

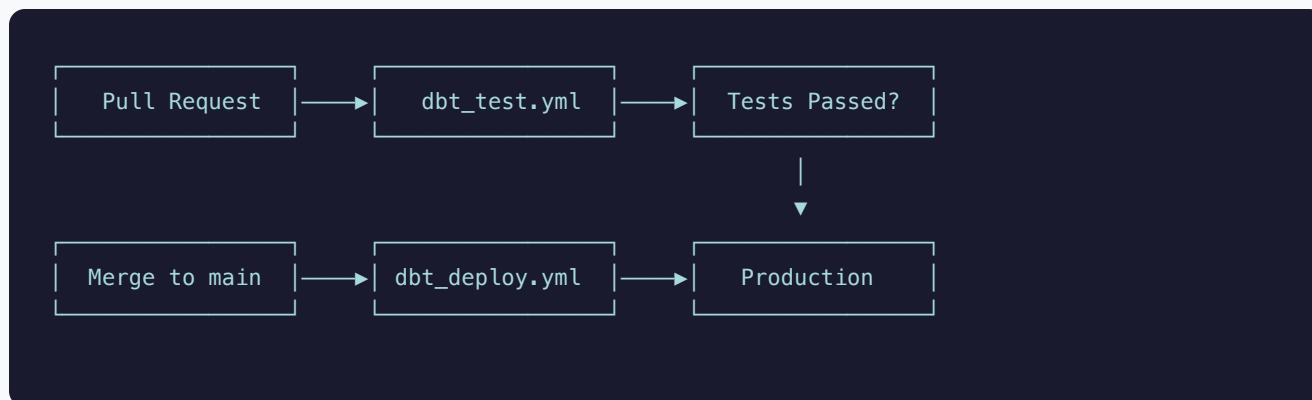
# FraudLens - CI/CD

Documentation GitHub Actions

## 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.

## Architecture CI/CD



## Workflows GitHub Actions

### 1. dbt\_test.yml - Tests sur Pull Request

Attribut	Valeur
----------	--------

Declencheur	Pull Request vers <code>main</code>
Condition	Modifications dans <code>dbt/**</code>
Fichier	<code>.github/workflows/dbt_test.yml</code>

### Etapes:

1. Checkout du code
2. Installation de Python 3.11
3. Installation de dbt-snowflake
4. Configuration de la cle privée Snowflake
5. Creation du profiles.yml
6. Verification de la connexion ( `dbt debug` )
7. Installation des packages ( `dbt deps` )
8. Compilation des modeles ( `dbt compile` )
9. Execution des tests ( `dbt test` )
10. Generation de la documentation ( `dbt docs generate` )

## 2. dbt\_deploy.yml - Deploiement sur main

Attribut	Valeur
Declencheur	Push sur <code>main</code>
Condition	Modifications dans <code>dbt/**</code>
Fichier	<code>.github/workflows/dbt_deploy.yml</code>

### Etapes:

1. Checkout du code
2. Installation de Python 3.11
3. Installation de dbt-snowflake

4. Configuration de la cle privée Snowflake
5. Creation du profiles.yml
6. Verification de la connexion
7. Installation des packages
8. Execution des modeles Staging ( `dbt run --select staging` )
9. Execution des modeles Silver ( `dbt run --select silver` )
10. Execution des modeles Gold ( `dbt run --select gold` )
11. Execution des tests
12. Generation de la documentation

### 3. lint.yml - Linting SQL (Optionnel)

Attribut	Valeur
Declencheur	Pull Request
Outil	SQLFluff

## Secrets GitHub

**Important:** Les secrets doivent etre configures dans **Settings > Secrets and variables > Actions**

Secret	Description	Exemple
<code>SNOWFLAKE_ACCOUNT</code>	Identifiant du compte Snowflake	rdsnbdu-gbc86569
<code>SNOWFLAKE_USER</code>	Nom d'utilisateur Snowflake	FISHER
<code>SNOWFLAKE_WAREHOUSE</code>	Nom du warehouse	FRAUDLENS_WH

<code>SNOWFLAKE_DATABASE</code>	Nom de la base de donnees	FRAUDLENS_DB
<code>SNOWFLAKE_PRIVATE_KEY</code>	Contenu de la cle private RSA	-----BEGIN PRIVATE KEY----- ~...

# Authentification Snowflake

## Methode: Cle Privée RSA

L'authentification utilise une cle privée RSA (fichier `.p8`) 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 entreprise

## 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
          role: ACCOUNTADMIN
```

```
database: ${ secrets.SNOWFLAKE_DATABASE }}
warehouse: ${ secrets.SNOWFLAKE_WAREHOUSE }}
schema: STAGING
threads: 4
```

EOF

## Flux de travail developpeur

1. Creer 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. Creer une Pull Request: `gh pr create`
5. Verifier les tests CI (dbt\_test.yml se lance automatiquement)
6. Merger la PR une fois les tests passes
7. Le deploiement (dbt\_deploy.yml) s'execute automatiquement

## Depannage

### Erreur: "No such file or directory: rsa\_key.p8"

Cause	Le chemin <code>~</code> n'est pas expanse dans YAML
Solution	Utiliser <code>\$HOME</code> au lieu de <code>~</code>

### Erreur: "profiles.yml not found"

Cause	dbt cherche le profiles.yml dans le mauvais repertoire
Solution	Ajouter <code>--profiles-dir \$HOME/.dbt</code> a toutes les commandes dbt

## Erreur: "Connection test failed"

Causes	Secrets mal configurés, cle privée incorrecte, compte Snowflake invalide
Vérification	Vérifier les secrets dans GitHub Settings, s'assurer que la cle privée est complète

## Bonnes pratiques

- **Tests avant merge:** Toujours attendre que les tests CI passent avant de merger une PR
- **Revue de code:** Faire reviewer les changements dbt par un pair
- **Commits atomiques:** Un commit = une modification logique
- **Messages clairs:** feat: Add new model, fix: Correct duplicate handling, docs: Update documentation
- **Ne jamais commit de secrets:** Les fichiers sensibles sont 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
```

