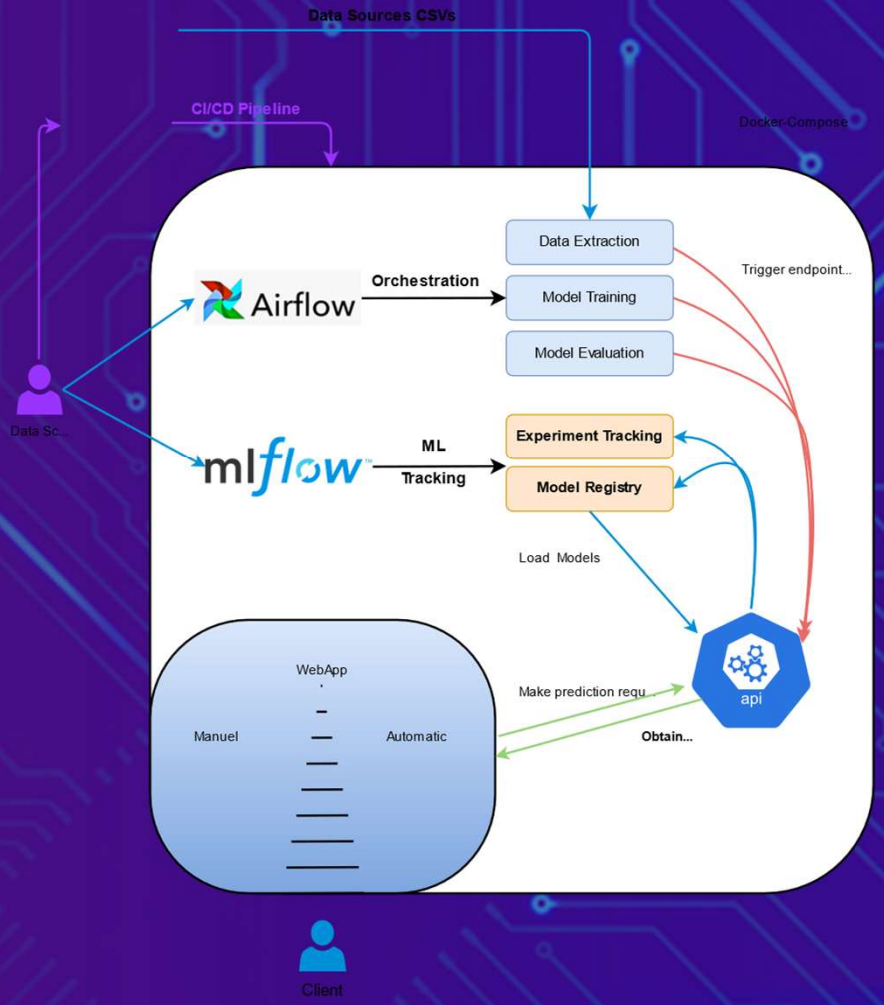
PROJET MLOps
Météo

Mentor : Antoine Fradin I Membres de l'équipe : Gael, Clément Simonin et Christophe Levra

1. Vue d’ensemble de l’architecture MLOps du projet météo, contexte, problématique & objectif
2. Collecte, préparation des données (Data Engineering), model train & DataViz
3. API de prédiction avec FastAPI, Database avec PostgreSQL & tests unitaires
4. Orchestration & Automatisation avec Airflow
5. Tracking & Registry avec MLflow
6. Tests et CI/CD avec GitHub actions
7. Interface utilisateur avec Streamlit frontend
8. Conclusion : Bilan du projet & Améliorations futures

1·0· Vue d'ensemble de l'architecture MLOps du projet météo



1·1· Contexte, problématique, défis méthodologiques & objectif projet

- **Contexte** : Prédire la pluie du lendemain à partir de données météo australiennes (objectif : RainTomorrow = Yes/No).

  → Données : +10 ans d'observations météo quotidiennes, issues de plusieurs stations australiennes.
  La variable cible à prédire RainTomorrow répond à la question cruciale : Va-t-il pleuvoir le lendemain? (Oui ou Non).

- **Problématique métier** : La prédiction météorologique représente un défi complexe de classification binaire où l'objectif est de prédire la probabilité de précipitations pour le lendemain avec une précision maximale.

  → Prédire avec fiabilité un phénomène rare (déséquilibre : majorité de "No").

- **Défis méthodologiques** :

  → Interactions non-linéaires entre les variables, saisonnalité, hétérogénéité géographique, latence de validation (24h), nécessité de modèles robustes et adaptatifs.

- **Objectif projet** :

  → Industrialiser la prédiction météo via un pipeline automatisé, sécurisé, traçable et réentraînable, du preprocessing à la prédiction, avec déploiement et interface utilisateur.

2.0. Collecte, préparation des données (Data Engineering), ...

- **Nettoyage & prétraitement rigoureux : imputation, encodage, standardisation, traitement des outliers.**
  - Imputation des valeurs manquantes : mode pour variables catégorielles, suppression pour colonnes critiques (RainToday, RainTomorrow).
  - Encodage des variables catégorielles : transformation binaire (RainToday, RainTomorrow → 0/1) et label encoding (Location, WindGustDir, etc…).
  - Standardisation des variables numériques : via StandardScaler.
  - Traitement des valeurs aberrantes : méthode IQR (valeurs extrêmes remplacées par des seuils).

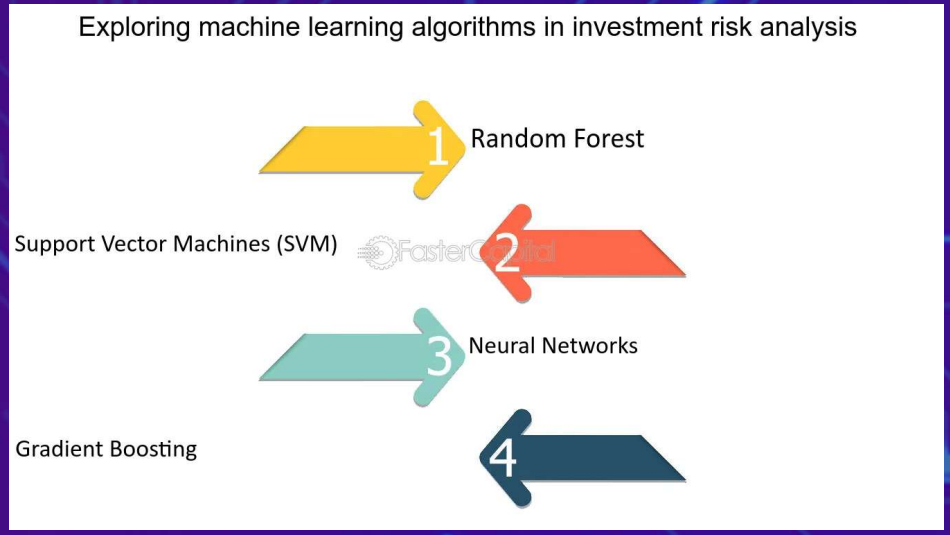
- **Sélection de variables clés (feature selection) : humidité, pression, pluie actuelle, etc…**
  - Variables conservées : Température, humidité, pression, direction/force du vent, RainToday, etc…
  - Critères : corrélation avec la variable cible + importance prédictive.
  - Variables supprimées : 'Date', 'Rainfall', 'Temp3pm', 'Temp9am', 'Pressure9am'

- **Analyse exploratoire : déséquilibre des classes (majorité de “No”), corrélations avec la cible.**
  - Étude de la corrélation entre les features météo et la variable cible.
  - Détection de déséquilibres de classe (majorité de "No" sur RainTomorrow).
  - Déséquilibre important sur RainTomorrow : majorité de "No" → justifie la métrique F1-score et le choix du seuil.
  - Corrélations observées :
    - Humidity3pm, RainToday et Cloud3pm = très bons prédicteurs.
    - Location = facteur discriminant (lié à la géographie).

Cf script prepare_data.py

2·1· Model Train

- **Model Train : Données transformées pour un apprentissage optimisé du modèle utilisé RandomForest.**
  - **Entraînement du modèle**
    - **Choix de l'algorithme** : **RandomForestClassifier avec GridSearch.:**
    - Validation croisée stratifiée (80/20).
    - Export et stockage du modèle entraîné au format .pkl + log dans MLflow.



Cf script train_model.py

2·2· DataViz - Visualisations clés

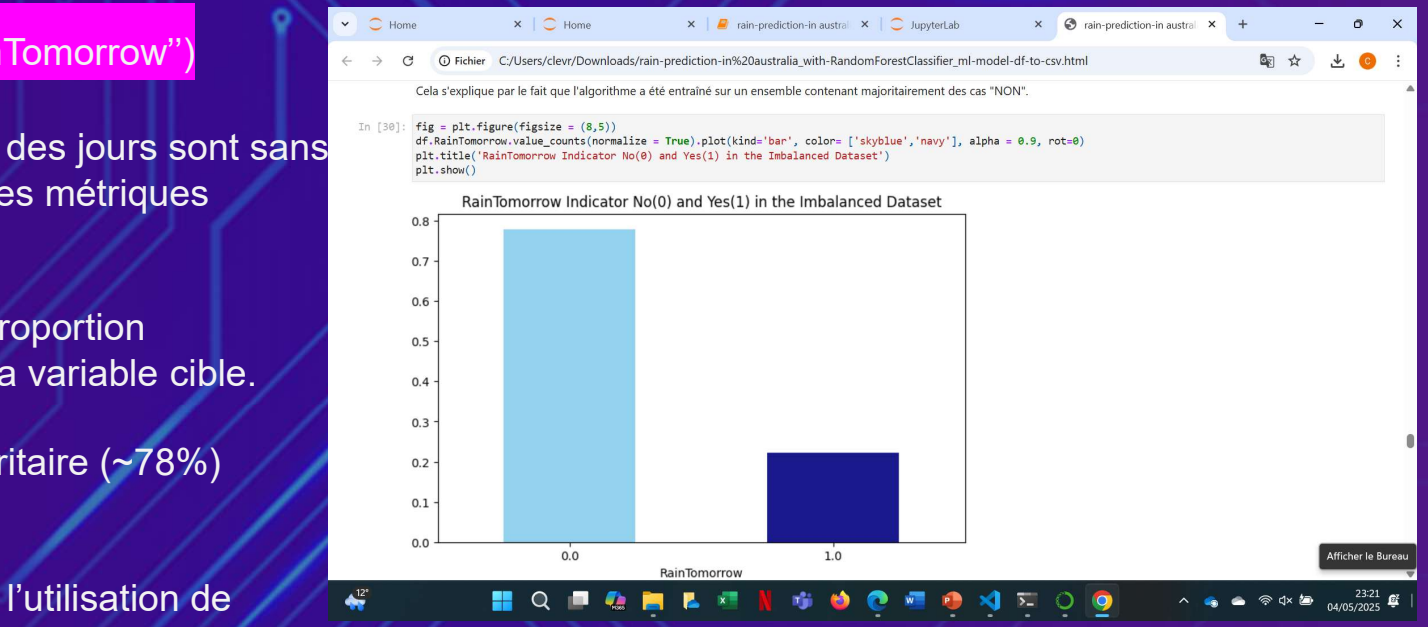
- Distribution de la Variable Cible (‘’RainTomorrow’’)

○ **Contexte** : On observe que la majorité des jours sont sans pluie. Ce déséquilibre impacte le choix des métriques d’évaluation et la difficulté du problème.

○ **Interprétation** : Ce barplot montre la proportion d’échantillons pour chaque classe dans la variable cible.

□ 0 (pas de pluie) est largement majoritaire (~78%)
□ 1 (pluie) est minoritaire (~22%)

Le dataset est déséquilibré : Cela justifie l’utilisation de métriques comme la courbe ROC et celle de précision-rappel, que vous pourrez observer dans les slides suivants, et la nécessité d’être attentif aux faux négatifs.

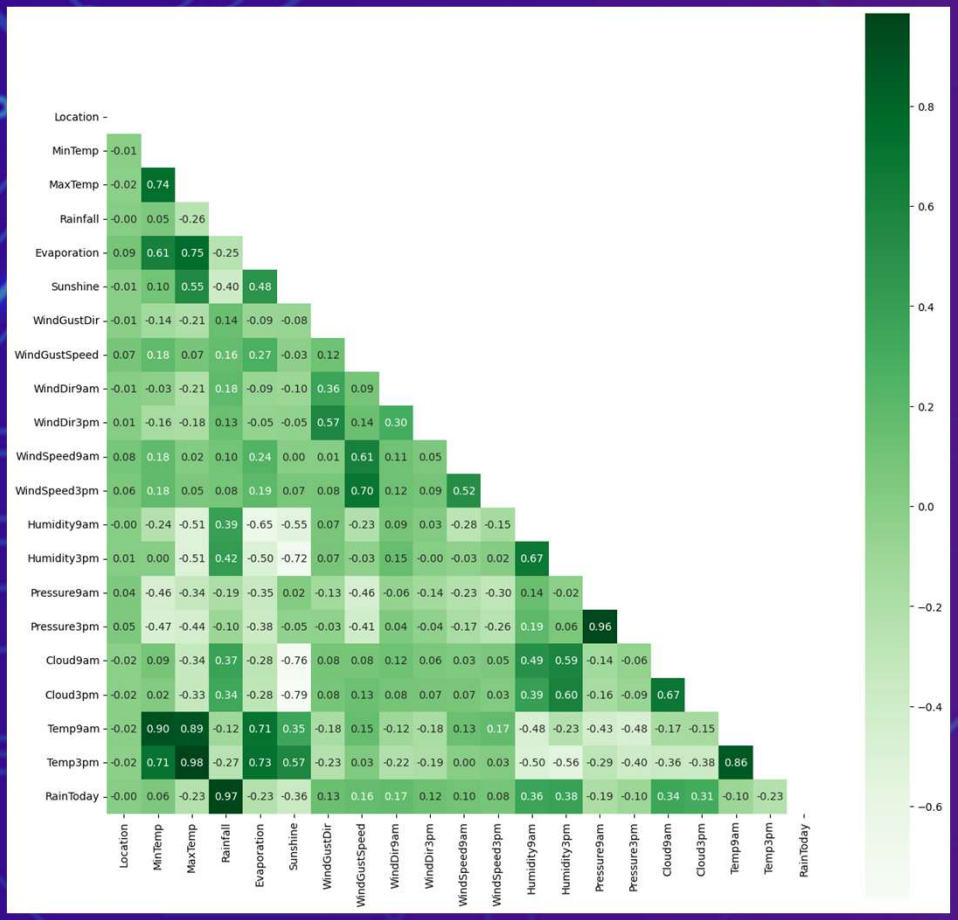


Barplot du déséquilibre des classes

2·2· DataViz - Visualisations clés

- Heatmap de Corrélation

Permet de visualiser les corrélations entre les différentes variables numériques du jeu de données et de justifier notre choix/sélection de certaines variables, avant la modélisation.



○ **Interprétation** : Cette heatmap présente les coefficients de corrélation entre les variables numériques. On remarque des corrélations très fortes entre certaines variables, par exemple :

- □ Temp9am et MaxTemp (0.90),
- □ Temp3pm et MaxTemp (0.98),
- □ Pressure9am et Pressure3pm (0.96),

→ *ce qui indique que ces variables sont très liées.*

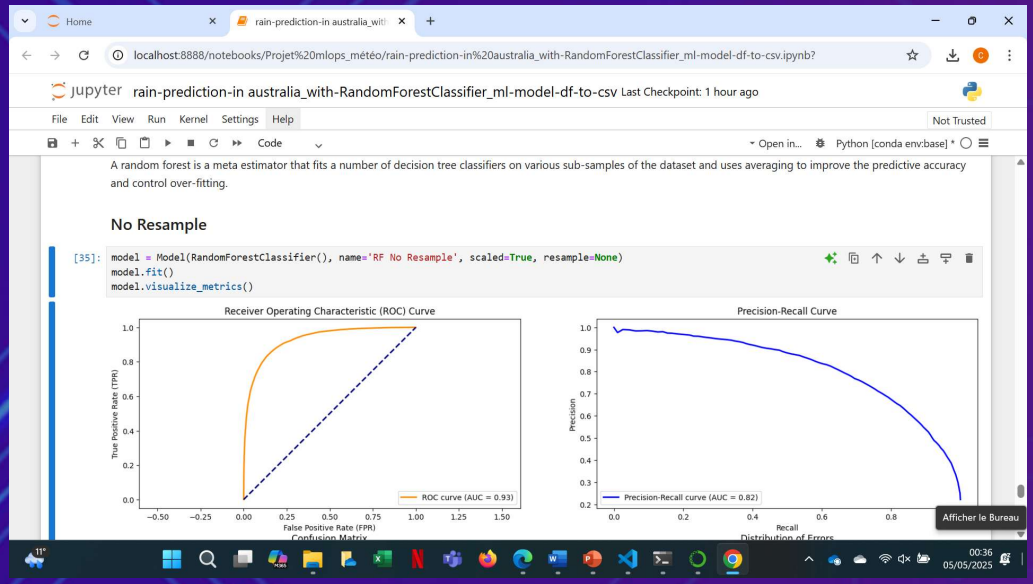
Certaines corrélations négatives sont aussi notables, par exemple entre Humidity3pm et Sunshine (-0.72). Ces corrélations suggèrent des risques de redondance (multicolinéarité) qui peuvent impacter certains modèles.
La heatmap révèle également que l'humidité à 3pm et la pression à 9am sont fortement corrélées avec la variable cible, ce qui justifie leur prise en compte dans le modèle.

2·2· DataViz - Visualisations clés

- Courbe AUC-ROC (Area Under the ROC Curve)

C'est LA référence pour évaluer la capacité discriminante du modèle, surtout en présence de classes déséquilibrées.

Elle mesure la capacité du modèle à distinguer les jours de pluie des jours sans pluie et à illustrer sa performance globale.



- **Interprétation AUC-ROC** : Elle montre la performance du modèle en termes de compromis entre le taux de vrais positifs (sensibilité) et le taux de faux positifs.

  - **AUC-ROC = 0.93** : L'aire sous la courbe (AUC) est de 0.93, ce qui indique une excellente capacité du modèle à distinguer les jours où il va pleuvoir de ceux où il ne pleuvra pas. Plus l'AUC est proche de 1, meilleur est le modèle.

- Courbe AUC-PR (Precision-Recall) : Cette courbe (en haut à droite) est également utile pour les datasets déséquilibrés.

- **Interprétation AUC-PR** : Elle montre la relation entre la précision des prédictions positives et le rappel (capacité à retrouver tous les vrais positifs).

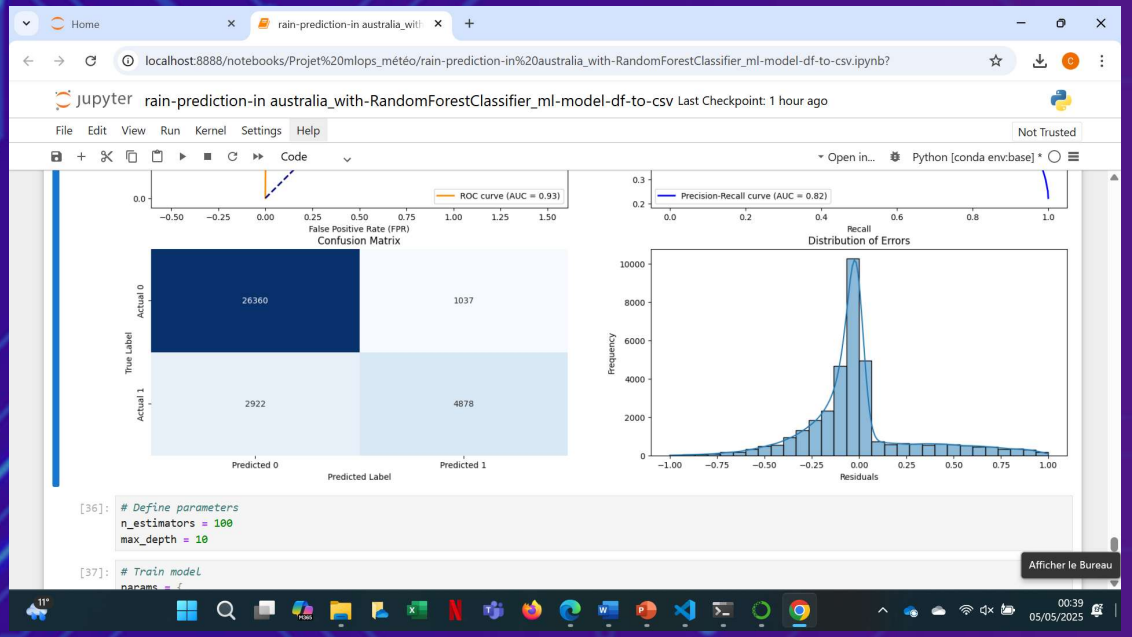
  - **AUC-PR = 0.82** : L'aire sous la courbe est de 0.82, ce qui est très correct, surtout dans un contexte de déséquilibre des classes. Elle montre que le modèle a une très bonne capacité de discrimination entre les jours pluvieux et non pluvieux.

2·2· DataViz - Visualisations clés

- Matrice de Confusion

L'exploitation de la matrice de confusion permet d'objectiver les performances du modèle sur la détection des jours de pluie, en mettant en évidence les taux d'erreurs critiques comme le**s faux négatifs.**

Cette analyse contribue au monitoring des métriques clés et à l'amélioration continue du modèle.



Nombre de prédictions correctes et incorrectes pour chaque classe :

- 26360 : Vrais négatifs (pas de pluie correctement prédit)
- 1037 : Faux positifs (prédit pluie alors qu'il n'y avait pas de pluie)
- 2922 : Faux négatifs (pas prédit pluie alors qu'il y avait pluie)
- 4878 : Vrais positifs (pluie correctement prédit)

La matrice de confusion indique que le modèle prédit correctement la majorité des jours sans pluie, mais a plus de difficultés sur les jours de pluie, ce qui se traduit par un certain nombre de faux négatifs. Cela s'explique par un déséquilibre dans le jeu de données (plus de jours sans pluie).

Key Metrics

Modèle Random Forest — Classification binaire

Key Metric 1

ACCURACY = 0.85

Key Metric 2

AUC-ROC = 0.93

Key Metric 3

AUC-PR = 0.82

3. API de prédiction avec FastAPI, Database avec PostgreSQL & Tests unitaires

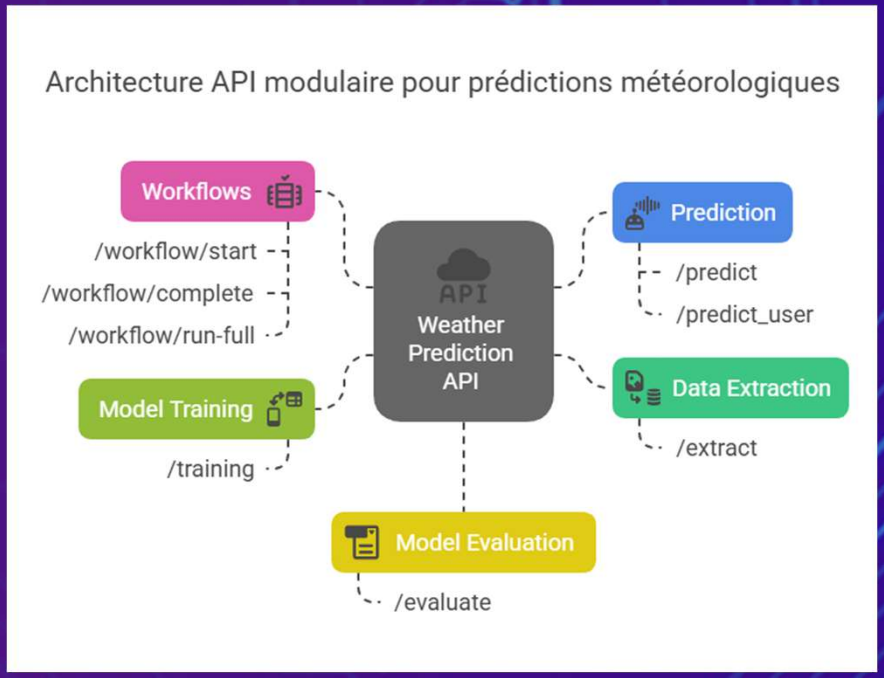


Schéma realisé avec la solution Napkin

- Architecture modulaire: Chaque endpoint répond à une tâche spécifique
- Flux de données complet: De l'extraction des données brutes à la prédiction
- Flexibilité d'utilisation: Prédictions automatiques ou via des données saisies par l'utilisateur
- Compatible avec un orchestrateur de workflow (Airflow)
- Compatible avec une solution de suivi de modèles (MLFlow)

Notre API respecte les principes REST tout en permettant un déploiement, une évolution et une maintenance simplifiée

3. API de prédiction avec FastAPI, Database avec PostgreSQL & Tests unitaires

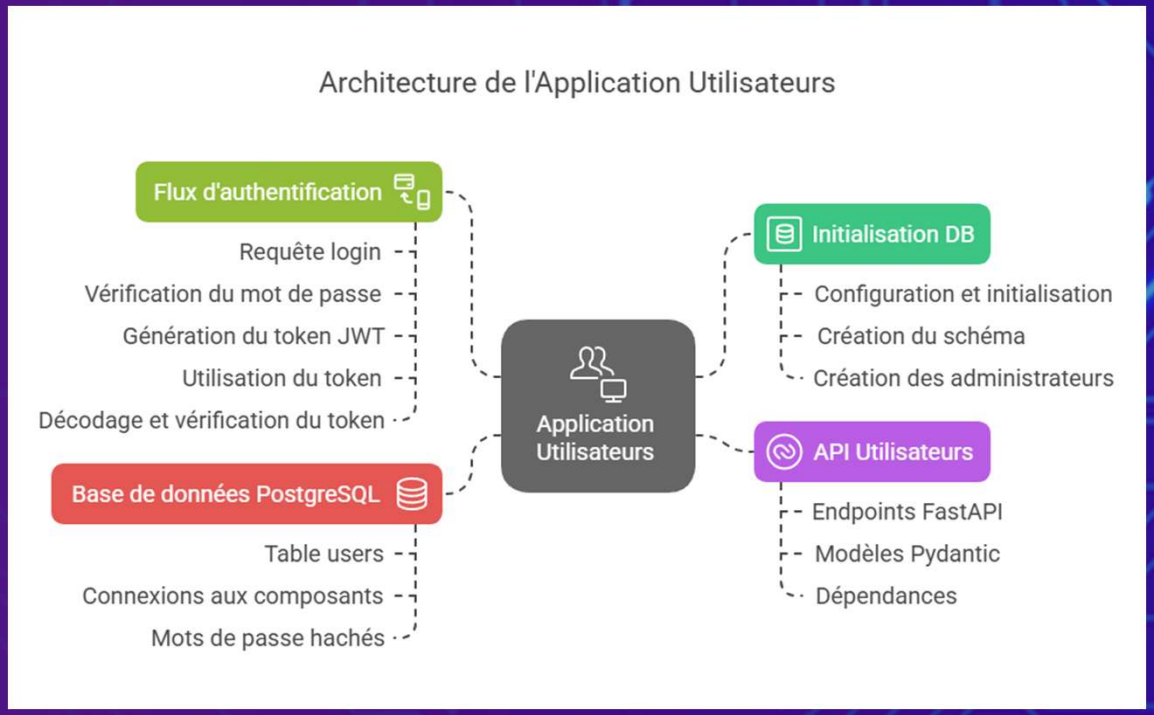


Schéma realisé avec la solution Napkin

- Le script est bien présent ainsi que la base de données et les endpoints, cependant, faute de temps,cela n'a pas été intégré à l'application.
- Sécurité mutlicouches: Hachage de mots de passe (bcrypt), tokens JWT pour s'authentifier (Oauth) avec des rôles limitant les accès (administrateur ou utilisateur)
- Suit les règles REST de l'API (endpoints spécifiques: /user (POST), /token, /user (GET) pour l'administrateur.
- Validation des entrées avec Pydantic et gestion des erreurs
- Contrôle des accès
- Initialisation de la base de données au lancement de l'application (init_db.py) et gère les operations en cours d'exécution (ajout d'utilisateur, authentication) – user_api.py

3. API de prédiction avec FastAPI, Database avec PostgreSQL & Tests unitaires

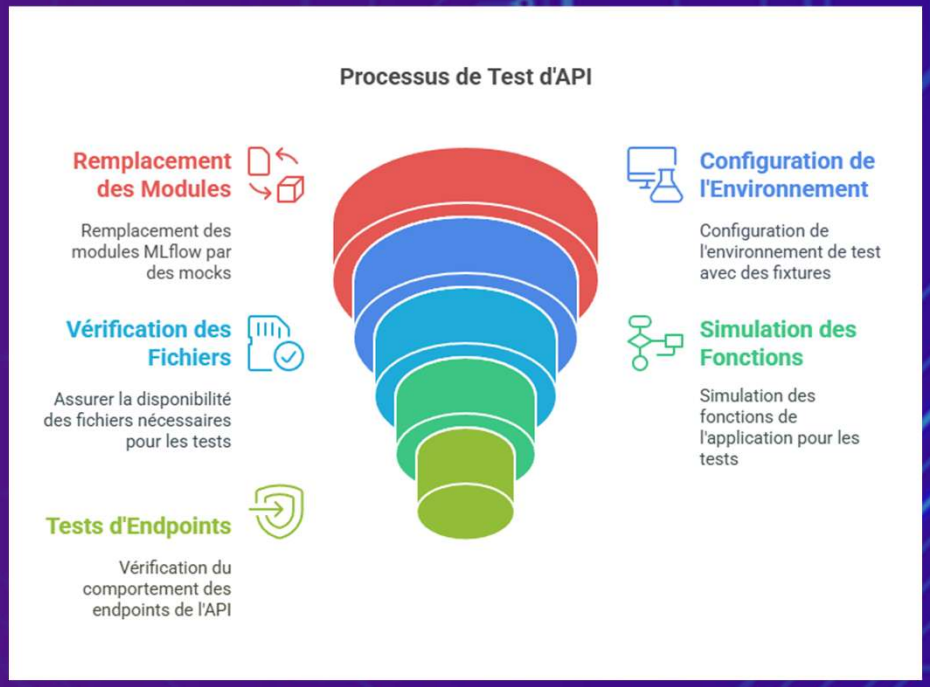


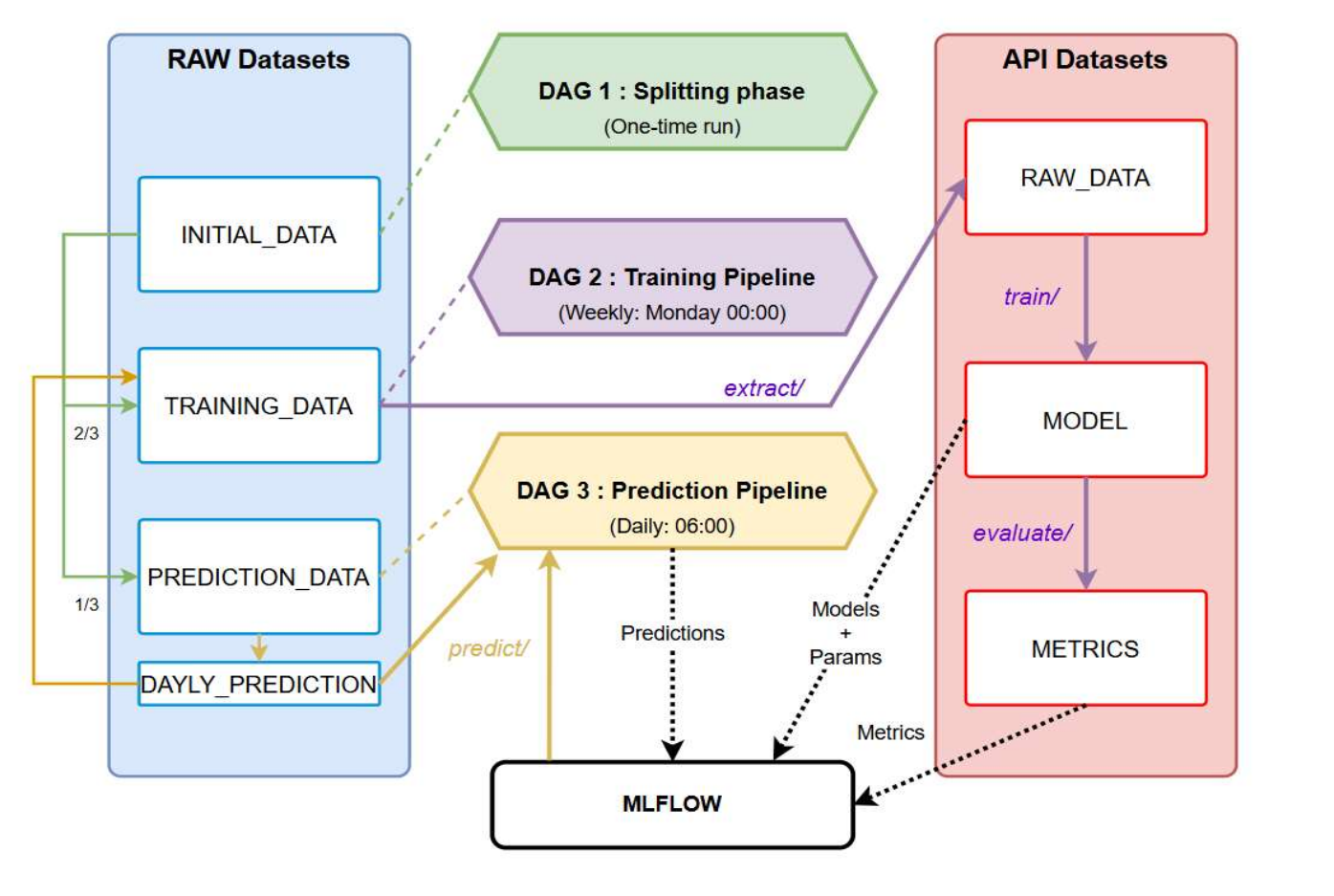
Schéma realisé avec la solution Napkin

- Difficulté initiale pour exécuter des tests sur des endpoints dont les fonctions mlflow (donc lié à l'exécution du serveur MLFlow)
- Simulation des modules MLFlow en utilisant des mocks (remplacements) – script mock_mlflow.py
- Utilisation de fixtures pour configurer l'environnement de test – script conftest.py:
  - Vérification de la présence de dossiers/fichiers nécessaires aux tests – fonction check_model_files.
  - Simulation des fonctions utilisées dans nos tests – function mock_application_functions.py
- Test des endpoints – script tests_unitaires.py
  - Tests d'appels: endpoinds extract, training, predict
  - Deux tests pour le endpoint /predict_user avec saisie de données valides et invalides.

Architecture de tests en 3 couches pour une isolation complète des tests. Ils s'exécutent rapidement sans dépendance à des services externes (MLFlow) tout en validant le comportement complet de l'application

4 Orchestration & Automatisation avec Airflow

- **Simulation d'un flux temps réel**
- **Architecture à 3 DAGs interconnectés**
- **Automatisation des workflows**



4.1 Avantages et enseignements d'Airflow

Gestion robuste des dépendances
FileSensors et fonction de comparaison de hash

Chainage entre DAGs
TriggerDagRunOperator

Persistance des données via XCOM
Transmission du run_id MLflow aux endpoints API

Résilience et observabilité
Reprise sur erreur, logs centralises, point de contrôle…

5 Tracking & Registry avec MLflow

Serveur de Tracking

Persistance des runs a travers les étapes du workflow

Configuration centralisée

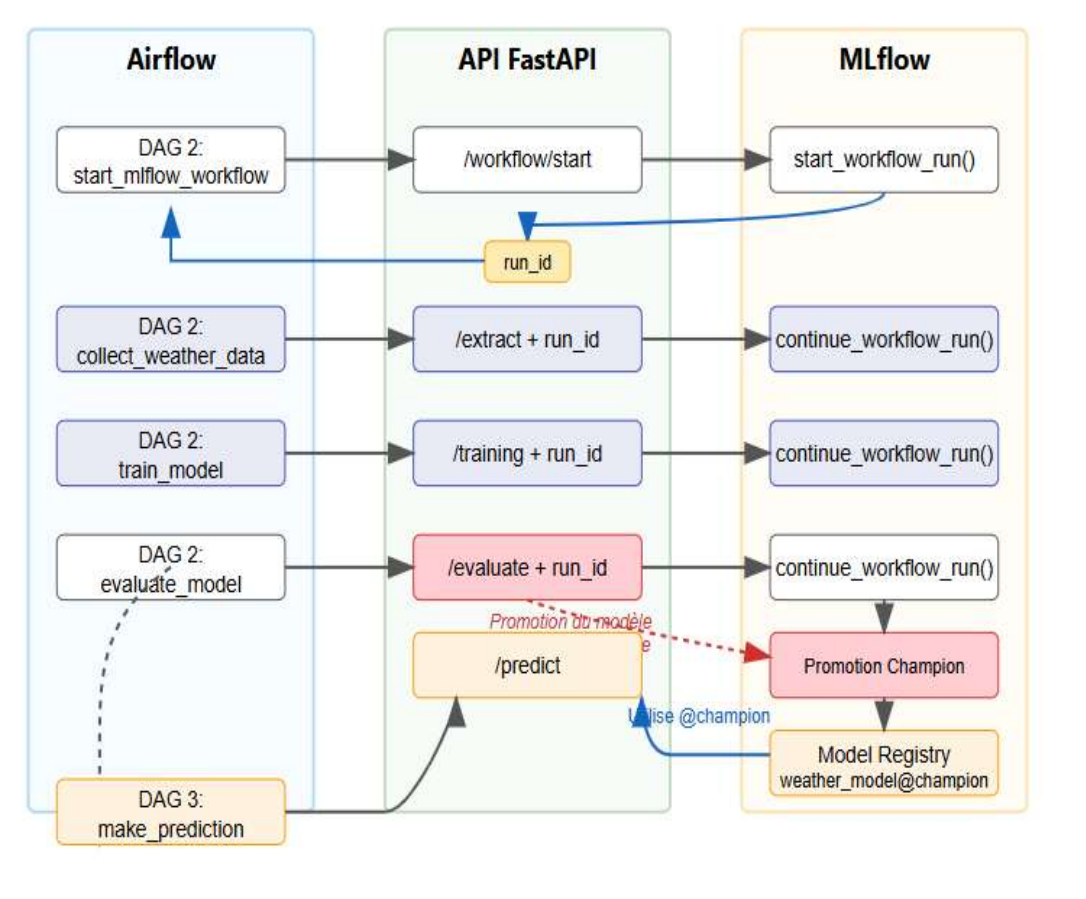
Mémoire des expériences

Registre de Modèles

Promotion automatique du champion basée sur les métriques

Utilisation du mécanisme d'alias pour la promotion du modèle

Nested runs basés sur le modèle champion



5·1 Avantages et Enseignements de MLflow

Flexibilité opérationnelle
Découpage entre Airflow et MLflow

Déploiement simplifié
L'utilisation d'alias pour la gestion des modèles a simplifié les procédures d'inférences et de Rollback

Documentation vivante
Tags, étiquettes, métadonnées… traçabilité complète

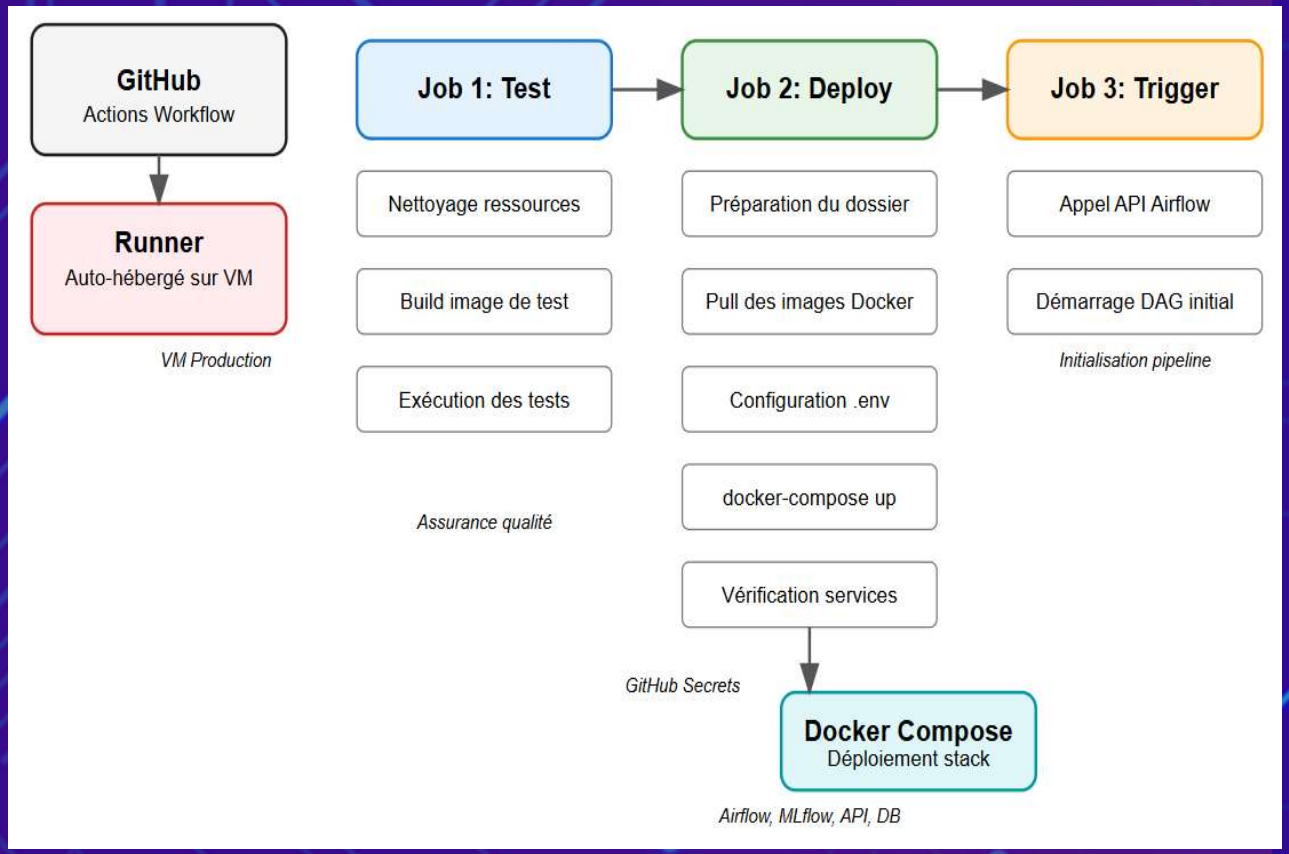
Conception réfléchie des le début du projet
Outil simple mais pas simpliste

6 Tests et CI/CD avec GitHub actions

Github action pour l'automatisation

Trois jobs séquentiels

Runner auto-hébergé



6·1 Choix techniques CI/CD

Docker-compose

Persistance des volumes
Simplicité de configuration
Orchestration multi-conteneurs
Cohérences entre environnement (prod, dev)

Runner auto-hébergé

Indépendance vis-à-vis de l'adresse IP
Pas besoin d'adresse publique
Accès directe aux services
Sécurité - Performance renforcées

7. Interface utilisateur avec Streamlit frontend

L'objectif est de présenter notre projet et de valider le bon fonctionnement de notre pipeline MLOps via une interface interactive.

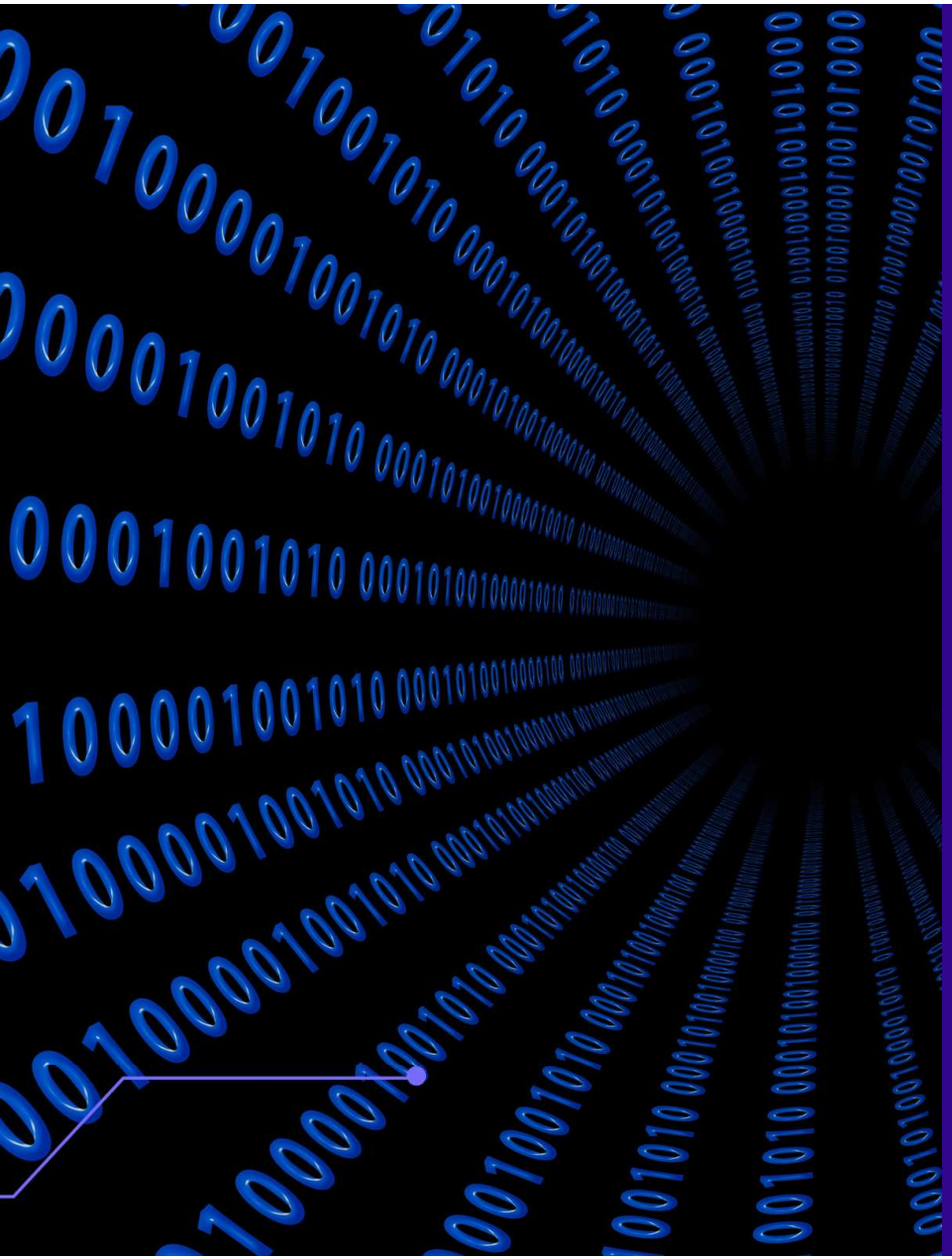
- **Visualiser les résultats du modèle (prédictions, probabilités, graphiques).**
  - Consommer l'API FastAPI via des appels HTTP en arrière-plan (requêtes GET ou POST en fonction du type de prédiction à réaliser.
  - Afficher des indicateurs de dérive ou de performance issus de MLflow.
- **Fonctionnalités du frontend.**
  - Saisie manuelle des données ou prédiction automatique basé sur la dernière lignes du fichier csv.
  - Affichage de la prédiction et probabilité.
  - Affichage d'une image « sun » ou « rain » en fonction de la probabilité la plus probable
- **Connexion API.**
  - Appel du endpoint /predict de FastAPI pour la prédiction automatique.
  - Envoi des données saisies au endpoint /predict_user pour la prédiction manuelle

8. Conclusion : Bilan du projet & Améliorations futures

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

- **Bilan du projet**
  - Résultats :
    - Bonne performance sur F1-score, système entièrement déployé et testable. Accuracy : ~85 %, AUC-ROC : ~0.93 et AUC-PR : ~0.82
  - Limites :
    - Données simulées, pas de flux temps réel.
    - Modèle unique (pas de segmentation régionale).
    - Réentraîne sur batch uniquement (pas de streaming).
    - Pas encore de lag features

- **Améliorations futures.**
  - Perspectives :
    - Passage en production avec Kubernetes ou infrastructure cloud (AWS/GCP).
    - Monitoring avec Grafana.
    - Modèles plus complexes (GBM, LSTM).
    - Notifications utilisateurs.



Projet MLOPS météo

This presentation template is free for everyone to use