNING - Using re
[2023-04-03, 18:40:01 UTC] {docker_operator_with_exposed_ports

INFO: Lectura del dataset.
Features: ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm',
Rows: 120.\n"

2023-04-03, 18:40:01 UTC] {docker_operator_with_exposed_ports

INFO: Metricas de entrenamiento.
Accuracy: 0.95.'

2023-04-03, 18:40:01 UTC] {taskinstance.py:1282} INFO - Markin
[2023-04-03, 18:40:01 UTC] {local_task_job.py:154} INFO - Tas
[2023-04-03, 18:40:01 UTC] {local_task_job.py:264} INFO - 1 d
```

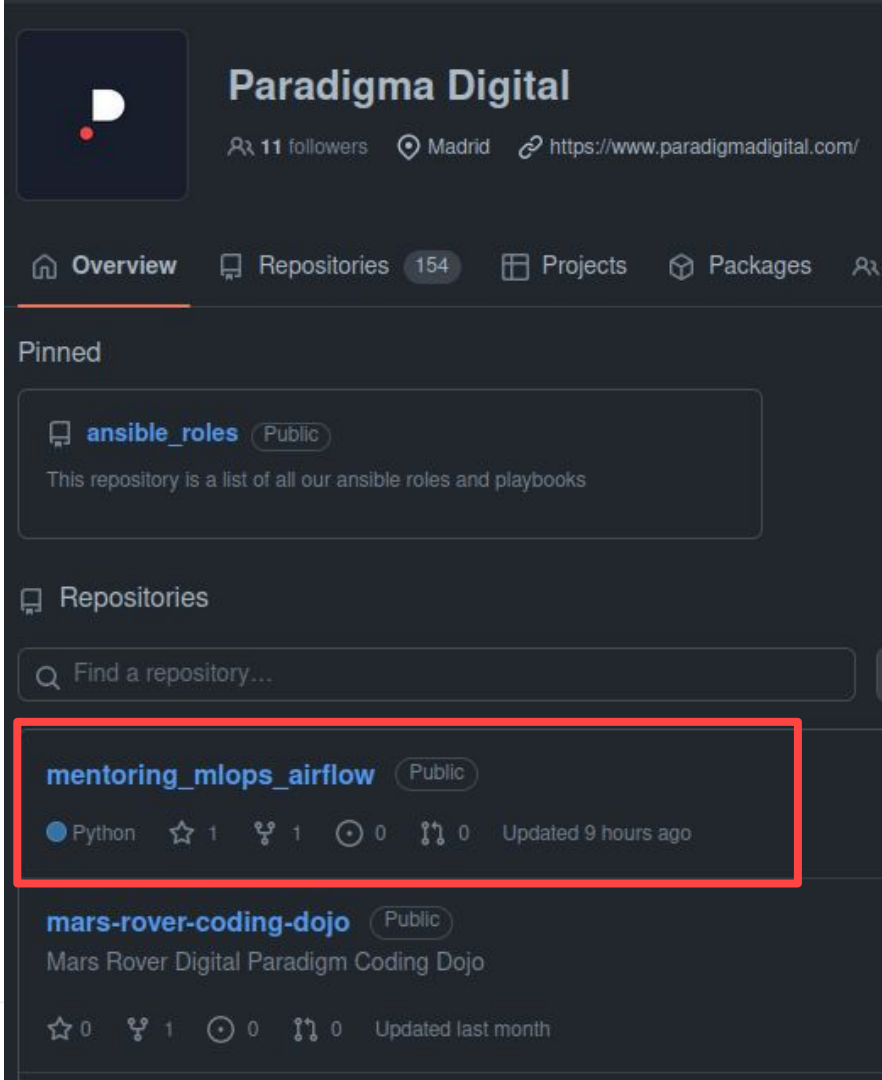
Título presentación proyecto.

El código es de todos.

Código en el Github de Paradigma.

Todo el código que hemos enseñado hoy y mucho más está en el Github de Paradigma!!

Repositorio: mentoring_mlops_airflow



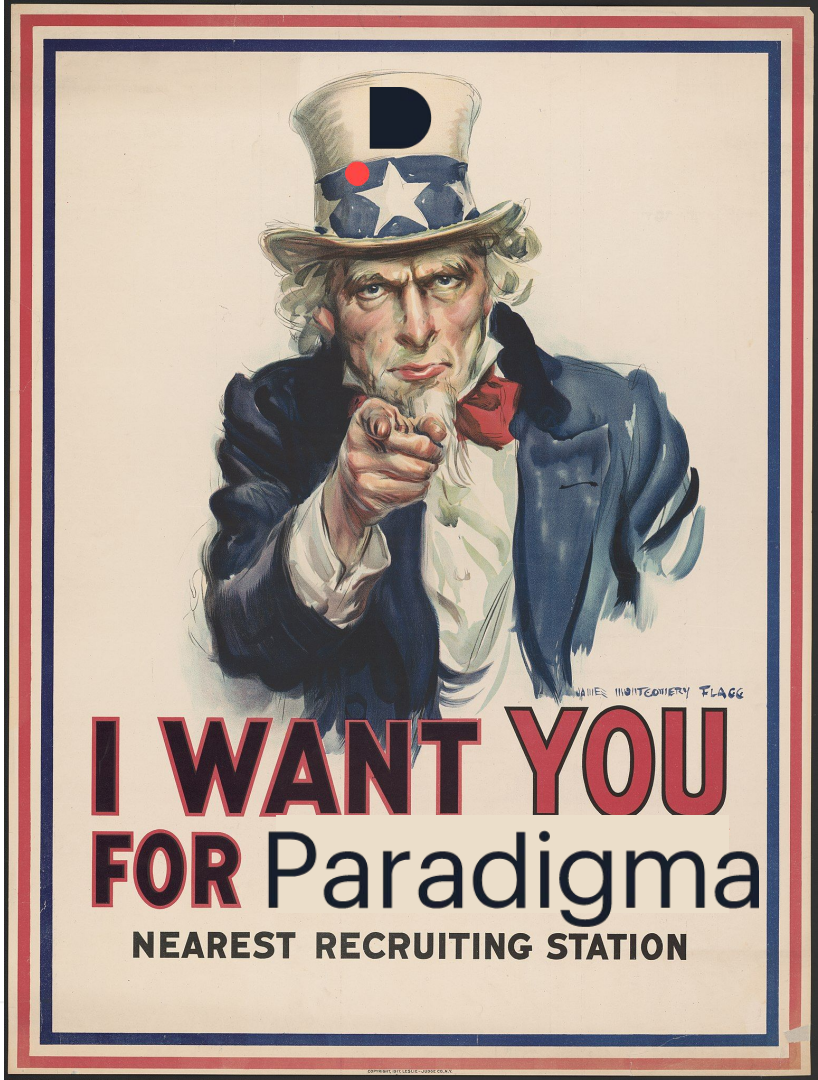
Título presentación proyecto.

¡Hasta la victoria camaradas!

We are hiring.

Únete a la revolución!!!

paradigmadigital.com/empleo/



Gracias!

Preguntas?