

Quickstart: Install SQL Server and create a database on Ubuntu

05/28/2019 • 7 minutes to read • +7

In this article

[Prerequisites](#)

[Install SQL Server](#)

[Install the SQL Server command-line tools](#)

[Connect locally](#)

[Create and query data](#)

[Performance best practices](#)

[Cross-platform data tools](#)

[Connecting from Windows](#)

[Other deployment scenarios](#)

[Next steps](#)

APPLIES TO: [SQL Server \(Linux only\)](#) [Azure SQL Database](#) [Azure SQL Data Warehouse](#) [Parallel Data Warehouse](#)

In this quickstart, you install SQL Server 2017 or SQL Server 2019 preview on Ubuntu 16.04. You then connect with **sqlcmd** to create your first database and run queries.

Tip

SQL Server 2017 ▾

Filter by title

TOOLS

> Tutorials

SQL Server on Linux

 SQL Server on Linux

> Overview

Quickstarts

 Install & Connect - Red Hat

 Install & Connect - SUSE

Install & Connect - Ubuntu

 Run & Connect - Docker

 Provision a SQL VM in Azure >

 Run & Connect - Cloud

> Tutorials

> Concepts

> Samples

> Resources

↓ Download PDF

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 16.04 on your own machine, go to <http://releases.ubuntu.com/xenial/>. 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](#).

ⓘ Note

Ubuntu 18.04 is not yet officially supported, but running SQL Server is possible with [modifications](#).

Install SQL Server

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

1. Import the public repository GPG keys:

```
bash
```



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

2. Register the Microsoft SQL Server Ubuntu repository:

```
bash
```



```
sudo add-apt-repository "$(wget -qO- https://packages.microsoft.com/confi
```

💡 Tip

If you want to try SQL Server 2019 , you must instead register the **Preview (2019)** repository. Use the following command for SQL Server 2019 installations:

```
bash
```



```
sudo add-apt-repository "$(wget -qO- https://packages.microsoft.com/confi
```

3. Run the following commands to install SQL Server:

```
bash
```

 Copy

```
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

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

Tip

The following SQL Server 2017 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

```
systemctl status mssql-server --no-pager
```

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

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

- ## 2. Register the Microsoft Ubuntu repository.

bash

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

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

```
bash
```

 Copy

```
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

```
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.

Sane Method

```
sudo ln -s /opt/mssql-tools/bin/* /usr/local/bin/
```

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

```
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

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

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

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



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



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

```
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

```
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
```

```
GO
```



Tip

To learn more about writing Transact-SQL statements and queries, see [Tutorial: Writing Transact-SQL Statements](#).

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
```

```
USE TestDB
```



2. Create new table named `Inventory`:

```
CREATE TABLE Inventory (
```

SQL

Copy

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

3. Insert data into the new table:

SQL

Copy

```
INSERT INTO Inventory VALUES (1, 'banana', 150); INSERT INTO Inventory V%
```

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

SQL

Copy

```
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

Copy

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

2. Execute the command:

```
SQL
GO
```

 Copy

Exit the `sqlcmd` command prompt

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

```
SQL
QUIT
```

 Copy

Performance best practices

After installing SQL Server on Linux, review the best practices for configuring Linux and SQL Server to improve performance for production scenarios. For more information, see [Performance best practices and configuration guidelines for SQL Server on Linux](#).

Cross-platform data tools

In addition to `sqlcmd`, you can use the following cross-platform tools to manage SQL Server:

Azure Data
Studio

A cross-platform GUI database management utility.

Visual Studio Code	A cross-platform GUI code editor that runs Transact-SQL statements with the mssql extension.
PowerShell Core	A cross-platform automation and configuration tool based on cmdlets.
mssql-cli	A cross-platform command-line interface for running Transact-SQL commands.

Connecting 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 \(SSDT\)](#)

Other deployment scenarios

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



Tip

For answers to frequently asked questions, see the [SQL Server on Linux FAQ](#).

Next steps

[Explore the tutorials for SQL Server on Linux](#)

Feedback

Send feedback about

[This product](#)

[This page](#)

