



# DelphiDay

italian conference

## FlayWay

Semplifica e automatizza le migrazioni del tuo database  
con Flyway



# Claudio Piffer

**PC Soft di Piffer Claudio**

@ claudio.piffer@gmail.com

https://github.com/claudio-piffer

linkedin.com/in/piffer-claudio-3599a16a



11-12 Giugno 2024  
Piacenza





# AGENDA

---

1. Migrazione
2. FlyWay
3. FlyWay con Docker



# FlyWay: Migrazione

# 1



# Migrazioni: domande

---

- Quale è lo stato attuale del DB?
- Lo script è già stato applicato? Sei sicuro?
- La patch allo script x è' stata applicata anche all'ambiente di test?
- Come posso configurare una nuova istanza del database con la versione x partendo da zero?





# Migrazioni: cosa sono

---

→ Le migrazioni del database sono una pratica fondamentale nello sviluppo del software che coinvolge la gestione strutturata e automatizzata delle modifiche apportate a uno schema di database o ai dati stessi nel corso del tempo. Queste modifiche possono includere l'aggiunta di nuove tabelle, l'eliminazione di colonne obsolete, la modifica di vincoli, l'inserimento di dati iniziali e altre operazioni che influenzano la struttura o i dati nel database



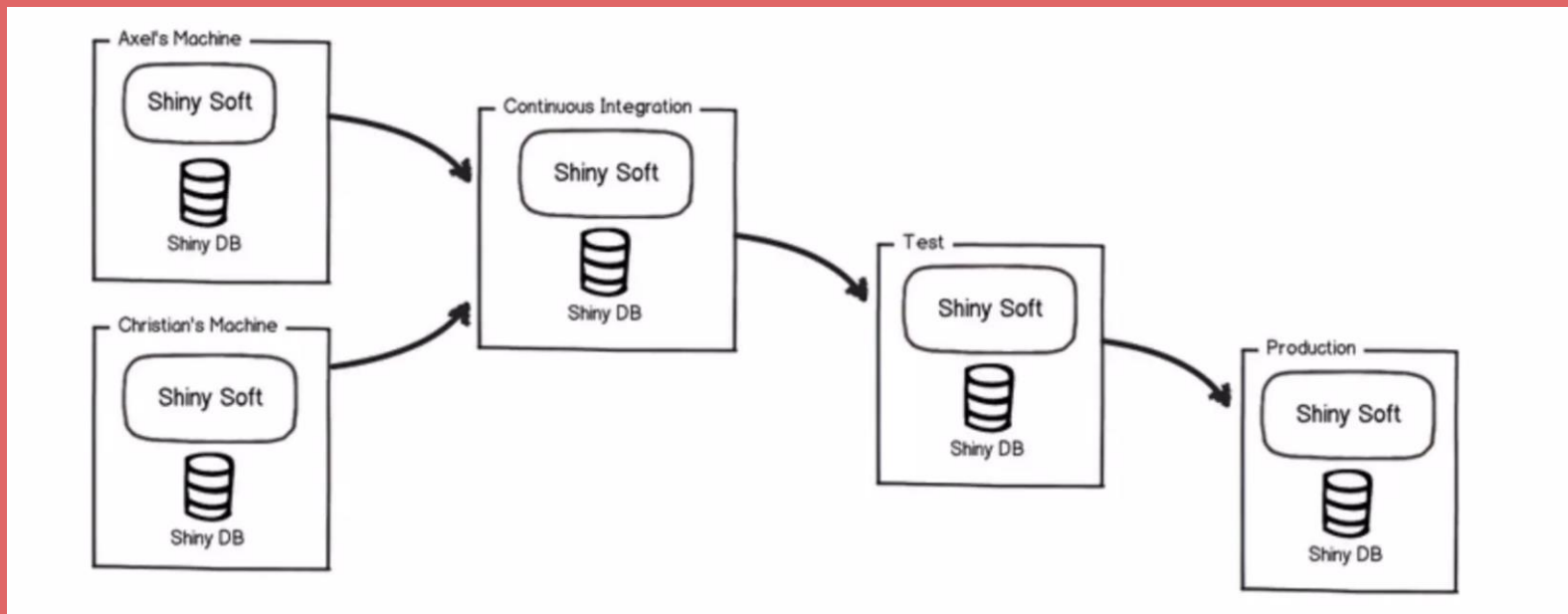
# Migrazioni: perché sono importanti

- **Consistenza del database:** mantengono il database in uno stato consistente tra ambienti diversi, come sviluppo, test e produzione. Ciò assicura che tutti gli sviluppatori e i sistemi stiano lavorando con la stessa versione dello schema del database
- **Aggiornamenti incrementali:** consentono l'applicazione di modifiche in modo incrementale e organizzato. Ogni migrazione rappresenta una singola modifica, semplificando il tracciamento delle modifiche nel tempo
- **Versionamento del database:** le migrazioni vengono versionate, permettendo di registrare quali modifiche sono state applicate, quando e da chi. Questo è fondamentale per la tracciabilità e la gestione del versionamento del software
- **Collaborazione del team:** favoriscono la collaborazione tra membri del team e team diversi, consentendo a più sviluppatori di lavorare contemporaneamente su parti diverse del database senza conflitti
- **Ripristino e rollback:** consentono operazioni di rollback per ripristinare il database a uno stato precedente in caso di errori durante le migrazioni o problemi imprevisti dopo l'implementazione.



# Migrazioni: perché sono importanti

→ In un team possono esistere più istanze del DB e queste devono essere tutte allineate alle relative versioni







# Migrazioni: perché sono importanti

- In un database ci sono molti «oggetti»:
- Tabelle e relazioni
- Viste
- Stored procedures
- Funzioni
- Packages
- Trigger
- Indici
- vincoli
- ...

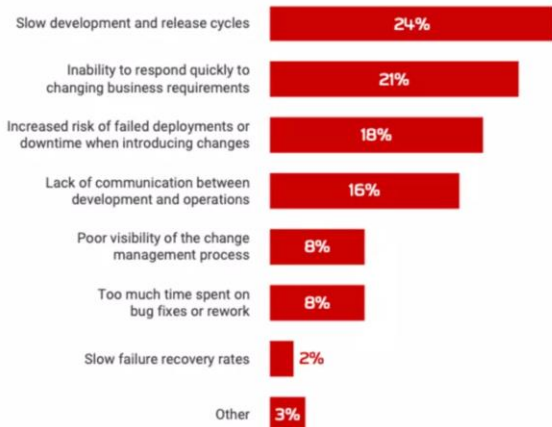




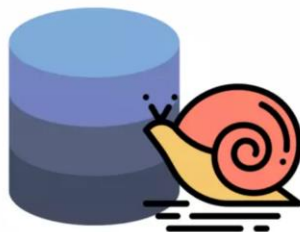
# Migrazioni: perché sono importanti

## 2020 State of Database DevOps

What do you consider the greatest drawback in traditional, siloed database development?



What do you consider the greatest challenge in integrating database changes into a DevOps process?





# FlyWay

# 2



# FlyWay

- ➔ **Gestione delle migrazioni del database:** Flyway è uno strumento di gestione delle migrazioni del database open-source che semplifica il processo di applicazione delle modifiche allo schema del database. Consente agli sviluppatori di definire, versionare e applicare le migrazioni del database in modo coerente e controllato
- ➔ **Versionamento delle migrazioni:** le migrazioni del database sono versionate e gestite come file di script SQL. Ogni migrazione è associata a una versione e può essere applicata in modo incrementale al database. Questo assicura una tracciabilità completa delle modifiche apportate al database nel tempo
- ➔ **Integrazione con sistemi di controllo versione:** Flyway può essere facilmente integrato con sistemi di controllo versione come Git, SVN e altri. Questo significa che le migrazioni del database possono essere gestite insieme al codice sorgente dell'applicazione, garantendo coerenza tra il codice e lo schema del database
- ➔ **Automazione e sicurezza:** Flyway automatizza l'applicazione delle migrazioni del database, riducendo il rischio di errori umani. Le migrazioni possono essere applicate in modo transazionale, il che significa che verranno eseguite solo se tutte le migrazioni precedenti sono riuscite, garantendo l'integrità dei dati e la sicurezza del database
- ➔ **Supporto per diversi database:** Flyway supporta una vasta gamma di database tra cui PostgreSQL, Oracle, SQL Server, Firebird, MySQL e altri. Ciò consente agli sviluppatori di utilizzare Flyway in progetti che coinvolgono più database eterogenei, mantenendo comunque una gestione uniforme delle migrazioni.



# FlyWay

<https://www.red-gate.com/products/flyway/editions>

La versione community è distribuita con licenza Apache versione 2.0

Flyway features & editions

<p><b>Community</b></p> <p>Perfect for individual developers, or non-commercial projects looking for a basic and reliable framework for versioning and automating the deployment of database changes</p> <p><b>Free</b> Open Source Support from the community</p> <p><a href="#">Download</a></p> <p><a href="#">Learn more &gt;</a></p> <p><b>Community includes:</b></p> <ul style="list-style-type: none"><li>• Flyway API/CLI core functionality</li><li>• Flyway Desktop GUI</li><li>• 6 basic commands: Migrate, Clean, Info, Validate, Baseline and Repair</li><li>• Support for current DB versions</li><li>• Community support</li></ul>	<p><b>Enterprise</b></p> <p>Ideal for medium and large enterprises who require secure and automated processes with compliance control to develop, test and deploy database changes</p> <p><b>Contact us</b> Get a quote Redgate support included</p> <p><a href="#">Get in touch</a></p> <p><a href="#">Free trial &gt;</a> <a href="#">Learn more &gt;</a></p> <p><b>Everything in Teams plus:</b></p> <ul style="list-style-type: none"><li>• Change reports**</li><li>• Diff detector**</li><li>• SQL code standard checks**</li><li>• Migration script auto-generation**</li><li>• Schema comparison**</li><li>• Static data versioning***</li><li>• Data comparison***</li><li>• Test Data Management (cloning)</li><li>• Support for restricted set of DB versions</li></ul> <p><small>**SQL Server, Oracle, PostgreSQL, and MySQL only</small></p> <p><small>***SQL Server and Oracle only</small></p>	<p><b>Teams</b></p> <p>Ideal for organizations looking to improve collaboration and fine tune their processes during development and the deployment of database changes</p> <p><b>€555* user/year</b> Redgate support included</p> <p><a href="#">Buy now</a></p> <p><a href="#">Free trial &gt;</a> <a href="#">Learn more &gt;</a></p> <p><small>*100 schema limit. Contact us if you have a requirement for more</small></p> <p><b>Everything in Community plus:</b></p> <ul style="list-style-type: none"><li>• Dry run</li><li>• Undo migration scripts</li><li>• Cherry pick</li><li>• Skip migrations (mark as deployed)</li><li>• Check-level versioning**</li><li>• Built in GUI client</li><li>• Support for older DB versions</li><li>• Access to technical support</li></ul> <p><small>**SQL Server, Oracle, PostgreSQL, and MySQL only</small></p>
--	---	---



# FlyWay






























---

- Esistono due versioni
  - CLI (quella che useremo in questo talk)
  - Desktop edition (che comprende anche la CLI edition)
- CLI: <https://documentation.red-gate.com/fd/command-line-184127404.html>
- Desktop: <https://www.red-gate.com/products/flyway/community/download>





# FlyWay: database supportati

 Oracle	 MySQL	 Aurora MySQL	 MariaDB	 Percona XtraDB cluster
 SQL Server	 PostgreSQL	 Azure Synapse	 Aurora PostgreSQL	 Redshift
 CockroachDB	 DB2	 SAP HANA	 Sybase ASE	 Informix
 HSQLDB	 H2	 Derby	 Snowflake	 SQLite
 Firebird	 Google BigQuery	 Google Cloud Spanner (Beta)	 Ignite	 SingleStoreDB
 TestContainers	 Titanium DB	 TimescaleDB	 YugabyteDB	



# FlyWay: punti di forza



## Plain old SQL

Plain SQL scripts (including placeholder replacement).  
No proprietary XML formats, no lock-in.



## No limits

Java-based migrations for advanced data  
transformations and handling with LOBs.



## Zero required dependencies

All you need is Java 7+ and your Jdbc driver and  
you're good to go!



## Convention over configuration

Filesystem and classpath scanning to automatically  
discover SQL and Java migrations.



## Highly reliable

Safe for cluster environments. Multiple machines can  
migrate in parallel.



## Cloud support

Full support for Amazon RDS, Microsoft SQL Azure,  
Google Cloud SQL, Heroku, and more.



## Auto-migration on startup

Ship migrations together with the application and run  
them automatically on startup using the API.



## Fail fast

Inconsistent database or failed migration prevents  
app from starting.



## Schema clean

Drop all tables, views, triggers, and more from a  
schema without dropping the schema itself.



# FlyWay: workflow

---

## Definizione delle migrazioni

Come primo step è necessario creare gli script SQL o classi Java/Groovy che rappresentano le migrazioni del database. Questi script contengono le operazioni DDL (Data Definition Language) e DML (Data Manipulation Language) necessarie per modificare lo schema del database/aggiornare, inserire i dati di base



# FlyWay: workflow

---

Organizzazione dei file di migrazione:

I file di migrazione devono seguire una convenzione di denominazione specifica e devono essere organizzati in una posizione specifica nel progetto. Questi file rappresentano le diverse versioni delle migrazioni del database



# FlyWay: convenzione nomi file

---

Flyway segue convenzioni specifiche per il nome dei file di migrazione. Queste convenzioni aiutano Flyway a determinare l'ordine in cui le migrazioni devono essere eseguite. Ecco le regole per il nome dei file di migrazione in Flyway:

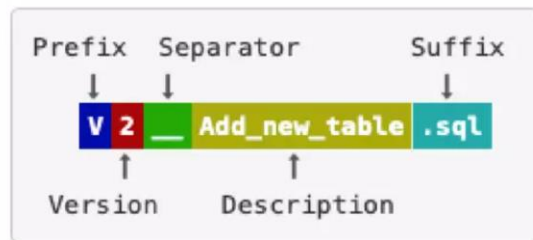
- Prefisso della versione
- Doppio underscore
- Descrizione
- Estensione del file



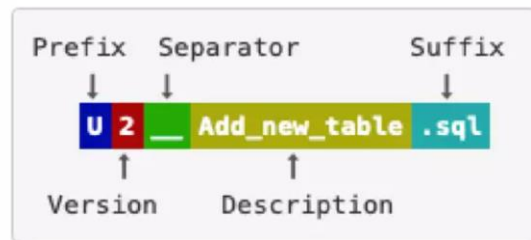
# FlyWay: convenzione nomi file

Convenzione nomi dei file:

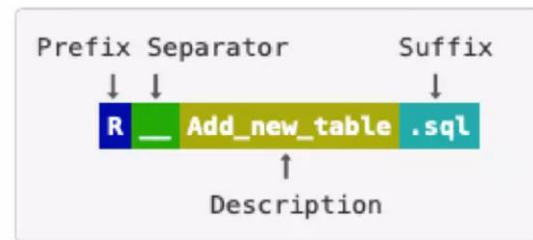
## Versioned Migrations



## Undo Migrations



## Repeatable Migrations







# FlyWay: convenzione nomi file

- ➔ **Versioni standard (V):** V1\_CreaTabella.sql: questi sono i file di migrazione standard e rappresentano le modifiche al database che devono essere applicate esattamente una volta. Flyway eseguirà questi file esattamente nell'ordine numerico del prefisso (V1, V2, ecc.).
- ➔ **Versioni di ripetizione (R):** R\_AggiornaTabella.sql: le versioni di ripetizione rappresentano modifiche che possono essere eseguite più di una volta. Questi script sono utili per operazioni che devono essere sempre aggiornate allo stato più recente, come le viste, le procedure memorizzate, le funzioni, e altri oggetti di database che possono essere ricreati in modo idempotente
- ➔ **Versioni di undo (U):** U1\_CreaTabella.sql: questi file contengono istruzioni per annullare le modifiche effettuate da una migrazione precedente. Sono utilizzati per eseguire operazioni di rollback e devono essere eseguiti manualmente se necessario. (non community edition)
- ➔ **Script di errore (E):** E1\_ErrorePersonalizzato.sql: i file di errore contengono comandi SQL che vengono eseguiti se una migrazione fallisce. Sono utilizzati per gestire situazioni di errore personalizzate.



# FlyWay: history table

---

Quando Flyway esegue migrazioni nel database, tiene traccia dello stato delle migrazioni applicate attraverso una tabella chiamata "schema\_version". Questa tabella è creata automaticamente da Flyway nel database di destinazione se non esiste già. La tabella schema\_version contiene informazioni sulle migrazioni applicate, come versione, descrizione, stato e data di esecuzione



# FlyWay: comandi principali



## Flyway Commands

Name	Description
<code>migrate</code>	Migrates the database
<code>clean</code>	Drops all objects in the configured schemas
<code>info</code>	Prints the details and status information about all the migrations
<code>validate</code>	Validates the applied migrations against the ones available on the classpath
<code>undo</code> <span>Flyway Teams</span>	Undoes the most recently applied versioned migrations
<code>baseline</code>	Baselines an existing database, excluding all migrations up to and including baselineVersion
<code>repair</code>	Repairs the schema history table



# FlyWay: comandi principali

- ➔ **migrate**: esegue le migrazioni nel database di destinazione. Questo è il comando principale che applica le nuove migrazioni che non sono ancora state eseguite nel database (<https://documentation.red-gate.com/fd/migrate-184127460.html>)
- ➔ **clean**: elimina lo schema del database attuale. Questo comando cancella tutte le tabelle del database, compresa la tabella schema\_version (<https://documentation.red-gate.com/fd/clean-184127458.html>)
- ➔ **info**: mostra lo stato delle migrazioni nel database di destinazione, indicando quali migrazioni sono state applicate, quale versione è attuale e se ci sono migrazioni in sospeso (<https://documentation.red-gate.com/fd/info-184127459.html>)
- ➔ **validate**: verifica l'integrità delle migrazioni applicate rispetto ai file di migrazione sul filesystem. Questo comando confronta i checksum dei file di migrazione con quelli memorizzati nel database (<https://documentation.red-gate.com/fd/validate-184127464.html>)
- ➔ **baseline**: crea un punto di partenza (baseline) nel database di destinazione, in modo che le migrazioni successive vengano applicate solo a partire da questo punto (<https://documentation.red-gate.com/fd/baseline-184127456.html>)
- ➔ **repair**: Corregge eventuali discrepanze tra lo stato delle migrazioni nel database e i file di migrazione sul filesystem (<https://documentation.red-gate.com/fd/repair-184127461.html>)



# FlyWay: callback

Le callbacks possono essere utilizzate per implementare logiche personalizzate, operazioni di pre-elaborazione o post-elaborazione, verifica dei dati, o qualsiasi altra logica necessaria. Ad esempio è possibile creare uno script .sql con i seguenti nomi (le callback sono richiamabili anche in java):

- beforeMigrate.sql: viene eseguito prima di ogni singola migrazione.
- afterMigrate.sql : viene eseguito dopo ogni singola migrazione.
- beforeValidate.sql : viene eseguito prima di ogni singola validazione.
- afterValidate.sql : viene eseguito dopo ogni singola validazione.
- beforeClean.sql : viene eseguito prima del processo di pulizia.
- afterClean.sql : viene eseguito dopo il processo di pulizia

→ La lista completa la trovate a questo link:

→ <https://documentation.red-gate.com/fd/callback-concept-184127466.html>



# FlyWay: file .conf

---

Il file di configurazione di Flyway è utilizzato per specificare le impostazioni e le configurazioni del processo di migrazione del database. Questo file, chiamato flyway.conf, è un file di testo che contiene vari parametri e valori di configurazione che vengono letti da Flyway quando viene eseguito





# FlyWay: file .conf

```
northwind_flyway.conf X
D: > DockerCourse > northwind_psql > flyway > conf > northwind > northwind_flyway.conf
27 # Uerby : jdbc:derby:<subsubprotocol>:<database>:<attribute=value>
28 # Firebird : jdbc:firebirdsql://<host>[:<port>]/<database>?<key1>=<value1>&<key2>=<value2>...
29 # H2 : jdbc:h2:<file>
30 # HSQLDB : jdbc:hsqldb:file:<file>
31 # Informix* : jdbc:informix-sqli://<host>:<port>/<database>:informixserver=dev
32 # MariaDB : jdbc:mariadb://<host>:<port>/<database>?<key1>=<value1>&<key2>=<value2>...
33 # MySQL : jdbc:mysql://<host>:<port>/<database>?<key1>=<value1>&<key2>=<value2>...
34 # Oracle : jdbc:oracle:thin:@<host>:<port>/<service>
35 # Oracle (TNS)** : jdbc:oracle:thin:@tns_entry
36 # PostgreSQL : jdbc:postgresql://<host>:<port>/<database>?<key1>=<value1>&<key2>=<value2>...
37 # SAP HANA* : jdbc:sap://<host>:<port>/<databaseName>=<database>
38 # Snowflake* : jdbc:snowflake://<account>.snowflakecomputing.com/<db>-<database>&warehouse=<warehouse>&role=<role>...
39 # SQL Server : jdbc:sqlserver://<host>:<port>/<databaseName>=<database>
40 # SQLite : jdbc:sqlite:<database>
41 # Sybase ASE : jdbc:jtds:sybase://<host>:<port>/<database>
42 # Redshift* : jdbc:redshift://<host>:<port>/<database>
43 flyway.url=jdbc:postgresql://127.0.0.1:45432/nwindpg
44
45 # Fully qualified classname of the JDBC driver (autodetected by default based on flyway.url)
46 flyway.driver=
47
48 # User to use to connect to the database. Flyway will prompt you to enter it if not specified, and if the JDBC
49 # connection is not using a password-less method of authentication.
50 flyway.user=nwind
51
52 # Password to use to connect to the database. Flyway will prompt you to enter it if not specified, and if the JDBC
53 # connection is not using a password-less method of authentication.
54 flyway.password=nwind$
55
56 # The maximum number of retries when attempting to connect to the database. After each failed attempt,
57 # Flyway will wait 1 second before attempting to connect again, up to the maximum number of times specified
58 # by connectRetries. (default: 0)
59 flyway.connectRetries=
60
61 # The SQL statements to run to initialize a new database connection immediately after opening it. (default: none)
62 flyway.initSql=
63
64 # The default schema managed by Flyway. This schema name is case-sensitive. If not specified, but <i>flyway.schemas</i> is, Flyway uses the first schema
65 # in that list. If that is also not specified, Flyway uses the default schema for the database connection.
66 # Consequences:
67 # - This schema will be the one containing the schema history table.
68 # - This schema will be the default for the database connection (provided the database supports this concept).
69 flyway.defaultSchema=northwind
70
71 # Comma-separated list of schemas managed by Flyway. These schema names are case-sensitive. If not specified, Flyway uses
72 # the default schema for the database connection. If <i>flyway.defaultSchema</i> is not specified, then the first of
73 # this list also acts as default schema.
```



# FlyWay: parametri

- **flyway.url**: URL del database a cui connettersi per eseguire le migrazioni
- **flyway.user** e **flyway.password**: le credenziali di accesso per il database. Flyway utilizzerà queste credenziali per autenticarsi al database
- **flyway.schemas**: uno o più nomi di schemi del database separati da virgole. Questo parametro specifica a quali schemi del database applicare le migrazioni
- **flyway.locations**: il percorso o i percorsi in cui Flyway troverà i file di migrazione. Questi possono essere percorsi del sistema di file o percorsi di classpath (inclusi all'interno del classpath dell'applicazione)
- **flyway.sqlMigrationPrefix** e **flyway.sqlMigrationSuffix**: prefisso e suffisso dei file di migrazione SQL. Ad esempio, con un prefisso V e un suffisso .sql, i file di migrazione potrebbero chiamarsi V1\_migrazione.sql
- **flyway.placeholders**: un elenco di coppie chiave-valore che rappresentano i segnaposto da utilizzare nei file di migrazione SQL. Questi possono essere sostituiti con valori specifici durante l'esecuzione delle migrazioni
- **flyway.outOfOrder**: un booleano che specifica se le migrazioni devono essere eseguite fuori ordine. Se è vero, le migrazioni vengono eseguite in base al numero di versione e non all'ordine di arrivo
- **flyway.cleanOnValidationError**: un booleano che indica se eliminare lo schema del database se una migrazione fallisce durante la validazione. Utile per mantenere il database pulito in caso di errori
- **flyway.baselineOnMigrate**: un booleano che specifica se creare una migrazione di base se non esiste. La migrazione di base è una versione iniziale del database, utilizzata come punto di partenza per le migrazioni successive
- **flyway.target**: la versione del database verso cui effettuare le migrazioni. Le migrazioni saranno eseguite fino a questa versione inclusa
- **flyway.connectRetries**: il numero massimo di tentativi di connessione al database in caso di fallimento
- **flyway.table**: il nome della tabella di controllo delle migrazioni. Di default, questa tabella è chiamata schema\_version
- **flyway.resolvers**: un elenco di nomi di classi che estendono MigrationResolver per personalizzare la logica di risoluzione delle migrazioni
- **flyway.callbacks**: un elenco di nomi di classi che implementano varie interfacce di callback di Flyway, come FlywayCallback o FlywayConfigurationCustomizer



# FlyWay: parametri base

```
# Redshift*           : jdbc:redshift://<host>:<port>/<database>
flyway.url=jdbc:postgresql://127.0.0.1:45432/nwindpg
```

```
# connection is not used
flyway.user=nwind
```

```
# connection is not used
flyway.password=nwind$
```

```
# - This schema will be the default schema
flyway.defaultSchema=northwind
```

```
# or in the schema specified
flyway.table=schema_version
```



# FlyWay e Docker

# 3



# FlyWaye Docker demo

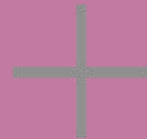
---



docker



PostgreSQL



Flyway



# FlyWay: CI/CD

---

Integrare Flyway in una pipeline CI/CD con Jenkins:

- ➔ **Configurazione del Job Jenkins:** configura un job Jenkins che monitora il repository del tuo progetto. Aggiungi un passaggio di build che esegue il comando Flyway (flyway migrate) per applicare le migrazioni del database. Usa variabili d'ambiente sicure in Jenkins per gestire le credenziali del database.
- ➔ **Trigger automatico delle migrazioni:** configura il job Jenkins in modo che venga attivato automaticamente ogni volta che ci sono nuovi commit nel repository. Ogni push nel repository può innescare automaticamente il processo di migrazione del database
- ➔ **Gestione degli errori e delle notifiche:** aggiungi gestione degli errori nel job Jenkins per gestire scenari in cui una migrazione fallisce. Puoi configurare Jenkins per inviare notifiche via email o Slack in caso di errori. Implementa un meccanismo per interrompere il processo di CI/CD se le migrazioni non vanno a buon fine, impedendo così il rilascio di un'applicazione con uno schema di database inconsistente.





# FlyWay: CI/CD

---

## Integrare Flyway in una pipeline CI/CD con GitLab CI:

- **File .gitlab-ci.yml**: crea un file .gitlab-ci.yml nel tuo repository GitLab per definire la pipeline CI/CD. Aggiungi un job dedicato alle migrazioni del database utilizzando Flyway.
- **Variabili d'ambiente protette**: utilizza le variabili d'ambiente protette di GitLab CI per gestire le credenziali del database in modo sicuro
- **Integrazione con GitLab Runner**: assicurati che GitLab Runner sia configurato per eseguire i job di CI/CD. Ogni push nel repository attiverà automaticamente la pipeline e, di conseguenza, eseguirà le migrazioni del database



THANK YOU