Installation

Contents

This section explains why certain Python libraries and technologies are used in this project, and provides step-by-step instructions for setting up the project. Python must be installed previously as well as an integrated development environment (IDE).

Prerequisites

Before you begin, ensure you have met the following requirements:

- **Operating System:** Windows 10 version 2004 and higher (Build 19041 and higher) or Windows 11

- **Python:** 3.13
- **Dependencies:** WSL 2.

Installation

Follow these steps to install **etl-ws-1**:

1. Clone the repository:

```
cd git clone https://github.com/ntlg72/etl-ws-1.git
```

2. Navigate to the project directory:

```
cd etl-ws-1
```

Python Virtual Environment & Dependencies

Virtual environments are essential for modern Python development, providing isolated spaces for each project to manage dependencies and avoid conflicts. By creating a dedicated virtual environment, projects gain their own set of installed packages, separate from the system's Python installation and other projects, preventing version clashes and namespace pollution. This isolation enables reproducible builds and simplifies project setup and deployment.

Implementation

1. In the project directory, use the following command to create the virtual environment:

```
py -m venv <environment_name>
```

2. The invocation of the activation script is platform-specific (_<*venv*>_ must be replaced by the path to the directory containing the virtual environment):

Commands to activate virtual environment

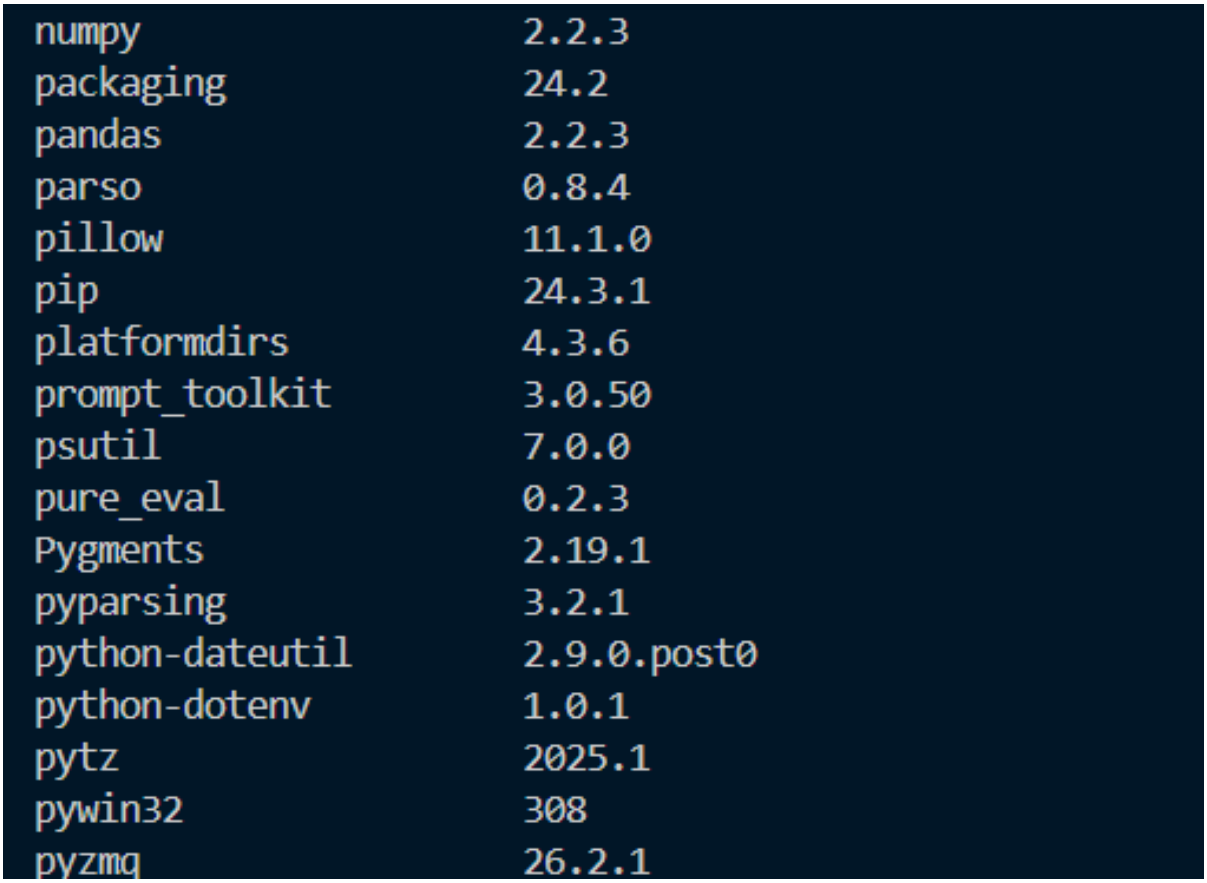
Platform	Shell	Command to activate virtual environment
Windows	cmd.exe	C:\> <venv>\Scripts\activate.bat
Windows	PowerShell	PS C:\> <venv>\Scripts\Activate.ps1

3. The project directory contains a *requirements.txt* file listing all necessary dependencies. To install them, while the virtual environment is activated, run:

```
pip install -r requirements.txt
```

You can check the installed dependencies using:

```
pip list
```



Installing WSL 2 and Docker for MySQL Deployment

WSL 2 (Windows Subsystem for Linux 2) provides a lightweight, virtualized Linux environment that integrates seamlessly with Windows, enabling developers to run Linux based tools and applications with improved performance and compatibility. Using a Dockerized MySQL image within WSL 2 allows for consistent, isolated, and portable development environments, which can be easily managed and shared. This approach ensures that the database setup is consistent across different development machines and avoids potential conflicts with other local services or applications.

> **Note**
>
> - A Dockerized MySQL image is preferred over a local installation because it offers better isolation (preventing dependency conflicts), simplified management (easy start/stop/remove), environment consistency (reducing deployment issues), and streamlined updates/maintenance (easy version control and rollback).
> - WSL 2 is used in this case because it provides a Linux kernel running within Windows, enabling Docker Desktop to efficiently run Linux containers (like the MySQL image) using a lightweight virtual machine.

Enabling WSL 2

1. Open PowerShell as Administrator.
2. Run:

```
wsl --install
```

3. Set WSL 2 as the default version:

```
wsl --set-default-version 2
```

Installing Ubuntu

1. Run the following command in PowerShell:

```
wsl.exe --install -d Ubuntu-24.04
```

2. Launch Ubuntu from the Start menu and complete the installation by creating a new user account.

Turning on Docker Desktop WSL 2

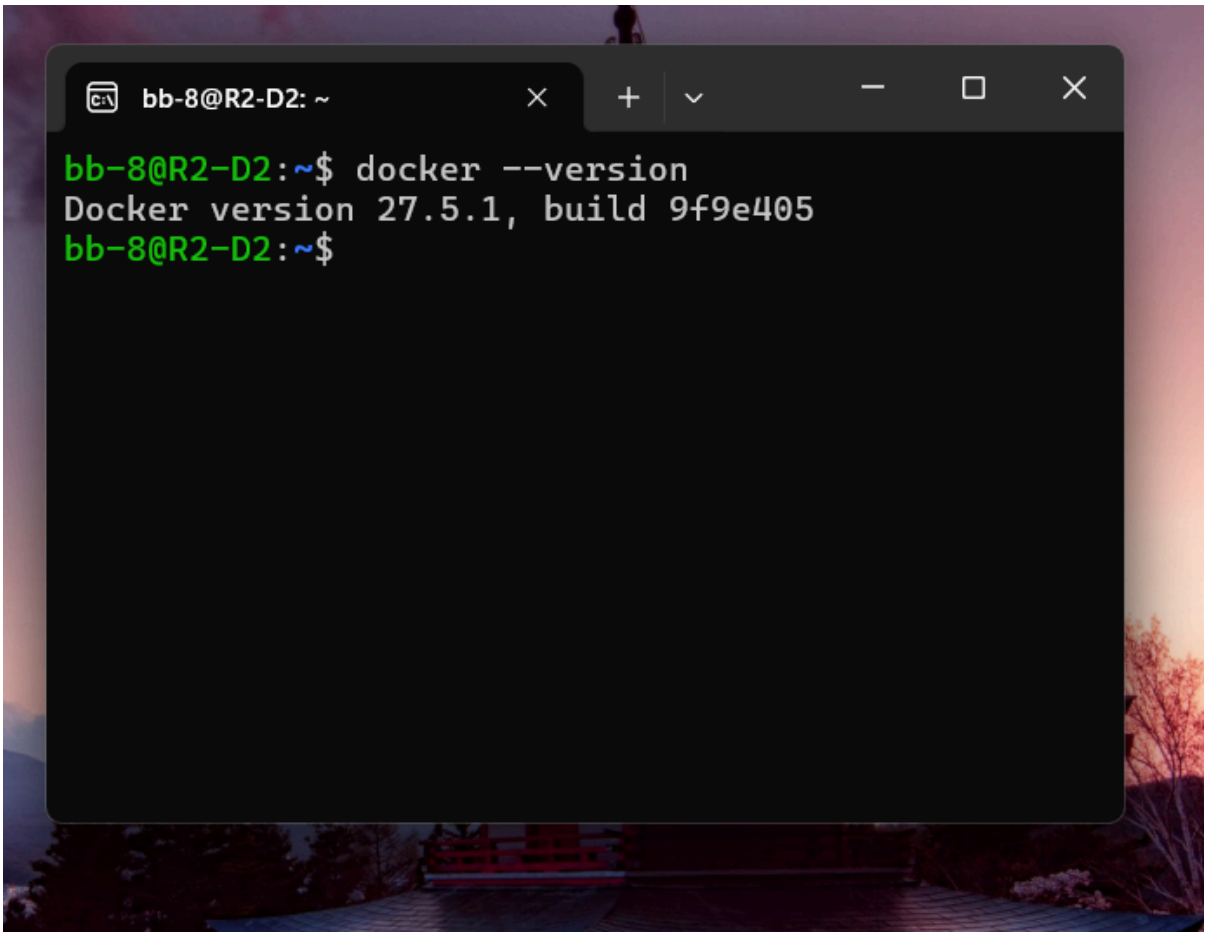
Important: Uninstall any previous versions of Docker Engine and CLI installed through Linux distributions.

1. Download and install the latest Docker Desktop for Windows.
2. Follow the installation instructions and enable WSL 2 when prompted.
3. Start Docker Desktop.
4. Navigate to **Settings > General** and select **Use WSL 2 based engine**.
5. Click **Apply & Restart**.

Confirming Docker Installation

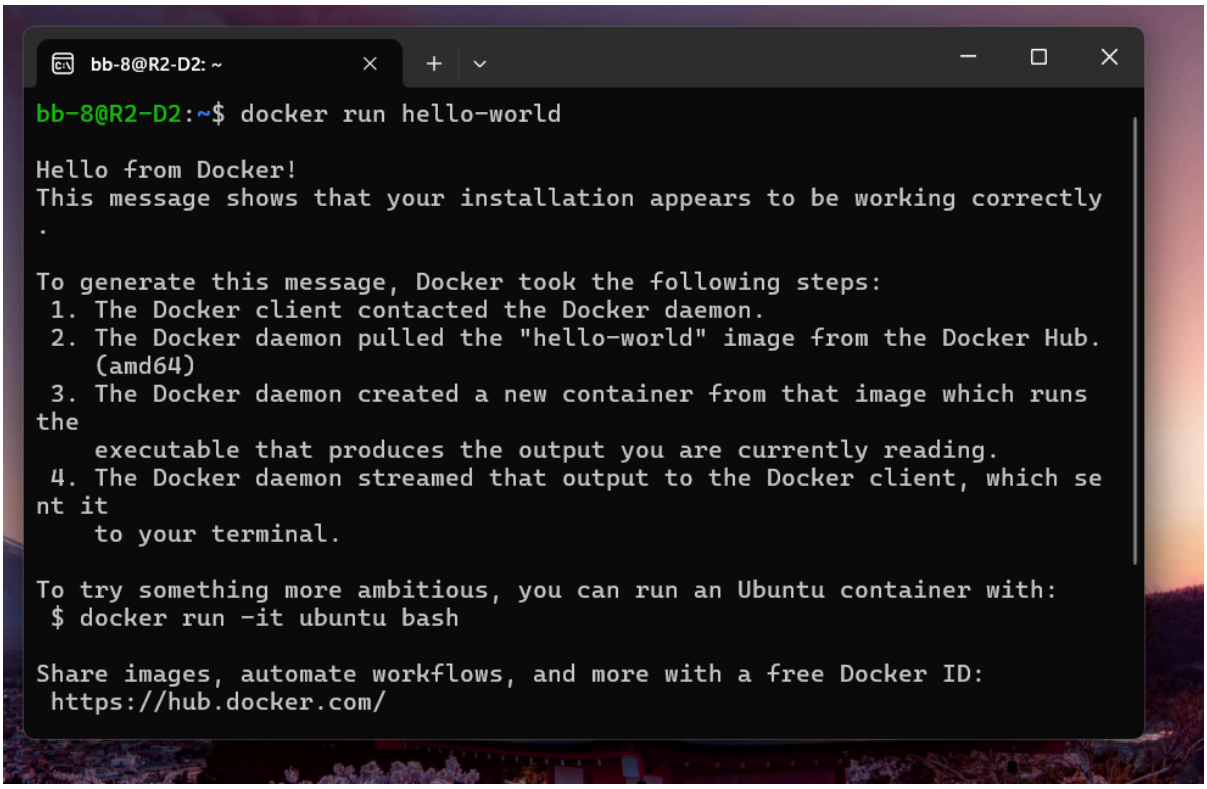
1. Open a WSL distribution (Ubuntu-24.04).
2. Display the version and build number by entering:

```
docker --version
```



3. Test the installation by running a simple built-in Docker image:

```
docker run hello-world
```



Setting Up MySQL Database with Docker

1. **Pull MySQL Image:**

   Open your Ubuntu 24.04 terminal and run the following command to pull the MySQL image:

   ```
   docker pull mysql
   ```

2. **Run MySQL Container:**

   Run the MySQL container and create a new database named *ws_001*:

   ```
   docker run -d --name mysql-container -e MYSQL_ROOT_PASSWORD=your_password -e MY
   ```

   Explanation:

   - *docker run -d* → Runs the container in detached mode (background).
   - *–name mysql-container* → Names the container *mysql-container*.
   - *-e MYSQL_ROOT_PASSWORD=your_password* → Sets the MySQL root password.
   - *-e MYSQL_DATABASE=ws_001* → Creates a default database named *ws_001*.
   - *-p 3307:3306* → Maps port *3307* on the host to *3306* inside the container.
     - **`3307` (Host Port):** This is the port on your _host machine_ (your WSL2 Ubuntu instance in this case) that you will use to access the MySQL server running inside the Docker container.
     - **`3306` (Container Port):** This is the port that the MySQL server is _listening on inside the Docker container_. MySQL's default port is 3306, and it's very likely that your MySQL Docker image is configured to use this default.
   - *mysql* → Uses the latest MySQL image from Docker Hub.

     Note that *mysql* is the name you want to assign to your container, and *your_password* is the password to be set for the MySQL root user.

3. **Check if the container is running:**

   ```
   docker ps
   ```

```
bb-8@R2-D2:~$ docker ps
CONTAINER ID   IMAGE     COMMAND
  CREATED        STATUS                 PORTS
                                NAMES
82f01f5a3fa8   mysql     "docker-entrypoint.s…"
  45 hours ago   Up 8 hours (Paused)   33060/tcp
, 0.0.0.0:3307->3306/tcp   mysql-container
```

5. **Access MySQL Container:**

   Access the MySQL container's shell:

   ```
   docker exec -it mysql-container mysql -u root -p
   ```

   Then, enter your password (*your_password*) to access the MySQL shell.

Usage

Setting up a .env file for MySQL Credentials in WSL2 Ubuntu 24.04

A *.env* file is needed to store your MySQL credentials securely, including the WSL2 IP address and the password set up.

1. **Locate the project directory:**

   Navigate to the directory where this repository has been cloned. This is where you'll create the *.env* file. In the terminal, it can be done through the following commands:

   ```
   cd /path/to/cloned/repository/directory
   ```

2. **Create the .env file:**

   In the project directory, create a new file named *.env* (no file extension). You can do this from the command line:

   ```
   touch .env
   ```

   Or using a text editor.

3. **Add your MySQL credentials to the .env file:**

Open the *.env* file with a text editor and add the following lines, replacing the placeholders with your actual values:

```
MYSQL_USER=root
MYSQL_PASSWORD=your_mysql_password
MYSQL_HOST=your_wsl2_ip_address
MYSQL_DATABASE=ws_001
MYSQL_PORT=3307
```

- **`MYSQL_USER`:** Your MySQL username.
- **`MYSQL_PASSWORD`:** The password you set for your MySQL user.
- **`MYSQL_HOST`:** This is _crucial_. You need the IP address of your WSL2 instance. See step 4 below to find this.
- **`MYSQL_DATABASE`:** The MySQL database created with the Docker command.
- **`MYSQL_PORT`:** The port MySQL is listening on. The one 3307.

4. **Find your WSL2 IP Address:**

   There are several ways to find the IP address of your WSL2 instance:

   - **From WSL:** Open your WSL2 terminal and run:

     ```
     ip addr show eth0 | grep "inet\b" | awk '{print $2}' | cut -d/ -f1
     ```

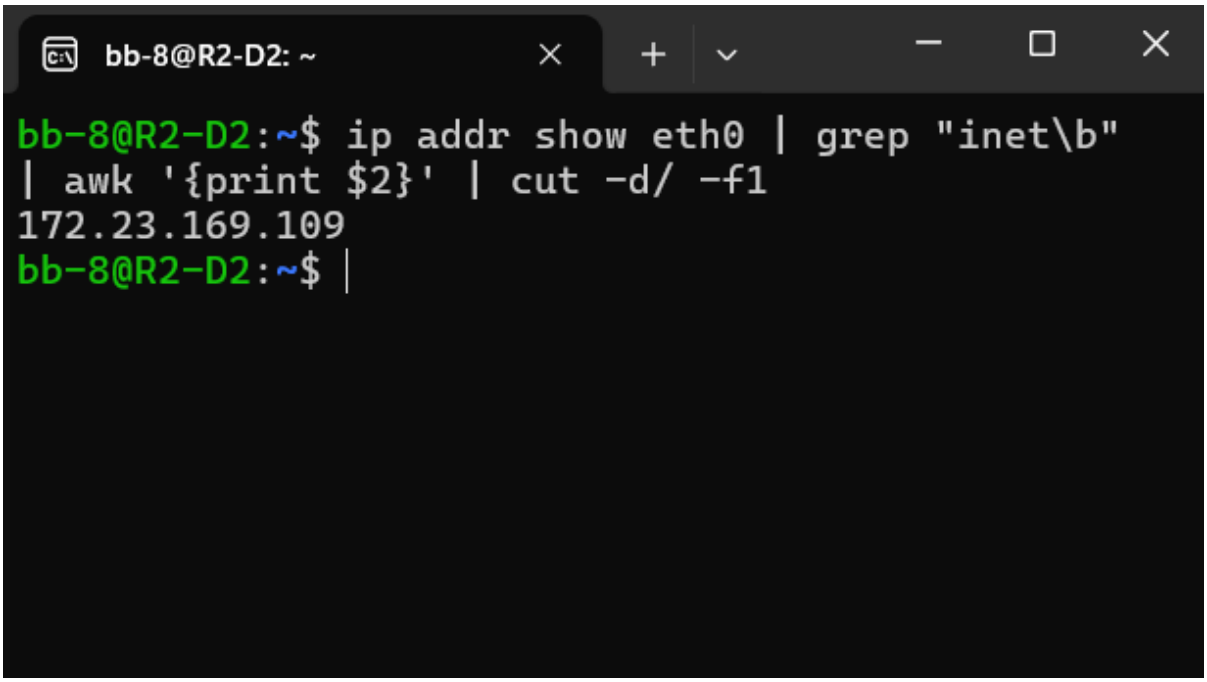
   - **From Windows (PowerShell):** Open PowerShell as administrator and run:

     ```
     wsl hostname -I
     ```

   - **From Windows (Command Prompt):** Open command prompt and run:

     ```
     wsl hostname -I
     ```

   The output will be the IP address of your WSL2 instance. Use this IP address for *MYSQL_HOST* in your *.env* file.



5. **Secure the .env file:**

   The *.env* file contains sensitive information. It's _extremely important_ to prevent it from being accidentally committed to version control (like Git). Add *.env* to your *.gitignore* file:

```
.env
```