**Activity 13: Virtual Machine (VM) and Container**

**สมาชิก**

|  |  |
| --- | --- |
| **ชื่อ-นามสกุล** | **เลขประจำตัวนิสิต** |
| นายสิปปภาส ชวานนท์ | 6630333721 |
| นายเนติภัทร โพธิพันธ์ | 6631331621 |
| นายวรลภย์ ศรีชัยนนท์ | 6632200221 |

**ส่วนที่ 1: เขียนโปรแกรม**

ในการเตรียมโปรแกรมก่อนที่จะบรรจุลง Docker Container จะต้องเขียนทั้งหมด 2 ไฟล์ คือ

1. **main.c** คือไฟล์โปรแกรมสั่งการว่าจะต้องทำอะไร
2. **Dockerfile** คือไฟล์โปรแกรมสั่งการว่าจะบรรจุลง Docker container อย่างไร

**เขียนโปรแกรม main.c**

1. เก็บ Environment variable ที่ชื่อว่า **secret\_user** ตอนผู้ใช้งานสั่ง run
2. แสดงผล **Hello, $secret\_user** บนหน้าจอผู้ใช้งาน ตัวอย่างเช่น

เมื่อใช้คำสั่ง **docker run -e secret\_user=Krerk my\_repo.localhost:50000/demo**

ให้แสดงผลว่า **Hello, Krerk** บนหน้าจอผู้ใช้งาน

1. หากไม่มีการระบุ **secret\_user** ตอนสั่ง run ให้แสดงผลว่า **Hello, Tendou Arisu** แทน

**#include <stdio.h>**

**#include <stdlib.h>**

**int main() {**

**// Get Environment variables**

**char\* user = getenv("secret\_user");**

**// Output**

**if (user) { printf("Hello, %s\n", user); }**

**else { printf("Hello, Tendou Arisu\n"); }**

**return 0;**

**}**

**เขียนโปรแกรม Dockerfile**

1. ตั้งค่า Base image ให้เป็นของ **Linux Alpine** (หากตั้งให้เป็น GCC โดยตรงอาจโดนฟ้องเรื่อง vulnerability ได้)
2. จากนั้นให้ install ตัวที่จำเป็นต้องใช้งานคือ **Standard C Library** (libc-dev) และ **GCC compiler** (gcc) จาก apk ซึ่งเป็น package manager ของ Alpine
3. ตั้งค่า working directory บน container ไปยัง **/app**
4. นำเข้าไฟล์ **main.c** ไปยัง container
5. สั่งให้รันคำสั่ง **gcc -o app main.c** เพื่อให้ GCC compile ไฟล์ **main.c** ออกมาเป็น binary file ที่ชื่อว่า **app.o**
6. รันไฟล์ **app.o** ด้วยคำสั่ง **./app** เป็นอันเสร็จสิ้นกระบวนการ

**FROM alpine:latest**

**RUN apk add --no-cache gcc libc-dev**

**WORKDIR /app**

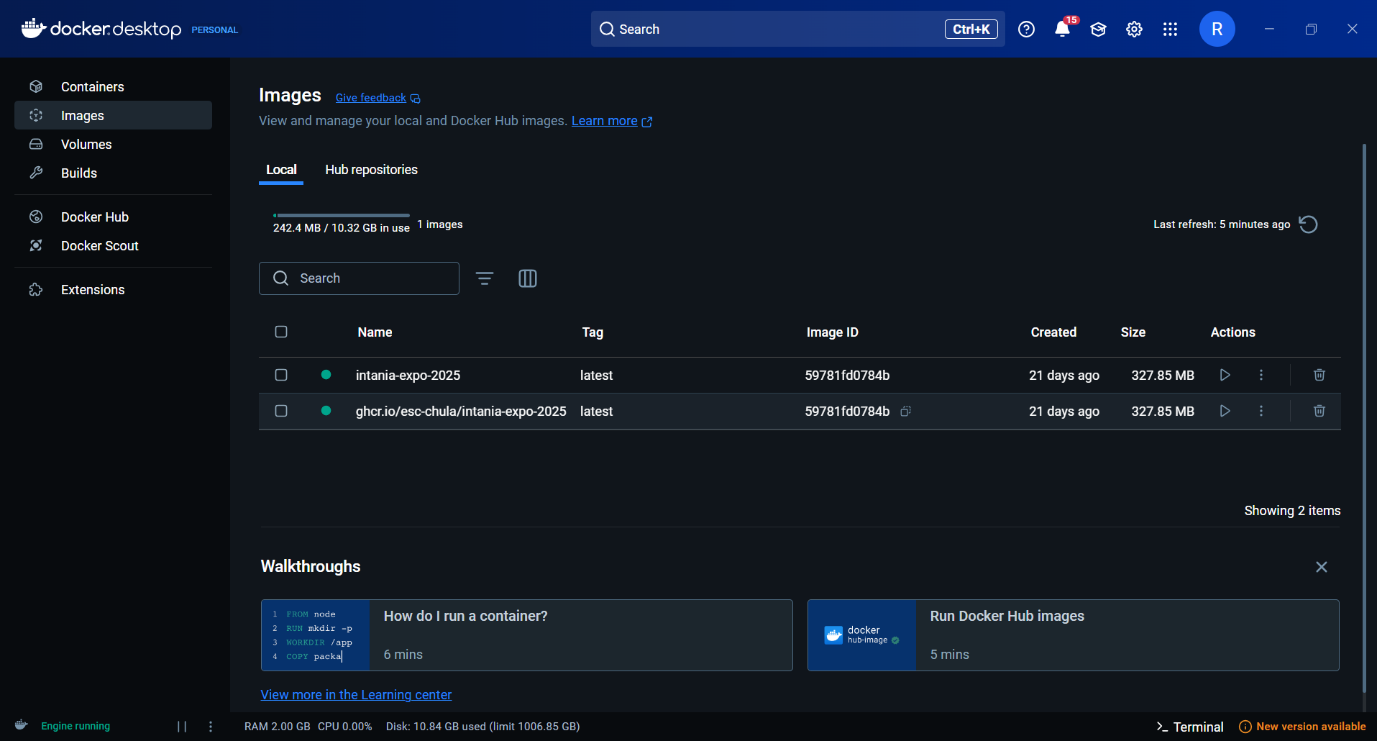
**COPY main.c .**

**RUN gcc -o app main.c**

**CMD ["./app"]**

**ส่วนที่ 2: สร้าง Docker Container**

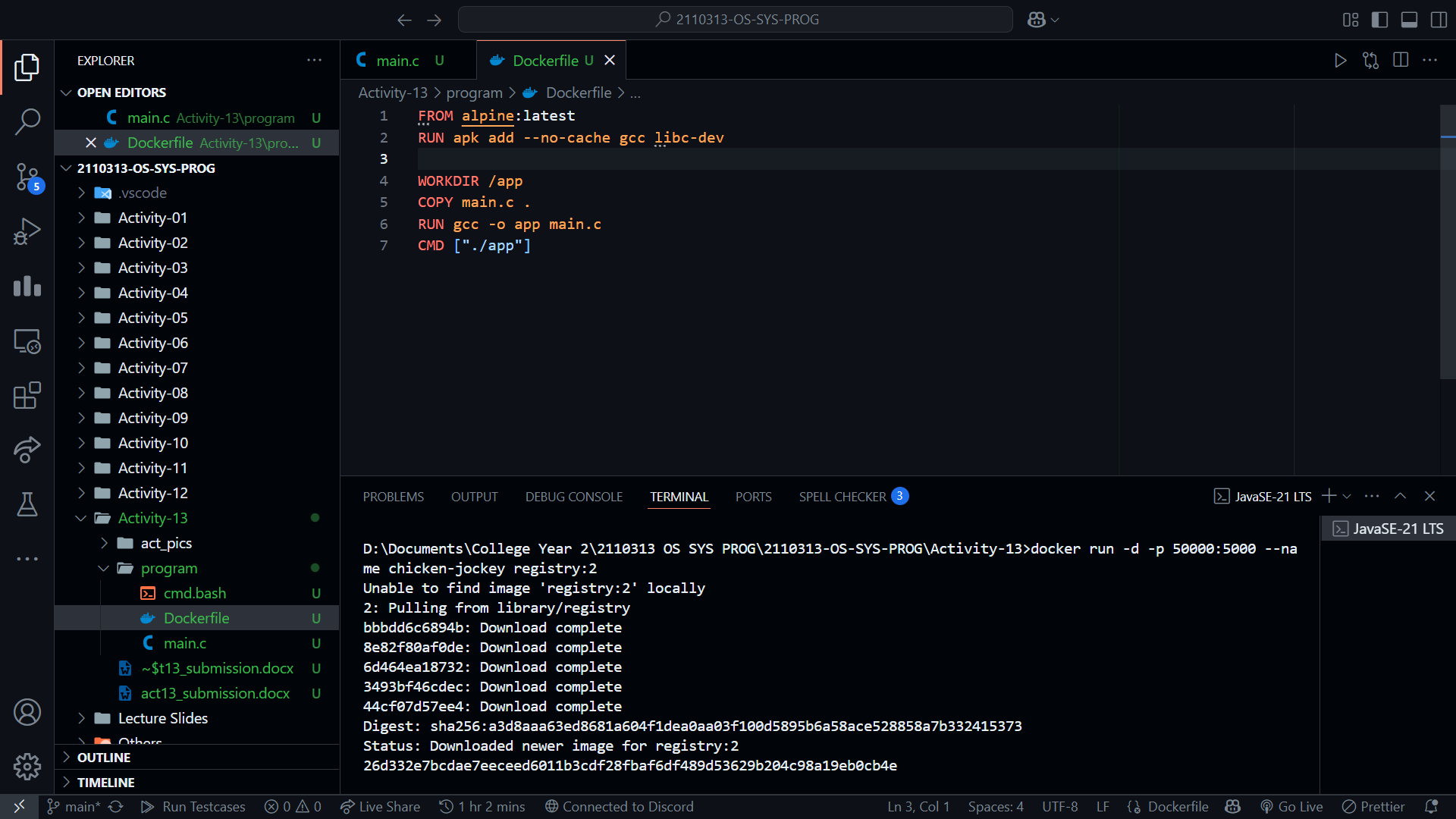
เปิดโปรแกรม **Docker Desktop** (Download <https://www.docker.com/products/docker-desktop/>)



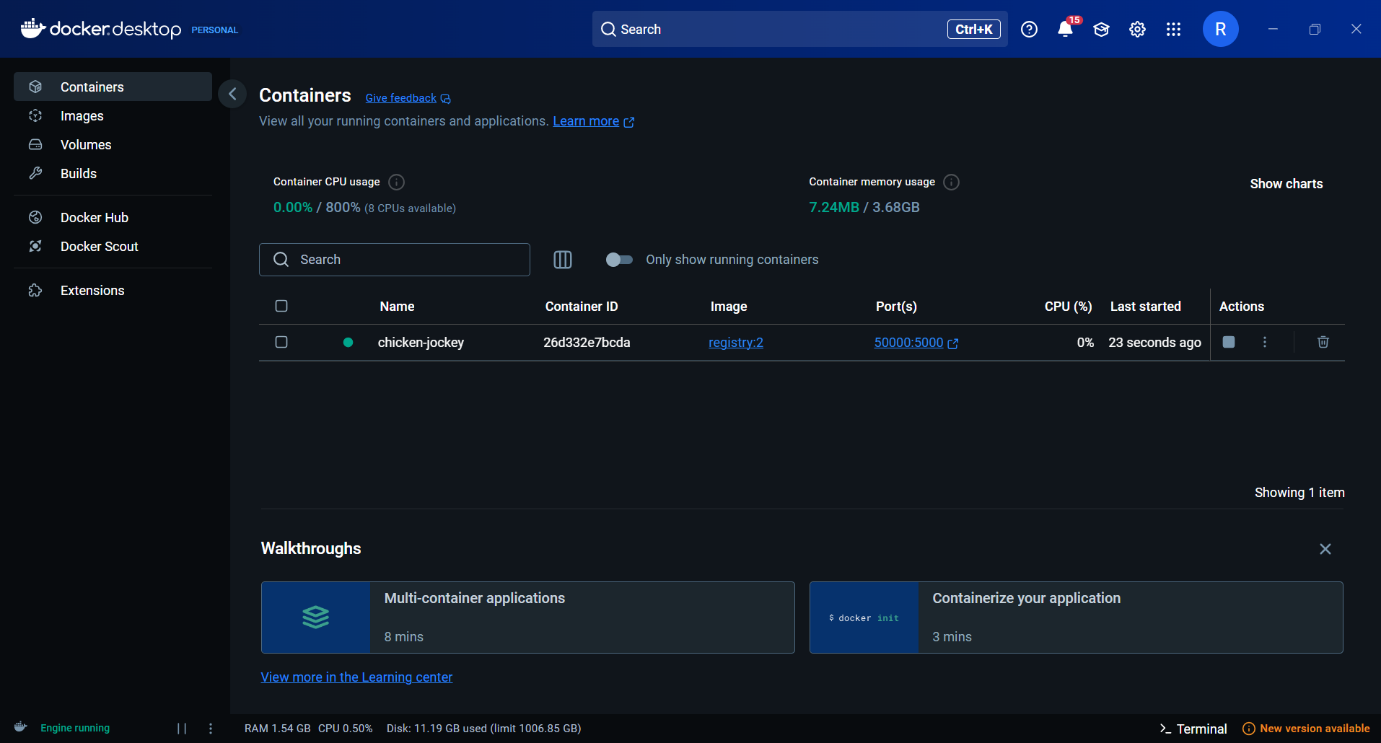
สร้าง **Local Docker Registry** เพื่อเป็น server สำหรับบรรจุ Docker image ของโปรแกรม โดยใช้คำสั่ง

**docker run -d -p 50000:5000 --name chicken-jockey registry:2**

รันคำสั่งนี้บน Terminal



เมื่อเปิด Docker Desktop บนหน้า Container จะพบว่ามี **chicken-jockey** ปรากฏขึ้นมา

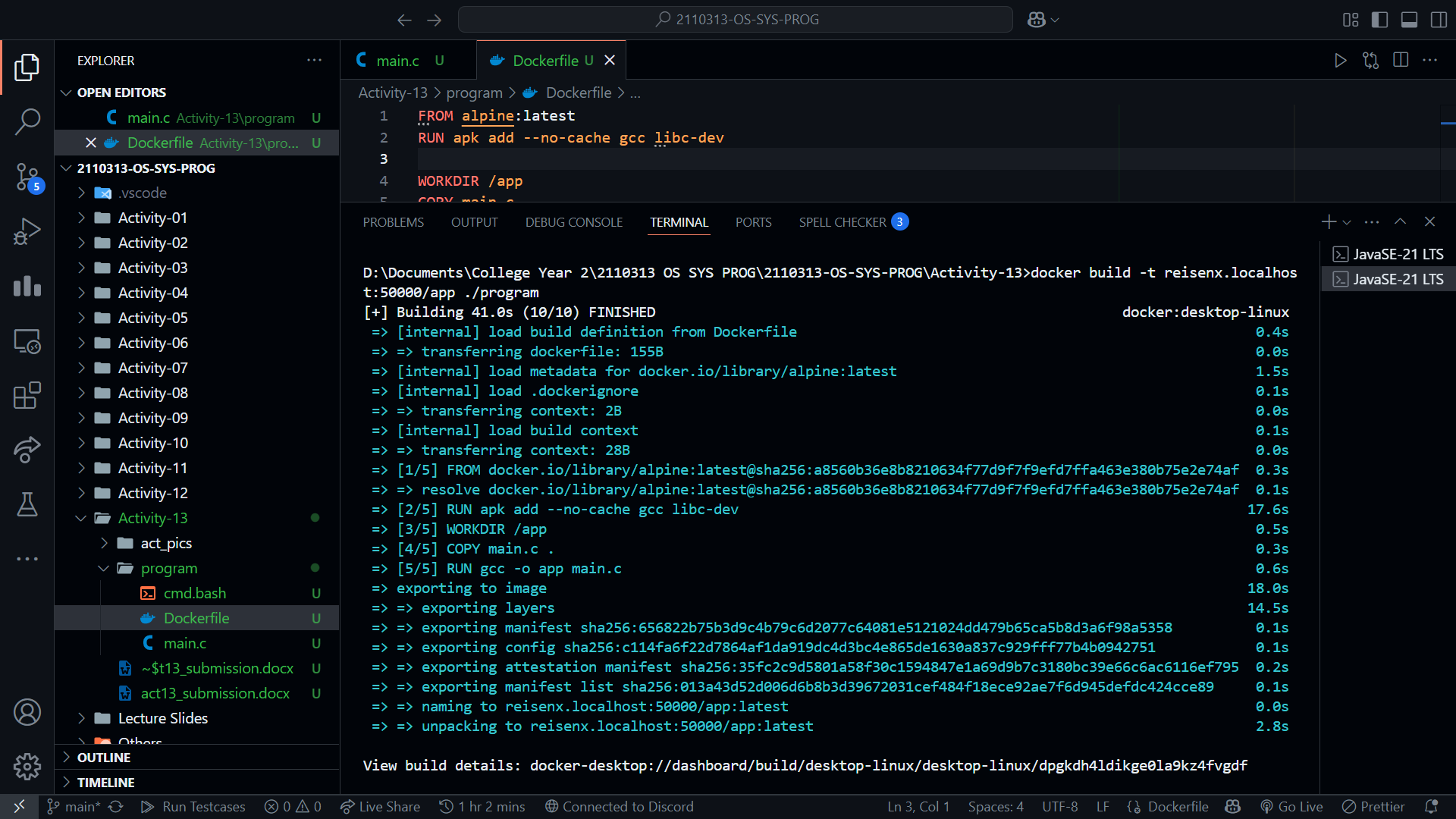


จากนั้นให้สร้าง **Docker Image** โดยใช้คำสั่งดังกล่าว

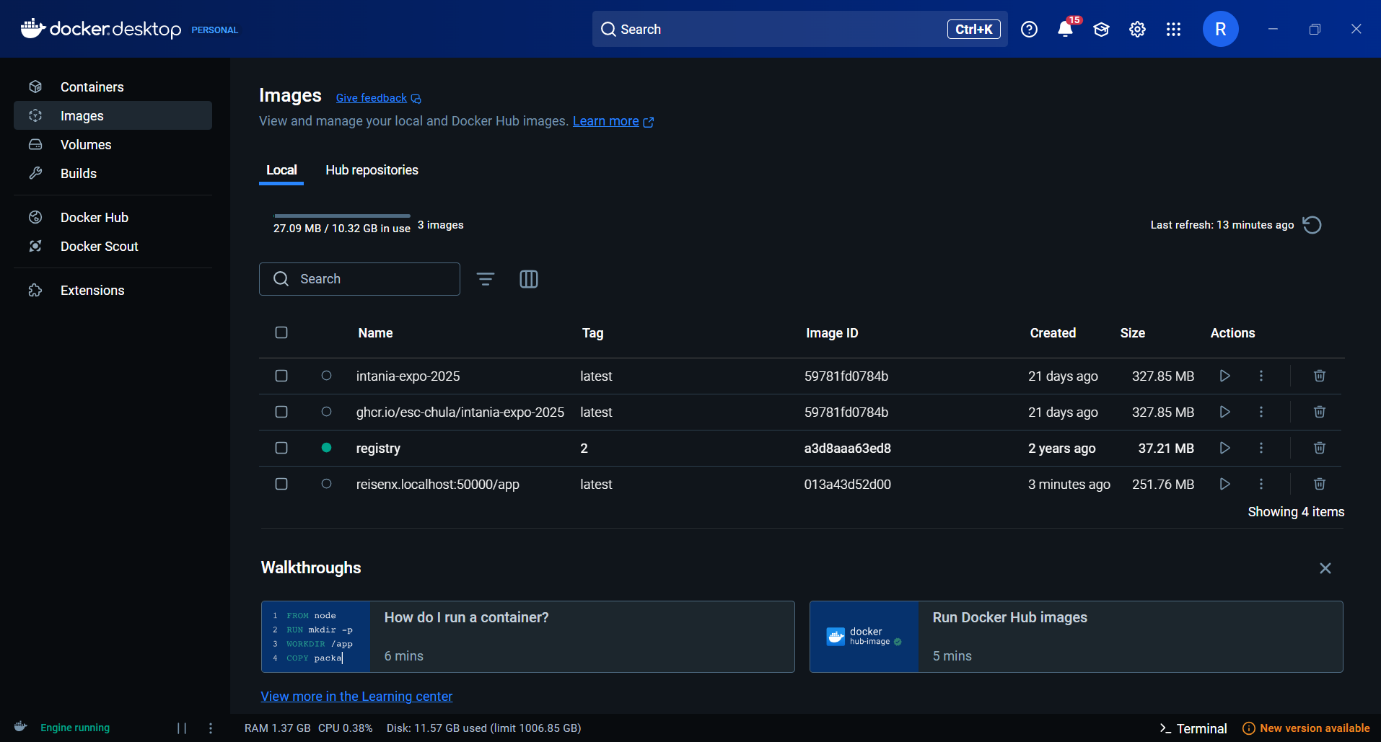
(ในกรณีนี้ไฟล์ main.c และ Dockerfile อยู่ใน folder ที่ชื่อว่า program)

**docker build -t reisenx.localhost:50000/app ./program**

รันคำสั่งนี้บน Terminal



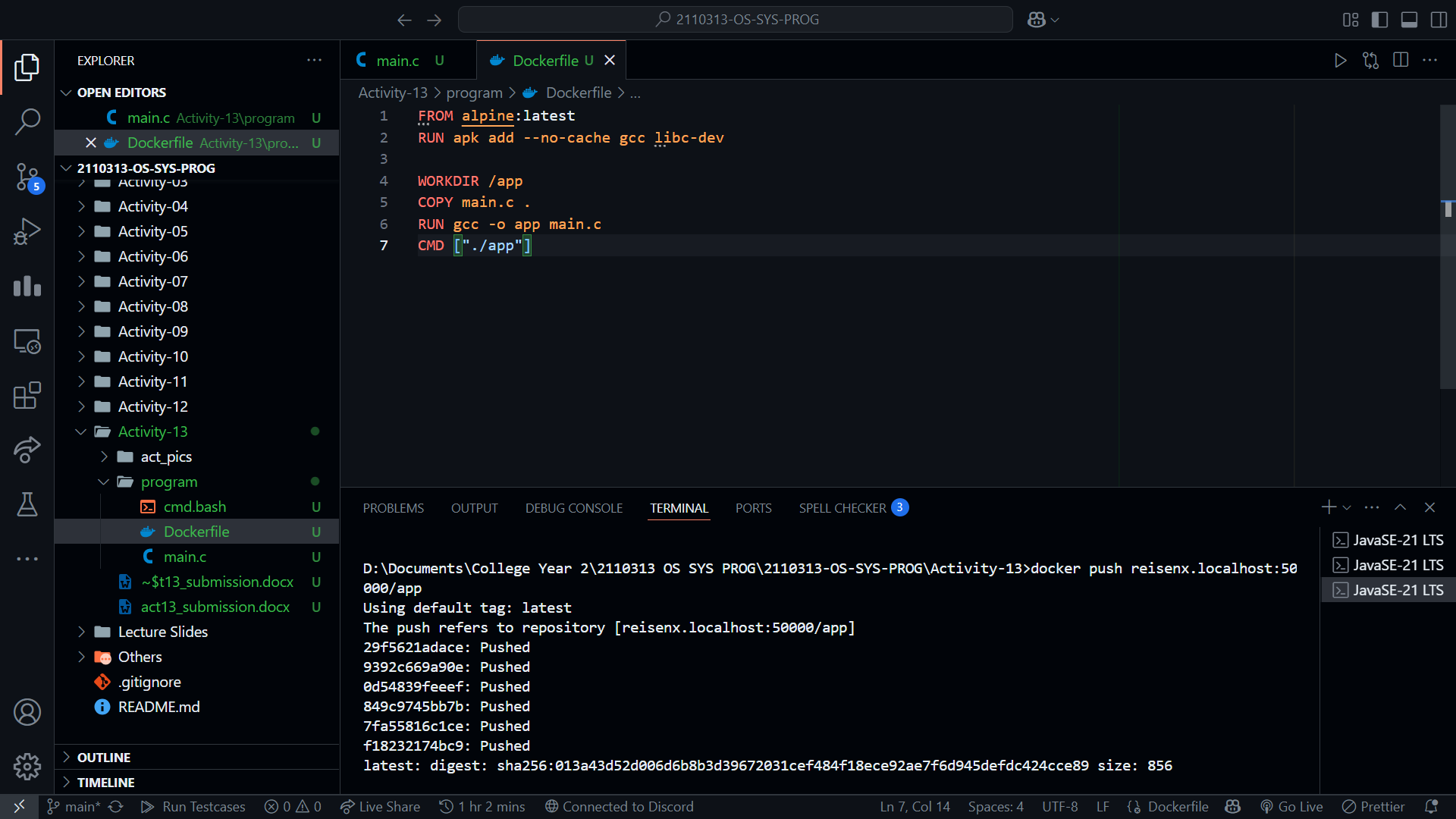
เมื่อเปิด Docker Desktop บนหน้า Image จะพบว่ามี **reisenx.localhost:50000/app** ปรากฏขึ้นมา



จากนั้นให้ **Push Docker image** ไปยัง Local Docker Registry ที่สร้างไว้ก่อนหน้า

**docker push reisenx.localhost:50000/app**

รันคำสั่งบน Terminal

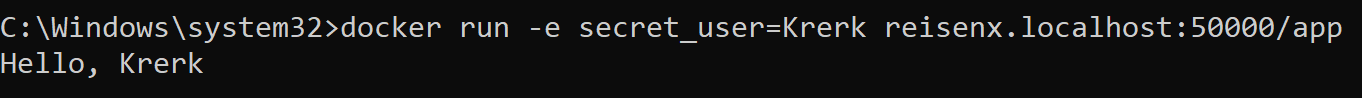


**ส่วนที่ 3: ทดลอง Run Docker Image**

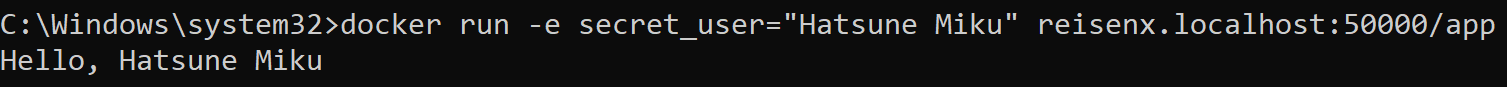
เปิด Terminal และรันคำสั่งดังกล่าว

**docker run -e secret\_user=Krerk reisenx.localhost:50000/app**

จะพบว่าหากกำหนดให้ **secret\_user = Krerk** จะแสดงผล **Hello, Krerk** ขึ้นมาบนหน้าจอ



และถ้าหากเปลี่ยนเป็น **secret\_user** เป็นชื่ออื่น ก็จะแสดงผลตามนั้น ดังภาพ



เมื่อเปิด Docker Desktop บนหน้า Image จะพบว่า **reisenx.localhost:50000/app** ขึ้นสถานะ IN USE สังเกตได้จากการที่มีจุดสีเขียวปรากฏขึ้นมาข้างหน้าชื่อ

