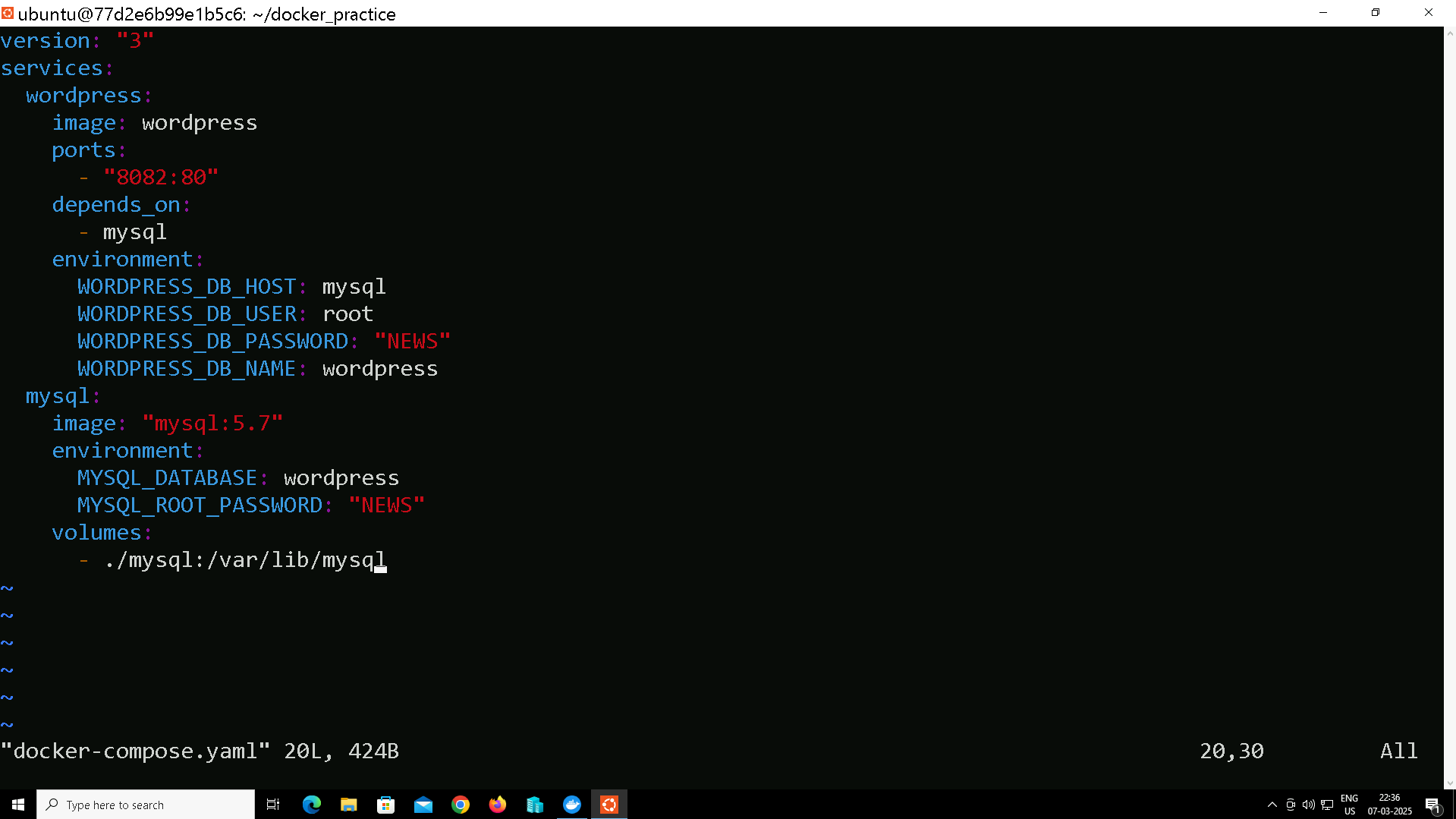
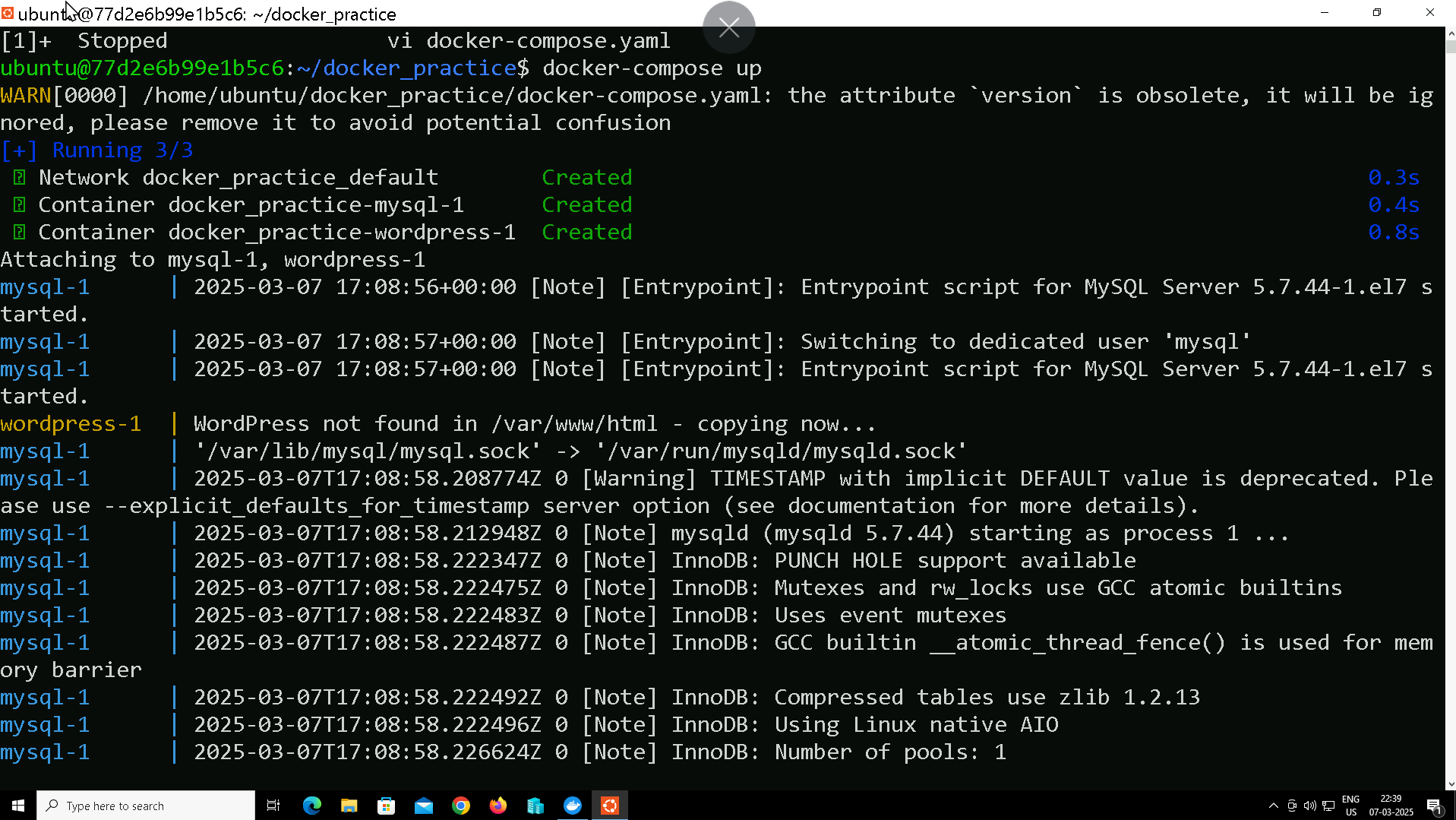
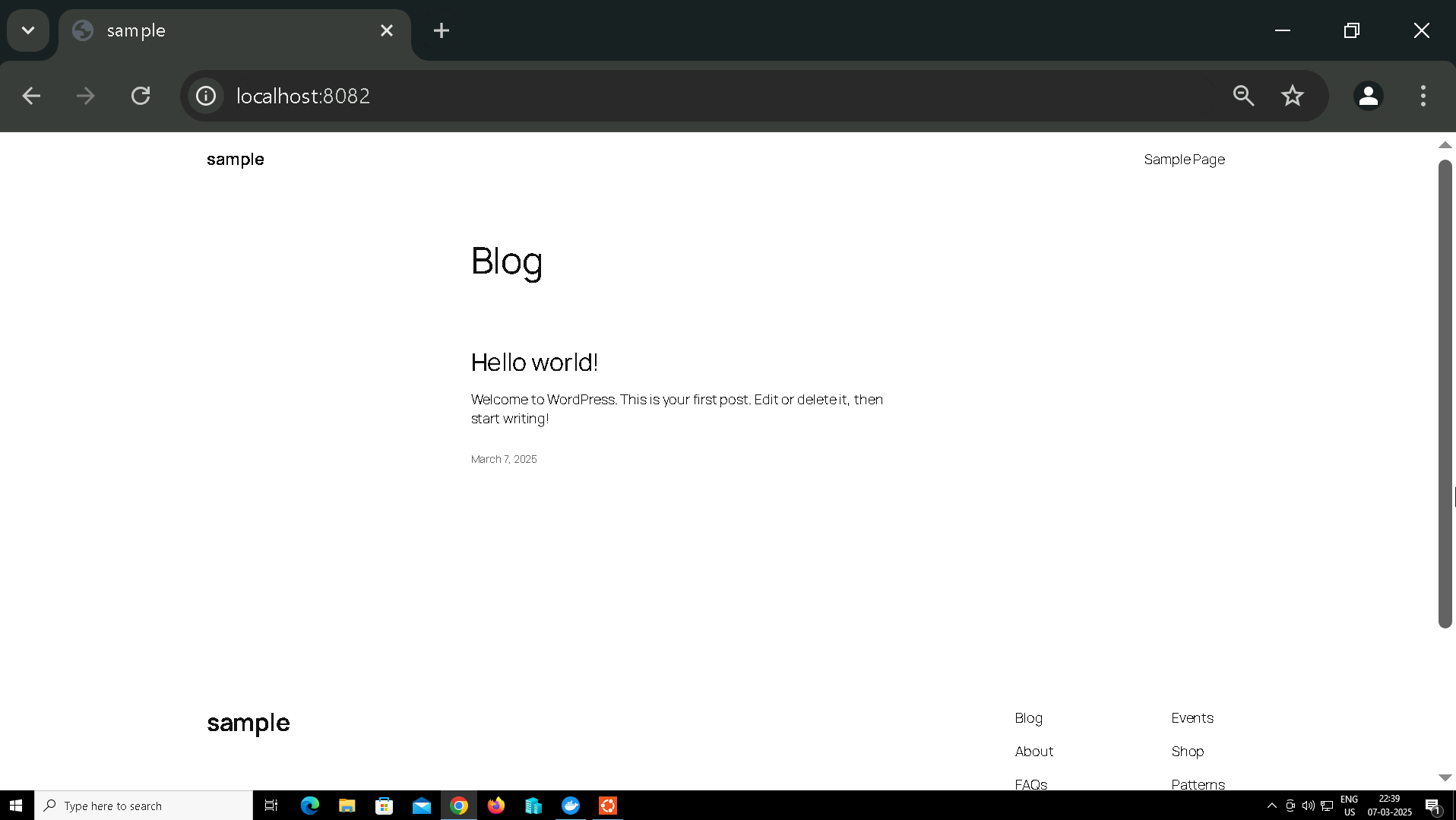
**Docker Compose**

**Task 1: Basic Docker Compose Setup**

* **Objective:** Create a basic docker-compose.yml file to run a web application and a database.
* **Instructions:**
  + Create a docker-compose.yml file that defines two services:
    - A web site using the wordpress image.
    - A database service using the mysql image.
  + Configure the database with environment variables (MYSQL\_USER and MYSQL\_PASSWORD).
  + Ensure the web service can connect to the database service (you can link the services or define a network).
  + Run docker-compose up to start the services.

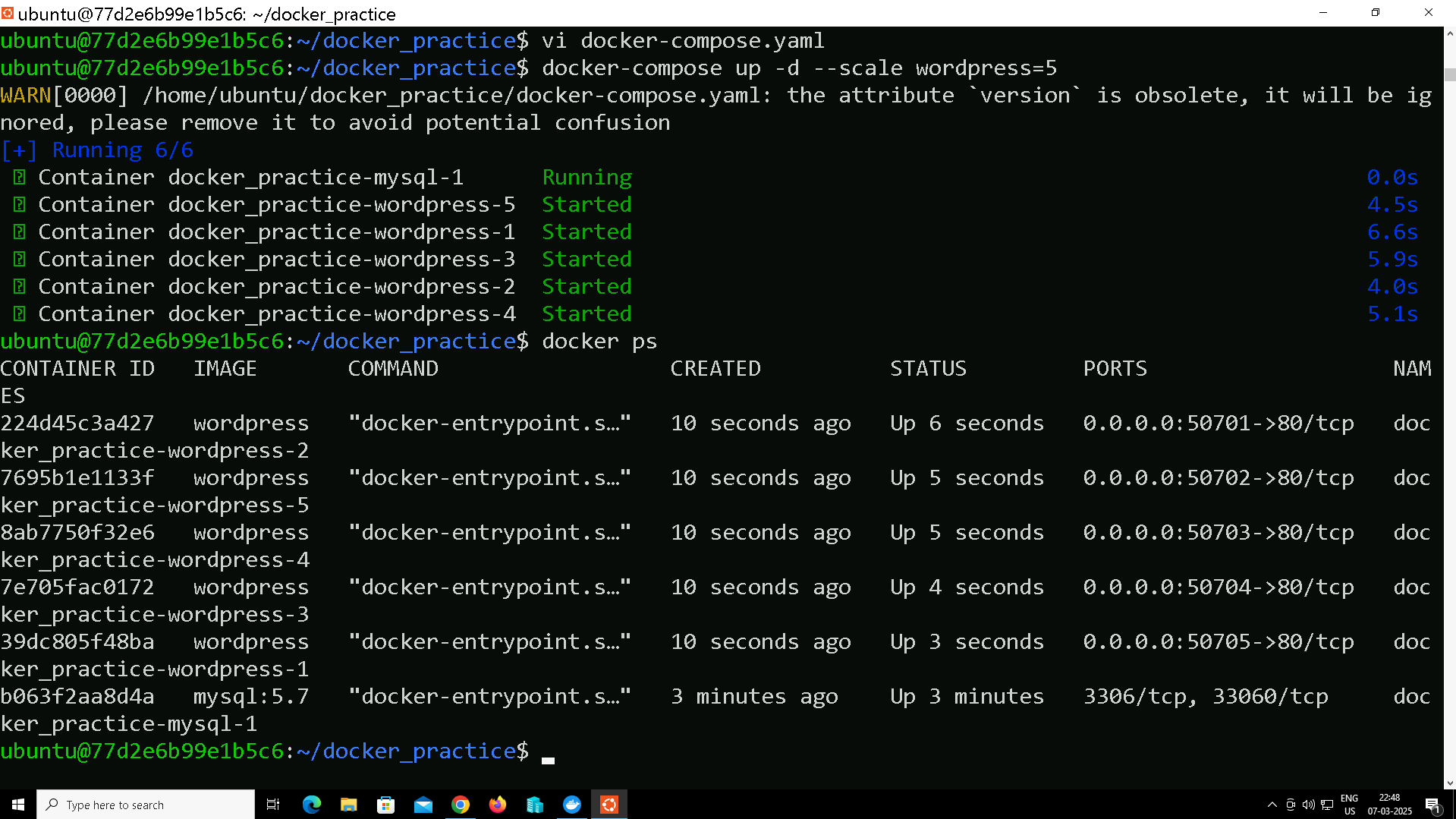
****

****

****

**Task 2: Scaling Services**

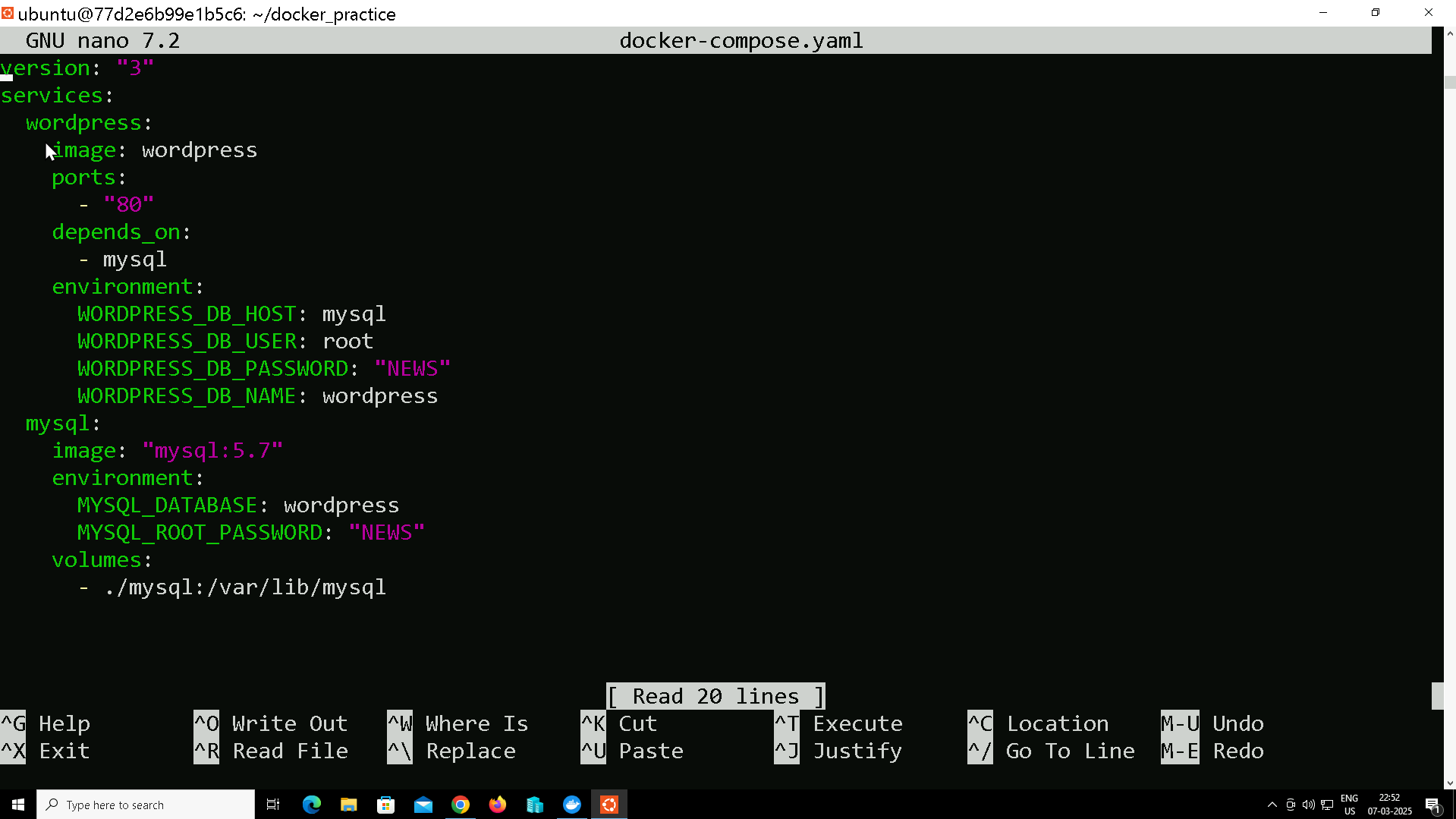
* **Objective**: Scale a service using Docker Compose.
* **Instructions**:
  1. Create a **docker-compose.yml** file for a web service using the **wordpress** image.
  2. Use the scale feature to run multiple replicas of the web service.
  3. Verify that multiple instances of the web service are running by using docker ps or checking the logs.



**Task 3: Docker Compose with Volumes**

* **Objective**: Use Docker volumes in your docker-compose.yml.
* **Instructions**:
  1. Create a docker-compose.yml file for a web application and a database service.
  2. Use a named volume to persist data for the database service (mysql).
  3. Ensure that the data persists even when you stop and restart the containers.
  4. Test the persistence by adding some data to the database and checking if it remains after restarting the containers.

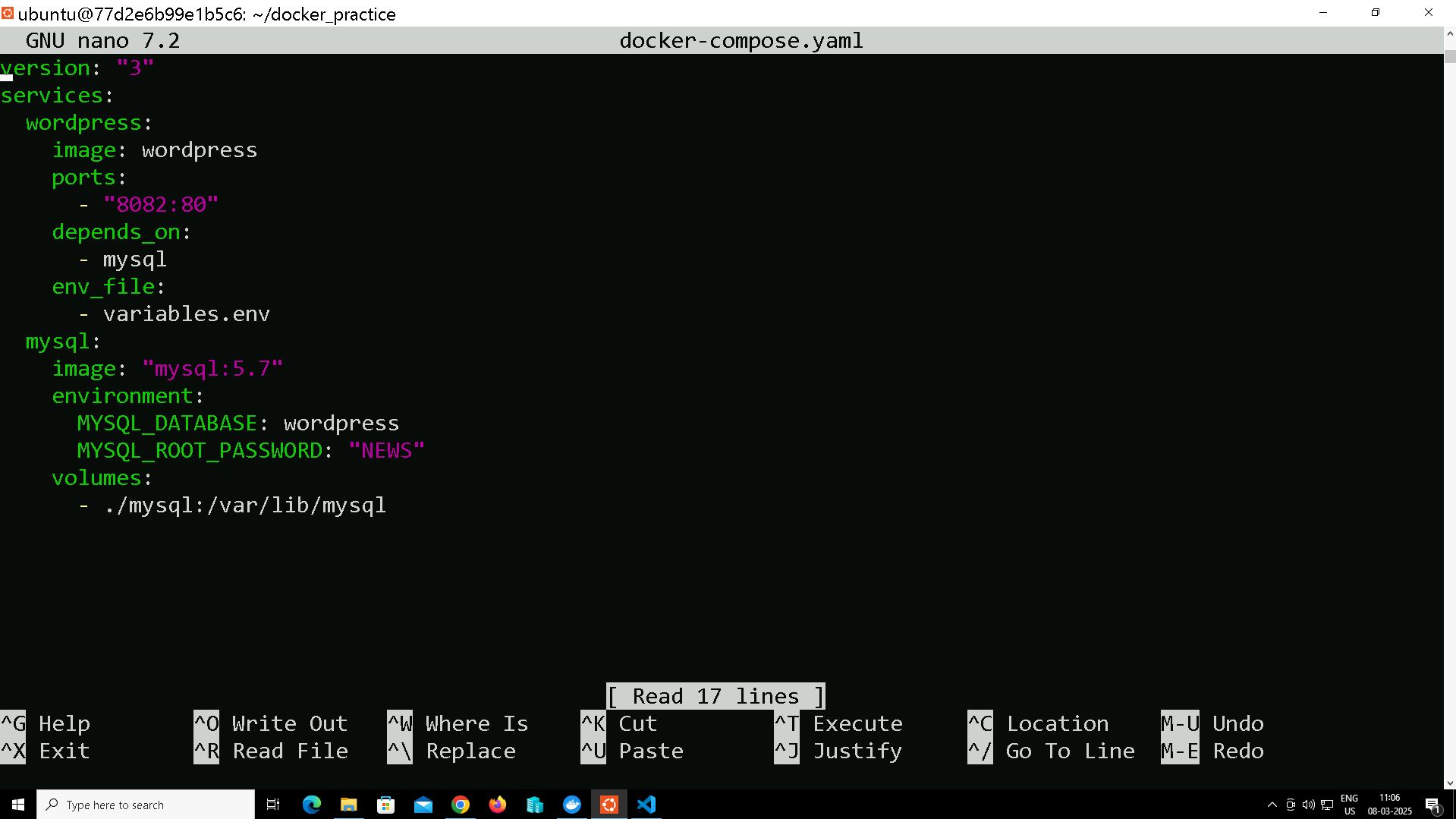
Soln:

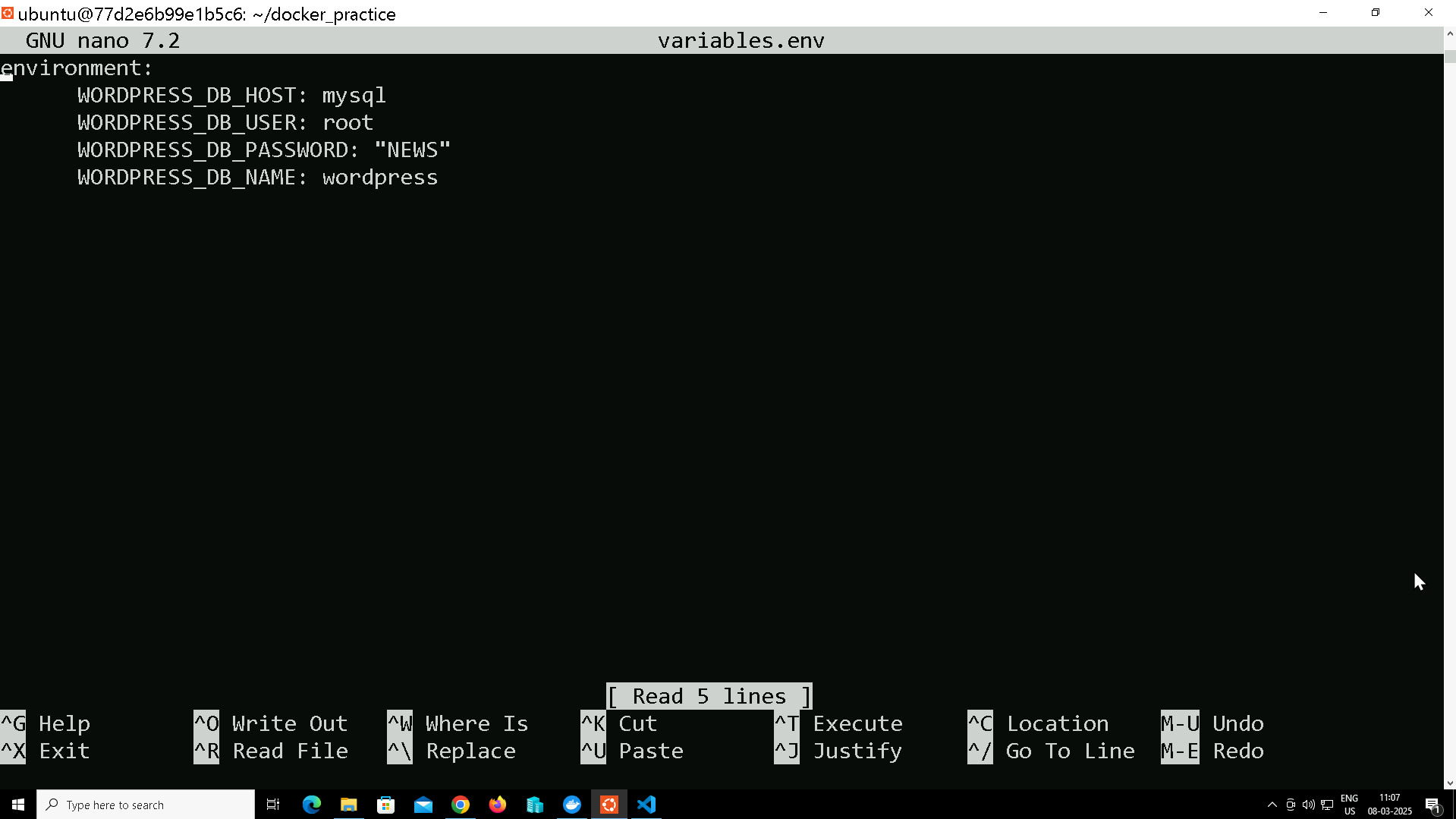


In this /var/lib/mysql have the data of the users even after restarting.

**Task 5: Environment Variables**

* **Objective: Configure environment variables for your Docker services.**

****

****