



Université Chouaib Doukkali
Ecole Nationale des Sciences Appliquées d'El Jadida
Département Télécommunications, Réseaux et Informatique
Module : MLops



TP

Filière : **SDIA**
Niveau : **2^{ème} Année**

Lab 6 :

Déploiement K8s d'un système MLops Churn

Réalisé Par :
Wassima RAICHI

Année Universitaire : 2025/2026

Objectif du lab 6 :

L'objectif principal de ce Lab est d'industrialiser un système de Machine Learning dédié à la prédiction du churn, en appliquant les bonnes pratiques MLOps et les principes de déploiement sur Kubernetes.

Plus précisément, ce Lab vise à :

- Conteneuriser une API ML FastAPI avec Docker,
- Versionner l'image Docker et la charger dans Minikube,
- Déployer l'API sur Kubernetes à l'aide de manifests YAML,
- Mettre en place des Services (NodePort), ConfigMaps et Secrets,
- Ajouter des probes de santé (liveness, readiness, startup) pour la fiabilité,
- Appliquer une NetworkPolicy pour contrôler la communication réseau interne,
- Tester l'API déployée, exécuter des prédictions et déclencher la détection de drift.

Introduction :

Dans un contexte industriel, le Machine Learning ne se limite pas à entraîner un modèle. La véritable difficulté réside dans son déploiement, sa maintenance, sa scalabilité et son monitoring en production. Ce Lab s'inscrit dans la continuité de la démarche MLOps (Machine Learning Operations) qui vise à intégrer les modèles dans un pipeline complet permettant leur exploitation à grande échelle.

Dans ce Lab 6, nous abordons la mise en production d'un système de prédiction du churn à travers une architecture conteneurisée et orchestrée par Kubernetes. À partir d'un modèle ML pré-entraîné encapsulé dans une API FastAPI, nous construisons une image Docker versionnée, puis nous la déployons sur Minikube via des manifests Kubernetes comprenant Deployment, Service, ConfigMap, Secret et NetworkPolicy.

Nous introduisons également des éléments essentiels pour une application “production-ready” tels que les probes (liveness, readiness, startup), la configuration externe, la gestion de secrets, l'ouverture contrôlée du réseau et le monitoring du modèle via la détection de drift. Ce Lab illustre ainsi les fondamentaux d'un déploiement MLOps robuste et reproductible.

Étape 1 : Préparer l'environnement Kubernetes

1. Vérification de Minikube

```
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> minikube version
minikube version: v1.37.0
commit: 65318f4cff9c12cc87ec9eb8f4cdd57b25047f3
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

Minikube est bien installé, et la version est compatible avec le lab.

2. Démarrage du cluster Kubernetes local

```
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> minikube start --driver=docker --kubernetes-version=v1.28.3
* minikube v1.37.0 sur Microsoft Windows 11 Pro 10.0.26200.7462 Build 26200.7462
* Kubernetes 1.34.0 est désormais disponible. Si vous souhaitez effectuer une mise à niveau, spécifiez : --kubernetes-version=v1.34.0
* Utilisation du pilote docker basé sur le profil existant
* Démarrage du nœud "minikube" primary control-plane dans le cluster "minikube"
* Extraction de l'image de base v0.0.48...
* Mise à jour du container docker en marche "minikube" ...
! Échec de la connexion à https://registry.k8s.io/ depuis l'intérieur du minikube container
* Pour extraire de nouvelles images externes, vous devrez peut-être configurer un proxy : https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
* Préparation de Kubernetes v1.28.3 sur Docker 28.4.0...
* Vérification des composants Kubernetes...
- Utilisation de l'image gcr.io/k8s-minikube/storage-provisioner:v5
* Modules activés: default-storageclass, storage-provisioner

! C:\Program Files\Docker\Docker\resources\bin\kubectl.exe est la version 1.32.2, qui peut comporter des incompatibilités avec Kubernetes 1.28.3.
- Vous voulez kubectl v1.28.3 ? Essayez 'minikube kubectl -- get pods -A'
* Terminé ! kubectl est maintenant configuré pour utiliser "minikube" cluster et espace de noms "default" par défaut.
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

Le cluster Kubernetes local est démarré et fonctionnel.

3. Vérification des nodes

```
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl get nodes
NAME      STATUS    ROLES      AGE     VERSION
minikube  Ready     control-plane   23h    v1.28.3
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

Le cluster a 1 node, il est en Ready, donc prêt à exécuter des workloads.

4. Création du namespace dédié

```
minikube ready control-plane 24s v1.28.3
PS C:\Users\hp> kubectl create namespace churn-mllops
namespace/churn-mllops created
PS C:\Users\hp> |
```

Le namespace dédié a été créé avec succès.

```
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl config set-context --current --namespace=churn-mllops
Context "minikube" modified.
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

Le contexte actif pointe maintenant vers churn-mllops, ce qui évite d'ajouter -n à chaque fois.

5. Vérification des namespaces

```
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl get ns
NAME        STATUS   AGE
churn-mlops Active   23h
default     Active   23h
kube-node-lease Active   23h
kube-public  Active   23h
kube-system  Active   23h
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

Le namespace `churn-mlops` est bien présent et actif.

```
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
churn-api-79797775cb-jsz8x  1/1     Running   2 (5m11s ago)  22h
churn-api-79797775cb-vwmqg  1/1     Running   2 (5m10s ago)  22h
churn-api-8dbffc749-hq25n   0/1     CrashLoopBackOff  15 (48s ago)  21h
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

⇒

Lors de cette étape, l'environnement Kubernetes a été initialisé correctement. Minikube a démarré avec Docker comme driver, le cluster possède un node en état Ready, et un namespace dédié `churn-mlops` a été créé et configuré comme contexte actif. Le cluster est donc prêt à recevoir des déploiements pour la suite du Lab.

Étape 2 : Préparer l'image Docker de l'API churn

1. Vérification de la version de Python

```
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> python --version
Python 3.12.6
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

2. Création de l'environnement virtuel

```
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> python -m venv venv_mllops
PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> .\venv_mllops\Scripts\activate
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

3. Mise à jour de pip

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> python -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (24.2)
Collecting pip
  Using cached pip-25.3-py3-none-any.whl.metadata (4.7 kB)
  Using cached pip-25.3-py3-none-any.whl (1.8 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.2
    Uninstalling pip-24.2:
      Successfully uninstalled pip-24.2
Successfully installed pip-25.3
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

4. Création du fichier requirements.txt

```

PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> pip install -r requirements.txt
Requirement already satisfied: fastapi in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from -r requirements.txt (line 1))
(0.124.4)
Requirement already satisfied: pydantic in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from -r requirements.txt (line 3))
(2.12.5)
Requirement already satisfied: scikit-learn==1.8.0 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from -r requirements.txt (line 4)) (1.8.0)
Collecting pandas==2.2.3 (from -r requirements.txt (line 5))
Using cached pandas-2.2.3-cp312-cp312-win_amd64.whl.metadata (19 kB)
Collecting numpy==2.1.3 (from -r requirements.txt (line 6))
Using cached numpy-2.1.3-cp312-cp312-win_amd64.whl.metadata (60 kB)
Collecting joblib==1.4.2 (from -r requirements.txt (line 7))
Using cached joblib-1.4.2-py3-none-any.whl.metadata (5.4 kB)
Collecting python-dotenv (from -r requirements.txt (line 8))
Using cached python_dotenv-1.2.1-py3-none-any.whl.metadata (25 kB)
Collecting loguru (from -r requirements.txt (line 9))
Downloading loguru-0.7.3-py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: uvicorn[standard] in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from -r requirements.txt (line 2)) (0.38.0)
Requirement already satisfied: scipy>=1.10.0 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from scikit-learn==1.8.0->-r requirements.txt (line 4)) (1.16.3)
Requirement already satisfied: threadpoolctl=3.2.0 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from scikit-learn==1.8.0->-r requirements.txt (line 4)) (3.6.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from pandas==2.2.3->-r requirements.txt (line 5)) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from pandas==2.2.3->-r requirements.txt (line 5)) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from pandas==2.2.3->-r requirements.txt (line 5)) (2025.3)
Requirement already satisfied: starlette<0.51.0,>=0.40.0 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from fastapi->-r requirements.txt (line 1)) (0.50.0)
Requirement already satisfied: typing-extensions>=4.8.0 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from fastapi->-r requirements.txt (line 1)) (4.15.0)
Requirement already satisfied: annotated-doc>=0.0.2 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from fastapi->-r requirements.txt (line 1)) (0.0.4)
Requirement already satisfied: annotated-types>=0.6.0 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from pydantic->-r requirements.txt (line 3)) (0.7.0)
Requirement already satisfied: pydantic-core==2.41.5 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from pydantic->-r requirements.txt (line 3)) (2.41.5)
Requirement already satisfied: typing-inspection>=0.4.2 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from pydantic->-r requirements.txt (line 3)) (0.4.2)

```

Installation des dépendances

```

(vENV_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> pip install -r requirements.txt
Requirement already satisfied: fastapi in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from -r requirements.txt (line 1))
(0.124.4)
Requirement already satisfied: pydantic in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from -r requirements.txt (line 3))
(2.12.5)
Requirement already satisfied: scikit-learn==1.8.0 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from -r requirements.txt (line 4)) (1.8.0)
Collecting pandas==2.2.3 (from -r requirements.txt (line 5))
Using cached pandas-2.2.3-cp312-cp312-win_amd64.whl.metadata (19 kB)
Collecting numpy==2.1.3 (from -r requirements.txt (line 6))
Using cached numpy-2.1.3-cp312-cp312-win_amd64.whl.metadata (60 kB)
Collecting joblib==1.4.2 (from -r requirements.txt (line 7))
Using cached joblib-1.4.2-py3-none-any.whl.metadata (5.4 kB)
Collecting python-dotenv (from -r requirements.txt (line 8))
Using cached python_dotenv-1.2.1-py3-none-any.whl.metadata (25 kB)
Collecting loguru (from -r requirements.txt (line 9))
Downloading loguru-0.7.3-py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: uvicorn[standard] in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from -r requirements.txt (line 2)) (0.38.0)
Requirement already satisfied: scipy>=1.10.0 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from scikit-learn==1.8.0->-r requirements.txt (line 4)) (1.16.3)
Requirement already satisfied: threadpoolctl=3.2.0 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from scikit-learn==1.8.0->-r requirements.txt (line 4)) (3.6.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from pandas==2.2.3->-r requirements.txt (line 5)) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from pandas==2.2.3->-r requirements.txt (line 5)) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from pandas==2.2.3->-r requirements.txt (line 5)) (2025.3)
Requirement already satisfied: starlette<0.51.0,>=0.40.0 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from fastapi->-r requirements.txt (line 1)) (0.50.0)
Requirement already satisfied: typing-extensions>=4.8.0 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from fastapi->-r requirements.txt (line 1)) (4.15.0)
Requirement already satisfied: annotated-doc>=0.0.2 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from fastapi->-r requirements.txt (line 1)) (0.0.4)
Requirement already satisfied: annotated-types>=0.6.0 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from pydantic->-r requirements.txt (line 3)) (0.7.0)
Requirement already satisfied: pydantic-core==2.41.5 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from pydantic->-r requirements.txt (line 3)) (2.41.5)
Requirement already satisfied: typing-inspection>=0.4.2 in c:\users\hp\onedrive\bureau\tp mllops\mllops-lab-01\venv_mllops\lib\site-packages (from pydantic->-r requirements.txt (line 3)) (0.4.2)

```

Étape 3 : Créer le dossier des manifests Kubernetes

1. Crédit du dossier k8s

```

(vENV_mllops) PS C:\Users\hp> cd "C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01"
(vENV_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> New-Item -ItemType Directory -Name k8s

Répertoire : C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01

Mode          LastWriteTime    Length Name
----          <-----          ----  --
d---  10/01/2026   14:14           k8s

```

2. Vérification du contenu du projet

(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> Get-ChildItem			
Répertoire : C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01			
Mode	LastWriteTime	Length	Name
--	--	--	--
da---l	27/12/2025 10:33		.dvc
da---l	27/12/2025 12:03		.github
da---l	05/01/2026 16:37		.venv
da---l	05/01/2026 00:23		data
da---l	27/12/2025 10:45		dvc_storage
d----	10/01/2026 14:14		k8s
da---l	14/12/2025 17:56		logs
da---l	05/01/2026 16:53		models
da---l	05/01/2026 00:23		registry
da---l	05/01/2026 00:22		reports
da---l	14/12/2025 19:44		src
da---l	14/12/2025 16:40		venv_mllops
-a---l	27/12/2025 10:12	142	.dvcignore
-a---l	27/12/2025 11:25	59	.gitignore
-a---l	05/01/2026 17:19	241	docker-compose.yml
-a---l	05/01/2026 16:46	464	Dockerfile
-a---l	05/01/2026 00:23	1456	dvc.lock
-a---l	05/01/2026 00:16	493	dvc.yaml
-a---l	14/12/2025 18:19	1883844	Lab 1 Du notebook au mini-système production-ready - Wassima RAICHI.pdf
-a---l	14/12/2025 19:47	1389609	Lab 2 Code Source management - Wassima RAICHI.pdf
-a---l	05/01/2026 17:10	135	payload.json
-a---l	14/12/2025 18:38	659	README.md
-a---l	05/01/2026 16:44	98	requirements.txt



Durant cette étape, un dossier k8s/ a été créé à la racine du projet pour regrouper l'ensemble des fichiers YAML nécessaires au déploiement Kubernetes. Cette organisation permet de séparer clairement :

- le code (src/)
- les artefacts ML (models/, registry/)
- les données (data/)
- les configurations de déploiement (k8s/)

Étape 4 : Construire l'image Docker (tag versionné)

1. Exécution du script de training

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> python ./src\train.py
C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01\src\train.py:195: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
    timestamp = datetime.utcnow().strftime("%Y%m%d_%H%M%S")
[OK] Alias stable : C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01\models\model.joblib
[METRICS] {
    "accuracy": 0.6433333333333333,
    "precision": 0.6687898089171974,
    "recall": 0.65625,
    "f1": 0.6624605678233438,
    "baseline_f1": 0.0
}
[OK] Modèle sauvegardé : C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01\models\churn_model_v1_20260111_140638.joblib
[DEPLOY] Refusé : F1 insuffisante ou baseline non battue.
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

Construction de l'image Docker

```

DEPLOY RELEASE : r1 insuffisance de baseline non battue.
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> docker build -t churn-api:v1 .
[+] Building 320.1s (10/10) FINISHED                                            docker:desktop-linux
=> [internal] load build definition from Dockerfile                         0.1s
=> => transferring dockerfile: 503B                                         0.0s
=> [internal] load metadata for docker.io/library/python:3.10-slim          1.7s
=> [internal] load .dockerignore                                           0.0s
=> => transferring context: 2B                                             0.0s
=> [1/5] FROM docker.io/library/python:3.10-slim@sha256:7b68a5fa7cf0d20b4cedb1dc9a134fdd394fe27edbc4c2519756c91d  0.0s
=> [internal] load build context                                         172.8s
=> => transferring context: 2.97GB                                         172.6s
=> CACHED [2/5] WORKDIR /app                                              0.0s
=> [3/5] COPY requirements.txt .                                         0.6s
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt              70.5s
=> [5/5] COPY . .                                                       46.2s
=> exporting to image                                                 27.8s
=> => exporting layers                                                 27.7s
=> => writing image sha256:5693a5fae6c1744f4afb608ec36c64f81dcb6a1da99ff5082fb0b8bb6234e833  0.0s
=> => naming to docker.io/library/churn-api:v1                           0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/3jcx2l8nl30u40x6euzescvur
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>

```

Docker :

- **lit le Dockerfile**
- **Copie les fichiers dans l'image**
- **Installe les dépendances via pip install**
- **Ajoute le modèle, le code, etc.**
- **Produit l'image churn-api:v1**

1. Vérification de l'image construite

```

(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> docker images | Select-String churn-api
churn-api                               v1           5693a5fae6c1   2 minutes ago   3.92GB
churn-api                               latest       b928002844bc   5 days ago    1.27GB

(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>

```

L'image Docker churn-api:v1 a été construite à partir du Dockerfile et est maintenant disponible localement avec un tag versionné.

Ce tag permettra de gérer différentes versions du modèle/API dans Kubernetes et d'éviter les collisions liées aux images latest.

Étape 5 : Charger explicitement l'image dans Minikube

1. Vérifier l'état de Minikube

```

(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> minikube status
minikube
type: Control Plane
host: Running
kublet: Running
apiserver: Running
kubeconfig: Configured

(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>

```

Le cluster est démarré et opérationnel, prêt à recevoir l'image.

2. Sauvegarder l'image Docker

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> docker save churn-api:v1 -o churn-api_v1.tar  
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

Cela exporte l'image locale churn-api:v1 dans un fichier churn-api_v1.tar.

3. Charger l'image dans Minikube

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> minikube image load churn-api_v1.tar  
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

Cette commande prend le .tar et l'ajoute dans le cache Docker utilisé par Minikube.

4. Vérifier l'import dans Minikube

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> minikube image ls | Select-String churn-api  
docker.io/library/churn-api:v1  
  
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

L'image a bien été chargée dans le runtime Docker de Minikube.

L'image Docker locale churn-api:v1 a été exportée en fichier (.tar) puis chargée dans le registre interne de Minikube via la commande minikube image load. La vérification avec minikube image ls a confirmé que l'image est disponible dans l'environnement Kubernetes.

Cette procédure permet de déployer des images locales sans les publier sur Docker Hub, ce qui est utile en contexte académique ou réseau privé.

Étape 6 : Deployment Kubernetes pour l'API churn

1. Crédation du manifest

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> New-Item k8s/deployment.yaml  
  
Répertoire : C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01\k8s  
  
Mode          LastWriteTime    Length Name  
----          -----        -- deployment.yaml  
-a---  10/01/2026   14:30           0 deployment.yaml  
  
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> notepad k8s/deployment.yaml  
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

2. Application du manifest

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl apply -f k8s/deployment.yaml  
deployment.apps/churn-api created  
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

Kubernetes a bien créé le Deployment churn-api

3. Suivi du rollout

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl rollout status deployment churn-api
deployment "churn-api" successfully rolled out
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

Le déploiement de la nouvelle version est terminé avec succès (K8s a créé les pods et les considère prêts au niveau du Deployment).

4. Vérification des Pods

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl get pods -l app=churn-api -o wide
NAME          READY   STATUS    RESTARTS   AGE      IP           NODE   NOMINATED NODE   READINESS
GATES
churn-api-79797775cb-jsz8x  1/1     Running   2 (139m ago)  24h     10.244.0.20  minikube <none>        <none>
churn-api-79797775cb-vwmag  1/1     Running   2 (139m ago)  24h     10.244.0.18  minikube <none>        <none>
churn-api-8dbfffc749-hq25n  0/1     Running   39 (66s ago)  24h     10.244.0.19  minikube <none>        <none>
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

- Les deux premiers pods sont :
 - READY 1/1
 - STATUS Running
 - Ils redémarré 2 fois (probablement lors de changements précédents de config/probes) mais maintenant tout va bien
- Le troisième pod est en CrashLoopBackOff :
 - READY 0/1
 - RESTARTS élevé (39) → ce pod échoue au démarrage (erreur dans le conteneur ou probes qui échouent)

Ce troisième pod est probablement un ancien ReplicaSet (ancienne version du Deployment) qui n'a pas été complètement nettoyé ou qui reste suite à un test de probes.

Mais ce qui est important pour l'étape 6 : le Deployment a bien créé et gère des pods fonctionnels.

Étape 7 : Exposer l'API via un Service NodePort

1. Crédation du fichier service.yaml

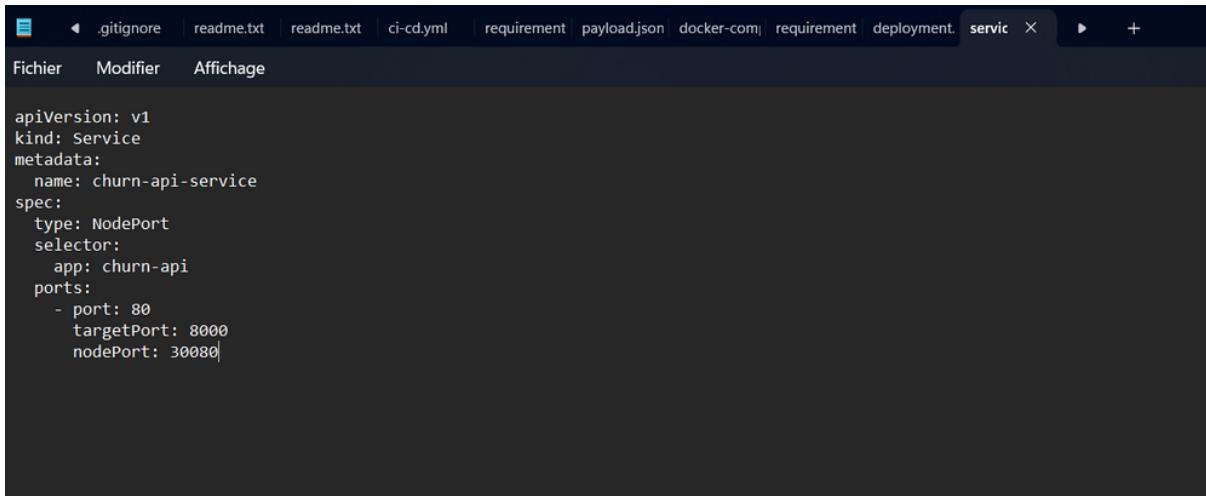
```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> New-Item k8s/service.yaml
```

```
Répertoire : C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01\k8s

Mode          LastWriteTime        Length Name
----          -----          ----  --
-a---  10/01/2026 14:35                 0 service.yaml
```

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> notepad k8s/service.yaml
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl apply -f k8s/service.yaml
service/churn-api-service created
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```



```
apiVersion: v1
kind: Service
metadata:
  name: churn-api-service
spec:
  type: NodePort
  selector:
    app: churn-api
  ports:
    - port: 80
      targetPort: 8000
      nodePort: 30080
```

Ce manifest dit à Kubernetes : “Envoie tout le trafic entrant sur 30080 vers les Pods app=churn-api sur leur port 8000”.

2. Vérification des Services

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl get svc churn-api-service
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)        AGE
churn-api-service  NodePort  10.108.81.153 <none>       80:30080/TCP  73s
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

3. Analyse détaillée

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl describe svc churn-api-service
Name:           churn-api-service
Namespace:      churn-mllops
Labels:          <none>
Annotations:    <none>
Selector:       app=churn-api
Type:           NodePort
IP Family Policy: SingleStack
IP Families:   IPv4
IP:             10.108.81.153
IPs:            10.108.81.153
Port:           <unset>  80/TCP
TargetPort:     8000/TCP
NodePort:       <unset>  30080/TCP
Endpoints:      10.244.0.20:8000,10.244.0.18:8000
Session Affinity: None
External Traffic Policy: Cluster
Internal Traffic Policy: Cluster
Events:         <none>
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

4. Accès à l'API via port-forward

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl port-forward svc/churn-api-service 30080:80
Forwarding from 127.0.0.1:30080 -> 8000
Forwarding from [::1]:30080 -> 8000
Handling connection for 30080
Handling connection for 30080
|
```

Exceptions
HTTPException - 500 si le modèle ne peut pas être chargé, - 400 si une erreur survient lors de la prédiction.

Parameters
No parameters

Request body **required**

application/json

```
{
  "tenure_months": 48,
  "num_complaints": 0,
  "avg_session_minutes": 60,
  "plan_type": "premium",
  "region": "EU",
  "request_id": "req-safe"
}
```

Execute Clear

Body Cookies Headers (4) Test Results Status: 200 OK Time: 2.36 s Size: 395 B Save Response

```

1
2   "request_id": "req-safe",
3   "model_version": "churn_model_v1_20260110_003135.joblib",
4   "prediction": 0,
5   "probability": 0.139973,
6   "latency_ms": 20.335,
7   "features": [
8     {
9       "tenure_months": 48,
10      "num_complaints": 0,
11      "avg_session_minutes": 60.0,
12      "plan_type": "premium",
13      "region": "EU"
14    }
15  ],
16  "status": "OK"
17 }
```

À cette étape, un Service Kubernetes de type NodePort a été créé afin d'exposer l'API FastAPI déployée dans le cluster. Le fichier service.yaml définit un selector vers les Pods app=churn-api, et configure la redirection du port externe 30080 vers le targetPort 8000 utilisé par FastAPI dans les conteneurs. Après application du manifest, la commande kubectl get svc a confirmé la création du service. L'accès à l'API via port-forward a permis de valider la connectivité à l'endpoint /health. Cette étape permet de rendre l'API utilisable depuis l'extérieur du cluster Kubernetes.

Étape 8 : Injecter la configuration MLOps via ConfigMap

1. Création du fichier ConfigMap

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> New-Item k8s/configmap.yaml
```

```
Répertoire : C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01\k8s
```

Mode	LastWriteTime	Length	Name
-a---	10/01/2026 14:49	0	configmap.yaml

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: churn-config
data:
  MODEL_NAME: "churn_model_v1"
  LOG_LEVEL: "info"

Mode LastWriteTime Length Name
---- ----- ---- -
-a--- 10/01/2026 14:49 0 configmap.yaml
```

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> notepad k8s/configmap.yaml
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

Ce fichier stocke la configuration de l'API churn dans Kubernetes

2. Application du ConfigMap

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl apply -f k8s/configmap.yaml
configmap/churn-config created
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

Le ConfigMap est maintenant enregistré dans le cluster

3. Vérification de la création

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl get configmap churn-config
NAME      DATA   AGE
churn-config  2    43s
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

Le 2 confirme qu'il contient 2 clés (MODEL_NAME + LOG_LEVEL)

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl describe configmap churn-config
Name:      churn-config
Namespace: churn-mllops
Labels:    <none>
Annotations: <none>

Data
=====
LOG_LEVEL:
-----
info

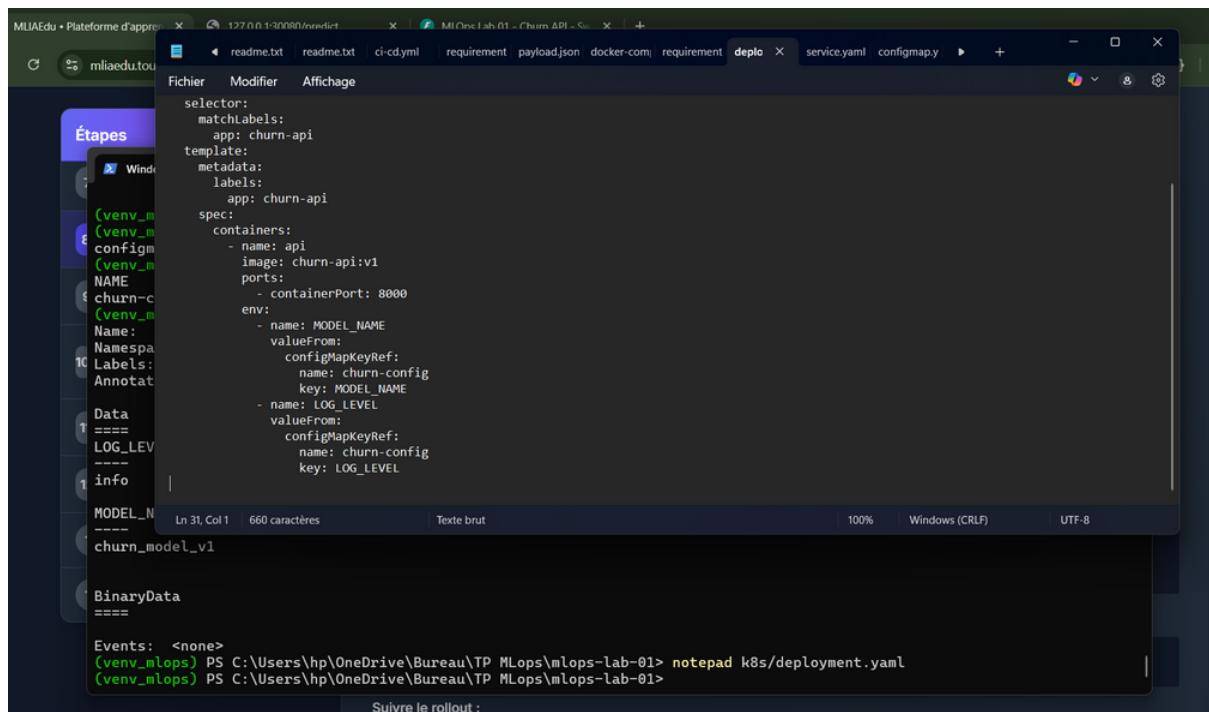
MODEL_NAME:
-----
churn_model_v1

BinaryData
=====

Events:  <none>
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

Le contenu est correct et lisible dans le cluster

4. Injection dans le Deployment



```
Etapes
Étapes
Fichier Modifier Affichage
selector:
  matchLabels:
    app: churn-api
template:
  metadata:
    labels:
      app: churn-api
  spec:
    containers:
      - name: api
        image: churn-api:v1
        ports:
          - containerPort: 8000
        env:
          - name: MODEL_NAME
            valueFrom:
              configMapKeyRef:
                name: churn-config
                key: MODEL_NAME
          - name: LOG_LEVEL
            valueFrom:
              configMapKeyRef:
                name: churn-config
                key: LOG_LEVEL
    MODEL_N
    LOGLEV
    info
    BinaryData
    Events: <none>
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> notepad k8s/deployment.yaml
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

Kubernetes injecte automatiquement les valeurs dans le conteneur au démarrage du Pod

5. Redéploiement du Deployment

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl apply -f k8s/deployment.yaml
deployment.apps/churn-api configured
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>

(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl rollout restart deployment churn-api
deployment.apps/churn-api restarted
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl rollout status deployment churn-api
deployment "churn-api" successfully rolled out
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

Les Pods ont bien été redémarrés avec la nouvelle configuration

6. Vérification dans un Pod

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl exec -it deploy/churn-api -- printenv MODEL_NAME
churn_model_v1
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl exec -it deploy/churn-api -- printenv LOG_LEVEL
info
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

Le ConfigMap est injecté dans les variables d'environnement du conteneur

Étape 9 : Gérer les secrets (MONITORING_TOKEN)

1. Encodage du secret en base64

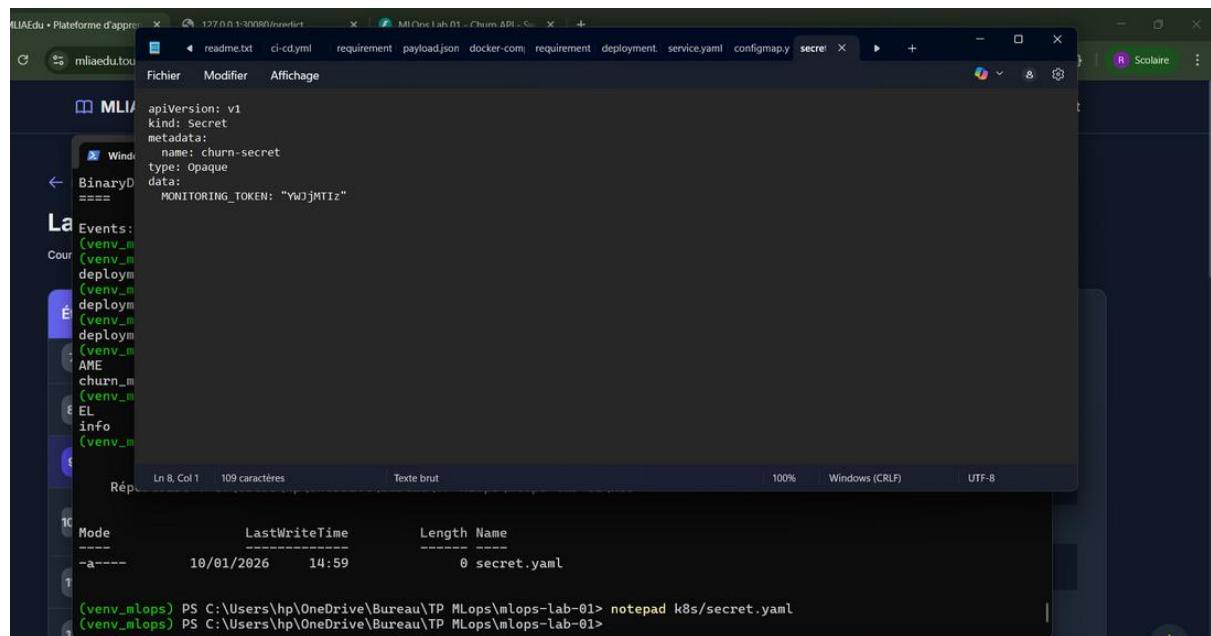
```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> [Convert]::ToBase64String([Text.Encoding]::UTF8.GetBytes("abc123"))
YwQjMTIz
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

2. Création du Secret Kubernetes

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> New-Item k8s/secret.yaml
```

Répertoire : C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01\k8s

Mode	LastWriteTime	Length	Name
-a---	10/01/2026 14:59	0	secret.yaml



On déclare un Secret churn-secret contenant 1 clé : MONITORING_TOKEN.

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl apply -f k8s/secret.yaml
secret/churn-secret created
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

3. Vérification du Secret

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl get secret churn-secret
NAME      TYPE      DATA   AGE
churn-secret  Opaque    1    26s
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl describe secret churn-secret
Name:          churn-secret
Namespace:    churn-mllops
Labels:        <none>
Annotations:  <none>

Type:  Opaque

Data
=====
MONITORING_TOKEN: 6 bytes
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

4. Injection du Secret dans le Deployment

```
Fichier Modifier Affichage
Étapes
Fichier Modifier Affichage
labels:
  app: churn-api
spec:
  containers:
    - name: api
      image: churn-api:v1
      ports:
        - containerPort: 8000
      env:
        - name: MODEL_NAME
          valueFrom:
            configMapKeyRef:
              name: churn-config
              key: MODEL_NAME
        - name: LOG_LEVEL
          valueFrom:
            configMapKeyRef:
              name: churn-config
              key: LOG_LEVEL
        - name: MONITORING_TOKEN
          valueFrom:
            secretKeyRef:
              name: churn-secret
              key: MONITORING_TOKEN
  ...
  annotations: <none>
  type: Opaque
  data
  =====
  MONITORING_TOKEN: 6 bytes
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> notepad k8s/deployment.yaml
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

Création d'une variable d'environnement MONITORING_TOKEN dans le conteneur, et va chercher sa valeur dans le secret churn-secret (clé MONITORING_TOKEN).

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl apply -f k8s/deployment.yaml
deployment.apps/churn-api configured
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
churn-api-79797775cb-jsz8x  1/1     Running   0          18s
churn-api-79797775cb-vwmqg  1/1     Running   0          20s
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

5. Vérification dans un Pod

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl exec -it deploy/churn-api -- printenv MONITORING_TOKEN
abc123
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```

Dans cette étape, un Secret Kubernetes nommé churn-secret a été créé pour stocker de manière sécurisée un token de monitoring (MONITORING_TOKEN). La valeur "abc123" a d'abord été encodée en base64, puis déclarée dans secret.yaml. Après application du manifest, le secret a été injecté dans le Deployment via la section env utilisant secretKeyRef. La commande kubectl exec ... printenv MONITORING_TOKEN a confirmé que le token est correctement disponible

dans le conteneur, tout en étant masqué dans les manifests YAML. Cette approche permet de séparer les données sensibles du code et de respecter les bonnes pratiques de sécurité en MLOps.

Étape 10 : Mise en place des endpoints de santé et des probes Kubernetes pour l'API Churn

1. Modification de src/api.py

```

status_code=503,
        detail="Aucun modèle courant. Lancer train.py (avec gate) d'abord."
    )
    name = CURRENT_MODEL_PATH.read_text(encoding="utf-8").strip()
    if not name:
        raise HTTPException(
            status_code=503,
            detail="current_model.txt vide."
        )
    return {
        "status": "ok",
        "current_model": name,
    }
@app.get("/ready")
def ready() -> dict[str, Any]:
    try:
        model_name = get_current_model_name()
        return {"status": "ready", "current_model": model_name}
    except Exception as exc:
        raise HTTPException(status_code=503, detail=str(exc))

```

La logique qui permettra à Kubernetes de savoir “Est-ce que je peux utiliser ce pod ? Oui / Non”

1. Reconstruction de l'image Docker

```

(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> docker build -t churn-api:v1 .
[+] Building 242.5s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 503B
=> [internal] load metadata for docker.io/library/python:3.10-slim
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 1.32GB
=> [1/5] FROM docker.io/library/python:3.10-slim@sha256:7b68a5fa7cf0d20b4cedb1dc9a134fdd394fe27edbc4c2519756c91d
=> CACHED [2/5] WORKDIR /app
=> CACHED [3/5] COPY requirements.txt .
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> [5/5] COPY .
=> exporting to image
=> => exporting layers
=> => writing image sha256:c6f065a97d3dd03fb1ad7b65cbd2f55ccb4529a11828436c3f4d9ddc5c39b310
=> => naming to docker.io/library/churn-api:v1
docker:desktop-linux
  0.1s
  0.0s
  2.0s
  0.0s
  0.0s
  0.0s
  114.6s
  114.3s
  0.0s
  0.0s
  0.0s
  84.2s
  41.1s
  41.0s
  0.0s
  0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/c73fbrd7ceu4jl8cfjywufzn9
(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>

```

La nouvelle version du code (avec les endpoints) est bien intégrée dans churn-api:v1.

2. Export + chargement dans Minikube

```

(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> docker save churn-api:v1 -o churn-api_v1.tar
(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>

```

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> minikube image load churn-api_v1.tar
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

Dans cette étape, trois endpoints de santé (/health, /startup, /ready) ont été ajoutés à l'API FastAPI afin de permettre à Kubernetes de moniterer l'état de l'application et du modèle. L'endpoint /health vérifie que l'API est vivante, /startup s'assure que le registry et le modèle courant sont correctement initialisés, et /ready confirme que le modèle est prêt à servir des requêtes. Après modification de src/api.py, l'image Docker churn-api:v1 a été reconstruite puis exportée et chargée dans Minikube. Cette étape prépare l'activation des probes Kubernetes (liveness, readiness, startup) dans le Deployment.

Étape 11 : Ajouter les probes (liveness / readiness / startup)

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> notepad k8s/deployment.yaml
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl apply -f k8s/deployment.yaml
deployment.apps/churn-api configured
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

```

Fichier Modifier Affichage
image: churn-api:v1
ports:
- containerPort: 8000
env:
- name: MODEL_NAME
  valueFrom:
    configMapKeyRef:
      name: churn-config
      key: MODEL_NAME
- name: LOG_LEVEL
  valueFrom:
    configMapKeyRef:
      name: churn-config
      key: LOG_LEVEL
- name: MONITORING_TOKEN
  valueFrom:
    secretKeyRef:
      name: churn-secret
      key: MONITORING_TOKEN
livenessProbe:
  httpGet:
    path: /health
    port: 8000
  initialDelaySeconds: 10
  periodSeconds: 30
readinessProbe:
  httpGet:
    path: /ready
    port: 8000
  initialDelaySeconds: 5
  periodSeconds: 10
startupProbe:
  httpGet:
    path: /startup
    port: 8000
  failureThreshold: 30
  periodSeconds: 5

```

livenessProbe : si /health ne répond plus → Kubernetes **redémarre** le conteneur.

readinessProbe : tant que /ready n'est pas OK → le pod n'est **pas ajouté** au Service (pas de trafic).

startupProbe : donne du temps à l'appli pour démarrer ; si /startup échoue trop souvent → K8s considère que le démarrage a échoué.

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl describe pod -l app=churn-api
Name:           churn-api-79797775cb-jsz8x
Namespace:      churn-mllops
Priority:       0
Service Account: default
Node:           minikube/192.168.49.2
Start Time:     Sat, 10 Jan 2026 15:05:01 +0100
Labels:         app=churn-api
Annotations:    pod-template-hash=79797775cb
Status:         Running
IP:             10.244.0.10
IPs:
  IP:          10.244.0.10
Controlled By: ReplicaSet/churn-api-79797775cb
Containers:
  api:
    Container ID: docker://7b310ab2e37766a8702b216f9720a83ad81f56b6d79de65be0b4fe4ef570b7e8
    Image:         churn-api:v1
    Image ID:     docker://sha256:852fd96e813161ec6a2f6acbb717d7c71f08402fb3067921a9f18393b5d40e0a
    Port:          8000/TCP
    Host Port:    0/TCP
    State:        Running
      Started:   Sat, 10 Jan 2026 15:05:02 +0100
    Ready:        True
    Restart Count: 0
    Environment:
      MODEL_NAME: <set to the key 'MODEL_NAME' of config map 'churn-config'> Optional: false
      LOG_LEVEL:  <set to the key 'LOG_LEVEL' of config map 'churn-config'> Optional: false
      MONITORING_TOKEN: <set to the key 'MONITORING_TOKEN' in secret 'churn-secret'> Optional: false
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-gxp6s (ro)
Conditions:
  Type        Status
  Initialized  True
  Ready        True
  ContainersReady  True
  PodScheduled  True
Volumes:
  kube-api-access-gxp6s:
    Type:       Projected (a volume that contains injected data from multiple sources)

```

```
Type:          Projected (a volume that contains injected data from multiple sources)
TokenExpirationSeconds: 3607
ConfigMapName:  kube-root-ca.crt
ConfigMapOptional: <nil>
DownwardAPI:   true
QoS Class:    BestEffort
Node-Selectors: <none>
Tolerations:   node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type  Reason  Age   From            Message
  ----  ----   --   ----            -----
  Normal Scheduled  26m  default-scheduler  Successfully assigned churn-mllops/churn-api-79797775cb-jsz8x to minikube
  Normal Pulled   26m  kubelet         Container image "churn-api:v1" already present on machine
  Normal Created   26m  kubelet         Created container api
  Normal Started   26m  kubelet         Started container api
```

```
Name:           churn-api-79797775cb-vwmqg
Namespace:      churn-mllops
Priority:       0
Service Account: default
Node:           minikube/192.168.49.2
Start Time:     Sat, 10 Jan 2026 15:04:59 +0100
Labels:         app=churn-api
Annotations:    pod-template-hash=79797775cb
Status:         Running
IP:             10.244.0.9
IPs:
  IP:          10.244.0.9
Controlled By: ReplicaSet/churn-api-79797775cb
Containers:
  api:
    Container ID: docker://69476a8341d1af6434143749ef28e7f9e7b78ea2b9e77b0ecc2fa31736166a1f
    Image:         churn-api:v1
    Image ID:     docker://sha256:852fd96e813161ec6a2f6acbb717d7c71f08402fb3067921a9f18393b5d40e0a
    Port:          8000/TCP
    Host Port:    0/TCP
    State:        Running
      Started:   Sat, 10 Jan 2026 15:05:00 +0100
```

```

Started:      Sat, 10 Jan 2026 15:05:00 +0100
Ready:       True
Restart Count: 0
Environment:
  MODEL_NAME:      <set to the key 'MODEL_NAME' of config map 'churn-config'>  Optional: false
  LOG_LEVEL:       <set to the key 'LOG_LEVEL' of config map 'churn-config'>  Optional: false
  MONITORING_TOKEN: <set to the key 'MONITORING_TOKEN' in secret 'churn-secret'>  Optional: false
Mounts:
  /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-2n8vg (ro)
Conditions:
  Type        Status
  Initialized  True
  Ready        True
  ContainersReady  True
  PodScheduled  True
Volumes:
  kube-api-access-2n8vg:
    Type:           Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:   kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI:    true
  QoS Class:      BestEffort
  Node-Selectors: <none>
  Tolerations:   node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                  node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type  Reason  Age   From            Message
  ----  -----  --   --              --
  Normal Scheduled  26m  default-scheduler  Successfully assigned churn-mlops/churn-api-79797775cb-vwmqg to minikube
  Normal Pulled   26m  kubelet         Container image "churn-api:v1" already present on machine
  Normal Created   26m  kubelet         Created container api
  Normal Started   26m  kubelet         Started container api

Name:            churn-api-7bc94549d6-6xmj4
Namespace:       churn-mlops
Priority:        0
Service Account: default
Node:            minikube/192.168.49.2
Start Time:     Sat, 10 Jan 2026 15:29:57 +0100

```

```

Labels:          app=churn-api
Annotations:    pod-template-hash=7bc94549d6
Status:         Running
IP:             10.244.0.11
IPs:
  IP:          10.244.0.11
Controlled By: ReplicaSet/churn-api-7bc94549d6
Containers:
  api:
    Container ID: docker://2e9d9a4d98deaa4856e90f93e90d13a0b3fcefa12e7d2d4aaac3392db74ed77a
    Image:        churn-api:v1
    Image ID:    docker://sha256:c6f065a97d3dd03fb1ad7b65cbd2f55ccb4529a11828436c3f4d9ddc5c39b310
    Port:         8000/TCP
    Host Port:   0/TCP
    State:
      Started:  Sat, 10 Jan 2026 15:29:58 +0100
    Ready:       False
    Restart Count: 0
    Liveness:    http-get http://:8000/health delay=10s timeout=1s period=30s #success=1 #failure=3
    Readiness:   http-get http://:8000/ready delay=5s timeout=1s period=10s #success=1 #failure=3
    Startup:    http-get http://:8000/startup delay=0s timeout=1s period=5s #success=1 #failure=30
    Environment:
      MODEL_NAME:      <set to the key 'MODEL_NAME' of config map 'churn-config'>  Optional: false
      LOG_LEVEL:       <set to the key 'LOG_LEVEL' of config map 'churn-config'>  Optional: false
      MONITORING_TOKEN: <set to the key 'MONITORING_TOKEN' in secret 'churn-secret'>  Optional: false
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-z8gh2 (ro)
Conditions:
  Type        Status
  Initialized  True
  Ready        False
  ContainersReady  False
  PodScheduled  True
Volumes:
  kube-api-access-z8gh2:
    Type:           Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:   kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI:    true

```

```

/var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-z8gh2 (ro)
Conditions:
  Type        Status
  Initialized  True
  Ready       False
  ContainersReady  False
  PodsScheduled  True
Volumes:
  kube-api-access-z8gh2:  Projected (a volume that contains injected data from multiple sources)
    Type:          TokenExpirationSeconds: 3607
    ConfigMapName: kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI: true
  QoS Class:      BestEffort
  Node-Selectors: <none>
  Tolerations:   node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                  node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

Events:
  Type  Reason  Age   From            Message
  ----  -----  --   --              --
  Normal Scheduled 101s  default-scheduler  Successfully assigned churn-mlops/churn-api-7bc94549d6-6xmj4 to minikube
  Normal Pulled   100s  kubelet         Container image "churn-api:v1" already present on machine
  Normal Created   100s  kubelet         Created container api
  Normal Started   99s  kubelet         Started container api
  Warning Unhealthy 96s   kubelet         Startup probe failed: Get "http://10.244.0.11:8000/startup": dial tcp 10.244.0.11:8000: connect: connection refused
  Warning Unhealthy 1s (x19 over 91s) kubelet         Startup probe failed: HTTP probe failed with statuscode: 404
(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>

```

```

Handling connection for 38888
(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl rollout restart deployment churn-api
deployment.apps/churn-api restarted
(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>

```

```

Handling connection for 38888
(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl rollout status deployment churn-api
Waiting for deployment "churn-api" rollout to finish: 1 out of 2 new replicas have been updated...
error: deployment "churn-api" exceeded its progress deadline
(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
churn-api-79797775cb-jsz8x  1/1     Running   3 (47m ago)  26h
churn-api-79797775cb-vwmqg  1/1     Running   3 (47m ago)  26h
churn-api-7cc44df87-lq5hk   0/1     Running   5 (36s ago)  13m
(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |

```

Étape 12 : Volume persistant pour registry + logs

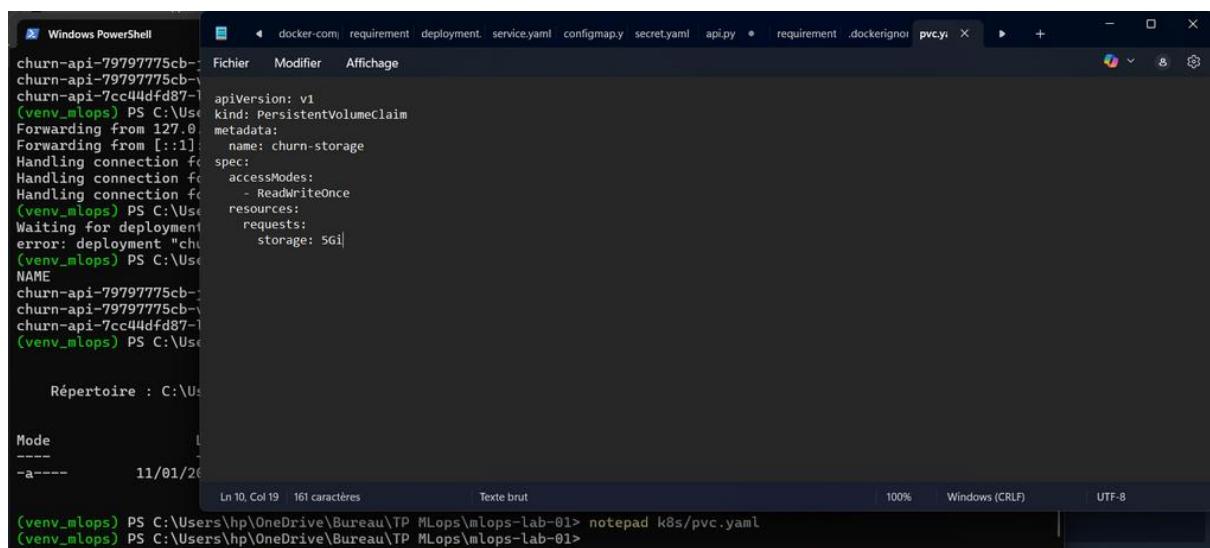
1. Crédation du PVC (PersistentVolumeClaim)

```
(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> New-Item k8s/pvc.yaml

Répertoire : C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01\k8s

Mode                LastWriteTime          Length Name
----                -----          ----  --
-a---  11/01/2026 17:49                 0  pvc.yaml

(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> |
```



```

Windows PowerShell
Fichier Modifier Affichage
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: churn-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> New-Item k8s/pvc.yaml
NAME
churn-api-79797775cb-jsz8x
churn-api-79797775cb-vwmqg
churn-api-7cc44df87-lq5hk
(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>

```

```

Répertoire : C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01\k8s

Mode                LastWriteTime          Length Name
----                -----          ----  --
-a---  11/01/2026 17:49                 0  pvc.yaml

(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> notePad k8s/pvc.yaml
(venv_mlops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>

```

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl apply -f k8s/pvc.yaml
persistentvolumeclaim/churn-storage created
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl get pvc
NAME           STATUS   VOLUME          CAPACITY   ACCESS MODES   STORAGECLASS   AGE
churn-storage   Bound    pvc-62416cf3-d144-45ca-84a1-e5e229a05624   5Gi        RWO          standard      23s
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

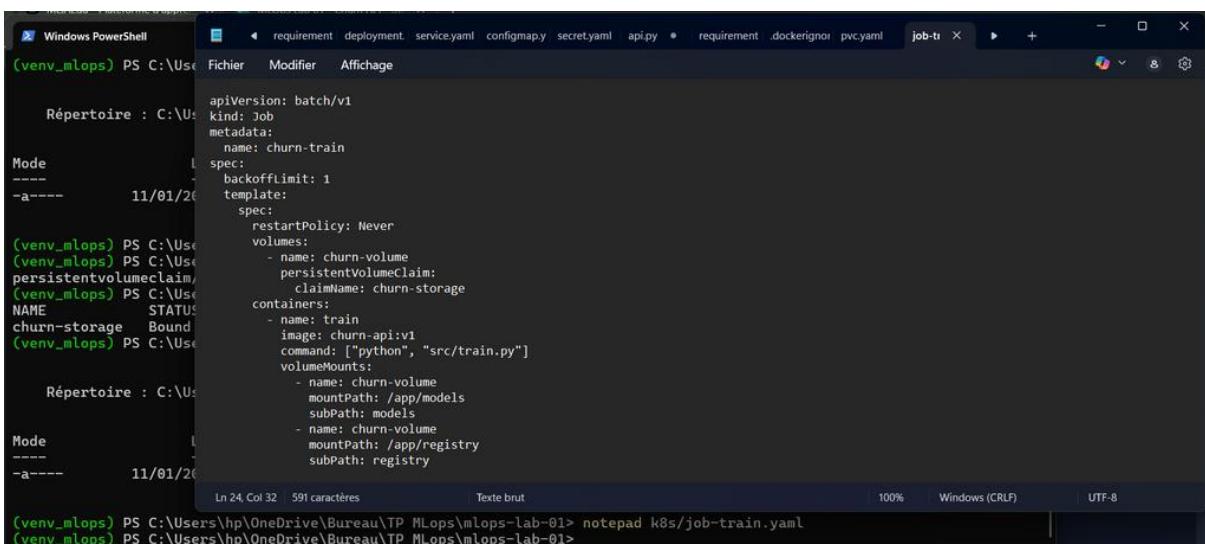
2. Crédation du Job d'entraînement

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> New-Item k8s/job-train.yaml

Répertoire : C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01\k8s

Mode          LastWriteTime       Length Name
----          -----          ----  --
-a---  11/01/2026     17:51          0 job-train.yaml

(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```



```
Windows PowerShell
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> Fichier Modifier Affichage
Répertoire : C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01\k8s
Mode          LastWriteTime       Length Name
----          -----          ----  --
-a---  11/01/2026     17:51          0 job-train.yaml

(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl apply -f job-train.yaml
job.batch/churn-train created
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

3. Application du Job

```
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01> kubectl apply -f k8s/job-train.yaml
job.batch/churn-train created
(venv_mllops) PS C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

À cette étape, un PersistentVolumeClaim (churn-storage) de 5 Gi a été créé afin de fournir un stockage persistant pour les modèles et le registry MLOps. Un Job Kubernetes (churn-train) a ensuite été défini pour exécuter le script src/train.py à l'intérieur du cluster, en montant ce volume sur les répertoires /app/models et /app/registry. Ainsi, les artefacts d'entraînement (modèles, fichier current_model.txt, logs...) sont conservés de manière durable indépendamment du cycle de vie des pods, ce qui est essentiel pour un système MLOps reproductible et résilient.

Étape 13 : NetworkPolicy

The terminal window shows the following commands and their outputs:

```

! C:\Windows\System32\cmd.exe
C:\Program Files\Docker\docker\resources\bin\kubectl.exe est la version 1.32
orer des incompatibilités avec Kubernetes 1.28.3
  Vous voulez kubectl v1.28.3 ? Essayez 'minikube kubectl -- get pods -A'
* Terminé ! kubectl est maintenant configuré pour utiliser "minikube" cluster
* "default" par défaut.

C:Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>kubectl get nodes
NAME      STATUS   ROLES      AGE   VERSION
minikube  Ready    control-plane  8d    v1.28.3

C:Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>New-Item k8s\networkpolicy.yaml
'New-Item' n'est pas reconnu en tant que commande interne
ou externe, un programme exécutable ou un fichier de commandes.

C:Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>New-Item k8s\networkpolicy.yaml
'New-Item' n'est pas reconnu en tant que commande interne
ou externe, un programme exécutable ou un fichier de commandes.

C:Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>notepad k8s\networkpolicy.yaml
C:Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>

```

The code editor window displays the following NetworkPolicy YAML configuration:

```

apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-internal-services
spec:
  podSelector:
    matchLabels:
      app: churn-api
  policyTypes:
    - Ingress
  ingress:
    - from:
        - podSelector: {}
  ports:
    - port: 8000
      protocol: TCP

```

- **podSelector: app: churn-api**

→ La règle s'applique aux pods ayant le label app=churn-api

- **policyTypes: Ingress**

→ On contrôle uniquement ce qui entre dans ces pods

- **ingress: from: podSelector: {}**

→ On autorise l'entrée depuis n'importe quel pod du cluster (podSelector vide = “tous les pods internes”)

- **ports: 8000/TCP**

→ On n'autorise le trafic que sur le port 8000 TCP

```
(venv_mllops) C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>notepad k8s\networkpolicy.yaml
(venv_mllops) C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>kubectl apply -f k8s\networkpolicy.yaml
networkpolicy.networking.k8s.io/allow-internal-services created
```

```
(venv_mllops) C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>kubectl get networkpolicy
NAME          POD-SELECTOR      AGE
allow-internal-services  app=churn-api  25s
```

Une NetworkPolicy allow-internal-services a été créée pour restreindre le trafic réseau entrant vers les Pods churn-api. Cette policy autorise uniquement les connexions provenant d'autres Pods du cluster sur le port 8000/TCP. Toute communication externe directe est bloquée, ce qui renforce la sécurité du microservice."

Étape 14 : Vérifications finales

1. Sélection des Pods par label

```
(venv_mllops) C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>kubectl get pods -l app=churn-api
NAME                  READY   STATUS    RESTARTS   AGE
churn-api-79797775cb-jsz8x  1/1     Running   4 (107m ago)   8d
churn-api-79797775cb-vwmqg  1/1     Running   4 (107m ago)   8d
churn-api-7cc44dfd87-lq5hk  0/1     Running   81 (6m4s ago)  7d17h
```

2. Service exposé

```
(venv_mllops) C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
churn-api-service  NodePort      10.108.81.153    <none>        80:30080/TCP  8d
```

3. Port-forward

```
(venv_mllops) C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>kubectl port-forward svc/churn-api-service 30080:80
Forwarding from 127.0.0.1:30080 -> 8000
Forwarding from [::1]:30080 -> 8000
Handling connection for 30080
Handling connection for 30080
```

4. Health-checks (liveness, readiness, startup)

```
(venv_mllops) C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>kubectl get pods -l app=churn-api
NAME                  READY   STATUS    RESTARTS   AGE
churn-api-79797775cb-jsz8x  1/1     Running   4 (108m ago)   8d
churn-api-79797775cb-vwmqg  1/1     Running   4 (108m ago)   8d
churn-api-7cc44dfd87-lq5hk  0/1     Running   81 (7m30s ago)  7d17h
```

5. ConfigMap & Secrets injectés

```
(venv_mllops) C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>kubectl exec -it churn-api-79797775cb-jsz8x -- python
src/monitor_drift.py
== Drift check sur 4 requêtes récentes ==
- tenure_months: mean_prod=27.000 | mean_train=30.246 | z=0.191
- num_complaints: mean_prod=1.500 | mean_train=1.174 | z=0.293
- avg_session_minutes: mean_prod=36.250 | mean_train=35.124 | z=0.095
Résultat : aucun drift détecté.

(venv_mllops) C:\Users\hp\OneDrive\Bureau\TP MLops\mllops-lab-01>
```

Conclusion

Ce Lab a permis de mettre en œuvre un pipeline complet de déploiement d'un modèle de Machine Learning dans un cluster Kubernetes en adoptant les bonnes pratiques MLOps. Nous avons conteneurisé une API FastAPI, versionné l'image Docker, déployé l'application sur Minikube et configuré les différents objets Kubernetes nécessaires à une exécution fiable et sécurisée.

L'ajout des ConfigMaps, Secrets et NetworkPolicies a permis d'introduire une séparation propre entre le code, la configuration et les credentials, tandis que les probes de santé ont assuré le bon fonctionnement et la disponibilité de l'API. Enfin, les tests d'accès (/health, /predict) ainsi que l'exécution du script de monitoring ont montré la capacité à superviser le comportement du modèle en production, étape cruciale pour détecter du concept drift.

À travers ce Lab, nous avons donc acquis une vision concrète d'un déploiement ML industriel “production-ready”, en démontrant comment un modèle de churn peut être mis en production sur Kubernetes avec résilience, observabilité et maintenabilité.