**MySQL** is a well-known open-source relational database management system and one of the most popular web server solutions. It stores and structures data in a meaningful manner, ensuring easy accessibility.

**Docker**is a set of platform-as-a-service products that support CI/CD development. It allows users to develop and deploy applications inside virtual environments, called containers. With a single image, Docker can boot up an application with all its libraries and dependencies.

**In this tutorial, learn how to deploy a MySQL Docker container and start working with the containerized database.**

**Running a MySQL Docker Container**

If you need to set up a database quickly and without using up too many resources, **deploying MySQL in a container** is a fast and efficient solution. This is only appropriate for small and medium-sized applications. Enterprise-level applications would not find a **MySQL Docker container** sufficient for their workload.

Using the Docker software for setting up your database is becoming increasingly popular for small-scale apps. Instead of having a separate server for database hosting, you can deploy a MySQL database container.

Multiple containers can run on your computer. The containers share the same kernel and libraries of the host while packaging the deployed application or software into single units. This makes the database extraordinarily lightweight and fast to spin up.

**Installing a MySQL Docker Container**

Setting up a database in Docker is simply building a container based on a MySQL image. Follow the steps outlined below to get your MySQL container up and running.

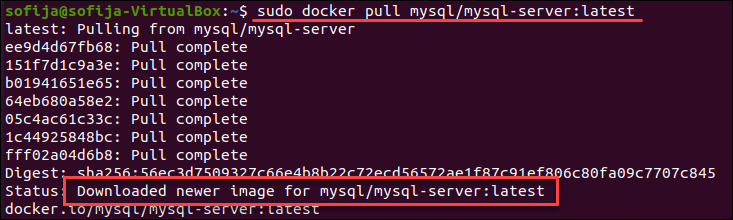
**Note:** This tutorial assumes you already have Docker on your system. If you don't have the software, take a look at one of our articles on how to [install Docker on CentOS](https://phoenixnap.com/kb/how-to-install-docker-centos-7), [installing Docker on Ubuntu](https://phoenixnap.com/kb/how-to-install-docker-on-ubuntu-18-04), or Docker guides for other operating systems.

**Step 1: Pull the MySQL Docker Image**

1. Start by pulling the appropriate Docker image for MySQL. You can download a specific version or opt for the latest release as seen in the following command:

**sudo docker pull mysql/mysql-server:latest**

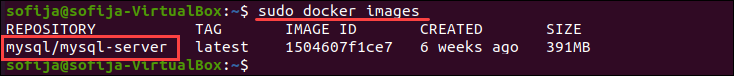
If you want a particular version of MySQL, replace **latest**with the version number.



2. Verify the image is now stored locally by [listing the downloaded Docker images](https://phoenixnap.com/kb/list-of-docker-commands-cheat-sheet#htoc-docker-commnads-for-container-and-image-information):

**sudo docker images**

The output should include **mysql/mysql-server** among the listed images.



**Step 2: Deploy the MySQL Container**

1. Once you have the image, move on to deploying a new MySQL container with:

**sudo docker run --name=[container\_name] -d [image\_tag\_name]**

* Replace **[container\_name]** with the name of your choice. If you do not provide a name, Docker generates a random one.
* The **-d** option instructs Docker to run the container as a service in the background.
* Replace **[image\_tag\_name]** with the name of the image downloaded in Step 1.

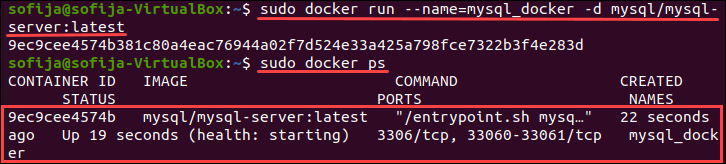
In this example, we create a container named **mysql\_docker** with the **latest** version tag:

**sudo docker run --name=[container\_name] -d mysql/mysql-server:latest**

2. Then, check to see if the MySQL container is running:

**docker ps**

You should see the newly created container listed in the output. It includes container details, one being the status of this virtual environment. The status changes from **health: starting** to **healthy**, once the setup is complete.



**Step 3: Connect to the MySQL Docker Container**

1. Before you can connect the MySQL server container with the host, you need to make sure the MySQL client package is installed:

**apt-get install mysql-client**

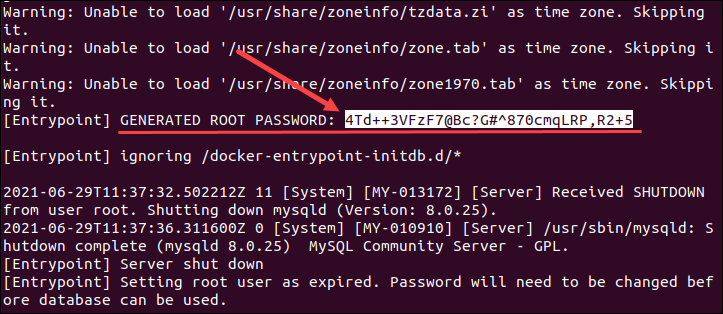
2. Then, open the logs file for the MySQL container to find the generated root password:

**sudo docker logs [container\_name]**

For the **mysql\_docker** container, we run:

**sudo docker logs mysql\_docker**

3. Scroll through the output and find the line **[Entrypoint] GENERATED ROOT PASSWORD**: , copy and paste the password in a notepad or text editor so you can use it later.



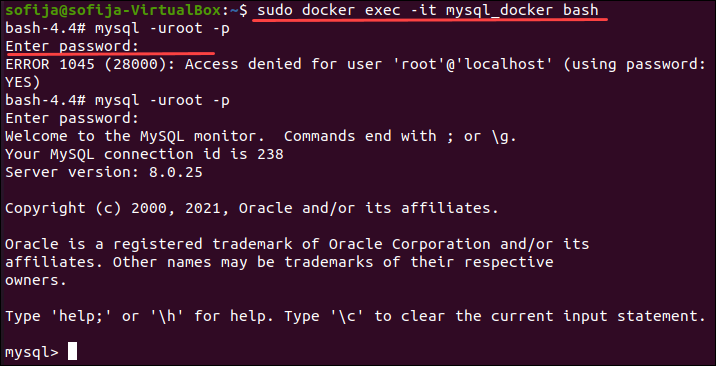
4. Next, go to the bash shell of the MySQL container by typing:

**sudo docker exec -it [container\_name] bash**

For the container created as an example, we run:

**sudo docker -it mysql\_docker bash**

3. Provide the root password you copied from the logs file, when prompted. With that, you have connected the MySQL client to the server.



4. Finally, change the server root password to protect your information:

mysql> **ALTER USER 'root'@'localhost' IDENTIFIED BY '[newpassword]';**

Replace **[newpassword]** with a strong password of your choice.

ChangeMySQL root password for MySQL container.