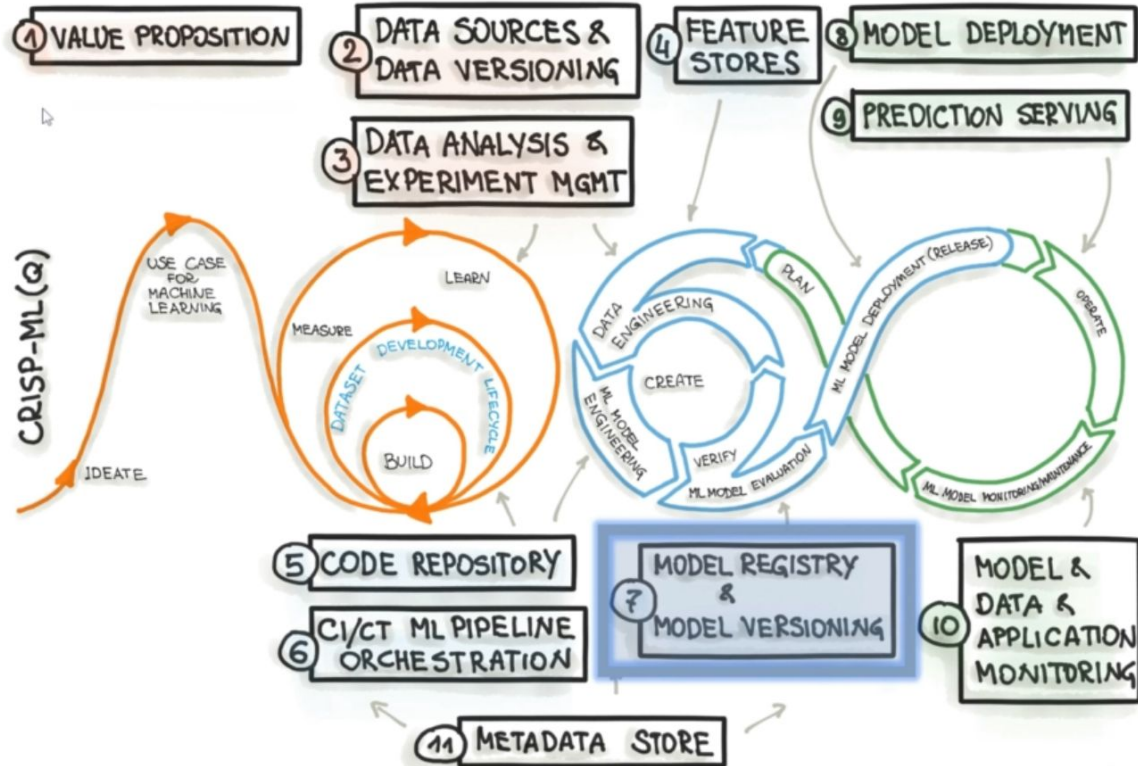




Registro y Versionado de Modelos

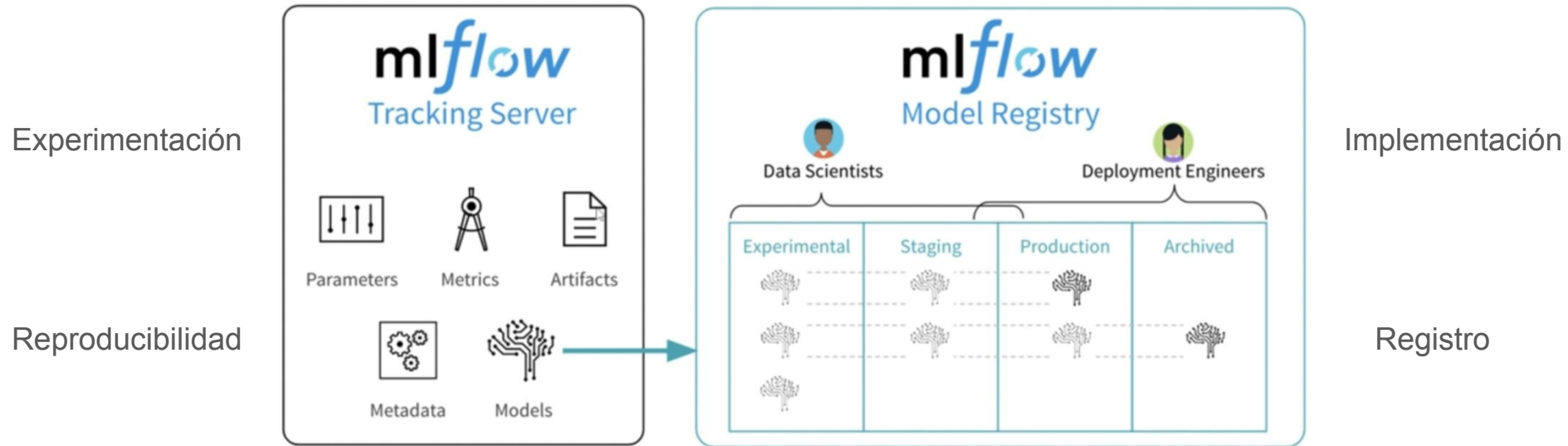
Diego Mosquera Uzcátegui
Abril, 2025

Etapas de MLOps





Herramientas para MLOps

Plataforma de código abierto para administrar el ciclo de vida de un modelo de ML



Registro y versionado con MLFlow

<https://mlflow.org/docs/latest/tutorials-and-examples/index.html>

 TRACKING Record and query experiments: code, data, config, and results.	 PROJECTS Package data science code in a format that enables reproducible runs on many platforms	 MODEL REGISTRY Store, annotate, and manage models in a central repository	 MODELS Deploy machine learning models in diverse serving environments
---	---	---	---

Versión actual del ecosistema de MLFlow

Componente	Descripción Actualizada
Tracking	Registro completo de runs: código, métricas, parámetros, artefactos, inputs/outputs, y ahora también prompts y respuestas de LLMs.
Projects	Ejecución reproducible de proyectos ML en diferentes entornos. Se integra con pipelines.
Model Registry	Versión centralizada de modelos con estados, metadata y transición entre entornos.
Models	Estándar de empaquetado, despliegue local y en la nube. Compatible con múltiples frameworks.
Pipelines (nuevo)	Plantillas de entrenamiento y despliegue con validaciones y automatización.
LLM Tracking (nuevo)	Observabilidad para modelos generativos y APIs LLM.

Registro de un modelo Scikit-learn y MLFlow

En google colab

Registro de un modelo Pycaret y MLFlow

En google colab

Versionado con DVC

(Data Version Control)

Data Version Control

- Herramienta open-source que extiende Git para gestionar versiones de datos, modelos y pipelines de machine learning.
- Desarrollada en Python.
- ¿Para qué sirve?
 - Rastrear **cambios en conjuntos de datos y artefactos** de ML (modelos, métricas, configuraciones).
 - Definir y ejecutar **pipelines reproducibles** (dvc repro).
 - Compartir datos y resultados sin sobrecargar el repositorio Git (usando “**remotos**” de **almacenamiento**).
- Complementa a Git para los proyectos de Ciencia de Datos e IA.

¿Por qué no usar solo Git directamente?

Git está optimizado para texto (código), no
para datos y modelos de ML (binarios)

Características principales

- Compatible con Git
- Control de Versiones de Datos simple
- Independiente de almacenamiento
- Reproducible
- Independiente del lenguaje y marco de desarrollo
- Ramificación de baja fricción
- Fácil de utilizar

Principales comandos DVC

```
$ dvc init # Inicializa el repositorio
$ dvc add . # Agrega los archivos que han sido cambiados
$ git commit -m "cambios realizados" # Commit de las actualizaciones con un mensaje
$ dvc remote add newremote s3://bucket/path # Apunta el repositorio a un bucket de S3
$ dvc push # Envía los cambios al repositorio DVC alojado en el bucket de S3 por defecto
$ dvc pull # Extrae el último cambio del repositorio DVC alojado en el bucket de S3 por defecto
```

Seguimiento de archivos con DVC

Los archivos y carpetas con datos grandes van al almacenamiento remoto de DVC. Los archivos .dvc pequeños van a Github.

```
$ dvc add data/raw/train
```

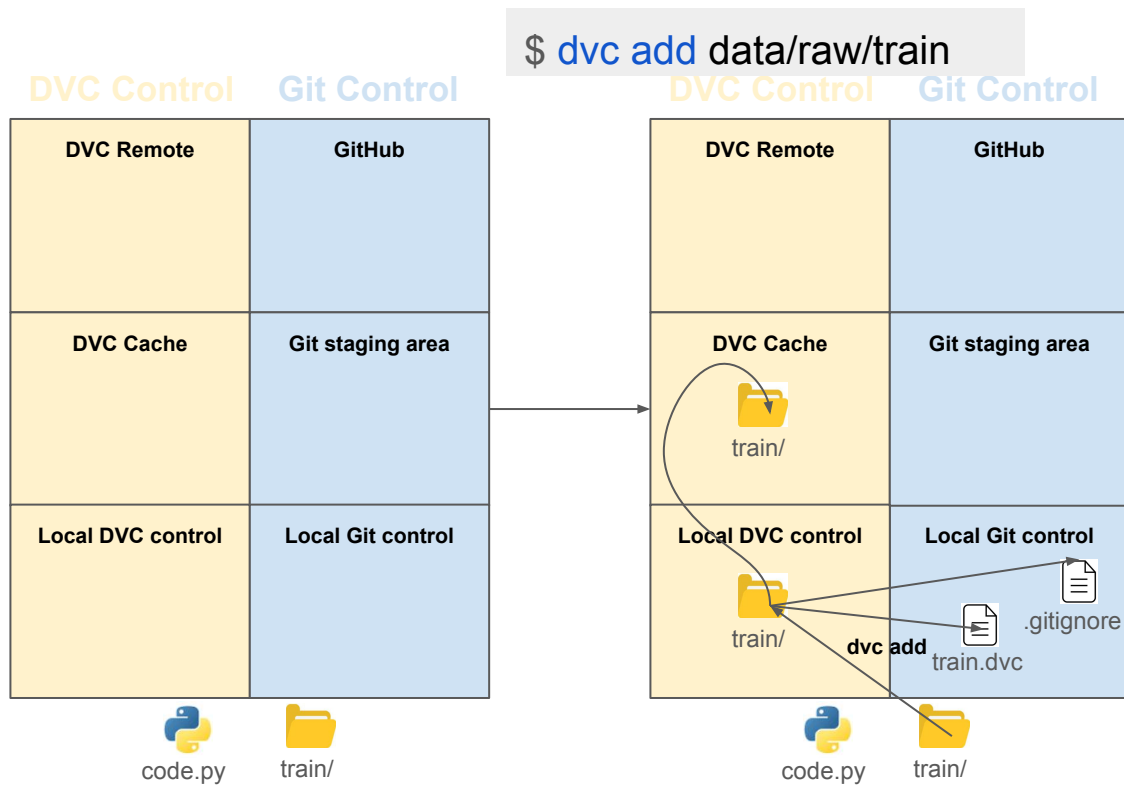
DVC Control Git Control

DVC Remote	GitHub
DVC Cache	Git staging area
Local DVC control	Local Git control



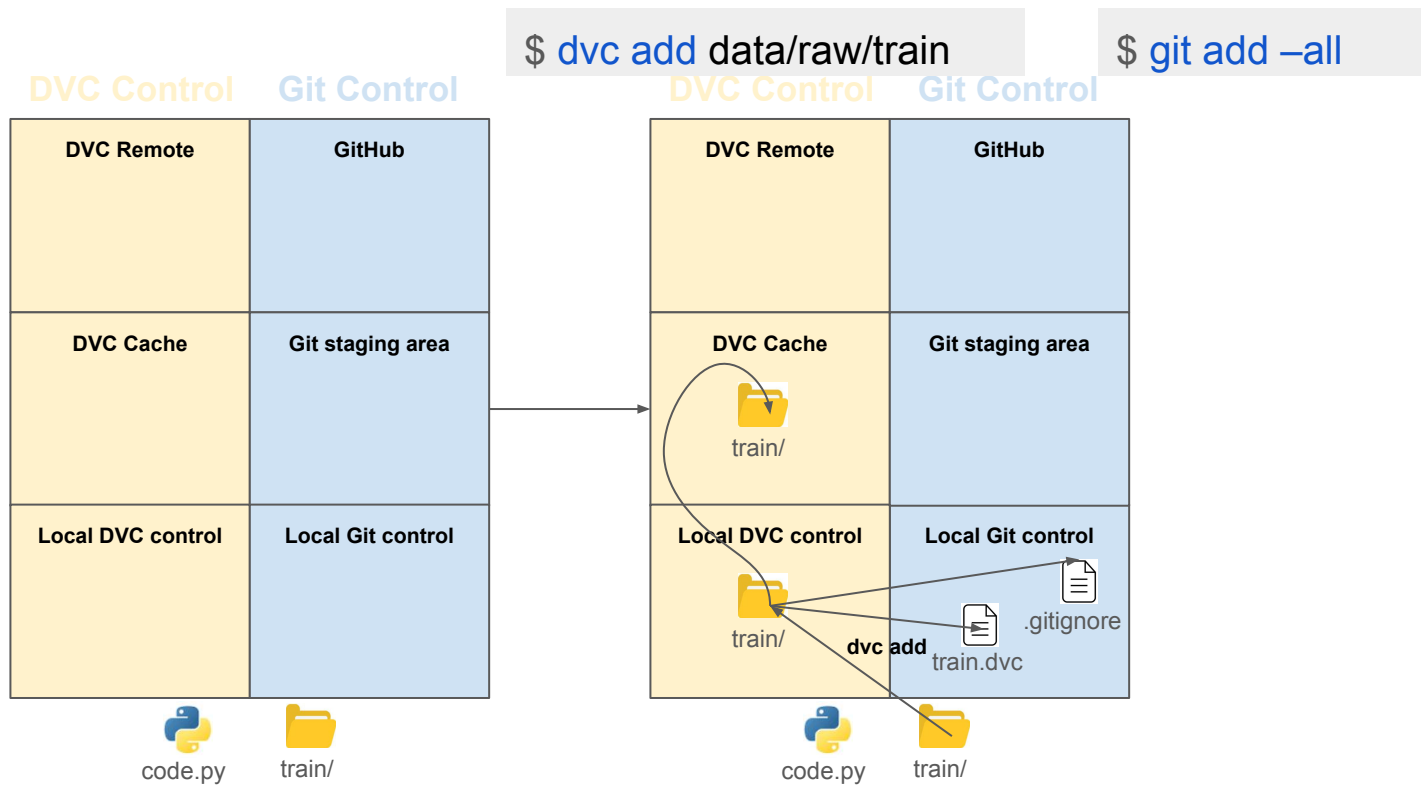
Seguimiento de archivos con DVC

Los archivos y carpetas con datos grandes van al almacenamiento remoto de DVC. Los archivos .dvc pequeños van a Github.



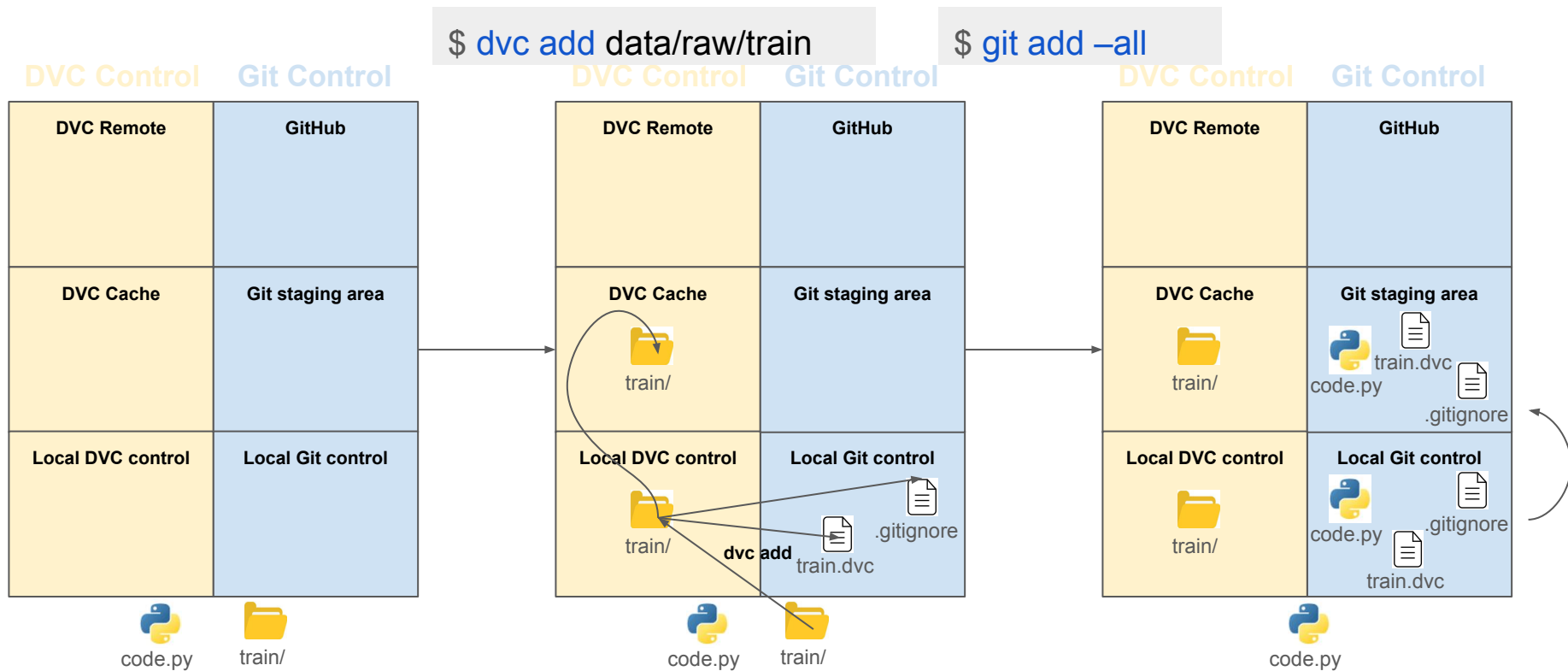
Seguimiento de archivos con DVC

Los archivos y carpetas con datos grandes van al almacenamiento remoto de DVC. Los archivos .dvc pequeños van a Github.



Seguimiento de archivos con DVC

Los archivos y carpetas con datos grandes van al almacenamiento remoto de DVC. Los archivos .dvc pequeños van a Github.

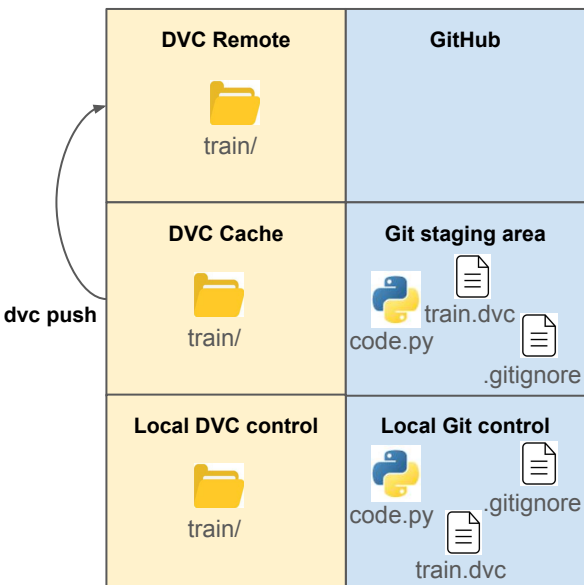


Seguimiento de archivos con DVC

Los archivos y carpetas con datos grandes van al almacenamiento remoto de DVC. Los archivos .dvc pequeños van a Github.

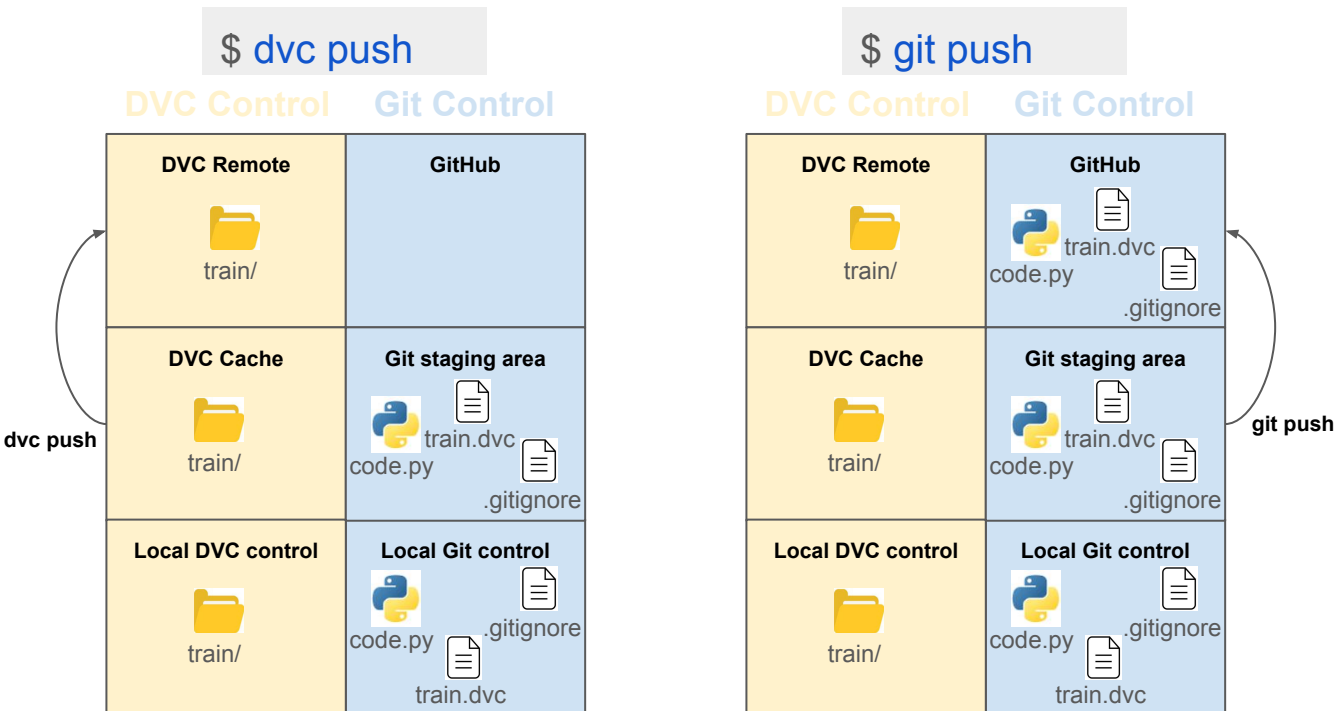
```
$ dvc push
```

DVC Control Git Control



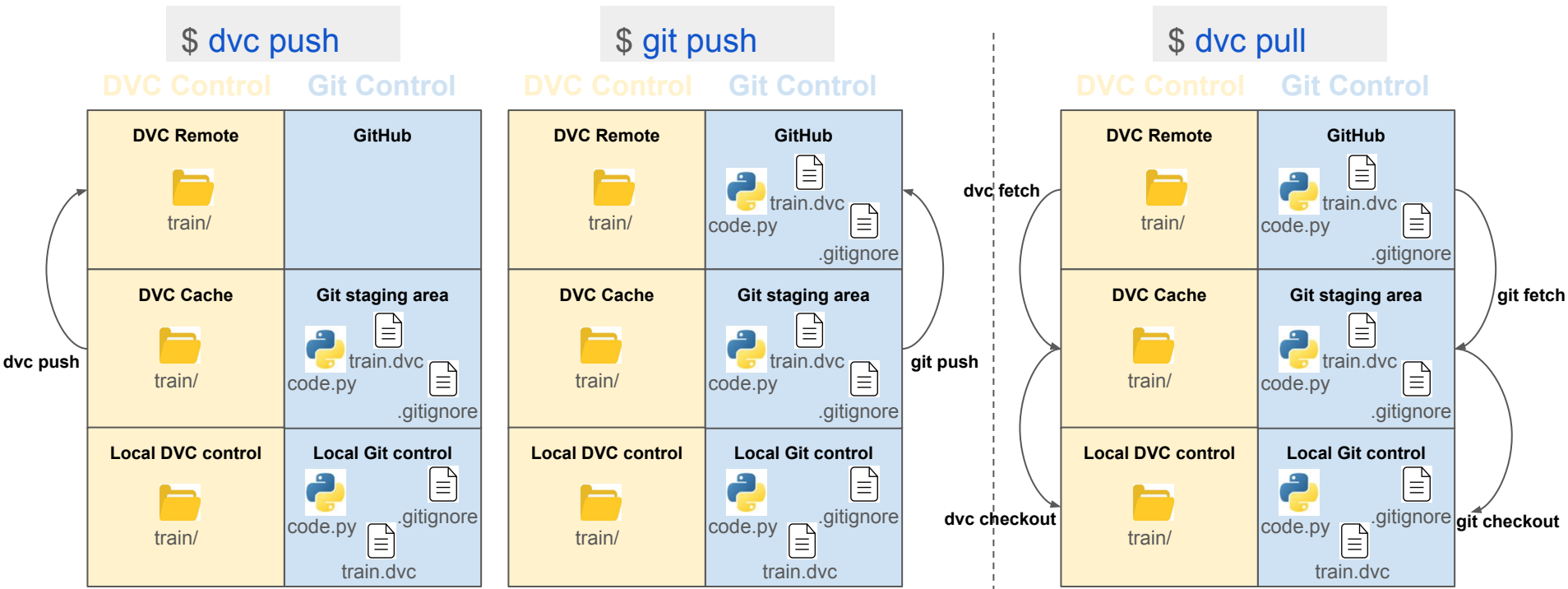
Seguimiento de archivos con DVC

Los archivos y carpetas con datos grandes van al almacenamiento remoto de DVC. Los archivos .dvc pequeños van a Github.



Seguimiento de archivos con DVC

Los archivos y carpetas con datos grandes van al almacenamiento remoto de DVC. Los archivos .dvc pequeños van a Github.



Archivos .dvc

Los archivos DVC son archivos YAML. La información se almacena en pares clave-valor y listas. La primera es md5 seguida de una cadena de caracteres.

```
yaml
```

```
# YAML
```

```
md5: 62bdac455a6574ed68a1744da1505745
```

```
outs:
```

```
- md5: 96652bd680f9b8bd7c223488ac97f151
```

```
  path: model.joblib
```

```
  cache: true
```

```
  metric: false
```

```
  persist: false
```

Uso de DVC

```
$ dvc init # Inicializa el repositorio
$ dvc add . # Agrega los archivos que han sido cambiados
$ git commit -m "cambios realizados" # Commit de las actualizaciones con un mensaje
$ dvc remote add newremote s3://bucket/path # Apunta el repositorio a un bucket de S3
$ dvc push # Envía los cambios al repositorio DVC alojado en el bucket de S3 por defecto
$ dvc pull # Extrae el último cambio del repositorio DVC alojado en el bucket de S3 por defecto
```

Pasos para probar DVC

1. Preparación del remote

- a. Crear una carpeta en google drive
- b. Crear una cuenta de servicio de google
 - i. Ir a la consola de google
 - ii. Ir a **IAM & Admin** → **Cuentas de servicio** y crea una **Service Account**
 - iii. En la pestaña de la cuenta, ir a **Claves** → **Crear clave**, elige JSON (por ejemplo `sa-credentials.json`).
- c. Compartir la carpeta de Drive con el e-mail de la service account con permisos de "Editor").

2. Preparación del local

- a. Crear el directorio del repositorio local
- b. Instalar el paquete de `dvc[gdrive]`
- c. Inicializar el repositorio y agregar el archivo de datos (por ejemplo, `.csv`)
- d. Configurar el repositorio remoto asociado
- e. Habilitar el uso de Service Account
- f. Editar el archivo de configuración y agregar el `gdrive_service_account_json_file_path =`
- g. Hacer el commit del archivo de configuración del repositorio remoto
- h. Hacer el push al repositorio remoto

Crear el directorio del repositorio local

Esta carpeta será utilizada como ubicación de los archivos del proyecto de ciencia de datos en el repositorio local. Por ejemplo:

C:\dvc-test

Instalar el paquete de dvc[gdrive]

```
pip install "dvc[gdrive]"
```


Inicializar el repositorio

Inicializar y hacer un primer commit

```
git init  
dvc init  
git commit -m "Iniciando DVC"
```

crear una carpeta /data dentro del directorio raíz del repositorio local

```
dvc add data # Esto genera el data.dvc (archivo YAML)  
git add data.dvc  
git commit -m "Agregando data.dvc"
```

Agregar un archivo de datos (por ejemplo .csv) a la subcarpeta /data

Configurar el repositorio remoto asociado

```
dvc remote add -d gdrive-remote gdrive://<FOLDER_ID>
```

Habilitar el uso de Service Account

```
dvc remote modify gdrive-remote gdrive_use_service_account true
```

Editar el archivo de configuración

Agregar

```
gdrive_service_account_json_file_path = ...
```

Hacer el commit del archivo de configuración

```
git commit .dvc/config -m "Configuracion del storage remoto"
```

Hacer el push al repositorio remoto

```
dvc push
```

Gestionar cambios en los datos

```
dvc add data
```

```
git commit data.dvc -m "Actualización de los datos"
```

```
dvc push
```

```
git push
```

Uso del repositorio remoto

```
dvc pull
```