CP353004/SC313 004 Software Engineering (2/2567)
ผศ.ดร.ชิตสุธา สุ่มเล็ก
Lab Worksheet

ชื่อ-นามสกุล จิรันธนิน สิทธิกุล รหัสนักศึกษา 653380282-6 Section 2

Lab#8 – Software Deployment Using Docker

วัตถุประสงค์การเรียนรู้

1. ผู้เรียนสามารถอธิบายเกี่ยวกับ Software deployment ได้
2. ผู้เรียนสามารถสร้างและรัน Container จาก Docker image ได้
3. ผู้เรียนสามารถสร้าง Docker files และ Docker images ได้
4. ผู้เรียนสามารถนำซอฟต์แวร์ที่พัฒนาขึ้นให้สามารถรันบนสภาพแวดล้อมเดียวกันและทำงานร่วมกันกับสมาชิกในทีมพัฒนาซอฟต์แวร์ผ่าน Docker hub ได้
5. ผู้เรียนสามารถเริ่มต้นใช้งาน Jenkins เพื่อสร้าง Pipeline ในการ Deploy งานได้

Pre-requisite

1. ติดตั้ง Docker desktop ลงบนเครื่องคอมพิวเตอร์ โดยดาวน์โหลดจาก https://www.docker.com/get-started
2. สร้าง Account บน Docker hub (https://hub.docker.com/signup)
3. กำหนดให้ $ หมายถึง Command prompt และ <> หมายถึง ให้ป้อนค่าของพารามิเตอร์ที่กำหนด

แบบฝึกปฏิบัติที่ 8.1 Hello world - รัน Container จาก Docker image

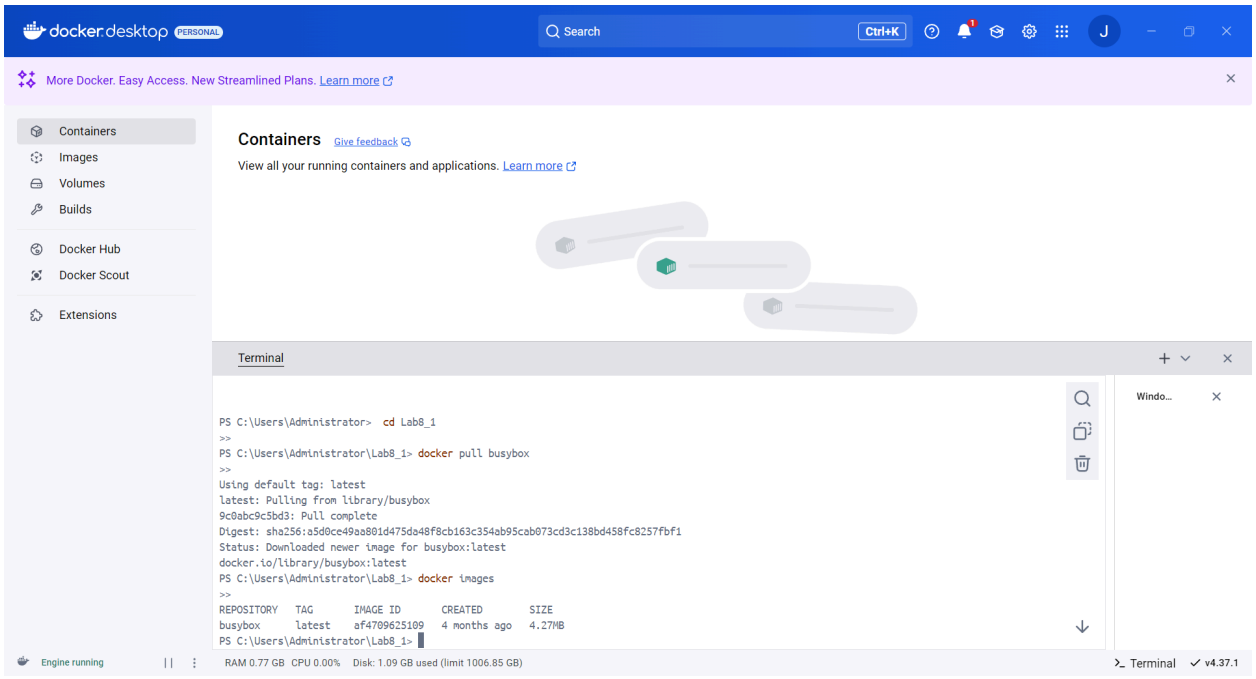
1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
1. เปิด Command line หรือ Terminal บน Docker Desktop จากนั้นสร้าง Directory ชื่อ Lab8_1

CP353004/SC313 004 Software Engineering (2/2567)
ผศ.ดร.ชิตสุธา สุ่มเล็ก
Lab Worksheet

2. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_1 เพื่อใช้เป็น Working directory
3. ป้อนคำสั่ง $ docker pull busybox หรือ $ sudo docker pull busybox สำหรับกรณีที่ติดปัญหา Permission denied (หมายเหตุ: BusyBox เป็น software suite ที่รองรับคำสั่งบางอย่างบน Unix - https://busybox.net)

4. ป้อนคำสั่ง $ docker images

[Check point#1] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ พร้อมกับตอบคำถามต่อไปนี้



(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

คอลัมน์ `REPOSITORY` แสดงชื่อของ **Docker image** ที่ใช้ในการสร้าง container นั้น ๆ

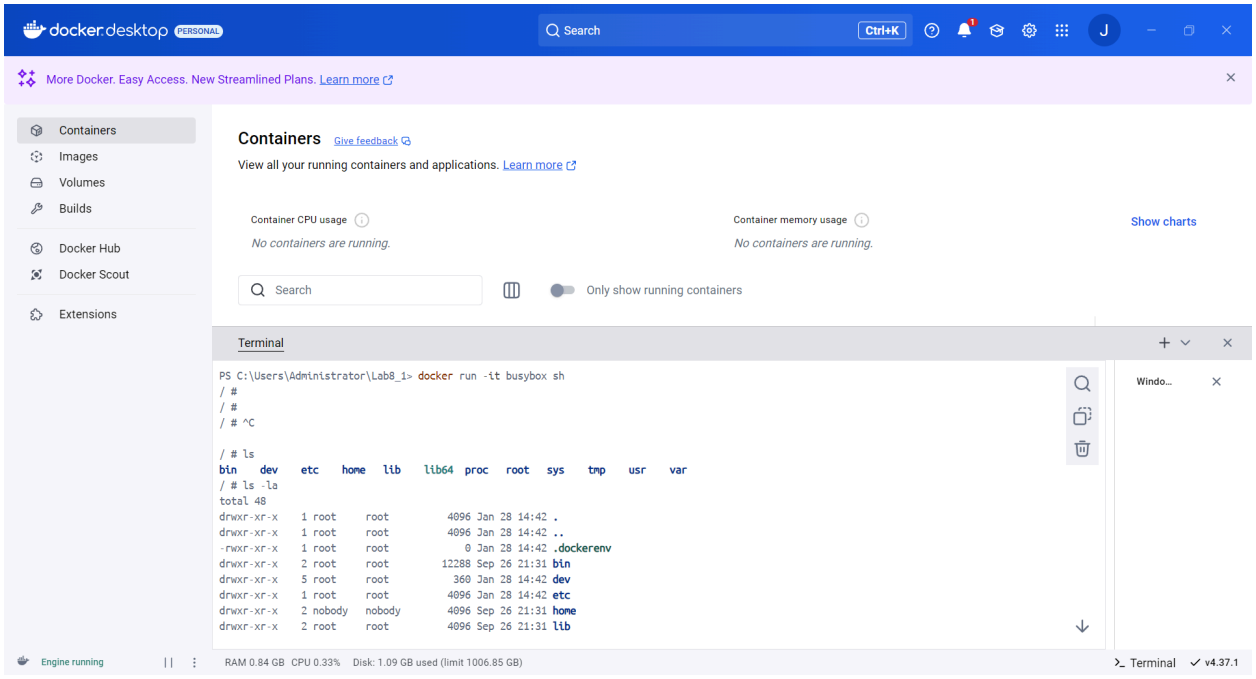
(2) Tag ที่ใช้บ่งบอกถึงอะไร

หากมีการระบุ **tag** ก็จะปรากฏในคอลัมน์นี้ด้วย เช่น `nginx:latest` หรือ `busybox:1.35`

CP353004/SC313 004 Software Engineering (2/2567)
ผศ.ดร.ชิตสุธา สุ่มเล็ก
Lab Worksheet

5. ป้อนคำสั่ง $ docker run busybox
6. ป้อนคำสั่ง $ docker run -it busybox sh
7. ป้อนคำสั่ง ls
8. ป้อนคำสั่ง ls -la
9. ป้อนคำสั่ง exit
10. ป้อนคำสั่ง $ docker run busybox echo “Hello ชื่อและนามสกุลของนักศึกษา from busybox”
11. ป้อนคำสั่ง $ docker ps -a

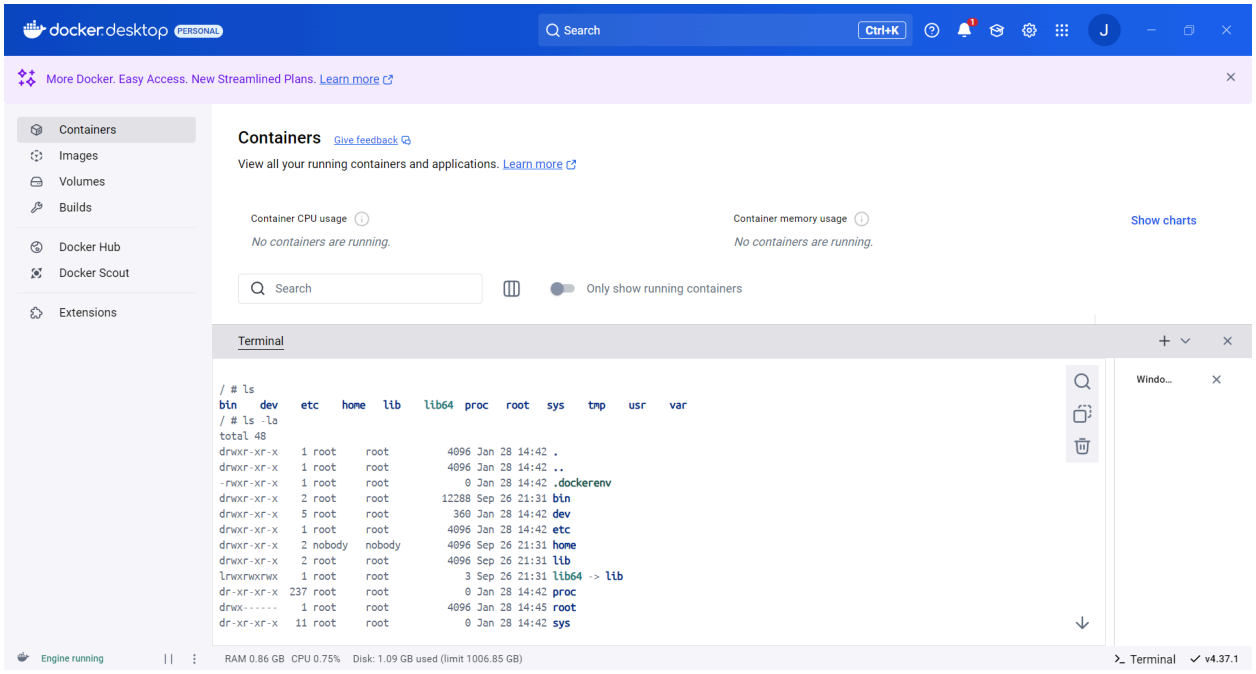
[Check point#2] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ตั้งแต่ขั้นตอนที่ 6-12 พร้อมกับตอบคำถามต่อไปนี้



CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet




```
PS C:\Users\Administrator\Lab8_1> docker run busybox echo "Hello จิรันธนิน สิทธิกุล from busybox"
>>
Hello จิรันธนิน สิทธิกุล from busybox
PS C:\Users\Administrator\Lab8_1>
```

```
Hello จิรันธนิน สิทธิกุล from busybox
PS C:\Users\Administrator\Lab8_1> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                      PORTS     NAMES
46f6932aa03a   busybox   "echo 'Hello จิรันธน…"   38 seconds ago   Exited (0) 37 seconds ago             affectionate_cartwright
d8acef8c08fa   busybox   "sh"                     10 minutes ago   Exited (0) 6 minutes ago              wizardly_snyder
67cd8aac6241   busybox   "sh"                     10 minutes ago   Exited (0) 10 minutes ago             busy_aryabhata
PS C:\Users\Administrator\Lab8_1>
```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

เมื่อใช้ `-it` ร่วมกัน จะสามารถเปิด **Interactive Shell** ภายใน Container ได้ เช่น `sh` หรือ `bash` เหมาะสำหรับการสำรวจ Container, แก้ไขไฟล์, หรือรันคำสั่งทีละคำสั่ง

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

คอลัมน์ STATUS แสดง **สถานะการทำงานของ Container** เช่น Up, Exited, Created, Paused, หรือ Restarting และยังแสดงเวลาที่เกี่ยวข้อง เช่น "Up 5 minutes" หรือ "Exited 2 minutes ago" เพื่อบอกสถานะปัจจุบันและประวัติการทำงาน

12. ป้อนคำสั่ง $ docker rm <container ID ที่ต้องการลบ>

[Check point#3] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 13

```
PS C:\Users\Administrator\Lab8_1> docker rm d8acef8c08fa
d8acef8c08fa
PS C:\Users\Administrator\Lab8_1>
```

แบบฝึกปฏิบัติที่ 8.2: สร้าง Docker file และ Docker image

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
2. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_2
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_2 เพื่อใช้เป็น Working directory
4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ (Windows) บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ Text Editor ที่มี

```
FROM busybox
CMD echo “Hi there. This is my first docker image.”
CMD echo “ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น”
```

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

```
$ cat > Dockerfile << EOF
FROM busybox
```

CMD echo “Hi there. This is my first docker image.”
CMD echo “ชื่อ-นามสกุล รหัสนักศึกษา ชื่อเล่น”
EOF

หรือใช้คำสั่ง

$ touch Dockerfile

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้
   $ docker build -t <ชื่อ Image> .
6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

[Check point#4] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 5 พร้อมกับตอบคำถามต่อไปนี้

```
PS C:\Users\Administrator> mkdir Lab8_2
>>


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/28/2025  10:10 PM                Lab8_2

PS C:\Users\Administrator> cd Lab8_2
>>
PS C:\Users\Administrator\Lab8_2> docker build -t my_first_image .
```

```
PS C:\Users\Administrator\Lab8_2> docker build -t my_first_image .
>>
[+] Building 0.1s (1/1) FINISHED                                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                              0.0s
 => => transferring dockerfile: 2B                                                                                                0.0s
ERROR: failed to solve: failed to read dockerfile: open Dockerfile: no such file or directory
PS C:\Users\Administrator\Lab8_2> docker build -t my_first_image .
>>
[+] Building 0.3s (5/5) FINISHED                                                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                              0.0s
 => => transferring dockerfile: 255B                                                                                              0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)  0.0s
 => WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last one will b  0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)  0.0s
 => [internal] load metadata for docker.io/library/busybox:latest                                                                 0.0s
 => [internal] load .dockerignore                                                                                                 0.0s
 => => transferring context: 2B                                                                                                   0.0s
```

```
PS C:\Users\Administrator\Lab8_2> docker images
>>
REPOSITORY       TAG       IMAGE ID       CREATED        SIZE
busybox          latest    af4709625109   4 months ago   4.27MB
my_first_image   latest    3292f059f081   4 months ago   4.27MB
PS C:\Users\Administrator\Lab8_2>
```

```
PS C:\Users\Administrator\Lab8_2> docker run my_first_image
>>
จิรันธนิน สิงธิกุล รหัสนักศึกษา 653380282-6 ชื่อเล่น ปั้น
PS C:\Users\Administrator\Lab8_2>
```

(1) คำสั่งที่ใช้ในการ run คือ
docker run my_first_image

(2) Option -t ในคำสั่ง $ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

Option `-t` ช่วยให้คุณกำหนดชื่อและ tag ให้กับ Docker image เพื่อให้สามารถจัดการและอ้างอิง image นั้นได้ง่ายขึ้นในภายหลัง.

แบบฝึกปฏิบัติที่ 8.3: การแชร์ Docker image ผ่าน Docker Hub

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เอาไว้
2. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_3
3. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_3 เพื่อใช้เป็น Working directory
4. สร้าง Dockerfile.swp ไว้ใน Working directory

สำหรับเครื่องที่ใช้ระบบปฏิบัติการวินโดวส์ บันทึกคำสั่งต่อไปนี้ลงในไฟล์โดยใช้ Text Editor ที่มี

FROM busybox
CMD echo “Hi there. My work is done. You can run them from my Docker image.”
CMD echo “ชื่อ-นามสกุล รหัสนักศึกษา”

CP353004/SC313 004 Software Engineering (2/2567)
ผศ.ดร.ชิตสุธา สุ่มเล็ก
Lab Worksheet

สำหรับเครื่องที่ใช้ระบบปฏิบัติการ MacOS หรือ Linux บนหน้าต่าง Terminal และป้อนคำสั่งต่อไปนี้

```
$ cat > Dockerfile << EOF
FROM busybox
CMD echo “Hi there. My work is done. You can run them from my Docker image.”
CMD echo “ชื่อ-นามสกุล รหัสนักศึกษา”
EOF
```

หรือใช้คำสั่ง

```
$ touch Dockerfile
```

แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

7. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้
   $ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8
5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง
   $ docker run <username ที่ลงทะเบียนกับ Docker Hub>/lab8

[Check point#5] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้ในขั้นตอนที่ 5

```
PS C:\Users\Administrator> mkdir Lab8_3


    Directory: C:\Users\Administrator


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/28/2025  10:39 PM                Lab8_3
```

```
PS C:\Users\Administrator> cd Lab8_3
PS C:\Users\Administrator\Lab8_3> docker build -t Jirunthanin/lab8 .
>>
[+] Building 0.3s (5/5) FINISHED                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                              0.1s
 => => transferring dockerfile: 288B                                                                              0.0s
 => [internal] load metadata for docker.io/library/busybox:latest                                                 0.0s
 => [internal] load .dockerignore                                                                                 0.1s
 => => transferring context: 2B                                                                                   0.0s
 => CACHED [1/1] FROM docker.io/library/busybox:latest                                                            0.0s
 => exporting to image                                                                                            0.0s
 => => exporting layers                                                                                           0.0s
 => => writing image sha256:344f4de4b0fcb10d1a54c6db81752a66b7c776bb14b9cc33a872c3e8bb1158a9                      0.0s
 => => naming to Jirunthanin/lab8                                                                                 0.0s


PS C:\Users\Administrator\Lab8_3> docker run Jirunthanin/lab8
>>
"ชื่อ-นามสกุล จิรันธนิน สิทธิกุล รหัสนักศึกษา 653380282-6"
PS C:\Users\Administrator\Lab8_3>
```

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง
   $ docker push <username ที่ลงทะเบียนกับ Docker Hub>/lab8
   ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push
   $ docker login แล้วป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง
   $ docker login -u <username> -p <password>
   $ docker login -u Jirunthanin -p Sunsiwakul010646
7. ไปที่ Docker Hub กด Tab ชื่อ Tags หรือไปที่ Repository ก็ได้

[Check point#6] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดง Repository ที่มี Docker image (<username>/lab8)

```
PS C:\Users\Administrator\Lab8_3> docker login
>>
Authenticating with existing credentials...
Login Succeeded
```

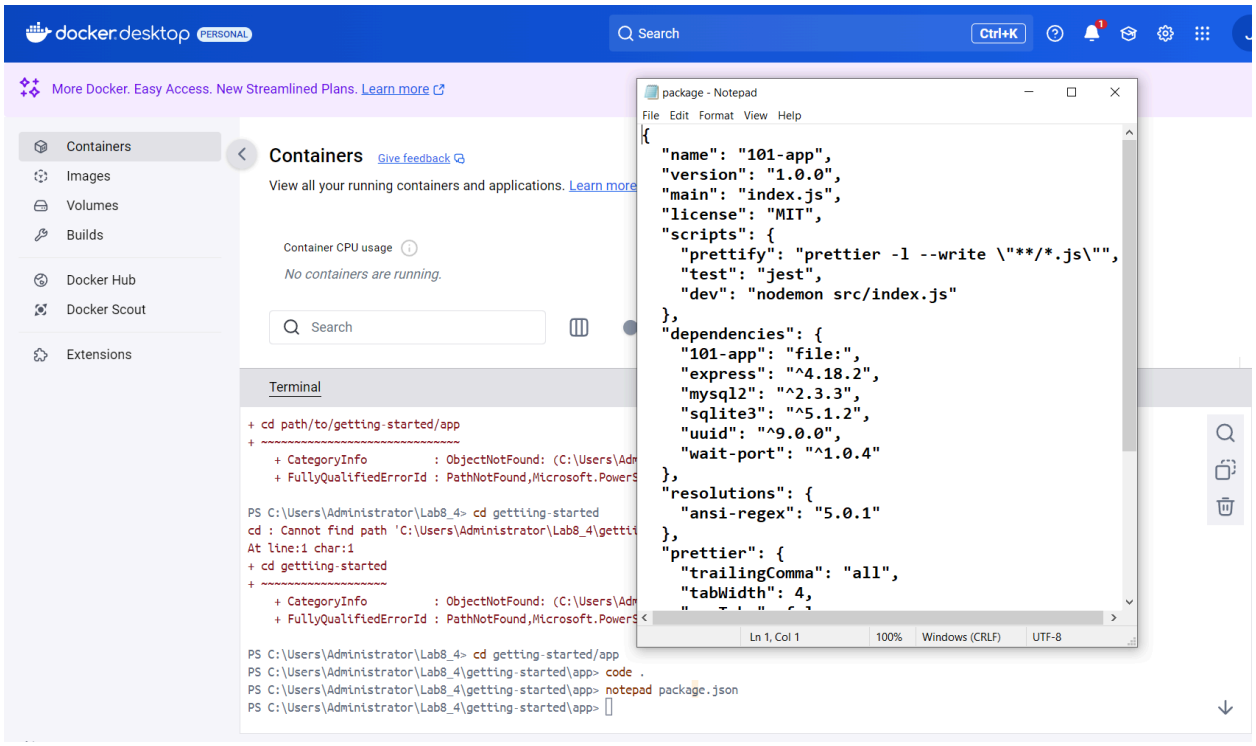
แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

---

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
2. ทำการ Clone ซอร์สโค้ดของเว็บแอปพลิเคชันจาก GitHub repository https://github.com/docker/getting-started.git ลงใน Directory ที่สร้างขึ้น โดยใช้คำสั่ง
   $ git clone https://github.com/docker/getting-started.git
3. เปิดดูองค์ประกอบภายใน getting-started/app เมื่อพบไฟล์ package.json ให้ใช้ Text editor ในการเปิดอ่าน

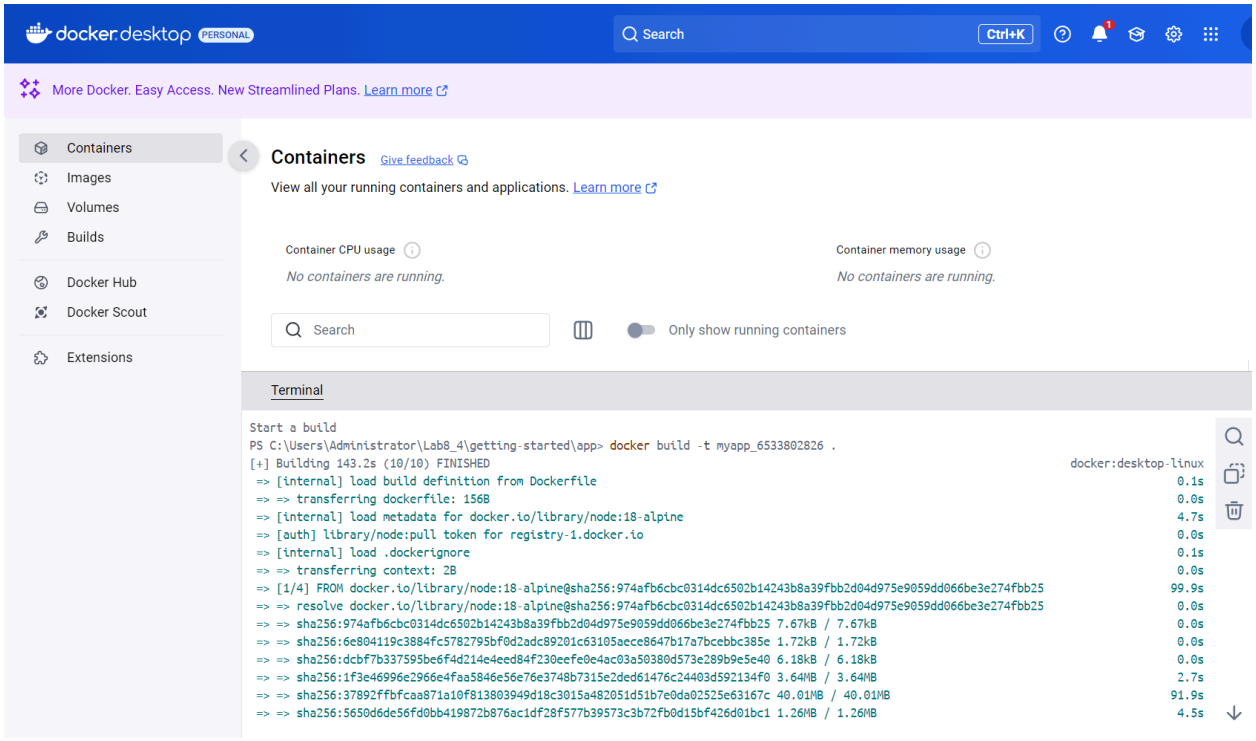
[Check point#7] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงที่อยู่ของ Source code ที่ Clone มาและเนื้อหาของไฟล์ package.json

```
PS C:\Users\Administrator> cd Lab8_4
PS C:\Users\Administrator\Lab8_4> git clone https://github.com/docker/getting-started.git
Cloning into 'getting-started'...
remote: Enumerating objects: 980, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 980 (delta 5), reused 1 (delta 1), pack-reused 971 (from 2)
Receiving objects: 100% (980/980), 5.28 MiB | 3.39 MiB/s, done.
Resolving deltas: 100% (523/523), done.
PS C:\Users\Administrator\Lab8_4>
```



4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปในไฟล์
   FROM node:18-alpine
   WORKDIR /app
   COPY . .
   RUN yarn install --production
   CMD ["node", "src/index.js"]
   EXPOSE 3000
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น myapp_รหัสนศ. ไม่มีขีด
   $ docker build -t <myapp_รหัสนศ. ไม่มีขีด> .

[Check point#8] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ



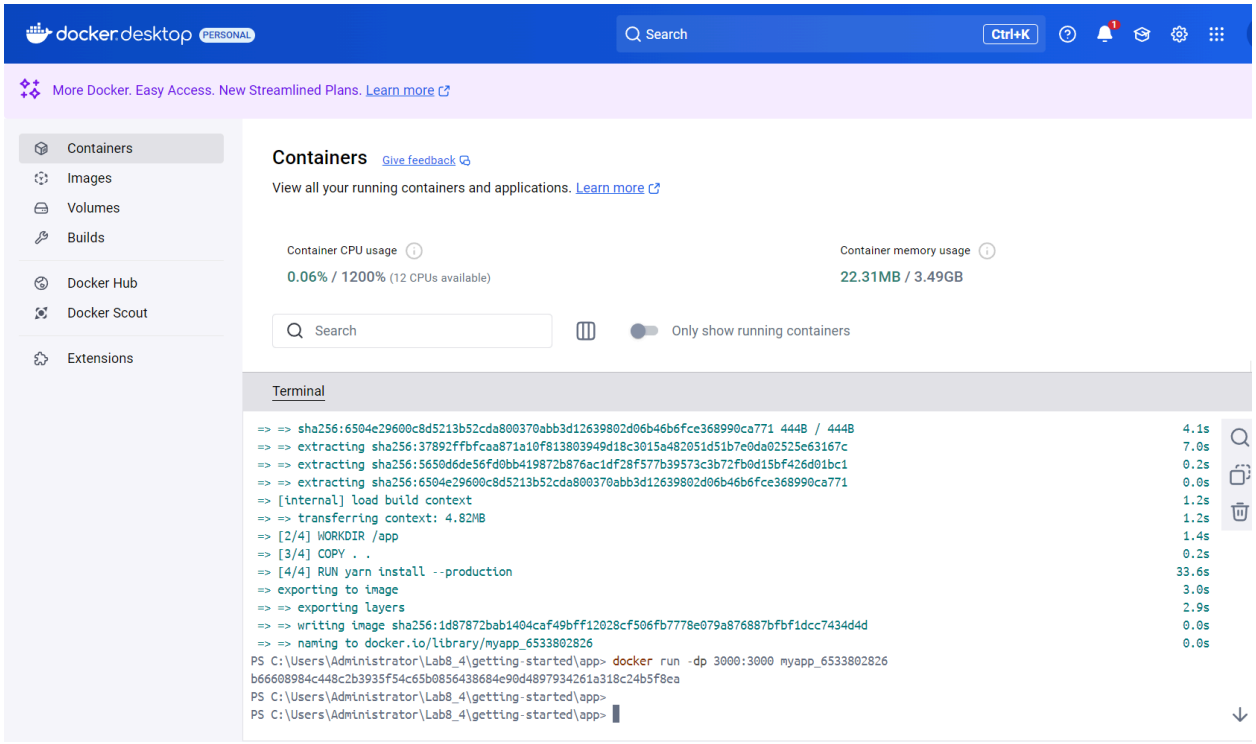
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง
   $ docker run -dp 3000:3000 <myapp_รหัสนศ. ไม่มีขีด>
7. เปิด Browser ไปที่ URL = http://localhost:3000

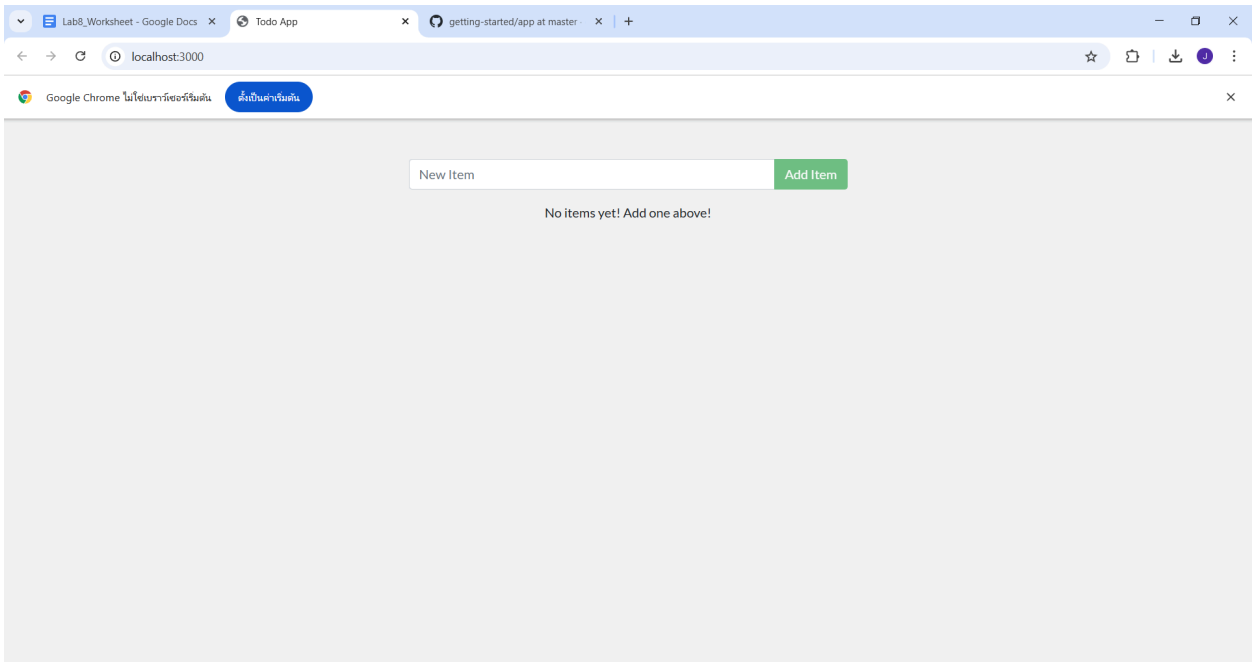
[Check point#9] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้บน Browser และ Dashboard ของ Docker desktop

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

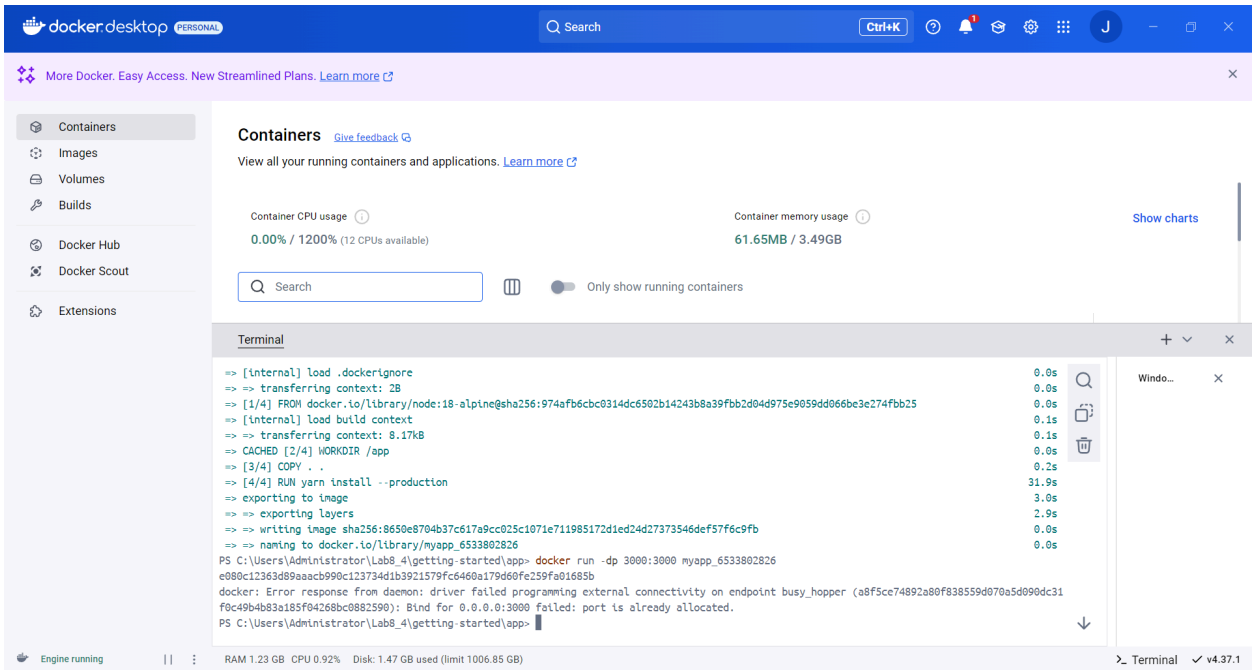




หมายเหตุ: นศ.สามารถทดลองเล่น Web application ที่ทำงานอยู่ได้

8. ทำการแก้ไข Source code ของ Web application ดังนี้
    a. เปิดไฟล์ src/static/js/app.js ด้วย Editor และแก้ไขบรรทัดที่ 56 จาก
    No items yet! Add one above! เป็น
    There is no TODO item. Please add one to the list. By ชื่อและนามสกุลของนักศึกษา
    b. Save ไฟล์ให้เรียบร้อย
9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5
10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้



(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

Error ที่เจอเกิดจากการที่พอร์ต `3000` ถูกใช้งานอยู่แล้วบนเครื่อง.

สามารถเลือกวิธีหยุดโปรเซสที่ใช้งานพอร์ตนั้นหรือเลือกใช้พอร์ตอื่นในการรันคอนเทนเนอร์.

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้
    a. ผ่าน Command line interface
        i. ใช้คำสั่ง $ docker ps เพื่อดู Container ID ที่ต้องการจะลบ
        ii. Copy หรือบันทึก Container ID ไว้
        iii. ใช้คำสั่ง $ docker stop b66608984c44 <Container ID ที่ต้องการจะลบ> เพื่อหยุดการทำงานของ Container ดังกล่าว
        iv. ใช้คำสั่ง $ docker rm b66608984c44 <Container ID ที่ต้องการจะลบ> เพื่อทำการลบ

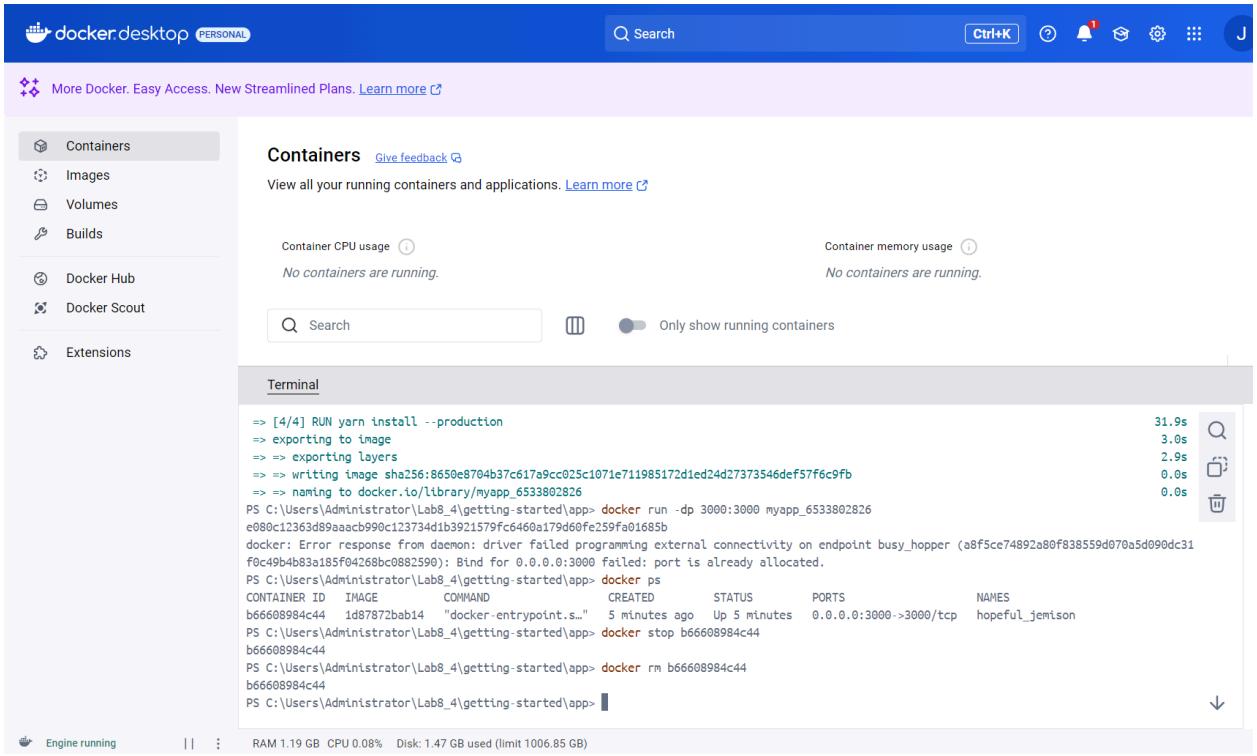
    b. ผ่าน Docker desktop
        i. ไปที่หน้าต่าง Containers
        ii. เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
        iii. ยืนยันโดยการกด Delete forever
12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6
13. เปิด Browser ไปที่ URL = http://localhost:3000

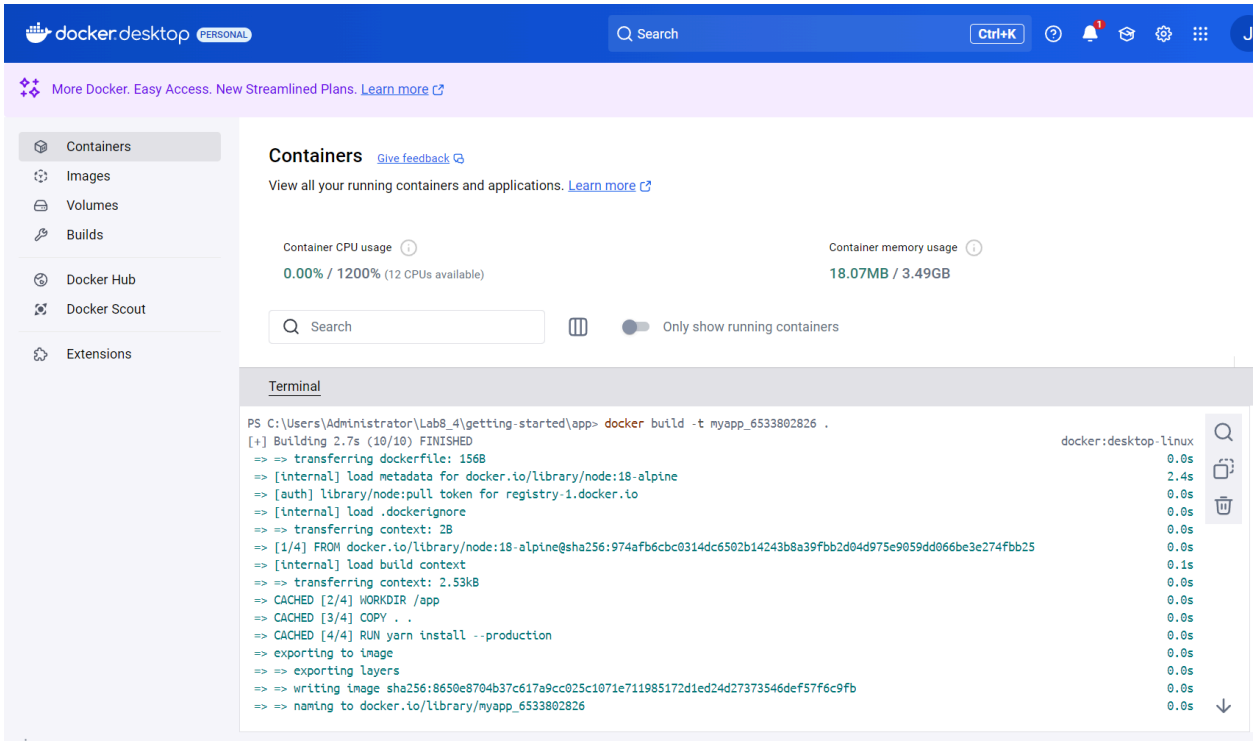
[Check point#11] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้บน Browser และ Dashboard ของ Docker desktop

CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet

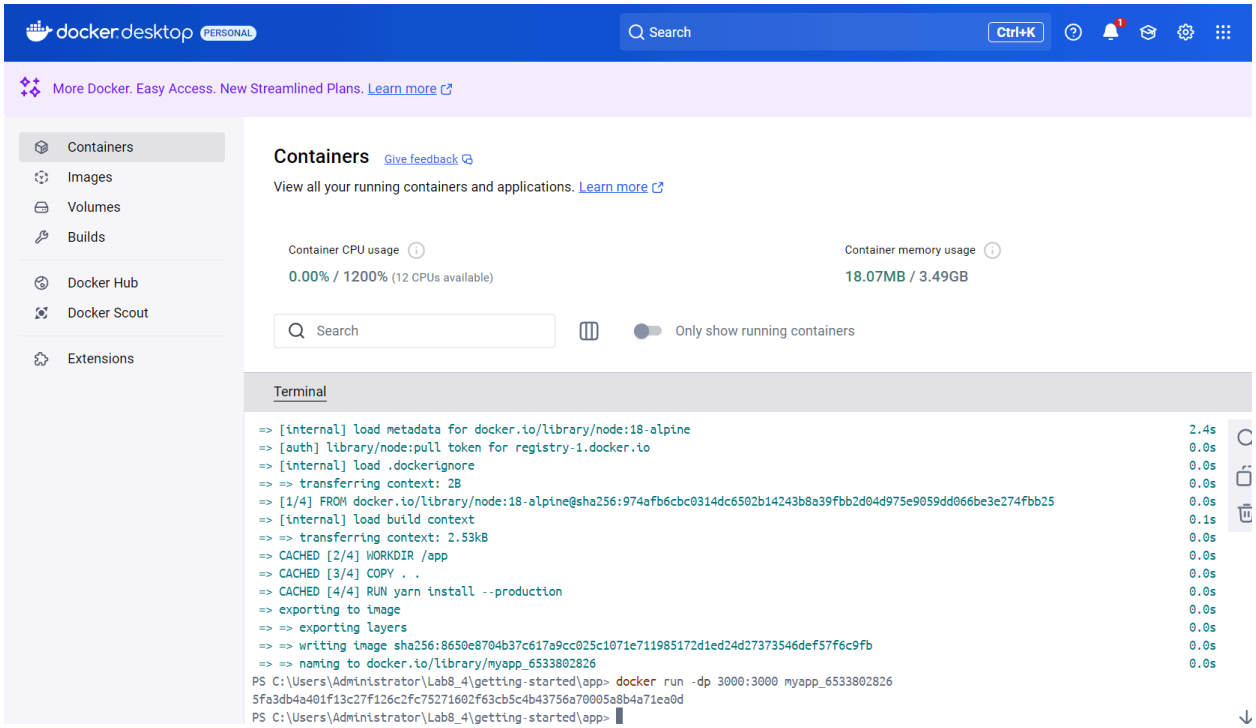


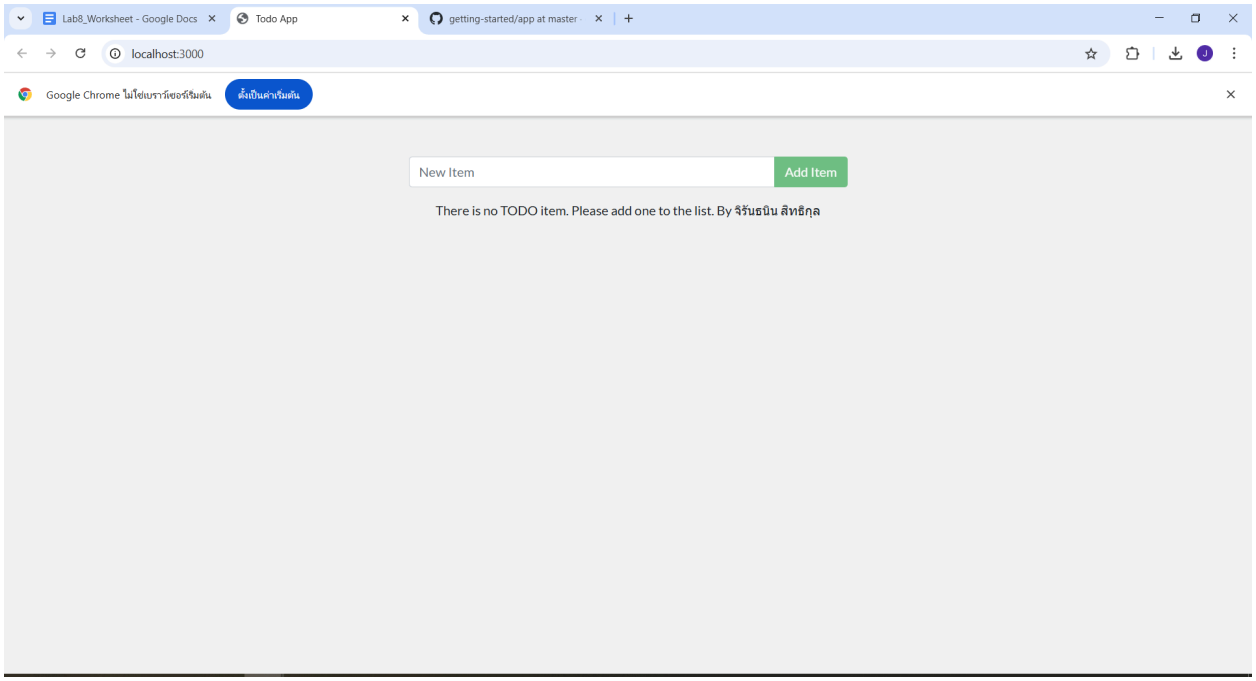


CP353004/SC313 004 Software Engineering (2/2567)

ผศ.ดร.ชิตสุธา สุ่มเล็ก

Lab Worksheet



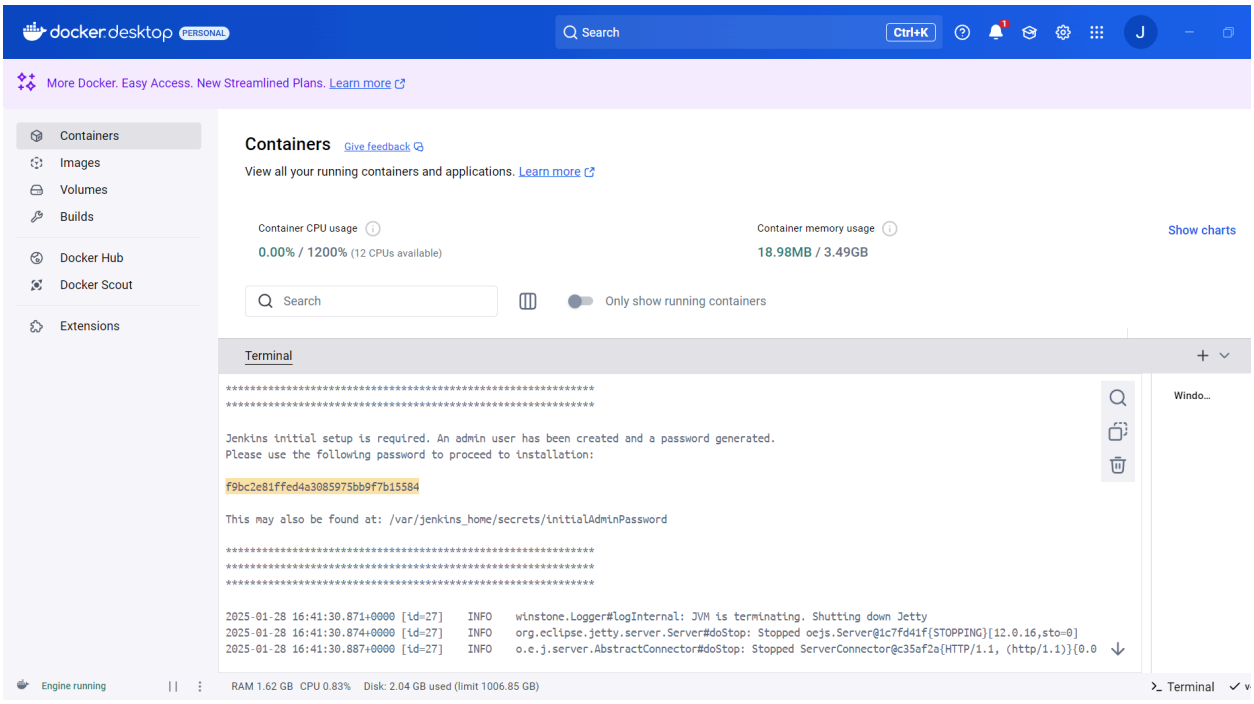


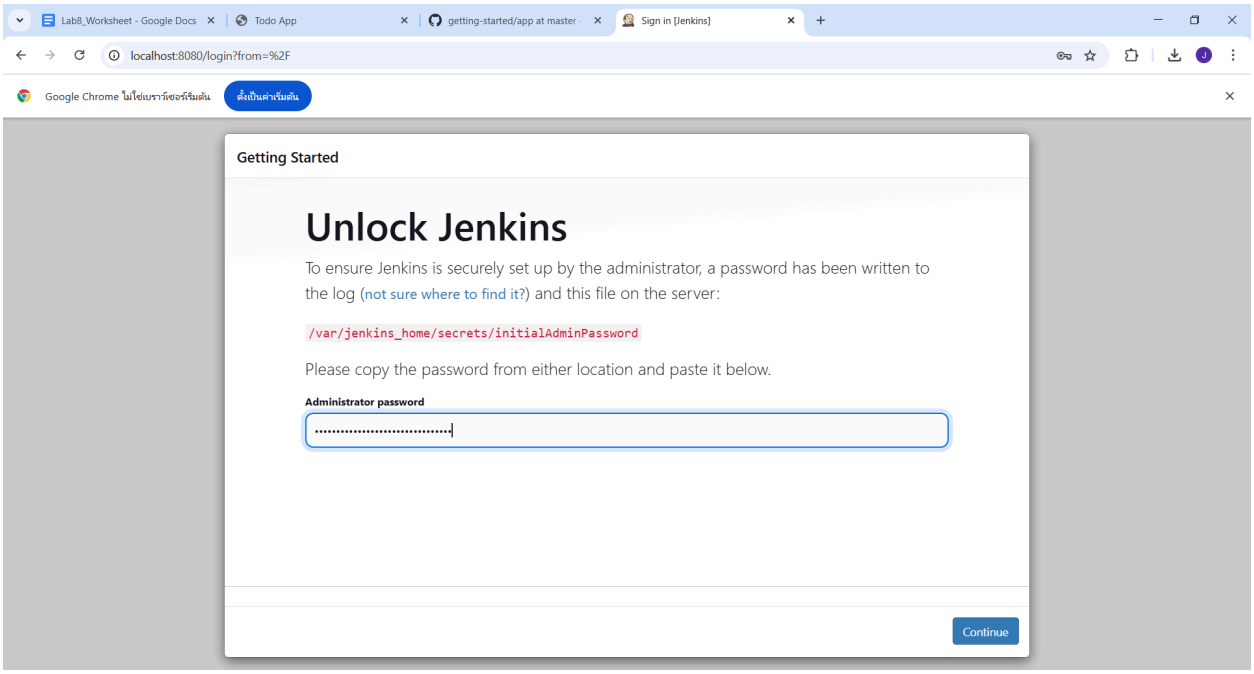
CP353004/SC313 004 Software Engineering (2/2567)
ผศ.ดร.ชิตสุธา สุ่มเล็ก
Lab Worksheet

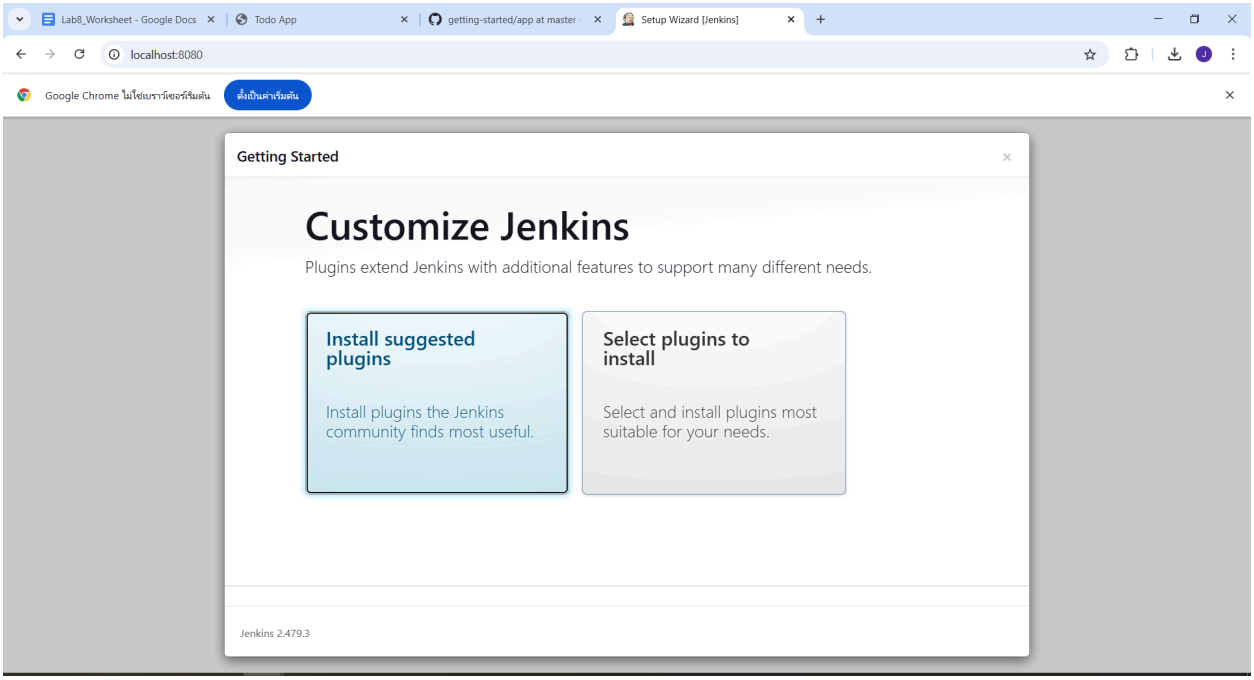
แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

1. เปิด Command line หรือ Terminal บน Docker Desktop
2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต
   $ docker run -p 8080:8080 -p 50000:50000
   --restart=on-failure jenkins/jenkins:lts-jdk17
   หรือ
   $ docker run -p 8080:8080 -p 50000:50000
   --restart=on-failure -v jenkins_home:/var/jenkins_home
   jenkins/jenkins:lts-jdk17
3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[Check point#12] Capture หน้าจอที่แสดงผล Admin password



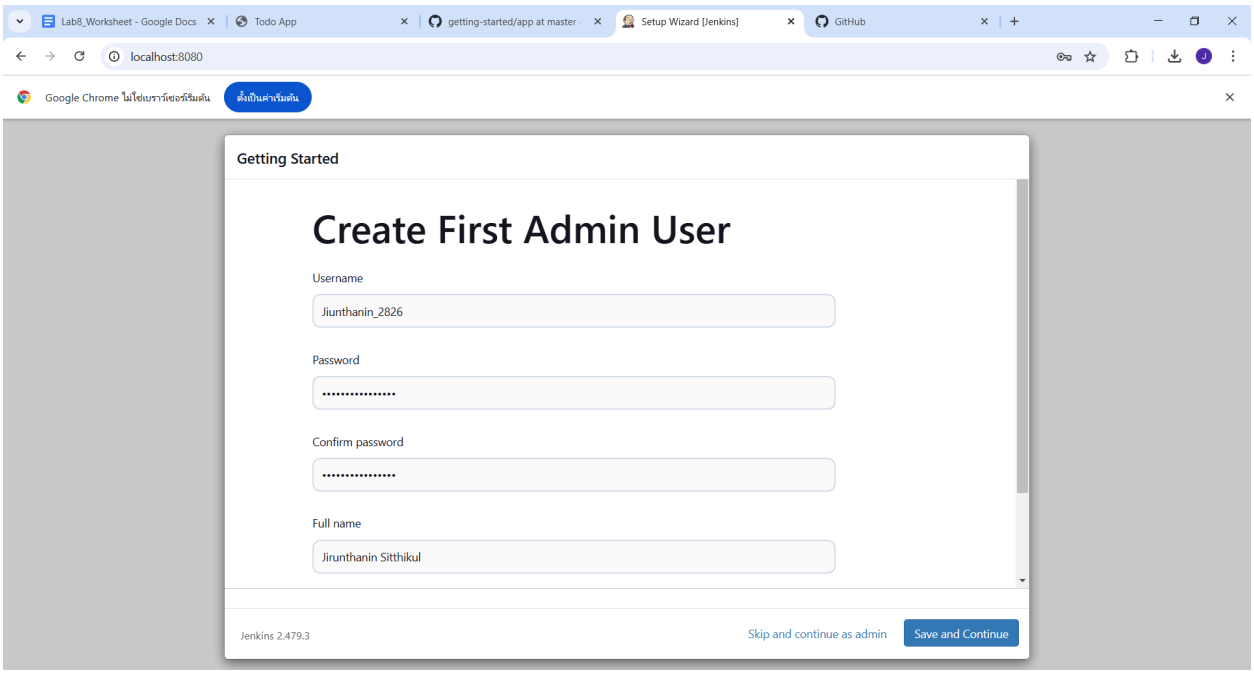


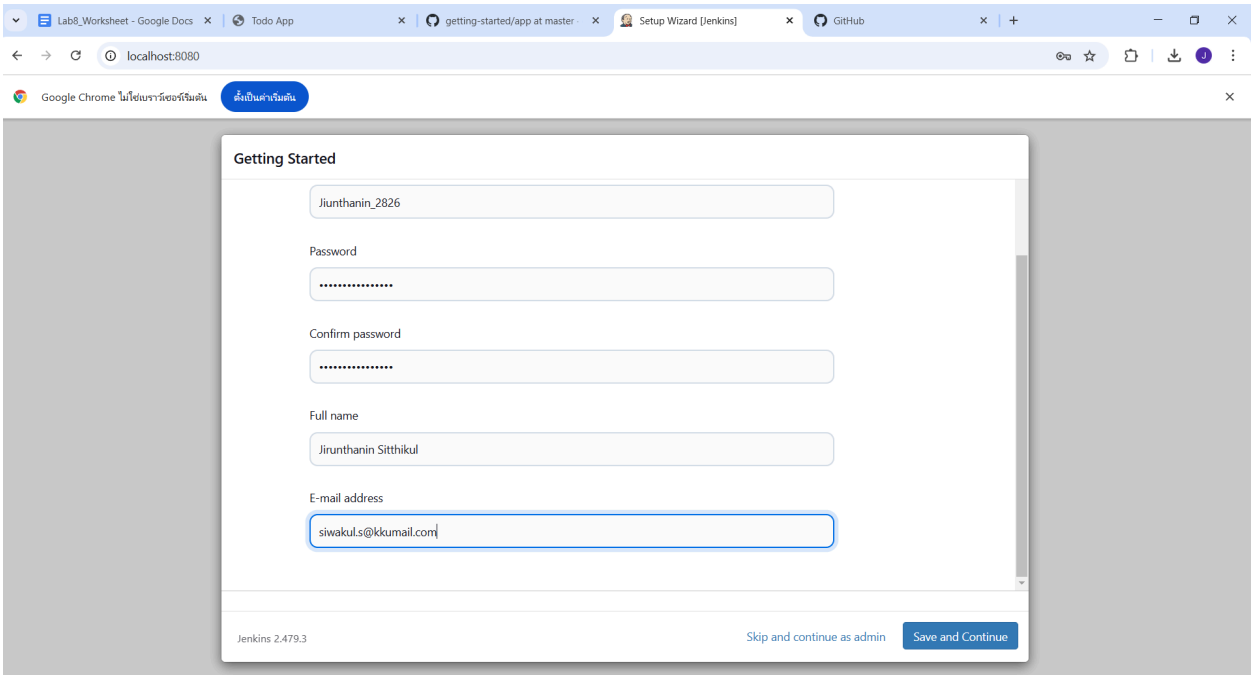


4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3

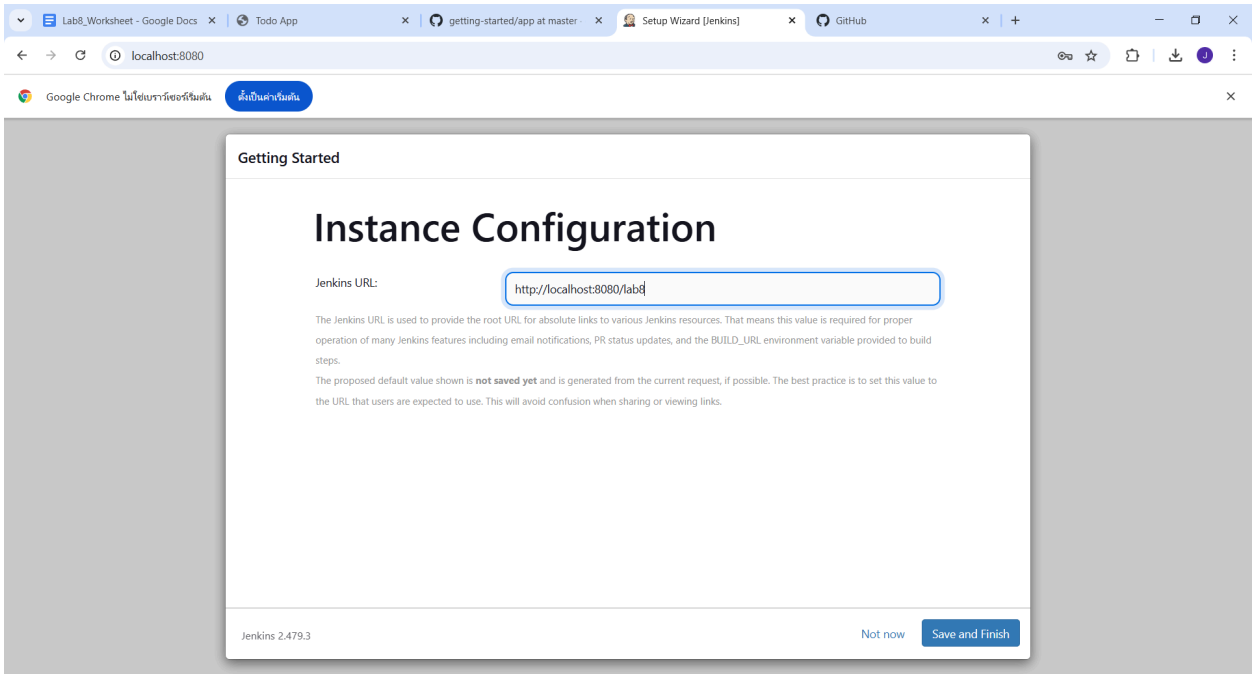
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษา พร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

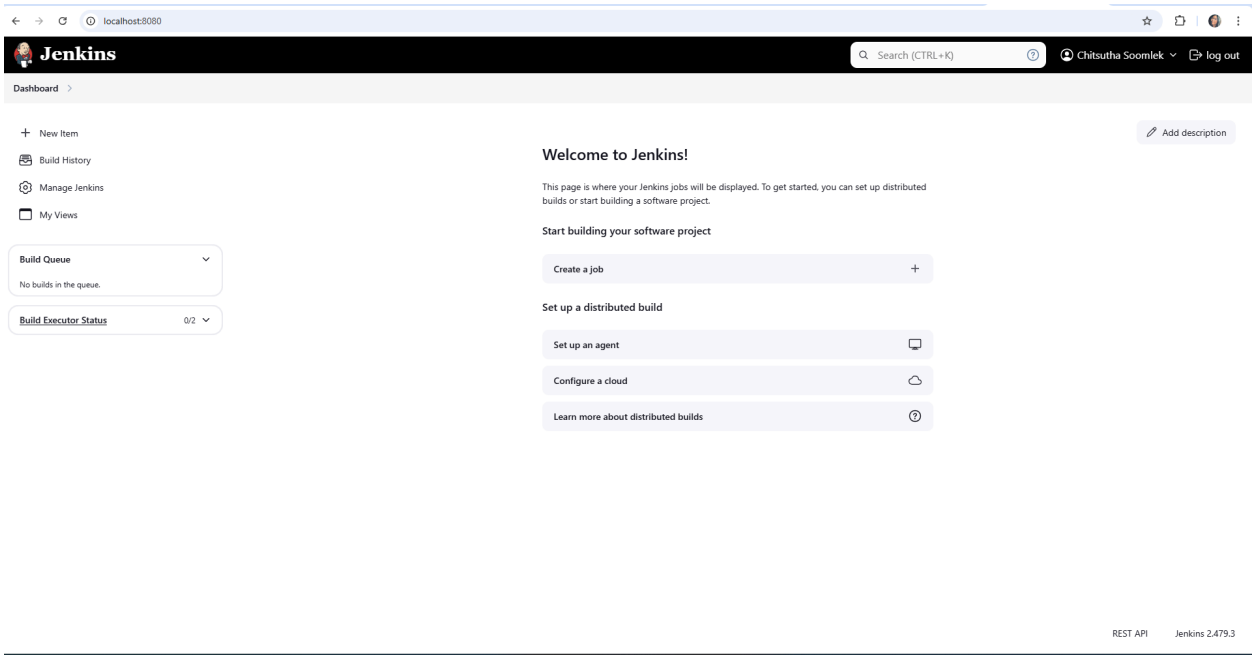




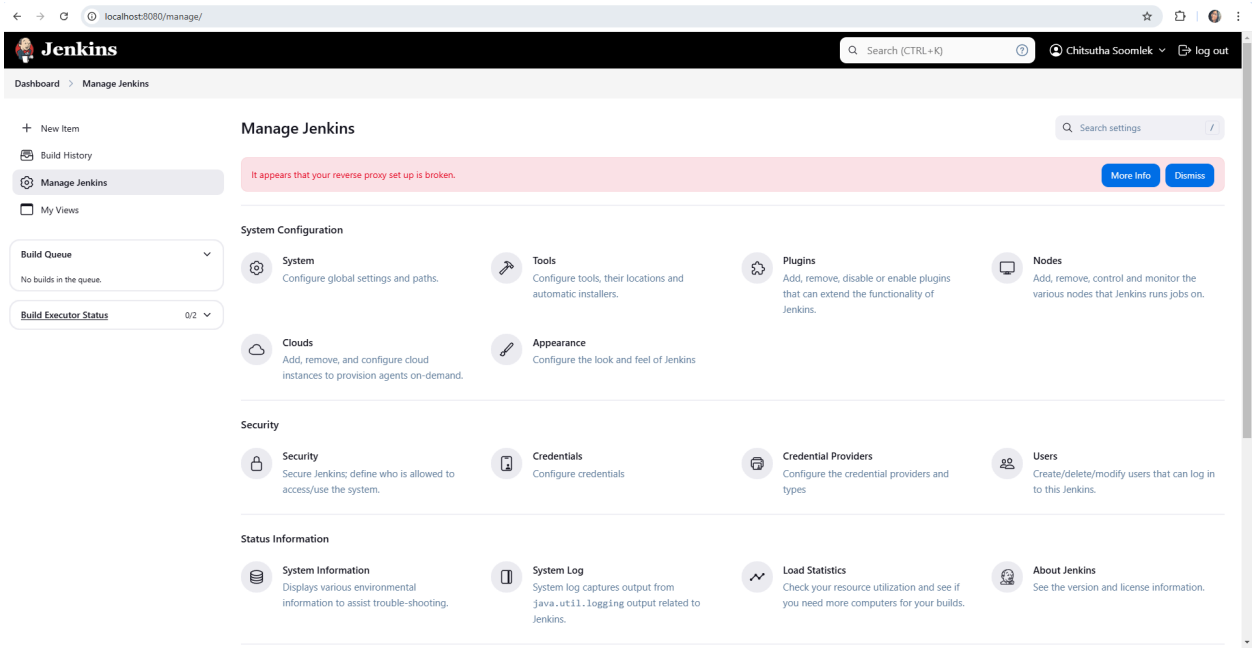
7. กำหนด Jenkins URL เป็น http://localhost:8080/lab8



8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

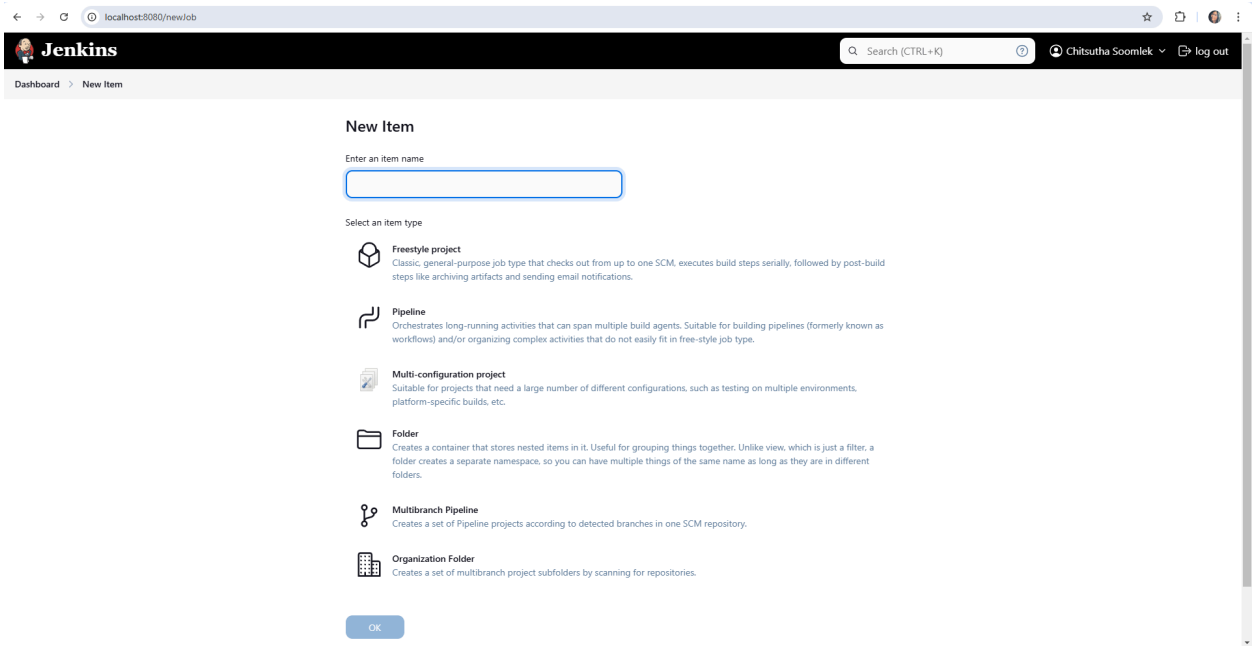


9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม

11. กลับไปที่หน้า Dashboard แล้วสร้าง Pipeline อย่างง่าย โดยกำหนด New item เป็น Freestyle project และตั้งชื่อเป็น UAT



12. นำไฟล์ .robot ที่ทำให้แบบฝึกปฏิบัติที่ 7 (Lab#7) ไปไว้บน Repository ของนักศึกษา จากนั้นตั้งค่าที่จำเป็นในหน้านี้ทั้งหมด ดังนี้

Description: Lab 8.5
GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)
Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที
Build Steps: เลือก Execute shell แล้วใส่คำสั่งในการรันไฟล์ .robot (หากไฟล์ไม่ได้อยู่ในหน้าแรกของ repository ให้ใส่ Path ไปถึงไฟล์ให้เรียบร้อยด้วย)

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

(1) คำสั่งที่ใช้ในการ Execute ไฟล์ .robot ใน Build Steps คือ robot testLogin.robot

Post-build action: เพิ่ม Publish Robot Framework test results -> ระบุไดเร็คทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่านแล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้ (เช่น 20, 80)

13. กด Apply และ Save
14. สั่ง Build Now

[Check point#15] Capture หน้าจอแสดงหน้าหลักของ Pipeline และ Console Output