SQL Server on Linux 2017

Product

All SQL

Version

Hide nothing

Analytics Platform System (PDW)

Version

2016

Azure SQL Data Warehouse

Version

latest

Azure SQL Database

Version

current

Azure SQL Database Managed Instance

Version

current

SQL Server

Version

2017

2016

2014

SQL Server Analysis Services

Version

2017

2016

SQL Server on Linux

Version

2017

Quickstart: Install SQL Server and create a database on Ubuntu

02/22/2018 6 minutes to read Contributors all

In this article

THIS TOPIC APPLIES TO: ✓ SQL Server (Linux only) ✗ Azure SQL Database ✗ Azure SQL Data Warehouse ✗ Parallel Data Warehouse

In this quickstart, you first install SQL Server 2017 on Ubuntu 16.04. Then connect with **sqlcmd** to create your first database and run queries.

Tip

This tutorial requires user input and an internet connection. If you are interested in the unattended or offline installation procedures, see Installation guidance for SQL Server on Linux.

Prerequisites

You must have a Ubuntu 16.04 machine with **at least 2 GB** of memory.

To install Ubuntu on your own machine, go to http://www.ubuntu.com/download/server. You can also create Ubuntu virtual machines in Azure. See Create and Manage Linux VMs with the Azure CLI.

Note

At this time, the Windows Subsystem for Linux for Windows 10 is not supported as an installation target.

For other system requirements, see System requirements for SQL Server on Linux.

Install SQL Server

To configure SQL Server on Ubuntu, run the following commands in a terminal to install the **mssql-server** package.

Important

If you have previously installed a CTP or RC release of SQL Server 2017, you must first remove the old repository before registering one of the GA repositories. For more information, see Change repositories from the preview repository to the GA repository.

1. Import the public repository GPG keys:

   bash Copy

   ```bash
   wget -qO- https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -
   ```

2. Register the Microsoft SQL Server Ubuntu repository:

   bash Copy

   ```bash
   sudo add-apt-repository "$(wget -qO- https://packages.microsoft.com/config/ubuntu/16.
   ```

   Note

   This is the Cumulative Update (CU) repository. For more information about your repository options and their differences, see Configure repositories for SQL Server on Linux.

3. Run the following commands to install SQL Server:

   bash Copy

   ```bash
   sudo apt-get update
   sudo apt-get install -y mssql-server
   ```

4. After the package installation finishes, run **mssql-conf setup** and follow the prompts to set the SA password and choose your edition.

   bash Copy

   ```bash
   sudo /opt/mssql/bin/mssql-conf setup
   ```

   Tip

If you are trying SQL Server 2017 in this tutorial, the following editions are freely licensed: Evaluation, Developer, and Express.

Note

Make sure to specify a strong password for the SA account (Minimum length 8 characters, including uppercase and lowercase letters, base 10 digits and/or non-alphanumeric symbols).

5. Once the configuration is done, verify that the service is running:

bash Copy

```bash
systemctl status mssql-server
```

6. If you plan to connect remotely, you might also need to open the SQL Server TCP port (default 1433) on your firewall.

At this point, SQL Server is running on your Ubuntu machine and is ready to use!

Install the SQL Server command-line tools

To create a database, you need to connect with a tool that can run Transact-SQL statements on the SQL Server. The following steps install the SQL Server command-line tools: sqlcmd and bcp.

Use the following steps to install the **mssql-tools** on Ubuntu.

1. Import the public repository GPG keys.

bash Copy

```bash
curl https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -
```

2. Register the Microsoft Ubuntu repository.

bash Copy

```bash
curl https://packages.microsoft.com/config/ubuntu/16.04/prod.list | sudo tee /etc/apt
```

3. Update the sources list and run the installation command with the unixODBC developer package.

   bash Copy

   ```bash
   sudo apt-get update
   sudo apt-get install mssql-tools unixodbc-dev
   ```

   Note

   To update to the latest version of **mssql-tools** run the following commands:

   bash Copy

   ```bash
   sudo apt-get update
   sudo apt-get install mssql-tools
   ```

4. **Optional**: Add `/opt/mssql-tools/bin/` to your **PATH** environment variable in a bash shell.

   To make **sqlcmd/bcp** accessible from the bash shell for login sessions, modify your **PATH** in the **~/.bash_profile** file with the following command:

   bash Copy

   ```bash
   echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bash_profile
   ```

   To make **sqlcmd/bcp** accessible from the bash shell for interactive/non-login sessions, modify the **PATH** in the **~/.bashrc** file with the following command:

   bash Copy

   ```bash
   echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc
   source ~/.bashrc
   ```

Tip

Sqlcmd is just one tool for connecting to SQL Server to run queries and perform management and development tasks. Other tools include:

- SQL Server Operations Studio (Preview)
- SQL Server Management Studio

- Visual Studio Code.
- mssql-cli (Preview)

Connect locally

The following steps use **sqlcmd** to locally connect to your new SQL Server instance.

1. Run **sqlcmd** with parameters for your SQL Server name (-S), the user name (-U), and the password (-P). In this tutorial, you are connecting locally, so the server name is `localhost`. The user name is `SA` and the password is the one you provided for the SA account during setup.

   bash Copy

   ```bash
   sqlcmd -S localhost -U SA -P '<YourPassword>'
   ```

   Tip

   You can omit the password on the command line to be prompted to enter it.

   Tip

   If you later decide to connect remotely, specify the machine name or IP address for the **-S** parameter, and make sure port 1433 is open on your firewall.

2. If successful, you should get to a **sqlcmd** command prompt: `1>`.

3. If you get a connection failure, first attempt to diagnose the problem from the error message. Then review the connection troubleshooting recommendations.

Create and query data

The following sections walk you through using **sqlcmd** to create a new database, add data, and run a simple query.

Create a new database

The following steps create a new database named `TestDB`.

1. From the **sqlcmd** command prompt, paste the following Transact-SQL command to create a test database:

SQL Copy

```sql
CREATE DATABASE TestDB
```

2. On the next line, write a query to return the name of all of the databases on your server:

SQL Copy

```sql
SELECT Name from sys.Databases
```

3. The previous two commands were not executed immediately. You must type `GO` on a new line to execute the previous commands:

SQL Copy

```sql
GO
```

Insert data

Next create a new table, `Inventory`, and insert two new rows.

1. From the **sqlcmd** command prompt, switch context to the new `TestDB` database:

SQL Copy

```sql
USE TestDB
```

2. Create new table named `Inventory`:

SQL Copy

```sql
CREATE TABLE Inventory (id INT, name NVARCHAR(50), quantity INT)
```

3. Insert data into the new table:

SQL Copy

```sql
INSERT INTO Inventory VALUES (1, 'banana', 150); INSERT INTO Inventory VALUES (2, 'or
```

4. Type `GO` to execute the previous commands:

   SQL

   ```
   GO
   ```

Select data

Now, run a query to return data from the `Inventory` table.

1. From the **sqlcmd** command prompt, enter a query that returns rows from the `Inventory` table where the quantity is greater than 152:

   SQL

   ```
   SELECT * FROM Inventory WHERE quantity > 152;
   ```

2. Execute the command:

   SQL

   ```
   GO
   ```

Exit the sqlcmd command prompt

To end your **sqlcmd** session, type `QUIT`:

SQL

```
QUIT
```

Connect from Windows

SQL Server tools on Windows connect to SQL Server instances on Linux in the same way they would connect to any remote SQL Server instance.

If you have a Windows machine that can connect to your Linux machine, try the same steps in this topic from a Windows command-prompt running **sqlcmd**. Just verify that you use the target Linux machine name or IP address rather than localhost, and make sure that TCP port 1433 is open. If you have any problems connecting from Windows, see connection troubleshooting recommendations.

For other tools that run on Windows but connect to SQL Server on Linux, see:

- SQL Server Management Studio (SSMS)
- Windows PowerShell
- SQL Server Data Tools (SSDT)

Additional resources

For other installation scenarios, see the following resources:

Upgrade	Learn how to upgrade an existing installation of SQL Server on Linux
Uninstall	Uninstall SQL Server on Linux
Unattended install	Learn how to script the installation without prompts
Offline install	Learn how to manually download the packages for offline installation

To explore other ways to connect and manage SQL Server, explore the following tools:

Visual Studio Code	A cross-platform GUI code editor that run Transact-SQL statements with the mssql extension.
SQL Server Operations Studio	A cross-platform GUI database management utility.
mssql-cli	A cross-platform command-line interface for running Transact-SQL commands.
SQL Server Management Studio	A Windows-based GUI database management utility that can connect to and manage SQL Server instances on Linux.

To learn more about writing Transact-SQL statements and queries, see Tutorial: Writing Transact-SQL Statements.

Tip

For answers to frequently asked questions, see the SQL Server on Linux FAQ.

Next steps

Explore the tutorials for SQL Server on Linux >

ⓘ **Note**

The feedback system for this content will be changing soon. Old comments will not be carried over. If content within a comment thread is important to you, please save a copy. For more information on the upcoming change, we invite you to read our blog post.