# **Getting a Local MySQL Database Up and Running**

## **Overview**

In this guide you’ll learn how to:

* Run a MySQL database locally using Docker.
* Verify that the database container is running.
* Connect to the database using VS Code (with the SQLTools extension).
* Create the essential users table required for your backend authentication.
* Set up a .gitignore file to prevent sensitive and unnecessary files from being committed to version control.

This local database will be used during development, and later you can switch to an Azure Database for MySQL with minimal changes.

## **1. Prerequisites**

* **Docker Desktop:** Ensure Docker is installed and running on your computer.  
   Download Docker Desktop
* **VS Code:** Install Visual Studio Code.  
   [Download VS Code](https://code.visualstudio.com/)
* **SQLTools Extension for VS Code:** This extension allows you to connect and manage databases directly from within VS Code.
* **Git:** Ensure Git is installed so you can manage version control effectively.

## **2. Launching a Local MySQL Instance with Docker**

### **A. Start the MySQL Container**

Open your terminal (Command Prompt, PowerShell, or your preferred shell) and run the following command:

docker run --name local-mysql -e MYSQL\_ROOT\_PASSWORD="password" -e MYSQL\_DATABASE=authdb -p 3306:3306 -d mysql:8.0

**Explanation:**

* **--name local-mysql**: Names the container “local-mysql.”
* **-e MYSQL\_ROOT\_PASSWORD="password"**: Sets the root password to password.
  + OBVIOUSLY YOU CHANGE THIS!!!
* **-e MYSQL\_DATABASE=authdb**: Automatically creates a database named authdb.
* **-p 3306:3306**: Maps port 3306 on your host machine to port 3306 in the container.
* **-d mysql:8.0**: Runs the MySQL version 8.0 image in detached mode.

### **B. Verify the Container is Running**

Run this command to list running Docker containers:

docker ps

You should see your local-mysql container in the list. To check logs for troubleshooting, run:

docker logs local-mysql

## **3. Connecting to the Database with VS Code**

### **A. Install the SQLTools Extension**

1. Open VS Code.
2. Click on the **Extensions** icon in the sidebar (or press Ctrl+Shift+X).
3. Search for **SQLTools** and install it.
4. Also install the **SQLTools MySQL/MariaDB Driver** extension.

### **B. Configure the Database Connection**

1. Press Ctrl+Shift+P to open the Command Palette.
2. Type SQLTools: Add new connection and select it.
3. Choose **MySQL/MariaDB** as the connection type.
4. Enter the following connection details:
   * **Connection Name:** Local MySQL
   * **Server/Host:** localhost
   * **Port:** 3306
   * **Username:** root
   * **Password:** password
   * **Database:** authdb (optional; sets a default database)
5. Save the connection.  
    You should now see the “Local MySQL” connection in the SQLTools panel.

## **4. Creating the users Table**

### **A. Open a New Query**

1. In the SQLTools panel, connect to your **Local MySQL** database.
2. Click the **New Query** button (or use the Command Palette and select SQLTools: New Query).

### **B. Create the Table**

Paste the following SQL command into the query editor:

CREATE TABLE IF NOT EXISTS users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL UNIQUE,

password VARCHAR(255) NOT NULL

);

Click the **Run Query** button (often represented by a play icon) to execute the command. This creates a users table with:

* An auto-incrementing id
* A unique username
* A password field to store the hashed password

### **C. Verify the Table**

Refresh your database view in SQLTools to ensure the users table now exists.

## **5. Setting Up the .gitignore File**

When developing your project, it’s important to keep sensitive files (like .env) and unnecessary files (like node\_modules) out of your version control system. To do this:

1. In your project’s root directory (the same directory that contains your backend folder), create a file named .gitignore.

Add the following lines to .gitignore:  
  
# Node modules directory

node\_modules/

# Environment variables

.env

# Docker files (if any)

docker-compose.yml

# Logs

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*

Save the file.

**Explanation:**

* **node\_modules/:** Prevents committing large dependency folders.
* **.env:** Protects sensitive configuration data (database credentials, JWT secret, etc.).
* **Docker files/logs:** Excludes files that are not necessary for the project’s source code.

## **6. Final Steps and Troubleshooting**

* **Docker Issues:** If the container isn’t running, confirm that Docker Desktop is active. Check logs with docker logs local-mysql for error details.
* **VS Code Connection:** If you have trouble connecting via SQLTools, verify that your connection details (host, port, username, password) are correct.
* **Table Creation:** Ensure that your SQL command runs without errors. Double-check that the database (authdb) exists and that you’re connected to the correct instance.

**Git Setup:** After creating your .gitignore, initialize a Git repository (if not already done) and commit your changes:  
  
git init

git add .

git commit -m "Initial commit with local database setup and .gitignore"

## **Conclusion**

By following this guide, you have successfully set up a local MySQL database using Docker, connected to it with VS Code, created a users table, and configured a .gitignore file to secure your sensitive files. With your local database now running, you’re ready to proceed to the backend setup and start building and testing your authentication system.

Happy coding!