How to set up SQL Source Control

The short guide for evaluators

SQL
SOURCE
CONTROL

redgate
ingeniously simple

Content

Introduction

SQL Source Control links your database to your source control system, so you can share changes to schemas and reference data, check a history of database development, and roll back anything you don't want to keep, without leaving SQL Server Management Studio.

If you don't have the installer for SQL Source Control, you can download it from our website.

This 19-page guide shows you how to set up and use SQL Source Control with your database and **Team Foundation Server**, **Subversion**, or **Git**, as well as how to get started if you don't have a source control system.

It covers how to:

1. Link your database to source control
2. Make an initial commit
3. Commit a change
4. Get a change
5. Set SQL Source Control options and share them with your team

If you already own SQL Source Control and need help getting set up, visit our documentation site to learn how to link your database to your source control system and how to use SQL Source Control.

Team Foundation Server & Subversion setup

The next few pages show you how to set up SQL Source Control with Team Foundation Server (TFS) or Subversion (SVN).

If you're using Git, you can safely skip ahead to page 9, where the setup instructions for Git begin.

If you don't have a source control system, skip to page 12, where the setup instructions for our evaluation repository begin.

For more information on setting up SQL Source Control with SVN or TFS, visit:

- Linking to source control with TFS
- Linking to source control with SVN

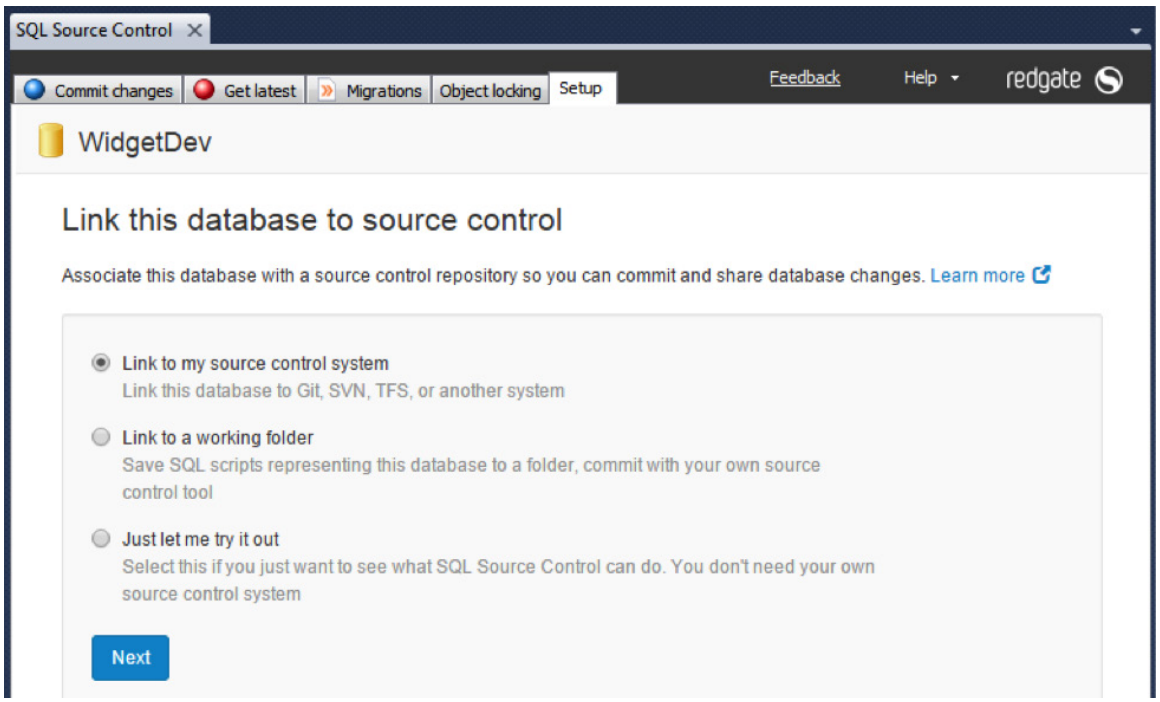
Linking a database to source control

To get started, you need to link your database to your source control repository.

If you don't want to try SQL Source Control with one of your own databases yet, you can create a dummy database full of objects using this script.

In SQL Server Management Studio's Object Explorer, select the database you want to link to source control.

In SQL Source Control, on the **Setup** tab, make sure **Link to my source control system** is selected and click **Next**:



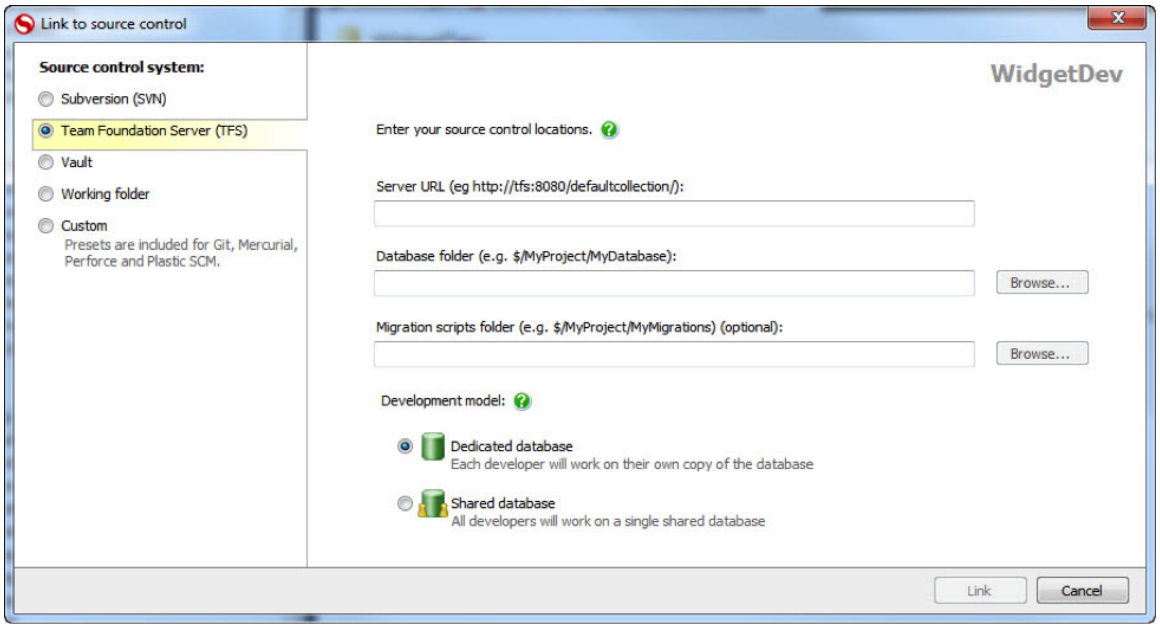
*Linking your database to source control in the **Setup** tab*

On the **Choose your source control system** page, select **Team Foundation Server (TFS)** or **Subversion (SVN)** and click **Next**.

The **Link to source control** dialog box opens.

Specifying URLs for Team Foundation Server

If you're using Subversion, skip ahead to the next page.



*The **Link to source control** dialog box, set up for Team Foundation Server*

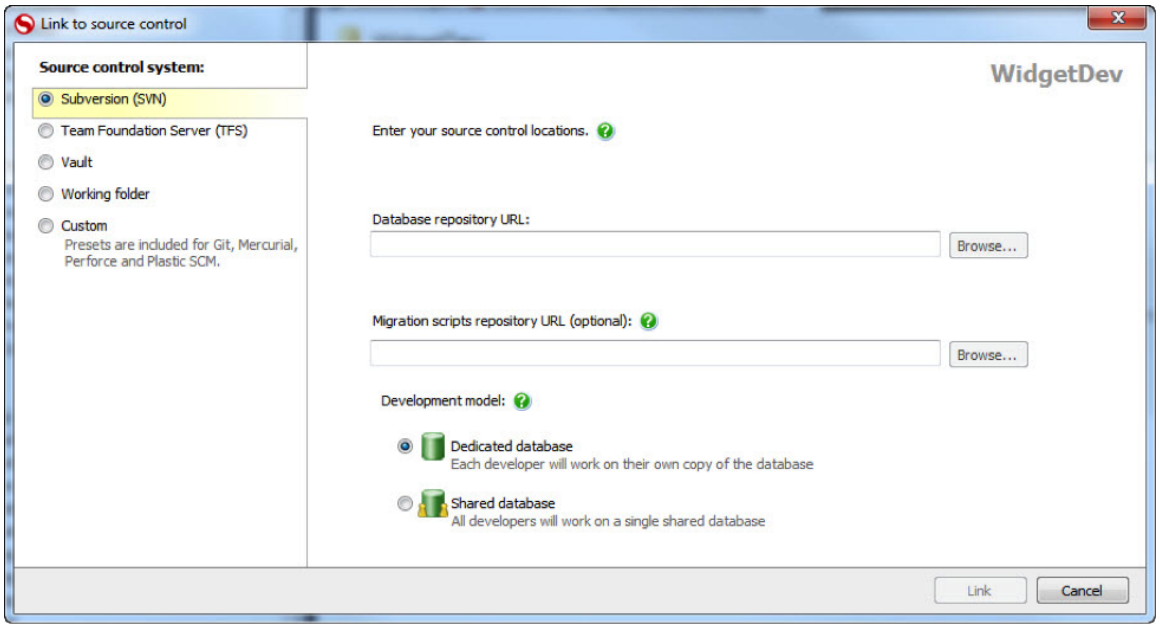
For TFS, specify the URL of your Team Foundation Server in the **Server URL** field. If you use a non-standard port, specify this in the URL (eg *http://myurl:8080/tfs*).

In the **Database folder** field, enter the folder in TFS where SQL Source Control will store SQL scripts.

If you're the first person to link the database to source control, specify an empty folder. If someone has already linked this database to source control, specify the folder they used.

For instructions on how to add migrations scripts, skip to the end of the next page.

Specifying a URL for Subversion



*The **Link to source control** dialog box, set up for Subversion*

For SVN, use the **Database repository URL** field to specify a folder in your Subversion repository where SQL Source Control will save SQL scripts. For example: *http://Subversion.Example.com/Databases/AdventureWorks*

If you're the first person to link the database to source control, specify an empty folder. If someone has already linked this database to source control, specify the folder they used.

How to add migration scripts

If you want to use migration scripts – customizable change scripts that SQL Compare uses during deployment – then under **Migration scripts repository URL**, specify an existing, empty folder in the repository.

The folder can't be in the database folder. For example, you can use *Repository folder\Migration scripts*, but not *Repository folder\Database folder\Migration scripts*.

Shared or Dedicated database development model?

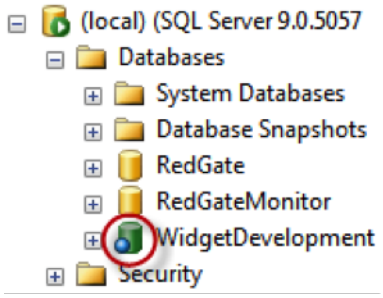
When you link a database, you tell SQL Source Control which development model your team uses. SQL Source Control changes its behavior slightly to suit the model your team uses.

If each developer has their own database, leave the **Development model** set to **Dedicated database**, the default setting. If you're linking to a database that will be used by multiple developers, make sure **Shared database** is selected.

Ready to go?

Once you've provided repository details, click **Link**.

The database icon in the Object Explorer changes to indicate that your database is linked to source control:



A database that's been linked to source control in the Object Explorer

To see how to start committing your changes, move to page 13.

Git setup

The next few pages show you how to set up SQL Source Control with Git.

If you've already set up SQL Source Control with TFS or SVN, you can safely skip ahead to page 13, where instructions for committing changes begin.

If you don't have a source control system, skip to page 12, where the setup instructions for our evaluation repository begin.

For more information on setting up SQL Source Control with Git, visit Linking to source control with Git.

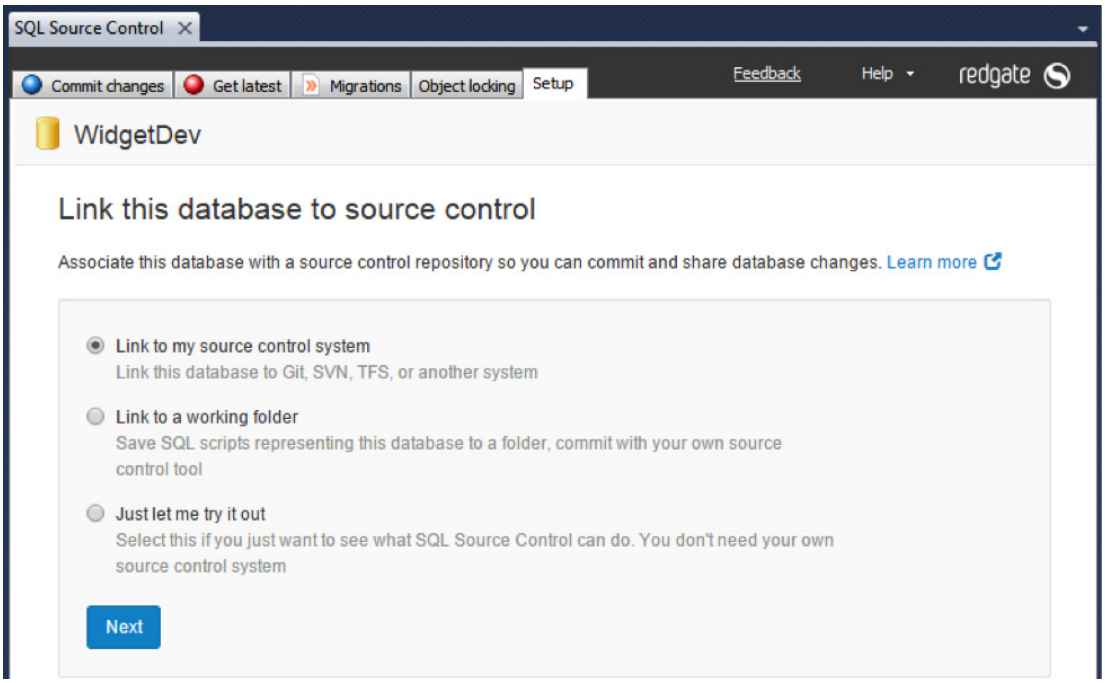
Linking a database to source control

To get started, you need to link your database to your source control repository.

If you don't want to try SQL Source Control with one of your own databases yet, you can create a dummy database full of objects using this script.

In SQL Server Management Studio's Object Explorer, select the database you want to link to source control.

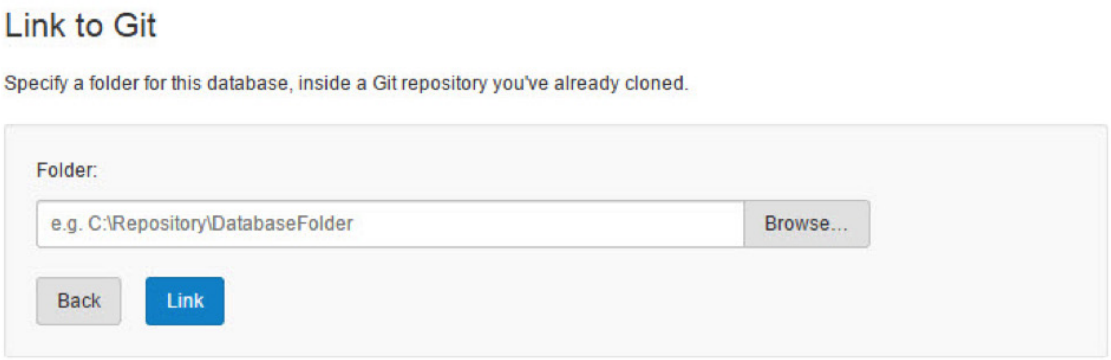
In SQL Source Control, on the **Setup** tab, make sure **Link to my source control system** is selected and click **Next**:



*Linking your database to source control in the **Setup** tab*

On the **Choose your source control system** page, select **Git** and click **Next**.

The **Link to Git** page opens:



*The **Link to Git** page in SQL Source Control*

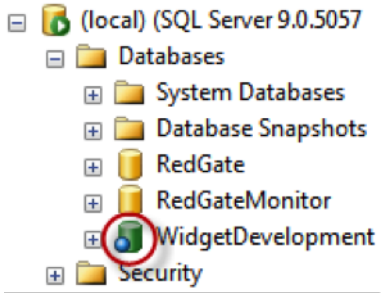
In the **Folder** field, specify a folder in an existing local Git repository where SQL Source Control will save SQL scripts, for example: *C:\Repository\DatabaseFolder*

If you're the first person to link the database to source control, specify an empty folder. If someone has already linked this database to source control, specify the folder they used.

Ready to go?

Once you've provided repository details, click **Link**.

The database icon in the Object Explorer changes to indicate that your database is linked to source control:



A database that's been linked to source control in the Object Explorer

To see how to start committing your changes, move to page 13.

Setup without a source control system

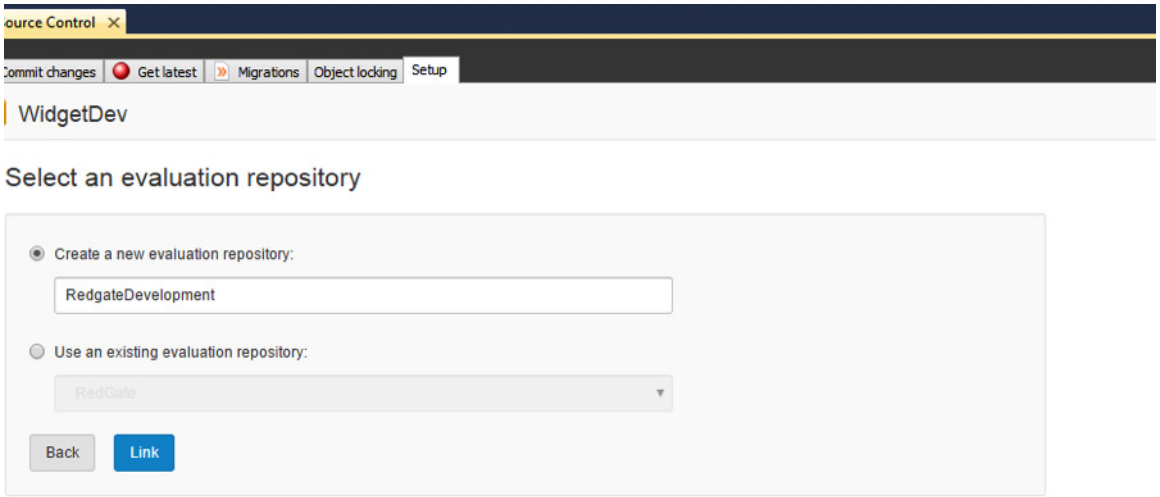
If you don't have a source control system to link to, we recommend you use our evaluation repository.

In SQL Server Management Studio's Object Explorer, select the database you want to link to source control.

In the SQL Source Control **Setup** tab, select **Just let me try it out** and click **Next**.

This opens the **Link to an evaluation repository** page. Click **Next**.

In the **Select an open evaluation repository** page, make sure **Create a new evaluation repository** is selected and click **Link**.



Using the built-in evaluation repository to try out SQL Source Control

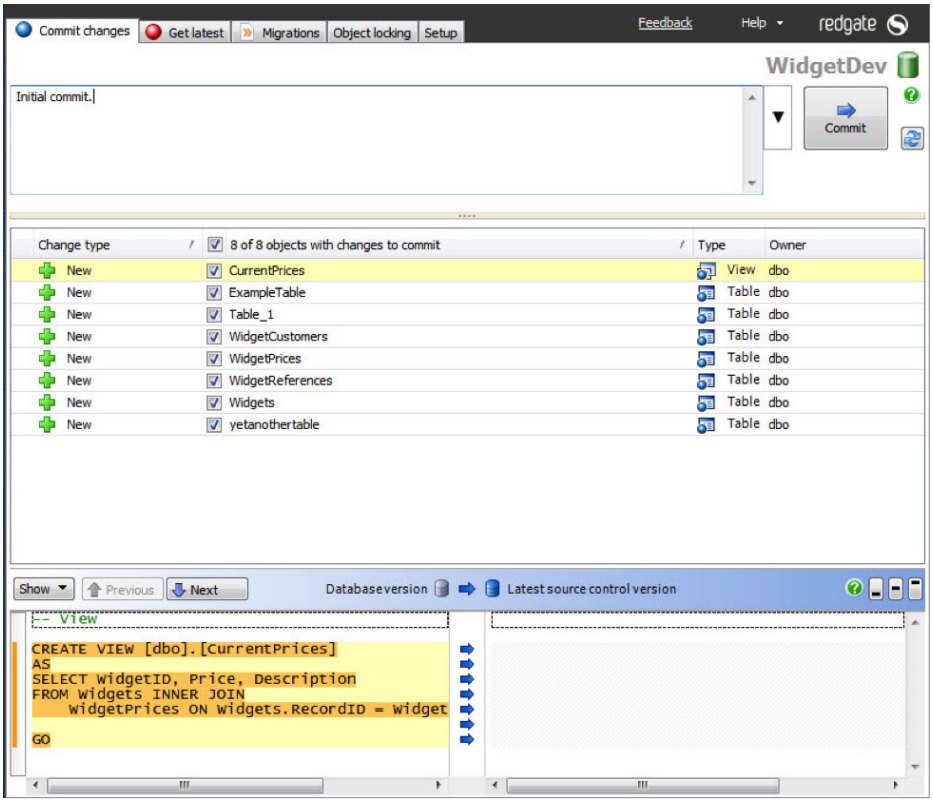
This creates a local Subversion repository on your computer. You don't have to set anything up yourself. We don't recommend you use it as a long-term source control system, however, as the evaluation repository doesn't work well for keeping backups or sharing changes. When you finish evaluating, you should link to your own source control system.

For more information, see Linking to the evaluation repository in the SQL Source Control documentation.

Making your first commit

Once you've linked your database to source control, you're ready to make your first commit (also known as the initial commit). This gets a copy of your database into source control.

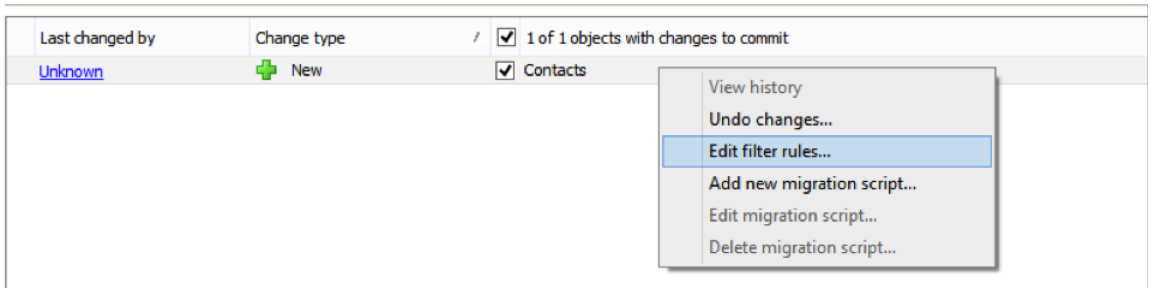
To make the initial commit, go to the **Commit changes** tab. This tab lists database changes that haven't been committed to source control yet:



*Committing database objects to source control for the first time in the **Commit changes** tab*

You may have some types of object you never want to commit to source control, for example, objects of a certain type (eg functions or views), or objects that belong to a particular schema. You can create a filter to exclude these objects so they never appear in the **Commit changes** or **Get latest** tabs.

To create a filter, right-click on the grid and select **Edit filter rules**.



*Editing filter rules in the **Commit changes** tab*

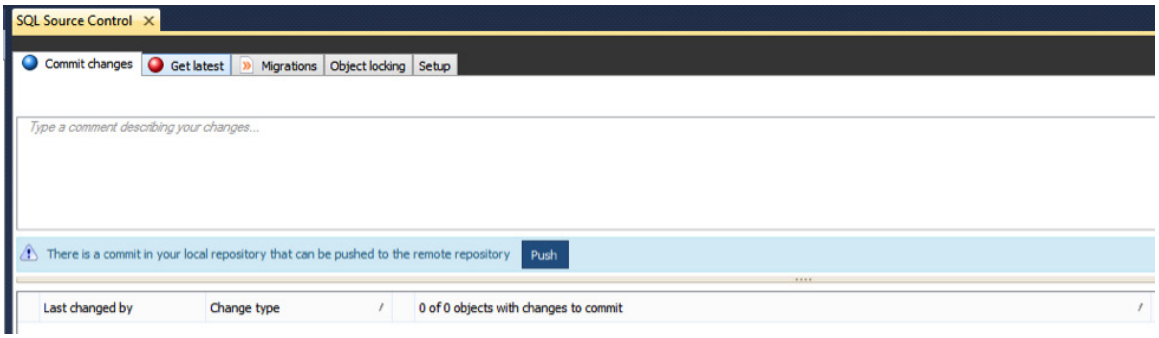
For more information, see Using filters to exclude objects in the SQL Source Control documentation.

To commit your database objects to source control, make sure all the objects are selected, write a commit message (eg “Initial commit”), and click **Commit**.

Pushing changes to a Git remote repository

To get changes into your remote repository, you need to push after you’ve committed them.

To do this, find the blue banner above the commit grid and click **Push**. This pushes all commits from the local repository to the remote repository.

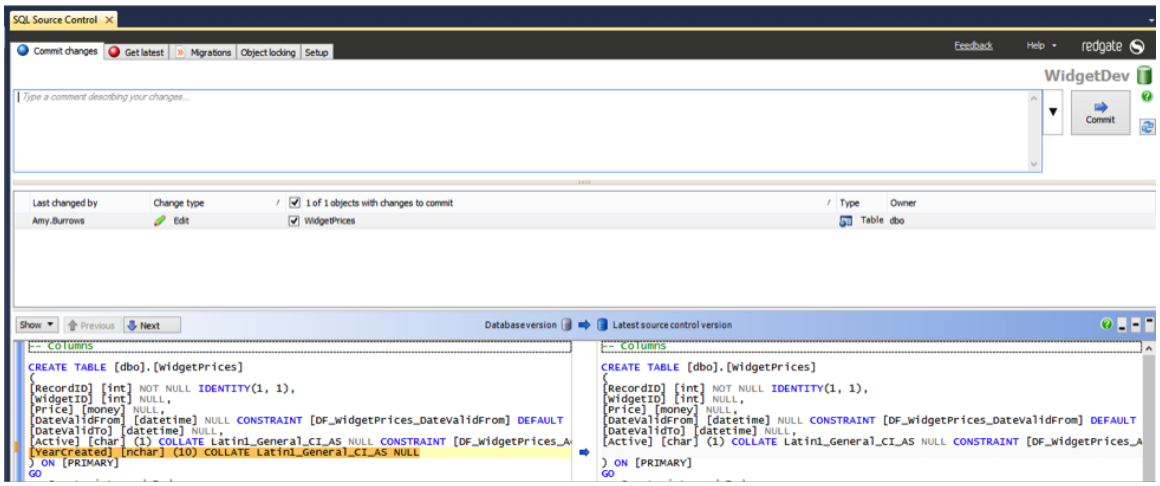


*Making a Git push in the **Commit changes** tab*

Committing a change

Try making a simple change to the database, such as adding a column to a table.

After you make the change, go to the **Commit changes** tab. The change appears in the list of changes to commit:



See which changes you haven't yet committed to source control, with line-by-line diffs, in the ***Commit changes*** *tab*

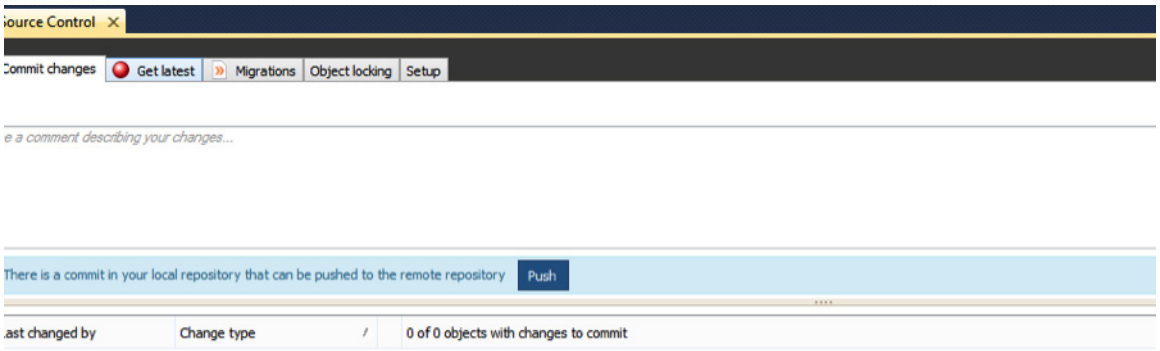
In the lower pane, you can see the differences between the SQL creation scripts for the new object and for the object in source control.

To commit the change, write a commit message and click **Commit**. Commit messages are useful when getting changes or reviewing history, so your team can quickly understand what's in each change and why it was made.

Pushing changes to a Git remote repository

To get changes into your remote repository, you need to push after you've committed them.

To do this, find the blue banner above the commit grid and click **Push**. This pushes all commits from the local repository to the remote repository.

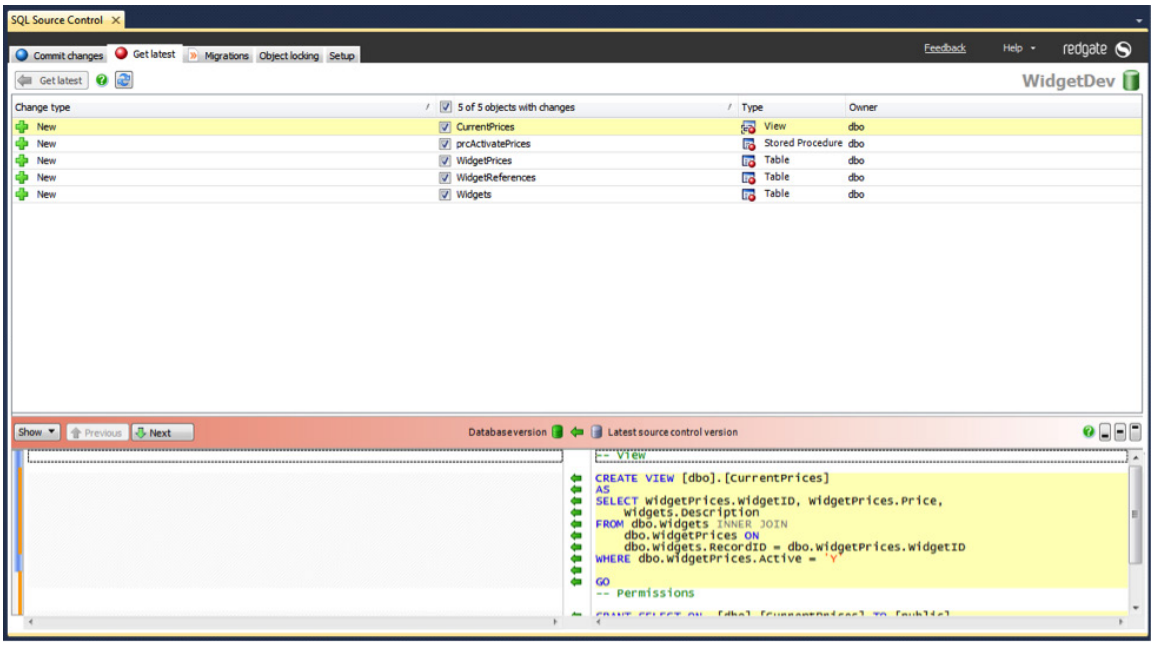


*Making a Git push in the **Commit changes** tab*

Getting a change

Let's check out the database from source control. Link an empty database to the same repository you linked your database to.

Afterwards, when you go to the **Get latest** tab, you'll see your database changes waiting to be retrieved from source control:



*Getting database changes in the **Get latest** tab*

When you click **Get latest**, the new database will be updated to match the other database.

Pulling changes from a Git remote repository

To get the latest changes from source control, you need to pull changes from the remote repository.

1. On the **Get latest** tab, click **Pull from remote repository**. Any changes from the remote repository are pulled to your local repository.
2. Click **Apply changes to database**. The changes from the local repository are deployed to the database.

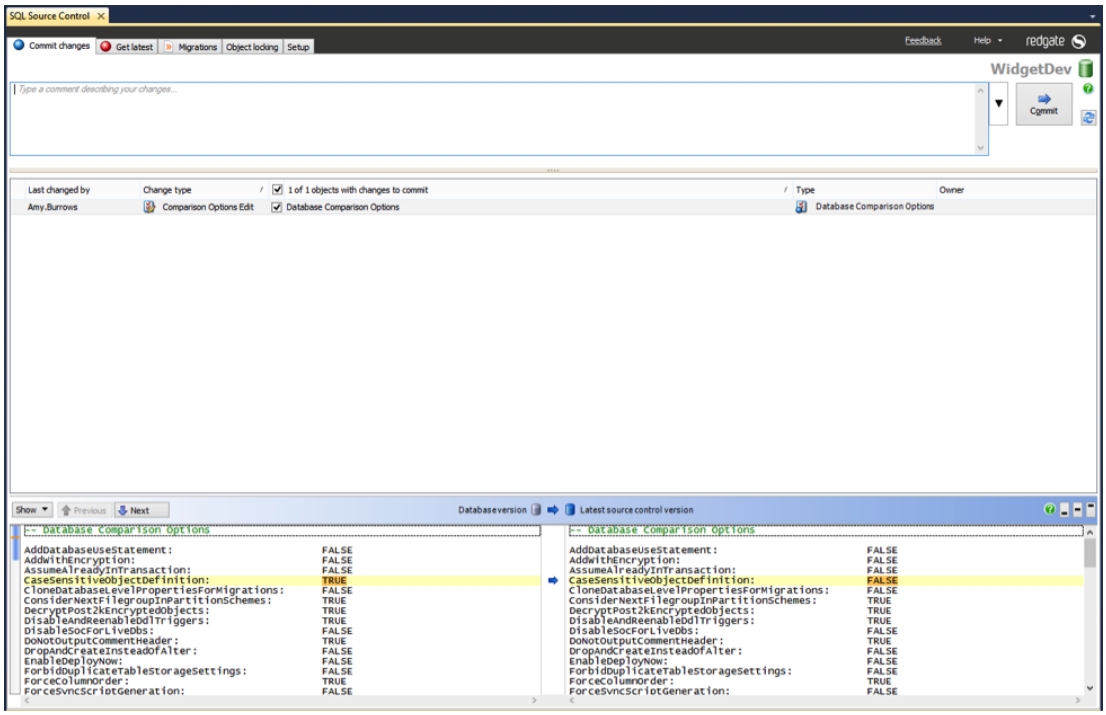
Setting SQL Source Control options

You can configure SQL Source Control to suit your development process:

- Using filters to exclude objects
- Change how SQL Source Control deploys database changes
- Link static data to source control

Sharing option changes with your team

When you edit an option, it appears in the **Commit changes** tab as a change to commit:



*Sharing a SQL Source Control option with your team, in the **Commit changes** tab*

After you commit, other people on your team can get your change to the options in the **Get latest** tab. This makes sure everyone works with the same options.

Next steps

Nice work – now you've covered all the basics of SQL Source Control.
So what next?

If your team is on board with the idea of database source control, then it's just a case of setting everyone up. But most teams want to do some extra investigation to be sure they're making a good investment.

To help your team decide, we've put together a guide full of free articles, eBooks, and videos, so you can:

- Show your DBA how source control keeps production data safe
- Help anyone who hasn't used source control before to learn the fundamentals
- Explain the advantages that database source control brings to a business:
    - Time and money savings
    - Strong regulatory compliance
    - Fast, reliable delivery of new products to customers

Download How to make the case for SQL Source Control in 5 steps

Or, we can show your team exactly how it would work in your environment.

Our technical team is more than happy to talk about your needs, and we can arrange custom demos for you on change management, productivity, or anything else you're interested in.

To speak to someone on our team email
SQLSourceControlQuestions@red-gate.com