

# Project Tundra: Quick Start Guide

Huxley Kendell, Solution Engineering – [Huxley.Kendell@Red-Gate.com](mailto:Huxley.Kendell@Red-Gate.com)

This is the QuickStart guide for my personal Project Tundra. Please note that it is currently under development, and while it is fully functional and tested, any changes to the code or the addition of variable groups may cause issues that need to be fixed. Therefore, it's important to test thoroughly and reach out for support if needed. This guide has been designed and used on SQL Server for Azure DevOps. If you wish to use a different CI/CD platform, you can follow the steps below but use that platform's technique for supplying variable groups. If you want to use this on a different DBMS, the concepts, YAML, and CI/CD implementation will still work, but you will need to regenerate the Tundra Flyway project, especially the migration scripts, for the equivalent of that DBMS.

Inside this repository, you will find two folders:

## Tundra:

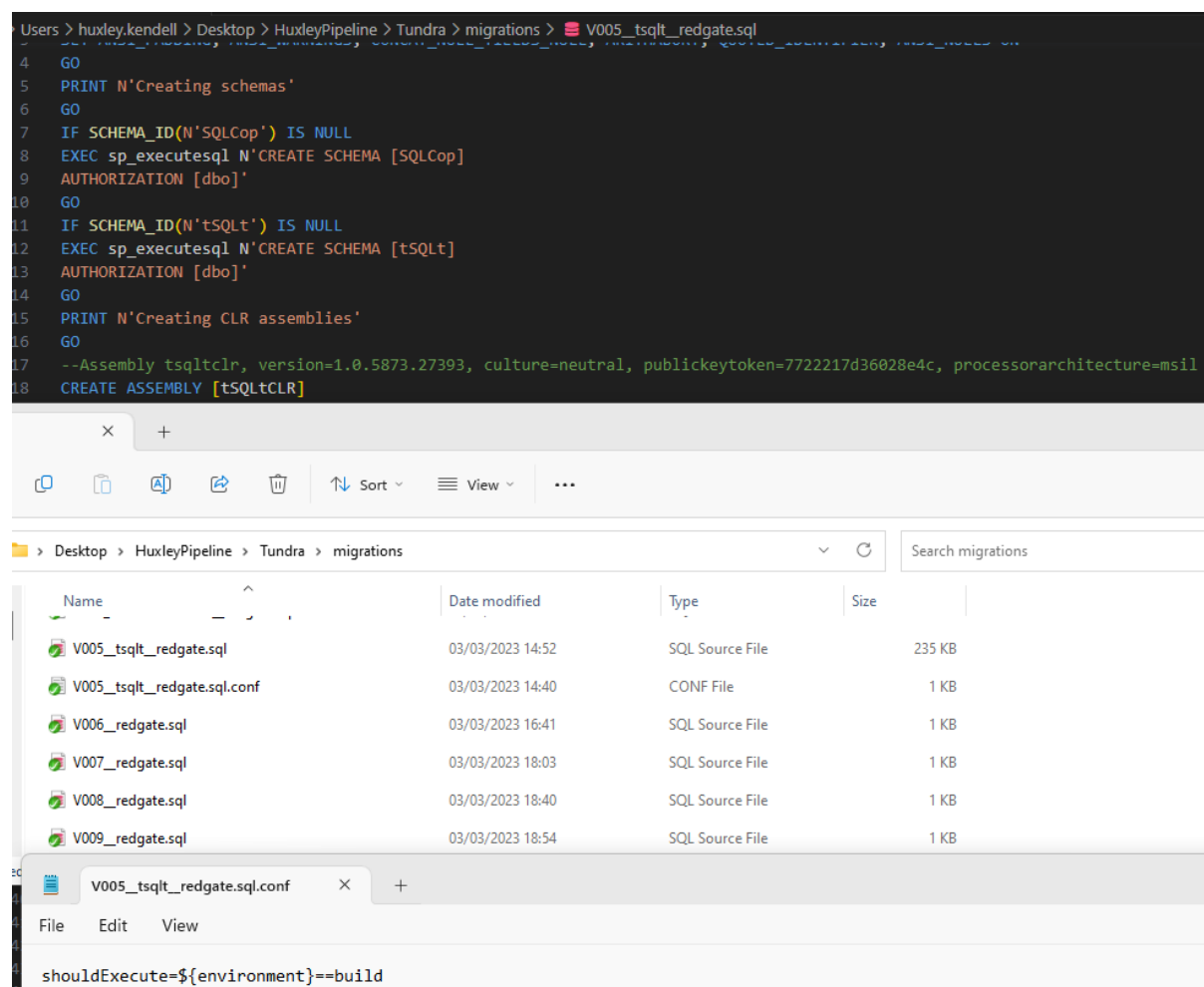
This is the Flyway project I use to test and develop this project. Although not necessary for the pipeline and YAML, it does contain certain repeatable migration scripts. If you are looking to reverse engineer, please refer to the V5 migration script, as we use it to deploy our tSQLt objects on our Build server only. I recommend using this folder as it allows you to test Flyway at its highest level without the need for extensive configuration before testing.

This folder also includes the necessary YAML file for the main pipeline, 'tSQLYAML.yml'. It does not include any hard-coded variables but relies on a set of Variable groups, which will be shown at the bottom of the screen. It is important to supply these Variable groups or their equivalents on your CI/CD platform; otherwise, Flyway will not have references to downstream environments, etc.

Name	Date modified	Type	Size
migrations	08/03/2023 14:40	File folder	
schema-model	08/03/2023 14:40	File folder	
TestResults	08/03/2023 14:40	File folder	
.gitignore	03/03/2023 10:58	Git Ignore Source File	1 KB
afterMigrate_tSQLt.ps1	03/03/2023 14:26	Windows PowerShell Sc...	1 KB
Filter.scpf	03/03/2023 10:58	SCPF File	5 KB
flyway.conf	03/03/2023 14:42	CONF File	23 KB
flyway.conf.bak	03/03/2023 11:04	BAK File	23 KB
flyway-dev.json	03/03/2023 17:25	Windows Media Player	2 KB
flyway-dev.user.json	03/03/2023 11:00	Windows Media Player	1 KB
tSQLYAML.yml	03/03/2023 18:41	Yaml Source File	3 KB

Looking deeper inside this folder, specifically into the migrations folder, you can find all the migration scripts. These scripts define our database structure and what needs to be deployed. However, they are not essential for the project to function, except for one script.

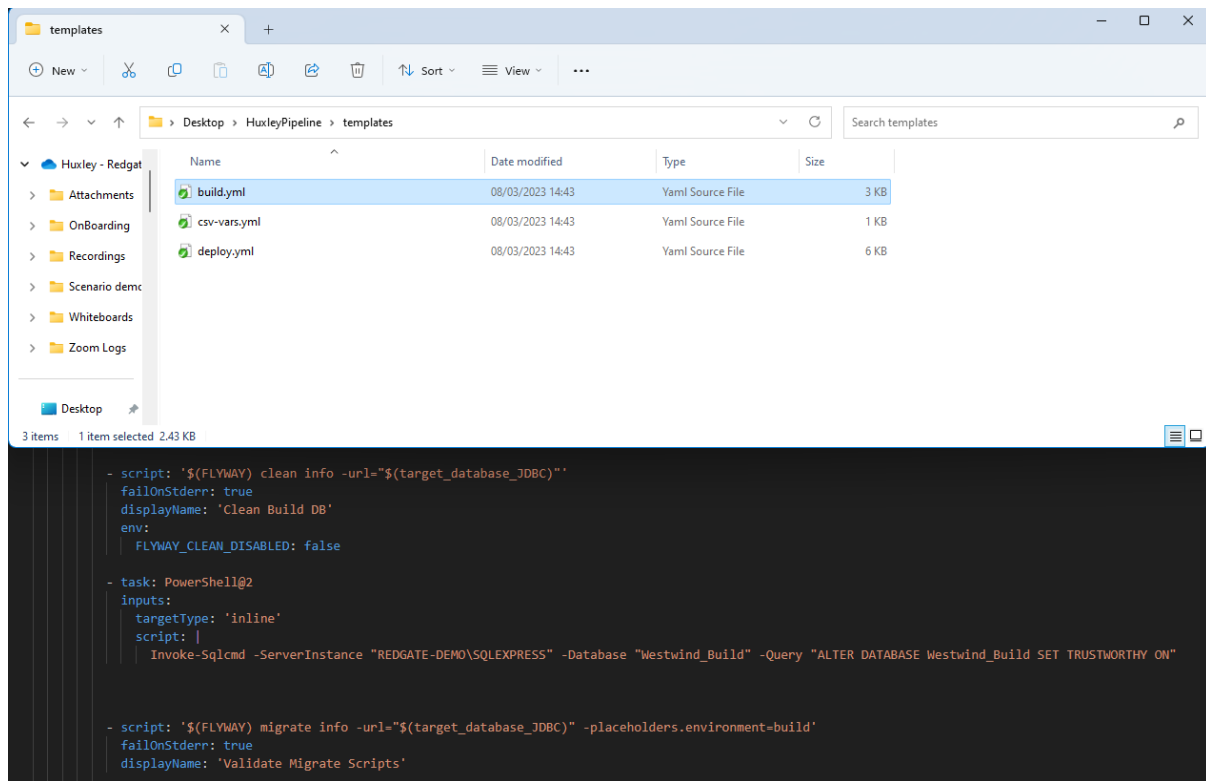
In this project, we focus on reliable deployments using various techniques, including unit testing. To perform unit testing, we need to have prerequisite objects, such as the tSQL framework and the COP tests we want to run. However, we do not want these objects to flow into our live environments for obvious reasons. Flyway allows us to have a single script dedicated to creating these objects, which only runs on our 'Build' environment. If you want to replicate this project, understanding how this file works is crucial for the pipeline to function properly.



## Templates:

This folder demonstrates the scalability and ease of access provided by Flyway when used with a CI/CD pipeline. The templates folder contains the YAML files for each stage of my pipeline, defining the structure and behavior before being supplied with more specific parameters in the main YAML file. These files are referenced by the parent YAML file in the Tundra folder, allowing us to set a template for our Build and Live Deployments. This saves us from typing the structure and behavior repeatedly for each deployment, making it highly scalable in terms of databases to deploy to or agents to run the deployments.

The Templates folder also contains a 'csv-vars.yml' file, which is used to declare additional variables. Feel free to check it out, but it should not require any changes.



### Set Up:

1. Firstly, fork this repository and clone all the files onto your local machine. Then, add both folders as separate repositories in the same Azure DevOps project. Do not add both folders to one repository in the CI/CD pipeline, as it will crash the pipeline.

### Create a repository

Repository type



Git

Repository name \*

Tundra

☒ Add a README

Add a .gitignore: None

Your repository will be initialized with a main branch.

2. Create a new pipeline using Azure Repos Git, select the Tundra repository, and use the 'tSQLYAML.yml' file as the existing pipeline file.

The first run of this pipeline may not work, but it will guide you in the direction of the missing variable groups. Refer to the images below, which show all the variable groups used. Add these variable groups and change the variables to match your local copy:

- **Build Credentials:** This variable group specifies the location of a database that can be used as a non-live sandbox for Flyway to clean and set up.

Variable group

Save

Clone

Security

Help

### Properties

Variable group name

build\_credentials\_variable\_group

Description

☒ Allow access to all pipelines

☐ Link secrets from an Azure key vault as variables ⓘ

### Variables

Name ↑	Value	🔒
databaseName	Westwind_Build	
password	DatabaseDevOps!	
target_database_JDBC	jdbc:sqlserver://localhost:databaseName=Westwind_Build;encrypt=true...	
userName	Redgate	

- **Flyway Vars:** This variable group includes the variables for your Flyway account and project. Make sure to supply a Flyway CLI key. Reach out to Redgate or myself if you do not have a trial and are interested.

Library > flyway\_vars

Variable group

Save

Clone

Security

Help

### Properties

Variable group name

flyway\_vars

Description

☒ Allow access to all pipelines

☐ Link secrets from an Azure key vault as variables ⓘ

### Variables

Name ↑	Value	🔒
AGENT_POOL	default	
BASELINE_VERSION	001.20230303110381	
FIRST_UNDO_SCRIPT	003.20230303115557	
FLYWAY_LICENSE_KEY	FL01970B114299DA8740C788EDEF20F313DD5668224A2ADB3E5F12AB...	
RELEASE_BRANCH	master	

- At least one agent needs to be configured to run. The easiest way to do this is to follow the guide provided by your chosen CI/CD platform for creating an agent and installing it locally on your machine or somewhere Flyway is installed.

Library >  redgate\_global\_vars

Variable group |  Save  Clone  Security  Help

## Properties


Variable group name

redgate\_global\_vars


Description

- ☒ Allow access to all pipelines
- ☐ Link secrets from an Azure key vault as variables ⓘ

## Variables

Name ↑	Value	
AGENT_POOL	default	
FLYWAY_LICENSE_KEY	FL01970B114299DA8740C788EDEF20F313DD5668224A2ADB3E5F12AB...	

- Details of a production database that can be deployed to, as well as a corresponding check database for checks. These databases can be configured and set up as empty if needed.
- Sensitive Variables, such as passwords can be marked as secure, showing why certain ones below appear as starred out.

Library >  westwind\_prod\_credentials

Variable group |  Save  Clone  Security  Help

## Properties


Variable group name

westwind\_prod\_credentials


Description





- ☒ Allow access to all pipelines
- ☐ Link secrets from an Azure key vault as variables ⓘ

## Variables

Name ↑	Value	
check_JDBC	jdbc:sqlserver://localhost;databaseName=Westwind_Check_Prod;encrypt=...	
check_password	*****	
check_userName	sa	
databaseName	Westwind	
password	*****	
target_database_JDBC	jdbc:sqlserver://localhost;databaseName=Westwind;encrypt=true;integ...	
userName	sa	

- Similarly, configure testing deployments and corresponding empty databases.

Library >  westwind\_test\_credentials

Variable group |  Save |  Clone |  Security |  Help

Properties

Variable group name




westwind\_test\_credentials

Description

☒ Allow access to all pipelines

☐ Link secrets from an Azure key vault as variables ⓘ

Variables

Name ↑	Value	
check_JDBC	jdbc:sqlserver://localhost;databaseName=Westwind_Check;encrypt=tru...	
check_password	*****	
check_userName	sa	
databaseName	Westwind_Test	
password	*****	
target_database_JDBC	 jdbc:sqlserver://localhost;databaseName=Westwind_Test;encrypt=true;i...	
userName	sa	