Flyway 101

Atelier Devops

L'équipe du PaaP

Auteur : Said Mezghanni

Plan

Plan

Flyway est un outil open source de migration de base de données
Le projet a commencé avec une première release en 2010.

L'outil est disponible sous deux éditions :

- Community Edition
- Teams Edition

Liste des outils offerts par Flyway :

- Command-line : permet d'exécuter Flyway en ligne de commande utile pour les agents docker.
- Maven-plugin : permet d'exécuter Flyway dans un build Maven.
- Gradle-plugin : permet d'exécuter Flyway dans des tasks Gradle.
- Java-API : permet d'exécuter Flyway dans le code Java.
- Plugins communautaire : npm, Jenkins, SpringBoot, Quarkus,...

Il est possible de configurer les outils Flyway par plusieurs moyens :

- variables d'environnements
- fichier de config : flyway.conf
- fichier de config par migration : myscript.sql.conf
- placeholders : permet de faire du templating sur les scripts de migrations suivant des paramètres prédéfinis
- config SSL : il faut injecter des certificats dans les formats Java (keystore et trust store) à la JVM où Flyway va s'exécuter puis configurer la connexion à la BD pour enforcer le ssl.

Liste des bases de données supportées par la version 7 :

Oracle, SQL Server (including Amazon RDS and Azure SQL Database), Azure Synapse (Formerly Data Warehouse), DB2, MySQL (including Amazon RDS, Azure Database & Google Cloud SQL), Aurora MySQL, MariaDB, Percona XtraDB Cluster, TestContainers, PostgreSQL (including Amazon RDS, Azure Database, Google Cloud SQL & Heroku), Aurora PostgreSQL, Redshift, CockroachDB, SAP HANA, Sybase ASE, Informix, H2, HSQLDB, Derby, Snowflake, SQLite and Firebird.

Plan

Définitions

Avec Flyway, toutes les modifications apportées à la base de données sont appelées **migrations**.
Les migrations peuvent être :

- versionnée
- répétable

Les migrations versionnées se présentent sous 2 formes :

- régulière
- annulée

Type de migrations

- **Les migrations versionnées** ont une version, une description et un checksum. La version doit être unique. Elles sont appliquées dans l'ordre une seule fois.
- **Les migrations d'annulation** : une migration régulière peut être annulée a l'aide d'une migration d'annulation de même version.
- **Les migrations répétables** ont une description et un checksum, mais pas de version. Elles sont réappliquées à chaque fois que leur checksum change.

Language de migration

Les migrations sont le plus souvent écrites en SQL. Ils sont généralement utilisés pour :

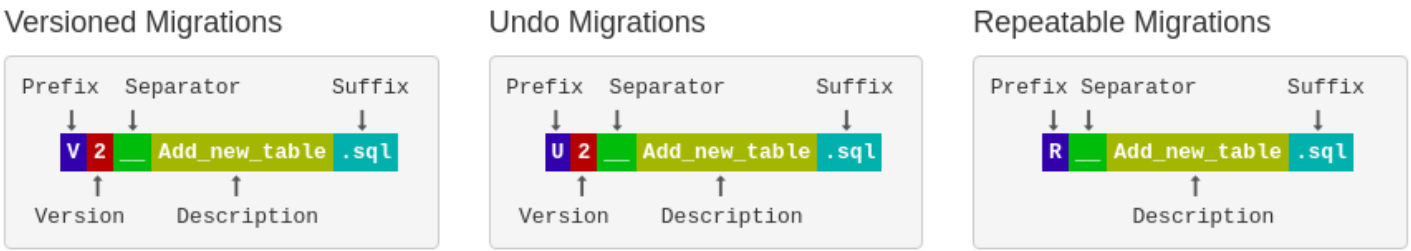
- modifications DDL (CREATE/ALTER/DROP : TABLES,VIEWS,...)
- modifications simples des données (CRUD)
- modifications simples des données en masse

Les migrations basées sur Java ou sur des scripts Bash, Python,... conviennent parfaitement à toutes les modifications qui ne peuvent pas être facilement exprimées à l'aide de SQL :

- modifications BLOB & CLOB
- modifications avancées des données en masse (recalculs, changements de format avancés,...)

Migrations SQL/Script

Nommage :

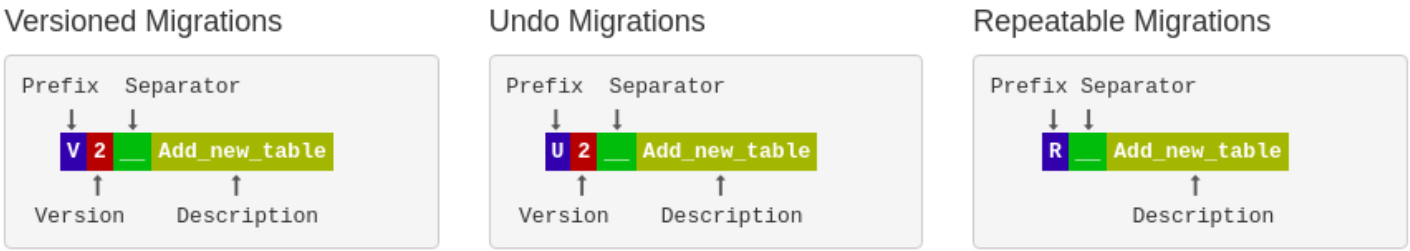


Emplacement des migrations :

- classpath : utilisable dans les builds Java avec les plugins Maven et Gradle
- filesystem : utilisable avec le cli et les plugins
- s3/gcs : à partir de la version 7

Migrations Java

Nommage :



Emplacement des migrations :

- classpath : utilisable dans les builds Java avec les plugins Maven et Gradle. Il faut compiler les classes avant de lancer la migration.

Table d'historique de schéma

Pour garder une trace des migrations qui ont déjà été appliquées, Flyway ajoute une table d'historique de schéma. Elle contient les checksums, les temps et l'état d'exécutions des migrations.

flyway_schema_history

installed_rank	version	description	type	script	checksum	installed_by	installed_on	execution_time	success
1	1	Initial Setup	SQL	V1__Initial_Setup.sql	1996767037	axel	2016-02-04 22:23:00.0	546	true
2	2	First Changes	SQL	V2__First_Changes.sql	1279644856	axel	2016-02-06 09:18:00.0	127	true
3	2.1	Refactoring	JDBC	V2_1__Refactoring		axel	2016-02-10 17:45:05.4	251	true

L'aspect transactionnel :

Par défaut, Flyway encapsule l'exécution de chaque migration dans une seule transaction.

Il est aussi possible de le configurer pour encapsuler l'exécution complète de toutes les migrations dans une seule transaction.

Fonctionnalités avancées

Flyway offre plusieurs fonctionnalités avancées :

- **Callbacks** : définir des callbacks en Java/SQL.
- **Error Overrides** : redéfinir le comportement en cas d'erreur (Teams).
- **Dry Runs** : prévisualiser les modifications que Flyway apportera à la BD (Teams).

Plan

Migrate/Undo

- **Migrate** : migre le schéma vers la dernière version. Flyway créera automatiquement la table d'historique de schéma si elle n'existe pas.
- **Undo** : annule la migration versionnée la plus récemment appliquée (Teams)

Options des commandes migrate/undo :

- **Cherry pick** : spécifiez les migrations à appliquer avec l'option *-cherryPick*. Exemple : *flyway migrate -cherryPick="5,create_view"*
- **Mark as applied** : avec l'option *skipExecutingMigrations=true*, l'exécution du script est ignorée, en mettant à jour la table d'historique de schéma.

Info/Validate

- **Validate** : valide les migrations appliquées par rapport à celles disponibles.
- **Info** : imprime les détails et les informations d'état de toutes les migrations. Les états possibles d'une migration :
  - **success** : la migration est réussie
  - **pending** : la migration est détectée et n'est pas appliquée
  - **failed** : lorsque la migration échoue (BD ne supporte pas les transactions)
  - **undone** : la migration versionnée est annulée par la commande undo
  - **outdated** : la migration répétable dont le checksum a changé
  - **futur** : la migration versionnée appliquée à une version supérieure à la version la plus élevée connue

Clean/Baseline/Repair

- **Clean** : supprime tous les objets dans les schémas configurés
- **Baseline** : sert à introduire Flyway dans les bases de données existant en les basant sur une version spécifique.
- **Repair** : répare la table d'historique de schéma en :
  - supprimant les entrées de migration ayant échoué
  - réalignant les checksums, les descriptions et les types des migrations appliquées avec les migrations disponibles
  - supprimant les migrations répétables supprimées

Plan

Cette démo est réalisé avec une version d'essai Teams :

1. Migrate et Undo en SQL
2. migration repetables avec chagement de checksum
3. migration Bash (bulk insert)
4. migration Java (anonymize)
5. migration Python

Cette démo est réalisé avec une version d'essai Teams :

1. migration au démarrage de l'application avec l'intégration Spring
2. migration à base de plain Java : utilisant une connexion JDBC
3. migration à base de Spring Bean : utilisant Hibernate/ Spring Data

Plan

Pros :

- riche en fonctionnalités
- supporte plusieurs langages/platform/BD
- facilite les tâches de maintenance pour des cas d'utilisation complexes

Cons :

- l'édition communautaire est limitée
- model de facturation par schéma pour la version Teams

Alternatives :

- Liquibase
- Alembic
- Orcas DB
- Dbmate

- Documentation : `https://flywaydb.org/documentation/`
- Blog : `https://flywaydb.org/blog/flyway-7.0.0-beta`
- FAQ : `https://flywaydb.org/documentation/faq`

Merci pour votre attention !