Lab 7 : Gestion du cycle de vie des modèles avec MLflow

Étape 1 : Initialisation de l'environnement et installation de MLflow

`pip install mlflow==2.20.1`

```
- avg_session_minutes: mean_prod=36.000 | mean_train=35.124 pip install mlflow==2.20.1
>> ultat : aucun drift détecté.
>> nv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01>
Collecting mlflow==2.20.1
  Downloading mlflow-2.20.1-py3-none-any.whl.metadata (30 kB)
Collecting mlflow-skinny==2.20.1 (from mlflow==2.20.1)
  Downloading mlflow_skinny-2.20.1-py3-none-any.whl.metadata (31 kB)
Collecting Flask<4 (from mlflow==2.20.1)
  Using cached flask-3.1.2-py3-none-any.whl.metadata (3.2 kB)
Collecting Jinja2<4,>=3.0 (from mlflow==2.20.1)
```

Étape 2 : Création explicite de l'espace de stockage MLflow

Instruction :Créer une structure dédiée au stockage MLflow.

CommandesÀ la racine du projet :

`mkdir mlflow/artifacts`

Résultat attendu :

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> mkdir mlflow/artifacts


    Directory: C:\Users\anoua\projects\MLOPS\mlops-lab-01\mlflow


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/15/2026   9:56 PM                artifacts
```

- **métadonnées (base de données),**

• **artefacts (modèles, fichiers).**

Étape 3 : Configuration du client MLflow

Instruction :Configurer l'URI du tracking server pour le projet.

`setx MLFLOW_TRACKING_URI http://127.0.0.1:5000`

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> setx MLFLOW_TRACKING_URI http://127.0.0.1:5000

SUCCESS: Specified value was saved.
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01>
```

VERFIFICATION :
`$Env:MLFLOW_TRACKING_URI`

```
PS C:\Users\anoua> $Env:MLFLOW_TRACKING_URI
http://127.0.0.1:5000
PS C:\Users\anoua>
```

Étape 4 : Démarrage du serveur MLflow (tracking server)

Instruction :Démarrer un serveur MLflow local avec SQLite et un artifact store dédié.

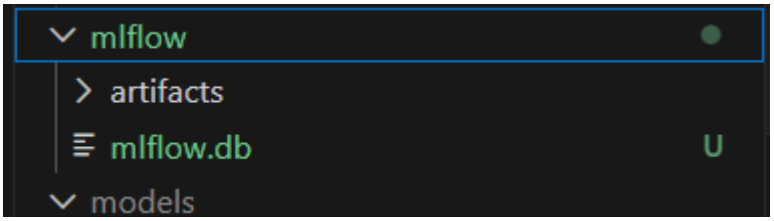
`mlflow server --backend-store-uri sqlite:///mlflow/mlflow.db --default-artifact-root ./mlflow/artifacts --host 127.0.0.1 --port 5000`

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> $Env:MLFLOW_TRACKING_URI
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> mlflow server --backend-store-uri sqlite:///mlf
low/mlflow.db --default-artifact-root ./mlflow/artifacts --host 127.0.0.1 --port 5000
2026/01/15 22:00:32 INFO mlflow.store.db.utils: Creating initial MLflow database tables...
2026/01/15 22:00:32 INFO mlflow.store.db.utils: Updating database tables
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> 451aebb31d03, add metric step
INFO  [alembic.runtime.migration] Running upgrade 451aebb31d03 -> 90e64c465722, migrate user column to tags
INFO  [alembic.runtime.migration] Running upgrade 90e64c465722 -> 181f10493468, allow nulls for metric valu
es
INFO  [alembic.runtime.migration] Running upgrade 181f10493468 -> df50e92ffc5e, Add Experiment Tags Table
```

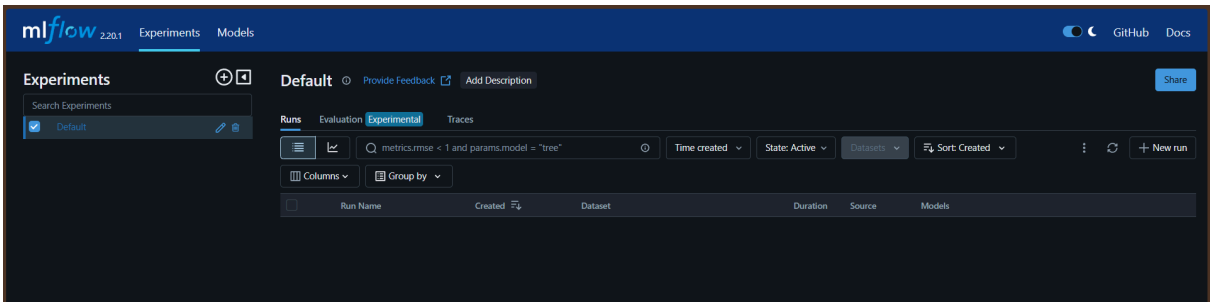
Résultat attendu :

un fichier `mlflow/mlflow.db` est créé,l'interface MLflow est accessible sur-http://127.0.0.1:5000

- **un fichier `mlflow/mlflow.db` est créé,**



- **l'interface MLflow est accessible sur http://127.0.0.1:5000**



Étape 5 : Instrumentation réelle de train.py

Instruction :Modifier `src/train.py` afin qu'une exécution d'entraînement enregistre automatiquement :(1) les paramètres et métriques, (2) le fichier modèle exporté, (3) une version du modèle dans le Model Registry MLflow.

1) Ajout des imports MLflow

Dans le bloc des imports en haut du fichier, ajouter :

```
import mlflow
import mlflow.sklearn
from mlflow.tracking import MlflowClient
```

2) Constante globale

Dans le bloc des constantes globales, ajouter :

```
MODEL_NAME: Final[str] = "churn_model
```

2) Insertion du bloc MLflow à l'endroit exact dans `main()`

Dans

```
main()
```

, repérer la ligne où le modèle est déjà sauvegardé sur disque :

```
joblib.dump(pipe, stable_model_path)
```

Immédiatement après cette ligne, insérer le bloc suivant :

```
#Associe le run à une expérience logique. Les exécutions seront regroupée
s sous ce nom dans l'interface MLflow.
mlflow.set_tracking_uri("http://127.0.0.1:5000")
mlflow.set_experiment("mlops-lab-01")

#Crée une exécution traçable correspondant à l'entraînement courant. Le n
om du run porte ici la version logique passée au script.
with mlflow.start_run(run_name=f"train-{version}") as run:
    run_id = run.info.run_id
    #Enregistre le contexte de l'entraînement. Les paramètres sont essentiel
s pour comprendre et reproduire une exécution.
    mlflow.log_param("version", version)
    mlflow.log_param("seed", seed)
    mlflow.log_param("gate_f1", gate_f1)
    #Enregistre l'ensemble des métriques calculées par le script (F1, accurac
y, etc.).
    #Chaque métrique devient comparable entre runs dans l'UI.
    mlflow.log_metrics(metrics)
    #Ajoute des informations descriptives (métadonnées) destinées à facilite
r la lecture humaine
    #quel fichier de données a été utilisé, quel fichier modèle a été produit.
    mlflow.set_tag("data_file", DATA_PATH.name)
    mlflow.set_tag("model_file", model_filename)
    #Attache le fichier modèle exporté (celui déjà généré via joblib.dump) au
run MLflow. Il apparaît dans la section Artifacts du run.
    mlflow.log_artifact(str(model_path), artifact_path="exported_models")
    #Enregistre le pipeline entraîné comme un modèle MLflow et le publie da
```

```
ns le Model Registry sous un nom stable (churn_model).
    #Chaque exécution crée une nouvelle version (v1, v2, ...) associée au ru
n.
    mlflow.sklearn.log_model(
        sk_model=pipe,
        artifact_path="model",
        registered_model_name="churn_model",
    )
```

4) Exécution

```
python src/train.py
```

5) Résultat attendu

Dans l'interface MLflow, un run apparaît dans l'expérience `mlops-lab-01` avec :

- **les paramètres `version`, `seed`, `gate_f1`,**
- **les métriques du dictionnaire `metrics`,**
- **un artefact `exported_models/<fichier_joblib>`.Dans le Model Registry, un modèle `churn_model` apparaît avec une nouvelle version créée par l'exécution.**

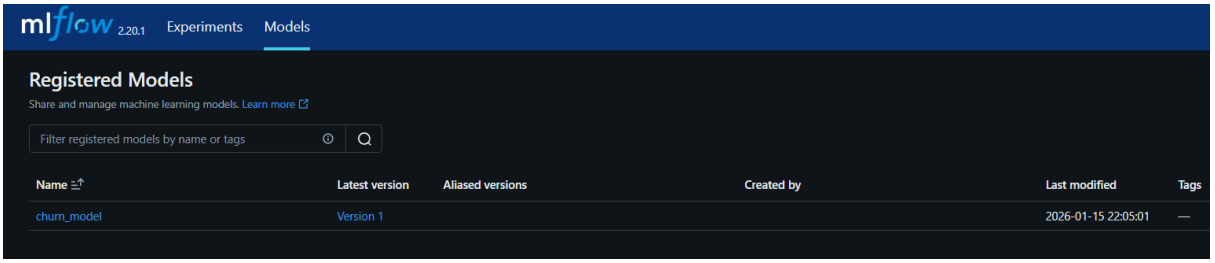
```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> python src/train.py
C:\Users\anoua\projects\MLOPS\mlops-lab-01\src\train.py:529: DeprecationWarning: datetime.datetime.utcnow()
 is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datet
imes in UTC: datetime.datetime.now(datetime.UTC).
  timestamp = datetime.utcnow().strftime("%Y%m%d_%H%M%S")
[METRICS] {
  "accuracy": 0.6433333333333333,
  "precision": 0.6687898089171974,
  "recall": 0.65625,
  "f1": 0.6624605678233438,
  "baseline_f1": 0.0
}
[OK] Modèle sauvegardé : C:\Users\anoua\projects\MLOPS\mlops-lab-01\models\churn_model_v1_20260115_210446.j
oblib
[DEPLOY] Modèle activé (current): churn_model_v1_20260115_210446.joblib
------------------------------------------------------------
2026/01/15 22:04:46 INFO mlflow.tracking.fluent: Experiment with name 'mlops-lab-01' does not exist. Creati
ng a new experiment.
2026/01/15 22:05:00 WARNING mlflow.models.model: Model logged without a signature and input example. Please
 set `input_example` parameter when logging the model to auto infer the model signature.
Successfully registered model 'churn_model'.
2026/01/15 22:05:01 INFO mlflow.store.model_registry.abstract_store: Waiting up to 300 seconds for model ve
rsion to finish creation. Model name: churn_model, version 1
Created version '1' of model 'churn_model'.
🏃 View run train-v1 at: http://127.0.0.1:5000/#/experiments/1/runs/081c997a0a6246e5b36cc5aacf3d80a2
🧪 View experiment at: http://127.0.0.1:5000/#/experiments/1
[DEPLOY] Modèle activé : churn_model_v1_20260115_210446.joblib
[DEPLOY] Alias stable : C:\Users\anoua\projects\MLOPS\mlops-lab-01\models\model.joblib
[MLflow] Run enregistré : 081c997a0a6246e5b36cc5aacf3d80a2 | Model Registry : churn_model
```

Étape 6 : Observation du registry MLflow

Instruction :Observer le modèle enregistré dans l'UI.

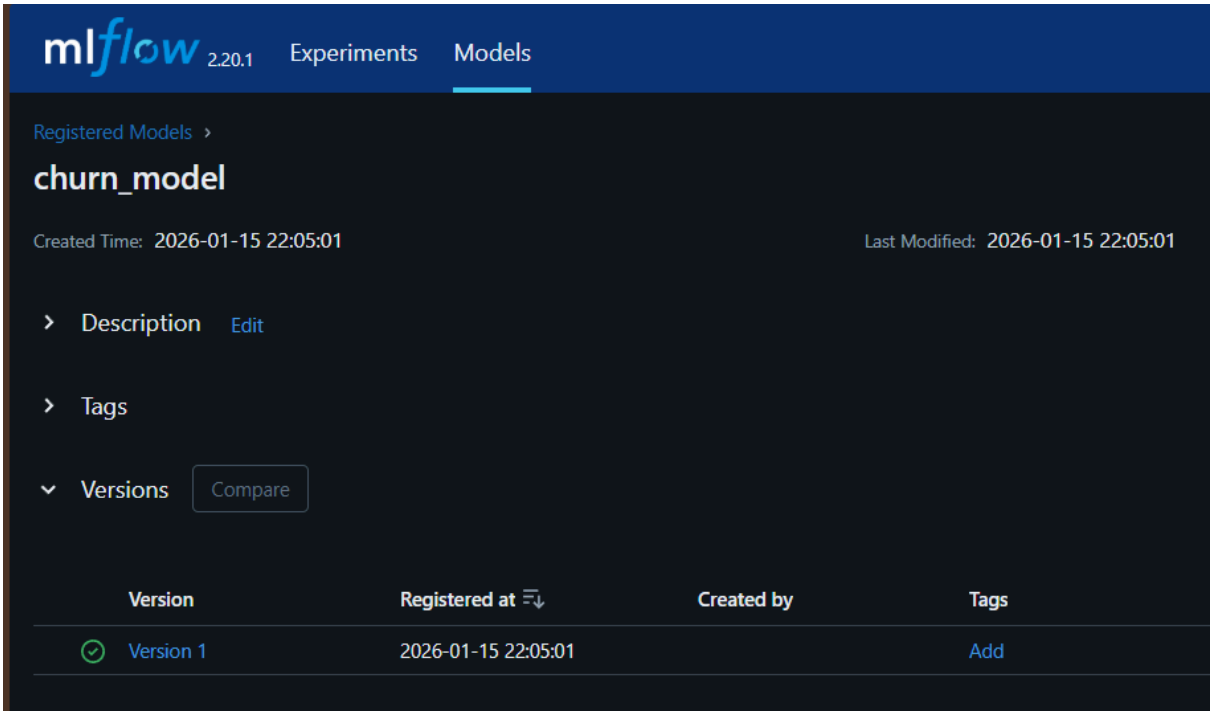
Action

- **Ouvrir : http://127.0.0.1:5000**
- **Onglet Models**
- **Sélectionner** `churn_model` **Résultat attendu :**



- **Version 1 créée**

- **Lien vers le run d'origine**



Explication pédagogique :

Le registry MLflow remplit maintenant le rôle conceptuel étudié dans le cours.

Étape 7 : Promotion d'un modèle (activation)

Instruction :Créer un script de promotion.

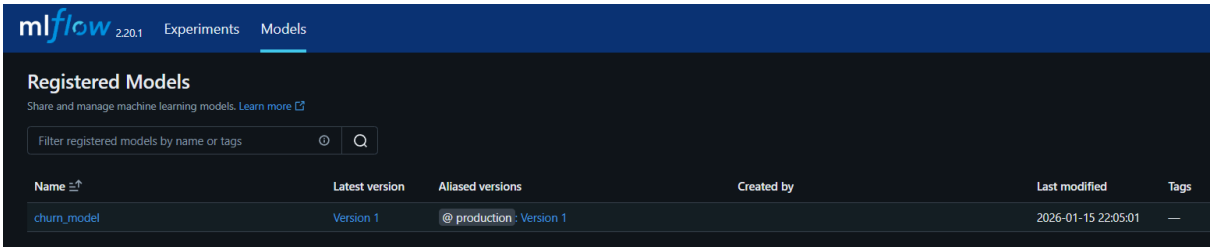
src > promote.py > ...

```python
"""
Script de promotion du modèle.

Promotionne la dernière version du modèle entraîné vers l'alias "production"
dans le MLflow Model Registry.

Flux:
1. Se connecte au tracking server MLflow (127.0.0.1:5000)
2. Cherche toutes les versions du modèle "churn_model"
3. Identifie la version la plus récente
4. Assigne l'alias "production" à cette version
5. Affiche le statut de la promotion

Usage:
    python src/promote.py
"""

import mlflow
from mlflow.tracking import MlflowClient


MODEL_NAME = "churn_model"
ALIAS = "production"


mlflow.set_tracking_uri("http://127.0.0.1:5000")
client = MlflowClient()
```

Résultat attendu :

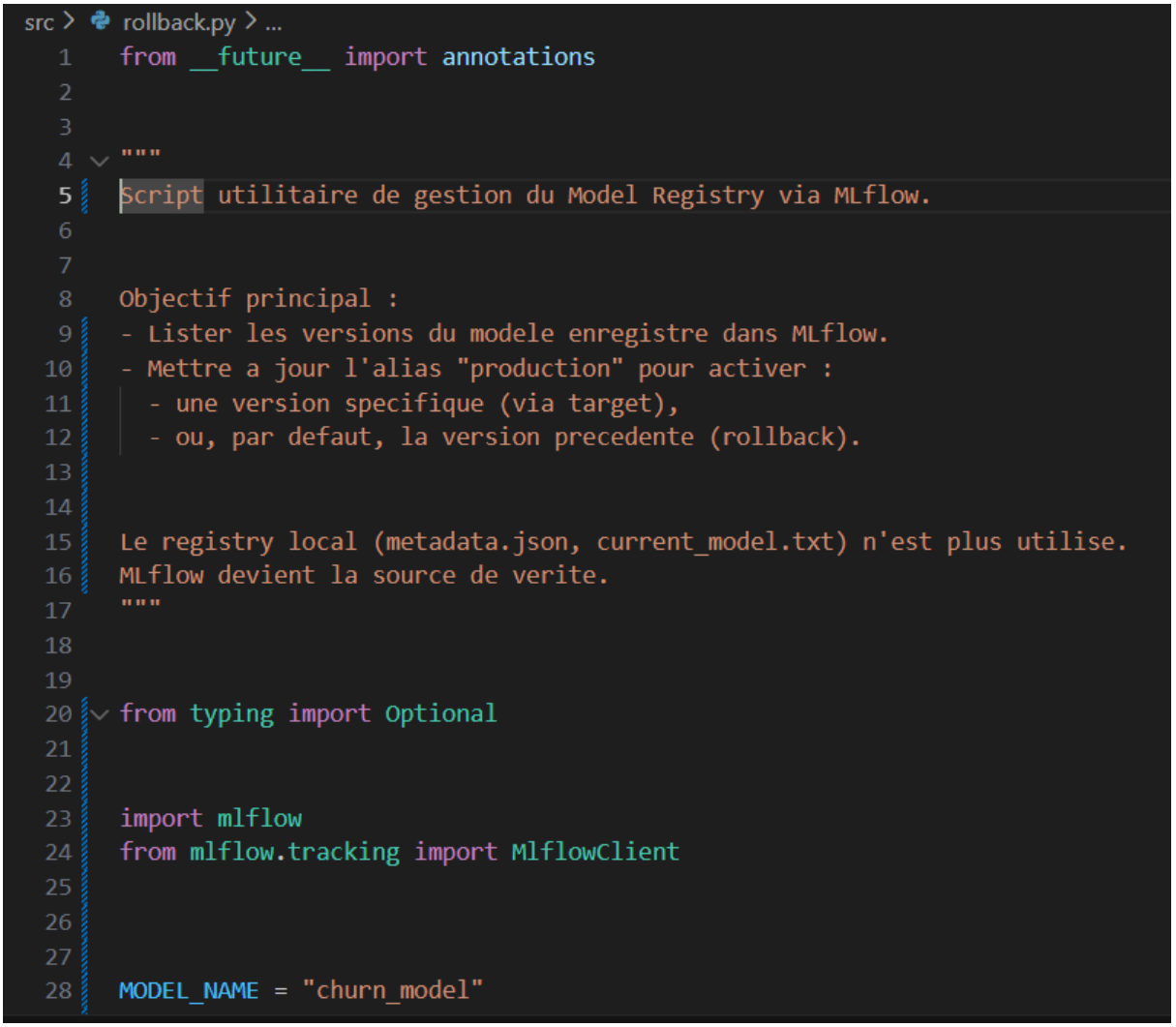
alias `production` visible dans l'UI MLflow.

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> python src/promote.py
Modèle activé : churn_model@production -> v1
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01>
```



Étape 8 : Rollback via MLflow Model Registry

Instruction :Modifier `src/rollback.py` afin qu'il n'écrive plus dans `registry/current_model.txt`, mais qu'il mette à jour l'alias MLflow `production` vers une version antérieure du modèle `churn_model`.



Interprétation du script

- **Le script ne lit plus `registry/metadata.json` et n'écrit plus `registry/current_model.txt` .La "source de vérité" devient le Model Registry MLflow.**
- **`client.search_model_versions(...)` récupère toutes les versions disponibles de `churn_model` .**
- **L'alias `production` remplace conceptuellement "current_model" :changer d'alias revient à changer de modèle actif, sans toucher au code ni aux fichiers.**
- **Deux modes :**
  - **`python src/rollback.py` : rollback automatique vers la version précédente.**
  - **`python -c "from src.rollback import main; main('3')"` : activation explicite de la version 3**

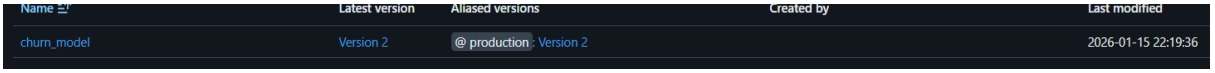
```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> python src/rollback.py
Traceback (most recent call last):
  File "C:\Users\anoua\projects\MLOPS\mlops-lab-01\src\rollback.py", line 131, in <module>
    main(sys.argv[1] if len(sys.argv) > 1 else None)
  File "C:\Users\anoua\projects\MLOPS\mlops-lab-01\src\rollback.py", line 115, in main
    raise ValueError(
ValueError: Rollback impossible : churn_model@production est deja sur la plus ancienne version (v1).
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01>
```

Après avoir entraîné au moins deux versions (v1, v2), exécuter :

```
Created version '2' of model 'churn_model'.
View run train-v1 at: http://127.0.0.1:5000/#/experiments/1/runs/c78d1cc3f4f54924bcf3737123180
View experiment at: http://127.0.0.1:5000/#/experiments/1
[DEPLOY] Modèle activé : churn_model_v1_20260115_211924.joblib
[DEPLOY] Alias stable : C:\Users\anoua\projects\MLOPS\mlops-lab-01\models\model.joblib
[MLflow] Run enregistré : c78d1cc3f4f54924bcf3737123180c0a | Model Registry : churn_model
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> python src/promote.py
Modèle activé : churn_model@production -> v2
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> python src/rollback.py
[OK] rollback => churn_model@production : v2 -> v1
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> python src/rollback.py 2
[OK] activation => churn_model@production = v2
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01>
```

Résultat attendu :

L'alias MLflow production **pointe vers la version précédente du modèle churn_model**



Étape 9 : API : chargement du modèle actif

Instruction :Adapter l'API pour charger le modèle actif depuis MLflow.

Imports à ajouter en `src/api.py`

En haut :

```
import mlflow
import mlflow.sklearn
from mlflow.tracking import MlflowClient
```

Ajoute ces constantes (près de tes constantes)

```
MLFLOW_TRACKING_URI = "http://127.0.0.1:5000"
MODEL_NAME = "churn_model"
ALIAS = "production"
MODEL_URI = f"models:/{MODEL_NAME}@{ALIAS}"
```

Remplace `get_current_model_name()` par ceci

```
def get_current_model_name() -> str:
    mlflow.set_tracking_uri(MLFLOW_TRACKING_URI)
    client = MlflowClient()
    mv = client.get_model_version_by_alias(MODEL_NAME, ALIAS)
    return f"{MODEL_NAME}@{ALIAS} (v{mv.version})"
```

Remplace `load_model_if_needed()` par ceci

```
def load_model_if_needed() -> tuple[str, Any]:
    mlflow.set_tracking_uri(MLFLOW_TRACKING_URI)

    # Cache key = model URI (alias), not local filename
    cache_key = MODEL_URI

    if _model_cache["name"] == cache_key and _model_cache["model"] is n
ot None:
        return cache_key, _model_cache["model"]

    model = mlflow.sklearn.load_model(MODEL_URI)

    _model_cache["name"] = cache_key
    _model_cache["model"] = model
    return cache_key, model
```

src > api.py > ...

```python
import joblib
import pandas as pd
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel, Field

import mlflow
import mlflow.sklearn
from mlflow.tracking import MlflowClient


# ---------------------------------------------------------------------------
# Constantes de chemin
# ---------------------------------------------------------------------------


ROOT: Path = Path(__file__).resolve().parents[1]
MODELS_DIR: Path = ROOT / "models"
REGISTRY_DIR: Path = ROOT / "registry"
CURRENT_MODEL_PATH: Path = REGISTRY_DIR / "current_model.txt"
LOG_PATH: Path = ROOT / "logs" / "predictions.log"

# Constantes MLflow
MLFLOW_TRACKING_URI = "http://127.0.0.1:5000"
MODEL_NAME = "churn_model"
ALIAS = "production"
MODEL_URI = f"models:/{MODEL_NAME}@{ALIAS}"


# Configuration du logging
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(name)s - %(levelname)s - [%(request_id)s] - %(message)s"
)
logger = logging.getLogger(__name__)
```

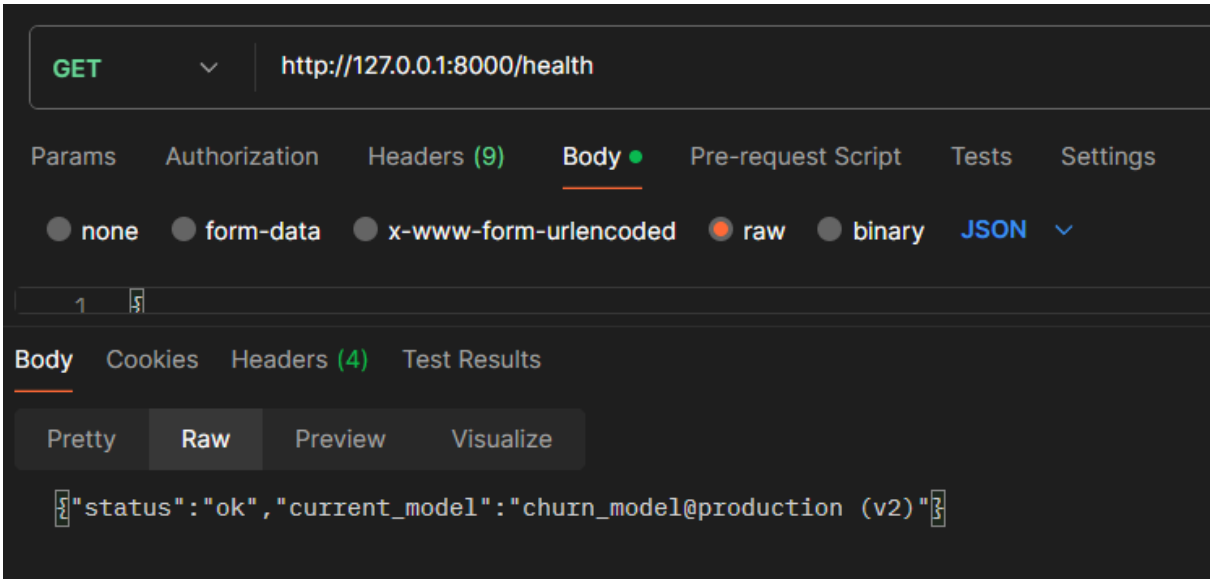
Start the FastApi server

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> uvicorn src.api:app --host 0.0.0.0 --port 8000
INFO:     Started server process [5904]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

Résultat attendu :

l'API sert toujours la version active,même après promotion ou rollback.

- **l'API sert toujours la version active,**
- **même après promotion ou rollback.**


GET
http://127.0.0.1:8000/health
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
JSON
1
{
Body
Cookies
Headers (4)
Test Results
Pretty
Raw
Preview
Visualize
{"status":"ok","current_model":"churn_model@production (v2)"}