



CI/CD in the ML era

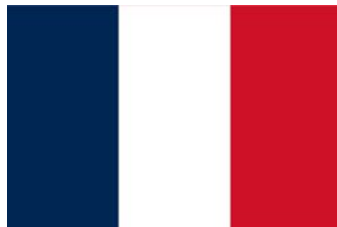
05/04/2024

Who am I?



Emmanuel-Lin
TOULEMONDE

About 10 years in Data Science, MLEng, MLOps



<https://eltoulemonde.fr/>

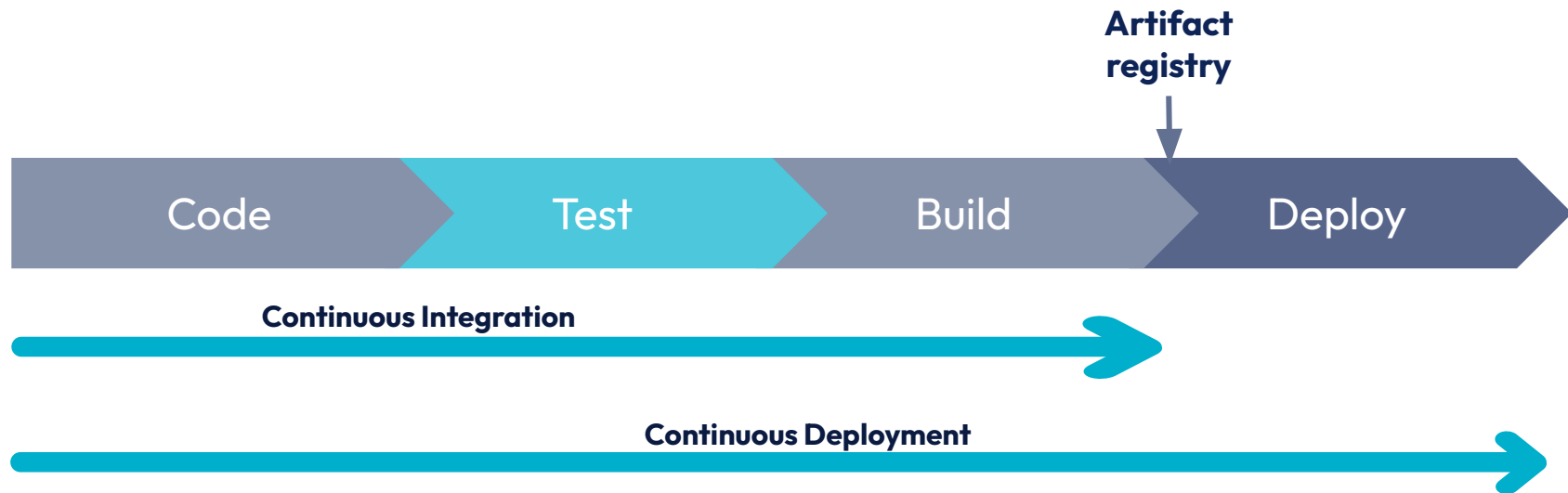
A topic extracted from Culture MLOps

CI/CD in the ML era

My intention is to show different patterns of what can be done and take a step back on what is done.



Continuous *



Example of CI/CD code

With GitHub Actions

```
1  name : validate code
2  on : push
3  jobs :
4    build:
5      runs-on: ubuntu-latest
6      steps:
7        - uses: actions/checkout@v4
8        - name: Setup Python
9          uses: actions/setup-python@v5
10         with:
11           python-version: '3.10'
12        - name : install dependencies
13          run : pip install -r requirements.txt
14        - name : install dev dependencies
15          run : python -m pytest tests
16
```



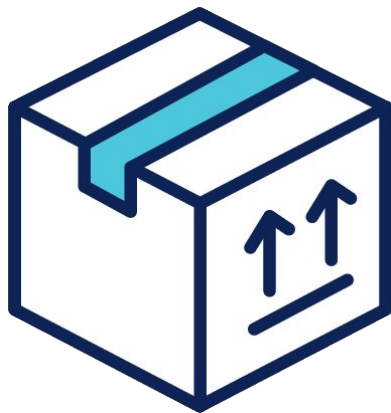
What CI / CD can do for us

*non-exhaustive list

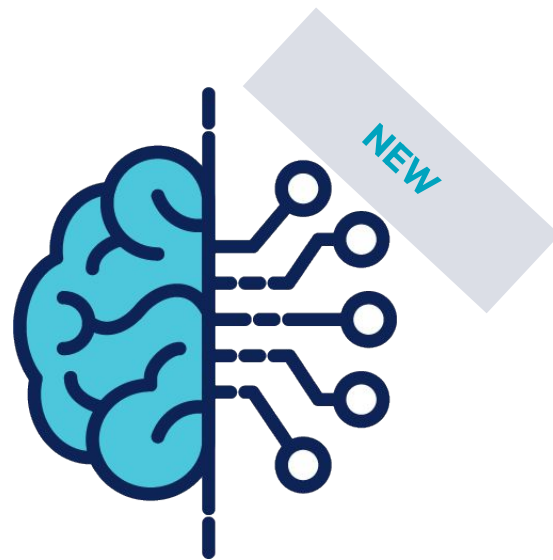


Why is it different in an ML context?

A new artifact to build and deploy



Packaged code



Model

Why is it different in an ML context?

A new event may justify the deployment of a new artefact

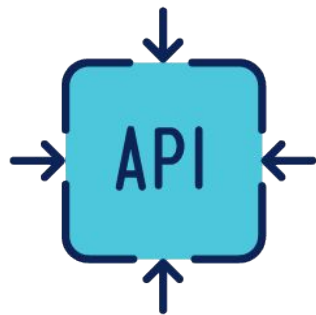


Code change



Data change

3 ways to use a model artifact

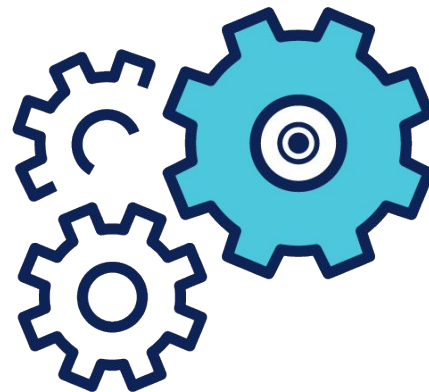


Use a prediction API



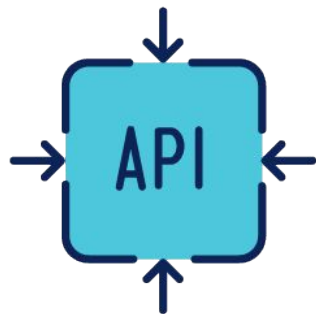
HUGGING FACE

Download an existing model
And run it on our infrastructure



Train or fine-tune a model

Artifact construction and management in these cases?

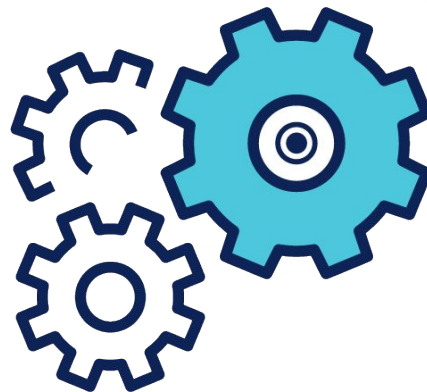


Like a software without ML



HUGGING FACE

Like a library



The topic of this talk

Our focus

1. Where to train models?
2. Where to store model versions?
3. When to load the desired model version?

All this through three examples

02

Making a prototype or demonstrator

Our needs

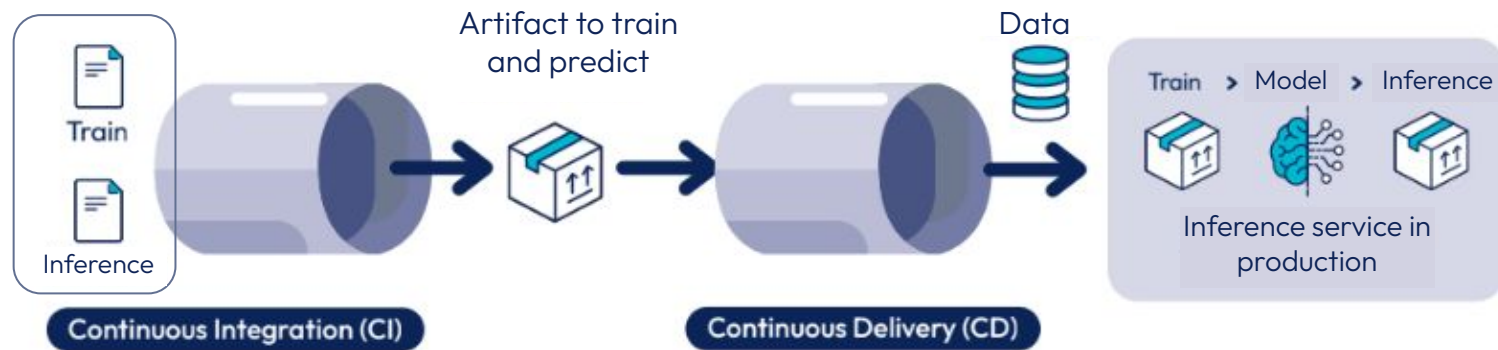
I want to **show** marketing teams that ML can help them.

The use case: show that I can predict **appeal for a product**.

My challenges:
Go fast - Be frugal

Where to train?

Inside inference service

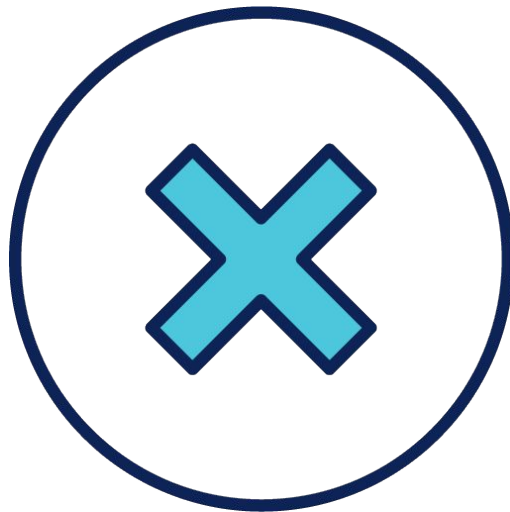


Code validation +
build artifact

CI/CD pipeline

Where can I store versions of the model?

In memory, during inference



Model is not persisted

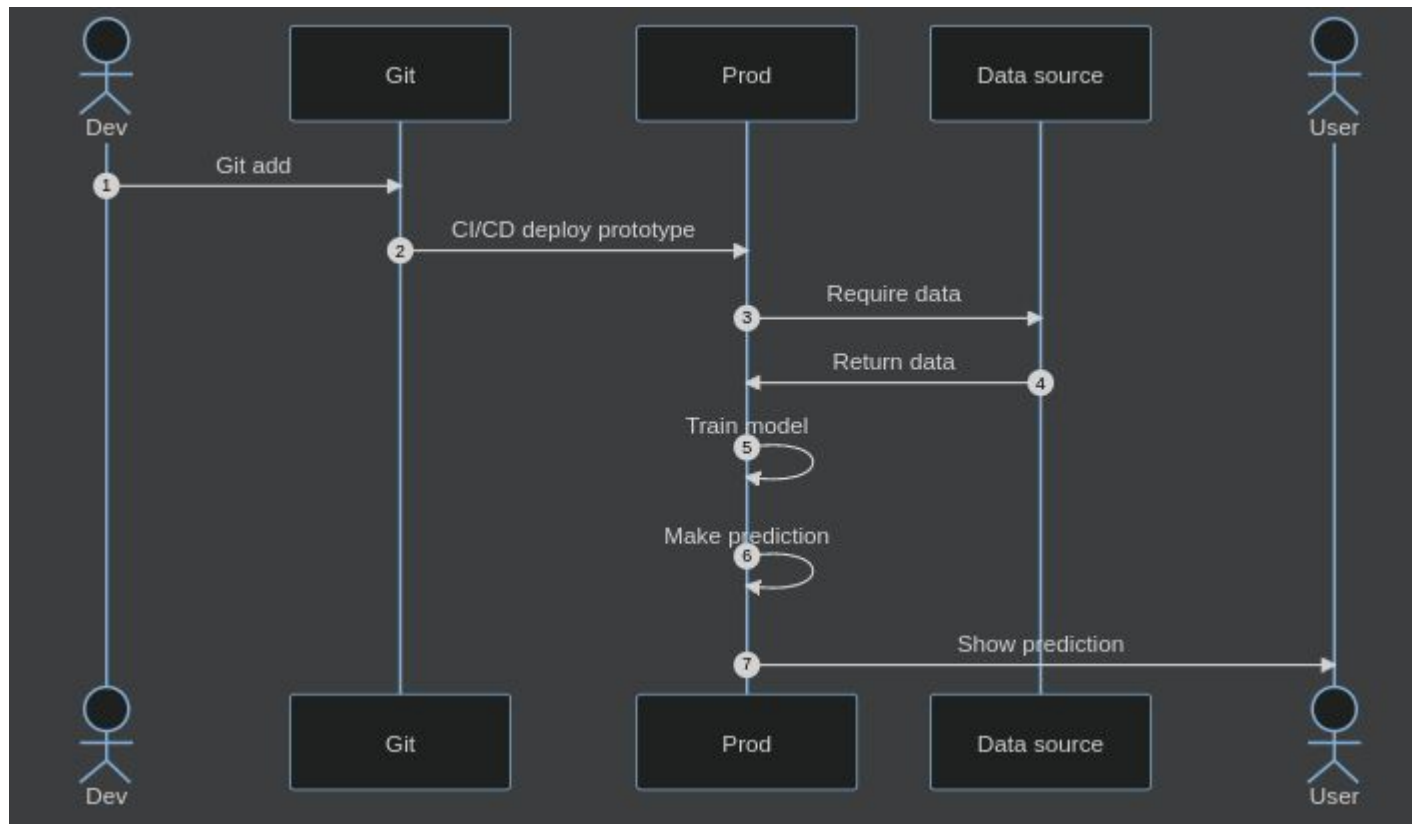
When to load model version?

During inference, in-memory

```
1  @app.get("/predict")
2  def predict(X):
3      model = Model()
4      X_train, y_train = get_train_data()
5      model.fit(X_train, y_train)
6      return model.predict(X)
```

Python code, training model before performing inference

Sequence diagram to get a prediction



03

Starting to develop a product

Our needs

I want to build a **product** in an **iterative way**.

The use case: I want to **serve** my first models to my **users**.

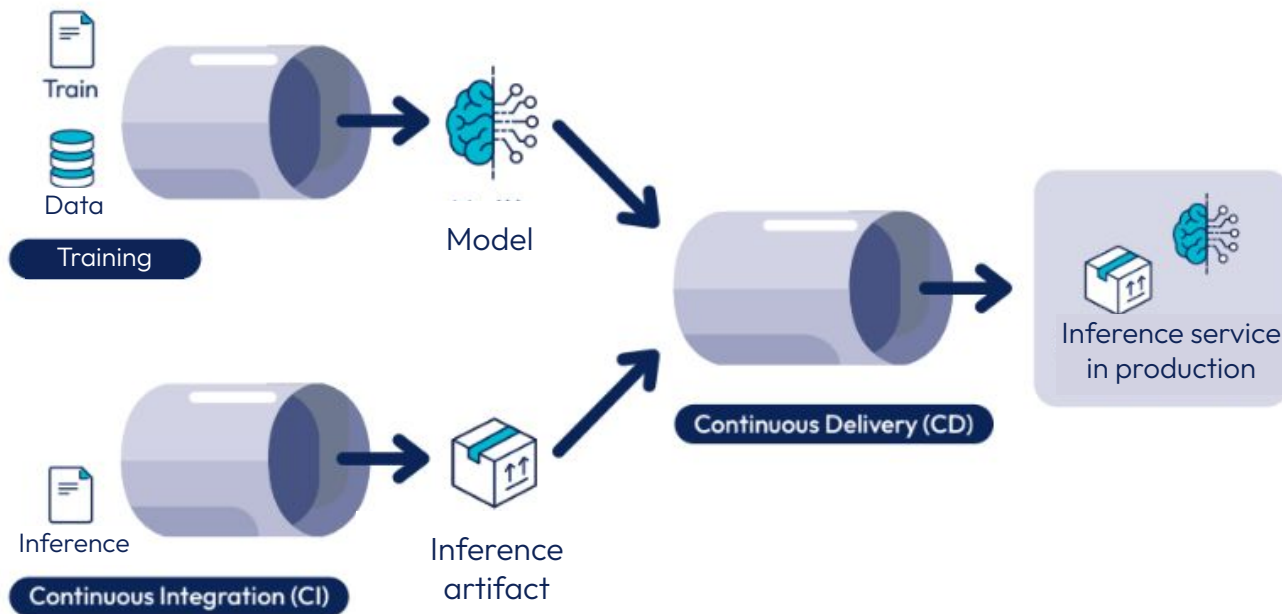
My challenges:

Start to **measure** the product value.

Being able to **pivot quickly** based on **first users feedbacks**.

Where to train?

In developer environment (like code)



CI/CD pipeline

Where to store model versions?

Git

```
~/my-project$ git add .  
~/my-project$ git commit -m "New model version"  
model version  
0 deletions(-)  
pickle  
~/my-project$ git push
```

Git command to save a new version of model

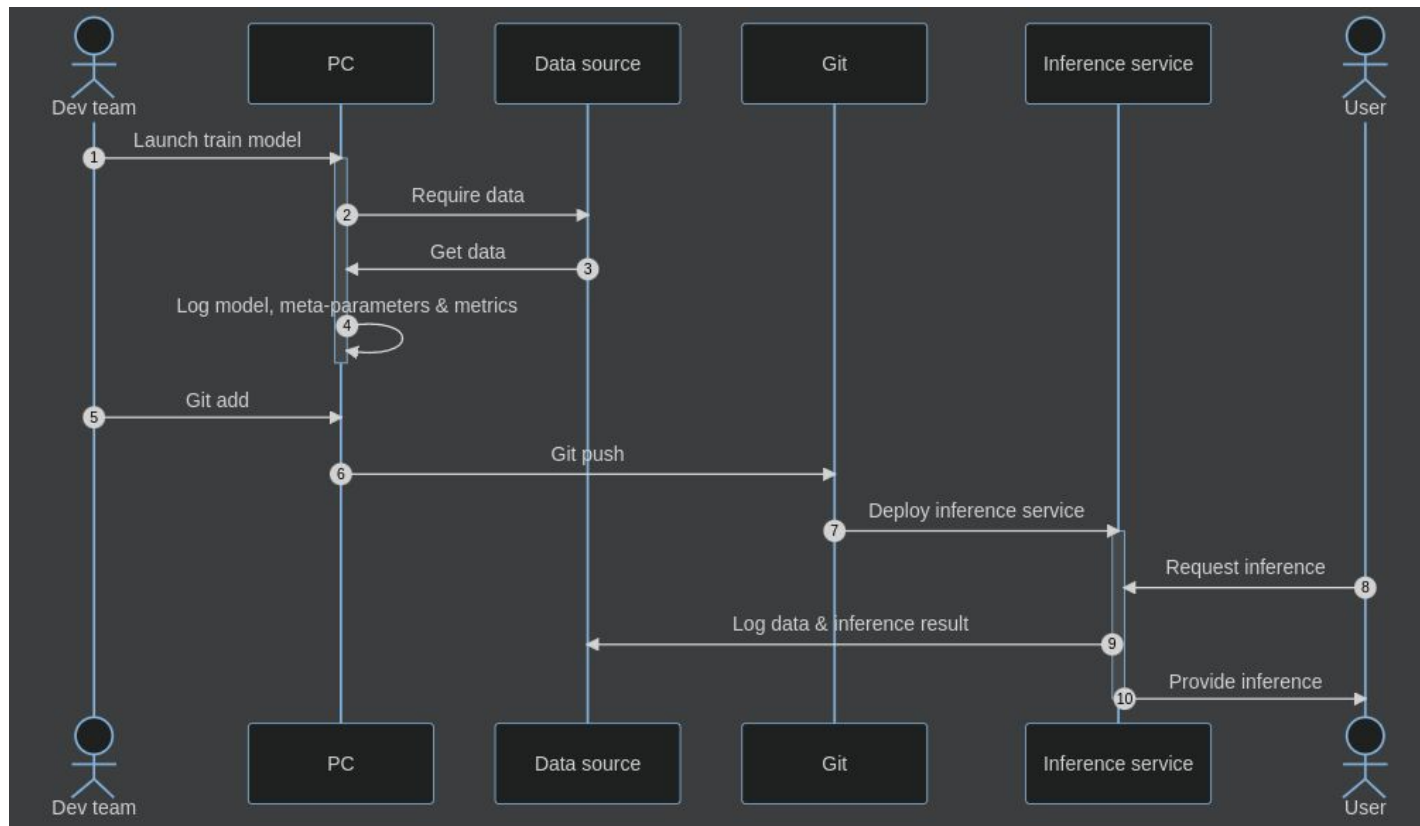
When to load the desired model version?

During deployment, with code artifact

```
1 FROM python:3.11-slim
2 COPY . /source
3 COPY model.joblib /model
4 WORKDIR /source
5 RUN pip install -r requirements.txt
6 CMD streamlit run main.py
```

Dockerfile copying model and code is the same artefact

Sequence diagram to get an inference



04

Scaling

Our needs

I want to **scale**.

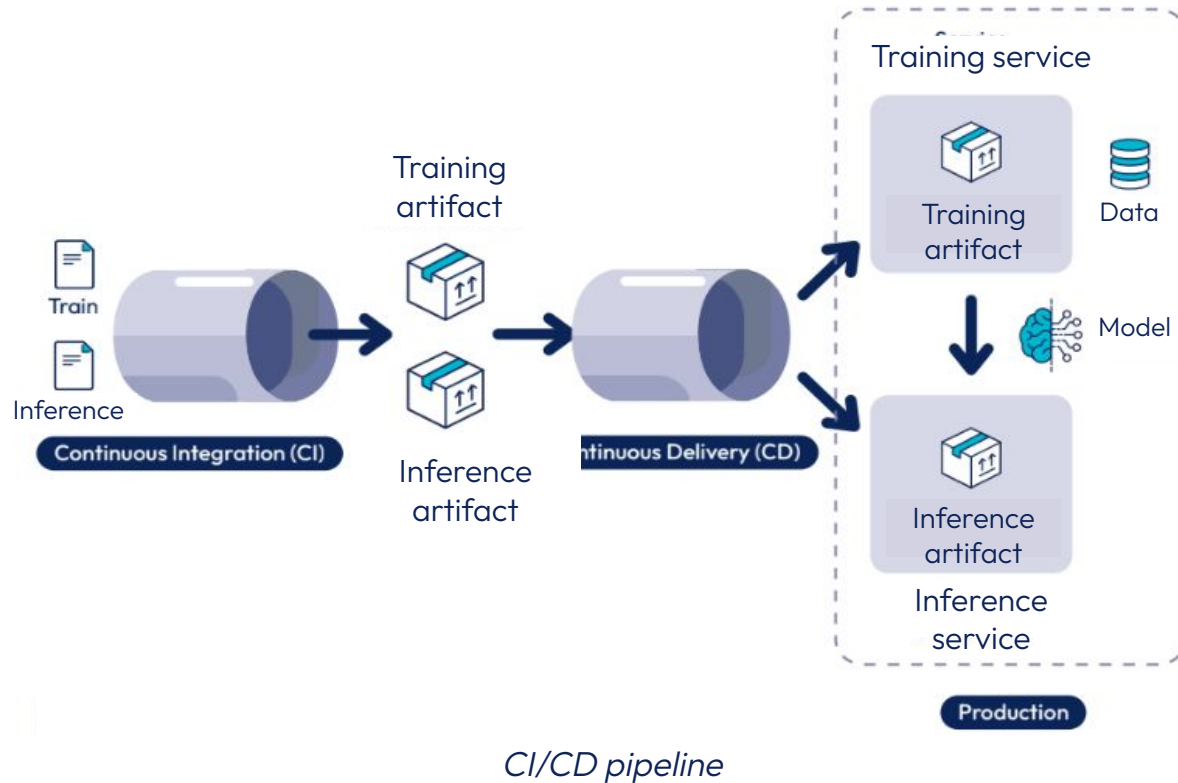
The use case : serving **lots of users**, I want to be able to **test and update quickly** models.

My challenges :

Have **a robust** production - acting in case of **drift** - keep on deploying **on demand**.

Where to train?

In a production service



Where to store model versions?

Blob, S3, etc.

```
1  def save_model(model, model_name):
2      with open("model.pickle") as f:
3          pickle.dump(f, model)
4      container_client = ContainerClient(
5          account_url="https://mydatalake.blob.core.windows.net/" ,
6          container_name="models",
7          credential="secret"
8      )
9      file_name = f"models/{model_name}.pickle"
10     blob_client = container_client.get_blob_client(file_name)
11     with open('model.pickle', 'rb') as f:
12         blob_client.upload_blob(f)
```

*Code of training service,
Saves produced model into a blob*

Where to store model versions?

Alternatives...

**In the same registry as code
artifact**



... but represent a weird artifact flow with a service that write onto registry instead of CI

**In a specialized model
registry**



... but it requires more pros to decide to have a new service

When to change model version?

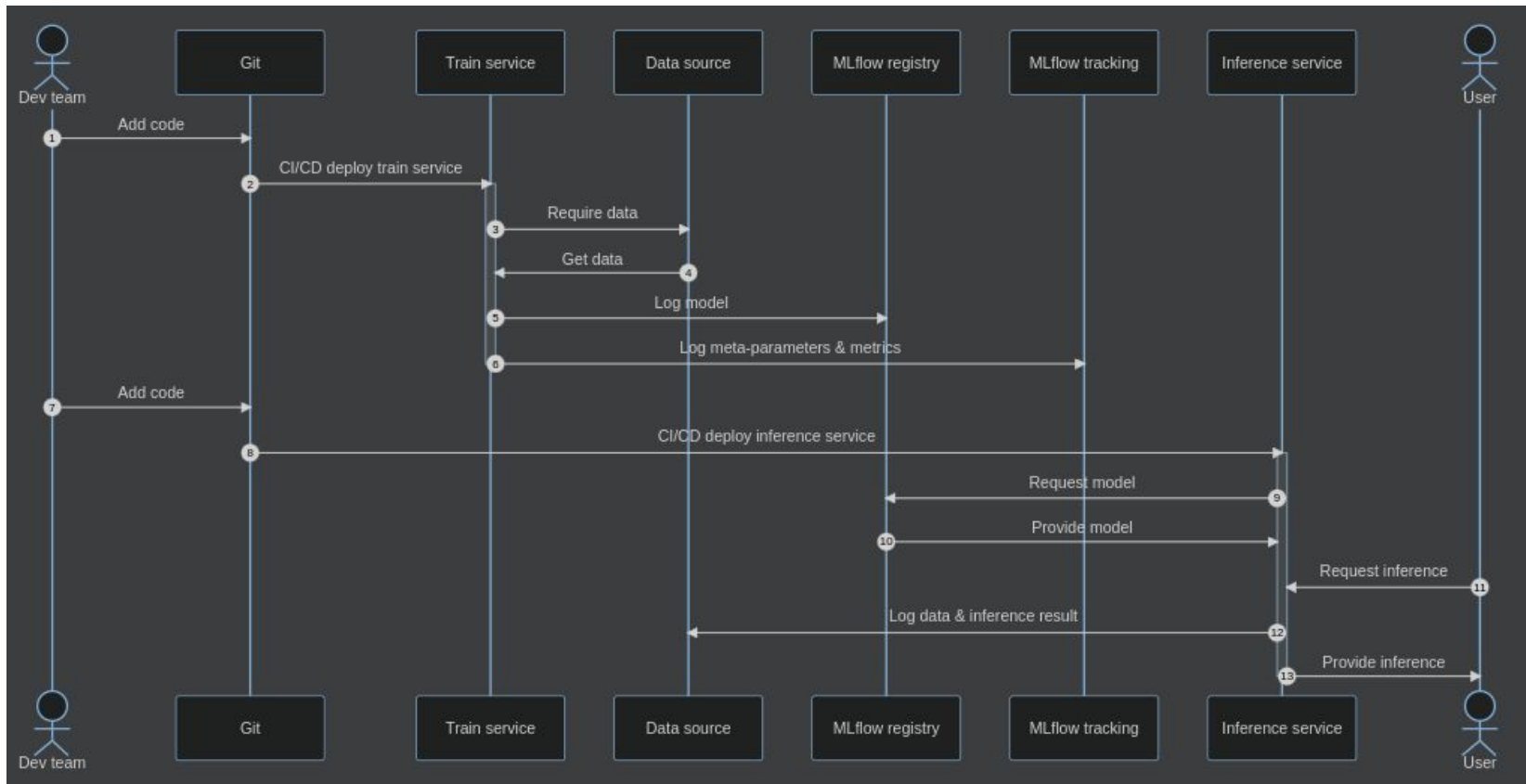
During inference, if it changed

```
1  @app.get("/predict")
2  def predict(X):
3      model_handler = ModelHandler()
4      model = model_handler.load()
5      return model.predict(X)
6
7  class ModelHandler:
8      def load(self):
9          if self.check_if_model_changed():
10             self.model_ = self._load()
11             return self.model_
```

Pseudocode of inference, that check if model was updated at each inference to update it if necessary.

Alternative : load model at each inference, or at each start of the service

Sequence diagram to get an inference




05

Conclusion


Where to train?

In a few words

	 When ?
Inside inference service	For a prototype or online learning
In data scientist env	Training is not frequent
In a dedicated production service	When training is triggered mainly by data changes


Where to store model versions?

In a few words

	 When to use it ?
Git	When train is performed in dev env
Storages such as blob	When train is performed in a dedicated service
Same registry as software	When train is performed in the CI
A specialized registry	When organization already have the tool, or other needs justify it

When to load a new version?

In a few words

	 When ?
During deployment	When train is performed in CI or in dev
At service start	When model change only on code change
During inference	When train can be triggered by a data change When inference are not too frequent (batch)
During inference, when model is updated	When train can be triggered by a data change When inferences are frequent

Takeaway

- CI/CD is a bit more complex in ML because there is a **new artifact** and a **new reason source of change**.
- There is no single good way to perform CI/CD when in an ML context: choose your pattern depending on your **context** and document then in **Architecture Decision Records**.
- Software tools are not always enough to deploy ML - but they can be, at least temporarily.
- **A software with ML can look like all three examples during its lifecycle:** CI/CD will need to change with its needs.



Thanks

Questions ?