Automated deployments

This page provides an overview of automating deployments using the PowerShell cmdlets.

SQL Change Automation PowerShell contains a set of cmdlets specifically designed for deployment. These offer a number of advantages:

- They create **deployment resources** that allow you to review the changes that will happen to the target database if you proceed with the deployment.
- They run pre- and post-deployment checks to make sure your target schema hasn't changed unexpectedly.

New-DatabaseReleaseArtifact

Once you have created a database (see Package and publish), you can use this as the source schema:

Example

```
 $production = New-DatabaseConnection -ServerInstance "prod01\sql2014" -Database "AdventureWorksProduction" -
Username "sa" -Password "p@ssw0rd"
 $dbRelease = New-DatabaseReleaseArtifact -Source "C:\packages\MyDatabase.1.0.0.nupkg" -Target $production
```

After running the cmdlet, you can either use:

- **`Export-DatabaseReleaseArtifact`** - to review the deployment resources, including the update script
- **`Use-DatabaseReleaseArtifact`** - to run the update script against the target database

Export-DatabaseReleaseArtifact

Use this cmdlet to export the output of the `New-DatabaseReleaseArtifact` cmdlet to disk, so you can review the deployment resources.

Example

```
Export-DatabaseReleaseArtifact $dbRelease -Path "C:\SQLChangeAutomationArtifacts\"
```

After running the cmdlet, open *C:\SQLChangeAutomationArtifacts* to see the database deployment resources. These include:

- an **Update.sql** file - the SQL script that will update the target database schema to match the source schema
- a **Reports** folder that lets you review a summary of changes between the two databases and check warnings
- a **States** folder that contains scripts folder representations of both the source and target schemas

Use-DatabaseReleaseArtifact

This cmdlet completes the deployment set up by `New-DatabaseReleaseArtifact`. It does this by running the update script in the deployment resources against the target database.

Example

```
Use-DatabaseReleaseArtifact $dbRelease -DeployTo $production
```

As part of the deployment, the `Use-DatabaseReleaseArtifact` cmdlet runs two checks:

- **pre-deployment check**
  Before running the update script, this checks the schema of the database you're deploying to hasn't changed since the creation of the database deployment resources. If the schema has changed, the update script will fail.
- **post-deployment check**
  After running the update script, this checks the schema of the database you're deploying to matches the source database. If they don't match, the cmdlet will give an error warning.

An example deployment script

Let's now combine the cmdlets we've looked at on this page to make a PowerShell script for a full deployment script:

Example

```
# Create a database Release
$production = New-DatabaseConnection -ServerInstance "prod01\sql2014" -Database "AdventureWorksProduction" -
Username "sa" -Password "p@ssw0rd"
$dbRelease = New-DatabaseReleaseArtifact -Source "C:\packages\MyDatabase.1.0.0.nupkg" -Target $production
Export-DatabaseReleaseArtifact $dbRelease -Path "C:\SQLChangeAutomationArtifacts\"

# Manually review the deployment resources

# Use the database Release to deploy to Production
Use-DatabaseReleaseArtifact $dbRelease -DeployTo $production
```

It's good practice to run lines 1-4 and 8-9 of the script above in separate stages, so you can review the deployment resources before deciding whether to continue with the release.

For example, if you're using Octopus Deploy, you can run the script from two separate steps in your deployment project, and add an intermediate *Manual intervention* step that pauses the deployment process.