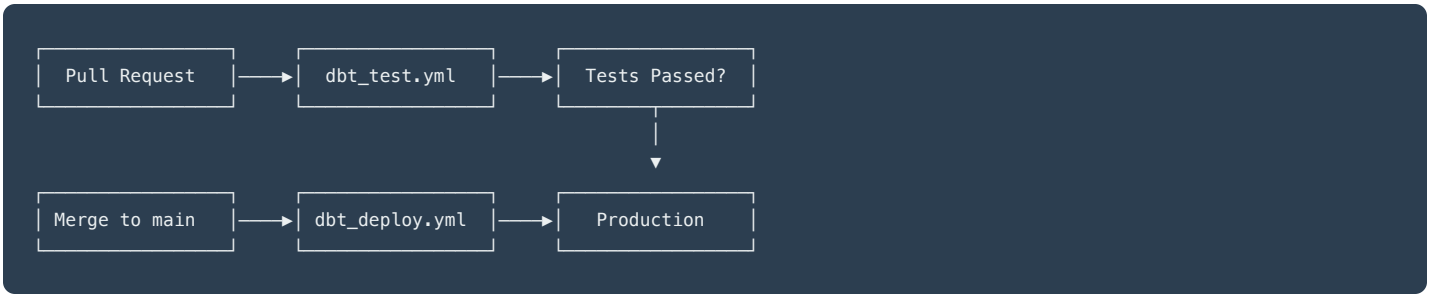
🚀 FraudLens - CI/CD Documentation

GitHub Actions Workflows

Vue d'ensemble

FraudLens utilise GitHub Actions pour automatiser les tests et le déploiement des modèles dbt. Les workflows assurent la qualité du code et le déploiement continu vers Snowflake.



Workflows GitHub Actions

1. dbt_test.yml - Tests sur Pull Request

Attribut	Valeur
Déclencheur	Pull Request vers main
Condition	Modifications dans `dbt/**`
Fichier	`.github/workflows/dbt_test.yml`

Étapes:

1. Checkout du code
2. Installation de Python 3.11
3. Installation de dbt-snowflake
4. Configuration de la clé privée Snowflake
5. Création du profiles.yml
6. Vérification de la connexion (`dbt debug`)
7. Installation des packages (`dbt deps`)
8. Compilation des modèles (`dbt compile`)
9. Exécution des tests (`dbt test`)
10. Génération de la documentation (`dbt docs generate`)

2. dbt_deploy.yml - Déploiement sur main

Attribut	Valeur
Déclencheur	Push sur main
Condition	Modifications dans `dbt/**`
Fichier	`.github/workflows/dbt_deploy.yml`

Étapes:

1. Checkout du code
2. Installation de Python 3.11 + dbt-snowflake
3. Configuration clé privée + profiles.yml
4. Exécution modèles Staging (`dbt run --select staging`)
5. Exécution modèles Silver (`dbt run --select silver`)
6. Exécution modèles Gold (`dbt run --select gold`)
7. Exécution des tests
8. Génération de la documentation

Secrets GitHub

> **Important:** Les secrets doivent être configurés dans Settings > Secrets and variables > Actions

Secret	Description	Exemple
SNOWFLAKE_ACCOUNT	Identifiant du compte	rdsnbdu-gbc86569
SNOWFLAKE_USER	Nom d'utilisateur	FISHER
SNOWFLAKE_WAREHOUSE	Nom du warehouse	FRAUDLENS_WH
SNOWFLAKE_DATABASE	Nom de la base	FRAUDLENS_DB
SNOWFLAKE_PRIVATE_KEY	Contenu clé RSA	-----BEGIN PRIVATE KEY-----...

Authentification Snowflake

Méthode: Clé Privée RSA

L'authentification utilise une clé privée RSA au lieu d'un mot de passe pour une sécurité renforcée.

Avantages:

- Plus sécurisé qu'un mot de passe
- Pas d'expiration automatique
- Compatible avec les politiques de sécurité enterprise

Configuration dans le workflow

```
- name: Setup Snowflake private key
  run: |
    mkdir -p $HOME/.snowflake
    echo "${{ secrets.SNOWFLAKE_PRIVATE_KEY }}" > $HOME/.snowflake/rsa_key.p8
    chmod 600 $HOME/.snowflake/rsa_key.p8

- name: Create profiles.yml for CI
  run: |
    mkdir -p $HOME/.dbt
    cat > $HOME/.dbt/profiles.yml << EOF
    fraudlens:
      target: prod
      outputs:
        prod:
          type: snowflake
          account: ${{ secrets.SNOWFLAKE_ACCOUNT }}
          user: ${{ secrets.SNOWFLAKE_USER }}
          private_key_path: $HOME/.snowflake/rsa_key.p8
          database: ${{ secrets.SNOWFLAKE_DATABASE }}
          warehouse: ${{ secrets.SNOWFLAKE_WAREHOUSE }}
          schema: STAGING
          threads: 4
    EOF
```

Flux de travail développeur

1. Créer une branche feature: `git checkout -b feature/ma-feature`
2. Modifier les fichiers dbt dans `dbt/models/`
3. Commit et Push: `git push -u origin feature/ma-feature`
4. Créer une Pull Request: `gh pr create`
5. Vérifier les tests CI (dbt_test.yml se lance automatiquement)
6. Merger la PR une fois les tests passés
7. Le déploiement (dbt_deploy.yml) s'exécute automatiquement

Dépannage

Erreur	Cause	Solution
No such file: rsa_key.p8	Le chemin ~ n'est pas expansé	Utiliser $HOME au lieu de ~
profiles.yml not found	Mauvais répertoire	Ajouter --profiles-dir $HOME/.dbt
Connection test failed	Secrets mal configurés	Vérifier les secrets GitHub

---

Bonnes pratiques

- **Tests avant merge:** Toujours attendre que les tests CI passent
- **Revue de code:** Faire reviewer les changements dbt par un pair
- **Commits atomiques:** Un commit = une modification logique
- **Messages clairs:** feat:, fix:, docs: prefixes
- **Ne jamais commit de secrets:** Fichiers sensibles dans .gitignore

---

Monitoring

```
# Voir les runs GitHub Actions
gh run list --limit 10

# Voir les logs d'un run
gh run view <run-id> --log

# Voir les logs d'échec
gh run view <run-id> --log-failed
```