

Getting Started with WSL

The Windows Subsystem for Linux (WSL) allows you to run a genuine Linux environment directly on your Windows machine. This is incredibly useful for developers, as it provides access to Linux-native tools and workflows without the need for a separate virtual machine or dual-booting. Modern WSL (WSL 2) runs a full Linux kernel, offering excellent performance and compatibility.

Installation is straightforward. Open a **Windows PowerShell** or **Command Prompt** as an **Administrator** and run the following command. This will, by default, install WSL and the latest Ubuntu distribution:

```
wsl --install -d ubuntu
```

After a restart, your Linux distribution will be available from the Start Menu. On first launch, you will be prompted to create a new username and password, which are specific to your Linux environment.



Essential WSL and Linux Commands

To work effectively, you need to be comfortable with the command-line interface. Your Linux environment is managed through a shell (usually **Bash**). Here are the foundational commands you will use daily.

For **navigating and viewing** your filesystem, the three most important commands are `pwd` (print working directory), which shows your current location; `ls` (list), which lists all files and directories in your current location; and `cd` (change directory), which allows you to move (e.g., `cd my_project` to enter a folder, or `cd ..` to go up one level).

For **managing files and directories**, you will use `mkdir <name>` to create a new directory and `touch <file>` to create an empty file. To manage them, `cp <source> <destination>` will copy a file, `mv <source> <destination>` will move or rename it, and `rm <file>` will remove it. To remove an empty directory, use `rmdir <name>`; to remove a directory and all its contents, use the more powerful `rm -r <name>`.

Finally, to **install and manage software** (on Ubuntu/Debian), you use the Advanced Package Tool (APT). You should first run `sudo apt update` to refresh your system's list of available software. To install a tool, you run `sudo apt install <package_name>` (e.g., `sudo apt install git`). The `sudo` command temporarily elevates your privileges to "super user" (admin) to perform system-level tasks like installing software.

Docker Run commands

Here are the `docker run` commands to create and start containers from the official PostgreSQL and MongoDB images.

Important: For databases, you must use volumes (`-v`) to persist your data. If you don't, all your data will be **deleted** when the container is removed.

1. PostgreSQL

This command starts a PostgreSQL container, sets a required password, and maps the default port.

```
docker run -d \
  --name my-postgres-db \
  -e POSTGRES_PASSWORD=mysecretpassword \
  -v postgres_data:/var/lib/postgresql/data \
  -p 5432:5432 \
  postgres
```

Command Breakdown:

- `docker run -d`: Runs the container in detached (background) mode.
- `--name my-postgres-db`: Gives your container a memorable name.
- `-e POSTGRES_PASSWORD=mysecretpassword`: **(Required)** Sets the password for the postgres superuser. **Change mysecretpassword to a strong password.**
- `-v postgres_data:/var/lib/postgresql/data`: Creates a named volume called `postgres_data` and mounts it to the container's data directory. This is the correct way to persist your database data.
- `-p 5432:5432`: Maps port `5432` on your local machine to port `5432` inside the container.
- `postgres`: The name of the official image to use.

2. MongoDB

This command starts a MongoDB container, sets the required root username and password, and maps the default port.

```
docker run -d \
  --name my-mongo-db \
  -e MONGO_INITDB_ROOT_USERNAME=mongoadmin \
  -e MONGO_INITDB_ROOT_PASSWORD=mysecretpassword \
  -v mongo_data:/data/db \
  -p 27017:27017 \
  mongo
```

Command Breakdown:

- `docker run -d`: Runs the container in detached mode.

- `--name my-mongo-db`: Gives your container a memorable name.
- `-e MONGO_INITDB_ROOT_USERNAME=mongoadmin`: **(Required)** Sets the root username for the database.
- `-e MONGO_INITDB_ROOT_PASSWORD=mysecretpassword`: **(Required)** Sets the password for the root user. **Change mysecretpassword to a strong password.**
- `-v mongo_data:/data/db`: Creates a named volume called `mongo_data` and mounts it to the container's data directory to persist your data.
- `-p 27017:27017`: Maps port `27017` on your local machine to port `27017` inside the container.
- `mongo`: The name of the official image to use.

3. Custom Node image

Stage 1: Use an official Node.js image as the base.

Using 'alpine' for a smaller, more secure image.

FROM node:20-alpine AS base

Set the working directory inside the container

WORKDIR /usr/src/app

Copy package.json and package-lock.json first

This leverages Docker's layer caching.

COPY package*.json ./

---

Stage 2: Install production dependencies

FROM base AS production-deps

Install *only* production dependencies

RUN npm ci --omit=dev

---

Stage 3: Install all (dev) dependencies and build the app

FROM base AS build

Install all dependencies (including devDependencies for building)

RUN npm install

Copy the rest of your application's source code

COPY . .

Compile TypeScript to JavaScript
This assumes your 'build' script is defined in package.json
RUN npm run build

---
Stage 4: Create the final production image
FROM node:20-alpine AS production

WORKDIR /usr/src/app

Copy the production dependencies from the 'production-deps' stage
COPY --from=production-deps /usr/src/app/node_modules ./node_modules
Copy the compiled JavaScript output from the 'build' stage
Assumes the build output is in a 'dist' folder
COPY --from=build /usr/src/app/dist ./dist
Copy the production package.json
COPY package.json .

Expose the port your app runs on
EXPOSE 3000

The command to run your application
CMD ["node", "dist/index.js"]

Managing Images

An image is a read-only template used to create containers.

- `docker pull <image_name>`
Downloads an image from Docker Hub (or another registry).
Example: `docker pull postgres`
- `docker build -t <your_image_name> .`
Builds a new image from a Dockerfile in the current directory.
Example: `docker build -t my-node-app .`
- `docker images`
Lists all images currently stored on your machine.
- `docker rmi <image_id_or_name>`
Removes a specific image from your machine.

Managing Containers

A container is a runnable instance of an image.

- `docker run <image_name>`
Creates and starts a new container from an image.
Example: `docker run hello-world`
- `docker run -d --name <container_name> -p <host_port>:<container_port> <image_name>`
A more common run command:
 - `-d`: Detached mode (runs in the background).
 - `--name`: Assigns a name to your container.
 - `-p`: Maps a port from your machine to the container.Example: `docker run -d --name my-app -p 3000:3000 my-node-app`
- `docker ps`
Lists all running containers.
 - `docker ps -a`: Lists *all* containers, including stopped ones.
- `docker stop <container_id_or_name>`
Stops a running container gracefully.
- `docker start <container_id_or_name>`
Starts a stopped container.
- `docker rm <container_id_or_name>`
Removes a stopped container.
 - `docker rm -f <container_id>`: Forces removal of a *running* container.
- `docker logs <container_id_or_name>`
Shows the log output from a container.
 - `docker logs -f <container_id>`: Follows the log output in real-time.
- `docker exec -it <container_id_or_name> <command>`
Runs a command inside an already running container. This is often used to open a shell.
Example: `docker exec -it my-postgres-db bash`

System & Volumes

Commands for managing your Docker system and data.

- **docker volume create <volume_name>**
Creates a named volume to persist data.
Example: **docker volume create postgres_data**
- **docker volume ls**
Lists all volumes.
- **docker system prune**
A useful cleanup command. It removes all stopped containers, unused networks, and dangling images.
 - **docker system prune -a --volumes**: A *much more aggressive* command. Removes all stopped containers, all unused images (not just dangling ones), and all unused volumes. Use with caution, as this can delete database data if it's in an unused volume.