

Lab 5 : Du Notebook au Déploiement Conteneurisé d'un Modèle de Machine Learning

Étape 1 : Vérifier l'installation de Docker

Instructions :

- Ouvrez un terminal (PowerShell ou bash).
- Vérifiez la version de Docker :

```
docker --version
```

- Vérifiez que le démon Docker fonctionne :

```
docker ps
```

```
PS C:\Users\anoua> docker --version
Docker version 29.1.3, build f52814d
PS C:\Users\anoua> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED    STATUS    PORTS    NAMES
PS C:\Users\anoua>
```

Étape 2 : Lancer un serveur Nginx dans un conteneur

Instructions :

Lancez un conteneur Nginx en arrière-plan :

- Lancez un conteneur Nginx en arrière-plan :

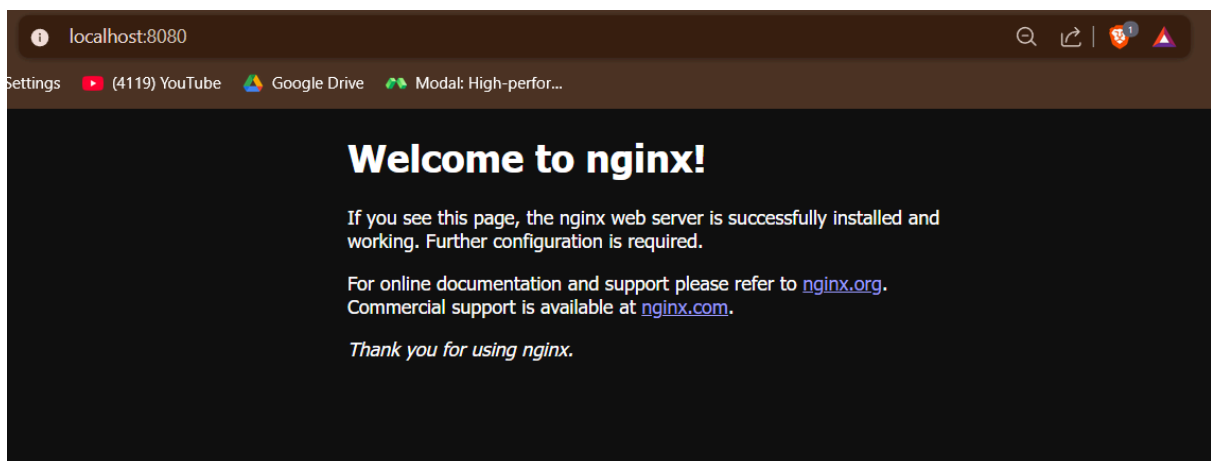
```
docker run -d -p 8080:80 --name demo-nginx nginx
```

```
PS C:\Users\anoua> docker run -d -p 8080:80 --name demo-nginx nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
02d7611c4eae: Pull complete
dcea87ab9c4a: Pull complete
35df28ad1026: Pull complete
99ae2d6d05ef: Pull complete
a2b008488679: Pull complete
d03ca78f31fe: Pull complete
d6799cf0ce70: Pull complete
Digest: sha256:ca871a86d45a3ec6864dc45f014b11fe626145569ef0e74deaffc95a3b15b430
Status: Downloaded newer image for nginx:latest
badcd23f1708315f956d11cb54ea64da76bd5659437ca18a312ad4cf6523fabb
PS C:\Users\anoua>
```

Ouvrez un navigateur et accédez à :

- Ouvrez un navigateur et accédez à :

http://localhost:8080



- Vérifiez que la page par défaut de Nginx s'affiche.
- Listez les conteneurs en cours d'exécution :

docker ps

```
PS C:\Users\anoua> docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS
NAMES
badcd23f1708   nginx    "/docker-entrypoint..." About a minute ago Up 47 seconds  0.0.0.0:8080->80/tcp, [::]:8080->
PS C:\Users\anoua>
```

- Arrêtez puis supprimez le conteneur :

```
docker stop demo-nginx
docker rm demo-nginx
```

```
PS C:\Users\anoua> docker stop demo-nginx
demo-nginx
PS C:\Users\anoua> docker rm demo-nginx
demo-nginx
PS C:\Users\anoua>
```

Étape 3 : Ouvrir un shell Linux isolé dans un conteneur

Instructions :

- Lancez un conteneur Linux interactif (exemple avec `ubuntu`) :

```
docker run -it --name demo-ubuntu ubuntu bash
```

```
PS C:\Users\anoua> docker run -it --name demo-ubuntu ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
20043066d3d5: Pull complete
Digest: sha256:c35e29c9450151419d9448b0fd75374fec4fff364a27f176fb458d472dfc9e54
Status: Downloaded newer image for ubuntu:latest
root@0f712daabcca:/#
```

- Dans le shell à l'intérieur du conteneur, exécutez quelques commandes :

```
ls
cat /etc/os-release
pwd
```

```

root@0f712daabcca:/# ls
bin boot dev etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@0f712daabcca:/# cat /etc/os-release
PRETTY_NAME="Ubuntu 24.04.3 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04.3 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
root@0f712daabcca:/# pwd
/
root@0f712daabcca:/#

```

- Installez un paquet (exemple) :

```

apt-get update
apt-get install -y curl

```

```

root@0f712daabcca:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [2898 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble/multiverse amd64 Packages [331 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble/universe amd64 Packages [19.3 MB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [33.1 kB]
Get:9 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1183 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1752 kB]
Get:11 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1808 kB]
Get:12 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [3059 kB]
Get:14 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1950 kB]
Get:15 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [2130 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Packages [35.9 kB]
Get:17 http://archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [34.6 kB]
Get:18 http://archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [49.5 kB]
Fetched 35.3 MB in 45s (792 kB/s)
Reading package lists... Done
root@0f712daabcca:/# apt-get install -y curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates krb5-locales libbrotli1 libcurl4t64 libgssapi-krb5-2 libk5crypto3 libkeyutils1 libkrb5-3
  libkrb5support0 libldap-common libldap2 libnghttp2-14 libpsl5t64 librtmp1 libsasl2-2 libsasl2-modules

```

- Quittez le shell :

```
exit
```

```

root@0f712daabcca:/# exit
exit
PS C:\Users\anoua> |

```

- Vérifiez que le conteneur existe toujours mais est arrêté :

```
docker ps -a
```

```
PS C:\Users\anoua> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
0f712daabcca   ubuntu   "bash"    4 minutes ago   Exited (0) 29 seconds ago   demo-ubuntu
PS C:\Users\anoua>
```

- Supprimez ce conteneur :

```
docker rm demo-ubuntu
```

```
PS C:\Users\anoua> docker rm demo-ubuntu
demo-ubuntu
PS C:\Users\anoua> |
```

Étape 4 : Comprendre la structure d'une commande docker run

Instructions :

Notez la structure générale :

```
docker run [options] image [commande] [arguments]
```

Lancez de nouveau Nginx avec des options clairement visibles :

```
PS C:\Users\anoua> docker run -d --name demo-nginx -p 8080:80 nginx
c8afc13927a2c3274bff3eca68517669275df26425c19ec46887309cac3ebf06
PS C:\Users\anoua>
```

- Identifiez le rôle de chaque élément :
- **d** : détaché (arrière-plan)
 - **-name demo-nginx** : nom du conteneur
 - **p 8080:80** : port hôte 8080 → port conteneur 80
 - **nginx** : image utilisée

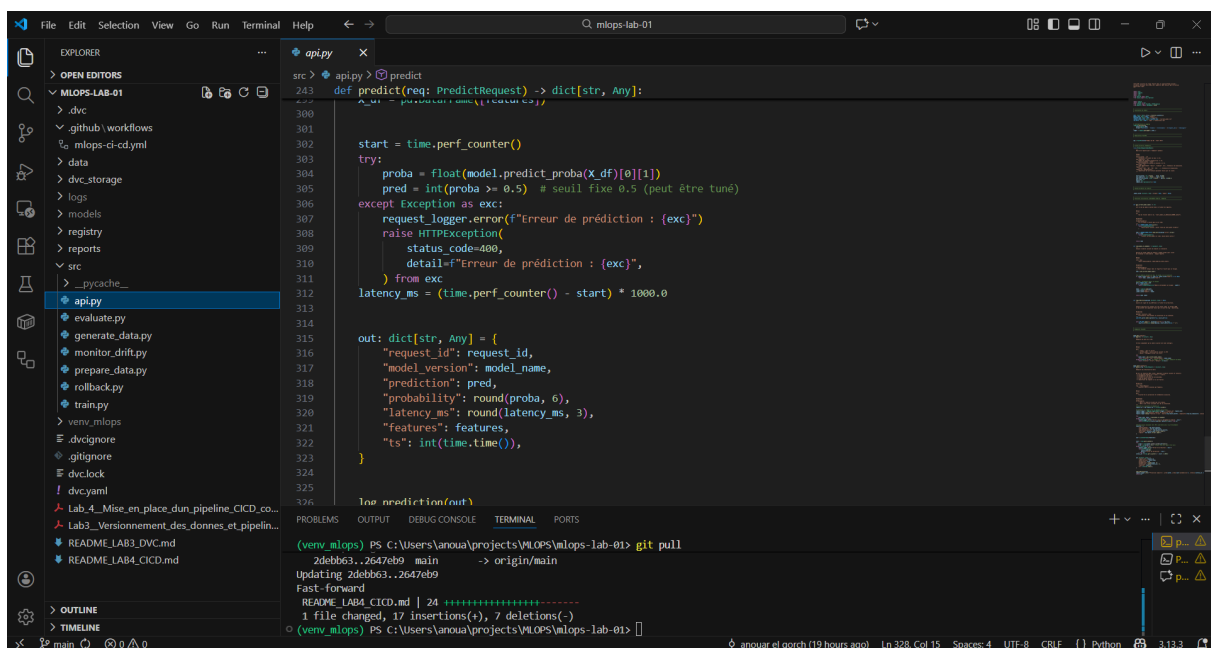
- Arrêtez et supprimez encore une fois le conteneur :

```
PS C:\Users\anoua> docker stop demo-nginx
demo-nginx
PS C:\Users\anoua> docker rm demo-nginx
demo-nginx
PS C:\Users\anoua>
```

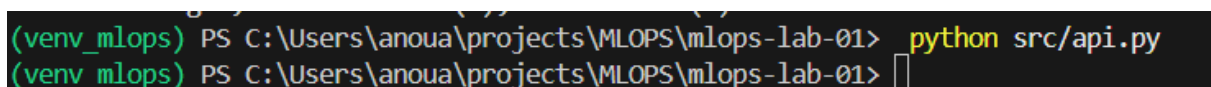
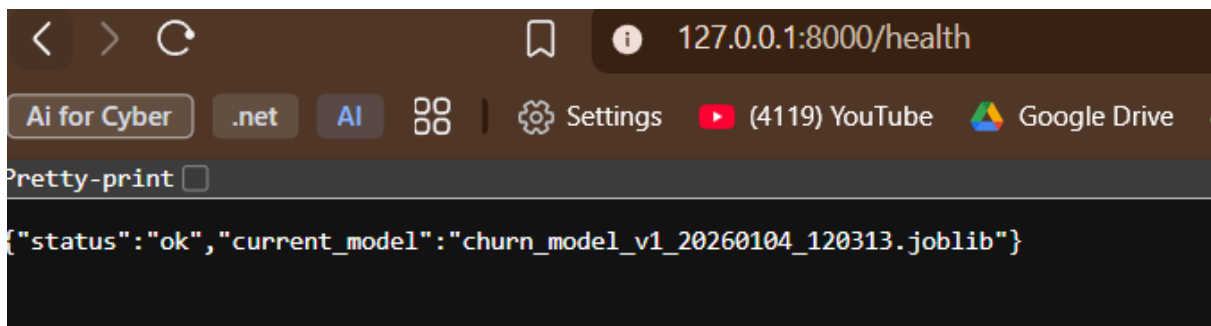
Étape 5 : Conteneuriser l'API churn du projet mlops-lab-01

- Vérifiez que l'arborescence contient au minimum :

```
mlops-lab-01/
├── data/
├── logs/
├── models/
├── registry/
└── src/
```



- Vérifiez que l'API fonctionne encore en local (optionnel mais recommandé) :



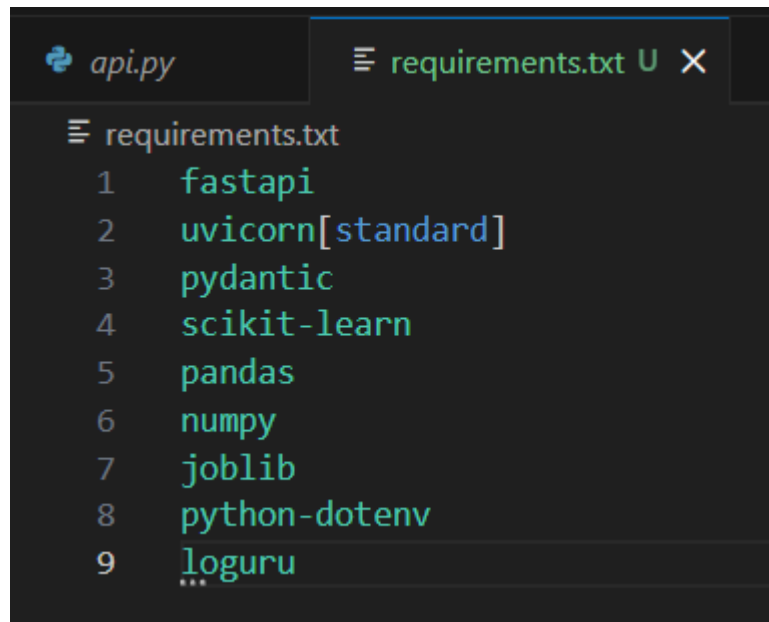
Étape 6 : Créer un fichier requirements.txt pour l'image Docker

Instructions :

1. Dans le dossier `mlops-lab-01`, créez un fichier `requirements.txt`.
2. Ajoutez le contenu suivant (version minimale) :

```
fastapi
uvicorn[standard]
pydantic
scikit-learn
pandas
numpy
joblib
```

1. Enregistrez le fichier. *Remarque : si tu as des libs supplémentaires dans ton lab (par ex. `python-dotenv`, `loguru` ...), ajoute-les ici.*

A screenshot of a code editor window. The top bar shows two tabs: 'api.py' and 'requirements.txt'. The 'requirements.txt' tab is active, displaying a list of Python dependencies. The text is as follows:

```
requirements.txt
1 fastapi
2 uvicorn[standard]
3 pydantic
4 scikit-learn
5 pandas
6 numpy
7 joblib
8 python-dotenv
9 loguru
```

Étape 7 : Créer un Dockerfile pour l'API churn

Instructions :

1. Dans le dossier `mlopslab-01`, créez un fichier nommé :
1. Collez le contenu suivant :
1. Sauvegardez le fichier.


```
api.py requirements.txt U Dockerfile U X
Dockerfile > ...
1 FROM python:3.10-slim (last pushed 3 days ago)
2
3 # 1) Préparer le dossier de travail dans le conteneur
4 WORKDIR /app
5
6 # 2) Copier les dépendances et les installer
7 COPY requirements.txt .
8 RUN pip install --no-cache-dir -r requirements.txt
9
10 # 3) Copier le reste du projet dans l'image
11 COPY . .
12
13 # 4) Exposer le port de l'API (8000 dans notre lab)
14 EXPOSE 8000
15
16 # 5) Commande de lancement de l'API FastAPI
17 CMD ["uvicorn", "src.api:app", "--host", "0.0.0.0", "--port", "8000"]
```

Étape 8 : Préparer un modèle actif avant de construire l'image

Instructions :

1. Assurez-vous qu'un modèle entraîné existe déjà dans `models/` :

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> ls models

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

Mode                LastWriteTime         Length Name
----                -
-a----             1/3/2026 10:12 PM           3690 churn_model_v1_20260103_211231.joblib
-a----             1/3/2026 10:13 PM           3690 churn_model_v1_20260103_211322.joblib
-a----             1/3/2026 10:14 PM           3690 churn_model_v1_20260103_211426.joblib
-a----             1/4/2026  1:02 PM           3690 churn_model_v1_20260104_120241.joblib
-a----             1/4/2026  1:02 PM           3690 churn_model_v1_20260104_120245.joblib
-a----             1/4/2026  1:03 PM           3690 churn_model_v1_20260104_120313.joblib
-a----             1/3/2026 10:20 PM           3690 churn_model_v2_20260103_212051.joblib
```

2. Assurez-vous que `registry/current_model.txt` contient bien le nom d'un modèle (une ligne du type) :

```
≡ requirements.txt U    Dockerfile U    ≡ current_model.txt X

registry > ≡ current_model.txt
1  churn_model_v1_20260104_120313.joblib
```

3. Vérifiez à nouveau le contenu de :

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> type registry/current_model.txt
churn_model_v1_20260104_120313.joblib
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> █
```

Étape 9 : Construire l'image Docker du projet churn

Instructions :

1. Dans le dossier `mlops-lab-01`, construisez l'image :

```
PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker build -t churn-api:latest .
[+] Building 472.1s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 501B
=> [internal] load metadata for docker.io/library/python:3.10-slim
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 497.75MB
=> [1/5] FROM docker.io/library/python:3.10-slim@sha256:7b68a5fa7cf0d20b4cedb1dc9a134fdd394fe27edbc4c2519756c91d21df2313
=> => resolve docker.io/library/python:3.10-slim@sha256:7b68a5fa7cf0d20b4cedb1dc9a134fdd394fe27edbc4c2519756c91d21df2313
=> => sha256:8715e552fa1374bdde269437d9a1c607c817289c2ebbce9ed9ab1aa9ca86763 1.29MB / 1.29MB
=> => sha256:9c27bc7ba63d1ac690daefc68302197d3ab9a91fc5c0e19f447cd57eda92d87c 13.82MB / 13.82MB
=> => sha256:7da4424a113245eb185ea22f2512cecb36f80ca1d0547c64b117f28495d3c3e5 250B / 250B
=> => sha256:7b68a5fa7cf0d20b4cedb1dc9a134fdd394fe27edbc4c2519756c91d21df2313 10.37kB / 10.37kB
=> => sha256:05be60a538e21a03887c1b1ecbc37e18a204d7abd2ff9d18ec9e95a868d83364 1.75kB / 1.75kB
=> => sha256:ce19342c5d49287e941ab558824eb6b0a4244066b18b5a1a9024b6c0f2f818a8 5.48kB / 5.48kB
=> => extracting sha256:8715e552fa1374bdde269437d9a1c607c817289c2ebbce9ed9ab1aa9ca86763 0.55s
=> => extracting sha256:9c27bc7ba63d1ac690daefc68302197d3ab9a91fc5c0e19f447cd57eda92d87c 2.8s
=> => extracting sha256:7da4424a113245eb185ea22f2512cecb36f80ca1d0547c64b117f28495d3c3e5 0.0s
=> [2/5] WORKDIR /app
=> [3/5] COPY requirements.txt .
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> [5/5] COPY . .
=> exporting to image
=> exporting layers
=> writing image sha256:9308ce70b31960c62ed562088fbad2cc8d03f166d8dbb851e622682e97761527
=> naming to docker.io/library/churn-api:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/b9hzq2zw114vzj4axop02wzy3
PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> █
```

1. Attendez la fin de la construction, vérifiez la présence de l'image :
Vous devez voir une ligne avec `churn-api` dans la colonne `REPOSITORY` .

```
PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker images
```

IMAGE	ID	DISK USAGE	CONTENT SIZE	EXTRA
churn-api:latest	9308ce70b319	968MB	0B	
nginx:latest	058f4935d1cb	152MB	0B	
ubuntu:latest	c3a134f2ace4	78.1MB	0B	

```
PS C:\Users\anoua\projects\MLOPS\mlops-lab-01>
```

Étape 10 : Lancer l'API churn dans un conteneur

Instructions :

1. Lancez un conteneur basé sur l'image et Vérifiez que le conteneur est en cours d'exécution :

```
PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker run -d --name churn-api-demo -p 8000:8000 churn-api:latest
2f5d9bf857e67eab3f56deee4aa9d4ae5e06c3fb6500555c0969146d4880ca3f
PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2f5d9bf857e6	churn-api:latest	"uvicorn src.api:app..."	55 seconds ago	Up 9 seconds	0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp	churn-api-de

```
PS C:\Users\anoua\projects\MLOPS\mlops-lab-01>
```

2. Testez le endpoint `/health` avec un client HTTP (Postman, curl, ou navigateur si tu as un GET simplifié) :Exemple avec `curl` :

GET ▼ http://localhost:8000/health Send ▼

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
Key	Value	

Body Cookies Headers (4) Test Results 200 OK 14 ms 196 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ ↔

```

1  [
2    "status": "ok",
3    "current_model": "churn_model_v1_20260104_120313.joblib"
4  ]

```

```

PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> curl http://localhost:8000/health

Security Warning: Script Execution Risk
Invoke-WebRequest parses the content of the web page. Script code in the web page might be run when the page is parsed.
RECOMMENDED ACTION:
Use the -UseBasicParsing switch to avoid script code execution.

Do you want to continue?

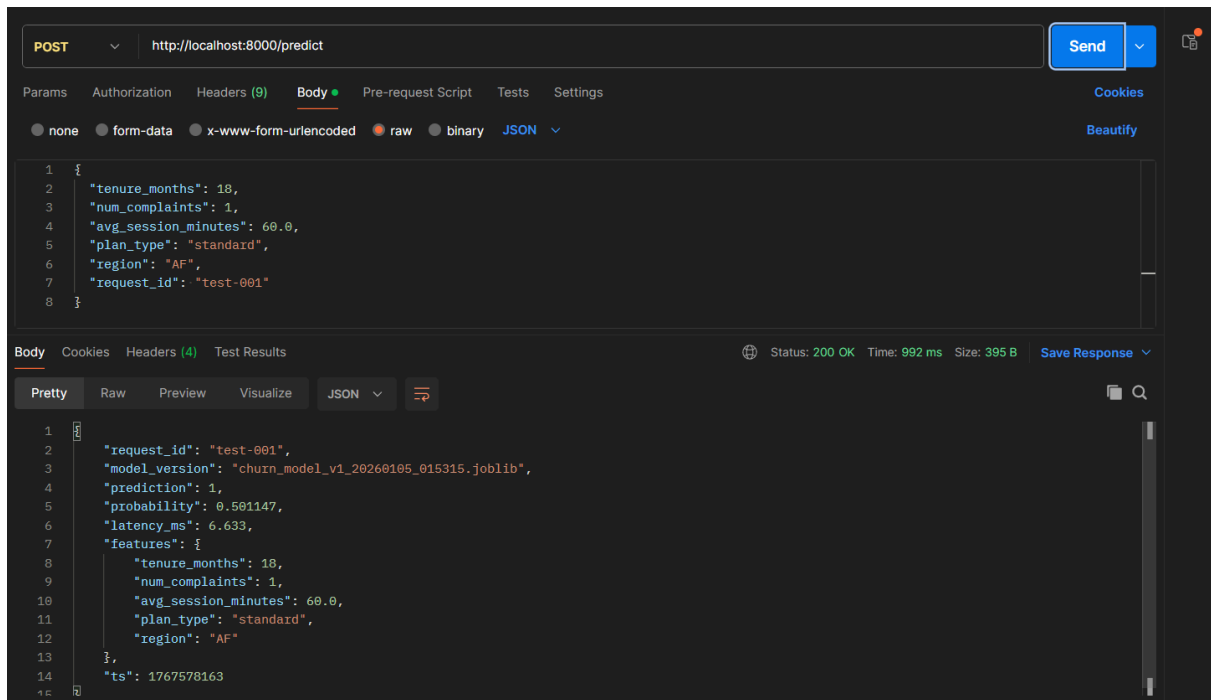
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): y

StatusCode      : 200
StatusDescription : OK
Content          : {"status":"ok","current_model":"churn_model_v1_20260104_120313.joblib"}
RawContent       : HTTP/1.1 200 OK
                   Content-Length: 71
                   Content-Type: application/json
                   Date: Sun, 04 Jan 2026 22:46:32 GMT
                   Server: uvicorn

                   {"status":"ok","current_model":"churn_model_v1_20260104_120313.joblib"}
Forms           : {}
Headers         : {[Content-Length, 71], [Content-Type, application/json], [Date, Sun, 04 Jan 2026 22:46:32 GMT], [Server, uvicorn]}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 71

```

3. Testez une requête POST `/predict` en envoyant un JSON conforme au lab (tenure, complaints, etc.).



Probleme de version sicket learn 1.8 non adapter a python 3.10 donc cahnger version python dans dockerfile vers 3.11

Étape 11 : Vérifier les logs générés à l'intérieur du conteneur

Instructions :

1. Listez les fichiers dans le conteneur :

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker exec -it churn-api-demo ls
Dockerfile
Lab3_Versionnement_des_donnes_et_pipelines_ML_avec_DVC.pdf
Lab_4_Mise_en_place_dun_pipeline_CICD_complet_pour_un_projet_Machine_Learning.pdf
README_LAB3_DVC.md
README_LAB4_CICD.md
data
dvc.lock
dvc.yaml
logs
models
registry
reports
requirements.txt
src
```

2. Vérifiez que l'application écrit des logs à l'exécution :

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker exec -it churn-api-demo ls logs
predictions.log
```

3. Affichez quelques lignes du fichier de logs des prédictions :

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker exec churn-api-demo cat /app/logs/predictions.log
{"request_id": "req-001", "model_version": "churn_model_v1_20260103_211426.joblib", "prediction": 1, "probability": 0.907065, "latency_ms": 6.894, "features": {"tenure_months": 6, "num_complaints": 3, "avg_session_minutes": 12.5, "plan_type": "basic", "region": "AF"}, "ts": 1767475090}
{"request_id": "req-safe", "model_version": "churn_model_v1_20260103_211426.joblib", "prediction": 0, "probability": 0.139973, "latency_ms": 4.661, "features": {"tenure_months": 48, "num_complaints": 0, "avg_session_minutes": 60.0, "plan_type": "premium", "region": "EU"}, "ts": 1767475152}
{"request_id": "test-001", "model_version": "churn_model_v1_20260105_015315.joblib", "prediction": 1, "probability": 0.501147, "latency_ms": 6.633, "features": {"tenure_months": 18, "num_complaints": 1, "avg_session_minutes": 60.0, "plan_type": "standard", "region": "AF"}, "ts": 1767578163}
{"request_id": "704867d1-b534-4de0-b2d7-d92eb3db2e16", "model_version": "churn_model_v1_20260105_015315.joblib", "prediction": 1, "probability": 0.501147, "latency_ms": 8.308, "features": {"tenure_months": 18, "num_complaints": 1, "avg_session_minutes": 60.0, "plan_type": "standard", "region": "AF"}, "ts": 1767578271}
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01>
```

4. Arrêtez et supprimez le conteneur une fois les tests terminés :

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker stop churn-api-demo
churn-api-demo
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker rm churn-api-demo
churn-api-demo
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01>
```

Étape 12 : Orchestration locale avec Docker Compose

```
docker-compose.yml U X  requirements.txt U  Dockerfile U  prediction U
1  version: "3.9"
2
3  services:
4    churn-api:
5      build: .
6      image: churn-api:latest
7      container_name: churn-api-compose
8      ports:
9        - "8000:8000"
10     environment:
11       LOG_LEVEL: "info"
12     # Optionnel : monter les logs sur l'hôte pour inspection
13     volumes:
14       - ./logs:/app/logs
```

Étape 13 : Démarrer l'API via Docker Compose

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker compose up
time="2026-01-05T12:19:29+01:00" level=warning msg="C:\\Users\\anoua\\projects\\MLOPS\\mlops-lab-01\\docker
-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential
confusion"
[+] Running 2/2
 ✓ Network mlops-lab-01_default Created 0.1s
 ✓ Container churn-api-compose Created 0.2s
Attaching to churn-api-compose
churn-api-compose | INFO: Started server process [1]
churn-api-compose | INFO: Waiting for application startup.
churn-api-compose | INFO: Application startup complete.
churn-api-compose | INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

Test get

```

PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> curl http://localhost:8000/health
StatusCode      : 200
StatusDescription : OK
Content         : {"status":"ok","current_model":"churn_model_v1_20260105_015315.joblib"}
RawContent      : HTTP/1.1 200 OK
                  Content-Length: 71
                  Content-Type: application/json
                  Date: Mon, 05 Jan 2026 11:19:41 GMT
                  Server: uvicorn

                  {"status":"ok","current_model":"churn_model_v1_20260105_015315.joblib"}
Forms          : {}
Headers        : {[Content-Length, 71], [Content-Type, application/json], [Date, Mon, 05 Jan 2026
                  11:19:41 GMT], [Server, uvicorn]}
Images         : {}
InputFields    : {}
Links          : {}
ParsedHtml     : mshtml.HTMLDocumentClass
RawContentLength : 71

```

test post

The screenshot shows a REST client interface with a POST request to `http://localhost:8000/predict`. The request body is a JSON object with the following fields:

```

{
  "tenure_months": 3,
  "num_complaints": 5,
  "avg_session_minutes": 10.0,
  "plan_type": "basic",
  "region": "NA",
  "request_id": "client-high-risk"
}

```

The response body is also shown, containing the following fields:

```

{
  "request_id": "client-high-risk",
  "model_version": "churn_model_v1_20260105_015315.joblib",
  "prediction": 1,
  "probability": 0.961668,
  "latency_ms": 21.967,
  "features": {
    "tenure_months": 3,
    "num_complaints": 5,
    "avg_session_minutes": 10.0,
    "plan_type": "basic",
    "region": "NA"
  },
  "ts": 1767613360
}

```

stopping


```

churn-api-compose | INFO:      172.18.0.1:56392 - "GET /health HTTP/1.1" 200 OK
churn-api-compose | 2026-01-05 11:42:38,851 - src.api - INFO - [client-high-risk] - Nouvelle requête de pr
édiction reçue
churn-api-compose | 2026-01-05 11:42:40,940 - src.api - INFO - [client-high-risk] - Prédiction complétée :
pred=1, proba=0.961668, latency=21.967ms
churn-api-compose | INFO:      172.18.0.1:54292 - "POST /predict HTTP/1.1" 200 OK
Gracefully Stopping... press Ctrl+C again to force
Container churn-api-compose Stopping
Container churn-api-compose Stopped

```

Étape 14 : lancer les services en arrière-plan et observer les logs

Instructions :

Lancez les services en mode détaché

```

(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker compose up -d
time="2026-01-05T12:44:36+01:00" level=warning msg="C:\\Users\\anoua\\projects\\MLOPS\\mlops-lab-01\\docker
-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential
confusion"
[+] Running 1/1
✓ Container churn-api-compose Started                                0.7s

```

Vérifiez les conteneurs en cours d'exécution :

```

(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
e9ef4c344838   churn-api:latest "uvicorn src.api:app..." 27 minutes ago Up 2 minutes   0.0.0.0:8000->8000/tcp, [::]:8000->8000/tcp
churn-api-compose
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01>

```

Affichez les logs du service :

```

(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker compose logs -f churn-api
-churn-api-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential
confusion"
churn-api-compose | INFO:      Started server process [1]
churn-api-compose | INFO:      Waiting for application startup.
churn-api-compose | INFO:      Application startup complete.
churn-api-compose | INFO:      Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
churn-api-compose | INFO:      172.18.0.1:56392 - "GET /health HTTP/1.1" 200 OK
churn-api-compose | 2026-01-05 11:42:38,851 - src.api - INFO - [client-high-risk] - Nouvelle requête de pr
édiction reçue
churn-api-compose | 2026-01-05 11:42:40,940 - src.api - INFO - [client-high-risk] - Prédiction complétée :
pred=1, proba=0.961668, latency=21.967ms
churn-api-compose | INFO:      172.18.0.1:54292 - "POST /predict HTTP/1.1" 200 OK
churn-api-compose | INFO:      Shutting down
churn-api-compose | INFO:      Waiting for application shutdown.
churn-api-compose | INFO:      Application shutdown complete.
churn-api-compose | INFO:      Finished server process [1]
churn-api-compose | INFO:      Started server process [1]
churn-api-compose | INFO:      Waiting for application startup.
churn-api-compose | INFO:      Application startup complete.
churn-api-compose | INFO:      Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)

```

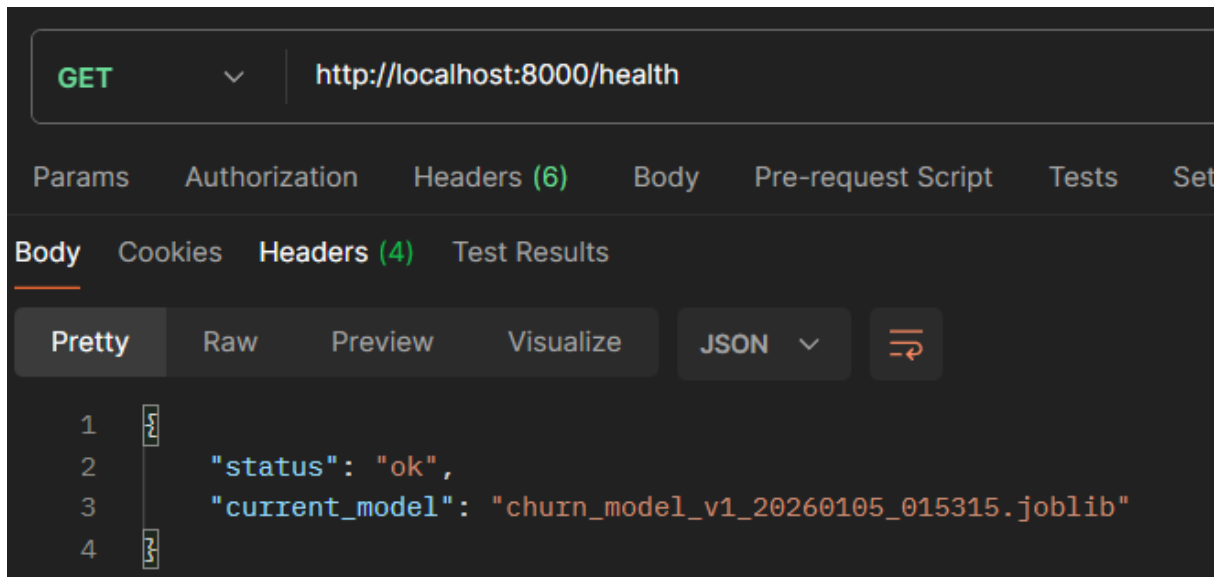
Testez `/health` et `/predict` pendant que les logs défilent. Arrêtez les services :

The screenshot shows a REST client interface with a POST request to `http://localhost:8000/predict`. The request body is a JSON object with the following fields:

```
{
  "tenure_months": 6,
  "num_complaints": 12,
  "avg_session_minutes": 2.0,
  "plan_type": "basic",
  "region": "AF",
  "request_id": "client-2222"
}
```

The response body is a JSON object with the following fields:

```
{
  "request_id": "client-2222",
  "model_version": "churn_model_v1_20260105_015315.joblib",
  "prediction": 1,
  "probability": 0.997941,
  "latency_ms": 20.182,
  "features": {
    "tenure_months": 6,
    "num_complaints": 12,
    "avg_session_minutes": 2.0,
    "plan_type": "basic",
    "region": "AF"
  },
  "ts": 1767613606
}
```



Arrêtez les services :

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> docker compose down
time="2026-01-05T12:48:35+01:00" level=warning msg="C:\\Users\\anoua\\projects\\MLOPS\\mlops-lab-01\\docker-
-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential
confusion"
[+] Running 2/2
 ✓ Container churn-api-compose Removed 1.1s
 ✓ Network mlops-lab-01_default Removed 0.4s
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01>
```

Étape 15 : lier Docker Compose au reste du cours (Git + DVC)

Assurez-vous que :

- le projet `mlops-lab-01` est versionné avec Git (lab Git),
- les données et modèles lourds sont suivis par DVC (lab DVC),
- l'API est conteneurisée via Docker (lab Docker).

```
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> git add .
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01> git commit -m "feat: ajout conteneurisation Docker de l'API churn"
[main 4ef1778] feat: ajout conteneurisation Docker de l'API churn
7 files changed, 690 insertions(+), 151 deletions(-)
create mode 100644 Dockerfile
create mode 100644 docker-compose.yml
create mode 100644 requirements.txt
(venv_mlops) PS C:\Users\anoua\projects\MLOPS\mlops-lab-01>
```

Notez dans ton cours que l'on a maintenant :

- **MLOps local : pipeline + API + monitoring**
- **Git : versionnement du code et de la structure**
- **DVC : versionnement des données / modèles (lab suivant)**
- **Docker / Compose : déploiement reproductible de l'API**