

# Contents

[Azure Data Studio documentation](#)

[Download Azure Data Studio](#)

[Release notes](#)

[Overview](#)

[Azure Data Studio](#)

[Quickstarts](#)

[Connect & query SQL Server](#)

[Connect & query Azure SQL Database](#)

[Connect & query Azure Synapse Analytics](#)

[Connect & query Azure Cosmos DB API for MongoDB](#)

[Connect & query PostgreSQL](#)

[Connect & explore Oracle](#)

[Tutorials](#)

[T-SQL Editor](#)

[Sample insight: slow queries](#)

[Sample insight: table space usage](#)

[Build a custom insight](#)

[Back up & restore](#)

[Concepts](#)

[Extensions](#)

[Extensions overview](#)

[Azure Arc extension](#)

[Azure SQL migration extension](#)

[Azure Monitor Logs extension](#)

[Azure Cosmos DB API for MongoDB extension](#)

[Database Migration Assessment for Oracle extension](#)

[Database Schema Conversion Toolkit](#)

[Database Schema Conversion Toolkit](#)

[Oracle to Microsoft SQL](#)

[Overview](#)

[Install and remove extension](#)

[Convert Oracle database objects to Microsoft SQL](#)

[Configure conversion](#)

[Conversion Messages](#)

[DSCT00000](#)

[DSCT00001](#)

[DSCT00002](#)

[DSCT00003](#)

[DSCT00004](#)

[DSCT00005](#)

[DSCT01000](#)

[DSCT01001](#)

[DSCT01002](#)

[DSCT01100](#)

[DSCT01101](#)

[DSCT01102](#)

[DSCT01103](#)

[Data Virtualization extension](#)

[Extension for Oracle](#)

[Kusto extension](#)

[Machine Learning extension](#)

[Machine Learning extension overview](#)

[Manage packages in database](#)

[Make predictions](#)

[Import and view models](#)

[Managed Instance extension](#)

[PostgreSQL extension](#)

[PowerShell extension](#)

[SandDance extension](#)

[Schema Compare extension](#)

[SQL Database Projects extension](#)

- [SQL Database Projects extension overview](#)
- [Getting Started with the SQL Database Projects extension](#)
- [Build and Publish a Project](#)
- [Build a Project from the Command Line](#)
- [Use SDK-style projects](#)
- [SQL Server Agent](#)
- [SQL Server Central Management Servers extension](#)
- [SQL Server dacpac extension](#)
- [SQL Server Import](#)
- [SQL Server Profiler](#)
- [Extension Authoring](#)
  - [Extension authoring](#)
  - [Extensibility overview](#)
  - [Extensibility APIs](#)
  - [Keymap extension](#)
  - [Dashboard extension](#)
  - [Notebook extension](#)
  - [Jupyter Book extension](#)
  - [Wizard extension](#)
- [Integrated terminal \(PowerShell, Bash\)](#)
- [Preview features](#)
- [Server groups](#)
- [Source control](#)
- [Table designer](#)
- [How-to guides](#)
  - [Notebooks](#)
    - [Notebooks overview](#)
    - [Notebooks views](#)
  - [Log Analytics Kernel](#)
  - [Kusto Kernel](#)
  - [Kqlmagic](#)
  - [Python Kernel](#)

- [Python Kernel Offline](#)
- [SQL Kernel](#)
- [Parameterization](#)
  - [Papermill Parameterization](#)
  - [URI Parameterization](#)
  - [Run with Parameters](#)
- [Deployment](#)
  - [Deploy Azure SQL Database](#)
  - [Deploy Azure SQL Edge](#)
  - [Deploy SQL Server on Azure Virtual Machines](#)
  - [Deploy SQL Server in a container](#)
- [Dashboards & insight widgets](#)
- [Code snippets](#)
- [Explore Azure SQL resources](#)
- [Keyboard shortcuts](#)
- [Usage data collection](#)
- [Windows authentication \(Kerberos\)](#)
- [Workspace & user settings](#)
- [References](#)
  - [SQL tools](#)
  - [Azure Developer tools](#)
- [Resources](#)
  - [FAQ](#)
  - [Troubleshooting](#)
  - [Azure Data Studio GitHub repo >](#)
  - [Report issues & suggestions >](#)
  - [SQL Server documentation >](#)



# Download and install Azure Data Studio

9/26/2022 • 8 minutes to read • [Edit Online](#)

Azure Data Studio is a cross-platform database tool for data professionals who use on-premises and cloud data platforms on Windows, macOS, and Linux.

Azure Data Studio offers a modern editor experience with IntelliSense, code snippets, source control integration, and an integrated terminal. It's engineered with the data platform user in mind, with the built-in charting of query result sets and customizable dashboards.

Use Azure Data Studio to query, design, and manage your databases and data warehouses wherever they are, on your local computer or in the cloud.

For more information about Azure Data Studio, visit [What is Azure Data Studio?](#).

## Download Azure Data Studio

Azure Data Studio 1.39.1 is the latest general availability (GA) version.

- Release number: 1.39.1
- Release date: August 30, 2022

PLATFORM	DOWNLOAD
Windows	<a href="#">User installer</a> (recommended) <a href="#">System installer</a> <a href="#">.zip file</a>
macOS	<a href="#">.zip file</a>
Linux	<a href="#">.deb file</a> <a href="#">.rpm file</a> <a href="#">.tar.gz file</a>

### NOTE

Azure Data Studio currently does not support the ARM architecture.

If you have comments or suggestions or want to report a problem with downloading Azure Data Studio, submit an issue to our team on the [Azure Data Studio feedback page](#).

## Install Azure Data Studio

### Windows installation

## IMPORTANT

Beginning with SQL Server Management Studio (SSMS) 18.7, Azure Data Studio is automatically installed alongside SSMS. Users of SQL Server Management Studio are now able to benefit from the innovations and features in Azure Data Studio. Azure Data Studio is a cross-platform and open-source desktop tool for your environments, whether in the cloud, on-premises, or hybrid.

To learn more about Azure Data Studio, check out [What is Azure Data Studio](#) or the [FAQ](#).

This release of Azure Data Studio includes a standard Windows installer experience and a .zip file.

We recommend the *user installer*, which simplifies installations and updates and doesn't require Administrator privileges. (It doesn't require Administrator privileges because the location is under your user Local AppData (LOCALAPPDATA) folder.) The user installer also provides a smoother background update experience. For more information, see [User setup for Windows](#).

### User installer (recommended)

1. Download and run the [Azure Data Studio user installer for Windows](#).
2. Start the Azure Data Studio app.

### System installer

1. Download and run the [Azure Data Studio system installer for Windows](#).
2. Start the Azure Data Studio app.

### .zip file

1. Download the [Azure Data Studio .zip file for Windows](#).
2. Go to the downloaded file and extract it.
3. Run `\azuredatastudio-windows\azuredatastudio.exe`.

### Unattended installation for Windows

You can also install Azure Data Studio by using a command prompt script.

For Windows, install Azure Data Studio in the background without prompts by doing the following:

1. Open the command prompt window with elevated permissions.
2. Run the following command:

```
<path where the azuredatastudio-windows-user-setup-x.xx.x.exe file is located> /VERYSILENT  
/MERGETASKS=!runcode
```

Example:

```
%systemdrive%\azuredatastudio-windows-user-setup-1.24.0.exe /VERYSILENT /MERGETASKS=!runcode
```

## NOTE

The following example also works with the system installer file.

```
<path where the azuredatascudio-windows-setup-x.xx.x.exe file is located> /VERYSILENT  
/MERGETASKS=!runcode>
```

In the preceding code, you can also pass */SILENT* instead of */VERYSILENT* to see the setup user interface.

3. If you've run the commands successfully, you can see Azure Data Studio installed.

## macOS installation

1. Download [Azure Data Studio for macOS](#).
2. To expand the contents of the .zip file, double-click it.
3. To make Azure Data Studio available in Launchpad, drag the *Azure Data Studio.app* file to the *Applications* folder.

## Linux installation

### Install with a .deb file

1. Download Azure Data Studio for Linux by using the [.deb](#) file.
2. To extract the .deb file, open a new terminal window, and then run the following commands:

```
cd ~  
sudo dpkg -i ./Downloads/azuredatascudio-linux-<version string>.deb
```

3. To start Azure Data Studio, run this command:

```
azuredatascudio
```

### NOTE

You might have missing dependencies. To install them, run the following command:

```
sudo apt-get install libunwind8
```

### Install with an .rpm file

1. Download Azure Data Studio for Linux by using the [.rpm](#) file.
2. To extract the file, open a new terminal window, and then run the following commands:

```
cd ~  
yum install ./Downloads/azuredatascudio-linux-<version string>.rpm
```

3. To start Azure Data Studio, run this command:

```
azuredatascudio
```

#### NOTE

You might have missing dependencies. To install them, run the following command:

```
yum install libXScrnSaver
```

#### Install with a .tar.gz file

1. Download Azure Data Studio for Linux by using the [.tar.gz](#) file.
2. To extract the file, open a new terminal window, and then run the following commands:

```
cd ~  
cp ~/Downloads/azuredatastudio-linux-<version string>.tar.gz ~  
tar -xvf ~/azuredatastudio-linux-<version string>.tar.gz  
echo 'export PATH="$PATH:~/azuredatastudio-linux-x64"' >> ~/.bashrc  
source ~/.bashrc
```

3. To start Azure Data Studio, run this command:

```
azuredatastudio
```

#### NOTE

You might have missing dependencies. To install them, run the following command:

```
sudo apt-get install libxss1 libgconf-2-4 libunwind8
```

#### Windows Subsystem for Linux

1. Install Azure Data Studio for Windows. Then, use the `azuredatastudio` command in a Windows Subsystem for Linux (WSL) terminal just as you would in a standard command prompt. By default, the application is stored in your *AppData* folder.
2. Start Azure Data Studio from the WSL command prompt. When you're using the default Windows installation, start the application by running the following command:

```
'/mnt/c/Users/<your user name>/AppData/Local/Programs/Azure Data Studio/azuredatastudio.exe'
```

## What's new with Azure Data Studio

For details about the latest release of Azure Data Studio, see [Release notes for Azure Data Studio](#).

## Download the GA release of Azure Data Studio

We recommend that you [download the general availability \(GA\) release of Azure Data Studio](#).

## Download the insiders build of Azure Data Studio

As an alternative, if you want to try out the beta features and send feedback, you can [download the insiders build of Azure Data Studio](#).

# Supported operating systems

Azure Data Studio runs on Windows, macOS, and Linux. The following versions are supported:

- Windows: 7 (with SP1), 8, 8.1, 10, 11
- Windows Server: 2016, 2019, 2022
- macOS: 10.12+, 11, 12
- Linux: Debian 9+, RHEL 7+, Ubuntu 18.04+

## NOTE

Incremental versions within a major operating system release may no longer be in support by the operating system. Consult the documentation for your operating system to ensure you have received applicable updates.

# Supported SQL Server versions

Azure Data Studio supports connecting to the following versions of SQL Server:

- Azure SQL Database
- Azure SQL Managed Instance
- SQL Server 2022 (16.x) Preview
- SQL Server 2019 (15.x)
- SQL Server 2017 (14.x)
- SQL Server 2016 (13.x)

# System requirements

Requirement Level	CPU Cores	RAM Memory
Recommended	4	8 GB
Minimum	2	4 GB

# Check for updates

To check for the latest updates, on the left pane, select **Manage** (gear icon), and then select **Check for Updates**.

To apply environment updates offline, [install the latest version](#) directly over your previously installed version. You don't need to uninstall earlier versions of Azure Data Studio. If an earlier version is present, the installer automatically updates to the latest version.

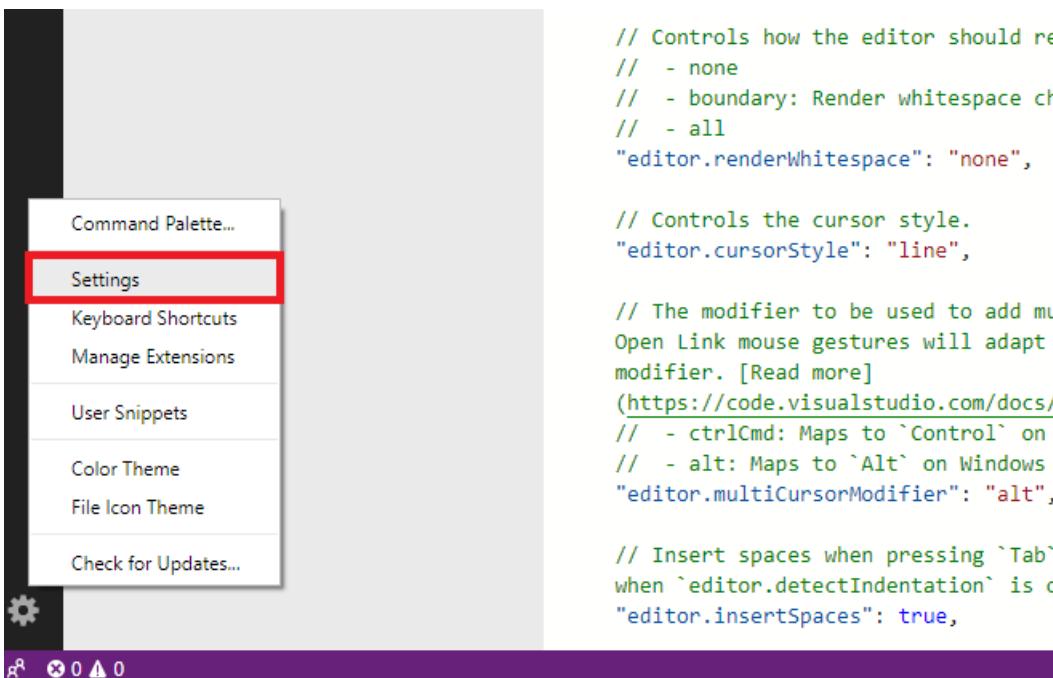
# Move user settings

If you're updating SQL Operations Studio to Azure Data Studio and want to keep your settings, keyboard shortcuts, or code snippets, do the following:

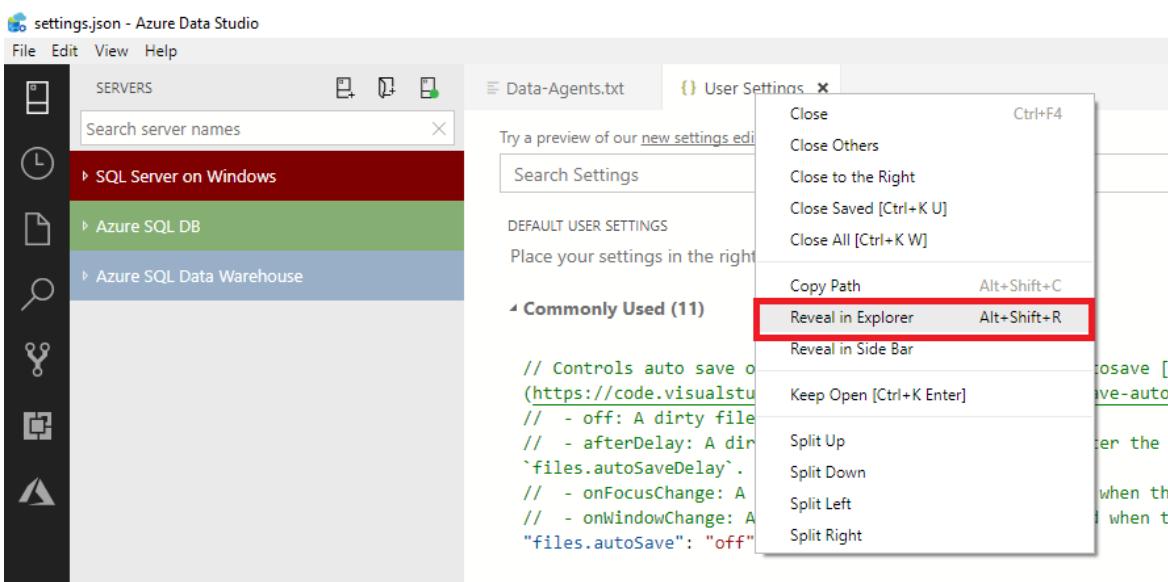
## NOTE

If you've already installed Azure Data Studio or you've never installed or customized SQL Operations Studio, you can ignore this section.

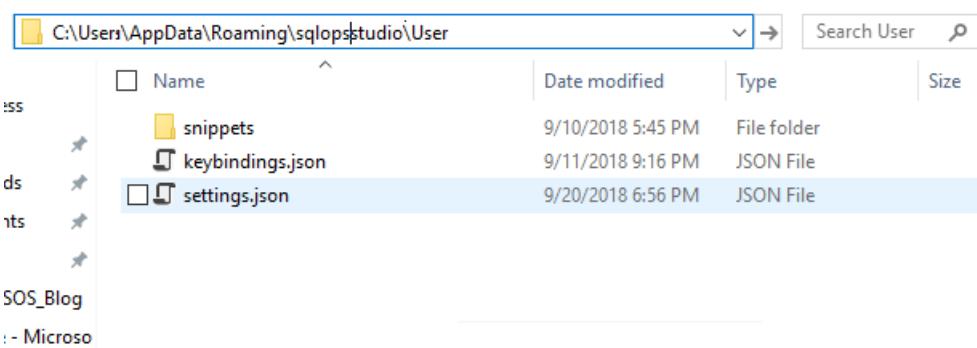
1. On the left pane, select **Manage** (gear icon) and then select **Settings**.



- At the top, right-click the User Settings tab, and then select Reveal in Explorer.



- Copy all files in this folder and save them in an easy-to-find location on your local drive, such as your *Documents* folder.



- In your updated version of Azure Data Studio, follow steps 1 and 2 and then, for step 3, paste the contents you saved into the folder. You can also manually copy over the settings, key bindings, or snippets in their respective locations.
- If you're overriding your current installation, before you do so, delete the old installation directory to avoid errors connecting to your Azure account for the resource explorer.

# Uninstall Azure Data Studio from Windows

If you installed Azure Data Studio by using the Windows installer, uninstall it just as you would any Windows application.

If you installed Azure Data Studio with a .zip file or other archive, delete that file.

# Uninstall Azure Data Studio from macOS

You can [uninstall apps](#) from the internet or disc on Mac by doing the following:

1. Select the **Finder** icon in the Dock, and then select **Applications** in the **Finder** sidebar.
2. Do one of the following:
  - If an app is in a folder, open the app's folder to check for an uninstaller. Double-click **Uninstall [App]** or **[App] Uninstaller**, and then follow the onscreen instructions.
  - If an app isn't in a folder or doesn't have an uninstaller, drag the app from the *Applications* folder to the Trash (at the end of the Dock).

To uninstall apps you've downloaded from the App Store, use Launchpad.

# Uninstall Azure Data Studio from Linux

## In Ubuntu/Debian

You can uninstall Azure Data Studio under Debian or Ubuntu Linux.

To list installed software type, run the following commands:

```
dpkg --list  
dpkg --list | less  
dpkg --list | grep azuredatastudio
```

To delete the software, run the following commands:

```
sudo apt-get remove azuredatastudio  
sudo apt-get remove apache
```

## In RedHat

Use the rpm or yum command to delete Azure Data Studio.

To list the installed software type, run the following commands:

```
rpm -qa | less  
rpm -qa azuredatastudio  
yum list | less  
yum list azuredatastudio
```

To get information about the azuredatastudio package, run the following commands:

```
rpm -qa azuredatastudio  
yum list azuredatastudio
```

To delete a package called azuredatastudio, run the following commands:

```
rpm -e azuredatastudio  
yum remove azuredatastudio
```

## Next steps

To learn more about Azure Data Studio, see the following resources:

- [Azure Data Studio release notes](#)
- [What is Azure Data Studio?](#)
- [SQL tools](#)
- [Azure Data Architecture Guide](#)
- [SQL Server Blog](#)

### ⓘ Get help for SQL tools

- [All the ways to get help](#)
- [Submit an Azure Data Studio Git issue](#)
- [Contribute to Azure Data Studio](#)
- [SQL Client Tools Forum](#)
- [SQL Server Data Tools - MSDN forum](#)
- [Support options for business users](#)

### ✎ Contribute to SQL documentation

Did you know that you can edit SQL content yourself? If you do so, not only do you help improve our documentation, but you also get credited as a contributor to the page.

For more information, see [How to contribute to SQL Server documentation](#)

[Microsoft Privacy Statement](#) and [Enable or disable usage data collection for Azure Data Studio](#).

# Release notes for Azure Data Studio

9/26/2022 • 41 minutes to read • [Edit Online](#)

This article provides details about updates, improvements, and bug fixes for the current and previous versions of Azure Data Studio.

## Current Azure Data Studio release

↓ [Download and install the latest release!](#)

### August 2022 (hotfix)

Azure Data Studio 1.39.1 is the latest general availability (GA) release.

- Release number: 1.39.1
- Release date: August 30, 2022

#### Bug fixes in 1.39.1

NEW ITEM	DETAILS
Object Explorer	Fixed bug that caused Database Trees in server connections to not expand in the Object Explorer.

## Azure Data Studio feedback

You can reference [Azure Data Studio feedback](#) for other known issues and to provide feedback to the product team.

## Previous Azure Data Studio releases and updates

AZURE DATA STUDIO RELEASE	BUILD NUMBER	RELEASE DATE	HOTFIX
August 2022	1.39.0	August 24, 2022	N/A
July 2022	1.38.0	July 27, 2022	N/A
June 2022	1.37.0	June 15, 2022	N/A
April 2022	1.36.0	April 20, 2022	hotfix
February 2022	1.35.0	February 24, 2022	hotfix
December 2021	1.34.0	December 15, 2021	N/A
October 2021	1.33.0	October 27, 2021	N/A
August 2021	1.32.0	August 18, 2021	N/A
July 2021	1.31.0	July 21, 2021	N/A

AZURE DATA STUDIO RELEASE	BUILD NUMBER	RELEASE DATE	HOTFIX
June 2021	1.30.0	June 17, 2021	N/A
May 2021	1.29.0	May 19, 2021	N/A
April 2021	1.28.0	April 15, 2021	N/A
March 2021	1.27.0	March 17, 2021	N/A
February 2021	1.26.0	February 18, 2021	N/A
December 2020	1.25.0	December 9, 2020	hotfix
November 2020	1.24.0	November 12, 2020	N/A
October 2020	1.23.0	October 14, 2020	N/A
September 2020	1.22.0	September 22, 2020	hotfix
August 2020	1.21.0	August 12, 2020	N/A
July 2020	1.20.0	July 15, 2020	hotfix
June 2020	1.19.0	June 15, 2020	N/A
May 2020	1.18.0	May 20, 2020	hotfix
April 2020	1.17.0	April 27, 2020	hotfix
March 2020	1.16.0	March 18, 2020	N/A
February 2020	1.15.0	February 13, 2020	hotfix
December 2019	1.14.0	December 19, 2019	hotfix
November 2019	1.13.0	November 4, 2019	hotfix
October 2019	1.12.0	October 2, 2019	hotfix 1 hotfix 2
September 2019	1.11.0	September 10, 2019	N/A
August 2019	1.10.0	August 15, 2019	N/A
July 2019	1.9.0	July 11, 2019	N/A
June 2019	1.8.0	June 6, 2019	N/A
May 2019	1.7.0	May 8, 2019	N/A
April 2019	1.6.0	April 18, 2019	N/A

AZURE DATA STUDIO RELEASE	BUILD NUMBER	RELEASE DATE	HOTFIX
March 2019	1.5.1	March 18, 2019	hotfix
February 2019	1.4.5	February 13, 2019	N/A
January 2019	1.3.8	January 09, 2019	hotfix

Download the previous release of Azure Data Studio.

#### NOTE

All previous versions of Azure Data Studio are not supported.

## August 2022

- Release number: 1.39.0
- Release date: August 24, 2022

### What's new in 1.39.0

NEW ITEM	DETAILS
Deployment Wizard	Azure Data Studio now supports SQL Server 2022 (16.x) Preview in the Deployment Wizard for both local and container installation.
Object Explorer	Added Ledger icons and scripting support to Object Explorer for Ledger objects.
Dashboard	Added hexadecimal values to support color detection.
Query Plan Viewer	Added the ability to copy text from cells in the Properties Pane of a query plan.
Query Plan Viewer	Introduced a "find node" option in plan comparison to search for nodes in either the original or added plan.
Table Designer	Now supports the ability to add included columns to a nonclustered index, and the ability to create filtered indexes.
SQL Projects	Publish options were added to the Publish Dialog.
Query History Extension	Added double-click support for query history to either open the query or immediately execute it, based on user configuration.

### Bug fixes in 1.39.0

NEW ITEM	DETAILS
Dashboard	Fixed an accessibility issue that prevented users from being able to access tooltip information using the keyboard.

NEW ITEM	DETAILS
Voiceover	Fixed a bug that caused voiceover errors across the Dashboard, SQL Projects, SQL Import Wizard, and SQL Migration extensions.
Schema Compare	Fixed a bug that caused the UI to jump back to the top of the options list after selecting/deselecting any option.
Schema Compare	Fixed a bug involving Schema Compare (.SCMP) file incompatibility with Database Project information causing errors when reading and using information stored in this file type.
Object Explorer	Fixed a bug that caused menu items in Object Explorer not to show up for non-English languages.
Table Designer	Fixed a bug that caused the History Table name not to be consistent with the current table name when working with System-Versioned Tables.
Table Designer	Fixed a bug in the Primary Key settings that caused the "Allow Nulls" option to be checked, but disabled, preventing users from changing this option.
Query Editor	Fixed a bug that prevented the SQLCMD in T-SQL from working correctly, giving false errors when running scripts in Azure Data Studio.
Query Editor	Fixed a bug that caused user-specified zoom settings to reset to default when selecting JSON values after query that returned JSON dataset was ran.
SQL Projects	Fixed a bug that caused the "Generate Script" command to not work correctly when targeting a new Azure SQL Database.
Notebooks	Fixed a bug that caused pasted images to disappear from editor after going out of edit mode.
Notebooks	Fixed a bug that caused a console error message to appear after opening a markdown file.
Notebooks	Fixed a bug that prevented markdown cell toolbar shortcuts from working after creating a new split view cell.
Notebooks	Fixed a bug that caused text cells to be erroneously created in split view mode when the notebook default text edit mode was set to "Markdown".

## July 2022

### What's new in 1.38.0

NEW ITEM	DETAILS
----------	---------

NEW ITEM	DETAILS
VS Code merges to 1.62	This release includes updates to VS Code from the three previous VS Code releases. Read <a href="#">their release notes</a> to learn more.
Table Designer	New column added to Table Designer for easier access to additional actions specific to individual rows.
Query Plan Viewer	The Top Operations pane view now includes clickable links to operations in each of its rows to show the runtime statistics which can be used to evaluate estimated and actual rows when analyzing a plan.
Query Plan Viewer	Improved UI on selected operation node in Execution Plan.
Query Plan Viewer	The keyboard command <b>CTRL + M</b> no longer executes queries. It now just enables or disables the actual execution plan creation when a query is executed.
Query Plan Viewer	Plan labels are now updated in the Properties window when plans are compared and the orientation is toggled from horizontal to vertical, and back.
Query Plan Viewer	Updates were made to the Command Palette. All execution plan commands are prefixed with "Execution Plan", so that they are easier to find and use.
Query Plan Viewer	A collapse/expand functionality is now available at the operator level to allow users to hide or display sections of the plan during analysis.
Query History	The Query History extension was refactored to be fully implemented in an extension. This makes the history view behave like all other extension views and also allows for searching and filtering in the view by selecting the view and typing in your search text.

## Bug fixes in 1.38.0

NEW ITEM	DETAILS
Table Designer	Error found in edit data tab when switching back to previously selected column when adding a new row. To fix this, editing the table is now disabled while new rows are being added and only reenabled afterwards.
Query Editor	Fixed coloring issues for new T-SQL functions in the Query Editor.
Query Plan Viewer	Fixed bug that caused custom zoom level spinner to allow values outside valid range.
Dashboard	Fixed issue that caused incorrect displaying of insight widgets on the dashboard.

NEW ITEM	DETAILS
Notebooks	Fixed issue where keyboard shortcuts and toolbar buttons were not working when first creating a Split View markdown cell.
Notebooks	Fixed issue where cell languages were not being set correctly when opening an ADS .NET Interactive notebook in VS Code.
Notebooks	Fixed issue where notebook was being opened as empty when exporting a SQL query as a notebook.
Notebooks	Disables install and uninstall buttons in Manage Packages dialog while a package is being installed or uninstalled.
Notebooks	Fixed issue where cell toolbar buttons were not refreshing when converting cell type.
Notebooks	Fixed issue where notebook was not opening if a cell contains an unsupported output type.
Schema Compare	Fixed issue where views and stored procedures were not correctly recognized by schema compare after applying changes.

## June 2022

### What's new in 1.37.0

NEW ITEM	DETAILS
Backup & Restore	Backup & Restore to URL is now available in preview for Azure SQL Managed Instances.
Table Designer	Added support for computed columns in Table Designer.
Table Designer	Can now specify where to add new columns and columns can now be re-arranged by mouse dragging.
Table Designer	Table Designer is now supported in the SQL Database Projects extension for editing tables in the SQL project.
Query Plan Viewer	Plan comparison is now available and includes visual indicators in the Properties pane for easier identification of differences.
Query Plan Viewer	Added a toolbar button to toggle the display for actual execution plans.
Query Plan Viewer	Larger query plans will now display additional precision for operator cost.
MongoDB Extension for Azure Cosmos DB (Preview)	This extension introduces support for access to Mongo resources for Cosmos DB.

### Bug fixes in 1.37.0

NEW ITEM	DETAILS
Table Designer	Fixed issue that caused app to not prompt user to save before closing.
Table Designer	Fixed issue that returned empty data set upon attempting to edit the first cell of a new row.
Table Designer	Improved resize to fit experience when zooming in on user interface as well as tab behavior issues.
Query Plan Viewer	Fixed bug that caused custom zoom level spinner to allow values outside valid range.
Schema Compare	Fixed issue with indexes not being added correctly when updating project from database.
Notebooks	Fixed inconsistencies with notebook cell behavior and toolbars.
Notebooks	Fixed issues with keyboard navigation.

## May 2022 (hotfix)

- Release number: 1.36.2
- Release date: May 20, 2022

### What's new in 1.36.2

NEW ITEM	DETAILS
Power BI	Introduced support for Power BI Datamart connectivity. Please see <a href="#">Announcing public preview of datamart in Power BI</a> .

### Bug fixes in 1.36.2

NEW ITEM	DETAILS
Query Plan Viewer	Fixed issue with execution plan zoom and operator icons.
Query Plan Viewer	Updated parallelism icon direction.

## April 2022 (hotfix)

- Release number: 1.36.1
- Release date: April 22, 2022

### Bug fixes in 1.36.1

NEW ITEM	DETAILS
Table Designer	Fix for timeout occurring when viewing table list

## April 2022

- Release number: 1.36.0
- Release date: April 20, 2022

## What's new in 1.36.0

NEW ITEM	DETAILS
Table Designer	Added support for System Versioning, Memory Optimized, and Graph Tables.
Query Plan Viewer	Added support icons and additional support for searching within plans. Added additional telemetry to Execution Plans. Provided support on plan toolbar to enable or disable tooltips. Added support for saving .sql plan files on Azure Data Studio
SQL Projects	Introduced new SQL Project format based on an SDK-Style project file
Azure SQL Migration Extension	Announcing General Availability of the Azure SQL Migration Extension
.NET Interactive Notebooks extension	This extension provides additional multi-language support to Jupyter Notebooks. Please reference <a href="#">.NET Notebooks in Visual Studio Code</a> for an introduction to using SQL and .NET interactive

## Bug fixes in 1.36.0

NEW ITEM	DETAILS
Table Designer	Added missing validation rules for primary key column specifications
Table Designer	Now able to add description to Primary Key ,Foreign Key, and check constraints
Table Designer	Fixed bug that prevents the primary key checkbox from being unchecked for unsupported primary key types
Query Plan Viewer	Added option to turn off tooltip in execution plan
Query Plan Viewer	Fixed display and sizing issues
Query Plan Viewer	Fixed latency issues while switching tabs when execution plan is shown
Query Editor	Fixed performance issues in Query Editor
Notebooks	Fixed keyboard navigation issues
Notebooks	Fixed .NET Interactive log errors on startup
Notebooks	Fixed inconsistencies with notebook URI handling

For a full list of bug fixes addressed for the April 2022 release, visit the [bugs and issues list on GitHub](#).

## Known issues in 1.36.0

For a list of the current known issues, visit the [issues list on GitHub](#).

## February 2022 (hotfix)

- Release number: 1.35.1
- Release date: March 17, 2022

#### Bug fixes in 1.35.1

NEW ITEM	DETAILS
Query Editor	Geometry Data Type Returned as Unknown Charset in Results Grid
Query Editor	Excel number format

### February 2022

- Release number: 1.35.0
- Release date: February 24, 2022

#### What's new in 1.35.0

NEW ITEM	DETAILS
Table Designer	Added functionality for creation and management of tables for SQL Servers. Built using DacFx framework
Query Plan Viewer	Added functionality for users to view a graphic view of estimated and actual query plans without need for an extension
Azure Arc Extension	Updated the Data Controller deployment wizard and the SQL Managed Instance - Azure Arc deployment wizard to reflect the deployment experience in Azure Portal

#### Bug fixes in 1.35.0

NEW ITEM	DETAILS
Azure Arc Extension	SQL Managed Instance-Azure Arc is now fixed for both indirect connectivity mode and direct connectivity mode
Notebooks	Support for keyboard navigation between cells to minimize mouse clicking

For a full list of bug fixes addressed for the February 2022 release, visit the [bugs and issues list on GitHub](#).

#### Known issues in 1.35.0

For a list of the current known issues, visit the [issues list on GitHub](#).

### December 2021

- Release number: 1.34.0
- Release date: December 15, 2021

#### What's new in 1.34.0

NEW ITEM	DETAILS
SQL Migration extension	Added 'Currently restoring backup file' in the migration progress details page of Azure SQL migration extension when backup files location is Azure Storage blob container

NEW ITEM	DETAILS
Notebooks	Added undo/redo support
SQL Database Projects extension	Support for project build with .NET 6 in SQL Database Projects extension
SQL Database Projects extension	Publish to container in SQL Database Projects extension
Extension update	<a href="#">SQL Database Projects</a>
Extension update	Lang packs
Extension update	Azure SQL Migration

#### Bug fixes in 1.34.0

NEW ITEM	DETAILS
SQL Migration	Fix for multiple database migrations when using network share as backup files location in Azure SQL migration extension
SQL Migration	Fix for multiple database migrations when using blob storage containers as backup files location in Azure SQL migration extension
SQL Migration	Fix to pre-populate target database names in the migration wizard in Azure SQL migration extension
Grid	Fix to column sorting in grids where the presence of null values could lead to unexpected results
Notebooks	Fix for Python upgrades when two or more notebooks were open

#### October 2021

- Release number: 1.33.0
- Release date: October 27, 2021

#### What's new in 1.33.0

NEW ITEM	DETAILS
Notebooks	Added Notebook Views support
Notebooks	Added split cell support
Notebooks	Added keyboard shortcuts for Markdown Toolbar cells
Notebooks	Large performance improvement for large notebooks <ul style="list-style-type: none"> <li>• Ctrl/Cmd + B = Bold Text</li> <li>• Ctrl/Cmd + I = Italicize Text</li> <li>• Ctrl/Cmd + U = Underline Text</li> <li>• Ctrl/Cmd + Shift + K = Add Code Block</li> <li>• Ctrl/Cmd + Shift + H = Highlight Text</li> </ul>

NEW ITEM	DETAILS
----------	---------

Notebooks	<p>Added Book improvements</p> <ul style="list-style-type: none"> <li>• Add a new section</li> <li>• Drag and Drop</li> </ul>
Extension update	<a href="#">Azure Monitor Logs</a>
Extension update	<a href="#">Schema Compare</a>
Extension update	<a href="#">SQL Database Projects</a>
Extension update	<a href="#">Machine Learning</a>
Extension update	<a href="#">Profiler</a>
Extension update	<a href="#">Import</a>
Extension update	<a href="#">Kusto</a>
Extension update	<a href="#">Dacpac</a>
Extension update	Lang packs
Extension update	<a href="#">Azure SQL Migration</a>
Extension update	<a href="#">CMS</a>
Extension update	Kusto

#### Bug fixes in 1.33.0

NEW ITEM	DETAILS
Notebook	Fixed Notebook linking
Notebook	Fixed horizontal scrollbar (when word wrap is off in MD Splitview / MD mode) in Notebooks
Notebook	Fixed vertical scrollbar for MD Splitview in Notebooks

For a full list of bug fixes addressed for the August 2021 release, visit the [bugs and issues list on GitHub](#).

#### Known issues in 1.33.0

For a list of the current known issues, visit the [issues list on GitHub](#).

#### August 2021

- Release number: 1.32.0

- Release date: August 18, 2021

#### **What's new in 1.32.0**

NEW ITEM	DETAILS
Notebooks	Large performance improvement for large notebooks
Extension (new)	<a href="#">Azure Monitor Logs</a>
Extension update	<a href="#">SchemaCompare</a>
Extension update	<a href="#">SQLDatabaseProjects</a>
Extension update	<a href="#">MachineLearning</a>
Extension update	<a href="#">Azure Arc</a>
Extension update	Lang packs

#### **Bug fixes in 1.32.0**

NEW ITEM	DETAILS
Database Projects	Fixed Seeing "Project already opened" message for every project I open or create
Schema Compare	Fixed Schema compare subsequent results after once are not loading in view
Lang packs	Fixed Localization for previously untranslated extensions.
Machine Learning	Fixed Machine Learning - View models dialog is broken

For a full list of bug fixes addressed for the August 2021 release, visit the [bugs and issues list on GitHub](#).

#### **Known issues in 1.32.0**

For a list of the current known issues, visit the [issues list on GitHub](#).

#### **July Hotfix 2021**

- Release number: 1.31.1
- Release date: July 29, 2021

#### **Bug fixes in 1.31.1**

NEW ITEM	DETAILS
Connections	Fixed Database connection toolbar missing for sql scripts
Connections	Fixed Connection dropped / is not maintained when saving / opening scripts
Connections	Script file opened from command line does not allow DB connection

For a full list of bug fixes addressed for the July Hotfix 2021 release, visit the [bugs and issues list on GitHub](#).

## Known issues in 1.31.1

For a list of the current known issues, visit the [issues list on GitHub](#).

## July 2021

- Release number: 1.31.0
- Release date: July 21, 2021

### What's new in 1.31.0

NEW ITEM	DETAILS
Notebooks	WYSIWYG link improvements
Extension update	<a href="#">SchemaCompare</a>
Extension update	<a href="#">SQLDatabaseProjects</a>

### Bug fixes in 1.31.0

NEW ITEM	DETAILS
Import Wizard	Fixed Import extension next button does not work in July release
Schema Compare	Fixed issue that Schema compare Select Source target dialog OK button not enabled
Notebooks	Fixed Export Notebook as SQL file has no query editor toolbar
SQL Server Big Data Clusters	Fixed Can't connect to BDC Clusters
Accessibility bug fixes	

For a full list of bug fixes addressed for the July 2021 release, visit the [bugs and issues list on GitHub](#).

## Known issues in 1.31.0

For a list of the current known issues, visit the [issues list on GitHub](#).

## June 2021

- Release number: 1.30.0
- Release date: June 17, 2021

### What's new in 1.30.0

NEW ITEM	DETAILS
Results Grid	Added filtering/sorting feature for query result grid in query editor and notebook, the feature can be invoked from the column headers. note that this feature is only available when you enable the preview features.
Results Grid	Added a status bar item to show summary of the selected cells if there are multiple numeric values
Notebooks	Added new book icon

NEW ITEM	DETAILS
Notebooks	Notebook URI Handler File Support
Python	Updated Python to 3.8.10

#### Bug fixes in 1.30.0

NEW ITEM	DETAILS
Notebooks	Fixed WYSIWYG Table cell adding new line in table cell
Notebooks	Fixed issue that Kusto notebook does not change kernels properly

For a full list of bug fixes addressed for the May 2021 release, visit the [bugs and issues list on GitHub](#).

#### Known issues in 1.30.0

For a list of the current known issues, visit the [issues list on GitHub](#).

### May 2021

- Release number: 1.29.0
- Release date: May 19, 2021

#### What's new in 1.29.0

NEW ITEM	DETAILS
Notebooks	Added run with parameters action. Learn more <a href="#">here</a>

#### Bug fixes in 1.29.0

NEW ITEM	DETAILS
Database	Fixed an issue with the title not changing when the database connection changes.
Extensions	Fixed an issue with no longer prompting the user to install 3rd party extensions.
General Azure Data Studio	Fixed an issue with the account button in the sidebar getting stuck loading.
General Azure Data Studio	Fixed an issue with the <i>Run With Parameters</i> silently failing if the parameters cell is empty or invalid.
General Azure Data Studio	Fixed an issue with the <i>Run With Parameters</i> not handling multiple parameters on the same line.
General Azure Data Studio	Fixed an issue with being unable to connect to Azure.
General Azure Data Studio	Fixed an issue with the wrong line number showing up in the output.
General Azure Data Studio	Fixed an issue when receiving an error when connecting to an Azure server in the Azure viewlet.

NEW ITEM	DETAILS
General Azure Data Studio	Fixed an issue with the loading spinner not being animated.
General Azure Data Studio	Fixed an issue with the links inserted in the split view/MD view not being inserted as a list item.
General Azure Data Studio	Fixed an issue with the <i>Issue Reporter</i> being blank.
Extensions	Fixed an issue with the extensions view having many filter options that aren't applicable to Azure Data Studio.
General Azure Data Studio	Fixed an issue with getting the Azure subscriptions API failing across Azure Data Studio.
General Azure Data Studio	Fixed an issue with the wrong event prefix.
General Azure Data Studio	Fixed an issue with converting the default tables from Excel, Word, and OneNote into markdown tables.
Notebooks	Fixed an issue with selecting notebooks from the notebooks viewlet recentering the viewlet vertically.
Notebooks	Fixed an issue with not connecting to SQL Server from SQL Notebook.
Notebooks	Fixed an issue with the notebook icons being sized incorrectly.
PowerShell	Fixed an issue with using the Cloud Shell (PowerShell).
SQL Database Project	Fixed an issue with showing an unnecessary horizontal scrollbar in the <i>create project from database</i> dialog SQL Database Project dashboard update.

For a full list of bug fixes addressed for the May 2021 release, visit the [bugs and issues list on GitHub](#).

#### Known issues in 1.29.0

For a list of the current known issues, visit the [issues list on GitHub](#).

### April 2021

April 15, 2021 / version: 1.28.0

#### What's new in 1.28.0

NEW ITEM	DETAILS
Extension update	Kusto (KQL)
Extension update	MachineLearning
Extension update	SchemaCompare
Extension update	SQLDatabaseProjects

NEW ITEM	DETAILS
Notebook features	Added <a href="#">Add Notebook</a> and <a href="#">Remove Notebook commands</a>

#### Bug fixes in 1.28.0

For the list of the bug fixes addressed for the April 2021 release, visit the [bugs and issues list on GitHub](#).

#### Known issues in 1.28.0

For the list of known issues, visit the [issues list on GitHub](#).

### March 2021

March 17, 2021 / version: 1.27.0

CHANGE	DETAILS
Bug Fixes	For a complete list of fixes, see <a href="#">Bugs and issues on GitHub</a> .
Extension(s) update	<a href="#">Dacpac</a> <a href="#">SQLDatabaseProjects</a>
New Notebook features	Added create book dialog

### February 2021

February 18, 2021 / version: 1.26.0

CHANGE	DETAILS
Bug Fixes	For a complete list of fixes see <a href="#">Bugs and issues on GitHub</a> .
Extension(s) update	<a href="#">Dacpac</a> <a href="#">Kusto (KQL)</a> <a href="#">MachineLearning</a> <a href="#">Profiler</a> <a href="#">SchemaCompare</a> <a href="#">SQLDatabaseProjects</a>
New Azure Arc features	Multiple data controllers now supported New connection dialog options like the <i>kube config file</i> Postgres dashboard enhancements
New Notebook features	Improved Jupyter server start-up time by 50% on Windows Added support to edit Jupyter Books through right-click Added URI notebook parameterization support and <a href="#">added notebook parameterization documentation</a>

### December 2020 (hotfix)

February 10, 2021 / version: 1.25.3

CHANGE	DETAILS
Fix bug #13899	Scrolling to the appropriate cross-reference links in Notebooks

CHANGE	DETAILS
Upgrade Electron to incorporate important bug fixes	N/A

## December 2020

December 9, 2020 / version: 1.25.0

CHANGE	DETAILS
Bug Fixes	For a complete list of fixes see <a href="#">Bugs and issues on GitHub</a> .
Database Projects extension update	Added workspaces and improved sidebar

## November 2020

November 12, 2020 / version: 1.24.0

CHANGE	DETAILS
Bug Fixes	For a complete list of fixes see <a href="#">Bugs and issues on GitHub</a> .
Connection dialog	Added new browse tab for connection dialog.
Extension(s) update	Released update to Postgres extension.
New notebook features	Added new features to SQL to notebook support. Added new features to Notebook parameterization support. Added new features to results streaming for SQL Notebooks.
Python installation	PROSE package has been removed from default Python installation.

## Known issues (1.24.0)

NEW ITEM	DETAILS	WORKAROUND
Azure Arc extension	<b>Known Issue:</b> The "Script to Notebook" button for Arc MIAA & PG deployments doesn't do field validation before scripting the notebook. This means that if users enter the password wrong in the password confirm inputs then they may end up with a notebook that has the wrong value for the password.	The "Deploy" button works as expected though so users should use that instead.
Object Explorer	Releases of Azure Data Studio before 1.24.0 have a breaking change in object explorer because of the engine's changes related to <a href="#">Azure Synapse Analytics serverless SQL pool</a> .	To continue utilizing object explorer in Azure Data Studio with Azure Synapse Analytics serverless SQL pool, you need to use Azure Data Studio 1.24.0 or later.

You can reference [Azure Data Studio feedback](#) for other known issues and to provide feedback to the product

team.

## October 2020

October 14, 2020 / version: 1.23.0

CHANGE	DETAILS
Azure SQL Edge	Support for Azure SQL Edge objects.
Bug Fixes	For a complete list of fixes see <a href="#">Bugs and issues on GitHub</a> .
Databases	Support for same database reference.
Extension updates	<a href="#">Azure Arc</a> <a href="#">azdata</a> <a href="#">Machine Learning</a> <a href="#">Kusto (KQL)</a> <a href="#">Schema Compare</a> <a href="#">SQL Assessment</a> <a href="#">SQL Database Projects</a> <a href="#">SQL Server Import</a>
New deployment features	Added Azure SQL DB and VM deployments.
PowerShell	Added PowerShell kernel results streaming support.

## September 2020 (hotfix)

September 30, 2020 / version: 1.22.1

CHANGE	DETAILS
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues on GitHub</a> .

## September 2020

September 22, 2020 / version: 1.22.0

CHANGE	DETAILS
New notebook features	<ul style="list-style-type: none"><li>Supports brand new text cell editing experience based on rich text formatting and seamless conversion to markdown, also known as WYSIWYG toolbar (What You See Is What You Get)</li><li>Supports Kusto kernel</li><li>Supports pinning of notebooks</li><li>Added support for new version of Jupyter Books</li><li>Improved Jupyter Shortcuts</li><li>Introduced perf loading improvements</li></ul>

CHANGE	DETAILS
SQL Database Projects extension	The SQL Database Projects extension brings project-based database development to Azure Data Studio. In this preview release, SQL projects can be created and published from Azure Data Studio.
Kusto (KQL) extension	Brings native Kusto experiences in Azure Data Studio for data exploration and data analytics against massive amount of real-time streaming data stored in Azure Data Explorer. This preview release supports connecting and browsing Azure Data Explorer clusters, writing KQL queries and authoring notebooks with Kusto kernel.
Azure Arc extension	Users can try out Azure Arc public preview through Azure Data Studio. This includes: <ul style="list-style-type: none"> <li>• Deploy data controller</li> <li>• Deploy Postgres</li> <li>• Deploy Managed Instance for Azure Arc</li> <li>• Connect to data controller</li> <li>• Access data service dashboards</li> <li>• Azure Arc Jupyter Book</li> </ul>
Deployment options	<ul style="list-style-type: none"> <li>• Azure SQL Database Edge (Edge will require Azure SQL Edge Deployment Extension)</li> </ul>
SQL Server Import extension GA	Announcing the GA of the SQL Server Import extension, features no longer in preview. This extension facilitates importing csv/txt files. Learn more about the extension in <a href="#">this article</a> .
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## August 2020

August 12, 2020 / version: 1.21.0

CHANGE	DETAILS
New notebook features	<ul style="list-style-type: none"> <li>• Move cell locations</li> <li>• Convert cells to Text Cell or Code cell</li> </ul>
Jupyter Books picker	Users can now choose Jupyter Books from GitHub releases and open seamlessly in Azure Data Studio
Search added to Notebooks Viewlet	Users can easily search through content across their notebooks and Jupyter Books
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## July 2020 (hotfix)

July 17, 2020 / version: 1.20.1

CHANGE	DETAILS
Fix bug #11372 Object Explorer drag-and-drop table incorrectly wraps table names	#11372
Fix bug #11356 Dark theme is now the default theme	#11356

## Known Issue

- Some users have reported connection errors from the new Microsoft.Data.SqlClient v2.0.0 included in this release. Users have found [following these instructions](#) to successfully connect

## July 2020

July 15, 2020 / version: 1.20.0

CHANGE	DETAILS
Added new Feature Tour	From welcome page and command palette, users can now launch a feature tour to get a walkthrough of commonly used features including Connections Viewlet, Notebooks viewlet, and Extensions Marketplace
New notebook features	<ul style="list-style-type: none"> <li>Header support in Markdown Toolbar</li> <li>Side-by-side Markdown Preview in Text Cells</li> </ul>
Drag and Drop Columns and Tables in Query Editor	Users can now directly drag and drop columns and tables from connections viewlet to query editor
Added Azure Account icon to Activity Bar	Users can now easily see where to sign in to Azure
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## June 2020

June 15, 2020 / version: 1.19.0

CHANGE	DETAILS
Added Azure Data Studio to Azure portal Integration	Users can now directly launch to Azure portal from an Azure SQL Database connection, Azure Postgres, and more.
New notebook features	<ul style="list-style-type: none"> <li>New Notebook toolbar</li> <li>New Edit Cell toolbar</li> <li>Python dependencies wizard UX updates</li> <li>Improved spacing across notebooks</li> </ul>
Announcing SQL Assessment API extension	This extension adds SQL Server best-practice assessment in Azure Data Studio. It exposes SQL Assessment API, which was previously available for use in PowerShell SqlServer module and SMO only, to let you evaluate your SQL Server instances and receive for them recommendations by SQL Server Team. Learn more about SQL Assessment API and what it's capable of <a href="#">in this article</a> .

CHANGE	DETAILS
Machine Learning Extension improvements	Now supports Azure SQL Managed Instance.
Data Virtualization extension improvements	Now supports MongoDB and Teradata
Postgres extension bug fixes	Fixed Azure MFA
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## May 2020 (hotfix)

May 27, 2020 / version: 1.18.1

CHANGE	DETAILS
Fix bug #10538 "Run Current Query" keybind no longer behaving as expected	<a href="#">#10538</a>
Fix bug #10537 Unable to open new or existing sql files on v1.18	<a href="#">#10537</a>

## May 2020

May 20, 2020 / version: 1.18.0

CHANGE	DETAILS
Announcing Redgate SQL Prompt extension	This extension lets you manage formatting styles directly within Azure Data Studio, so you can create and edit your styles without leaving the IDE.
Announcing Machine Learning Extension	This extension enables you to: <ul style="list-style-type: none"> <li>Manage Python and R packages with SQL Server machine learning services with Azure Data Studio.</li> <li>Use ONNX model to make predictions in Azure SQL Edge.</li> <li>View ONNX models in an Azure SQL Edge database.</li> <li>Import ONNX models from a file or Azure Machine Learning into Azure SQL Edge database.</li> <li>Create a notebook to run experiments.</li> </ul>
New notebook features	<ul style="list-style-type: none"> <li>Added new Python Dependencies Wizard to make it easier to install Python dependencies</li> <li>Added underline support for Markdown Toolbar</li> </ul>
Parameterization for Always Encrypted	Allows you to run queries that insert, update, or filter by encrypted database columns.
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## April 2020 (hotfix)

April 30, 2020 / version: 1.17.1

CHANGE	DETAILS
Fix bug #10197 Can't connect via MFA	#10197

## April 2020

April 27, 2020 / version: 1.17.0

CHANGE	DETAILS
Improved welcome page	UI update on the welcome page to make it easier to see common actions and highlighting extensions.
New notebook features	<ul style="list-style-type: none"> <li>Added Markdown toolbar when editing text cells to help write with Markdown</li> <li>Revamped Jupyter Books viewlet to become a Notebooks viewlet where you can manage Jupyter Books and notebooks together</li> <li>Added support for persisting charts when saving a notebook</li> <li>Added support for KQL magic in Python notebooks</li> </ul>
Improved dashboards	Dashboards throughout Azure Data Studio have been updated with latest design patterns, including an actions toolbar. This also applies to many extensions.
Added Cloud Shell integration in the Azure view.	
Support for Always Encrypted and Always Encrypted with secure enclaves.	
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## March 2020

March 18, 2020 / version: 1.16.0

CHANGE	DETAILS
Added charting support in SQL Notebooks	When running a SQL query in a code cell, users can now create and save charts.
Added Create Jupyter Book experience	Users can now create their own Jupyter Books using a notebook.
Added Azure AD support for Postgres extension	
Fixed many accessibility bugs	<a href="#">List of accessibility bugs</a>

CHANGE	DETAILS
VS Code merges to 1.42	This release includes updates to VS Code from the 3 previous VS Code releases. <a href="#">Read their release notes</a> to learn more.
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## February (hotfix)

February 19, 2020 / version: 1.15.1

CHANGE	DETAILS
Fix bug #9149 Show active connections	<a href="#">#9149</a>
Fix bug #9061 Edit Data grid doesn't properly resize when showing or hiding SQL Pane	<a href="#">#9061</a>

## February 2020

February 13, 2020 / version: 1.15.0

CHANGE	DETAILS
New Azure Sign-in improvement	Added improved Azure Sign-in experience, including removal of copy/paste of device code to make a more seamless connected experience.
Find in Notebook support	Users can now use Ctrl+F in a notebook. Find in Notebook support searches line by line through both code and text cells.
VS Code merges from 1.38 to 1.42	This release includes updates to VS Code from the 3 previous VS Code releases. <a href="#">Read their release notes</a> to learn more.
Fix for the "white/blank screen" issue reported by many users.	
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## Known Issue

- Users on macOS Catalina will need to right-click Azure Data Studio and then click open.

## December 2019 (hotfix)

December 26, 2019 / version: 1.14.1

CHANGE	DETAILS
Fix bug #8747 OE Expansion fails	<a href="#">#8747</a>

## December 2019

December 19, 2019 / version: 1.14.0

CHANGE	DETAILS
Changed attach to connection dropdown in Notebooks to only list the currently active connection	<a href="#">#8129</a>
Added bigdatacluster.ignoreSslVerification setting to allow ignoring TLS/SSL verification errors when connecting to a BDC	<a href="#">#8582</a>
Allow changing default language flavor for offline query editors	<a href="#">#8419</a>
GA status for Big Data Cluster/SQL 2019 features	<a href="#">#8269</a>
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## November 2019 (hotfix)

November 15, 2019 / version: 1.13.1

CHANGE	DETAILS
Fix bug #8210 Copy/Paste results are out of order	

## November 2019

November 4, 2019 / version: 1.13.0

CHANGE	DETAILS
New SQL Server 2019 support	<ul style="list-style-type: none"><li>• Deploy SQL Server 2019 big data cluster with BDC Deploy wizard</li><li>• Manage cluster health with controller dashboard</li><li>• Manage HDFS access control lists using Security ACLs Dialog</li><li>• Add mounts using HDFS Tiering Dialog</li><li>• Troubleshoot using built-in Jupyter Book, SQL Server 2019 guide</li><li>• Renamed to SQL vNext extension Data virtualization extension</li><li>• Added Teradata and Mongo support in External Table Wizard</li></ul>
New notebook features	<ul style="list-style-type: none"><li>• Announcing PowerShell notebooks</li><li>• Announcing collapsible code cells</li><li>• Perf improvements in Notebooks</li><li>• View the full list of improvements <a href="#">here</a></li></ul>
Announcing Jupyter Books	Jupyter Books are a collection of notebooks and markdown files organized in a table of contents.

CHANGE	DETAILS
New SQL Server Deploy wizard	Now includes support for deploying: <ul style="list-style-type: none"><li>• SQL Server 2019 on Windows</li><li>• SQL Server 2017 on Windows</li><li>• SQL Server 2019 on Docker</li><li>• SQL Server 2017 on Docker</li></ul>
Announcing GA of Schema Compare extension	<ul style="list-style-type: none"><li>• SQLCMD mode</li><li>• Localization support</li><li>• Accessibility fixes</li><li>• Security bugs</li></ul>
Announcing GA of SQL Server Dacpac extension	<ul style="list-style-type: none"><li>• Localization support</li><li>• Accessibility fixes</li><li>• Security bugs</li></ul>
Announcing Visual Studio IntelliCode extension	Visual Studio IntelliCode now supports SQL, which allows for smarter suggestions of reserved keywords.
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## October 2019 (hotfix 2)

October 11, 2019 / version: 1.12.2

CHANGE	DETAILS
Disable automatically starting the EH in inspect mode	

## October 2019 (hotfix)

October 8, 2019 / version: 1.12.1

CHANGE	DETAILS
Fixed issue for quotes and backslashes in Notebooks to escape correctly.	

## October 2019

October 2, 2019 / version: 1.12.0

CHANGE	DETAILS
Release of Query History extension	The SQL History extension saves all past queries executed in an Azure Data Studio session and lists them in execution order. Users can see open the query, execute the query, delete the query, pause query history, or delete all query history entries.

CHANGE	DETAILS
New Copy/Paste Results	We have added more ways to copy/paste results from the results grid.
Update to PowerShell extension	
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## Known Issues

- Notebooks
  - [7080](#) Rare Case when Notebook is Serialized Incorrectly

## September 2019

September 10, 2019 / version: 1.11.0

CHANGE	DETAILS
Enable SQLCMD Mode	Query editor now supports toggling of SQLCMD mode to write and edit queries as SQLCMD scripts
Community Extension: Query Editor Boost	Query Editor Boost is an open-source extension focused on enhancing the Azure Data Studio query editor for users who are frequently writing queries. <ul style="list-style-type: none"> <li>• Save the current query as a snippet</li> <li>• Switch databases using Ctrl+U</li> <li>• New Query from template</li> <li>• View the full list of improvements <a href="#">here</a></li> </ul>
Notebook Improvements	<ul style="list-style-type: none"> <li>• Performance improvements for supporting larger notebook files</li> <li>• View the full list of improvements <a href="#">here</a></li> </ul>
Visual Studio Code August Release Merge 1.38	Latest improvements can be found <a href="#">here</a> .
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## Known Issues

- Notebooks
  - [7080](#) Rare Case when Notebook is Serialized Incorrectly

## August 2019

August 15, 2019 / version: 1.10.0

CHANGE	DETAILS
Release of SandDance 1.3.1 extension	<ul style="list-style-type: none"> <li>• Smart chart detection</li> <li>• 3D Visualizations</li> <li>• Data filtering</li> </ul>

CHANGE	DETAILS
Notebook Improvements	<ul style="list-style-type: none"> <li>• Add code or text cell in-line</li> <li>• Added ability to right-click SQL results grid to save result as CSV, JSON, etc.</li> <li>• Improvement to notebook loading performance for loading JSON faster</li> <li>• View the full list of improvements <a href="#">here</a></li> </ul>
SQL Server 2019 Support	<p>This release includes support for extra SQL Server 2019 Big Data Cluster features including:</p> <ul style="list-style-type: none"> <li>• Reduced time taken to load table and column information on the object-mapping page.</li> <li>• Fixed a bug with loading existing database scoped credentials on the connection details page.</li> <li>• Increased default sample size used for PROSE parsing.</li> </ul>
Dacpac extension now supports Azure AD	
Visual Studio Code July Release Merge 1.37	Latest improvements can be found <a href="#">here</a> .
Resolved bugs and issues	For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a> .

## July 2019

July 11, 2019 / version: 1.9.0

CHANGE	DETAILS
Release of SentryOne Plan Explorer extension	<p>Our valued Microsoft partner, SentryOne, will be shipping their <a href="#">SentryOne Plan Explorer extension for Azure Data Studio</a>.</p> <p>This is a free extension, which provides enhanced plan diagrams for queries run in Azure Data Studio, with optimized layout algorithms and intuitive color-coding to help quickly identify the most expensive operators affecting query performance. To learn more about the extension, check out SentryOne's blog post <a href="#">here</a>.</p>
New Features coming to Schema Compare	<ul style="list-style-type: none"> <li>• Schema Compare File Support (.SCMP)</li> <li>• Cancel Schema Compare Support</li> <li>• Complete changes can be found <a href="#">here</a></li> </ul>
Notebook Improvements	<ul style="list-style-type: none"> <li>• Plotly Python Support</li> <li>• Open Notebook from Browser</li> <li>• Python Package Management Dialog</li> <li>• Performance and Markdown Enhancements</li> <li>• Keyboard Shortcuts Update</li> <li>• Bug Fixes and Minor Features can be found <a href="#">here</a></li> </ul>
SQL Server 2019 Support	<p>This release includes support for extra SQL Server 2019 Big Data Cluster features including:</p> <ul style="list-style-type: none"> <li>• Service Endpoints table within the Management Dashboard that lists all key services in the cluster.</li> <li>• Cluster Status Notebook shows how you can query &amp; troubleshoot cluster status across all services and pods.</li> </ul>

CHANGE	DETAILS
Updated Language Packs Available	<p>There are now 10 language packs available in the Extension Manager marketplace. Simply, search for the specific language using the extension marketplace and install. Once you install the selected language, Azure Data Studio will prompt you to restart with the new language.</p>
SQL Server Profiler Update	<p>The SQL Server Profiler extension has been updated to include new features including:</p> <ul style="list-style-type: none"> <li>• Filtering by Database Name</li> <li>• Copy &amp; Paste Support</li> <li>• Save/Load Filter</li> </ul> <p>A full list of improvements for SQL Server Profiler Extension can be found <a href="#">here</a>.</p>
Visual Studio Code May Release Merge 1.35	Latest improvements can be found <a href="#">here</a> .
Resolved bugs and issues	<p>In previous releases of Azure Data Studio, if a user database was selected when connecting from the Connection dialog, the resulting Object Explorer entry was scoped entirely to that single database. Beginning in this release, that behavior is being changed so that server level properties are also shown in the object explorer.</p> <p>For a complete list of fixes see <a href="#">Bugs and issues, on GitHub</a>.</p>

## June 2019

June 6, 2019 / version: 1.8.0

CHANGE	DETAILS
Release of Central Management Servers (CMS) extension	Central Management Servers store a list of instances of SQL Server that is organized into one or more central management server groups. Users can connect to their own existing CMS servers and manage their servers like adding and removing servers. To learn more, you can read <a href="#">here</a>
Release of Database Administration Tool Extensions for Windows	This extension launches two of the most used experiences in SQL Server Management Studio from Azure Data Studio. Users can right-click on many different objects (such as Databases, Tables, Columns, Views, and more) and select Properties to view the SSMS Properties Dialog for that object. In addition, users can right-click on a database and select Generate Scripts to launch the well-known SSMS Generate Scripts Wizard.
Schema Compare Improvements	<ul style="list-style-type: none"> <li>• Added Exclude/Include Options</li> <li>• Generate Script opens script after being generated</li> <li>• Removed double scroll bars</li> <li>• Formatting and layout improvements</li> <li>• Complete changes can be found <a href="#">here</a></li> </ul>
Moved Messages section to own tab	When users ran SQL queries, results and messages were on stacked panels. Now they are in separate tabs in one panel like in SSMS.

CHANGE	DETAILS
SQL Notebook Improvements	<ul style="list-style-type: none"> <li>Users can now choose to use their own Python 3 or Anaconda installs in notebooks</li> <li>Multiple Stabilities + fit/finish fixes</li> <li>View the full list of improvements <a href="#">here</a></li> </ul>
Visual Studio Code April Release Merge 1.34	Latest improvements can be found <a href="#">here</a>
Resolved bugs and issues.	See <a href="#">Bugs and issues, on GitHub</a> .

## Known Issues

- Database Administration Tool Extensions for Windows
  - Can't launch properties from disconnected server node
  - Can't launch properties for Azure servers
  - Not all objects have property dialogs
  - Dialogs take a long time to start up
  - Errors launching servers with some types of connections (such as Azure AD)
- Notebooks
  - [5838](#) Allow users to use system Python for Notebooks
- Schema Compare
  - [5804](#) Schema Compare tasks show default cancel context menu, which doesn't work

## May 2019

May 8, 2019 / version: 1.7.0

CHANGE	DETAILS
Release of Schema Compare extension	Schema Compare is a well-known feature in SQL Server Data Tools (SSDT), and its primary use case is to compare and visualize the differences between databases and .dacpac files and to execute actions to make them the same.
Moved Task view to Output Window	Users can now view the status of long running tasks like Backup, Restore, and Schema Compare in the Task view in Output window
Added Welcome page	<ul style="list-style-type: none"> <li>Links to common actions like New Query, New File, New Notebook</li> <li>Links to documentation and GitHub</li> </ul>

CHANGE	DETAILS
SQL Notebook Improvements	<ul style="list-style-type: none"> <li>Markdown rendering improvements, including better support for notes and tables</li> <li>Usability improvements to the toolbar</li> <li>Markdown links for trusted notebooks no longer require Cmd/Ctrl + click and can be clicked directly</li> <li>Improvements in cleaning up Jupyter processes after closing notebooks and reducing errors when starting multiple notebooks concurrently</li> <li>Improvements to SQL notebook connections to ensure errors don't occur when running 2 notebooks against the same database</li> <li>Improvements to notebook autoscrolling to the currently executing cell when clicking the Run Cells button from the toolbar</li> <li>General stability and performance improvements</li> </ul>
Resolved bugs and issues.	See <a href="#">Bugs and issues, on GitHub</a> .

## April 2019

April 18, 2019 / version: 1.6.0

CHANGE	DETAILS
Renamed Servers tab to Connections	
Moved Azure Resource Explorer as an Azure viewlet under Connections	Users can now view their Azure SQL instances through Azure viewlet in the Connections view and expand to view objects under each server or database.
SQL Notebook Improvements	<ul style="list-style-type: none"> <li>Added button on toolbar to clear output for all cells</li> <li>Added button on toolbar to run all cells</li> <li>Fixed connection name instead of server name (if set) in the Attach to dropdown</li> <li>Fix for images in markdown not rendering when using relative image paths</li> <li>Improved functionality in notebook grids by adding double-click auto resize column size and improved mousewheel support</li> <li>Improvements to error handling and python install resiliency when installing python through notebooks</li> <li>Improvements to "select all" functionality when selecting notebook cells</li> <li>Improvements to notebook connections to prevent closing a notebook and impacting an object explorer connection</li> <li>Improved notebook experience to display a message to the user when notebook disconnected and needs a connection to run cells</li> <li>Improved support for unsaved notebooks to rehydrate in Azure Data Studio when Azure Data Studio is started again</li> </ul>
Resolved bugs and issues.	See <a href="#">Bugs and issues, on GitHub</a> .

## March 2019 (Hotfix)

March 22, 2019 / version: 1.5.2 / Hotfix release

CHANGE	DETAILS
Fixed a few issues discovered in 1.5.1.	<p>See <a href="#">March Hotfix Release, on GitHub</a>.</p> <ul style="list-style-type: none"> <li>• Fixed issue where user couldn't close notebook opened from the "Open Notebook" task in the Dashboard</li> <li>• Fixed issue where Notebook JSON has extra } after save</li> <li>• Fixed issue where notebook grids weren't responding to theme changes</li> <li>• Fixed issue where full notebook path was shown in the tab header. Now only the filename is shown.</li> </ul>

## March 2019

March 18, 2019 / version: 1.5.1

CHANGE	DETAILS
Added PostgreSQL extension for Azure Data Studio	<p>Supported features:</p> <ul style="list-style-type: none"> <li>• Connection Dialog</li> <li>• Object Explorer</li> <li>• Query Editor</li> <li>• Charting</li> <li>• Dashboards</li> <li>• Snippets</li> <li>• Edit Data</li> <li>• Notebooks</li> </ul>
Added SQL Notebooks	<p>Added SQL Kernel support to built-in Notebook viewer:</p> <ul style="list-style-type: none"> <li>• Supports T-SQL</li> <li>• Support PGSQL</li> </ul>
Added PowerShell Extension	Brings over the <a href="#">PowerShell extension</a> experience from VS Code.
Added SQL Server dacpac extension	Removes Data-Tier Application Wizard from SQL Server Import extension into a new extension.
Added Community extension QueryPlan.show	Adds integration support to visualize query plans
Updated SQL Server 2019 Preview extension	<ul style="list-style-type: none"> <li>• Jupyter Notebook support, specifically Python3, and Spark kernels, have moved into the core Azure Data Studio tool.</li> <li>• Bug fixes to External Data Wizard</li> </ul>
Resolved bugs and issues.	See <a href="#">Bugs and issues, on GitHub</a> .

## Known Issues

- [#4427](#): Clicking Run on Cell Before Kernel is Ready for Spark Results in Fatal Error **Workaround**: Wait until kernels are loaded until running any cells
- [#4493](#): Azure Data Studio launched from SSMS using SQL auth - prompts user for password **Workaround**: Use Windows Auth for now.
- [#4494](#): Unable to install SQL notebook feature  
**Workaround**: Follow workaround steps [here](#).

- [#4503](#): Azure Data Studio can't be Opened Directly from Download Azure Data Studio Folder (Mac)  
**Workaround:** Restart computer after unzipping the app. Will be investigated.
- [#4539](#): Notebook Save As loses connection context  
**Workaround:** Will be fixed in next release.
- [#4458](#): Dacpac Extract crashes SqlToolsService if invalid version is used  
**Workaround:** Restart Azure Data Studio and ensure correct version is used.
- New Notebook and Open Notebook icons are lost  
**Workaround:** The legacy connection type is deprecated. We recommend connecting to the SQL Server endpoint and you'll get all the actions (New Notebook, Spark Job) as expected.

## February 2019

February 13, 2019 / version: 1.4.5

CHANGE	DETAILS
Added Admin pack for SQL Server extension pack.	This makes it easier to install SQL Server admin-related extensions. This includes: <ul style="list-style-type: none"> <li>• <a href="#">SQL Server Agent</a></li> <li>• <a href="#">SQL Server Profiler</a></li> <li>• <a href="#">SQL Server Import</a></li> </ul>
Added filtering extended event support in Profiler extension.	
Added Save as XML feature that can save T-SQL results as XML.	
Added Data-Tier Application Wizard improvements.	<ul style="list-style-type: none"> <li>• Added Generate script button</li> <li>• Added view to give warnings of possible data loss during deployment.</li> </ul>
Updates to the SQL Server 2019 Preview extension.	See <a href="#">Data Virtualization extension</a> .
Results streaming enabled by default for long running queries.	
Resolved bugs and issues.	See <a href="#">Bugs and issues, on GitHub</a> .

## January 2019 (Hotfix)

January 16, 2019 / version: 1.3.9 / Hotfix release

CHANGE	DETAILS
Fixed a few issues discovered in 1.3.8.	See <a href="#">January Hotfix Release, on GitHub</a> .  For detailed information, see: <ul style="list-style-type: none"> <li>• <a href="#">Change Log, on GitHub</a>.</li> <li>• <a href="#">Releases, on GitHub</a>.</li> </ul>

## January 2019

January 09, 2019 / version: 1.3.8

CHANGE	DETAILS
Added a new user installer for Windows.	Unlike the existing system installer, the new user installer doesn't require administrator privileges. This also enables an easier upgrade experience for non-administrators.
Added Azure Active Directory Authentication support.	
Announcing Idera SQL DM Performance Insights (preview).	
Data-Tier Application Wizard support in SQL Server Import extension.	
Update to the SQL Server 2019 Preview extension.	See <a href="#">Data Virtualization extension</a> .
SQL Server Profiler improvements.	
Results Streaming for large queries (preview).	
Community extensions: sp_executesql to sql and New Database.	
Resolved bugs and issues.	See <a href="#">Bugs and issues, on GitHub</a> .

## Next steps

See one of the following quickstarts to get started:

- [Download Azure Data Studio](#)
- [Connect & Query SQL Server](#)
- [Connect & Query Azure SQL Database](#)
- [Connect & Query Azure Synapse Analytics](#)

Contribute to Azure Data Studio:

- <https://github.com/Microsoft/azuredatastudio>

# What is Azure Data Studio?

9/26/2022 • 4 minutes to read • [Edit Online](#)

Azure Data Studio is a cross-platform database tool for data professionals using on-premises and cloud data platforms on Windows, macOS, and Linux.

Azure Data Studio offers a modern editor experience with IntelliSense, code snippets, source control integration, and an integrated terminal. It's engineered with the data platform user in mind, with built-in charting of query result sets and customizable dashboards.

The source code for Azure Data Studio and its data providers is available on GitHub under a source code EULA that provides rights to modify and use the software, but not to redistribute it or host it in a cloud service. For more information, see [Azure Data Studio FAQ](#).

## [Download and Install Azure Data Studio](#)

## SQL code editor with IntelliSense

Azure Data Studio offers a modern, keyboard-focused SQL coding experience that makes your everyday tasks easier with built-in features, such as multiple tab windows, a rich SQL editor, IntelliSense, keyword completion, code snippets, code navigation, and source control integration (Git). Run on-demand SQL queries, view and save results as text, JSON, or Excel. Edit data, organize your favorite database connections, and browse database objects in a familiar object browsing experience. To learn how to use the SQL editor, see [Use the SQL editor to create database objects](#).

## Smart SQL code snippets

SQL code snippets generate the proper SQL syntax to create databases, tables, views, stored procedures, users, logins, roles, and to update existing database objects. Use smart snippets to quickly create copies of your database for development or testing purposes, and to generate and execute CREATE and INSERT scripts.

Azure Data Studio also provides functionality to create custom SQL code snippets. To learn more, see [Create and use code snippets](#).

## Customizable Server and Database Dashboards

Create rich customizable dashboards to monitor and quickly troubleshoot performance bottlenecks in your databases. To learn about insight widgets, and database (and server) dashboards, see [Manage servers and databases with insight widgets](#).

## Connection management (server groups)

Server groups provide a way to organize connection information for the servers and databases you work with. For details, see [Server groups](#).

## Integrated Terminal

Use your favorite command-line tools (for example, Bash, PowerShell, sqlcmd, bcp, and ssh) in the Integrated Terminal window right within the Azure Data Studio user interface. To learn about the integrated terminal, see [Integrated terminal](#).

# Extensibility and extension authoring

Enhance the Azure Data Studio experience by extending the functionality of the base installation. Azure Data Studio provides extensibility points for data management activities, and support for extension authoring.

To learn about extensibility in Azure Data Studio, see [Extensibility](#). To learn about authoring extensions, see [Extension authoring](#).

## Feature comparison with SQL Server Management Studio (SSMS)

### Use Azure Data Studio if you:

- Are mostly editing or executing queries.
- Need the ability to quickly chart and visualize result sets.
- Can execute most administrative tasks via the integrated terminal using sqlcmd or PowerShell.
- Have minimal need for wizard experiences.
- Do not need to do deep administrative or platform related configuration.
- Need to run on macOS or Linux.

### Use SQL Server Management Studio if you:

- Are doing complex administrative or platform configuration.
- Are doing security management, including user management, vulnerability assessment, and configuration of security features.
- Need to make use of performance tuning advisors and dashboards.
- Use database diagrams and table designers.
- Need access to Registered Servers.
- Make use of live query stats or client statistics.

### Shell features

FEATURE	AZURE DATA STUDIO	SSMS
Azure Sign-In	Yes	Yes
Dashboard	Yes	
Extensions	Yes	
Integrated Terminal	Yes	
Object Explorer	Yes	Yes
Object Scripting	Yes	Yes
Project System	Yes	
Select from Table	Yes	Yes
Source Code Control	Yes	
Task Pane	Yes	

FEATURE	AZURE DATA STUDIO	SSMS
Themes, including Dark Mode	Yes	
Azure Resource Explorer	Preview	
Generate Scripts Wizard		Yes
Object Properties		Yes
Table Designer	Preview	Yes

## Query Editor

FEATURE	AZURE DATA STUDIO	SSMS
Chart Viewer	Yes	
Export Results to CSV, JSON, XLSX	Yes	
Results to File		Yes
Results to Text		Yes
IntelliSense	Yes	Yes
Snippets	Yes	Yes
Show Plan	Preview	Yes
Client Statistics		Yes
Live Query Stats		Yes
Query Options		Yes
Spatial Viewer		Yes
SQLCMD	Yes	Yes

## Operating System Support

FEATURE	AZURE DATA STUDIO	SSMS
Windows	Yes	Yes
macOS	Yes	
Linux	Yes	

## Data Engineering

FEATURE	AZURE DATA STUDIO	SSMS
External Data Wizard	Preview	
HDFS Integration	Preview	
Notebooks	Preview	

## Database Administration

FEATURE	AZURE DATA STUDIO	SSMS
Backup / Restore	Yes	Yes
Flat File Import	Yes	Yes
SQL Agent	Preview	Yes
SQL Profiler	Preview	Yes
Always On		Yes
Always Encrypted		Yes
Copy Data Wizard		Yes
Data Tuning Advisor		Yes
Database Diagrams		Yes
Error Log Viewer		Yes
Maintenance Plans		Yes
Multi-Server Query		Yes
Policy Based Management		Yes
PolyBase		Yes
Query Store		Yes
Registered Servers		Yes
Replication		Yes
Security Management		Yes
Service Broker		Yes
SQL Assessment	Preview	Yes

FEATURE	AZURE DATA STUDIO	SSMS
SQL Mail		Yes
Template Explorer		Yes
Vulnerability Assessment		Yes
XEvent Management		Yes

## Database Development

FEATURE	AZURE DATA STUDIO	SSMS
Import\Export DACPAC	Yes	Yes
SQL Projects	Preview	
Schema Compare	Yes	

## SQL Tools Service

Azure Data Studio uses the [SqlToolsService](#) as the SQL API layer to the application. SQL Tools Service is .NET-based and is open source under the MIT license. For SQL connectivity, SQL Tools Service uses [Microsoft.Data.SqlClient](#) as the SQL driver.

## Next steps

- [Download and Install Azure Data Studio](#)
- [Azure Data Studio FAQ](#)
- [Connect and query SQL Server](#)
- [Connect and query Azure SQL Database](#)
- [Download SQL Server Management Studio \(SSMS\)](#)

## ⓘ Get help for SQL tools

- [All the ways to get help](#)
- [Submit an Azure Data Studio Git issue](#)
- [Contribute to Azure Data Studio](#)
- [SQL Client Tools Forum](#)
- [SQL Server Data Tools - MSDN forum](#)
- [Support options for business users](#)

## ✍ Contribute to SQL documentation

Did you know that you can edit SQL content yourself? If you do so, not only do you help improve our documentation, but you also get credited as a contributor to the page.

For more information, see [How to contribute to SQL Server documentation](#)

# Quickstart: Use Azure Data Studio to connect and query SQL Server

9/26/2022 • 2 minutes to read • [Edit Online](#)

This quickstart shows how to use Azure Data Studio to connect to SQL Server, and then use Transact-SQL (T-SQL) statements to create the *TutorialDB* used in Azure Data Studio tutorials.

## Prerequisites

To complete this quickstart, you need Azure Data Studio, and access to SQL Server.

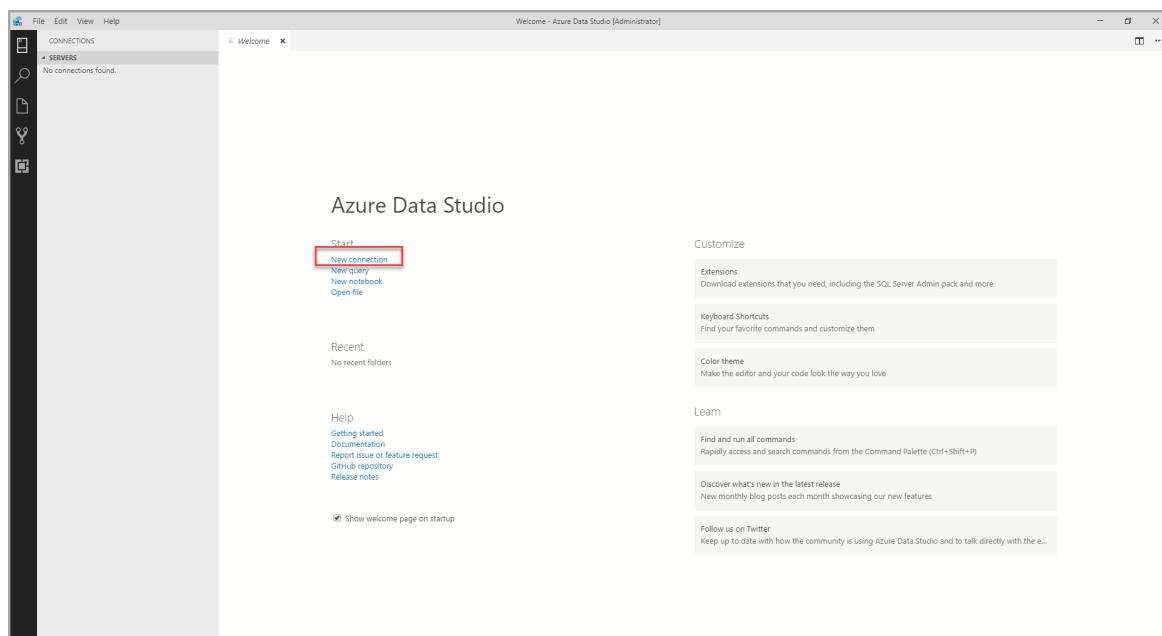
- [Install Azure Data Studio](#).

If you don't have access to a SQL Server, select your platform from the following links (make sure you remember your SQL Login and Password!):

- [Windows - Download SQL Server 2019 Developer Edition](#)
- [macOS - Download SQL Server 2019 on Docker](#)
- [Linux - Download SQL Server 2019 Developer Edition](#) - You only need to follow the steps up to *Create and Query Data*.

## Connect to a SQL Server

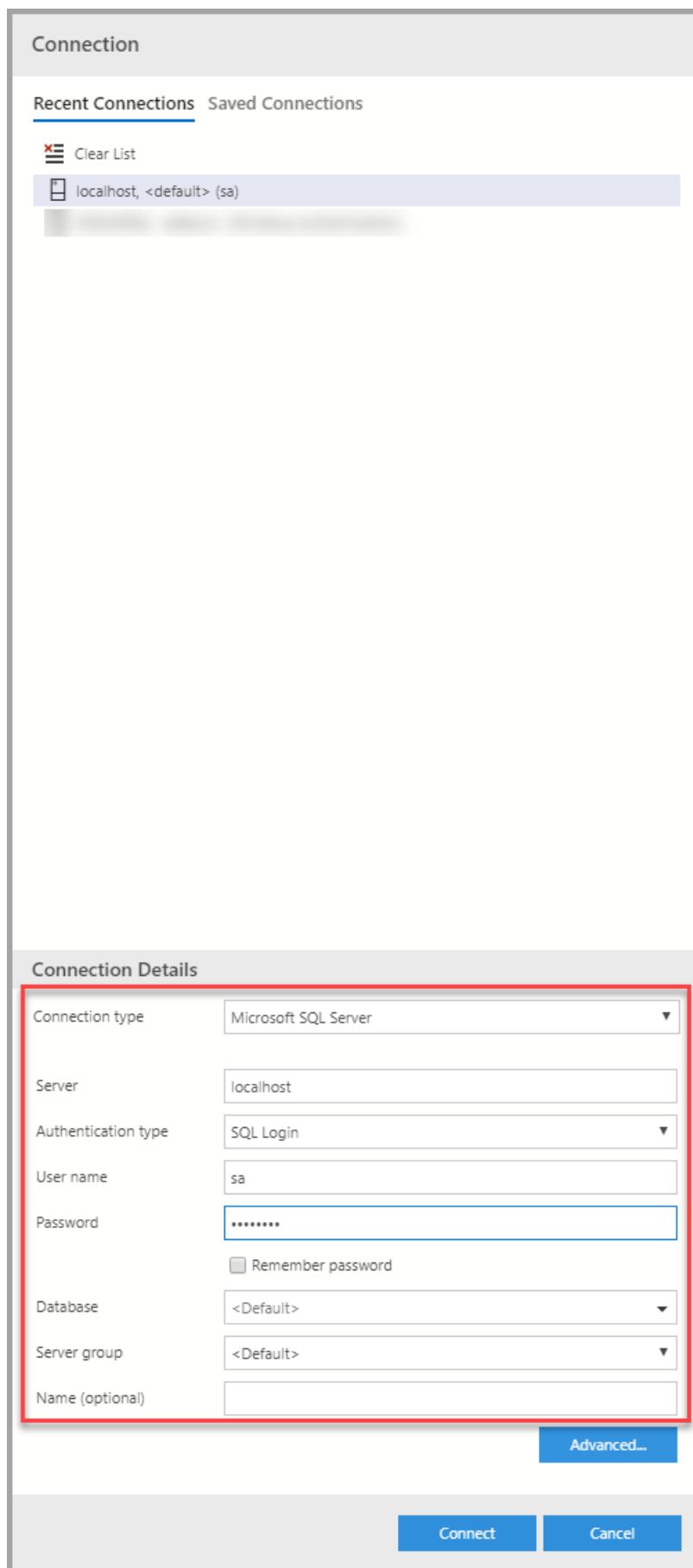
1. Start [Azure Data Studio](#).
2. The first time you run Azure Data Studio the **Welcome** page should open. If you don't see the **Welcome** page, select **Help > Welcome**. Select **New Connection** to open the **Connection** pane:



3. This article uses *SQL Login*, but *Windows Authentication* is supported. Fill in the fields as follows:

- **Server Name:** Enter server name here. For example, localhost.
- **Authentication Type:** SQL Login
- **User name:** User name for the SQL Server

- **Password:** Password for the SQL Server
- **Database Name:** <Default>
- **Server Group:** <Default>



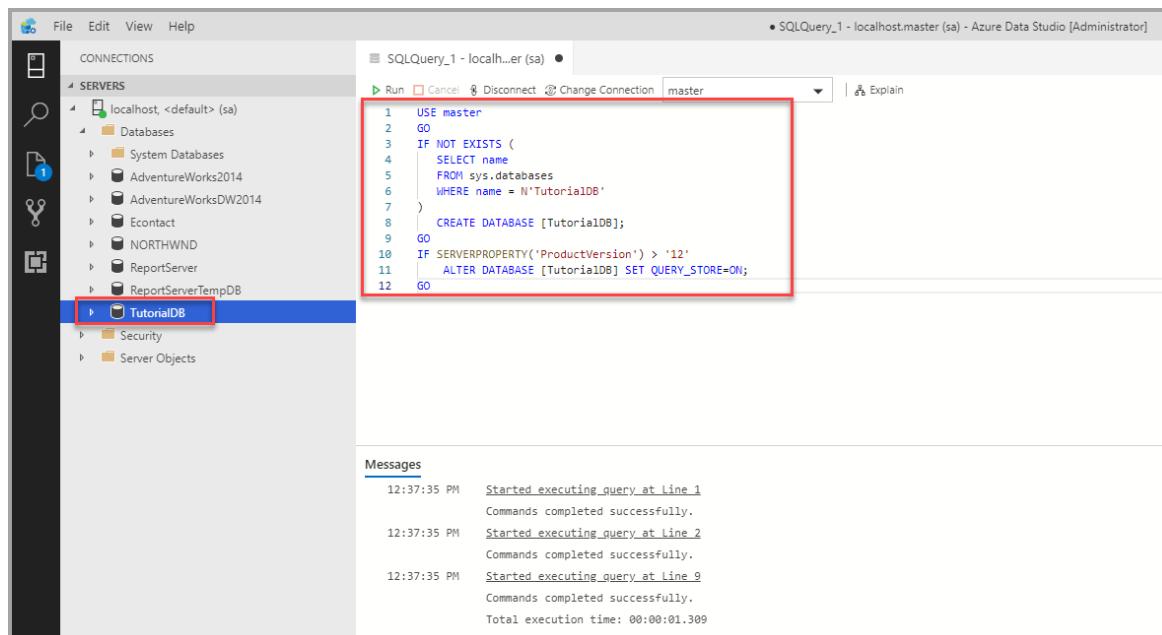
Create a database

The following steps create a database named **TutorialDB**:

1. Right-click on your server, **localhost**, and select **New Query**.
2. Paste the following snippet into the query window: and then select **Run**.

```
USE master
GO
IF NOT EXISTS (
    SELECT name
    FROM sys.databases
    WHERE name = N'TutorialDB'
)
CREATE DATABASE [TutorialDB];
GO
IF SERVERPROPERTY('ProductVersion') > '12'
    ALTER DATABASE [TutorialDB] SET QUERY_STORE=ON;
GO
```

After the query completes, the new **TutorialDB** appears in the list of databases. If you don't see it, right-click the **Databases** node and select **Refresh**.



## Create a table

The query editor is still connected to the *master* database, but we want to create a table in the *TutorialDB* database.

1. Change the connection context to **TutorialDB**:

```
1 USE master
2 GO
3 IF NOT EXISTS (
4     SELECT name
5     FROM sys.databases
6     WHERE name = N'TutorialDB'
7 )
8 CREATE DATABASE [TutorialDB];
9 GO
10 IF SERVERPROPERTY('ProductVersion') > '12'
11     ALTER DATABASE [TutorialDB] SET QUERY_STORE=ON;
12 GO
```

Messages

12:37:35 PM Started executing query at Line 1  
Commands completed successfully.  
12:37:35 PM Started executing query at Line 2  
Commands completed successfully.  
12:37:35 PM Started executing query at Line 9  
Commands completed successfully.  
Total execution time: 00:00:01.309

2. Paste the following snippet into the query window and click **Run**:

**NOTE**

You can append this too, or overwrite the previous query in the editor. Note that clicking **Run** executes only the query that is selected. If nothing is selected, clicking **Run** executes all queries in the editor.

```
-- Create a new table called 'Customers' in schema 'dbo'
-- Drop the table if it already exists
IF OBJECT_ID('dbo.Customers', 'U') IS NOT NULL
    DROP TABLE dbo.Customers;
GO
-- Create the table in the specified schema
CREATE TABLE dbo.Customers
(
    CustomerId int NOT NULL PRIMARY KEY, -- primary key column
    Name nvarchar(50) NOT NULL,
    Location nvarchar(50) NOT NULL,
    Email nvarchar(50) NOT NULL
);
GO
```

After the query completes, the new **Customers** table appears in the list of tables. You might need to right-click the **TutorialDB > Tables** node and select **Refresh**.

## Insert rows

- Paste the following snippet into the query window and click **Run**:

```
-- Insert rows into table 'Customers'  
INSERT INTO dbo.Customers  
([CustomerId], [Name], [Location], [Email])  
VALUES  
( 1, N'Orlando', N'Australia', N''),  
( 2, N'Keith', N'India', N'keith0@adventure-works.com'),  
( 3, N'Donna', N'Germany', N'donna0@adventure-works.com'),  
( 4, N'Janet', N'United States', N'janet1@adventure-works.com')  
GO
```

## View the data returned by a query

- Paste the following snippet into the query window and click **Run**:

```
-- Select rows from table 'Customers'  
SELECT * FROM dbo.Customers;
```

The screenshot shows the SSMS interface. On the left is the Object Explorer with a tree view of servers, databases, and tables. The 'TutorialDB' database is selected. In the center, there are two query panes: 'SQLQuery\_1 - localhost...DB (sa)' and 'SQLQuery\_2 - localhost...DB (sa)'. The second pane is active and contains the query: 'SELECT \* FROM dbo.Customers;'. Below the panes is the 'Results' grid, which displays the following data:

	CustomerId	Name	Location	Email
1	1	Orlando	Australia	
2	2	Keith	India	keith0@adventure-works.com
3	3	Donna	Germany	donna0@adventure-works.com
4	4	Janet	United States	janet1@adventure-works.com

## Next steps

Now that you've successfully connected to SQL Server and run a query try out the [Code editor tutorial](#).

# Quickstart: Use Azure Data Studio to connect and query Azure SQL Database

9/26/2022 • 3 minutes to read • [Edit Online](#)

In this quickstart, you'll use Azure Data Studio to connect to an Azure SQL Database server. You'll then run Transact-SQL (T-SQL) statements to create and query the TutorialDB database, which is used in other Azure Data Studio tutorials.

## Prerequisites

To complete this quickstart, you need Azure Data Studio, and an Azure SQL Database server.

- [Install Azure Data Studio](#)

If you don't have an Azure SQL server, complete one of the following Azure SQL Database quickstarts.

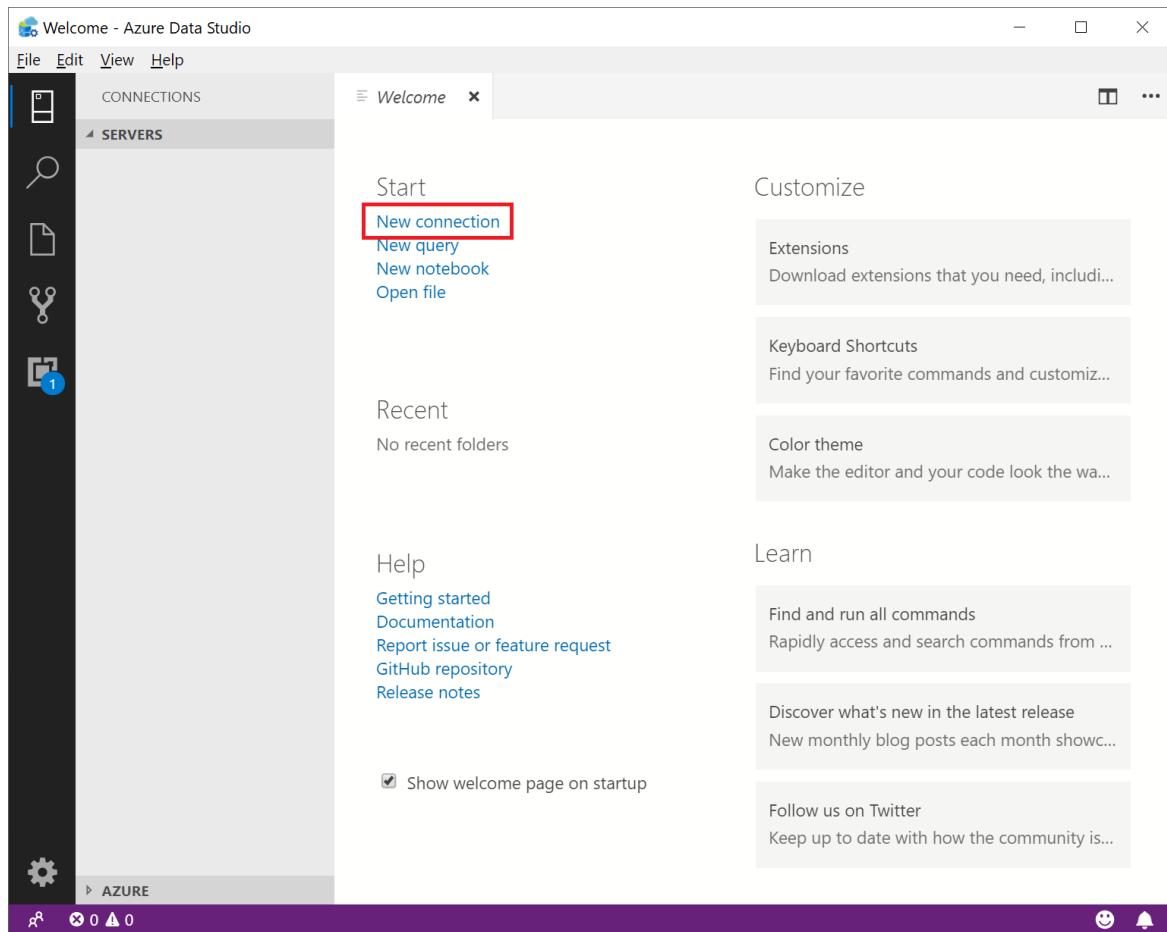
Remember the fully qualified server name and sign in credentials for later steps:

- [Create DB - Portal](#)
- [Create DB - CLI](#)
- [Create DB - PowerShell](#)

## Connect to your Azure SQL Database server

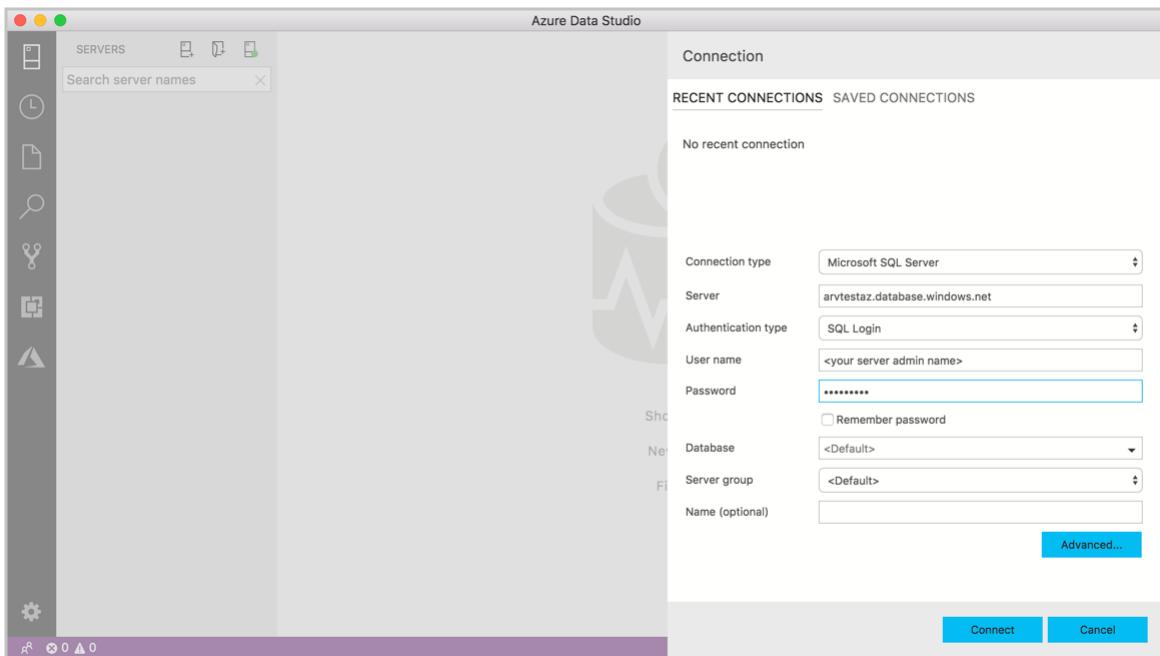
Use Azure Data Studio to establish a connection to your Azure SQL Database server.

1. The first time you run Azure Data Studio the **Welcome** page should open. If you don't see the **Welcome** page, select **Help > Welcome**. Select **New Connection** to open the **Connection** pane:

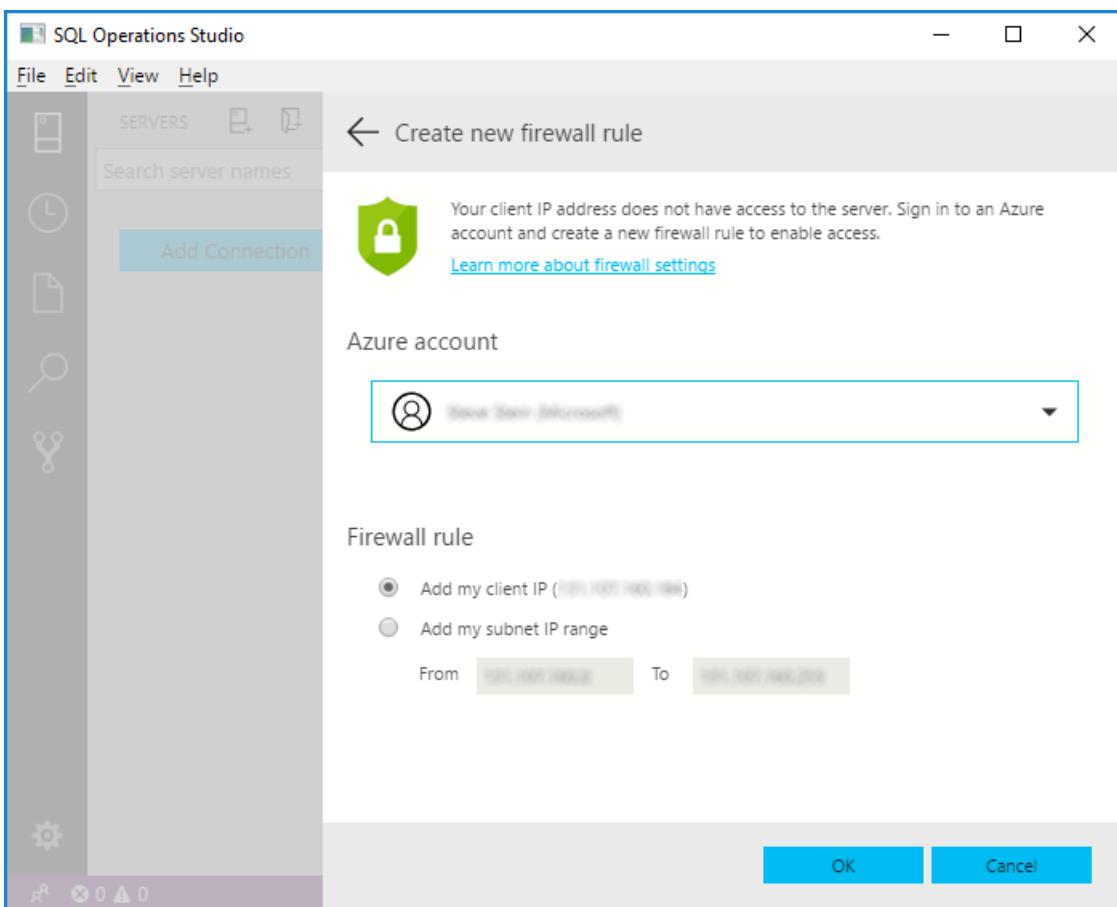


2. This article uses SQL sign-in, but also supports Windows authentication. Fill in the following fields using the server name, user name, and password for your Azure SQL server:

SETTING	SUGGESTED VALUE	DESCRIPTION
Server name	The fully qualified server name	Something like: <b>servername.database.windows.net.</b>
Authentication	SQL Login	This tutorial uses SQL Authentication.
User name	The server admin account user name	The user name from the account used to create the server.
Password (SQL Login)	The server admin account password	The password from the account used to create the server.
Save Password?	Yes or No	Select Yes if you don't want to enter the password each time.
Database name	<i>leave blank</i>	You're only connecting to the server here.
Server Group	Select <Default>	You can set this field to a specific server group you created.



3. Select **Connect**.
4. If your server doesn't have a firewall rule allowing Azure Data Studio to connect, the **Create new firewall rule** form opens. Complete the form to create a new firewall rule. For details, see [Firewall rules](#).



After successfully connecting, your server opens in the SERVERS sidebar.

## Create the tutorial database

The next sections create the TutorialDB database that's used in other Azure Data Studio tutorials.

1. Right-click on your Azure SQL server in the SERVERS sidebar and select **New Query**.

2. Paste this SQL into the query editor.

```
IF NOT EXISTS (
    SELECT name
    FROM sys.databases
    WHERE name = N'TutorialDB'
)
CREATE DATABASE [TutorialDB]
GO

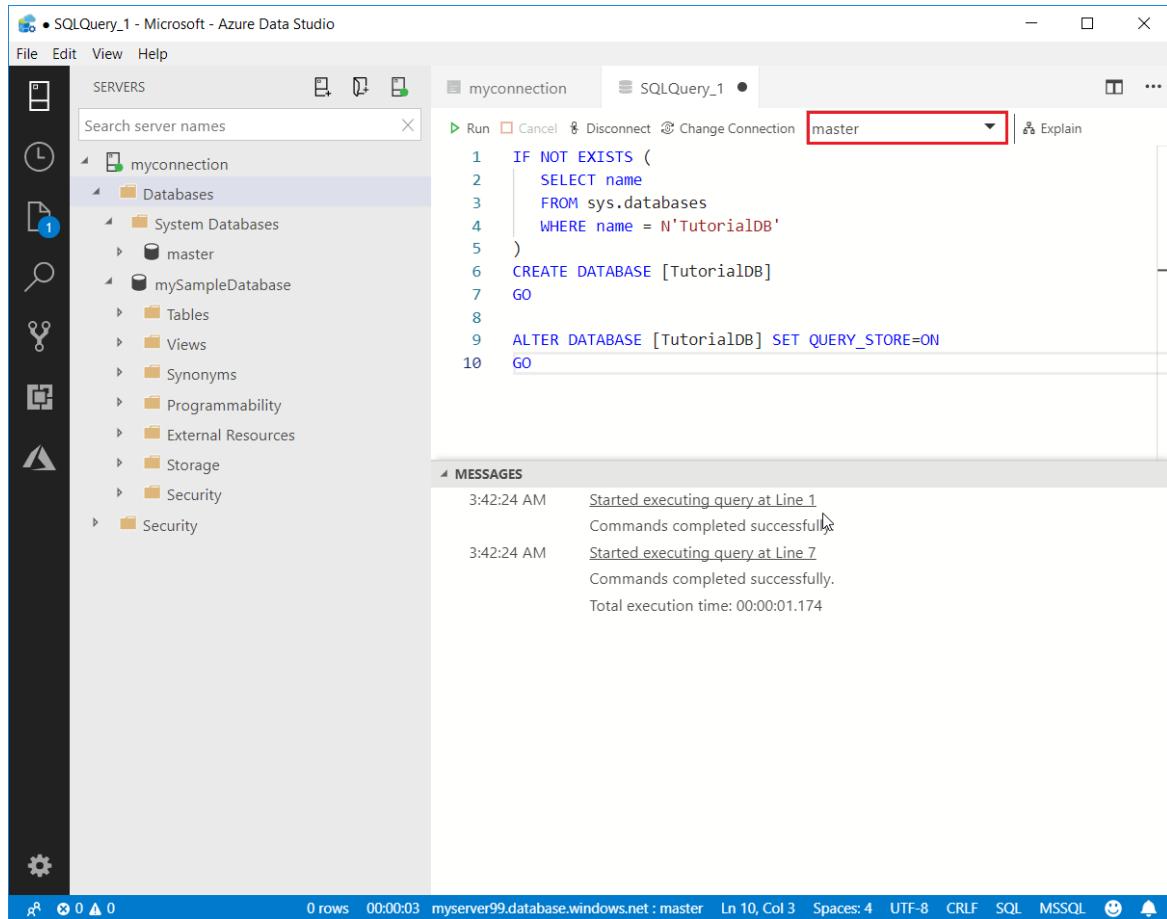
ALTER DATABASE [TutorialDB] SET QUERY_STORE=ON
GO
```

3. From the toolbar, select **Run**. Notifications appear in the **MESSAGES** pane showing query progress.

## Create a table

The query editor is connected to the **master** database, but we want to create a table in the **TutorialDB** database.

1. Connect to the **TutorialDB** database.



2. Create a `Customers` table.

Replace the previous query in the query editor with this one and select **Run**.

```
-- Create a new table called 'Customers' in schema 'dbo'  
-- Drop the table if it already exists  
IF OBJECT_ID('dbo.Customers', 'U') IS NOT NULL  
DROP TABLE dbo.Customers  
GO  
-- Create the table in the specified schema  
CREATE TABLE dbo.Customers  
(  
    CustomerId      INT      NOT NULL  PRIMARY KEY, -- primary key column  
    Name            [NVARCHAR](50) NOT NULL,  
    Location        [NVARCHAR](50) NOT NULL,  
    Email           [NVARCHAR](50) NOT NULL  
);  
GO
```

## Insert rows into the table

Replace the previous query with this one and select **Run**.

```
-- Insert rows into table 'Customers'  
INSERT INTO dbo.Customers  
    ([CustomerId],[Name],[Location],[Email])  
VALUES  
    ( 1, N'Orlando', N'Australia', N''),  
    ( 2, N'Keith', N'India', N'keith0@adventure-works.com'),  
    ( 3, N'Donna', N'Germany', N'donna0@adventure-works.com'),  
    ( 4, N'Janet', N'United States', N'janet1@adventure-works.com')  
GO
```

## View the result

Replace the previous query with this one and select **Run**.

```
-- Select rows from table 'Customers'  
SELECT * FROM dbo.Customers;
```

The query results display:

The screenshot shows the Microsoft Azure Data Studio interface. On the left is a sidebar with icons for servers, databases, tables, views, synonyms, programmability, external resources, storage, security, and security. The 'myconnection' node under 'Databases' is expanded, showing 'System Databases' (master), 'mySampleDatabase' (Tables, Views, Synonyms, Programmability, External Resources, Storage, Security), and 'Security'. The main area has tabs for 'myconnection' and 'SQLQuery\_1'. The 'SQLQuery\_1' tab contains the following SQL code:

```
-- Select rows from table 'Customers'  
SELECT * FROM dbo.Customers;
```

The results pane shows a table with four rows of data:

	CustomerID	Name	Location	Email
1	1	Orlando	Australia	
2	2	Keith	India	keith0@adventure-works.com
3	3	Donna	Germany	donna0@adventure-works.com
4	4	Janet	United States	janet1@adventure-works.com

The messages pane at the bottom shows the following output:

```
3:47:46 AM Started executing query at Line 1  
(4 rows affected)  
Total execution time: 00:00:00.214
```

At the bottom of the interface, there are status indicators for rows (4 rows), execution time (00:00:00), connection (myserver99.database.windows.net : TutorialDB), and other settings.

## Clean up resources

Later quickstart articles build upon the resources created here. If you plan to work through these articles, be sure not to delete these resources. Otherwise, in the Azure portal, delete the resources you no longer need. For details, see [Clean up resources](#).

## Next steps

Now that you've successfully connected to an Azure SQL database and run a query, try the [Code editor tutorial](#).

# Quickstart: Use Azure Data Studio to connect and query data using a dedicated SQL pool in Azure Synapse Analytics

9/26/2022 • 4 minutes to read • [Edit Online](#)

This quickstart shows connecting to a dedicated SQL pool in Azure Synapse Analytics using Azure Data Studio.

## Prerequisites

To complete this quickstart, you need Azure Data Studio, and a dedicated SQL pool in Azure Synapse Analytics.

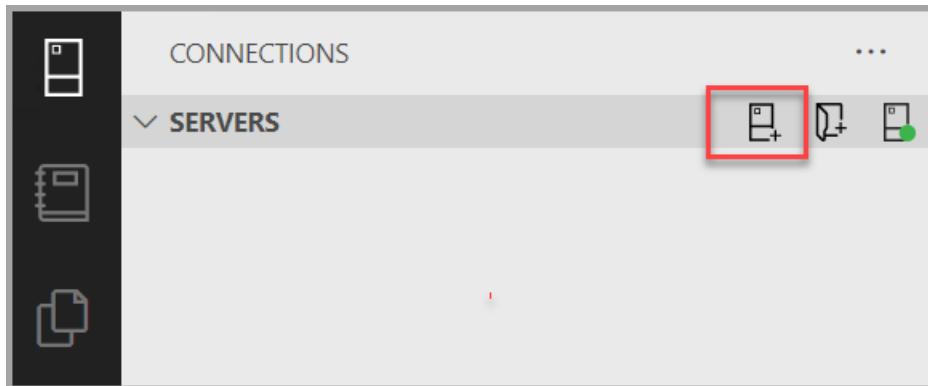
- [Install Azure Data Studio](#).

If you don't already have a dedicated SQL pool in Azure Synapse Analytics workspace, see [Quickstart: Create a dedicated SQL pool using Synapse Studio](#). For a standalone dedicated SQL pool (formerly SQL DW) outside of an Azure Synapse workspace, see [Create a dedicated SQL pool](#). Remember the server name, and login credentials! For more information on the differences, see [What's the difference between Azure Synapse dedicated SQL pools \(formerly SQL DW\) and dedicated SQL pools in an Azure Synapse Analytics workspace?](#).

## Connect to your dedicated SQL pool

Use Azure Data Studio to establish a connection to your Azure Synapse Analytics server.

1. The first time you run Azure Data Studio the **Connection** page should open. If you don't see the **Connection** page, select **Add Connection**, or the **New Connection** icon in the **SERVERS** sidebar:



2. This article uses *SQL Login*, but *Windows Authentication* is also supported. Fill in the fields as follows using the server name, user name, and password for *your* Azure SQL server:

SETTING	SUGGESTED VALUE	DESCRIPTION

SETTING	SUGGESTED VALUE	DESCRIPTION
Server name	The fully qualified server name	<p>For example the name should look like to this:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">myworkspace.sql.azuresynapse.net</div> <p>for a dedicated SQL endpoint in an Azure Synapse Analytics workspace (recommended), or for a standalone dedicated SQL pool (formerly SQL DW) outside of an Azure Synapse Analytics workspace, use the Azure SQL logical server:</p> <div style="border: 1px solid black; padding: 2px; display: inline-block;">sqlpoolservername.database.windows.net</div>
Authentication	SQL Login	SQL Authentication is used in this tutorial.
User name	The server admin account	This is the account that you specified when you created the server.
Password (SQL Login)	The password for your server admin account	This is the password that you specified when you created the server.
Save Password?	Yes or No	Select Yes if you don't want to enter the password each time.
Database name	<i>leave blank</i>	
Server Group	<i>leave blank</i>	

- If your server doesn't have a firewall rule allowing Azure Data Studio to connect, the **Create new firewall rule** form opens. Complete the form to create a new firewall rule. For details, see [Firewall rules](#).
- After successfully connecting your server opens in the *Servers* sidebar.

## Create a database

You can use an existing dedicated SQL pool for this exercise, including the one created by default, or you can [create a new pool](#).

For dedicated SQL pools in Azure Synapse, navigate to the Azure Synapse Analytics workspace.

- Select **Manage**.
- In the **Analytics pools** section, select **SQL pools**.
- Select **+ New** to create a new dedicated SQL pool.
- Give the dedicated SQL pool a new name 

TutorialDB

 and pick a performance level. Set to the smallest size to reduce costs for this quickstart.
- Select **Create**.

For standalone dedicated SQL pools (formerly SQL DW), navigate to the **Dedicated SQL pool (formerly SQL DW)** in the Azure portal.

- Select the link for **Server name**, such as 

contosowdw.database.windows.net

, to go to the Azure portal page for the logical SQL server.

2. Select + **Create dedicated SQL pool** (formerly **SQL DW**) to create another dedicated SQL pool in the logical SQL server.
3. Give the dedicated SQL pool a new name `TutorialDB` and pick a performance level. Set to the smallest size to reduce costs for this quickstart.
4. Select **Review + Create**.

## Create a table

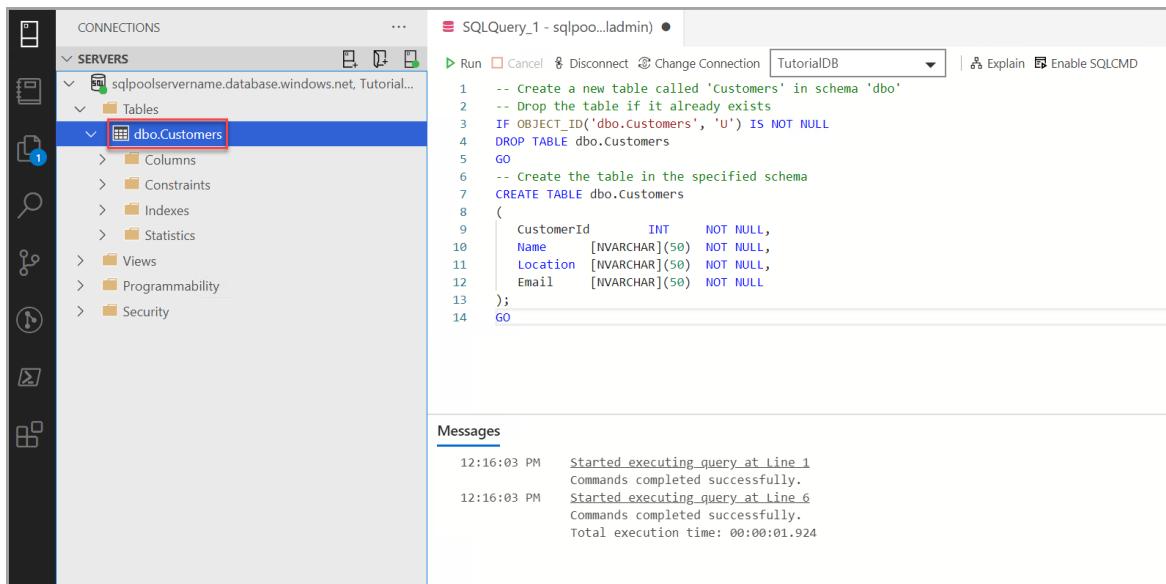
The query editor is still connected to the `master` database, but we want to create a table in the `TutorialDB` database.

1. Change the connection context to `TutorialDB`:
2. Paste the following snippet into the query editor and select **Run**:

### NOTE

You can append this to, or overwrite the previous query in the editor. Note that selecting **Run** executes only the query that is selected. If nothing is selected, selecting **Run** executes all queries in the editor.

```
-- Create a new table called 'Customers' in schema 'dbo'
-- Drop the table if it already exists
IF OBJECT_ID('dbo.Customers', 'U') IS NOT NULL
DROP TABLE dbo.Customers
GO
-- Create the table in the specified schema
CREATE TABLE dbo.Customers
(
    CustomerId      INT      NOT NULL,
    Name            [NVARCHAR](50) NOT NULL,
    Location        [NVARCHAR](50) NOT NULL,
    Email           [NVARCHAR](50) NOT NULL
);
GO
```



## Insert rows

1. Paste the following snippet into the query editor and select **Run**:

```
-- Insert rows into table 'Customers'
INSERT INTO dbo.Customers
([CustomerId],[Name],[Location],[Email])
SELECT 1, N'Orlando',N'Australia', N'' UNION ALL
SELECT 2, N'Keith', N'India', N'keith0@adventure-works.com' UNION ALL
SELECT 3, N'Donna', N'Germany', N'donna0@adventure-works.com' UNION ALL
SELECT 4, N'Janet', N'United States', N'janet1@adventure-works.com'
```

## View the result

- Paste the following snippet into the query editor and select **Run**:

```
-- Select rows from table 'Customers'
SELECT * FROM dbo.Customers;
```

- The results of the query are displayed:

	CustomerId	Name	Location	Email
1	1	Orlando	Australia	keith0@adventure-works.com
2	2	Keith	India	
3	3	Donna	Germany	donna0@adventure-works.com
4	4	Janet	United States	janet1@adventure-works.com

"/>

## Clean up resources

If you don't plan to continue working with the sample databases created in this article, then [delete the resource group](#).

## Next steps

For more information, visit [Connecting to Synapse SQL with Azure Data Studio](#).

Now that you've successfully connected to an Azure Synapse Analytics and ran a query, try out the [Code editor tutorial](#).

# Quickstart: Use Azure Data Studio to connect and query Azure Cosmos DB API for MongoDB (Preview)

9/26/2022 • 3 minutes to read • [Edit Online](#)

This quickstart shows how to use Azure Data Studio to connect to the Azure Cosmos DB API for MongoDB.

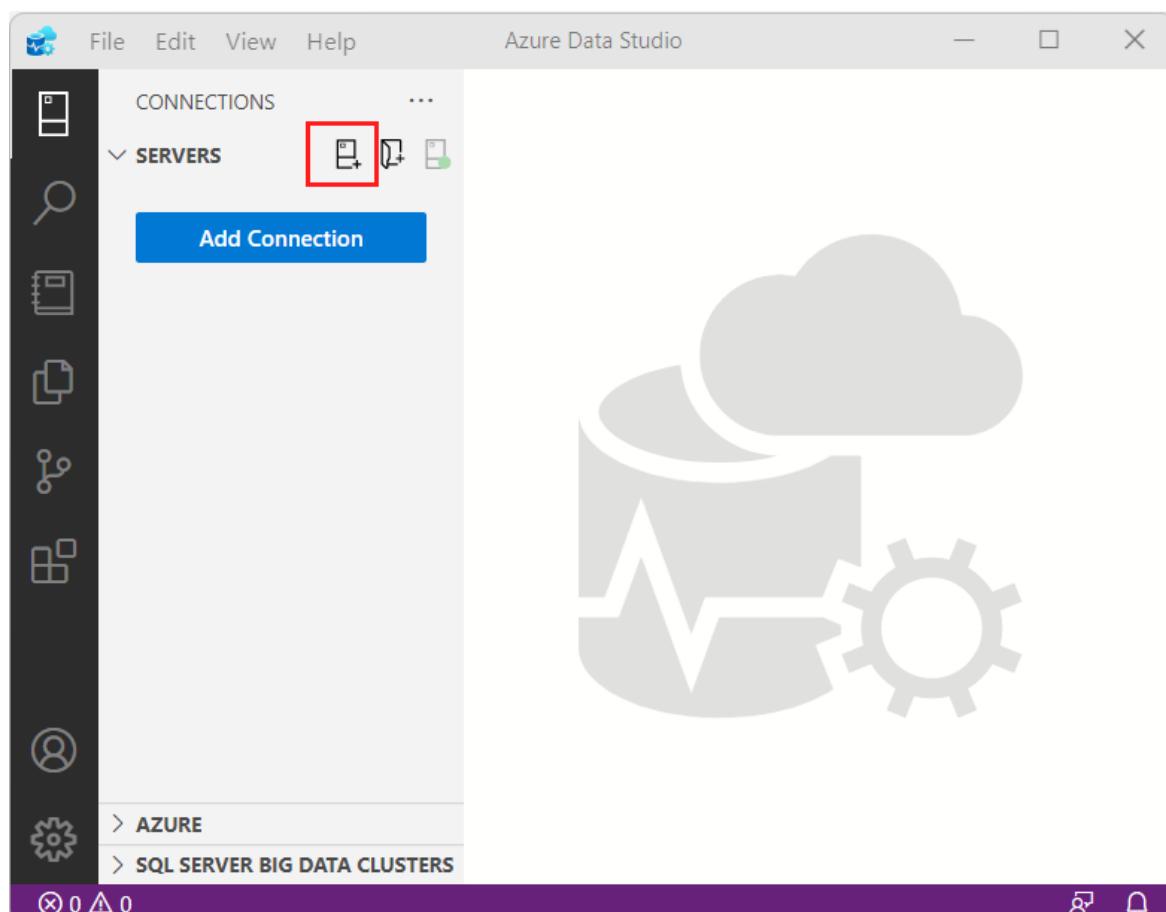
## Prerequisites

To complete this quickstart, you need Azure Data Studio and an Azure Cosmos DB API account.

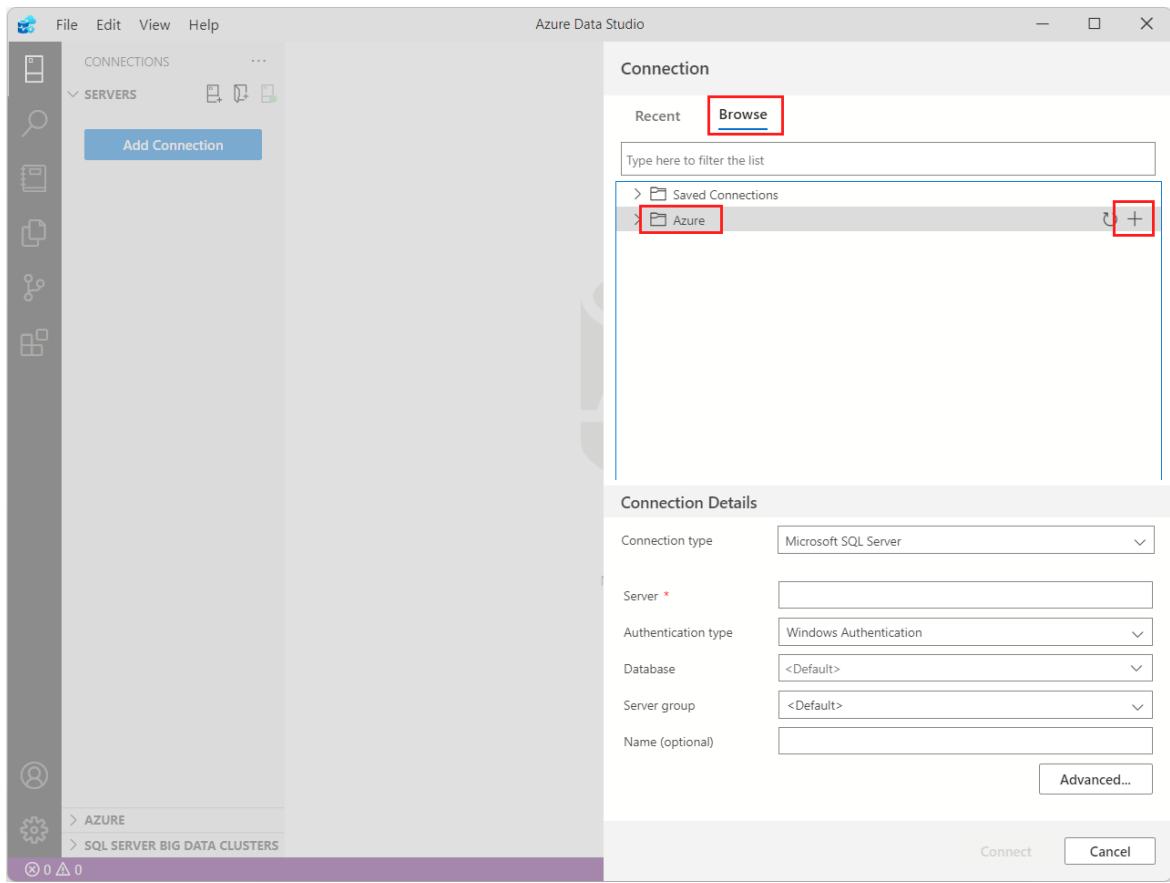
- [Install Azure Data Studio](#).
- [Install the Azure Cosmos DB API for MongoDB extension for Azure Data Studio](#).
- [Create an Azure Cosmos DB API for MongoDB account](#).

## Connect to an Azure Cosmos DB API for MongoDB account

1. Start **Azure Data Studio**.
2. The first time you start Azure Data Studio the **Connection** dialog opens. If the **Connection** dialog doesn't open, select the **New Connection** icon in the **SERVERS** page:

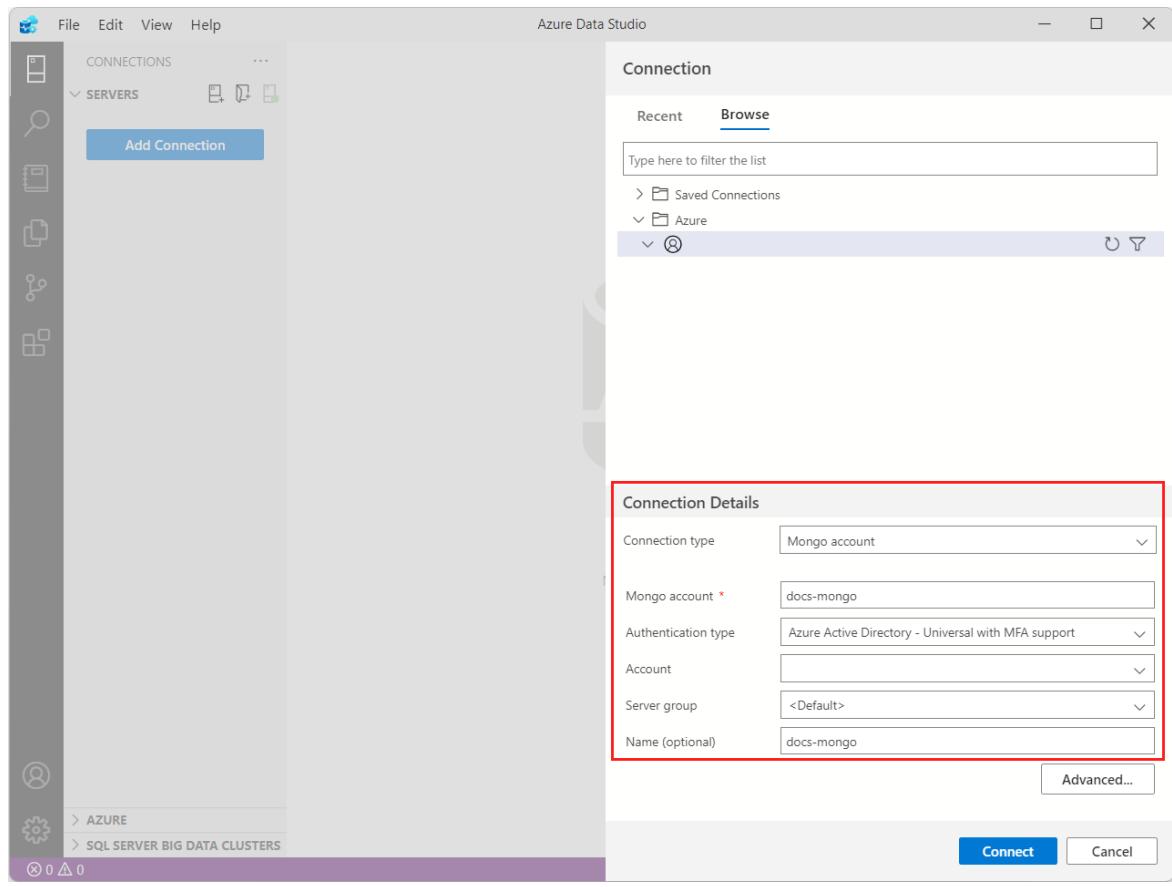


3. In the dialog that appears, select **Browse**, and then select the + icon to sign in to Azure.



4. Once you're signed in to Azure, select the Azure Cosmos DB account you'd like to access or fill in the **Connection Details** from fields using the details in this table.

SETTING	VALUE	DESCRIPTION
<b>Connection type</b>	<i>Mongo account</i>	Set this value to <i>Mongo account</i> to use with Azure Cosmos DB API for MongoDB.
<b>Mongo account</b>	<i>[cosmos-account-name]</i>	Name of Azure Cosmos DB account.
<b>Authentication type</b>	<i>Azure Active Directory - Universal with MFA support</i>	Choose between <i>Azure Active Directory</i> or <i>Basic</i> authentication options.
<b>Account</b>	<i>[current-aad-user-name]</i>	Only visible if using <i>Azure Active Directory</i> authentication. Select the Azure AD account that you wish to use for authentication.
<b>Username</b>		Only visible if using <i>Basic</i> authentication. Enter the username for authentication.
<b>Connection String</b>		Only visible if using <i>Basic</i> authentication. Enter the connection string for authentication.
<b>Server group</b>	<i>&lt;Default&gt;</i>	
<b>Name (optional)</b>		Enter any unique name to use in the SERVERS sidebar.



## 5. Select Connect.

### IMPORTANT

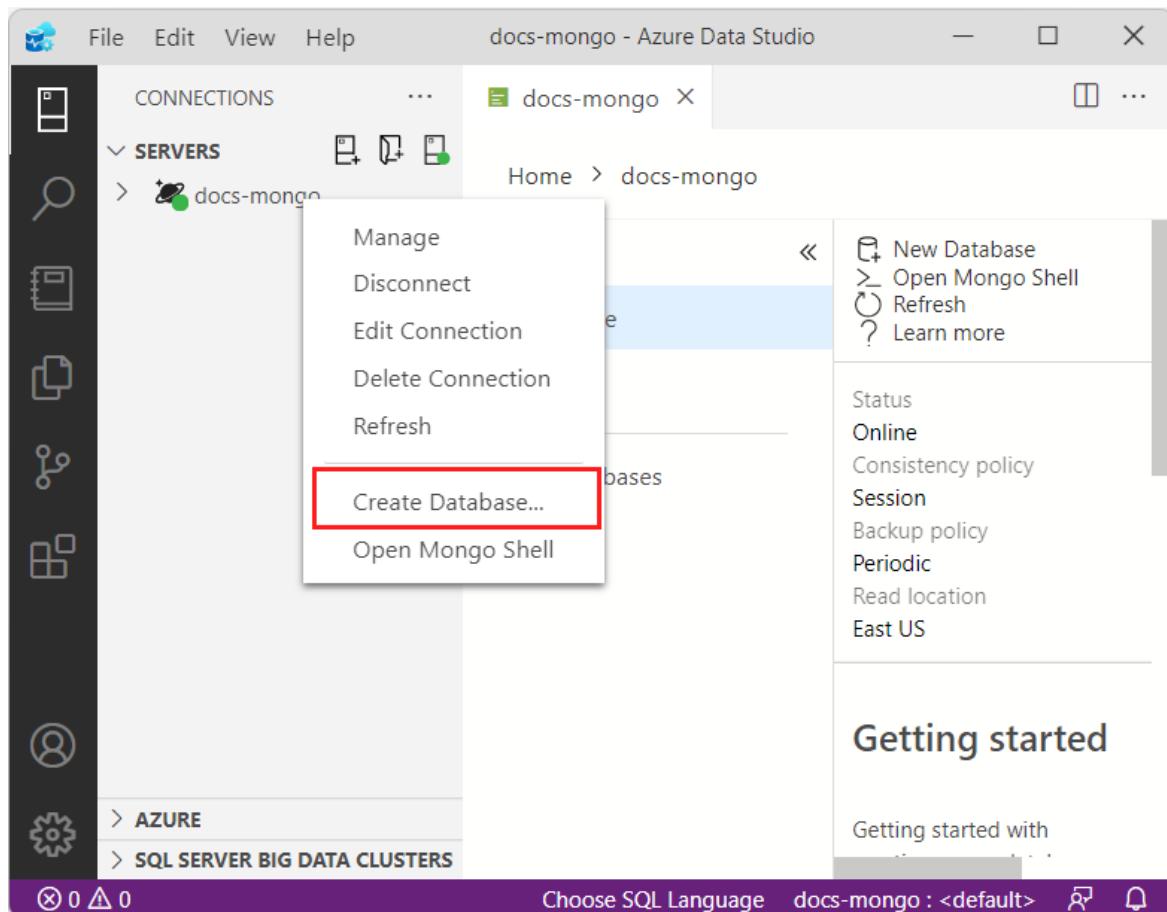
If you selected *Azure Active Directory* authentication, you may be prompted to select a MongoDB connection string value to use for your connection.

Your server will open in the SERVERS sidebar after you've successfully connected.

## Create a database and collection

The following steps will create a database named **SourceDatabase** and a collection named **People**:

1. Open the context menu for your account, and select **Create Database**.



2. In the **Database** and **Collection** popup dialogs, use the details in this table.

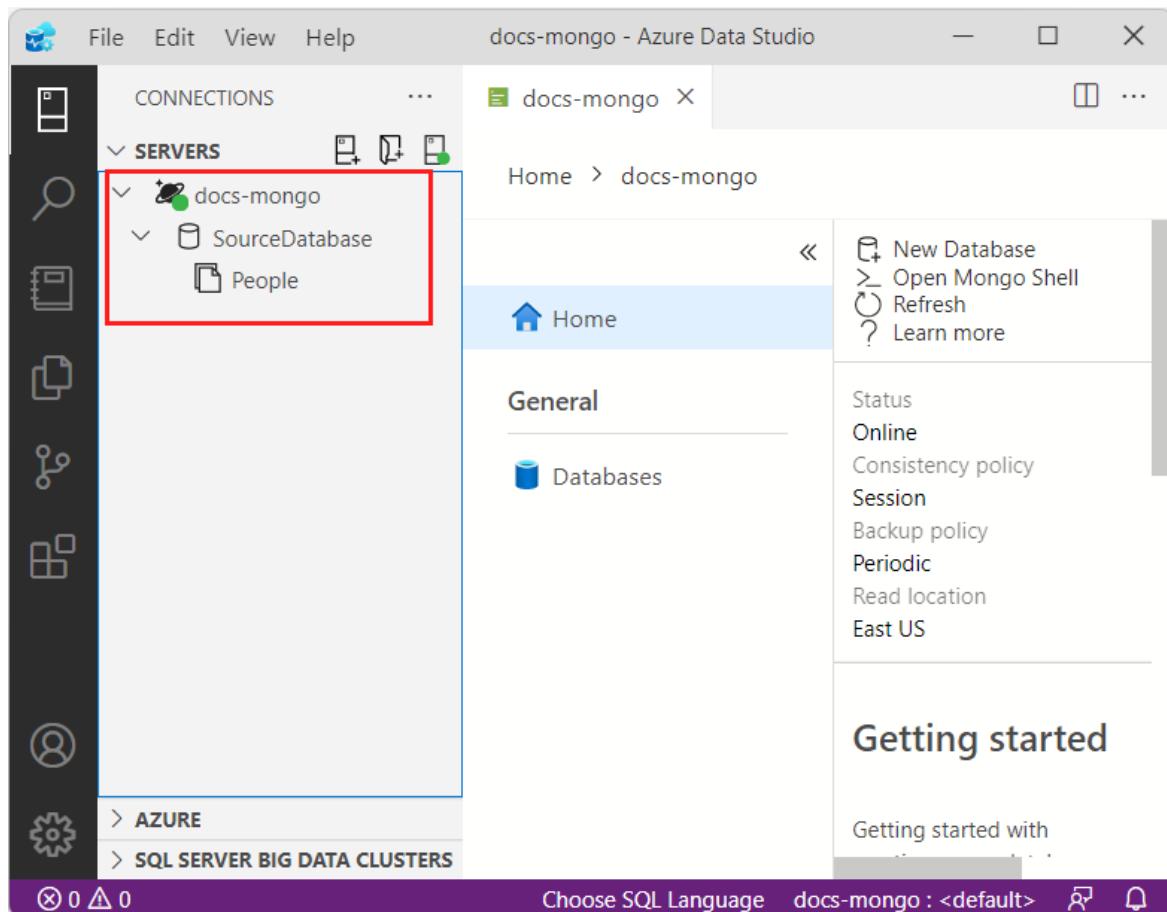
PROMPT	VALUE
Database name	<i>SourceDatabase</i>
Collection name	<i>People</i>

3. After the operation completes, the new database should appear in the list of databases.

**TIP**

If the database does not appear in the list, select **Refresh**.

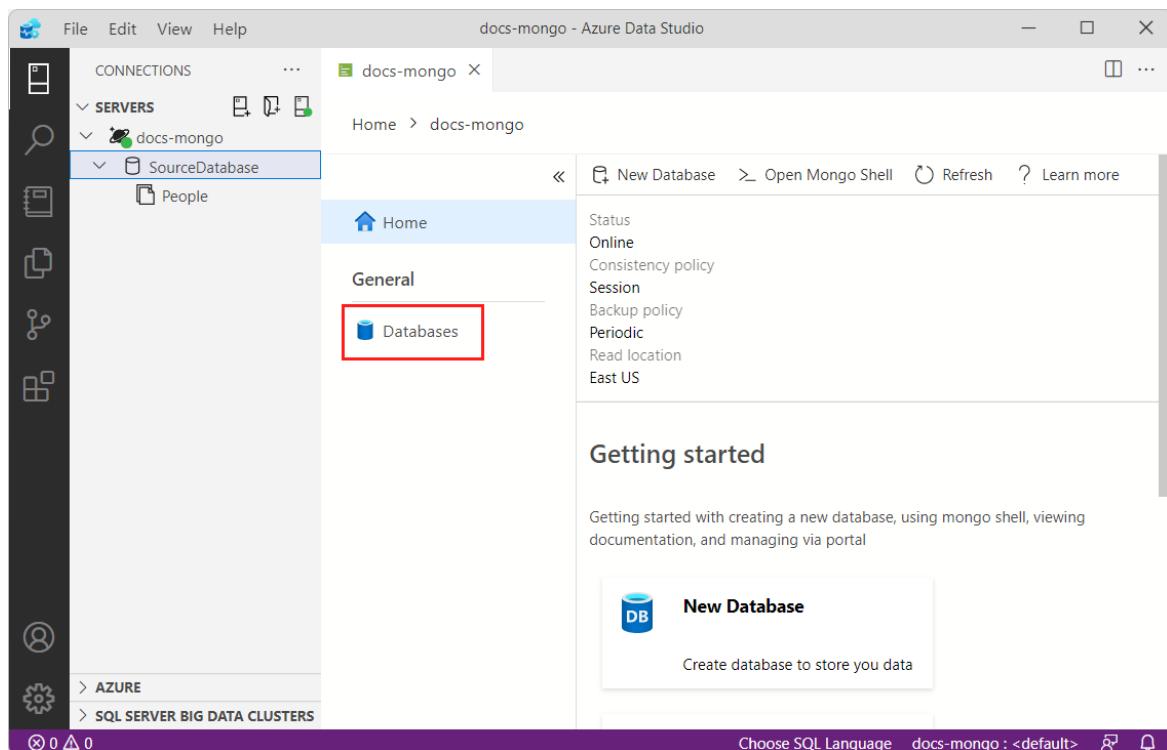
4. Expand the **SourceDatabase** and **People** nodes in the **SERVERS** sidebar.



## Create a sample collection

The following steps will populate the **People** collection with a sample data set:

1. Select **Databases** to navigate to the list of databases in your account.



2. Select the **SourceDatabase** item in the list of databases.

The screenshot shows the Azure Data Studio interface for a database named 'docs-mongo'. In the left sidebar, under 'CONNECTIONS' and 'SERVERS', 'docs-mongo' is selected. Under 'SourceDatabase', there is a single collection named 'People'. The main pane displays the 'Database overview' with a table:

Database	Data Usage ...	Collections	Throughput Sh...
SourceDatabase	0	1	

A red box highlights the 'SourceDatabase' row.

3. In the header menu, select **Import Sample Data**. In the confirmation dialog, select **Yes**.

The screenshot shows the Azure Data Studio interface for the 'SourceDatabase' database. In the left sidebar, 'SourceDatabase' is selected. In the center, there are two options: 'New Collection' and 'Import Sample Data'. A red box highlights the 'Import Sample Data' button.

4. Wait for the import operation to complete.

**TIP**

The import operation may take a few minutes to finish.

## Query the data

The following steps will use the **Mongo Shell** to execute a query and view the results of the query:

1. In the menu for your database, select **Open Mongo Shell**.

The screenshot shows the Azure Data Studio interface with the 'SourceDatabase' connection selected. In the left sidebar, under 'COLLECTIONS', the 'Persons' collection is highlighted. At the top right, there is a red box around the 'Open Mongo Shell' button. Below the collections list, there are two cards: 'New Collection' (Create a new collection to store your data) and 'Import Sample Data' (Create a new collection using one of our sample datasets). The main area is titled 'Collection overview' with the sub-instruction 'Click on a collection to work with the data'. A table below shows collection statistics:

Collection	Data Usage (KB)	Documents	Throughput
Persons	0	Unknown	
People	0	Unknown	

At the bottom, the status bar shows 'docs-mongo : <default>'.

2. Execute the following query in the shell.

```
db.Persons.find().pretty()
```

**TIP**

The `pretty()` method in Mongo displays the results of a query in a format that's easy to read. To learn more, see the [official MongoDB documentation for cursor.pretty](#).

3. Observe the output of the query within the shell. The output should contain

The screenshot shows the Azure Data Studio interface with the 'SourceDatabase' connection selected. The 'TERMINAL' tab is active, showing the Mongo shell. The command `db.Persons.find().pretty()` is entered and its output is displayed in a red box. The output shows a single document with fields like '\_id', 'type', 'customer\_id', and 'title'.

## Next steps

In this quickstart, you've learned how to create an API for MongoDB account, create a database and a collection in Azure Data Studio, and query sample data. You can now use Azure Data Studio to review real data in your

database.

- Learn about the [extension available for Azure Cosmos DB API for MongoDB](#).
- Interested in importing MongoDB data into Azure Cosmos DB? See [Migrate MongoDB to Azure Cosmos DB's API for MongoDB](#)
- Trying to do capacity planning for a migration to Azure Cosmos DB? You can use information about your existing database cluster for capacity planning.
  - If all you know is the number of vCores and servers in your existing database cluster, read about [estimating request units using vCores or vCPUs](#)
  - If you know typical request rates for your current database workload, read about [estimating request units using Azure Cosmos DB capacity planner](#)

# Quickstart: Use Azure Data Studio to connect and query PostgreSQL

9/26/2022 • 2 minutes to read • [Edit Online](#)

This quickstart shows how to use Azure Data Studio to connect to PostgreSQL, and then use SQL statements to create the database `tutorialdb` and query it.

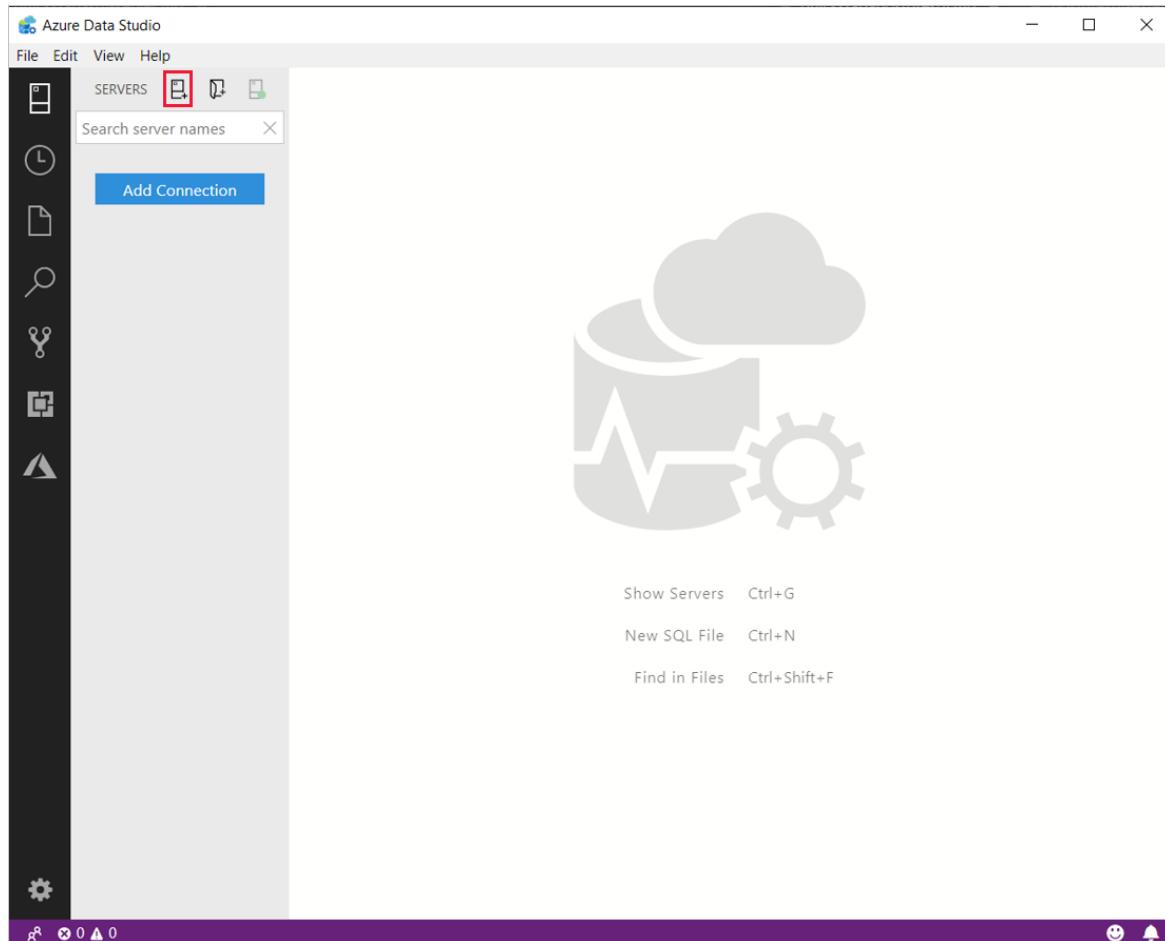
## Prerequisites

To complete this quickstart, you need Azure Data Studio, the PostgreSQL extension for Azure Data Studio, and access to a PostgreSQL server.

- [Install Azure Data Studio](#).
- [Install the PostgreSQL extension for Azure Data Studio](#).
- [Install PostgreSQL](#). (Alternatively, you can create a Postgres database in the cloud using `az postgres up`).

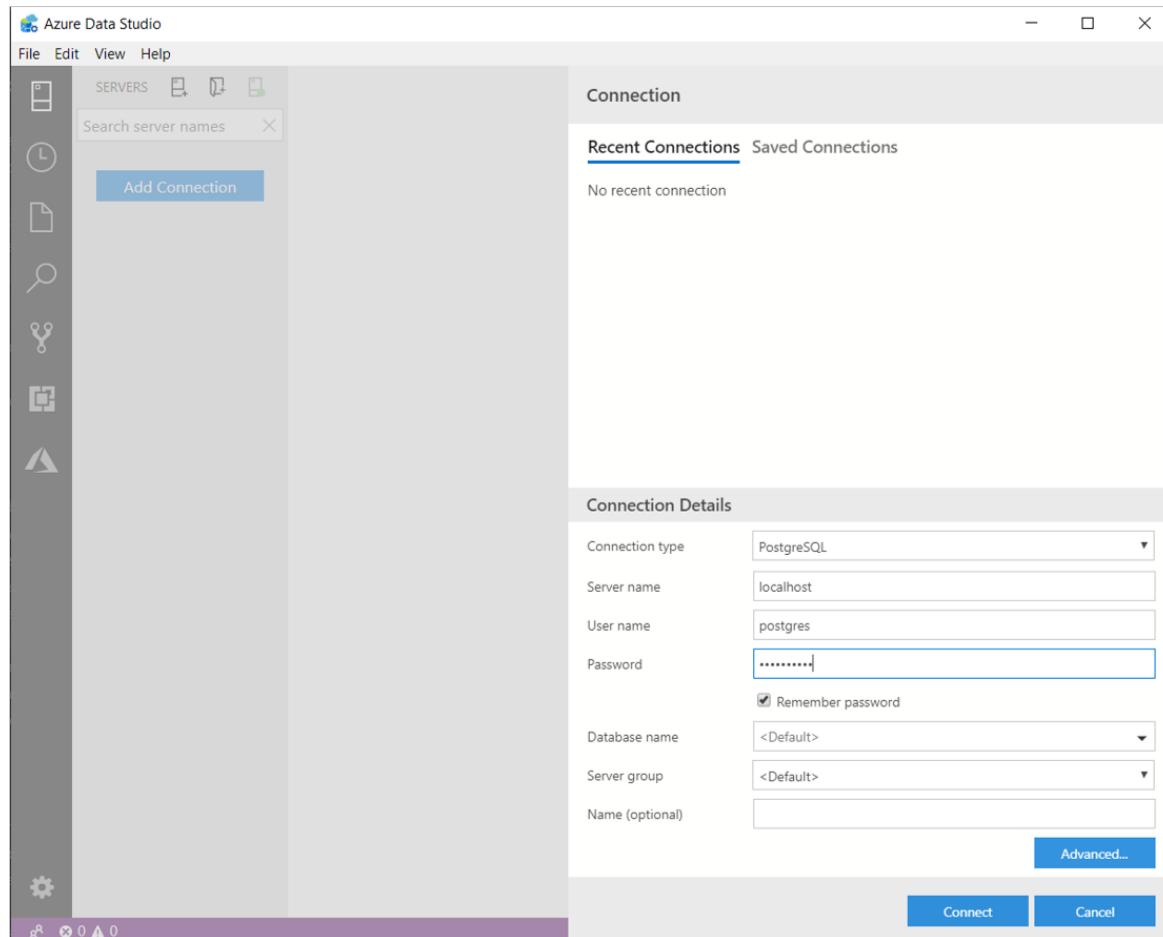
## Connect to PostgreSQL

1. Start **Azure Data Studio**.
2. The first time you start Azure Data Studio the **Connection** dialog opens. If the **Connection** dialog doesn't open, click the **New Connection** icon in the **SERVERS** page:



3. In the form that pops up, go to **Connection type** and select **PostgreSQL** from the drop-down.

4. Fill in the remaining fields using the server name, user name, and password for your PostgreSQL server.



SETTING	EXAMPLE VALUE	DESCRIPTION
Server name	localhost	The fully qualified server name
User name	postgres	The user name you want to log in with.
Password (SQL Login)	<i>password</i>	The password for the account you are logging in with.
Password	<i>Check</i>	Check this box if you don't want to enter the password each time you connect.
Database name	<Default>	Fill this if you want the connection to specify a database.
Server group	<Default>	This option lets you assign this connection to a specific server group you create.
Name (optional)	<i>leave blank</i>	This option lets you specify a friendly name for your server.

5. Select **Connect**.

After successfully connecting, your server opens in the SERVERS sidebar.

## Create a database

The following steps create a database named **tutorialdb**:

1. Right-click on your PostgreSQL server in the SERVERS sidebar and select **New Query**.
2. Paste this SQL statement in the query editor that opens up.

```
CREATE DATABASE tutorialdb;
```

3. From the toolbar select **Run** to execute the query. Notifications appear in the **MESSAGES** pane to show query progress.

### TIP

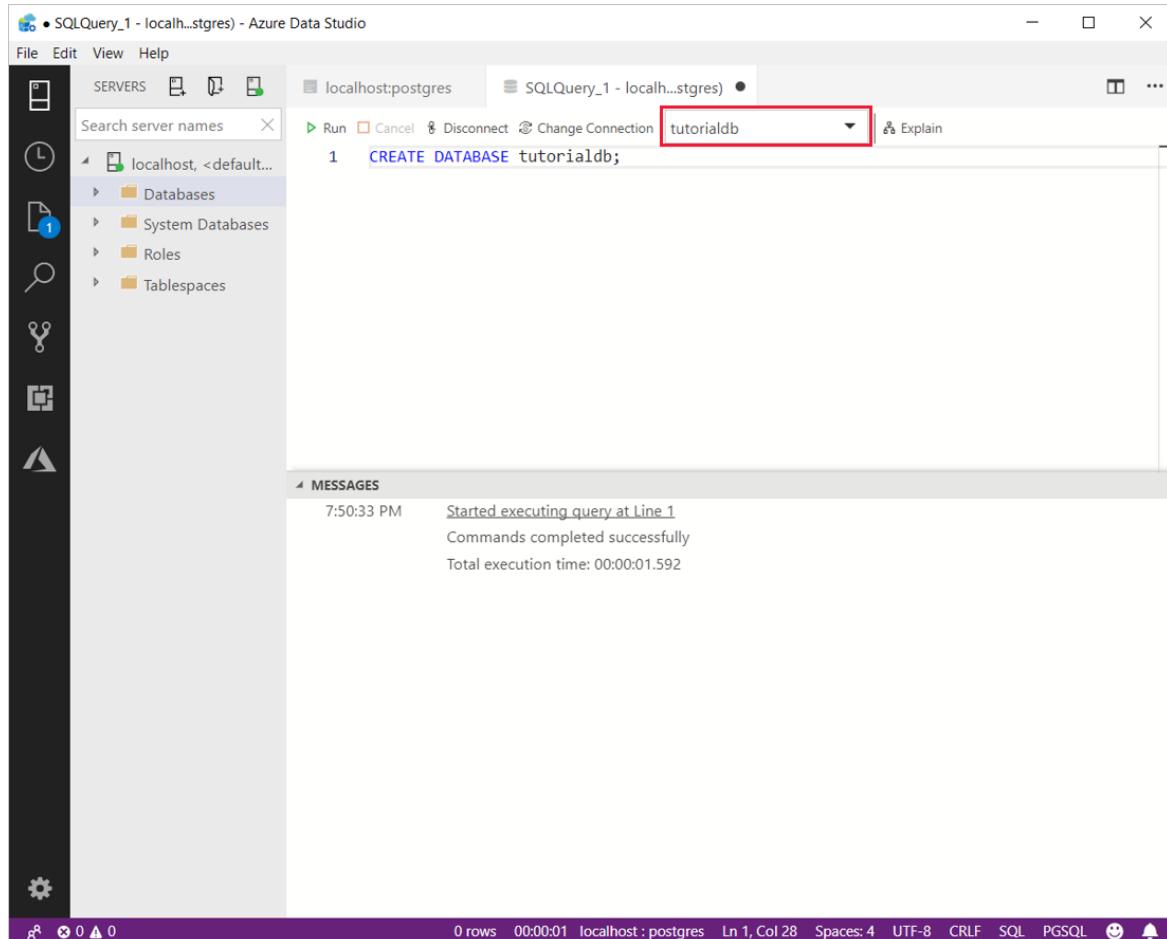
You can use **F5** on your keyboard to execute the statement instead of using **Run**.

After the query completes, right-click **Databases** and select **Refresh** to see **tutorialdb** in the list under the **Databases** node.

## Create a table

The following steps create a table in the **tutorialdb**:

1. Change the connection context to **tutorialdb** using the drop-down in the query editor.



2. Paste the following SQL statement into the query editor and click **Run**.

## NOTE

You can either append this or overwrite the existing query in the editor. Clicking **Run** executes only the query that is highlighted. If nothing is highlighted, clicking **Run** executes all queries in the editor.

```
-- Drop the table if it already exists
DROP TABLE IF EXISTS customers;
-- Create a new table called 'customers'
CREATE TABLE customers(
    customer_id SERIAL PRIMARY KEY,
    name VARCHAR (50) NOT NULL,
    location VARCHAR (50) NOT NULL,
    email VARCHAR (50) NOT NULL
);
```

## Insert rows

Paste the following snippet into the query window and click **Run**:

```
-- Insert rows into table 'customers'
INSERT INTO customers
    (customer_id, name, location, email)
VALUES
    ( 1, 'Orlando', 'Australia', ''),
    ( 2, 'Keith', 'India', 'keith0@adventure-works.com'),
    ( 3, 'Donna', 'Germany', 'donna0@adventure-works.com'),
    ( 4, 'Janet', 'United States','janet1@adventure-works.com');
```

## Query the data

1. Paste the following snippet into the query editor and click **Run**:

```
-- Select rows from table 'customers'
SELECT * FROM customers;
```

2. The results of the query are displayed:

A screenshot of the Azure Data Studio interface. The top navigation bar shows 'File', 'Edit', 'View', and 'Help'. Below it is a 'SERVERS' sidebar with a tree view of database objects: 'localhost, <default...>' expanded to show 'Databases', 'System Databases', 'Roles', and 'Tablespaces'. The main area has tabs for 'localhost:postgres' and 'SQLQuery\_1 - localhost:postgres'. A toolbar above the query editor includes 'Run', 'Cancel', 'Disconnect', 'Change Connection' (set to 'tutorialdb'), 'Explain', and other icons. The query editor contains the following SQL:

```
-- Select rows from table 'customers'  
SELECT * FROM customers;
```

The results pane, titled 'RESULTS', displays a table with four rows of data:

	customer_id	name	location	email
1	1	Orlando	Australia	NULL
2	2	Keith	India	keith0@adve...
3	3	Donna	Germany	donna0@adve...
4	4	Janet	United States	janet1@adve...

The messages pane at the bottom shows the execution log:

```
8:02:50 PM Started executing query at Line 1  
(4 row(s) affected)  
Total execution time: 00:00:00.039
```

At the bottom of the interface, there are status indicators (0 errors, 0 warnings), connection information ('localhost : tutorialdb'), and file format settings ('4 rows', '00:00:00', 'Ln 2, Col 25', 'Spaces: 4', 'UTF-8', 'CRLF', 'SQL', 'PGSQL').

## Next Steps

Learn about the scenarios available for Postgres in [Azure Data Studio](#).

# Quickstart: Use Azure Data Studio to connect and explore Oracle databases

9/26/2022 • 2 minutes to read • [Edit Online](#)

This quickstart shows how to use Azure Data Studio to connect to Oracle database and explore existing database objects.

## NOTE

Currently extension provides a read-only view of the database objects. Running queries is not supported.

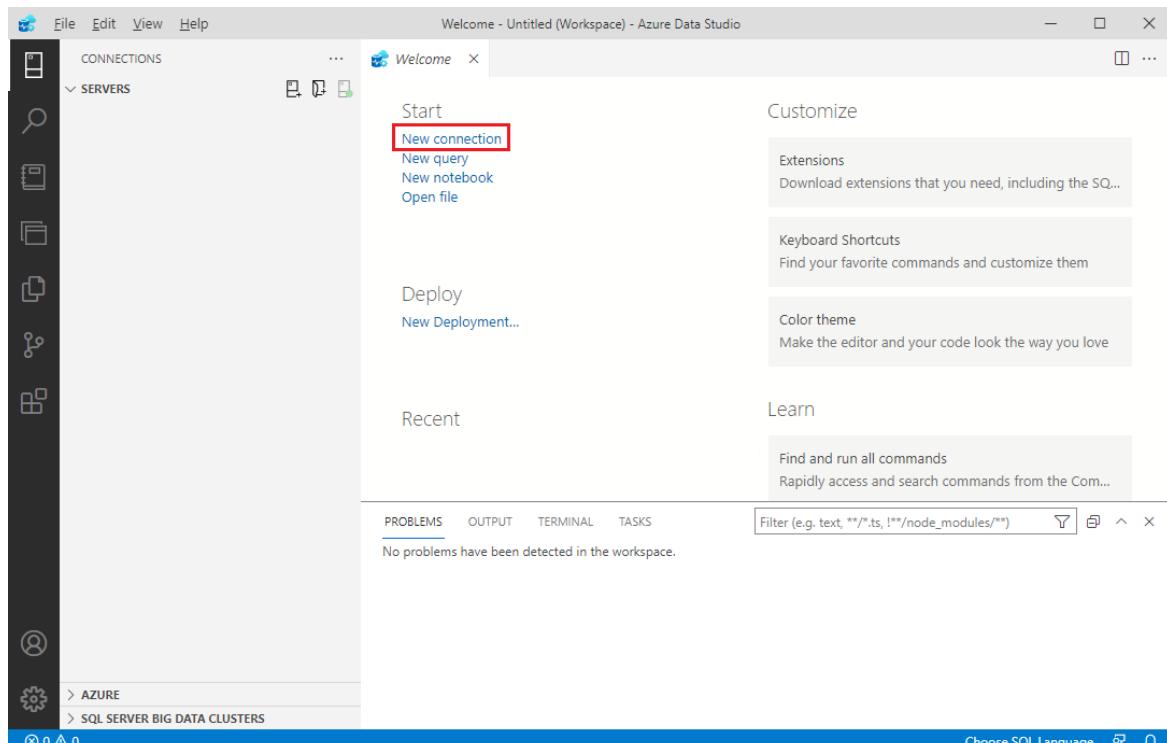
## Prerequisites

To complete this quickstart, you need Azure Data Studio and access to Oracle database.

- [Install Azure Data Studio](#).
- [Install the Extension for Oracle](#).

## Connect to an Oracle database

1. Start [Azure Data Studio](#).
2. The first time you run Azure Data Studio the **Welcome** page should open. If you don't see the **Welcome** page, select **Help > Welcome**. Select **New Connection** to open the **Connection** pane:



3. In the connection dialog, select **Oracle** as **Connection type** and fill in the following fields:

- **Data source:** Enter server name, for example, `localhost:1521`. **Data source** can be defined in three different ways: *Easy Connect/Easy Connect Plus*, *TNS Alias* or *Connect Descriptor*. For more information on the format of each method, please refer to the Oracle documentation on **Data**

Source attribute of the ODP.NET connection string.

**NOTE**

In order to use *TNS Alias* as a data source, make sure that `ORACLE_HOME` environment variable is set and `TNSNAMES.ORA` file is located in the `<ORACLE_HOME>/network/admin` folder.

- **User Id:** User name for the Oracle database
- **Password:** Password for the Oracle database
- **Server Group:** <Default>

**Connection**

Recent      Browse

No recent connection

**Connection Details**

Connection type	Oracle
Data Source	localhost
User Id	SYSTEM
Password	*****
<input type="checkbox"/> Remember password	
Server group	<Default>
Name (optional)	

Advanced...

**Connect**

**Cancel**

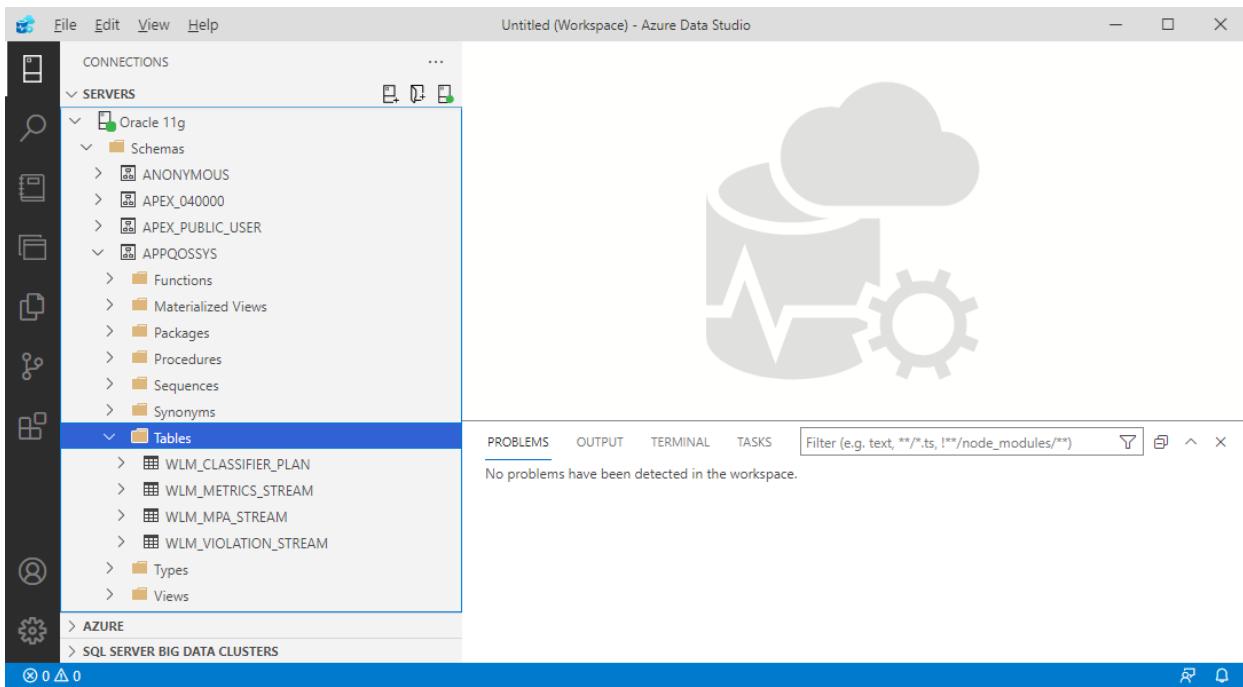
**NOTE**

Make sure that the Oracle database user used to connect has permissions to enumerate and access objects you are planning to explore. At least `CONNECT` and `SELECT ANY DICTIONARY` permissions should be granted to the connecting user.

4. Click **Connect** to establish the connection.

## Explore database objects

After connection to the Oracle database is established, you will be able to navigate and explore existing database objects using **Connections** view:



# Tutorial: Use the Transact-SQL editor to create database objects - Azure Data Studio

9/26/2022 • 5 minutes to read • [Edit Online](#)

Creating and running queries, stored procedures, scripts, etc. are the core tasks of database professionals. This tutorial demonstrates the key features in the T-SQL editor to create database objects.

In this tutorial, you learn how to use Azure Data Studio to:

- Search database objects
- Edit table data
- Use snippets to quickly write T-SQL
- View database object details using *Peek Definition* and *Go to Definition*

## Prerequisites

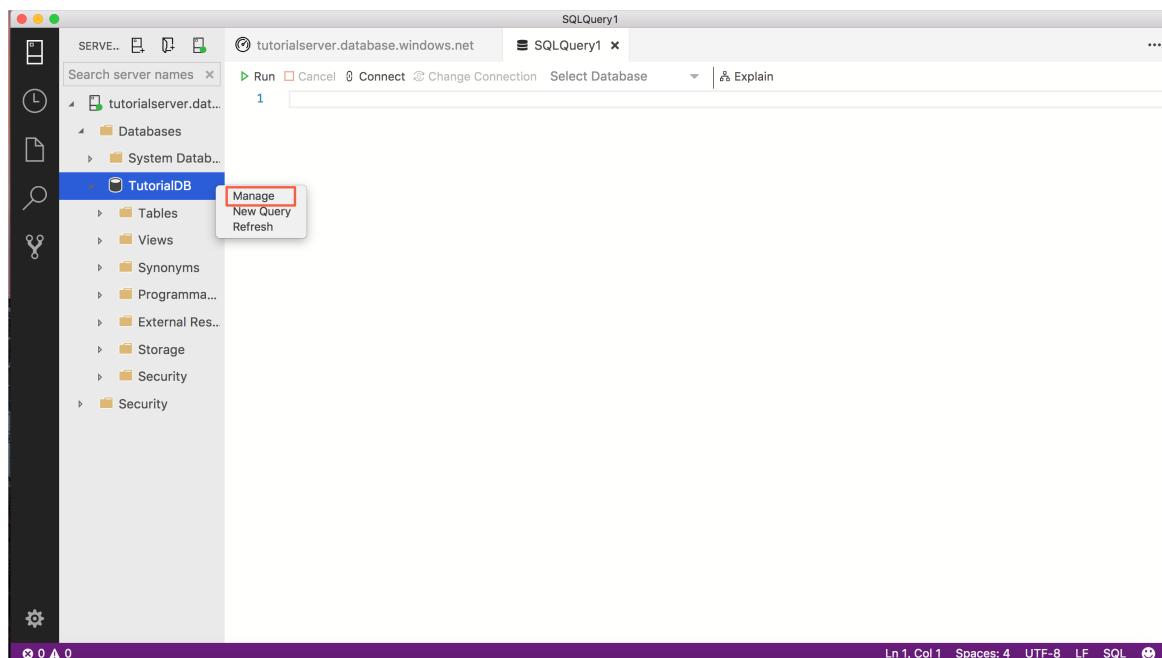
This tutorial requires the SQL Server or Azure SQL Database *TutorialDB*. To create the *TutorialDB* database, complete one of the following quickstarts:

- [Connect and query SQL Server using Azure Data Studio](#)
- [Connect and query Azure SQL Database using Azure Data Studio](#)

## Quickly locate a database object and perform a common task

Azure Data Studio provides a search widget to quickly find database objects. The results list provides a context menu for common tasks relevant to the selected object, such as *Edit Data* for a table.

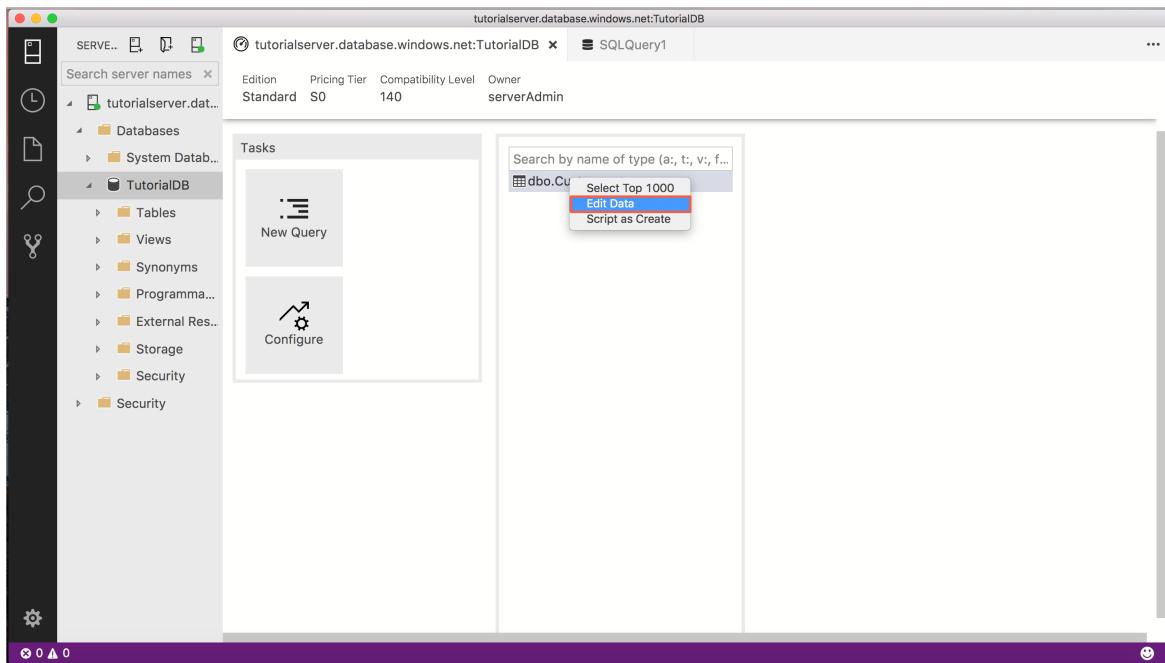
1. Open the SERVERS sidebar (**Ctrl+G**), expand **Databases**, and select **TutorialDB**.
2. Open the *TutorialDB Dashboard* by right-clicking **TutorialDB** and selecting **Manage** from the context menu:



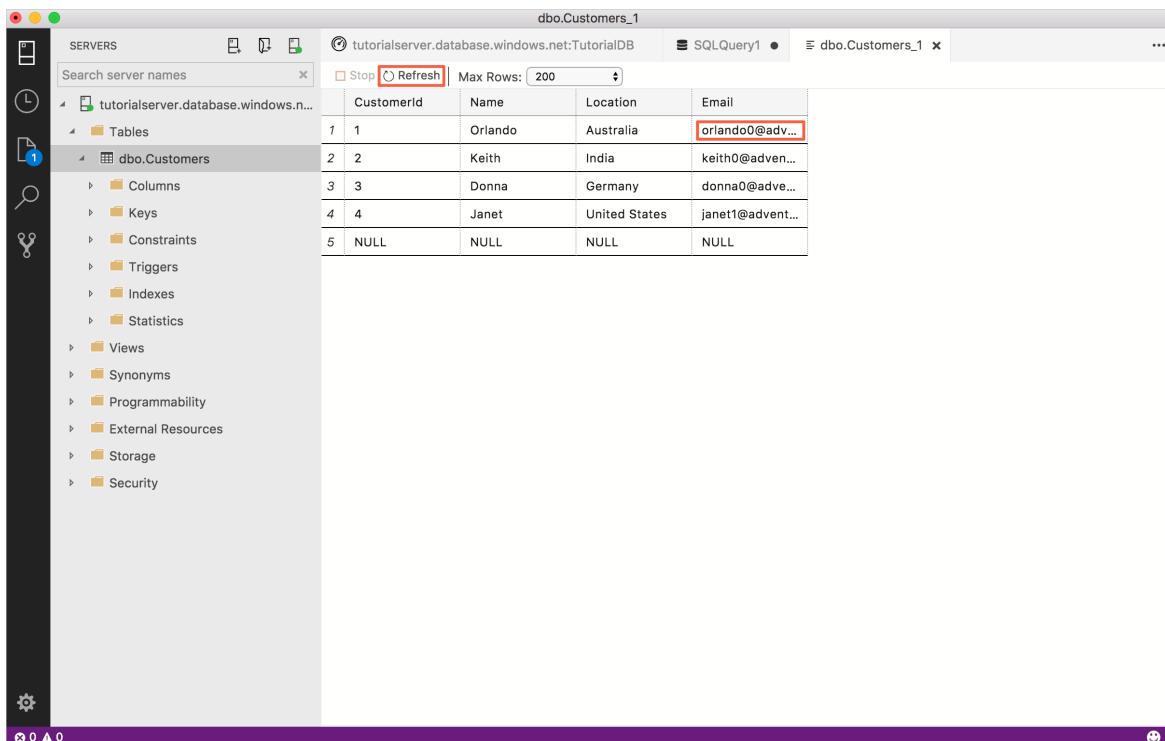
3. On the dashboard, right-click **dbo.Customers** (in the search widget) and select **Edit Data**.

### TIP

For databases with many objects, use the search widget to quickly locate the table, view, etc. that you're looking for.



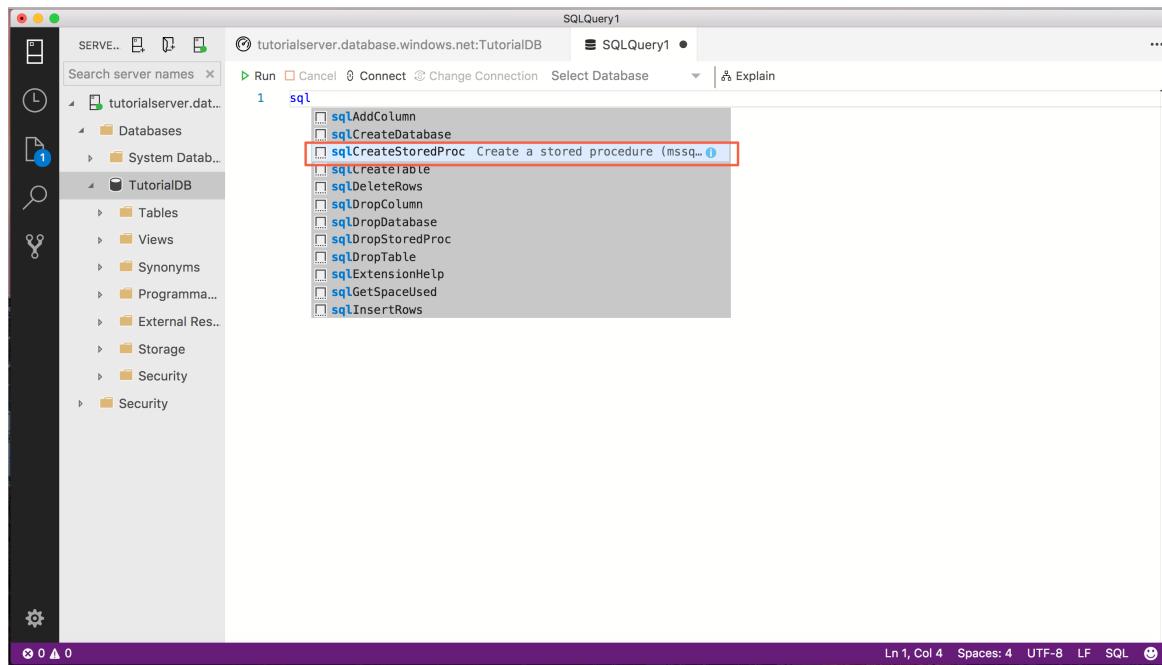
4. Edit the **Email** column in the first row, type *orlando0@adventure-works.com*, and press **Enter** to save the change.



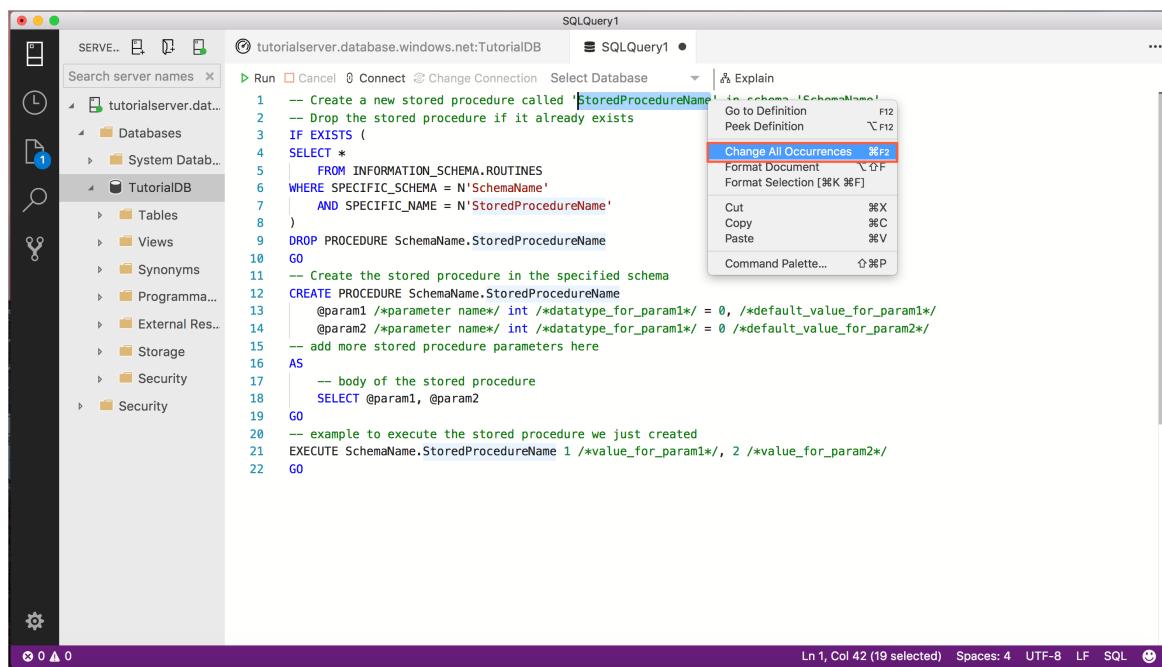
## Use T-SQL snippets to create stored procedures

Azure Data Studio provides many built-in T-SQL snippets for quickly creating statements.

1. Open a new query editor by pressing **Ctrl+N**.
2. Type **sql** in the editor, arrow down to **sqlCreateStoredProcedure**, and press the **Tab** key (or **Enter**) to load the create stored procedure snippet.



3. The create stored procedure snippet has two fields set up for quick edit, *StoredProcedureName* and *SchemaName*. Select *StoredProcedureName*, right-click, and select **Change All Occurrences**. Now type *getCustomer* and all *StoredProcedureName* entries change to *getCustomer*.



4. Change all occurrences of *SchemaName* to *dbo*.
5. The snippet contains placeholder parameters and body text that needs updating. The *EXECUTE* statement also contains placeholder text because it doesn't know how many parameters the procedure will have. For this tutorial update the snippet so it looks like the following code:

```

-- Create a new stored procedure called 'getCustomer' in schema 'dbo'
-- Drop the stored procedure if it already exists
IF EXISTS (
    SELECT *
    FROM INFORMATION_SCHEMA.ROUTINES
    WHERE SPECIFIC_SCHEMA = N'dbo'
    AND SPECIFIC_NAME = N'getCustomer'
)
DROP PROCEDURE dbo.getCustomer
GO
-- Create the stored procedure in the specified schema
CREATE PROCEDURE dbo.getCustomer
@ID int
-- add more stored procedure parameters here
AS
-- body of the stored procedure
SELECT c.CustomerId,
c.Name,
c.Location,
c.Email
FROM dbo.Customers c
WHERE c.CustomerId = @ID
FOR JSON PATH

GO
-- example to execute the stored procedure we just created
EXECUTE dbo.getCustomer 1
GO

```

6. To create the stored procedure and give it a test run, press F5.

The stored procedure is now created, and the **RESULTS** pane displays the returned customer in JSON. To see formatted JSON, click the returned record.

## Use Peek Definition

Azure Data Studio provides the ability to view an objects definition using the peek definition feature. This section creates a second stored procedure and uses peek definition to see what columns are in a table to quickly create the body of the stored procedure.

1. Open a new editor by pressing **Ctrl+N**.
2. Type *sql*/in the editor, arrow down to *sqlCreateStoredProcedure*, and press the *Tab* key (or *Enter*) to load the create stored procedure snippet.
3. Type in *setCustomer* for *StoredProcName* and *dbo* for *SchemaName*
4. Replace the @param placeholders with the following parameter definition:

```
@json_val nvarchar(max)
```

5. Replace the body of the stored procedure with the following code:

```
INSERT INTO dbo.Customers
```

6. In the *INSERT*line you just added, right-click *dbo.Customers* and select **Peek Definition**.

```

    1  SET ANSI_NULLS ON
    2  GO
    3  SET QUOTED_IDENTIFIER ON
    4  GO
    5  CREATE TABLE [dbo].[Customers](
    6      [CustomerId] [int] NOT NULL,
    7      [Name] [nvarchar](50) NOT NULL,
    8      [Location] [nvarchar](50) NOT NULL,
    9      [Email] [nvarchar](50) NOT NULL
   10 ) ON [PRIMARY]
   11 GO
   12
   13

```

7. The table definition appears so you can quickly see what columns are in the table. Refer to the column list to easily complete the statements for your stored procedure. Finish creating the INSERT statement you added previously to complete the body of the stored procedure, and close the peek definition window:

```

INSERT INTO dbo.Customers (CustomerId, Name, Location, Email)
SELECT CustomerId, Name, Location, Email
FROM OPENJSON (@json_val)
WITH( CustomerId int,
      Name nvarchar(50),
      Location nvarchar(50),
      Email nvarchar(50)
)

```

8. Delete (or comment out) the *EXECUTE* command at the bottom of the query.

9. The entire statement should look like the following code:

```

-- Create a new stored procedure called 'setCustomer' in schema 'dbo'
-- Drop the stored procedure if it already exists
IF EXISTS (
    SELECT *
        FROM INFORMATION_SCHEMA.ROUTINES
        WHERE SPECIFIC_SCHEMA = N'dbo'
        AND SPECIFIC_NAME = N'setCustomer'
)
DROP PROCEDURE dbo.setCustomer
GO
-- Create the stored procedure in the specified schema
CREATE PROCEDURE dbo.setCustomer
    @json_val nvarchar(max)
AS
    -- body of the stored procedure
    INSERT INTO dbo.Customers (CustomerId, Name, Location, Email)
    SELECT CustomerId, Name, Location, Email
    FROM OPENJSON (@json_val)
    WITH( CustomerId int,
          Name nvarchar(50),
          Location nvarchar(50),
          Email nvarchar(50)
)
GO

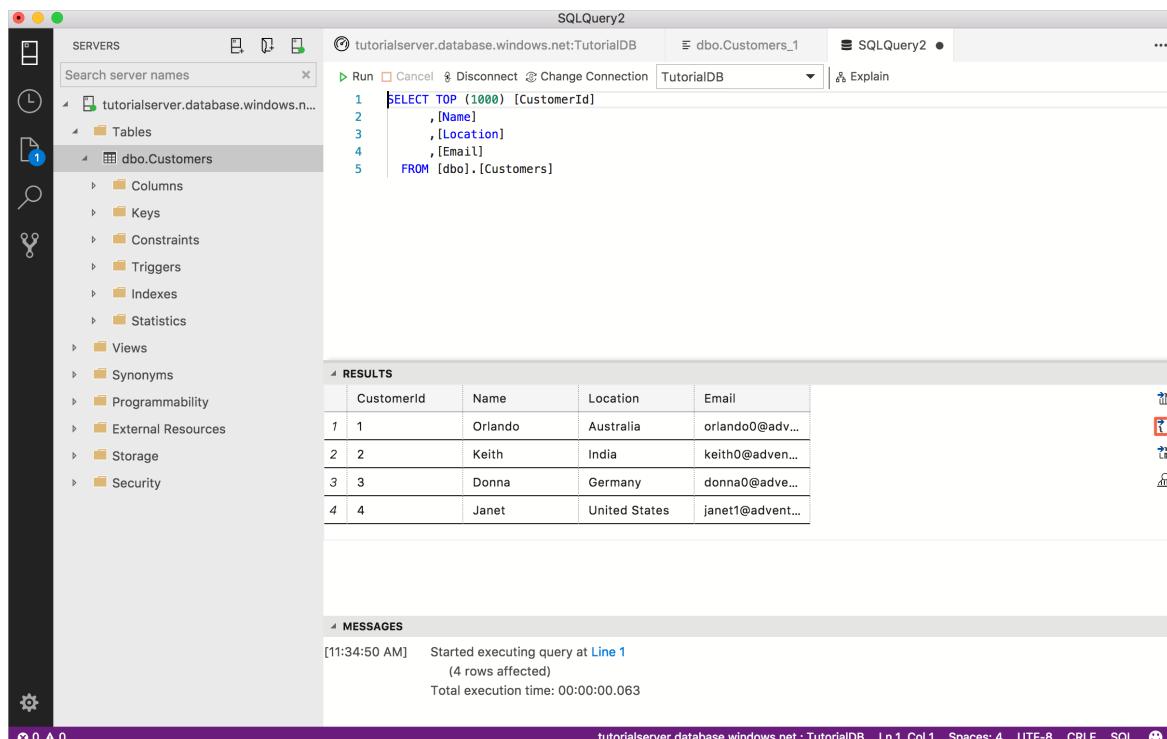
```

- To create the *setCustomer* stored procedure, press F5.

## Use save query results as JSON to test the *setCustomer* stored procedure

The *setCustomer* stored procedure created in the previous section requires JSON data be passed into the *@json\_val* parameter. This section shows how to get a properly formatted bit of JSON to pass into the parameter so you can test the stored procedure.

- In the SERVERS sidebar right-click the *dbo.Customers* table and click **SELECT TOP 1000 Rows**.
- Select the first row in the results view, make sure the entire row is selected (click the number 1 in the left-most column), and select **Save as JSON**.
- Change the folder to a location you'll remember so you can delete the file later (for example desktop) and click **Save**. The JSON formatted file opens.



- Select the JSON data in the editor and copy it.
- Open a new editor by pressing **Ctrl+N**.
- The previous steps show how you can easily get the properly formatted data to complete the call to the *setCustomer* procedure. You can see the following code uses the same JSON format with new customer details so we can test the *setCustomer* procedure. The statement includes syntax to declare the parameter and run the new get and set procedures. You can paste the copied data from the previous section and edit it so it is the same as the following example, or simply paste the following statement into the query editor.

```
-- example to execute the stored procedure we just created
declare @json nvarchar(max) =
N'[
{
    "CustomerID": 5,
    "Name": "Lucy",
    "Location": "Canada",
    "Email": "lucy0@adventure-works.com"
}
]'

EXECUTE dbo.setCustomer @json_val = @json
GO

EXECUTE dbo.getCustomer @ID = 5
```

7. Execute the script by pressing **F5**. The script inserts a new customer and returns the new customer's information in JSON format. Click the result to open a formatted view.

The screenshot shows the SSMS interface with the following details:

- Servers View:** Shows the database structure, with the `dbo.Customers` table selected.
- SQL Query Editor:** Contains the T-SQL script provided in the code block.
- Results View:** Displays the JSON output from the execution. The output is:
 

```
1 [{"CustomerID":5,"Name":"Lucy","Location":"Canada","Email":"lucy0@adventure-works.com"}]
```
- Messages View:** Shows the status of the execution.

## Next steps

In this tutorial, you learned how to:

- Quick search schema objects
- Edit table data
- Writing T-SQL script using snippets
- Learn about database object details using Peek Definition and Go to Definition

To learn how to enable the **five slowest queries** widget, complete the next tutorial:

[Enable the slow queries sample insight widget](#)

# Tutorial: Add the *five slowest queries* sample widget to the database dashboard

9/26/2022 • 2 minutes to read • [Edit Online](#)

This tutorial demonstrates the process of adding one of the built-in Azure Data Studio sample widgets to the *database dashboard* to quickly view a database's five slowest queries. You also learn how to view the details of the slow queries and query plans using Azure Data Studio features. During this tutorial, you learn how to:

- Enable Query Store on a database
- Add a pre-built insight widget to the database dashboard
- View details about the database's slowest queries
- View query execution plans for the slow queries

Azure Data Studio includes several insight widgets out-of-the-box. This tutorial shows how to add the *query-data-store-db-insight* widget, but the steps are basically the same for adding any widget.

## Prerequisites

This tutorial requires the SQL Server or Azure SQL Database *TutorialDB*. To create the *TutorialDB* database, complete one of the following quickstarts:

- [Connect and query SQL Server using Azure Data Studio](#)
- [Connect and query Azure SQL Database using Azure Data Studio](#)

## Turn on Query Store for your database

The widget in this example requires *Query Store* to be enabled.

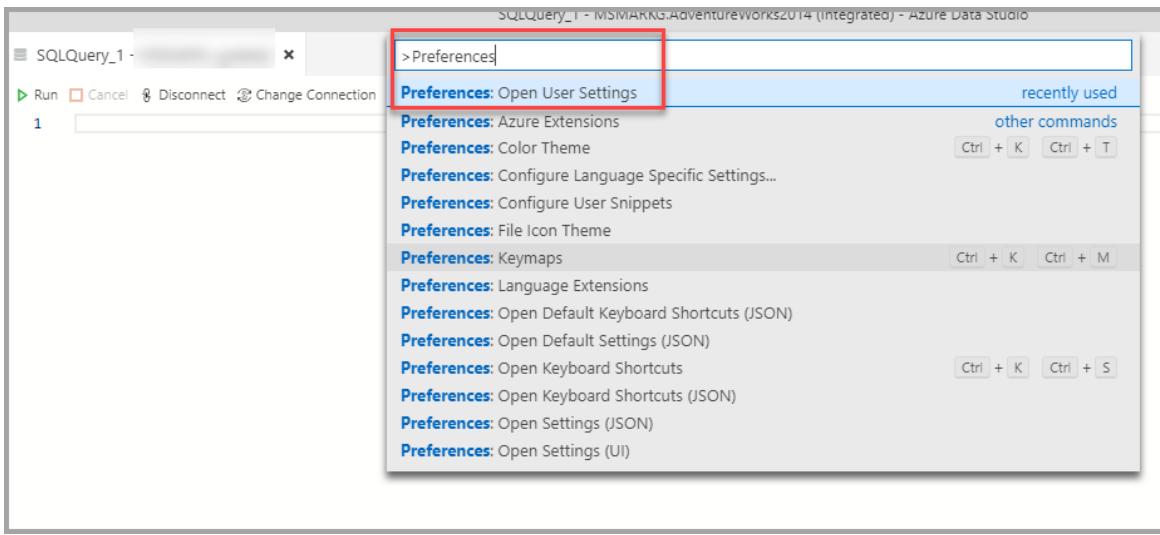
1. Right-click the **TutorialDB** database (in the SERVERS sidebar) and select **New Query**.
2. Paste the following Transact-SQL (T-SQL) statement in the query editor, and click **Run**:

```
ALTER DATABASE TutorialDB SET QUERY_STORE = ON
```

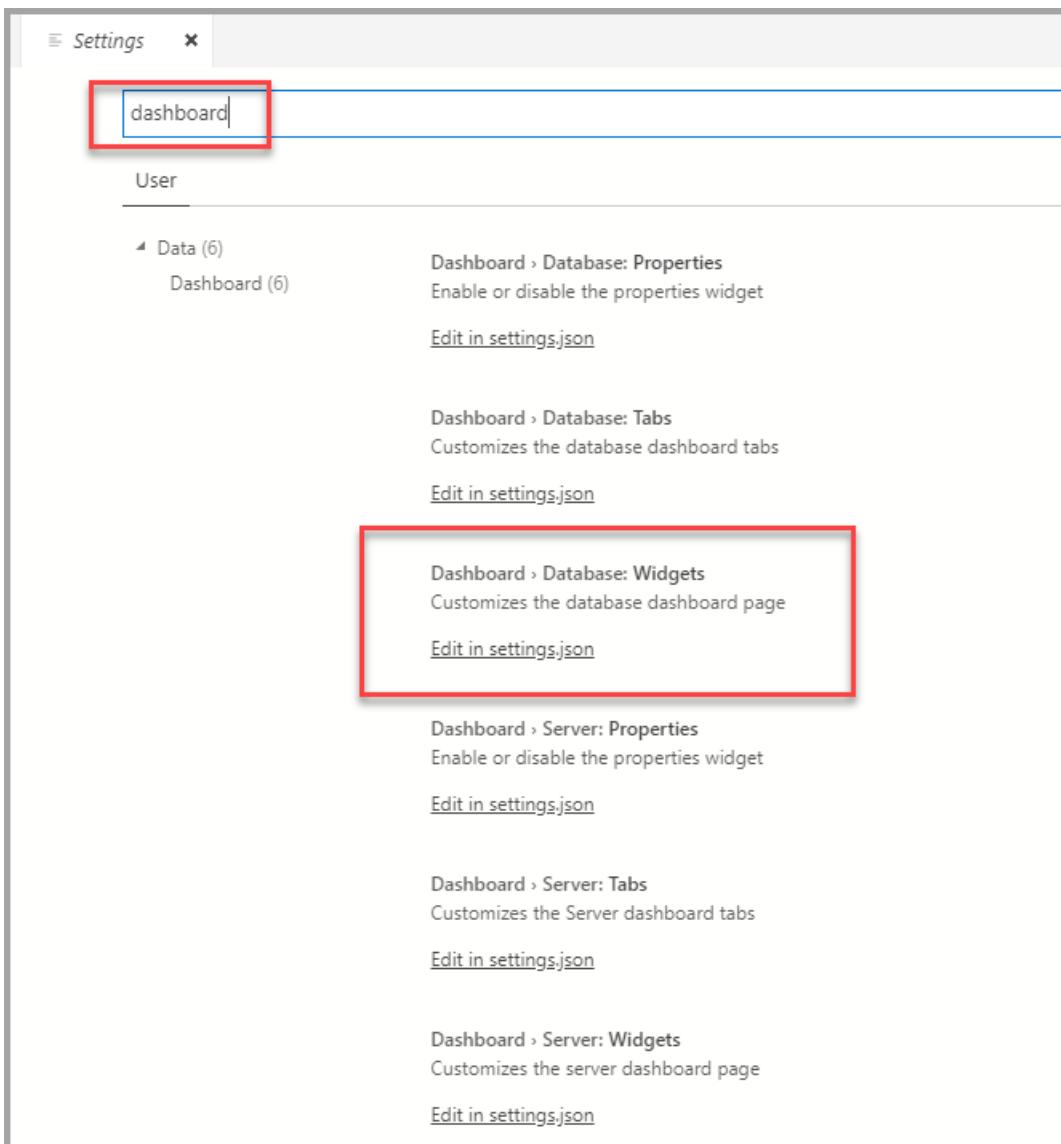
## Add the slow queries widget to your database dashboard

To add the *slow queries* widget to your dashboard, edit the *dashboard.database.widgets* setting in your *User Settings* file.

1. Open *User Settings* by pressing **Ctrl+Shift+P** to open the *Command Palette*.
2. Type *settings* in the search box and select **Preferences: Open User Settings**.



3. Type `dashboard` in the settings search box and locate `dashboard.database.widgets`, and then click `edit in settings.json`.



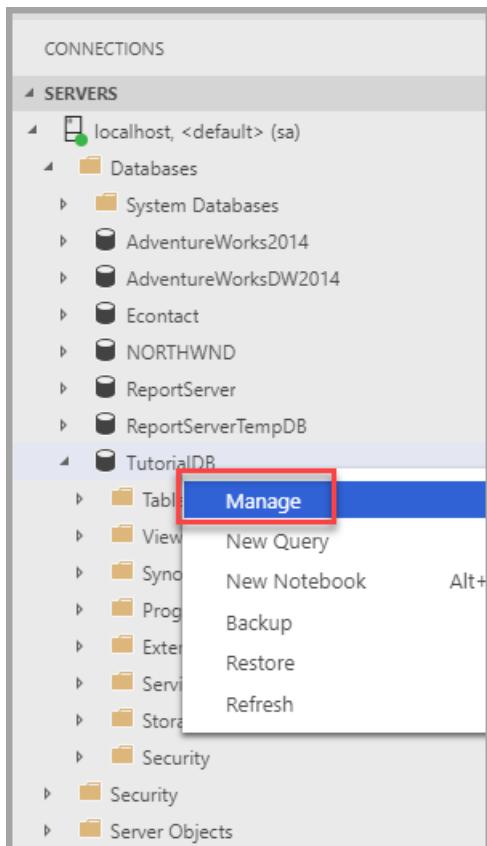
4. In `settings.json`, add the following code below:

```

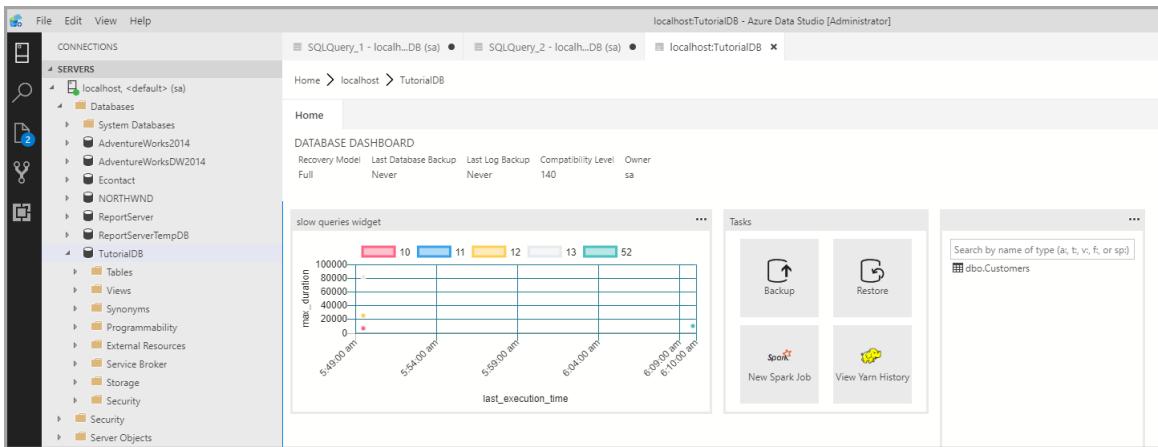
"dashboard.database.widgets": [
    {
        "name": "slow queries widget",
        "gridItemConfig": {
            "sizex": 2,
            "sizey": 1
        },
        "widget": {
            "query-data-store-db-insight": null
        }
    },
    {
        "name": "Tasks",
        "gridItemConfig": {
            "sizex": 1,
            "sizey": 1
        },
        "widget": {
            "tasks-widget": {}
        }
    },
    {
        "gridItemConfig": {
            "sizex": 1,
            "sizey": 2
        },
        "widget": {
            "explorer-widget": {}
        }
    }
]

```

5. Press **Ctrl+S** to save the modified **User Settings**.
6. Open the *Database dashboard* by navigating to **TutorialDB** in the **SERVERS** sidebar, right-click, and select **Manage**.



7. The insight widget appears on the dashboard:



## View insight details for more information

1. To view additional information for an insight widget, click the ellipses (...) in the upper right, and select **Show Details**.
2. To show more details for an item, select any item in **Chart Data** list.

Insights

ITEMS

	query_id	max_duration
11	8100	
12	25192	
13	81721	
52	9702	
54	286097	

ITEM DETAILS

Property	Value
query_id	12
ast_execution_time	2019-08-02 05:49:30
max_duration	25192
plan_id	12
query_sql_text	(@_msparam_0 nvarchar(4000))SELECT
query_plan	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan">

<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan">

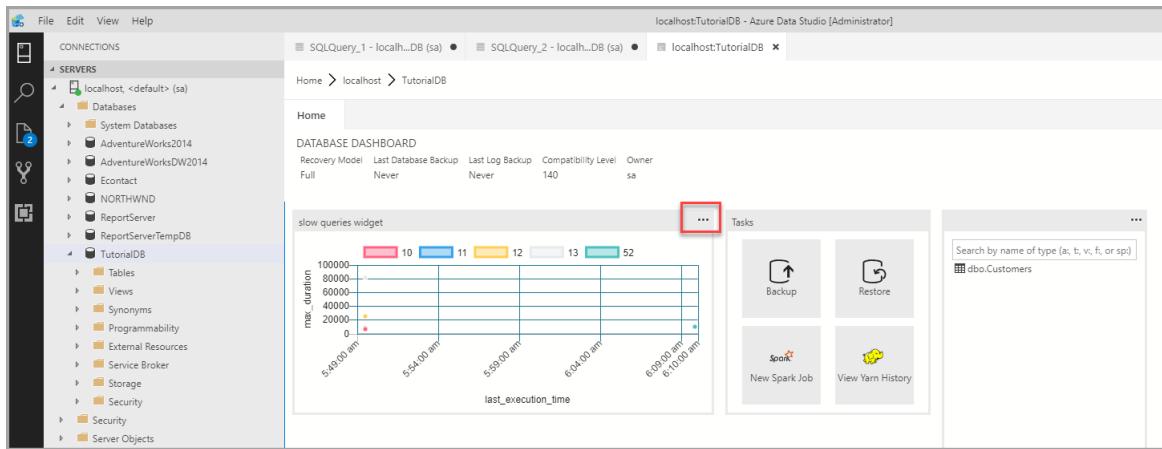
**Close**

3. Close the **Insights** pane.

## View the query plan

1. Right-click on the **TutorialDB** database and select *Manage*
2. From the *slow query widget* To view additional information for an insight widget, click the ellipses (...) in

the upper right, and select Run Query.



3. Now you should see a new query window with the results.

A screenshot of the Azure Data Studio query editor. The top bar shows 'Run', 'Cancel', 'Disconnect', 'Change Connection' set to 'NORTHWND', and 'Explain'. The code in the editor is:

```
1 declare @qds_status int = (SELECT actual_state
2 FROM sys.database_query_store_options)
3 if @qds_status > 0
4 Begin
5 WITH SlowestQry AS(
6     SELECT TOP 5
7         q.query_id,
8         MAX(rs.max_duration) max_duration
9     FROM sys.query_store_query_text AS qt
10    JOIN sys.query_store_query AS q
11       ON qt.query_text_id = q.query_text_id
12    JOIN sys.query_store_plan AS p
13       ON q.query_id = p.query_id
14    JOIN sys.query_store_runtime_stats AS rs
15       ON p.plan_id = rs.plan_id
16    WHERE rs.last_execution_time > DATEADD(week, -1, GETUTCDATE())
17    AND is_internal_query = 0
18    GROUP BY q.query_id
19    ORDER BY MAX(rs.max_duration) DESC)
20 SELECT
21     q.query_id,
22     format(rs.last_execution_time, 'yyyy-MM-dd hh:mm:ss') as last_execution_time
```

The results grid, highlighted with a red border, contains the following data:

	query_id	last_execution_time	max_duration	plan_id
1	20	2019-08-01 08:57:16	23044	20
2	21	2019-08-01 08:57:16	5941	21
3	32	2019-08-01 08:57:17	209686	32
4	35	2019-08-01 09:07:22	6400	35
5	32	2019-08-01 09:07:37	6005	32
6	21	2019-08-01 09:07:37	469	21
7	20	2019-08-01 09:07:37	94	20
8	33	2019-08-01 09:08:12	45282	33
9	35	2019-08-01 09:08:39	6400	35
10	33	2019-08-01 09:10:04	77735	33
11	32	2019-08-01 09:11:16	4385	36
12	20	2019-08-01 09:11:16	222	20
13	21	2019-08-01 09:11:16	772	21

4. Click Explain.

The screenshot shows the SQL Server Management Studio interface with three windows:

- SQLQuery\_1 - localh...DB (sa)
- SQLQuery\_2 - localh...DB (sa)
- localhost:TutorialDB

The SQLQuery\_1 window contains the following T-SQL code:

```
1 declare @qds_status int = (SELECT actual_state
2 FROM sys.database_query_store_options)
3 if @qds_status > 0
4 Begin
5 WITH SlowestQry AS(
6     SELECT TOP 5
7         q.query_id,
8             MAX(rs.max_duration) max_duration
9     FROM sys.query_store_query_text AS qt
10    JOIN sys.query_store_query AS q
11        ON qt.query_text_id = q.query_text_id
12    JOIN sys.query_store_plan AS p
13        ON q.query_id = p.query_id
14    JOIN sys.query_store_runtime_stats AS rs
15        ON p.plan_id = rs.plan_id
16    WHERE rs.last_execution_time > DATEADD(week, -1, GETUTCDATE())
17    AND is_internal_query = 0
18    GROUP BY q.query_id
19    ORDER BY MAX(rs.max_duration) DESC)
20 SELECT
21     q.query_id,
22     format(rs.last_execution_time,'yyyy-MM-dd hh:mm:ss') as [last_execution_time],
23     rs.max_duration,
24     p.plan_id
25 FROM sys.query_store_query_text AS qt
26    JOIN sys.query_store_query AS q
27        ON qt.query_text_id = q.query_text_id
28    JOIN sys.query_store_plan AS p
29        ON q.query_id = p.query_id
30    JOIN sys.query_store_runtime_stats AS rs
31        ON p.plan_id = rs.plan_id
32    JOIN SlowestQry tq
33        ON tq.query_id = q.query_id
34    WHERE rs.last_execution_time > DATEADD(week, -1, GETUTCDATE())
35    AND is_internal_query = 0
36    order by format(rs.last_execution_time,'yyyy-MM-dd hh:mm:ss')
37 END
38 else
39 select 0 as [query_id], getdate() as [QDS is not enabled], 0 as [max_duration]
```

5. View the query's execution plan:

Run Cancel Disconnect Change Connection TutorialDB Explain

```

1 declare @qds_status int = (SELECT actual_state
2 FROM sys.database_query_store_options)
3 if @qds_status > 0
4 Begin
5 WITH SlowestQry AS(
6     SELECT TOP 5
7         q.query_id,
8         MAX(rs.max_duration) max_duration
9     FROM sys.query_store_query_text AS qt
10    JOIN sys.query_store_query AS q
11       ON qt.query_text_id = q.query_text_id
12    JOIN sys.query_store_plan AS p
13       ON q.query_id = p.query_id
14    JOIN sys.query_store_runtime_stats AS rs
15       ON p.plan_id = rs.plan_id
16    WHERE rs.last_execution_time > DATEADD(week, -1, GETUTCDATE())
17    AND is_internal_query = 0
18    GROUP BY q.query_id
19    ORDER BY MAX(rs.max_duration) DESC)
20 SELECT
21     q.query_id,
22     format(rs.last_execution_time,'yyyy-MM-dd hh:mm:ss') as [last_execution_time],
23     rs.max_duration,
24     p.plan_id
25 FROM sys.query_store_query_text AS qt
26    JOIN sys.query_store_query AS q
27       ON qt.query_text_id = q.query_text_id
28    JOIN sys.query_store_plan AS p
29       ON q.query_id = p.query_id

```

Results Messages Query Plan Top Operations

Query 1  
declare @qds\_status int = (SELECT actual\_state FROM sys.database\_query\_store\_options)

```

graph LR
    A[SELECT] --> B[Compute Scalar  
Cost: 0%]
    B --> C[Nested Loops  
(Left Outer Join)  
Cost: 0%]
    C --> D[Constant Scan  
Cost: 0%]
    D --> E[Assert  
Cost: 0%]
    E --> F[Stream Aggregate  
(Aggregate)  
Cost: 37%]
    F --> G[Table-valued function  
[QUERY_STORE_OPTIONS]  
Cost: 62%]

```

## Next steps

In this tutorial, you learned how to:

- Enable Query Store on a database
- Add an insight widget to the database dashboard
- View details about the database's slowest queries
- View query execution plans for the slow queries

To learn how to enable the **table space usage** sample insight, complete the next tutorial:

[Enable the table space sample insight widget](#)

# Tutorial: Enable the table space usage sample insight widget using Azure Data Studio

9/26/2022 • 2 minutes to read • [Edit Online](#)

This tutorial demonstrates how to enable an insight widget on the database dashboard, providing an at-a-glance view about the space usage for all tables in a database. During this tutorial, you learn how to:

- Quickly turn on an insight widget using a built-in insight widget example
- View the details of table space usage
- Filter data and view label detail on an insight chart

## Prerequisites

This tutorial requires the SQL Server or Azure SQL Database *TutorialDB*. To create the *TutorialDB* database, complete one of the following quickstarts:

- [Connect and query SQL Server using Azure Data Studio](#)
- [Connect and query Azure SQL Database using Azure Data Studio](#)

## Turn on a management insight on Azure Data Studio's database dashboard

Azure Data Studio has a built-in sample widget to monitor the space used by tables in a database.

1. Open *User Settings* by pressing **Ctrl+Shift+P** to open the *Command Palette*.
2. Type *settings* in the search box and select **Preferences: Open User Settings**.
3. Type *dashboard* in Settings Search input box and locate **dashboard.database.widgets**.
4. To customize the **dashboard.database.widgets** settings, you need to edit the **dashboard.database.widgets** entry in the **USER SETTINGS** section.

dashboard.data

User

✓ Data (3)

**Dashboard** (3)

Dashboard › Database: Properties

Enable or disable the properties widget

[Edit in settings.json](#)

Dashboard › Database: Tabs

Customizes the database dashboard tabs

[Edit in settings.json](#)

Dashboard › Database: Widgets

Customizes the database dashboard page

[Edit in settings.json](#)

If there is no **dashboard.database.widgets** in the **USER SETTINGS** section, hover over the **dashboard.database.widgets** text in the **DEFAULT SETTINGS** column and click the *gear* icon that appears to the left of the text and click **Copy as Setting JSON**. If the pop-up says **Replace in Settings**, don't click it! Go to the **USER SETTINGS** column to the right and locate the **dashboard.database.widgets** section and advance to the next step.

5. In the **dashboard.database.widgets** section, add the following lines:

```
{  
    "name": "Space Used by Tables",  
    "gridItemConfig": {  
        "sizex": 2,  
        "sizey": 1  
    },  
    "widget": {  
        "table-space-db-insight": null  
    }  
},
```

The **dashboard.database.widgets** section should look similar to the following image:

```

1  {
2    "workbench.enablePreviewFeatures": true,
3    "datasource.connectionGroups": [
4      {
5        "name": "ROOT",
6        "id": "C777F068-202E-4480-B475-FA416154D458"
7      }
8    ],
9    "dashboard.database.widgets": [
10      {
11        "name": "Space Used by Tables",
12        "gridItemConfig": {
13          "sizex": 2,
14          "sizey": 1
15        },
16        "widget": {
17          "table-space-db-insight": null
18        }
19      },
20      {
21        "name": "slow queries widget",
22        "gridItemConfig": {
23          "sizex": 2,
24          "sizey": 1
25        },
26        "widget": {
27          "query-data-store-db-insight": null
28        }
29      },
30      {
31        "name": "Tasks",
32        "gridItemConfig": {
33          "sizex": 1,
34          "sizey": 1
35        },
36        "widget": {
37          "tasks-widget": {}
38        }
39      },
40      {
41        "gridItemConfig": {
42          "sizex": 1,
43          "sizey": 2
44        }
45      }
46    ]
47  }

```

6. Press **Ctrl+S** to save the settings.
7. Open database dashboard by right-clicking **TutorialDB** and click **Manage**.
8. View the *table space* insight widget as shown in the following image:

The screenshot shows the 'DATABASE DASHBOARD' for the 'TutorialDB'. The 'Space Used by Tables' chart for the 'Customers' table is highlighted with a red box. The chart displays the following data for the 'Customers' table:

Series	Value
rows_count	4
total_pages	9
used_pages	2
data_pages	1
total_space_MB	9
used_space_MB	2
data_space_MB	1

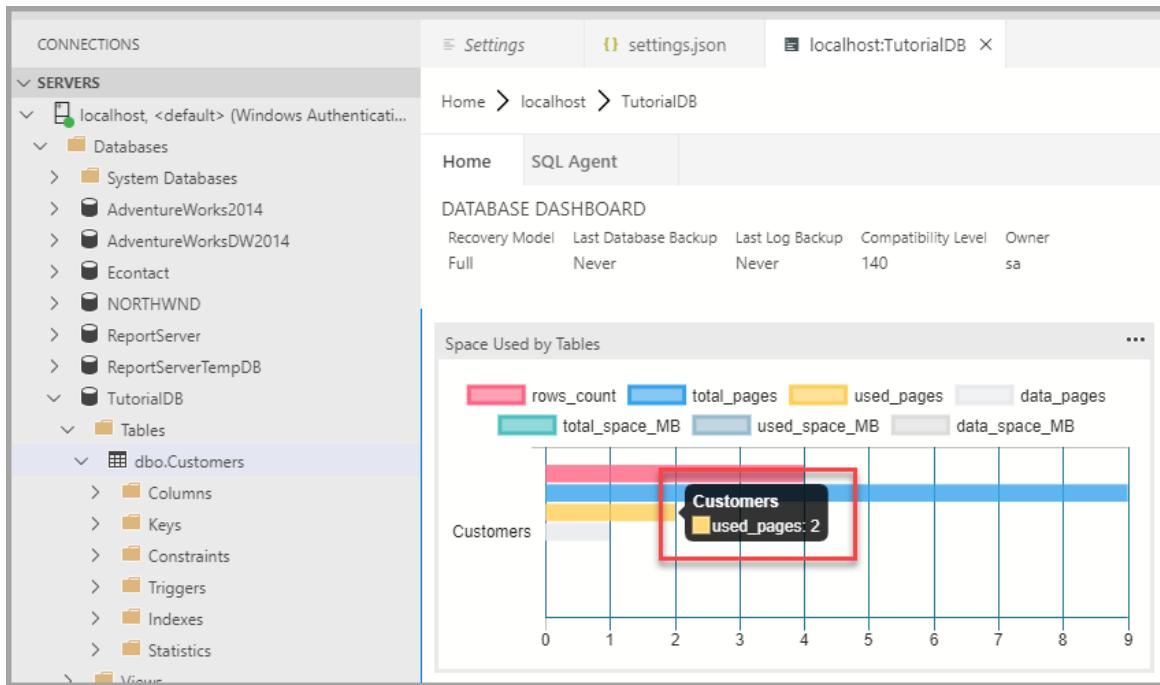
## Working with the insight chart

Azure Data Studio's insight chart provides filtering and mouse-hover details. To try out the following steps:

1. Click and toggle the *row\_count* legend on the chart. Azure Data Studio shows and hides data series as

you toggle a legend on or off.

2. Hover the mouse pointer over the chart. Azure Data Studio shows more information about the data series label and its value as shown in the following screenshot.



## Next steps

In this tutorial, you learned how to:

- Quickly turn on an insight widget using a built-in insight widget sample.
- View the details of table space usage.
- Filter data and view label detail on an insight chart

To learn how to build a custom insight widget, complete the next tutorial:

[Build a custom insight widget](#)

# Tutorial: Build a custom insight widget

9/26/2022 • 2 minutes to read • [Edit Online](#)

This tutorial demonstrates how to use your own insight queries to build custom insight widgets.

During this tutorial you learn how to:

- Run your own query and view it in a chart
- Build a custom insight widget from the chart
- Add the chart to a server or database dashboard
- Add details to your custom insight widget

## Prerequisites

This tutorial requires the SQL Server or Azure SQL Database *TutorialDB*. To create the *TutorialDB* database, complete one of the following quickstarts:

- [Connect and query SQL Server using Azure Data Studio](#)
- [Connect and query Azure SQL Database using Azure Data Studio](#)

## Run your own query and view the result in a chart view

In this step, run a sql script to query the current active sessions.

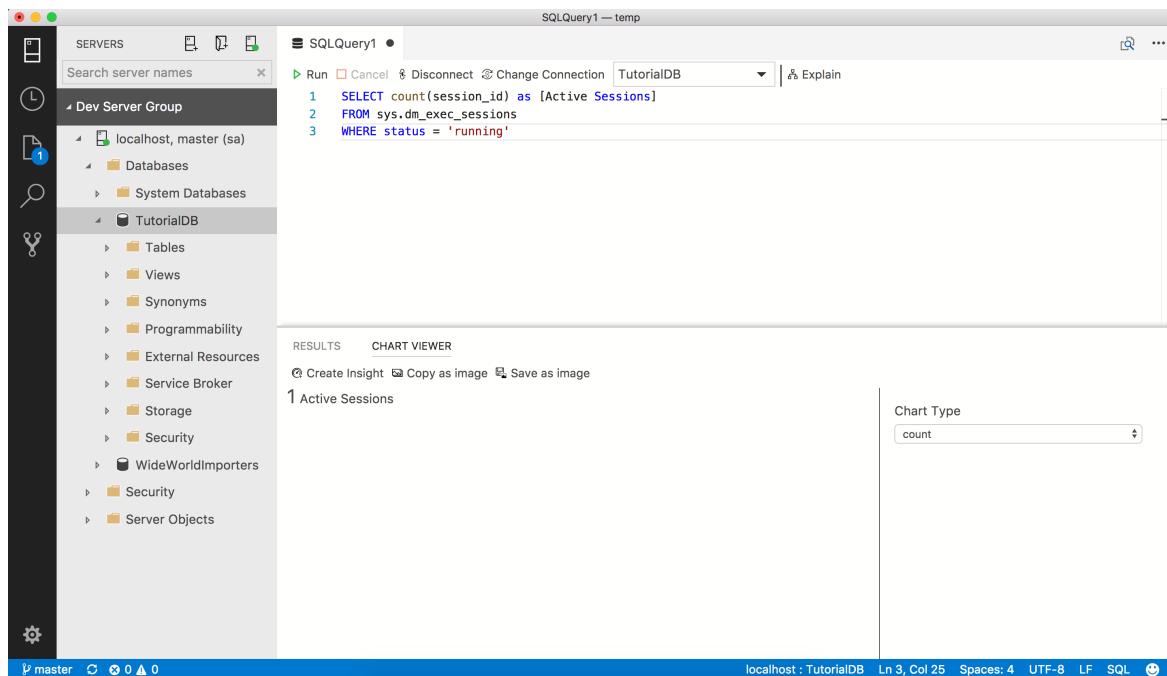
1. To open a new editor, press **Ctrl+N**.
2. Change the connection context to **TutorialDB**.
3. Paste the following query into the query editor:

```
SELECT count(session_id) as [Active Sessions]
FROM sys.dm_exec_sessions
WHERE status = 'running'
```

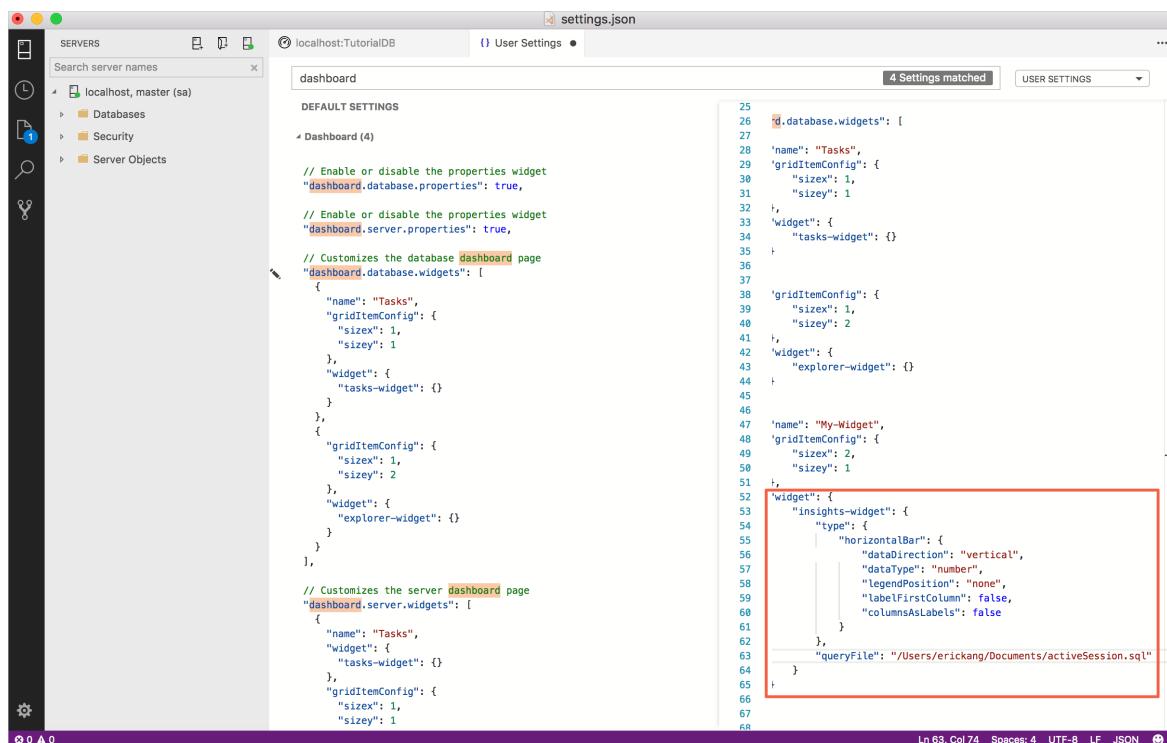
4. Save the query in the editor to a \*.sql file. For this tutorial, save the script as *activeSession.sql*.
5. To execute the query, press **F5**.
6. After the query results are displayed, click **View as Chart**, then click the **Chart Viewer** tab.
7. Change **Chart Type** to **count**. These settings render a count chart.

## Add the custom insight to the database dashboard

1. To open the insight widget configuration, click **Create Insight** on *Chart Viewer*.



2. Copy the insight configuration (the JSON data).
3. Press **Ctrl+Comma** to open *User Settings*.
4. Type *dashboard* in *Search Settings*.
5. Click **Edit** for *dashboard.database.widgets*.

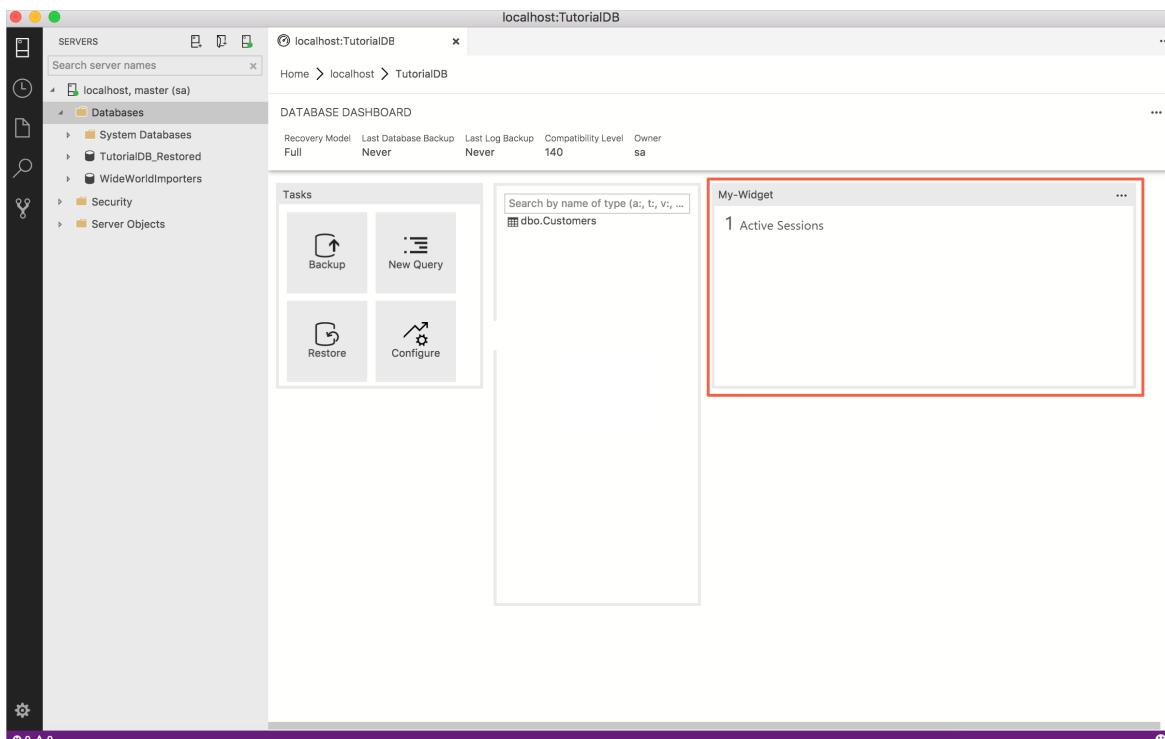


```

"dashboard.database.widgets": [
    {
        "name": "My-Widget",
        "gridItemConfig": {
            "sizeX": 2,
            "sizeY": 1
        },
        "widget": {
            "insights-widget": {
                "type": {
                    "count": {
                        "dataDirection": "vertical",
                        "dataType": "number",
                        "legendPosition": "none",
                        "labelFirstColumn": false,
                        "columnsAsLabels": false
                    }
                },
                "queryFile": "{your file folder}/activeSession.sql"
            }
        }
    }
]

```

7. Save the *User Settings* file and open the *TutorialDB* database dashboard to see the active sessions widget:



## Add details to custom insight

1. To open a new editor, press **Ctrl+N**.
2. Change the connection context to **TutorialDB**.
3. Paste the following query into the query editor:

```

SELECT session_id AS [SID], login_time AS [Login Time], host_name AS [Host Name], program_name AS [Program Name], login_name AS [Login Name]
FROM sys.dm_exec_sessions
WHERE status = 'running'

```

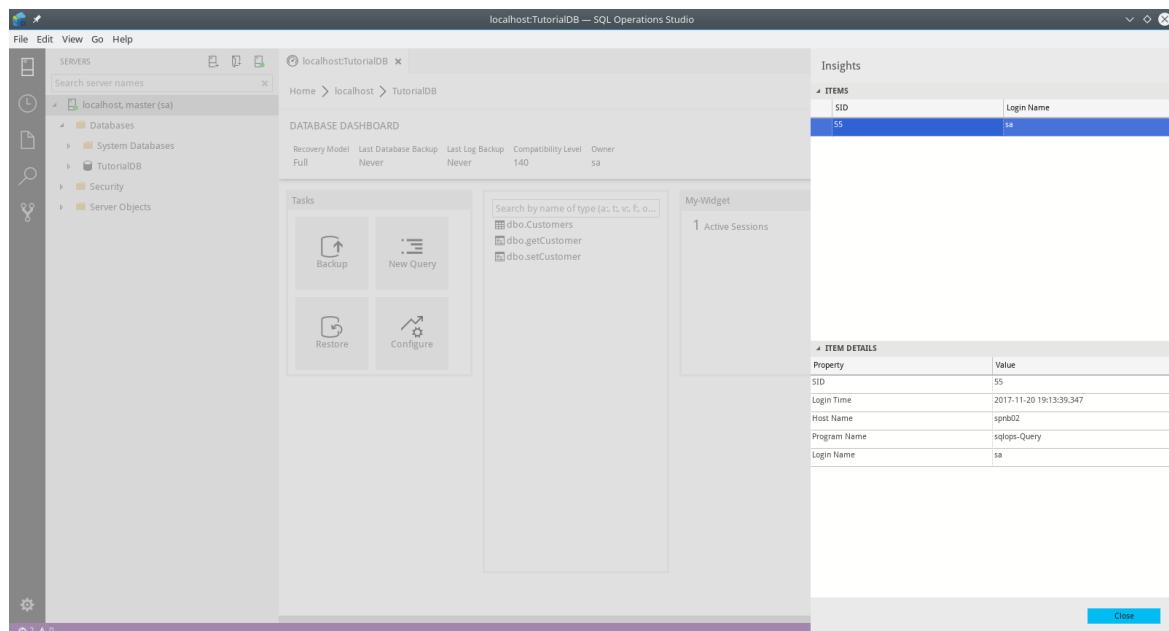
4. Save the query in the editor to a \*.sql file. For this tutorial, save the script as *activeSessionDetail.sql*.

5. Press **Ctrl+Comma** to open *User Settings*.

6. Edit the existing *dashboard.database.widgets* node in your settings file:

```
"dashboard.database.widgets": [
    {
        "name": "My-Widget",
        "gridItemConfig": {
            "sizex": 2,
            "sizey": 1
        },
        "widget": {
            "insights-widget": {
                "type": {
                    "count": {
                        "dataDirection": "vertical",
                        "dataType": "number",
                        "legendPosition": "none",
                        "labelFirstColumn": false,
                        "columnsAsLabels": false
                    }
                },
                "queryFile": "{your file folder}/activeSession.sql",
                "details": {
                    "queryFile": "{your file folder}/activeSessionDetail.sql",
                    "label": "SID",
                    "value": "Login Name"
                }
            }
        }
    }
]
```

7. Save the *User Settings* file and open the *TutorialDB* database dashboard. Click the ellipsis (...) button next to *My-Widget* to show the details:



## Next steps

In this tutorial, you learned how to:

- Run your own query and view it in a chart

- Build a custom insight widget from the chart
- Add the chart to a server or database dashboard
- Add details to your custom insight widget

To learn how to backup and restore databases, complete the next tutorial:

[Backup and restore databases.](#)

# Tutorial: Back up and restore databases using Azure Data Studio

9/26/2022 • 2 minutes to read • [Edit Online](#)

In this tutorial, you learn how to use Azure Data Studio to:

- Back up a database.
- View the backup status.
- Generate the script used to perform the backup.
- Restore a database.
- View the status of the restore task.

## Prerequisites

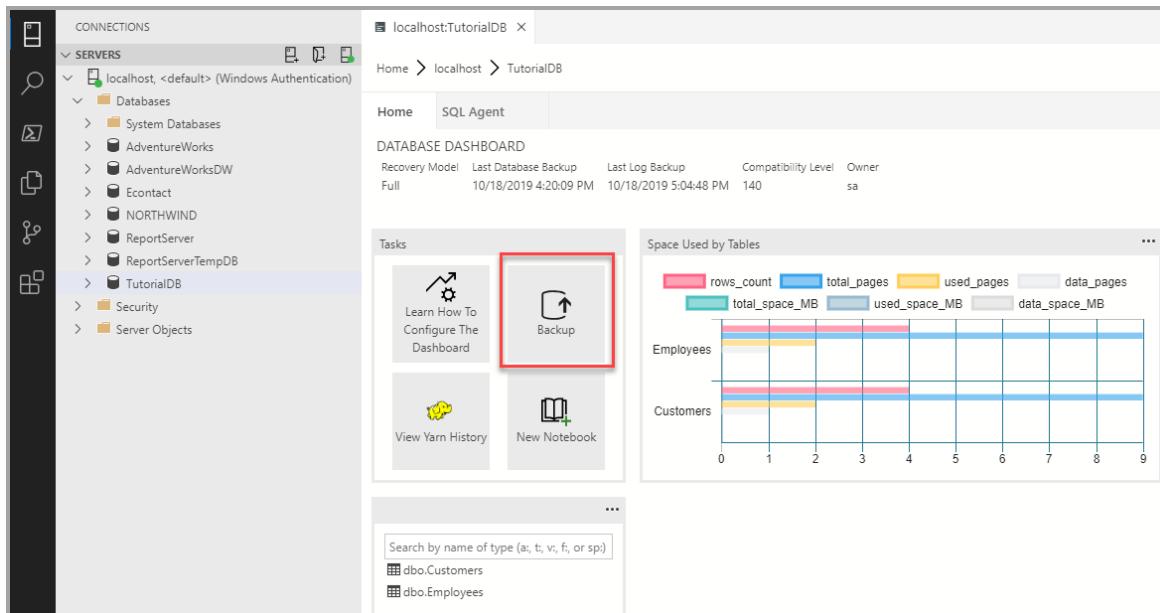
This tutorial requires SQL Server *TutorialDB*. To create the TutorialDB database, complete the following quickstart:

- [Use Azure Data Studio to connect and query SQL Server](#)

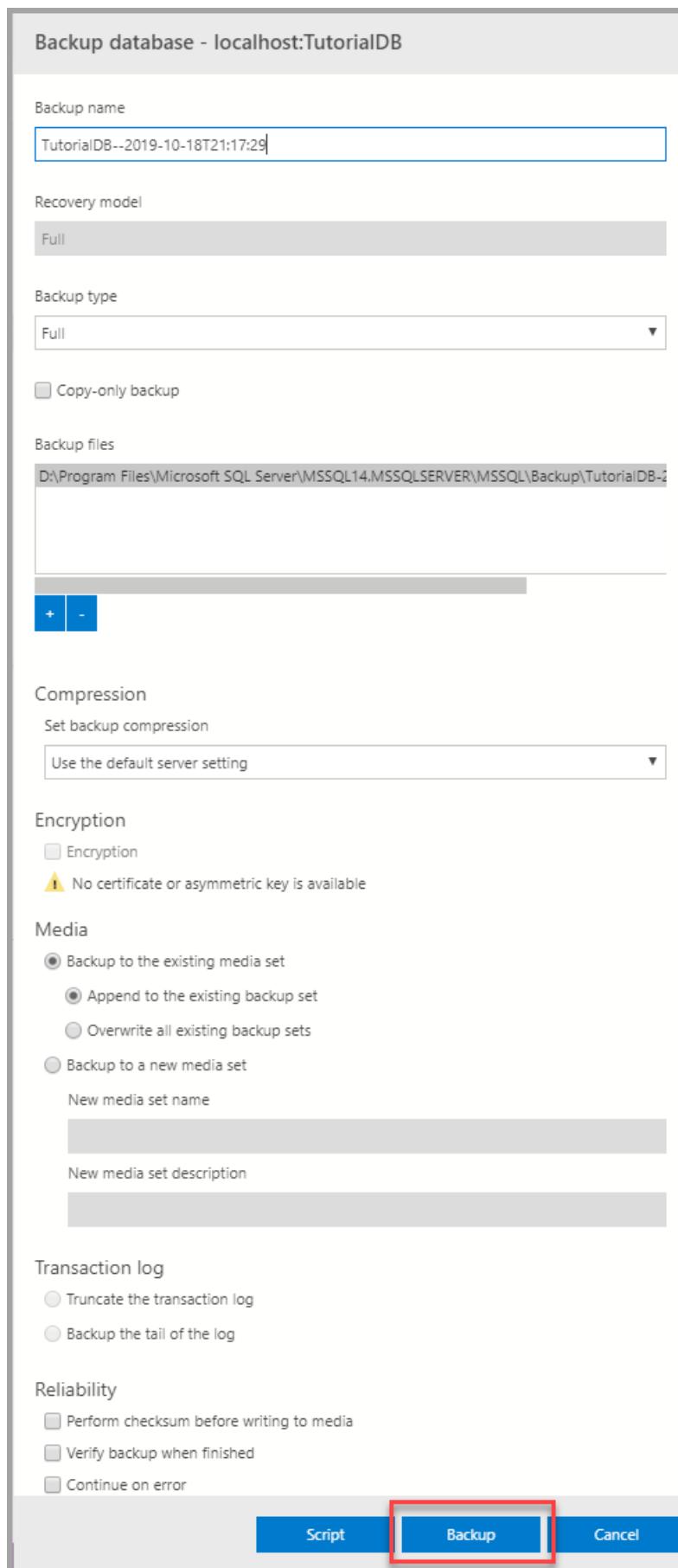
This tutorial requires a connection to a SQL Server database. Azure SQL Database has automated backups, so Azure Data Studio doesn't perform Azure SQL Database backup and restore. For more information, see [Learn about automatic SQL Database backups](#).

## Back up a database

1. Open the TutorialDB database dashboard by opening the **SERVERS** sidebar. Then select **Ctrl+G**, expand **Databases**, right-click **TutorialDB**, and select **Manage**.
2. Open the **Backup database** dialog box by selecting **Backup** on the **Tasks** widget.



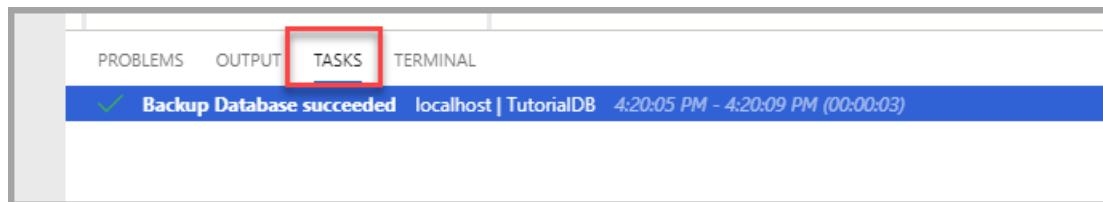
3. This tutorial uses the default backup options, so select **Backup**.



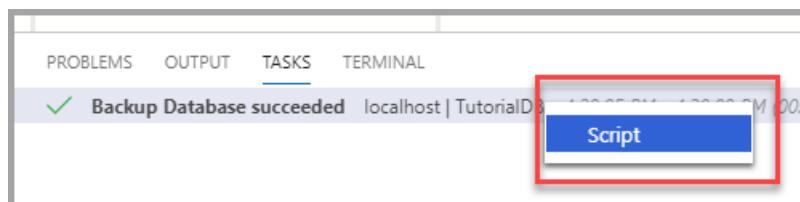
After you select **Backup**, the **Backup database** dialog box disappears and the backup process begins.

## View the backup status and the backup script

1. The **Task History** pane appears, or select **Ctrl+T** to open it.

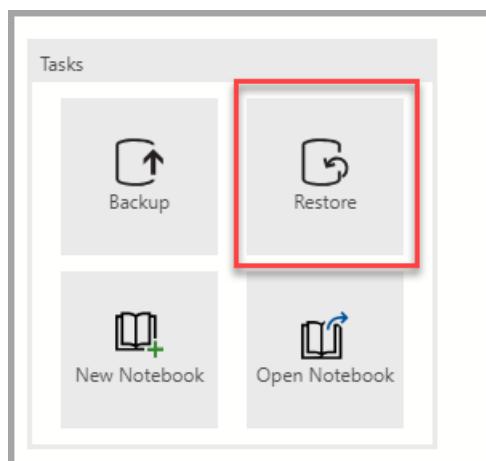


2. To view the backup script in the editor, right-click **Backup Database succeeded** and select **Script**.

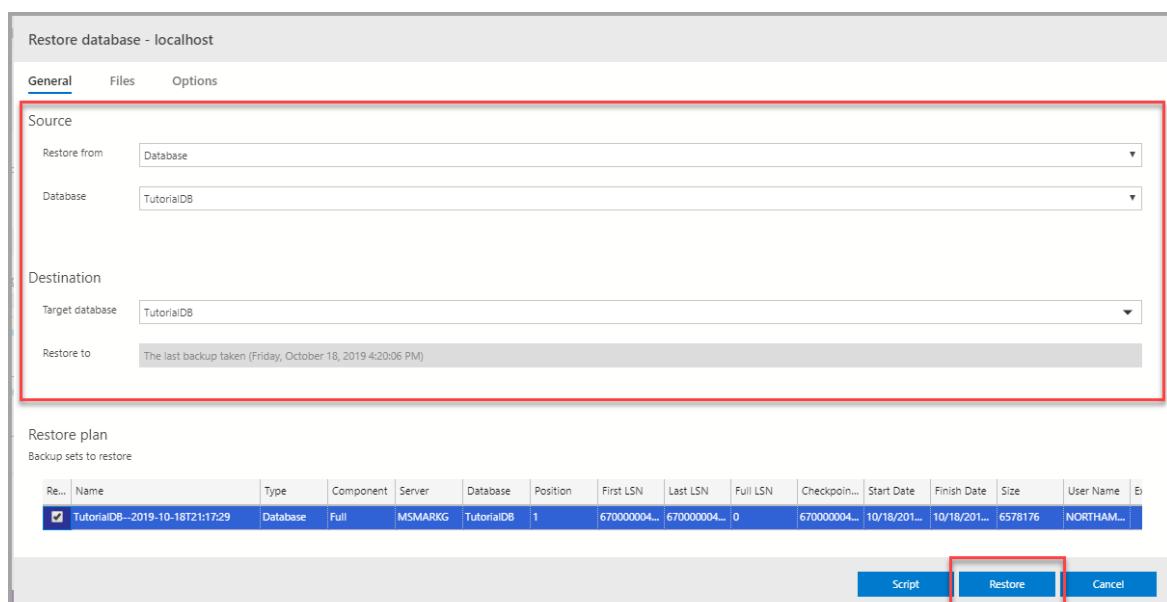


## Restore a database from a backup file

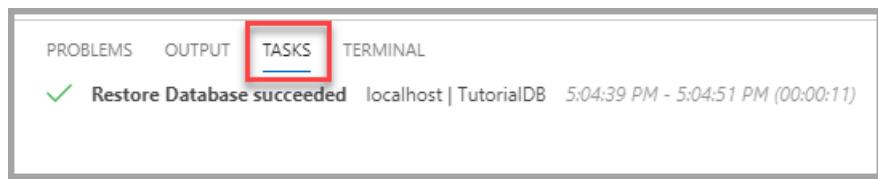
1. Open the SERVERS sidebar by selecting **Ctrl+G**. Then right-click your server, and select **Manage**.
2. Open the **Restore database** dialog box by selecting **Restore** on the **Tasks** widget.



3. Select **Backup file** in the **Restore from** box.
4. Select the ellipses (...) in the **Backup file path** box, and select the latest backup file for *TutorialDB*.
5. Enter **TutorialDB\_Restored** in the **Target database** box in the **Destination** section to restore the backup file to a new database. Then select **Restore**.



6. To view the status of the restore operation, select **Ctrl+T** to open the **Task History**.



# Extend the functionality of Azure Data Studio

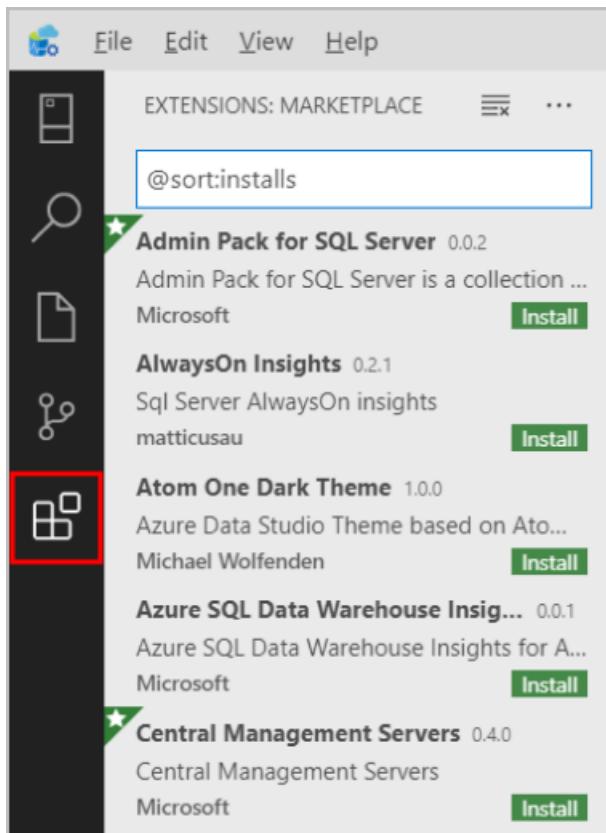
9/26/2022 • 4 minutes to read • [Edit Online](#)

Extensions in Azure Data Studio provide an easy way to add more functionality to the base Azure Data Studio installation.

Extensions are provided by the Azure Data Studio team (Microsoft), as well as the third-party community (you!). For more information about creating extensions, see [Extension authoring](#).

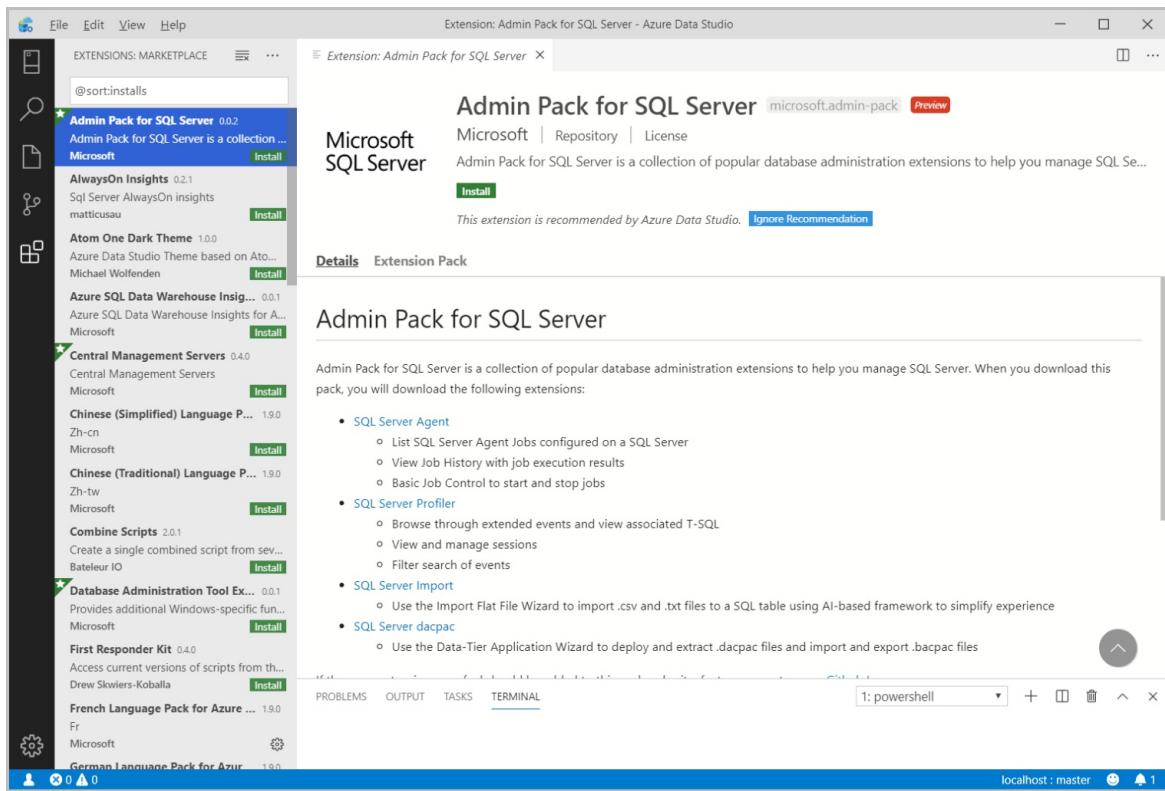
## Add Azure Data Studio extensions

1. Access the available extensions by selecting the Extensions Icon, or by selecting **Extensions** in the **View** menu. You can use the **View: Show Extensions** command, available in the **Command Palette** (F1 or `Ctrl+Shift+P`).



You can also quickly access the extensions manager by pressing `Ctrl+Shift+X` (Windows/Linux) or `Command+Shift+X` (Mac).

2. Select an available extension to view its details.



3. Select the extension you want and **Install** it.
4. Once installed, **Reload** to enable the extension in Azure Data Studio (only required when installing an extension for the first time).

If you're having problems accessing the Extensions Manager on Azure Data Studio, you can download the extension you need on our [GitHub Wiki](#).

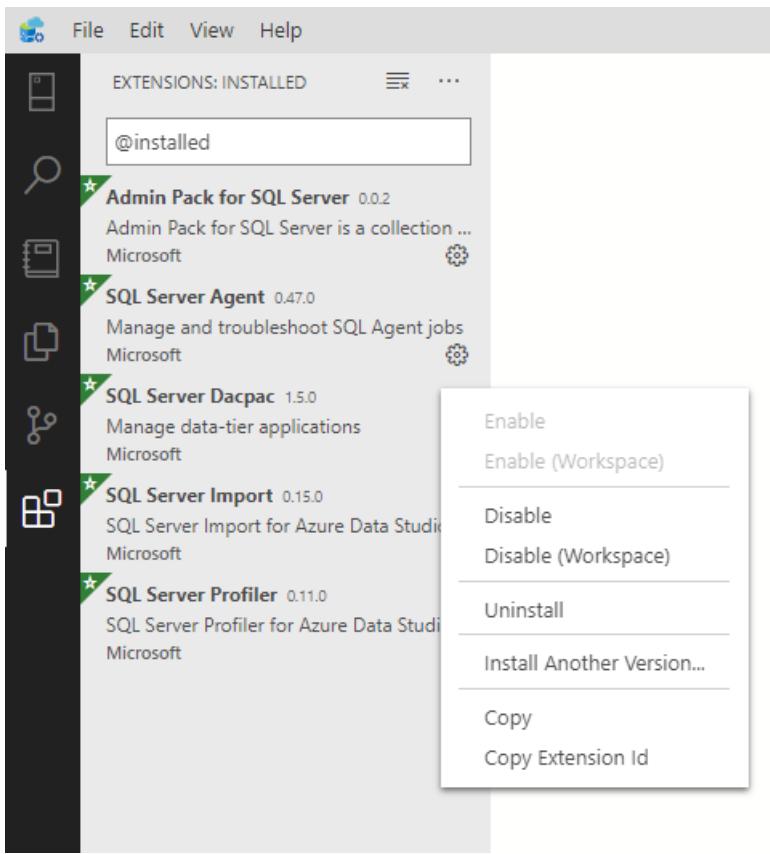
## Manage extensions

### List installed extensions

The default Extensions view shows the extensions that are currently enabled, all extensions that are recommended for you, and a collapsed container of all currently disabled extensions. The **Extensions: Show Installed Extensions** command, available in the **Command Palette** or the **More Actions** (...) drop-down menu, shows a list of all installed extensions, including disabled extensions.

### Uninstall an extension

To uninstall an extension, click the gear icon on the right of an extension entry and choose **Uninstall** from the drop-down menu. This uninstalls the selected extension and will prompt you to reload Azure Data Studio.

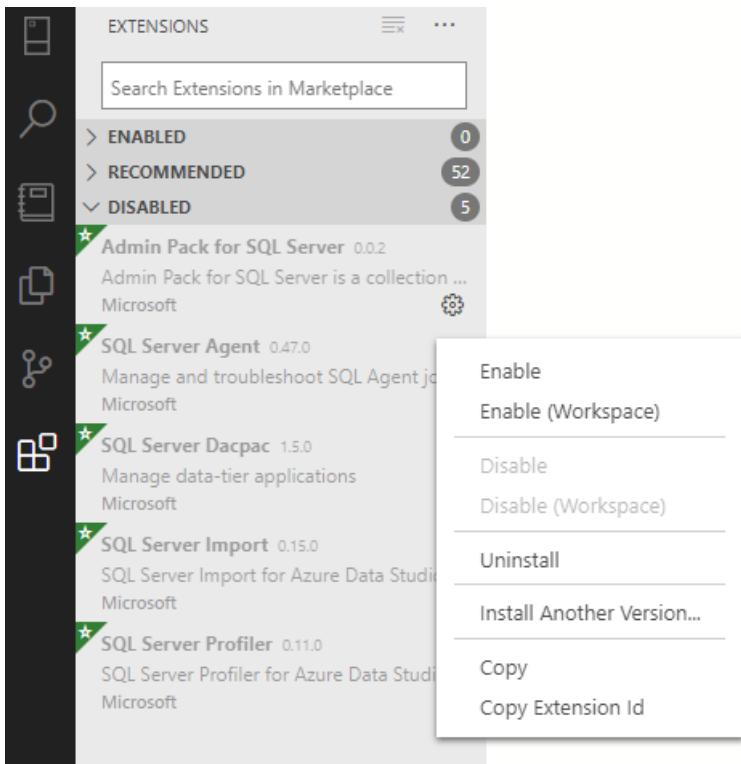


## Disable an extension

You may temporarily disable an extension instead of permanently removing an extension. You can either disable an extension across all Azure Data Studio sessions (**Disable**) or just for your current Workspace (**Disable (Workspace)**). You can also disable all of your currently installed extensions through the **Command Palette** with the commands **Extensions: Disable All Extensions** and **Extensions: Disable All Extensions (Workspace)**.

## Enable an extension

If an extension has been disabled, it will be in the **Disabled** section of the extension list and marked as **Disabled**. You can re-enable it with the **Enable** or **Enable (Workspace)** commands in the drop-down menu. The **Command Palette** also lets you enable all extensions with the commands **Extensions: Enable All Extensions** and **Extensions: Enable All Extensions (Workspace)**.



## Updating an extension

Azure Data Studio automatically checks for and installs updates for any of your installed extensions.

You can check for updates to the extension and have them automatically updated by configuring **Auto Update** in Azure Data Studio settings.

To enable auto updates:

1. Select the **Settings** icon in Azure Data Studio.
2. Select the checkbox under **User > Features > Extensions > Auto Check Updates**.
3. Select the dropdown under **User > Features > Extensions > Auto Update** and select either **All Extensions** or **Only Enabled Extensions**.

A screenshot of the Azure Data Studio User settings page. The top navigation bar has tabs for **User** (highlighted with a red box) and **Workspace**. The left sidebar lists Commonly Used features: Text Editor, Workbench, Window, and a expanded **Features** section which includes: Explorer, Search, Testing, SCM, **Extensions** (highlighted with a red box), Terminal, Task, Problems, Output, and Comments. In the main content area, there is a heading **Extensions**. Below it is a section for **Auto Check Updates** with a checked checkbox and a descriptive text: "When enabled, automatically checks extensions for updates. If an extension has an update available, it will be downloaded and installed automatically." A red arrow points from the **Extensions** link in the sidebar to this checkbox. Below that is a section for **Auto Update** with a dropdown menu. The menu has four options: "All Extensions" (selected and highlighted with a red box), "All Extensions true", "Only Enabled Extensions onlyEnabledExtensions", and "None false". A red arrow points from the "All Extensions" option in the dropdown to the "All Extensions" link in the sidebar. At the bottom of the dropdown menu, there is explanatory text: "Download and install updates automatically for all extensions." and a note: "Configured via HandlerExtension.Id".

#### NOTE

If you want to update the extension manually, you can disable **Auto Update** and install the updates from the extension in the Marketplace.

## Install from a VSIX

You can manually install an Azure Data Studio extension packaged in a `.vsix` file using the **Install from VSIX** command in the Extensions view command drop-down, or the **Extensions: Install from VSIX** command in the Command Palette and point to the extension's `.vsix` file.

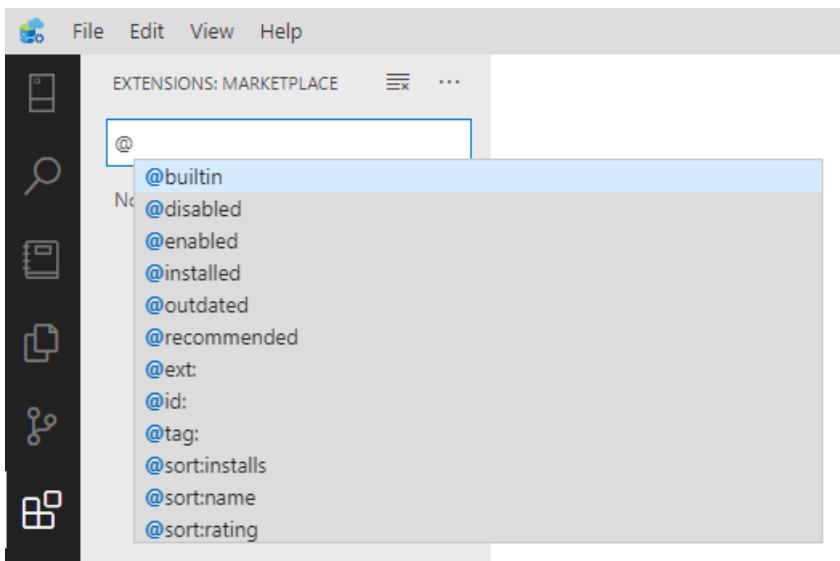
## Access installed Azure Data Studio extensions

Each extension enhances your experience in Azure Data Studio in a different way. As a result, the entry point for extensions can vary. Refer to your installed extension's individual documentation for information on how its features can be accessed once it's installed.

## Extensions view filters

The Extensions view search box supports filters to help you find and manage extensions. The commands **Show Installed Extensions** and **Show Recommended Extensions** use filters such as `@installed` and `@recommended` in the search box.

You can see a complete listing of all filters and sort commands by typing `@` in the extensions search box and navigating through the suggestions:



Here are the Extensions view filters:

- `@builtin` - Show extensions that come with Azure Data Studio. Grouped by type (Programming Languages, Themes, etc.).
- `@disabled` - Show disabled installed extensions.
- `@enabled` - Show enabled installed extensions. Extensions can be individually enabled/disabled.
- `@installed` - Show installed extensions.
- `@outdated` - Show outdated installed extensions. A newer version is available on the Marketplace.
- `@recommended` - Show recommended extensions. Grouped as Workspace specific or general use.
- `@category` - Show extensions belonging to specified category. Below are a few of supported categories. For a complete list, type `@category` and follow the options in the suggestion list:

- `@category:themes`
- `@category:formatters`
- `@category:snippets` These filters can be combined as well. For example, `@installed @category:themes` displays all installed themes.

If no filter is provided, the Extensions view displays currently installed and recommended extensions.

## Sorting

You can sort extensions with the `@sort` filter, which can take the following values:

- `installs` - Sort by extension gallery installation count, in descending order.
- `rating` - Sort by extension gallery rating (1-5 stars), in descending order.
- `name` - Sort alphabetically by extension name.

## Common questions

### Where are extensions installed?

Extensions are installed in a per user extensions folder. Depending on your platform, the location is in the following folder:

- Windows `%USERPROFILE%\azuredatascudio\extensions`
- macOS `~/.azuredatascudio/extensions`
- Linux `~/.azuredatascudio/extensions`

# Azure Arc extension for Azure Data Studio

9/26/2022 • 2 minutes to read • [Edit Online](#)

The [Azure Arc extension](#) is an extension for creating and managing Azure Arc data services resources.

**Key actions include:**

- Create a resource
  - Data Controller
  - SQL Managed Instance for Azure Arc
  - PostgreSQL for Azure Arc
- Manage a resource
  - View Data Controller dashboard
  - View SQL Managed Instance for Azure Arc dashboard
  - View PostgreSQL for Azure Arc dashboard
- Run Azure Arc Jupyter Book

## Prerequisite

[Download and Install Azure Data Studio](#).

## Install the extensions

1. [Install Azure CLI](#)
2. [Install Azure `azdata` extension](#)
3. Install Azure CLI extension for Azure Data Studio from the Azure Data Studio gallery
4. Install the Azure Arc extension for Azure Data Studio from the gallery
5. Restart Azure Data Studio

## Sign in with Azure account

1. Select Accounts on bottom left
2. Select Add Account
3. This action will launch a browser. Sign in to your Azure Account.

## Create a resource

This extension supports deployment of Azure Arc data controllers, PostgreSQL for Azure Arc, and SQL Managed Instance for Azure Arc. Deployments can be done through the built-in Deployment wizard.

1. Select Connections viewlet on left Activity Bar
2. Select the three dots and select **New Deployment**
3. Follow prompts to create a new Azure Arc resource.

## Manage a resource

After deploying an Azure Arc data controller from azdata, Azure portal, or Azure Data Studio, you can connect to it from Azure Data Studio.

1. Open Connections viewlet on the left activity bar.
2. Expand **Azure Arc controllers**
3. Select **Connect controller**
4. Fill out parameters and connect.

Once connected, you can view the resources deployed on the data controller. You can then right-click and select **Manage** to access the resource dashboard.

These dashboards will provide additional information about the resource, including option to open in Azure portal.

## Next steps

To learn more about Azure Arc data services, [check our documentation](#).

# Azure SQL migration extension for Azure Data Studio

9/26/2022 • 3 minutes to read • [Edit Online](#)

The Azure SQL migration extension for [Azure Data Studio](#) enables you to use the SQL Server assessment and migration capability in Azure Data Studio.

You can use the Azure SQL migration extension in Azure Data Studio to migrate SQL Server databases to either Azure SQL Managed Instance (Platform-as-a-Service) or to SQL Server on Azure Virtual Machines (Infrastructure-as-a-Service).

The Azure SQL migration extension in Azure Data Studio brings together a simplified assessment, Azure recommendation and migration experience that delivers the following capabilities:

- A responsive user interface that provides an easy-to-navigate step-by-step wizard to deliver an integrated assessment, Azure recommendation and migration experience.
- An enhanced assessment engine that can assess SQL Server instances and identify databases that are ready for migration to Azure SQL Managed Instance or SQL Server on Azure Virtual Machines.
- An Azure recommender to generate right-sized recommendation for Azure SQL Managed Instance SKU or SQL Server on Azure Virtual Machines SKU to meet your database(s)' performance requirements with minimal cost.
- A reliable Azure service powered by Azure Database Migration service that orchestrates data movement activities to deliver a seamless migration experience with minimal downtime.
- The ability to run migrations in either online (for migrations that require minimal downtime) or offline (for migrations where downtime persists through the duration of the migration) modes to suit your business requirements.
- The flexibility to install self hosted integration runtime to provide your own compute for access to source SQL Server and backups in your on-premises environment.

To learn more about the concepts for migrating databases using the Azure SQL migration extension in Azure Data Studio, see [Database migrations using Azure Data Studio](#).

This article describes how to install the Azure SQL migration extension through Azure Data Studio.

## Prerequisites

If you don't have an Azure subscription, create a [free Azure account](#) before you begin.

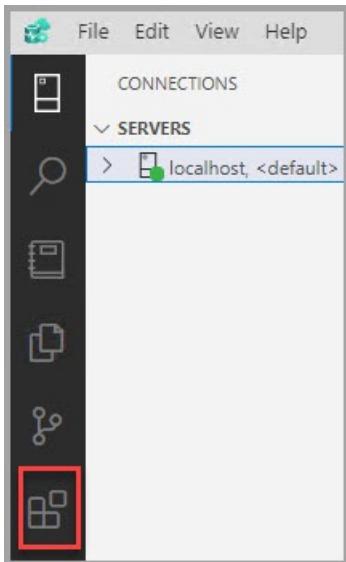
The following prerequisites are also required to install this extension:

- [Azure Data Studio installed](#).

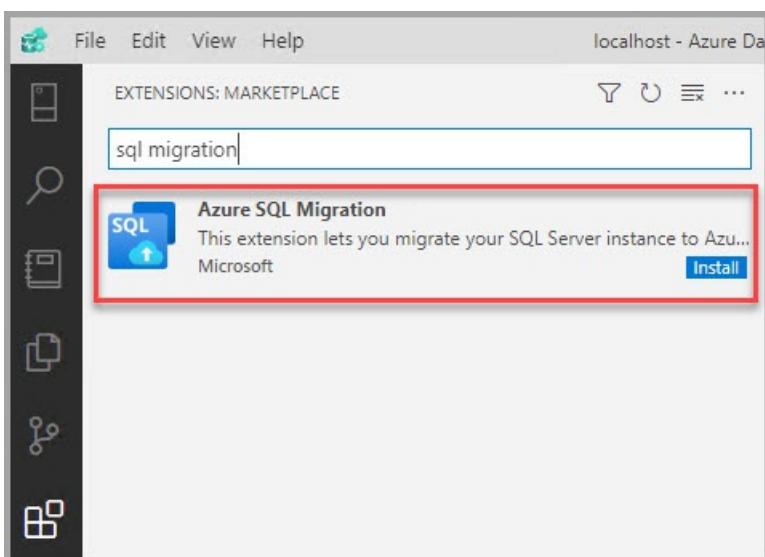
## Install the Azure SQL migration extension

To install the Azure SQL migration extension in Azure Data Studio, follow the steps below.

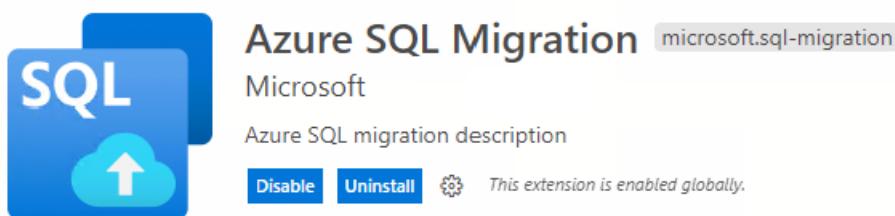
1. Open the extensions manager in Azure Data Studio. You can either select the extensions icon or select **Extensions** in the View menu.



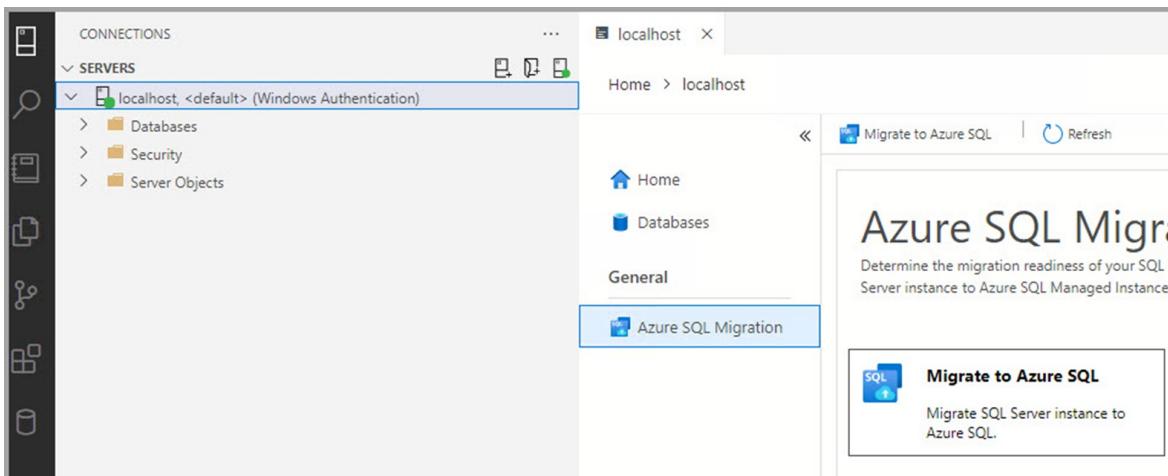
2. Type in *Azure SQL Migration* in the search bar.
3. Select the **Azure SQL Migration** extension and view its details.
4. Select **Install**.



5. You can see the Azure SQL migration extension in the extension list once installed.



6. You can connect to the SQL Server instance in Azure Data Studio. Right-click the instance name and select **Manage** to see the instance dashboard and the **Azure SQL Migration** extension landing page.



## Set up auto update for the extension

You can check for updates to the extension and have them automatically updated by configuring **Auto Update** in Azure Data Studio settings.

To enable auto updates:

1. Select the **Settings** icon in Azure Data Studio.
2. Select the checkbox under **User > Features > Extensions > Auto Check Updates**.
3. Select the dropdown under **User > Features > Extensions > Auto Update** and select either **All Extensions** or **Only Enabled Extensions**.

The screenshot shows the 'User' settings page in Azure Data Studio. The 'User' tab is selected. In the 'Extensions' section, the 'Features' and 'Extensions' buttons are highlighted with red boxes. A red arrow points to the 'Auto Check Updates' section, which contains a checked checkbox with the description: 'When enabled, automatically checks extensions for updates. If an update is available, it will be downloaded and installed automatically.' Another red arrow points to the 'Auto Update' section, which shows a dropdown menu with three options: 'All Extensions', 'All Extensions true', and 'Only Enabled Extensions onlyEnabledExtensions'. The 'Only Enabled Extensions onlyEnabledExtensions' option is highlighted with a red box.

### NOTE

If you want to update the extension manually, you can disable **Auto Update** and install the updates from the extension in the Marketplace.

## Features

### Azure SQL target readiness assessment and database migrations

The Azure SQL migration extension supports assessment, generate Azure recommendations and database migrations for the following Azure SQL targets.

- SQL Server on Azure Virtual Machines (SQL VM)
- Azure SQL Managed Instance (SQL MI)

## Migration modes

The following migration modes are supported for the corresponding Azure SQL targets.

- **Online** - The source SQL Server database is available for read and write activity while database backups are continuously restored on target Azure SQL. Application downtime is limited to duration for the cutover at the end of migration.
- **Offline** - The source database cannot be used for write activity while database backup files are restored on the target Azure SQL database. Application downtime persists through the start until the completion of the migration process.

## Support matrix

AZURE SQL TARGET	MIGRATION MODE
Azure SQL Managed Instance	Online
Azure SQL Managed Instance	Offline
SQL Server on Azure VM	Online
SQL Server on Azure VM	Offline

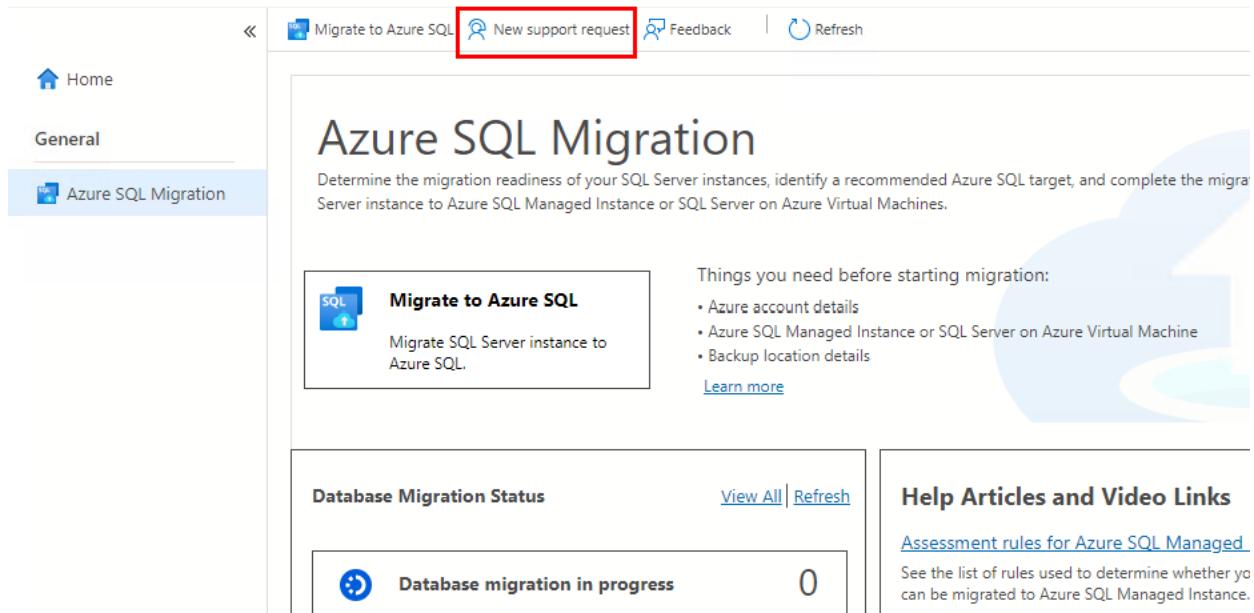
### TIP

For information on pre-requisites, features and migration workflow, see [Migration using Azure Data Studio](#)

## Get help from Microsoft support

You can raise a support request for a Microsoft engineer to assist if you encounter issues or errors with your database migrations using this extension.

Select **New support request** on the extension to navigate to the Azure portal where you can submit a support request.



The screenshot shows the Azure SQL Migration extension interface. At the top, there's a navigation bar with 'Home' and 'General' tabs, and a 'Migrate to Azure SQL' button with a red box around it. Below the navigation bar, the main title is 'Azure SQL Migration'. A sub-section titled 'Migrate to Azure SQL' describes migrating SQL Server instances to Azure SQL Managed Instance or SQL Server on Azure VM. To the right, there's a list of 'Things you need before starting migration:' which includes 'Azure account details', 'Azure SQL Managed Instance or SQL Server on Azure Virtual Machine', and 'Backup location details'. There's also a 'Learn more' link. At the bottom, there are sections for 'Database Migration Status' (showing 'Database migration in progress' with a value of 0) and 'Help Articles and Video Links'.

**NOTE**

Select the **Feedback** button if you have any suggestions or feedback to improve the extension.

## Next steps

- [Database migrations using Azure Data Studio](#)
- [Download Azure Data Studio](#)
- [Azure Data Studio release notes](#)
- [Azure Data Studio extensions](#)

# Azure Monitor Logs extension for Azure Data Studio (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

The Azure Monitor Logs extension for [Azure Data Studio](#) enables you to connect and query a [Log Analytics workspace](#).

This extension is currently in preview.

## Prerequisites

If you don't have an Azure subscription, create a [free Azure account](#) before you begin.

The following prerequisites are also required:

- [Azure Data Studio installed](#).
- [Log Analytics workspace](#).

## Install the Azure Monitor Logs extension

To install the Azure Monitor Logs extension in Azure Data Studio, follow the steps below.

1. Open the extensions manager in Azure Data Studio. You can either select the extensions icon or select **Extensions** in the View menu.
2. Type in *Monitor logs* in the search bar.
3. Select the **Azure Monitor Logs** extension and view its details.
4. Select **Install**.



### Azure Monitor Logs microsoft.azuremonitor

Microsoft | Repository | License | v0.1.0

Azure Monitor extension for Azure Data Studio

[Install](#)

## How to connect to a Log Analytics workspace

### Find your Log Analytics workspace

Find your Log Analytics Workspace in the [Azure portal](#), then find the Workspace ID.

myloganalyticsworkspace

Log Analytics workspace

Search (Ctrl+F)

Delete

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Settings

Resource group (change) : azuremonitorlogsmg

Status : Active

Location : South Central US

Tags (change) Click here to add tags

Workspace Name : myloganalyticsworkspace

Workspace ID : 0072c3ec-927d-4706-abab-bc4d0f06efaa

Pricing tier : Pay-as-you-go

Access control mode : Use resource or workspace permissions

Operational issues : OK

## Connection details

To set up an Azure Monitor Log workspace, follow the steps below.

1. Select **New connection** from the **Connections** pane.
2. Fill in the **Connection Details** information.
  - a. For **Connection type**, select *Azure Monitor Logs*.
  - b. For **Workspace ID**, enter in your Log Analytics Workspace ID.
  - c. For **Authentication type**, use the default - *Azure Active Directory - Universal with MFA account*.
  - d. For **Account**, use your account information.
  - e. For **Database**, select the same Workspace ID.
  - f. For **Server group**, use *Default*.
    - a. You can use this field to organize your servers in a specific group.
  - g. For **Name (optional)**, leave blank.
    - a. You can use this field to give your server an alias.

**Connection Details**

Connection type	Azure Monitor Logs
Workspace ID	0072c3ec-927d-4706-abab-bc4d0f06efaa
Authentication type	Azure Active Directory - Universal with MFA support
Account	(dropdown menu)
Database	myloganalyticsworkspace (0072c3ec-927d-4706-abab-bc4d0f06efaa)
Server group	<Default>
Name (optional)	(dropdown menu)
Advanced...	
<b>Connect</b> <b>Cancel</b>	

## How to query a Log Analytics Workspace in Azure Data Studio

Now that you have set up a connection to your Log analytics workspace, you can query the workspace using Kusto (KQL).

To create a new query tab, you can either select **File > New Query**, use **Ctrl + N**, or right-click the database and select **New Query**.

Once you have your new query tab open, then enter your Kusto query.

There are two main tables in Azure Log Analytics (Azure Monitor Logs) workspace that capture Azure SQL events for an Azure SQL database:

1. [AzureDiagnostics](#)
2. [AzureMetric](#)

To execute the samples below, you must be logged in to an [Azure SQL database](#).

Here are some samples of KQL queries:

```
AzureDiagnostics  
| summarize count() by OperationName
```

```
AzureDiagnostics  
| where LogicalServerName_s == "<servername>"  
| summarize count() by Category
```

**NOTE**

Allow about 15 minutes before the log results appear.

For more information about writing Azure Monitor Logs, visit [Azure Monitor documentation](#)

## Next steps

- [Create and run a notebook with Azure Monitor Logs](#)
- [Create diagnostic settings to send platform logs and metrics to different destinations](#)
- [SQL to Kusto cheat sheet](#)
- [What is Azure Monitor Logs?](#)
- [Using SandDance visualizations](#)

# Azure Cosmos DB API for MongoDB extension (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

The Azure Cosmos DB API for MongoDB extension for Azure Cosmos DB (Preview) enables you to manage and query your MongoDB resources using the capabilities of Azure Data Studio.

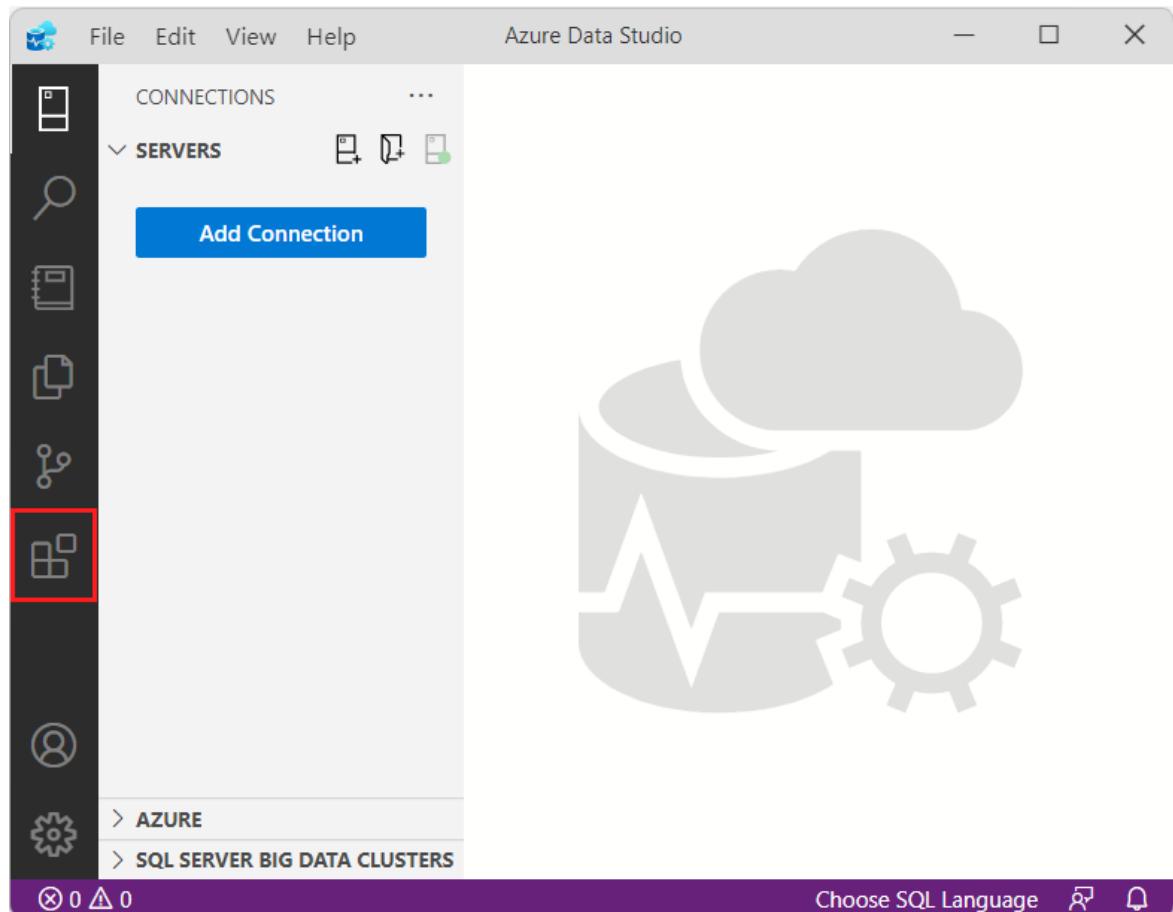
The Azure Data Studio functionality available for Azure Cosmos DB API for MongoDB includes:

- Connection manager & query editor
- Provisioning and scaling containers
- [Integrated terminal](#)

## Install the Azure Cosmos DB API for MongoDB extension (Preview)

If you don't already have Azure Data Studio installed, see its [install instructions](#).

1. Select the extensions icon from the sidebar in Azure Data Studio.



2. Type *mongo* into the search bar. Select the **Azure Cosmos DB API for MongoDB extension**.
3. Select **Install**. Once installed, select **Reload** to activate the extension in Azure Data Studio.

## Next steps

Learn how to connect and query [Azure Cosmos DB API for MongoDB](#).

# Database Migration Assessment for Oracle extension (Preview)

9/26/2022 • 5 minutes to read • [Edit Online](#)

The Database Migration Assessment for Oracle extension in Azure Data Studio helps you assess an Oracle workload for migrating to SQL. The extension identifies an appropriate Azure SQL target with right-sizing recommendations, and how complex the migration can be.

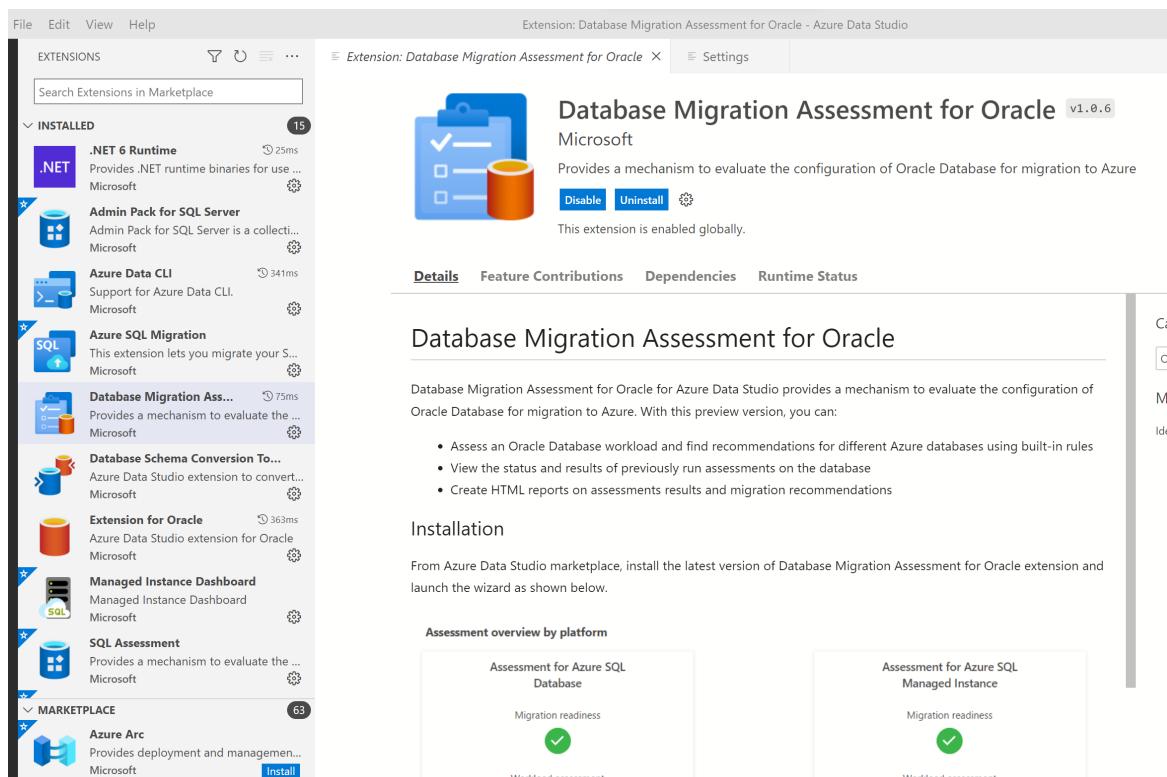
## Pre-requisites

- An [Oracle database](#) running version 10 g or higher.
- [Azure Data Studio version 1.37 or above](#).
- Oracle user should have CONNECT and SELECT ANY DICTIONARY privileges assigned.

## Install Azure Data Studio extension

Follow the steps below to install the Database Migration Assessment for Oracle extension in Azure Data Studio.

1. Open the extensions manager in Azure Data Studio. You can select the extensions icon or select **Extensions** in the View menu.
2. Type *Oracle* in the search bar.
3. Select the **Database Migration Assessment for Oracle** extension and view its details.
4. Select **Install**.

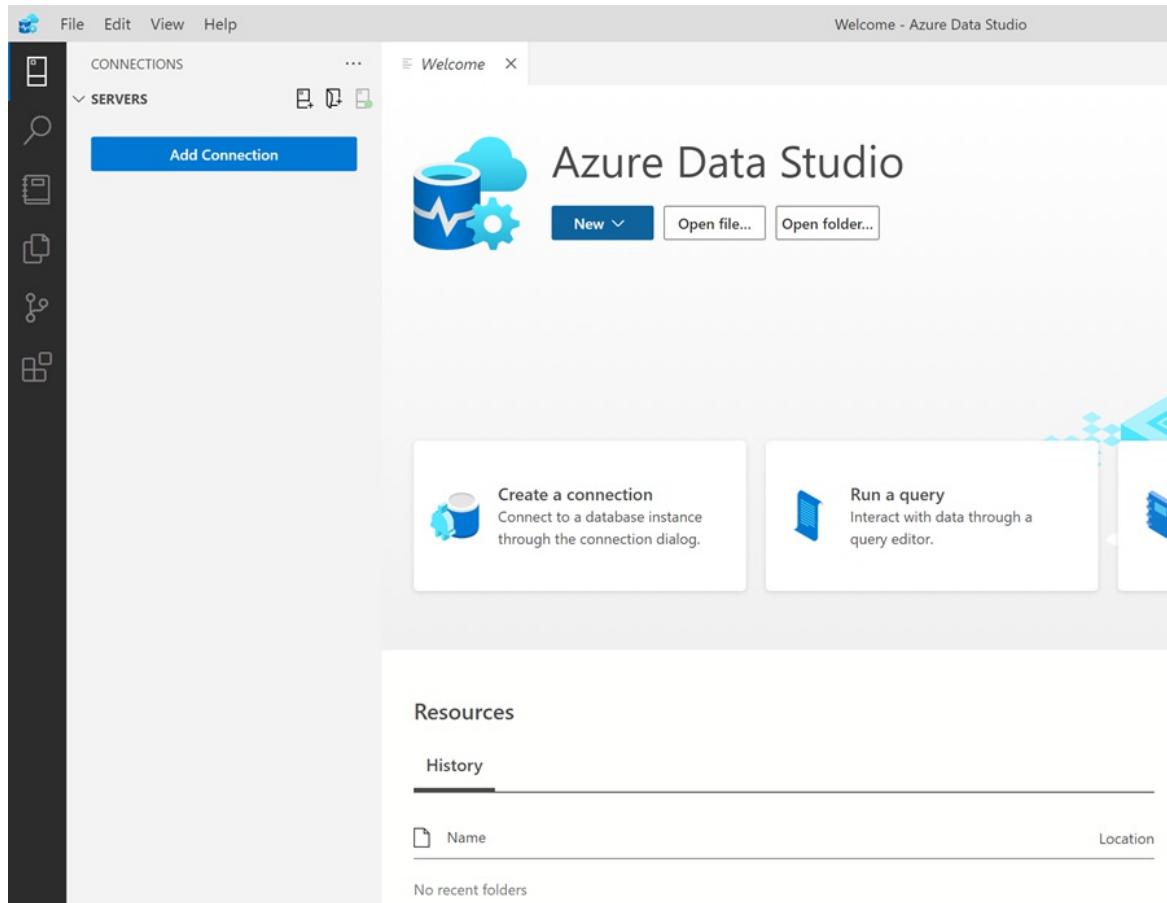


## Run assessment

Once the assessment extension installs, the next step is to connect to Oracle your database, collect metadata

information from your Oracle instance and generate an assessment report.

1. Go to the connections icon in the menu bar.



2. In the **Connection Details**, fill out the fields.

- a. In **Connection type** field, select **Oracle**.
- b. In the **Data Source** field, type in your Oracle server name and instance.
  - a. You can provide the TNS name as well. (Make sure that `ORACLE_HOME` environment variable is set and TNSNAMES.ORA file is located in the `<ORACLE_HOME>/network/admin` folder.)
  - c. In the **User Id** field, provide the database username.
  - d. In the **Password** field, provide the database password.

## Connection

Recent

Browse

 Clear List

 localhost/orcl (system)

## Connection Details

Connection type

Oracle

Data Source \*

localhost/orcl

User Id \*

system

Password

Remember password

Server group

<Default>

Name (optional)

Demo

Advanced...

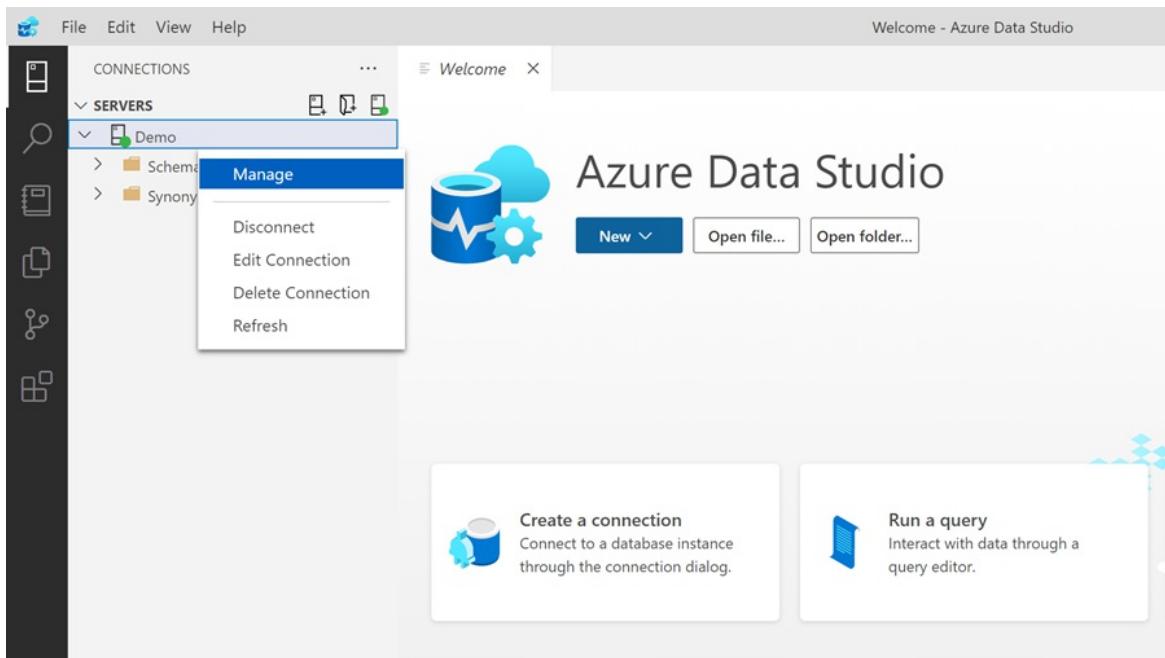
Connect

Cancel

3. Select **Connect**.

4. Now, a new connection appears in the connection details menu.

5. Right-click on the Oracle connection and select **Manage**.



## 6. Select Migration Assessment.

Using this assessment, you will be able to evaluate different target Azure database platforms and select the best using factors like migration readiness, feature compatibility and optimized operational cost. You can provide default assessment settings using the "Settings" tab and then run a new assessment.

**Run new assessment**  
Run assessment of your oracle database.

**Last five assessments**

No assessment has been done.  
Run a new assessment.

**Run new assessment**

**Help Articles**

[Migrate Oracle to Azure SQL](#)  
This guide helps you to migrate your Oracle schema to Azure SQL.

[Migration guide: Oracle to Azure SQL](#)  
This guide teaches you to migrate your Oracle schemas to Azure SQL.

[Migration guide: Oracle to Azure SQL Data](#)  
This guide teaches you to migrate your Oracle schemas to Azure SQL Database...

## 7. Select Run new Assessment.

The screenshot shows the 'Database Migration Assessment for Oracle' dashboard. At the top, there's a navigation bar with 'Home', 'General', and 'Migration Assessment' (which is selected). Below the navigation is a toolbar with 'Refresh', 'Help + Support', and 'Feedback'. The main content area has a title 'Database Migration Assessment for Oracle' and a sub-instruction: 'Using this assessment, you will be able to evaluate different target Azure database platforms and select the best using factors like migration readiness, feature compatibility and optimized operational cost. You can provide default assessment settings using the "Settings" tab and then run a new assessment'. There are two main sections: 'Run new assessment' (with a blue icon) and 'Last five assessments' (with a grey icon). The 'Last five assessments' section contains the message 'No assessment has been done. Run a new assessment.' and a 'Run new assessment' button. To the right, there's a 'Help Articles' sidebar with three links: 'Migrate Oracle to Azure SQL' (with a blue icon), 'Migration guide: Oracle to Azure SQL' (with a blue icon), and 'Migration guide: Oracle to Azure SQL Data' (with a blue icon).

8. Provide your assessment details.

- a. In the **Assessment name** field, enter a title.
  - a. For example, *demo1*.
- b. Enter your **Assessment setting**.
  - a. In the **Target Platform** field, enter the destination migration database.
    - a. For example, **SQL**.
  - b. In the **Scale factor** field, enter the multiplier value.
    - a. If the recommended SKU needs to consider other peak load, the scale factor multiplier should be greater than 1. Example: Burst Load, Seasonal usage, future capacity planning etc. Whereas, when partial Oracle schema workload is considered migrated, then the multiplier should be less than 1.
  - c. In the **Percentile utilization** field, enter the percentile value for sizing the Azure target.
    - a. The percentile value of the performance sample set to be considered for sizing the Azure target.

## Oracle Assessment

Run a new assessment for your oracle database

Enter the information below to start Oracle Assessment

Assessment name \* ⓘ

demo1

### Assessment settings

Select the assessment settings. Assessment report will be generated based on these settings.

Target platform \* ⓘ

SQL

Scale factor \* ⓘ

1.0

Percentile utilization \* ⓘ

95th percentile

9. Now, you see the new assessment in the last five assessments section.

The screenshot shows the Azure Database Migration Assessment for Oracle dashboard. On the left, there's a navigation sidebar with 'Home', 'General', and 'Migration Assessment' selected. The main content area has tabs for 'Dashboard' and 'Assessments', with 'Assessments' selected. Below the tabs are buttons for 'Refresh', 'Help + Support', and 'Feedback'. The main title is 'Database Migration Assessment for Oracle'. A call-to-action button 'Run new assessment' with the sub-instruction 'Run assessment of your oracle database.' is visible. To the right, there's a 'Last five assessments' section and a 'Help Articles' section. The 'Last five assessments' table has one row for 'demo1' with a status of 'In progress'. The 'Help Articles' section lists three guides: 'Migrate Oracle to Azure SQL', 'Migration guide: Oracle to Azure SQL', and 'Migration guide: Oracle to Azure SQL Data'. Each article has a brief description and a link icon.

Assessment name	Status
demo1	In progress

**Last five assessments**

Assessment name	Status
demo1	In progress

**Help Articles**

- [Migrate Oracle to Azure SQL](#)  
This guide helps you to migrate your Oracle schema to Azure SQL.
- [Migration guide: Oracle to Azure SQL](#)  
This guide teaches you to migrate your Oracle schemas to Azure SQL.
- [Migration guide: Oracle to Azure SQL Data](#)  
This guide teaches you to migrate your Oracle schemas to Azure SQL Database...

10. On Click on the assessment link to view assessment details page. You can view the latest assessment status.

The screenshot shows the Azure portal's Assessment page for an assessment named "demo1".

**Assessment Details:**

- Host name: ORACLE12C
- Version: 12.2.0.1
- Assessment status: In progress
- Assessment start time: 27/4/2022, 01:22
- Assessment name: demo1
- Percentile utilization: 95th percentile
- Instance name: orcl
- Edition: 12c Enterprise Edition
- Performance data collection window: --
- Assessment time taken: --
- Target platform: SQL
- Scale factor: 1.0

**Assessment Steps:**

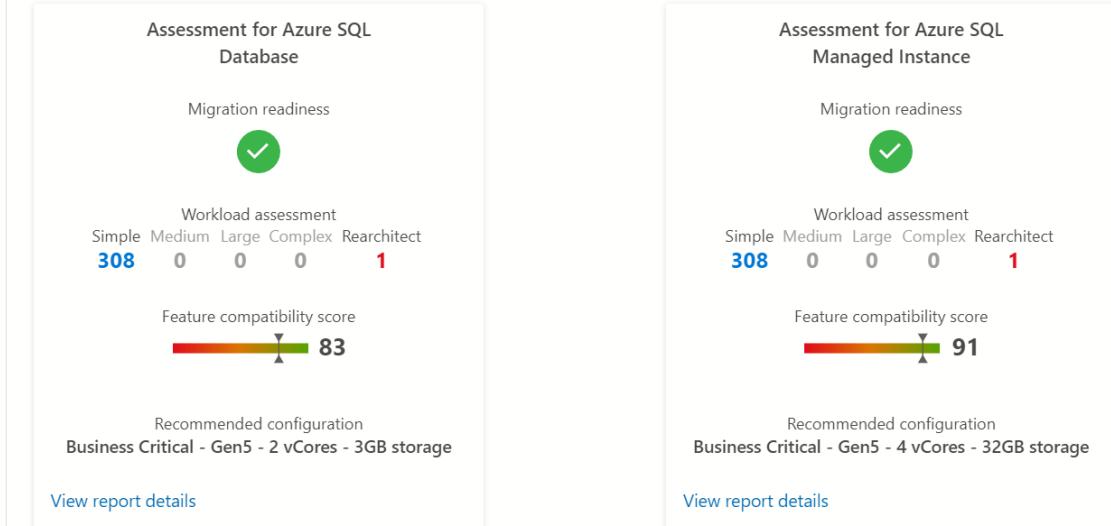
Assessment steps	Status	Time taken
Static data collection	Success	6 seconds
Feature data collection	Warning	5 seconds
Performance data collection	In progress	--
Schema workload collection	Waiting	--
Feature compatibility assessment	Waiting	--
Schema workload assessment	Waiting	--
SKU recommendation	Waiting	--

- Click refresh or wait until the Assessment completes. The assessment status page refreshes frequently. The default value is 15 seconds. The status updates to show you 1 of 4 statuses - **success**, **failed**, **in-progress** or **canceled**.

## View assessment

Once the Assessment is complete, a consolidated output is generated for each Azure SQL target. Currently, these targets include **SQL Server on Azure Virtual Machines**, **Azure SQL Database**, and **Azure SQL Managed Instance**.

Assessment start time	: 21/4/2022, 18:15	Assessment time taken	: 2 minutes
Assessment name	: parallel1	Target platform	: SQL
Percentile utilization	: 95th percentile	Scale factor	: 1.0

**Assessment overview by platform**

Each card has three sections. The card shows the overall feasibility of the migration to the SQL target. Also drills down on various workloads under various migration complexity categories. The feature compatibility section that provides the feature assessment review result. SKU recommendation provides the proper sizing of the target.

When you click on view detail report, it first shows the summary of the Assessment.

✓ Download report

Summary Database details SKU recommend... Feature compat...

ORACLE12C | abcd - Azure SQL Managed Instance | Summary

✓ Essentials

**Assessment overview**

**Assessment for Azure SQL Managed Instance**

Migration readiness ✗

Workload assessment

Simple	Medium	Large	Complex	Rearchitect
19	1	0	3	3

Feature compatibility score 91

Recommended configuration  
BusinessCritical - Gen5 - 4 vCores - 32GB storage

**Summary**

The assessed Oracle workload could be converted to the Azure SQL Managed Instance with low to medium confidence level.

The assessment module covers the database storage objects and code objects. Database storage objects include schemas, tables, indexes, constraints, sequences, synonyms, types, clusters, and database links. Database code objects include procedures, functions, triggers, views, materialized views, packages, and its associated constructs. There are 19 simple, 1 medium, 0 large, 3 complex and 3 rearchitect schema workload with a total of estimated 25263 hours for converting to the Azure SQL Managed Instance target.

The complexity of the Migration readiness is derived from a statistical model applied on the workload summary. Each category is decided based on the following factors:

Code objects	0	100	200	400	800 or greater
Storage objects	0	500	1000	2000	3000 or greater
Efficient estimated hours	0	350	700	1400	2300 or greater
	Simple	Medium	Large	Complex	Rearchitect

For more accurate code conversion complexity, kindly use [Sql Server Migration Assistant for Oracle](#)

The following database details tab provides the breakup per schema basis. It shows the list of schemas are discovered, the migration feasibility, and the estimated time to convert schema in hours.

Download report

Summary Database details SKU recommend... Feature compat...

### abcd - Azure SQL Managed Instance | Database details

**Database summary**

Name	: ORCL	Database role	: Primary
Storage used	: 7.09 GB	Number of schema	: 26
Number of owners	: 45	Number of objects	: 43038
Log mode	: Noarchivelog	Open mode	: Read write
Flashback on	: No	RAC enabled	: No

**Workload summary**

Schema name	Tables	SP, Triggers, Functions, Views	Collection Types	Packages	JSON Column	Materialized Views	Data Size (MB)	Complexity	Estimated hours
ANONYMOUS	2151	26	0	0	0	0	7	Simple	918
C##DEEP	2150	26	0	0	0	0	7	Simple	918
C##DOCTEAM	2150	26	0	0	0	0	7	Simple	918
C##MOVIEDB	2173	72	5	2	0	6	26	Simple	987
C##PRASANNA	4	2	0	0	0	0	1	Simple	3
C##PROXY_VARSHA	0	0	0	0	0	0	0	Simple	0
C##RAMYA	2124	2	14	11	0	0	2	Complex	931
C##SRIVEN	1	0	0	0	0	0	1	Simple	1
C##SSMA	0	0	0	0	0	0	0	Simple	0
C##TESTDMAOEXT	0	0	0	0	0	0	0	Simple	0
C##TESTODA10	3016	849	52	12	12	0	2	Rearchitect	2186
C##TESTODA11	1778	150	13	11	36	12	4	Medium	950
C##TESTODA12	1823	250	23	28	55	54	9	Complex	1160
C##TESTODA13	1749	810	0	0	0	0	3	Rearchitect	1498

This estimation is based on a statistical model that applies to the object count, lines of code, enabled features and size of the database. For more accurate estimate on the code conversion, use [SQL Server migration Assistant for Oracle](#).

#### NOTE

The following system or Oracle built-in schemas are excluded in the workload assessment  
 'SYSTEM','CTXSYS','DBSNMP','EXFSYS','LBACSYS','MDSYS','MGMT\_VIEW'  
 'OLAPSYS','ORDDATA','OWBSYS','ORDPLUGINS','ORDSYS','OUTLN','SI\_INFORMTN\_SCHEMA','SYS  
 'SYSMAN','WK\_TEST','WKSYS','WKPROXY','WMSYS','XDB','DIP','MDDATA','ORACLE\_OCM',  
 'SPATIAL\_CSADMIN\_USR','SPATIAL\_WFSADMIN\_USR','XS\$NULL','PERFSTAT','SQLPLAIN','DMSYS','TSMSSYS','WKSYS',  
 'DVSYS','OJVMSYS','GSMADMIN\_INTERNAL','APPQOSSYS','DVFSYS','DVF','AUDSYS','MGMT\_VIEW','ODM','ODM\_MTR','TRA  
 CESRV','MTMSYS','OWBSYS\_AUDIT','WEBSYS','WK\_PROXY','OSE\$HTTP\$ADMIN','DBMS\_PRIVILEGE\_CAPTURE','CSMIG','MG  
 DSYS','SDE','DBSFWUSER','APEX','FLOW\_'

The SKU recommendation provides the suitable Azure SQL target, its service tier and the metric thresholds that have been used to provide the recommended SKU.

## abcd - Azure SQL Managed Instance | SKU recommendation

The recommended target SKU, monthly cost and estimated effort to remediate all issues is shown below:

### Recommended configuration

**BusinessCritical -**  
Gen5 - 4 vCores -  
32GB storage

#### ► Recommendation reason

The CPU requirement of 2 vCore(s) is within the range for this 4 vCore(s) SKU

The storage requirement of 7 GiB of data files is within the SKU limit. The closest provisioning that would match is 32 GiB

The memory requirement is of 6 GB which is within SKU limit

The IO latency requirement of 3.00 ms is within the threshold limits of Business Critical service tier

The IOPS requirement is 6 IOPS for data and log files which is within SKU limit

#### ► Source properties

CPU count	Memory (allocated) (GB)	OS version	Storage (GB)
2	6	Windows	8

The SKU recommendation evaluates various performance metrics - CPU, memory, IOPS, latency and storage. Based on the usage and the configuration data, the recommender provides the suitable SQL target and the appropriate service tier.

### NOTE

If automatic workload repository (AWR) feature is enabled at Oracle instance, then SKU recommender will use the DBA\_HIST\_ views to gather the performance metrics metadata. Otherwise, the recommender will use server configuration and other system views information for sizing the Azure SQL target.

The feature assessment provides the Oracle to Azure SQL mapped features and the effort required for migrating those capabilities to Azure SQL target.

[Download report](#)

## demo1 - Azure SQL Database | Feature compatibility

Overall feature compatibility score: 54 | Total Features: 79

The list below shows the features of source Oracle database, and their compatibility with target. Click to see details.

act

Feature name	Azure SQL equivalent	Service Tier	Feature migration compatibility
Active Data Guard	None	general purpose, business critical	0
Data Redaction	Dynamic Data Masking	general purpose, business critical	5

### Data Redaction

Dynamic data masking helps prevent unauthorized access to sensitive data by enabling customers to specify how much sensitive data to reveal with minimal impact on the application layer. With I data in the database is not changed. DDM is easy to use with existing applications, since masking rules are applied in the query results. Many applications can mask sensitive data without modifying queries.

You can download the report for offline viewing by selecting download combined reports or download

individual reports.

The screenshot shows the Azure Data Studio interface with the 'Demo' tab selected. On the left, there's a navigation sidebar with 'Home', 'General', and 'Oracle Assessment' selected. The main area displays an 'Assessments' section for 'demo1'. It includes details like Host name, Assessment name, Performance data collection window, Assessment time taken, and Percentile utilization. Below this is an 'Assessment overview by platform' section, which is currently focused on 'Assessment for Azure SQL Database'. This section shows Migration readiness (green checkmark), Remediation actions (Simple, 11), and a Feature compatibility score of 54. It also recommends a GeneralPurpose - Gen5 - 2 vCores - 2048MB storage configuration. There's a link to 'View report details'. Another section for 'Assessment for Azure Virtual' is partially visible at the bottom. A 'Save As' dialog box is overlaid on the screen, prompting the user to save the report as an HTML file named 'DatabaseAssessmentCombinedReport\_20220320163949' to the 'admin' directory. The dialog has 'Save' and 'Cancel' buttons.

You can also cancel an ongoing assessment, delete an assessment and move assessments to another directory.

## Change Assessment Directory

1. Go to extension marketplace and search for Database Migration Assessment for Oracle.
2. Click on the manage icon and click extension settings.
3. Provide the new assessment path under Oracle Assessment: Assessment Path.

The screenshot shows the 'Extension Settings' page for the 'Database Migration Assessment for Oracle' extension. At the top, it shows the extension name and a 'Settings' button. Below this is a search bar with the query '@ext:microsoft.azuredatastudio-dma-oracle'. The main area is titled 'User' and contains a single setting: 'Oracle Assessment: Assessment Path', which is currently set to '@ext:microsoft.azuredatastudio-dma-oracle'. A note below the setting says 'Local path to store your assessments.' There are also 'Extensions (1)' and a 'Setting Found' link.

# Troubleshoot

## Logging

The extension has the errors, warning and other diagnostic logging written in the default log directory.

- Windows - C:\Users\<username>\dmaoracle\logs\
- Linux - ~/dmaoracle/logs
- Mac - /Users/<username>/dmaoracle/logs

By default, the extension stores last 7 log files.

To change the log directory, update LogDirectory property in the extension settings file.

Windows - C:\Users\<username>\azuredatastudio\extensions\microsoft.azuredatastudio-dma-oracle-<VersionNumber>\bin\service\Properties\ConfigSettings\extension-settings.json

Linux - ~/azuredatastudio/extensions/microsoft.azuredatastudio-dma-oracle-<VersionNumber>/bin/service/Properties/ConfigSettings/extension-settings.json

Mac - /Users/<username>/azuredatastudio/extensions/microsoft.azuredatastudio-dma-oracle-<VersionNumber>/bin/service/Properties/ConfigSettings/extension-settings.json

## Known issues

- Path provided does not exist.

Reason: Missing files or missing permission on the assessment folder.

Possible Solution:

- a. User has read and write permission on the assessment folder.
- b. If there is a missing file or folder, delete the assessment and generate a new assessment.

- Encountered connection timeout exception while interacting with Oracle.

Reason: Failed to Connect to Oracle Instance.

Possible Solution:

- a. Check if the port Oracle is running on is blocked by firewall rules.
- b. Perform tnsping and see if the Service ID gets resolved.

- Unable to access Database Migration Assessment reports for Oracle after upgrading to version 2.0

Reason: When Database Migration Assessment for Oracle extension is upgraded to latest version (v2.0 and above), the assessment reports will be inaccessible.

Workaround: Uninstall and install the prior version of Database migration assessment (v1.4.1) to download the previously assessed reports. Please follow the steps below to use previous version of the extension

### 1. Pre-requisites

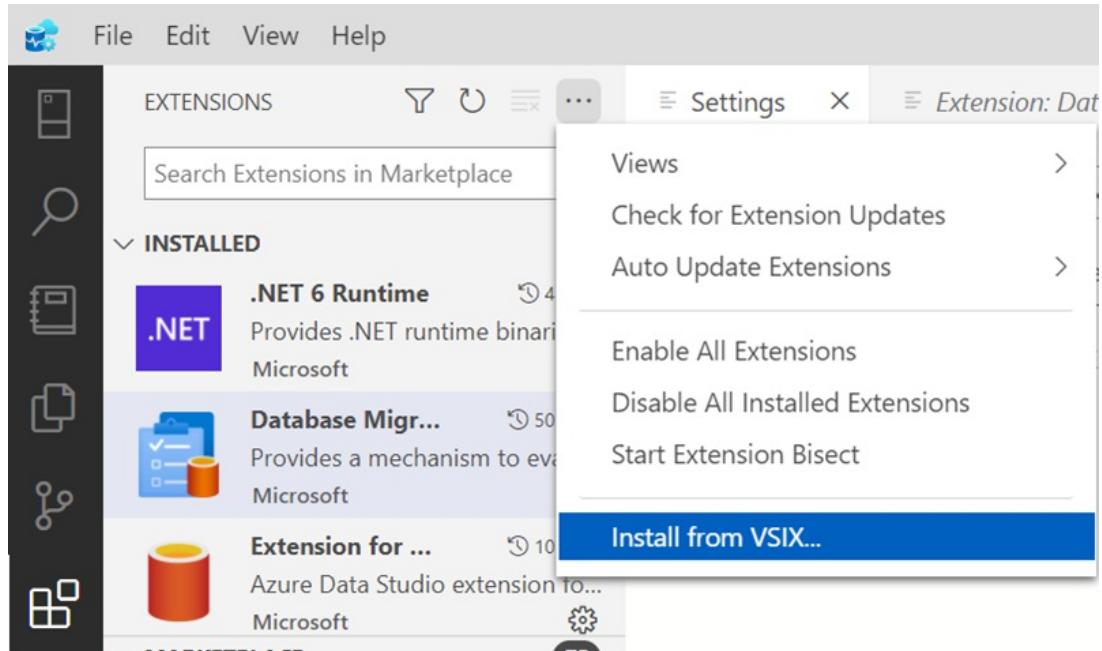
- Current assessment Directory: Path is given under Extension settings > Oracle Assessment: Assessment Path > Local path to store your assessments.
- Previous version build: Download the Previous [build](#) (v1.4.1) on local system.

### 2. Troubleshooting steps

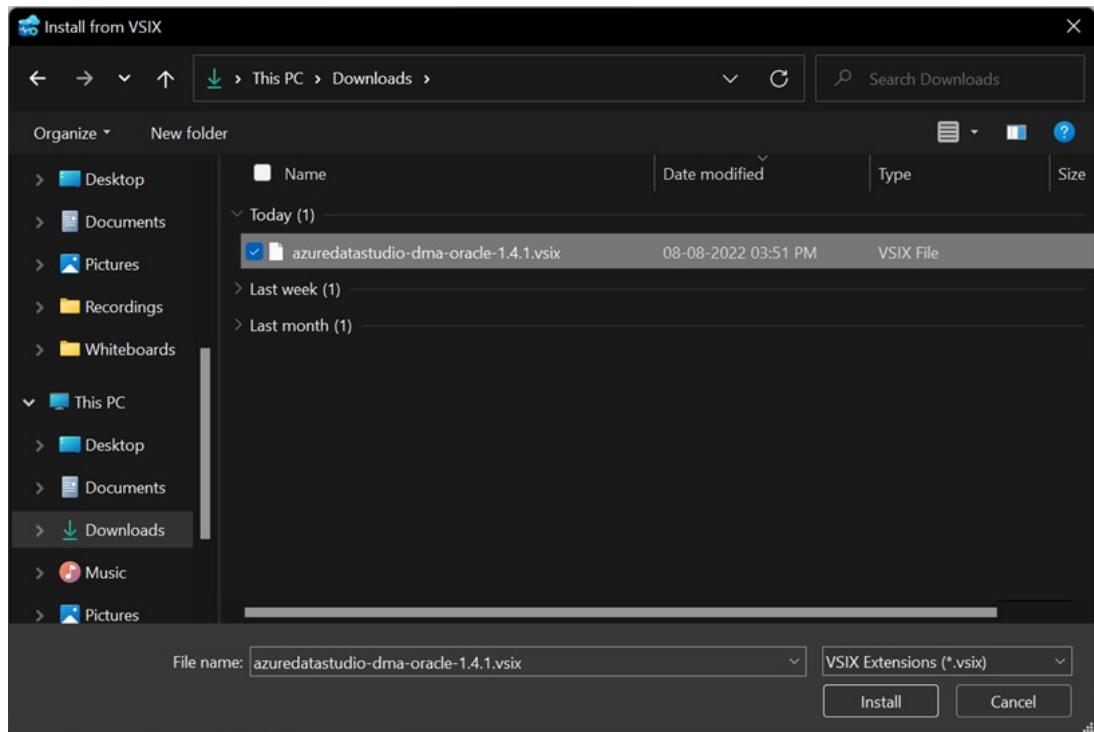
- Open the extensions manager in Azure Data Studio. You can select the extensions icon or select

Extensions in the View menu.

- To Install, click on View and More Actions (3 dots) as shown below > Install from VSIX. Screen capture shows view and More Actions to chose Install from vsix.



- Locate the VSIX file of v1.4.1 saved locally and click Install.



- Close and reopen the Azure data studio.

Ensure to set the extension assessment directory same as noted in pre-requisites.

- Using this extension version v1.4.1, saved assessments reports can be accessed, downloaded and saved in HTML format.

## Next steps

- [Azure SQL Migration extension](#)
- [Oracle extension](#)

- Data Schema Conversion
- Add extensions

# Database Schema Conversion Toolkit (DSCT)

9/26/2022 • 2 minutes to read • [Edit Online](#)

Database Schema Conversion Toolkit (DSCT) is a set of extensions for [Azure Data Studio](#) designed to automate database schema migration between different database platforms. This documentation explains the installation of the extensions, common scenarios, and what database platforms are compatible.

## Supported sources and targets

- [Oracle to Microsoft SQL](#)

Supports migration from Oracle database to Microsoft SQL Server, Azure SQL Database, Azure SQL Managed Instance, and Azure Synapse Analytics.

## Copyright notice

Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

## Next steps

- [Install Database Schema Conversion Toolkit \(Oracle to Microsoft SQL\)](#)

# Database Schema Conversion Toolkit (Oracle to Microsoft SQL) (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

The Database Schema Conversion Toolkit (Oracle to Microsoft SQL) is an [Azure Data Studio](#) extension for migrating Oracle database schemas to Microsoft SQL platform. While the [SQL Server Migration Assistant](#) is the Microsoft migration tool for enterprise customers and complex migrations (including Oracle customers with the ability to provision a Windows VM in their environment), the Database Schema Conversion Toolkit in Azure Data Studio enables the previously unsupported migration and modernization of Oracle workloads in exclusively Linux environments. Oracle data marts and reporting solutions are key opportunities to leverage the new Database Schema Conversion Toolkit.

This section introduces you to Database Schema Conversion Toolkit (Oracle to Microsoft SQL) and provides step-by-step instructions for migrating Oracle databases.

## Supported sources

Oracle databases version 11.2.0.4 and above are supported as the source.

### NOTE

Database Schema Conversion Toolkit (Oracle to Microsoft SQL) extension relies on the [Extension for Oracle](#) to establish connection to the Oracle database.

## Supported targets

The Database Schema Conversion Toolkit (Oracle to Microsoft SQL) extension currently supports the following targets:

- Microsoft SQL Server 2012 and above
- Azure SQL Database
- Azure SQL Managed Instance
- Azure Synapse Analytics

## Currently supported objects

The Database Schema Conversion Toolkit (Oracle to Microsoft SQL) supports automated conversion of the following database objects:

- Users (schemas)
- Basic table definitions (columns, indexes, constraints)
- Sequences
- Synonyms

**NOTE**

If your database contains a significant amount of objects that the Database Schema Conversion Toolkit does not currently support, you may consider using [SQL Server Migration Assistant for Oracle](#), which provides automated conversion for additional object types, but can only be used on Windows.

# Install and remove Database Schema Conversion Toolkit (Oracle to Microsoft SQL)

9/26/2022 • 2 minutes to read • [Edit Online](#)

To use Database Schema Conversion Toolkit (Oracle to Microsoft SQL) features, you will need to [install the Azure Data Studio](#) first.

Once Azure Data Studio is installed, select **Extensions** under the **View** menu and search for "Database Schema Conversion Toolkit" on the Extensions Marketplace. Select **Database Schema Conversion Toolkit (Oracle to Microsoft SQL)** and select the **Install** button in the search results list.

## NOTE

The Database Schema Conversion Toolkit (Oracle to Microsoft SQL) extension depends on **Extension for Oracle** and **SQL Database Projects** extensions. These extensions will be automatically installed.

## Upgrading Database Schema Conversion Toolkit (Oracle to Microsoft SQL)

Once available, latest version of the Database Schema Conversion Toolkit will be automatically installed, if extensions [Auto Update](#) option is enabled in Azure Data Studio.

## Removing Database Schema Conversion Toolkit (Oracle to Microsoft SQL)

When you have finished migrating databases, you might want to uninstall the Database Schema Conversion Toolkit extension.

To remove the Database Schema Conversion Toolkit extension, in Azure Data Studio, select **Extensions** under the **View** menu and search for "Database Schema Conversion Toolkit (Oracle to Microsoft SQL)" in the list of installed extensions. Select the extension and select **Uninstall** to remove the extension.

## NOTE

A restart of Azure Data Studio may be necessary to complete the removal process.

## Next steps

- [Convert Oracle database objects to Microsoft SQL](#)
- [Extend the functionality of Azure Data Studio - Add Azure Data Studio extensions](#)
- [Extend the functionality of Azure Data Studio - Uninstall an extension](#)

# Convert Oracle database objects to Microsoft SQL

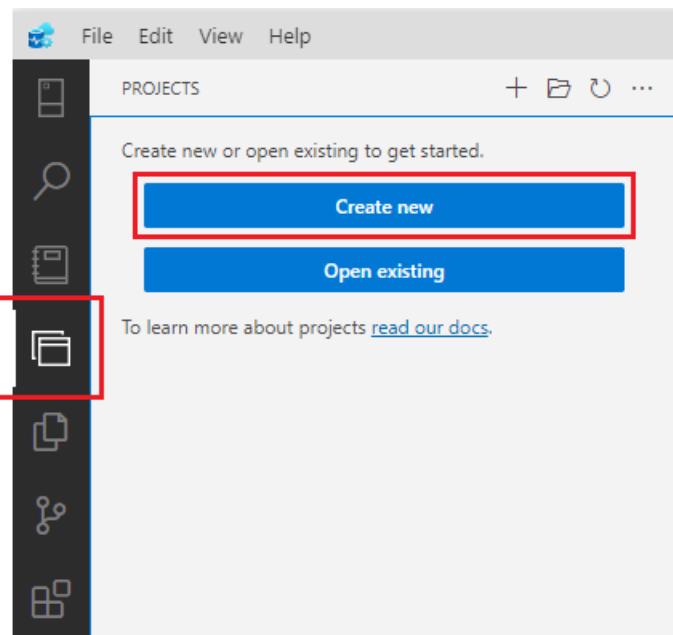
9/26/2022 • 3 minutes to read • [Edit Online](#)

Once Database Schema Conversion Toolkit (Oracle to Microsoft SQL) extension is [installed](#), you can start converting your Oracle database objects to the Microsoft SQL platform.

## Create target SQL Database project

First step in the conversion process is to prepare the target SQL Database project, where generated T-SQL scripts will be stored. In order to do this:

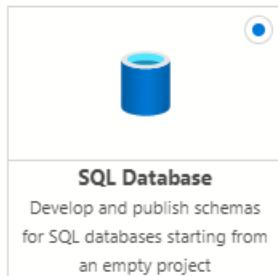
1. Navigate to the **Projects** view and select **Create new** to create new SQL Database project



2. In the **Create new project** dialog, select **SQL Database** as a **Type**, provide the **Name** for your project, choose the **Location** where you want your project to be stored and select desired **Target Platform**

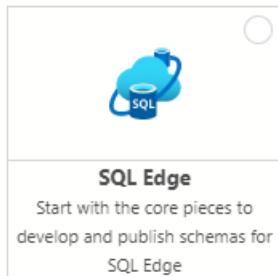
## Create new project

Type \*



### SQL Database

Develop and publish schemas for SQL databases starting from an empty project



### SQL Edge

Start with the core pieces to develop and publish schemas for SQL Edge

Name \*

SampleConversion

Location \*

C:\Demo\



Target Platform \*

SQL Server 2019

Create

Cancel

#### NOTE

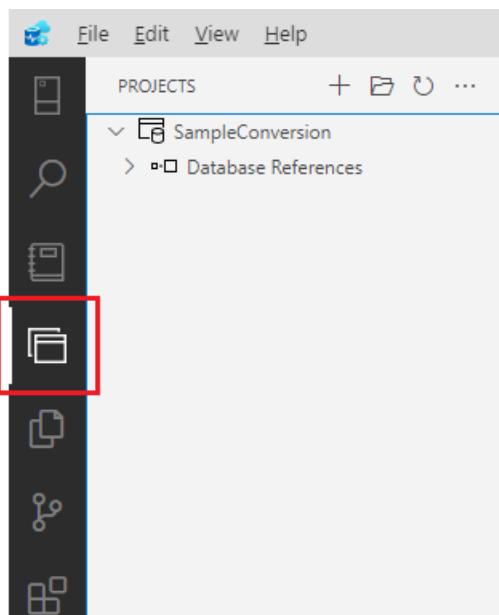
When targeting on-premises SQL Server, only SQL Server 2012 and above is supported by the **Database Schema Conversion Toolkit (Oracle to Microsoft SQL)**.

3. Select **Create** to complete the SQL Database project creation

#### NOTE

At this point Azure Data Studio may need to restart for the project to be created and added to the workspace.

4. You should be able to see an empty SQL Database project under the **Projects** view



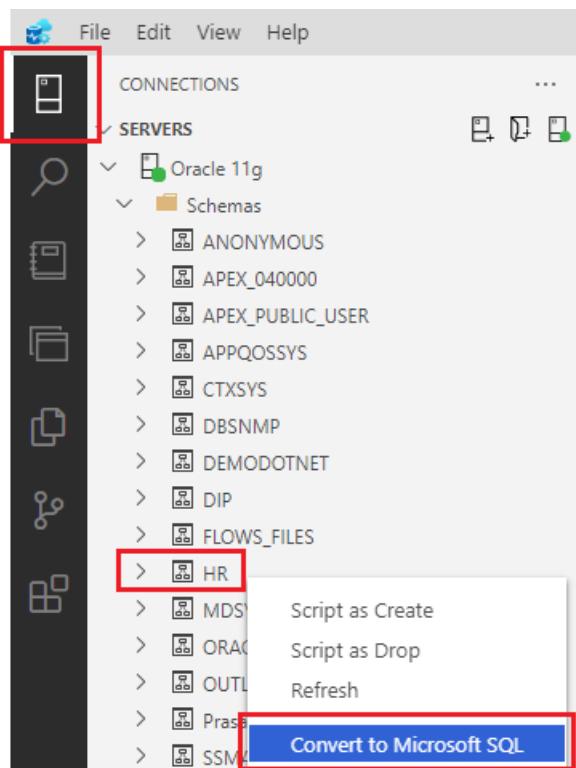
## Connect to the source Oracle database

Next step in the schema conversion process is to [connect to an Oracle database](#).

## Convert Oracle database objects

When connection to the Oracle database is established, you can start converting database objects.

1. In the **Connections** view, right-click an Oracle schema that you wish to convert to Microsoft SQL and select **Convert to Microsoft SQL**. If you want to convert multiple schemas at once, right-click the entire server element.

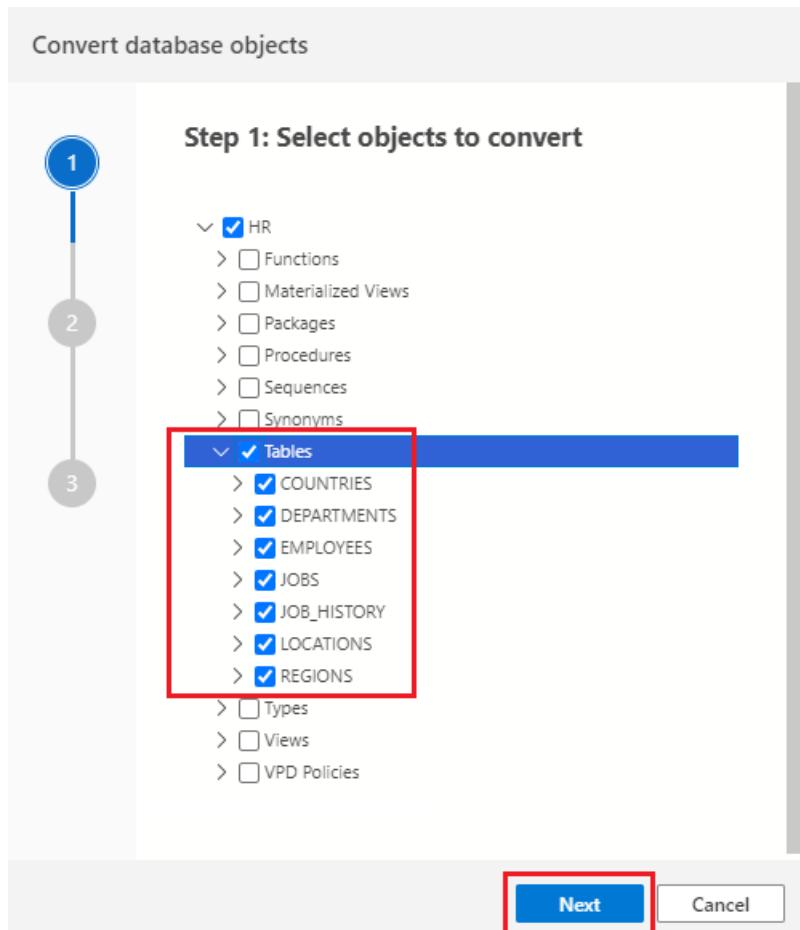


### NOTE

Conversion option is available at the server and schema levels only. If you need to convert individual objects, you will be able to narrow down the objects selection in the next step.

The Convert database objects wizard dialog will appear.

2. On the **Select objects to convert** step, select schema objects that you wish to convert.



**NOTE**

Parent objects will be automatically selected. For example, when selecting an index, parent table will also be included to make sure all necessary context is available to the conversion.

Select **Next** when done to advance to the next step.

3. On the **Select target database project** step, select previously created SQL Database project, where you want converted T-SQL scripts to be stored.

## Convert database objects

Step 2: Select target database project

1

2

3

SQL Database project to store converted objects \*

SampleConversion

Overwrite existing objects

Previous Next Cancel

If your SQL Database project already contains script files that were previously generated by the conversion, then you may consider selecting an **Overwrite existing objects** option to replace existing script files.

Select **Next** when done to advance to the next step.

4. On the **Conversion settings** step you can optionally load a [conversion configuration file](#).

Convert database objects

Step 3: Conversion settings

1

2

3

Configuration file

c:\Demo\SampleConversion\hr-to-dbo.json

Load

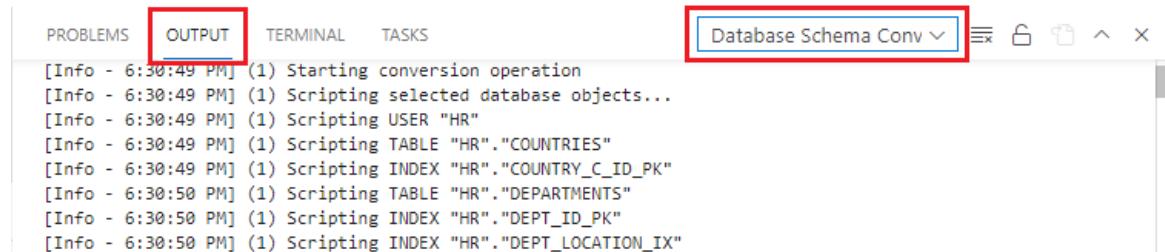
Previous Done Cancel

Select **Done** to complete the wizard and start the schema conversion.

5. Conversion task will be started.

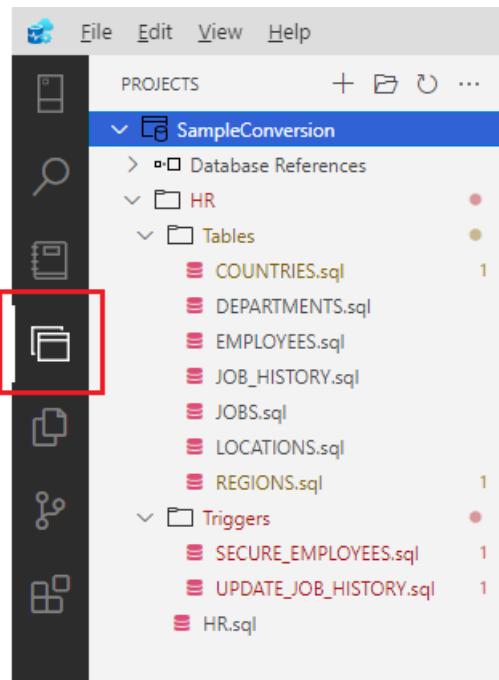


You can track conversion progress by switching to **Database Schema Conversion Toolkit (Oracle to Microsoft SQL)** on the Output panel.

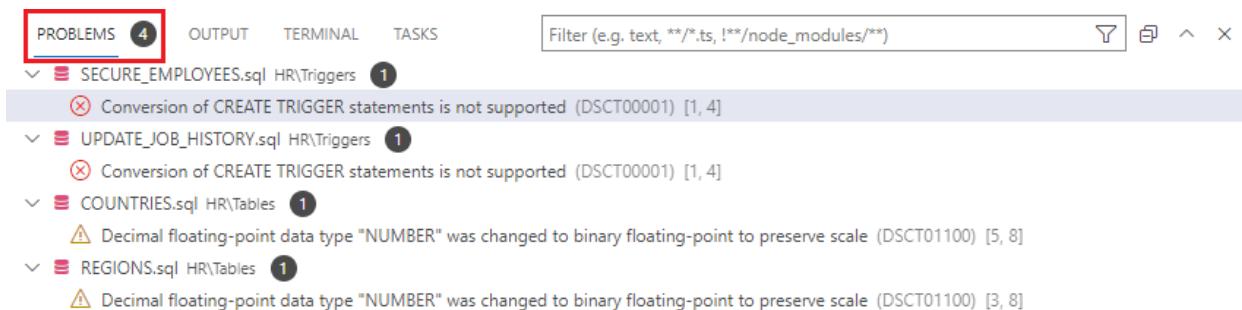


## Review converted scripts

Once conversion has completed, converted script files will be added to the target SQL Database project.



If there are any conversion messages, they will appear on the **PROBLEMS** panel and you can navigate to the conversion message location by selecting the message row.



Conversion messages will also be embedded as comments in the converted SQL scripts. Once addressed, the comments can be removed.

#### **NOTE**

You can close the Azure Data Studio at any time and reopen the SQL Database project later. In order to discover conversion messages in the SQL Database project that was converted previously, open **Command Palette...** from the **View** menu and run **DSCT (Oracle to Microsoft SQL): Discover problems** command.

## Deploy the target Microsoft SQL database

When all problems produced during conversion are addressed, you can [deploy your database](#) to the target server.

## Next steps

- [Extension for Oracle](#)
- [SQL Database Projects extension](#)

# Configure conversion

9/26/2022 • 5 minutes to read • [Edit Online](#)

The Database Schema Conversion Toolkit (Oracle to Microsoft SQL) is aiming to provide a solution to database schema migrations. While default conversion is fairly reasonable, there are cases when users may want to adjust some of the conversion settings to better suit their needs.

## Configuration user interface

When [performing a database schema conversion](#) through the Database Schema Conversion Toolkit conversion wizard UI, the **Conversion settings** step will allow you to tweak basic conversion settings.

In more complex scenarios custom JSON configuration file with advanced options can be provided, as explained in the next section.

## Advanced conversion configuration

There are some conversion settings that are not currently exposed through the user interface. These settings can be adjusted through the JSON configuration file.

Basic structure of the configuration file looks as following:

```
{  
  "options" : <simple-conversion-options>,  
  "dataTypeMappings": [  
    <data-type-mapping>,  
    ...  
  ],  
  "nameMappings": [  
    <object-name-mapping>,  
    ...  
  ]  
}
```

Following sections will cover each configuration region in more details.

### Simple conversion options

The `options` configuration region has the following schema:

```
{  
  "msSqlDialect":  
    "AzureSynapse"  
    | "AzureSqlDatabase"  
    | "AzureSqlManagedInstance"  
    | "SqlServer2012"  
    | "SqlServer2014"  
    | "SqlServer2016"  
    | "SqlServer2017"  
    | "SqlServer2019",  
  "quoteIdentifiers": true | false,  
  "isMsSqlCaseSensitive": true | false,  
}
```

Following table describes all possible configuration options in this region:

OPTION NAME	DESCRIPTION
<code>msSqlDialect</code>	Determines which Microsoft SQL platform dialect to use when converting the source object definitions. This option will be derived from the target SQL Database project and you should not need to set it explicitly.
<code>quoteIdentifiers</code>	Determines whether all identifiers should be quoted in converted SQL scripts. Default is <code>true</code> . It is recommended to set it to <code>true</code> , as quotation might be required when special characters are used in identifier names.
<code>isMsSqlCaseSensitive</code>	Controls whether <a href="#">DSCT01000</a> conversion message will be produced during conversion. This option will be derived from the default collation of the target SQL Database project and you should not need to set it explicitly.

## Data type mappings

The `dataTypeMappings` configuration region consists of multiple data type mapping records. Each data type mapping record has the following schema:

```
{
  "source": {
    "type": "<oracle-data-type-name>",
    "arguments": [
      <argument-value-matching-expression>,
      ...
    ]
  },
  "target": {
    "type": "<ms-sql-data-type-name>",
    "arguments": [
      <argument-value-expression>,
      ...
    ]
  }
}
```

The `source` section defines the source data type that is being mapped and consists of two parts:

- `type` is the name of the Oracle data type to map;
- `arguments` is the collection of matching expressions that will further filter a data type based on its arguments values.

The source `arguments` collection defines matching expressions for data type arguments based on their position. The collection should contain one string expression for each data type argument. Supported expressions are:

EXPRESSION	MEANING
<code>"&lt;number&gt;"</code>	Matches the exact value of an argument.
<code>"*"</code>	Matches special <code>*</code> argument value. For example, the first argument in the <code>NUMBER(*, 5)</code> data type definition.

EXPRESSION	MEANING
"X..Y"	Matches an argument value in the [x, y] range, where x and y can be either <number> or *. The * in the range expression means <i>unbound</i> . To match any argument value the "*..*" range expression can be used.

Some data types may have their arguments specified within the type name, for example

INTERVAL DAY (2) TO SECOND (6). In such cases, the type name would be INTERVAL DAY TO SECOND, while 2 and 6 are considered first and second arguments respectively.

The target section of the data type mapping record defines the Microsoft SQL data type that should be used in the target database and consists of two parts:

- type is the name of the Microsoft SQL data type to map to;
- arguments is the collection of expressions that define values for the target data type arguments.

The arguments collection defines data type arguments value expressions based on the arguments position. The collection should contain one string expression for each data type argument. Supported expressions are:

EXPRESSION	MEANING
"<number>"	Specifies an exact value of an argument.
"\$<number>"	Specifies that a value of the <number> source argument should be used. The index is 1-based. For example, \$2 will be replaced with the value of the second argument of the matched source data type.

The following example demonstrates how to map the VARCHAR2 Oracle data type that holds 4000 characters or less, to the NVARCHAR Microsoft SQL data type of the same length as the source data type:

```
{
  "source": {
    "type": "VARCHAR2",
    "arguments": [
      "*..4000"
    ]
  },
  "target": {
    "type": "NVARCHAR",
    "arguments": [
      "$1"
    ]
  }
}
```

#### IMPORTANT

Data type mappings should be defined from the least specific to the more specific, as they will be applied in reverse order. In other words, every subsequent data type mapping overrides (entirely or in part) previously defined mappings.

#### NOTE

Database Schema Conversion Toolkit comes with the built-in data type mappings that cover common scenarios, thus custom data type mappings are not required in most cases.

### Object name mappings

The `nameMappings` configuration region consists of multiple name mapping records. Each name mapping record has the following schema:

```
{  
    "source": [  
        {  
            "type":  
                "constraint"  
                | "index"  
                | "materializedview"  
                | "sequence"  
                | "synonym"  
                | "table"  
                | "tablecolumn"  
                | "tabletrigger"  
                | "user"  
                | "view",  
            "name": "<source-object-name>"  
        },  
        ...  
    ],  
    "target": "<target-object-name>"  
}
```

The `source` collection defines name parts of the source identifier that is being mapped. For example, to define the target name for a source schema, the following source collection may be used:

```
[  
    { "type" : "user", "name": "MySchema" }  
]
```

If you want to define target name for the source table, then two-part identifier name should be specified in the `source` collection as following:

```
[  
    { "type" : "user", "name": "MySchema" },  
    { "type" : "table", "name": "MyTable" }  
]
```

This will match a table with multi-part name `"MySchema"."MyTable"`. Most database objects will require multi-part names to be specified in the `source` collection.

Following table describes supported source object types:

TYPE	DESCRIPTION
<code>constraint</code>	Name of the constraint object
<code>index</code>	Name of the index object

TYPE	DESCRIPTION
materializedview	Name of the materialized view object
sequence	Name of the sequence object
synonym	Name of the synonym object
table	Name of the table object
tablecolumn	Name of the table column
tabletrigger	Name of the table trigger object
user	Name of the database schema
view	Name of the view object

The `target` property is always a simple string that defines new name for the source object that matches `source` multi-part identifier.

#### NOTE

It is not possible to change the schema for just one object. The `target` property only specifies a one-part name for the matching source object, not a multi-part name.

## Examples

Following example demonstrates entire configuration file that maps the `VARCHAR2` Oracle data type that holds 4000 characters or less, to the `NVARCHAR` Microsoft SQL data type of the same length as the source data type, while also replacing `HR` Oracle schema with `dbo` in the target database:

```
{  
  "dataTypeMappings": [  
    {  
      "source": {  
        "type": "VARCHAR2",  
        "arguments": [  
          "*..4000"  
        ]  
      },  
      "target": {  
        "type": "NVARCHAR",  
        "arguments": [  
          "$1"  
        ]  
      }  
    }  
  ],  
  "nameMappings": [  
    {  
      "source": [  
        { "type": "user", "name": "HR" }  
      ],  
      "target": "dbo"  
    }  
  ]  
}
```

When this configuration is used all converted objects from `HR` schema will be defined under the `dbo` schema and all matching references to the `VARCHAR2` data type will be replaced with the `NVARCHAR`.

# DSCT00000: Failed to parse SQL fragment (Error)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit isn't able to parse input SQL text.

## Background

The Database Schema Conversion Toolkit contains a custom implementation of the parser for different SQL dialects to understand and interpret definitions of objects in the source database. Such custom parser implementation may not support the entire spectrum of statements and clauses, or may have some discrepancies compared to the actual database engine. Database Schema Conversion Toolkit will produce a `DSCT00000` error in the converted output if an unsupported statement or clause is encountered.

## Possible remedies

Statements and clauses that cannot be parsed will need to be converted manually.

# DSCT00001: Conversion of statements is not supported (Error)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit isn't able to convert certain statements.

## Background

When it comes to automated database schema conversion, there are cases when some statements aren't converted. This can be because of multiple different reasons. The most common are:

- Conversion of a certain statement isn't yet implemented in the Database Schema Conversion Toolkit
- Conversion of statement cannot be automated because of lack of context
- Conversion of statement cannot be automated because target platform doesn't provide similar functionality, and this functionality cannot be emulated

## Example

To better understand the necessity of the context during conversion, consider the following example:

```
EXECUTE
    "SELECT SUM(AMOUNT) " +
    "FROM " + :SalesTableName
    "WHERE " + :FilterClause
```

In this case, parts of the SQL statement (source table name and filter clause) will be specified externally to the location where the `EXECUTE` statement is defined. To properly convert the above statement, DSCT will need to know runtime values of the parameters `:SalesTableName` and `:FilterClause` to ensure that the resulting statement is correct in the target database. While `EXECUTE` statement itself is widely supported across the database platforms and can be converted, the dynamic SQL string cannot be converted, as it has an incomplete definition in the given context.

Another example is related to the difference in supported features across different database platforms. Consider the following unique index:

```
CREATE UNIQUE INDEX IX_EMPLOYEES_ID
    ON EMPLOYEES (ID)
```

When converting this statement targeting Azure Synapse, a `DSCT00001` error message will be produced, as Azure Synapse doesn't currently support unique indexes.

## Possible remedies

Statements that cannot be automatically converted will need to be converted manually.

**NOTE**

When migrating from Oracle database to Microsoft SQL platform, you may also consider using [SQL Server Migration Assistant for Oracle](#), which provides automated conversion for an extended set of statements, but can only be used on Windows.

# DSCT00002: Conversion of syntax element is not supported (Error)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit isn't able to convert certain syntax elements.

## Background

When it comes to automated database schema conversion, there are cases when some clauses are not converted. This can be due to multiple different reasons.

The most common are:

- Conversion of a certain clause isn't yet implemented in the Database Schema Conversion Toolkit
- Conversion of the clause can't be automated because target platform doesn't provide similar functionality, and this functionality can't be emulated

## Example

To better understand how limitations of the target platform affect the conversion, consider the following table definition:

```
CREATE TABLE EMPLOYEES
(
    ID INT NOT NULL,
    NAME VARCHAR2(20),
    DEPARTMENT_ID INT
        REFERENCES DEPARTMENTS (ID)
)
```

When converting this table definition targeting Azure Synapse, a `DSCT00002` error message will be produced, as Azure Synapse doesn't currently support foreign key constraints.

## Possible remedies

Clauses that can't be automatically converted will need to be converted manually.

### NOTE

When migrating from Oracle database to Microsoft SQL platform, you may also consider using [SQL Server Migration Assistant for Oracle](#), which provides automated conversion for an extended set of clauses, but can only be used on Windows.

# DSCT0003: Identifier was not resolved (Error)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit isn't able to resolve an identifier reference.

## Background

To perform correct schema conversion, the Database Schema Conversion Toolkit needs to establish object references within the database schema. This process is referred to as "binding" or "linking". During this process, Database Schema Conversion Toolkit attempts to match object references with their declarations. If Database Schema Conversion Toolkit can't find a suitable declaration for the encountered object reference, then **DSCT0003** error message is produced.

## Example

Consider the below table definition:

```
CREATE TABLE EMPLOYEES
(
    ID INT NOT NULL,
    NAME VARCHAR(20),
    DEPARTMENT_ID INT
        REFERENCES DEPARTMENTS (ID)
)
```

Without any additional table definitions included in the conversion, Database Schema Conversion Toolkit will not be able to resolve the foreign key reference to **DEPARTMENTS.ID** and validate that its data type matches the **DEPARTMENTS\_ID** column, which the target database might require.

## Possible remedies

To eliminate these types of errors, make sure all referenced objects are included in the conversion.

# DSCT00004: Conversion of data type is not supported (Error)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit isn't able to convert certain data types.

## Background

When migrating database schema between different database platforms, not all data types from the source database can have a corresponding match in the new target database. The Database Schema Conversion Toolkit will attempt to pick the best match in the target database, based on the source data type definition, but sometimes it isn't possible, as none of the data types in the target database provide the same functionality as the source. In such cases a `DSCT00004` error message is produced.

## Example

Consider the below table definition using `INTERVAL` data type as an example:

```
CREATE TABLE EXAMPLE
(
    COLUMN1 INTERVAL YEAR TO MONTH
)
```

When converting this table definition targeting Microsoft SQL platform, a `DSCT00004` error message will be produced, as Microsoft SQL does not currently support `INTERVAL` data types.

## Possible remedies

Data type conversion issues will need to be addressed manually, with the knowledge of the application logic. Consider changing the data type to one of the supported types in the target database and updating references to it. In some cases a generic data type, like `VARCHAR`, may be used with a set of additional "accessor" routines (functions/procedures) to serialize and deserialize complex values.

# DSCT00005: Conversion of system identifier is not supported (Error)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit isn't able to convert certain built-in system identifiers.

## Background

Different database platforms provide different built-in functionality, including everything from catalog tables and views (for example, `sys.tables`, `sys.columns` in Microsoft SQL) to special functions and procedures. Not all built-in system objects from the source database have a corresponding match in the new target database. The Database Schema Conversion Toolkit will attempt to emulate the functionality of the source database, but sometimes it isn't possible. In such cases a `DSCT00005` error message is produced.

## Example

Consider the below query using Oracle's `SYS.ALL_UNUSED_COL_TABS` data dictionary:

```
SELECT
    TABLE_NAME
FROM
    SYS.ALL_UNUSED_COL_TABS
WHERE
    OWNER = 'TEST'
```

When converting this query targeting Microsoft SQL platform, a `DSCT00005` error message will be produced, as Microsoft SQL does not have a concept of unused table columns.

## Possible remedies

Review the converted code to determine whether this functionality is business-critical or does not apply to the new database platform. Depending on a certain scenario it may not be necessary to transfer the logic in question to the new database platform, like in the example above, where there is simply no concept of the unused columns in the new database platform. In other situations, the code that failed to convert might be a part of the application business logic which cannot be removed. In such cases, the logic will need to be converted manually.

# DSCT0100: Identifier is case-sensitive and may cause issues, if target database is not using CS collation (Warning)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit may produce identifier collision in converted code.

## Background

Databases may be configured to allow identifiers that are only different in case to be considered different, for example `foo` vs `FOO`. In case source database respects the casing of the identifier names, target database should be configured in the same way. If Database Schema Conversion Toolkit encounters an identifier where case matters, but target database is not configured to respect the casing, then `DSCT0100` warning message is produced.

## Example

Consider the below definition of the Oracle table:

```
CREATE TABLE "foo"
(
    ID INT
)
```

In this case, a table will be named lower-case `foo` and will be considered different to an unquoted name `foo` or an upper-case `FOO`. When Database Schema Conversion Toolkit encounters such identifiers, a warning message is produced.

## Possible remedies

To eliminate these warnings, consider changing the collation of the target database to case-sensitive. Alternatively, you can review all instances of the warning message to make sure such identifiers will not produce undesired side-effects when used within case-insensitive database.

# DSCT01001: Index will be defined in the schema of the target table (Info)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit may change the schema in which the index is defined.

## Background

Some database platforms allow indexes to be defined in a schema different than that of a target table. In Microsoft SQL indexes always exist in the same schema as a target table.

## Example

Consider the following table and index definitions as an example:

```
CREATE TABLE SCHEMA_A.EXAMPLE
(
    COLUMN1 INT
);

CREATE INDEX SCHEMA_B.IX_EXAMPLE_COLUMN1
    ON SCHEMA_A.EXAMPLE (COLUMN1)
```

Since table `EXAMPLE` is defined in `SCHEMA_A`, but the index `IX_EXAMPLE_COLUMN1` targeting said table is defined in `SCHEMA_B`, Database Schema Conversion Toolkit produces `DSCT01001` informational message.

## Possible remedies

In general, this shouldn't be a concern, but Database Schema Conversion Toolkit produces an informational message, so that you can review and make sure there are no other dependencies on the index location, or all dependencies are updated to work with the new location.

# DSCT01002: Column will allow NULL values to be inserted (Warning)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit may produce converted code that allows `NULL` values to be inserted.

## Background

When you define default value for a column, some database platforms allow the default to be used to replace explicitly inserted `NULL` values. In Microsoft SQL default value is never used when `NULL` value was explicitly provided and `NULL` is inserted, so the behavior will be different in these cases.

## Example

Consider the following table definitions as an example:

```
CREATE TABLE DEF_VALUE_NULL
(
    COLUMN1 INT DEFAULT ON NULL 42
)
```

The `ON NULL` clause will not be converted when targeting Microsoft SQL platform and `DSCT01002` warning message will be produced.

## Possible remedies

Review the converted code and make sure new behavior is acceptable. If the change in behavior is not acceptable, consider adding a table trigger to replace `NULL` values on insert.

# DSCT01100: Decimal floating-point data type was changed to binary floating-point to preserve scale (Warning)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit may change data type from decimal floating-point to binary floating-point.

## Background

Oracle database support decimal floating-point data types that allow higher scale than Microsoft SQL `decimal` / `numeric` data types. For example Oracle's `NUMBER` data type represents decimal floating-point and allows scale ranging from `-84` to `127`, compared to maximum allowed `38` for Microsoft SQL `decimal` / `numeric` data types. In order to preserve the scale in such cases, the Database Schema Conversion Toolkit will change the data type to `float`.

## Example

Consider the following Oracle table definitions as an example:

```
CREATE TABLE EXAMPLE
(
    COLUMN1 NUMBER(10, 50)
)
```

The `NUMBER(10, 50)` data type allows `10` significant decimal digits at a scale of `10^50`. Such values will not fit into the Microsoft SQL `decimal` / `numeric` data types.

## Possible remedies

Ensure that use of binary floating-point is acceptable. If full decimal precision is required by the application scenario, consider using `VARCHAR` datatype to store numbers as-is.

# DSCT01101: Sequence will not guarantee the order of values (Info)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit may produce converted sequences that will not guarantee the order of values.

## Background

Oracle database supports an additional `ORDER` option when creating a sequence, which guarantees that sequence numbers are issued in the order of requests. This is useful when sequence numbers are used as timestamps. Microsoft SQL platform does not have a similar option, and sequence numbers are not guaranteed to be generated in order of requests.

## Example

Consider the below sequence with the `ORDER` option specified.

```
CREATE SEQUENCE ORDERED_SEQ
    MINVALUE 0
    MAXVALUE 1000
    ORDER
```

The `ORDER` option will not be converted when targeting the Microsoft SQL platform, and `DSCT01101` informational message will be produced.

## Possible remedies

In general, it is very rare when the order in which sequence numbers are issued matters, but Database Schema Conversion Toolkit produces an informational message, so that you can review and ensure that the order of generated numbers is not significant in the specific use case.

If you identify that order matters, then consider creating a helper table that can be used to generate sequential numbers in the desired order.

# DSCT01102: Session-scoped sequence was converted as global sequence (Info)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit changes the scope of the sequence during conversion.

## Background

Oracle database supports session-scoped sequences, which generate unique values within a single session. Microsoft SQL does not have similar functionality and only supports global sequences.

## Example

Consider the below session-scoped sequence.

```
CREATE SEQUENCE SESSION_SEQ  
  SESSION
```

The `SESSION` option will not be converted when targeting Microsoft SQL platform and `DSCT01102` informational message will be produced.

## Possible remedies

In general, it is very rare when sequence values must be reused between different sessions, but Database Schema Conversion Toolkit produces an informational message, so that you can review each case.

If your scenario depends on the fact that values are reused between sessions, consider creating a helper table storing sequence number per `@@SPID` with associated routines to retrieve next value and clean it up. Another approach would be to update the application to use session-scoped temp tables, for example `#MySequence1`, `#MySequence2`, etc. each of which holds last generated sequence value. Such tables will be automatically dropped when session ends.

# DSCT01103: Synonym will be defined in the default schema (Warning)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article covers the reason why Database Schema Conversion Toolkit changes the schema in which synonym is defined.

## Background

Oracle database supports public synonyms which can be accessed from any schema without an explicit schema prefix. Microsoft SQL only supports synonyms defined within a schema of the database and not globally.

## Example

Consider the below public synonym:

```
CREATE PUBLIC SYNONYM PUBSYN  
FOR "EXAMPLE"."TABLE1"
```

Since given synonym is defined as "public", the Database Schema Conversion Toolkit will not produce any explicit schema reference in the converted code and synonym will be created in the default schema for the target database.

## Possible remedies

Make sure all references outside of the converted code to the synonym are updated to include schema identifier.

# Data Virtualization extension for Azure Data Studio

9/26/2022 • 7 minutes to read • [Edit Online](#)

The Data Virtualization extension for Azure Data Studio provides support for the [External Table Wizard with ODBC data sources](#).

## Install the Data Virtualization extension

To install the Data Virtualization extension, visit [Extend the functionality of Azure Data Studio](#).

## Changes in release 1.0

- Extension renamed to Data Virtualization.
- Create External Table wizard:
  - Included guided notebooks for virtualization MongoDB and Teradata sources.
  - Added dialog to fill out variables in MongoDB and Teradata virtualization notebooks.

## Changes in release 0.16

- Create External Table wizard:
  - Improved error handling when loading tables and views on object-mapping page.

## Changes in release 0.15

- Create External Table wizard:
  - Reduced time taken to load table and column information on the object-mapping page.
  - Fixed a bug with loading existing database scoped credentials on the connection details page.
- Create External Table from CSV Files Wizard:
  - Increased default sample size used for PROSE parsing.

## Changes in release 0.14.1

- Support for CTP 3.1 data source support

## Changes in release 0.12.1

- The **SQL Server big data cluster** connection type has been removed in this release. All functionality previously available from the SQL Server big data cluster connection is now available in the SQL Server connection.
- HDFS browsing can be found under the **Data Services** folder
- For notebooks, the PySpark and other big data kernels work when connected to the SQL Server master instance in your SQL Server big data cluster.
- Create External Table wizard:
  - Support for creating External Table using existing External Data Source.
  - Performance improvements across the wizard.
  - Improved handling of object names with special characters. In some cases, these caused the wizard to fail
  - Reliability improvements for the Object-Mapping page.

- Removed system databases - `DWConfiguration`, `DWDiagnostics`, `DWQueue` - from the databases dropdown.
- Support for setting the External File Format object's name in the **Create External Table from CSV Files** wizard.
- Added a refresh button to the first page of the **Create External Table from CSV Files** wizard.

## Release Notes (v0.11.0)

- Jupyter Notebook support, specifically support for the Python3 and Spark kernels, has been moved into Azure Data Studio. This extension is no longer required in order to use Notebooks.
- Multiple bug fixes in the External Data wizards:
  - Oracle type mappings have been updated to match changes shipped in SQL Server 2019 CTP 2.3.
  - Fixed an issue where new schemas typed into the table-mapping controls were being lost.
  - Fixed an issue where checking a Database node in the table-mappings didn't result in all tables and views being checked.

## Release Notes (v0.10.2)

### SQL Server 2019 support

Support for SQL Server 2019 has been updated. After connecting to a SQL Server Big Data Cluster instance, a new *Data Services* folder appears in the explorer tree. The folder has launch points for actions such as opening a new notebook against the connection, submitting Spark jobs, and working with HDFS. Some actions, such as *Create External Data* over an HDFS file/folder, the *SQL Server 2019* extension must be installed.

### Notebook support

We have made significant updates to the notebook user interface. Our focus is on making it easy to read notebooks that are shared with you. This meant removing all outline boxes around cells unless selected or hovered, adding hover support for easy cell-level actions without need to select a cell, and clarifying execution state by adding execution count, an animated *stop running* button and more. We also added keyboard shortcuts for *New Notebook* (`Ctrl+Shift+N`), *Run Cell* (`F5`), *New Code Cell* (`Ctrl+Shift+C`), *New Text Cell* (`Ctrl+Shift+T`). We aim to have all key actions launchable by shortcut so let us know what you're missing!

Other improvements and fixes include:

- The *SQL Server 2019* extension now prompts users to pick an install directory for Python dependencies. It also no longer includes Python in the `.vsix file`, reducing overall extension size. The Python dependencies support Spark and Python3 kernels.
- Support for launching a new notebook from the command line has been added. Launch with the arguments `--command=notebook.command.new --server=myservername` should open a new notebook and connect to this server.
- Performance fixes for notebooks with a large code length in cells. If code cells are over 250 lines, a scrollbar is added.
- Improved `.ipynb` file support. Version 3 or higher is now supported.

#### NOTE

Saving files updates to version 4 or higher.

- The `notebook.enabled` user setting has been removed now that the built-in Notebook viewer is stable.
- High Contrast theme is now supported with a number of fixes to object layout in this case.

- Fixed #3680 where outputs sometimes showed a number of `,,` characters incorrectly.
- Fixed #3602 Editor disappears for cells after navigating away from Azure Data Studio.
- Support has been added to use Grid views for the `application/vnd.dataresource+json` output MIME type. This means many notebooks that use this (for example by setting `pd.options.display.html.table_schema` in a Python notebook) have nicer tabular outputs.

#### Known issues

- When opening a Notebook, the install python dialog appears. Canceling this install results in the Kernels and Attach To dropdowns not showing expected values. The workaround is to complete the Python installation.
- When a notebook is opened with a kernel that isn't supported, the kernels and *attach to* dropdowns causes Azure Data Studio to stop responding. Close Azure Data Studio and ensure you use a kernel that is supported (Python3, Spark | R, Spark | Scala, PySpark, PySpark3).
- Spark UI link fails when using PySpark3 or other Spark kernels against the SQL Server endpoint. As a workaround, select Spark UI from the Dashboard, or connect using the SQL Server big data cluster connection type as this has the correct Spark UI hyperlink.

#### Extensibility improvements

A number of improvements that help extenders were added in this release.

- A new `ObjectExplorerNodeProvider` API allows extensions to contribute folders under SQL Server or other Connection nodes. This is how the `Data Services` node is added under SQL Server 2019 instances but could be used to add Monitoring or other folders easily to the UI.
- Two new context key values are available to help show/hide contributions to the dashboard.
  - `mssql:iscluster` indicates if this is a SQL Server 2019 Big Data Cluster
  - `mssql:servermajorversion` has the server version (15 for SQL Server 2019, 14 for SQL Server 2017, and so on). This can help if features should only be shown for SQL Server 2017 or greater, for example.

## Release Notes (v0.8.0)

#### Notebooks:

- Adding cells before / after existing cells are now supported by selecting the "More Actions" cell button
- **Add New Connection** option has been added to the connections in the "Attach To" dropdown
- A **Reinstall Notebook Dependencies** command has been added to assist with Python package updates, and solve cases where install was halted partway through by closing the application. This can be run from the command palette (use `Ctrl/Cmd+Shift+P` and type `Reinstall Notebook Dependencies`)
- The PROSE python package has been updated to 1.1.0 and includes a number of bug fixes. Use the **Reinstall Notebook Dependencies** command to update this package
- A **Clear Output** command is now supported by selecting the **More Actions** cell button
- Fixed the following customer reported issues:
  - Notebook session couldn't start on Windows due to PATH issues
  - Notebook couldn't be started from the root folder of a drive, such as C:\ or D:\
  - [#2820](#) Unable to edit notebooks created from ADS in VS Code
  - Spark UI link now works when running a Spark kernel
  - Renamed "Managed Packages" to "Install Packages"

#### Create External Data:

- Error messages are copyable and have been separated into a summary and detailed view for easier
- Improved UI layout and improved reliability and error handling

- Fixed the following customer reported issues:
  - Tables with invalid column mappings are shown as disabled and a warning explains the error

## Release Notes (v0.7.2)

- Azure Resource Explorer is now built into Azure Data Studio and has been removed from this extension.  
Thank you for your feedback on this!
- Improved performance of notebooks with many Markdown cells.
- Autosize code cells in Notebook. This still has a minimum size based on the cell toolbar.
- Notify user when installing Notebook dependencies. On Windows in particular this can take a long time, so notifications are now shown in the Tasks view.
- Support reinstalling Notebook dependencies. This is useful if the user previously closed Azure Data Studio partway through installation.
- Support canceling cell execution in Notebook.
- Improved reliability when using Create External Data wizard, specifically when connection errors occur.
- Block use of Create External Data wizard if PolyBase isn't enabled or running in the target server.
- Spelling and naming fixes related to SQL Server 2019 and Create External Data.
- Removed a large number of errors from the Azure Data Studio debug console.

# Extension for Oracle (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

The extension for Oracle enables you to connect to the Oracle databases and explore existing objects using the capabilities of Azure Data Studio.

The Azure Data Studio functionality available for Oracle includes:

- Connection manager and objects explorer
- [Keyboard shortcuts](#)
- [Source control integration](#)
- [Workspace & user settings](#)

## NOTE

Currently extension provides a read-only view of the database objects. Running queries is not supported.

## Install the Extension for Oracle

If you don't already have Azure Data Studio installed, see its [installation instructions](#).

1. Select **Extensions** under the **View** menu in Azure Data Studio
2. Type "*Oracle*" into the search bar on the Extensions Marketplace and select the **Extension for Oracle**.
3. Click **Install** to install the extension.

## Copyright notice

Oracle, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

## Next steps

Learn how to connect to and explore [Oracle database objects from Azure Data Studio](#).

# Kusto (KQL) extension for Azure Data Studio (Preview)

9/26/2022 • 3 minutes to read • [Edit Online](#)

The Kusto (KQL) extension for [Azure Data Studio](#) enables you to connect and query to [Azure Data Explorer](#) clusters.

Users can write and run KQL queries and author notebooks with the [Kusto kernel](#) complete with IntelliSense.

By enabling the native Kusto (KQL) experience in Azure Data Studio, data engineers, data scientists, and data analysts can quickly observe trends and anomalies against massive amounts of data stored in Azure Data Explorer.

This extension is currently in preview.

## Prerequisites

If you don't have an Azure subscription, create a [free Azure account](#) before you begin.

The following prerequisites are also required:

- [Azure Data Studio installed](#).
- [An Azure Data Explorer cluster and database](#).

## Install the Kusto (KQL) extension

To install the Kusto (KQL) extension in Azure Data Studio, follow the steps below.

1. Open the extensions manager in Azure Data Studio. You can either select the extensions icon or select **Extensions** in the View menu.
2. Type in *Kusto* in the search bar.
3. Select the **Kusto (KQL)** extension and view its details.
4. Select **Install**.



## How to connect to an Azure Data Explorer cluster

### Find your Azure Data Explorer cluster

Find your Azure Data Explorer cluster in the [Azure portal](#), then find the URI for the cluster.

The screenshot shows the Azure Data Explorer Cluster Overview page. At the top, there's a search bar and a set of actions: Add database, Stop, Refresh, Move, Delete, and Feedback. Below that, a message says 'To use Azure Data Explorer, create at least one database.' A table provides cluster details: Resource group (change) is 'mydataexplorercluster'; Location is 'West US'; Subscription (change) is listed; Subscription ID is listed; State is 'Running'; URI is 'https://mydataexplorercluster.westus.kusto.windows.net' (highlighted with a red box); Data Ingestion URI is 'https://ingest-mydataexplorercluster.westus.kusto.windows.net'; and Compute specifications are 'D13\_v2'. On the left, a sidebar lists Overview, Activity log, Access control (IAM), Tags, and Diagnose and solve problems.

However, you can get started immediately using the `help.kusto.windows.net` cluster.

For this article, we're using data from the `help.kusto.windows.net` cluster for samples.

## Connection details

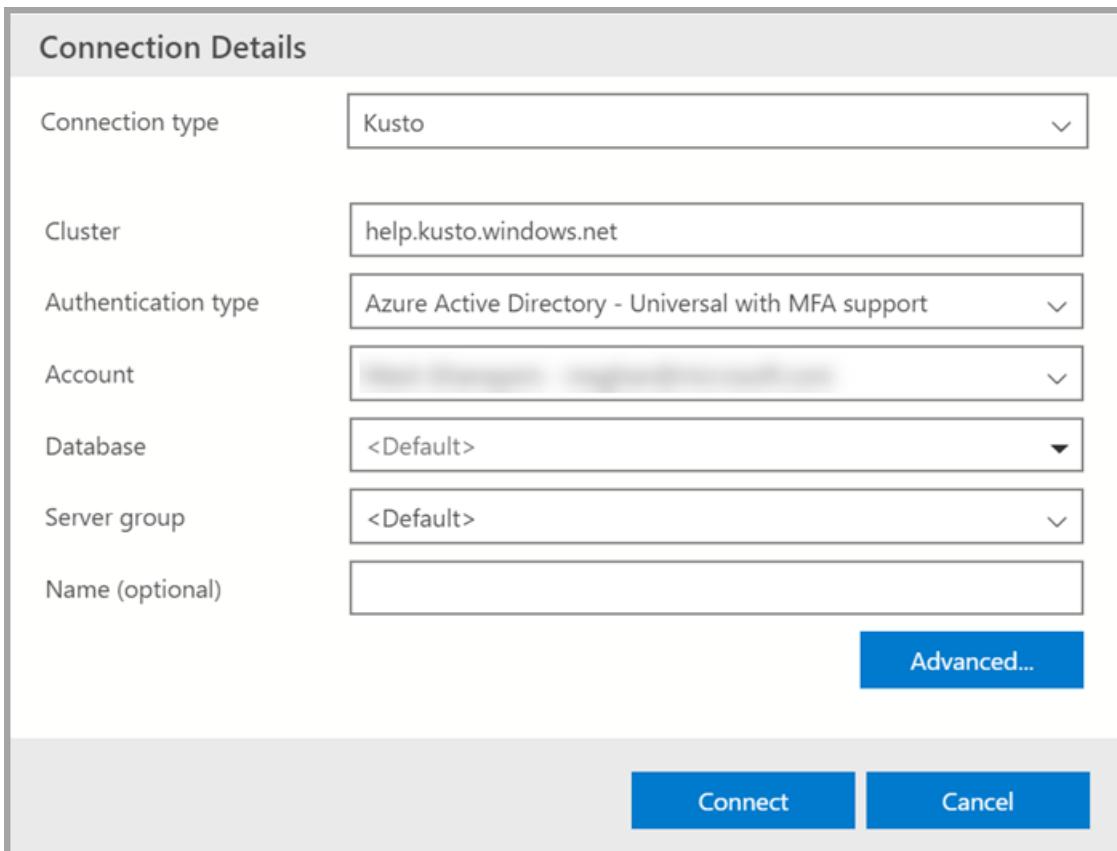
To set up an Azure Data Explorer cluster to connect to, follow the steps below.

1. Select **New connection** from the **Connections** pane.
2. Fill in the **Connection Details** information.
  - a. For **Connection type**, select *Kusto*.
  - b. For **Cluster**, enter in your Azure Data Explorer cluster.

### NOTE

When entering the cluster name, don't include the `https://` prefix or a trailing `/`.

- c. For **Authentication Type**, use the default - *Azure Active Directory - Universal with MFA account*.
- d. For **Account**, use your account information.
- e. For **Database**, use *Default*.
- f. For **Server Group**, use *Default*.
  - a. You can use this field to organize your servers in a specific group.
- g. For **Name (optional)**, leave blank.
  - a. You can use this field to give your server an alias.



## How to query an Azure Data Explorer database in Azure Data Studio

Now that you have set up a connection to your Azure Data Explorer cluster, you can query your database(s) using Kusto (KQL).

To create a new query tab, you can either select **File > New Query**, use *Ctrl + N*, or right-click the database and select **New Query**.

Once you have your new query tab open, then enter your Kusto query.

Here are some samples of KQL queries:

```
StormEvents  
| limit 1000
```

```
StormEvents  
| where EventType == "Waterspout"
```

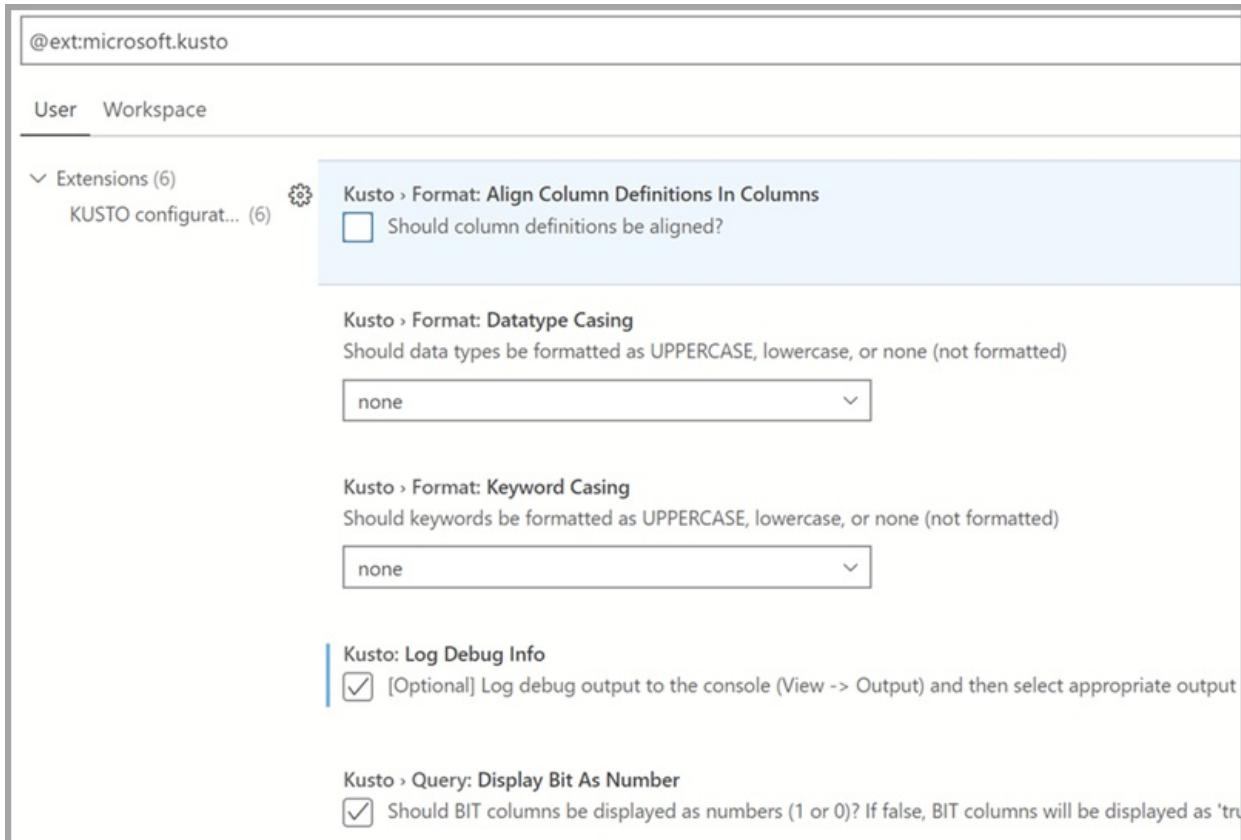
For more information about writing KQL queries, visit [Write queries for Azure Data Explorer](#)

## View extension settings

To change the settings for the Kusto extension, follow the steps below.

1. Open the extension manager in Azure Data Studio. You can either select the extensions icon or select **Extensions** in the View menu.
2. Find the **Kusto (KQL)** extension.
3. Select the **Manage** icon.
4. Select the **Extension Settings** icon.

The extensions settings look like this:



## SandDance visualization

The [SandDance extension](#) with the Kusto (KQL) extension in Azure Data Studio bring rich interactive visualization together. From the KQL query result set, select the **Visualizer** button to launch [SandDance](#).

Azure Data Studio interface showing a KQL query and its results:

```
1 StormEvents
2 | extend Damage = DamageProperty + DamageCrops, EventTime = bin(StartTime, 7d)
3 | where State in ("TEXAS", "WASHINGTON", "FLORIDA")
4 | project EventTime, State, EventType, Damage
```

The results table displays the following data:

	EventTime	State	EventType	Damage
1	2007-09-17 00:00:00	FLORIDA	Heavy Rain	0
2	2007-09-17 00:00:00	FLORIDA	Tornado	6200000
3	2007-12-10 00:00:00	FLORIDA	Thunderstorm Wind	5000
4	2007-12-10 00:00:00	FLORIDA	Thunderstorm Wind	0
5	2007-12-10 00:00:00	FLORIDA	Thunderstorm Wind	10000
6	2007-12-24 00:00:00	FLORIDA	Funnel Cloud	0
7	2007-12-10 00:00:00	FLORIDA	Thunderstorm Wind	6000
8	2007-12-17 00:00:00	FLORIDA	Tornado	15000
9	2007-12-10 00:00:00	TEXAS	Thunderstorm Wind	5000
10	2007-12-10 00:00:00	TEXAS	Tornado	12000
11	2007-12-17 00:00:00	TEXAS	Hail	0
12	2007-12-17 00:00:00	TEXAS	Hail	0

## Known issues

DETAILS	WORKAROUND
In Kusto notebook, Changing a database connection on a saved alias connection is stuck after an error in code cell execution	Close and reopen the Notebook, then connect to the right cluster with the database
In Kusto Notebook, changing a database connection on a non-saved alias connection doesn't work	Create a new connection from Connection Viewlet and save it with an alias. Then create a new notebook and connect to the newly saved connection)
In Kusto Notebook, the database dropdown isn't populated when creating a new ADX connection	Create a new connection from Connection Viewlet and save it with an alias. Then create a new notebook and connect to the newly saved connection)

You can file a [feature request](#) to provide feedback to the product team.

You can file a [bug](#) to provide feedback to the product team.

## Next steps

- [Create and run a Kusto notebook](#)
- [Kqlmagic notebook in Azure Data Studio](#)
- [SQL to Kusto cheat sheet](#)
- [What is Azure Data Explorer?](#)
- [Using SandDance visualizations](#)

# Machine Learning extension for Azure Data Studio (Preview)

9/26/2022 • 3 minutes to read • [Edit Online](#)

The Machine Learning extension for [Azure Data Studio](#) enables you to manage packages, import machine learning models, make predictions, and create notebooks to run experiments for your SQL databases. This extension is currently in preview.

## Prerequisites

The following prerequisites need to be installed on the computer you run Azure Data Studio.

- [Python 3](#). Once you have installed Python, you need to specify the local path to a Python installation under [Extension Settings](#). If you have used a [Python kernel notebook](#) in Azure Data Studio, the extension will use the path from the notebook by default.
- [Microsoft ODBC driver 17 for SQL Server](#) for Windows, macOS, or Linux.
- [R 3.5](#) (optional). Other version than 3.5 is currently not supported. Once you have installed R 3.5, you need to enable R and specify the local path to an R installation under [Extension Settings](#). This is only required if you want to manage R packages in your database.

### Trouble installing Python 3 from within ADS?

If you attempt to install Python 3 but get an error about TLS/SSL, add these two, optional components:

*sample error:*

```
$: ~/0.0.1/bin/python3 -m pip install --user "jupyter>=1.0.0" --extra-index-url https://prose-python-packages.azurewebsites.net
WARNING: pip is configured with locations that require TLS/SSL, however the ssl module in Python is not available.
Looking in indexes: https://pypi.org/simple, https://prose-python-packages.azurewebsites.net
Requirement already satisfied: jupyter
```

*install these:*

- [Homebrew](#) (optional). Install homebrew, then run `brew update` from the command line.
- [openssl](#) (optional). Next run `brew install openssl`.

## Install the extension

To install the Machine Learning extension in Azure Data Studio, follow the steps below.

1. Open the extensions manager in Azure Data Studio. You can either select the extensions icon or select Extensions in the View menu.
2. Select the **Machine Learning** extension and view its details.
3. Select **Install**.
4. Select **Reload** to enable the extension. This is only required the first time you install an extension).

# Extension settings

To change the settings for the Machine Learning extension, follow the steps below.

1. Open the extension manager in Azure Data Studio. You can either select the extensions icon or select Extensions in the View menu.
2. Find the **Machine Learning** extension under **enabled** extensions.
3. Select the **Manage** icon.
4. Select the **Extension Settings** icon.

The extensions settings look like this:

@ext:microsoft.machine-learning

4 Settings Found

User

Extensions (4)

Machine Learning... (4)

Enable Python package management in database.

Enable R package management in database.

Machine Learning: Python Path

Local path to a preexisting Python installation used by Machine Learning.

Machine Learning: R Path

Local path to a preexisting R installation used by Machine Learning.

## Enable Python

To use the Machine Learning extension as well as the Python package management in your database, follow the steps below.

### IMPORTANT

The Machine Learning extension requires Python to be enabled and configured to most functionality to work, even if you do not wish to use the Python package management in database functionality.

1. Ensure that **Machine Learning: Enable Python** is enabled. This setting is enabled by default.
2. Provide the path to your pre-existing Python installation under **Machine Learning: Python Path**. This can either be the full path to the Python executable or the folder the executable is in. If you have used a [Python kernel notebook](#) in Azure Data Studio, the extension will use the path from the notebook by default.

## Enable R

To use the Machine Learning extension for R package management in your database, follow the steps below.

1. Ensure that **Machine Learning: Enable R** is enabled. This setting is disabled by default.
2. Provide the path to your pre-existing R installation under **Machine Learning: R Path**. This has to be the full path to the R executable.

# Use the Machine Learning extension

To use the Machine Learning extension in Azure Data Studio, follow the steps below.

1. Open the **Connections** viewlet in Azure Data Studio.
2. Right-click on your server and select **Manage**.
3. Select **Machine Learning** in the left side menu under **General**.

Follow the links under **Next steps** to see how you can use the Machine Learning extension for manage packages, make predictions, and import models in your database.

## Next steps

- [Manage packages in database](#)
- [Make predictions](#)
- [Import or view models](#)
- [Notebooks in Azure Data Studio](#)
- [SQL machine learning documentation](#)
- [Machine learning and AI with ONNX in SQL Edge \(preview\)](#)

# Manage packages in database with Machine Learning extension for Azure Data Studio (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

Learn how to manage Python or R packages in your database with the [Machine Learning extension for Azure Data Studio](#).

## IMPORTANT

Manage packages in database with the Machine Learning extension currently only supports [Machine Learning Services](#) on SQL Server 2019.

## Prerequisites

- Install and configure [Machine Learning extension for Azure Data Studio](#). You need to specify the [Python or R installation paths in the Extension Settings](#) for the package management to work.
- The `sqlmlutils` package. If the package is not already installed, the Machine Learning extension will prompt you to install this.
- For SQL Server on Linux, Windows authentication is not supported. Therefore, you need to use SQL authentication to connect to your SQL Server on Linux.

## Manage Python packages

You can install and uninstall Python packages with the Machine Learning extension.

### Install new Python package

Follow the steps below to install Python packages in your database.

1. Select **Manage packages in database**.
2. If you're asked to install `sqlmlutils`, select **Yes**.
3. Under the **Installed** tab, select **Python** under the **Package Type** and select your database under **Location**.
4. Select the **Add new** tab.
5. Enter the Python package you like to install in **Search Python packages** and select **Search**.
6. Verify the package is listed under **Package Name** and the desired version is listed under **Package Version**.
7. Select **Install**.
8. Select **Close** and verify that the package was installed successfully under **Tasks**.

### Uninstall a Python package

Follow the steps below to uninstall Python packages in your database.

#### **NOTE**

You can only uninstall packages that have been installed in your database. It is not possible to uninstall a package that has been pre-installed with SQL Server Machine Learning Services.

1. Select **Manage packages in database**.
2. If you're asked to install **sqlmlutils**, select **Yes**.
3. Under the **Installed** tab, select **Python** under the **Package Type** and select your database under **Location**.
4. Select the package(s) you would like to uninstall.
5. Select **Uninstall selected packages**.
6. Select **Close** and verify that the package was installed successfully under **Tasks**.

## Manage R packages

You can install and uninstall R packages with the Machine Learning extension.

### **Install new R package**

Follow the steps below to install Python packages in your database.

1. Select **Manage packages in database**.
2. If asked to install **sqlmlutils**, select **Yes**.
3. Under the **Installed** tab, select **R** under the **Package Type** and select your database under **Location**.
4. Select the **Add new** tab.
5. Enter the R package you like to install in **Search R packages** and select **Search**.
6. Verify the package is listed under **Package Name** and the desired version is listed under **Package Version**.
7. Select **Install**.
8. Select **Close** and verify that the package was installed successfully under **Tasks**.

### **Uninstall an R package**

Follow the steps below to uninstall R packages in your database.

#### **NOTE**

You can only uninstall packages that have been installed in your database. It is not possible to uninstall a package that has been pre-installed with SQL Server Machine Learning Services.

1. Select **Manage packages in database**.
2. If asked to install **sqlmlutils**, select **Yes**.
3. Under the **Installed** tab, select **R** under the **Package Type** and select your database under **Location**.
4. Select the package(s) you would like to uninstall.
5. Select **Uninstall selected packages**.

6. Select **Close** and verify that the package was installed successfully under **Tasks**.

## Next steps

- [Machine Learning extension in Azure Data Studio](#)
- [Make predictions](#)
- [Import or view models](#)
- [Notebooks in Azure Data Studio](#)
- [SQL machine learning documentation](#)

# Make predictions with Machine Learning extension for Azure Data Studio (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

Learn how to use the [Machine Learning extension for Azure Data Studio](#) to make predictions with an ONNX model in your database. The extension will generate a T-SQL script using [PREDICT](#) to make predictions on the dataset stored in your table with a model that is previously imported, resides in a local file, or from Azure Machine Learning.

## IMPORTANT

Make predictions with the Machine Learning extension currently only supports [Machine Learning Services in Azure SQL Managed Instance](#) and [Azure SQL Edge with ONNX](#).

## Prerequisites

- Install and configure [Machine Learning extension for Azure Data Studio](#). You need to specify the [Python installation paths in the Extension Settings](#).
- The `onnxruntime`, `mlflow`, and `mlflow-dbstore` Python packages. If the packages are not already installed, the Machine Learning extension will prompt you to install them.

## Make predictions from ONNX model

Follow the steps below to use an ONNX model to make predictions.

1. Select **Make predictions**.
2. If you're asked to install `onnxruntime`, `mlflow`, and `mlflow-dbstore`, select **Yes**.
3. Choose where your model is located and select **Next**. You can use:
  - **Imported models**. Choose this to use a model that is already stored in your database. Choose the **Model database** and **Model table** where your model is located, select the model you want to use, and select **Next**.
  - **File upload**. Choose this to use a model from a file. Select the model file under **Source files** and select **Next**.
  - **Azure Machine Learning**. Choose this to use a model from Azure Machine Learning. First, **Sign in** to Azure. Then select your **Azure account**, **Azure subscription**, **Azure resource group**, and **Azure ML workspace**. Select the model you want to use and select **Next**.
4. Map the source data to your model.
  - Select the **Source database** and **Source table** containing the data set for which you want to apply the prediction.
  - Map the columns under **Model Input mapping** and **Model output**. The extension will automatically map columns that have the same name and data type.
5. Select **Predict**.

Azure Data Studio will create a new T-SQL query with the [PREDICT](#), which you can use to make predictions on your data.

## Next steps

- [Machine Learning extension in Azure Data Studio](#)
- [Manage packages in database](#)
- [Import or view models](#)
- [Notebooks in Azure Data Studio](#)
- [SQL machine learning documentation](#)
- [Machine Learning Services in Azure SQL Managed Instance](#)
- [Machine learning and AI with ONNX in SQL Edge \(Preview\)](#)

# Import or view models with Machine Learning extension for Azure Data Studio (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

Learn how to use the [Machine Learning extension for Azure Data Studio](#) to import an ONNX model or view already imported models in your database.

## IMPORTANT

Import and view in a database with the Machine Learning extension currently only supports [Machine Learning Services in Azure SQL Managed Instance](#) and [Azure SQL Edge with ONNX](#).

## Prerequisites

- Install and configure [Machine Learning extension for Azure Data Studio](#). You need to specify the [Python installation paths in the Extension Settings](#).
- The `onnxruntime`, `mlflow`, and `mlflow-dbstore` Python packages. If the packages are not already installed, the Machine Learning extension will prompt you to install them.

## View models

Follow the steps below to view ONNX models that are stored in your database.

1. Select **Import or view models**.
2. If you're asked to install `onnxruntime`, `mlflow`, and `mlflow-dbstore`, select **Yes**.
3. Select the **Models database** and **Models table** where your models are stored in.

This will show a list of your models. You can edit the model name and description, or delete a model from the list.

## Import a new model

Follow the steps below to import an ONNX model in your database.

1. Select **Import or view models**.
2. If you're asked to install `onnxruntime`, `mlflow`, and `mlflow-dbstore`, select **Yes**.
3. Select **Import models**.
4. Select the **Models database** you want to store the imported model in.
5. Select the **Models table** you want to store the imported model in. You can either choose an **Existing table** or create a **New table**. Select **Next**.
6. Select where your model is located and Select **Next**. You can use:
  - **File upload**. Choose this to use a model from a file. Select the model file under **Source files** and Select **Next**.
  - **Azure Machine Learning**. Choose this to use a model from Azure Machine Learning. First, **Sign in**

to Azure. Then select your Azure account, Azure subscription, Azure resource group, and Azure ML workspace. Select the model you want to use and Select **Next**.

7. Enter the model **Name** and **Description** and Select **Deploy** to store the model in your database.

**NOTE**

The Machine Learning extension is currently in preview. Therefore, the table schema where the models are stored might change in the future.

## Next steps

- [Machine Learning extension in Azure Data Studio](#)
- [Manage packages in database](#)
- [Make predictions](#)
- [SQL machine learning documentation](#)
- [Machine Learning Services in Azure SQL Managed Instance](#)
- [Machine learning and AI with ONNX in SQL Edge \(Preview\)](#)

# Azure SQL Managed Instance dashboard for Azure Data Studio (Preview)

9/26/2022 • 3 minutes to read • [Edit Online](#)

The Azure SQL Managed Instance extension provides a dashboard for working with an [Azure SQL Managed Instance](#) in [Azure Data Studio](#). This extension provides the following features:

- Shows SQL Managed Instance properties, including vCores and used storage
- Monitors CPU and storage usage for previous two hours
- Displays configuration warnings and tuning recommendations
- Shows state of database replicas
- Displays filtered error logs

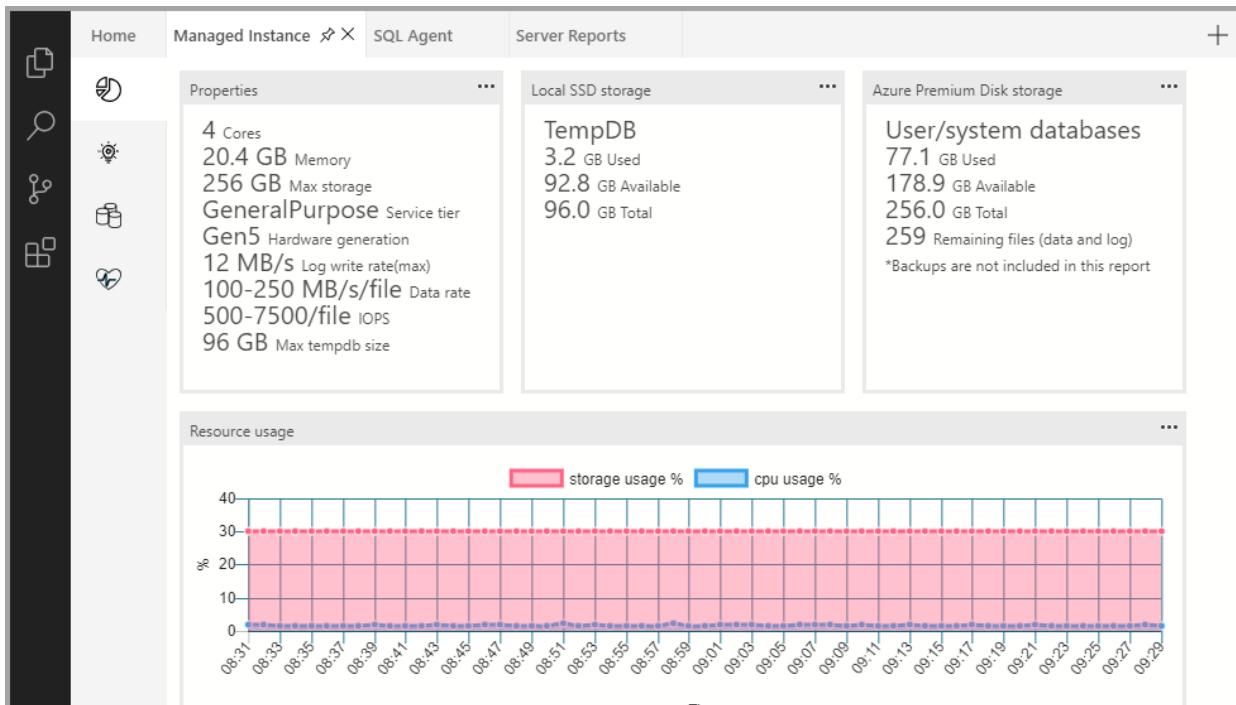
## Install

You can install the official release of this extension. Follow the steps in the [Azure Data Studio documentation](#). In the **Extensions** pane, search for "Managed Instance", and install it there. After it's installed, you'll get notified automatically about any future extension updates.

With the extension installed, you'll see a **Managed Instance** tab in Azure Data Studio. Here you can find information specific for your managed instance.

## Properties

The extension displays technical characteristics and some resource usage of your managed instance.



The top pane shows the following details:

- **Properties.** Get basic information about your managed instance, including available vCores, memory, and storage. Also find your current service tier, hardware generation, and IO characteristics such as instance log write throughput or file I/O throughput characteristics.

- **Local SSD storage.** On the general-purpose service tier, **TempDB** files are stored locally. On the business-critical service tier, *all* database files are placed on local SSD storage. In this section, you can see how much space on the local storage is used by your managed instance.
- **Azure Premium Disk Storage.** If you have the general-purpose service tier, both user and system database files are placed on Azure Premium storage. In this section, you can see the amount of data used, the number of files, and the available storage. On the business-critical service tier, this section is empty.
- **Resource usage.** View the percentage of storage and CPU that your managed instance used over the previous two hours. This way, you can increase the instance size if it's nearing the limit.

## Recommendations

When you select the second pane in the **Managed Instance** tab, you get recommendations and alerts to help optimize your managed instance.

name	reason	score	state	script	details
MEMORY_PRESSURE	PLE 48 lower than 658	100	Investigate	N/A: Add more memory or fin...	Page life expectancy 48 is lower than 658 on MSSQL\$F1B6051ABD8
HIGH_VLF_COUNT	307 VLF in testdb	95	Mitigate	USE [testdb];DBCC SHRINKFIL...	307 VLF can cause unavailability of testdb after failover. Shrink log
SLOW_MEMORY_LOAD	262ms to load pages in memory.	80	Mitigate	N/A. Investigate data IO statis...	Slow memory load in testdb indicates IO issues on data files. On GP
PAGEIOLATCH_SH	Data IO limit or memory pressure.	50	Investigate	On GP increase data file with l...	Instance may not succeed to save the memory pages to the data fil

You might see some of the following recommendations:

- **Reaching storage space limit.** Either delete unnecessary data or increase instance storage size. Databases that reach storage limit might fail to process even read queries.
- **Reaching instance throughput limit.** Notifies you when you're loading near the limit of your service tier: 22 MB/s for general-purpose or 48 MB/s for business-critical. Be aware that your managed instance will limit your load to ensure that backups can be taken.
- **Memory pressure.** Low page life expectancy or numerous `PAGEIOLATCH` wait statistics might indicate that your instance is evicting pages from the memory and constantly trying to load more pages from disk.
- **Log file limits.** If your log files approach the [file I/O limits on the general-purpose service tier](#), you might need to increase the log file size to get better performance.
- **Data file limits.** If your data files approach the [file I/O limits on the general-purpose service tier](#), you might need to increase file size to get better performance. This issue might cause memory pressure and slow down backups.
- **Availability issues.** A high number of virtual log files can affect performance. If there's a process failure, such issues might result in longer database recovery on the general-purpose service tier.

Periodically review these recommendations, investigate the root causes, and take action to correct any issues. The SQL Managed Instance extension provides scripts you can run to mitigate some of the reported issues.

## Replicas

The third pane in the **Managed Instance** tab shows you the state of database replicas in your managed instance.

The screenshot shows the 'Managed Instance' pane of the extension. On the left is a dark sidebar with icons for Home, Database, Agent, Reports, and Help. The main area has tabs for 'Home', 'Managed Instance', 'SQL Agent', and 'Server Reports'. The 'Managed Instance' tab is active, displaying a table titled 'Database replicas' with columns: Database, Is primary, Endpoint, Sync progress, Is local, State, Lag (sec), Send rate (kbps), and Redo rate (kbps). The table lists several databases: AdventureWorks2014, AdventureWorksDW2014, master, model, msdb, SQLTestDB, Test, testdb, tpch300, and WideWorldImporters, all in a synchronized state.

On the general-purpose service tier, every database has a single (primary) replica. On a business-critical tier instance, every database has one primary and three secondary replicas, one of which is used for read-only workloads. On the **Replicas** pane, you can monitor the synchronization process and verify that all secondary replicas are synchronized with the primary replica.

## Logs

The fourth pane of **Managed Instance** shows the most recent and relevant SQL error log entries.

The screenshot shows the 'Error log entries' pane of the extension. The sidebar and tabs are identical to the previous screenshot. The main area displays a table titled 'Error log entries' with columns: LogDate, ProcessInfo, and LogText. The table lists numerous log entries from October 2019, primarily from spid145s, detailing memory pressure events and logpool shrinking operations across various databases.

Although your managed instance generates a large number of log entries, most of them are internal/system information. Also, some log entries show physical database names (`GUID` values) instead of actual logical database names.

The SQL Managed Instance extension filters out unnecessary log entries based on the [Dimitri Furman method](#). The extension also displays actual logical file names instead of physical names.

## Reporting problems

If you experience any problems with the SQL Managed Instance extension, go to the [Extension GitHub project](#) and report your issue.

# Code of conduct

This project has adopted the [Microsoft Open Source Code of Conduct](#).

For more information, see the [Code of Conduct FAQ](#) or contact [opencode@microsoft.com](mailto:opencode@microsoft.com) with any additional questions or comments.

## Next steps

For more information, visit [the GitHub project](#).

# PostgreSQL extension (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

The PostgreSQL extension (preview) enables you to connect to, query, and develop for Postgres using the capabilities of Azure Data Studio.

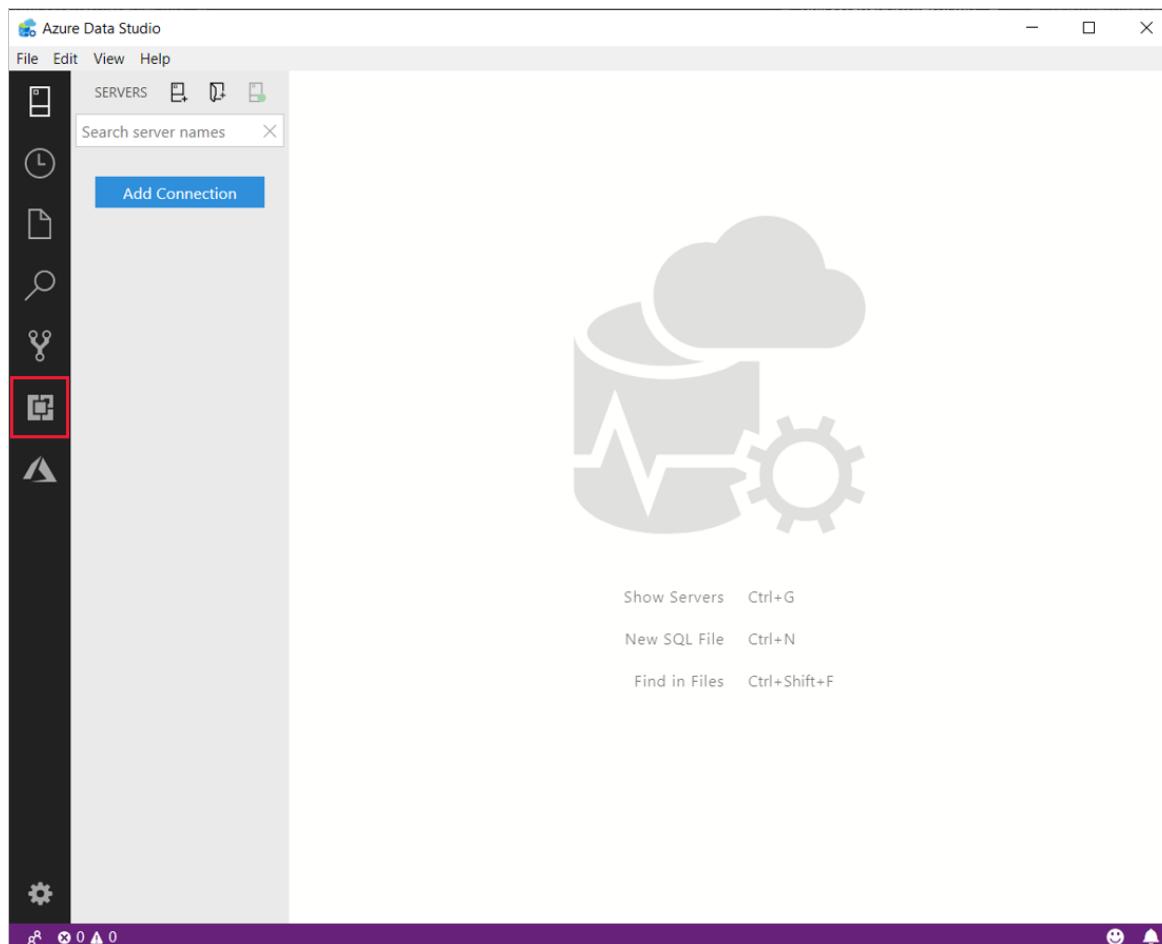
The Azure Data Studio functionality available for PostgreSQL includes:

- Connection manager & query editor
- Customizable dashboards & insight widgets
- Code snippets
- [Integrated terminal](#)
- [Keyboard shortcuts](#)
- [Source control integration](#)
- [Workspace & user settings](#)

## Install the PostgreSQL extension (preview)

If you don't already have Azure Data Studio installed, see its [install instructions](#).

1. Select the extensions icon from the sidebar in Azure Data Studio.



2. Type 'postgresql' into the search bar. Select the PostgreSQL extension.

3. Select **Install**. Once installed, select **Reload** to activate the extension in Azure Data Studio.

## Next steps

Learn how to connect and query [Postgres from Azure Data Studio](#).

# PowerShell Editor Support for Azure Data Studio

9/26/2022 • 6 minutes to read • [Edit Online](#)

This extension provides rich PowerShell editor support in [Azure Data Studio](#). Now you can write and debug PowerShell scripts using the excellent IDE-like interface that Azure Data Studio provides.

The screenshot shows the Azure Data Studio interface. On the left is a sidebar with various icons for connections, servers, databases, and other data sources. The main area has a tab titled "00\_Backups.ps1 - Azure Data Studio". The code editor contains a PowerShell script:

```
13 <# List all Databases #>
14 Get-SqlDatabase -ServerInstance localhost\sql2017
15
16                                     Get-SqlDatabase [-Name] <string> [[-Path] <string>]
17                                     [-Script] [-WhatIf] [-Confirm] [<CommonParameters>]
18 <# Backup all databases on every instance Listed in Registered Servers #>
19 DIR 'SQLSERVER:\SQLRegistration\Database Engine Server Group' |
20 Foreach {
21     Get-SqlDatabase -ServerInstance $_.Name |
22     Where { $_.Name -ne 'tempdb' } |
23     Backup-SqlDatabase -CompressionOption On
24 }
25
26
```

Below the code editor is a terminal window showing the output of the PowerShell command:

```
PS C:\temp> Get-SqlDatabase -ServerInstance localhost\sql2017
```

Name	Status	Size	Space Available	Recovery Model	Compat. Level	Owner
AdventureWorks2017	Normal	336.00 MB	56.88 MB	Simple	140	sa
AdventureWorksDW2017	Normal	208.00 MB	40.57 MB	Simple	140	sa
AirStats	Normal	23 GB	8.4 GB	Simple	130	sa
BlankDB	Normal	144.00 MB	60.22 MB	Full	140	sa
BlankDW	Normal	144.00 MB	47.95 MB	Simple	140	sa
Cleansing	Normal	80.00 MB	4.77 MB	Full	140	sa
Logging	Normal	80.00 MB	5.12 MB	Full	140	sa
master	Normal	6.00 MB	368.00 KB	Simple	140	sa
model	Normal	16.00 MB	5.53 MB	Full	140	sa
msdb	Normal	48.44 MB	784.00 KB	Simple	140	sa
PBIRS	Normal	144.00 MB	55.95 MB	Full	140	sa
PBIRSTempDB	Normal	16.00 MB	4.20 MB	Simple	140	sa
PerformanceV3	Normal	1.6 GB	98.63 MB	Full	140	sa
PopulatedDB	Normal	15 GB	6.9 GB	Full	140	sa
SandBox	Normal	80.00 MB	5.23 MB	Full	140	sa
SSISDB	Normal	779.50 MB	24.21 MB	Full	140	sa
tempdb	Normal	72.00 MB	61.25 MB	Simple	140	sa
TSQLV3	Normal	80.00 MB	1.57 MB	Full	140	sa
WideWorldImporters	Normal	3.3 GB	2.6 GB	Simple	130	sa

At the bottom of the terminal, it says "PS C:\temp> []". The status bar at the bottom right shows "Ln 14, Col 51 Spaces: 4 UTF-8 with BOM CRLF PowerShell 5.1 😊 📡".

## Features

- Syntax highlighting
- Code snippets
- IntelliSense for cmdlets and more
- Rule-based analysis provided by [PowerShell Script Analyzer](#)
- Go to Definition of cmdlets and variables
- Find References of cmdlets and variables
- Document and workspace symbol discovery
- Run selected selection of PowerShell code using F8
- Launch online help for the symbol under the cursor using Ctrl+F1
- Basic interactive console support!

## Installing the Extension

You can install the official release of the PowerShell extension by following the steps in the [Azure Data Studio documentation](#). In the Extensions pane, search for "PowerShell" extension and install it there. You will get notified automatically about any future extension updates!

You can also install a VSIX package from our [Releases page](#) and install it through the command line:

```
azuredataprofessional --install-extension PowerShell-<version>.vsix
```

## Platform support

- **Windows 7 through 10** with Windows PowerShell v3 and higher, and PowerShell Core
- **Linux** with PowerShell Core (all PowerShell-supported distributions)
- **macOS** with PowerShell Core

Read the [FAQ](#) for answers to common questions.

## Installing PowerShell Core

If you are running Azure Data Studio on macOS or Linux, you may also need to install PowerShell Core.

PowerShell Core is an Open Source project on [GitHub](#). For more information on installing PowerShell Core on macOS or Linux platforms, see the following articles:

- [Installing PowerShell Core on Linux](#)
- [Installing PowerShell Core on macOS](#)

## Example Scripts

There are some example scripts in the extension's `examples` folder that you can use to discover PowerShell editing and debugging functionality. Check out the included `README.md` file to learn more about how to use them.

This folder can be found at the following path:

```
$HOME/.azuredataprofessional/extensions/microsoft.powershell-<version>/examples
```

or if you're using the preview version of the extension

```
$HOME/.azuredataprofessional/extensions/microsoft.powershell-preview-<version>/examples
```

To open/view the extension's examples in Azure Data Studio, run the following code from your PowerShell command prompt:

```
azuredataprofessional (Get-ChildItem $Home\.azuredataprofessional\extensions\microsoft.powershell-*\examples)[-1]
```

## Creating and opening files

To create and open a new file inside the editor, use the `New-EditorFile` from within the PowerShell Integrated Terminal.

```
PS C:\temp> New-EditorFile ExportData.ps1
```

This command works for any file type, not just PowerShell files.

```
PS C:\temp> New-EditorFile ImportData.py
```

To open one or more files in Azure Data Studio, use the `Open-EditorFile` command.

```
Open-EditorFile ExportData.ps1, ImportData.py
```

### No focus on console when executing

For those users who are used to working with SSMS, you're used to being able to execute a query, and then being able to re-execute it again without having to switch back to the query pane. In this case, the default behavior of the code editor may feel strange to you. To keep the focus in the editor when you execute with F8 change the following setting:

```
"powershell.integratedConsole.focusConsoleOnExecute": false
```

The default is `true` for accessibility purposes.

Be aware this setting will prevent the focus from changing to the console, even when you use a command that explicitly calls for input, like `Get-Credential`.

## SQL PowerShell Examples

In order to use these examples (below), you need to install the `SqlServer` module from the [PowerShell Gallery](#).

```
Install-Module -Name SqlServer
```

#### NOTE

With version `21.1.18102` and up, the `SqlServer` module supports [PowerShell Core](#) 6.2 and up, in addition to Windows PowerShell.

In this example, we use the `Get-SqlInstance` cmdlet to Get the Server SMO objects for ServerA & ServerB. The default output for this command will include the Instance name, version, Service Pack, & CU Update Level of the instances.

```
Get-SqlInstance -ServerInstance ServerA, ServerB
```

Here is a sample of what that output will look like:

Instance Name	Version	ProductLevel	UpdateLevel	HostPlatform	HostDistribution
-----	-----	-----	-----	-----	-----
ServerA	13.0.5233	SP2	CU4	Windows	Windows Server 2016 Datacenter
ServerB	14.0.3045	RTM	CU12	Linux	Ubuntu

The `SqlServer` module contains a Provider called `SQLRegistration` which allows you to programmatically access the following types of saved SQL Server connections:

- Database Engine Server (Registered Servers)
- Central Management Server (CMS)

- Analysis Services
- Integration Services
- Reporting Services

In the following example, we will do a `dir` (alias for `Get-ChildItem`) to get the list of all SQL Server instances listed in your Registered Servers file.

```
dir 'SQLSERVER:\SQLRegistration\Database Engine Server Group' -Recurse
```

Here is a sample of what that output could look like:

Mode	Name
-	-----
-	ServerA
-	ServerB
-	localhost\SQL2017
-	localhost\SQL2016Happy
-	localhost\SQL2017

For many operations that involve a database, or objects within a database, the `Get-SqlDatabase` cmdlet can be used. If you supply values for both the `-ServerInstance` and `-Database` parameters, only that one database object will be retrieved. However, if you specify only the `-ServerInstance` parameter, a full list of all databases on that instance will be returned.

Here is a sample of what that output will look like:

Name	Status	Size	Space Available	Recovery Model	Compat. Level	Owner
-----	-----	-----	-----	-----	-----	-----
AdventureWorks2017	Normal	336.00 MB	57.01 MB	Simple	140	sa
master	Normal	6.00 MB	368.00 KB	Simple	140	sa
model	Normal	16.00 MB	5.53 MB	Full	140	sa
msdb	Normal	48.44 MB	1.70 MB	Simple	140	sa
PBIRS	Normal	144.00 MB	55.95 MB	Full	140	sa
PBIRSTempDB	Normal	16.00 MB	4.20 MB	Simple	140	sa
SSISDB	Normal	325.06 MB	26.21 MB	Full	140	sa
tempdb	Normal	72.00 MB	61.25 MB	Simple	140	sa
WideWorldImporters	Normal	3.2 GB	2.6 GB	Simple	130	sa

This next example uses the `Get-SqlDatabase` cmdlet to retrieve a list of all databases on the ServerB instance, then presents a grid/table (using the `Out-GridView` cmdlet) to select which databases should be backed up. Once the user clicks on the "OK" button, only the highlighted databases will be backed up.

```
Get-SqlDatabase -ServerInstance ServerB |
Out-GridView -PassThru |
Backup-SqlDatabase -CompressionOption On
```

This example, again, gets list of all SQL Server instances listed in your Registered Servers file, then calls the `Get-SqlAgentJobHistory` which reports every failed SQL Agent Job since Midnight, for each SQL Server instance listed.

```
dir 'SQLSERVER:\SQLRegistration\Database Engine Server Group' -Recurse |
WHERE { $_.Mode -ne 'd' } |
FOREACH {
    Get-SqlAgentJobHistory -ServerInstance $_.Name -Since Midnight -OutcomesType Failed
}
```

In this example, we will do a `dir` (alias for `Get-ChildItem`) to get the list of all SQL Server instances listed in your Registered Servers file, and then use the `Get-SqlDatabase` cmdlet to get a list of Databases for each of those instances.

```
dir 'SQLSERVER:\SQLRegistration\Database Engine Server Group' -Recurse |
WHERE { $_.Mode -ne 'd' } |
FOREACH {
    Get-SqlDatabase -ServerInstance $_.Name
}
```

Here is a sample of what that output will look like:

Name	Status	Size	Space	Recovery	Compat.	Owner
			Available	Model	Level	
AdventureWorks2017	Normal	336.00 MB	57.01 MB	Simple	140	sa
master	Normal	6.00 MB	368.00 KB	Simple	140	sa
model	Normal	16.00 MB	5.53 MB	Full	140	sa
msdb	Normal	48.44 MB	1.70 MB	Simple	140	sa
PBIRS	Normal	144.00 MB	55.95 MB	Full	140	sa
PBIRSTempDB	Normal	16.00 MB	4.20 MB	Simple	140	sa
SSISDB	Normal	325.06 MB	26.21 MB	Full	140	sa
tempdb	Normal	72.00 MB	61.25 MB	Simple	140	sa
WideWorldImporters	Normal	3.2 GB	2.6 GB	Simple	130	sa

## Reporting Problems

If you experience any problems with the PowerShell Extension, see [the troubleshooting docs](#) for information on diagnosing and reporting issues.

### Security Note

For any security issues, see [here](#).

## Contributing to the Code

Check out the [development documentation](#) for more details on how to contribute to this extension!

## Maintainers

- [Keith Hill - @r\\_keith\\_hill](#)
- [Tyler Leonhardt - @TylerLeonhardt](#)
- [Rob Holt](#)

## License

This extension is [licensed under the MIT License](#). For details on the third-party binaries that we include with releases of this project, see the [third-party notices](#) file.

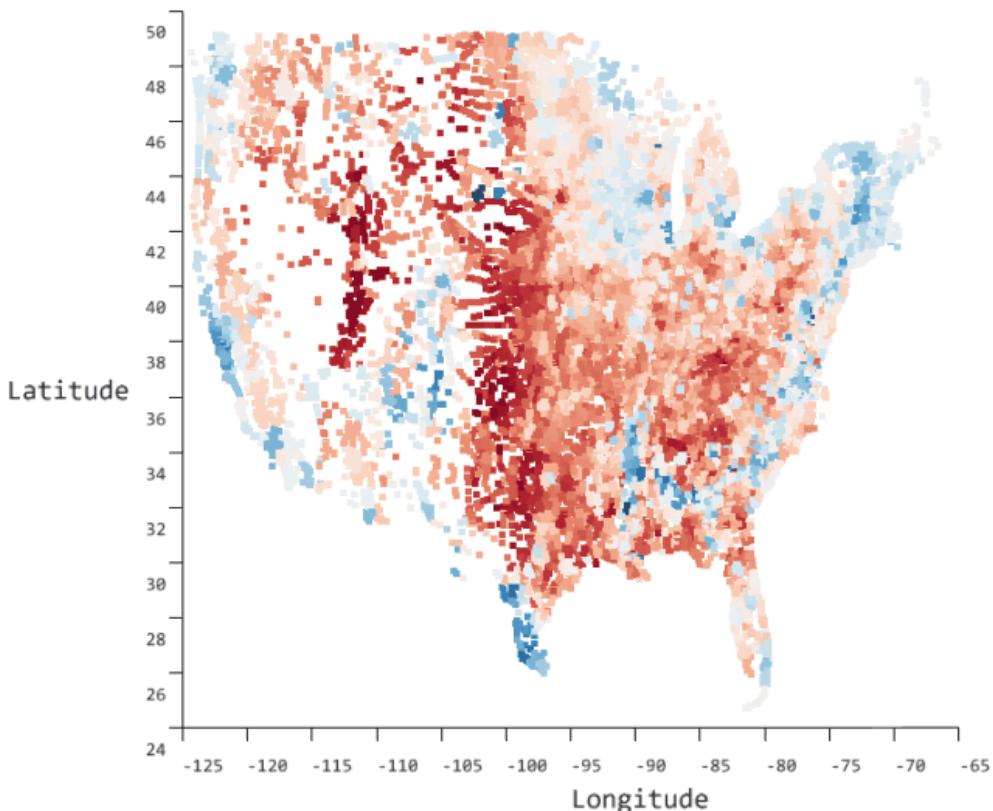
## Code of Conduct

This project has adopted the [Microsoft Open Source Code of Conduct](#). For more information, see the [Code of Conduct FAQ](#) or contact [opencode@microsoft.com](mailto:opencode@microsoft.com) with any additional questions or comments.

# SandDance for Azure Data Studio (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

Azure Data Studio now offers a way to create quick visualizations for your data. This extension is helpful when you are trying to look at the data and understand what's going on. We use a technology called SandDance from Microsoft Research, which can generate in-place visualizations of the data.



By using easy-to-understand views, SandDance helps you find insights about your data, which in turn help you tell stories supported by data, build cases based on evidence, test hypotheses, dig deeper into surface explanations, support decisions for purchases, or relate data into a wider, real world context.

SandDance uses unit visualizations, which apply a one-to-one mapping between rows in your database and marks on the screen. Smooth animated transitions between views help you to maintain context as you interact with your data.

## Usage

### **View .csv or .tsv files**

This includes local files or files on HDFS in your SQL Server 2019 Big Data Cluster.

Starting from the File menu, use Open Folder or [Ctrl+K Ctrl+O] to open the directory containing the .CSV file. Next, from within the Explorer panel, Right-click on the .csv or .tsv file and choose *View in SandDance*.

Right-click on a .csv or .tsv file in HDFS if you are connected to SQL Server 2019 Big Data Cluster and choose *View in SandDance*.

## **View SQL query results**

Starting from a SQL query window, execute a query to get a results grid. Click the Visualizer icon on the right side of the Results pane.

## Known Issues

Currently, we do not cap the row count that is visualized. However, memory consumption goes up proportionally to the number of rows, so we recommend that the data set or view is limited to around 100k rows.

See [known issues](#)

## Release Notes

### **1.0.0**

Initial release of azdata-sanddance

## Next Steps

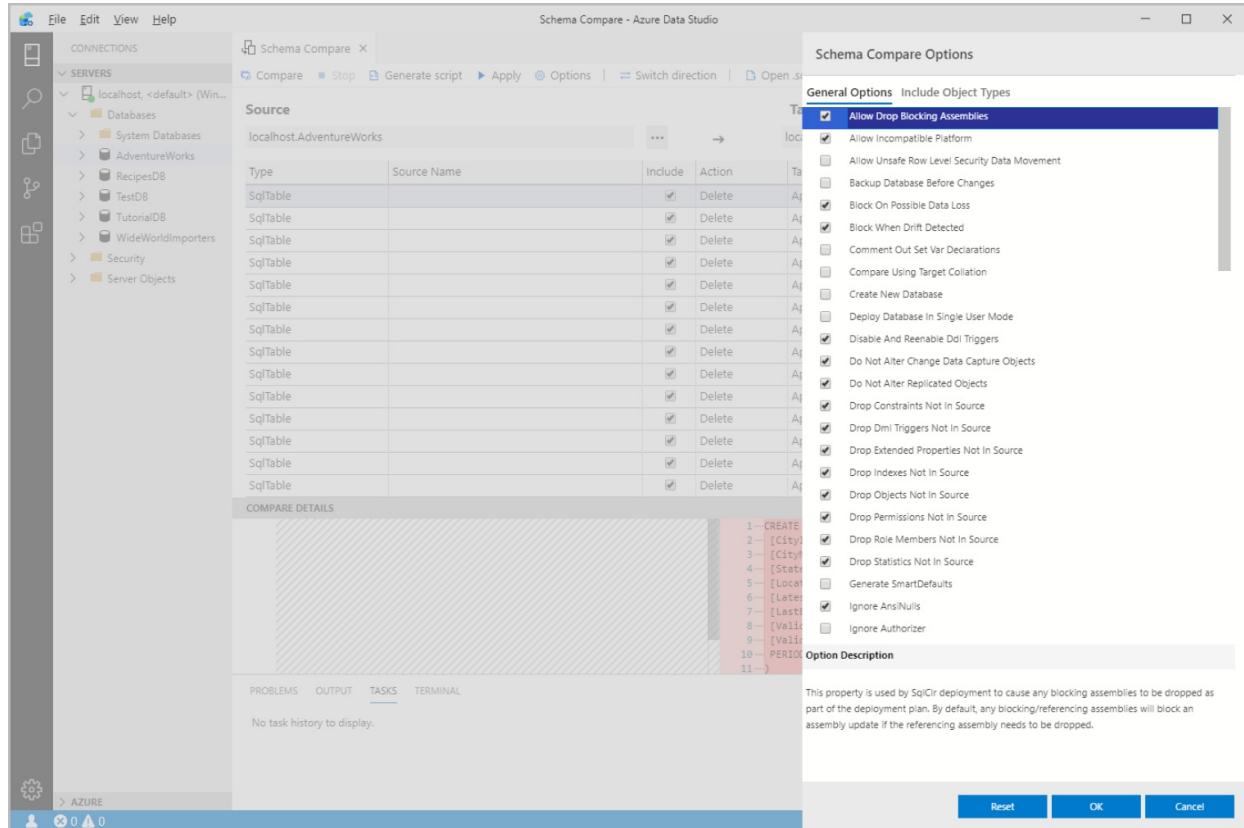
To learn more, [visit the GitHub repo](#).

# Schema Compare extension

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article provides an overview of the Schema Compare extension for Azure Data Studio. The Schema Compare extension provides an easy-to-use way to compare two database definitions and apply the differences from the source to the target. This includes active database connections, dacpac files, and [SQL projects](#).

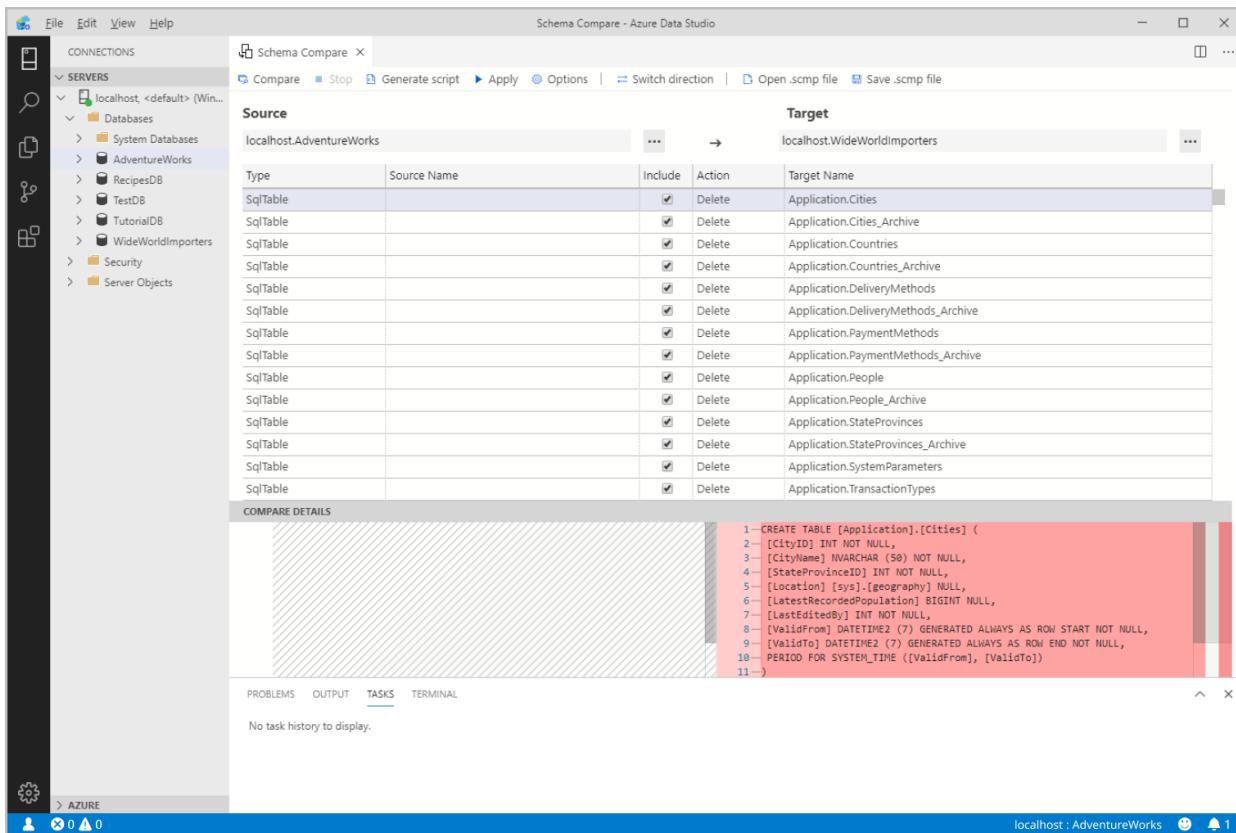
It can be tedious to manually manage and synchronize different database versions. The Schema Compare extension simplifies the process of comparing databases and gives you full control when synchronizing them - you can selectively filter specific differences and categories of differences before applying any changes. The Schema Compare extension is a reliable tool that saves you time and code.



## Features

The Schema Compare extension provides the following features:

- Compare schemas between two dacpac files, databases, or SQL projects.
- View results as a set of actions to take against the target for it to match the source.
- Selectively exclude actions listed in results.
- Set options that control the scope of the comparison.
- Apply changes directly to the target or generate a script to apply changes at a later time.
- Save the comparison.



## Install the extension

To install the Schema Compare extension, follow these steps:

1. In Azure Data Studio, select the Extensions icon to view available extensions.

Extension	Description	Provider
Database Schema Conversion Toolkit (Oracle to Microsoft SQL)	Azure Data Studio extension to convert Oracle Database objects to Microsoft	Microsoft
SQL Database Projects	The SQL Database Projects extension for Azure Data Studio allows us Microsoft	Microsoft
<b>SQL Server Schema Compare</b>	Schema Compare tool for dacpac and databases	Microsoft
Schema Visualization	Visualization of databases	Jens Hunt

2. Search for the **Schema Compare** extension and select it to view its details. Select **Install** to add the

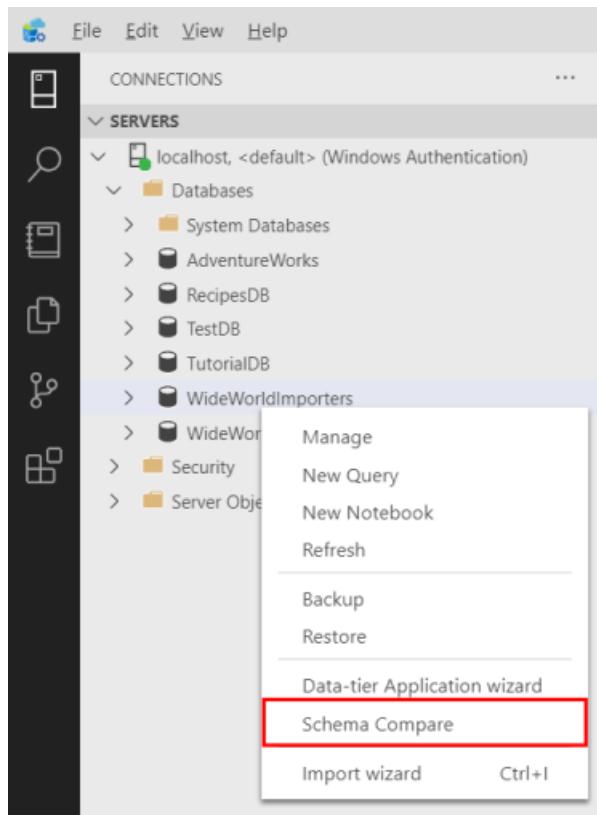
extension.

- Once installed, **Reload** to enable the extension in Azure Data Studio (only required when installing an extension for the first time).

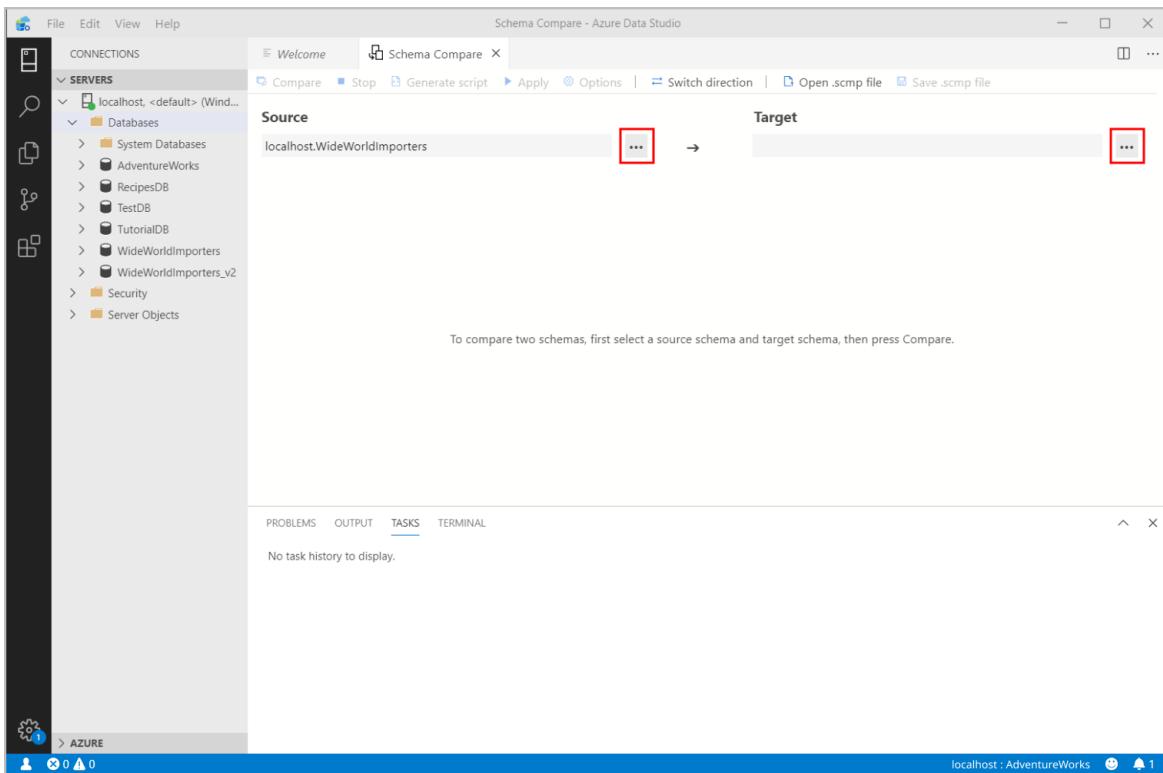
## Compare schemas

To compare schemas, open the Schema Compare dialog box. To do so, follow these steps:

- To open the Schema Compare dialog box, right-click a database in Object Explorer and select **Schema Compare**. The database you select is set as the **Source** database in the comparison.



- Select one of the ellipses (...) to change the **Source** and **Target** of your Schema Compare and select OK.



3. To customize your comparison, select the **Options** button in the toolbar.
4. Select **Compare** to view the results of the comparison.

## Update an existing SQL project from a database

To update an existing SQL project from a database, follow these steps:

1. Install both the **Schema Compare** and **SQL Database Project** extensions.
2. From a **Database dashboard** select the **Update Project from Database** option in the toolbar.
3. Select the existing SQL project and the desired file structure for new objects.
4. Choose **View changes in Schema Compare** to review the changes before applying them to the SQL project.

## Next steps

- [Learn more about Schema Compare in SSDT](#)
- [Explore SQL database projects in Azure Data Studio](#)

# SQL Database Projects extension (Preview)

9/26/2022 • 3 minutes to read • [Edit Online](#)

The SQL Database Projects extension (preview) is an Azure Data Studio and VS Code extension for developing SQL databases including for SQL Server, Azure SQL Database, and Azure SQL Managed Instance in a project-based development environment. A SQL project is a local representation of SQL objects that comprise the schema for a single database, such as tables, stored procedures, or functions.

## Extension features

The SQL Database Projects extension provides the following features:

- Create a new blank project.
- Create a new project from a connected database or from an [OpenAPI](#) specification file.
- Open a project previously created in [Azure Data Studio](#), [VS Code](#) or in [SQL Server Data Tools](#).
- Edit a project by adding or removing objects (tables, views, stored procedures) or custom scripts in the project.
- Organize files/scripts in folders.
- Add references to system databases or a user dacpac.
- Build a single project.
- Deploy a single project.
- Load connection details (SQL Windows authentication) and SQLCMD variables from deployment profile.

Watch this short 10-minute video for an introduction to the SQL Database Projects extension in Azure Data Studio:

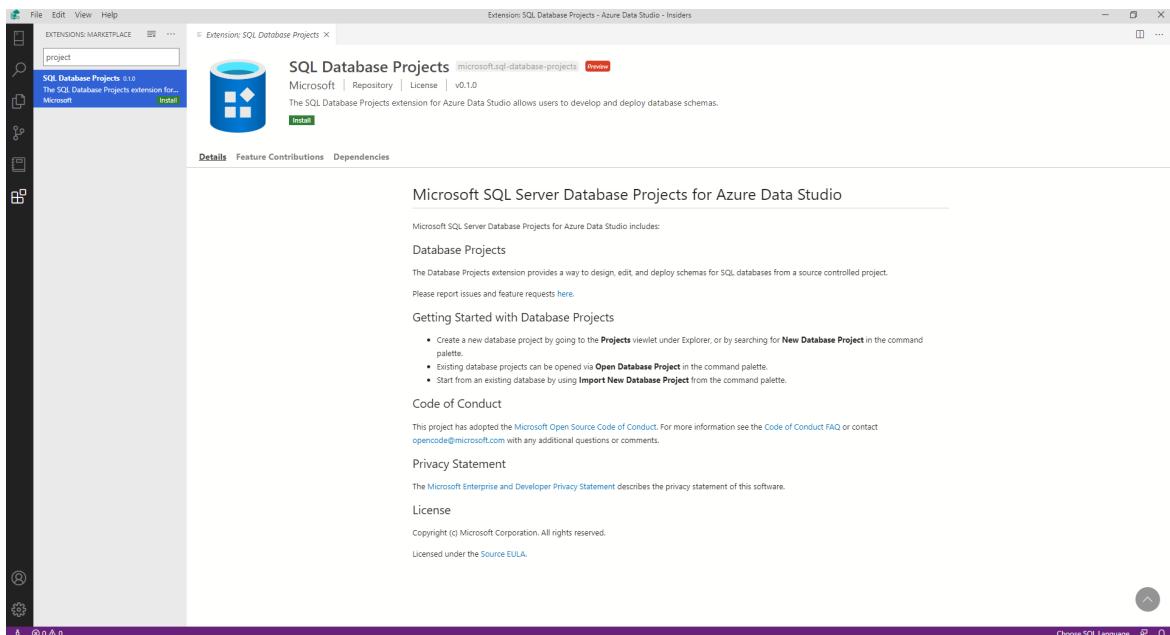
## Install

You can install the SQL Database Project extension in Azure Data Studio and VS Code.

### Azure Data Studio

To install the SQL Database Project extension in Azure Data Studio, follow these steps:

1. Open the extensions manager to access the available extensions. To do so, either select the extensions icon or select **Extensions** in the **View** menu.
2. Identify the *SQL Database Projects* extension by typing all or part of the name in the extension search box. Select an available extension to view its details.



3. Select the extension you want and choose to **Install** it.
4. Select **Reload** to enable the extension (only required the first time you install an extension).
5. Select the **Projects** icon from the activity bar.

#### NOTE

- It is recommended to install the [Schema Compare extension](#) alongside the SQL Database Projects extension for full functionality.

## VS Code

The SQL Database Projects extension is installed with the [mssql](#) extension for VS Code.

## Dependencies

The SQL Database Projects extension has a dependency on the .NET Core SDK and AutoRest.Sql

### .NET Core SDK

The .NET Core SDK is required for project build functionality and you will be prompted to install the .NET Core SDK if it cannot be detected by the extension. The .NET Core SDK (v3.1.x) can be downloaded and installed from <https://dotnet.microsoft.com/download/dotnet-core/3.1>.

If you would like to [check currently installed versions](#) of the dotnet SDK, open a terminal and run the following command:

```
dotnet --list-sdks
```

To force the SQL Database Projects extension to use the v3.1.x version of the .NET Core SDK when multiple versions are installed, add a [package.json](#) file to the folder that contains the SQL project.

Unsupported .NET Core SDK versions may result in error messages such as:

- `error MSB4018: The "SqlBuildTask" task failed unexpectedly.`  
`error MSB4018: System.TypeInitializationException: The type initializer for 'SqlSchemaModelStaticState' threw an exception. ---> System.IO.FileNotFoundException: Could not load file or assembly 'System.Runtime, Version=4.2.2.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a'. The system cannot find the file specified. [c:\Users\ .sqlproj]`

(where the linked non-existing file has an unmatched closing square bracket).

## AutoRest.Sql

The SQL extension for [AutoRest](#) is automatically downloaded and used by the SQL Database Projects extension when a SQL project is generated from an OpenAPI specification file.

## Limitations

Currently, the SQL Database Project extension has the following limitations:

- Loading files as a link is not supported in Azure Data Studio today, however the files will be loaded at the top level in the tree and the build will incorporate these files as expected.
- SQLCLR objects in projects are not supported in the .NET Core version of DacFx.
- Tasks (build/publish) are not user-defined.
- Publish targets defined by DacFx.
- WSL environment support is limited.

## Workspace

SQL database projects are contained within a logical workspace in Azure Data Studio and VS Code. A workspace manages the folder(s) visible in the Explorer pane. **All SQL projects within the folders open in the current workspace are available in the SQL Database Projects view by default.**

You can manually add and remove projects from a workspace through the interface in the **Projects** pane. The settings for a workspace can be manually edited in the `.code-workspace` file, if necessary.

In the following example `.code-workspace` file, the `folders` array lists all folders included in the Explorer pane and the `dataworkspace.excludedProjects` array within `settings` lists all the SQL projects included in the **Projects** pane.

```
{
  "folders": [
    {
      "path": "."
    },
    {
      "name": "WideWorldImportersDW",
      "path": "..\\WideWorldImportersDW"
    }
  ],
  "settings": {
    "dataworkspace.excludedProjects": [
      "AdventureWorksLT.sqlproj"
    ]
  }
}
```

## Next steps

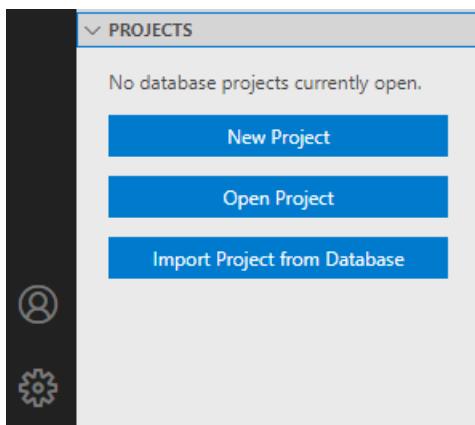
- [Getting Started with the SQL Database Projects extension](#)
- [Build and Publish a project with SQL Database Projects extension](#)

# Getting started with the SQL Database Projects extension (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article describes three ways to get started with the SQL Database Projects extension:

1. Create a new database project by going to the **Projects** view or by searching for **New Database Project** in the command palette.
2. Existing database projects can be opened via **Open Database Project** in the command palette.
3. Start from an existing database by using **Import New Database Project** from the command palette.



## Create an empty database project

In the **Projects** view select the **New Project** button and enter a project name in the text input that appears. In the "Select a Folder" dialog that appears, select a directory for the project's folder, .sqlproj file, and other contents to reside in. The empty project is opened and visible in the **Projects** view for editing.

## Open an existing project

In the **Projects** view, select the **Open Project** button and open an existing .sqlproj file from the file picker that appears. Existing projects can originate from Azure Data Studio, VS Code or [Visual Studio SQL Server Data Tools](#).

The existing project is opened and its contents are visible in the **Projects** view for editing.

## Create a database project from an existing database

In the **Project** view select the **Import Project from Database** button and connect to a SQL Server. Once the connection is established, select a database from the list available databases and set the name of the project.

Finally, select a target structure of the extraction. The new project is opened and contains SQL scripts for the contents of the selected database.

## Build and publish

Deploying the database project is achieved in the SQL Database Projects extension by building the project into a [data-tier application file](#) (DACPAC) and publishing to a supported platform. For more on this process, see [Build and Publish a Project](#).

## Schema compare

The SQL Database Projects extension interacts with the [Schema Compare extension](#), if installed, to compare the contents of a project to a dacpac or existing database. The resulting schema comparison can be used to view and apply the differences from source to target.

## Next steps

- [Build and Publish a project with SQL Database Projects extension](#)

# Build and Publish a project

9/26/2022 • 2 minutes to read • [Edit Online](#)

The build process in the SQL Database Projects extension (preview) for Azure Data Studio and VS Code allows for *dacpac* creation in Windows, macOS, and Linux environments. The project can be deployed to a local or cloud environment with the publish process.

## Prerequisites

- Install and configure [SQL Database Projects extension for Azure Data Studio or VS Code](#).

## Build a Database Project

In the **Projects** viewlet right-click the *.sqlproj* root node and select **Build**.

The output pane will automatically appear with the output from the build process. A successful build will conclude with the message:

```
... exited with code: 0
```

## Publish a Database Project

### **Publish to an existing SQL instance**

*This functionality is available in Azure Data Studio and VS Code, however the interfaces will appear slightly different between the 2 applications.*

After a project is successfully compiled through the [build process](#), the database can be published to a SQL Server instance. To publish a database project, in the **Projects** viewlet right-click the *.sqlproj* root node and select **Publish**.

In the **Publish Database** dialog that appears, specify a server connection and the database name to be created.

### **Publish the SQL project and deploy to a local Container**

*This functionality is currently only available in VS Code.*

After a project is successfully compiled through the [build process](#), the database can be published to a new development instance of SQL Server in a local container. To publish a database project to a local container, in the **Projects** viewlet right-click the *.sqlproj* root node and select **Publish**.

In VS Code, the options for publish appear as:

- Publish to existing server
- Publish to new server in a container

To create a new container with a development SQL instance and publish the SQL project contents to it, select the option "Publish to new server in a container".

Creating a new container exposes the following options:

- SQL Server port number: the port on which the SQL server's 1433 port will be forwarded to your workstation
- SQL Server admin password: the *sa* password for the new instance
- SQL Server docker image: the version base of the container

## Next steps

- [SQL Database Projects extension](#)
- [Build SQL database projects from command line](#)

# Build a database project from command line

9/26/2022 • 2 minutes to read • [Edit Online](#)

While the SQL Database Project extension (preview) provides a graphical user interface to [build a database project](#), a command line build experience is also available for Windows, macOS, and Linux environments. The steps to build a project from the command line are different between [SDK-style SQL projects](#) and the previous non-SDK-style SQL project format. This article outlines the prerequisites and syntax needed to build a SQL project to dacpac from the command line for both SQL project types.

## SDK-style SQL projects

Using [Microsoft.Build.Sql](#) with [SDK-style SQL projects](#) is the preferred method for working with SQL projects from the command line.

To build an SDK-style SQL project from the command line on Windows, macOS, or Linux, use the following command:

```
dotnet build /p:NetCoreBuild=true
```

### NOTE

The SDK-style SQL projects is currently in preview.

## Non-SDK-style SQL projects

### Prerequisites

1. Install and configure [SQL Database Projects extension](#).
2. The following .NET Core dlls and the target file `Microsoft.Data.Tools.Schema.SqlTasks.targets` are required to build a SQL database project from the command line from all platforms supported by the Azure Data Studio extension for SQL Database Projects. These files are created by the extension during the first build completed in the Azure Data Studio interface and placed in the extension's folder under `BuildDirectory`. For example, on Linux, these files are placed in `~\azuredatadistro\extensions\microsoft.sql-database-projects-x.x.x\BuildDirectory\`. Copy these 11 files to a new and accessible folder or note their location. This location will be referred to as `DotNet Core build folder` in this document.

- Microsoft.Data.SqlClient.dll
- Microsoft.Data.Tools.Schema.Sql.dll
- Microsoft.Data.Tools.Schema.SqlTasks.targets
- Microsoft.Data.Tools.Schema.Tasks.Sql.dll
- Microsoft.Data.Tools.Utilities.dll
- Microsoft.SqlServer.Dac.dll
- Microsoft.SqlServer.Dac.Extensions.dll
- Microsoft.SqlServer.TransactSql.ScriptDom.dll
- Microsoft.SqlServer.Types.dll
- System.ComponentModel.Composition.dll

- System.IO.Packaging.dll
3. If the project was created in Azure Data Studio - skip ahead to [Build the project from the command line](#). If the project was created in SQL Server Data Tools (SSDT), open the project in the Azure Data Studio SQL Database project extension. Opening the project in Azure Data Studio automatically updates the `sqlproj` file with three edits, noted below for your information:

a. Import conditions

```
<Import Condition="\"$(NetCoreBuild)" == 'true'" Project="$(NETCoreTargetsPath)\Microsoft.Data.Tools.Schema.SqlTasks.targets"/>
<Import Condition="\"$(NetCoreBuild)" != 'true' AND "$(SQLDBExtensionsRefPath)" != '' Project="$(SQLDBExtensionsRefPath)\Microsoft.Data.Tools.Schema.SqlTasks.targets"/>
<Import Condition="\"$(NetCoreBuild)" != 'true' AND "$(SQLDBExtensionsRefPath)" == '' Project="$(MSBuildExtensionsPath)\Microsoft\VisualStudio\v$(VisualStudioVersion)\SSDT\Microsoft.Data.Tools.Schema.SqlTasks.targets"/>
```

b. Package reference

```
<ItemGroup>
    <PackageReference Condition="\"$(NetCoreBuild)" == 'true'" Include="Microsoft.NETFramework.ReferenceAssemblies" Version="1.0.0" PrivateAssets="All"/>
</ItemGroup>
```

c. Clean target, necessary for supporting dual editing in SQL Server Data Tools (SSDT) and Azure Data Studio

```
<Target Name="AfterClean">
    <Delete Files="$(BaseIntermediateOutputPath)\project.assets.json"/>
</Target>
```

### Build the project from the command line

From the full .NET folder, use the following command:

```
dotnet build "<sqlproj file path>" /p:NetCoreBuild=true /p:NETCoreTargetsPath=<DotNet Core build folder>"
```

For example, from `/usr/share/dotnet` on Linux:

```
dotnet build "/home/myuser/Documents/DatabaseProject1/DatabaseProject1.sqlproj" /p:NetCoreBuild=true
/p:NETCoreTargetsPath="/home/myuser/.azuredatastudio/extensions/microsoft.sql-database-projects-
x.x.x/BuildDirectory"
```

## Next steps

- [SQL Database Projects extension](#)
- [SDK-style SQL projects in SQL Database Projects extension](#)
- [Publish SQL database projects](#)

# Use SDK-style SQL projects with the SQL Database Projects extension (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article introduces [Microsoft.Build.Sql](#) for SDK-style SQL projects in the SQL Database Projects extension in Azure Data Studio or VS Code. SDK-style SQL projects are especially advantageous for applications shipped through pipelines or built in cross-platform environments. The initial announcement is available in [TechCommunity](#).

## NOTE

The SDK-style SQL projects is currently in preview.

## Create an SDK-style database project

You can create an SDK-style database project from a blank project, or from an existing database.

### Blank project

In the **Projects** view, select the **New Project** button and enter a project name in the text input that appears. In the **Select a Folder** dialog box that appears, choose a directory for the project's folder, `.sqlproj` file, and other contents to reside in.

By default the selection for **SDK-style project (Preview)** is checked. When the dialog is completed, the empty project is opened and visible in the **Projects** view for editing.

### From an existing database

In the **Project** view, select the **Import Project from Database** button and connect to a SQL Server. Once the connection is established, select a database from the list of available databases and set the name of the project. Select a target structure of the extraction.

By default the selection for **SDK-style project (Preview)** is checked. When the dialog is completed, the new project is opened and contains SQL scripts for the contents of the selected database.

## Build and publish

From the Azure Data Studio and VS Code interfaces, building and publishing an SDK-style SQL project is completed in the same way as the previous SQL project format. For more on this process, see [Build and Publish a Project](#).

To build an SDK-style SQL project from the command line on Windows, macOS, or Linux, use the following command:

```
dotnet build /p:NetCoreBuild=true
```

The `.dacpac` resulting from building an SDK-style SQL project is compatible with tooling associated with the data-tier application framework (`.dacpac`, `.bacpac`), including [SqlPackage](#).

## Next steps

- Build and Publish a project with the SQL Database Projects extension

# SQL Server Agent extension (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

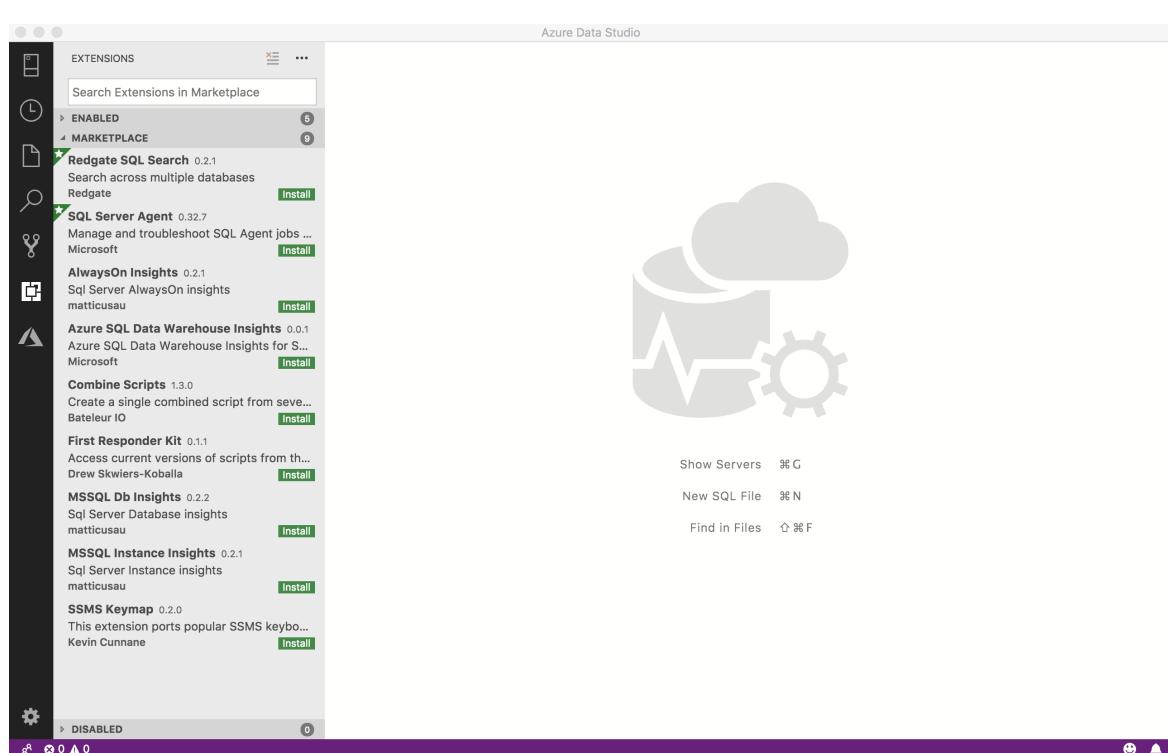
The SQL Server Agent extension (preview) is an extension for managing and troubleshooting SQL Agent jobs and configuration. This extension is currently in preview.

Key actions include:

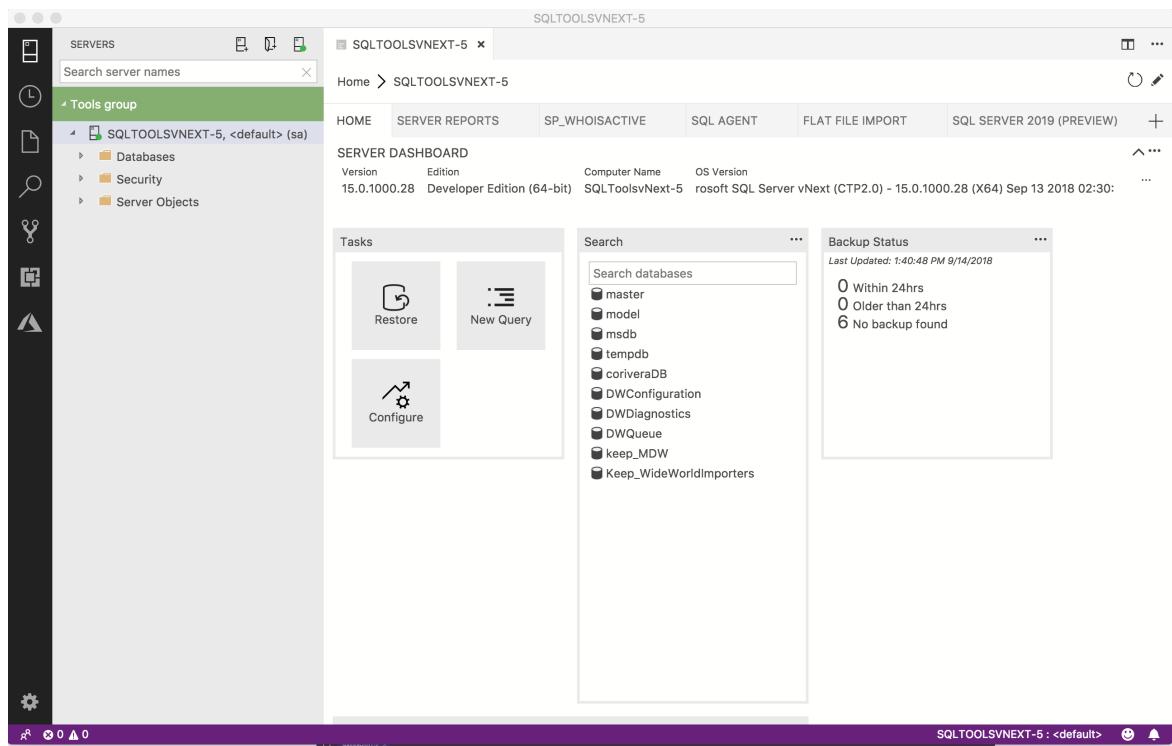
- List SQL Server Agent Jobs Configured on a SQL Server
- View Job History with job execution results
- Basic Job Control to start and stop jobs

## Install the SQL Server Agent extension

1. To open the extensions manager and access the available extensions, select the extensions icon, or select **Extensions** in the **View** menu.
2. Select an available extension to view its details.

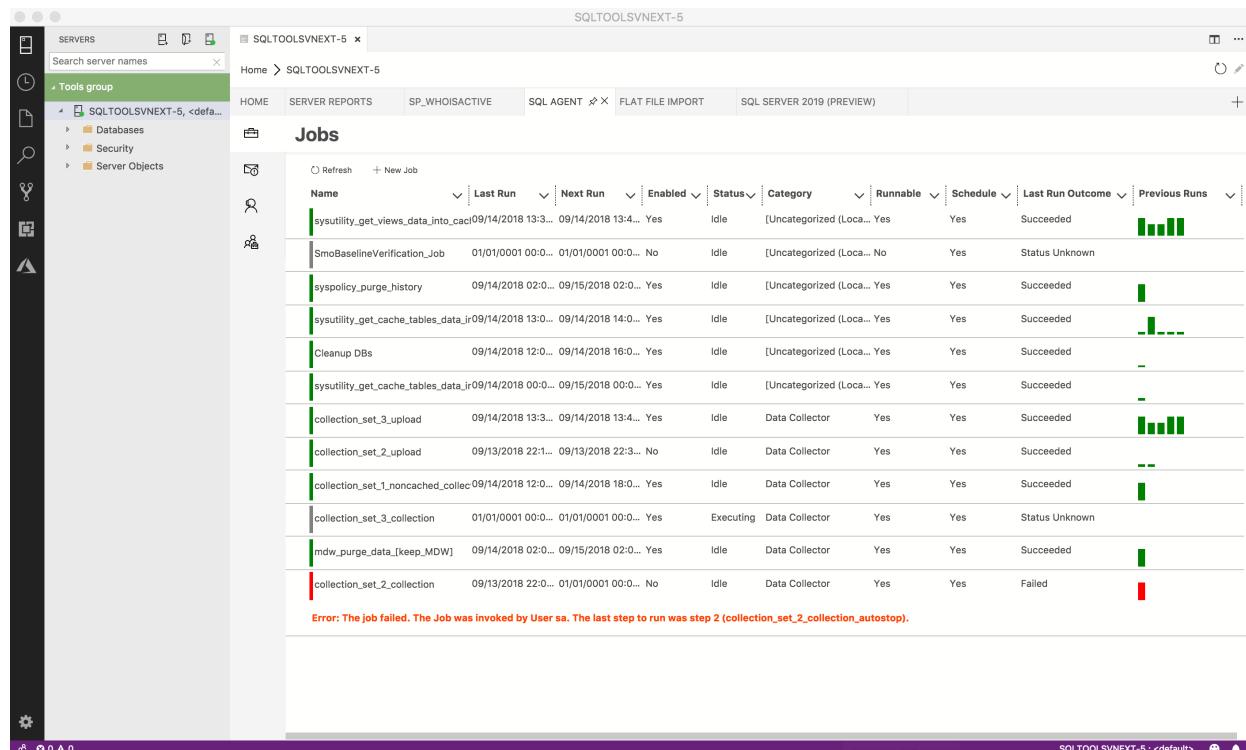


3. Select the extension you want and **Install** it.
4. Select **Reload** to enable the extension (only required the first time you install an extension).
5. Navigate to your management dashboard by right-clicking your server or database and selecting **Manage**.
6. Installed extensions appear as tabs on your management dashboard:



## View jobs

When you connect to the SQL Server Agent extension, the first thing you see is a list of all your Agent jobs.



Name	Last Run	Next Run	Enabled	Status	Category	Runnable	Schedule	Last Run Outcome	Previous Runs
sysutility_get_views_data_into_caci	09/14/2018 13:3...	09/14/2018 13:4...	Yes	Idle	[Uncategorized (Loca...]	Yes	Yes	Succeeded	██████
SmoBaselineVerification_Job	01/01/2001 00:0...	01/01/2001 00:0...	No	Idle	[Uncategorized (Loca...]	No	Yes	Status Unknown	
syspolicy_purge_history	09/14/2018 02:0...	09/15/2018 02:0...	Yes	Idle	[Uncategorized (Loca...]	Yes	Yes	Succeeded	█
sysutility_get_cache_tables_data_ir	09/14/2018 13:0...	09/14/2018 14:0...	Yes	Idle	[Uncategorized (Loca...]	Yes	Yes	Succeeded	█
Cleanup DBs	09/14/2018 12:0...	09/14/2018 16:0...	Yes	Idle	[Uncategorized (Loca...]	Yes	Yes	Succeeded	-
sysutility_get_cache_tables_data_ir	09/14/2018 00:0...	09/15/2018 00:0...	Yes	Idle	[Uncategorized (Loca...]	Yes	Yes	Succeeded	-
collection_set_3_upload	09/14/2018 13:3...	09/14/2018 13:4...	Yes	Idle	Data Collector	Yes	Yes	Succeeded	██████
collection_set_2_upload	09/13/2018 22:1...	09/13/2018 22:3...	No	Idle	Data Collector	Yes	Yes	Succeeded	-
collection_set_1_noncached_collec	09/14/2018 12:0...	09/14/2018 18:0...	Yes	Idle	Data Collector	Yes	Yes	Succeeded	█
collection_set_3_collection	01/01/2001 00:0...	01/01/2001 00:0...	Yes	Executing	Data Collector	Yes	Yes	Status Unknown	
mdw_purge_data_[keep_MDW]	09/14/2018 02:0...	09/15/2018 02:0...	Yes	Idle	Data Collector	Yes	Yes	Succeeded	█
collection_set_2_collection	09/13/2018 22:0...	01/01/2001 00:0...	No	Idle	Data Collector	Yes	Yes	Failed	█

## Next steps

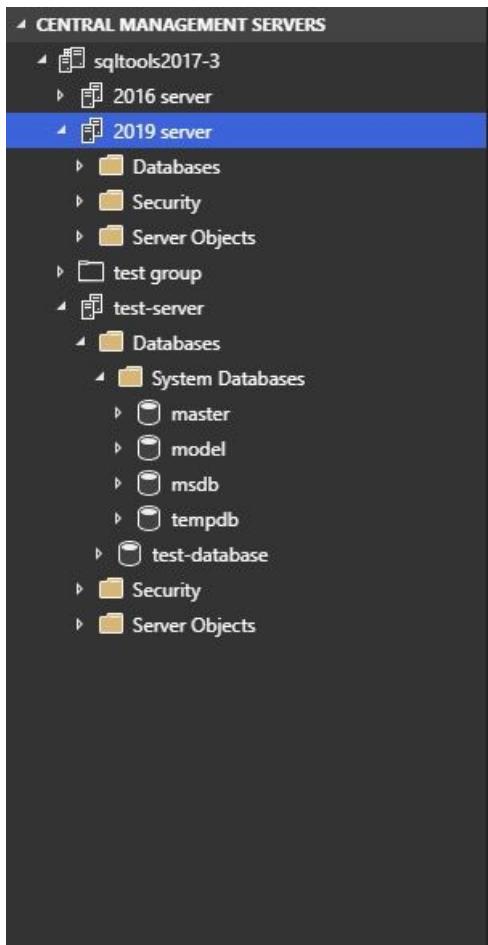
To learn more about SQL Server Agent, [check our documentation](#).

# SQL Server Central Management Servers extension (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

The Central Management Servers extension allows users to store a list of instances of SQL Server that is organized into one or more groups. Actions that are taken using a CMS group act on all servers in the server group.

This experience is currently in its initial preview. Report issues and feature requests [here](#).



## Install the SQL Server Central Management Servers extension

1. To open the extensions manager and access the available extensions, select the extensions icon, or select **Extensions** in the **View** menu.
2. Select an available extension to view its details.
3. Select the extension you want (SQL Server Central Management Servers) and **Install** it.

### How do I start Central Management Servers?

Central Management Servers can be viewed by clicking on the Connections icon (Ctrl/Cmd + G). The first time you download the extension, the CMS view will be minimized, and you can open it by click on **Central Management Servers**

## Next steps

To learn more conceptually about Central Management Servers, [you can read more here](#).

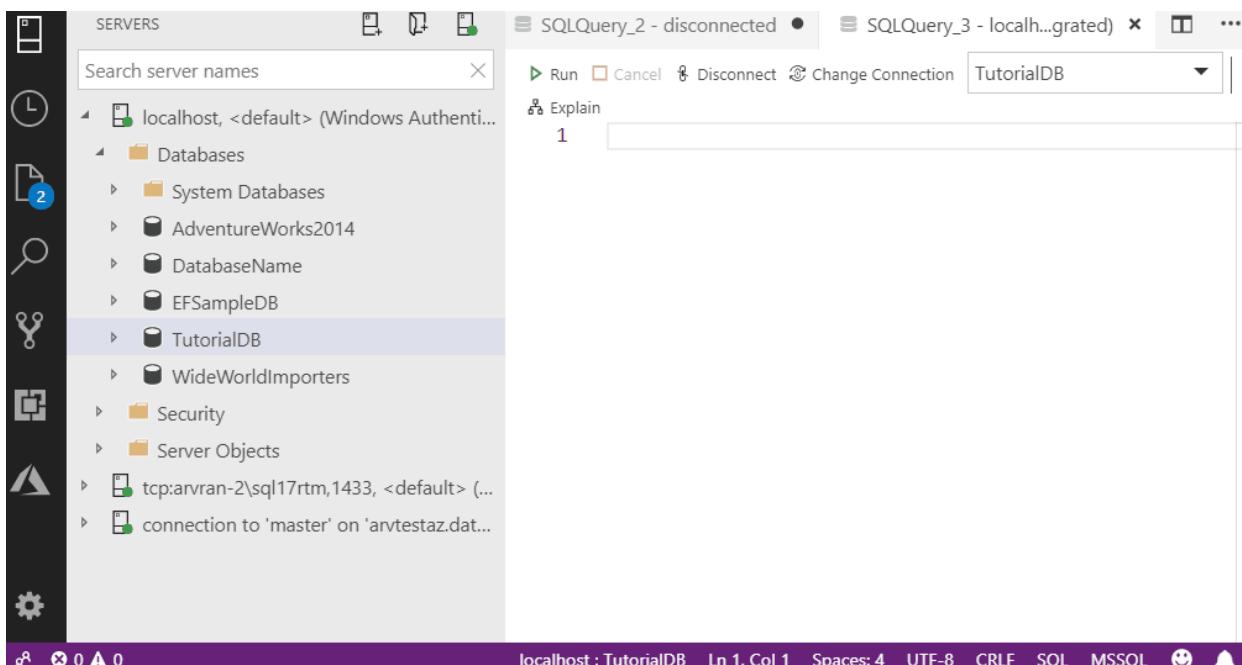
# SQL Server dacpac extension

9/26/2022 • 2 minutes to read • [Edit Online](#)

The **Data-tier Application Wizard** provides an easy-to-use wizard experience to deploy and extract dacpac files and import and export bacpac files.

## Features

- Deploy a dacpac to a SQL Server instance
- Extract a SQL Server instance to a dacpac
- Create a database from a bacpac
- Export schema and data to a bacpac

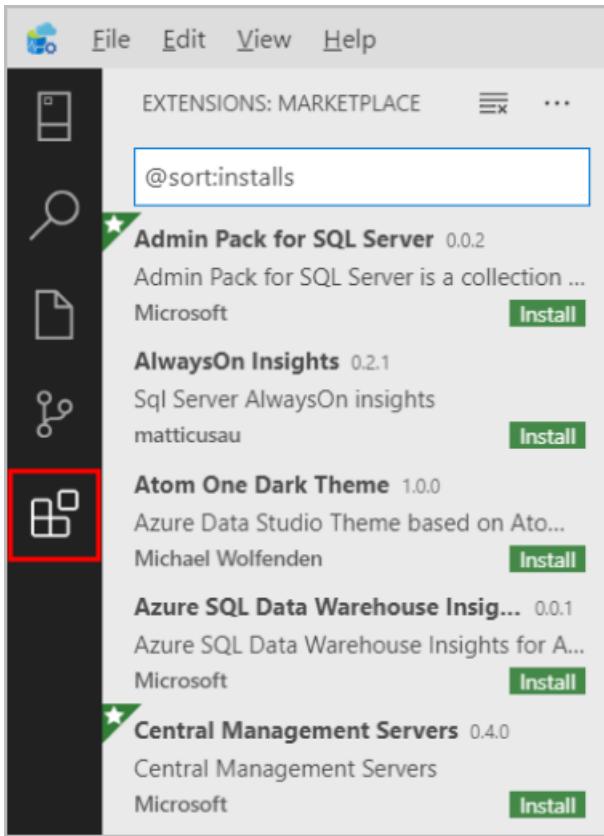


## Why would I use the Data-tier Application Wizard?

The wizard makes it easier to manage dacpac and bacpac files, which simplifies the development and deployment of data-tier elements that support your application. To learn more about using Data-tier applications, [check out our documentation](#).

## Install the extension

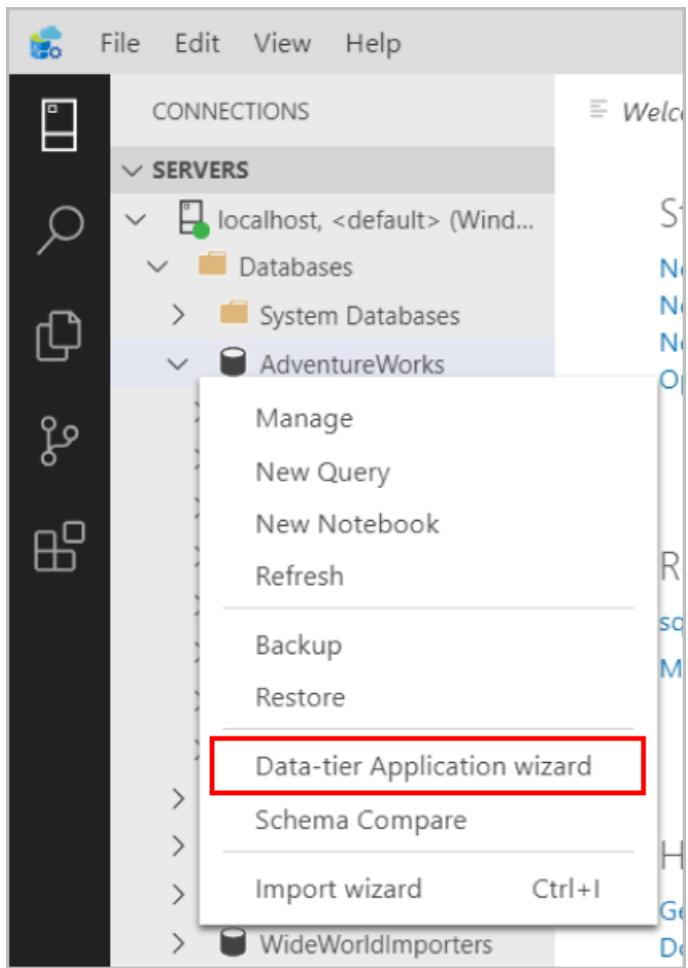
1. Select the Extensions Icon to view the available extensions.



2. Search for the **SQL Server dacpac** extension and select it to view its details. Select **Install** to add the extension.
3. Once installed, **Reload** to enable the extension in Azure Data Studio (only required when installing an extension for the first time).

## Launch the Data-tier Application Wizard

To launch the wizard, right-click the Databases folder or right-click a specific database in the Object Explorer. Then, select **Data-tier Application Wizard**.



## Next steps

To learn more about dacpacs, [check out our documentation](#). Please report issues and feature requests [here](#).

# SQL Server Import extension

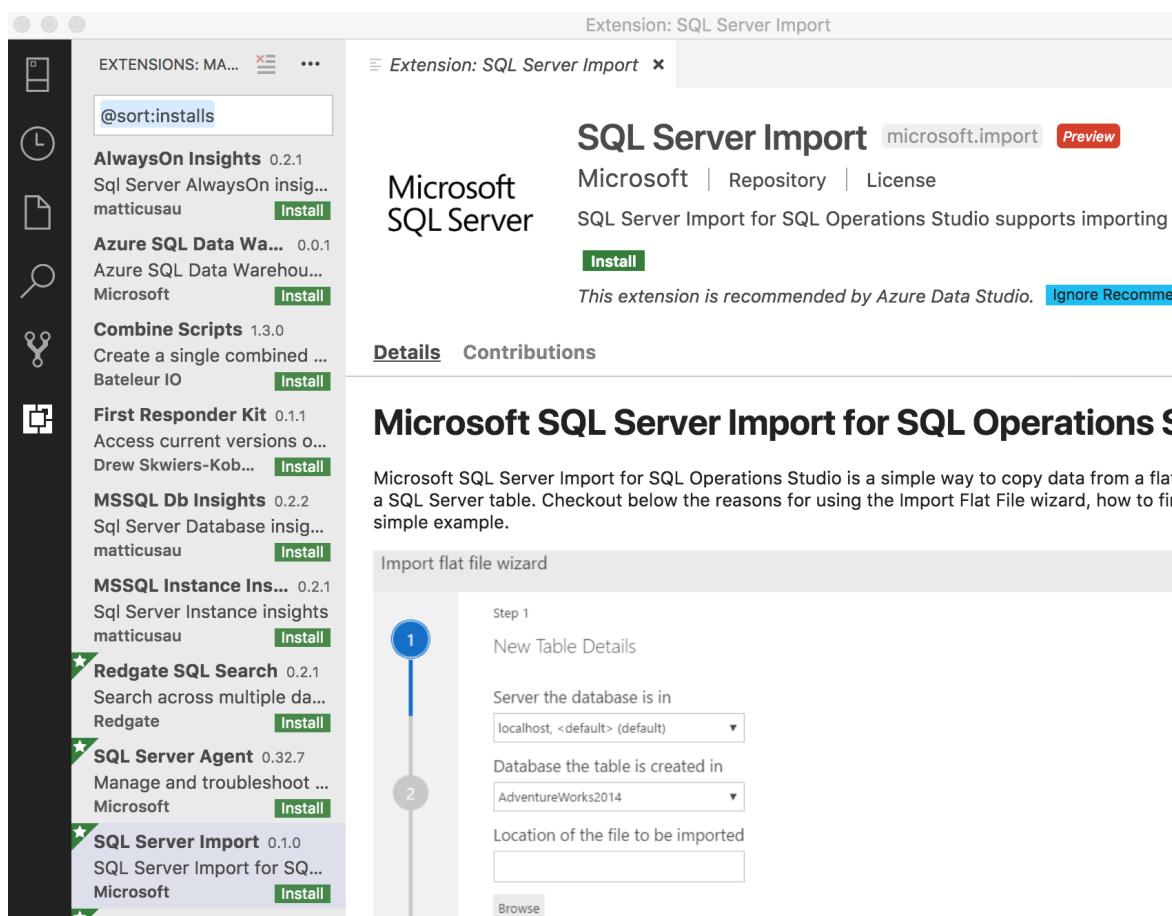
9/26/2022 • 2 minutes to read • [Edit Online](#)

The SQL Server Import extension converts .txt and .csv files into a SQL table. This wizard utilizes a Microsoft Research framework known as [Program Synthesis using Examples \(PROSE\)](#) to intelligently parse the file with minimal user input. It's a powerful framework for data wrangling, and it's the same technology that powers Flash Fill-in Microsoft Excel

To learn more about the SSMS version of this feature, you can read [this article](#).

## Install the SQL Server Import extension

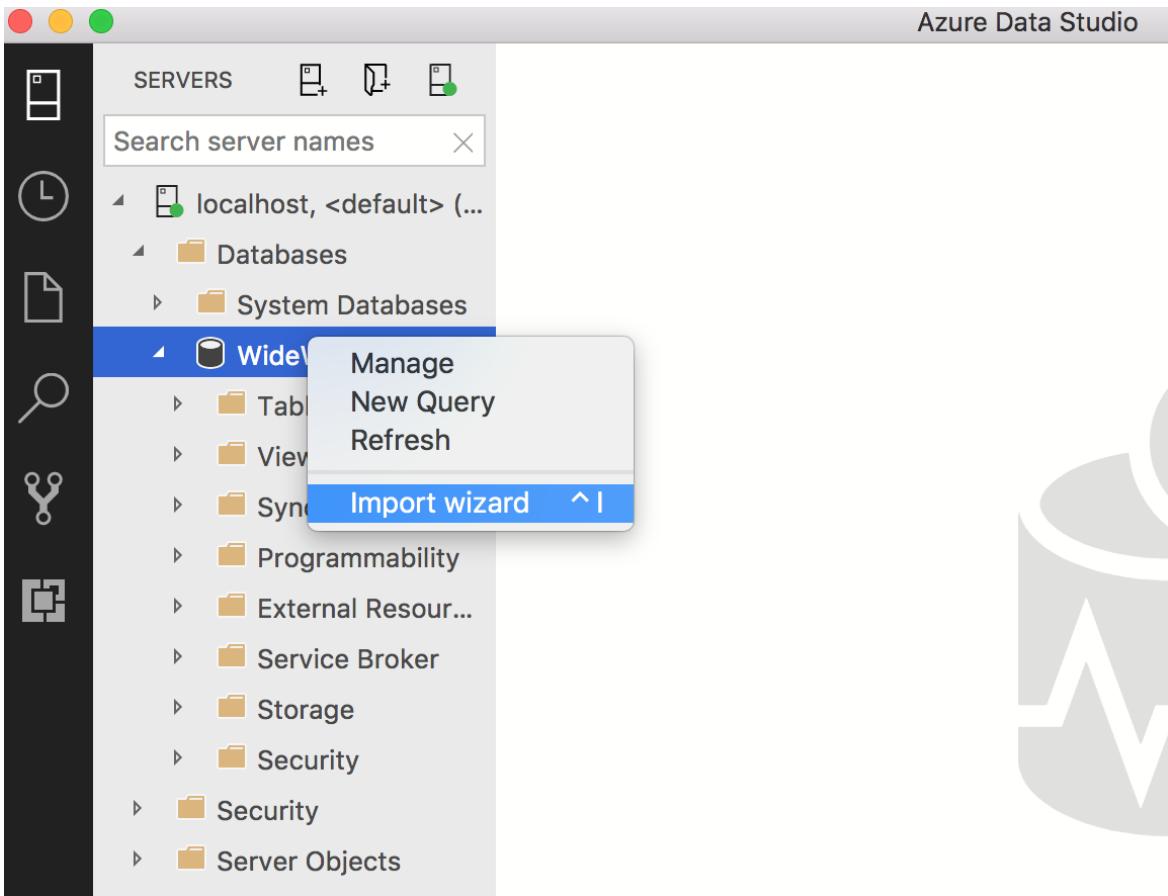
1. To open the extensions manager and access the available extensions, select the extensions icon, or select **Extensions** in the **View** menu.
2. Select an available extension to view its details.



3. Select the extension you want and **Install** it.

## Start Import Wizard

1. To start SQL Server Import, first make a connection to a server in the Servers tab.
2. After you make a connection, drill down to the target database that you want to import a file into a SQL table.
3. Right-click on the database and select **Import Wizard**.

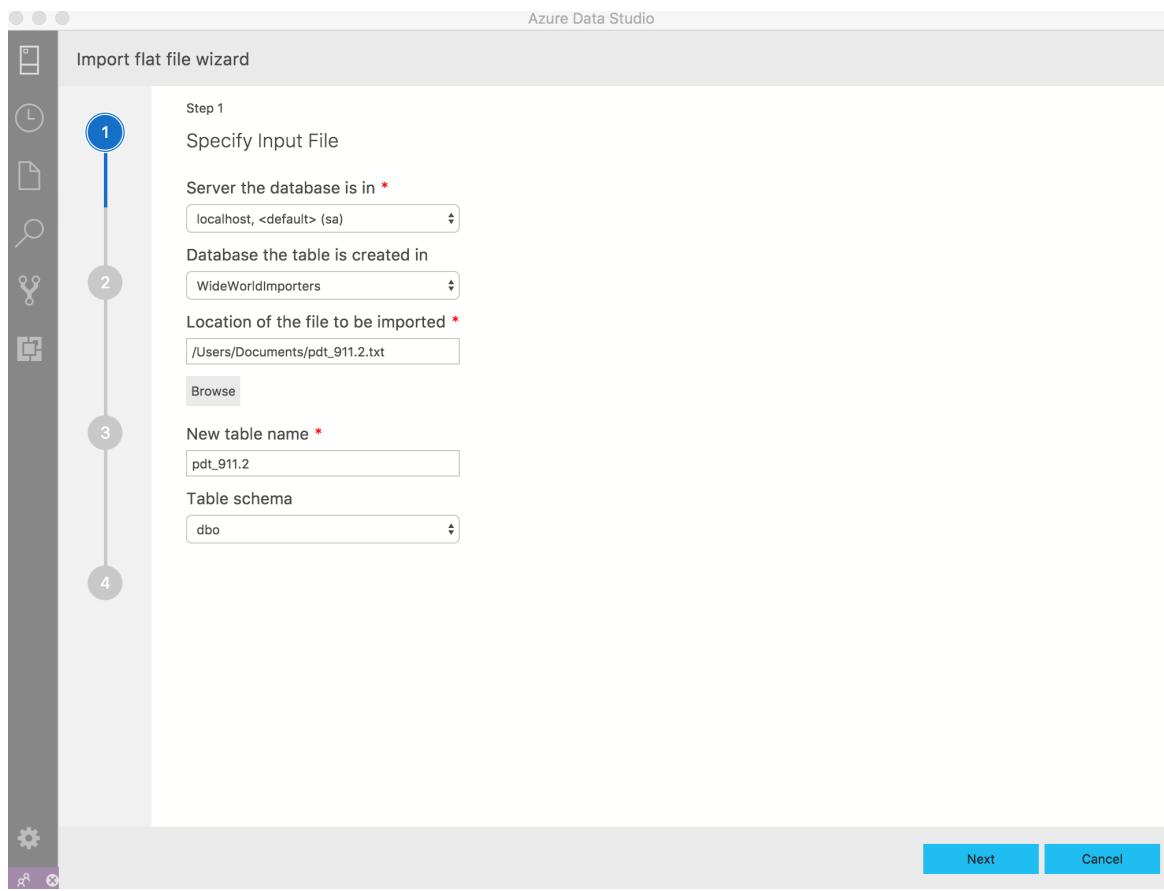


## Importing a file

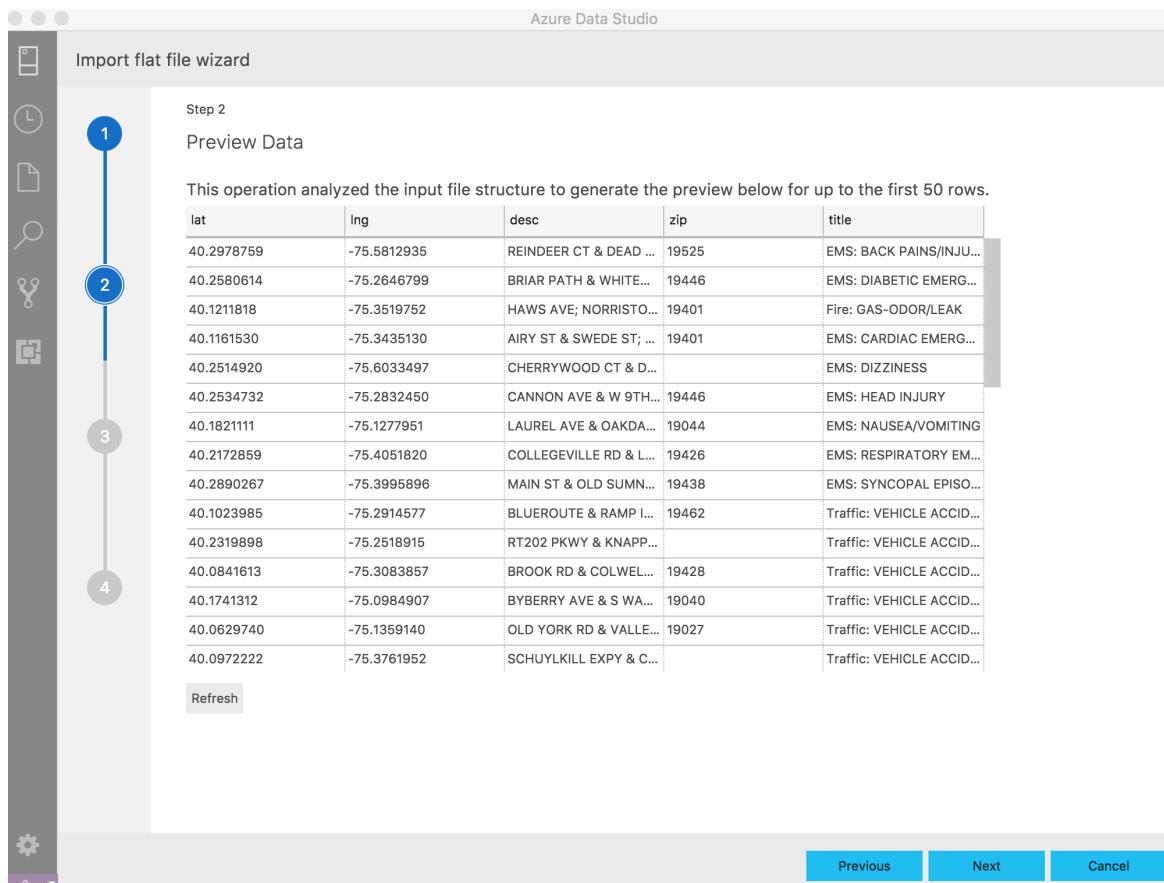
1. When you Right-click to launch the wizard, the server and database are already autofilled. If there are other active connections, you can select in the dropdown.

Select a file by selecting **Browse**. It should autofill the table name based on the file name, but you can also change it yourself.

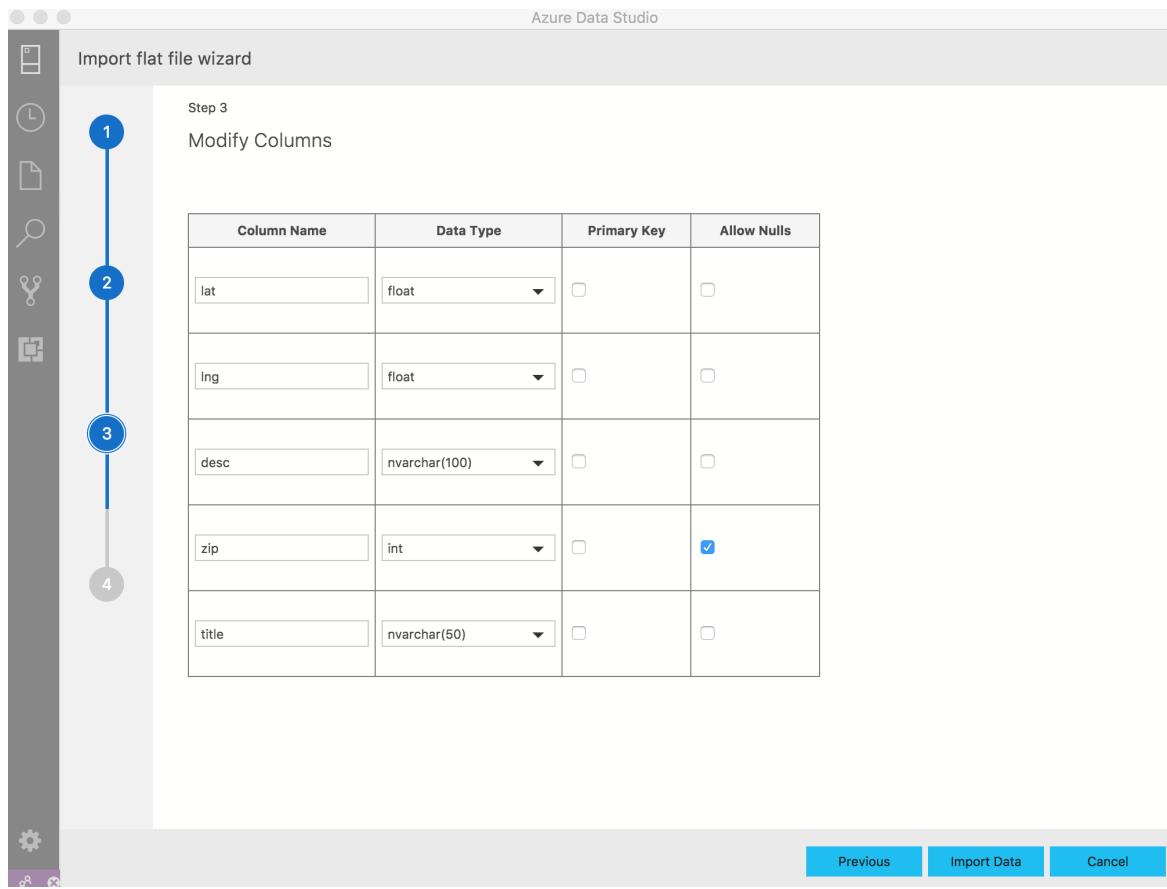
By default, the schema will be dbo but you can change it. select **Next** to proceed.



2. The wizard will generate a preview based on the first 50 rows. Select **Next** to proceed or create new columns calculated based on the contents of existing columns by selecting **Create derived column**. For more information, see the section on [Derived columns](#).

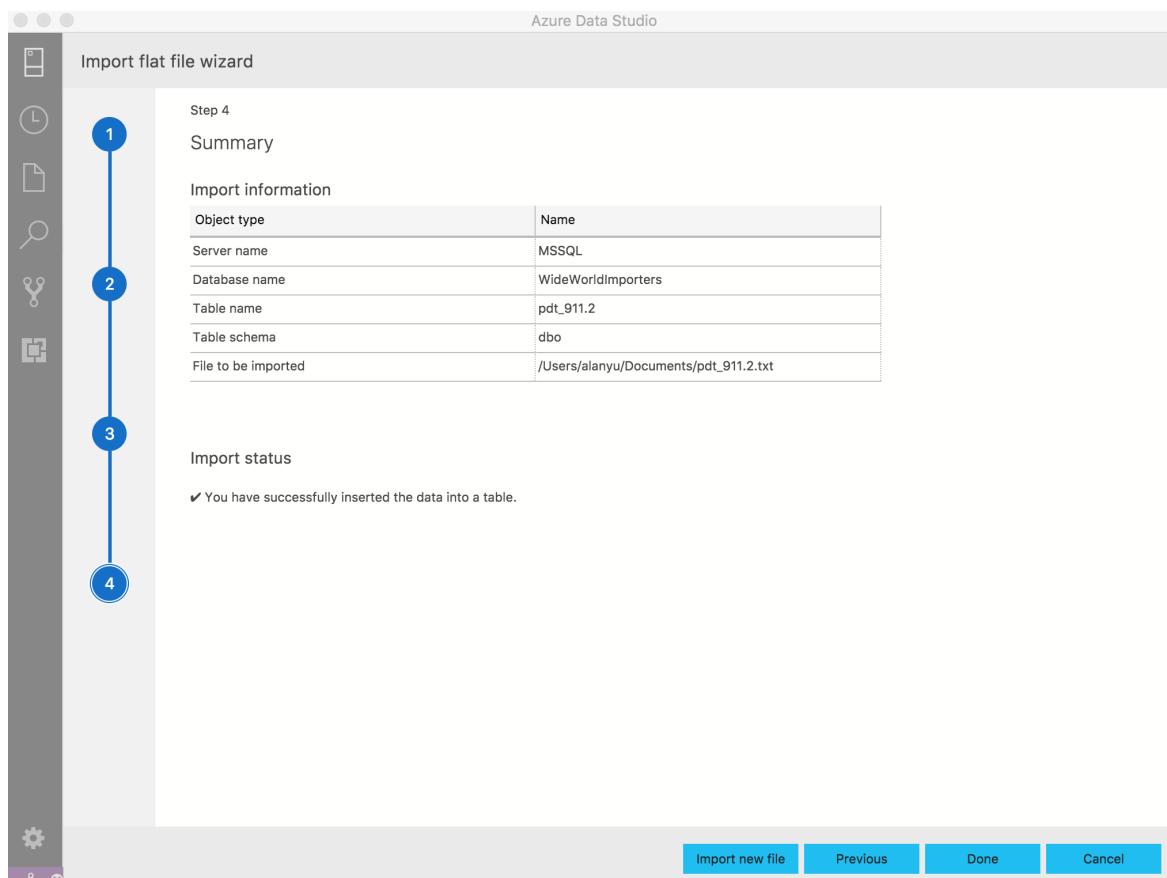


3. On this page, you can make changes to column name, data type, whether it's a primary key, or to allow nulls. You can make as many changes as you like. select **Import Data** to proceed.



4. This page gives a summary of the actions chosen. You can also see whether your table inserted successfully or not.

You can either select **Done**, **Previous** if you need to make changes, or **Import new file** to quickly import another file.



5. Verify if your table successfully imported by refreshing your target database or running a SELECT query on the table name.

## Derived columns

Columns can be added dynamically based on the contents of existing columns. This functionality is especially useful when you want to create a new column based on the contents of another column during the data import. This functionality also uses the [Program Synthesis](#) framework.

When you enter the **Create derived column** portion of the Import Wizard, you will begin by selecting one or more columns that the new column will be based on. Following the column selection you can enter the name of the new column and begin providing one or more values to be used in determining the calculation of the new column. As you enter values, you can pause and select **Preview Transformation** to assess the results of the transformation against a sample of the data.

When you are satisfied with the data transformation displayed, select **Done** to return to the primary Import Wizard and proceed to the next step or continue to add derived columns.

## Next steps

- To learn more about the Import Wizard, read the [initial blog post](#).
- To learn more about PROSE, read the [documentation](#).

# SQL Server Profiler extension (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

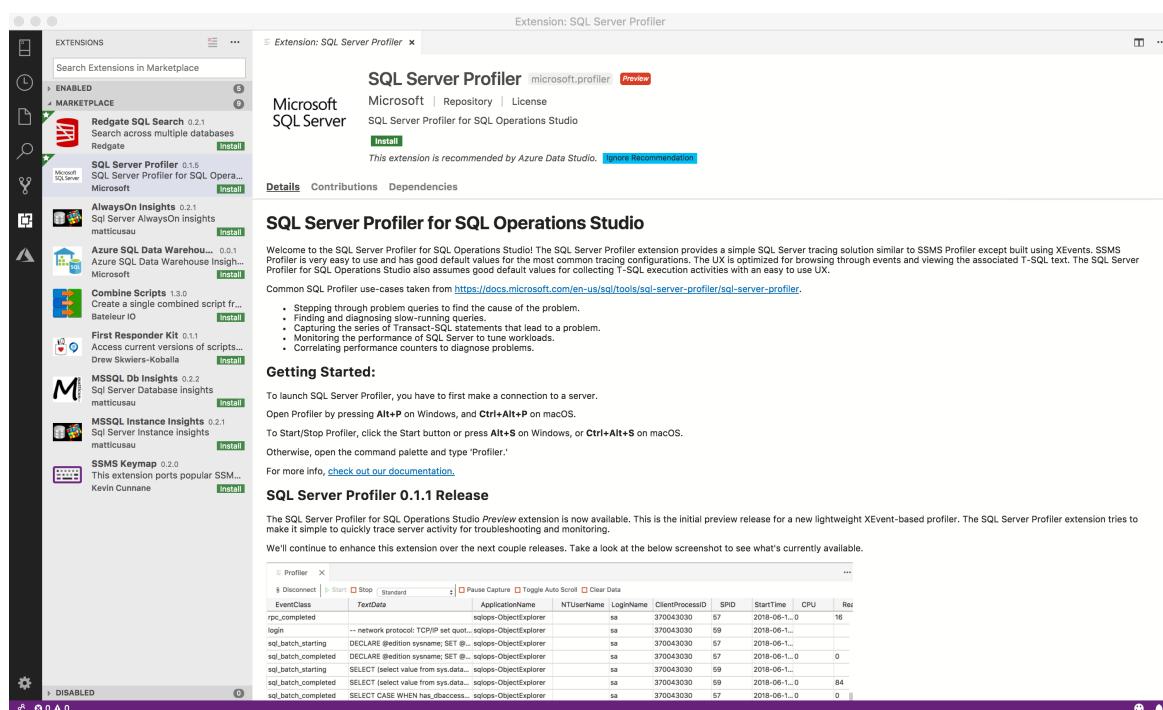
The SQL Server Profiler extension (preview) provides a simple SQL Server tracing solution similar to [SQL Server Management Studio \(SSMS\) Profiler](#) except built using [Extended Events](#). SQL Server Profiler is easy to use and has good default values for the most common tracing configurations. The UX is optimized for browsing through events and viewing the associated Transact-SQL (T-SQL) text. The SQL Server Profiler for Azure Data Studio also assumes good default values for collecting T-SQL execution activities with an easy to use UX. This extension is currently in preview.

## Common SQL Profiler use-cases:

- Stepping through problem queries to find the cause of the problem.
- Finding and diagnosing slow-running queries.
- Capturing the series of Transact-SQL statements that lead to a problem.
- Monitoring the performance of SQL Server to tune workloads.
- Correlating performance counters to diagnose problems.

## Install the SQL Server Profiler extension

1. To open the extensions manager and access the available extensions, select the extensions icon, or select **Extensions** in the **View** menu.
2. Select an available extension to view its details.



3. Select the extension you want and **Install** it.
4. Select **Reload** to enable the extension (only required the first time you install an extension).

## Start Profiler

1. To start Profiler, first make a connection to a server in the Servers tab.

2. After you make a connection, type Alt + P to launch Profiler.

3. To start Profiler, type Alt + S. You can now start seeing Extended Events.

The screenshot shows the SQL Server Profiler interface. The main pane displays a table of event data with columns: EventClass, TextData, ApplicationName, NTUserName, LoginName, ClientProcessID, SPID, StartTime, CPU, Reads, Writes, and Duration. Below the table is a detailed view pane showing specific event properties like Label, Value, and a large text area for the query. The status bar at the bottom indicates the session name is SQLTOOLSVNEXT-5, and the current state is running.

EventClass	TextData	ApplicationName	NTUserName	LoginName	ClientProcessID	SPID	StartTime	CPU	Reads	Writes	Duration
sql_batch_completed	use (coriveraDB)	sql:ops-languageSe...	sa	673399356	152	2018-09-14T21:04...	0	0	0	156	
rpc_completed		sql:ops-languageSe...	sa	673399356	152	2018-09-14T21:05...	31000	1217	0	36136	
attention		sql:ops-languageSe...	sa	673399356	152	2018-09-14T21:05...				56	
logout		sql:ops-languageSe...	sa	673399356	152	2018-09-14T21:05...	31000	1664	0	1267000	
login	-- network protocol...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...					
sql_batch_starting	SELECT ISNULL(SE...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...					
sql_batch_completed	SELECT ISNULL(SE...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...	0	0	0	353	
sql_batch_starting	SELECT CONVERT(...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...					
sql_batch_completed	SELECT CONVERT(...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...	0	0	0	229	
sql_batch_starting	set LOCK_TIMEOUT...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...					
sql_batch_completed	set LOCK_TIMEOUT...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...	0	0	0	49	
sql_batch_starting	SET ANSI_NULLS...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...					
sql_batch_completed	SET ANSI_NULLS...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...				110	
sql_batch_starting	SELECT SERVERPR...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...					
sql_batch_completed	SELECT SERVERPR...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...	0	5	0	228	
sql_batch_starting	SELECT OSVersion...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...					
sql_batch_completed	SELECT OSVersion...	sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...	0	0	0	362	
logout		sql:ops-GeneralCon...	sa	673399356	152	2018-09-14T21:05...	0	17	0	423000	
login	-- network protocol...	sql:ops-languageSe...	sa	673399356	152	2018-09-14T21:05...					
sql_batch_starting	SELECT ISNULL(SE...	sql:ops-LanguageSe...	sa	673399356	187	2018-09-14T21:05...					
sql_batch_completed	SELECT ISNULL(SE...	sql:ops-LanguageSe...	sa	673399356	187	2018-09-14T21:05...	0	11	0	411	
sql_batch_starting	SELECT CONVERT(...	sql:ops-LanguageSe...	sa	673399356	187	2018-09-14T21:05...					
sql_batch_completed	SELECT CONVERT(...	sql:ops-LanguageSe...	sa	673399356	187	2018-09-14T21:05...	0	0	0	294	
logout		sql:ops-languageSe...	sa	673399356	152	2018-09-14T21:05...	0	23	0	110000	
sql_batch_starting	SELECT CONVERT(...	sql:ops-Query	sa	673399356	187	2018-09-14T21:05...					

TEXT DETAILS

Label	Value
EventClass	sql_batch_starting
StartTime	2018-09-14T21:05:00.560Z
TextData	SELECT SERVERPROPERTY('EngineEdition'), SERVERPROPERTY('productversion'), SERVERPROPERTY ('productlevel'), SERVERP...
SPID	152
LoginName	sa
NTUserName	
ClientProcessID	673399356
ApplicationName	sql:ops-GeneralConnection

SQLTOOLSVNEXT-5 : coriveraDB | Ln 1, Col 1 | Spaces: 4 | UTF-8 | CRLF | SQL | ▲ 1

4. To stop Profiler, type Alt + S. This hotkey is a toggle.

## Next steps

To learn more about Profiler and extended events, see [Extended Events](#).

# Extend functionality by creating Azure Data Studio extensions

9/26/2022 • 2 minutes to read • [Edit Online](#)

Extensions in Azure Data Studio provide an easy way to add more functionality to the base Azure Data Studio installation.

Extensions are provided by the Azure Data Studio team (Microsoft), as well as the third-party community (you!).

## Author an extension

If you're interested in extending Azure Data Studio, you can create your own extension and publish it to the extensions gallery.

### Write an extension

#### Prerequisites

To develop an extension, you need [Node.js](#) installed and available in your `$PATH`. Node.js includes npm, the Node.js Package Manager, which is used to install the extension generator.

To create your new extension, you can use the Azure Data Studio extension generator. The Yeoman [extension generator](#) is a beneficial starting point for extension projects. To start the generator, enter the following command in a command prompt:

```
npm install -g yo generator-azuredatastudio
yo azuredatastudio
```

For an in-depth guide on how to get started with your extension template, see [keymap extension](#), which walks you through the creation of an extension.

#### Extensibility references

To learn about Azure Data Studio extensibility, see [Extensibility overview](#). You can also see examples of how to use the API in existing [samples](#).

## Debug an extension

You can debug your new extension by using the Visual Studio Code extension [Azure Data Studio Debug](#).

To debug your extension:

1. Open your extension with [Visual Studio Code](#).
2. Install the Azure Data Studio Debug extension.
3. Select F5, or select the **Debug** icon and then select **Start**.
4. A new instance of Azure Data Studio starts in a special mode (Extension Development Host). This new instance is now aware of your extension.

## Create an extension package

After writing your extension, you need to create a VSIX package that installs in Azure Data Studio. You can use [vscode-vsce](#) (Visual Studio Code Extensions) to create the VSIX package.

```
npm install -g vsce
cd myExtensionName
vsce package
# The myExtensionName.vsix file has now been generated
```

With a VSIX package, you can share your extension locally and privately by sharing the .vsix file and using the command **Extensions: Install From VSIX File** from the command palette to install the extension in Azure Data Studio.

## Publish an extension

To publish your new extension to Azure Data Studio:

1. Add your extension to the [extensions gallery](#).
2. We currently don't have support to host third-party extensions. Instead of downloading the extension, Azure Data Studio has the option to browse to a download page. To set a download page for your extension, set the value of the asset **Microsoft.AzureDataStudio.DownloadPage**.
3. Create a PR against release/extensions branch.
4. Send a review request to the team.

Your extension will be reviewed and added to the extensions gallery.

### Publish extension updates

The process to publish updates is similar to publishing the extension. Make sure the version is updated in package.json.

## Next steps

See one of the following extension authoring tutorials for step-by-step instructions on how to get started:

- [Keymap extension tutorial](#)
- [Dashboard extension tutorial](#)
- [Notebook extension tutorial](#)
- [Jupyter Book extension tutorial](#)
- [Wizard extension tutorial](#)

# Azure Data Studio extensibility

9/26/2022 • 3 minutes to read • [Edit Online](#)

Azure Data Studio has several extensibility mechanisms to customize the user experience and make those customizations available to the entire user community. The core Azure Data Studio platform is built upon Visual Studio Code, so most of the Visual Studio Code extensibility APIs are available. Additionally, we've provided additional extensibility points for data management-specific activities.

Some of the key extensibility points are:

- Visual Studio Code extensibility APIs
- Azure Data Studio extension authoring tools
- Manage Dashboard tab panel contributions
- Insights with Actions experiences
- Azure Data Studio extensibility APIs
- Custom Data Provider APIs

## Visual Studio Code extensibility APIs

Because the core Azure Data Studio platform is built upon Visual Studio Code, details about the Visual Studio Code extensibility APIs are found in the [Extension Authoring](#) and [Extension API](#) documentation on the Visual Studio Code website.

### NOTE

Azure Data Studio releases are aligned with a recent version of VS Code, however the included VS Code engine may not be the current VS Code release. For example, in November 2020 the VS Code engine in Azure Data Studio is 1.48 and the current VS Code version is 1.51. The error message "Unable to install extension '<name>' as it is not compatible with VS Code <version>" when installing an extension is caused by an extension that has a later VS Code engine version defined in the package manifest (`package.json`). You can verify the VS Code engine version in your Azure Data Studio through the [Help](#) menu under **About**.

## Manage Dashboard tab panel contributions

For details, see [Contribution Points](#) and [Context Variables](#).

## Azure Data Studio extensibility APIs

For details, see [Extensibility APIs](#).

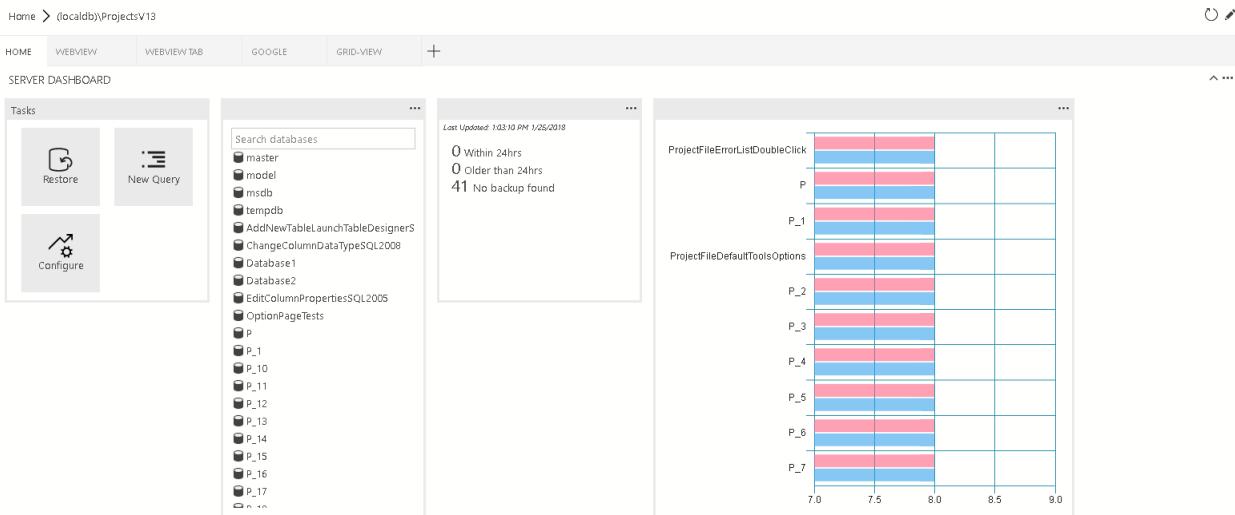
## Contribution points

This section covers the various contribution points that are defined in the package.json extension manifest.

The IntelliSense is supported inside azuredatastudio.

### Dashboard contribution points

Contribute a tab, container, and/or insight widget to the dashboard.



#### dashboard.tabs

Dashboard.tabs creates the tab sections inside the dashboard page. It expects an object or an array of objects.

```
"dashboard.tabs": [
{
  "id": "test-tab1",
  "title": "Test 1",
  "description": "The test 1 displays a list of widgets.",
  "when": "connectionProvider == 'MSSQL' && !mssql:iscloud",
  "alwaysShow": true,
  "container": {
    ...
  }
}
]
```

#### dashboard.containers

Instead of specifying dashboard container inline (inside the dashboard.tab). You can register containers using dashboard.containers. It accepts an object or an array of the object.

```
"dashboard.containers": [
{
  "id": "innerTab1",
  "container": {
    ...
  }
},
{
  "id": "innerTab2",
  "container": {
    ...
  }
}
]
```

To refer to registered container, specify the id of the container

```
"dashboard.tabs": [
{
...
"container": {
"innerTab1": {
}
}
}
]
```

#### dashboard.insights

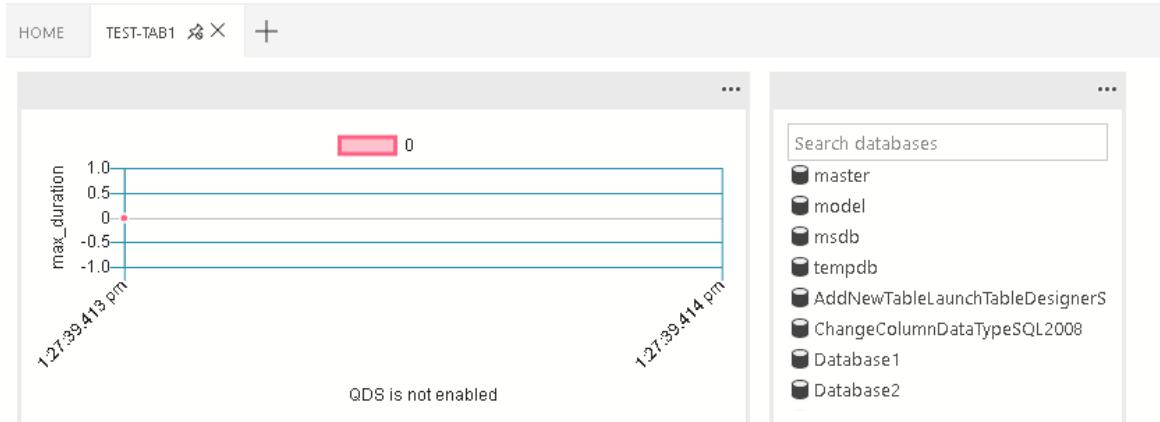
You can register insights using dashboard.insights. This is similar to [Tutorial: Build a custom insight widget](#)

```
"dashboard.insights": {
"id": "my-widget"
"type": {
"count": {
"dataDirection": "vertical",
"dataType": "number",
"legendPosition": "none",
"labelFirstColumn": false,
"columnsAsLabels": false
},
"queryFile": "{your file folder}/activeSession.sql"
}
```

## Dashboard container types

There are currently four supported container types:

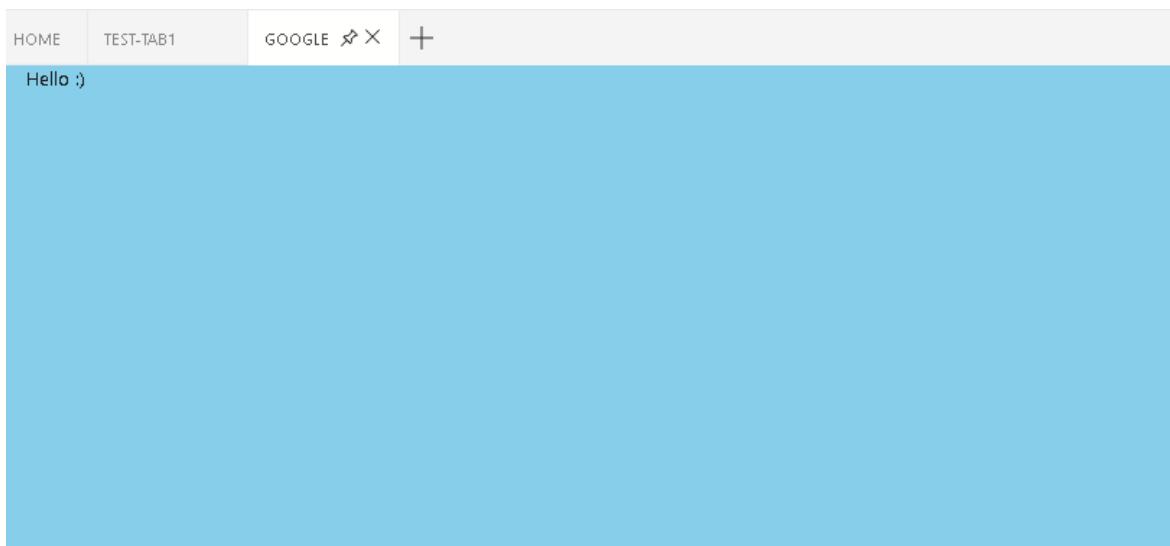
### 1. widgets-container



The list of widgets that will be displayed in the container. It's a flow layout. It accepts the list of widgets.

```
"container": {  
  "widgets-container": [  
    {  
      "widget": {  
        "query-data-store-db-insight": {  
          }  
        }  
      },  
      {  
        "widget": {  
          "explorer-widget": {  
            }  
          }  
        }  
      ]  
    }
```

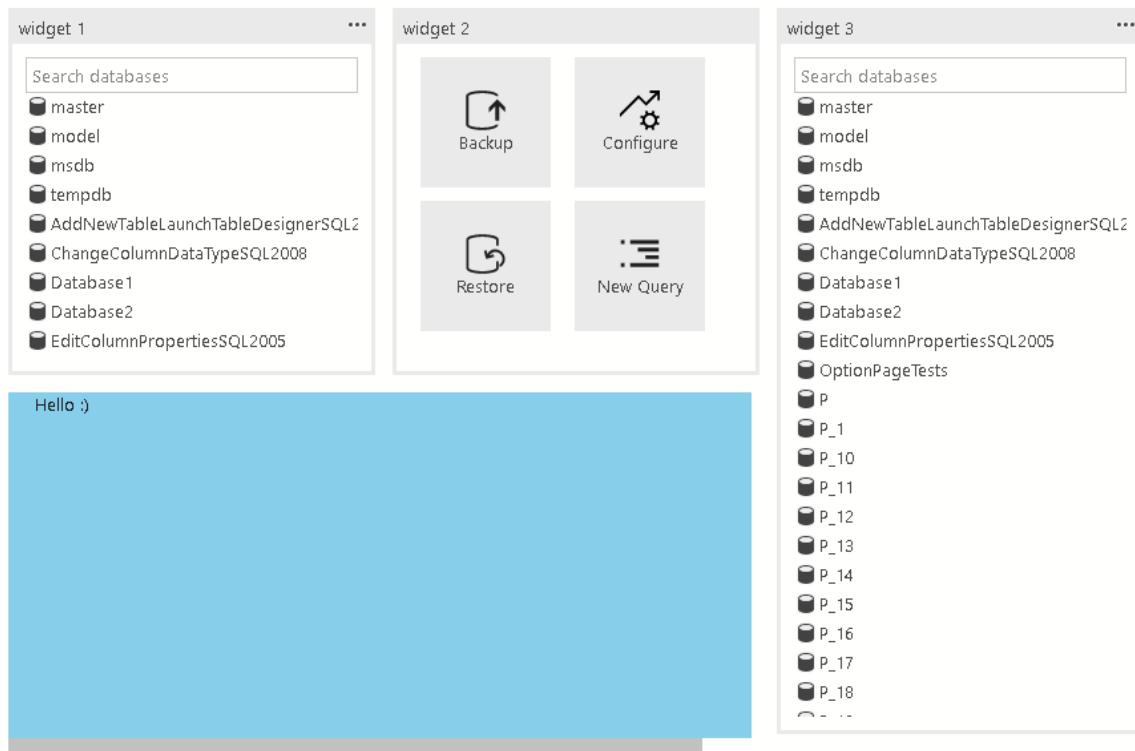
2. `webview-container`



The webview will be displayed in the entire container. It expects webview id to be the same is tab ID

```
"container": {  
  "webview-container": null  
}
```

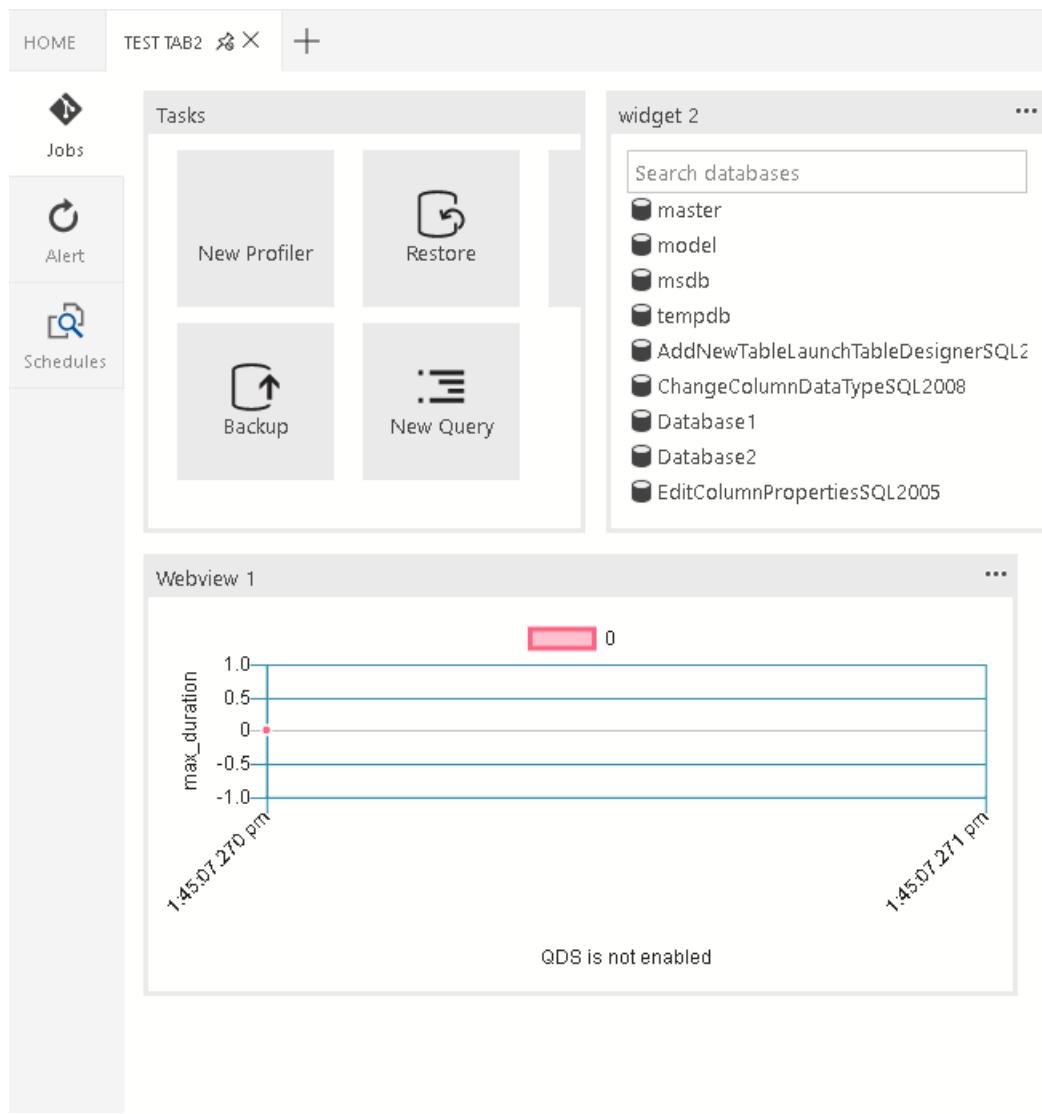
3. `grid-container`



The list of widgets or webviews that will be displayed in the grid layout

```
"container": {
  "grid-container": [
    {
      "name": "widget 1",
      "widget": {
        "explorer-widget": {}
      },
      "row":0,
      "col":0
    },
    {
      "name": "widget 2",
      "widget": {
        "tasks-widget": {
          "backup",
          "restore",
          "configureDashboard",
          "newQuery"
        }
      },
      "row":0,
      "col":1
    },
    {
      "name": "Webview 1",
      "webview": {
        "id": "google"
      },
      "row":1,
      "col":0,
      "colspan":2
    },
    {
      "name": "widget 3",
      "widget": {
        "explorer-widget": {}
      },
      "row":0,
      "col":3,
      "rowspan":2
    },
  ]
}
```

4. `nav-section`



The navigation section will be displayed in the container

```
"container": {  
  "nav-section": [  
    {  
      "id": "innerTab1",  
      "title": "inner-tab1",  
      "icon": {  
        "light": "./icons/tab1Icon.svg",  
        "dark": "./icons/tab1Icon_dark.svg"  
      },  
      "container": {  
        ...  
      }  
    },  
    {  
      "id": "innerTab2",  
      "title": "inner-tab2",  
      "icon": {  
        "light": "./icons/tab2Icon.svg",  
        "dark": "./icons/tab2Icon_dark.svg"  
      },  
      "container": {  
        ...  
      }  
    }  
  ]  
}
```

# Context variables

For general information about context in Visual Studio Code and subsequently Azure Data Studio, see [Extensibility](#).

In Azure Data Studio, we have specific context around database connections available for extensions.

## Dashboard

In dashboard, we provide the following context variables:

CONTEXT VARIABLE	DESCRIPTION
<code>connectionProvider</code>	A string of the identifier for the provider of the current connection. Ex. <code>connectionProvider == 'MSSQL'</code> .
<code>serverName</code>	A string of the server name of the current connection. Ex. <code>serverName == 'localhost'</code> .
<code>databaseName</code>	A string of the database name of the current connection. Ex. <code>databaseName == 'master'</code> .
<code>connection</code>	The full connection profile object for the current connection ( <code>IConnectionProfile</code> )
<code>dashboardContext</code>	A string of the context of the page the dashboard is currently on. Either 'database' or 'server'. Ex. <code>dashboardContext == 'database'</code>

# Azure Data Studio extensibility APIs

9/26/2022 • 4 minutes to read • [Edit Online](#)

Azure Data Studio provides an API that extensions can use to interact with other parts of Azure Data Studio, such as Object Explorer. These APIs are available from the [src/sql/azdata.d.ts](#) file and are described below.

## Connection Management

`azdata.connection`

### Top-level Functions

- `getCurrentConnection(): Thenable<azdata.connection.Connection>` Gets the current connection based on the active editor or Object Explorer selection.
- `getActiveConnections(): Thenable<azdata.connection.Connection[]>` Gets a list of all the user's connections that are active. Returns an empty list if there are no such connections.
- `getCredentials(connectionId: string): Thenable<{ [name: string]: string }>` Gets a dictionary containing the credentials associated with a connection. These would otherwise be returned as part of the options dictionary under a `azdata.connection.Connection` object but get stripped from that object.

`Connection`

- `options: { [name: string]: string }` The dictionary of connection options
- `providerName: string` The name of the connection provider (e.g. "MSSQL")
- `connectionId: string` The unique identifier for the connection

### Example Code

```
> let connection = azdata.connection.getCurrentConnection();
connection: {
  providerName: 'MSSQL',
  connectionId: 'd97bb63a-466e-4ef0-ab6f-00cd44721dcc',
  options: {
    server: 'mairvine-sql-server',
    user: 'sa',
    authenticationType: 'sqlLogin',
    ...
  },
  ...
}
> let credentials = azdata.connection.getCredentials(connection.connectionId);
credentials: {
  password: 'abc123'
}
```

## Object Explorer

`azdata.objectexplorer`

### Top-level Functions

- `getNode(connectionId: string, nodePath?: string): Thenable<azdata.objectexplorer.ObjectExplorerNode>` Get an Object Explorer node corresponding to the given connection and path. If no path is given, it

returns the top-level node for the given connection. If there is no node at the given path, it returns `undefined`. Note: The `nodePath` for an object is generated by the SQL Tools Service backend and is difficult to construct by hand. Future API improvements will allow you to get nodes based on metadata you provide about the node, such as name, type and schema.

- `getActiveConnectionNodes(): Thenable<azdata.objectexplorer.ObjectExplorerNode[]>` Get all active Object Explorer connection nodes.
- `findNodes(connectionId: string, type: string, schema: string, name: string, database: string, parentObjectNames: string[]): Thenable<azdata.objectexplorer.ObjectExplorerNode[]>`  
Find all Object Explorer nodes that match the given metadata. The `schema`, `database`, and `parentObjectNames` arguments should be `undefined` when they are not applicable. `parentObjectNames` is a list of non-database parent objects, from highest to lowest level in Object Explorer, that the desired object is under. For example, when searching for a column "column1" that belongs to a table "schema1.table1" and database "database1" with connection ID `connectionId`, call `findNodes(connectionId, 'Column', undefined, 'column1', 'database1', ['schema1.table1'])`. Also see the [list of types that Azure Data Studio supports by default](#) for this API call.

## ObjectExplorerNode

- `connectionId: string` The id of the connection that the node exists under
- `nodePath: string` The path of the node, as used for a call to the `getNode` function.
- `nodeType: string` A string representing the type of the node
- `nodeSubType: string` A string representing the subtype of the node
- `nodeStatus: string` A string representing the status of the node
- `label: string` The label for the node as it appears in Object Explorer
- `isLeaf: boolean` Whether the node is a leaf node and therefore has no children
- `metadata: azdata.ObjectMetadata` Metadata describing the object represented by this node
- `errorMessage: string` Message shown if the node is in an error state
- `isExpanded(): Thenable<boolean>` Whether the node is currently expanded in Object Explorer
- `setExpandedState(expandedState: vscode.TreeItemCollapsibleState): Thenable<void>` Set whether the node is expanded or collapsed. If the state is set to None, the node will not be changed.
- `setSelected(selected: boolean, clearOtherSelections?: boolean): Thenable<void>` Set whether the node is selected. If `clearOtherSelections` is true, clear any other selections when making the new selection. If it is false, leave any existing selections. `clearOtherSelections` defaults to true when `selected` is true and false when `selected` is false.
- `getChildren(): Thenable<azdata.objectexplorer.ObjectExplorerNode[]>` Get all the child nodes of this node. Returns an empty list if there are no children.
- `getParent(): Thenable<azdata.objectexplorer.ObjectExplorerNode>` Get the parent node of this node. Returns `undefined` if there is no parent.

## Example Code

```

private async interactWithOENode(selectedNode: azdata.objectexplorer.ObjectExplorerNode): Promise<void> {
    let choices = ['Expand', 'Collapse', 'Select', 'Select (multi)', 'Deselect', 'Deselect (multi)'];
    if (selectedNode.isLeaf) {
        choices[0] += ' (is leaf)';
        choices[1] += ' (is leaf)';
    } else {
        let expanded = await selectedNode.isExpanded();
        if (expanded) {
            choices[0] += ' (is expanded)';
        } else {
            choices[1] += ' (is collapsed)';
        }
    }
    let parent = await selectedNode.getParent();
    if (parent) {
        choices.push('Get Parent');
    }
    let children = await selectedNode.getChildren();
    children.forEach(child => choices.push(child.label));
    let choice = await vscode.window.showQuickPick(choices);
    let nextNode: azdata.objectexplorer.ObjectExplorerNode = undefined;
    if (choice === choices[0]) {
        selectedNode.setExpandedState(vscode.TreeItemCollapsibleState.Expanded);
    } else if (choice === choices[1]) {
        selectedNode.setExpandedState(vscode.TreeItemCollapsibleState.Collapsed);
    } else if (choice === choices[2]) {
        selectedNode.setSelected(true);
    } else if (choice === choices[3]) {
        selectedNode.setSelected(true, false);
    } else if (choice === choices[4]) {
        selectedNode.setSelected(false);
    } else if (choice === choices[5]) {
        selectedNode.setSelected(false, true);
    } else if (choice === 'Get Parent') {
        nextNode = parent;
    } else {
        let childNode = children.find(child => child.label === choice);
        nextNode = childNode;
    }
    if (nextNode) {
        let updatedNode = await azdata.objectexplorer.getNode(nextNode.connectionId, nextNode.nodePath);
        this.interactWithOENode(updatedNode);
    }
}

vscode.commands.registerCommand('mssql.objectexplorer.interact', () => {
    azdata.objectexplorer.getActiveConnectionNodes().then(activeConnections => {
        vscode.window.showQuickPick(activeConnections.map(connection => connection.label + ' ' +
connection.connectionId)).then(selection => {
            let selectedNode = activeConnections.find(connection => connection.label + ' ' + connection.connectionId
=== selection);
            this.interactWithOENode(selectedNode);
        });
    });
});

```

## Proposed APIs

We have added proposed APIs to allow extensions to display custom UI in dialogs, wizards, and document tabs, among other capabilities. See the [proposed API types file](#) for more documentation, though be aware that these APIs are subject to change at any time. Examples of how to use some of these APIs can be found in the ["sqlservices" sample extension](#).

# Create an Azure Data Studio Keymap extension

9/26/2022 • 4 minutes to read • [Edit Online](#)

This tutorial demonstrates how to create a new Azure Data Studio extension. The extension creates familiar SSMS key bindings in Azure Data Studio.

In this article you learn how to:

- Create an extension project
- Install the extension generator
- Create your extension
- Add custom key-bindings to your extension
- Test your extension
- Package your extension
- Publish your extension to the marketplace

## Prerequisites

Azure Data Studio is built on the same framework as Visual Studio Code, so extensions for Azure Data Studio are built using Visual Studio Code. To get started, you need the following components:

- [Node.js](#) installed and available in your `$PATH`. Node.js includes [npm](#), the Node.js Package Manager, which is used to install the extension generator.
- [Visual Studio Code](#) to debug the extension.
- The Azure Data Studio [Debug extension](#) (optional). This lets you test your extension without needing to package and install it into Azure Data Studio.
- Ensure `azuredatascudio` is in your path. For Windows, make sure you choose the `Add to Path` option in setup.exe. For Mac or Linux, run `Install 'azuredatascudio'` command in PATH from the Command Palette in Azure Data Studio.

## Install the extension generator

To simplify the process of creating extensions, we've built an [extension generator](#) using Yeoman. To install it, run the code in the command prompt below:

```
npm install -g yo generator-azuredatascudio
```

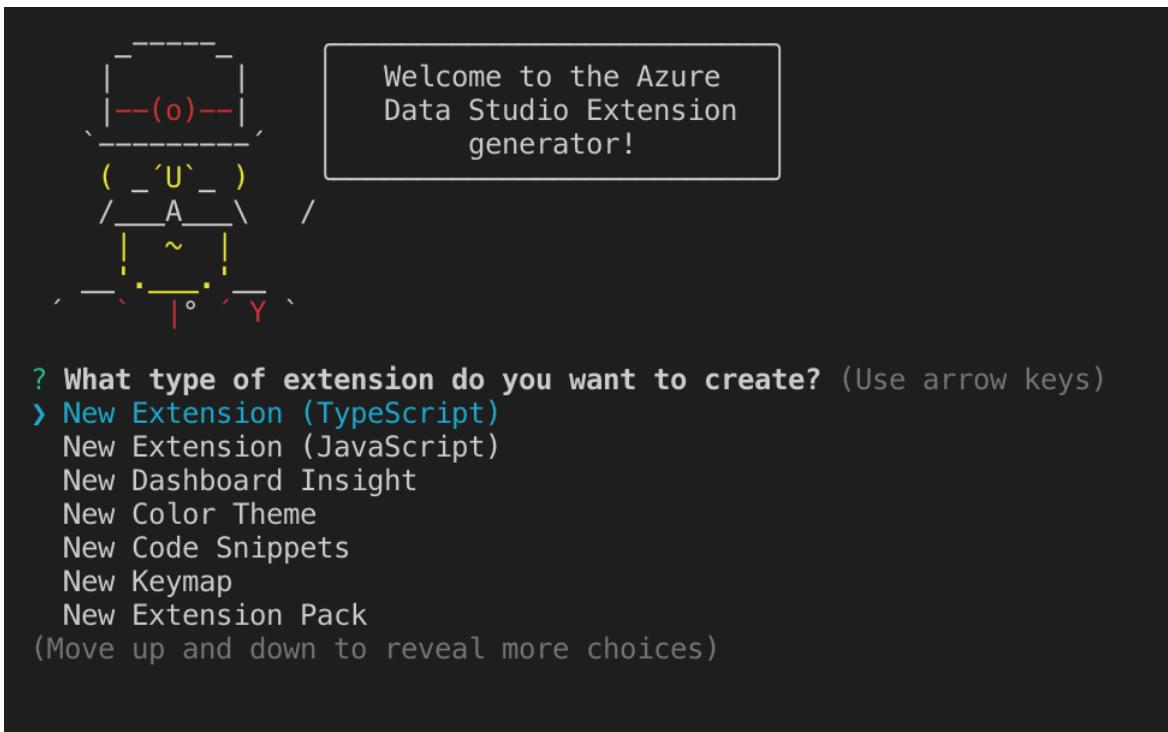
## Create your keymap extension

To create an extension:

1. Launch the extension generator with the following command:

```
yo azuredatascudio
```

2. Choose **New Keymap** from the list of extension types:



3. Follow the steps to fill in the extension name (for this tutorial, use `ssmskeymap2`), and add a description.

Completing the previous steps creates a new folder. Open the folder in Visual Studio Code and you're ready to create your own key binding extension!

### Add a keyboard shortcut

#### Step 1: Find the shortcuts to replace

Now that we have our extension ready to go, add some SSMS keyboard shortcuts (or keybindings) into Azure Data Studio. I used [Andy Mallon's Cheatsheet](#) and RedGate's keyboard shortcuts list for inspiration.

The top things I saw missing were:

- Run a query with the actual execution plan enabled. This is **Ctrl+M** in SSMS and doesn't have a binding in Azure Data Studio.
- Having **CTRL+SHIFT+E** as a second way of running a query. User feedback indicated that this was missing.
- Having **ALT+F1** run `sp_help`. We added this in Azure Data Studio but since that binding was already in use, we mapped it to **ALT+F2** instead.
- Toggle full screen (**SHIFT+ALT+ENTER**).
- **F8** to show Object Explorer / Servers view.

It's easy to find and replace these key bindings. Run *Open Keyboard Shortcuts* to show the **Keyboard Shortcuts** tab in Azure Data Studio, search for *query* and then choose **Change Key binding**. Once you're done changing the keybinding, you can see the updated mapping in the `keybindings.json` file (run *Open Keyboard Shortcuts* to see it).

Keyboard Shortcuts x

query

For advanced customizations open and edit [keybindings.json](#)

Command	Keybinding	Source	When
Cancel <b>Query</b> cancelQueryKeyboardAction	⌘ ⌘ F5 PauseBreak	Default	—
Run Current <b>Query</b> runCurrentQueryKeyboardAction	⌘ ⌘ F5 Copy	Default	—
Run Current <b>Query</b> with Actual Plan runCurrentQueryWithActualPlanKeyboardAction	⌘ ⌘ K ⌘ K Change Keybinding [⌘K ⌘K]	Default	—
Run Current <b>Query</b> with Actual Plan runCurrentQueryWithActualPlanKeyboardAction	⌘ ⌘ Backspace Remove Keybinding [⌘K ⌘Backspace]	Default	—
Run <b>Query</b> runQueryKeyboardAction	Reset Keybinding	Default	—
Run <b>Query</b> runQueryKeyboardAction	Show Conflicts	Default	—
Toggle <b>Query</b> Results toggleQueryResultsKeyboardAction	F5 ^ ⌘ ⌘ R	Default	queryEditorVisible

keybindings.json x ..

```

1 // Place your key bindings in this file to overwrite the defaults
2 [
3   {
4     "key": "f8",
5     "command": "workbench.view.connections"
6   },
7   {
8     "key": "ctrl+m",
9     "command": "runCurrentQueryWithActualPlanKeyboardAction"
10 }
11 ]

```

## Step 2: Add shortcuts to the extension

To add shortcuts to the extension, open the `package.json` file (in the extension) and replace the `contributes` section with the code below:

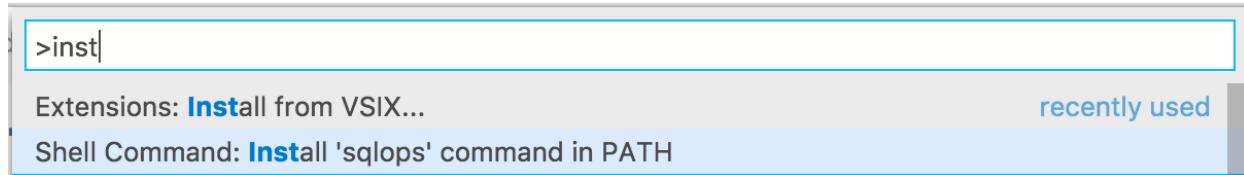
```
"contributes": {  
  "keybindings": [  
    {  
      "key": "shift+cmd+e",  
      "command": "runQueryKeyboardAction"  
    },  
    {  
      "key": "ctrl+cmd+e",  
      "command": "workbench.view.explorer"  
    },  
    {  
      "key": "alt+f1",  
      "command": "workbench.action.query.shortcut1"  
    },  
    {  
      "key": "shift+alt+enter",  
      "command": "workbench.action.toggleFullScreen"  
    },  
    {  
      "key": "f8",  
      "command": "workbench.view.connections"  
    },  
    {  
      "key": "ctrl+m",  
      "command": "runCurrentQueryWithActualPlanKeyboardAction"  
    }  
  ]  
}
```

## Test your extension

Ensure `azuredatastudio` is in your PATH by running the `Install azuredatastudio` command in PATH command in Azure Data Studio.

Ensure the Azure Data Studio Debug extension is installed in Visual Studio Code.

Select F5 to launch Azure Data Studio in debug mode with the extension running:



The screenshot shows the SSMS interface. At the top, there are tabs for 'Keyboard Shortcuts' and 'Default Keybindings'. On the right, there's a connection status bar for 'SQLQuery1'. Below the tabs, there are buttons for 'Run', 'Cancel', 'Disconnect', and 'Change Connection'. A dropdown menu labeled 'Select Database' is open. To the right of it is an 'Explain' button. The main area contains a query window with the following content:

```
1  select * from sys.tables
```

The screenshot shows the 'TOP OPERATIONS' tab selected in the SSMS interface. It displays the execution plan for the query 'select \* from sys.tables'. The plan consists of three nested loops. The innermost loop is a 'Compute Scalar' operation with a cost of 0%. This is followed by two 'Nested Loops' operations, each with a 'Left Outer Join' logical operation. The outermost loop is another 'Nested Loops' operation. Below the diagram, there is a table with the following data:

Physical Operation	Nested Loops
Logical Operation	Left Outer Join
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	5
Actual Number of Batches	0

Key maps are one of the quickest extensions to create, so your new extension should now be successfully working and ready to share.

## Package your extension

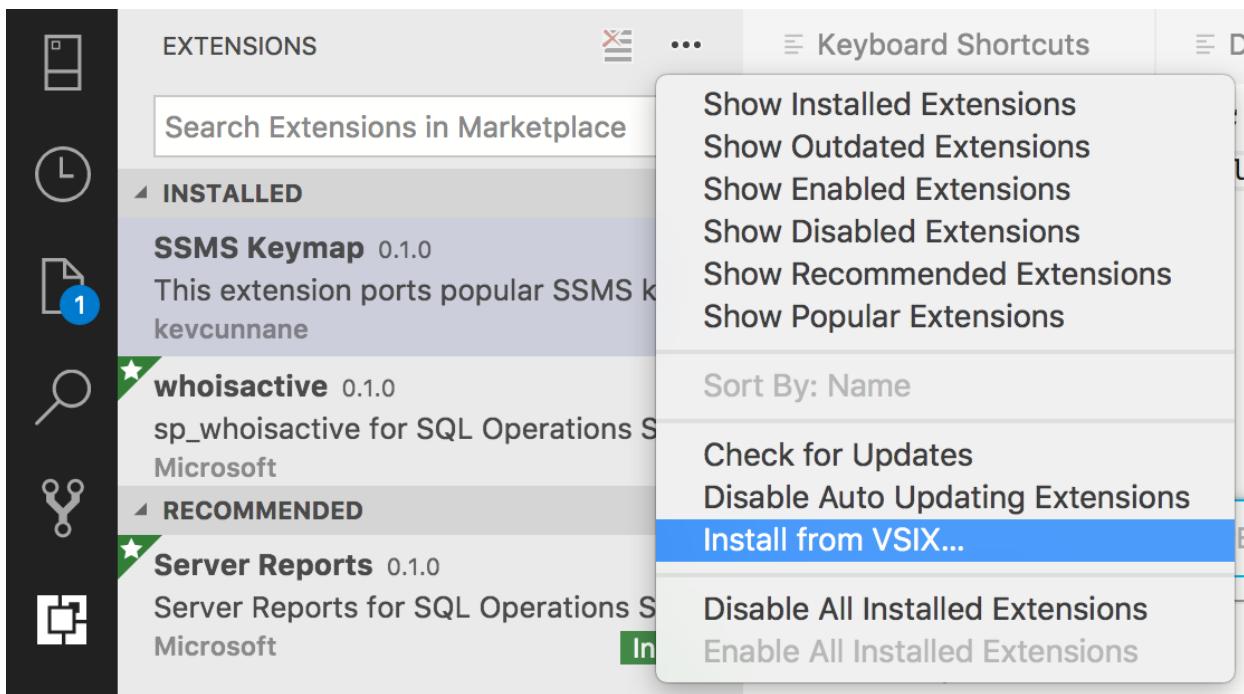
To share with others, you need to package the extension into a single file. This can be published to the Azure Data Studio extension Marketplace, or shared among your team or community. To do this, you need to install another npm package from the command line:

```
npm install -g vsce
```

Navigate to the base directory of the extension, and run `vsce package`. I had to add in a couple of extra lines to stop the `vsce` tool from complaining:

```
"repository": {
    "type": "git",
    "url": "https://github.com/kevcunnane/ssmskeymap.git"
},
"bugs": {
    "url": "https://github.com/kevcunnane/ssmskeymap/issues"
},
```

Once this was done, my `ssmskeymap-0.1.0.vsix` file was created and ready to install and share with the world!



## Publish your extension to the Marketplace

The Azure Data Studio extension Marketplace is under construction, but the current process is to host the extension VSIX somewhere (for example, a GitHub Release page) then submit a PR updating [this JSON file](#) with your extension info.

## Next steps

In this tutorial, you learned how to:

- Create an extension project
- Install the extension generator
- Create your extension
- Add custom key-bindings to your extension
- Test your extension
- Package your extension
- Publish your extension to the marketplace

We hope after reading this you'll be inspired to build your own extension for Azure Data Studio. We have support for Dashboard Insights (pretty graphs that run against your SQL Server), a number of SQL-specific APIs, and a huge existing set of extension points inherited from Visual Studio Code.

If you have an idea but are not sure how to get started, open an issue or tweet at the team: [azuredatrstudio](#).

You can always refer to the [Visual Studio Code extension guide](#) because it covers all the existing APIs and patterns.

To learn how to work with T-SQL in Azure Data Studio, complete the T-SQL Editor tutorial:

[Use the Transact-SQL editor to create database objects](#)

# Create an Azure Data Studio dashboard extension

9/26/2022 • 4 minutes to read • [Edit Online](#)

This tutorial demonstrates how to create a new Azure Data Studio dashboard extension. The extension contributes to the Azure Data Studio connection dashboard, so you can extend the functionality of Azure Data Studio in a manner easily visible to users.

In this article you learn how to:

- Install the extension generator.
- Create your extension.
- Contribute to the dashboard in your extension.
- Test your extension.
- Package your extension.
- Publish your extension to the marketplace.

## Prerequisites

Azure Data Studio is built on the same framework as Visual Studio Code, so extensions for Azure Data Studio are built by using Visual Studio Code. To get started, you need the following components:

- [Node.js](#) installed and available in your `$PATH`. Node.js includes [npm](#), the Node.js Package Manager, which is used to install the extension generator.
- [Visual Studio Code](#) to debug the extension.
- The Azure Data Studio [Debug extension](#) (optional). The Debug extension lets you test your extension without needing to package and install it in Azure Data Studio.
- Ensure `azuredatascudio` is in your path. For Windows, make sure you choose the **Add to Path** option in setup.exe. For Mac or Linux, run **Install 'azuredatascudio' command in PATH** from the Command Palette in Azure Data Studio.

## Install the extension generator

To simplify the process of creating extensions, we've built an [extension generator](#) using Yeoman. To install it, run the following command from the command prompt:

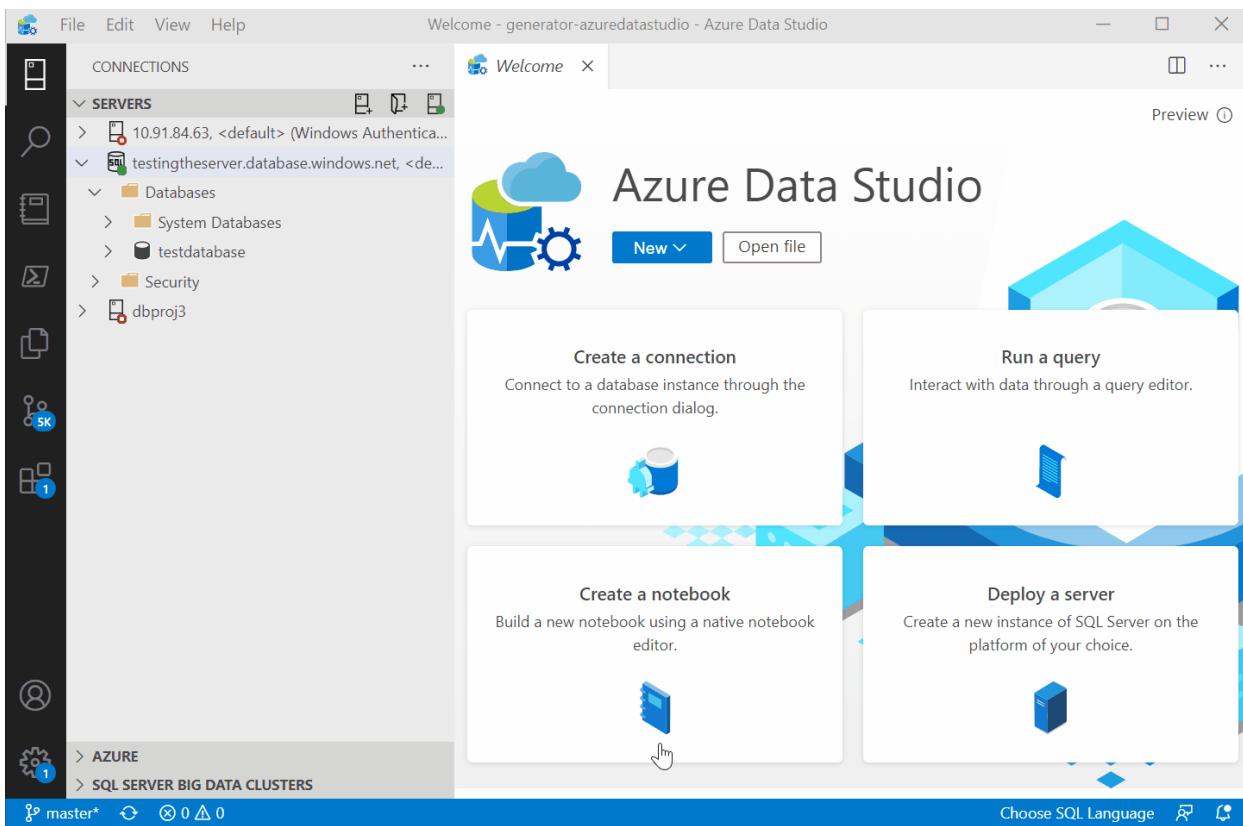
```
npm install -g yo generator-azuredatascudio
```

## Create your dashboard extension

### Introduction to the dashboard

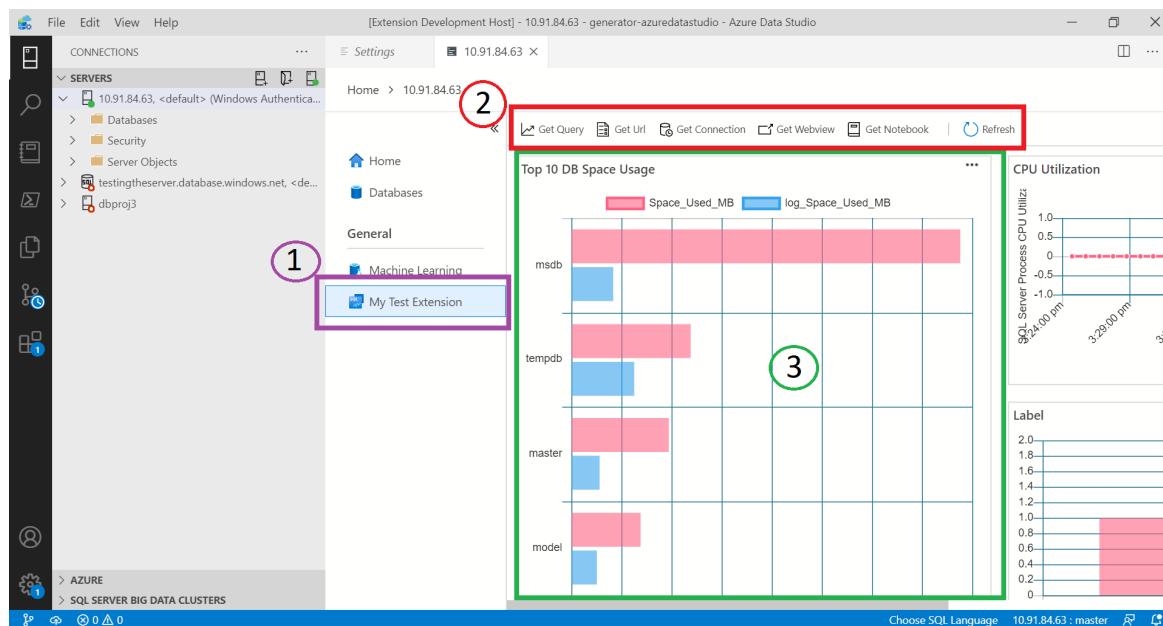
The Azure Data Studio connection dashboard is a powerful tool that summarizes and provides insight into a user's connections.

There are two variations of the dashboard. The *server dashboard* summarizes the entire server, and the *database dashboard* summarizes an individual database. You can access either dashboard by right-clicking a server or a database in the **Connections** viewlet of Azure Data Studio, and selecting **Manage**.



There are three key contribution points for extensions to add functionality to the dashboard:

- Full Dashboard tab:** A separate tab in the dashboard for your extension. Can be added to either a server or database dashboard. Customizable with widgets, a toolbar, and a navigation section.
- Homepage Actions:** Action buttons at the top of the connection toolbar.
- Widgets:** Graphs that run against your SQL Server.



## Run the extension generator

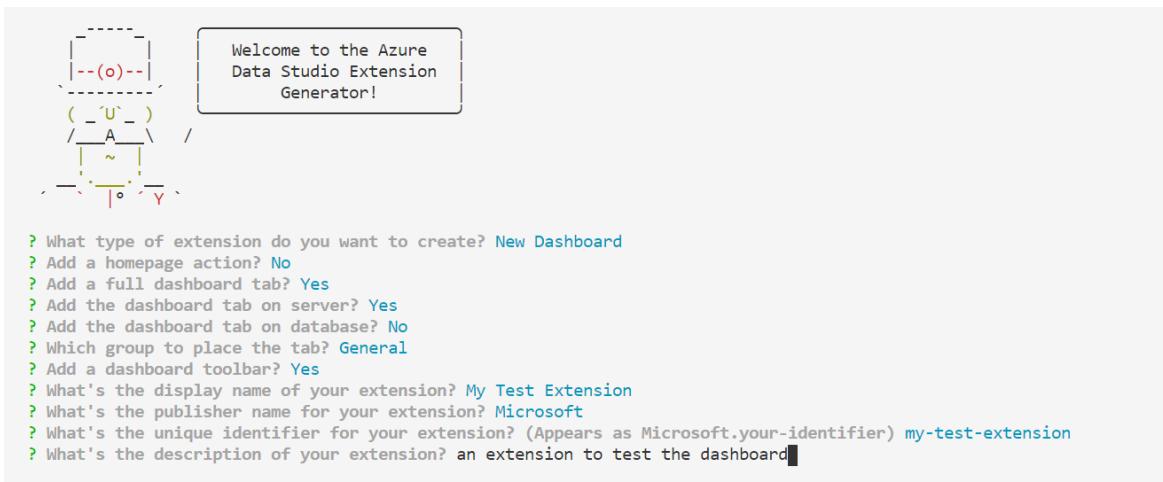
To create an extension:

- Start the extension generator with the following command:

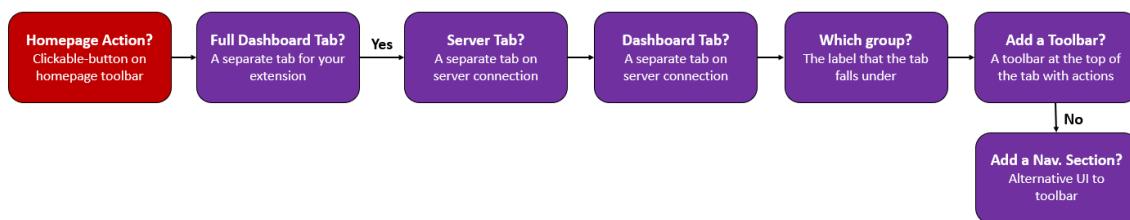
```
yo azuredatascudio
```

- Choose **New Dashboard** from the list of extension types.

3. Fill in the prompts, as shown, to create an extension that contributes a tab to the server dashboard.



There are many prompts, so here's a little more information on what each question means:



Completing the previous steps creates a new folder. Open the folder in Visual Studio Code, and you're ready to create your own dashboard extension.

### Run the extension

Let's see what the dashboard template gives us by running the extension. Before you run it, ensure that the **Azure Data Studio Debug extension** is installed in Visual Studio Code.

Select **F5** in Visual Studio Code to launch Azure Data Studio in Debug mode with the extension running. Then, you can see how this default template contributes to the dashboard.

Next, we look at how to modify this default dashboard.

### Develop the dashboard

The most important file to get started with extension development is `package.json`. This file is the manifest file, where the dashboard contributions are registered. Note the `dashboard.tabs`, `dashboard.insights`, and `dashboard.containers` sections.

Here are some changes to try out:

- Play around with the insights types, which include bar, horizontalBar, and timeSeries.
- Write your own queries to run against your SQL Server connection.
- See this [sample insight tutorial](#) or this [tutorial](#) for specific insight tutorials.

## Package your extension

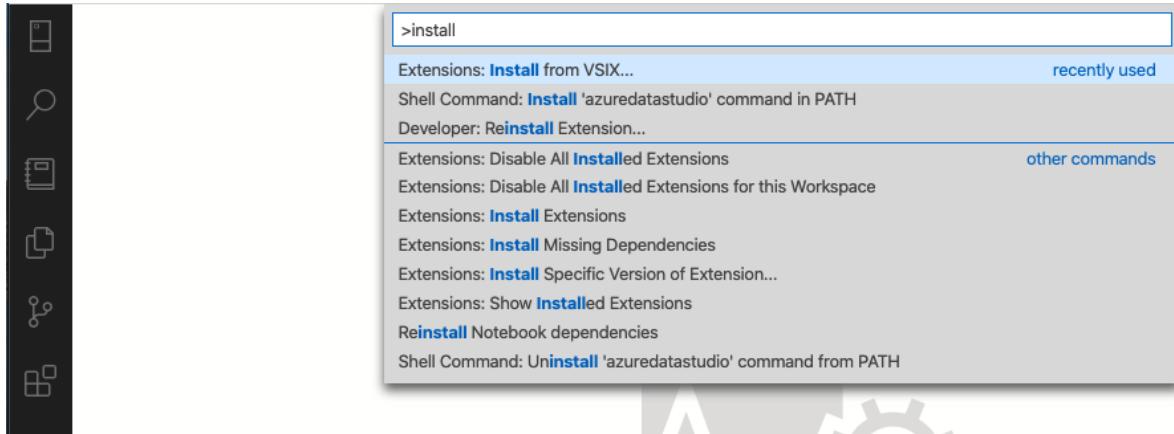
To share with others, you need to package the extension into a single file. Your extension can be published to the Azure Data Studio extension marketplace or shared with your team or community. To do this step, you need to install another npm package from the command line.

```
npm install -g vsce
```

Edit the `README.md` file to your liking. Then go to the base directory of the extension, and run `vsce package`. You can optionally link a repository with your extension or continue without one. To add one, add a similar line to your `package.json` file.

```
"repository": {  
    "type": "git",  
    "url": "https://github.com/anjalia/my-test-extension.git"  
}
```

After these lines are added, a `my-test-extension-0.0.1.vsix` file is created and ready to install in Azure Data Studio.



## Publish your extension to the marketplace

The Azure Data Studio extension marketplace is under construction. The current process is to host the extension VSIX somewhere, for example, on a GitHub release page. Then you submit a pull request that updates [this JSON file](#) with your extension information.

## Next steps

In this tutorial, you learned how to:

- Install the extension generator.
- Create your extension.
- Contribute to the dashboard in your extension.
- Test your extension.
- Package your extension.
- Publish your extension to the marketplace.

We hope that after reading this article you're inspired to build your own extension for Azure Data Studio. We have support for Dashboard Insights (attractive graphs that run against your SQL Server), a number of SQL-specific APIs, and a huge existing set of extension points inherited from Visual Studio Code.

If you have an idea but aren't sure how to get started, open an issue or tweet the team at [azuredatadstudio](#).

For more information, the [Visual Studio Code extension guide](#) covers all the existing APIs and patterns.

To learn how to work with T-SQL in Azure Data Studio, complete the T-SQL Editor tutorial:

[Use the Transact-SQL editor to create database objects](#)

# Create a Jupyter Notebook extension

9/26/2022 • 5 minutes to read • [Edit Online](#)

This tutorial demonstrates how to create a new Jupyter Notebook Azure Data Studio extension. The extension ships a sample Jupyter Notebook that can be opened and run in Azure Data Studio.

In this article you learn how to:

- Create an extension project.
- Install the extension generator.
- Create your notebook extension.
- Run your extension.
- Package your extension.
- Publish your extension to the marketplace.

## APIs used

- `azdata.nb.showNotebookDocument`

### Extension use cases

There are a few different reasons why you would create a notebook extension:

- Share interactive documentation
- Save and have constant access to that notebook
- Provide coding problems for users to follow along with
- Version and keep track of notebook updates

## Prerequisites

Azure Data Studio is built on the same framework as Visual Studio Code, so extensions for Azure Data Studio are built by using Visual Studio Code. To get started, you need the following components:

- [Node.js](#) installed and available in your `$PATH`. Node.js includes [npm](#), the Node.js Package Manager, which is used to install the extension generator.
- [Visual Studio Code](#) to debug the extension.
- Ensure `azuredatastudio` is in your path. For Windows, make sure you choose the **Add to Path** option in setup.exe. For Mac or Linux, run **Install 'azuredatastudio' command in PATH** from the Command Palette in Azure Data Studio.

## Install the extension generator

To simplify the process of creating extensions, we've built an [extension generator](#) using Yeoman. To install it, run the following command from the command prompt:

```
npm install -g yo generator-azuredatastudio
```

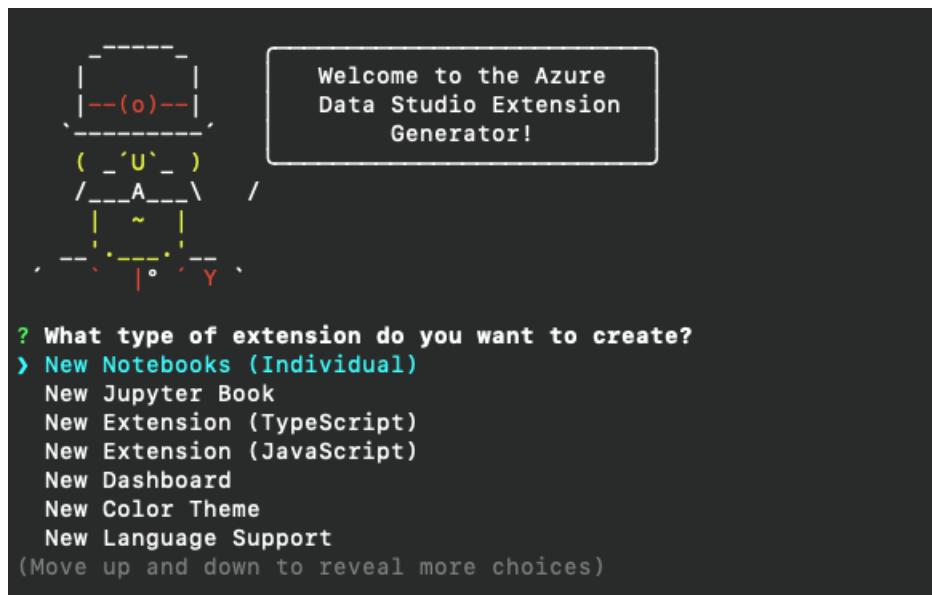
## Create your extension

To create an extension:

1. Start the extension generator with the following command:

```
yo azuredatastudio
```

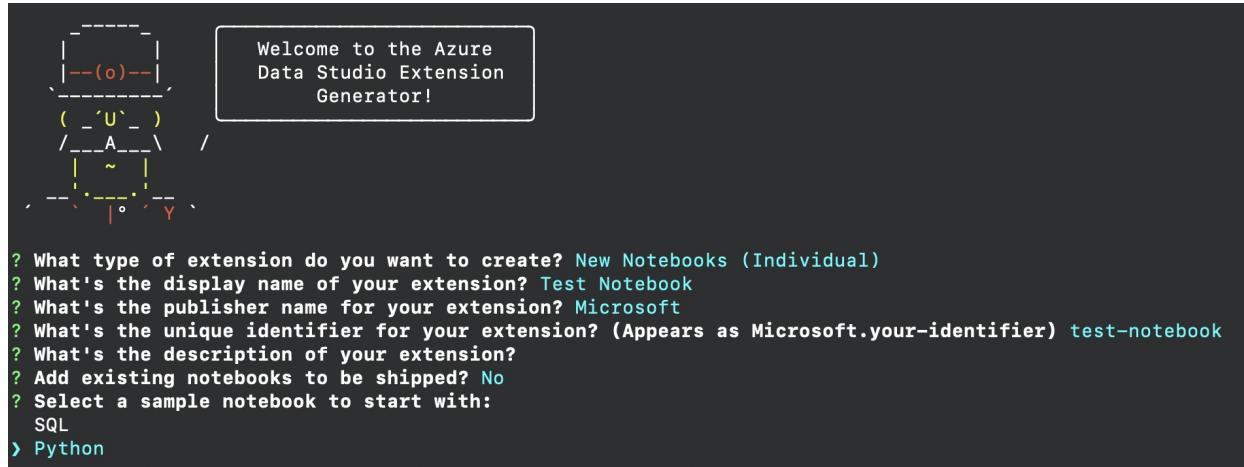
2. Choose **New Notebooks (Individual)** from the list of extension types.



3. Follow the steps to fill in the extension name. For this tutorial, use **Test Notebook**. Then fill in a publisher name. For this tutorial, use **Microsoft**. Finally, add a description.

Now, this is where some branching exists. You can either add Jupyter Notebooks that you've already created or you can use sample notebooks provided to you through the generator.

For this tutorial, we'll use a sample Python notebook:

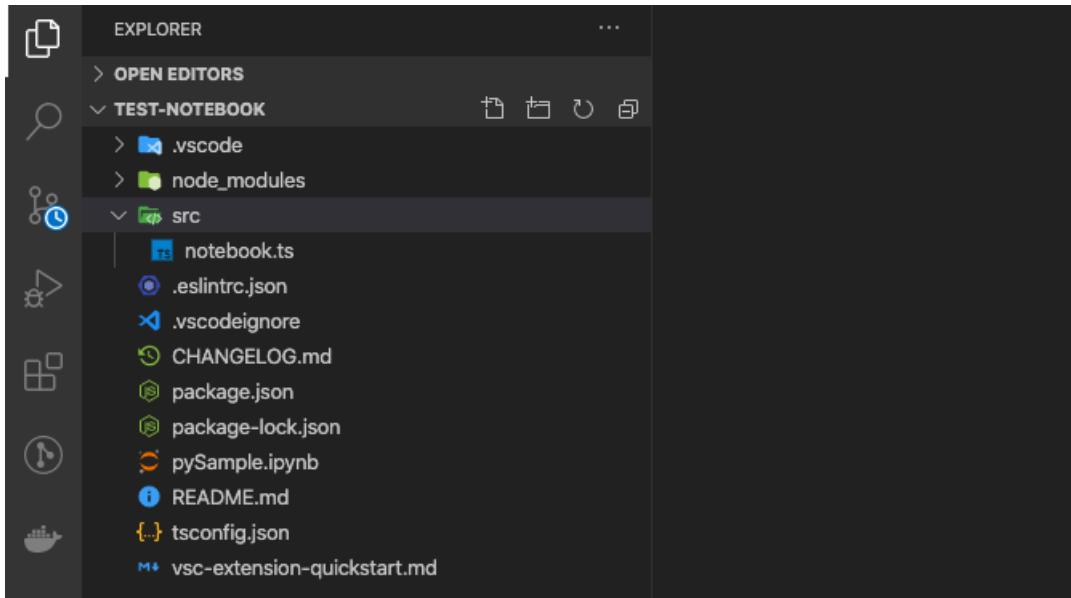


If you have notebooks that you're interested in shipping, answer that you have existing notebooks you want to ship. Provide the absolute file path where all of your notebooks or markdown files live.

Completing the previous steps creates a new folder with the sample notebook. Open the folder in Visual Studio Code, and you're ready to ship your new notebook extension.

## Understand your extension

This is what your project should currently look like:



The `vsc-extension-quickstart.md` file provides you with a reference of the important files. The `README.md` file is where you can provide documentation for your new extension. Note the `package.json`, `notebook.ts`, and `pySample.ipynb` files.

If there are any files or folders that you don't want to publish, you can include their names in the `.vscodeignore` file.

Let's take a look at `notebook.ts` to understand what our newly formed extension is doing.

```
// This function is called when you run the command `Launch Notebooks: Test Notebook` from the
// command palette in Azure Data Studio. If you want any additional functionality
// to occur when you launch the book, add it to the activate function.
export function activate(context: vscode.ExtensionContext) {
    context.subscriptions.push(vscode.commands.registerCommand('launchNotebooks.test-notebook', () => {
        let notebooksToDisplay: Array<string> = processNotebooks();
        notebooksToDisplay.forEach(name => {
            azdata.nb.showNotebookDocument(vscode.Uri.file(name));
        });
    }));
}

// Add other code here if you want to register another command.
}
```

This is the main function in `notebook.ts` that's called whenever we run our extension through the command **Launch Notebooks: Test Notebook**. We create our new command by using the `vscode.commands.registerCommand` API. The following definition inside the braces is the code that runs each time we call our command. For each notebook that's found from our `processNotebooks` function, we open it up in Azure Data Studio by using `azdata.nb.showNotebookDocument`.

The `package.json` file also plays an important role in registering our command **Launch Notebooks: Test Notebook**.

```
"activationEvents": [
  "onCommand:launchNotebooks.test-notebook"
],
"main": "./out/notebook.js",
"contributes": {
  "commands": [
    {
      "command": "launchNotebooks.test-notebook",
      "title": "Launch Notebooks: Test Notebook"
    }
  ]
}
```

We have an activation event for the command, and we've also added specific contribution points. These contribution points show up in the extension marketplace, where extensions are published, when users are looking at your extension. If you want to add more commands, be sure to add them to the `activationEvents` field. For more options, see [Activation events](#).

## Package your extension

To share with others, you need to package the extension into a single file. Your extension can be published to the Azure Data Studio extension marketplace or shared with your team or community. To do this step, you need to install another npm package from the command line.

```
npm install -g vsce
```

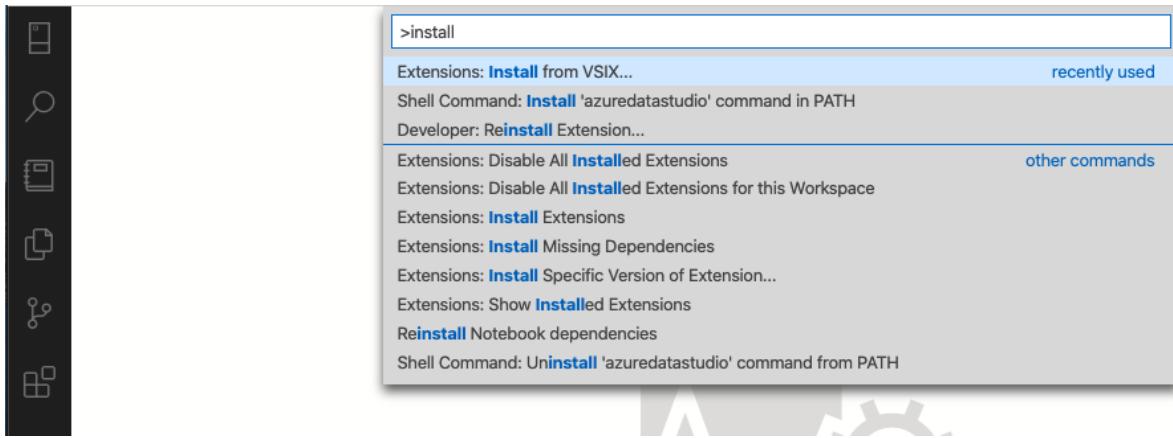
Edit the `README.md` file to your liking. Then go to the base directory of the extension, and run `vsce package`. You can optionally link a repository with your extension or continue without one. To add one, add a similar line to your `package.json` file.

```
"repository": {
  "type": "git",
  "url": "https://github.com/laurajjiang/testnotebook.git"
}
```

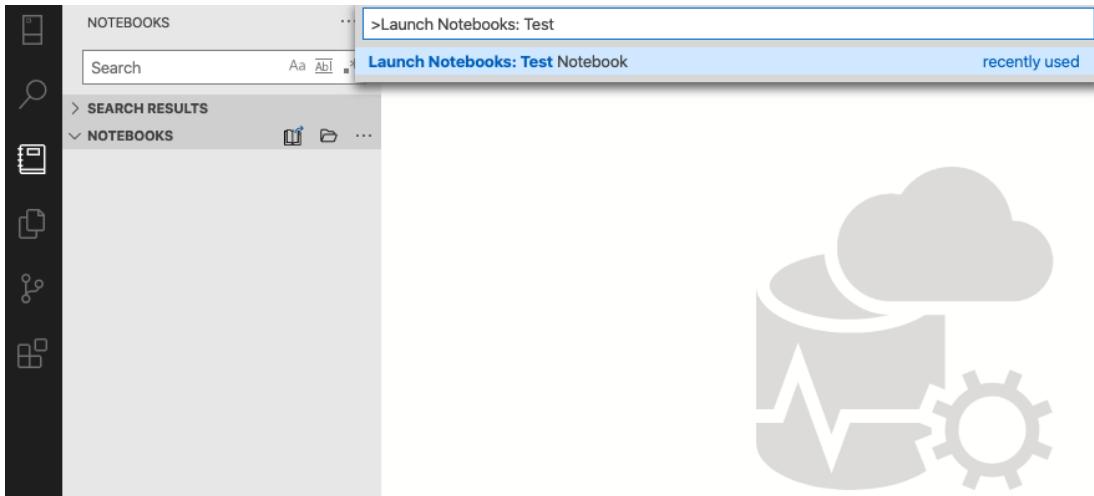
After these lines are added, a `my_notebook-0.0.1.vsix` file is created and ready to install and share with the world.

## Run your extension

To run and test your extension, open Azure Data Studio and open the command palette by selecting **Ctrl+Shift+P**. Find the command **Extensions: Install from VSIX**, and go to the folder that contains your new extension.



Your extension should now show up in your extension panel in Azure Data Studio. Open the command palette again, and you'll find the new command that we created with our extension **Launch Book: Test Book**. Upon running, it should open the Jupyter Book that we packaged with our extension.



Congratulations! You built and can now ship your first Jupyter Notebook extension.

## Publish your extension to the marketplace

The Azure Data Studio extension marketplace is under construction. To publish, host the extension VSIX somewhere, for example, on a GitHub release page. Then submit a pull request that updates [this JSON file](#) with your extension information.

## Next steps

In this tutorial, you learned how to:

- Create an extension project.
- Install the extension generator.
- Create your notebook extension.
- Create your extension.
- Package your extension.
- Publish your extension to the marketplace.

We hope that after reading this article you'll be inspired to build your own extension for Azure Data Studio.

If you have an idea but aren't sure how to get started, open an issue or tweet the team at [azuredatadstudio](#).

For more information, the [Visual Studio Code extension guide](#) covers all the existing APIs and patterns.

# Create a Jupyter Book extension

9/26/2022 • 6 minutes to read • [Edit Online](#)

This tutorial demonstrates how to create a new Jupyter Book Azure Data Studio extension. The extension ships a sample Jupyter Book that can be opened and run in Azure Data Studio.

In this article you learn how to:

- Create an extension project.
- Install the extension generator.
- Create your Jupyter Book extension.
- Run your extension.
- Package your extension.
- Publish your extension to the marketplace.

## APIs used

- `bookTreeView.openBook`

### Extension use cases

There are a few different reasons why you would create a Jupyter Book extension:

- Share organized and sectioned interactive documentation
- Share a full book (similar to an e-book but distributed through Azure Data Studio)
- Version and keep track of Jupyter Book updates

The main difference between a Jupyter Book and a notebook extension is that a Jupyter Book provides you with organization. Tens of notebooks can be split into different chapters in a Jupyter Book, but the notebook extension is intended to ship a small number of individual notebooks.

## Prerequisites

Azure Data Studio is built on the same framework as Visual Studio Code, so extensions for Azure Data Studio are built by using Visual Studio Code. To get started, you need the following components:

- [Node.js](#) installed and available in your `$PATH`. Node.js includes [npm](#), the Node.js Package Manager, which is used to install the extension generator.
- [Visual Studio Code](#) to make any changes to your extension and debug the extension.
- Ensure `azuredatascudio` is in your path. For Windows, make sure you choose the **Add to Path** option in setup.exe. For Mac or Linux, run **Install 'azuredatascudio' command in PATH** from the Command Palette in Azure Data Studio.

## Install the extension generator

To simplify the process of creating extensions, we've built an [extension generator](#) using Yeoman. To install it, run the following command from the command prompt:

```
npm install -g yo generator-azuredatascudio
```

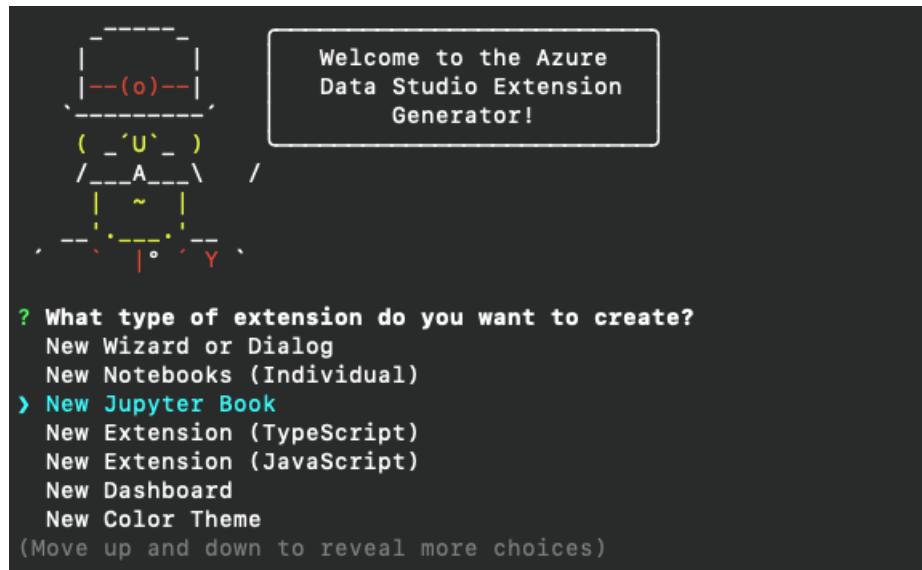
# Create your extension

To create an extension:

1. Start the extension generator with the following command:

```
yo azuredatastudio
```

2. Choose **New Jupyter Book** from the list of extension types.

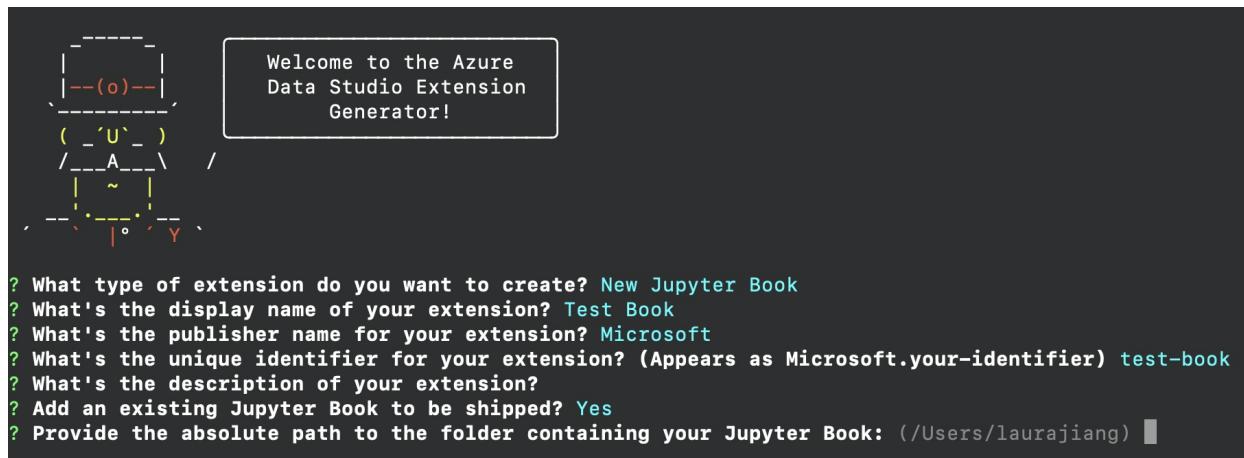


3. Follow the steps to fill in the extension name. For this tutorial, use **Test Book**. Then fill in a publisher name. For this tutorial, use **Microsoft**. Finally, add a description.

You select to either provide an existing Jupyter Book, use a provided sample book, or work to create a new Jupyter Book. All three options are shown.

## Provide an existing book

If you want to ship a book that you've already created, provide the absolute file path to the folder where your book contents live. You can then be ready to move on to learning about the extension and shipping it.



## Use the sample book

If you don't have an existing book or notebooks, you can use the provided sample in the generator.

```
  _--(o)--_
 /_-'U`_ \
 /__A__\ /
 | ~ |
 | .-.-. |
 , --| o | Y

Welcome to the Azure
Data Studio Extension
Generator!

? What type of extension do you want to create? New Jupyter Book
? What's the display name of your extension? Test Book
? What's the publisher name for your extension? Microsoft
? What's the unique identifier for your extension? (Appears as Microsoft.your-identifier) test-book
? What's the description of your extension?
? Add an existing Jupyter Book to be shipped? No
? Do you have existing notebooks you would like to create a Jupyter Book out of? No
```

The sample book demonstrates what a simple Jupyter Book looks like. If you want to learn about customizing a Jupyter Book, see the following section on how to create a new book with existing notebooks.

### Create a new book

If you have notebooks that you want to package into a Jupyter Book, you can. The generator asks if you want chapters in your book, and if so, how many and their titles. You can see what the selection process looks like here. Use the Spacebar to select which notebooks you want to place in each chapter.

```
  _--(o)--_
 /_-'U`_ \
 /__A__\ /
 | ~ |
 | .-.-. |
 , --| o | Y

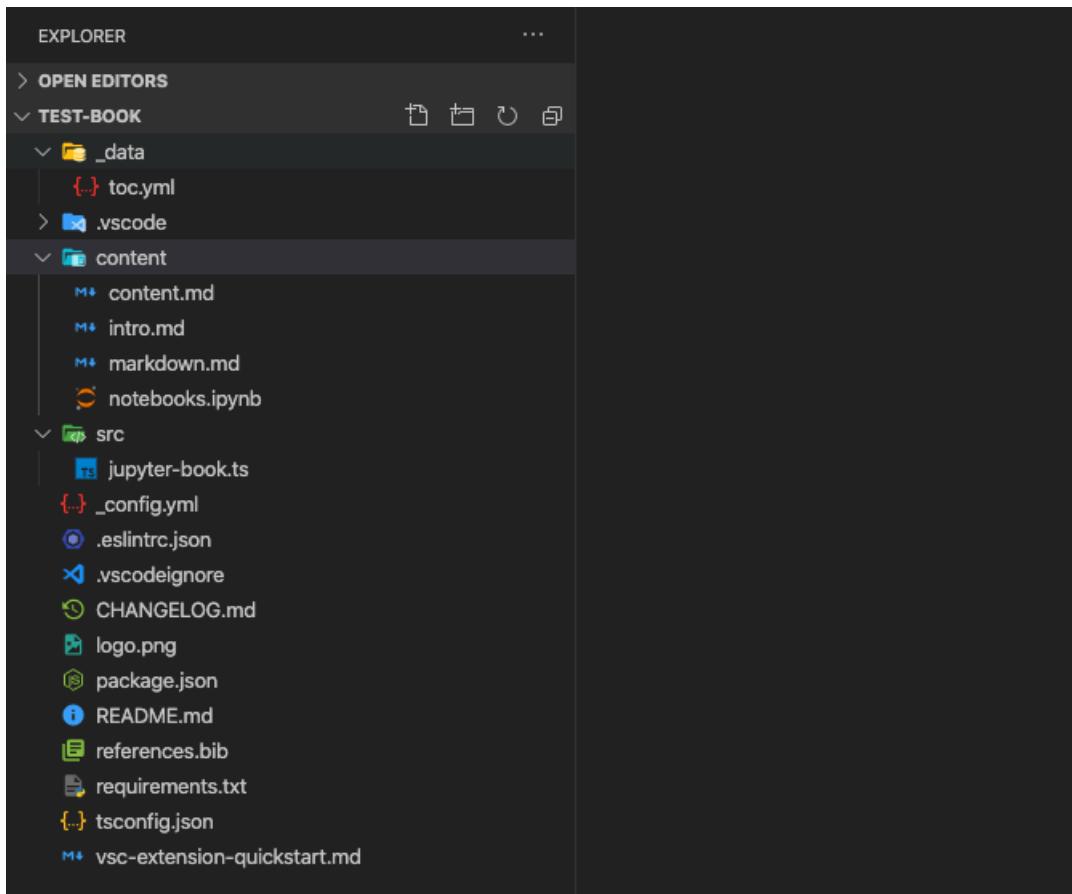
Welcome to the Azure
Data Studio Extension
Generator!

? What type of extension do you want to create? New Jupyter Book
? What's the display name of your extension? Test Book
? What's the publisher name for your extension? Microsoft
? What's the unique identifier for your extension? (Appears as Microsoft.your-id
entifier) test-book
? What's the description of your extension?
? Add an existing Jupyter Book to be shipped? No
? Do you have existing notebooks you would like to create a Jupyter Book out of? Yes
? Provide the absolute path to the folders where your notebooks currently exist. /users/laurajiang/desktop/notebooks
? Would you like more than one chapter in your book? Yes
11 notebook(s) found.
? How many chapters would you like in your book? 2
? List the name(s) of your chapter(s), separated by a comma for each new chapter. (e.g.: '1 - Chapter 1, 2 - Chapter 2') Chapter 1, Chapter 2
? Select notebooks for your chapter, Chapter 1: (Press <space> to select, <a> to toggle all, <i> to invert selection)
>o class01_ppi_R.ipynb
o class01_ppi_R_template.ipynb
o class01_ppi_bash.ipynb
o class01_ppi_python3.ipynb
o class01_ppi_python3_template.ipynb
o class02a_igraph_R.ipynb
o class02a_igraph_R_template.ipynb
(Move up and down to reveal more choices)
```

Completing the previous steps creates a new folder with your new Jupyter Book. Open the folder in Visual Studio Code, and you're ready to ship your Jupyter Book extension.

## Understand your extension

This is what your project should currently look like:



The `vsc-extension-quickstart.md` file provides you with a reference of the important files. The `README.md` file is where you can provide documentation for your new extension. Note the `package.json`, `jupyter-book.ts`, `content`, and `toc.yml` files. The `content` folder holds all notebook or markdown files. The `toc.yml` structures your Jupyter Book and is autogenerated if you opted to create a custom Jupyter Book through the extension generator.

If you created a book by using the generator and opted for chapters in your book, your folder structure would look a little different. Instead of your markdown and Jupyter Notebook files living in the `content` folder, there would be subfolders that correspond to the titles that you chose for your chapters.

If there are any files or folders that you don't want to publish, you can include their names in the `.vscodeignore` file.

Let's take a look at `jupyter-book.ts` to understand what our newly formed extension is doing.

```
// This function is called when you run the command `Launch Book: Test Book` from the
// command palette in Azure Data Studio. If you want any additional functionality
// to occur when you launch the book, add it to the activate function.
export function activate(context: vscode.ExtensionContext) {
    context.subscriptions.push(vscode.commands.registerCommand('launchBook.test-book', () => {
        processNotebooks();
    }));
}

// Add other code here if you want to register another command.
}
```

The `activate` function is the main action of your extension. Any commands that you want to register should appear inside the `activate` function, similar to our `launchBook.test-book` command. Inside the `processNotebooks` function, we find our extension folder that holds our Jupyter Book and call `bookTreeView.openBook` by using our extension's folder as a parameter.

The `package.json` file also plays an important role in registering our extension's command.

```
"activationEvents": [
  "onCommand:launchBook.test-book"
],
"main": "./out/notebook.js",
"contributes": {
  "commands": [
    {
      "command": "launchBook.test-book",
      "title": "Launch Book: Test Book"
    }
  ]
}
```

The activation event, `onCommand`, triggers the function that we registered when we invoke the command. There are a few other activation events that are possible for additional customization. For more information, see [Activation events](#).

## Package your extension

To share with others, you need to package the extension into a single file. Your extension can be published to the Azure Data Studio extension marketplace or shared with your team or community. To do this step, you need to install another npm package from the command line.

```
npm install -g vsce
```

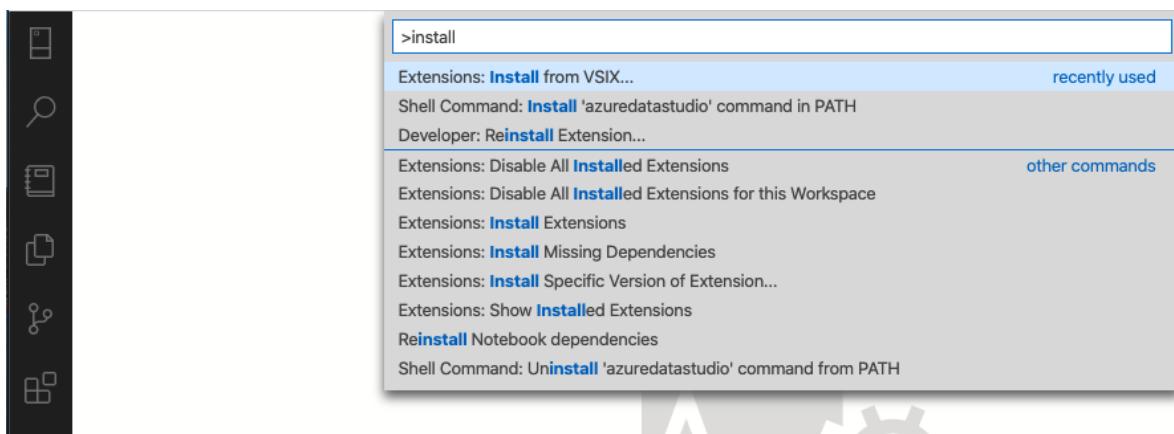
Edit the `README.md` file to your liking. Then go to the base directory of the extension, and run `vsce package`. You can optionally link a repository with your extension or continue without one. To add one, add a similar line to your `package.json` file.

```
"repository": {
  "type": "git",
  "url": "https://github.com/laurajjiang/testbook.git"
}
```

After these lines are added, a `my test-book-0.0.1.vsix` file is created and ready to install in Azure Data Studio.

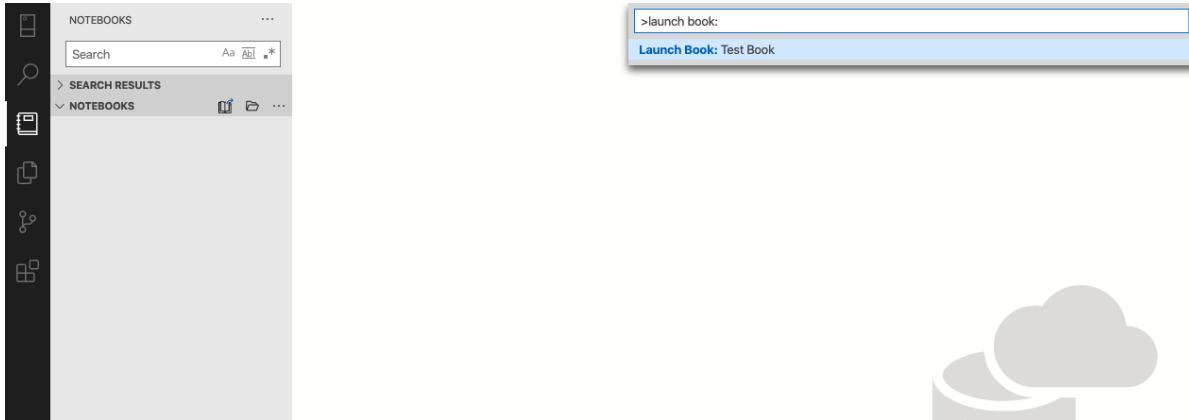
## Run your extension

To run and test your extension, open Azure Data Studio and open the command palette by selecting **Ctrl+Shift+P**. Find the command **Extensions: Install from VSIX**, and go to the folder that contains your new extension. It should now show up in your extension panel in Azure Data Studio.



Open the command palette again, and find the command that we registered, **Launch Book: Test Notebook**.

Upon running, it should open the Jupyter Book that we packaged with our extension.



Congratulations! You built and can now ship your first Jupyter Book extension. For more information on Jupyter Books, see [Books with Jupyter](#).

## Publish your extension to the Marketplace

The Azure Data Studio extension marketplace is under construction. To publish, host the extension VSIX somewhere, for example, on a GitHub release page. Submit a pull request that updates [this JSON file](#) with your extension information.

## Next steps

In this tutorial, you learned how to:

- Create an extension project.
- Install the extension generator.
- Create your Jupyter Book extension.
- Package your extension.
- Publish your extension to the marketplace.

We hope that after reading this article you'll have ideas on Jupyter Books that you'd like to share with the Azure Data Studio community.

If you have an idea but aren't sure how to get started, open an issue or tweet the team at [azuredatrstudio](#).

For more information, the [Visual Studio Code extension guide](#) covers all the existing APIs and patterns.

# Create an Azure Data Studio Wizard extension

9/26/2022 • 4 minutes to read • [Edit Online](#)

This tutorial demonstrates how to create a new **Azure Data Studio Wizard extension**. The extension contributes a wizard to interact with users in Azure Data Studio.

In this article you learn how to:

- Install the extension generator
- Create your extension
- Add a custom wizard to your extension
- Test your extension
- Package your extension
- Publish your extension to the marketplace

## Prerequisites

Azure Data Studio is built on the same framework as Visual Studio Code, so extensions for Azure Data Studio are built using Visual Studio Code. To get started, you need the following components:

- [Node.js](#) installed and available in your `$PATH`. Node.js includes [npm](#), the Node.js Package Manager, which is used to install the extension generator.
- [Visual Studio Code](#) to debug the extension.
- The Azure Data Studio [Debug extension](#) (optional). This lets you test your extension without needing to package and install it into Azure Data Studio.
- Ensure `azuredatascudio` is in your path. For Windows, make sure you choose the `Add to Path` option in `setup.exe`. For Mac or Linux, run `Install 'azuredatascudio' command in PATH` from the Command Palette in Azure Data Studio.

## Install the extension generator

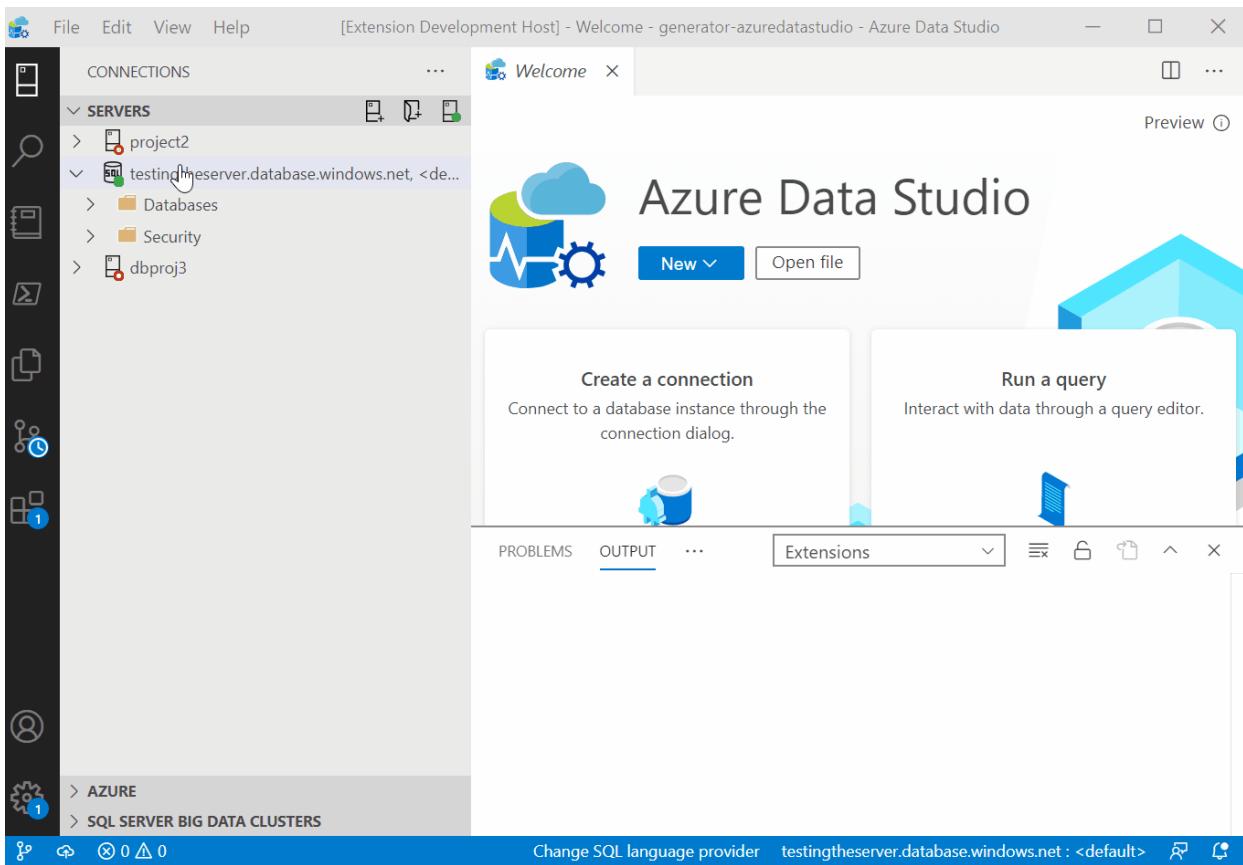
To simplify the process of creating extensions, we've built an [extension generator](#) using Yeoman. To install it, run the following from the command prompt:

```
npm install -g yo generator-azuredatascudio
```

## Create your wizard extension

### Introduction to wizards

Wizards are a user interface type that present step-by-step pages for users to fill in, in order to accomplish a task. Common examples include software setup wizards and troubleshooting wizards. For example:



Because wizards are often helpful when working with data and extending the functionality of Azure Data Studio, Azure Data Studio offers APIs to create your own custom wizards. We will be walking through how to generate a Wizard template and modify it to create your own custom wizard.

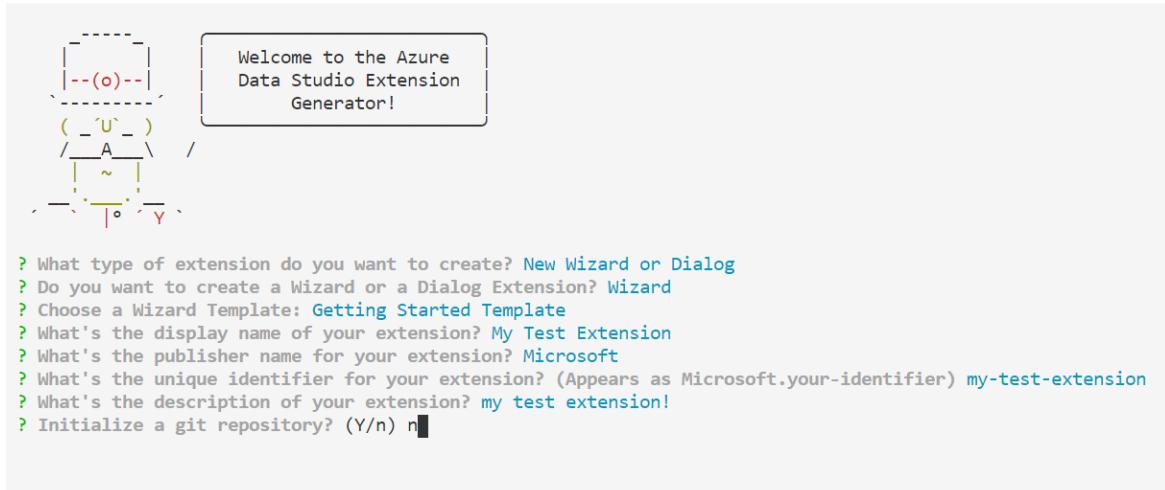
### Run the extension generator

To create an extension:

1. Launch the extension generator with the following command:

```
yo azuredatascudio
```

2. Choose **New Wizard or Dialog** from the list of extension types. Then select **Wizard**, followed by the **Getting Started Template**
3. Follow the steps to fill in the extension name (for this tutorial, use **My Test Extension**), and add a description.

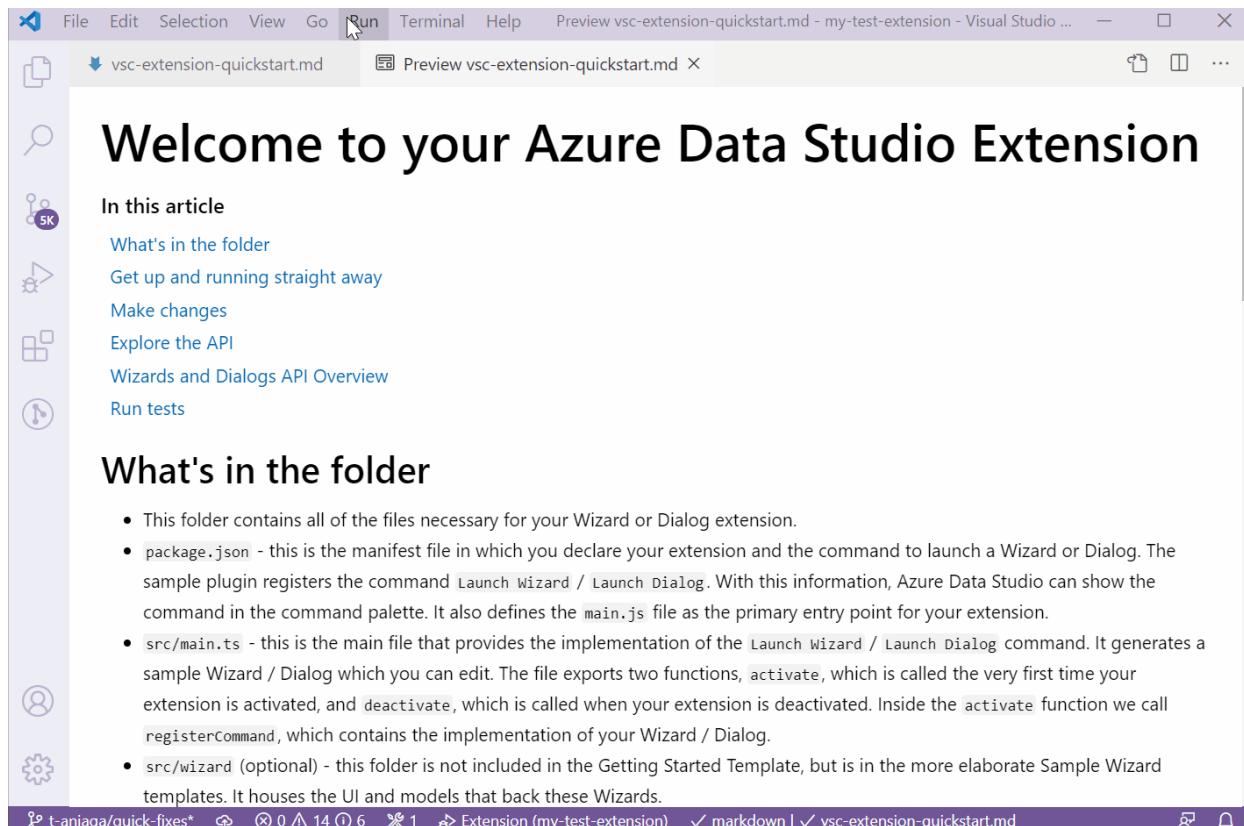


Completing the previous steps creates a new folder. Open the folder in Visual Studio Code and you're ready to create your own wizard extension!

## Run the extension

Let's see what the wizard template gives us by running the extension. Before running, ensure that the **Azure Data Studio Debug extension** is installed in Visual Studio Code.

Select F5 in VS Code to launch Azure Data Studio in debug mode with the extension running. Then, in Azure Data Studio, run the **Launch Wizard** command from the Command Palette (Ctr+Shift+P) in the new window. This will launch the default wizard that this extension contributes:



Next, we will look at how to modify this default wizard.

## Develop the wizard

The most important files to get started with extension development are `package.json`, `src/main.ts`, and `vsc-extension-quickstart.md`:

- `package.json`: This is the manifest file, where the **Launch Wizard** command is registered. This is also where `main.ts` is declared the main program entry point.
- `main.ts`: Contains the code to add UI elements to the Wizard, like pages, text, and buttons
- `vsc-extension-quickstart.md`: Contains technical documentation that may be a helpful reference when developing

Let's make a change to the wizard: we'll add a 4th, blank page. Modify `src/main.ts` as shown below, and you should see an additional page show up when you launch the wizard.

```

//----- Creating the Wizard -----//  

// The command has been defined in the package.json file
// This method defines what occurs when the command is entered in the ADS command palette
context.subscriptions.push(vscode.commands.registerCommand('my-test-extension.launchWizard', () => {
    // Create wizard
    let wizard : azdata.window.Wizard = azdata.window.createWizard('Sample Wizard');
    // Create wizard pages
    let page1 : azdata.window.WizardPage = azdata.window.createWizardPage('Sample Page 1');
    let page2 : azdata.window.WizardPage = azdata.window.createWizardPage('Sample Page 2');
    let page3 : azdata.window.WizardPage = azdata.window.createWizardPage('Sample Page 3');
    let page4 : azdata.window.WizardPage = azdata.window.createWizardPage('Sample Page 4');
    // Populate pages with content
    registerContentPage1(page1);
    registerContentPage2(page2);
    registerContentPage3(page3);

    page4.registerContent(async (view) => {
        // await view.initializeModel();
    });

    wizard.pages = [page1, page2, page3, page4]; // set wizard's content field to the wizard pages
    wizard.generateScriptButton.hidden = true;
    wizard.open(); // open wizard
}););

```

Once you are familiar with the template, here are some additional ideas to try:

- Add a button with a width of 300 to your new page
- Add a flex component to put your button in
- Add an action to your button. For example, when the button is clicked, launch a file-opening dialog or open a query editor.

## Package your extension

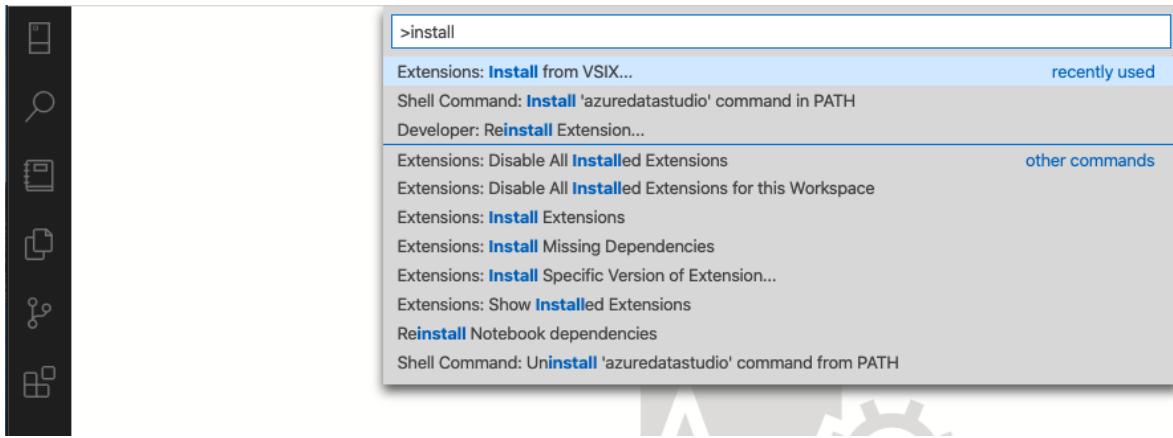
To share with others you need to package the extension into a single file. This can be published to the Azure Data Studio extension marketplace, or shared among your team or community. To do this, you need to install another npm package from the command line:

```
npm install -g vsce
```

Edit the `README.md` to your liking, then navigate to the base directory of the extension, and run `vsce package`. You can optionally link a repository with your extension or continue without one. To add one, add a similar line to your `package.json` file.

```
"repository": {
    "type": "git",
    "url": "https://github.com/anjalia/my-test-extension.git"
}
```

After these lines were added, a `my-test-extension-0.0.1.vsix` file was created and ready to install in Azure Data Studio.



## Publish your extension to the marketplace

The Azure Data Studio extension marketplace is not totally implemented yet, but the current process is to host the extension VSIX somewhere (for example, a GitHub Release page) then submit a PR updating [this JSON file](#) with your extension info.

## Next steps

In this tutorial, you learned how to:

- Install the extension generator
- Create your extension
- Add a custom wizard to your extension
- Test your extension
- Package your extension
- Publish your extension to the marketplace

We hope after reading this you'll be inspired to build your own extension for Azure Data Studio. We have support for Dashboard Insights (pretty graphs that run against your SQL Server), a number of SQL-specific APIs, and a huge existing set of extension points inherited from Visual Studio Code.

If you have an idea but are not sure how to get started, please open an issue or tweet at the team: [azuredatadstudio](#).

You can always refer to the [Visual Studio Code extension guide](#) because it covers all the existing APIs and patterns.

To learn how to work with T-SQL in Azure Data Studio, complete the T-SQL Editor tutorial:

[Use the Transact-SQL editor to create database objects](#)

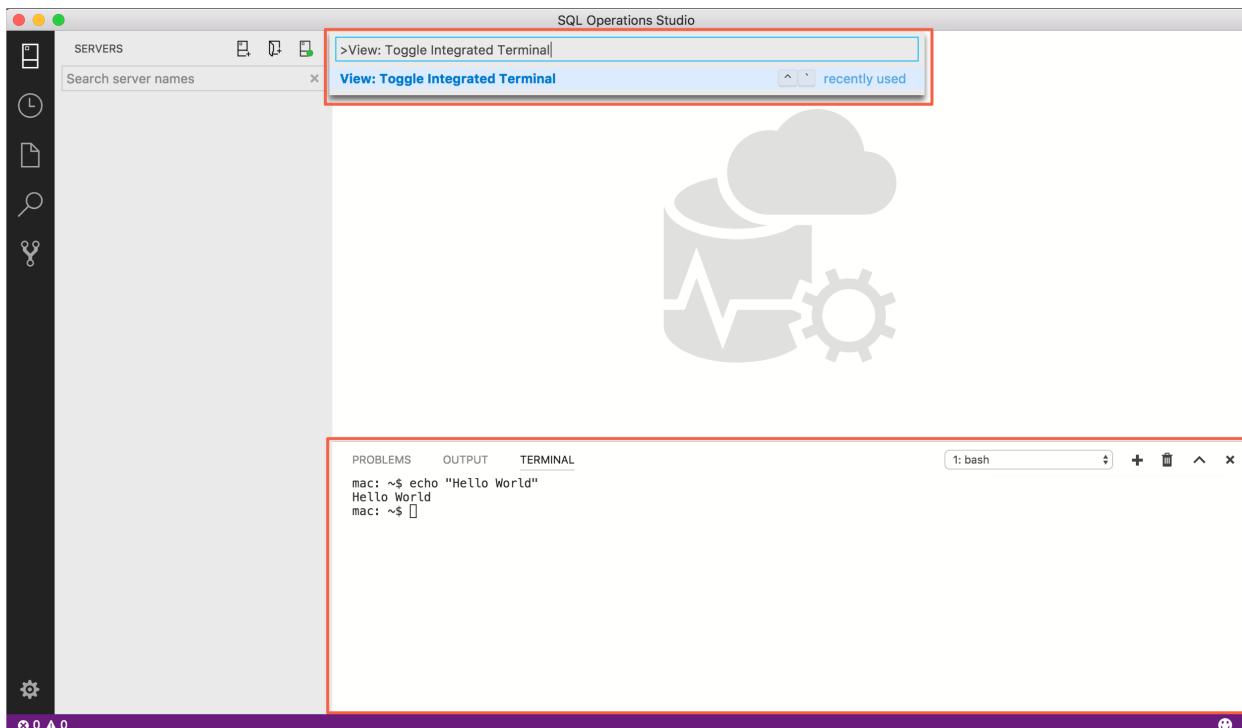
# Integrated Terminal

9/26/2022 • 4 minutes to read • [Edit Online](#)

In Azure Data Studio, you can open an integrated terminal, initially starting at the root of your workspace. This can be convenient as you don't have to switch windows or alter the state of an existing terminal to perform a quick command-line task.

To open the terminal:

- Use the **Ctrl+`** keyboard shortcut with the backtick character.
- Use the **View | Integrated Terminal** menu command.
- From the **Command Palette (Ctrl+Shift+P)**, use the **View:Toggle Integrated Terminal** command.

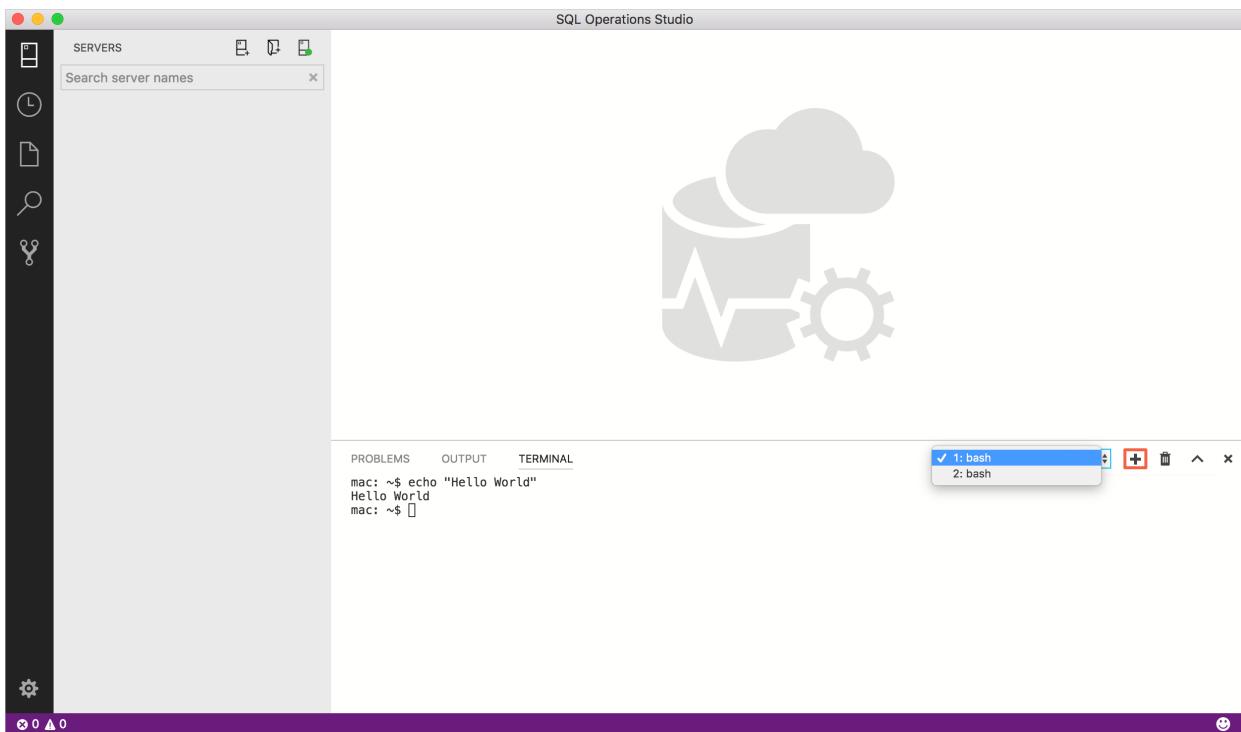


## NOTE

You can still open an external shell with the Explorer **Open in Command Prompt** command (**Open in Terminal** on Mac or Linux) if you prefer to work outside Azure Data Studio.

## Managing Multiple Terminals

You can create multiple terminals open to different locations and easily navigate between them. Terminal instances can be added by hitting the plus icon on the top-right of the **TERMINAL** panel or by triggering the **Ctrl+Shift+`** command. This creates another entry in the dropdown list that can be used to switch between them.



Remove terminal instances by pressing the trash can button.

#### TIP

If you use multiple terminals extensively, you can add key bindings for the `focusNext`, `focusPrevious` and `kill` commands outlined in the [Key Bindings section](#) to allow navigation between them using only the keyboard.

## Configuration

The shell used defaults to `$SHELL` on Linux and macOS, PowerShell on Windows 10 and Windows 11, and `cmd.exe` on earlier versions of Windows. These can be overridden manually by setting `terminal.integrated.shell.*` in [settings](#). Arguments can be passed to the terminal shell on Linux and macOS using the `terminal.integrated.shellArgs.*` settings.

### Windows

Correctly configuring your shell on Windows is a matter of locating the right executable and updating the setting. Below are a list of common shell executables and their default locations:

```
// 64-bit cmd if available, otherwise 32-bit
"terminal.integrated.shell.windows": "C:\\Windows\\sysnative\\cmd.exe"
// 64-bit PowerShell if available, otherwise 32-bit
"terminal.integrated.shell.windows": "C:\\Windows\\sysnative\\WindowsPowerShell\\v1.0\\powershell.exe"
// Git Bash
"terminal.integrated.shell.windows": "C:\\Program Files\\Git\\bin\\bash.exe"
// Bash on Ubuntu (on Windows)
"terminal.integrated.shell.windows": "C:\\Windows\\sysnative\\bash.exe"
```

#### NOTE

To be used as an integrated terminal, the shell executable must be a console application so that `stdin/stdout/stderr` can be redirected.

**TIP**

The integrated terminal shell is running with the permissions of Azure Data Studio. If you need to run a shell command with elevated (administrator) or different permissions, you can use platform utilities such as `runas.exe` within a terminal.

## Shell arguments

You can pass arguments to the shell when it is launched.

For example, to enable running bash as a login shell (which runs `.bash_profile`), pass in the `-l` argument (with double quotes):

```
// Linux
"terminal.integrated.shellArgs.linux": ["-l"]
```

## Terminal Display Settings

You can customize the integrated terminal font and line height with the following settings:

- `terminal.integrated.fontFamily`
- `terminal.integrated.fontSize`
- `terminal.integrated.lineHeight`

## Terminal Key Bindings

The **View: Toggle Integrated Terminal** command is bound to **Ctrl+`** to quickly toggle the integrated terminal panel in and out of view.

Below are the keyboard shortcuts to quickly navigate within the integrated terminal:

KEY	COMMAND
<b>Ctrl+`</b>	Show integrated terminal
<b>Ctrl+Shift+`</b>	Create new terminal
<b>Ctrl+Up</b>	Scroll up
<b>Ctrl+Down</b>	Scroll down
<b>Ctrl+PageUp</b>	Scroll page up
<b>Ctrl+PageDown</b>	Scroll page down
<b>Ctrl+Home</b>	Scroll to top
<b>Ctrl+End</b>	Scroll to bottom
<b>Ctrl+K</b>	Clear the terminal

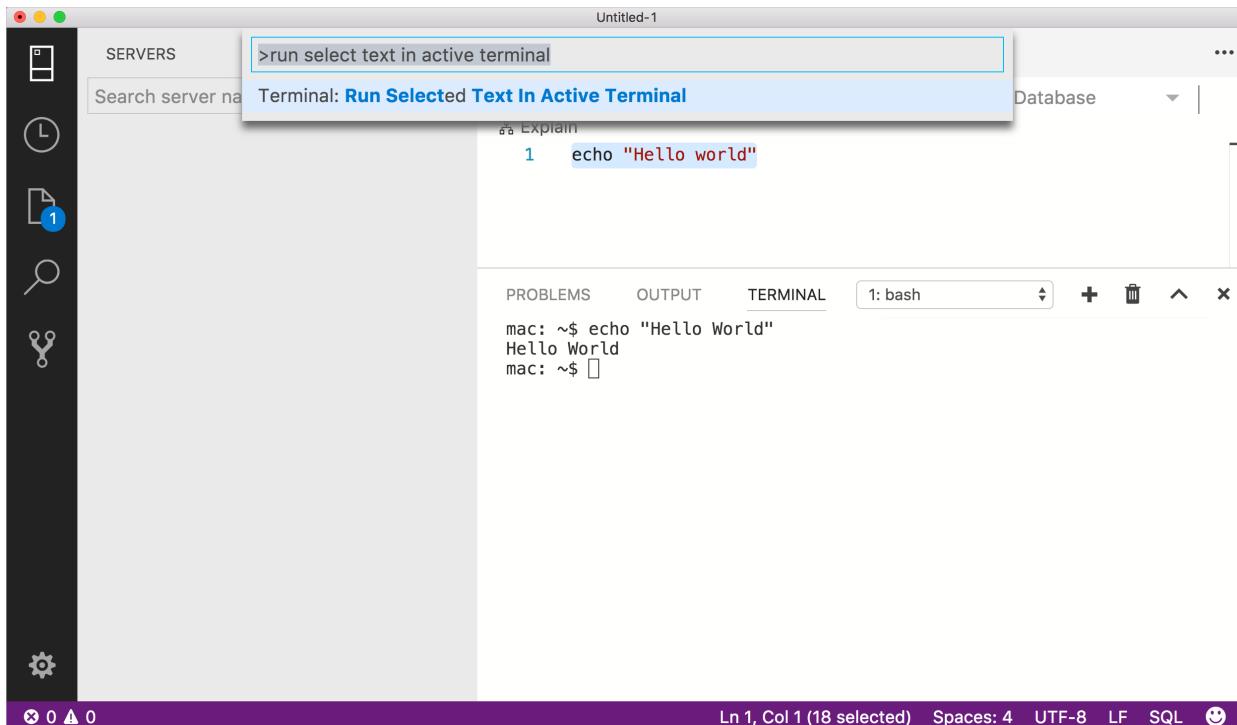
Other terminal commands are available and can be bound to your preferred keyboard shortcuts.

They are:

- `workbench.action.terminal.focus` : Focus the terminal. This is like toggle but focuses the terminal instead of hiding it, if it is visible.
- `workbench.action.terminal.focusNext` : Focuses the next terminal instance.
- `workbench.action.terminal.focusPrevious` : Focuses the previous terminal instance.
- `workbench.action.terminal.kill` : Remove the current terminal instance.
- `workbench.action.terminal.runSelectedText` : Run the selected text in the terminal instance.
- `workbench.action.terminal.runActiveFile` : Run the active file in the terminal instance.

## Run Selected Text

To use the `runSelectedText` command, select text in an editor and run the command **Terminal: Run Selected Text in Active Terminal** via the **Command Palette** (**Ctrl+Shift+P**). The terminal attempts to run the selected text:



If no text is selected in the active editor, the line that the cursor is on is run in the terminal.

## Copy & Paste

The keybindings for copy and paste follow platform standards:

- Linux: **Ctrl+Shift+C** and **Ctrl+Shift+V**
- Mac: **Cmd+C** and **Cmd+V**
- Windows: **Ctrl+C** and **Ctrl+V**

## Find

The Integrated Terminal has basic find functionality that can be triggered with **Ctrl+F**.

If you want **Ctrl+F** to go to the shell instead of launching the Find widget on Linux and Windows, you need to remove the keybinding like so:

```
{ "key": "ctrl+f", "command": "-workbench.action.terminal.focusFindWidget",
    "when": "terminalFocus" },
```

## Rename terminal sessions

Integrated Terminal sessions can now be renamed using the **Terminal: Rename** (

`workbench.action.terminal.rename`) command. The new name is displayed in the terminal selection drop-down.

### Forcing key bindings to pass through the terminal

While focus is in the integrated terminal, many key bindings won't work because the keystrokes are passed to and consumed by the terminal itself. The `terminal.integrated.commandsToSkipShell` setting can be used to get around this. It contains an array of command names whose key bindings skip processing by the shell and instead be processed by the Azure Data Studio key binding system. By default this includes all terminal key bindings in addition to a select few commonly used key bindings.

# Preview features in Azure Data Studio

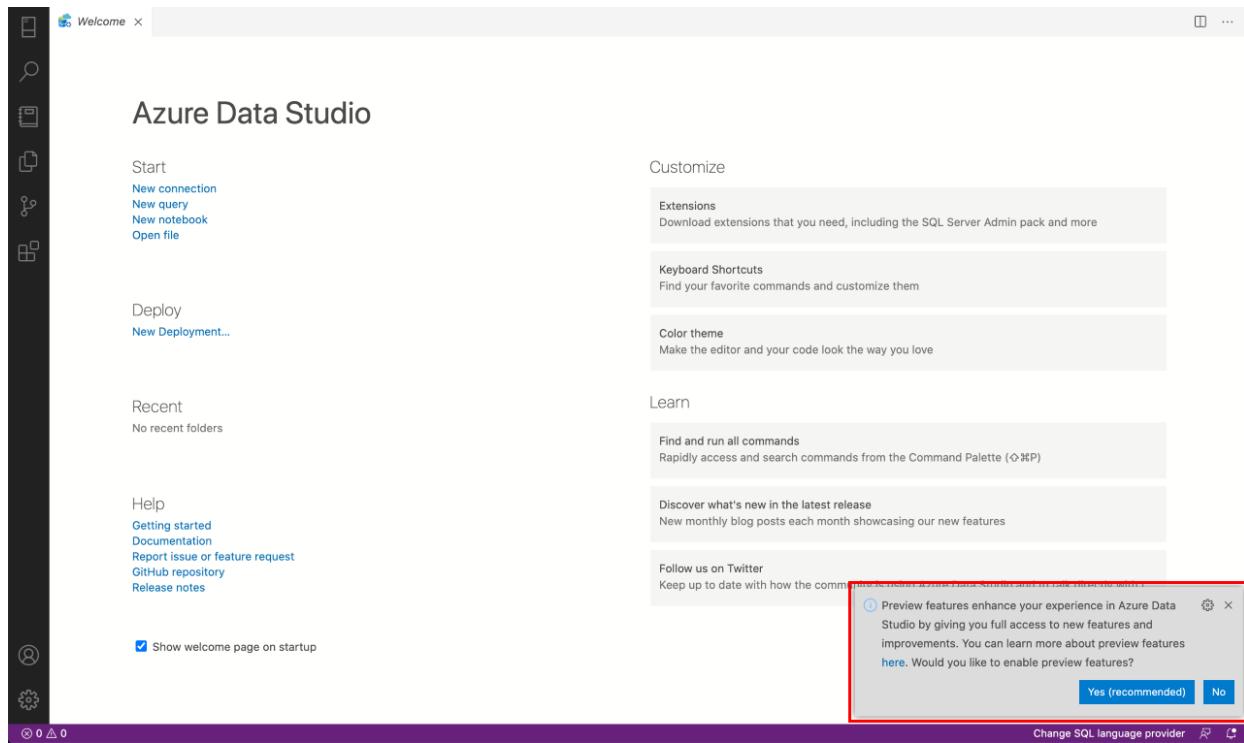
9/26/2022 • 2 minutes to read • [Edit Online](#)

In Azure Data Studio, new features and improvements are often first released as preview features before they're made generally available (GA). The amount of time a feature remains in preview can vary based on user feedback, quality checks, and long-term road maps. By enabling preview features, you get full access to Azure Data Studio features and the chance to provide early feedback.

## How do I enable preview features?

### On first launch

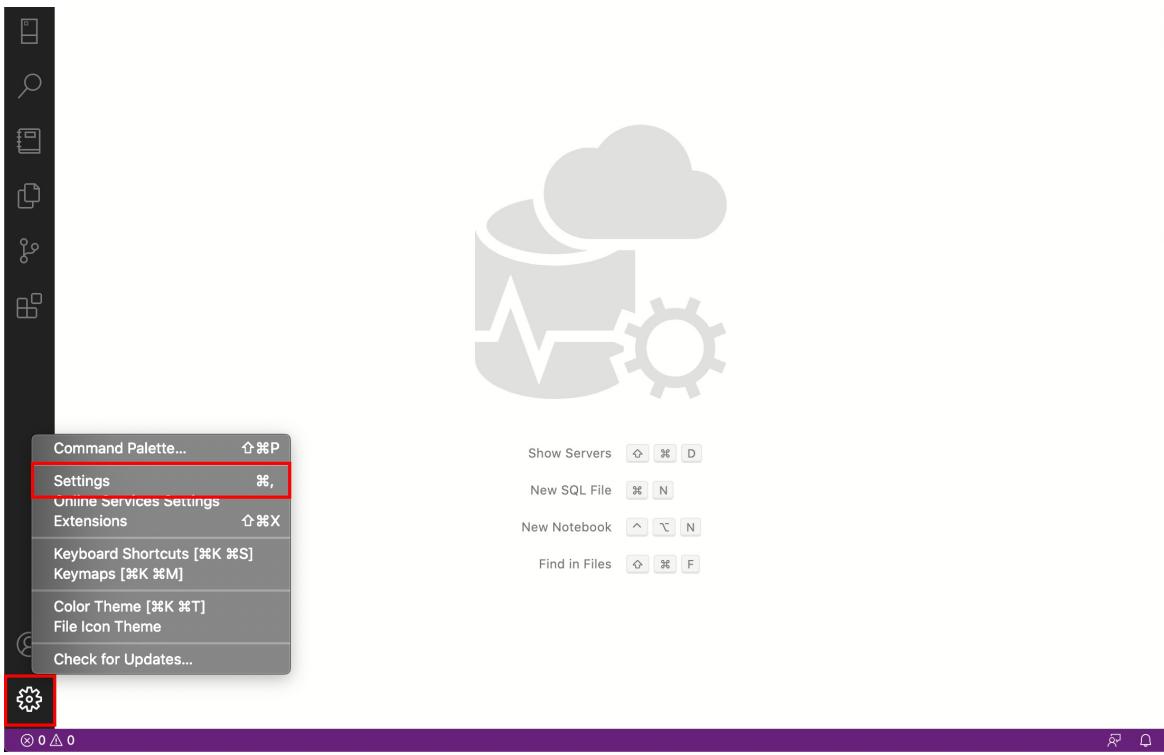
If you're a new user, you can opt into preview features when you launch Azure Data Studio for the first time. On startup, a toast notification will appear in the bottom-right corner of the screen that gives you the option to enable or disable preview features. Select **Yes (recommended)** to enable preview features.



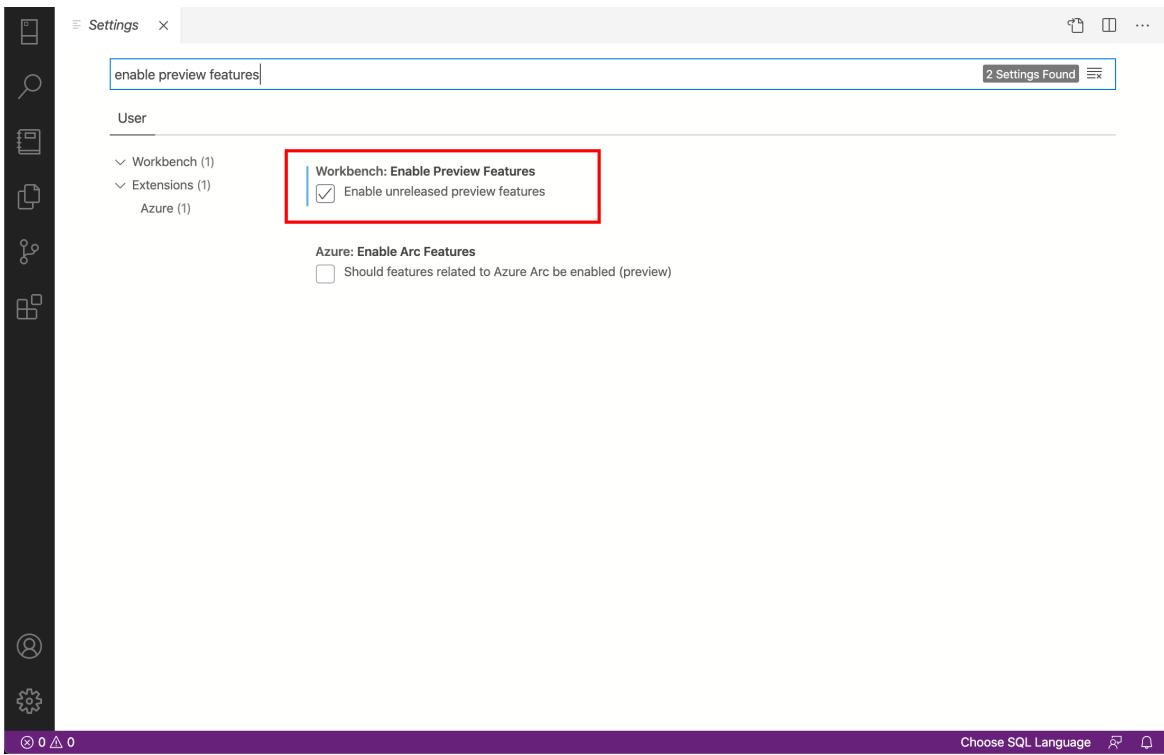
### In Settings

You can enable or disable preview features anytime in your Settings.

1. Select the Gear icon in the bottom-left corner and then select **Settings** from the context menu. The **Settings** tab will open.



2. Type "enable preview features" in the search bar.
3. To enable preview features, check the checkbox for **Enable unreleased preview features** under **Workbench: Enable Preview Features**. To disable preview features, clear the checkbox.



## List of preview features in Azure Data Studio

### General features in preview

- Azure portal integration
- Backup / Restore
- Deployments
  - SQL Edge
- Feature tour

## Notebook features in preview

- Dotnet interactive support

## First-party extensions in preview

- Azure Monitor Logs
- Azure SQL Edge Deployment
- Central Management Servers
- Cosmos
- Database Administration Tool Extensions for Windows
- Database Migration Assessment for Oracle
- Kusto
- Managed Instance Dashboard
- Oracle
- PostgreSQL
- Query History
- SandDance for Azure Data Studio
- Server Reports
- SQL Assessment
- SQL Database Projects
- SQL Server Agent
- SQL Server Profiler
- whoisactive

## Next steps

- [Azure Data Studio](#)

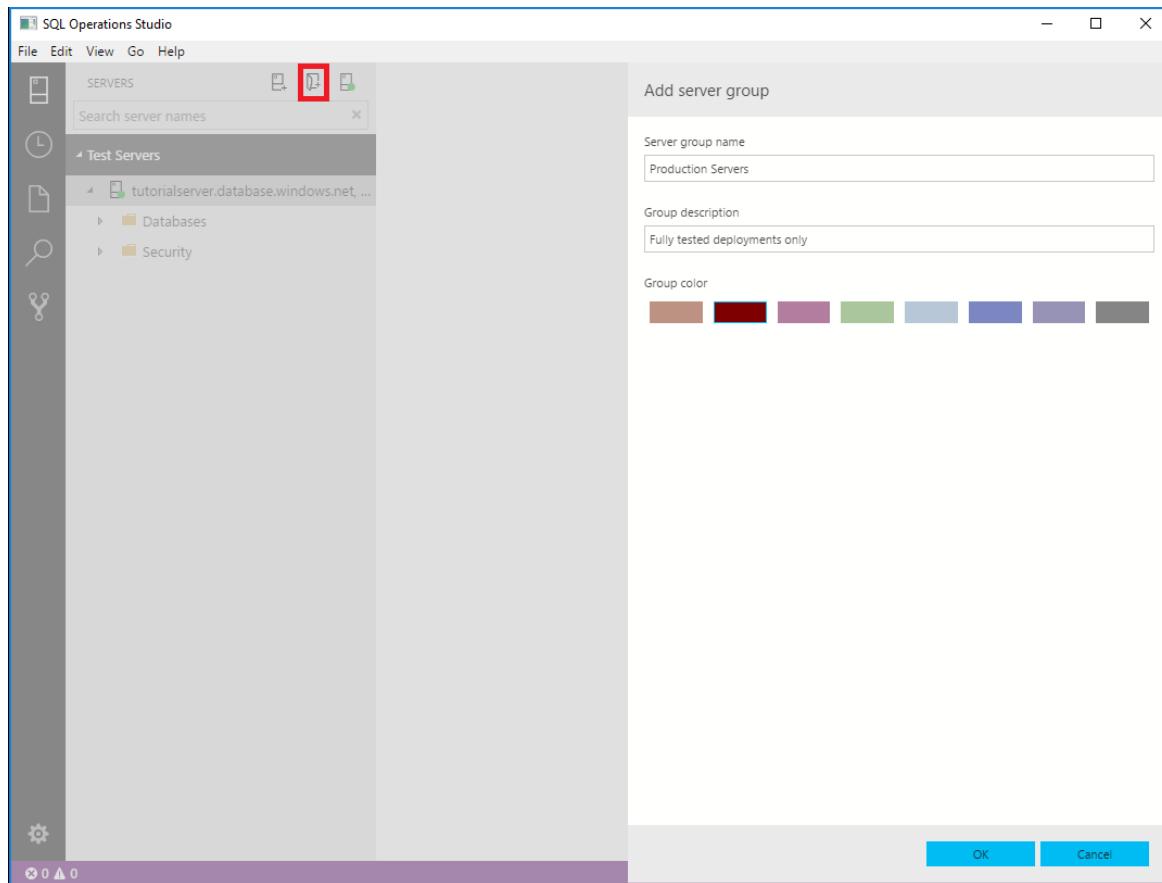
# Server groups in Azure Data Studio

9/26/2022 • 2 minutes to read • [Edit Online](#)

Server groups provide a way to organize your connections to the servers and databases you work with. When you create server groups, the configuration details are saved into *User Settings*.

## Create and edit server groups

1. Click **New Server Group** at the top of the *SERVERS* sidebar.
2. Enter a group name and select a color for the group. Optionally, add a description.



To edit an existing server group, right-click the group, and select **Edit Server Group**.

To edit available server group colors, edit the *Server Groups* values in [User Settings](#).

### TIP

You can drag and drop servers between different Server Groups.

## Additional resources

- [Workspace and User settings](#)

# Source control in Azure Data Studio

9/26/2022 • 2 minutes to read • [Edit Online](#)

Azure Data Studio supports Git for version/source control.

## Git support in Azure Data Studio

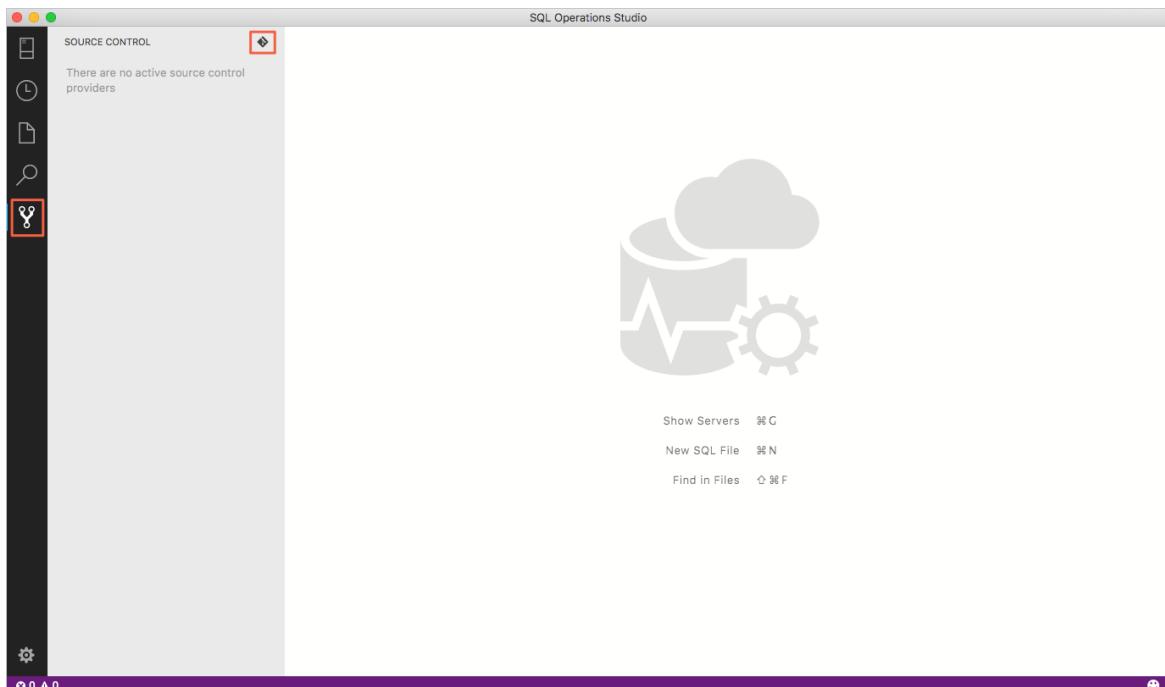
Azure Data Studio ships with a Git source control manager (SCM), but you still need to [install Git \(version 2.0.0 or later\)](#) before these features are available.

## Open an existing Git repository

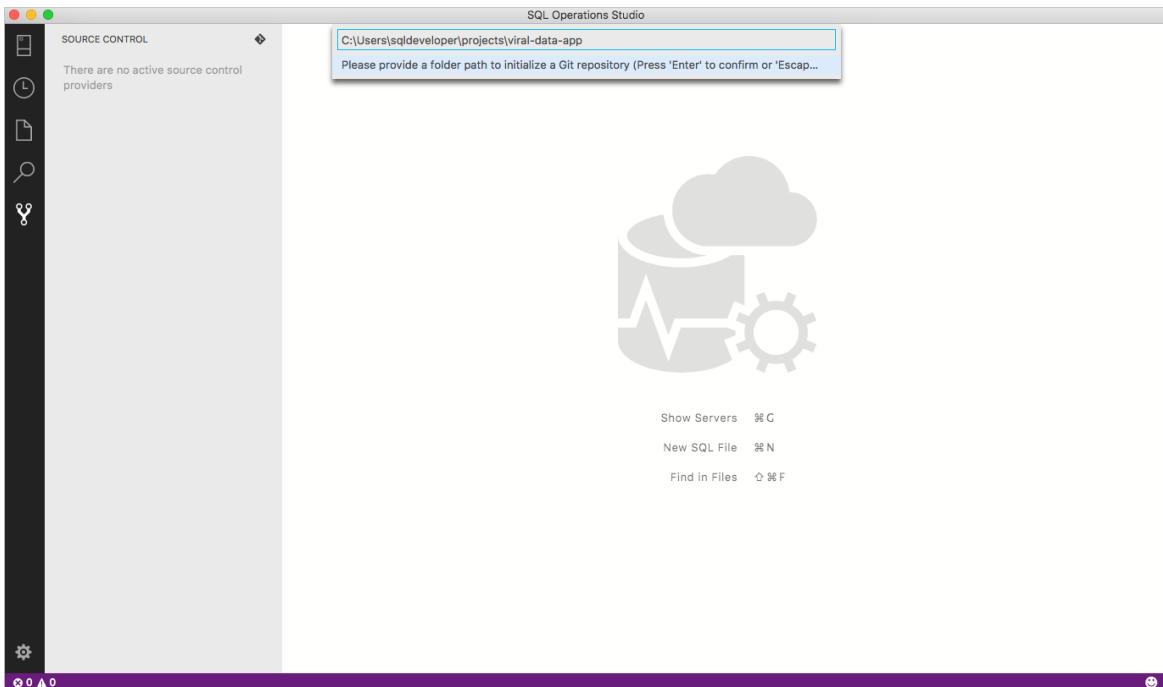
1. Under the **File** menu, select **Open Folder...**
2. Browse to the folder that contains your files tracked by git, and select **Select Folder**. Subfolders in your local repository are okay to select here.

## Initialize a new git repository

1. Select **Source Control**, then select the git icon.



2. Enter the path to the folder you want to initialize as a Git repository and press **Enter**.



## Working with Git repositories

Azure Data Studio inherits its Git implementation from VS Code, but doesn't currently support additional SCM providers. For the details about working with Git after you open or initialize a repository, see [Git support in VS Code](#).

## Additional resources

- [Git documentation](#)

# Table Designer in Azure Data Studio (Preview)

9/26/2022 • 10 minutes to read • [Edit Online](#)

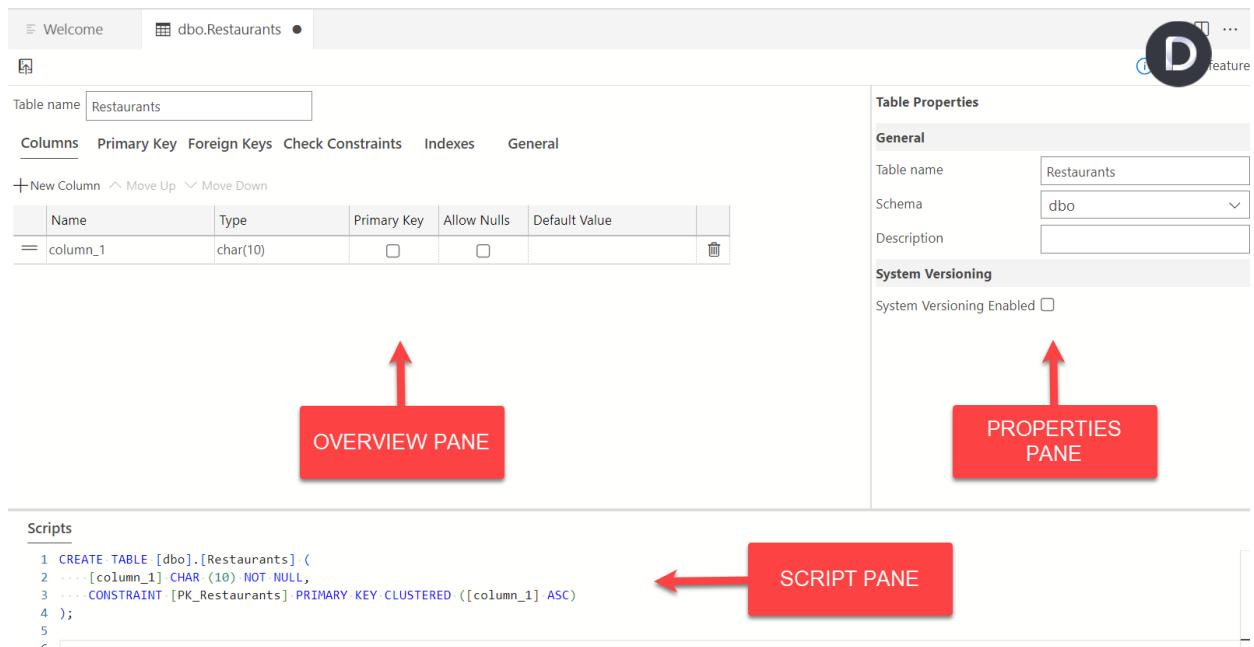
The Table Designer (preview) in Azure Data Studio provides a visual editor experience alongside the Transact-SQL Editor for creating and editing table structure, including table-specific programming objects, for SQL Server databases.

## Why Table Designer?

The Table Designer in Azure Data Studio provides users an efficient way to configure and manage tables, using primary and foreign keys, indices, constraints, triggers, etc. directly on the graphical user interface (GUI).

## Overview of Table Designer

The Table Designer consists of a window split into three separate panes. The first pane, is the Overview/General pane of the table design. This consists of tabs for the columns, primary and foreign keys, constraint checks, indexes, and a general tab. The second pane is used for defining the properties of your table. Lastly, the third pane is the Script pane for the read-only T-SQL script that represents actions performed on the GUI as well as query results. The size of these panes can be adjusted by clicking on the edge lines and mouse dragging with the mouse to resize.



## Creating and editing tables

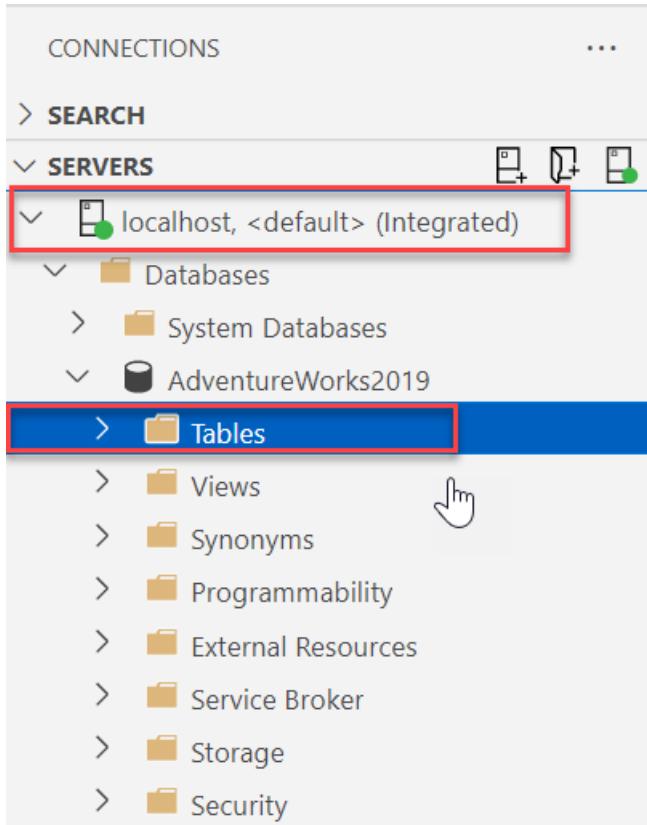
### Creating a table

#### NOTE

We will be using the "AdventureWorks2019" sample database throughout this tutorial. If you haven't already, please refer to [AdventureWorks sample databases](#) to download this sample database.

Before creating a table, you need to ensure that you have an active connection where your database is located. For a tutorial on connecting to the SQL Server, check out the [Use Azure Data Studio to connect and query SQL Server tutorial](#) article. To locate the Tables folder, click the drop-down menu of the databases folder in your active local connection and locate the AdventureWorks2019 database. Then, click the drop-down of this

database to view the Tables folder.



#### NOTE

That your server needs to be connected. This is indicated by the green dot at the bottom right of the icon in the server connection. If red (this means the connection is inactive or disconnected), left select once on the connection and this will spin up your connection.

To create a table, right-click the Tables folder in the connection pane and select "New Table". This opens up the Table Designer view seen below.

A default table is created with the default name of "NewTable", which has one column by default. You can edit the name of your table and add new columns to it. Let us update our table name to "Cities". We do this by selecting inside this name field, typing the new table name (Cities) and pressing "Enter" to save the change. Lets add a new column by selecting the "New Column" button above the columns grid. In the Columns grid, select the name value "column 1", replace with "CityName", and select the Primary Key column checkbox to make this the primary key for the table. To read more on Primary Keys, [see this SQL Server documentation on Primary Keys](#).

Select the "column 2", replace with "Population"

The screenshot shows the 'dbo.Cities' table definition. The 'CityName' column is highlighted with a red box, and the 'Primary Key' checkbox in its row is also highlighted with a red box, indicating it is selected.

Name	Type	Primary Key	Allow Nulls	Default Value
CityName	int	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Delete"/>
Population	int	<input type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Delete"/>

Now that we have created a new table, we need to save this change to our AdventureWorks2019 database. To do this, select the "Publish" icon as seen below. Publishing can also be done by using the "Ctrl/Cmd + S" command.

The screenshot shows the 'dbo.Cities' table properties window. The 'Publish' icon (a blue square with a white arrow) is highlighted with a red box.

Once this icon is selected, a dialog box appears for the user to either publish the changes directly to the AdventureWorks2019 database via the graphical user interface or generate the SQL command which can then be manually executed.

The screenshot shows the 'Preview Database Updates' dialog box. It includes sections for 'Highlights', 'Tables that will be rebuilt', 'User actions', 'Table rebuild', 'Supporting actions', and buttons for 'Update Database', 'Generate Script', and 'Cancel'. The 'Update Database' and 'Generate Script' buttons are highlighted with red boxes.

**Highlights**

**Tables that will be rebuilt**

- [dbo].[Cities]

**User actions**

**Table rebuild**

- [dbo].[Cities] (SqlTable)

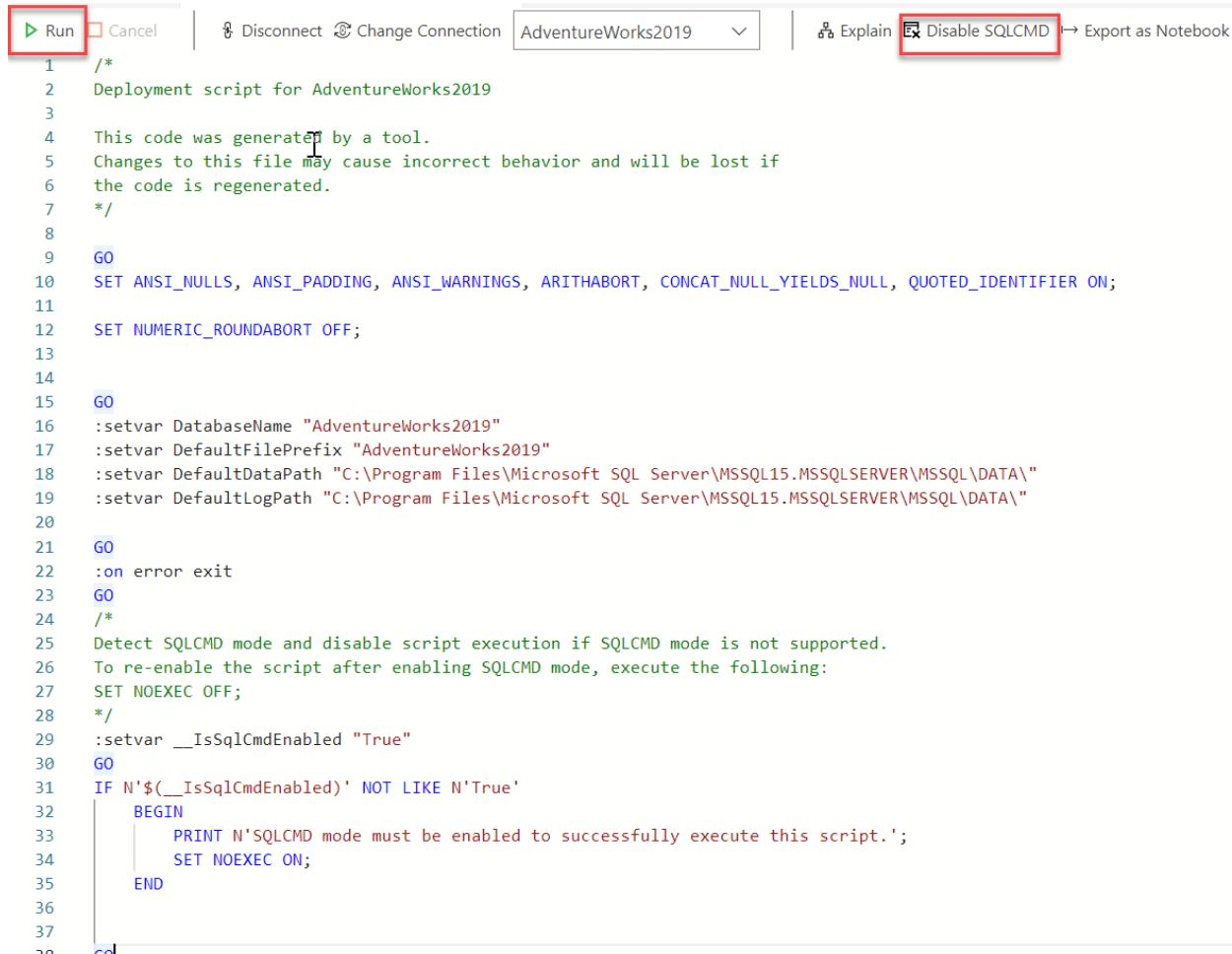
**Supporting actions**

- None

**Buttons:**

- Update Database
- Generate Script
- Cancel

If you choose to manually run the SQL command to update this database, you need to select "Run" to perform this action. See below for this illustration.



```
1  /*
2  Deployment script for AdventureWorks2019
3
4  This code was generated by a tool.
5  Changes to this file may cause incorrect behavior and will be lost if
6  the code is regenerated.
7 */
8
9  GO
10 SET ANSI_NULLS, ANSI_PADDING, ANSI_WARNINGS, ARITHABORT, CONCAT_NULL_YIELDS_NULL, QUOTED_IDENTIFIER ON;
11
12 SET NUMERIC_ROUNDABORT OFF;
13
14
15 GO
16 :setvar DatabaseName "AdventureWorks2019"
17 :setvar DefaultFilePrefix "AdventureWorks2019"
18 :setvar DefaultDataPath "C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\
19 :setvar DefaultLogPath "C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\
20
21 GO
22 :on error exit
23 GO
24 /*
25 Detect SQLCMD mode and disable script execution if SQLCMD mode is not supported.
26 To re-enable the script after enabling SQLCMD mode, execute the following:
27 SET NOEXEC OFF;
28 */
29 :setvar __IsSqlCmdEnabled "True"
30 GO
31 IF N'$(__IsSqlCmdEnabled)' NOT LIKE N'True'
32 BEGIN
33     PRINT N'SQLCMD mode must be enabled to successfully execute this script.';
34     SET NOEXEC ON;
35 END
36
37
38 GO
```

**Messages**

```
11:27:59 AM Started executing query at Line 1
Commands completed successfully.
11:27:59 AM Started executing query at Line 10
Commands completed successfully.
11:27:59 AM Started executing query at Line 24
Commands completed successfully.
11:27:59 AM Started executing query at Line 31
Commands completed successfully.
11:27:59 AM Started executing query at Line 39
Commands completed successfully.
11:27:59 AM Started executing query at Line 43
Update complete.
Total execution time: 00:00:00.001
```

You see that this script was ran successfully as described in the "Messages" pane.

#### NOTE

Please note that if you decide to use the "Generate Script" option, you need to enable the SQL command mode in order to prevent errors in the SQL command script. Do this by clicking the "Enable SQLCMD" icon. In the image above, the "Enable SQLCMD" option is already selected, which is why you see "Disable SQLCMD".

You can save this SQLQuery as a file on your local desktop by using the "Ctrl/Cmd + S" command. Any unsaved work is indicated by the dark shaded dot in the tab of the workspace that you have open. First image below shows work that isn't saved as indicated by the black dot.

The screenshot shows the Table Designer interface. At the top, there's a navigation bar with 'Welcome' and a highlighted item 'dbo.Cities'. Below it is a table name input field containing 'Cities'. A menu bar with tabs like 'Columns', 'Primary Key', 'Foreign Keys', etc., is visible. A 'New Column' button is present. The main area shows a grid for defining columns:

Name	Type	Primary Key	Allow Nulls	Default Value
CityName	nvarchar(50)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<span style="color: red;">●</span>
Population	int	<input type="checkbox"/>	<input type="checkbox"/>	<span style="color: red;">●</span>

This image below shows work that has been saved. Notice black dot disappeared.

The screenshot shows the Table Designer interface after saving changes. The 'Primary Key' column for the 'CityName' field is no longer selected, indicated by the absence of a black dot in the 'Primary Key' column header. The rest of the table definition remains the same.

To confirm that your table has been created, traverse your AdventureWorks2019 database in the object explorer under "Tables" to locate your "Cities" table. Newly created tables are seen by order of how recently they were created, so you should see yours near top of the list. If you don't see your table, right click on the "Tables" folder and select "Refresh".

The screenshot shows the Object Explorer pane. The 'TutorialDB' database is selected. Under the 'Tables' node, a context menu is open, with the 'Refresh' option highlighted.

#### Editing an existing table

To edit an existing table in the Table Designer, right-click on the table in the Connections pane and select "Design" from the menu. This opens up the Table Designer view, which then allows you to make edits where necessary. See below.

The screenshot shows the SSMS interface with the 'TutorialDB' database selected. In the left sidebar, 'Tables' is expanded, and 'dbo.Cities' is selected. A context menu is open over the second column of the table design grid, with the 'Design' option highlighted by a red box.

Name	Type	Primary Key	Allow Nulls	Default Value
\$edge_id	nvarchar(1000)	<input type="checkbox"/>	<input type="checkbox"/>	
\$from_id	nvarchar(1000)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Select Top 1000	nvarchar(1000)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
Edit Data	nvarchar(50)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

### Changing column properties

You can change column properties in the main pane as shown above or in the properties pane. To view the properties pane for a column, select the column, for example "City Name" as shown below, the properties pane will then show the properties specific to columns.

The screenshot shows the SSMS interface with the 'TutorialDB' database selected. In the left sidebar, 'Tables' is expanded, and 'dbo.Cities' is selected. The 'Properties' pane on the right is open for the 'CityName' column, with a red box highlighting it. The 'General' section shows the column properties: Name (CityName), Type (nvarchar), Length (50), Primary Key (checked), and Allow Nulls (unchecked). The 'Script' pane at the bottom contains the T-SQL code for creating the table.

Name	Type	Primary Key	Allow Nulls	Default Value
CityName	nvarchar(50)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Population	int	<input type="checkbox"/>	<input type="checkbox"/>	

```

CREATE TABLE [dbo].[Cities] (
    [CityName] NVARCHAR (50) NOT NULL,
    [Population] INT NOT NULL,
    CONSTRAINT [PK_Cities] PRIMARY KEY CLUSTERED ([CityName] ASC)
);

```

The script pane will update to display the read-only Transact-SQL version of your table for reference (as shown in the image above). After making changes, select the "Publishing Changes..." button or "CTRL/CMD + S"

### Deleting columns

To delete a column, simply select the trash icon at the end of the column as highlighted below.

Table name

[Columns](#) [Primary Key](#) [Foreign Keys](#) [Check Constraints](#) [Indexes](#) [General](#)

[+ New Column](#)

Name	Type	Primary Key	Allow Nulls	Default Value	
CityName	nvarchar(50)	<input checked="" type="checkbox"/>	<input type="checkbox"/>		
Population	int	<input type="checkbox"/>	<input type="checkbox"/>		

### Deleting a table

To delete a table, expand on the object-dropdown, and then the "Tables" folder. When your table is located, right-click on the table and select "Script as Drop", and select "Run". This generates and runs the script that is responsible for deleting your table from the database.



#### NOTE

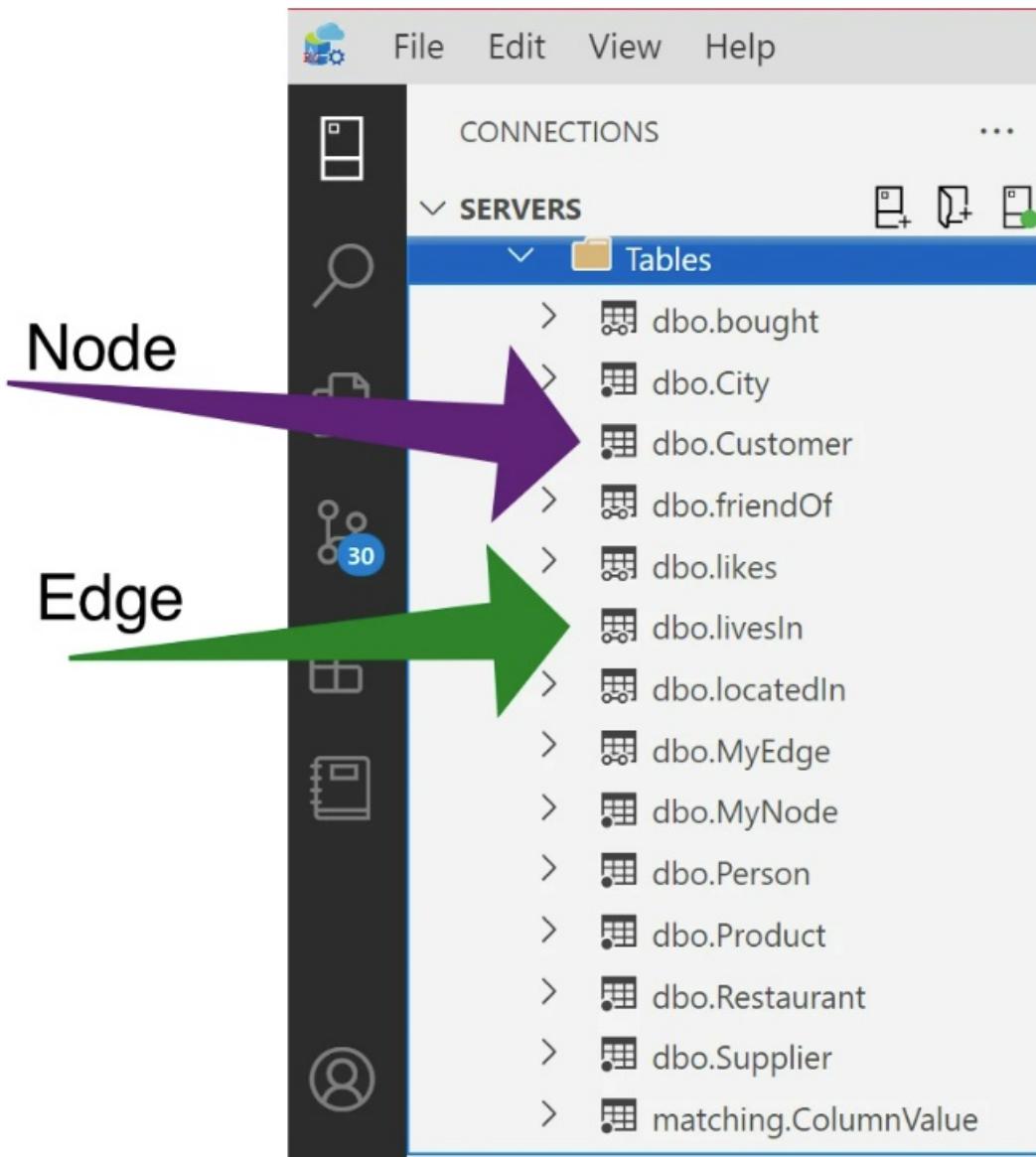
If your object explorer is open while dropping a table, you may still see this table name showing in the list of tables. Refreshing your table will take care of this.

## Table Types

The Table Designer supports different table types and properties that dictate relationships within entities and the overall structure of the table.

### Graph Tables

Graph tables are used to establish relationships between entities in your database using node and edge table relationships. In Azure Data Studio, you can easily create these relationships directly on the GUI without having to manually type out long T-SQL statements. To read more on graph tables, check out [this documentation on SQL Graph Architecture](#). The table type can be seen in the Connections Pane by the icon shown to the left of the table name. A single dot represents a node table, while the two unshaded dots represent edge tables as shown below.



#### **Creating a Graph Table**

Creating a graph table is simplified in Azure Data Studio. This can be done directly in the Table Properties pane.

#### **NOTE**

That to create graph tables, a new table has to be created. Graph tables can't be implemented for existing tables.

In this example below, we will use the GUI to create two node tables; Person, City and an edge table called "lives" with an edge constraint to establish the relationship between the two aforementioned node tables, that is "Person lives in City". For a T-SQL script version of this example, see [Create a graph database and run some pattern matching queries using T-SQL](#).

Create the node tables

#### **NOTE**

That the Graph Tables can only be implemented on new table types, and not already existing ones.

So, lets drop the "Cities" table we already created and re-create it. Before saving this new table, in the Table Properties pane, change the Graph Table option to "Node" from the drop-down.

Table name **Cities**

Columns Primary Key Foreign Keys Check Constraints Indexes General

+ New Column

Name	Type	Primary Key	Allow Nulls	Default Value
CityName	nvarchar(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="Delete"/>
Population	int	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="Delete"/>

Table Properties

**General**

Table name **Cities**  
Schema **dbo**  
Description

**Graph Table**

Type **Node**

**System Versioning**

System Versioning Enabled

**Memory Optimized**

Memory Optimized

**Type**

Specifies the table type.

Scripts

```

1 CREATE TABLE [dbo].[Cities] (
2     [CityName] NVARCHAR (50) NULL,
3     [Population] INT .....
4 ) AS NODE;

```

Notice that the script is updated to include the "as node" syntax.

Once this configuration is set, publish this update to the database. Once published, you will see that a new column for the node ID is created and this ID is referenced in the script pane as shown below.

Table name **Cities**

Columns Primary Key Foreign Keys Check Constraints Indexes General

+ New Column

Name	Type	Primary Key	Allow Nulls	Default Value
<b>\$node_id</b>	nvarchar(1000)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="button" value="Delete"/>
CityName	nvarchar(50)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="Delete"/>
Population	int	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="button" value="Delete"/>

Scripts

```

1 CREATE TABLE [dbo].[Cities] (
2     [CityName] NVARCHAR (50) NULL,
3     [Population] INT .....
4     INDEX [GRAPH_UNIQUE_INDEX_A3F0C343F57C457EA0042AB95A5F9059] UNIQUE NONCLUSTERED ($node_id)
5 ) AS NODE;

```

Create another node table called "Persons", repeating the steps above.

#### Create the edge table

Now, as mentioned earlier, we'll create our edge table, "lives". To do this, right-click on the Tables folder to create a new table. Change the name of this table from its default to "lives". To indicate that this is an edge table, click on the Graph Table Type drop-down in the Table Properties pane and select "Edge". See below.

Table name: lives

Columns Primary Key Foreign Keys Check Constraints Indexes Edge Constraints General

+ New Column

Name	Type	Primary Key	Allow Nulls	Default Value	
\$edge_id	nvarchar(1000)	<input type="checkbox"/>	<input type="checkbox"/>		<input type="button" value="Delete"/>
\$from_id	nvarchar(1000)	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="button" value="Delete"/>
\$to_id	nvarchar(1000)	<input type="checkbox"/>	<input checked="" type="checkbox"/>		<input type="button" value="Delete"/>
livesIn	nvarchar(50)	<input checked="" type="checkbox"/>	<input type="checkbox"/>		<input type="button" value="Delete"/>

Table Properties

General

Table name: lives  
Schema: dbo  
Description:

Graph Table

Type: Edge

System Versioning

System Versioning Enabled:

Memory Optimized

Memory Optimized:

Scripts

```

1 CREATE TABLE [dbo].[lives] (
2     [livesIn] NVARCHAR (50) NOT NULL,
3     CONSTRAINT [PK_lives] PRIMARY KEY CLUSTERED ([livesIn] ASC),
4     INDEX [GRAPH_UNIQUE_INDEX_4448E18FF673448D864F8C36F3ED0BCC] UNIQUE NONCLUSTERED ($edge_id)
5 ) AS EDGE;

```

To create the relationship between the node and edge, select "Edge Constraints". Select the plus sign next to "New Edge Constraint" to create a new constraint. A default name of "EC-1" is provided as seen below. Feel free to change the name as you please. For the purpose of this tutorial, we will leave this as is.

Table name: lives

Columns Primary Key Foreign Keys Check Constraints Indexes Edge Constraints General

+ New Edge Constraint

Name	Clauses	
EC_1		<input type="button" value="Delete"/>

Edge Constraint Properties

Name: EC\_1  
Is Enabled:   
On Delete Action: No Action  
Clauses:  
+ New Clause  
From Table: To Table

Scripts Issues (1)

(X) Edge constraint 'EC\_1' does not have any clauses specified.

As seen in script pane in the image above, a warning appears indicating that a clause has not been specified. We need to create the clause that will establish the edge constraint between our "Persons" and "Cities" node tables. On the "Edge Constraint Properties" pane, under "Clauses", select "+ New Clause". The "From" and "To" values will now appear in the clauses section. Hover over the "From Table" to view the drop-down, select "dbo.Persons" and from the "To Table" drop-down, select "dbo.Cities". Be sure to publish changes to save your work. See below for the overview of what the Table Designer view should look like after performing these steps described above.

Table name: lives

Columns Primary Key Foreign Keys Check Constraints Indexes Edge Constraints General

+ New Edge Constraint

Name	Clauses	
EC_1	(dbo.Cities,dbo.Persons)	<input type="button" value="Delete"/>

Edge Constraint Properties

Name: EC\_1  
Is Enabled:   
On Delete Action: No Action  
Clauses:  
+ New Clause  
From Table: To Table  
dbo.Cities: dbo.Persons

Scripts

```

1 CREATE TABLE [dbo].[lives] (
2     [livesIn] NVARCHAR (50) NOT NULL,
3     CONSTRAINT [PK_lives] PRIMARY KEY CLUSTERED ([livesIn] ASC),
4     CONSTRAINT [EC_1] CONNECTION ([dbo].[cities] TO [dbo].[persons]),
5     INDEX [GRAPH_UNIQUE_INDEX_4448E18FF673448D864F8C36F3ED0BCC] UNIQUE NONCLUSTERED ($edge_id)
6 ) AS EDGE;

```

You can add as little or as many constraints as needed depending on the nature of the relationships between the different tables in your database.

## Memory-Optimized Tables

Creating Memory-Optimized Tables can be done directly in Azure Data Studio. All it takes is checking a box at

the time of table creation and the table in your database is memory-optimized!

#### NOTE

That to create Memory-Optimized tables, a new table has to be created. Memory-Optimized tables can't be implemented on existing tables.

Memory-Optimized tables must have non-clustered primary key. For an introduction to Memory-Optimized Tables, check out the [Introduction to Memory-Optimized Tables](#) article.

#### Create a Memory-Optimized Table

To create a Memory-Optimized Table, we need to ensure that a filegroup has been created for our database. To read more on this, check out this documentation on [the memory optimized filegroup](#).

Next, create your table by right-clicking on the "Tables" folder and selecting "New Table". This will bring up the main pane and Table Properties pane where you can opt to have your table Memory Optimized. Be sure to assign a non-clustered primary key like shown below.

The screenshot shows the 'New Table 1' dialog box. The 'Table name' is 'NewTable'. Under 'Columns', 'column\_1' is listed with 'Is Ascending' checked. The 'Primary Key' checkbox is checked (highlighted with a red box). The 'Is Clustered' checkbox is unchecked (highlighted with a red box). In the bottom-left corner, there is a 'Scripts' section containing the following T-SQL code:

```
1 CREATE TABLE [dbo].[NewTable] (
2     [column_1] INT NOT NULL,
3     CONSTRAINT [PK_NewTable] PRIMARY KEY NONCLUSTERED ([column_1] ASC)
4 )
5 WITH (MEMORY_OPTIMIZED = ON);
6
7
```

Next, head over to the Table Properties pane. Select the Memory Optimized checkbox. This will generate the durability drop-down where you can choose whether you want only the Schema to be stored in memory or both the Schema and Data. Choosing "Schema" only saves the schema of your database to memory. As you can see below, the script is updated to reflect the changes.

The screenshot shows the 'New Table' creation dialog in Azure Data Studio. In the 'General' tab, the table name is set to 'NewTable', schema to 'dbo', and the primary key is named 'PK\_NewTable'. Under 'System Versioning', the 'Memory Optimized' checkbox is checked, and 'Durability' is set to 'Schema Only'. The 'Scripts' pane at the bottom contains the generated SQL script:

```

1 CREATE TABLE [dbo].[NewTable] (
2     [column_1] INT NOT NULL,
3     CONSTRAINT [PK_NewTable] PRIMARY KEY NONCLUSTERED ([column_1] ASC)
4 )
5 WITH (DURABILITY = SCHEMA_ONLY, MEMORY_OPTIMIZED = ON);
6
7

```

Choosing "Schema and Data" saves both the schema and Data to memory if there's a disaster. Notice the change in the script.

The screenshot shows the 'New Table' creation dialog in Azure Data Studio. In the 'General' tab, the table name is set to 'NewTable', schema to 'dbo', and the primary key is named 'PK\_NewTable'. Under 'System Versioning', the 'Memory Optimized' checkbox is checked, and 'Durability' is set to 'Schema and Data'. The 'Scripts' pane at the bottom contains the generated SQL script:

```

1 CREATE TABLE [dbo].[NewTable] (
2     [column_1] INT NOT NULL,
3     CONSTRAINT [PK_NewTable] PRIMARY KEY NONCLUSTERED ([column_1] ASC)
4 )
5 WITH (DURABILITY = SCHEMA_AND_DATA, MEMORY_OPTIMIZED = ON);
6
7

```

## System-Versioned Tables

System-versioned Tables, also known as Temporal Tables can also be configured directly on Azure Data Studio. If you're new to system versioning, check out [Temporal Tables on SQL Server](#). System-versioning tables must have the period columns defined.

### Creating a System-Versioned Table

Create a table called "Department" with the column properties as seen below. To do this, right-click on the Tables folder in the Connections pane and select "New Table". Next, in the Table Properties pane, select the "System Versioning Enabled" check box. You can decide to rename this history table or leave the default name as is.

Table name: Department

Columns: Primary Key Foreign Keys Check Constraints Indexes General

+ New Column

Name	Type	Primary Key	Allow Nulls	Default Value	
DeptID	int	<input checked="" type="checkbox"/>	<input type="checkbox"/>		<input type="button" value="Delete"/>
DeptName	nvarchar(50)	<input type="checkbox"/>	<input type="checkbox"/>		<input type="button" value="Delete"/>
ValidFrom	datetime2(7)	<input type="checkbox"/>	<input type="checkbox"/>		<input type="button" value="Delete"/>
ValidTo	datetime2(7)	<input type="checkbox"/>	<input type="checkbox"/>		<input type="button" value="Delete"/>

Table Properties

General

Table name: Department  
Schema: dbo  
Description:  
Graph Table  
Type: None

System Versioning

System Versioning Enabled   
Auto Create History Table   
New History Table Name: MSSQL\_NewTable\_History

Memory Optimized

Memory Optimized

System Versioning Enabled   
Specifies whether the table is system versioning enabled.

Scripts Issues (1)

System versioned table must have the period columns defined.

Next, we need to define the period columns and indicate the table is system-versioned. To define the period columns, select "ValidFrom" and then in the column properties under "System Versioning", set the "Generated Always As" to "Row Start", repeat for "ValidTo" but set to "Row End". With the period columns defined, select the table to show the Table Properties and check the "System-Versioning Enabled" checkbox. The script pane will now be updated to show the T-SQL version of this system-verisoned table.

Table name: Department

Columns: Primary Key Foreign Keys Check Constraints Indexes General

+ New Column

Name	Type	Primary Key	Allow Nulls	Default Value	
DeptID	int	<input checked="" type="checkbox"/>	<input type="checkbox"/>		<input type="button" value="Delete"/>
DeptName	nvarchar(50)	<input type="checkbox"/>	<input type="checkbox"/>		<input type="button" value="Delete"/>
ValidFrom	datetime2(7)	<input type="checkbox"/>	<input type="checkbox"/>		<input type="button" value="Delete"/>
ValidTo	datetime2(7)	<input type="checkbox"/>	<input type="checkbox"/>		<input type="button" value="Delete"/>

Column Properties

General

Name: ValidFrom  
Description:  
Type: datetime2  
Length:  
Default Value:  
Allow Nulls:  
Primary Key:  
Precision:  
Scale: 7  
Default Constraint Name:  
Identity Specification

Is Identity:  
Identity Seed:  
Identity Increment:  
System Versioning

Generated Always As: Row start  
Is Hidden:

Scripts

```

1 CREATE TABLE [dbo].[Department] [
2     [DeptID] INT NOT NULL,
3     [DeptName] VARCHAR (50) NOT NULL,
4     [ValidFrom] DATETIME2 (7) GENERATED ALWAYS AS ROW START NOT NULL,
5     [ValidTo] DATETIME2 (7) GENERATED ALWAYS AS ROW END NOT NULL,
6     CONSTRAINT [PK_Department] PRIMARY KEY CLUSTERED ([DeptID] ASC),
7     PERIOD FOR SYSTEM_TIME ([ValidFrom], [ValidTo])
8 ]
9 WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE=[dbo].[MSSQL_Department_History], DATA_CONSISTENCY_CHECK=ON));
10
11

```

## Next steps

- Download Azure Data Studio

# Use Jupyter Notebooks in Azure Data Studio

9/26/2022 • 4 minutes to read • [Edit Online](#)

Applies to: SQL Server 2019 (15.x)

Jupyter Notebook is an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text. Usage includes data cleaning and transformation, numerical simulation, statistical modeling, data visualization, and machine learning.

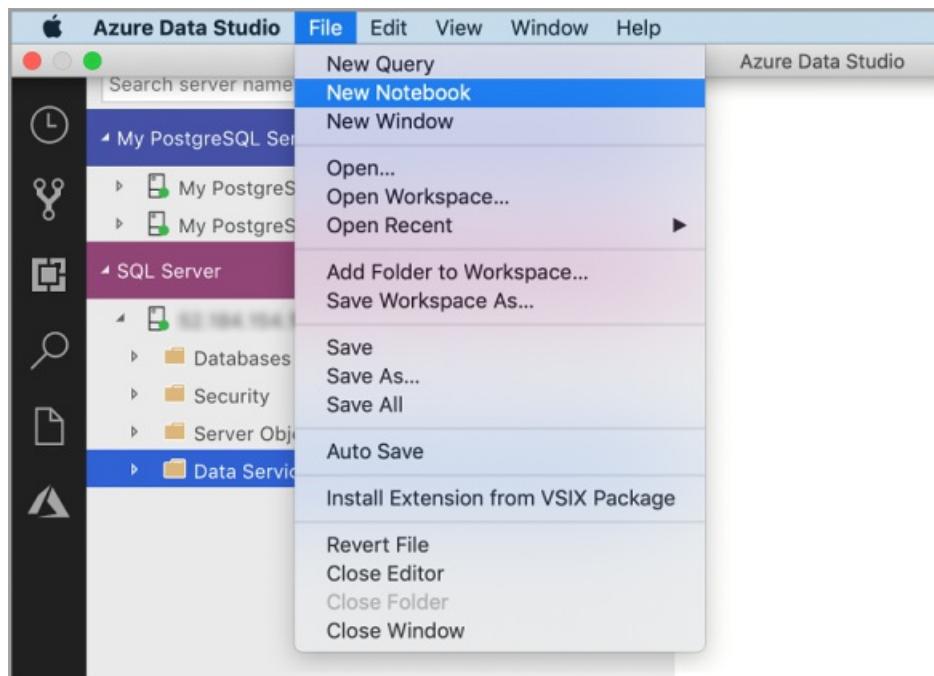
This article describes how to create a new notebook in the latest release of [Azure Data Studio](#) and how to start authoring your own notebooks using different kernels.

Watch this short 5-minute video for an introduction to notebooks in Azure Data Studio:

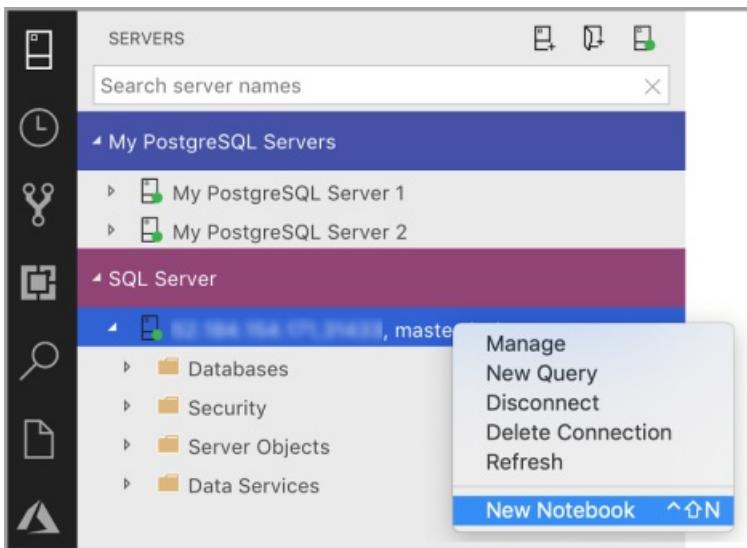
## Create a notebook

There are multiple ways to create a new notebook. In each case, a new file named `Notebook-1.ipynb` opens.

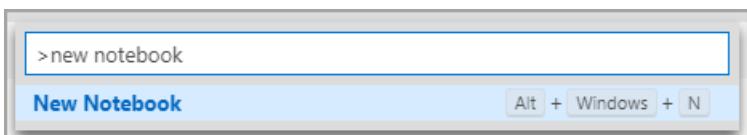
- Go to the **File Menu** in Azure Data Studio and select **New Notebook**.



- Right-click a **SQL Server** connection and select **New Notebook**.



- Open the command palette (**Ctrl+Shift+P**), type "new notebook", and select the **New Notebook** command.



## Connect to a kernel

Azure Data Studio notebooks support a number of different kernels, including SQL Server, Python, PySpark, and others. Each kernel supports a different language in the code cells of your notebook. For example, when connected to the SQL Server kernel, you can enter and run T-SQL statements in a notebook code cell.

**Attach to** provides the context for the kernel. For example, if you're using SQL Kernel, then you can attach to any of your SQL Server instances. If you're using Python3 Kernel you attach to **localhost** and you can use this kernel for your local Python development.

SQL Kernel can also be used to connect to PostgreSQL server instances. If you're a PostgreSQL developer and want to connect the notebooks to your PostgreSQL Server, then download the [PostgreSQL extension](#) in the Azure Data Studio extension Marketplace and connect to the PostgreSQL server.

If you're connected to SQL Server 2019 big data cluster, the default **Attach to** is the end point of the cluster. You can submit Python, Scala, and R code using the Spark compute of the cluster.

KERNEL	DESCRIPTION
SQL Kernel	Write SQL Code targeted at your relational database.
PySpark3 and PySpark Kernel	Write Python code using Spark compute from the cluster.
Spark Kernel	Write Scala and R code using Spark compute from the cluster.
Python Kernel	Write Python code for local development.

For more information on specific kernels, see:

- [Create and run a SQL Server notebook](#)
- [Create and run a Python notebook](#)
- [Kqlmagic extension in Azure Data Studio](#) - this extends the capabilities of the Python kernel

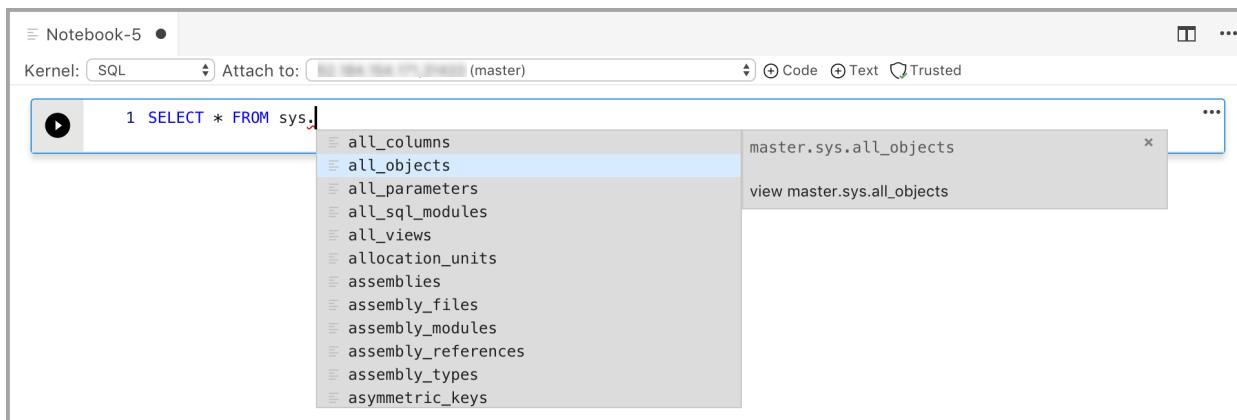
## Add a code cell

Code cells allow you to run code interactively within the notebook.

Add a new code cell by clicking the **+Cell** command in the toolbar and selecting **Code cell**. A new code cell is added after the currently selected cell.

Enter code in the cell for the selected kernel. For example, if you're using the SQL kernel, you can enter T-SQL commands in the code cell.

Entering code with the SQL kernel is similar to a SQL query editor. The code cell supports a modern SQL coding experience with built-in features such as a rich SQL editor, IntelliSense, and built-in code snippets. Code snippets allow you to generate the proper SQL syntax to create databases, tables, views, stored procedures, and to update existing database objects. Use code snippets to quickly create copies of your database for development or testing purposes and to generate and execute scripts.

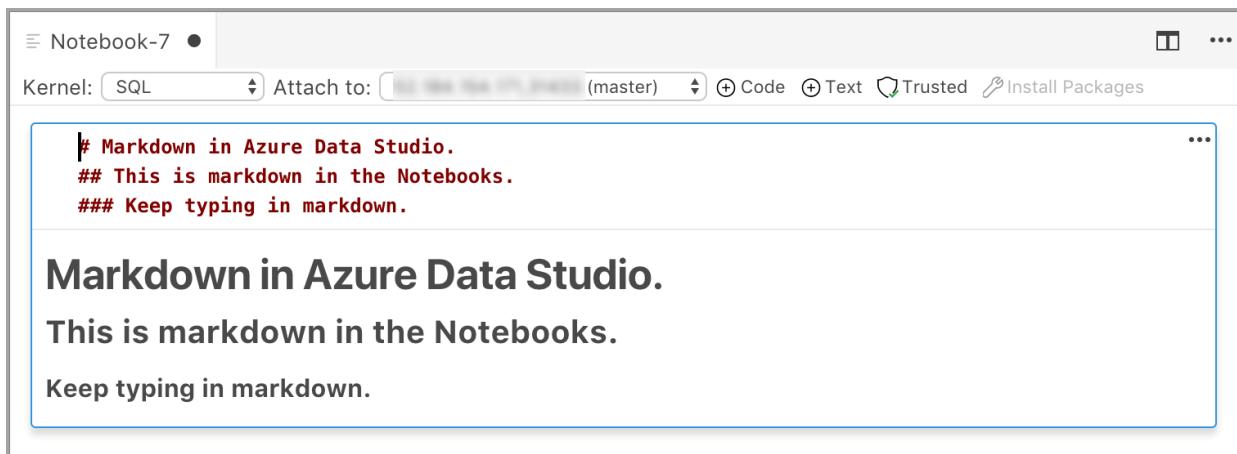


## Add a text cell

Text cells allow you to document your code by adding Markdown text blocks in between code cells.

Add a new text cell by clicking the **+Cell** command in the toolbar and selecting **Text cell**.

The cell starts in edit mode in which you can type Markdown text. As you type, a preview is shown below.



Selecting outside the text cell shows the Markdown text.

The screenshot shows a notebook titled "Notebook-7". The kernel is set to "SQL". The cell contains the following text:

```
Markdown in Azure Data Studio.  
This is markdown in the Notebooks.  
Keep typing in markdown.
```

If you click in the text cell again, it changes to edit mode.

## Run a cell

To run a single cell, click **Run cell** (the round black arrow) to the left of the cell or select the cell and press F5. You can run all cells in the notebook by clicking **Run all** in the toolbar - the cells are run one at a time and execution stops if an error is encountered in a cell.

Results from the cell are shown below the cell. You can clear the results of all the executed cells in the notebook by selecting the **Clear Results** button in the toolbar.

## Save a notebook

To save a notebook, do one of the following.

- Type Ctrl+S
- Select **Save** from the **File** menu
- Select **Save As...** from the **File** menu
- Select **Save All** from the **File** menu - this saves all open notebooks
- In the command palette, enter **File: Save**

Notebooks are saved as `.ipynb` files.

## Trusted and Non Trusted

The notebooks open in Azure Data Studio are defaulted to **Trusted**.

If you open a notebook from some other source, it opens in **Non-Trusted** mode and then you can make it **Trusted**.

## Examples

The following examples demonstrate using different kernels to run a simple "Hello World" command. Select the kernel, enter the example code in a cell, and click **Run cell**.

### Pyspark

The screenshot shows a pyspark session. The code entered is:

```
1 print("Hello World")
```

The output shows the application starting and the "Hello World" message being printed.

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
80	application_1543890300284_0092	pyspark3	idle	Link	Link	✓

SparkSession available as 'spark'.  
Hello World

### Spark | Scala language

```
1 print("Hello World")
```

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
84	application_1543890300284_0096	spark	idle	Link	Link	✓

SparkSession available as 'spark'.  
Hello World

## Spark | R language

```
1 print('Hello World in R')
```

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
77	application_1543890300284_0089	sparkr	idle	Link	Link	✓

SparkSession available as 'spark'.  
[1] "Hello World in R"

## Python 3

Kernel: Python 3 Attach to: localhost + Code + Text Trusted ⚡ Install Packages

```
[1] 1 print("Hello World !")
```

Hello World !

## Next steps

- [Create and run a SQL Server notebook.](#)
- [Create and run a Python notebook](#)
- [Run Python and R scripts in Azure Data Studio notebooks with SQL Server Machine Learning Services.](#)
- [Deploy SQL Server big data cluster with Azure Data Studio notebook.](#)
- [Manage SQL Server Big Data Clusters with Azure Data Studio notebooks.](#)
- [Run a sample notebook using Spark.](#)

# Notebook views (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

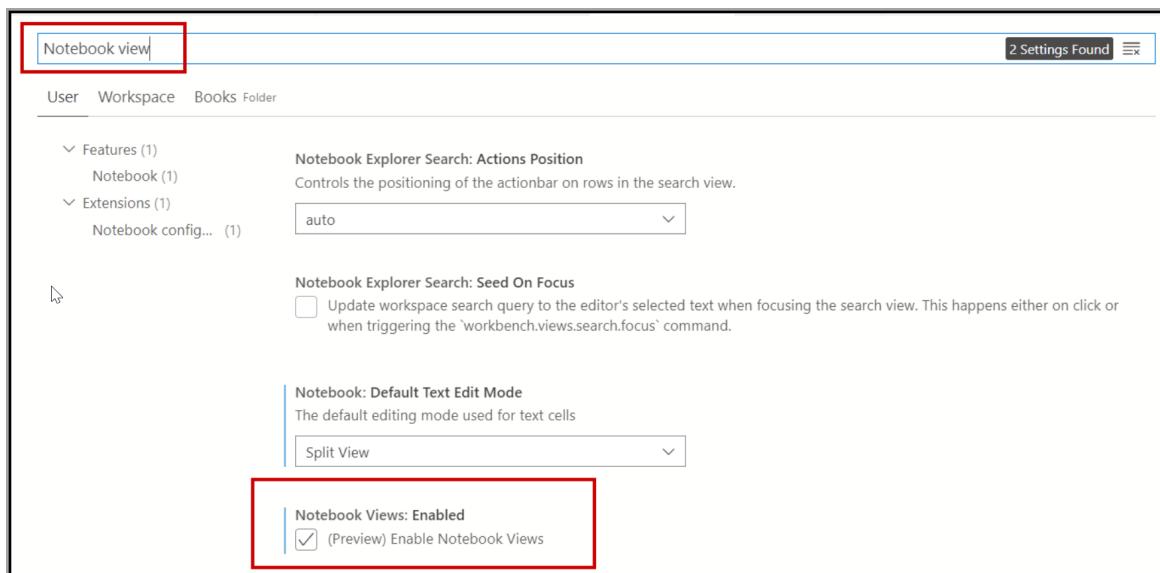
Notebook Views is ideal for consumers of notebook results such as data analysis outcomes. Notebook authors can hide complexities and select the cells that are relevant for consumers.

One notebook can have multiple views and work for any kernel.

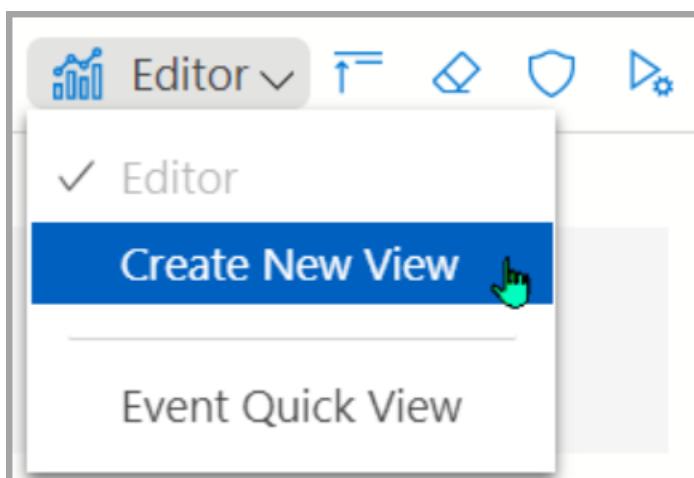
Notebook Views is now available in [Azure Data Studio](#), starting with the October 2021 release (1.33.0).

## Get started

1. Enable Notebook Views from Settings.
2. Go to **Settings**, then search for `notebook view`. Ensure that **Notebook Views: Enabled** is selected.



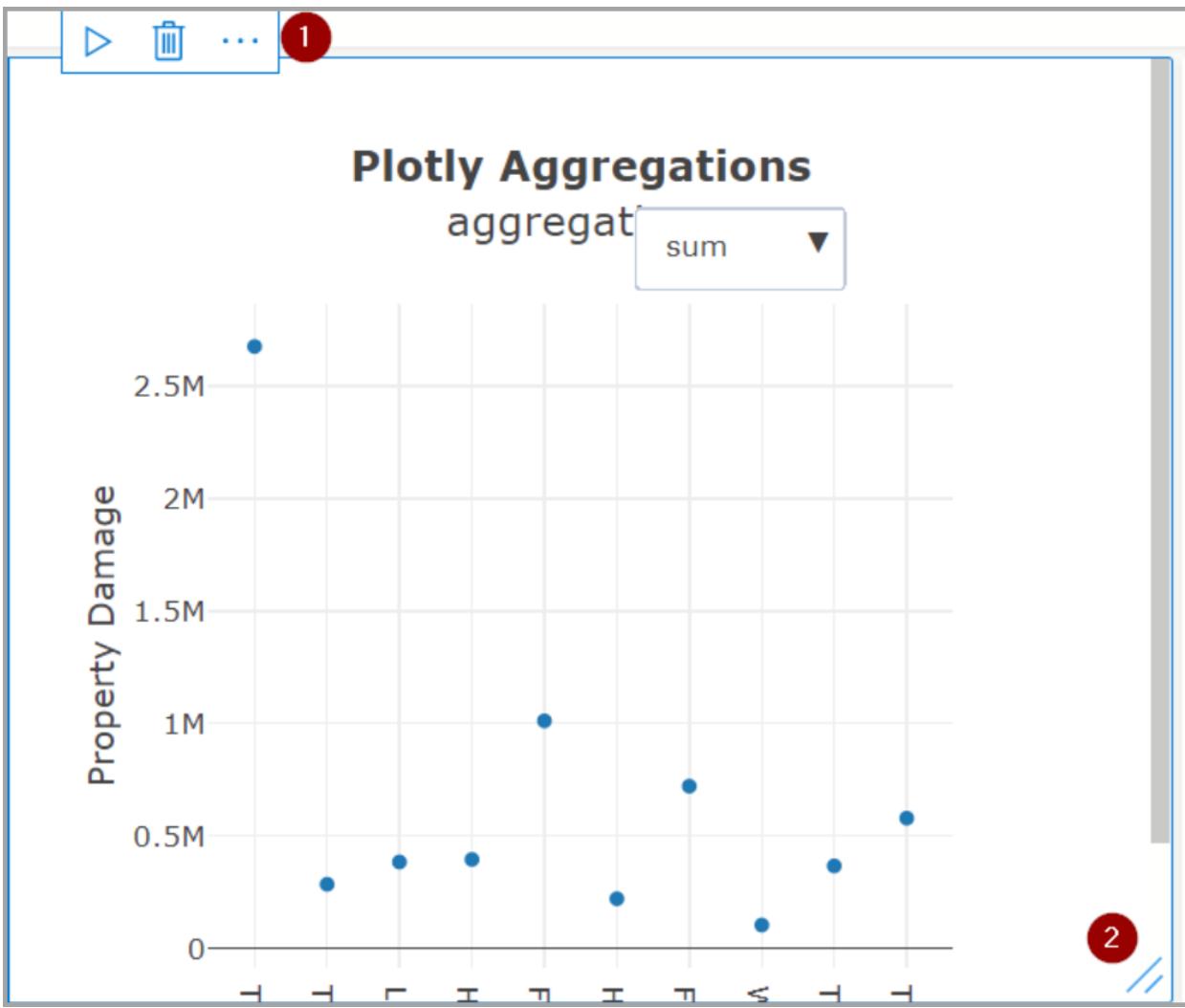
3. In an existing notebook, select **Editor** and choose **Create New View**.



Now you can see that all the cells (both text and code cells) have been added to the View canvas.

## Resize, move, and delete cells

From here, you can delete, resize, and move the cells from the view.

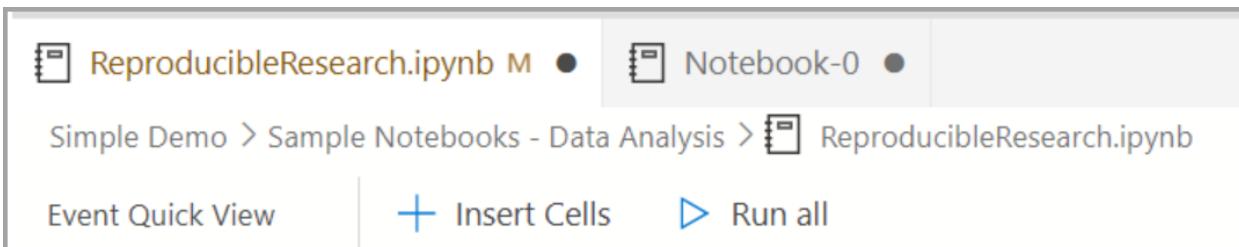


- To delete, select the cell and select the **Delete** button on top of the cell (circle 1 in the screenshot).
- To resize, select the cell and hover to the bottom-right corner (circle 2 of the screenshot), and your cursor should be in resize mode.
- To move, select the cell and drag the cell to the desired location.
- To run the cell only, select the cell to activate the cell, then select the **Run** button from the top menu (circle 1 of the screenshot).

#### NOTE

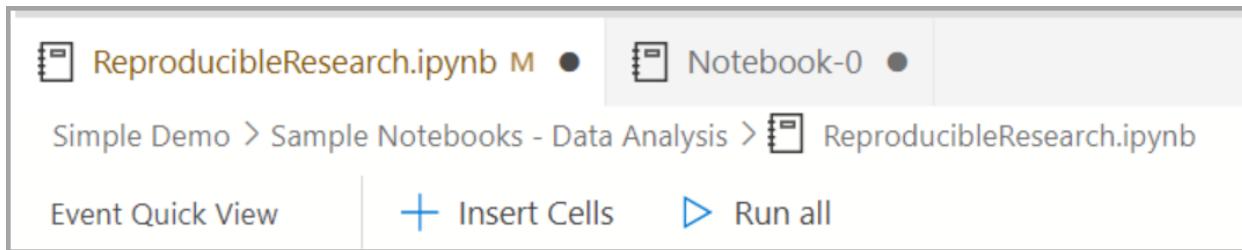
If the cell in the notebook has dependencies on the previous cells, this might result in an error.

## Insert and run all cell views



- To add new cells, select the **Insert Cells** button on the top right of the Editor view.
- To refresh the view, select **Run All**. This runs all the cells in the notebook, in the background, even if they aren't on the view itself.

## Manage views



- To navigate to another view or create a new one, use the View drop-down list
- To rename the view, select the gear button on the top right of the view.
- To delete the view, select the **Delete** button on the top right of the view.

## Next steps

- [Notebook guidance](#)

# Create and run a notebook with Azure Monitor Logs (Preview)

9/26/2022 • 3 minutes to read • [Edit Online](#)

This article shows you how to create and run an [Azure Data Studio notebook](#) using the [Azure Monitor Log extension](#) to connect to a [Log Analytics workspace](#) to view results for an Azure SQL database.

Once the Azure Monitor Logs extension is installed, you can connect to your Azure Monitor Log workspaces, browse the tables, write/execute KQL queries against workspaces and write/execute Notebooks connected to the Log Analytics kernel.

With the Azure Monitor Log extension, you can change the kernel option to [Log Analytics](#).

This feature is currently in preview.

## Prerequisites

If you don't have an Azure subscription, create a [free Azure account](#) before you begin.

The following prerequisites are also required:

- [Azure SQL Database](#)
- [Azure Data Studio installed](#).
- [Log Analytics workspace](#).
- [Azure Monitor Log extension](#).

## Connect to an Azure Monitor Logs (Log Analytics) workspace

There are two main tables in a Log Analytics (Azure Monitor Logs) workspace that capture Azure SQL events:

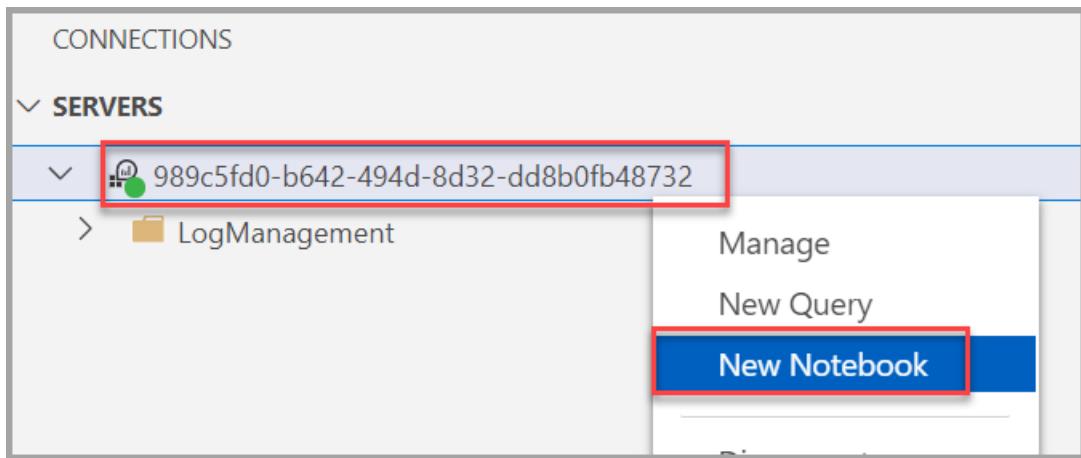
In the examples below, this article uses the [AzureDiagnostics](#) table and the [AzureMetric](#) table in a Log Analytics workspace, which store some Azure SQL event data. We have previously configured Azure SQL to write its selected events to a Log Analytics workspace. Learn more about how to do this [here](#).

You can connect to a [Log Analytics workspace](#).

## Create an Azure Monitor Log notebook

The following steps show how to create a notebook file in Azure Data Studio:

1. In Azure Data Studio, connect to your Log Analytics workspace.
2. Navigate to the **Connections** pane and under the **Servers** window, right-click the Log analytics workspace and select *New Notebook*. You can also go to **File > New Notebook**.



3. Select *Log Analytics* for the **Kernel**. Confirm that the **Attach to** menu is set to the workspace name.



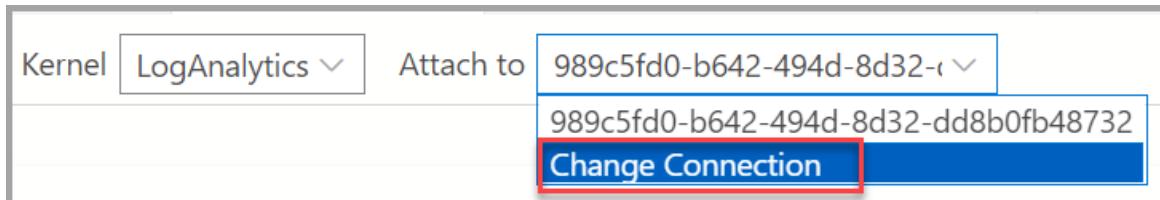
You can save the notebook using the **Save** or **Save as...** command from the **File** menu.

To open a notebook, you can use the **Open file...** command in the **File** menu, select **Open file** on the **Welcome** page, or use the **File: Open** command from the command palette.

## Change the connection

To change the Azure Monitor Log connection for a notebook:

1. Select the **Attach to** menu from the notebook toolbar and then select **Change Connection**.



### NOTE

Ensure that the workspace value is populated. Azure Monitor Log notebooks require to have the Workspace ID specified in the **Server** field.

2. Now, you can either select a recent connection workspace or enter new connection details to connect.

## Connection

Recent      [Browse](#)

[!\[\]\(8add110465e026ef9286a19aa9003de9\_img.jpg\) Clear List](#)

 989c5fd0-b642-494d-8d32-dd8b0fb48732

### Connection Details

Connection type	Azure Monitor Logs
Workspace ID *	
Authentication type	Azure Active Directory - Universal with MFA support
Account	
Database	<Default>
Server group	<Do not save>
Name (optional)	

[Advanced](#)

## Run a code cell

You can create cells containing KQL queries that you can run in place by selecting the **Run cell** button to the cell's left. The results are shown in the notebook after the cell runs.

For example:

Add a new code cell by selecting the **+Code** command in the toolbar.

Click on **+ Code** or **+ Text** to add a code or text cell

## Query log results for an Azure SQL database

You can connect to your Log Analytics workspace(s) to browse tables, write KQL queries against workspaces, and create notebooks connected to the Log Analytics kernel.

### NOTE

Allow about 15 minutes before the log results appear.

Some sample queries are listed below.

### AzureMetrics

Here's a sample query to view AzureMetrics results.

AzureMetrics

1 AzureMetrics

Commands completed successfully.

Total execution time: 00:00:01.859

Columns: TenantId, SourceSystem, TimeGenerated, ResourceId, OperationType

	TenantId	SourceSystem	TimeGenerated	ResourceId	OperationType
1	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:18:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
2	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:19:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
3	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:20:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
4	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:18:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
5	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:19:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
6	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:20:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
7	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:18:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
8	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:19:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
9	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:20:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
10	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:18:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
11	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:19:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
12	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:20:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
13	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:18:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
14	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:19:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
15	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:20:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
16	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:18:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
17	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:19:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
18	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:20:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	
19	989c5fd0-b642-494d-8d32-dd8b...	Azure	2021-11-02 17:18:00	/SUBSCRIPTIONS/AB39DBC6-61CD...	

## AzureDiagnostics

Here's a sample query to view AzureDiagnostics results.

The AzureDiagnostics table tends to have more details than AzureMetrics.

```
AzureDiagnostics
| project-away TenantId, ResourceId, SubscriptionId, _ResourceId, ResourceGroup // hide sensitive info
| take 10
```

```

1 AzureDiagnostics
2 | project-away TenantId, ResourceId, SubscriptionId, _ResourceId, ResourceGroup // hide sensitive info
3 | take 10

```

Commands completed successfully.

Total execution time: 00:00:00.611

	TimeGenerated	Category	ResourceProvider	Resource	ResourceType	OperationName	Re
1	2021-11-02 13:15:08	DatabaseWaitStatistics	MICROSOFT.SQL	MARKGTEST	SERVERS/DATABASES	DatabaseWaitStatistcsEvent	
2	2021-11-02 13:15:08	DatabaseWaitStatistics	MICROSOFT.SQL	MARKGTEST	SERVERS/DATABASES	DatabaseWaitStatistcsEvent	
3	2021-11-02 13:15:08	DatabaseWaitStatistics	MICROSOFT.SQL	MARKGTEST	SERVERS/DATABASES	DatabaseWaitStatistcsEvent	
4	2021-11-02 13:15:08	DatabaseWaitStatistics	MICROSOFT.SQL	MARKGTEST	SERVERS/DATABASES	DatabaseWaitStatistcsEvent	
5	2021-11-02 13:15:08	DatabaseWaitStatistics	MICROSOFT.SQL	MARKGTEST	SERVERS/DATABASES	DatabaseWaitStatistcsEvent	
6	2021-11-02 13:15:08	DatabaseWaitStatistics	MICROSOFT.SQL	MARKGTEST	SERVERS/DATABASES	DatabaseWaitStatistcsEvent	
7	2021-11-02 13:16:17	QueryStoreRuntimeStatistics	MICROSOFT.SQL	MARKGTEST	SERVERS/DATABASES	QueryStoreRuntimeStatisticsE...	
8	2021-11-02 13:16:17	QueryStoreRuntimeStatistics	MICROSOFT.SQL	MARKGTEST	SERVERS/DATABASES	QueryStoreRuntimeStatisticsE...	
9	2021-11-02 13:16:17	QueryStoreRuntimeStatistics	MICROSOFT.SQL	MARKGTEST	SERVERS/DATABASES	QueryStoreRuntimeStatisticsE...	
10	2021-11-02 13:16:17	QueryStoreWaitStatistics	MICROSOFT.SQL	MARKGTEST	SERVERS/DATABASES	QueryStoreWaitStatisticsEvent	

## Analyze events by Diagnostic Settings

Here's a query to analyze the number of events by Operation Name.

### NOTE

Each row in AzureDiagnostic represents an event for a specific Operation or category. Some SQL actions may result in generating multiple events of different types.

```

AzureDiagnostics
| summarize count() by OperationName

```

```

1 AzureDiagnostics
2 | summarize count() by OperationName

```

Commands completed successfully.

Total execution time: 00:00:01.442

	OperationName	count_
1	DatabaseWaitStatistcsEvent	19
2	QueryStoreRuntimeStatisticsE...	5
3	QueryStoreWaitStatisticsEvent	2
4	AutomaticTuningSettingsSna...	16
5	ErrorEvent	15

You can try some more examples from the [Azure Monitor Logs samples repo](#).

## Next steps

- [Azure Monitor Log extension](#)
- [Azure Monitor Log samples](#)
- [Notebook guidance](#)

# Create and run a Kusto (KQL) notebook (Preview)

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article shows you how to create and run an [Azure Data Studio notebook](#) using the [Kusto \(KQL\) extension](#), connecting to an Azure Data Explorer cluster.

With the Kusto (KQL) extension, you can change the kernel option to **Kusto**.

This feature is currently in preview.

## Prerequisites

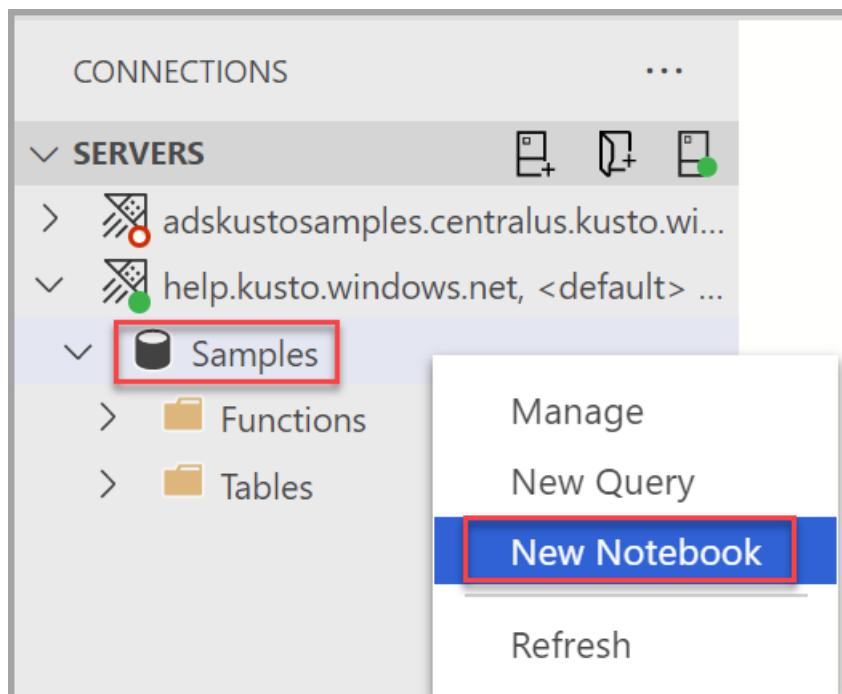
If you don't have an Azure subscription, create a [free Azure account](#) before you begin.

- [An Azure Data Explorer cluster with a database that you can connect to](#).
- [Azure Data Studio](#).
- [Kusto \(KQL\) extension for Azure Data Studio](#).

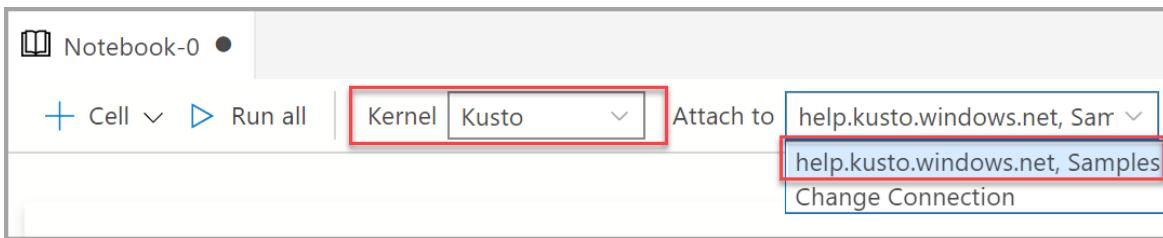
## Create a Kusto (KQL) notebook

The following steps show how to create a notebook file in Azure Data Studio:

1. In Azure Data Studio, connect to your Azure Data Explorer cluster.
2. Navigate to the **Connections** pane and under the **Servers** window, right-click the Kusto database and select **New Notebook**. You can also go to **File > New Notebook**.



3. Select **Kusto** for the **Kernel**. Confirm that the **Attach to** menu is set to the cluster name and database. For this article, we use the help.kusto.windows.net cluster with the Samples database data.



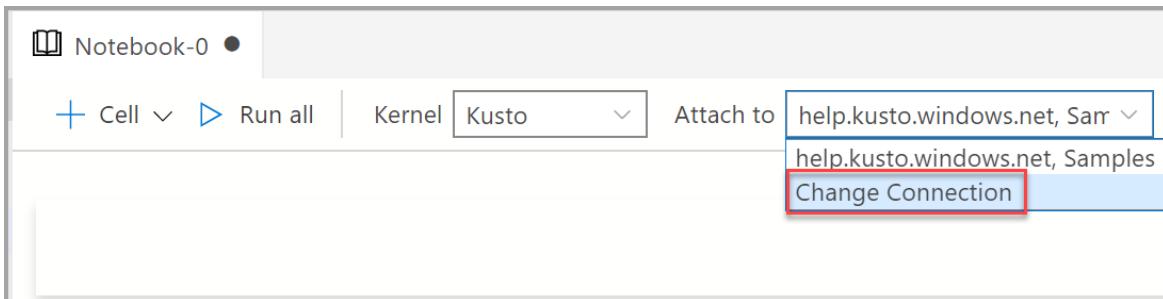
You can save the notebook using the **Save** or **Save as...** command from the **File** menu.

To open a notebook, you can use the **Open file...** command in the **File** menu, select **Open file** on the **Welcome** page, or use the **File: Open** command from the command palette.

## Change the connection

To change the Kusto connection for a notebook:

1. Select the **Attach to** menu from the notebook toolbar and then select **Change Connection**.



### NOTE

Ensure that the database value is populated. Kusto notebooks require to have the database specified.

2. Now you can either select a recent connection server or enter new connection details to connect.

## Connection

Recent Connections Saved Connections

 Clear List

- help.kusto.windows.net,
- adskustosamples.centralus.kusto.windows.net, .

### Connection Details

Connection type	Microsoft SQL Server
Server	<input type="text"/>
Authentication type	Azure Active Directory - Universal with MFA support
Account	<input type="text"/>
Database	<Default>
Server group	<input type="text"/>
Name (optional)	<input type="text"/>
<a href="#">Advanced...</a>	

[Connect](#)

[Cancel](#)

#### NOTE

Specify the cluster name without the `https://`.

## Run a code cell

You can create cells containing KQL queries that you can run in place by selecting the **Run cell** button to the cell's left. The results are shown in the notebook after the cell runs.

For example:

1. Add a new code cell by selecting the **+Code** command in the toolbar.

Click on **+ Code** or **+ Text** to add a code or text cell

2. Copy and paste the following example into the cell and select **Run cell**. This example queries the `StormEvents` data for a specific event type.

```
StormEvents  
| where EventType == "Waterspout"
```



```
1  StormEvents  
2  | where EventType == "Waterspout"
```

Commands completed successfully.

Total execution time: 00:00:00.177



	StartTime	EndTime	EpisodeId	EventId	State
1	2007-09-29 08:11:00	2007-09-29 08:11:00	11091	61032	ATLANTIC S
2	2007-06-27 13:36:00	2007-06-27 14:00:00	7369	42340	ATLANTIC S
3	2007-12-07 11:25:00	2007-12-07 11:30:00	12124	66278	ATLANTIC S
4	2007-12-20 09:05:00	2007-12-20 09:15:00	12127	66284	ATLANTIC S
5	2007-12-20 11:15:00	2007-12-20 11:30:00	12127	66286	ATLANTIC S

## Save the result or show chart

If you run a script that returns a result, you can save that result in different formats using the toolbar displayed above the result.

- Save As CSV
- Save As Excel
- Save As JSON
- Save As XML
- Show Chart

```
StormEvents  
| limit 10
```

```
[2] 1     ....StormEvents  
2     ....| limit 10
```

Commands completed successfully.

Total execution time: 00:00:00.238



	StartTime	EndTime
1	2007-09-29 08:11:00	2007-09-29 08:11:00
2	2007-09-18 20:00:00	2007-09-19 18:00:00
3	2007-09-20 21:57:00	2007-09-20 22:05:00

## Known issues

DETAILS	WORKAROUND
Query result only shows column headers.	N/A

You can file a [feature request](#) to provide feedback to the product team.

You can file a [bug](#) to provide feedback to the product team.

## Next steps

Learn more about notebooks:

- [Kusto \(KQL\) extension for Azure Data Studio](#)
- [How to use notebooks in Azure Data Studio](#)
- [Create and run a Python notebook](#)
- [Create and run a SQL Server notebook](#)

# Create and run a notebook with Kqlmagic

9/26/2022 • 4 minutes to read • [Edit Online](#)

Kqlmagic is a command that extends the capabilities of the Python kernel in [Azure Data Studio notebooks](#). You can combine Python and [Kusto query language \(KQL\)](#) to query and visualize data using rich Plotly library integrated with `render` commands. Kqlmagic brings you the benefit of notebooks, data analysis, and rich Python capabilities all in the same location. Supported data sources with Kqlmagic include [Azure Data Explorer](#), [Application Insights](#), and [Azure Monitor logs](#).

This article shows you how to create and run a notebook in Azure Data Studio using the Kqlmagic extension for an Azure Data Explorer cluster, an Application Insights log, and Azure Monitor logs.

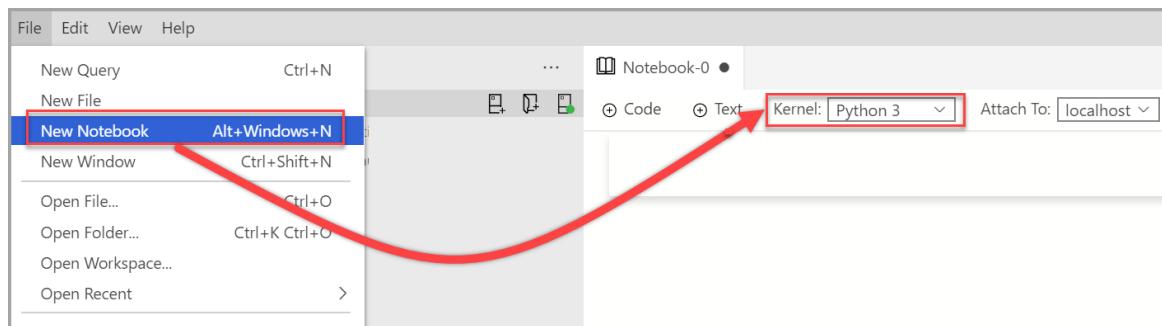
## Prerequisites

- [Azure Data Studio](#)
- [Python](#)

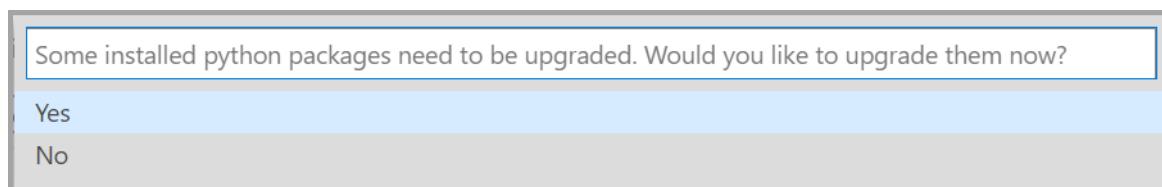
## Install and set up Kqlmagic in a notebook

The steps in this section all run within an Azure Data Studio notebook.

1. Create a new notebook and change the **Kernel** to *Python 3*.



2. You may be prompted to upgrade your Python packages when your packages need updating.



3. Install Kqlmagic:

```
import sys  
!{sys.executable} -m pip install Kqlmagic --no-cache-dir --upgrade
```

Verify it's installed:

```
import sys  
!{sys.executable} -m pip list
```

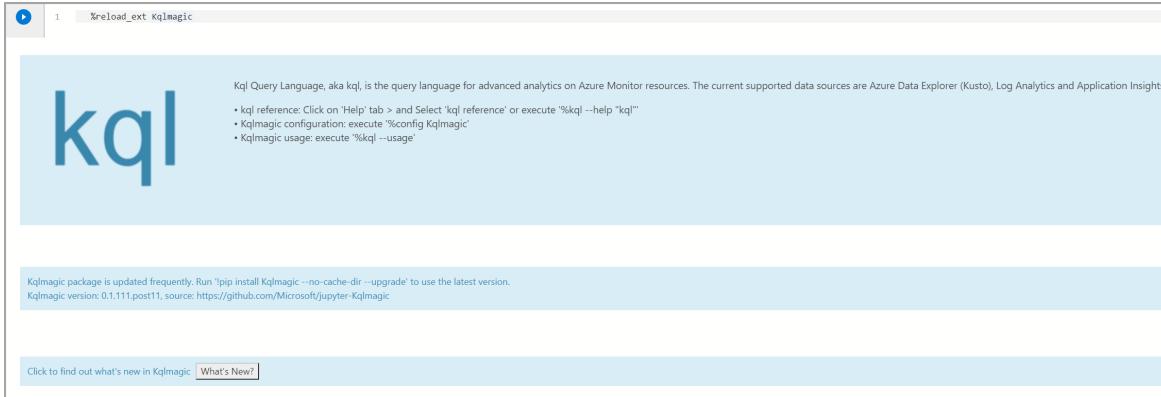
```
jupyter
jupyter-client
jupyter-console
jupyter-core
kiwisolver
Kqlmagic
lxml
Markdown
MarkupSafe
```

#### 4. Load Kqlmagic:

```
%reload_ext Kqlmagic
```

##### NOTE

If this step fails, then close the file and reopen it.



#### 5. You can test if Kqlmagic is loaded properly by browsing the help documentation or by checking for the version.

```
%kql --help "help"
```

##### NOTE

If `Samples@help` is asking for a password, then you can leave it blank and press **Enter**.

```
1 %kql --help "help"
```

## Overview

Help command is a tool to get more information on a topics that are relevant to Kqlmagic. t usage: %kql --help "topic"

## Topics

- **usage** - How to use Kqlmagic.

-**config** - Lists Kqlmagic default options. The same as --config without parameters.

- **faq** - Reference to Kqlmagic FAQ

- **conn** - Lists the available connection string variation, and how their are used to authenticatie to data sources.

- **query / kql** - [Reference to resources Kusto Query language, aka kql, documentation](#)

- **options** - Lists the available options, and their behavior impact on the submit query command.

- **commands** - Lists the available commands, and what they do.

- **proxies** - How to use Kqlmagic via proxies.

To see which version of Kqlmagic is installed, run the command below.

```
%kql --version
```

## Kqlmagic with an Azure Data Explorer cluster

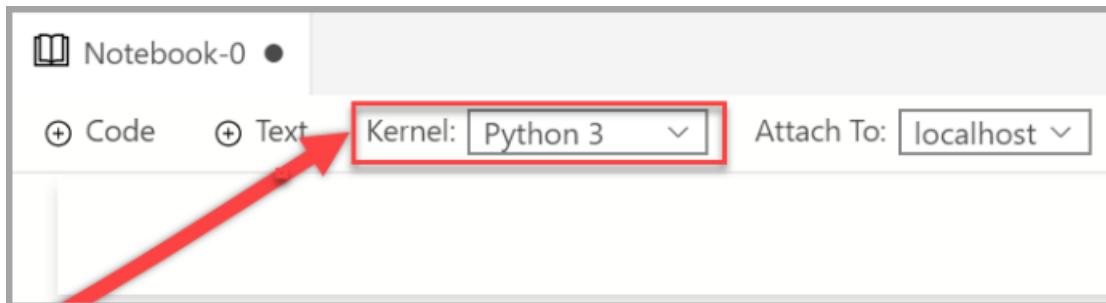
This section explains how to run data analysis using Kqlmagic with an Azure Data Explorer cluster.

### Load and authenticate Kqlmagic for Azure Data Explorer

#### NOTE

Every time you create a new notebook in Azure Data Studio you must load the Kqlmagic extension.

1. Verify the **Kernel** is set to *Python3*.



2. Load Kqlmagic:

```
%reload_ext Kqlmagic
```

The screenshot shows the Kqlmagic extension interface in Azure Data Studio. At the top, there's a toolbar with a play button and the text "%reload\_ext Kqlmagic". Below the toolbar is a large blue header with the "kql" logo. The main content area contains text about the Kql Query Language, its reference, configuration, and usage. It also mentions the package version (0.1.111.post11) and source (https://github.com/Microsoft/jupyter-Kqlmagic). A small note at the bottom says "Click to find out what's new in Kqlmagic" with a "What's New?" link.

### 3. Connect to the cluster and authenticate:

```
%kql azureDataExplorer://code;cluster='help';database='Samples'
```

#### NOTE

If you are using your own ADX cluster, you must include the region in the connection string as follows:

```
%kql azuredataexplorer://code;cluster='mycluster.westus';database='mykustodb'
```

You use Device Login to authenticate. Copy the code from the output and select **authenticate** which opens a browser where you need to paste the code. Once you authenticate successfully, you can come back to Azure Data Studio to continue with the rest of the script.

The screenshot shows the Azure Data Studio terminal window. The command "%kql azureDataExplorer://code;cluster='help';database='Samples'" is entered. A red box highlights the output "EBYXEEGR2" followed by the instruction "Copy code to clipboard and authenticate". Below the terminal, a message "popup schema Samples@help" is shown in a separate box.

## Query and visualize for Azure Data Explorer

Query data using the [render operator](#) and visualize data using the ploy.ly library. This query and visualization supplies an integrated experience that uses native KQL.

### 1. Analyze top 10 storm events by state and frequency:

```
%kql StormEvents | summarize count() by State | sort by count_ | limit 10
```

If you're familiar with the Kusto Query Language (KQL), you can type the query after `%kql`.

```
1 %kql StormEvents | summarize count() by state | sort by count_ | limit 10
```

\* Samples@help



	State	count_
1	TEXAS	4701
2	KANSAS	3166
3	IOWA	2337
4	ILLINOIS	2022
5	MISSOURI	2016
6	GEORGIA	1983
7	MINNESOTA	1881
8	WISCONSIN	1850
9	NEBRASKA	1766
10	NEW YORK	1750

Done (00:00.253): 10 records

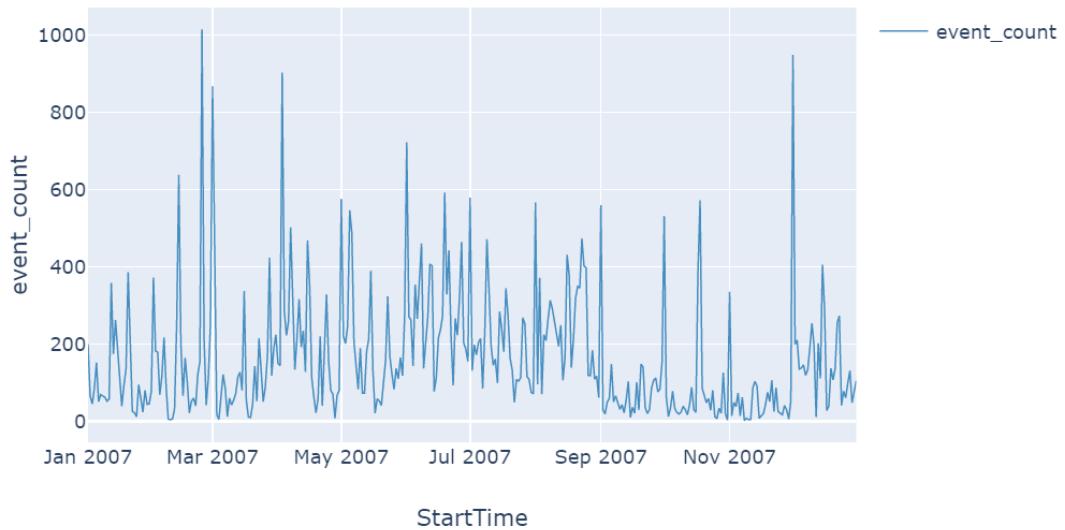
## 2. Visualize a timeline chart:

```
%kql StormEvents \
| summarize event_count=count() by bin(StartTime, 1d) \
| render timechart title= 'Daily Storm Events'
```

```
1 %%kql StormEvents \
2 | summarize event_count=count() by bin(StartTime, 1d) \
3 | render timechart title= 'Daily Storm Events'
```

\* Samples@help

## Daily Storm Events



Done (00:00:251): 365 records

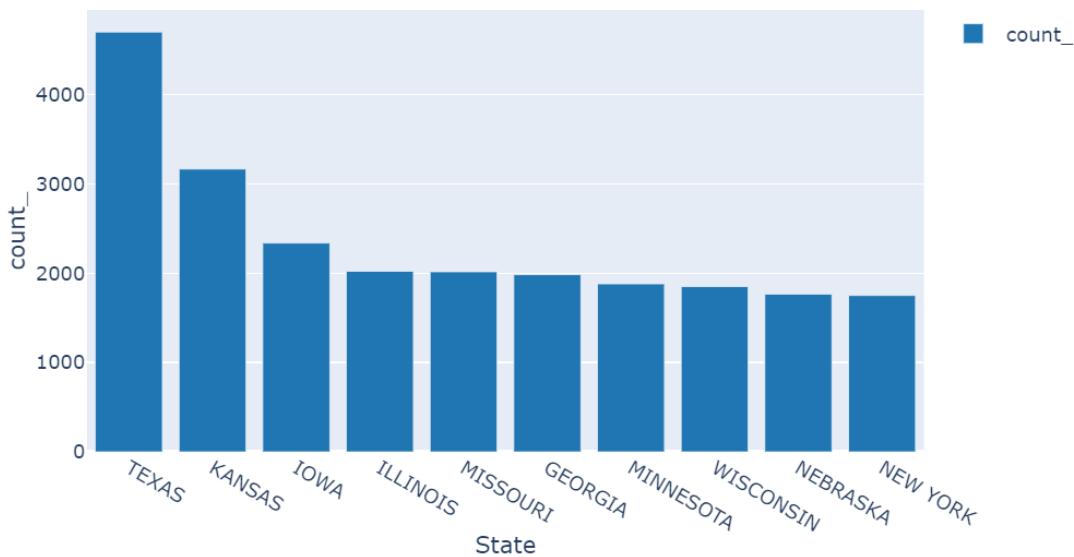
3. Multiline Query sample using `%%kql`.

```
%%kql
StormEvents
| summarize count() by State
| sort by count_
| limit 10
| render columnchart title='Top 10 States by Storm Event count'
```

```
1 %%kql
2   StormEvents
3   | summarize count() by state
4   | sort by count_
5   | limit 10
6   | render columnchart title='Top 10 States by Storm Event count'
```

\* Samples@help

Top 10 States by Storm Event count

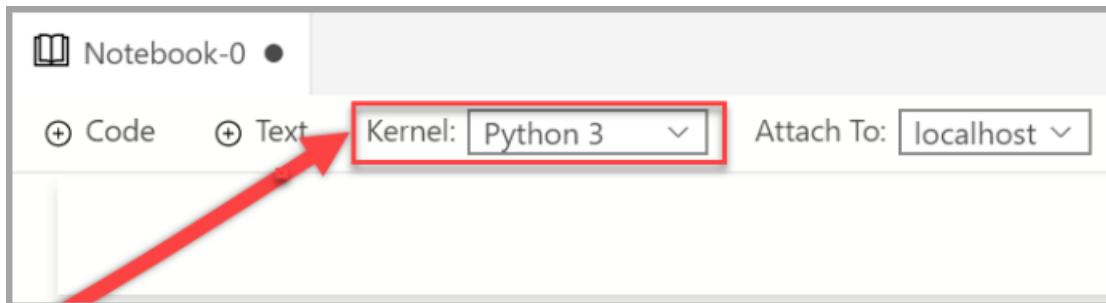


Done (00:01.199): 10 records

## Kqlmagic with Application Insights

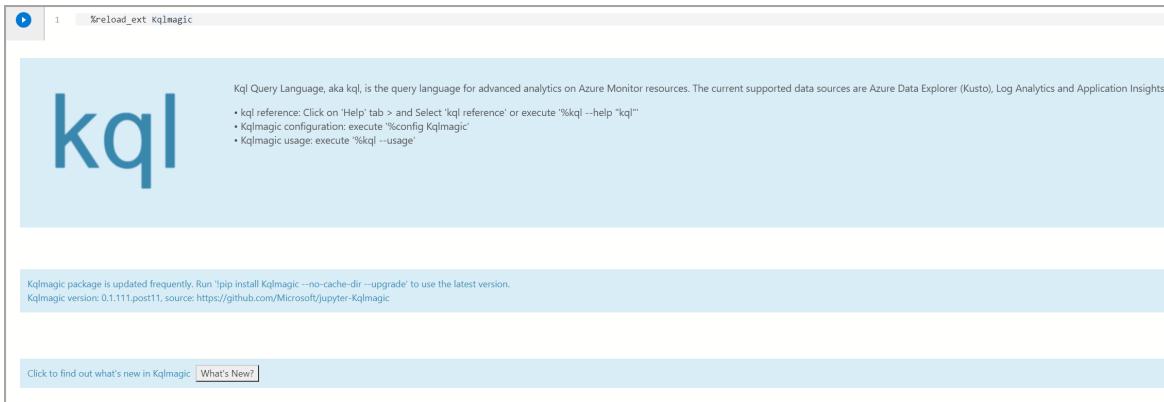
### Load and authenticate Kqlmagic for Application Insights

1. Verify the Kernel is set to *Python3*.



2. Load Kqlmagic:

```
%reload_ext Kqlmagic
```



### NOTE

Every time you create a new notebook in Azure Data Studio you must load the Kqlmagic extension.

### 3. Connect and authenticate.

First, you must generate an API key for your Application Insights resource. Then, use the Application ID and API key to connect to Application Insights from the notebook:

```
%kql appinsights://appid='DEMO_APP';appkey='DEMO_KEY'
```

### Query and visualize for Application Insights

Query data using the [render operator](#) and visualize data using the ploy.ly library. This query and visualization supplies an integrated experience that uses native KQL.

#### 1. Show Page Views:

```
%%kql  
pageViews  
| limit 10
```

```
1 %%kql
2 pageViews
3 | limit 10
```

\* DEMO\_APP@applicationinsights

The screenshot shows a table with 10 rows of data. The columns are timestamp, id, name, url, duration, performanceBucket, itemType, and customDimensions. All rows show 'Home Page' as the name and 'pageView' as the itemType. The timestamp column shows dates from January 11, 2020, at 09:46:37.524Z to 09:50:00.024Z.

	timestamp	id	name	url	duration	performanceBucket	itemType	customDimensions
1	2020-01-11T09:46:37.524Z		Home Page		null		pageView	None
2	2020-01-11T09:47:45.024Z		Home Page		null		pageView	None
3	2020-01-11T09:44:00.041Z		Tickets		null		pageView	None
4	2020-01-11T09:05:00.024Z		Home Page		null		pageView	None
5	2020-01-11T09:50:00.041Z		Home Page		null		pageView	None
6	2020-01-11T09:47:20.024Z		Home Page		null		pageView	None
7	2020-01-11T09:50:00.041Z		Home Page		null		pageView	None
8	2020-01-11T09:50:00.024Z		Home Page		null		pageView	None
9	2020-01-11T09:48:45.041Z		Home Page		null		pageView	None
10	2020-01-11T09:50:00.024Z		Home Page		null		pageView	None

Done (00:00:705): 10 records

#### NOTE

Use your mouse to drag on an area of the chart to zoom in to the specific date(s).

## 2. Show Page views in a timeline chart:

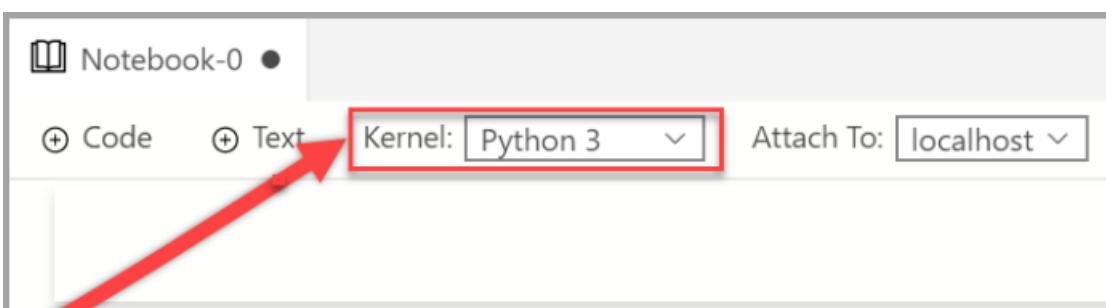
```
%%kql
pageViews
| summarize event_count=count() by name, bin(timestamp, 1d)
| render timechart title= 'Daily Page Views'
```



## Kqlmagic with Azure Monitor logs

### Load and authenticate Kqlmagic for Azure Monitor logs

1. Verify the Kernel is set to *Python3*.



2. Load Kqlmagic:

```
%reload_ext Kqlmagic
```

The screenshot shows the Kqlmagic extension's help page in Azure Data Studio. It features a large 'kql' logo at the top left. Below it, there's a section with bullet points about Kql Query Language and its configuration. A note at the bottom says 'Kqlmagic package is updated frequently. Run 'pip install Kqlmagic --no-cache-dir --upgrade' to use the latest version.' There's also a 'What's New?' link.

```
1 %reload_ext Kqlmagic
```

Kql Query Language, aka kql, is the query language for advanced analytics on Azure Monitor resources. The current supported data sources are Azure Data Explorer (Kusto), Log Analytics and Application Insights.  
• kql reference: Click on 'Help' tab > and Select 'kql reference' or execute '%kql --help "kql"'  
• Kqlmagic configuration: execute '%config Kqlmagic'  
• Kqlmagic usage: execute '%kql --usage'

Kqlmagic package is updated frequently. Run 'pip install Kqlmagic --no-cache-dir --upgrade' to use the latest version.  
Kqlmagic version: 0.1.111.post11, source: <https://github.com/Microsoft/jupyter-Kqlmagic>

Click to find out what's new in Kqlmagic [What's New?](#)

### NOTE

Every time you create a new notebook in Azure Data Studio you must load the Kqlmagic extension.

### 3. Connect and authenticate:

```
%kql loganalytics://workspace='DEMO_WORKSPACE';appkey='DEMO_KEY';alias='myworkspace'
```

```
1 %kql loganalytics://workspace='DEMO_WORKSPACE';appkey='DEMO_KEY';alias='myworkspace'|
```

```
popup schema myworkspace@loganalytics
```

## Query and visualize for Azure Monitor Logs

Query data using the [render operator](#) and visualize data using the ploy.ly library. This query and visualization supplies an integrated experience that uses native KQL.

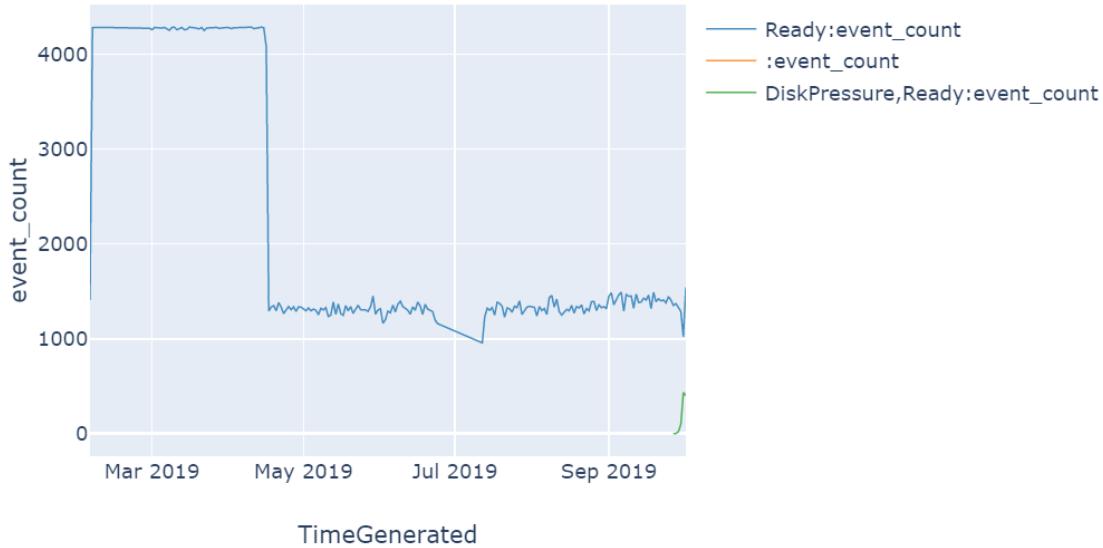
### 1. View a timeline chart:

```
%%kql
KubeNodeInventory
| summarize event_count=count() by Status, bin(TimeGenerated, 1d)
| render timechart title= 'Daily Kubernetes Nodes'
```

```
1 %%kql
2     KubeNodeInventory
3     | summarize event_count=count() by Status, bin(TimeGenerated, 1d)
4     | render timechart title= 'Daily Kubernetes Nodes'
```

DEMO\_APP@applicationinsights  
Samples@help  
\* myworkspace@loganalytics

## Daily Kubernetes Nodes



Done (00:00:840): 234 records

## Next steps

Learn more about notebooks and Kqlmagic:

- [Kusto \(KQL\) extension for Azure Data Studio \(Preview\)](#)
- [Create and run a Kusto \(KQL\) notebook \(Preview\)](#)
- [Use a Jupyter Notebook and Kqlmagic extension to analyze data in Azure Data Explorer](#)
- [Extension \(Magic\) to Jupyter Notebook and Jupyter lab, that enable notebook experience working with Kusto, Application Insights, and LogAnalytics data.](#)
- [Kqlmagic](#)
- [How to use notebooks in Azure Data Studio](#)

# Create and run a Python notebook

9/26/2022 • 2 minutes to read • [Edit Online](#)

Applies to: SQL Server 2019 (15.x)

This tutorial demonstrates how to create and run a notebook in Azure Data Studio using the Python kernel.

## Prerequisites

- [Azure Data Studio installed](#)

## Create a notebook

The following steps show how to create a notebook file in Azure Data Studio:

1. Open Azure Data Studio. If you're prompted to connect to a SQL Server, you may connect or click **Cancel**.
2. Select **New Notebook** in the **File** menu.
3. Select **Python 3** for the Kernel. **Attach to** is set to "localhost".



You can save the notebook using the **Save** or **Save as...** command from the **File** menu.

To open a notebook, you can use the **Open file...** command in the **File** menu, select **Open file** on the **Welcome** page, or use the **File: Open** command from the command palette.

## Change the Python kernel

The first time you connect to the Python kernel in a notebook, the **Configure Python for Notebooks** page is displayed. You can select either:

- **New Python installation** to install a new copy of Python for Azure Data Studio, or
- **Use existing Python installation** to specify the path to an existing Python installation for Azure Data Studio to use

To view the location and version of the active Python kernel, create a code cell and run the following Python commands:

```
import os
import sys
print(sys.version_info)
print(os.path.dirname(sys.executable))
```

To connect to a different installation of Python:

1. From the **File** menu, select **Preferences** and then **Settings**.
2. Scroll to **Notebook configuration** under **Extensions**.
3. Under **Use Existing Python**, uncheck the option "Local path to a preexisting python installation used by Notebooks."

#### 4. Restart Azure Data Studio.

When Azure Data Studio starts and you connect to the Python kernel, the **Configure Python for Notebooks** page is displayed. You can choose to create a new Python installation or specify a path to an existing installation.

## Run a code cell

You can create cells containing SQL code that you can run in place by clicking the **Run cell** button (the round black arrow) to the left of the cell. The results are shown in the notebook after the cell finishes running.

For example:

1. Add a new Python code cell by selecting the **+Code** command in the toolbar.



2. Copy and paste the following example into the cell and click **Run cell**. This example does simple math and the result appears below.

```
a = 1
b = 2
c = a/b
print(c)
```

The code cell shows the following output:  
1 a = 1  
2 b = 2  
3 c = a/b  
4 print(c)  
5 |  
0.5

## Next steps

Learn more about notebooks:

- [Extend Python with Kqlmagic](#)
- [How to use notebooks in Azure Data Studio](#)
- [Create and run a SQL Server notebook](#)

# Install Python in an offline Windows environment

9/26/2022 • 2 minutes to read • [Edit Online](#)

This tutorial demonstrates how to install and use the Python kernel in an offline Windows environment with notebooks.

## Prerequisites

- [Azure Data Studio installed](#)

## Download Python and Dependencies

1. On a machine that has internet access, download the latest Azure Data Studio Python package from here: <https://go.microsoft.com/fwlink/?linkid=2163338>. Unzip the file into a local directory (for example: C:\azuredatascience-py).

### NOTE

The latest Azure Data Studio Python version is 3.8.10.

2. In a terminal, navigate to the Python directory.

```
cd C:\azuredatascience-py
```

3. Create a text file named **requirements.txt** with the following contents.

```
pandas>=0.24.2
jupyter>=1.0.0
sparkmagic>=0.12.9
powershell-kernel>=0.1.3
```

4. Create a sub directory named **wheelhouse**.

```
mkdir wheelhouse
```

5. Run the following command to download the required dependencies to the sub directory.

```
python.exe -m pip download -r requirements.txt -d wheelhouse
```

## Install Python on a machine that doesn't have internet access

1. On a machine that doesn't have internet access, copy the Python folder from above to a local directory (for example: C:\azuredatascience-py).
2. In a terminal, navigate to the Python folder.

```
cd C:\azuredatascience-py
```

3. Run the following to install the dependencies.

```
python.exe -m pip install -r requirements.txt --no-index --find-links wheelhouse
```

## Use the Python Installation in Azure Data Studio

1. Open Azure Data Studio
2. From the Command Palette, search for Configure Python for Notebooks.
3. In the Configure Python for Notebooks wizard, select *Use existing Python installation*, and browse to the installed Python location (for example: C:\azuredatascience-pyton).

Once the wizard is completed, open a new notebook and change the kernel to Python.

**NOTE**

The steps above can be repeated for any additional packages needed.

# Create and run a SQL Server notebook

9/26/2022 • 2 minutes to read • [Edit Online](#)

Applies to: SQL Server 2019 (15.x)

This tutorial demonstrates how to create and run a notebook in Azure Data Studio using SQL Server.

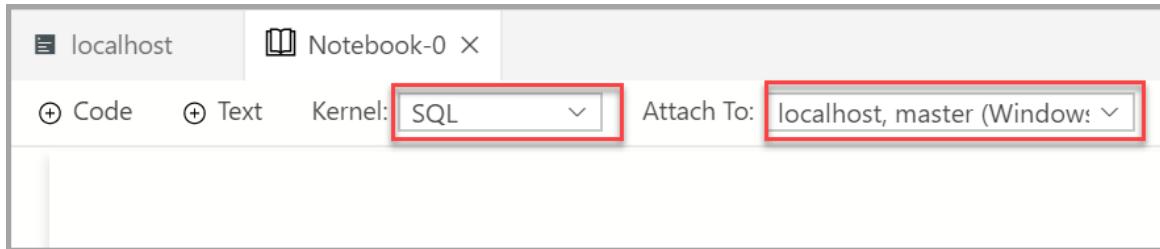
## Prerequisites

- [Azure Data Studio installed](#)
- SQL Server installed
  - [Windows](#)
  - [Linux](#)

## Create a notebook

The following steps show how to create a notebook file in Azure Data Studio:

1. In Azure Data Studio, connect to your SQL Server.
2. Select under the **Connections** in the **Servers** window. Then select **New Notebook**.
3. Wait for the **Kernel** and the target context (**Attach to**) to be populated. Confirm that the **Kernel** is set to **SQL**, and set **Attach to** for your SQL Server (in this example it's *localhost*).



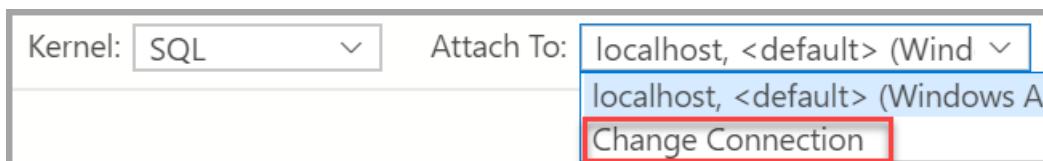
You can save the notebook using the **Save** or **Save as...** command from the **File** menu.

To open a notebook, you can use the **Open file...** command in the **File** menu, select **Open file** on the **Welcome** page, or use the **File: Open** command from the command palette.

## Change the SQL connection

To change the SQL connection for a notebook:

1. Select the **Attach to** menu from the notebook toolbar and then select **Change Connection**.



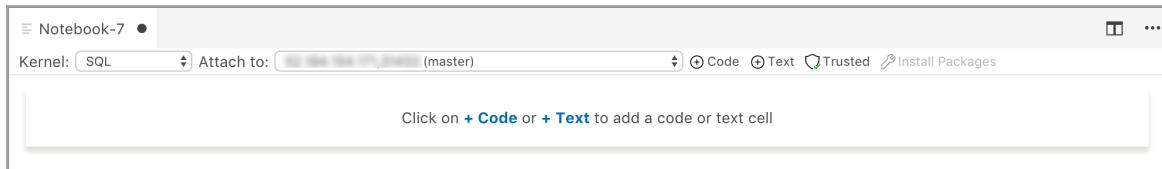
2. Now you can either select a recent connection server or enter new connection details to connect.

## Run a code cell

You can create cells containing SQL code that you can run in place by clicking the **Run cell** button (the round black arrow) to the left of the cell. The results are shown in the notebook after the cell finishes running.

For example:

1. Add a new code cell by selecting the **+Code** command in the toolbar.



2. Copy and paste the following example into the cell and click **Run cell**. This example creates a new database.

```
USE master
GO

-- Drop the database if it already exists
IF EXISTS (
    SELECT name
    FROM sys.databases
    WHERE name = N'TestNotebookDB'
)
DROP DATABASE TestNotebookDB
GO

-- Create the database
CREATE DATABASE TestNotebookDB
GO
```

The screenshot shows a Jupyter Notebook cell with numbered code lines. A red box highlights the first line, which contains a play button icon. The cell output below shows three lines of text: "Commands completed successfully." repeated twice, followed by "Total execution time: 00:00:01.114".

```
1 USE master
2 GO
3
4 -- Drop the database if it already exists
5 IF EXISTS (
6     SELECT name
7     FROM sys.databases
8     WHERE name = N'TestNotebookDB'
9 )
10 DROP DATABASE TestNotebookDB
11 GO
12
13 -- Create the database
14 CREATE DATABASE TestNotebookDB
15 GO
```

Commands completed successfully.  
Commands completed successfully.  
Commands completed successfully.  
Total execution time: 00:00:01.114

## Save the result

If you run a script that returns a result, you can save that result in different formats using the toolbar displayed above the result.

- Save As CSV
- Save As Excel
- Save As JSON
- Save As XML

For example, the following code returns the result of PI.

```
SELECT PI() AS PI;  
GO
```

[6]        1 SELECT PI() AS PI;  
            2 GO

(1 row affected)

Total execution time: 00:00:00.024



	PI
1	3.14159265358979

## Next steps

Learn more about notebooks:

- [How to use notebooks in Azure Data Studio](#)
- [Create and run a Python notebook](#)
- [Run a sample notebook using Spark](#)

# Create a parameterized notebook by using Papermill

9/26/2022 • 2 minutes to read • [Edit Online](#)

*Parameterization* in Azure Data Studio is running the same notebook with a different set of parameters.

This article shows you how to create and run a parameterized notebook in Azure Data Studio by using the Python kernel.

## NOTE

Currently, you can use parameterization with Python, PySpark, PowerShell, and .NET Interactive kernels.

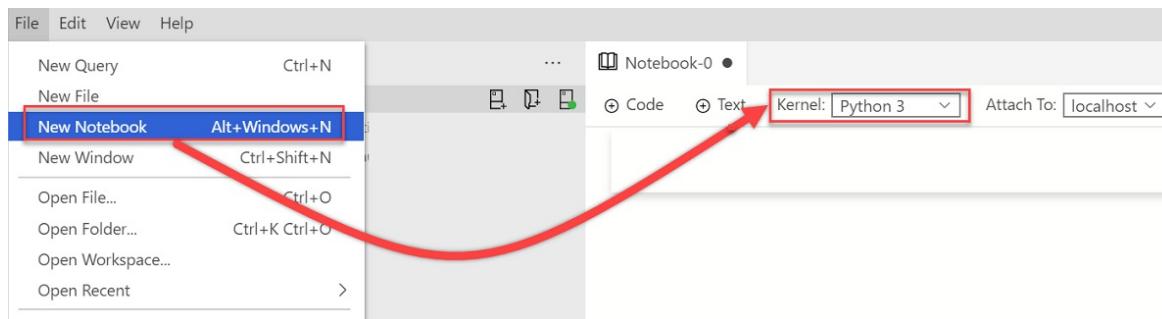
## Prerequisites

- [Azure Data Studio](#)
- [Python](#)

## Install and set up Papermill in Azure Data Studio

All the steps in this section run inside an Azure Data Studio notebook.

1. Create a new notebook. Change Kernel to Python 3:



2. If you're prompted to upgrade your Python packages when your packages need updating, select Yes:

Some installed python packages need to be upgraded. Would you like to upgrade them now?

Yes

No

3. Install Papermill:

```
import sys
!{sys.executable} -m pip install papermill --no-cache-dir --upgrade
```

Verify that Papermill is installed:

```
import sys
!{sys.executable} -m pip list
```

oset	0.1.3
packaging	20.4
pandas	1.1.4
pandocfilters	1.4.3
<b>papermill</b>	<b>2.2.2</b>
paramiko	2.7.2
parso	0.7.1
pathspec	0.8.1
pexpect	4.8.0
	...

4. To verify that Papermill installed correctly, check the version of Papermill:

```
import papermill  
papermill
```

```
<module 'papermill' from  
'/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-  
packages/papermill/__init__.py'>
```

## Parameterization example

You can use an [example notebook file](#) to go through the steps in this article:

1. Go to the [notebook file in GitHub](#). Select **Raw**.
2. Select Ctrl+S or right-click, and then save the file with the .ipynb extension.
3. Open the file in Azure Data Studio.

## Set up a parameterized notebook

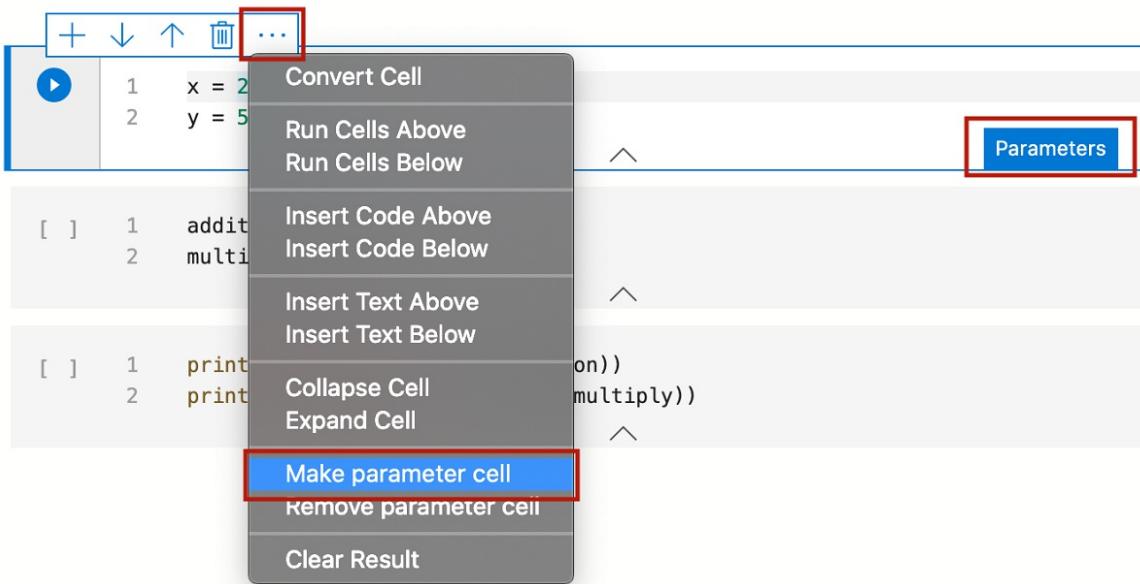
You can begin with the example notebook open in Azure Data Studio or complete the following steps to create a notebook. Then, try using different parameters. All the steps run inside an Azure Data Studio notebook.

1. Verify that **Kernel** is set to **Python 3**:



2. Make a new code cell. Select **Parameters** to tag the cell as a parameters cell.

```
x = 2.0  
y = 5.0
```



3. Add other cells to test different parameters:

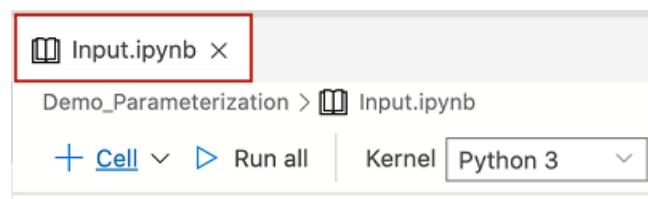
```
addition = x + y
multiply = x * y
```

```
print("Addition: " + str(addition))
print("Multiplication: " + str(multiply))
```

After all cells are run, the output will look similar to this example:

```
[1]   1   x = 2.0
      2   y = 5.0
      ^
[2]   1   addition = x + y
      2   multiply = x * y
      ^
[3]   1   print("Addition: " + str(addition))
      2   print("Multiplication: " + str(multiply))
      ^
Addition: 7.0
Multiplication: 10.0
```

4. Save the notebook as *Input.ipynb*



## Execute a Papermill notebook

You can execute Papermill in two ways:

- Command-line interface (CLI)
- Python API

## Parameterized CLI execution

To execute a notebook by using the CLI, in the terminal, enter the `papermill` command with the input notebook, the location for the output notebook, and options.

### NOTE

To learn more, see the [Papermill CLI documentation](#).

1. Execute the input notebook with new parameters:

```
papermill Input.ipynb Output.ipynb -p x 10 -p y 20
```

This command executes the input notebook with new values for parameters *x* and *y*.

2. A new cell labeled `# Injected-Parameters` contains the new parameter values that were passed in via the CLI. The new `# Injected-Parameters` values are used for the new output that's shown in the last cell:

```
[1] 1 x = 2.0
    2 y = 5.0
```

Parameters

```
[2] 1 # Injected-Parameters
    2 x = 10
    3 y = 20
    4
```

```
1 addition = x + y
2 multiply = x * y
```

```
[4] 1 print("Addition: " + str(addition))
    2 print("Multiplication: " + str(multiply))
```

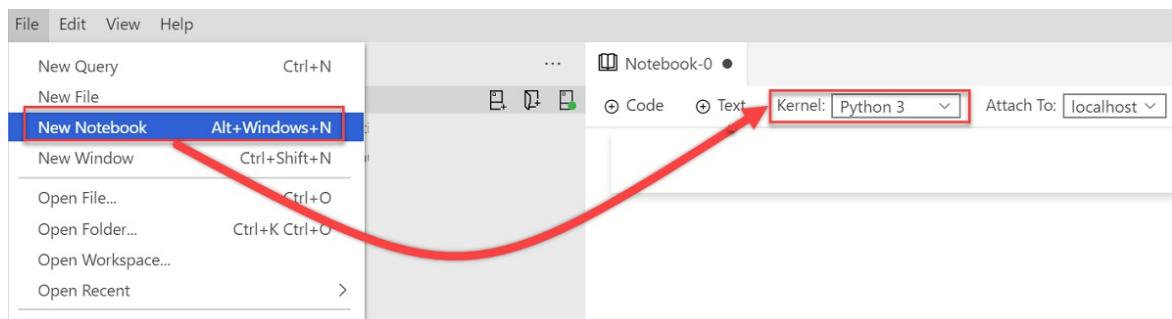
```
Addition: 30
Multiplication: 200
```

## Parameterized Python API execution

### NOTE

To learn more, see the [Papermill Python documentation](#).

1. Create a new notebook. Change Kernel to Python 3:



2. Add a new code cell. Then, use the Papermill Python API to execute and generate the output parameterized notebook:

```
import papermill as pm

pm.execute_notebook(
    '/Users/vasubhog/GitProjects/AzureDataStudio-Notebooks/Demo_Parameterization/Input.ipynb',
    '/Users/vasubhog/GitProjects/AzureDataStudio-Notebooks/Demo_Parameterization/Output.ipynb',
    parameters = dict(x = 10, y = 20)
)
```

A screenshot of the Azure Data Studio interface showing a code cell with the following content:

```
import papermill as pm

pm.execute_notebook(
    '/Users/vasubhog/GitProjects/AzureDataStudio-Notebooks/Demo_Parameterization/Input.ipynb',
    '/Users/vasubhog/GitProjects/AzureDataStudio-Notebooks/Demo_Parameterization/Output.ipynb',
    parameters = dict(x = 10, y = 20)
)
```

The cell has a play button icon and a line number 1. Above the cell are buttons for '+ Cell', 'Run all', 'Kernel: Python 3', and 'Attach to: localhost'. A red arrow points from the 'Kernel: Python 3' dropdown in the top interface to the 'Kernel: Python 3' dropdown in this screenshot.

3. A new cell labeled `# Injected-Parameters` contains the new parameter values that were passed in. The new `# Injected-Parameters` values are used for the new output that's shown in the last cell:

A screenshot of the Azure Data Studio interface showing three code cells:

- Cell 1: Prints `x = 2.0` and `y = 5.0`. It has a play button icon and a line number 1.
- Cell 2: Contains the code `# Injected-Parameters`, `x = 10`, and `y = 20`. It has a play button icon and a line number 1.
- Cell 3: Contains the code `addition = x + y` and `multiply = x * y`. It has a play button icon and a line number 1.

Below the second cell is a blue button labeled 'Parameters'. At the bottom of the interface, the output is displayed:

```
Addition: 30
Multiplication: 200
```

## Next steps

Learn more about notebooks and parameterization:

- [How to use notebooks in Azure Data Studio](#)
- [Papermill parameterization documentation](#)
- [URI parameterization](#)
- [Run with Parameters](#)

# Create a parameterized notebook by using the notebook URI

9/26/2022 • 2 minutes to read • [Edit Online](#)

*Parameterization* in Azure Data Studio is running the same notebook with a different set of parameters.

This article shows you how to create and run a parameterized notebook in Azure Data Studio by using the Python kernel.

## NOTE

Currently, you can use parameterization with Python, PySpark, PowerShell, and .NET Interactive kernels.

## Prerequisites

- [Azure Data Studio](#)
- [Python](#)

## URI parameterization

URI parameterization programmatically adds parameters to the query of the Azure Data Studio URI to open the notebook in Azure Data Studio with new parameters.

Azure Data Studio notebook URI supports HTTPS, HTTP, and FILE URI schema and uses this format:

```
azuredatastudio://microsoft.notebook/open?url=
```

To pass in parameters with an Azure Data Studio notebook URI, use this format:

```
azuredatastudio://microsoft.notebook/open?url=LinkToNotebook?x=1&y=2
```

In the URI query, use `&` to indicate a new parameter to be injected.

## Parameterization example

You can use an [example notebook file](#) to go through the steps in this article:

1. Go to the [notebook file in GitHub](#). Select **Raw**.
2. Select Ctrl+S or right-click, and then save the file with the .ipynb extension.
3. Open the file in Azure Data Studio.

Here are the contents and structure of the notebook:

```
x = 2.0
y = 5.0
```

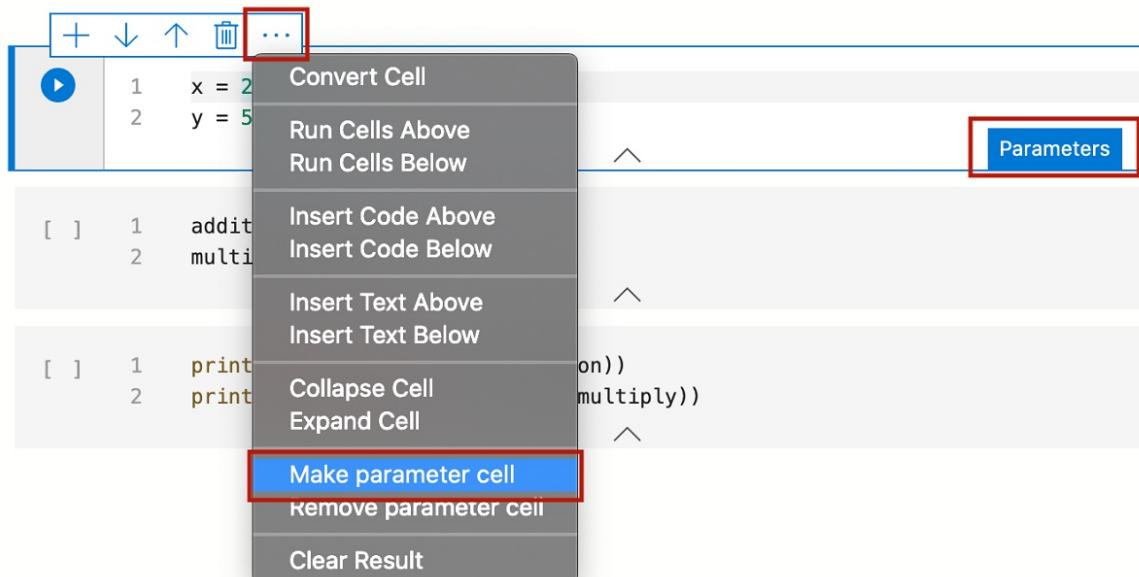
```
addition = x + y
multiply = x * y
```

```
print("Addition: " + str(addition))
print("Multiplication: " + str(multiply))
```

## Set up a notebook for parameterization

You can begin with the example notebook open in Azure Data Studio or complete the following steps to create a notebook. Then, try using different parameters. All the steps run inside an Azure Data Studio notebook.

1. The first code cell must be tagged with parameters. If you're using a new notebook file, make a new code cell. Select **Parameters** to tag the cell as a parameters cell.



2. You can use either the search bar of any browser or a Markdown cell to open the notebook URI link.

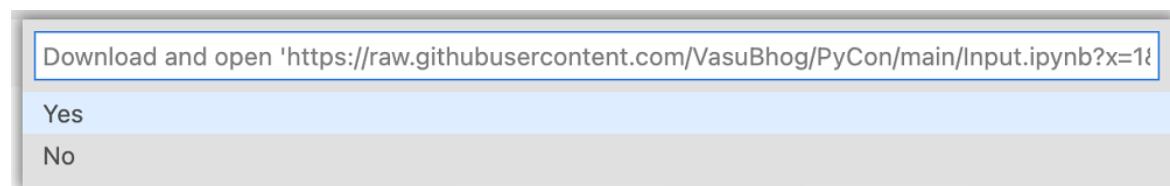
Using the Azure Data Studio notebook URI (shown next), you can parameterize the example input notebook on GitHub.

Copy the following notebook URI to parameterize the example input notebook on GitHub with new values for *x* and *y*. Paste the URI in a browser search bar:

```
azuredatrstudio://microsoft.notebook/open?url=https://raw.githubusercontent.com/microsoft/sql-server-samples/master/samples/applications/azure-data-studio/parameterization.ipynb?x=10&y=20
```



When you open the link from the web browser, you're prompted to open the notebook in Azure Data Studio. Select **Open Azure Data Studio**.



3. You're prompted to download and open the notebook with new parameters.

Select **Yes**, and then view the new parameterized notebook. On the toolbar, select **Run all** to run all cells and see the new output.

A new cell labeled `# Injected-Parameters` contains the new parameter values that were passed in:

```
[1] 1 x = 2.0  
2 y = 5.0
```

Parameters

```
[2] 1 # Injected-Parameters  
2 x = 10  
3 y = 20  
4
```

```
1 addition = x + y  
2 multiply = x * y
```

```
[4] 1 print("Addition: " + str(addition))  
2 print("Multiplication: " + str(multiply))
```

```
Addition: 30  
Multiplication: 200
```

## Next steps

Learn more about notebooks and parameterization:

- [How to use notebooks in Azure Data Studio](#)
- [Papermill parameterization](#)
- [Run with Parameters](#)

# Create a parameterized notebook by using the Run with Parameters action

9/26/2022 • 2 minutes to read • [Edit Online](#)

*Parameterization* in Azure Data Studio is running the same notebook with a different set of parameters.

This article shows you how to create and run a parameterized notebook in Azure Data Studio by using the Python kernel.

## NOTE

Currently, you can use parameterization with Python, PySpark, PowerShell, and .NET Interactive kernels.

## Prerequisites

- [Azure Data Studio](#)
- [Python](#)

## Run with Parameters action

When you use the Run with Parameters notebook action, the user can input new parameters in the UI to quickly set new parameters for your notebook. The user can then run the notebook with the new parameters.

## NOTE

It's important to format the parameter cell with each new parameter on a new line.

## Parameterization example

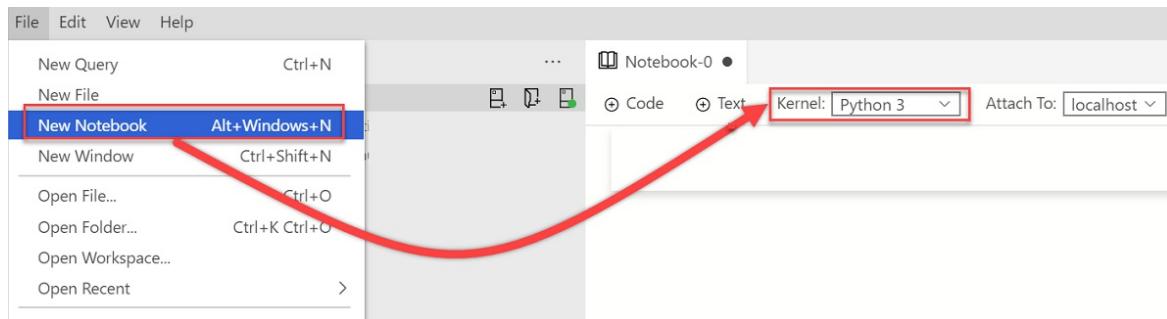
You can use an [example notebook file](#) to go through the steps in this article:

1. Go to the [notebook file in GitHub](#). Select **Raw**.
2. Select Ctrl+S or right-click, and then save the file with the .ipynb extension.
3. Open the file in Azure Data Studio.

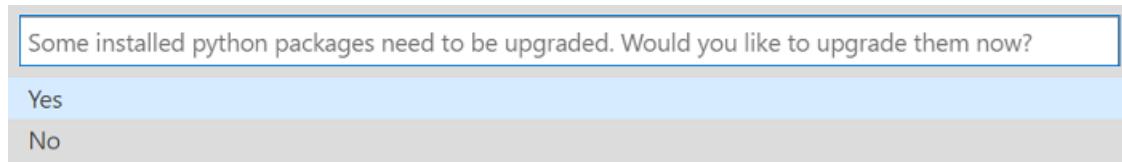
## Set up a notebook for parameterization

You can begin with the example notebook open in Azure Data Studio or complete the following steps to create a notebook. Then, try using different parameters. All the steps run inside an Azure Data Studio notebook.

1. Create a new notebook. Change Kernel to **Python 3**:



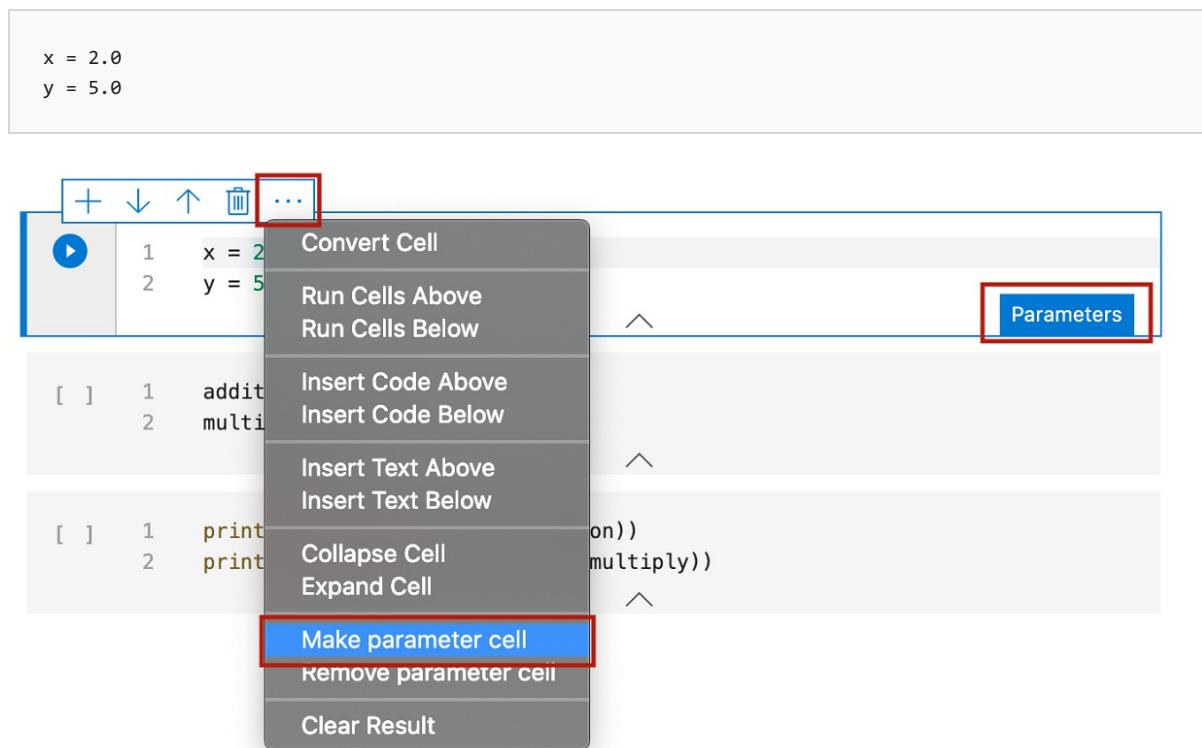
2. If you're prompted to upgrade your Python packages when your packages need updating, select **Yes**:



3. Verify that **Kernel** is set to **Python 3**:



4. Make a new code cell. Select **Parameters** to tag the cell as a parameters cell.



5. Add other cells to test different parameters:



The output will look similar to this example:

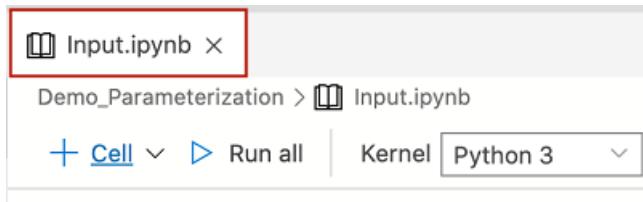
```

[1] 1 x = 2.0
    2 y = 5.0
    ^
Parameters

[2] 1 addition = x + y
    2 multiply = x * y
    ^
[3] 1 print("Addition: " + str(addition))
    2 print("Multiplication: " + str(multiply))
    ^
Addition: 7.0
Multiplication: 10.0

```

6. Save the notebook as *Input.ipynb*.

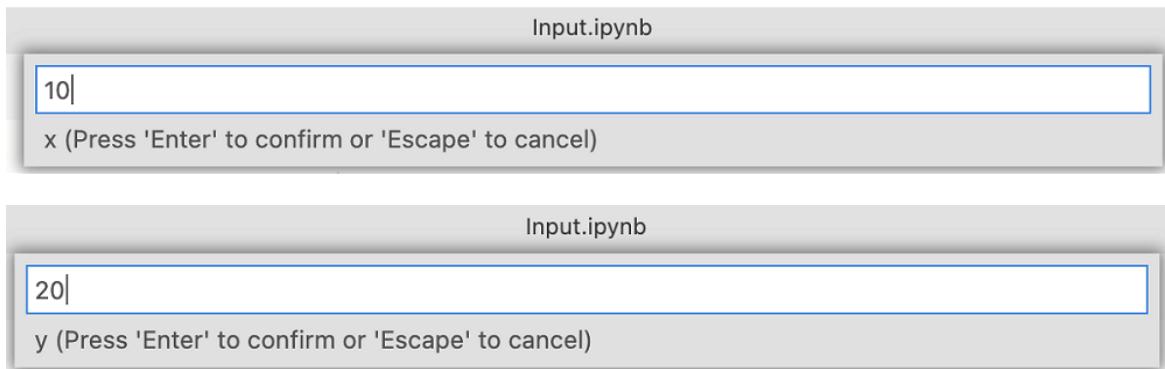


## Run the notebook with parameters

1. On the notebook toolbar, select the Run with Parameters icon:



2. A series of new dialogs prompt you to input new parameters for  $x$  and  $y$ :



3. After you enter the new parameters, view the new parameterized notebook. On the toolbar, select Run all to run all cells and see the new output. A new cell labeled '# Injected-Parameters' contains the new parameter values that were passed in:

```
[1] 1 x = 2.0
2 y = 5.0
```

Parameters

```
[2] 1 # Injected-Parameters
2 x = 10
3 y = 20
4
```

```
1 addition = x + y
2 multiply = x * y
```

```
[4] 1 print("Addition: " + str(addition))
2 print("Multiplication: " + str(multiply))
```

```
Addition: 30
Multiplication: 200
```

## Next steps

Learn more about notebooks and parameterization:

- [How to use notebooks in Azure Data Studio](#)
- [Papermill parameterization](#)
- [URI parameterization](#)

# Create an Azure SQL Database using Azure Data Studio

9/26/2022 • 2 minutes to read • [Edit Online](#)

You can create an Azure SQL Database using Azure Data Studio through the deployment wizard and notebooks.

## Pre-requisites

- [Azure Data Studio](#) is installed
- An active Azure account and subscription. If you do not have one, [create a free account](#).

## Use the deployment wizard

Follow these steps to use the deployment wizard, which will guide you through the required settings in a simple UI experience.

First, find and select Azure SQL Database in the deployment wizard.

1. In Azure Data Studio, select the Connections viewlet on the left-side navigation.
2. Select the ... button at the top of the Connections panel and choose **New Deployment...**
3. In the deployment wizard, select the **Azure SQL Database** tile and check the terms acceptance checkbox
4. In the Resource type dropdown, select what you would like to create - either a database, database server, or elastic pool. If you don't have any SQL databases in Azure, we recommend starting by creating a database.
5. Select **Create in Azure portal** if you chose to create a server or pool or select **Select** if you chose to create a database.

If you chose to create a database, follow the steps below.

1. Sign in to your Azure account if you have not already. You can refresh your connection if you have issues on this page of the wizard.
2. Select your desired subscription and server. Then select **Next**.
3. Enter a database name.
4. Enter a firewall rule name and the IP range of your local client machine running Azure Data Studio. This is so you can connect to the server and database from ADS right after it has been created.
5. Select **Next**.
6. Review the parameters you have entered and then select **Script to Notebook**.

Once the Notebook opens, you can review the content and the code and make changes if you like. In particular you can change the compute and storage settings from the default if you have specific performance or cost preferences. However, be aware that any changes you make to the Notebook could potentially cause validation errors.

The last step is to select **Run all** to run all cells in the Notebook. Once this completes you should have a fully created and running SQL Database that you can connect to and query from ADS.

## Next steps

After you've created your database, server, or pool, you can [connect and query](#) your database in Azure Data Studio.

# Deploy Azure SQL Edge with Azure Data Studio (Preview)

9/26/2022 • 3 minutes to read • [Edit Online](#)

Azure SQL Edge is a relational database engine optimized for IoT and Azure IoT Edge deployments. It provides capabilities to create a high-performance data storage and processing layer for IoT applications and solutions. This article shows you how to deploy an Azure SQL Edge instance with Azure Data Studio and the deployment scenarios that are supported with the deployment wizard.

The following scenarios are supported by the deployment wizard in Azure Data Studio:

- Local container instance
- Remote container instance
- New Azure IoT Hub and VM
- Existing Device of an Azure IoT Hub
- Multiple Devices of an Azure IoT Hub

REQUIRED TOOLS	DOCKER	AZURE CLI
Local container instance	x	
Remote container instance		
New Azure IoT Hub and VM		x
Existing Device of an Azure IoT Hub		x
Multiple Devices of an Azure IoT Hub		x

## NOTE

Azure SQL Edge deployment (preview) is available through the extension "Azure SQL Edge Deployment Extension" while these features are in preview. To make the features available to you, please install the extension in Azure Data Studio.

To begin a deployment in Azure Data Studio, open the menu in the **Connections** view and select the **New Deployment...** option. For all Azure SQL Edge scenarios, select **Azure SQL Edge** on the following screen and the desired scenario from the **Deployment target** drop-down. The deployment wizard generates a TSQL notebook that can be executed to complete the deployment. It should be noted that connection information, including the SQL admin password, is contained in the notebook by default.

Select the deployment options

Azure SQL Edge (Preview) is an optimized relational database engine geared for IoT and IoT Edge deployments.

I accept Microsoft Privacy Statement and Microsoft Azure SQL Edge License Agreement. \*

Options

Deployment target: Local container instance

Required tools

Tool	Description	Status	Version	Required Version	Discovered Path or Additional Information
docker	Packages and runs applications in isolated contain...	Installed	19.3.12		

## Local container instance

Azure SQL Edge can be deployed to a Docker container on the localhost by specifying the container name, `sa` user password, and the host port for Azure SQL Edge connectivity. After completing the deployment, several links are available at the bottom of the notebook for further actions:

- **Select here to connect to the Azure SQL Edge instance:** Creates a connection in Azure Data Studio to the new Azure SQL Edge instance
- **Open the terminal window:** Opens a terminal (existing or new) in Azure Data Studio
- **Stop the container:** Copies a command into the terminal that stops the container when executed by the user
- **Remove the container:** Copies a command into the terminal that removes the container when executed by the user

## Remote container instance

Azure SQL Edge can be deployed to a Docker container on a remote host with Docker installed by specifying the container information and the target/host machine information. After completing the deployment, a connection link is available at the bottom of the notebook. Because of the remote container host environment, to stop or remove the container, commands must be copied to execute in a remote shell.

## New Azure IoT Hub and VM

The Azure SQL Edge deployment wizard can create several Azure resources needed to deploy an edge-enabled virtual machine (VM) connected to an Azure IoT hub. An active Azure subscription is required. This deployment creates:

- Resource group (if current resource group isn't selected)
- Network security group
- Virtual machine
- Static public IP address

Optionally, a dacpac file can be zipped in a folder and deployed to the new Azure SQL Edge instance as a part of the process. If a dacpac file is provided, an Azure Blob Storage account is created in the same resource group.

## Existing Device of an Azure IoT Hub

If you have an existing IoT hub and a connected device, Azure SQL Edge can be deployed to the device based on the resource group, IoT hub name, and the device ID. The IP address provided during the deployment wizard is utilized to generate a quick connect link at the bottom of the notebook.

Optionally, a dacpac file can be zipped in a folder and deployed to the new Azure SQL Edge instance as a part of the process. If a dacpac file is provided, an Azure Blob Storage account is created in the same resource group.

## Multiple Devices of an Azure IoT Hub

If you have an existing IoT hub and connected devices, Azure SQL Edge can be deployed to the device based on the resource group, IoT hub name, and a [target condition](#) to select device(s). The IP address provided during the deployment wizard is utilized to generate a quick connect link at the bottom of the notebook.

Optionally, a dacpac file can be zipped in a folder and deployed to the new Azure SQL Edge instance as a part of the process. If a dacpac file is provided, an Azure Blob Storage account is created in the same resource group.

## Next steps

- [Learn more about Azure SQL Edge](#)

# Create SQL Server on Azure Virtual Machines using Azure Data Studio

9/26/2022 • 2 minutes to read • [Edit Online](#)

You can create an SQL virtual machine (VM) using Azure Data Studio through the deployment wizard and notebooks.

## Pre-requisites

- [Azure Data Studio](#) is installed
- An active Azure account and subscription. If you don't have one, [create a free account](#).

## Use the deployment wizard

Follow these steps to use the deployment wizard, which will guide you through the required settings in a simple UI experience.

First, find, and select Azure SQL VM in the deployment wizard.

1. In Azure Data Studio, select the Connections viewlet on the left-side navigation.
2. Select the ... button at the top of the Connections panel and choose **New Deployment**.
3. In the deployment wizard, select the **Azure SQL VM** tile and check the terms acceptance checkbox
4. If prompted to, install the required tools and then select the **Select** button at the bottom.

Next, enter all of the required parameters in the Azure SQL VM wizard.

1. Sign in to your Azure account if you'ven't already. You can refresh your connection if you've issues on this page of the wizard.
2. Select your desired subscription, resource group, and region. Then select **Next**.
3. Enter a unique Virtual Machine name and your username and password credentials.
4. Select your preferred image, SKU, and version, and then select your preferred VM size. You can learn more about [available VM sizes](#) to help you make your selection. Then select **Next**.
5. Either select an existing virtual network from the dropdown, or check the **New virtual network** checkbox to enter a name for a new virtual network.
6. Do the same for selecting or creating a subnet and public IP address.
7. If you would like to connect to your VM via Remote Desktop (RDP), check the **Enable RDP inbound port** checkbox. Then select **Next**.
8. Enter your preferred SQL Server settings. The recommendation for testing out this experience is to set SQL connectivity to **Public (internet)**, enter port 1433, and enable SQL authentication with your preferred username and password. Then select **Next**.
9. Review the parameters you've entered and then select **Script to Notebook**.

Once the Notebook opens, you can review the content and the code and make changes if you like. However making changes isn't recommended since it could cause validation errors.

The last step is to select **Run all** to run all cells in the Notebook. Once this completes, you should have a fully created and running:

- An Azure virtual machine
- A SQL virtual machine
- A Virtual Network, subnet, and public IP address
- A network security group and a network interface
- Access to RDP into the VM
- Access to a variety of SQL Server manageability features to easily control the Backup schedule, patching schedule, the SQL Server edition and licensing, the storage performance configurations, and much more.

## Next steps

To learn more about how to migrate your data to the new SQL VM, see the following article.

[Migrate a database to a SQL VM](#)

For other information about using SQL Server in Azure, see [SQL Server on Azure Virtual Machines](#) and the [Frequently Asked Questions](#).

# Deploy SQL Server in a container using Azure Data Studio

9/26/2022 • 2 minutes to read • [Edit Online](#)

You can deploy SQL Server in a container using Azure Data Studio through the deployment wizard and notebooks.

## Pre-requisites

- [Azure Data Studio](#) is installed
- Docker Engine 1.8+ on any supported Linux distribution or Docker for Mac/Windows. For more information, see [Install Docker](#)

## Use the deployment wizard

Follow these steps to use the deployment wizard, which will guide you through the required settings in a simple UI experience.

First, find and select Azure SQL Database in the deployment wizard.

1. In Azure Data Studio, select the Connections viewlet on the left-side navigation.
2. Select the ... button at the top of the Connections panel and choose **New Deployment...**
3. In the deployment wizard, select the **SQL Server container image** tile and select **Select** to move to the next screen.
4. In the Version dropdown, select which image of SQL Server you would like to deploy. The wizard will check whether you have a running instance of Docker at this time. Select **Select** to move to the next screen.
5. Enter the container name, a password for the *sa* account of suitable complexity (at least eight characters long and contain characters from three of the following four sets: uppercase letters, lowercase letters, base-10 digits, and symbols), and the port in the host environment which the container will map to.
6. Review the parameters you have entered and then select **Open Notebook**.

Once the Notebook opens, you can review the content and the code and make changes if you like. However, be aware that any changes you make to the Notebook could potentially cause validation errors.

The last step is to select **Run all** to run all cells in the Notebook. Once this completes you should have a fully created and running SQL Database that you can connect to and query from ADS.

## Next steps

After you've created your database, server, or pool, you can [connect and query](#) your database in Azure Data Studio.

For more information on deploying SQL Server in containers, see [Quickstart: Run SQL Server Linux container images with Docker](#).

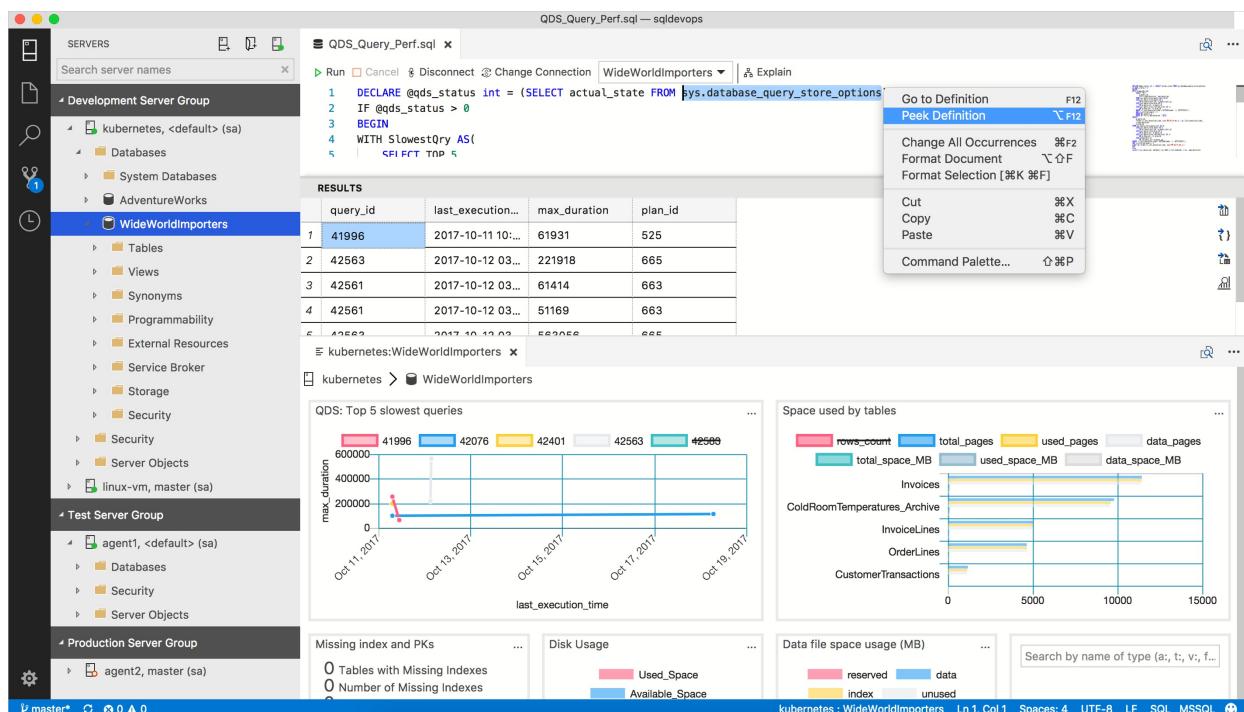
# Manage servers and databases with Insight widgets in Azure Data Studio

9/26/2022 • 2 minutes to read • [Edit Online](#)

Insight widgets take the Transact-SQL (T-SQL) queries you use to monitor servers & databases and turns them into insightful visualizations.

Insights are customizable charts and graphs that you add to server and database monitoring dashboards. View at-a-glance insights of your servers and databases, then drill into more details, and launch management actions that you define.

You can build awesome server and database management dashboards similar to the following example:



To jump in and start creating different types of insight widgets, check out the following tutorials:

- [Build a custom insight widget](#)
- [Enable built-in insight widgets](#)
  - [Enable the performance monitoring insight](#)
  - [Enable the table space usage insight](#)

## SQL Queries

Azure Data Studio tries to avoid introducing yet another language or heavy user interface so it tries to use T-SQL as much as possible with minimal JSON configuration. Configuring insight widgets with T-SQL leverages the countless number of existing sources of useful T-SQL queries that can be turned into insightful widgets.

Insight widgets are composed of one or two T-SQL queries:

- *Insight widget query* is mandatory, and is the query that returns the data that appears in the widget.\*
- *Insight details query* is only required if you are creating an insight details page.\*

An insight widget query defines a dataset that renders a count, chart, or graph. Insight details query is used to

list relevant insight detail information in a tabular format in the insight details panel.

Azure Data Studio executes insight widget queries and maps the query result set to a chart's dataset then renders it. When users open up an insight's details, it executes the insight details query and prints out the result in a grid view within the dialog.

The basic idea is to write a T-SQL query in a way so it can be used as a dataset of a count, chart, and graph widget.

## Summary

The T-SQL query and its result set determine the insight widget behavior. Writing a query for a chart type or mapping a right chart type for existing query is the key consideration to build an effective insight widget.

## Additional resources

- [Query Editor](#)

# Create and use code snippets to quickly create Transact-SQL (T-SQL) scripts in Azure Data Studio

9/26/2022 • 2 minutes to read • [Edit Online](#)

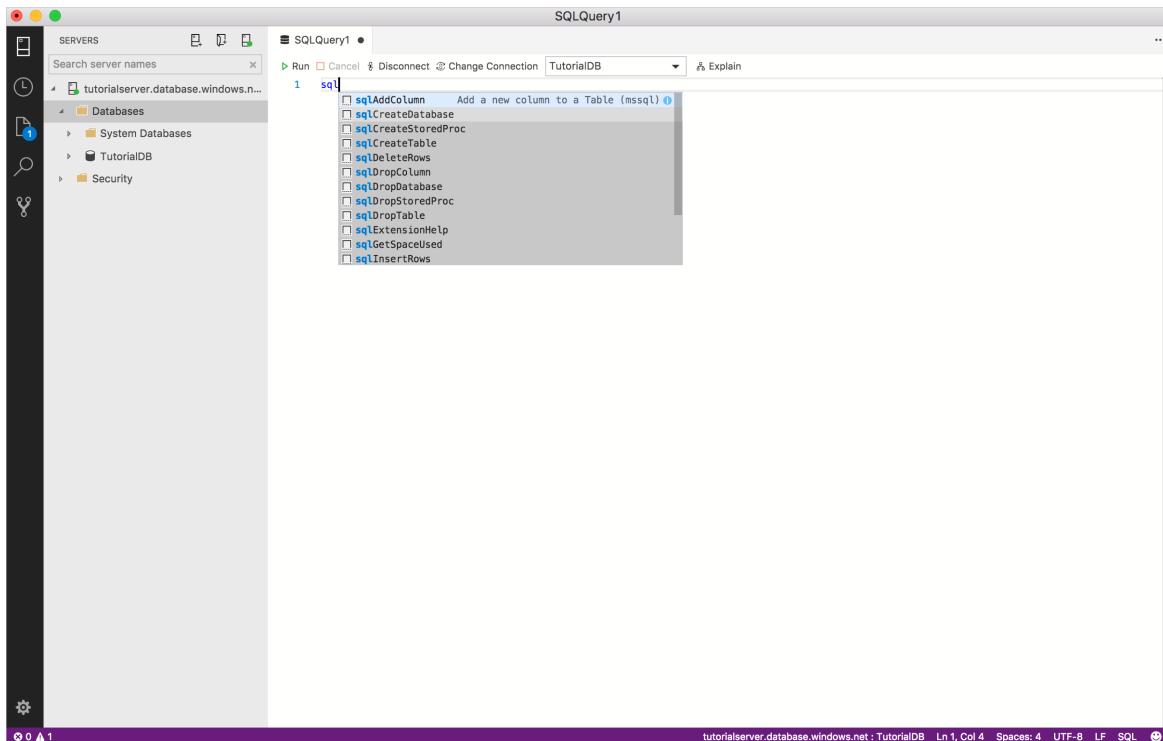
Code snippets in Azure Data Studio are templates that make it easy to create databases and database objects.

Azure Data Studio provides several T-SQL snippets to assist you with quickly generating the proper syntax.

User-defined code snippets can also be created.

## Using built-in T-SQL code snippets

1. To access the available snippets, type `sq`/in the query editor to open the list:



2. Select the snippet you want to use, and it generates the T-SQL script. For example, select `sqlCreateTable`.

```
1 -- Create a new table called 'TableName' in schema 'SchemaName'
2 -- Drop the table if it already exists
3 IF OBJECT_ID('SchemaName.TableName', 'U') IS NOT NULL
4 DROP TABLE SchemaName.TableName
5 GO
6 -- Create the table in the specified schema
7 CREATE TABLE SchemaName.TableName
8 (
9     TableNameId INT NOT NULL PRIMARY KEY, -- primary key column
10    Column1 NVARCHAR(50) NOT NULL,
11    Column2 NVARCHAR(50) NOT NULL
12    -- specify more columns here
13 );
14 GO
```

3. Update the highlighted fields with your specific values. For example, replace *TableName* and *Schema* with the values for your database:

```
1 -- Create a new table called 'TableFromSnippet' in schema 'dbo'
2 -- Drop the table if it already exists
3 IF OBJECT_ID('dbo.TableFromSnippet', 'U') IS NOT NULL
4 DROP TABLE dbo.TableFromSnippet
5 GO
6 -- Create the table in the specified schema
7 CREATE TABLE dbo.TableFromSnippet
8 (
9     TableFromSnippetId INT NOT NULL PRIMARY KEY, -- primary key column
10    Column1 NVARCHAR(50) NOT NULL,
11    Column2 NVARCHAR(50) NOT NULL
12    -- specify more columns here
13 );
14 GO
```

If the field you want to change is no longer highlighted (this happens when moving the cursor around the editor), right-click the word you want to change, and select **Change all occurrences**:

The screenshot shows the SSMS interface with a context menu open over a T-SQL script in the SQL Query Editor. The menu includes options like Go to Definition, Peek Definition, Change All Occurrences (which is highlighted with a red box), Format Document, Cut, Copy, Paste, and Command Palette... . The script in the editor is:

```

1  -- Create a new table called 'TableFromSnippet' in schema 'dbo'
2  -- Drop the table if it already exists
3  IF OBJECT_ID('dbo.TableFromSnippet', 'U') IS NOT NULL
4  DROP TABLE dbo.TableFromSnippet
5  GO
6  -- Create the table in the specified schema
7  CREATE TABLE dbo.TableFromSnippet
8  (
9      TableFromSnippetId INT NOT NULL PRIMARY KEY, -- primary key
10     Column1 [NVARCHAR](50) NOT NULL,
11     Column2 [NVARCHAR](50) NOT NULL
12     -- specify more columns here
13 );
14 GO

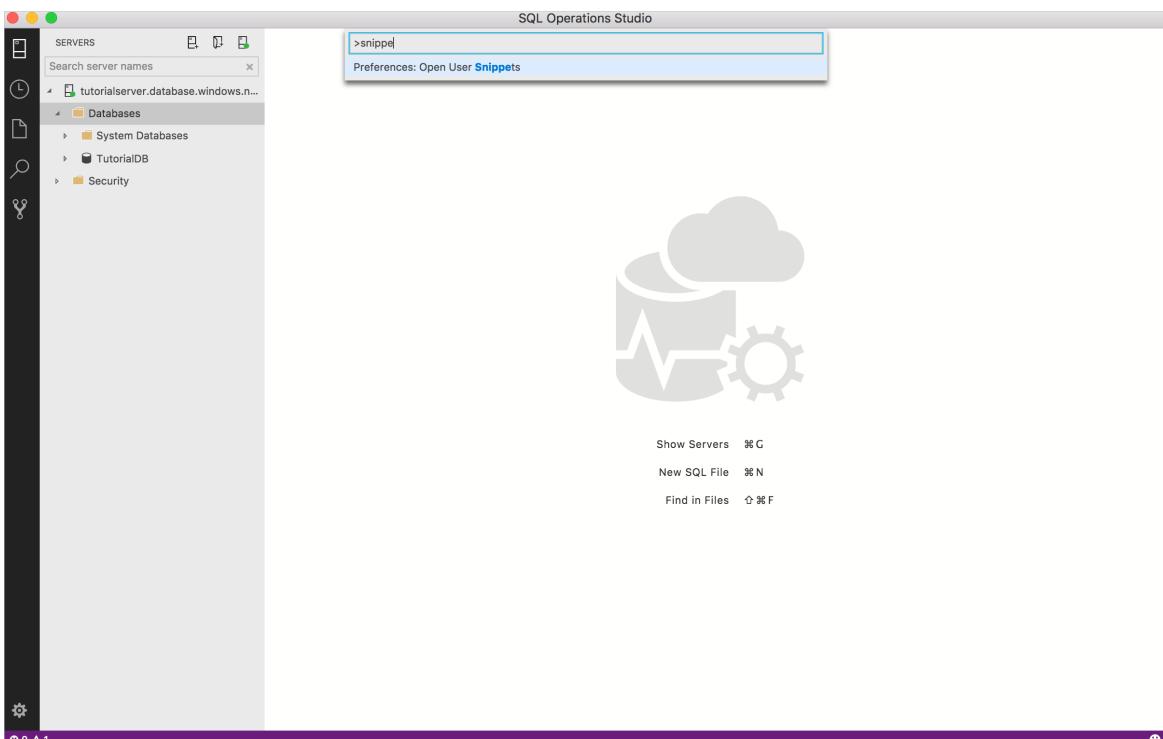
```

4. Update or add any additional T-SQL you need for the selected snippet. For example, update *Column1*, *Column2*, and add more columns.

## Creating SQL code snippets

You can define your own snippets. To open up the SQL snippet file for editing:

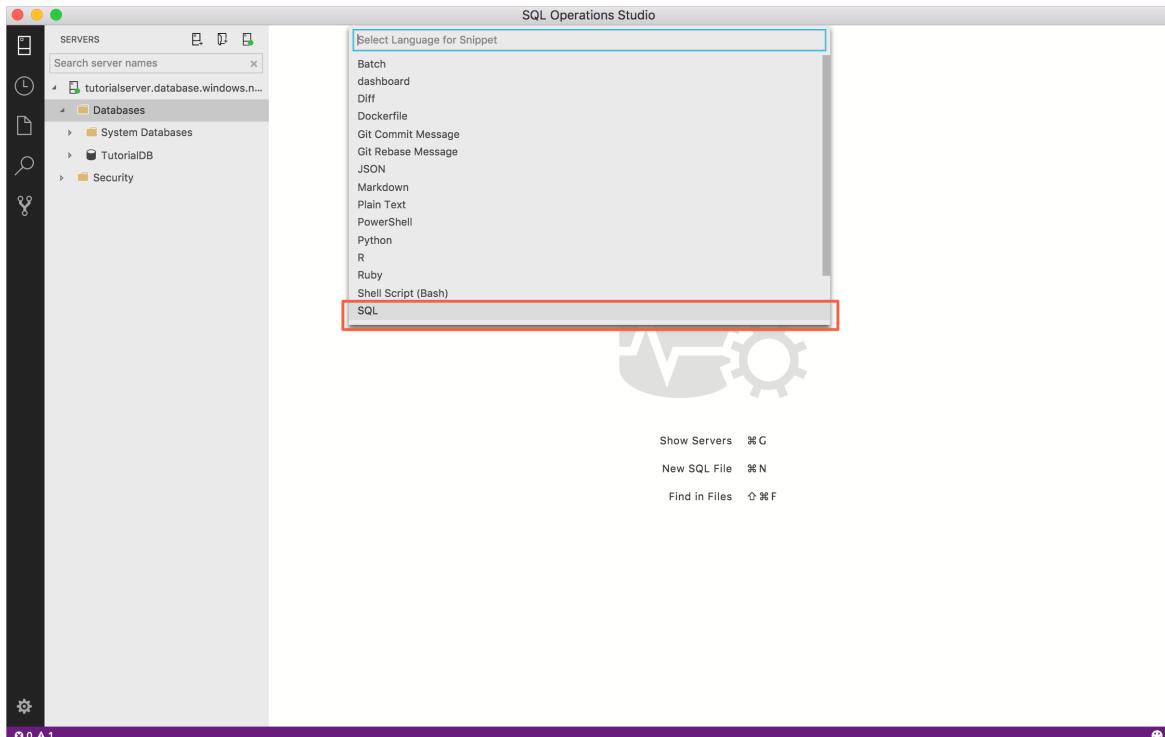
1. Open the *Command Palette* (**Shift+Ctrl+P**), and type *snip*, and select **Preferences: Open User Snippets**:



2. Select **SQL**:

## NOTE

Azure Data Studio inherits its code snippet functionality from Visual Studio Code so this article specifically discusses using SQL snippets. For more detailed information, see [Creating your own snippets](#) in the Visual Studio Code documentation.



3. Paste the following code into *sql.json*:

```
{  
  "Select top 5": {  
    "prefix": "sqlSelectTop5",  
    "body": "SELECT TOP 5 * FROM ${1:TableName}",  
    "description": "User-defined snippet example 1"  
  },  
  "Create Table snippet":{  
    "prefix": "sqlCreateTable2",  
    "body": [  
      "-- Create a new table called '${1:TableName}' in schema '${2:SchemaName}'",  
      "-- Drop the table if it already exists",  
      "IF OBJECT_ID('${2.$1}', 'U') IS NOT NULL",  
      "DROP TABLE $2.$1",  
      "GO",  
      "-- Create the table in the specified schema",  
      "CREATE TABLE $2.$1",  
      "(",  
      "$1Id INT NOT NULL PRIMARY KEY, -- primary key column",  
      "Column1 [NVARCHAR](50) NOT NULL,",  
      "Column2 [NVARCHAR](50) NOT NULL",  
      "-- specify more columns here",  
      ");",  
      "GO"  
    ],  
    "description": "User-defined snippet example 2"  
  }  
}
```

4. Save the *sql.json* file.

5. Open a new query editor window by clicking **Ctrl+N**.

6. Type **sql**, and you see the two user snippets you just added; *sql/CreateTable2* and *sql>SelectTop5*.

Select one of the new snippets and give it a test run!

## Next steps

For information about the SQL editor, see [Code editor tutorial](#).

# Explore and manage Azure SQL resources with Azure Resource Explorer

9/26/2022 • 2 minutes to read • [Edit Online](#)

In this document, you learn how you can explore and manage Azure SQL Server, Azure SQL database, and Azure SQL Managed Instance resources through Azure Resource Explorer in Azure Data Studio.

## NOTE

The Azure Resource Explorer is supported in SQL Server 2019. After that, you can install the extension through [extension manager](#) or through **File > Install Package from VSIX Package**.

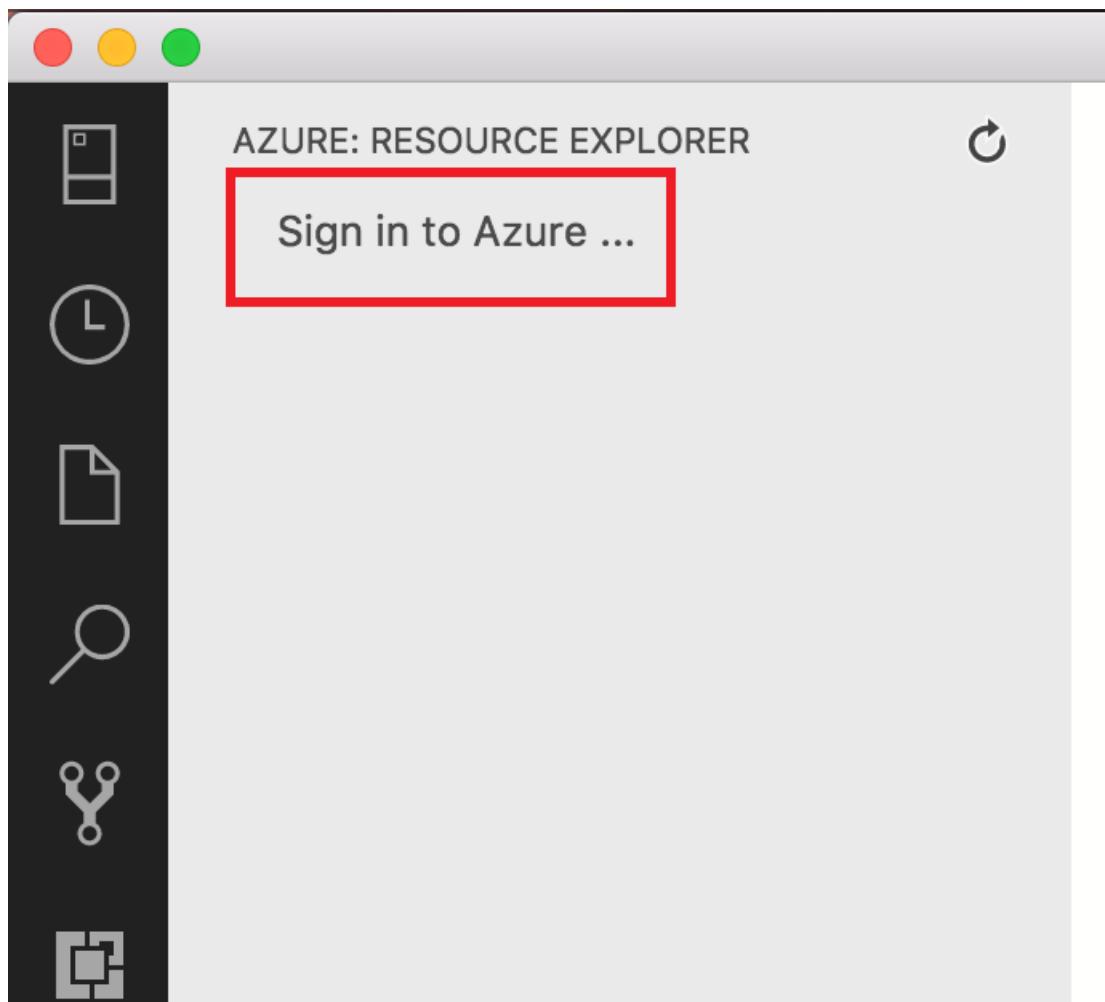
## Connect to Azure

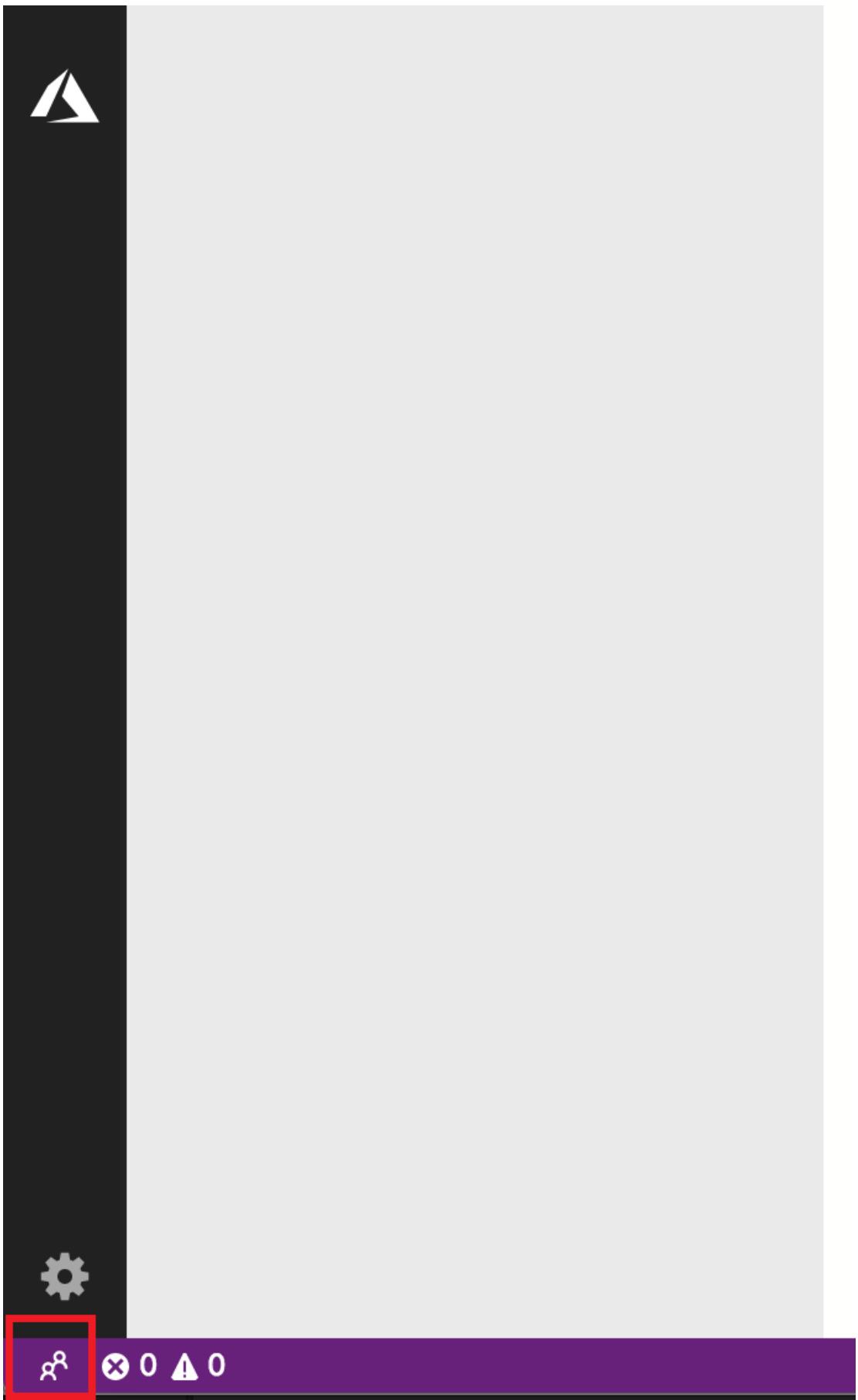
After installing the SQL plugin, an Azure icon appears in the left menu bar. Click the icon to open Azure Resource Explorer. If you don't see the Azure icon, right click the left menu bar, and select **Azure Resource Explorer**.

### Add an Azure account

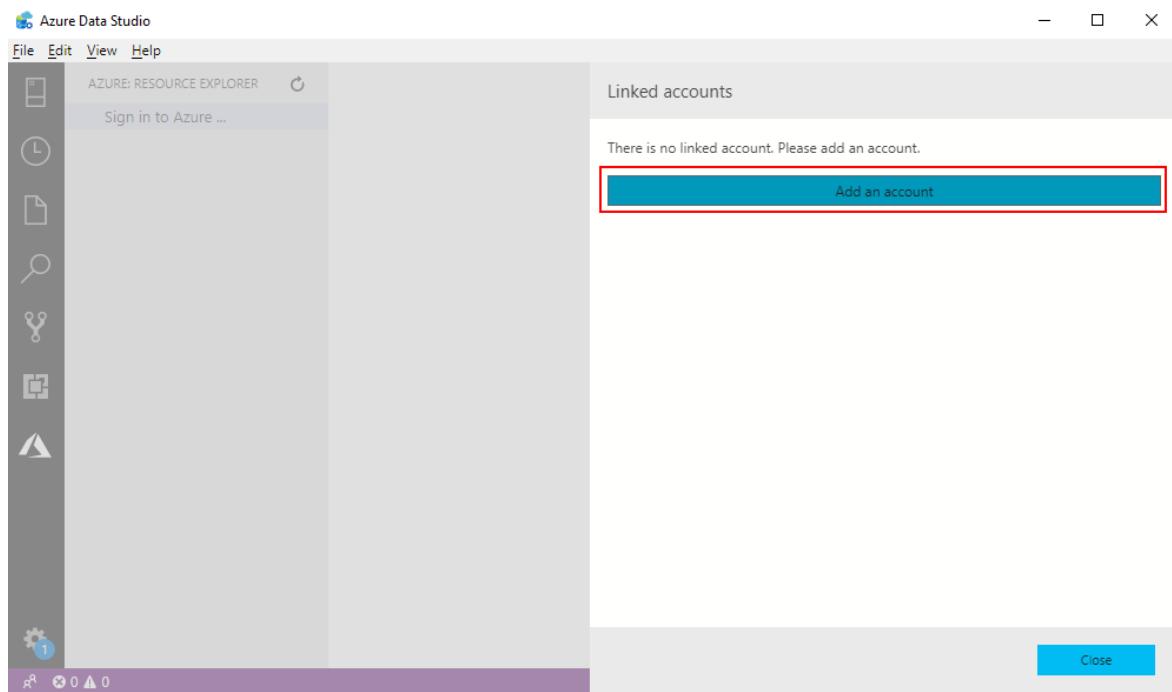
To view the SQL resources associated with an Azure account, you must first add the account to Azure Data Studio.

1. Open **Linked Accounts** dialog through the account management icon on the left bottom, or through **Sign in to Azure...** link in Azure Resource Explorer.

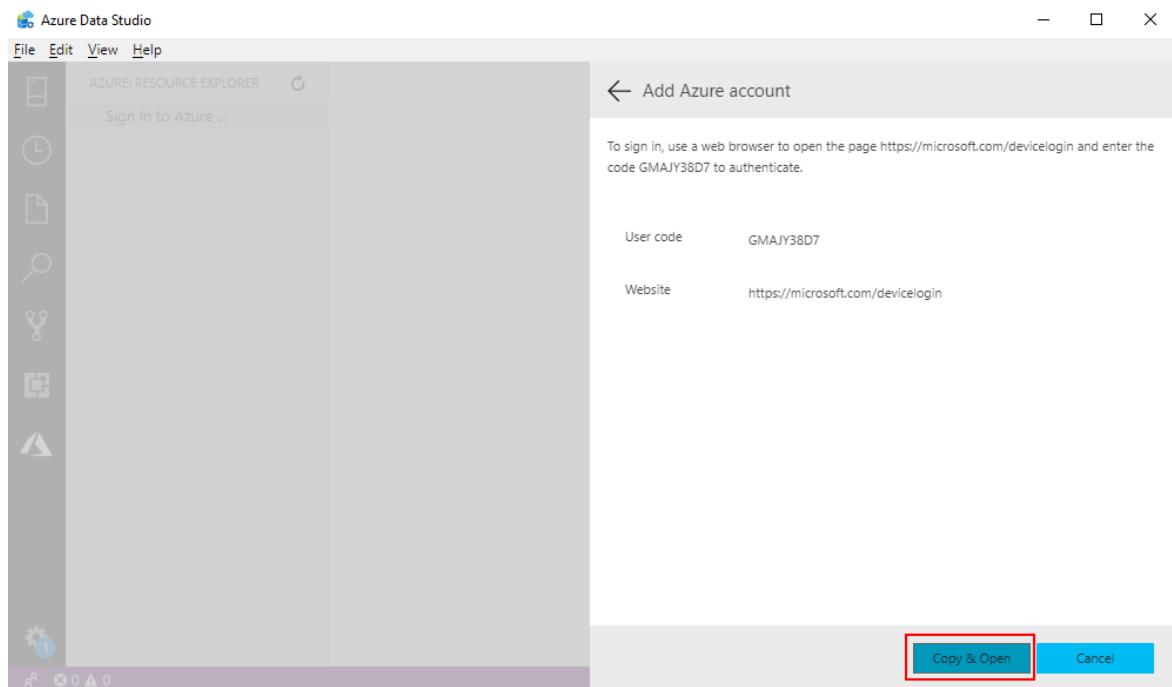




2. In the **Linked Accounts** dialog, click **Add an account**.



3. Click **Copy and Open** to open the browser for authentication.



4. Paste the **User code** in the web page and click **Continue** to authenticate.

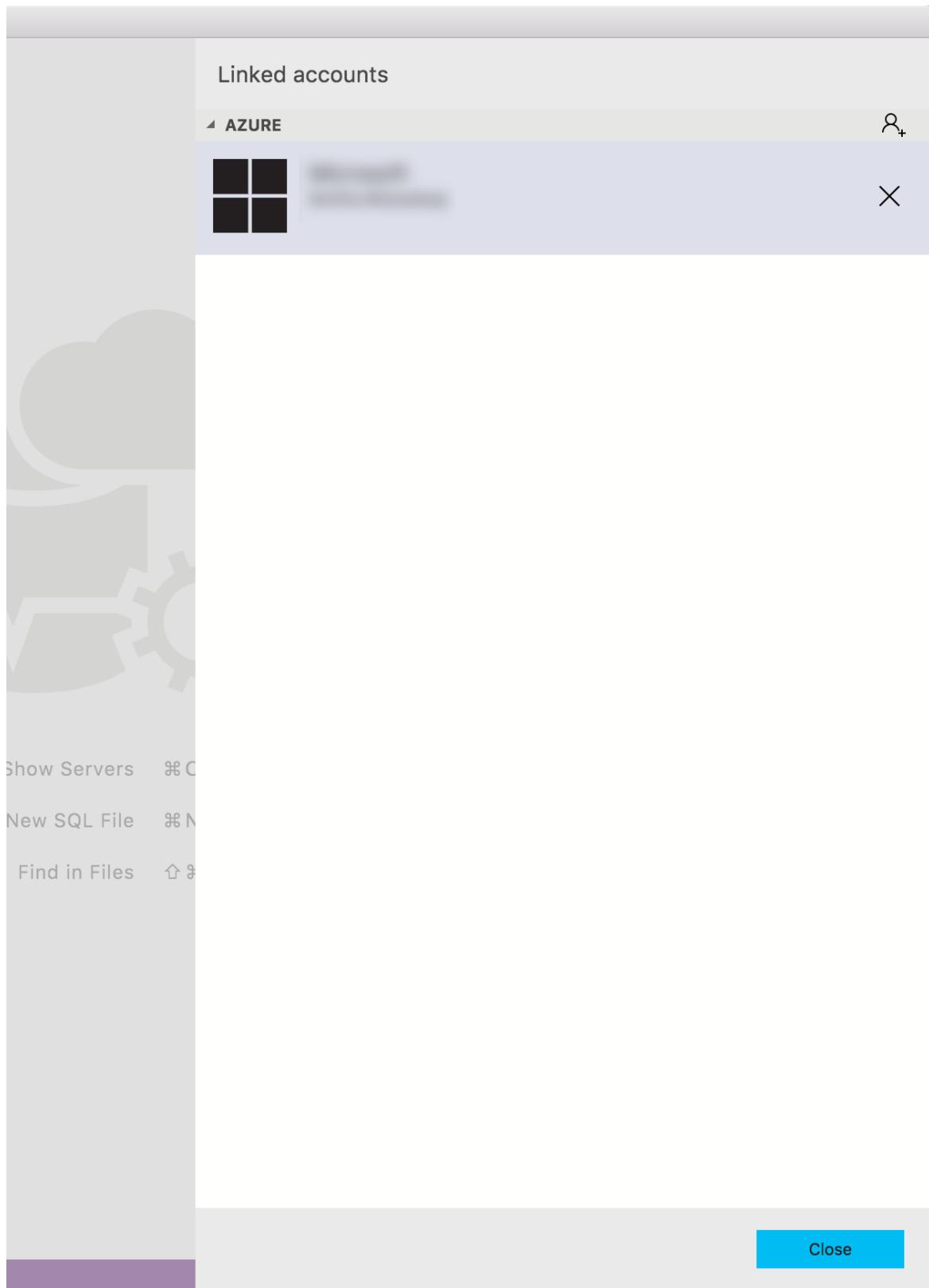
# Device Login

Enter the code that you received from the application on your device

## Azure Data Studio

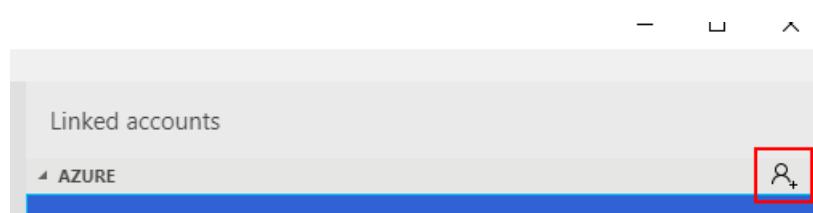
Click Cancel if this isn't the application you were trying to sign in to on your device.

5. In Azure Data Studio you should now find the logged in Azure account in **Linked Accounts** dialog.



### Add more Azure accounts

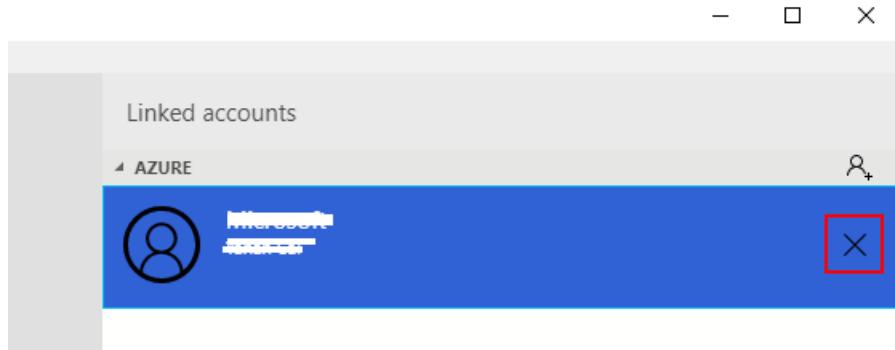
Multiple Azure accounts are supported in Azure Data Studio. To add more Azure accounts, click the button on the right top of **Linked Accounts** dialog and follow the same steps with Add an Azure account section to add more Azure accounts.



## Remove an Azure account

To remove an existing logged in Azure account:

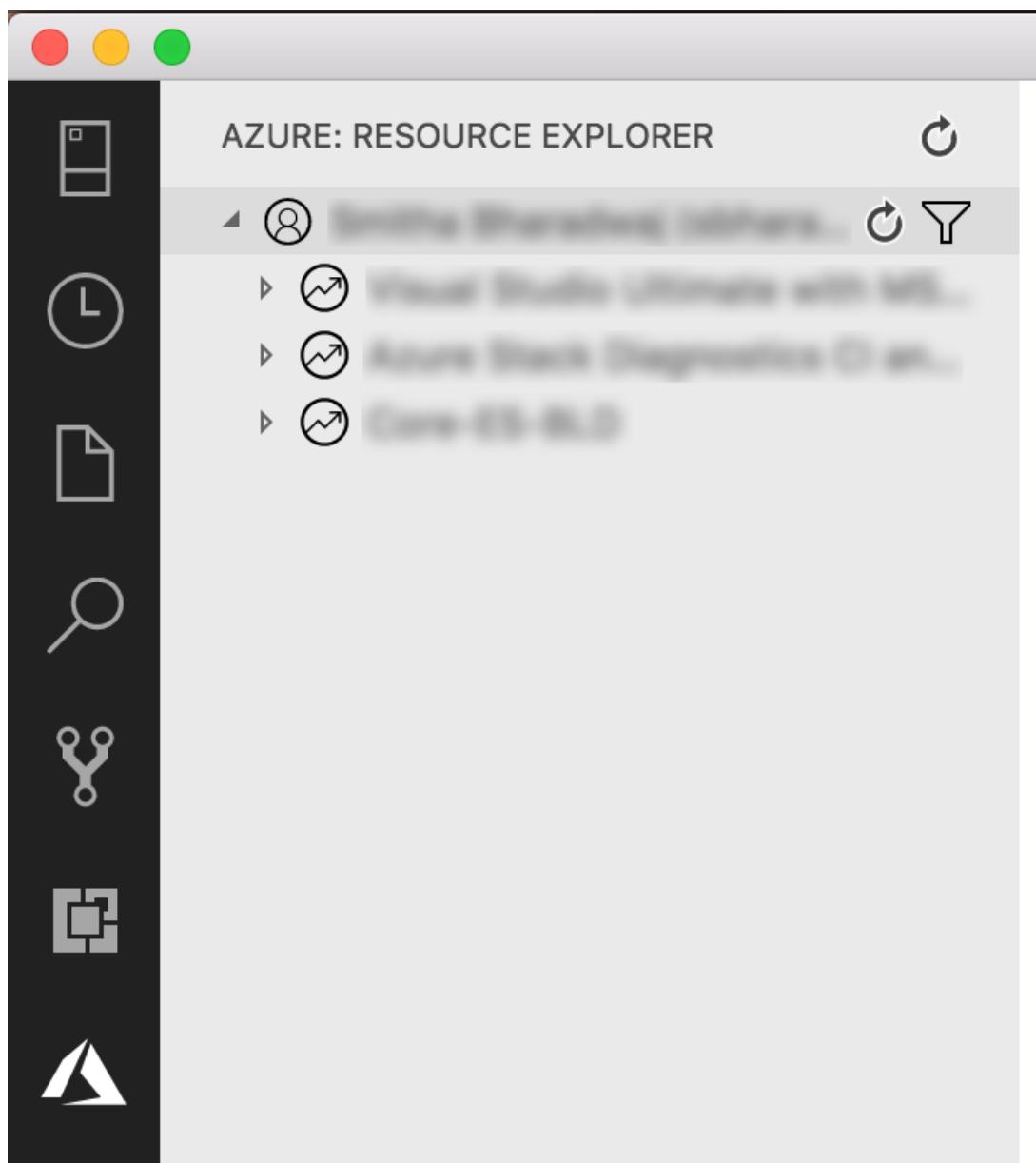
1. Open **Linked Accounts** dialog through the account management icon on the left bottom.
2. Click the X button at the right of the Azure account to remove it.

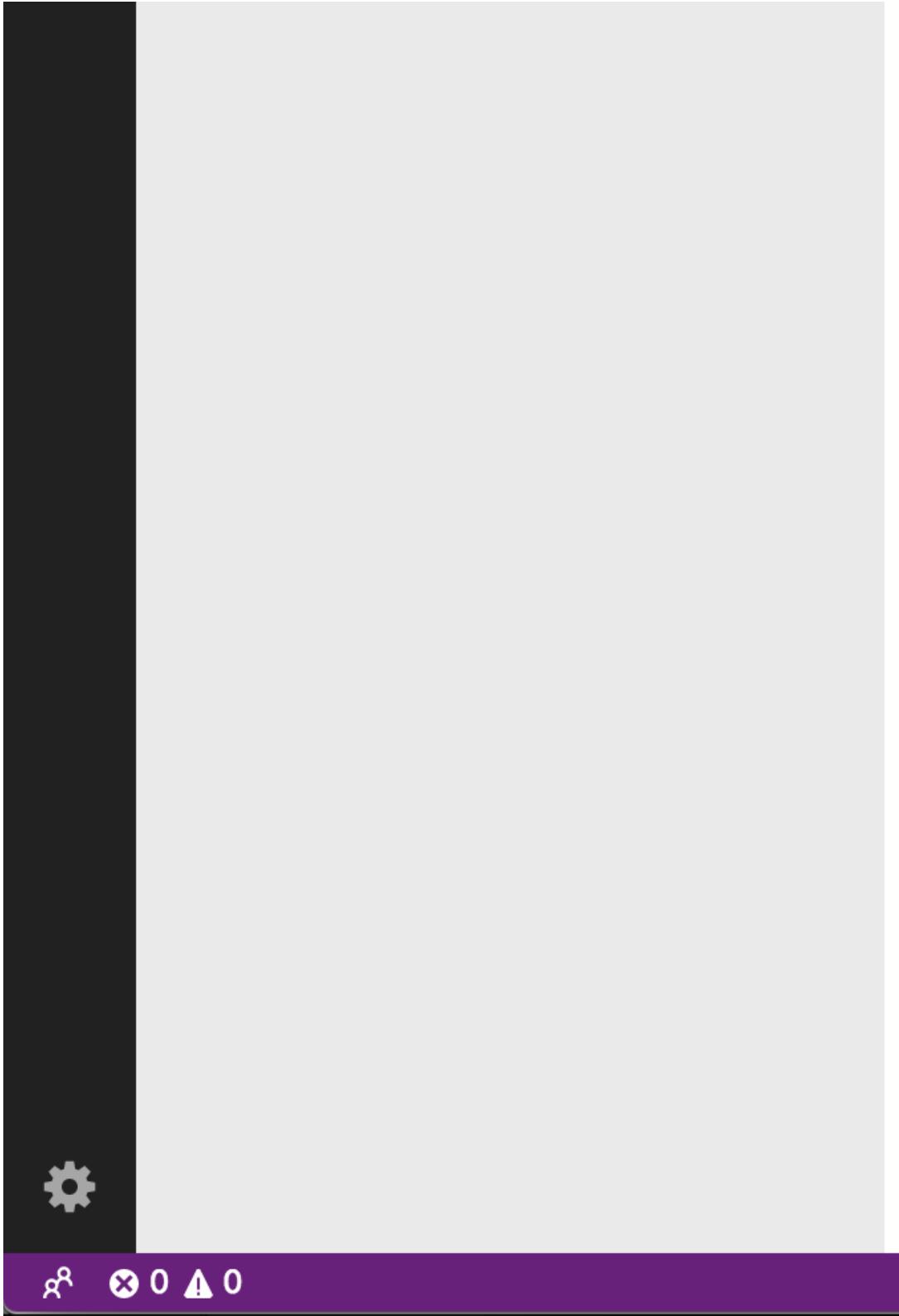


## Filter subscription

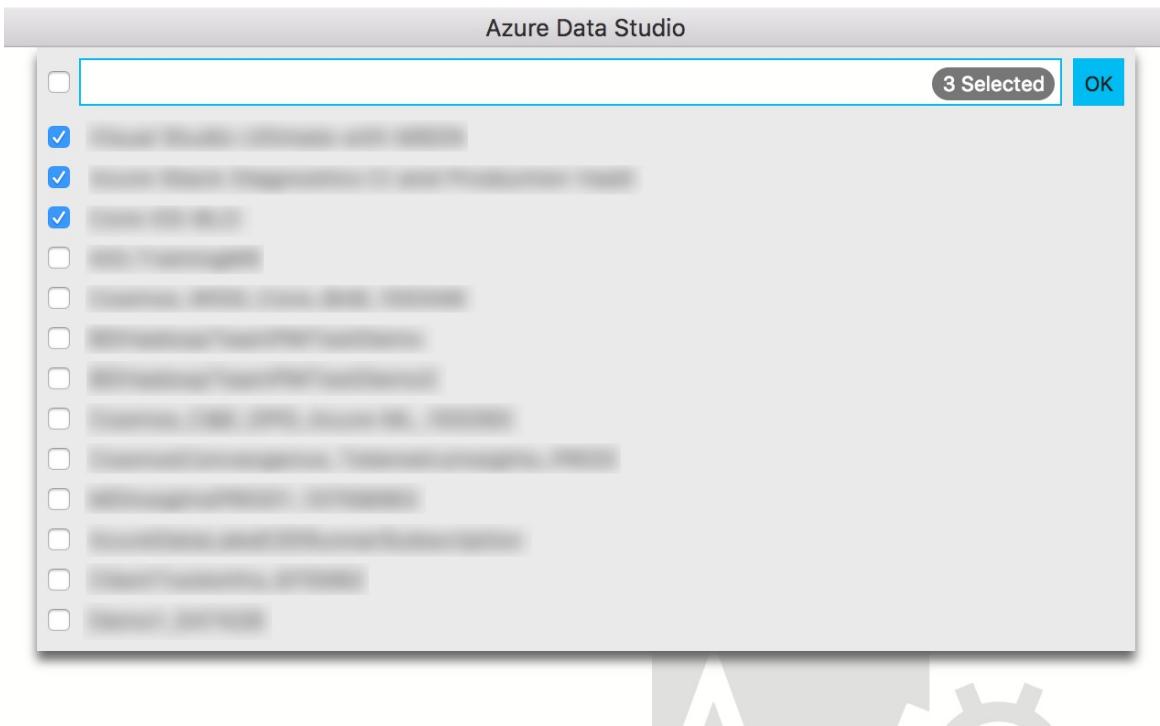
Once logged in to an Azure account, all subscriptions associated with that Azure account display in Azure Resource Explorer. You can filter subscriptions for each Azure account.

1. Click the **Select Subscription** button at right of the Azure account.





2. Select the check boxes for the account subscriptions you want to browse and then click **OK**.



## Explore Azure SQL resources

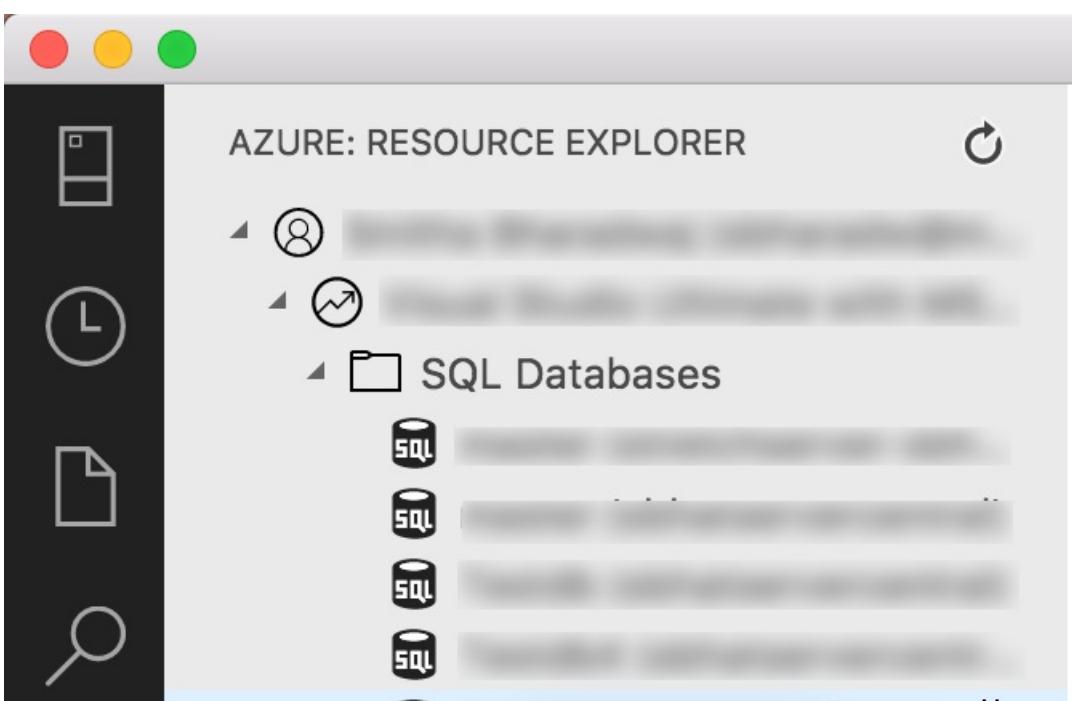
To navigate an Azure SQL resource in Azure Resource Explorer, expand the Azure accounts and resource type group.

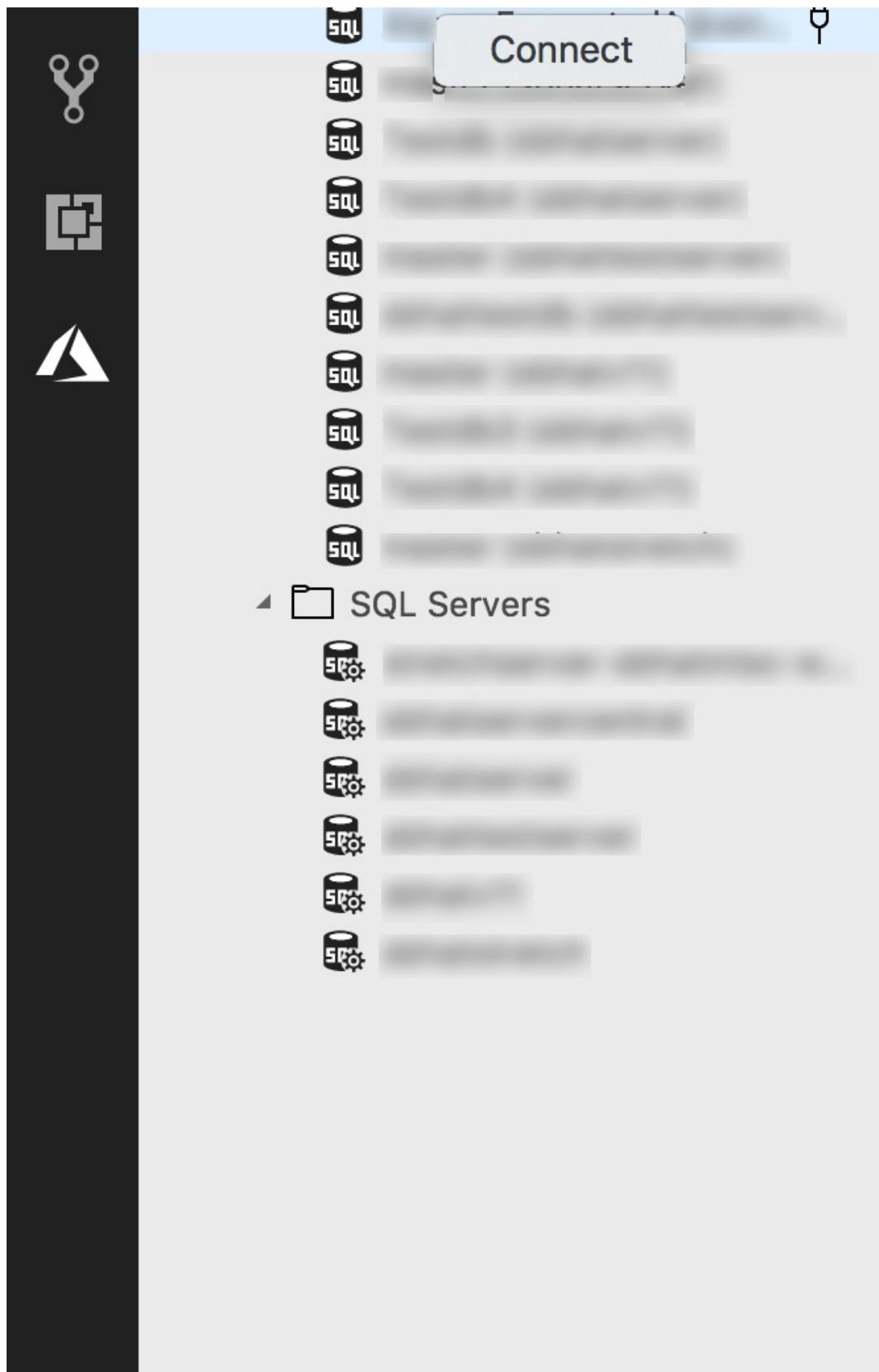
Azure Resource Explorer supports Azure SQL Server, Azure SQL Database and Azure SQL Managed Instance currently.

## Connect to Azure SQL resources

Azure Resource Explorer provide quick access that helps you connect to SQL Servers and databases for query and management.

1. Explore the SQL resource you would like to connect with from the tree view.
2. Right click the resource and select **Connect**, you can also find the connect button at the right of the resource.





3. In the opened **Connection** dialog, enter your password and click **Connect**.

## Connection

RECENT CONNECTIONS SAVED CONNECTIONS

### Recent history

- [REDACTED]
- [REDACTED]



Connection type	Microsoft SQL Server
Server	[REDACTED]
Authentication type	SQL Login
User name	[REDACTED]
Password	<input type="password"/>
<input type="checkbox"/> Remember password	
Database	[REDACTED]
Server group	<Default>
Name (optional)	[REDACTED]

[Advanced...](#)

Connect

Cancel

4. The **Servers** window automatically opens with the new connected SQL server/database after connection succeeds.

## Next steps

- [Use Azure Data Studio to connect and query Azure SQL database](#)
- [Use Azure Data Studio to connect and query data in Azure Synapse Analytics](#)

# Keyboard shortcuts in Azure Data Studio

9/26/2022 • 2 minutes to read • [Edit Online](#)

This article provides the steps to quickly view, edit, and create keyboard shortcuts in Azure Data Studio.

Because Azure Data Studio inherits its key binding functionality from Visual Studio Code, detailed information about advanced customizations, using different keyboard layouts, etc., is in the [Key Bindings for Visual Studio Code](#) article. Some key binding features may not be available (for example, Keymap extensions are not supported in Azure Data Studio).

## Open the Keyboard Shortcuts editor

To view all currently defined keyboard shortcuts:

Open the **Keyboard Shortcuts** editor from the **File** menu: **File > Preferences > Keyboard Shortcuts** (Azure Data Studio > Preferences > Keyboard Shortcuts on Mac).

In addition to displaying current key bindings, the **Keyboard Shortcuts** editor lists the available commands that do not have keyboard shortcuts defined. The **Keyboard Shortcuts** editor enables you to easily change, remove, reset, and define new key bindings.

## Edit existing keyboard shortcuts

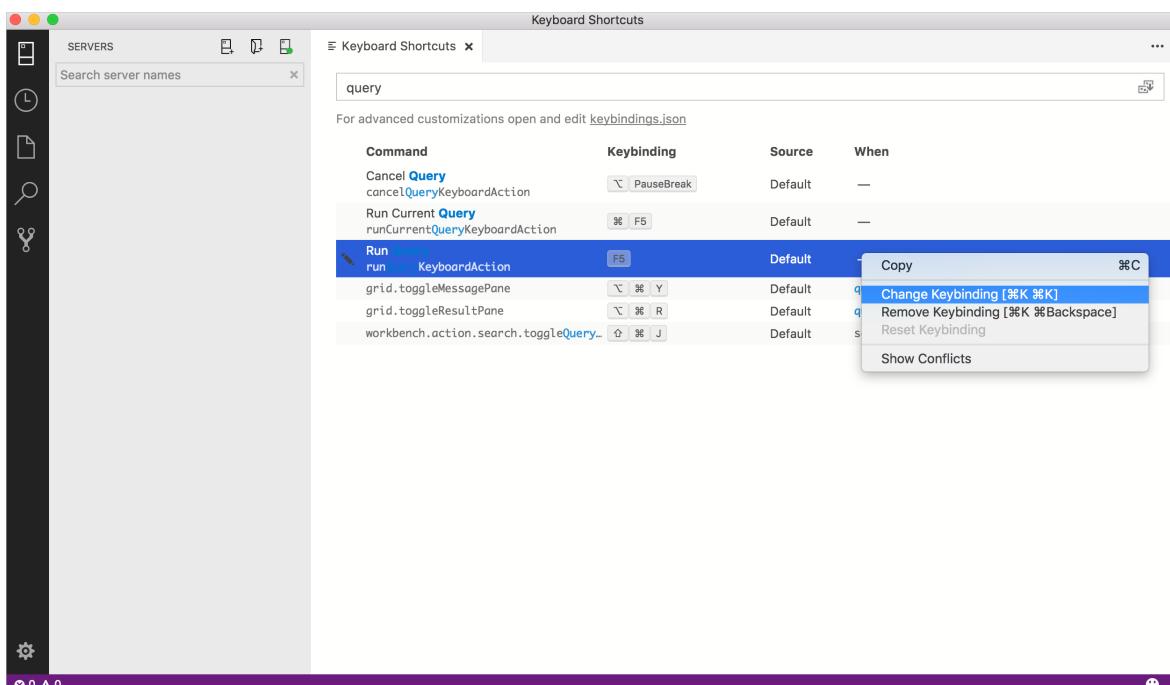
To change the key binding for an existing keyboard shortcut:

1. Locate the keyboard shortcut you want to change by using the search box or scrolling through the list.

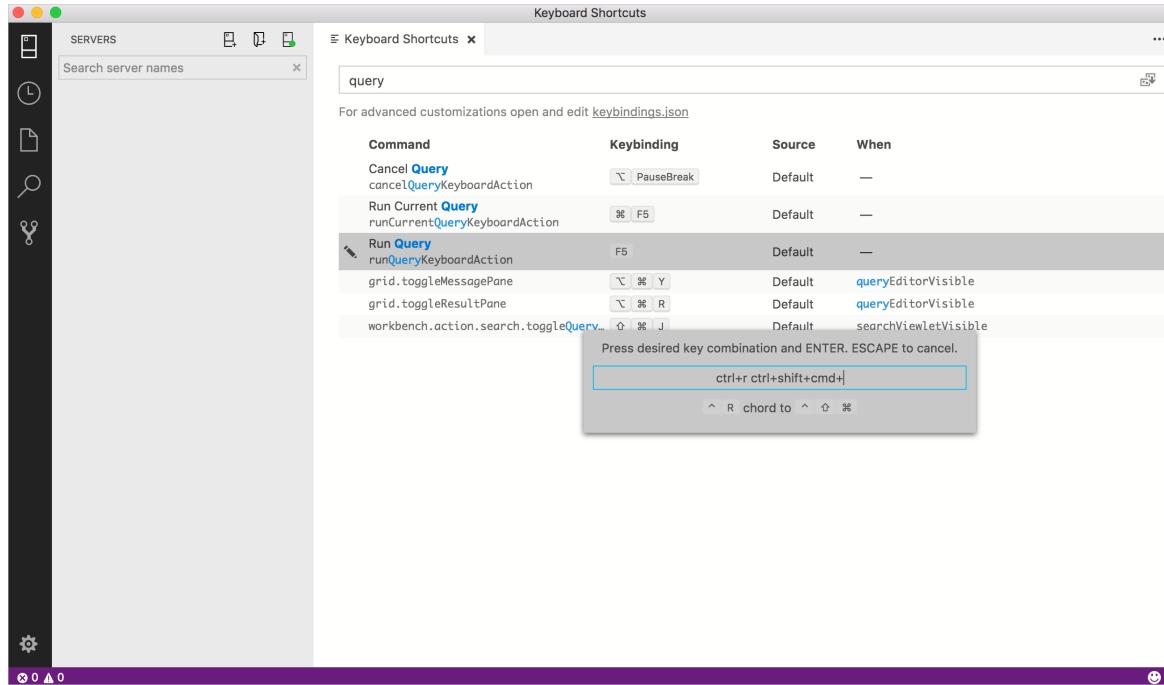
### TIP

Search by key, by command, by source, etc. to return all relevant keyboard shortcuts.

2. Right-click the desired entry and select **Change Key binding**



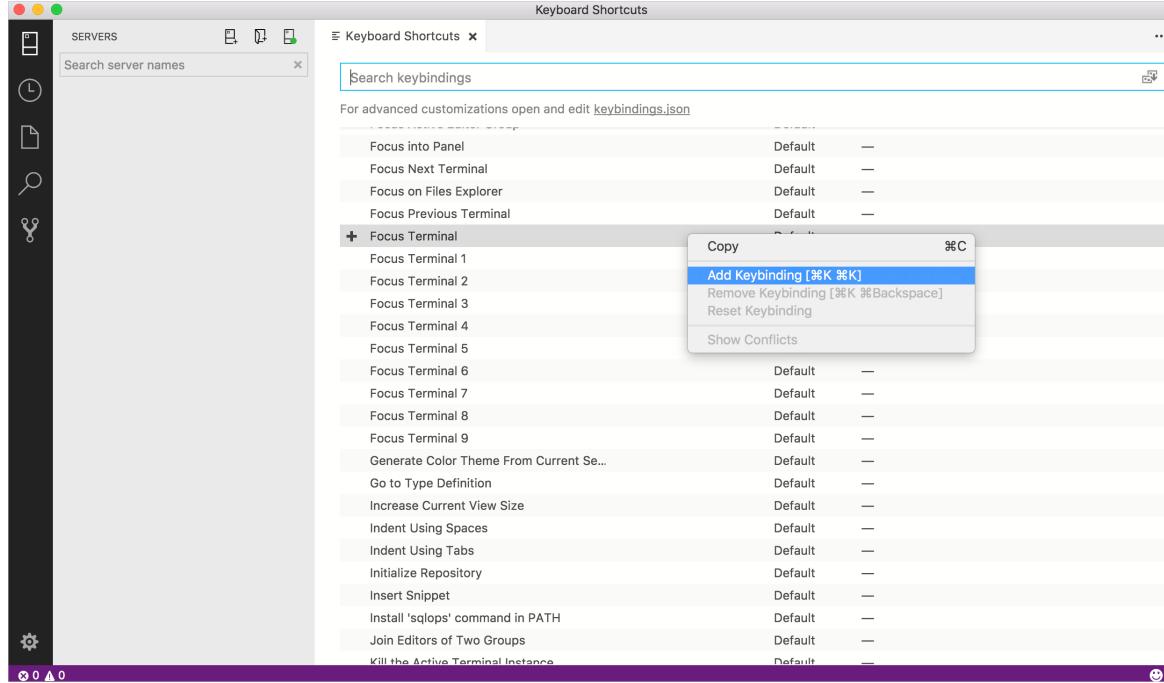
3. Press the desired combination of keys, then press **Enter** to save it.



# Create new keyboard shortcuts

To create new keyboard shortcuts:

1. Right-click a command that doesn't have any key binding and select Add Key binding.



2. Press the desired combination of keys, then press **Enter** to save it.

# Enable or disable usage data collection for Azure Data Studio

9/26/2022 • 2 minutes to read • [Edit Online](#)

Azure Data Studio contains Internet-enabled features that can collect and send anonymous feature usage and diagnostic data to Microsoft.

Azure Data Studio may collect standard computer, use, and performance information that may be transmitted to Microsoft and analyzed to improve the quality, security, and reliability of Azure Data Studio.

Azure Data Studio doesn't collect your name or address, but Azure Data Studio gathers data that helps approximate a single user for diagnostic purposes (based on a hash of the network adapter NIC).

Several updates made to Azure Data Studio to help ensure data privacy.

- Making it more accessible to opt-out of telemetry collection by placing a notification in the product for all existing and new users.
- Reviewing and classifying the telemetry that we send.
- Ensuring that we have valid data retention policies in place for any data we collect, for example, crash dumps.

For details, see the [Microsoft Privacy Statement](#), and [SQL Server Privacy supplement](#).

## Audit feature usage and diagnostic data

To see feature usage data that is collected by Azure Data Studio, follow the steps below:

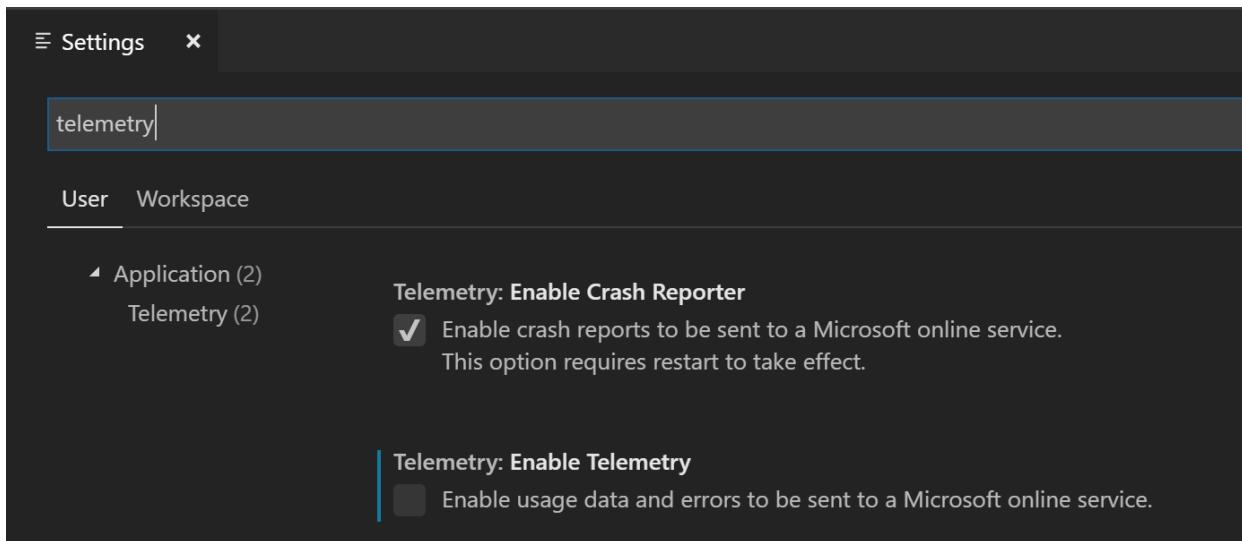
1. Launch Azure Data Studio.
2. Open the command palette and choose the **Developer: Set Log Level...** command
3. Select **Trace** from the options
4. Open the Output panel (Ctrl+Shift+U)
5. Select **Log (Telemetry)** from the dropdown
6. Select **View**, then Select **Output** in the main menu to show the **Output** window.
7. When the **Output** window is visible, choose **Log (Window)** in the **Show output from:** menu.

When tracing telemetry events, the events are also logged to a local file telemetry.log, which you can view using the **Developer: Open Log File...** command and choosing **Telemetry** from the dropdown.

## Disable telemetry reporting

To not send usage data to Microsoft, you can set the `telemetry.enableTelemetry` [user settings](#) to `false`.

Go to **File > Preferences > Settings** (macOS: **Code > Preferences > Settings**) and search for `telemetry`, and uncheck the **Telemetry: Enable Telemetry** setting. With this setting, you silence all telemetry events from Azure Data Studio going forward. Telemetry information may have been collected and sent up until the point when you disable the setting.



If you use the JSON editor for your settings, add the following line:

```
"telemetry.enableTelemetry": false
```

## Disable crash reporting

Azure Data Studio collects data about any crashes that occur and sends it to Microsoft to help improve our products and services.

If you don't want to send crash data to Microsoft, you can change the `enable-crash-reporter` runtime argument to `false`.

1. Open the Command Palette (`kb(workbench.action.showCommands)`).
2. Run the **Preferences: Configure Runtime Arguments** command.
3. This command opens a `argv.json` file to configure runtime arguments.
4. Edit `"enable-crash-reporter": false`.
5. Restart Azure Data Studio.

## Extensions and telemetry

Azure Data Studio lets you add features to the product by installing Microsoft and third-party extensions. These extensions may be collecting their own usage data and are not controlled by the `telemetry.enableTelemetry` setting. Consult the specific extension's documentation to learn about its telemetry reporting and whether it can be disabled.

## More resources

- [Workspace and User settings](#)
- [GDPR section of the Service Trust portal](#)
- [GDPR section of the Microsoft Trust Center](#)

## See also

- [Configure usage and diagnostic data collection for SQL Server](#)
- [Local audit for SQL Server usage and diagnostic data collection](#)

# Connect Azure Data Studio to SQL Server using Kerberos

9/26/2022 • 3 minutes to read • [Edit Online](#)

Azure Data Studio supports connecting to SQL Server by using Kerberos.

To use integrated authentication (Windows Authentication) on macOS or Linux, you need to set up a *Kerberos ticket* that links your current user to a Windows domain account.

## Prerequisites

To get started, you need:

- Access to a Windows domain-joined machine to query your Kerberos domain controller.
- SQL Server should be configured to allow Kerberos authentication. For the client driver running on Unix, integrated authentication is supported only by using Kerberos. For more information, see [Using Kerberos integrated authentication to connect to SQL Server](#). There should be [service principal names \(SPNs\)](#) registered for each instance of SQL Server you're trying to connect to. For more information, see [Register a Service Principal Name for Kerberos Connections](#).

## Check if SQL Server has a Kerberos setup

Sign in to the host machine of SQL Server. From the Windows command prompt, use `setspn -L %COMPUTERNAME%` to list all the SPNs for the host. Verify there are entries that begin with MSSQLSvc/HostName.Domain.com. These entries mean that SQL Server has registered an SPN and is ready to accept Kerberos authentication.

If you don't have access to the host of the SQL Server instance, then from any other Windows OS joined to the same Active Directory, you could use the command `setspn -L <SQLSERVER_NETBIOS>`, where `<SQLSERVER_NETBIOS>` is the computer name of the host of the SQL Server instance.

## Get the Kerberos Key Distribution Center

Find the Kerberos Key Distribution Center (KDC) configuration value. Run the following command on a Windows computer that's joined to your Active Directory domain.

Start `cmd.exe` and run `nlttest`.

```
nlttest /dsgetdc:DOMAIN.COMPANY.COM (where "DOMAIN.COMPANY.COM" maps to your domain's name)

Sample Output
DC: \\dc-33.domain.company.com
Address: \\2111:4444:2111:33:1111:ecff:ffff:3333
...
The command completed successfully
```

Copy the DC name that's the required KDC configuration value. In this case, it's dc-33.domain.company.com.

## Join your OS to the Active Directory domain controller

**Ubuntu**

```
sudo apt-get install realmd krb5-user software-properties-common python-software-properties packagekit
```

Edit the `/etc/network/interfaces` file so that your Active Directory domain controller's IP address is listed as dns-nameserver. For example:

```
<...>
# The primary network interface
auto eth0
iface eth0 inet dhcp
dns-nameservers **<AD domain controller IP address>**
dns-search **<AD domain name>**
```

#### NOTE

The network interface (eth0) might differ for different machines. To find out which one you're using, run ifconfig and copy the interface that has an IP address and transmitted and received bytes.

After editing this file, restart the network service:

```
sudo ifdown eth0 && sudo ifup eth0
```

Now check that your `/etc/resolv.conf` file contains a line like the following one:

```
nameserver **<AD domain controller IP address>**
```

```
sudo realm join contoso.com -U 'user@CONTOSO.COM' -v
<...>
* Success
```

#### Red Hat Enterprise Linux

```
sudo yum install realmd krb5-workstation
```

Edit the `/etc/sysconfig/network-scripts/ifcfg-eth0` file (or other interface config file as appropriate) so that your Active Directory domain controller's IP address is listed as a DNS server:

```
<...>
PEERDNS=no
DNS1=**<AD domain controller IP address>**
```

After editing this file, restart the network service:

```
sudo systemctl restart network
```

Now check that your `/etc/resolv.conf` file contains a line like the following one:

```
nameserver **<AD domain controller IP address>**
```

```
sudo realm join contoso.com -U 'user@CONTOSO.COM' -v  
<...>  
* Success
```

## Configure KDC in krb5.conf with macOS

This section discusses the [Kerberos configuration file](#).

Edit the `/etc/krb5.conf` file in an editor of your choice. Configure the following keys:

```
sudo vi /etc/krb5.conf

[libdefaults]
default_realm = DOMAIN.COMPANY.COM

[realms]
DOMAIN.COMPANY.COM = {
    kdc = dc-33.domain.company.com
}
```

Then save the krb5.conf file and exit.

### NOTE

The domain must be in ALL CAPS.

## Test the ticket granting ticket retrieval

Get a Ticket Granting Ticket (TGT) from KDC.

```
kinit username@DOMAIN.COMPANY.COM
```

View the available tickets by using klist. If the kinit was successful, you should see a ticket.

```
klist  
  
krbtgt/DOMAIN.COMPANY.COM@ DOMAIN.COMPANY.COM.
```

## Connect by using Azure Data Studio

1. Create a new connection profile.
2. Select **Windows Authentication** as the authentication type.
3. Complete the connection profile, and select **Connect**.

After successfully connecting, your server appears in the **SERVERS** sidebar.

# Modify User and Workspace Settings

9/26/2022 • 2 minutes to read • [Edit Online](#)

It's easy to configure Azure Data Studio to your liking through settings. Nearly every part of Azure Data Studio's editor, user interface, and functional behavior has options you can modify.

Azure Data Studio provides two different scopes for settings:

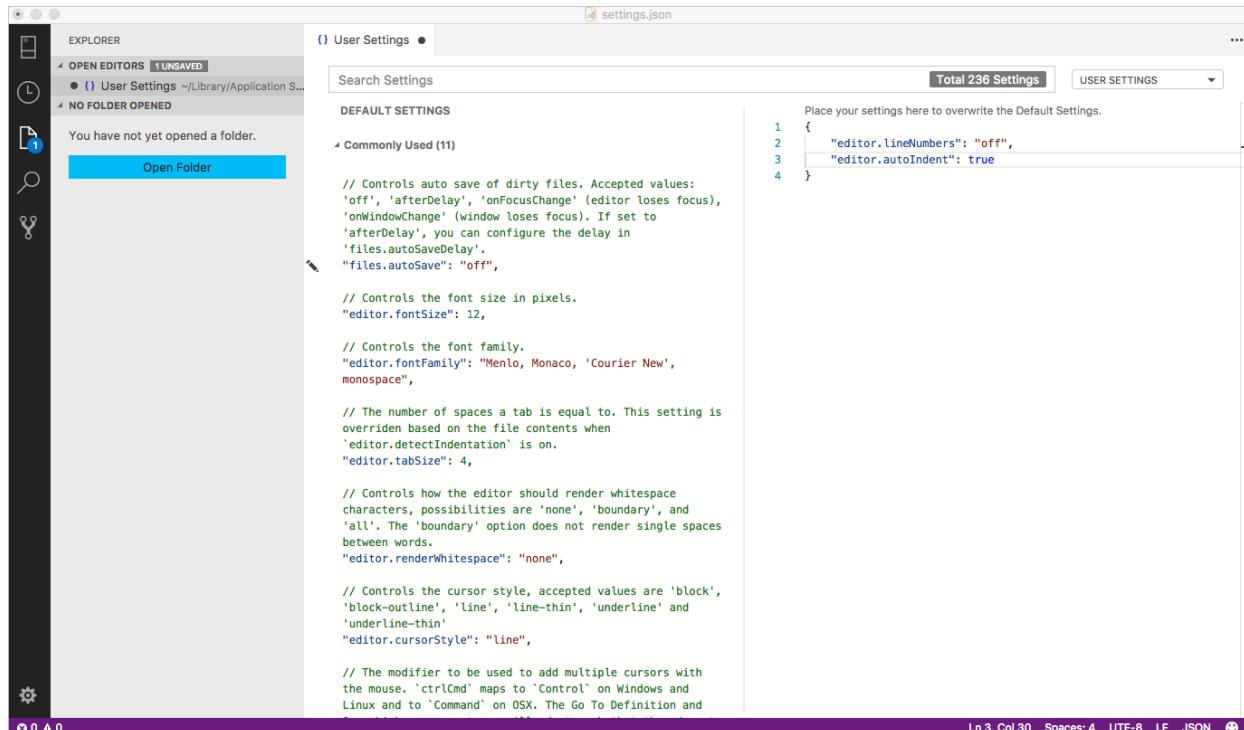
- **User** - These settings apply globally to any instance of Azure Data Studio you open.
- **Workspace** - Workspace settings are settings specific to a folder on your computer, and are only available when the folder is open in the Explorer sidebar. Settings defined on this scope override the user scope.

## Creating User and Workspace Settings

The menu command **File > Preferences > Settings** (**Code > Preferences > Settings** on Mac) provides the entry point to configure user and workspace settings. You're provided with a list of Default Settings. Copy any setting that you want to change to the appropriate `settings.json` file. The tabs on the right let you switch quickly between the user and workspace settings files.

You can also open the user and workspace settings from the **Command Palette (Ctrl+Shift+P)** with **Preferences: Open User Settings** and **Preferences: Open Workspace Settings** or use the keyboard shortcut (**Ctrl+,**).

The following example disables line numbers in the editor and configures lines of code to be indented automatically.



Changes to settings are reloaded by Azure Data Studio after the modified `settings.json` file is saved.

### NOTE

Workspace settings are useful for sharing project-specific settings across a team.

# Settings File Locations

Depending on your platform, the user settings file is located here:

- **Windows** `%APPDATA%\azuredatastudio\User\settings.json`
- **Mac** `$HOME/Library/Application Support/azuredatastudio/User/settings.json`
- **Linux** `$HOME/.config/azuredatastudio/User/settings.json`

The workspace setting file is located under the `.Azure Data Studio` folder in your project.

## Hot Exit

Azure Data Studio remembers unsaved changes to files when you exit by default. In Visual Studio Code, this is the same as the hot exit feature.

By default, hot exit's off. Enable hot exit by editing the `files.hotExit` setting. For details, see [Hot Exit \(in the Visual Studio Code documentation\)](#).

## Tab color

To simplify identifying what connections you're working with, open tabs in the editor can have their colors set to match the color of the Server Group the connection belongs to. By default, tab colors are off by default. Enable tab colors by editing the `sql.tabColorMode` setting.

## Additional resources

Because Azure Data Studio inherits its user and workspace settings functionality from Visual Studio Code, detailed information about settings is in the [Settings for Visual Studio Code](#) article.

# SQL tools overview

9/26/2022 • 3 minutes to read • [Edit Online](#)

Applies to: ✓ SQL Server (all supported versions) ✓ Azure SQL Database ✓ Azure SQL Managed Instance ✓ Azure Synapse Analytics ✓ Analytics Platform System (PDW)

To manage your database, you need a tool. Whether your databases run in the cloud, on Windows, on macOS, or on [Linux](#), your tool doesn't need to run on the same platform as the database.

You can view the links to the different SQL tools in the following tables.

## NOTE

To download SQL Server, see [Install SQL Server](#).

## Recommended tools

The following tools provide a graphical user interface (GUI).

TOOL	DESCRIPTION	OPERATING SYSTEM
 <a href="#">Azure Data Studio</a>	A light-weight editor that can run on-demand SQL queries, view and save results as text, JSON, or Excel. Edit data, organize your favorite database connections, and browse database objects in a familiar object browsing experience.	Windows macOS Linux
 <a href="#">SQL Server Management Studio (SSMS)</a>	Manage a SQL Server instance or database with full GUI support. Access, configure, manage, administer, and develop all components of SQL Server, Azure SQL Database, and Azure Synapse Analytics. Provides a single comprehensive utility that combines a broad group of graphical tools with a number of rich script editors to provide access to SQL for developers and database administrators of all skill levels.	Windows
 <a href="#">SQL Server Data Tools (SSDT)</a>	A modern development tool for building SQL Server relational databases, Azure SQL databases, Analysis Services (AS) data models, Integration Services (IS) packages, and Reporting Services (RS) reports. With SSDT, you can design and deploy any SQL Server content type with the same ease as you would develop an application in <a href="#">Visual Studio</a> .	Windows

TOOL	DESCRIPTION	OPERATING SYSTEM
 <b>Visual Studio Code</b>	The <a href="#">mssql extension</a> for Visual Studio Code is the official SQL Server extension that supports connections to SQL Server and rich editing experience for T-SQL in Visual Studio Code. Write T-SQL scripts in a light-weight editor.	Windows macOS Linux

## Command-line tools

The tools below are the main command-line tools.

TOOL	DESCRIPTION	OPERATING SYSTEM
<a href="#">bcp</a>	The <b>bcp</b> bulk copy program utility ( <b>bcp</b> ) bulk copies data between an instance of Microsoft SQL Server and a data file in a user-specified format.	Windows macOS Linux
<a href="#">mssql-cli (preview)</a>	<b>mssql-cli</b> is an interactive command-line tool for querying SQL Server. Also, query SQL Server with a command-line tool that features IntelliSense, syntax high-lighting, and more.	Windows macOS Linux
<a href="#">mssql-conf</a>	<b>mssql-conf</b> configures SQL Server running on Linux.	Linux
<a href="#">mssql-scripter (preview)</a>	<b>mssql-scripter</b> is a multi-platform command-line experience for scripting SQL Server databases.	Windows macOS Linux
<a href="#">sqlcmd</a>	<b>sqlcmd</b> utility lets you enter Transact-SQL statements, system procedures, and script files at the command prompt.	Windows macOS Linux
<a href="#">sqlpackage</a>	<b>sqlpackage</b> is a command-line utility that automates several database development tasks.	Windows macOS Linux
<a href="#">SQL Server PowerShell</a>	<b>SQL Server PowerShell</b> provides cmdlets for working with SQL.	Windows macOS Linux

## Migration and other tools

These tools are used to migrate, configure, and provide other features for SQL databases.

TOOL	DESCRIPTION
<a href="#">Configuration Manager</a>	Use SQL Server Configuration Manager to configure SQL Server services and configure network connectivity. Configuration Manager runs on Windows

TOOL	DESCRIPTION
<a href="#">Database Experimentation Assistant</a>	Use Database Experimentation Assistant to evaluate a targeted version of SQL for a given workload.
<a href="#">Data Migration Assistant</a>	The Data Migration Assistant tool helps you upgrade to a modern data platform by detecting compatibility issues that can impact database functionality in your new version of SQL Server or Azure SQL Database.
<a href="#">Distributed Replay</a>	Use the Distributed Replay feature to help you assess the impact of future SQL Server upgrades. Also use Distributed Replay to help assess the impact of hardware and operating system upgrades, and SQL Server tuning.
<a href="#">ssbdiagnose</a>	The ssbdiagnose utility reports issues in Service Broker conversations or the configuration of Service Broker services.
<a href="#">SQL Server Migration Assistant</a>	Use SQL Server Migration Assistant to automate database migration to SQL Server from Microsoft Access, DB2, MySQL, Oracle, and Sybase.

If you're looking for additional tools that aren't mentioned on this page, see [SQL Command Prompt Utilities](#) and [Download SQL Server extended features and tools](#)



# Azure Data Studio Troubleshooting

9/26/2022 • 2 minutes to read • [Edit Online](#)

Azure Data Studio tracks issues and feature requests using on a [GitHub repository issue tracker](#) for the `azuredatastudio` repository.

## If you've experienced any issue

Report issues to [GitHub Issue Tracker](#) and let us know any details that will help reproduce the error. Include any [log information](#) from the log file.

## Writing good bug reports and feature requests

File a single issue per problem and feature request.

- Don't enumerate multiple bugs or feature requests in the same issue.
- Don't add your issue as a comment to an existing issue unless it's for the identical input. Many issues look similar, but have different causes.

The more information you can provide, the more likely someone will be successful reproducing the issue and finding a fix.

Include the following information with each issue:

- Version of Azure Data Studio
- Reproducible steps (1... 2... 3...) and what you expected versus what you actually saw.
- Images, animations, or a link to a video. Images and animations illustrate repro-steps but don't replace them.
- A code snippet that demonstrates the issue or a link to a code repository we can easily pull down onto our machine to recreate the issue.

### NOTE

Because we need to copy and paste the code snippet, including a code snippet as a media file (i.e. .gif) is not sufficient.

- Errors in the Dev Tools Console (Help | Toggle Developer Tools)

Please remember to do the following:

- Search the issue repository to see if there exists a duplicate.
- Simplify your code around the issue so we can better isolate the problem.

Don't feel bad if we can't reproduce the issue and ask for more information!

## How to set the logging level

### Azure Data Studio

Run the `Developer: Set Log Level...` command to select the log level for the current session. This value is NOT persisted over multiple sessions - so if you restart Azure Data Studio it will revert back to the default `Info` level.

If you want to enable debug logging for startup then set the log level to `Debug` and run the `Developer: Reload Window` command

## MSSQL (Built-In Extension)

If the `Mssql: Log Debug Info` user setting is set to true, then debug log info will be sent to the `MSSQL` output channel.

The `Mssql: Tracing Level` user setting is used to control the verbosity of the logging.

## Debug log location

From Azure Data Studio, run the `Developer: Open Logs Folder` command to open the path to the logs. There's many different types of log files that write there, a few of the commonly used ones are:

1. `renderer#.log` (for example, `renderer1.log`) - this file is the log file for the main process.
2. `telemetry.log` - When the log level is set to `Trace` this file will contain the telemetry events sent by Azure Data Studio
3. `exthost#/exthost.log` - Log file for the extension host process (this is only the process itself, not the extensions running inside it)
4. `exthost#/Microsoft.mssql` - Logs for the mssql extension, which contains much of the core logic for MSSQL-related features
  - `sqltools.log` is the log for SQL Tools Service
5. `exthost#/output_logging #####` - these folders contain the messages displayed in the `Output` panel in Azure Data Studio. Each file is named `#-<Channel Name>` so for example the `Notebooks` output channel may output to a file named `3-Notebooks.log`.

If you are asked to provide logs, zip up the entire folder to ensure that the correct logs are included.

## Next Steps

- [Report an issue](#)
- [What is Azure Data Studio](#)

