

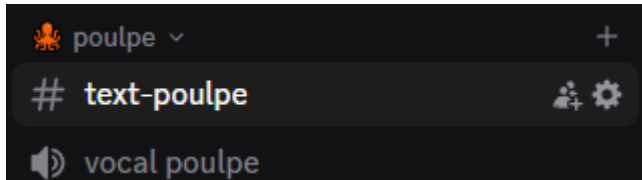
Améliorer une solution d'IA en continu

Préparation - jour 1.....	1
Discord Webhook.....	1
Dockerfile pour Uptime Kuma et une api dans un conteneur.....	2
Configurez Uptime Kuma :	2
Notification.....	3
Intro Prefect — pipeline « random-check ».....	5
Étapes à suivre.....	5
Vérifier que tout tourne dans l'UI Prefect (port 4200).....	7
Conteneurisez votre code.....	8
Créer une api - jour 2.....	9
Route predict.....	9
Route health.....	9
Route generate.....	9
Route retrain.....	9
Tests unitaires.....	9
Monitoring et application - jour 3.....	10
Mise en place de la documentation.....	10
Prometheus et Grafana.....	10
Loguru.....	10
Route retrain.....	10
Uptime Kuma.....	10
UI.....	10
API Token et migration Alembic.....	11
Première restitution - jour 4.....	11
Automatisation.....	11
Mise en place de l'Automatisation.....	11
Discord Webhook.....	11
Template.....	11
Projet IA - jours 5-6-7-8 et oui pas de week end cette semaine.....	12
Projet 1.....	12
Projet 2.....	12
Projet 3.....	12
Projet 4.....	12
Bonus - Intégration de celery pour une gestion asynchrone du modèle....	12

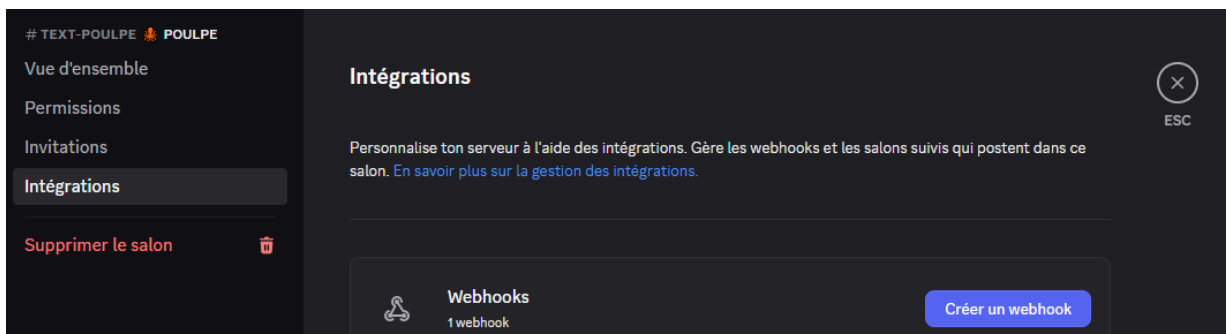
Préparation - jour 1 - groupe

Discord Webhook

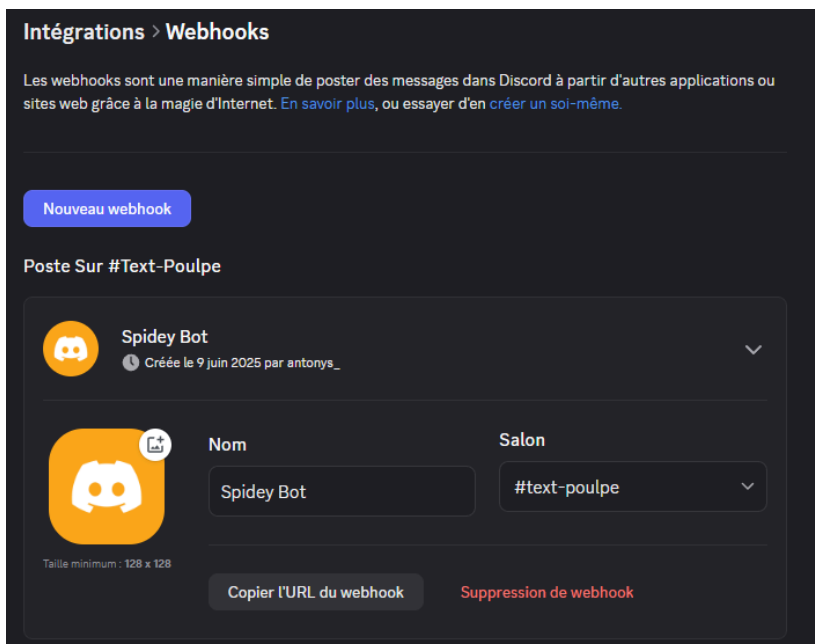
cliquez sur la roue (paramètre)



Puis intégration



Puis copier l'url du webhook



Docker-compose pour Uptime Kuma et une api dans un conteneur

```
version: '3.8'

services:
  app:
    build: .
    container_name: fastapi_app
    ports:
      - "8000:8000"
    restart: unless-stopped
  uptime-kuma:
    image: louislam/uptime-kuma:latest
    container_name: uptime_kuma
    ports:
      - "3001:3001"
    volumes:
      - uptime-kuma-data:/app/data
    restart: unless-stopped
volumes:
  uptime-kuma-data:
```

Configurez Uptime Kuma :

- Accédez à <http://localhost:3001> dans votre navigateur.
- Ajoutez un nouveau service dans Uptime Kuma avec l'URL http://fastapi_app:8000/health.
- Configurez le ping toutes les 30 secondes pour vérifier que l'API fonctionne correctement.

Général

Type de sonde

HTTP(s)

Nom d'affichage

health

URL

http://fastapi_app:8000/health

Intervalle de vérification (Vérifier toutes les 30 secondes)

30

Essais

3

Nombre d'essais avant que le service ne soit déclaré hors ligne et qu'une

Notifications

☒ Ma notification Discord numéro (1) [Modifier](#)

Créer une notification

Proxy

Non disponible, merci de le configurer.

Configurer le proxy

Options HTTP

Méthode

GET

Notification

Créer une notification

Type de notification

Discord

Nom d'affichage

Ma notification Discord numéro (1)

URL vers le webhook Discord

https://discord.com/api/webhooks/13817455069964575

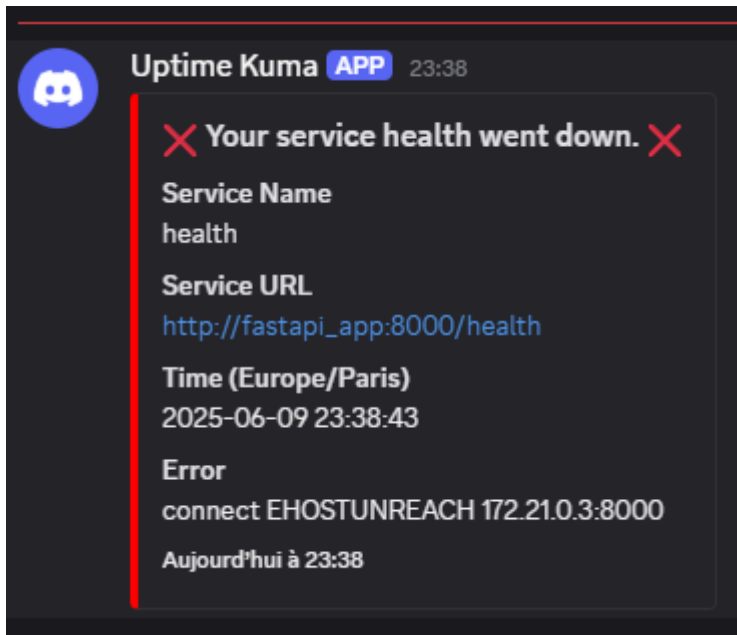
Vous pouvez l'obtenir en allant dans « Paramètres du serveur » -> « Intégrations » -> « Consulter les webhooks » -> « Nouveau Webhook »

Nom du robot (affiché)

Uptime Kuma

Si vous arrêtez votre API (docker stop ID_du_conteneur)

```
UptimeKuma> docker stop ddf72e341a95
```



exemple dans un script, directement dans un code python avec requests

```
def send_discord_embed(message):  
    """Envoyer un message à un canal Discord via un Webhook."""  
  
    data = {"embeds": [{  
        "title": "Résultats du pipeline",  
        "description": message,  
        "color": 5814783,  
        "fields": [{  
            "name": "Status",  
            "value": "Succès",  
            "inline": True  
        }]  
    }]}  
  
    response = requests.post(DISCORD_WEBHOOK_URL, json=data)  
  
    if response.status_code != 204:  
        print(f"Erreur lors de l'envoi de l'embed : {response.status_code}")  
    else:  
        print("Embed envoyé avec succès !")  
  
send_discord_embed("Le traitement des données est terminé avec succès.")
```

Intro Prefect — pipeline « random-check »

Dans cet atelier, vous construirez progressivement un pipeline Prefect qui s'exécute toutes les 30 secondes : il génère un nombre aléatoire et, s'il est inférieur à 0,5, déclenche un retrain (échec + retries) ; sinon, il affiche ok.

L'exercice poursuit deux objectifs :

1. Explorer la syntaxe @task / @flow et les concepts d'orchestration : logs, retries, planification.
2. Passer d'une exécution locale à une exécution conteneurisée (Docker Compose) sans modifier le code Python.

Le tirage aléatoire joue ici le rôle d'un test de performance : un résultat $< 0,5$ symbolise la dérive d'un modèle qu'il faut ré-entraîner.

Étapes à suivre

Initialisez un projet propre

```
python -m venv .venv  
  
source .venv/bin/activate  
# ou .venv\Scripts\Activate.ps1 sous Windows  
  
pip install "prefect>=3.1"
```

Créez flow.py et importez :

```
from prefect import flow, task  
  
from prefect.logging import get_run_logger
```

Expérimentez @task, @flow et get_run_logger

Options à découvrir : retries, retry_delay_seconds, logger.warning, logger.info, raise.

Implémentez la tâche

```
@task(retries=2, retry_delay_seconds=1)
def check_random(): ...
```

Implémentez le flow

```
@flow
def periodic_check(): ...
```

Planifiez l'exécution

```
if __name__ == "__main__":
    periodic_check.serve(
        name="every-10s",
        interval=10
    )
```

Le bloc `if __name__ == "__main__":` sert à lancer le scheduler et le worker intégrés lorsque vous exécutez directement le fichier ; il est ignoré si le module est importé ailleurs.

Ajoutez des logs pour savoir si votre routine est ok ou doit être entraîné à nouveau.

En suivant ces étapes, vous obtiendrez d'abord un pipeline fonctionnant en local, puis vous pourrez l'envelopper dans Docker Compose pour tester le même code dans des conteneurs.

Vérifier que tout tourne dans l'UI Prefect (port 4200)

Ouvrez un terminal dédié et lancez :

```
prefect server start
```

Une fois le serveur démarré, lancez votre code

```
python flow.py
```

Normalement ça ne marchera pas car l'OS doit connaître l'adresse de l'UI, des variables d'environnements doivent être données

```
# PowerShell
$Env:PREFECT_API_URL = "http://127.0.0.1:4200/api"

# Bash / zsh / WSL
export PREFECT_API_URL=http://127.0.0.1:4200/api
```

ou dans votre code python avec os:

```
# PYTHONIOENCODING : évite les UnicodeDecodeError sous Windows
# PREFECT_API_URL : indique au SDK où se trouve l'API Prefect
os.environ.setdefault("PYTHONIOENCODING", "utf-8")

os.environ.setdefault("PREFECT_API_URL",
"http://127.0.0.1:4200/api")
```


Conteneurisez votre code

Créez un dockerfile pour lancer votre [flow.py](#)

```
version: "3.9"

services:
  prefect-server:
    image: prefecthq/prefect:3-latest
    command: prefect server start --host 0.0.0.0
    container_name: prefect-server
    ports:
      - "4200:4200"          # UI + API
    volumes:
      - prefect_data:/root/.prefect # persistence SQLite
    restart: unless-stopped

  random-check-with-server:
    build: .
    depends_on:
      - prefect-server
    environment:
      - PREFECT_API_URL=http://prefect-server:4200/api
      - PYTHONIOENCODING=utf-8
    restart: unless-stopped

volumes:
  prefect_data:
```

Créer une api - jour 2

Approche agile

Création des groupes, mise en place du dépôt github et de la posture agile (kanban, US, EPIC etc....)

Route predict

Retourne la prédiction sur le dernier dataset généré en base de données

Détail de la prédiction: c'est une régression logistique toute simple entraîné sur un dataset à 2 variables

Route health

retourne ok, 200

Route generate

génère un dataset basé sur des nombres aléatoires, le dataset est linéaire avec 2 features, une des deux features change de signe avec l'heure (modulo de l'heure par 2 $\rightarrow a-0.5$) il donne 2 classes 0 ou 1

le dataset est stocké en base de données avec un numéro pour savoir de quel génération de dataset il s'agit

Route retrain

recupère le dernier dataset avec la target et re entraîne à chaud le modèle

Mise en place de MLFlow dans la route retrain

Tests unitaires

- health: retourne bien 200
- predict: retourne bien une valeur 0 ou 1
- generate: la base de données est testées
- imagine d'autre tests pour ces routes

Monitoring et application - jour 3

Mise en place de la documentation

Daily, readme.md et du CI/CD sur GitHub Action.

Prometheus et Grafana

mise en place du monitoring avec prometheus et grafana sur l'utilisation des ressources de l'application

Loguru

mise en place de la journalisation dans des fichiers de logs

Route retrain

avant de faire le réentraînement, faire la prédiction et mesurer les performances de la régression logistique. Si la métrique choisie est inférieure à un seuil alors seulement on lance le réentraînement.

Uptime Kuma


Mettre en place Uptime Kuma dans le docker compose directement pour qu'il ping l'API toutes les minutes pour s'assurer que tout se passe bien, sinon lancer une alerte

UI

Mettre en place un petit streamlit avec un bouton pour chaque route et un système d'authentification

- journalisation
- CI/CD

API Token et migration Alembic

- Modifier la base de données  OPCO - sqlalchemy
- Mettre en place Les tokens

Première restitution - jour 4

Daily

Préparation de slides afin de présenter le travail effectué et d'un google doc (ou autre) qui détaille les mises en place des services qui étaient inconnues, y ajouter les difficultés que vous avez rencontrées et comment vous les avez surpassées.

Automatisation

Veille technologique sur PREFECT et comment l'utiliser pour automatiser l'entraînement continu de votre application

Mise en place de l'Automatisation

Mettre en place prefect dans votre projet, attention il ne gère pas l'API, nettoyer les routes, la route retrain n'est plus utile

Discord Webhook

Grâce au token fournie pour votre canal mettre en place une communication entre votre API/PREFECT ainsi que uptime kuma et discord pour vous donner les résultats (drifting or not drifting)

Template

Créez un template à partir de votre travail, ce sera la base de vos projets chef d'oeuvre

Mettez à jour votre journal de recherche et vos slides

Présentation du travail des groupes, analyse réflexive

Projet IA - jours 5-6-7-8 et oui pas de week end cette semaine

Reprenez votre template et développez un projet d'IA plus profond. De votre choix ou parmi la liste suivante.

Projet 1

Reprenez le projet Doggy and cat adventure et adaptez le à la reconnaissance audio/photo de personnes qui constituent votre groupe.

Projet 2

Idem mais sans audio et avec la vidéo en continu avec fine tuning de yolo 11

Projet 3

Fine Tuning de Yolo V11 (ultralytics) pour pierre, ciseau feuille

<https://universe.roboflow.com/roboflow-58fyf/rock-paper-scissors-sxsw>

Projet 4

Créez

Bonus - Intégration de celery pour une gestion asynchrone du modèle

Faites faire les tâches de prédiction (API) et d'entraînement (PREFECT) par celery pour ne pas bloquer votre application pendant l'utilisation du modèle.