Flyway AutoPilot Learning Exercise Book

This Exercise book aims to guide you through the process of not only using Flyway AutoPilot, but truly understanding and onboarding with Flyway. Throughout this document we will present frequent tasks and challenges that Flyway is used for, with instructional guides and learning points, aimed to scale anyone from simply sharing a change with a fellow developer, all the way to creating scalable CI/CD deployments.

Use this guide how you see fit, and please consult any documentation found here!

Flyway
AutoPilot

For Further Support, or queries please contact:

Huxley Kendell
Solutions Engineering
Huxley.Kendell@red-gate.com

redgate

Welcome to the world of Flyway, where database migrations are streamlined, and your database management becomes a well-oiled machine. This exercise book is your essential partner on the path to mastering Flyway and harnessing the advanced capabilities of Flyway AutoPilot as a sandbox environment to learn. Whether you're an experienced database professional or a newcomer looking to streamline your database operations, this book is tailored to guide you through the journey of becoming a proficient Flyway user.

Your Learning Journey

As you flip through these pages, you'll embark on a structured journey that covers every facet of Flyway, ensuring that you're prepared to use Flyway effectively. Here's a brief glimpse of what you'll achieve by the time you complete this guide:

- **Setting Up Your Environment:** Begin by creating the ideal workspace for your learning experience, allowing you to test on easily provisioned environments, or your own depending on experience and needs.
- **Learning Core Concepts:** Dive into the foundational principles of Flyway, including version control, schema evolution, and the creation of migration scripts.
- **Deploying with Flyway:** Explore the world of automated database migrations, ensuring that your deployment processes are efficient and error-free.
- **Achieving Scalable CI/CD Deployments:** Discover how to seamlessly integrate Flyway into your Continuous Integration/Continuous Deployment (CI/CD) pipelines, making your database migrations scalable and hassle-free.

Your Learning Experience

This exercise book is designed with your needs in mind. Whether you prefer a step-by-step approach or need quick answers to specific challenges, this book accommodates various learning styles. Each section corresponds to a specific skill or concept, allowing you to focus on what matters most to you at any given time.

Additionally, we encourage you to explore the official Flyway documentation for in-depth knowledge and keep your skills up to date as Flyway evolves.

We're excited to be your guide on this journey to mastering Flyway and Flyway AutoPilot. By the time you reach the end of this exercise book, you'll have the knowledge and confidence to tackle real-world database management tasks with ease. Let's embark on this exciting learning adventure together!

Contents

Background

1. Flyway

- It is important that we have installed Flyway Desktop on to the machines wishing to participate, Flyway AutoPilot is configured to allow for teams to work together so the scope does not neem to be limited to individual members.
- A link to download everything needed can be found here!

2. DBMS of Choice

- We will need access to the DBMS of your choice, Flyway AutoPilot currently supports MS SQL Server & PostgreSQL, with more on the roadmap. So, make sure each member has access to a shared or dedicated environment to test and learn.

3. IDE of Choice

- Access to some kind of DBMS IDE is also needed, we recommend SQL Server Management Studio (SSMS) for SQL Server and PG Admin for PG, but the choice is up to you!

4. GitHub + GitHub Actions

- Ability to create a GitHub repository, and a GitHub Actions pipeline. This will be configured automatically but access is still needed.

Challenges:

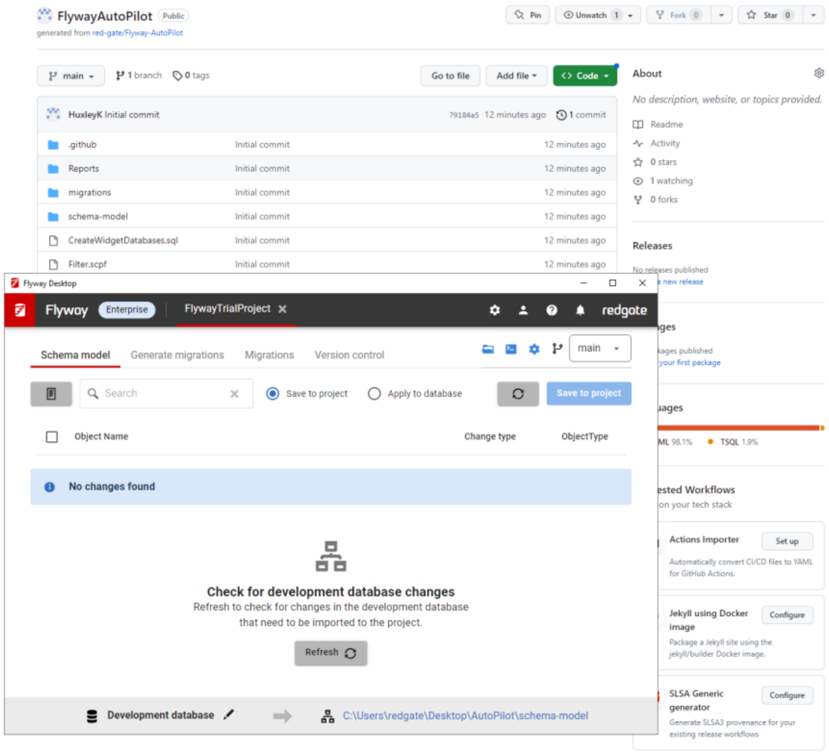
1. Install AutoPilot (Optional)

1. The first step to completing this exercise book is too make sure we have Flyway AutoPilot installed, this gives us the custom environment to properly learn and onboard.

2. This is NOT a mandatory step, you are more than welcome to use this book to guide learning on a different environment, but Flyway AutoPilot is set up to follow these scenarios, if you desire to this without AutoPilot make sure to have a project set-up and refer to Redgate Documentation for additional support!

3. A link to everything you could need is here!
   Once you have completed the video and setup, come back to the exercise book to

being your learning!

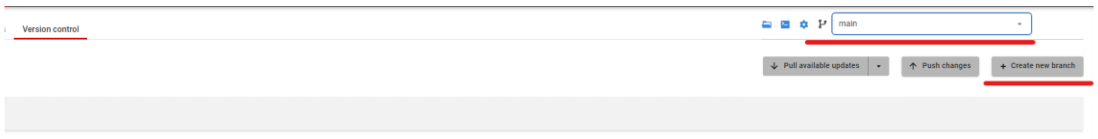


No changes found

Check for development database changes
Refresh to check for changes in the development database that need to be imported to the project.

2. Create Additional Branch

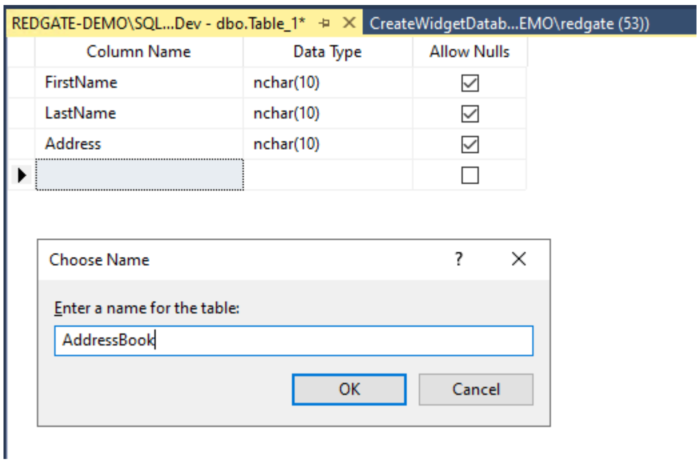
1. Flyway will pick up any branches that were prior made on the Git repository, or made elsewhere! But should the need come up during development or during the use of Flyway, we can also create a branch on the Desktop too, allowing us to keep changes isolated by branches and then merge them into main when ready.
2. Flyway works exceptionally where when coupled with a good branching strategy, so lets head to the “Version Control” tab and see for ourselves how easy it is! Click “+ Create New Branch” and give it a name!



3. We can then change to any branch, this means any schema / migration changes we make are currently isolated to that branch, allowing other teams / projects to code independently!

3. Make & Capture Change

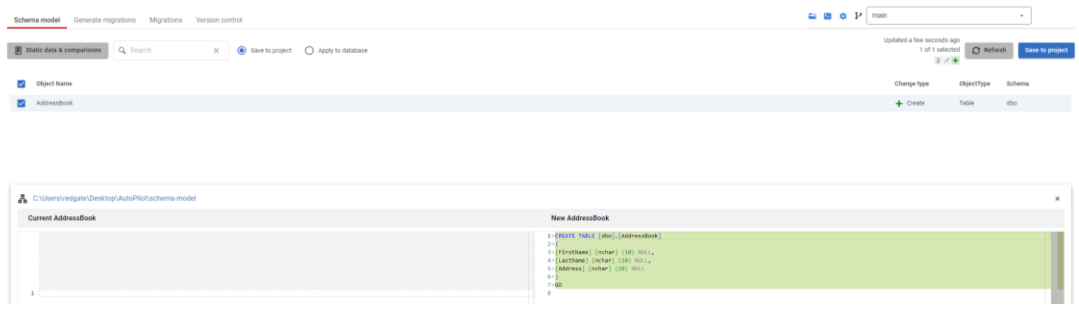
1. The very first thing we want to showcase is simple developer integration, the ability to make a change on a dev environment and share that change with other developers. Head over to the IDE of your choice, and create a object of your choice inside the 'WidgetDev' Database!
   The screenshot below shows me creating a table!


REDGATE-DEMO\SQL...Dev - dbo.Table_1*
CreateWidgetDatab...EMO\redgate (53))
Choose Name
OK
Cancel

Column Name	Data Type	Allow Nulls
FirstName	nchar(10)	✓
LastName	nchar(10)	✓
Address	nchar(10)	✓
		☐

Enter a name for the table: AddressBook

2. Once the change has been made, head over to Flyway and click into the "Schema Model" Tab. This will discover any changes you have made! You can click on the object name for a breakdown of how this object has evolved, and once happy you can decide to bring this object into the Flyway Project!



Save to project [✓] Apply to database

Updated a few seconds ago
1 of 1 selected

Object Name	Change type	ObjectType	Schema
AddressBook	Create	Table	dbo

C:\Users\redgate\Desktop\AutoPilot\schema-model

Current AddressBook

New AddressBook

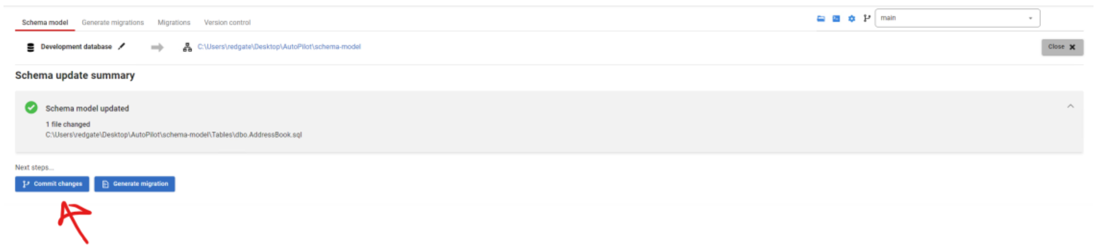
```
CREATE TABLE [dbo].[AddressBook]
(
[FirstName] [nchar] (10) NULL,
[LastName] [nchar] (10) NULL,
[Address] [nchar] (10) NULL
)
GO
```

By clicking Save to Project, we are not deploying any changes, just simply bringing it into our Flyway Project, to track and use at a later time!

3. This feature allows us to track any and all objects inside our database and gain a higher level of visibility into how our DB evolves.

4. Share Changes via Version Control

1. It is then important that these changes leave our own personal copy of the repository, so our next step is to try and share via Version Control.
2. After we Save a change to the project, like we did in Step 2., we will be given a prompt to “Commit Changes”!



C:\Users\redgate\Desktop\AutoPilot\schema-model

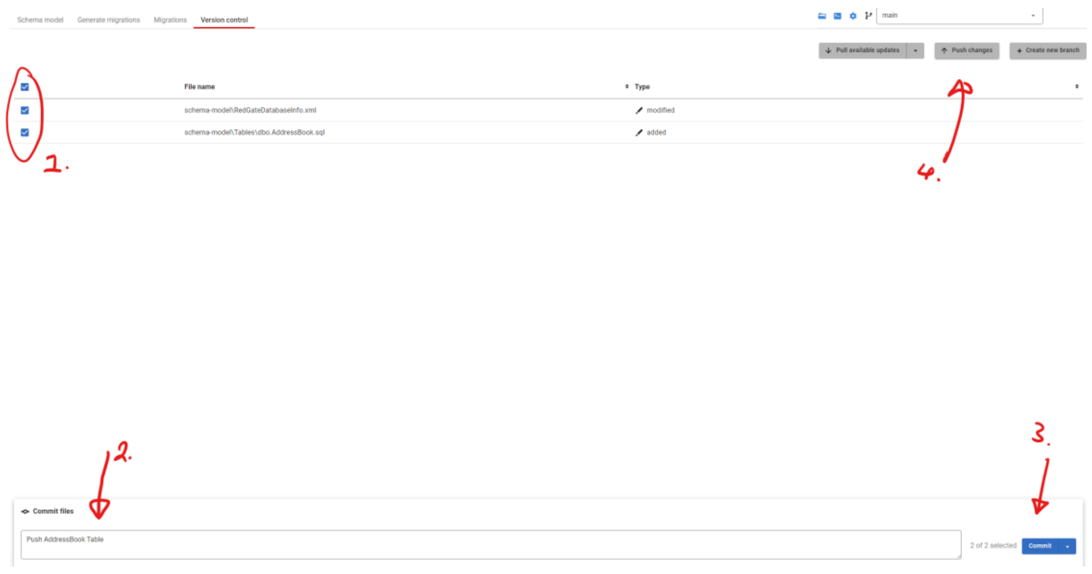
Schema update summary

Schema model updated

1 file changed
C:\Users\redgate\Desktop\AutoPilot\schema-model\Tables\dbo.AddressBook.sql

Next steps...

3.1. If this is skipped, we can navigate to the same page by clicking into the “Version Control” tab. Follow the image below to construct a commit, and push into the central repository!



	File name	Type
✓	schema-model\RedGateDatabaseInfo.xml	modified
✓	schema-model\Tables\dbo.AddressBook.sql	added

Push AddressBook Table

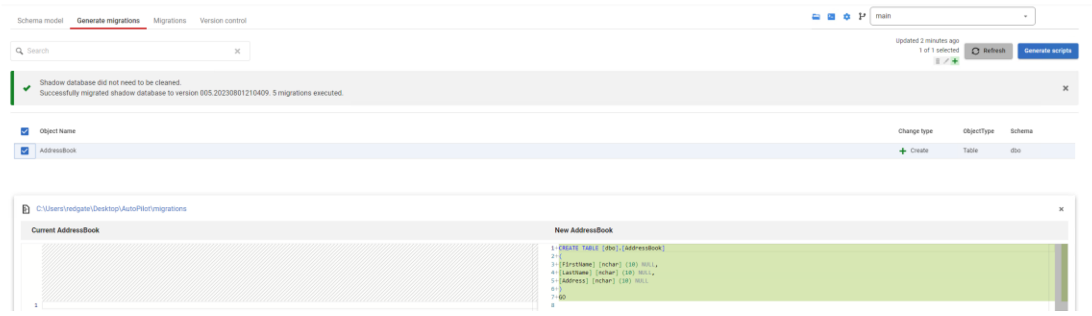
3. The key steps, as shown above, include:
   - Choosing the changes to the repository that you wish to share.

- Writing a Commit Message to communicate the changes with others.
- Committing & Pushing the changes!
- Remember to check which branch you are using before committing any changes!

4. As mentioned, this allows us to not only track changes inside our repository, but to seamlessly share any changes between one another. Once we start creating migration scripts, it will also allows us to pass them to our CI/CD pipelines! But that's for a later challenge!

5. Create Migration Script

1. With the state of development now being ahead of production, it is time to use one of Flyways biggest value points and generate a migration script encompassing our changes. When ready, head over to the "Generate Migrations" tab!
2. Flyway will begin to compare our newly updated Schema Model (this is where Step 2 Saved our changes too) against the state of Production! The first screen (Seen below) shows showcases all the changes that we as a Dev Team have made to the schema model, that has yet to be turned into a Versioned Script. Click on one of the object names, to see how the object has evolved.



Shadow database did not need to be cleaned.
Successfully migrated shadow database to version 005.20230801210409. 5 migrations executed.

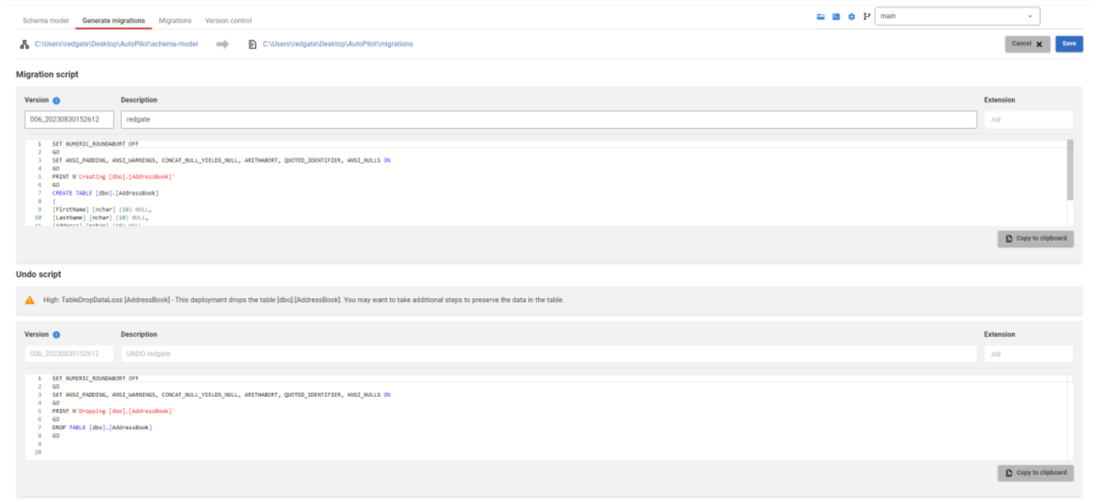
Object Name	Change type	ObjectType	Schema
AddressBook	Create	Table	dbo

C:\Users\redgate\Desktop\AutoPilot\migrations

Current AddressBook | New AddressBook

```
CREATE TABLE [dbo].[AddressBook]
(
[FirstName] [nchar] (10) NULL,
[LastName] [nchar] (10) NULL,
[Address] [nchar] (10) NULL
)
GO
```

3. We have the choice now to decide what changes we should turn into a script, choose the new object we made, and click generate scripts!



Migration script

Version: 006_20230830152612
Description: redgate
Extension: .sql

```
SET NUMERIC_ROUNDABORT OFF
GO
SET ANSI_PADDING, ANSI_WARNINGS, CONCAT_NULL_YIELDS_NULL, ARITHABORT, QUOTED_IDENTIFIER, ANSI_NULLS ON
GO
PRINT N'Creating [dbo].[AddressBook]'
GO
CREATE TABLE [dbo].[AddressBook]
(
[FirstName] [nchar] (10) NULL,
[LastName] [nchar] (10) NULL,
```

Undo script

High: TableDropDataLoss [AddressBook] - This deployment drops the table [dbo].[AddressBook]. You may want to take additional steps to preserve the data in the table.

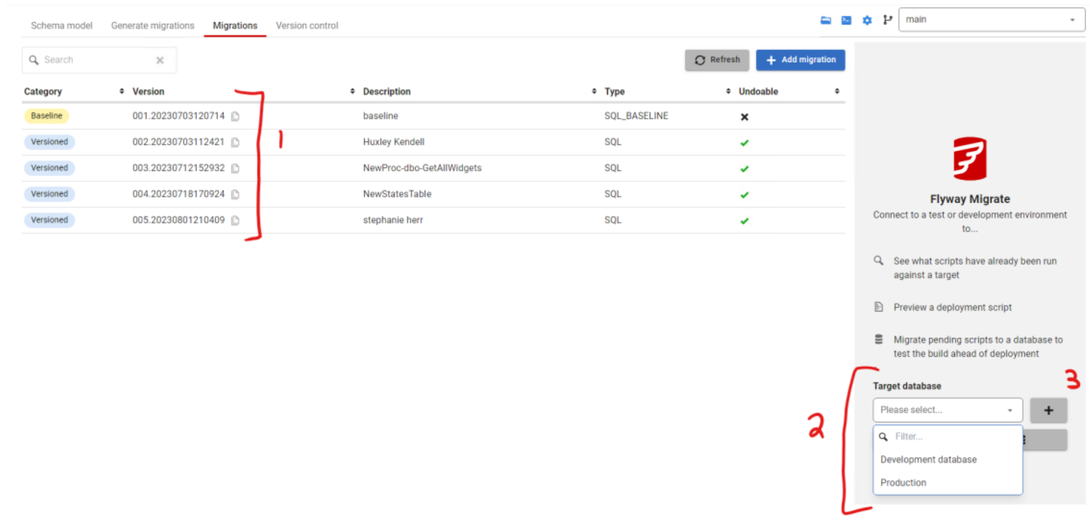
Version: 006_20230830152612
Description: UNDO-redgate
Extension: .sql

```
SET NUMERIC_ROUNDABORT OFF
GO
SET ANSI_PADDING, ANSI_WARNINGS, CONCAT_NULL_YIELDS_NULL, ARITHABORT, QUOTED_IDENTIFIER, ANSI_NULLS ON
GO
PRINT N'Dropping [dbo].[AddressBook]'
GO
DROP TABLE [dbo].[AddressBook]
GO
```

4. As you should of seen, Flyway creates two scripts for us! A forward migration script, and a corresponding rollback script. This not only streamlines development by saving any time spent on writing scripts, but also gives instant access to rollback any changes we did not anticipate! Save the script, and we are ready to begin deploying!

6. Add a New Target Database

1. An important part of Flyway is its ability to deploy, head over to the migrations tab to begin to see what we can do!



Category	Version	Description	Type	Undoable
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗
Versioned	002.20230703112421	Huxley Kendell	SQL	✓
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓
Versioned	004.20230718170924	NewStatesTable	SQL	✓
Versioned	005.20230801210409	stephanie herr	SQL	✓

Flyway Migrate
Connect to a test or development environment to...

See what scripts have already been run against a target

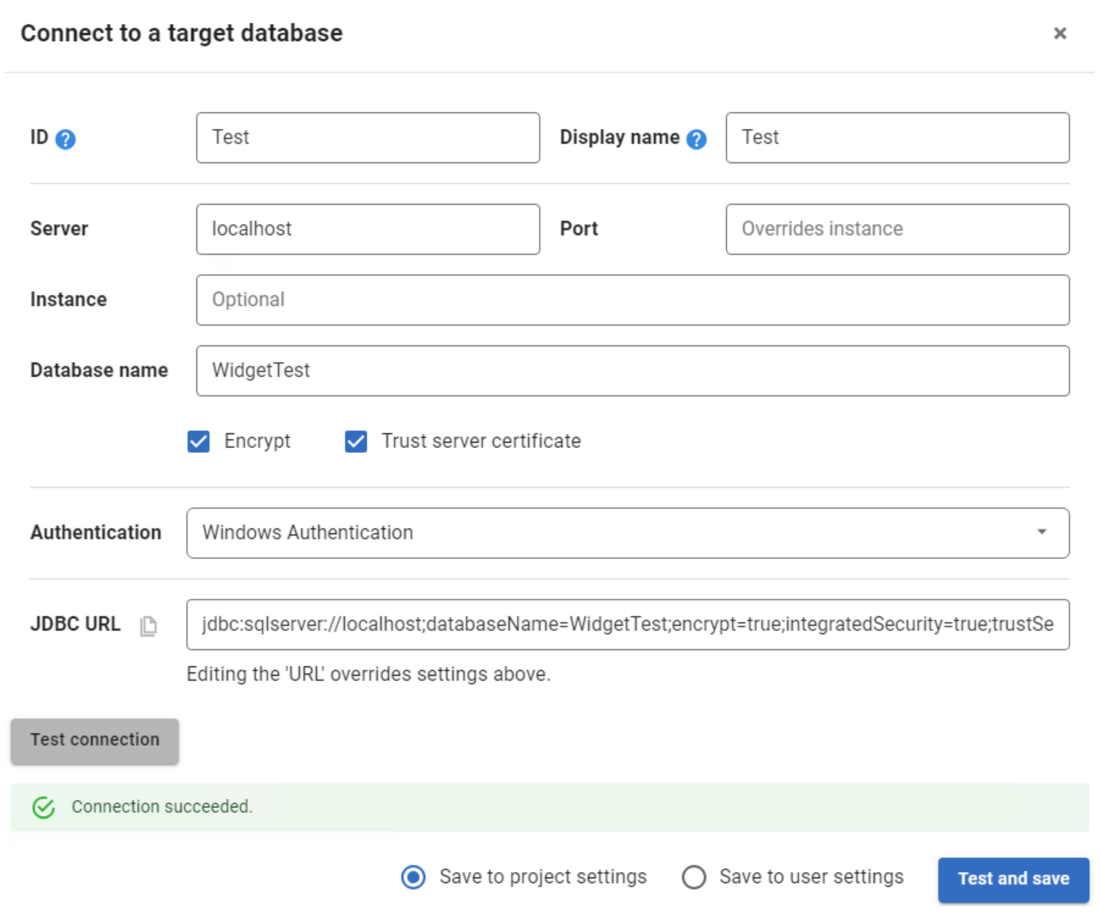
Preview a deployment script

Migrate pending scripts to a database to test the build ahead of deployment

Target database
Please select...
Filter...
Development database
Production

- On this page we get an overview of all the scripts that make up our Flyway Project (1).
- We can also chose to Target an environment, this will give a much clearer picture of the state of our deployments, as we track the version of each Database (2).

2. As you may notice, we only have our Development & Production environments currently, lets try and add the Testing environment we have by clicking the plus button (3)!
3. We need to provide a few details before we can add our new environment, luckily for us Flyway makes it easy. Reference the image below to see the information we need to supply:


Connect to a target database

ID: Test
Display name: Test
Server: localhost
Port: Overrides instance
Instance: Optional
Database name: WidgetTest
[✓] Encrypt [✓] Trust server certificate
Authentication: Windows Authentication
JDBC URL: jdbc:sqlserver://localhost;databaseName=WidgetTest;encrypt=true;integratedSecurity=true;trustSe
Editing the 'URL' overrides settings above.

Connection succeeded.

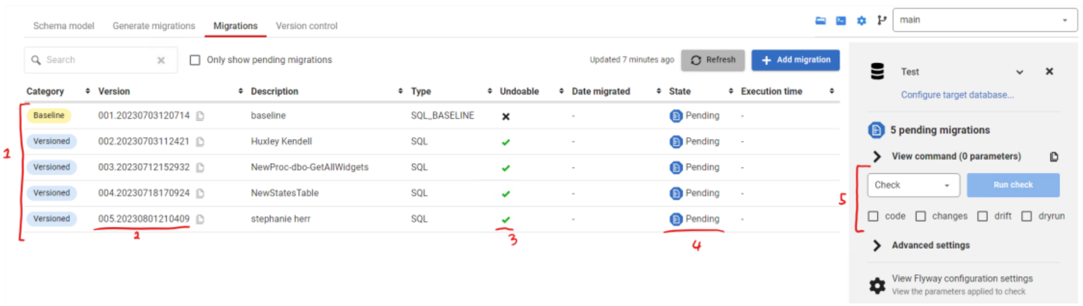
[● Save to project settings] ○ Save to user settings

- **ID:** This is a parameter we can assign the DB, to allow us to reference it by name instead of the longer JDBC when it comes to using the CLI.
- **Display Name:** This is the name Flyway will display for this environment.
- Server, Port & Instance: These reference the location of our instance!
- **Database Name:** The name of the DB we intend to add!
- If you need to **Encrypt** or tell Flyway to **Trust Server Certificate** we can also enable this here!
- **Authentication:** Set the kind of Authentication we should use, this will be dictated by the configuration of your Server.
- **JDBC:** Flyway makes use of its Java API for communicating with Databases. The JDBC will be created for you when you add the information above, you can use this to copy the URL of your Database or to manually add it and watch Flyway change the parameters above!

- Save To Project Settings: Saving to the Project settings will share this with anyone else on the repository, so it is smart for any shared configuration to be saved here, such as downstream environments.

- Save To User Settings: Saving to User settings is useful as this will not be shared with anyone else, so settings like a dedicated Dev environment which are personal, should be saved to user settings to avoid overwriting any changes!

4. Fill out the details for our testing environment as seen in the image above, or for your own requirements, and click Test & Save! Once done, it will display more information into the state of our Database as seen below!



Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	Pending	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	-	Pending	-
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	-	Pending	-
Versioned	004.20230718170924	NewStatesTable	SQL	✓	-	Pending	-
Versioned	005.20230801210409	stephanie herr	SQL	✓	-	Pending	-

- **Category:** This tells us what type of script it is. This can be:

Baseline: This sets the starting point for our scripts, any pre-existing parts of the database will be comprised here. These scripts are used when deploying to a fresh database, and allow us to separate subsequent changes made with Flyway from objects that are already in production!

Versioned: These are scripts that encompass the changes needed to move to a new version of the database, versioned scripts allow us to manage the state of different DBs much easier, and with Flyway we don't have to worry about deploying scripts manually. This can make rolling forward, and backwards much more streamlined!

Undo: These are scripts that allow us to rollback a version script, each Version Script you make can and should have a rollback script, meaning whenever we deploy a change, we also a script to undo the change!

Repeatable: These are scripts that we want to run multiple times, anytime Flyway detects that the script has been altered, it will deploy the script on the next migration!

- Version: This indicates the specific version of the database, so we can see any information around that specific version of the database and the script used to move to it!

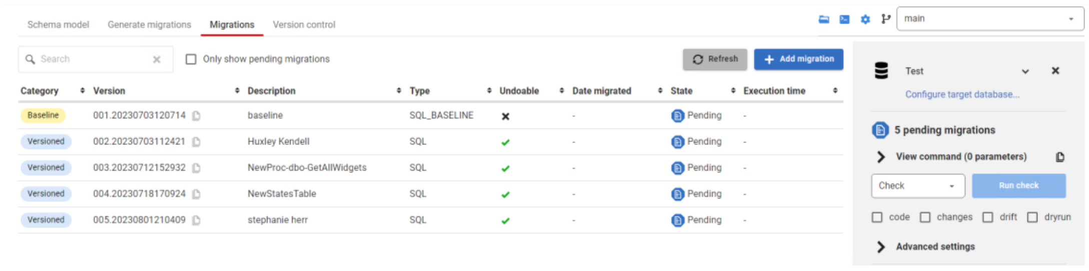
- Undoable: Indicates if this version is undoable, allowing us to rollback to the prior

version instantly. If it is not undoable, a rollback script needs to be supplied!
- State: State of this version, can either be successful which indicates the script has already been deployed or Pending, indicating the script is yet to be deployed to this environment.
Also marked on the image (5) is the **command window**, this is where we can perform Flyway commands against this particular database, alongside any additional parameters. This will be explored throughout this exercise book!

7. Create a Report

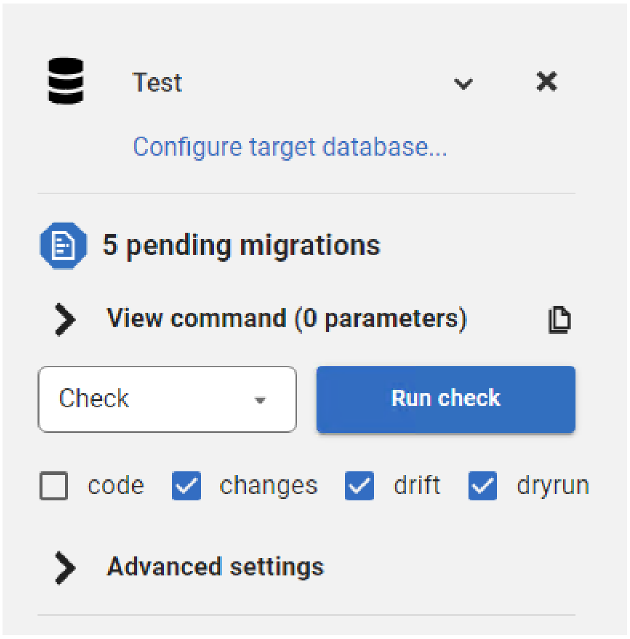
1. A useful feature of Flyway is its ability to generate reporting, this can include:
   - Code: This analyse your code, and checks for any rules that have been broken that you set!
   - Changes: This shows how objects have evolved, and the code differences used to make these changes.
   - Drift: This checks for any drift, drift indicates any changes that have been made to the database outside of the scope of Flyway, so either direct changes to the environment or scripts that have been deployed elsewhere.
   - Dry run: This produces a dry run script, which showcases what is going to happen to this database on the next migration, pre-deployment! Allowing you to analyse changes before they happen!
   All of these reports can be collated into one HTML report, and can be created retroactively allowing for much higher visibility before we make changes to downstream environments!
2. We have successfully added our Test database as a target, and one common best practice is to generate a set of reports before we decide to deploy any scripts, so let’s try and do that. The first step is to make sure we are targeting our Test DB in the Migrations tab and select the Check Command inside the command window as

seen below.



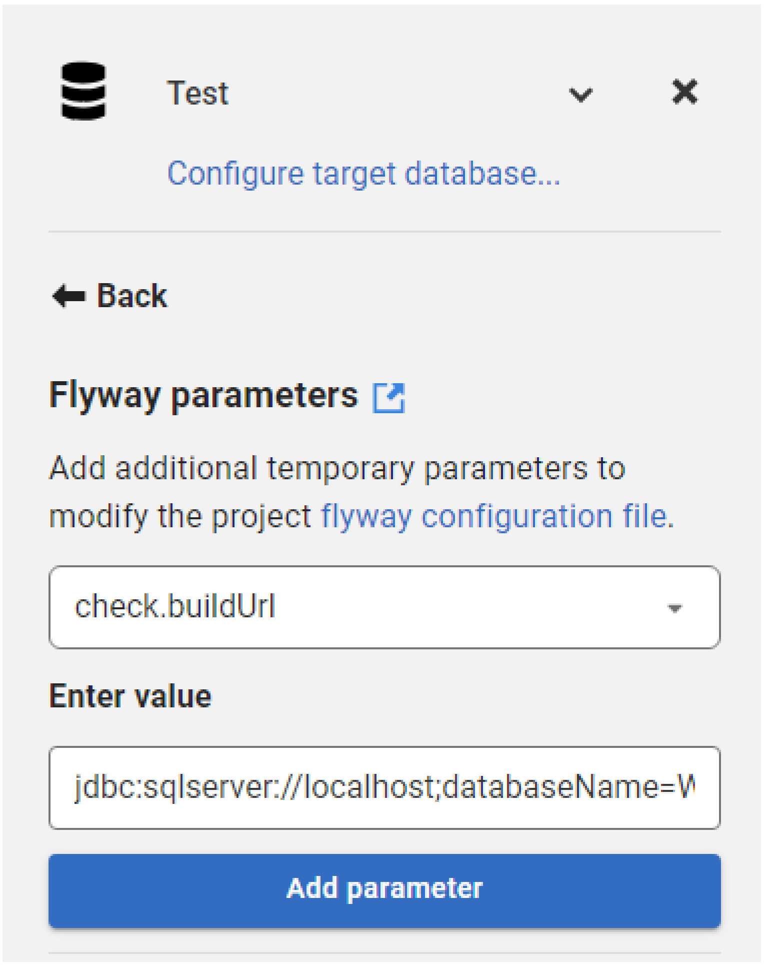
Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	Pending	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	-	Pending	-
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	-	Pending	-
Versioned	004.20230718170924	NewStatesTable	SQL	✓	-	Pending	-
Versioned	005.20230801210409	stephanie herr	SQL	✓	-	Pending	-

3. We need to know decide what kind of report we wish to generate, for this example we can try and generate a report that includes code changes, a summary of any drift on our database and a copy of the dry run script! Now let's select the type of reports we want!



☐ code ☑ changes ☑ drift ☑ dryrun

4. As we are doing this retroactively, before deploying. Flyway will need to be provided with a database to act as a sandbox and generate the report, to do this we need to supply it with the 'Check' database we have inside of our Server. This is a chance to showcase the additional parameters functionality of flyway, to begin to customise our commands. To do this particular parameter, click into 'Advanced Settings' and find the 'check.buildUrl' parameter.


Test

Configure target database...

Back

Flyway parameters

Add additional temporary parameters to modify the project flyway configuration file.

check.buildUrl

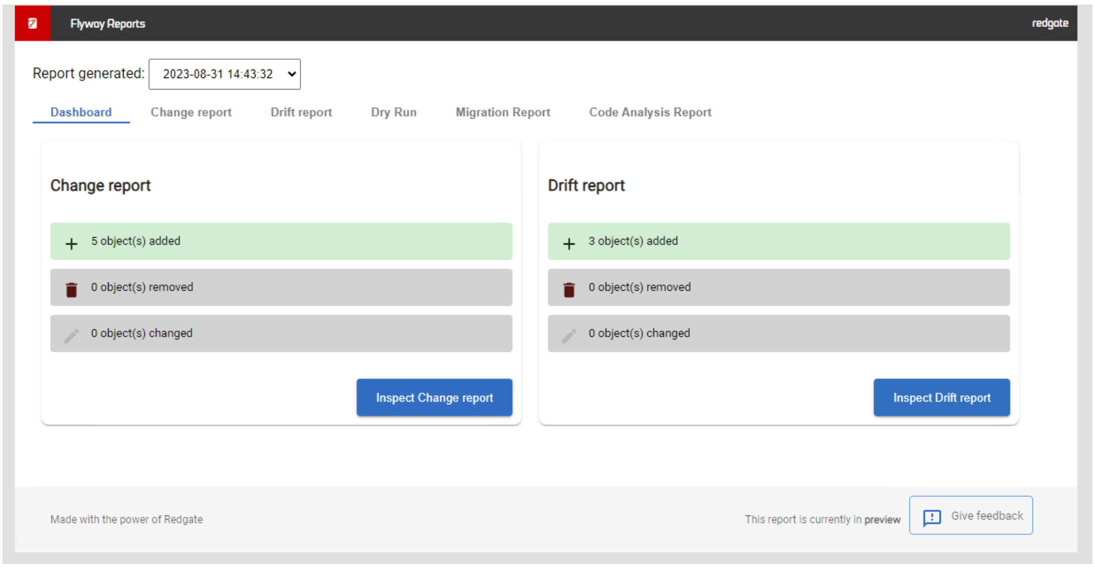
Enter value

jdbc:sqlserver://localhost;databaseName=W

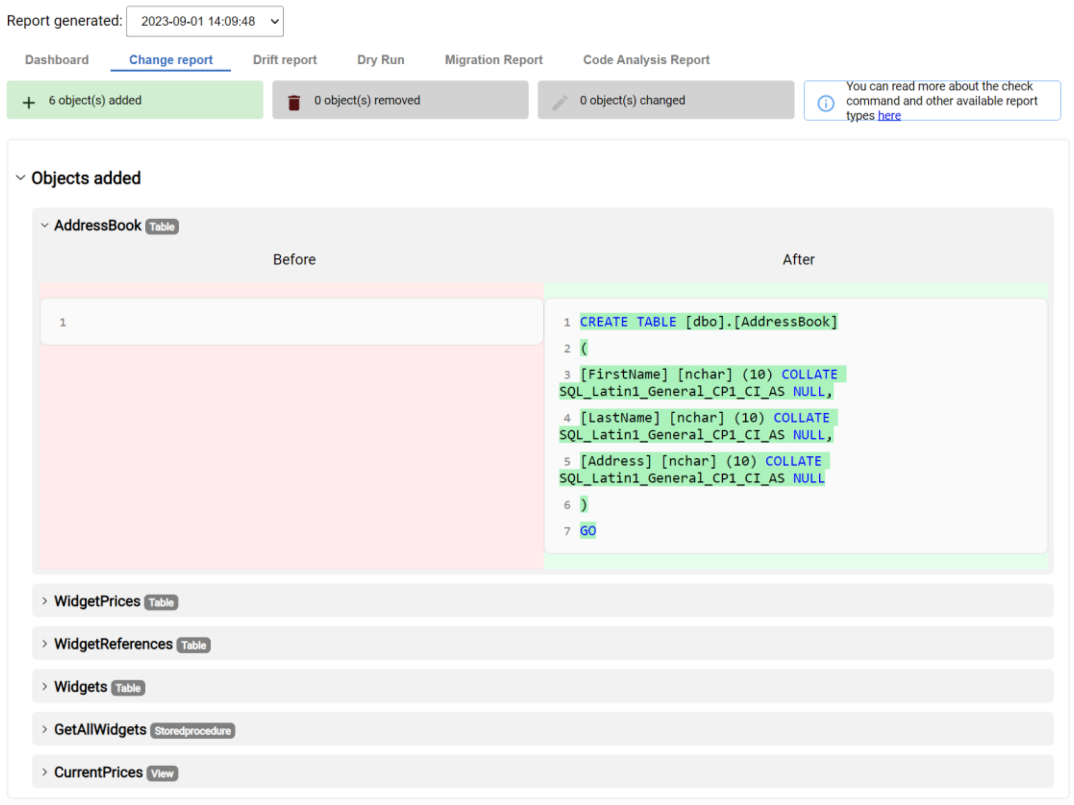
Add parameter

If you need support finding the 'Check.BuildUrl' reference back to Challenge 5 and add the Check Database as a new target to be provided with a JDBC!

5. The final step is to run the command and review the report it generates for us!



As we should be able to see, it provides us with all the reports we requested giving us a much higher level of visibility into each of our databases.

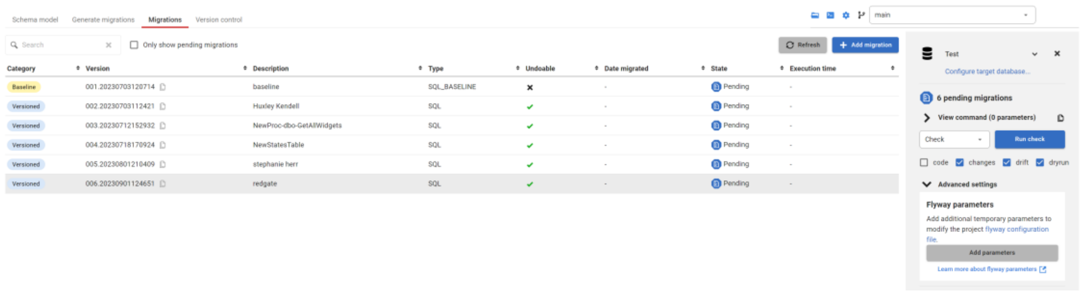


```
CREATE TABLE [dbo].[AddressBook]
(
[FirstName] [nchar] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
[LastName] [nchar] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL,
[Address] [nchar] (10) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
)
GO
```

Remember, we can fully automate this process!

8. View DryRun Script

1. Now we have generated a report on the future deployment, we can have a much higher level of confidence going into our deployment. Navigate your way back to the migrations tab, and target our 'Test' Database so we can begin the process of now deploying those changes we have seen.



Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	Pending	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	-	Pending	-
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	-	Pending	-
Versioned	004.20230718170924	NewStatesTable	SQL	✓	-	Pending	-
Versioned	005.20230801210409	stephanie herr	SQL	✓	-	Pending	-
Versioned	006.20230901124651	redgate	SQL	✓	-	Pending	-

Test

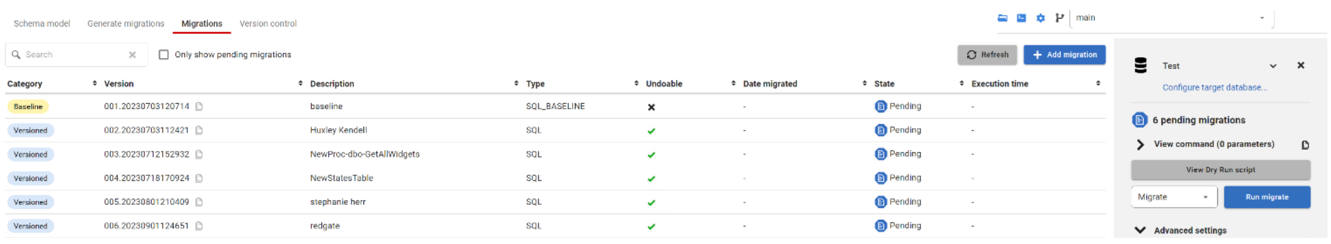
6 pending migrations

Check: ☐ code ☑ changes ☑ drift ☑ dryrun

Flyway parameters

Add additional temporary parameters to modify the project flyway configuration file.

2. The final level of clarity we can gain from Flyway before making our changes, is by viewing a DryRun Script of the actual deployment, just like we generated in our Report the challenge before.
3. The command we will use to deploy our changes is a basic 'Migrate' command, without any additional parameters a migrate command will check for any pending scripts, and deploy them in order.



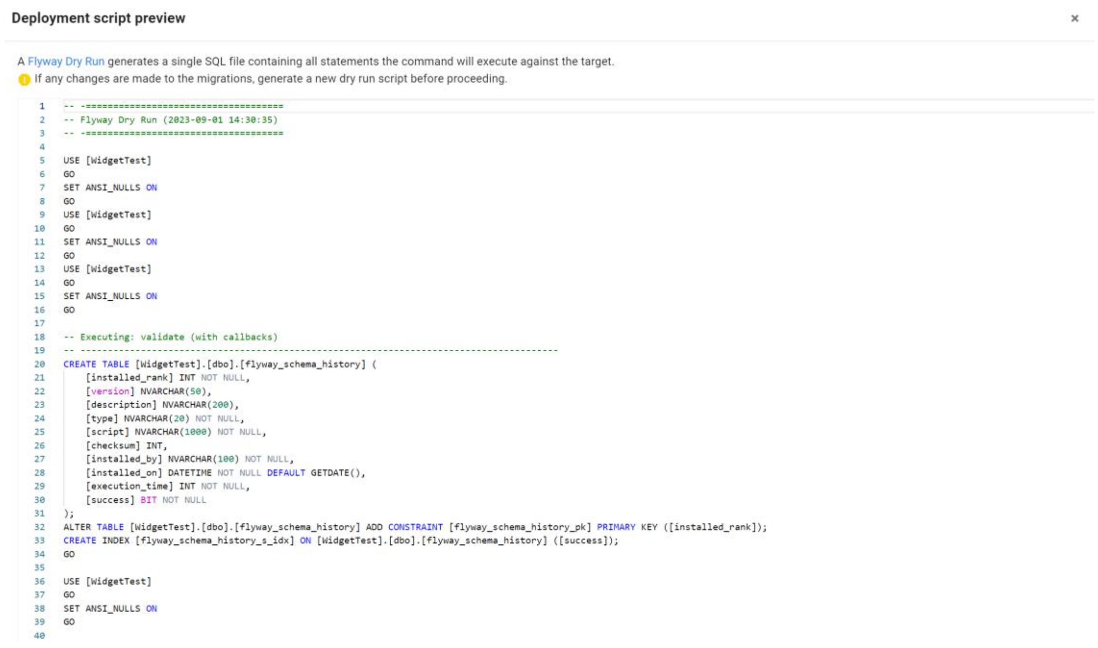
Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	Pending	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	-	Pending	-
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	-	Pending	-
Versioned	004.20230718170924	NewStatesTable	SQL	✓	-	Pending	-
Versioned	005.20230801210409	stephanie herr	SQL	✓	-	Pending	-
Versioned	006.20230901124651	redgate	SQL	✓	-	Pending	-

Test

6 pending migrations

Migrate

4. Before we deploy, try and click “View Dry Run Script” to view the deployment before we go ahead and do so!



Deployment script preview

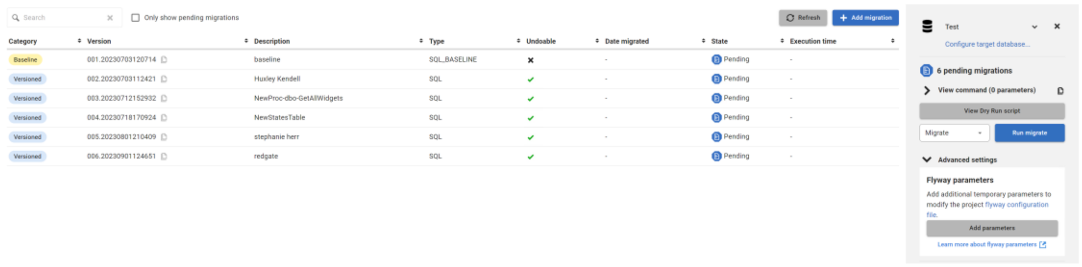
A Flyway Dry Run generates a single SQL file containing all statements the command will execute against the target.
If any changes are made to the migrations, generate a new dry run script before proceeding.

```
-- -====================================
-- Flyway Dry Run (2023-09-01 14:30:35)
-- -====================================

USE [WidgetTest]
GO
SET ANSI_NULLS ON
GO
USE [WidgetTest]
GO
SET ANSI_NULLS ON
GO
USE [WidgetTest]
GO
SET ANSI_NULLS ON
GO

-- Executing: validate (with callbacks)
-- -------------------------------------------------------------------------------------
CREATE TABLE [WidgetTest].[dbo].[flyway_schema_history] (
    [installed_rank] INT NOT NULL,
    [version] NVARCHAR(50),
    [description] NVARCHAR(200),
    [type] NVARCHAR(20) NOT NULL,
    [script] NVARCHAR(1000) NOT NULL,
    [checksum] INT,
    [installed_by] NVARCHAR(100) NOT NULL,
    [installed_on] DATETIME NOT NULL DEFAULT GETDATE(),
    [execution_time] INT NOT NULL,
    [success] BIT NOT NULL
);
ALTER TABLE [WidgetTest].[dbo].[flyway_schema_history] ADD CONSTRAINT [flyway_schema_history_pk] PRIMARY KEY ([installed_rank]);
CREATE INDEX [flyway_schema_history_s_idx] ON [WidgetTest].[dbo].[flyway_schema_history] ([success]);
GO

USE [WidgetTest]
GO
SET ANSI_NULLS ON
GO

```

5. If everything in the Dry Run Script looks good, we can either Run Migrate straight from this page, or close and run the Migrate tab. For the sake of this exercise, anaylse the dry run script and click close so we can follow the next challenge!

9. Deploy to New Target

1. We now have a very high level of confidence in our deployment, so the final step is to run the migrate command and bring our Test database to the latest version!
2. If you are not there already, head to the migrate tab, double check we are targeting our Test environment then click ‘Run Migrate’.

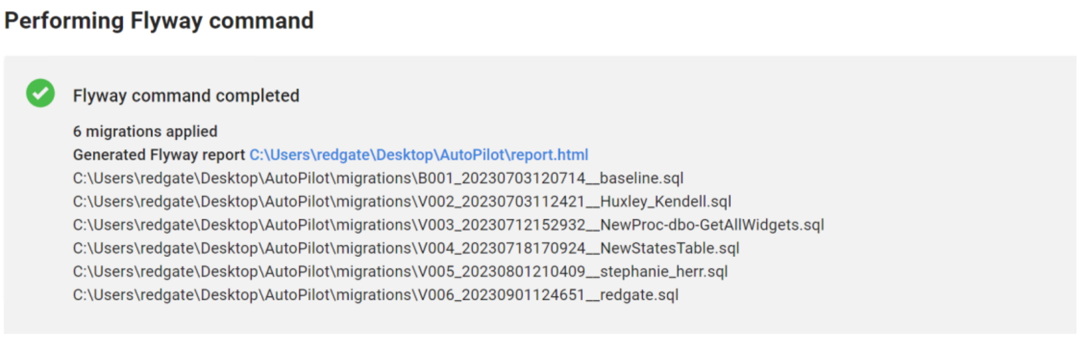


Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	Pending	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	-	Pending	-
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	-	Pending	-
Versioned	004.20230718170924	NewStatesTable	SQL	✓	-	Pending	-
Versioned	005.20230801210409	stephanie herr	SQL	✓	-	Pending	-
Versioned	006.20230901124651	redgate	SQL	✓	-	Pending	-

Test
Configure target database...
6 pending migrations
View command (0 parameters)

Flyway parameters
Add additional temporary parameters to modify the project flyway configuration file.

3. Once the migration has complete, we will see a display letting us know all the successful deployments, once happy click close!

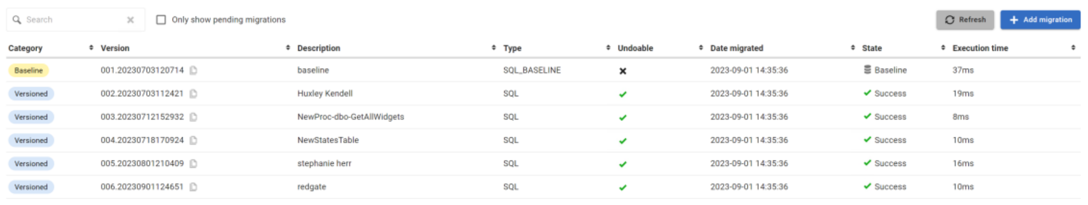


Performing Flyway command

Flyway command completed

```
6 migrations applied
Generated Flyway report C:\Users\redgate\Desktop\AutoPilot\report.html
C:\Users\redgate\Desktop\AutoPilot\migrations\B001_20230703120714__baseline.sql
C:\Users\redgate\Desktop\AutoPilot\migrations\V002_20230703112421__Huxley_Kendell.sql
C:\Users\redgate\Desktop\AutoPilot\migrations\V003_20230712152932__NewProc-dbo-GetAllWidgets.sql
C:\Users\redgate\Desktop\AutoPilot\migrations\V004_20230718170924__NewStatesTable.sql
C:\Users\redgate\Desktop\AutoPilot\migrations\V005_20230801210409__stephanie_herr.sql
C:\Users\redgate\Desktop\AutoPilot\migrations\V006_20230901124651__redgate.sql
```

4. Once we click close, we will see that Flyway refreshes its view over the database, and has updated the table to let us know the state of deployments, you should see that all of our scripts have been successfully deployed!

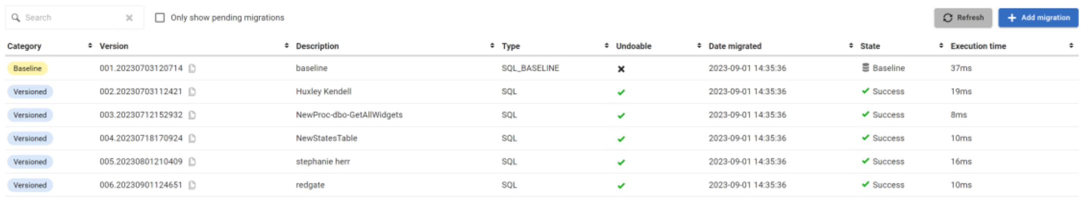


Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	2023-09-01 14:35:36	Baseline	37ms
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	2023-09-01 14:35:36	✓ Success	19ms
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	2023-09-01 14:35:36	✓ Success	8ms
Versioned	004.20230718170924	NewStatesTable	SQL	✓	2023-09-01 14:35:36	✓ Success	10ms
Versioned	005.20230801210409	stephanie herr	SQL	✓	2023-09-01 14:35:36	✓ Success	16ms
Versioned	006.20230901124651	redgate	SQL	✓	2023-09-01 14:35:36	✓ Success	10ms

10. Rollback Changes

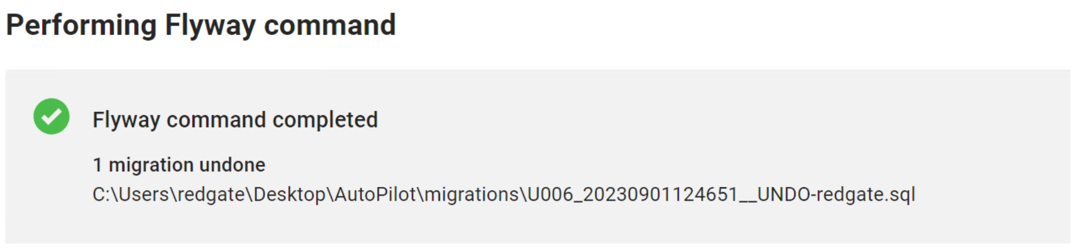
1. So, we have seen now how we can roll forward and update the state of databases with ease, but what if we deployed changes that had effects we now want to undo. Maybe certain changes did not align with applicational code, regardless it is incredibly important to have the ability to instantly rollback a change.
2. Remaining inside the migrations tab of Flyway, we can change our command to the "Undo" command. The undo command, like our migrations command, has a desired functionality without any parameters, and that is to simply rollback the most

recent script that has been deployed that it can.



Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	2023-09-01 14:35:36	Baseline	37ms
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	2023-09-01 14:35:36	✓ Success	19ms
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	2023-09-01 14:35:36	✓ Success	8ms
Versioned	004.20230718170924	NewStatesTable	SQL	✓	2023-09-01 14:35:36	✓ Success	10ms
Versioned	005.20230801210409	stephanie herr	SQL	✓	2023-09-01 14:35:36	✓ Success	16ms
Versioned	006.20230901124651	redgate	SQL	✓	2023-09-01 14:35:36	✓ Success	10ms

3. In our scenario (seen above) version 6 has been deployed, and can be undone, so that will be the version a basic undo command will target.
4. Try and see if you can undo the changes we just deployed, by changing our command to undo and clicking "Run Undo".

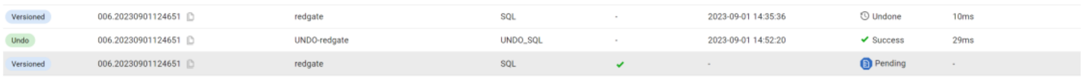


Performing Flyway command

Flyway command completed

1 migration undone
C:\Users\redgate\Desktop\AutoPilot\migrations\U006_20230901124651__UNDO-redgate.sql

5. Hopefully you will see that Flyway was able to rollback and execute version 6s corresponding Undo Script, head back to the migrations tab to visualise our changes!



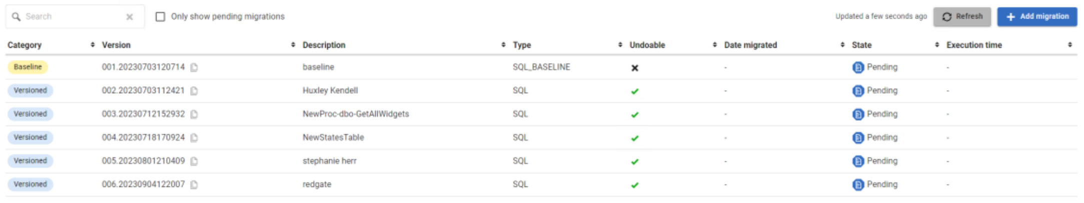
Versioned	006.20230901124651	redgate	SQL	-	2023-09-01 14:35:36	Undone	10ms
Undo	006.20230901124651	UNDO-redgate	UNDO_SQL	-	2023-09-01 14:52:20	✓ Success	29ms
Versioned	006.20230901124651	redgate	SQL	✓	-	Pending	-

6. This shows that version 6 was deployed, but the undo script has been deployed and now version 6 is back to pending!

11. Perform an Info Command via CLI

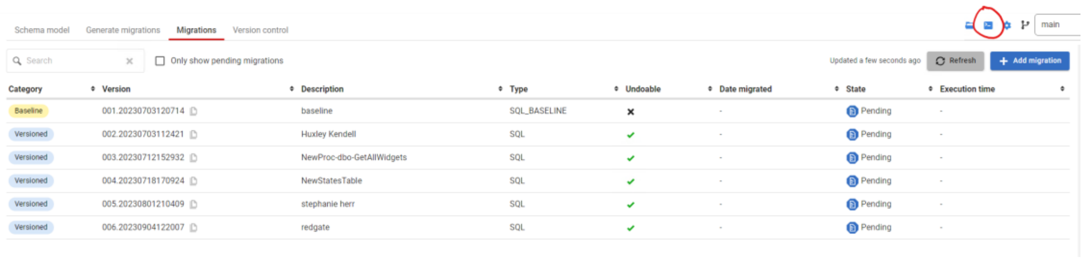
1. One of Flyways massive benefits is its flexibility, any and all actions you perform inside Flyway Desktop can be performed inside the Command-line interface (CLI)!
2. Navigate yourself to the migrations tab, and target any database! As you can see, Flyway generates a graphic table showing all of our versions, and what the state of the database is. As we have covered before, and we can see in the screenshot

below!



Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	Pending	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	-	Pending	-
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	-	Pending	-
Versioned	004.20230718170924	NewStatesTable	SQL	✓	-	Pending	-
Versioned	005.20230801210409	stephanie herr	SQL	✓	-	Pending	-
Versioned	006.20230904122007	redgate	SQL	✓	-	Pending	-

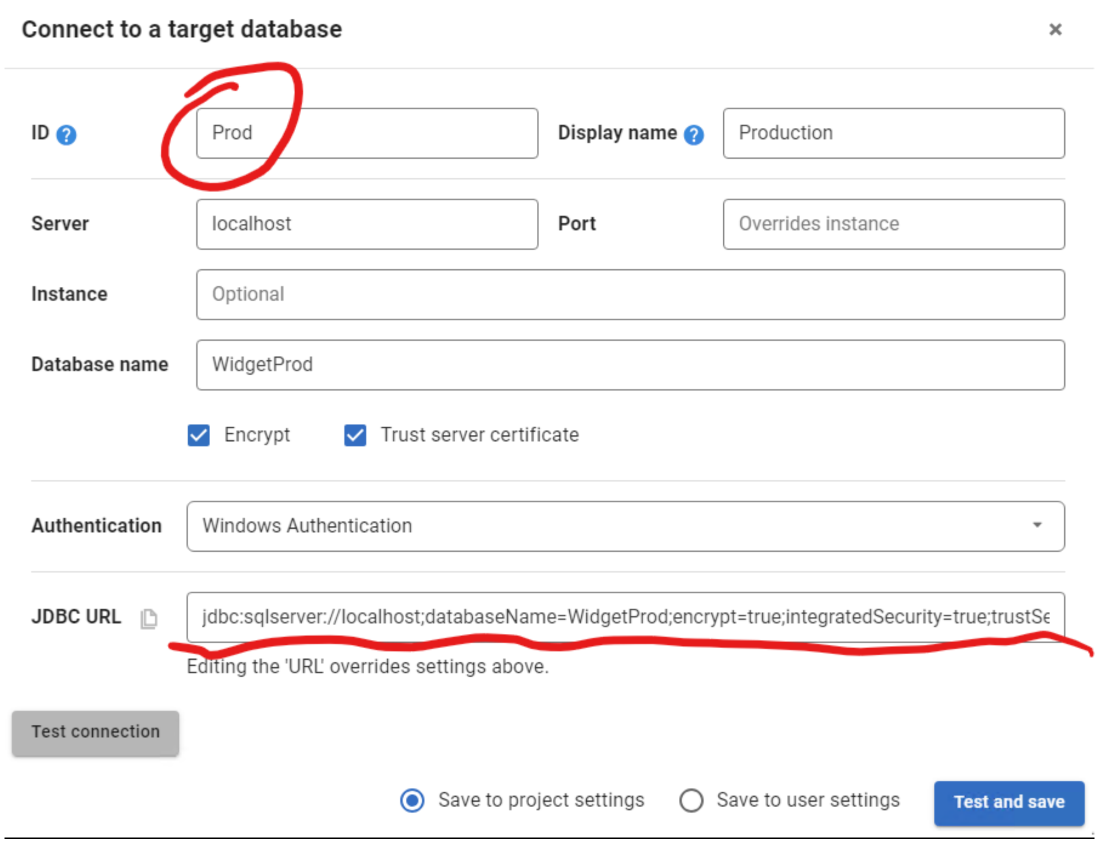
3. Now let's try and reproduce this functionality, but over the CLI. You should see (see below if not) a button in the top right of Flyway desktop, to take you straight to a console inside the Flyway project, click this to begin!



Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	Pending	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	-	Pending	-
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	-	Pending	-
Versioned	004.20230718170924	NewStatesTable	SQL	✓	-	Pending	-
Versioned	005.20230801210409	stephanie herr	SQL	✓	-	Pending	-
Versioned	006.20230904122007	redgate	SQL	✓	-	Pending	-

4. The final step is to provide the console with a Flyway Info command, which is what Flyway desktop uses to produce the table! We can do that with the following:

   All Flyway CLI commands begin with the syntax of "***Flyway*".**
   The syntax we need after this is "***Info*".** This grabs all the information over the specified DB, just as Flyway does internally to produce the table!
   We then need to tell Flyway which DB we wish to perform the command against, we can do this two ways. We either use the parameter of *"**-Url={JDBC URL of Target DB}**"* or "***Environment={ID}***"
   Both of these parameters can be found inside the "configure target database" (seen below)



Connect to a target database ×

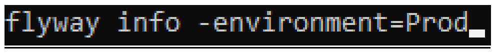
ID	Prod	Display name	Production
Server	localhost	Port	Overrides instance
Instance	Optional		
Database name	WidgetProd		

☑ Encrypt ☑ Trust server certificate

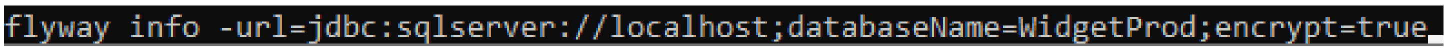
Authentication: Windows Authentication

JDBC URL: jdbc:sqlserver://localhost;databaseName=WidgetProd;encrypt=true;integratedSecurity=true;trustSe

Editing the 'URL' overrides settings above.

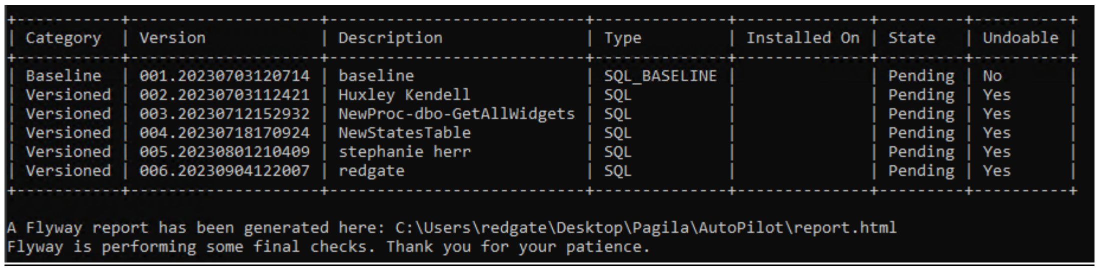


```
flyway info -environment=Prod
```



```
flyway info -url=jdbc:sqlserver://localhost;databaseName=WidgetProd;encrypt=true
```

5. You should see the following, successful output:

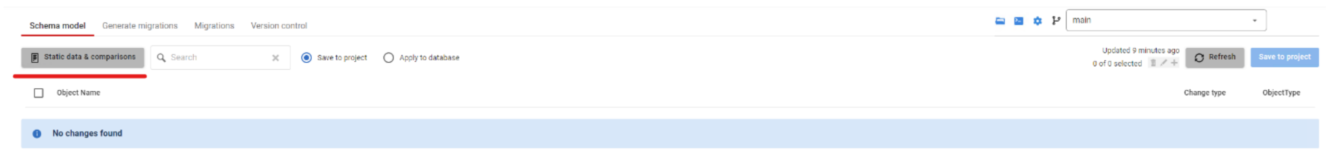


```
+-----------+--------------------+----------------------------+-------------+--------------+---------+----------+
| Category  | Version            | Description                | Type        | Installed On | State   | Undoable |
+-----------+--------------------+----------------------------+-------------+--------------+---------+----------+
| Baseline  | 001.20230703120714 | baseline                   | SQL_BASELINE |             | Pending | No       |
| Versioned | 002.20230703112421 | Huxley Kendell             | SQL         |              | Pending | Yes      |
| Versioned | 003.20230712152932 | NewProc-dbo-GetAllWidgets  | SQL         |              | Pending | Yes      |
| Versioned | 004.20230718170924 | NewStatesTable             | SQL         |              | Pending | Yes      |
| Versioned | 005.20230801210409 | stephanie herr             | SQL         |              | Pending | Yes      |
| Versioned | 006.20230904122007 | redgate                    | SQL         |              | Pending | Yes      |
+-----------+--------------------+----------------------------+-------------+--------------+---------+----------+

A Flyway report has been generated here: C:\Users\redgate\Desktop\Pagila\AutoPilot\report.html
Flyway is performing some final checks. Thank you for your patience.
```

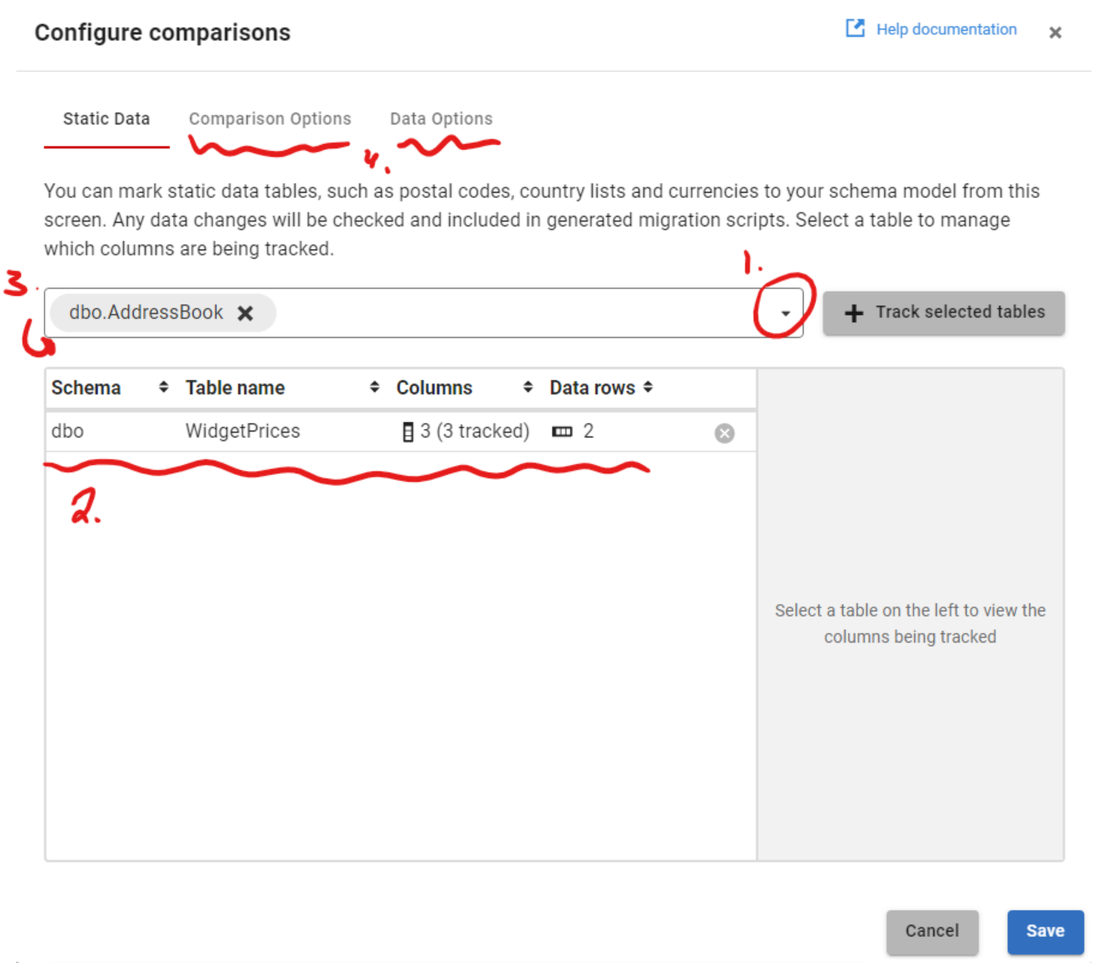
You'll notice it also produces a report with its finding, feel free to check that out too!

12. Track Static Data inside of Dev

1. Flyway aims to be your “end-to-end” DevOps tool, and apart of that value is making sure we can track not just schema changes but data changes as well. Flyway comes built in with the ability to track Static Data! Head to the Schema Model tab, to begin this challenge!



2. For this challenge, click into Static Data & Comparisons, and try and add a table we wish to track static data for!


Configure comparisons

Help documentation

Static Data | Comparison Options | Data Options

You can mark static data tables, such as postal codes, country lists and currencies to your schema model from this screen. Any data changes will be checked and included in generated migration scripts. Select a table to manage which columns are being tracked.

dbo.AddressBook

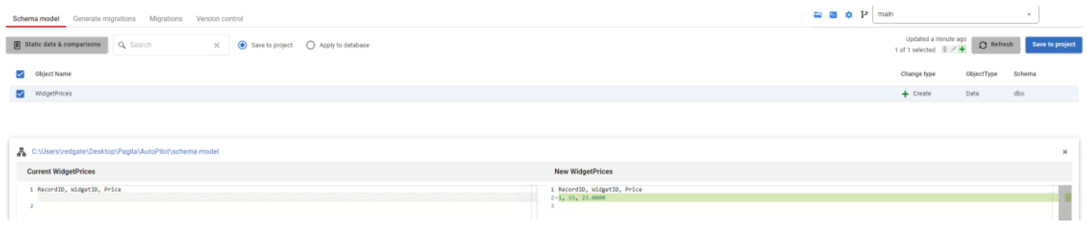
Schema	Table name	Columns	Data rows
dbo	WidgetPrices	3 (3 tracked)	2

Select a table on the left to view the columns being tracked

3. We can choose our tables via the dropdown list (1)
   You can then customise which columns are being tracked by clicking on (2)
   You can manually add objects to track the data for by typing into (3)
   You can also customise how we track schema and data, as well as deploying, by clicking into either points labelled at (4).

4. Click Save, and watch Flyway scan for new data. Feel free to add any data into our tables, and to finish this challenge track that data changes by Saving them to the

project as we learnt before!



Updated a minute ago
1 of 1 selected

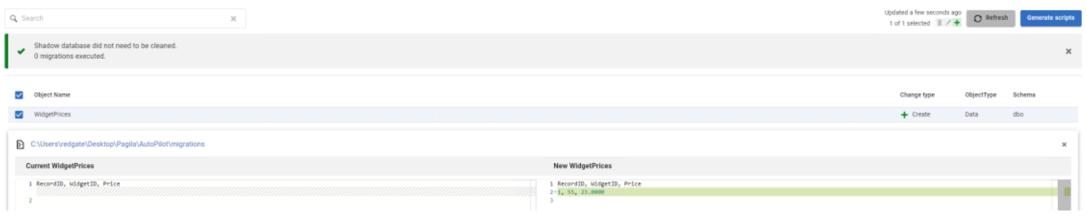
Object Name	Change type	ObjectType	Schema
WidgetPrices	Create	Data	dbo

C:\Users\redgate\Desktop\Pagila\AutoPilot\schema-model

Current WidgetPrices	New WidgetPrices
RecordID, WidgetID, Price	RecordID, WidgetID, Price
	1, 55, 23.0000

13. Create DML Migration Script

1. We learnt how to track our static data, and by saving this data in our project we can share between other developers and track any changes with ease.
2. The next step to being able to effectively handle any and all static data, is to be able to turn our changes, into a DML script (Flyway is able to generate scripts that include DML changes, DDL changes, or a combination of both).
   So in the same process as before, head to the Generate Migrations tab, and create a migration script that includes DML inside of it!



Updated a few seconds ago
1 of 1 selected

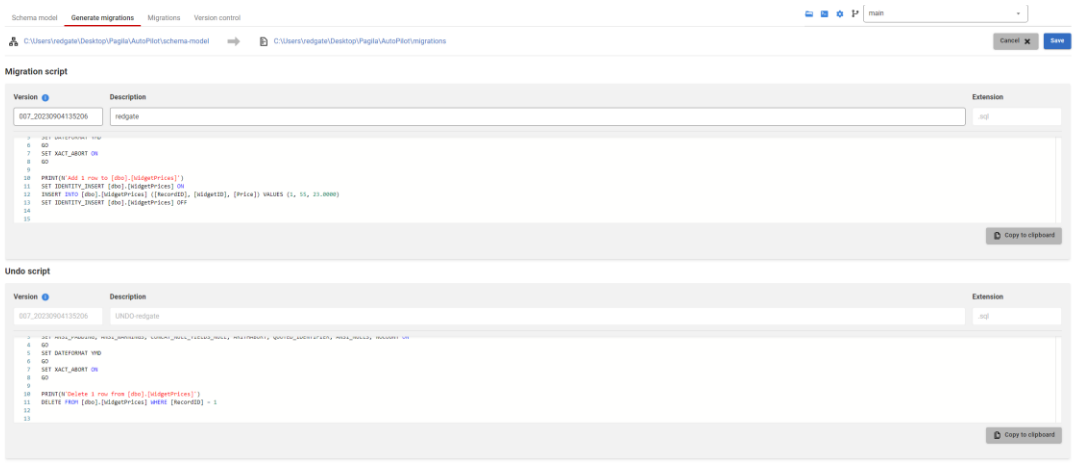
Shadow database did not need to be cleaned.
0 migrations executed.

Object Name	Change type	ObjectType	Schema
WidgetPrices	Create	Data	dbo

C:\Users\redgate\Desktop\Pagila\AutoPilot\migrations

Current WidgetPrices	New WidgetPrices
RecordID, WidgetID, Price	RecordID, WidgetID, Price
	1, 55, 23.0000

3. Check over the scripts produced, but we should now be able to successful insert our data changes and roll them back!


Schema model
Generate migrations
Migrations
Version control
main
C:\Users\redgate\Desktop\Pagila\AutoPilot\schema-model
C:\Users\redgate\Desktop\Pagila\AutoPilot\migrations
Cancel
Save
Copy to clipboard
Copy to clipboard

Migration script

Version: 007_20230904135206
Description: redgate
Extension: .sql

```sql
GO
SET XACT_ABORT ON
GO

PRINT(N'Add 1 row to [dbo].[WidgetPrices]')
SET IDENTITY_INSERT [dbo].[WidgetPrices] ON
INSERT INTO [dbo].[WidgetPrices] ([RecordID], [WidgetID], [Price]) VALUES (1, 55, 23.0000)
SET IDENTITY_INSERT [dbo].[WidgetPrices] OFF
```

Undo script

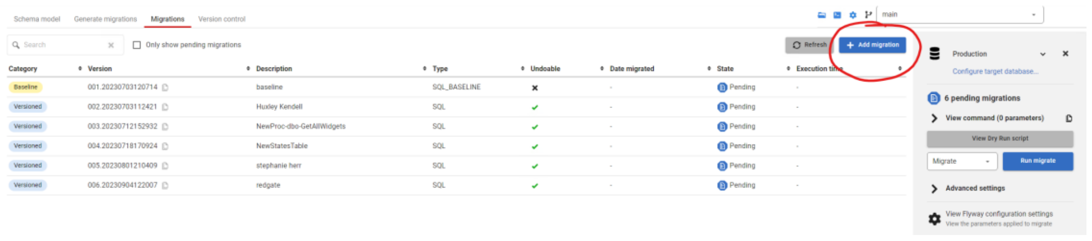
Version: 007_20230904135206
Description: UNDO-redgate
Extension: .sql

```sql
GO
SET DATEFORMAT YMD
GO
SET XACT_ABORT ON
GO

PRINT(N'Delete 1 row from [dbo].[WidgetPrices]')
DELETE FROM [dbo].[WidgetPrices] WHERE [RecordID] = 1
```

14. Create & Deploy Repeatable Script

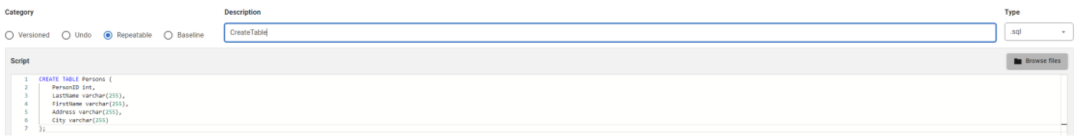
1. Now is a good time to explore a new type of Script we can use, having seen both our Versioned Scripts and our Undo Scripts.
   These scripts are called repeatable scripts, and we have touched upon them briefly so far. The fundamentals of our R scripts is that they run on every deployment, so long as it has been altered since the last time it deployed itself.

2. So, lets head over to the migrations tab and see about adding one for ourselves! Once inside the migrations tab, click "+ Add Migration"



Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	Pending	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	-	Pending	-
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	-	Pending	-
Versioned	004.20230718170924	NewStatesTable	SQL	✓	-	Pending	-
Versioned	005.20230801210409	stephanie herr	SQL	✓	-	Pending	-
Versioned	006.20230904122007	redgate	SQL	✓	-	Pending	-

3. Select Repeatable as the category, and give it some kind of description that will summise the contents of the script as that will be its name! For me, I created a script that creates a table, with the description “CreateTable”, and making sure the type was “.sql” as can be seen below.



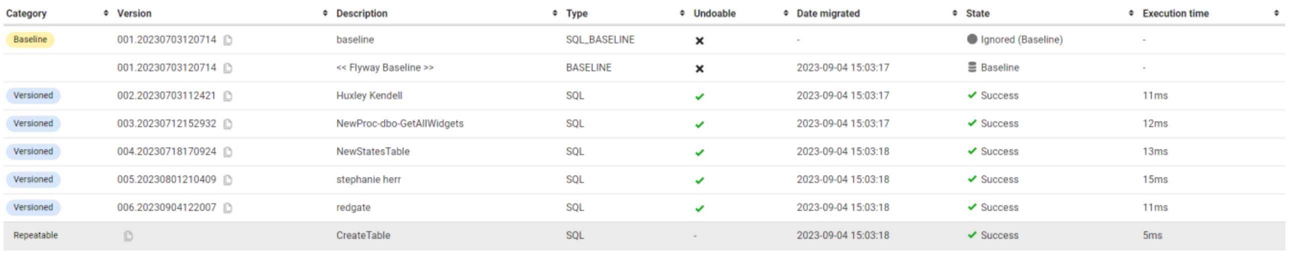
Category: ○ Versioned ○ Undo [✓ Repeatable] ○ Baseline

Description: CreateTable

Type: .sql

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

4. Click Save, and as the script has not run before on this DB, we can hit migrate and watch it deploy!

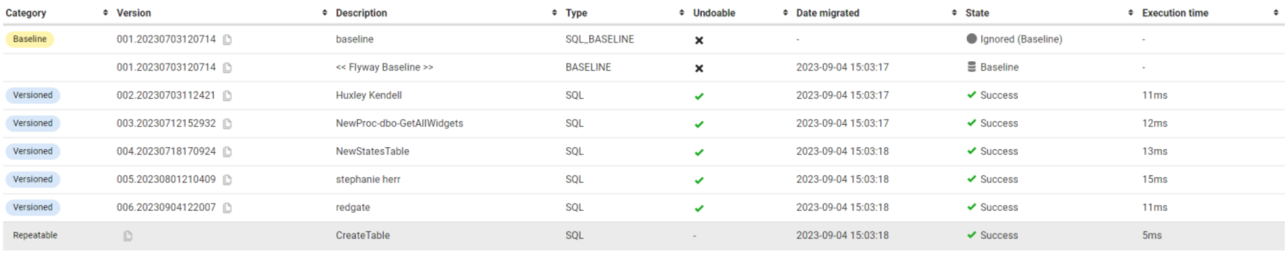


Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	● Ignored (Baseline)	-
	001.20230703120714	<< Flyway Baseline >>	BASELINE	✗	2023-09-04 15:03:17	Baseline	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	2023-09-04 15:03:17	✓ Success	11ms
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	2023-09-04 15:03:17	✓ Success	12ms
Versioned	004.20230718170924	NewStatesTable	SQL	✓	2023-09-04 15:03:18	✓ Success	13ms
Versioned	005.20230801210409	stephanie herr	SQL	✓	2023-09-04 15:03:18	✓ Success	15ms
Versioned	006.20230904122007	redgate	SQL	✓	2023-09-04 15:03:18	✓ Success	11ms
Repeatable		CreateTable	SQL	-	2023-09-04 15:03:18	✓ Success	5ms

You can make the script whatever for this example, feel free to copy my code, but try and make it something you can easily change and re-run!

15. Change Repeatable Script

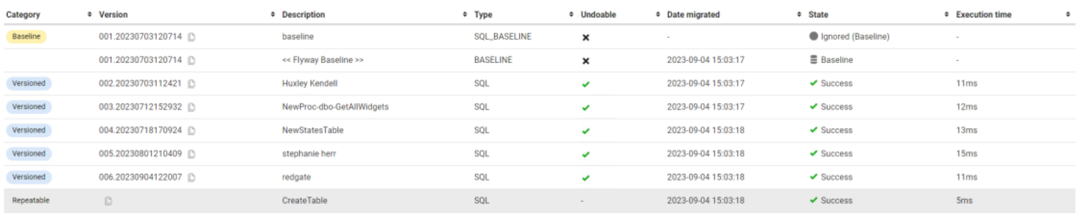
1. Now that we have our repeatable
2. script, and have run it successfully, we can try and see how we get it to run again! Flyway has what we call a checksum against each script, in this use case is very



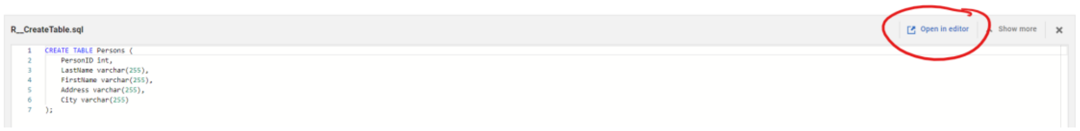
Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	● Ignored (Baseline)	-
	001.20230703120714	<< Flyway Baseline >>	BASELINE	✗	2023-09-04 15:03:17	Baseline	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	2023-09-04 15:03:17	✓ Success	11ms
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	2023-09-04 15:03:17	✓ Success	12ms
Versioned	004.20230718170924	NewStatesTable	SQL	✓	2023-09-04 15:03:18	✓ Success	13ms
Versioned	005.20230801210409	stephanie herr	SQL	✓	2023-09-04 15:03:18	✓ Success	15ms
Versioned	006.20230904122007	redgate	SQL	✓	2023-09-04 15:03:18	✓ Success	11ms
Repeatable		CreateTable	SQL	-	2023-09-04 15:03:18	✓ Success	5ms

helpful at making sure we do not deploy changes that have already been deployed! Before we move on, we want to make sure our repeatable script is marked as successfully deployed as seen above!

3. So, lets change our script and see about getting it to *repeat* itself. Flyway makes this quite easy, click on our Repeatable script and we will be given the option to "Open in editor", click this and we can begin changing our name!



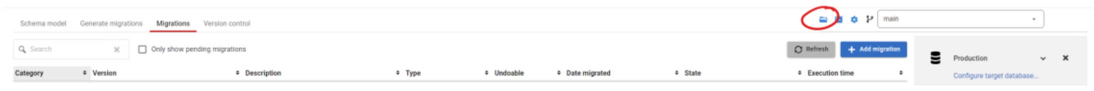
Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	● Ignored (Baseline)	-
	001.20230703120714	<< Flyway Baseline >>	BASELINE	✗	2023-09-04 15:03:17	Baseline	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	2023-09-04 15:03:17	✓ Success	11ms
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	2023-09-04 15:03:17	✓ Success	12ms
Versioned	004.20230718170924	NewStatesTable	SQL	✓	2023-09-04 15:03:18	✓ Success	13ms
Versioned	005.20230801210409	stephanie herr	SQL	✓	2023-09-04 15:03:18	✓ Success	15ms
Versioned	006.20230904122007	redgate	SQL	✓	2023-09-04 15:03:18	✓ Success	11ms
Repeatable		CreateTable	SQL	-	2023-09-04 15:03:18	✓ Success	5ms



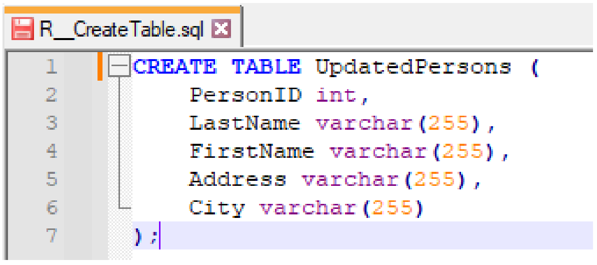
R__CreateTable.sql

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

Just in case it opens somewhere, undesirable. We can also use Flyways handy "Folder" icon next to the CLI icon used before, to open directly into our Project folder. Navigate to the "Migrations" folder, and we can edit our script from there too!



4. Make whatever change you see fit, remembering that this is going to get deployed so make it a change that will work! If in doubt, use the updated script I made, I simply changed the name of the table it was going to create!
The important thing however, is that the script is altered!

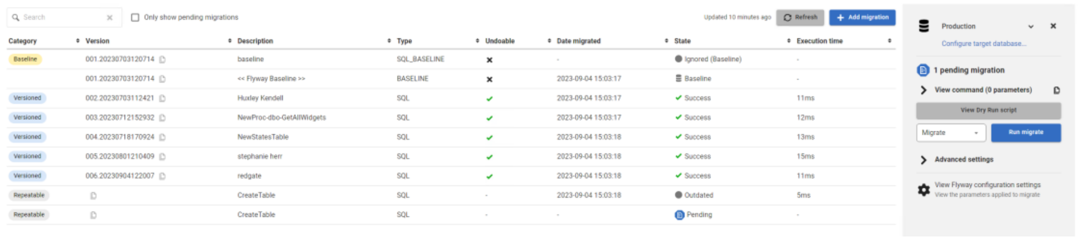


R__CreateTable.sql

```
CREATE TABLE UpdatedPersons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

5. Head back to Flyway, hit refresh on the migrations page, and we should now see that Flyway now wants to deploy the script, and it has been marked as pending, as

expected!

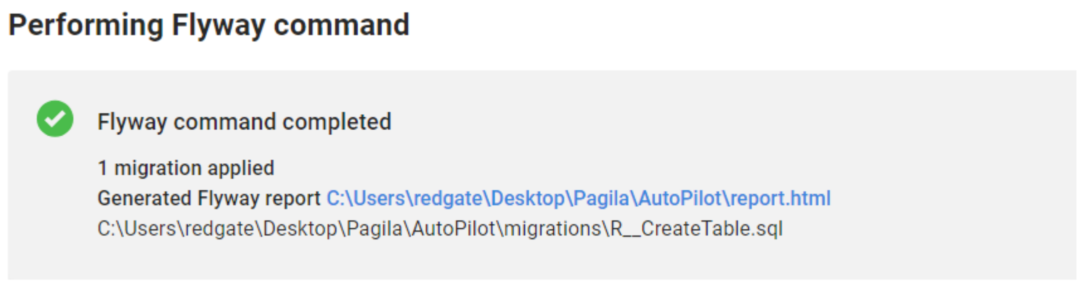


Search | Only show pending migrations | Updated 10 minutes ago | Refresh | Add migration

Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	● Ignored (Baseline)	-
	001.20230703120714	<< Flyway Baseline >>	BASELINE	✗	2023-09-04 15:03:17	Baseline	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	2023-09-04 15:03:17	✓ Success	11ms
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	2023-09-04 15:03:17	✓ Success	12ms
Versioned	004.20230718170924	NewStatesTable	SQL	✓	2023-09-04 15:03:18	✓ Success	13ms
Versioned	005.20230801210409	stephanie herr	SQL	✓	2023-09-04 15:03:18	✓ Success	15ms
Versioned	006.20230904122007	redgate	SQL	✓	2023-09-04 15:03:18	✓ Success	11ms
Repeatable		CreateTable	SQL	-	2023-09-04 15:03:18	● Outdated	5ms
Repeatable		CreateTable	SQL	-	-	Pending	-

Production
Configure target database...
1 pending migration
View command (0 parameters)
View Dry Run script
Migrate
Run migrate
Advanced settings
View Flyway configuration settings
View the parameters applied to migrate

6. Hit migrate to then deploy any pending scripts, including our repeatable script!



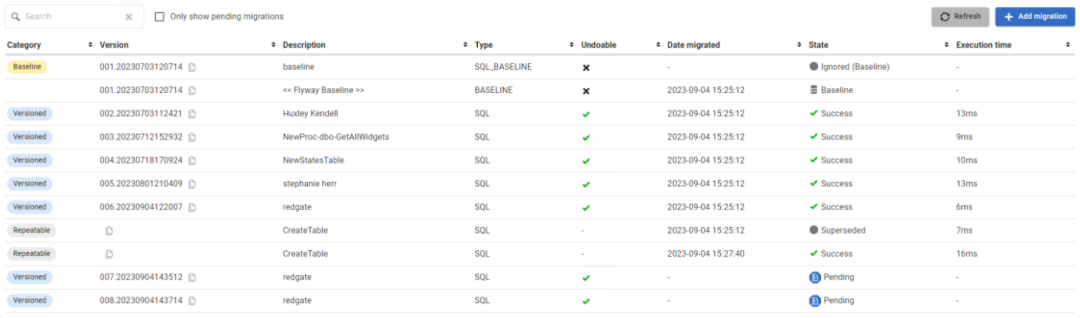
Performing Flyway command

Flyway command completed

1 migration applied
Generated Flyway report C:\Users\redgate\Desktop\Pagila\AutoPilot\report.html
C:\Users\redgate\Desktop\Pagila\AutoPilot\migrations\R__CreateTable.sql

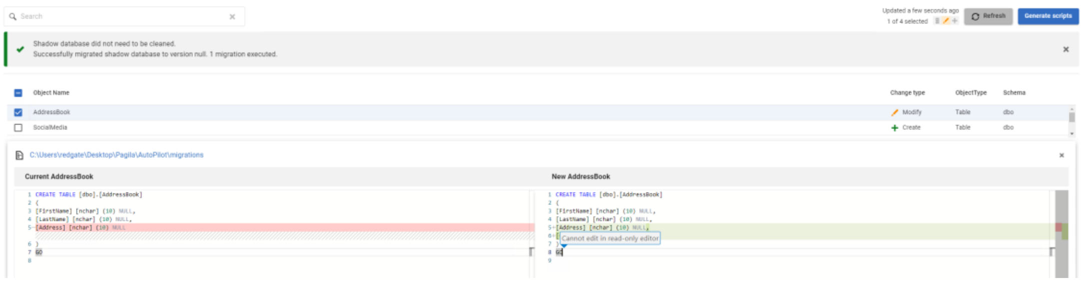
16. Create Two Migration Scripts, Cherry Pick Deployment

1. An important aspect of Flyway, is making sure we customise it too match your own needs. One way we can do this is to use additional parameters!
2. Before we can do this, we need some pre-requisite scripts to deploy, this is a good chance to test your own knowledge! Try and make two separate migration scripts now, without deploying to any downstream environments! You should be able to get to a screen similar to mine!



Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	● Ignored (Baseline)	-
	001.20230703120714	<< Flyway Baseline >>	BASELINE	✗	2023-09-04 15:25:12	Baseline	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	2023-09-04 15:25:12	✓ Success	13ms
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	2023-09-04 15:25:12	✓ Success	9ms
Versioned	004.20230718170924	NewStatesTable	SQL	✓	2023-09-04 15:25:12	✓ Success	10ms
Versioned	005.20230801210409	stephanie herr	SQL	✓	2023-09-04 15:25:12	✓ Success	13ms
Versioned	006.20230904122007	redgate	SQL	✓	2023-09-04 15:25:12	✓ Success	6ms
Repeatable		CreateTable	SQL	-	2023-09-04 15:25:12	● Superseded	7ms
Repeatable		CreateTable	SQL	-	2023-09-04 15:27:40	✓ Success	16ms
Versioned	007.20230904143512	redgate	SQL	✓	-	Pending	-
Versioned	008.20230904143714	redgate	SQL	✓	-	Pending	-

The simplest way to do so, is to create two changes, and only tick one box when creating a migration script, as seen in the screenshot below!



Shadow database did not need to be cleaned.
Successfully migrated shadow database to version null. 1 migration executed.

Object Name	Change type	ObjectType	Schema
✓ AddressBook	Modify	Table	dbo
☐ SocialMedia	Create	Table	dbo

C:\Users\redgate\Desktop\Pagila\AutoPilot\migrations

Current AddressBook

```
CREATE TABLE [dbo].[AddressBook]
(
[FirstName] [nchar] (10) NULL,
[LastName] [nchar] (10) NULL,
[Address] [nchar] (10) NULL
)
GO
```

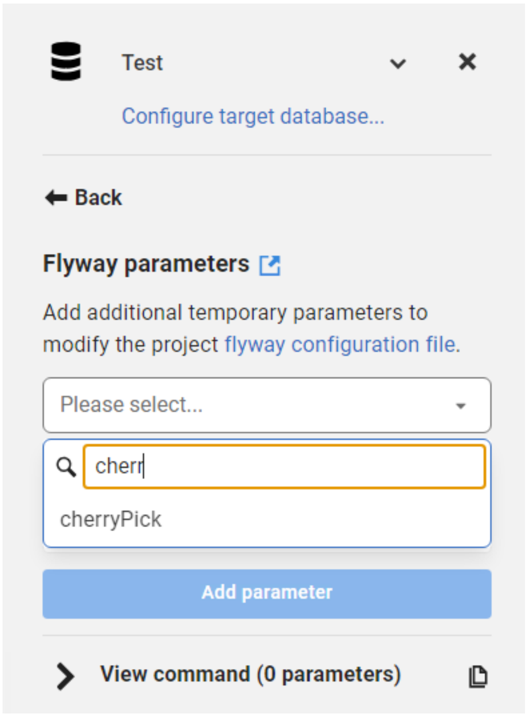
New AddressBook

```
CREATE TABLE [dbo].[AddressBook]
(
[FirstName] [nchar] (10) NULL,
[LastName] [nchar] (10) NULL,
[Address] [nchar] (10) NULL,
[
)
GO
```

Remember, if at any time you get stuck, you can always refresh the repository off the main, and start over!

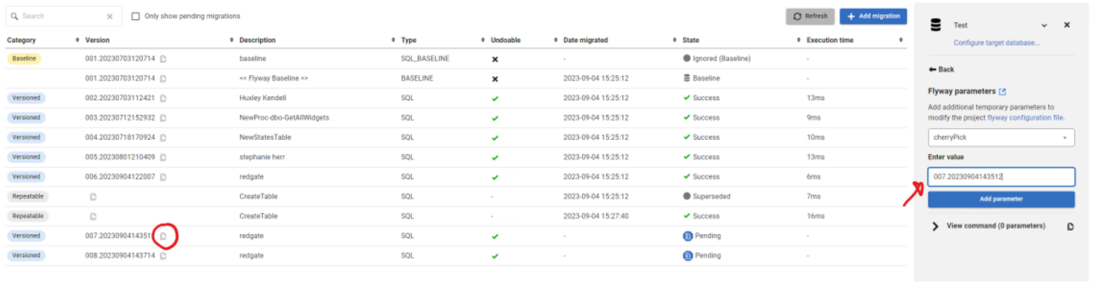
3. As we know, a basic migration is going to deploy both of these scripts, with the goal of making your life as DBA easier without the need to specify.
   But sometimes we want more control, so before we hit migrate go down into

Advanced settings and find the "CherryPick" parameter!



Add additional temporary parameters to modify the project flyway configuration file.

4. Flyway handily lets up copy the version name of the script, which is the value needed for the parameter, so copy the version we want to deploy and add the parameter!



Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Baseline	001.20230703120714	baseline	SQL_BASELINE	✗	-	● Ignored (Baseline)	-
	001.20230703120714	<< Flyway Baseline >>	BASELINE	✗	2023-09-04 15:25:12	Baseline	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	2023-09-04 15:25:12	✓ Success	13ms
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	2023-09-04 15:25:12	✓ Success	9ms
Versioned	004.20230718170924	NewStatesTable	SQL	✓	2023-09-04 15:25:12	✓ Success	10ms
Versioned	005.20230801210409	stephanie herr	SQL	✓	2023-09-04 15:25:12	✓ Success	13ms
Versioned	006.20230904122007	redgate	SQL	✓	2023-09-04 15:25:12	✓ Success	6ms
Repeatable		CreateTable	SQL	-	2023-09-04 15:25:12	● Superseded	7ms
Repeatable		CreateTable	SQL	-	2023-09-04 15:27:40	✓ Success	16ms
Versioned	007.2023090414351	redgate	SQL	✓	-	Pending	-
Versioned	008.20230904143714	redgate	SQL	✓	-	Pending	-

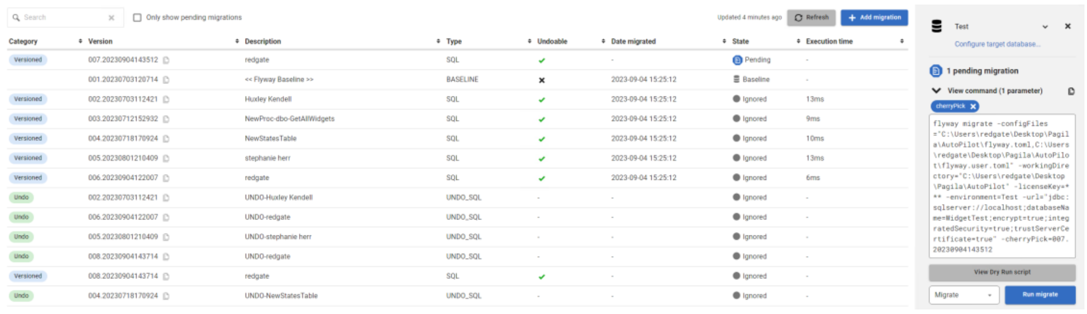
Add additional temporary parameters to modify the project flyway configuration file.

cherryPick

Enter value

007.20230904143512

5. Once we add the parameter, we will be presented with the view of what Flyway intends to do. We should see that the only script not ignored is the one we selected, if so then click migrate and watch Flyway CherryPick our deployments!

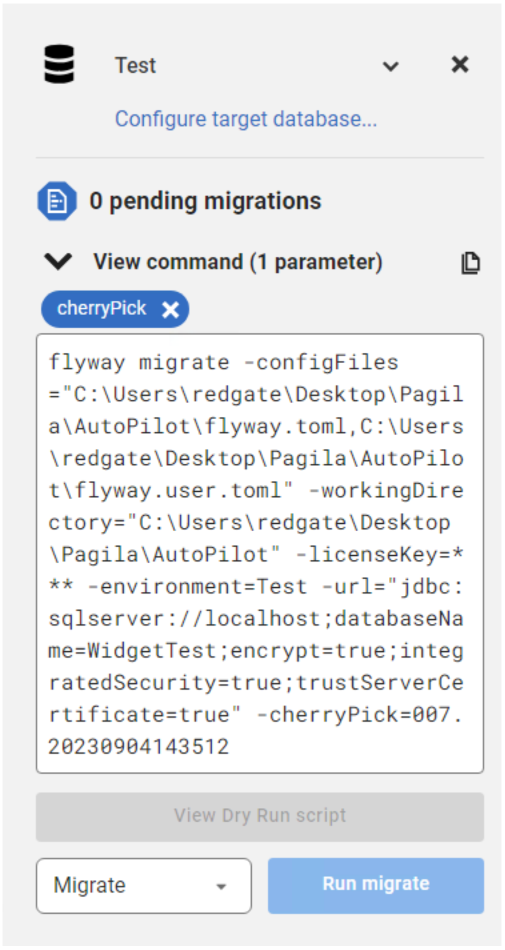


Category	Version	Description	Type	Undoable	Date migrated	State	Execution time
Versioned	007.20230904143512	redgate	SQL	✓	-	Pending	-
	001.20230703120714	<< Flyway Baseline >>	BASELINE	✗	2023-09-04 15:25:12	Baseline	-
Versioned	002.20230703112421	Huxley Kendell	SQL	✓	2023-09-04 15:25:12	Ignored	13ms
Versioned	003.20230712152932	NewProc-dbo-GetAllWidgets	SQL	✓	2023-09-04 15:25:12	Ignored	9ms
Versioned	004.20230718170924	NewStatesTable	SQL	✓	2023-09-04 15:25:12	Ignored	10ms
Versioned	005.20230801210409	stephanie herr	SQL	✓	2023-09-04 15:25:12	Ignored	13ms
Versioned	006.20230904122007	redgate	SQL	✓	2023-09-04 15:25:12	Ignored	6ms
Undo	002.20230703112421	UNDO-Huxley Kendell	UNDO_SQL	-	-	Ignored	-
Undo	006.20230904122007	UNDO-redgate	UNDO_SQL	-	-	Ignored	-
Undo	005.20230801210409	UNDO-stephanie herr	UNDO_SQL	-	-	Ignored	-
Undo	008.20230904143714	UNDO-redgate	UNDO_SQL	-	-	Ignored	-
Versioned	008.20230904143714	redgate	SQL	✓	-	Ignored	-
Undo	004.20230718170924	UNDO-NewStatesTable	UNDO_SQL	-	-	Ignored	-

```
flyway migrate -configFiles
="C:\Users\redgate\Desktop\Pagil
a\AutoPilot\flyway.toml,C:\Users
\redgate\Desktop\Pagila\AutoPilo
t\flyway.user.toml" -workingDire
ctory="C:\Users\redgate\Desktop
\Pagila\AutoPilot" -licenseKey=*
** -environment=Test -url="jdbc:
sqlserver://localhost;databaseNa
me=WidgetTest;encrypt=true;integ
ratedSecurity=true;trustServerCe
rtificate=true" -cherryPick=007.
20230904143512
```

6. It is also important to make sure, as we get more experienced, we know how to do this via the CLI. So when it comes to implementing CI/CD we feel more comfortable! Whilst the box above the migrate command, seen below, does demonstrate and allow for copying the script, lets try and build it out ourselves!
   The output as seen below, is incredibly useful in the CI/CD, as it includes every

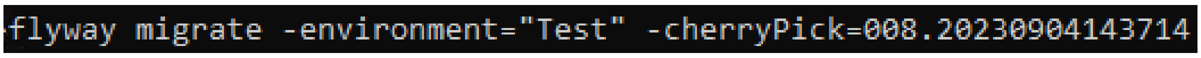
parameter you may need!



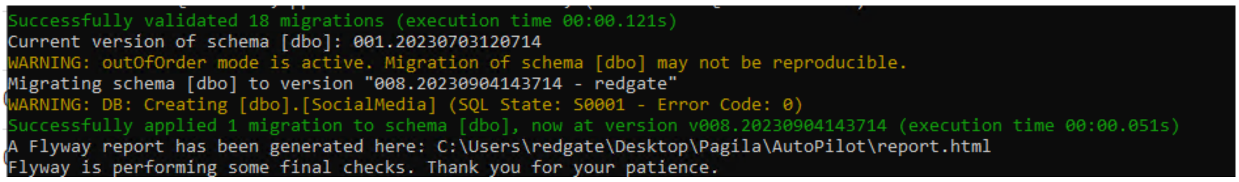
```
flyway migrate -configFiles
="C:\Users\redgate\Desktop\Pagil
a\AutoPilot\flyway.toml,C:\Users
\redgate\Desktop\Pagila\AutoPilo
t\flyway.user.toml" -workingDire
ctory="C:\Users\redgate\Desktop
\Pagila\AutoPilot" -licenseKey=*
** -environment=Test -url="jdbc:
sqlserver://localhost;databaseNa
me=WidgetTest;encrypt=true;integ
ratedSecurity=true;trustServerCe
rtificate=true" -cherryPick=007.
20230904143512
```

7. So open up the CLI as we have learnt, and enter the following parameters with the values they need:
   All Flyway CLI commands begin with the syntax of "***Flyway*". **
   The syntax we need after this is "***Migrate*". ** This grabs all the information over the specified DB and deploys any pending scripts, unless specified otherwise!
   We then need to tell Flyway which DB we wish to perform the command against, we can do this two ways. We either use the parameter of ***"-Url={JDBC URL of Target DB}"*** or ***"Environment={ID}"***
   Both of these parameters can be found inside the "configure target database"
   Now we need to add our additional parameter of ***"-cherryPick={VersionID of Script}".***

8. Our command should look to be the following, use the only pending script you have as the one we wish to cherryPick!

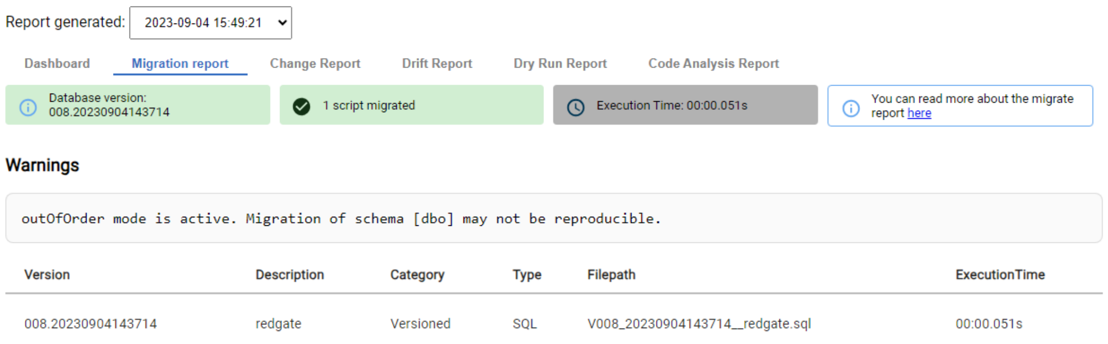


```
flyway migrate -environment="Test" -cherryPick=008.20230904143714
```



```
Successfully validated 18 migrations (execution time 00:00.121s)
Current version of schema [dbo]: 001.20230703120714
WARNING: outOfOrder mode is active. Migration of schema [dbo] may not be reproducible.
Migrating schema [dbo] to version "008.20230904143714 - redgate"
WARNING: DB: Creating [dbo].[SocialMedia] (SQL State: S0001 - Error Code: 0)
Successfully applied 1 migration to schema [dbo], now at version v008.20230904143714 (execution time 00:00.051s)
A Flyway report has been generated here: C:\Users\redgate\Desktop\Pagila\AutoPilot\report.html
Flyway is performing some final checks. Thank you for your patience.
```

9. We can perform two things to validate our changes, head to Flyway Desktop and refresh the migrations tab OR look at the report it generates after our CLI deployment!



Report generated: 2023-09-04 15:49:21

Database version: 008.20230904143714

1 script migrated

Execution Time: 00:00.051s

You can read more about the migrate report here

Warnings

outOfOrder mode is active. Migration of schema [dbo] may not be reproducible.

Version	Description	Category	Type	Filepath	ExecutionTime
008.20230904143714	redgate	Versioned	SQL	V008_20230904143714__redgate.sql	00:00.051s

17. Deploy Via Git Actions Pipeline

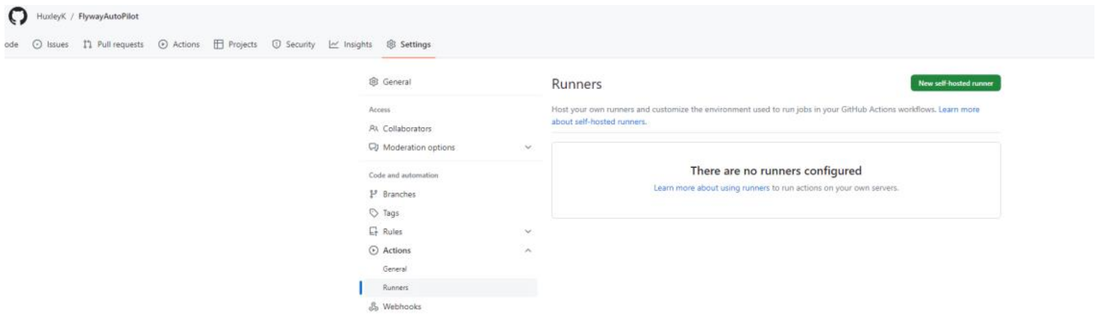
1. What we have learnt so far should have onboarded you well, and opened your mind to how we can use Flyway in many different ways! And after some more exploring you should become fluid and very comfortable operating with Flyway, with the end goal being CI/CD use!

2. Luckily for you, if you have not already seen, Flyway AutoPilot comes with full CI/CD usage ready!

3. First of all, before moving over. Make sure we have at least one migration script pending before, otherwise it'll make for a very uneventful pipeline deployment!

   Secondly make sure we have configured two things:

   GitHub Actions Runner! The intro video should of covered this, but if in any doubt head to the repository you made to get all the files etc, and go to "Settings > Actions > Runners". Any runners will appear here, but it is incredibly easy to create a new one!



Runners

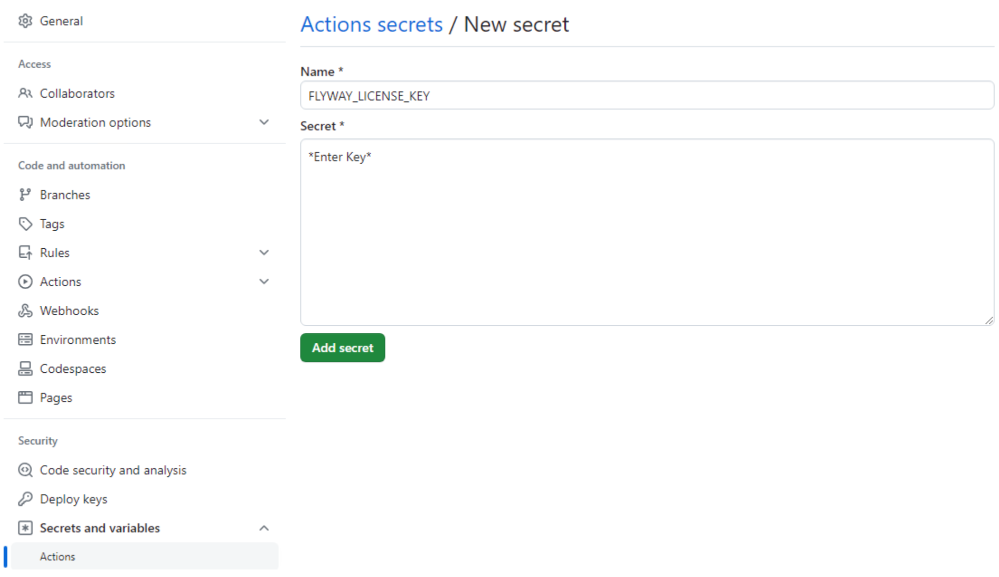
Host your own runners and customize the environment used to run jobs in your GitHub Actions workflows. Learn more about self-hosted runners.

There are no runners configured

Learn more about using runners to run actions on your own servers.

   If no runner appears, click "New Self-Hosted Runner" and follow the simple creation steps! If any issues, revert back to the video here!

   Make sure the **Flyway-License-Key** is set. Inside the Repository head to "Settings > Secrets and Variables > Actions". Click "New Repository Secret" and enter the following (including your own key, these can be gained from Redgate, if inside your 14 day trial please skip)



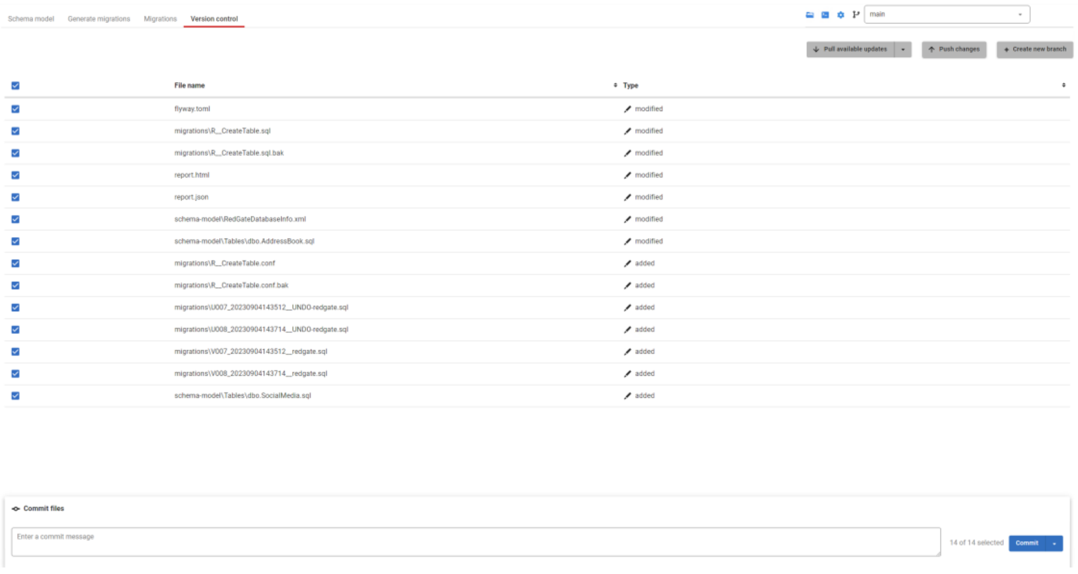
Name *

FLYWAY_LICENSE_KEY

Secret *

Enter Key

4. Now that we have everything ready, we just need to make sure the migration script that has yet to be deployed, is inside our repository!
   This is a case of creating a pending script, making sure we are on the correct branch (this can be merged from GitHub if not) and then pushing our changes!

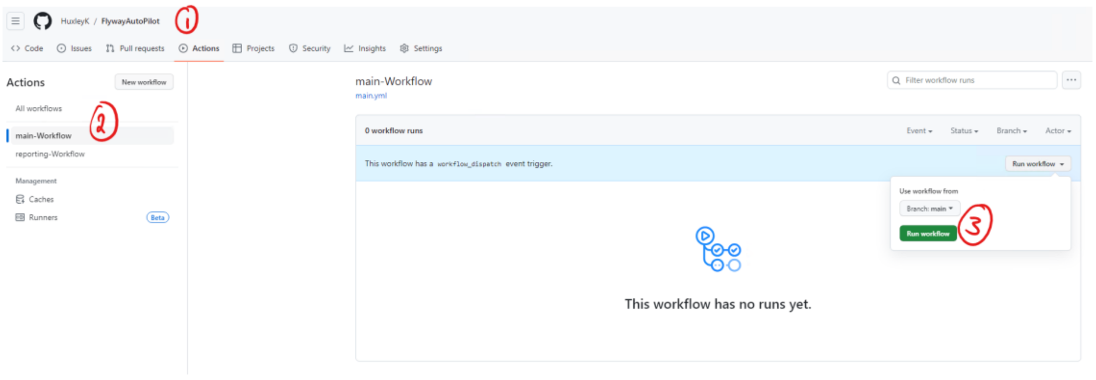


✓	File name	Type
✓	flyway.toml	modified
✓	migrations\R__CreateTable.sql	modified
✓	migrations\R__CreateTable.sql.bak	modified
✓	report.html	modified
✓	report.json	modified
✓	schema-model\RedGateDatabaseInfo.xml	modified
✓	schema-model\Tables\dbo.AddressBook.sql	modified
✓	migrations\R__CreateTable.conf	added
✓	migrations\R__CreateTable.conf.bak	added
✓	migrations\U007_20230904143512__UNDO-redgate.sql	added
✓	migrations\U008_20230904143714__UNDO-redgate.sql	added
✓	migrations\V007_20230904143512__redgate.sql	added
✓	migrations\V008_20230904143714__redgate.sql	added
✓	schema-model\Tables\dbo.SocialMedia.sql	added

Enter a commit message

14 of 14 selected

5. Head over to our repository, and click into GitHub “Actions”, from there make sure we run the “main-Workflow” YAML and watch as our Pipeline performs its tasks!



This workflow has a `workflow_dispatch` event trigger.

This workflow has no runs yet.

6. As seen in image below, our pipeline has three stages. Feel free to click into any stage and watch it perform its tasks.
   This pipeline performs the following:

   Build Deployment:
   This deploys to a build database, a non-live database. First it performs a **Flyway Clean** command, this removes any scripts that have been generated and allows us to start from fresh, important when testing if our scripts will stand up!
   Next we perform a **Flyway Migrate**, allowing us to deploy not just our new script but all of our scripts together, giving a higher level of security!
   Finally we perform a **Flyway Undo** while **CherryPicking** the first undo script, this then tests all of our rollback scripts by rollback to the first version possible!
   The build is a very good stage to run any T-SQL tests or Automated tests needed, as it is a perfect copy of Production that is non-live!

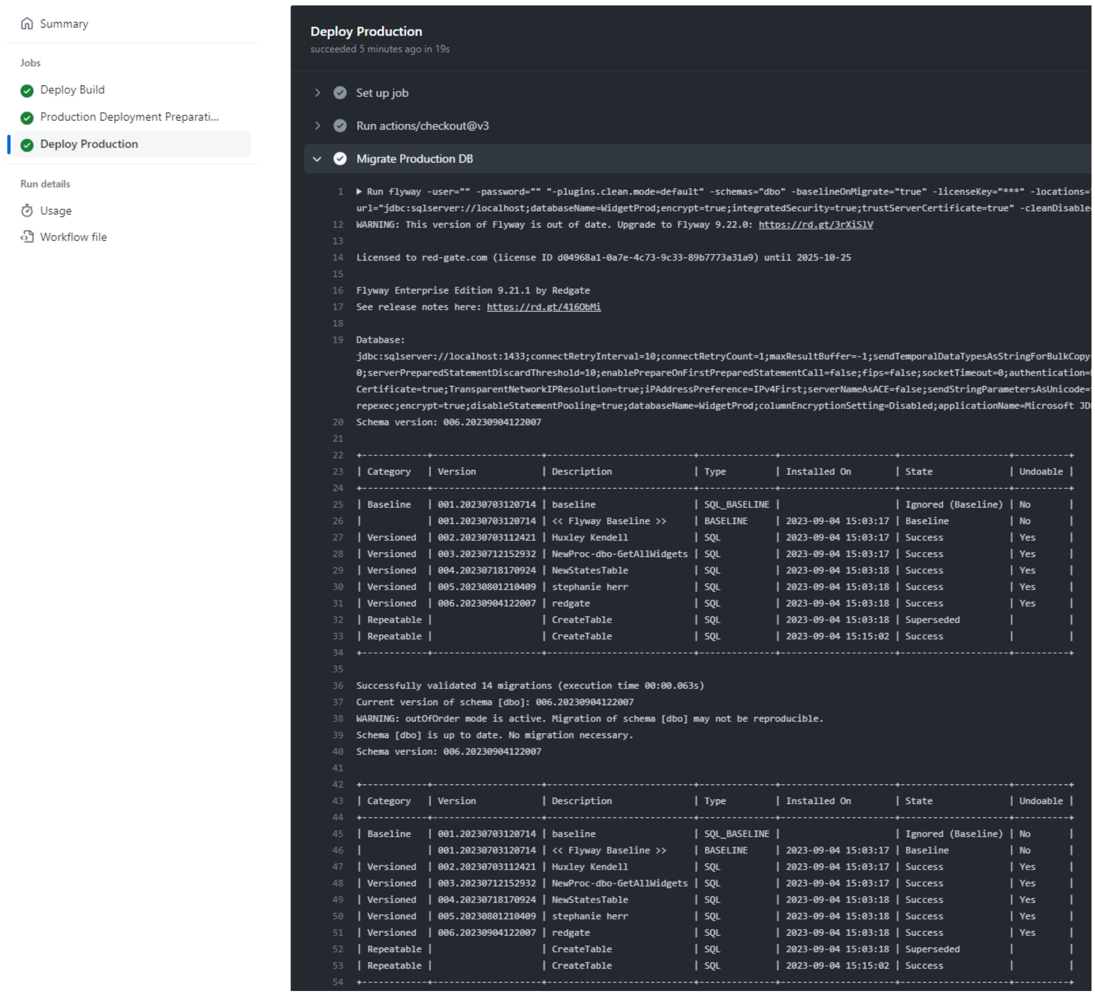
   Production Report Generation:
   The second stage is important for generating our reporting, and allowing for much higher visibility.
   This stage performs the **Flyway Check** commands, allowing us to generate all four types of reporting about our Flyway deployment to the production DB, before we deploy!

Production Deployment:

Once we have deployed to a Build DB, and generated reports. We move straight to Deploying our single new script to the production DB, this uses a basic **Flyway Migrate** command, meaning if we have 1 or 100 pending scripts, the outcome will be the same. Our Production DB at the latest version!

There is much to be desired about this pipeline, but it acts as a great starting point for you, we will explore how this can be improved later on but feel free to explore (it can always be re-made).

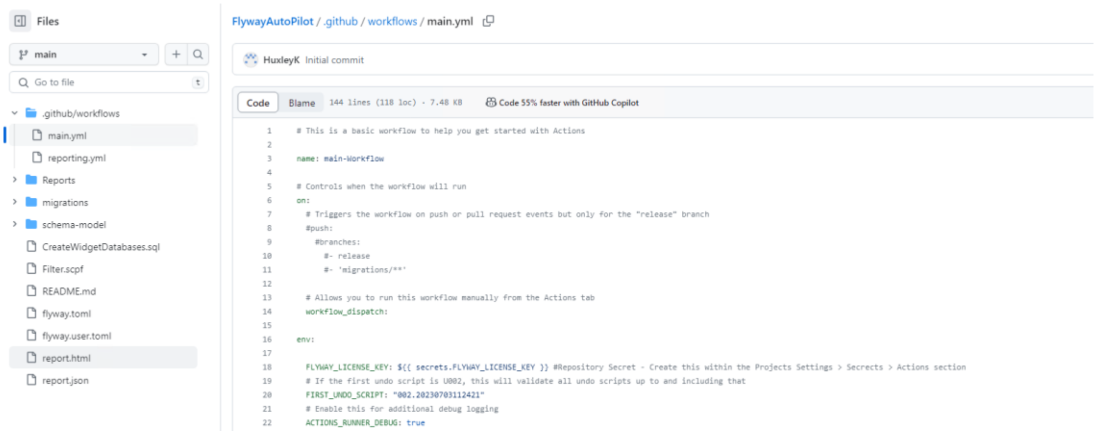
Check out the artifacts a the bottom if not already, to see the reports it makes for us!



```
▶ Run flyway -user="" -password="" "-plugins.clean.mode=default" -schemas="dbo" -baselineOnMigrate="true" -licenseKey="***" -locations=
url="jdbc:sqlserver://localhost;databaseName=WidgetProd;encrypt=true;integratedSecurity=true;trustServerCertificate=true" -cleanDisable
WARNING: This version of Flyway is out of date. Upgrade to Flyway 9.22.0: https://rd.gt/3rXiSlV

Licensed to red-gate.com (license ID d04968a1-0a7e-4c73-9c33-89b7773a31a9) until 2025-10-25

Flyway Enterprise Edition 9.21.1 by Redgate
See release notes here: https://rd.gt/416ObMi

Database:
jdbc:sqlserver://localhost:1433;connectRetryInterval=10;connectRetryCount=1;maxResultBuffer=-1;sendTemporalDataTypesAsStringForBulkCopy
0;serverPreparedStatementDiscardThreshold=10;enablePrepareOnFirstPreparedStatementCall=false;fips=false;socketTimeout=0;authentication=
Certificate=true;TransparentNetworkIPResolution=true;iPAddressPreference=IPv4First;serverNameAsACE=false;sendStringParametersAsUnicode=
repexec;encrypt=true;disableStatementPooling=true;databaseName=WidgetProd;columnEncryptionSetting=Disabled;applicationName=Microsoft JD
Schema version: 006.20230904122007

+------------+--------------------+---------------------------+--------------+---------------------+--------------------+----------+
| Category   | Version            | Description               | Type         | Installed On        | State              | Undoable |
+------------+--------------------+---------------------------+--------------+---------------------+--------------------+----------+
| Baseline   | 001.20230703120714 | baseline                  | SQL_BASELINE |                     | Ignored (Baseline) | No       |
|            | 001.20230703120714 | << Flyway Baseline >>     | BASELINE     | 2023-09-04 15:03:17 | Baseline           | No       |
| Versioned  | 002.20230703112421 | Huxley Kendell            | SQL          | 2023-09-04 15:03:17 | Success            | Yes      |
| Versioned  | 003.20230712152932 | NewProc-dbo-GetAllWidgets | SQL          | 2023-09-04 15:03:17 | Success            | Yes      |
| Versioned  | 004.20230718170924 | NewStatesTable            | SQL          | 2023-09-04 15:03:18 | Success            | Yes      |
| Versioned  | 005.20230801210409 | stephanie herr            | SQL          | 2023-09-04 15:03:18 | Success            | Yes      |
| Versioned  | 006.20230904122007 | redgate                   | SQL          | 2023-09-04 15:03:18 | Success            | Yes      |
| Repeatable |                    | CreateTable               | SQL          | 2023-09-04 15:03:18 | Superseded         |          |
| Repeatable |                    | CreateTable               | SQL          | 2023-09-04 15:15:02 | Success            |          |
+------------+--------------------+---------------------------+--------------+---------------------+--------------------+----------+


Successfully validated 14 migrations (execution time 00:00.063s)
Current version of schema [dbo]: 006.20230904122007
WARNING: outOfOrder mode is active. Migration of schema [dbo] may not be reproducible.
Schema [dbo] is up to date. No migration necessary.
Schema version: 006.20230904122007

+------------+--------------------+---------------------------+--------------+---------------------+--------------------+----------+
| Category   | Version            | Description               | Type         | Installed On        | State              | Undoable |
+------------+--------------------+---------------------------+--------------+---------------------+--------------------+----------+
| Baseline   | 001.20230703120714 | baseline                  | SQL_BASELINE |                     | Ignored (Baseline) | No       |
|            | 001.20230703120714 | << Flyway Baseline >>     | BASELINE     | 2023-09-04 15:03:17 | Baseline           | No       |
| Versioned  | 002.20230703112421 | Huxley Kendell            | SQL          | 2023-09-04 15:03:17 | Success            | Yes      |
| Versioned  | 003.20230712152932 | NewProc-dbo-GetAllWidgets | SQL          | 2023-09-04 15:03:17 | Success            | Yes      |
| Versioned  | 004.20230718170924 | NewStatesTable            | SQL          | 2023-09-04 15:03:18 | Success            | Yes      |
| Versioned  | 005.20230801210409 | stephanie herr            | SQL          | 2023-09-04 15:03:18 | Success            | Yes      |
| Versioned  | 006.20230904122007 | redgate                   | SQL          | 2023-09-04 15:03:18 | Success            | Yes      |
| Repeatable |                    | CreateTable               | SQL          | 2023-09-04 15:03:18 | Superseded         |          |
| Repeatable |                    | CreateTable               | SQL          | 2023-09-04 15:15:02 | Success            |          |
+------------+--------------------+---------------------------+--------------+---------------------+--------------------+----------+
```

18. Add Additional Deployment Step inside YML

1. So far the pipeline has worked great, but we want it to better reflect actual deployments. One thing that is missing is deploying to a Test database first, to allow for more thorough testing. Head into the GitHub repository, and let's edit the "Main.yml" file like I am in the image below!



```yaml
# This is a basic workflow to help you get started with Actions

name: main-Workflow

# Controls when the workflow will run
on:
  # Triggers the workflow on push or pull request events but only for the "release" branch
  #push:
    #branches:
      #- release
      #- 'migrations/**'

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:

env:

  FLYWAY_LICENSE_KEY: ${{ secrets.FLYWAY_LICENSE_KEY }} #Repository Secret - Create this within the Projects Settings > Secrects > Actions section
  # If the first undo script is U002, this will validate all undo scripts up to and including that
  FIRST_UNDO_SCRIPT: "002.20230703112421"
  # Enable this for additional debug logging
  ACTIONS_RUNNER_DEBUG: true
```

2. Let us first break-down our production YML before trying ourselves:

```yaml
prod:
  name: Deploy Production
  # The type of runner that the job will run on
  runs-on: self-hosted
  environment: 'prod' #Ensure this environment name is setup in the projects Settings>Environment area. Ensuring any reviewers are also configured
  if: ${{ true }} #Set this variable to false to temporarily disable the job
  needs: prod-preparation
```

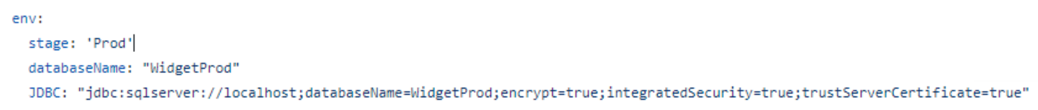
The first few lines allow us to set some configuration:

"name: Deploy Production" being what it will display as!

"runs-on" Here we can specify a different set of runners if needed, but as ours are self hosted that's all we need to put!

"environment:" this simply declares an environment variable, if we need to call it later.

"needs: prod-preparation" Does this stage need something else to run first, before it can deploy! In our case, prod needs the reports to be made first so that is why it has been set!



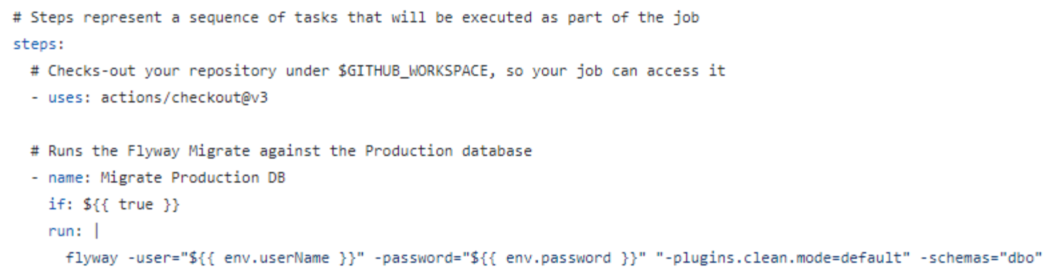
```
env:
  stage: 'Prod'
  databaseName: "WidgetProd"
  JDBC: "jdbc:sqlserver://localhost;databaseName=WidgetProd;encrypt=true;integratedSecurity=true;trustServerCertificate=true"
```

Here we are declaring variables to be used later on, this means we can set it once, instead of potentially writing multiple times!

“Stage:” What stage does this correspond too, in our case Production!

“databaseName:” What is the name of Database, needed for certain reports and some connection strings!

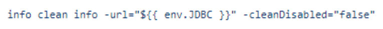
“JDBC” Very important, this is added to our Flyway commands so they can be set once in the YAML, instead of having to write Full flyway commands, whatever we set here is what is used for the commands.



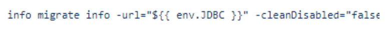
```
# Steps represent a sequence of tasks that will be executed as part of the job
steps:
  # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
  - uses: actions/checkout@v3

  # Runs the Flyway Migrate against the Production database
  - name: Migrate Production DB
    if: ${{ true }}
    run: |
      flyway -user="${{ env.userName }}" -password="${{ env.password }}" "-plugins.clean.mode=default" -schemas="dbo"
```

Finally, we set the actual commands we need to run, this is where we get most freedom. We can intermix Flyway with other commands or scripts, but the important thing is you have full control over the DB with Flyway.

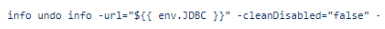
In our Build stage, we run multiple commands, like clean, migrate and undo. As seen below:



```
info clean info -url="${{ env.JDBC }}" -cleanDisabled="false"
```

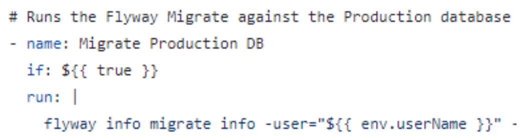


```
info migrate info -url="${{ env.JDBC }}" -cleanDisabled="false
```



```
info undo info -url="${{ env.JDBC }}" -cleanDisabled="false" -
```

But with this particular stage, all we need is a Flyway info migrate (we can attach info to any additional commands to perform an info before / after / before & after).

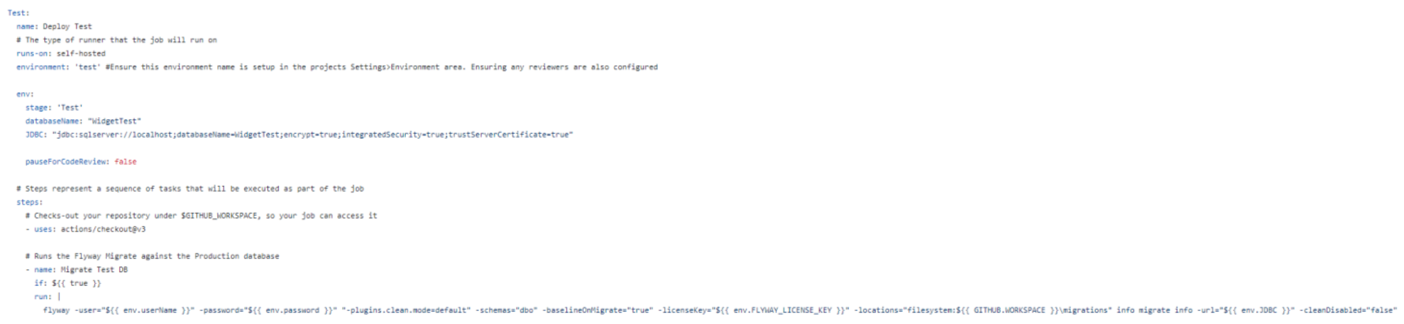


```
# Runs the Flyway Migrate against the Production database
- name: Migrate Production DB
  if: ${{ true }}
  run: |
    flyway info migrate info -user="${{ env.userName }}" -
```

As we can see, for our Production deployments, all we needed was an info command (to see that prior state), a migrate (to deploy any pending scripts) and a final info command (see updated state)!

3. Ideally, we want our Test deployments to come before it deploys to Production, but after it generates its reports! This gives us ample time to test anything needed, with the appropriate reporting to aid us.
   So, let's see if we can update our YAML.
   We ideally want all of our deployments to be repeatable and follow the same standard, so let's base this off our Prod deployments.

4. Starting on line '119', see if you can do this yourselves, if not follow the example I did below to get it working:

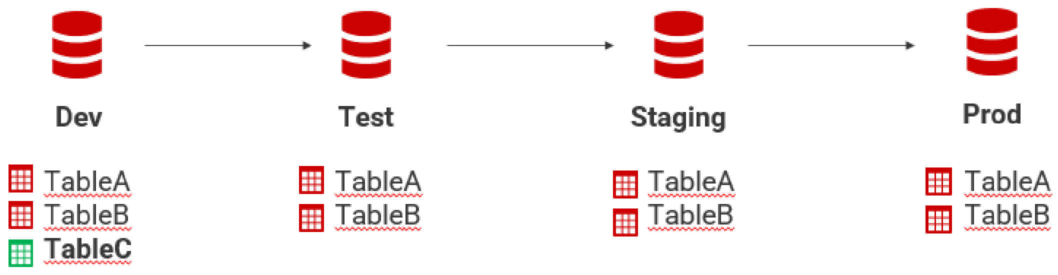


```
Test:
  name: Deploy Test
  # The type of runner that the job will run on
  runs-on: self-hosted
  environment: 'test' #Ensure this environment name is setup in the projects Settings>Environment area. Ensuring any reviewers are also configured

  env:
    stage: 'Test'
    databaseName: "WidgetTest"
    JDBC: "jdbc:sqlserver://localhost;databaseName=WidgetTest;encrypt=true;integratedSecurity=true;trustServerCertificate=true"

    pauseForCodeReview: false

  # Steps represent a sequence of tasks that will be executed as part of the job
  steps:
    # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
    - uses: actions/checkout@v3

    # Runs the Flyway Migrate against the Production database
    - name: Migrate Test DB
      if: ${{ true }}
      run: |
        flyway -user="${{ env.userName }}" -password="${{ env.password }}" "-plugins.clean.mode=default" -schemas="dbo" -baselineOnMigrate="true" -licenseKey="${{ env.FLYWAY_LICENSE_KEY }}" -locations="filesystem:${{ GITHUB.WORKSPACE }}\migrations" info migrate info -url="${{ env.JDBC }}" -cleanDisabled="false"
```

5. Ideally we should be able to re-run this by clicking "Actions > Main-Workflow > Run Workflow (do not rerun job as it will use old YML).
   If we see a successful run congratulations, if not try again! Remember we can always revert to our stating point which we know works!

19. Re-Create Baseline Script

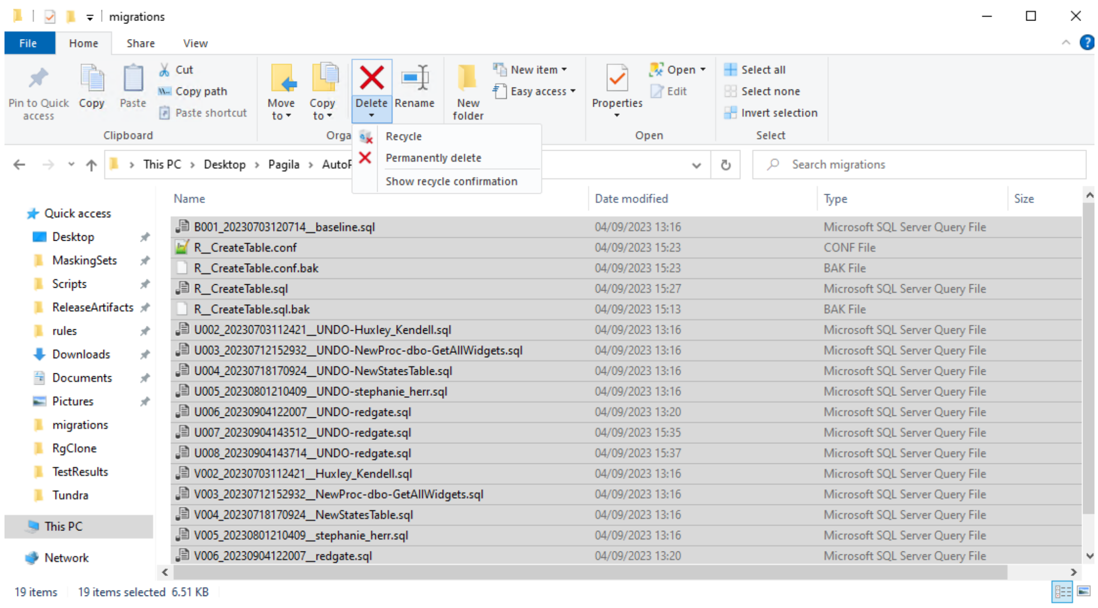
- Baselining can be very hard, *if* we do not understand what it is doing. Baselining is a powerful tool for Flyway, and allows us to designate a starting point for Flyway to work from, when working with an existing database, we recommend creating a baseline script that reflects the state of your objects and reference data that have already been deployed to production so that you can generate new migrations from this point. A baseline script is also necessary to spin up new environments for testing purposes or when deploying to new client sites.



(Everything Red already existed, and is therefore in the baseline script, green did not. So, it is not included!)

- Let's try and generate a new baseline script off of production, as we have deployed new objects there. This is also handy to know in case we want to ever update our baseline, potentially due to having made too many migration scripts and / or wanting a fresh start!

The best way to do this is to navigate to our Flyway project, and delete all of our scripts! (they will still be safe in the repository if we need to undo our changes!)



Name	Date modified	Type	Size
B001_20230703120714__baseline.sql	04/09/2023 13:16	Microsoft SQL Server Query File	
R__CreateTable.conf	04/09/2023 15:23	CONF File	
R__CreateTable.conf.bak	04/09/2023 15:23	BAK File	
R__CreateTable.sql	04/09/2023 15:27	Microsoft SQL Server Query File	
R__CreateTable.sql.bak	04/09/2023 15:13	BAK File	
U002_20230703112421__UNDO-Huxley_Kendell.sql	04/09/2023 13:16	Microsoft SQL Server Query File	
U003_20230712152932__UNDO-NewProc-dbo-GetAllWidgets.sql	04/09/2023 13:16	Microsoft SQL Server Query File	
U004_20230718170924__UNDO-NewStatesTable.sql	04/09/2023 13:16	Microsoft SQL Server Query File	
U005_20230801210409__UNDO-stephanie_herr.sql	04/09/2023 13:16	Microsoft SQL Server Query File	
U006_20230904122007__UNDO-redgate.sql	04/09/2023 13:20	Microsoft SQL Server Query File	
U007_20230904143512__UNDO-redgate.sql	04/09/2023 15:35	Microsoft SQL Server Query File	
U008_20230904143714__UNDO-redgate.sql	04/09/2023 15:37	Microsoft SQL Server Query File	
V002_20230703112421__Huxley_Kendell.sql	04/09/2023 13:16	Microsoft SQL Server Query File	
V003_20230712152932__NewProc-dbo-GetAllWidgets.sql	04/09/2023 13:16	Microsoft SQL Server Query File	
V004_20230718170924__NewStatesTable.sql	04/09/2023 13:16	Microsoft SQL Server Query File	
V005_20230801210409__stephanie_herr.sql	04/09/2023 13:16	Microsoft SQL Server Query File	
V006_20230904122007__redgate.sql	04/09/2023 13:20	Microsoft SQL Server Query File	

19 items   19 items selected 6.51 KB

- Once we have deleted the scripts, Flyway will recognise this and once we try and generate a new script will prompt us to first generate a baseline!

Additional Learning:

- Flyway, Redgate university website:

https://www.red-gate.com/hub/university/courses/flyway

- Getting Started:

https://flywaydb.org/documentation/getstarted/

- Flyway Desktop:

https://documentation.red-gate.com/fd/database-development-using-flyway-desktop-138346961.html

- CLI / API:

https://documentation.red-gate.com/fd/flyway-cli-and-api-183306238.html

- CI / CD:

https://flywaydb.org/hub