

# Deploy and connect to SQL Server Linux containers

Article • 01/10/2024


Applies to:  [SQL Server](#) - Linux

This article explains how to deploy and connect to SQL Server Linux containers.


For other deployment scenarios, see:

- [Windows](#)
- [Linux](#)
- [Container cluster on Azure](#)


## Note

This article specifically focuses on using the `mssql-server-linux` image. SQL Server deployments in Windows containers are not covered by support. For development and testing, you can create your own custom container images to work with SQL Server in Windows containers. Sample files are available on [GitHub](#) . Sample files are for reference only.

## Important

Before choosing to run a SQL Server container for production use cases, please review our [support policy for SQL Server Containers](#)  to ensure that you are running on a supported configuration.

This 6-minute video provides an introduction into running SQL Server on containers:

[https://channel9.msdn.com/Shows/Data-Exposed/SQL-Server-2019-in-Containers/player?WT.mc\\_id=dataexposed-c9-niner&nocookie=true&locale=en-us&embedUrl=%2Fsql%2Flinux%2Fsql-server-linux-docker-container-deployment](https://channel9.msdn.com/Shows/Data-Exposed/SQL-Server-2019-in-Containers/player?WT.mc_id=dataexposed-c9-niner&nocookie=true&locale=en-us&embedUrl=%2Fsql%2Flinux%2Fsql-server-linux-docker-container-deployment) 

## Pull and run the container image

To pull and run the Docker container images for SQL Server, follow the prerequisites and steps in the following quickstart:

- [Run the SQL Server 2017 container image with Docker](#)

- [Run the SQL Server 2019 container image with Docker](#)
- [Run the SQL Server 2022 container image with Docker](#)

This configuration article provides additional usage scenarios in the following sections.

## Connect and query

You can connect and query SQL Server in a container from either outside the container or from within the container. The following sections explain both scenarios.

### Tools outside the container

You can connect to the SQL Server instance on your container host from any external Linux, Windows, or macOS tool that supports SQL connections. Some common tools include:

- [sqlcmd](#)
- [Azure Data Studio](#)
- [Visual Studio Code](#)
- [SQL Server Management Studio \(SSMS\) on Windows](#)

The following example uses **sqlcmd** to connect to SQL Server running in a container. The IP address in the connection string is the IP address of the host machine that is running the container.

#### ⓘ Note

Newer versions of **sqlcmd** (in **mssql-tools18**) are secure by default. If using version 18 or higher, you need to add the **No** option to **sqlcmd** to specify that encryption is optional, not mandatory.

Bash

```
sqlcmd -S 10.3.2.4 -U SA -P '<YourPassword>'
```

If you mapped a host port that wasn't the default **1433**, add that port to the connection string. For example, if you specified **-p 1400:1433** in your **docker run** command, then connect by explicitly specifying port 1400.

Bash

```
sqlcmd -S 10.3.2.4,1400 -U SA -P '<YourPassword>'
```

## Tools inside the container

Starting with SQL Server 2017 (14.x), the [SQL Server command-line tools](#) are included in the container image. If you attach to the image with an interactive command-prompt, you can run the tools locally.

1. Use the `docker exec -it` command to start an interactive bash shell inside your running container. In the following example `e69e056c702d` is the container ID.

Bash

```
docker exec -it e69e056c702d "bash"
```

### 💡 Tip

You don't always have to specify the entire container ID. You only have to specify enough characters to uniquely identify it. So in this example, it might be enough to use `e6` or `e69` rather than the full ID. To find out the container ID, run the command `docker ps -a`.

2. Once inside the container, connect locally with `sqlcmd` by using its full path.

Bash

```
/opt/mssql-tools18/bin/sqlcmd -S localhost -U SA -P '<YourPassword>'
```

### ⓘ Note

Newer versions of `sqlcmd` are secure by default. For more information about connection encryption, see [sqlcmd utility](#) for Windows, and [Connecting with sqlcmd](#) for Linux and macOS. If the connection doesn't succeed, you can add the `-No` option to `sqlcmd` to specify that encryption is optional, not mandatory.

3. When finished with `sqlcmd`, type `exit`.

- When finished with the interactive command-prompt, type `exit`. Your container continues to run after you exit the interactive bash shell.

## Check the container version

If you want to know the version of SQL Server in a running container, run the following command to display it. Replace `<Container ID or name>` with the target container ID or name. Replace `<YourStrong!Passw0rd>` with the SQL Server password for the system administrator (SA) account.

Bash

```
sudo docker exec -it <Container ID or name> /opt/mssql-tools/bin/sqlcmd \  
-S localhost -U SA -P '<YourStrong!Passw0rd>' \  
-Q 'SELECT @@VERSION'
```

### ⓘ Note

Newer versions of `sqlcmd` are secure by default. For more information about connection encryption, see [sqlcmd utility](#) for Windows, and [Connecting with sqlcmd](#) for Linux and macOS. If the connection doesn't succeed, you can add the `-N` option to `sqlcmd` to specify that encryption is optional, not mandatory.

You can also identify the SQL Server version and build number for a target container image. The following command displays the SQL Server version and build information for the `mcr.microsoft.com/mssql/server:2022-latest` image. It does this by running a new container with an environment variable `PAL_PROGRAM_INFO=1`. The resulting container instantly exits, and the `docker rm` command removes it.

Bash

```
sudo docker run -e PAL_PROGRAM_INFO=1 --name sqlver \  
-ti mcr.microsoft.com/mssql/server:2022-latest && \  
sudo docker rm sqlver
```

The previous commands display version information similar to the following output:

Output

```
sqlservr  
Version 16.0.1000.6
```

Build ID d81e9b6de06534e649bd57dd609aa3050f5e380f361b7f8a80a80ee-b71e7422c

Build Type release

Git Version 2aede92f

Built at Tue Nov 01 06:11:40 GMT 2022

PAL

Build ID 754097e8f0db68f559e1cbc9d46952ac9fd518b5da9f12964e-f40fc9033720e3

Build Type release

Git Version d88e3e1130

Built at Tue Nov 01 06:08:02 GMT 2022

Packages

system.security	mssql-16.0.1000.6_26_offi-
cial-release	
system.certificates	mssql-16.0.1000.6_26_offi-
cial-release	
sqlagent	16.0.1000.6
system.wmi	10.0.17763.2061.202107231
system.netfx	4.7.0.0.202104262
system	mssql-16.0.1000.6_26_offi-
cial-release	
system.common	10.0.17763.2061.202107231
sqlservr	16.0.1000.6
secforwarderxplat	16.0.1000.6

## Run a specific SQL Server container image

### ⓘ Note

- Starting with SQL Server 2019 (15.x) CU3, Ubuntu 18.04 is supported.
- Starting with SQL Server 2019 (15.x) CU10, Ubuntu 20.04 is supported.
- You can retrieve a list of all available tags for mssql/server at <https://mcr.microsoft.com/v2/mssql/server/tags/list>.

There are scenarios where you might not want to use the latest SQL Server container image. To run a specific SQL Server container image, use the following steps:

1. Identify the Docker `tag` for the release you want to use. To view the available tags, see the [Microsoft Artifact Registry](#).
2. Pull the SQL Server container image with the tag. For example, to pull the `2019-CU18-ubuntu-20.04` image, replace `<image_tag>` in the following command with `2019-CU18-ubuntu-20.04`.

Bash

```
docker pull mcr.microsoft.com/mssql/server:<image_tag>
```

3. To run a new container with that image, specify the tag name in the `docker run` command. In the following command, replace `<image_tag>` with the version you want to run.

#### Important

The `SA_PASSWORD` environment variable is deprecated. Use `MSSQL_SA_PASSWORD` instead.

Bash

```
docker run -e 'ACCEPT_EULA=Y' -e 'MSSQL_SA_PASSWORD=  
<YourStrong!Passw0rd>' -p 1401:1433 -d  
mcr.microsoft.com/mssql/server:<image_tag>
```

These steps can also be used to downgrade an existing container. For example, you might want to roll back or downgrade a running container for troubleshooting or testing. To downgrade a running container, you must be using a persistence technique for the data folder. Follow the same steps outlined in the [upgrade section](#), but specify the tag name of the older version when you run the new container.

## Run RHEL-based container images

The documentation for SQL Server Linux container images points to Ubuntu-based containers. Beginning with SQL Server 2019 (15.x), you can use containers based on Red Hat Enterprise Linux (RHEL). An example of the image for RHEL will look like

```
mcr.microsoft.com/mssql/rhel/server:2019-CU15-rhel-8.
```

For example, the following command pulls the Cumulative Update 18 for SQL Server 2019 (15.x) container that uses RHEL 8:

Bash

```
sudo docker pull mcr.microsoft.com/mssql/rhel/server:2019-CU18-rhel-  
8.4
```

# Run production container images

The [quickstart](#) in the previous section runs the free Developer edition of SQL Server from the Microsoft Artifact Registry. Most of the information still applies if you want to run production container images, such as Enterprise, Standard, or Web editions. However, there are a few differences that are outlined here.

- You can only use SQL Server in a production environment if you have a valid license. You can obtain a free SQL Server Express production license [here](#). SQL Server Standard and Enterprise edition licenses are available through [Microsoft Volume Licensing](#).
- The Developer container image can be configured to run the production editions as well.

To run a production edition, review the requirements and run procedures in the [quickstart](#). You must specify your production edition with the `MSSQL_PID` environment variable. The following example shows how to run the latest SQL Server 2022 (16.x) container image for the Enterprise Core edition.

Bash

```
docker run --name sqlenterprise \  
-e 'ACCEPT_EULA=Y' -e 'MSSQL_SA_PASSWORD=<YourStrong!Passw0rd>' \  
-e 'MSSQL_PID=EnterpriseCore' -p 1433:1433 \  
-d mcr.microsoft.com/mssql/server:2022-latest
```

## Important

By passing the value `Y` to the environment variable `ACCEPT_EULA` and an edition value to `MSSQL_PID`, you are expressing that you have a valid and existing license for the edition and version of SQL Server that you intend to use. You also agree that your use of SQL Server software running in a container image will be governed by the terms of your SQL Server license.

## Note

For a full list of possible values for `MSSQL_PID`, see [Configure SQL Server settings with environment variables on Linux](#).

# Run multiple SQL Server containers

Docker provides a way to run multiple SQL Server containers on the same host machine. Use this approach for scenarios that require multiple instances of SQL Server on the same host. Each container must expose itself on a different port.

The following example creates two SQL Server 2022 (16.x) containers and maps them to ports `1401` and `1402` on the host machine.

Bash

```
docker run -e 'ACCEPT_EULA=Y' -e 'MSSQL_SA_PASSWORD=<YourStrong!Pass-w0rd>' -p 1401:1433 -d mcr.microsoft.com/mssql/server:2022-latest
docker run -e 'ACCEPT_EULA=Y' -e 'MSSQL_SA_PASSWORD=<YourStrong!Pass-w0rd>' -p 1402:1433 -d mcr.microsoft.com/mssql/server:2022-latest
```

Now there are two instances of SQL Server running in separate containers. Clients can connect to each SQL Server instance by using the IP address of the container host and the port number for the container.

## ⓘ Note

Newer versions of `sqlcmd` (in `mssql-tools18`) are secure by default. If using version 18 or higher, you need to add the `No` option to `sqlcmd` to specify that encryption is optional, not mandatory.

Bash

```
sqlcmd -S 10.3.2.4,1401 -U SA -P '<YourPassword>'
sqlcmd -S 10.3.2.4,1402 -U SA -P '<YourPassword>'
```

# Upgrade SQL Server in containers

To upgrade the container image with Docker, first identify the tag for the release for your upgrade. Pull this version from the registry with the `docker pull` command:

command

```
docker pull mcr.microsoft.com/mssql/server:<image_tag>
```



This updates the SQL Server image for any new containers you create, but it doesn't update SQL Server in any running containers. To do this, you must create a new container with the latest SQL Server container image and migrate your data to that new container.


1. Make sure you are using one of the [data persistence techniques](#) for your existing SQL Server container. This enables you to start a new container with the same data.
2. Stop the SQL Server container with the `docker stop` command.
3. Create a new SQL Server container with `docker run` and specify either a mapped host directory or a data volume container. Make sure to use the specific tag for your SQL Server upgrade. The new container now uses a new version of SQL Server with your existing SQL Server data.

#### Important

Upgrade is only supported between RC1, RC2, and GA at this time.

4. Verify your databases and data in the new container.
5. Optionally, remove the old container with `docker rm`.

## Related content

- Get started with SQL Server 2022 (16.x) container images on Docker by going through the [quickstart](#)
- [Reference additional configuration and customization to Docker containers](#)
- See the [mssql-docker GitHub repository](#)  for resources, feedback, and known issues
- [Troubleshooting SQL Server Docker containers](#)
- [Explore high availability for SQL Server containers](#)
- [Secure SQL Server Docker containers](#)

## 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](#)

---

# Feedback

Was this page helpful?



Yes



No

[Provide product feedback](#)  | [Get help at Microsoft Q&A](#)