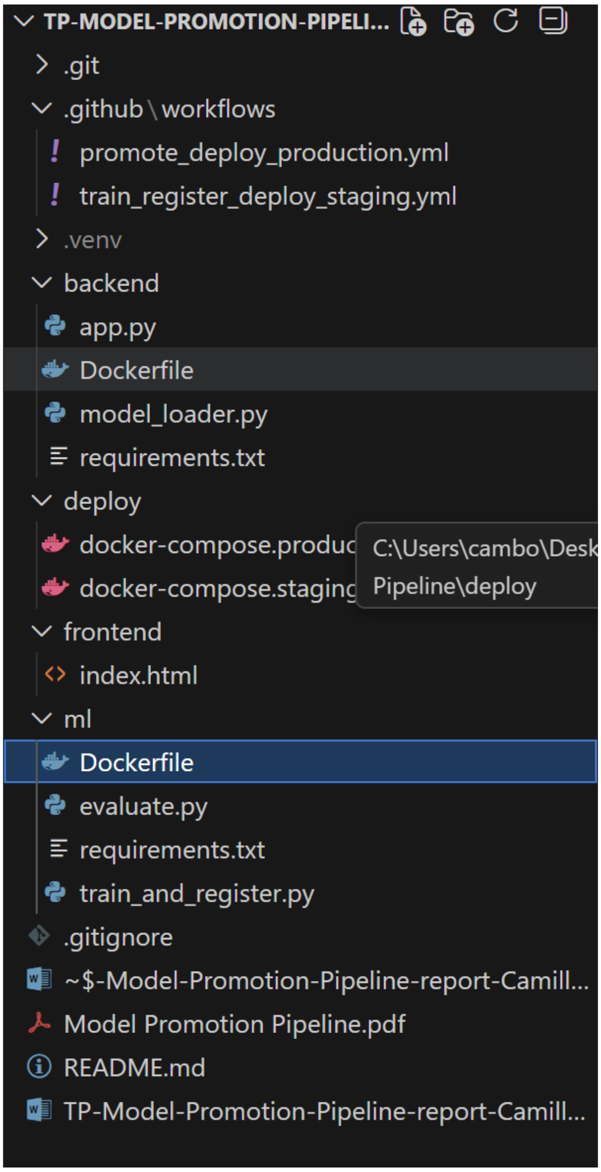
28/01/2026

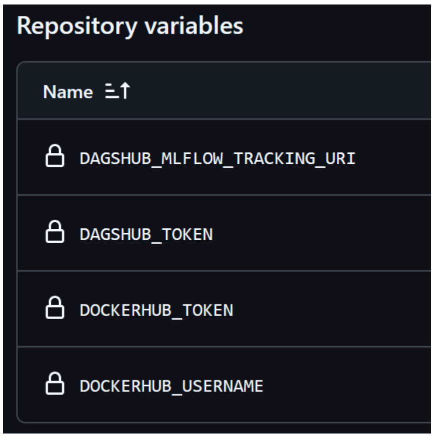
TP-Model-Promotion-Pipeline

Machine Learning in Production

Camille BORDES
GROUP DAI

Configuration :





Task 1: Run Candidate -> Staging workflow

```
Evaluate gate                                                                 0s
▾ Run echo '{"run_id": "973906543027475ca08b3f182da8ddd7", "accuracy": 0.8975, "model_version": 1}' | python ml/evaluate.py
  echo '{"run_id": "973906543027475ca08b3f182da8ddd7", "accuracy": 0.8975, "model_version": 1}' | python ml/evaluate.py
  shell: /usr/bin/bash -e {0}
  env:
    pythonLocation: /opt/hostedtoolcache/Python/3.11.14/x64
    PKG_CONFIG_PATH: /opt/hostedtoolcache/Python/3.11.14/x64/lib/pkgconfig
    Python_ROOT_DIR: /opt/hostedtoolcache/Python/3.11.14/x64
    Python2_ROOT_DIR: /opt/hostedtoolcache/Python/3.11.14/x64
    Python3_ROOT_DIR: /opt/hostedtoolcache/Python/3.11.14/x64
    LD_LIBRARY_PATH: /opt/hostedtoolcache/Python/3.11.14/x64/lib
    ACCURACY_THRESHOLD: 0.90
{"passed": false, "accuracy": 0.8975, "threshold": 0.9, "model_version": 1, "run_id": "973906543027475ca08b3f182da8ddd7"}
Error: Process completed with exit code 2.
```

- **Model version number** : 1.
- **Accuracy** : 0.8975.
- **Did the gate pass?** : **No**

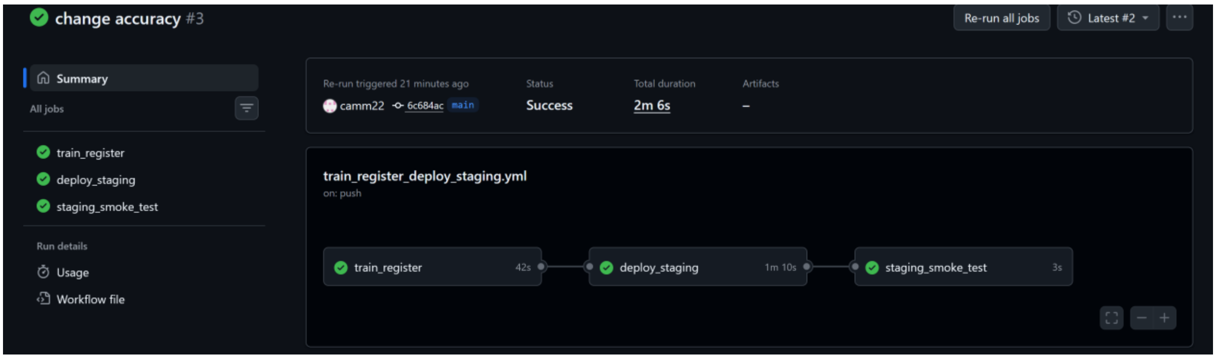
Task 2: Explain what "staging" proves

What staging tests that offline evaluation does not ?

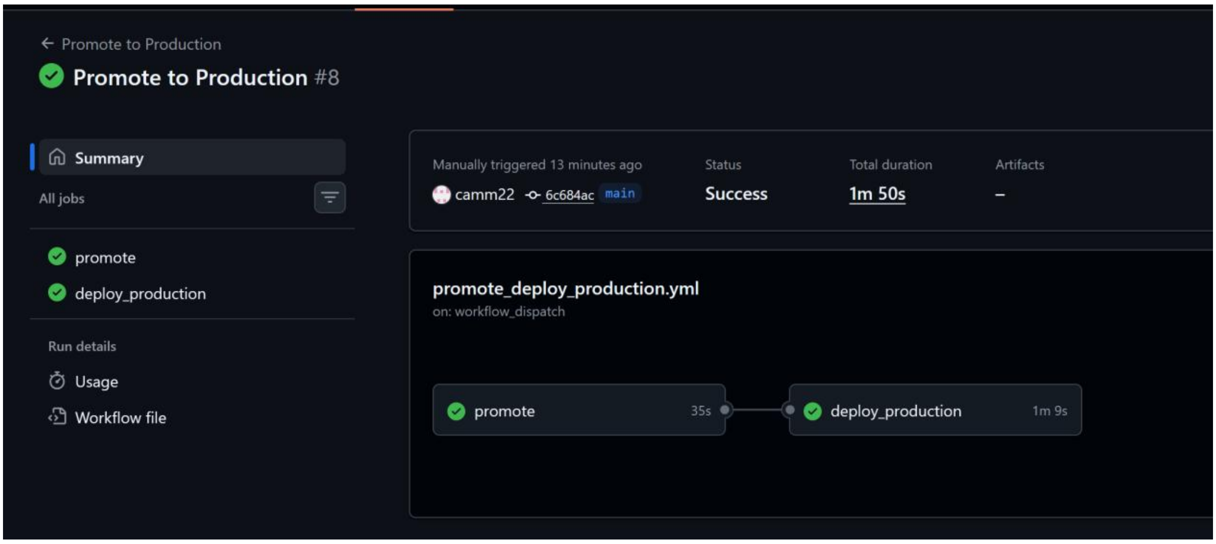
In a production pipeline, staging serves to prove operational readiness by validating factors that offline evaluation cannot, such as the successful integration of the model within a Flask API, the correctness of Docker environment configurations including all dependencies, and the model's ability to handle real-time API requests and schema compatibility through the /predict endpoint. While offline evaluation focuses solely on mathematical metrics like accuracy, staging ensures the entire system, from model loading in MLflow to serving predictions, works as a cohesive unit in a production-like setting.

Task 3: Promote to production

I changed the ACCURACY_THRESHOLD to 0.85 to the model with the Candidate to Staging workflow :



So now we can launch the Promote to Production :



Task 4: Prove production uses registry stage, not "latest code"

Staging:

```
PS C:\Users\cambo> curl http://localhost:8000/health

Avertissement de sécurité : risque d'exécution de script
Invoke-WebRequest analyse le contenu de la page web. Il se peut que le code de script de la page web s'exécute lors de l'analyse de la page.
      ACTION RECOMMANDÉE :
      Utilisez le commutateur -UseBasicParsing pour éviter l'exécution du code de script.

      Voulez-vous continuer ?

[O] Oui  [T] Oui pour tout  [N] Non  [U] Non pour tout  [S] Suspendre  [?] Aide (la valeur par défaut est « N ») : t


StatusCode        : 200
StatusDescription : OK
Content           : {"stage":"Staging","status":"ok"}

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 34
                    Content-Type: application/json
                    Date: Wed, 28 Jan 2026 23:37:01 GMT
                    Server: Werkzeug/3.1.5 Python/3.11.14

                    {"stage":"Staging","status":"ok"}
                    ...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 34], [Content-Type, application/json], [Date, Wed, 28 Jan 2026 23:37:01 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 34
```

Production:

```
PS C:\Users\cambo> curl http://localhost:8001/health


StatusCode        : 200
StatusDescription : OK
Content           : {"stage":"Production","status":"ok"}

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 37
                    Content-Type: application/json
                    Date: Wed, 28 Jan 2026 23:42:59 GMT
                    Server: Werkzeug/3.1.5 Python/3.11.14

                    {"stage":"Production","status":"ok...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 37], [Content-Type, application/json], [Date, Wed, 28 Jan 2026 23:42:59 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 37
```

Discussion questions

1. Why is it dangerous to deploy "whatever just merged to main" as the model?

- **Performance Uncertainty**: A code merge only guarantees that the scripts are syntactically correct and pass unit tests, but it does not guarantee that the resulting model meets performance requirements.
- **Quality Control**: Without a "Quality Gate," a model with poor accuracy or high loss could be deployed to production, potentially causing business failures.

- **Stochastic Nature**: Model training is often stochastic; the same code can produce different results across runs, making it essential to validate the specific artifact (the model file) rather than just the code.

2. What does the registry stage give you that a Git tag does not?

- **Dynamic Decoupling**: Registry stages (Staging/Production) allow you to update which model is "live" without changing the code or rebuilding Docker images.
- **State Management**: It provides a centralized source of truth for the model's lifecycle that is independent of the Git branch, allowing different environments (ports 8000 vs 8001) to pull specific versions based on their status.
- **Auditability**: Registries maintain metadata about who promoted a model and when, which is more difficult to track and restrict via standard Git tags in a collaborative ML environment.

3. If staging passes but production fails, what could be the causes?

- **Data Drift**: The real-world data in production may differ significantly from the test/validation data used in staging, leading to poor model performance.
- **Scale and Load**: Production often handles much higher traffic, which might expose latency issues or memory leaks that weren't visible in the staging "smoke test."
- **Configuration Drift**: Discrepancies in environment variables, hardware (CPU vs GPU), or software dependencies between the two environments.

4. Where should DVC fit in a serious pipeline?

- **Training Data Snapshot**: DVC should version the exact dataset used for training to ensure reproducibility of the model version stored in MLflow.
- **Evaluation Dataset Snapshot**: It must version the "Golden Dataset" used by the Quality Gate to ensure that model comparisons over time are consistent.
- **Drift Reference Dataset**: DVC should store the baseline data distribution used to detect if production data has started to deviate from training data.

5. What should be added to the gate beyond accuracy?

- **Latency**: Ensure the model predicts within a specific time limit (e.g., < 100ms) to meet User Experience requirements.
- **Schema Checks**: Verify that the input data format and feature types haven't changed, preventing the API from crashing.

- **Fairness Constraints**: Check for biases against specific subgroups to ensure ethical and legal compliance.
- **Adversarial/Robustness Tests**: Test the model against edge cases or intentionally noisy data to ensure it doesn't fail catastrophically in unexpected scenarios