Build Guide: Two‑Tier Flask + MySQL (Compose Outcome) + Nexus Publishing

Student Project Instructions

August 25, 2025

# 0) Project Layout (assumed)

project-root/

├─ app/ # your Flask source (or another folder name)

│ └─ ... # must read MYSQL\_\* envs and listen on 0.0.0.0:5000

├─ Dockerfile # already exists (builds the Flask image)

├─ requirements.txt # already exists

├─ message.sql # already exists; seed data for MySQL on first run

└─ docker-compose.yml # final outcome file (provided)

Notes:

* The Flask app exposes '/' and (optionally) '/health' on port 5000.
* The app reads env vars: MYSQL\_HOST, MYSQL\_USER, MYSQL\_PASSWORD, MYSQL\_DB.
* message.sql runs on the first MySQL initialization (fresh data directory).

# 1) Bring up the two‑tier stack with Compose

**Build Guide: Two‑Tier Flask + MySQL (Outcome Compose) + Nexus Publishing**

This guide turns your existing source (Flask app), **Dockerfile**, **requirements.txt**, and **message.sql** into the final two‑tier stack defined by your docker-compose.yml. It then adds optional steps to publish and consume the app image via a local **Sonatype Nexus** registry.

**0) Project Layout (assumed)**

project-root/

├─ app/ # your Flask source (or another folder name)

│ └─ ... # must read MYSQL\_\* envs and listen on 0.0.0.0:5000

├─ Dockerfile # already exists (builds the Flask image)

├─ requirements.txt # already exists

├─ message.sql # already exists; seed data for MySQL on first run

└─ docker-compose.yml # final outcome file (provided)

**Expectations:**

* The Flask app exposes / and (optionally) /health on port **5000**.
* The app reads env vars: MYSQL\_HOST, MYSQL\_USER, MYSQL\_PASSWORD, MYSQL\_DB.
* message.sql creates the schema/seed content; MySQL runs files in /docker-entrypoint-initdb.d/ on first initialization.

**1) Build the Flask image (local)**

From project-root/:

# Enable BuildKit for speed (optional)

$env:COMPOSE\_DOCKER\_CLI\_BUILD=1; $env:DOCKER\_BUILDKIT=1 # PowerShell

# or: export COMPOSE\_DOCKER\_CLI\_BUILD=1; export DOCKER\_BUILDKIT=1

# Build a local image tag for dev

docker build -t twotier/flask-app:1.0 .

# Quick local test (optional)

docker run --rm -p 5000:5000 \

-e MYSQL\_HOST=localhost -e MYSQL\_USER=admin -e MYSQL\_PASSWORD=admin -e MYSQL\_DB=devops \

twotier/flask-app:1.0

Visit [**http://localhost:5000/**](http://localhost:5000/) (stop container with Ctrl+C).

**2) Bring up the two‑tier stack with Compose**

Your outcome docker-compose.yml (describes two services: **mysql** and **flask-app**.

Run it:

docker compose up -d

**What Compose will do**

* **mysql**
  + Image: mysql:5.7
  + Binds: ./mysql-data:/var/lib/mysql (persistent data)
  + Seeds DB on first run by mounting ./message.sql to /docker-entrypoint-initdb.d/message.sql.
  + Env (example):
    - MYSQL\_ROOT\_PASSWORD=root
    - MYSQL\_DATABASE=devops
    - MYSQL\_USER=admin
    - MYSQL\_PASSWORD=admin
* **flask-app**
  + Exposes **5000:5000**
  + Depends on **mysql**
  + Env to reach MySQL, e.g.: MYSQL\_HOST=mysql, MYSQL\_USER=admin, MYSQL\_PASSWORD=admin, MYSQL\_DB=devops.
  + healthcheck hitting <http://localhost:5000/health>.
    - test: ["CMD-SHELL", "curl -f http://localhost:5000/health || exit 1"]
    - interval: 10s
    - timeout: 5s
    - retries: 5
    - start\_period: 30s

Check status & logs:

docker compose ps

docker compose logs -f --tail=100 flask-app

Open the app: [**http://localhost:5000/**](http://localhost:5000/)  
(Optional) Health: [**http://localhost:5000/health**](http://localhost:5000/health)

Stop the stack:

docker compose down

**3) Publish & pull via Nexus**

These steps let you publish your Flask image to a **local Nexus** Docker registry and then run the stack using Nexus endpoints.