

Activity 13: Virtual Machine (VM) and Container

สมาชิก

ชื่อ-นามสกุล	เลขประจำตัวนิสิต
นายสีปภาส 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
3. 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

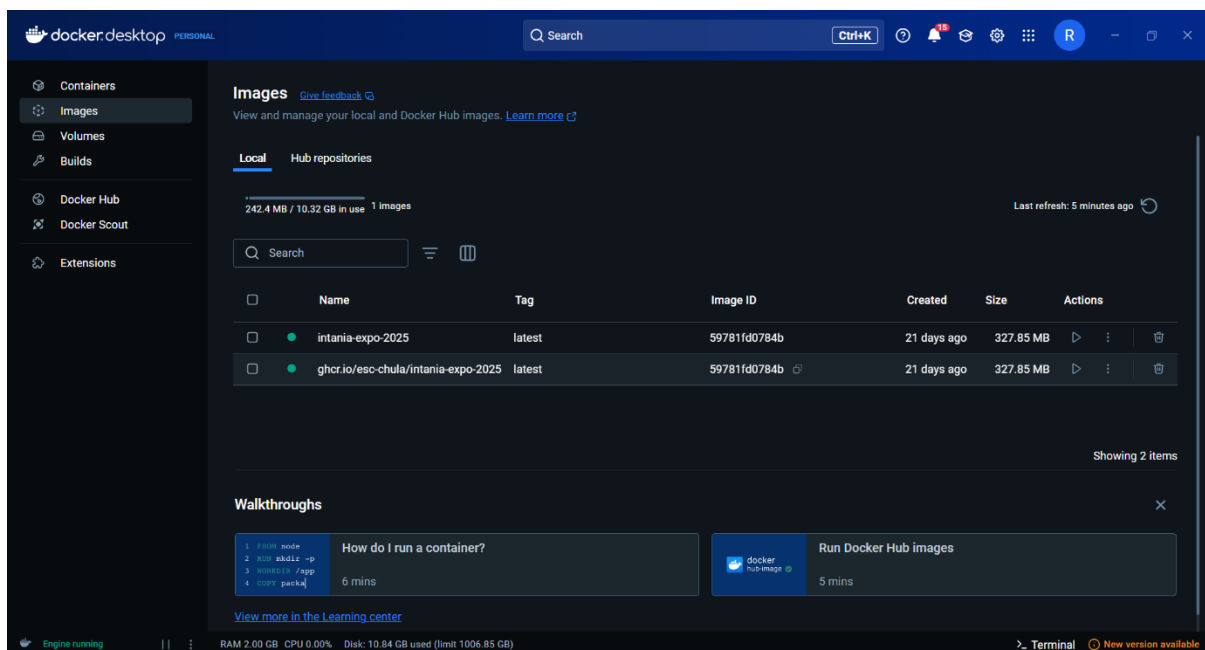
1. ตั้งค่า Base image ให้เป็นของ **Linux Alpine** (หากตั้งให้เป็น GCC โดยตรงอาจโดนฟ้องเรื่อง vulnerability ได้)
2. จากนั้นให้ install ตัวที่จำเป็นต้องใช้งานคือ **Standard C Library (libc-dev)** และ **GCC compiler (gcc)** จาก apk ซึ่งเป็น package manager ของ Alpine
3. ตั้งค่า working directory บน container 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

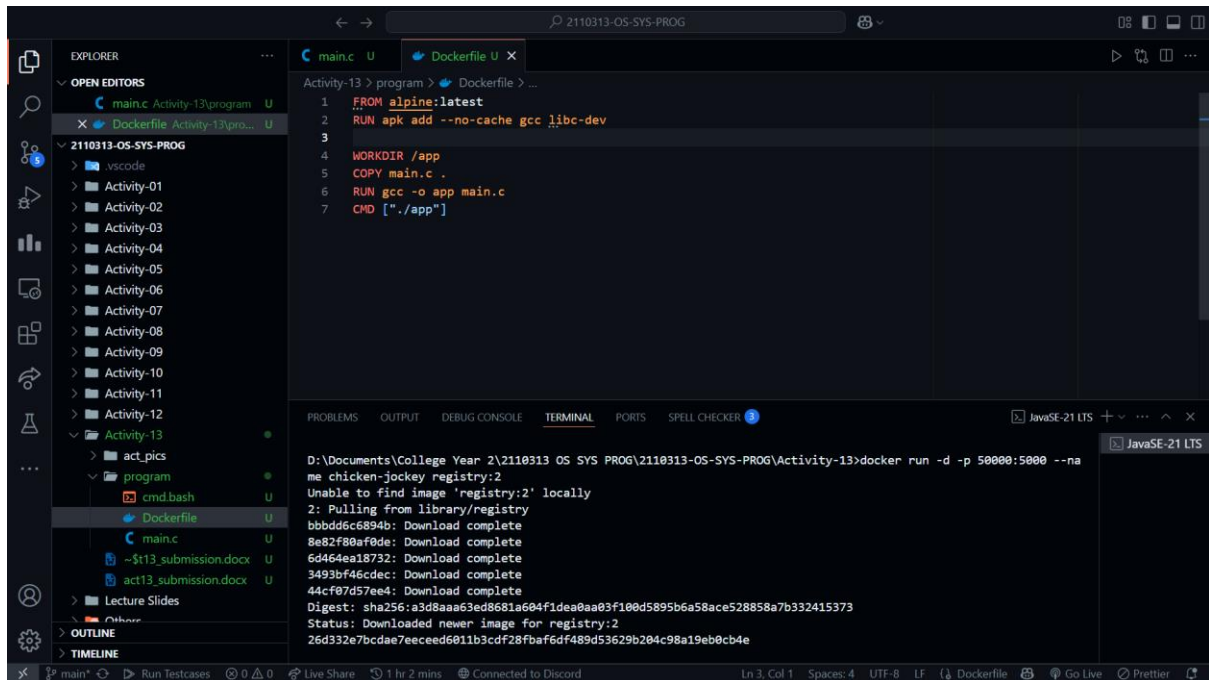
เปิดโปรแกรม Docker Desktop (Download <https://www.docker.com/products/docker-desktop/>)



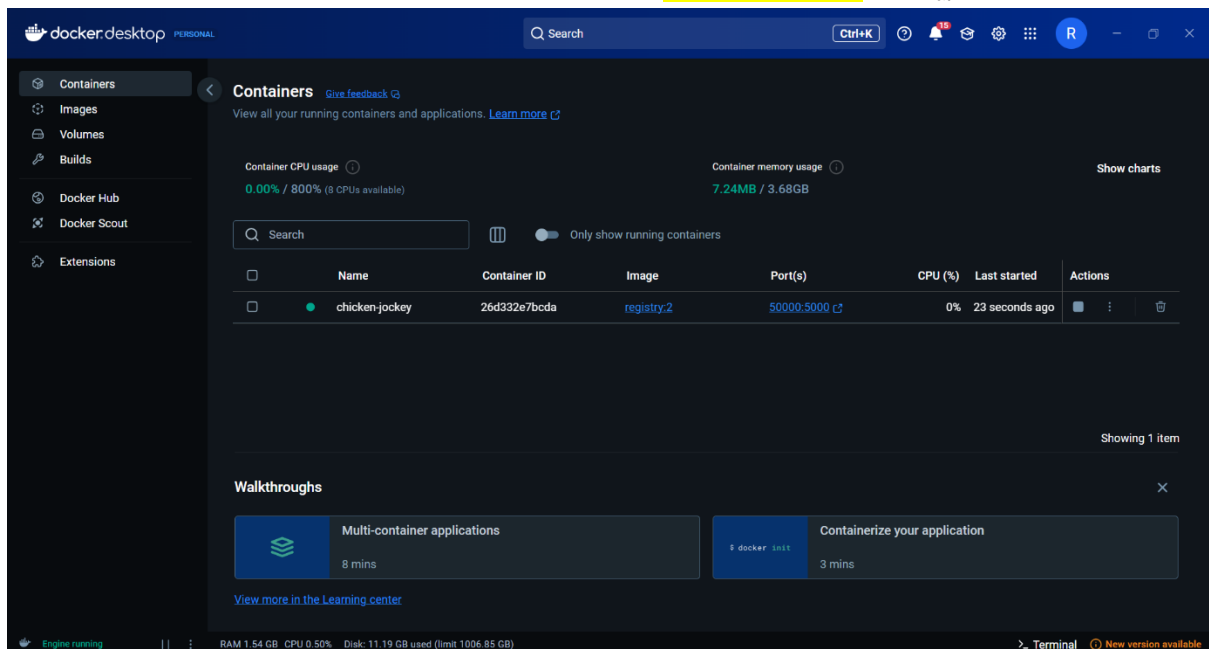
สร้าง Local Docker Registry เพื่อเป็น server สำหรับบรรจุ Docker image ของโปรแกรม โดยใช้คำสั่ง

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

รันคำสั่งนี้บน Terminal



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

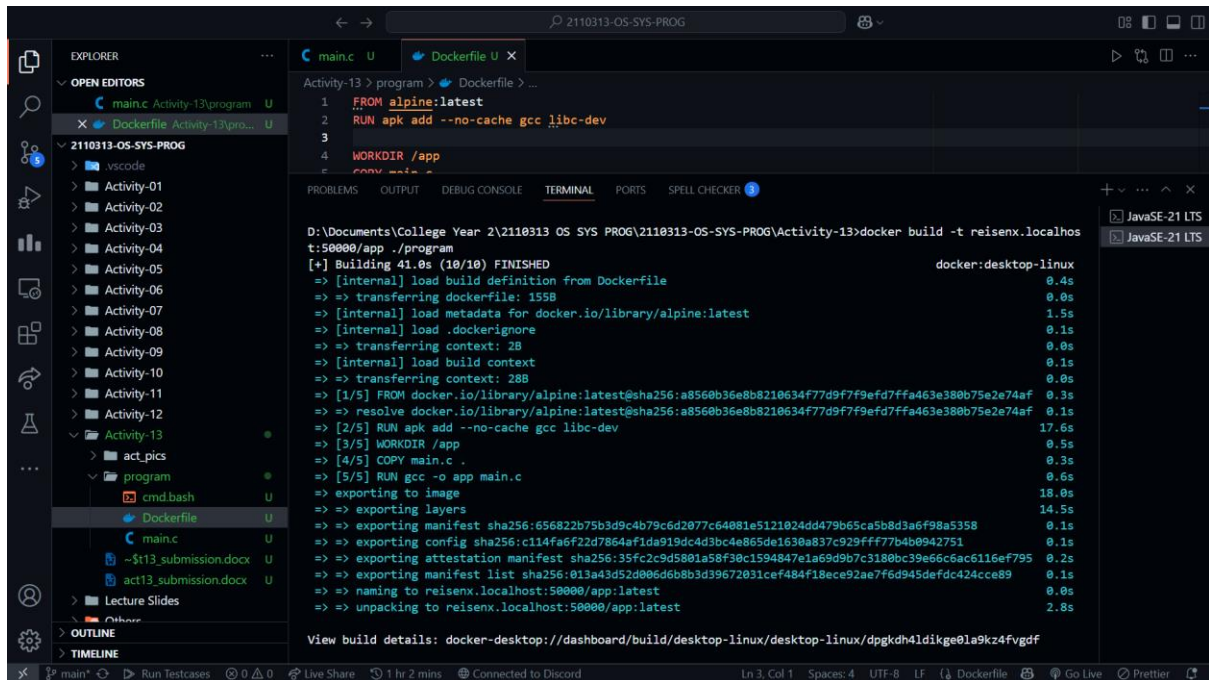


จากนั้นให้สร้าง Docker Image โดยใช้คำสั่งดังกล่าว

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

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

รันคำสั่งนี้บน Terminal

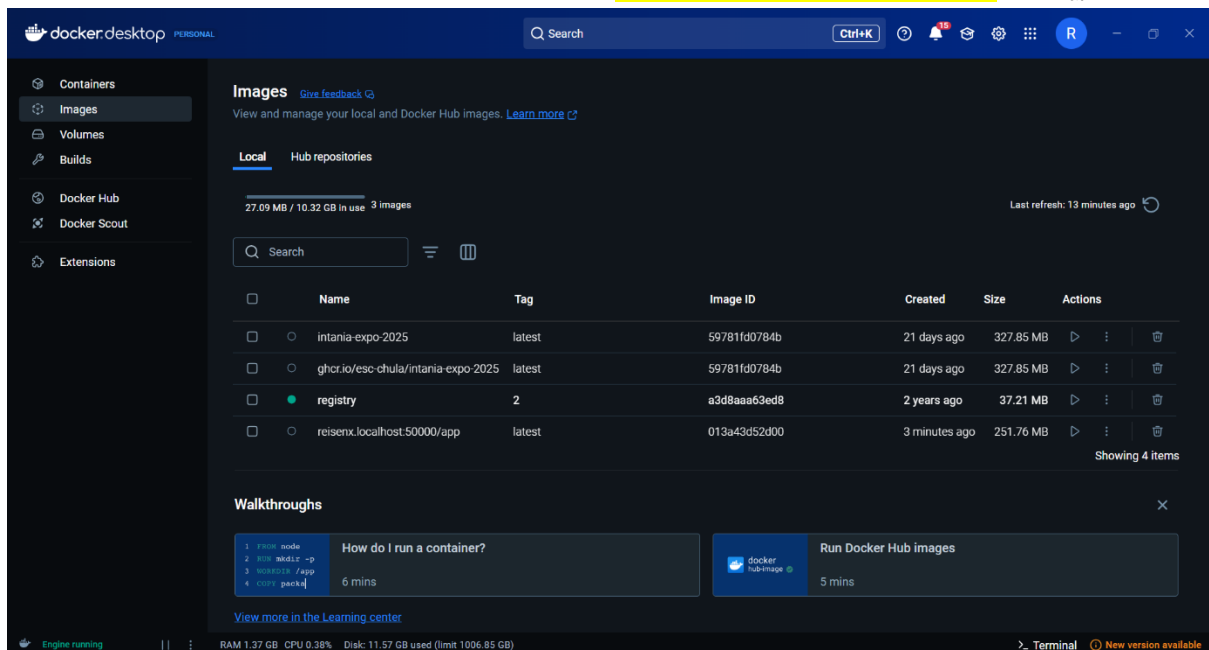


The screenshot shows the VS Code interface with the Dockerfile open in the editor. The Dockerfile contains the following content:

```
1 FROM alpine:latest
2 RUN apk add --no-cache gcc libc-dev
3
4 WORKDIR /app
5 COPY main.c .
6 RUN gcc -o app main.c
```

The terminal output shows the execution of the command `docker build -t reisenx.localhost:50000/app ./program`. The output indicates that the build was successful, with the image exported to the local host.

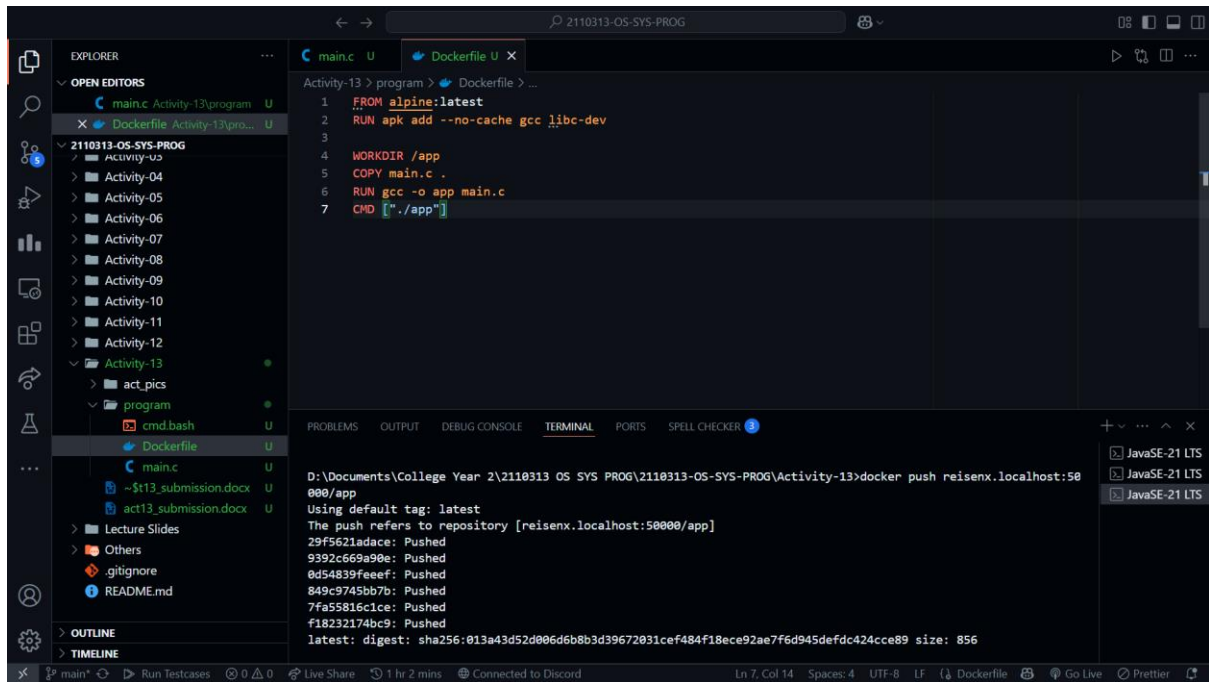
เมื่อเปิด Docker Desktop บนหน้า Image จะพบว่ามี **reisenx.localhost:50000/app** ปรากฏขึ้นมา



จากนั้นให้ Push Docker image ไปยัง Local Docker Registry ที่สร้างไว้ก่อนหน้านี้

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

รันคำสั่งบน Terminal



The screenshot shows a Visual Studio Code editor with a Dockerfile open in the 'Activity-13 > program' directory. The Dockerfile contains the following instructions:

```
1 FROM alpine:latest
2 RUN apk add --no-cache gcc libc-dev
3
4 WORKDIR /app
5 COPY main.c .
6 RUN gcc -o app main.c
7 CMD ["/app"]
```

The terminal at the bottom shows the execution of the command `docker push reisenx.localhost:50000/app`. The output indicates that the image was pushed successfully to the local registry. The digest for the 'latest' tag is `sha256:013a43d52d006d6b8b3d39672031cef484f18ece92ae7f6d945defdc424cce89` and the size is 856 bytes.

```
D:\Documents\College Year 2\2110313 OS SYS PROG\2110313-OS-SYS-PROG\Activity-13>docker push reisenx.localhost:50000/app
Using default tag: latest
The push refers to repository [reisenx.localhost:50000/app]
29f5621adace: Pushed
9392c669a90e: Pushed
0d54839feef: Pushed
849c9745bb7b: Pushed
7fa55816c1ce: Pushed
f18232174bc9: Pushed
latest: digest: sha256:013a43d52d006d6b8b3d39672031cef484f18ece92ae7f6d945defdc424cce89 size: 856
```

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

```
C:\Windows\system32>docker run -e secret_user=KrerK reisenx.localhost:50000/app
Hello, Krerk
```

และถ้าหากเปลี่ยนเป็น **secret_user** เป็นชื่ออื่น ก็แสดงผลตามนั้น ดังภาพ

```
C:\Windows\system32>docker run -e secret_user="Hatsune Miku" reisenx.localhost:50000/app
Hello, Hatsune Miku
```

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

