ชื่อ-นามสกุล เอกะรัตตะฆัต สืบเหล่างิ้ว รหัสนักศึกษา 653380030-3 Section 1

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
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

```
PS D:\coding\SE\Lab8_1> docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
9c0abc9c5bd3: Pull complete
Digest: sha256:a5d0ce49aa801d475da48f8cb163c354ab95cab073cd3c138bd458fc8257fbf1
Status: Downloaded newer image for busybox:latest
docker.io/library/busybox:latest

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview busybox
PS D:\coding\SE\Lab8_1> ls
PS D:\coding\SE\Lab8_1> docker images
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
busybox      latest    af4709625109   3 months ago   4.27MB
PS D:\coding\SE\Lab8_1>
```

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร

ชื่อของ docker image

(2) Tag ที่ใช้บ่งบอกถึงอะไร

Version release

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

```
PS D:\coding\SE\Lab8_1> docker run busybox
PS D:\coding\SE\Lab8_1> docker run -it busybox
/ # ls
bin    dev    etc    home   lib    lib64  proc   root   sys    tmp    usr    var
/ #  ls -la
total 48
drwxr-xr-x    1 root     root          4096 Jan 22 03:34 .
drwxr-xr-x    1 root     root          4096 Jan 22 03:34 ..
-rwxr-xr-x    1 root     root             0 Jan 22 03:34 .dockerenv
drwxr-xr-x    2 root     root         12288 Sep 26 21:31 bin
drwxr-xr-x    5 root     root           360 Jan 22 03:34 dev
drwxr-xr-x    1 root     root          4096 Jan 22 03:34 etc
drwxr-xr-x    2 nobody   nobody        4096 Sep 26 21:31 home
drwxr-xr-x    2 root     root          4096 Sep 26 21:31 lib
lrwxrwxrwx    1 root     root             3 Sep 26 21:31 lib64 -> lib
dr-xr-xr-x  259 root     root             0 Jan 22 03:34 proc
drwx------    1 root     root          4096 Jan 22 03:34 root
dr-xr-xr-x   11 root     root             0 Jan 22 03:34 sys
drwxrwxrwt    2 root     root          4096 Sep 26 21:31 tmp
drwxr-xr-x    4 root     root          4096 Sep 26 21:31 usr
drwxr-xr-x    4 root     root          4096 Sep 26 21:31 var
/ # exit
PS D:\coding\SE\Lab8_1> docker run busybox echo "Hello Aekarattakhat Sueplaongio From busybox"
Hello Aekarattakhat Sueplaongio From busybox
PS D:\coding\SE\Lab8_1> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS                          PORTS     NAMES
021da5df3e3f   busybox   "echo 'Hello Aekarat…"   7 seconds ago        Exited (0) 6 seconds ago                  condescending_matsumoto
cde9e33434c6   busybox   "sh"                     About a minute ago   Exited (0) About a minute ago             boring_goodall
048cd4b7b6f9   busybox   "sh"                     2 minutes ago        Exited (0) 2 minutes ago                  serene_tharp
PS D:\coding\SE\Lab8_1>
```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

เชื่อมต่อ cli ของ docker contanier ที่รันอยู่ให้แสดงออกมา

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

สถานะของ docker container ในขณะนั้น

12. ป้อนคำสั่ง $ docker rm <container ID ที่ต้องการลบ>

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

```
PS D:\coding\SE\Lab8_1> docker rm 021da5df3e3f
021da5df3e3f
PS D:\coding\SE\Lab8_1> docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS                      PORTS     NAMES
cde9e33434c6   busybox   "sh"      15 minutes ago   Exited (0) 15 minutes ago             boring_goodall
048cd4b7b6f9   busybox   "sh"      16 minutes ago   Exited (0) 16 minutes ago             serene_tharp
PS D:\coding\SE\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 D:\coding\se\lab8_2> docker build -t lab8_2 .
[+] Building 0.1s (5/5) FINISHED                                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                                          0.0s
 => => transferring dockerfile: 192B                                                          0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS si  0.0s
 => WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because   0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS si  0.0s
 => [internal] load metadata for docker.io/library/busybox:latest                             0.0s
 => [internal] load .dockerignore                                                             0.0s
 => => 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 sha256:3177fe0d93460f8c19f4204b4dd7f3f9db40daa3c7adcfca80a3bb5444942824  0.0s
 => => naming to docker.io/library/lab8_2                                                     0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/rl7of2dj7ydhvpjm9xy7me8sy

 3 warnings found (use docker --debug to expand):
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)
 - MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same stage because only the last on

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS D:\coding\se\lab8_2> docker run -it lab8_2
เอกะรี ตตะฆั ต สิ บเหล่ างั้ ว
```

(1) คำสั่งที่ใช้ในการ run คือ

docker run -it lab8_2

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

ตัวช่วยจัดการกับ input และ output ในการเชื่อมต่อกับคอนเทนเนอร์

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 บันทึกคำสั่งต่อไปนี้ลงในไฟล์ โดยใช้ 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”
```

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 D:\coding\se\lab8_3> docker build -t aekarattakhat/lab8 .
[+] Building 0.1s (5/5) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 192B
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavi
 => WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavi
 => [internal] load metadata for docker.io/library/busybox:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => CACHED [1/1] FROM docker.io/library/busybox:latest
 => exporting to image
 => => exporting layers
 => => writing image sha256:3177fe0d93460f8c19f4204b4dd7f3f9db40daa3c7adcfca80a3bb5444942824
 => => naming to docker.io/aekarattakhat/lab8

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/7gn61dkshbxce

 3 warnings found (use docker --debug to expand):
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior rela
 - MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the same st
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior rela

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS D:\coding\se\lab8_3> docker run aekarattakhat/lab8
เอกะรั ตตะฆั ต สึ บเหล่ างิ้  ว
PS D:\coding\se\lab8_3>
```

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

   $ docker push <username ที่ลงทะเบียนกับ Docker Hub>/lab8

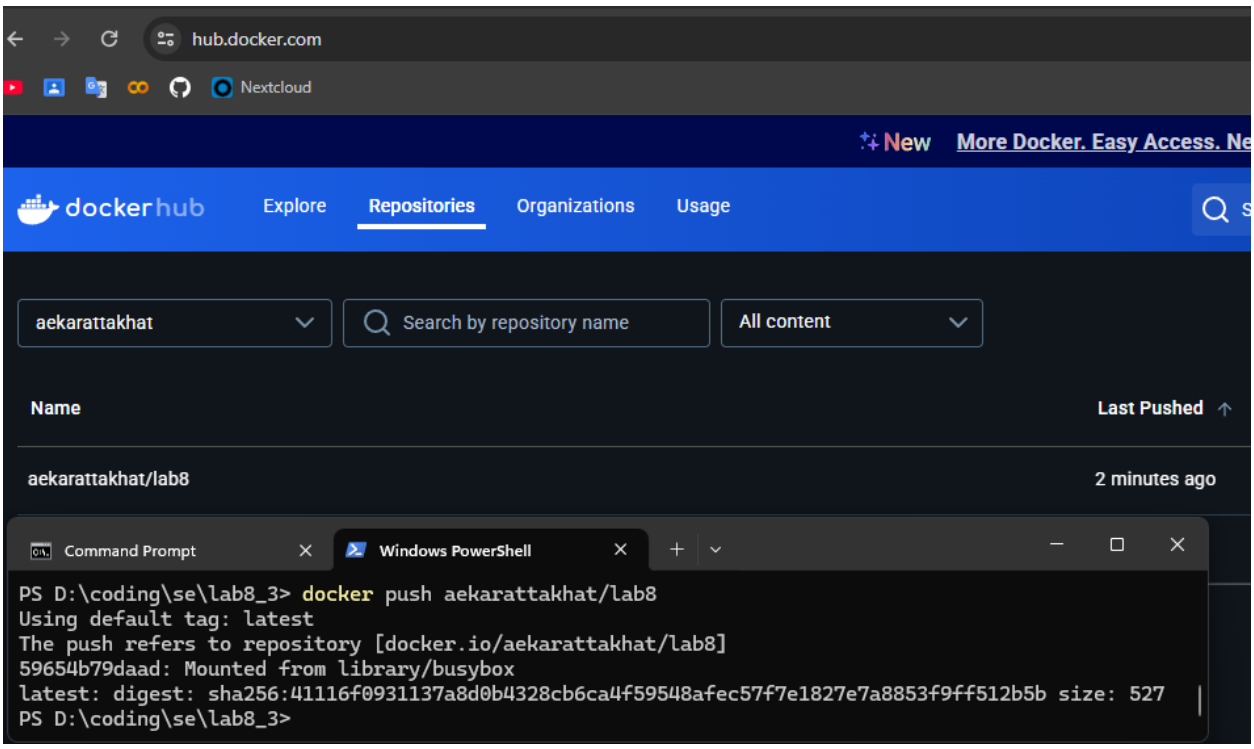
   ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

   $ docker login แล้วป้อน Username และ Password ตามที่ระบุใน Command prompt หรือใช้คำสั่ง

   $ docker login -u <username> -p <password>

7. ไปที่ Docker Hub กด Tab ชื่อ Tags หรือไปที่ Repository ก็ได้

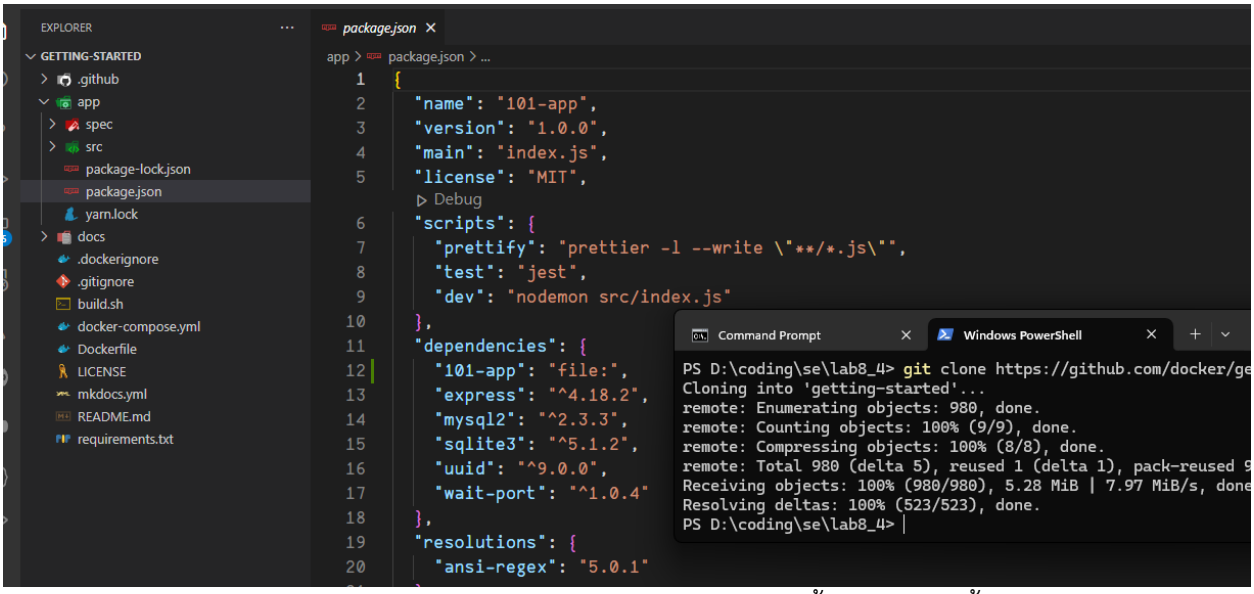
[Check point#6] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดง Repository ที่มี Docker image (<username>/lab8)



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



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

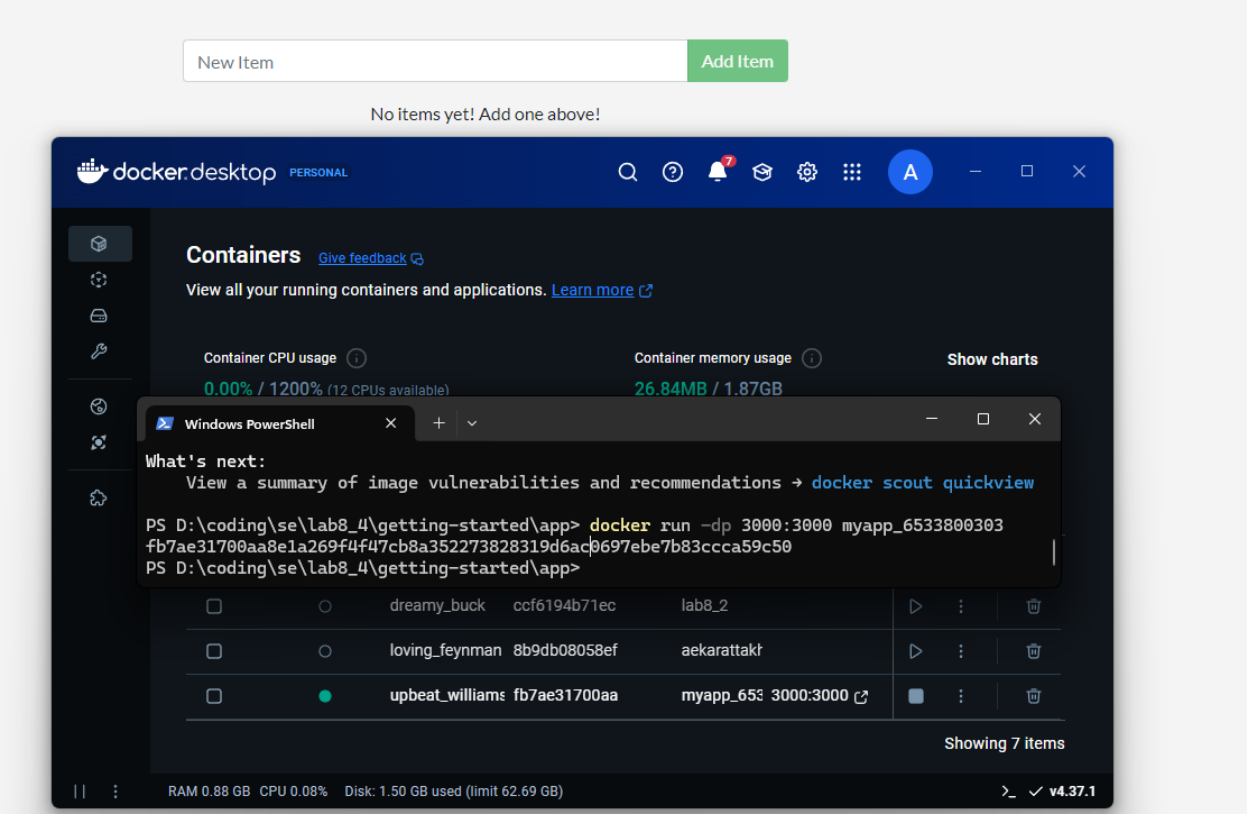
```
PS D:\coding\se\lab8_4\getting-started\app> docker build -t myapp_6533800303 .
[+] Building 24.4s (10/10) FINISHED                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                               0.0s
 => => transferring dockerfile: 156B                                               0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                  4.5s
 => [auth] library/node:pull token for registry-1.docker.io                        0.0s
 => [internal] load .dockerignore                                                  0.0s
 => => transferring context: 2B                                                    0.0s
 => [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243  5.8s
 => => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243  0.0s
 => => sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd06 7.67kB / 7.67kB  0.0s
 => => sha256:6e804119c3884fc5782795bf0d2adc89201c63105aece8647b1 1.72kB / 1.72kB  0.0s
 => => sha256:dcbf7b337595be6f4d214e4eed84f230eefe0e4ac03a50380d5 6.18kB / 6.18kB  0.0s
 => => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c2 3.64MB / 3.64MB  0.7s
 => => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e 40.01MB / 40.01MB  4.5s
 => => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d 1.26MB / 1.26MB  1.2s
 => => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d  0.1s
 => => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce36 444B / 444B  1.3s
 => => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da0252  1.0s
 => => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf4  0.0s
 => => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368  0.0s
 => [internal] load build context                                                  0.4s
 => => transferring context: 4.82MB                                                0.4s
 => [2/4] WORKDIR /app                                                             0.2s
 => [3/4] COPY . .                                                                 0.1s
 => [4/4] RUN yarn install --production                                           13.1s
 => exporting to image                                                             0.6s
 => => exporting layers                                                            0.6s
 => => writing image sha256:3dd03bba2c00f5848b3c7347928e9fb754b8f427d01ade921233d  0.0s
 => => naming to docker.io/library/myapp_6533800303                                0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/exkma9c
9ss8307f4wqvpewm4g

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

   $ docker run -dp 3000:3000 <myapp_รหัสนศ. ไม่มีขีด>

7. เปิด Browser ไปที่ URL = http://localhost:3000

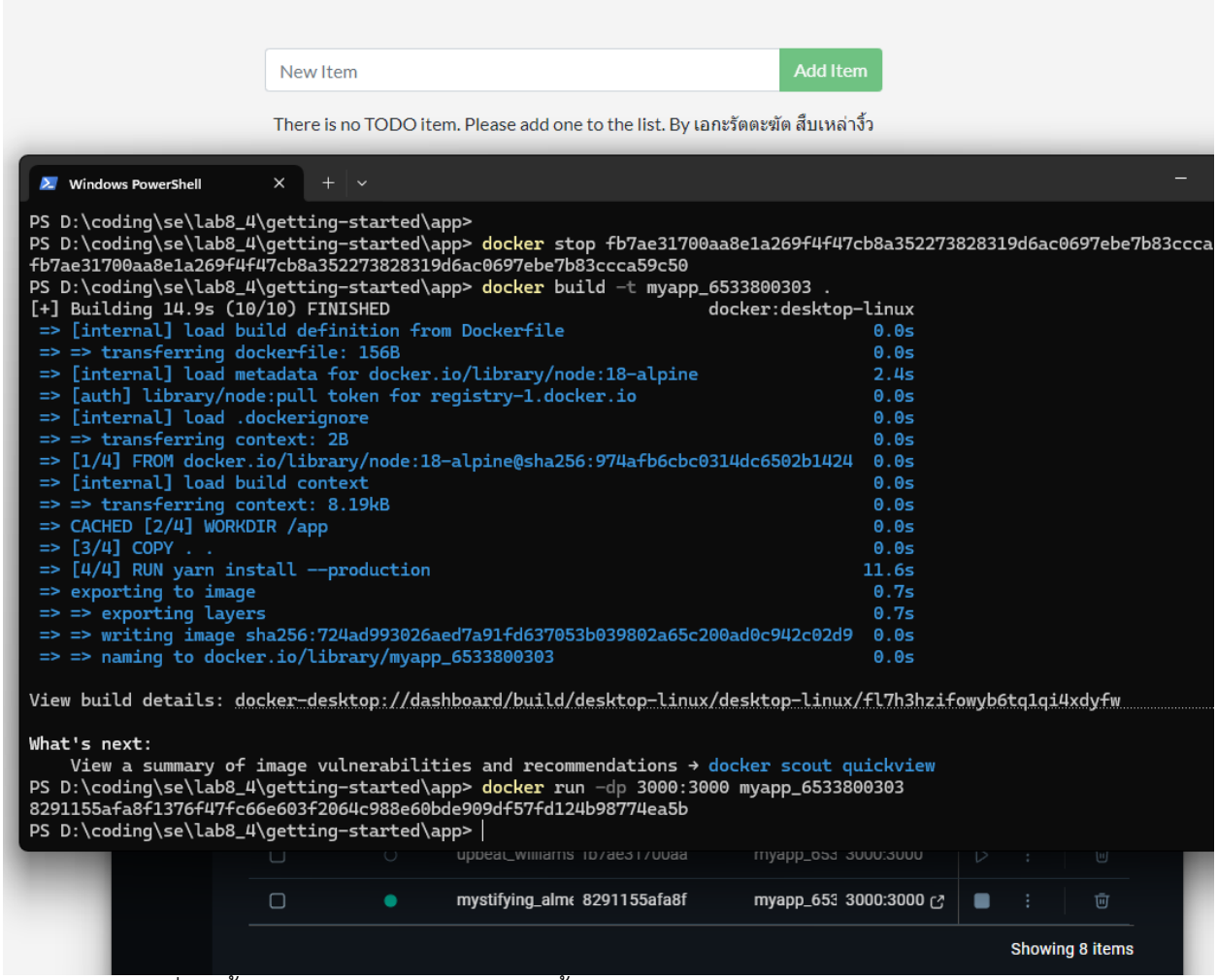
[Check point#9] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้บน Browser และ Dashboard ของ Docker desktop



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

ไม่มี Error เกิดขึ้น

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 <Container ID ที่ต้องการจะลบ> เพื่อหยุดการทำงานของ Container ดังกล่าว

iv. ใช้คำสั่ง $ docker rm <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

```
PS D:\coding\se\lab8_4\getting-started\app> docker ps -a
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS
8291155afa8f   myapp_6533800303     "docker-entrypoint.s…"   6 minutes ago    Up 6 minutes
fb7ae31700aa   3dd03bba2c00         "docker-entrypoint.s…"   13 minutes ago   Exited (0) 7 minutes ago
8b9db08058ef   aekarattakhat/lab8   "/bin/sh -c 'echo \"ι…"   33 minutes ago   Exited (0) 33 minutes ago
ccf6194b71ec   lab8_2               "/bin/sh -c 'echo \"ι…"   2 hours ago      Exited (0) 2 hours ago
79a5a2a9fa91   lab8_2               "/bin/sh -c 'echo \"ι…"   2 hours ago      Exited (0) 2 hours ago
cf702b2fbd2f   a5f120a1587f         "/bin/sh -c 'echo “ι…"   2 hours ago      Exited (0) 2 hours ago
cde9e33434c6   busybox              "sh"                     5 days ago       Exited (0) 5 days ago
048cd4b7b6f9   busybox              "sh"                     5 days ago       Exited (0) 5 days ago
PS D:\coding\se\lab8_4\getting-started\app> docker rm fb7ae31700aa
fb7ae31700aa
PS D:\coding\se\lab8_4\getting-started\app>
```

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

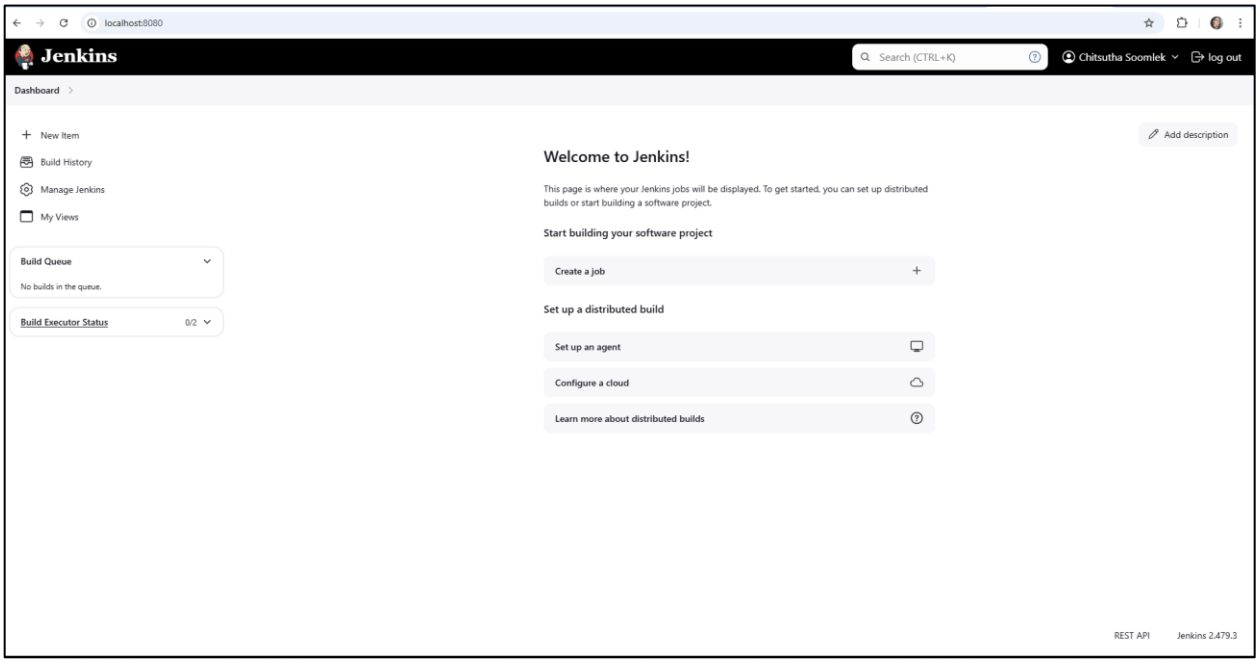
[Check point#12] Capture หน้าจอที่แสดงผล Admin password

```
************************************************************
************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

90e77ae1695a49fdbba9c545d9ece5c1

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

************************************************************
************************************************************
************************************************************
```

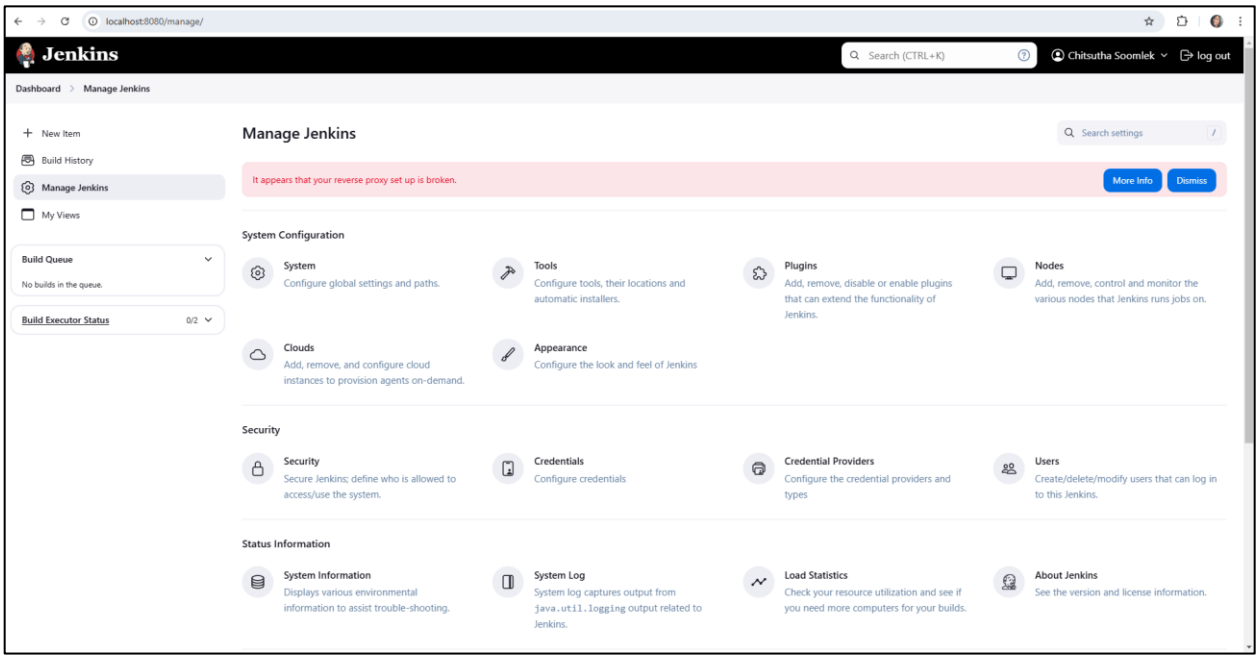
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

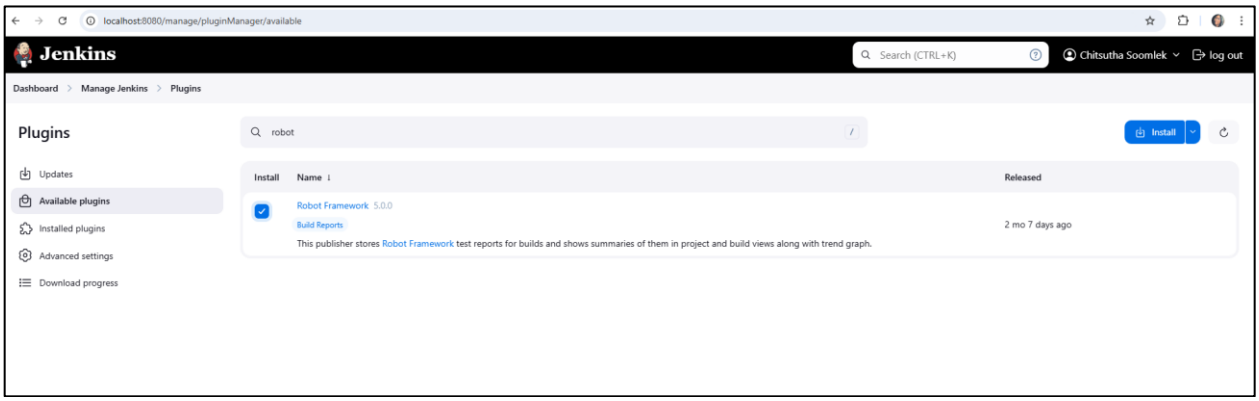
7. กำหนด Jenkins URL เป็น http://localhost:8080/lab8
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ



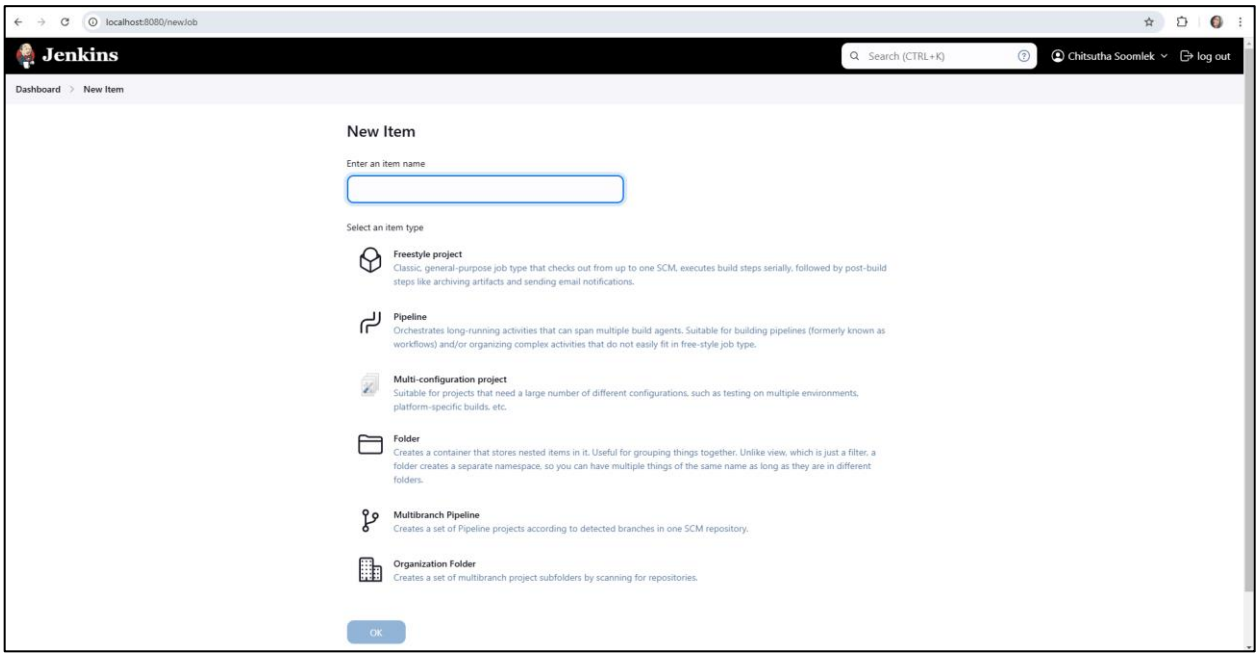
9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



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

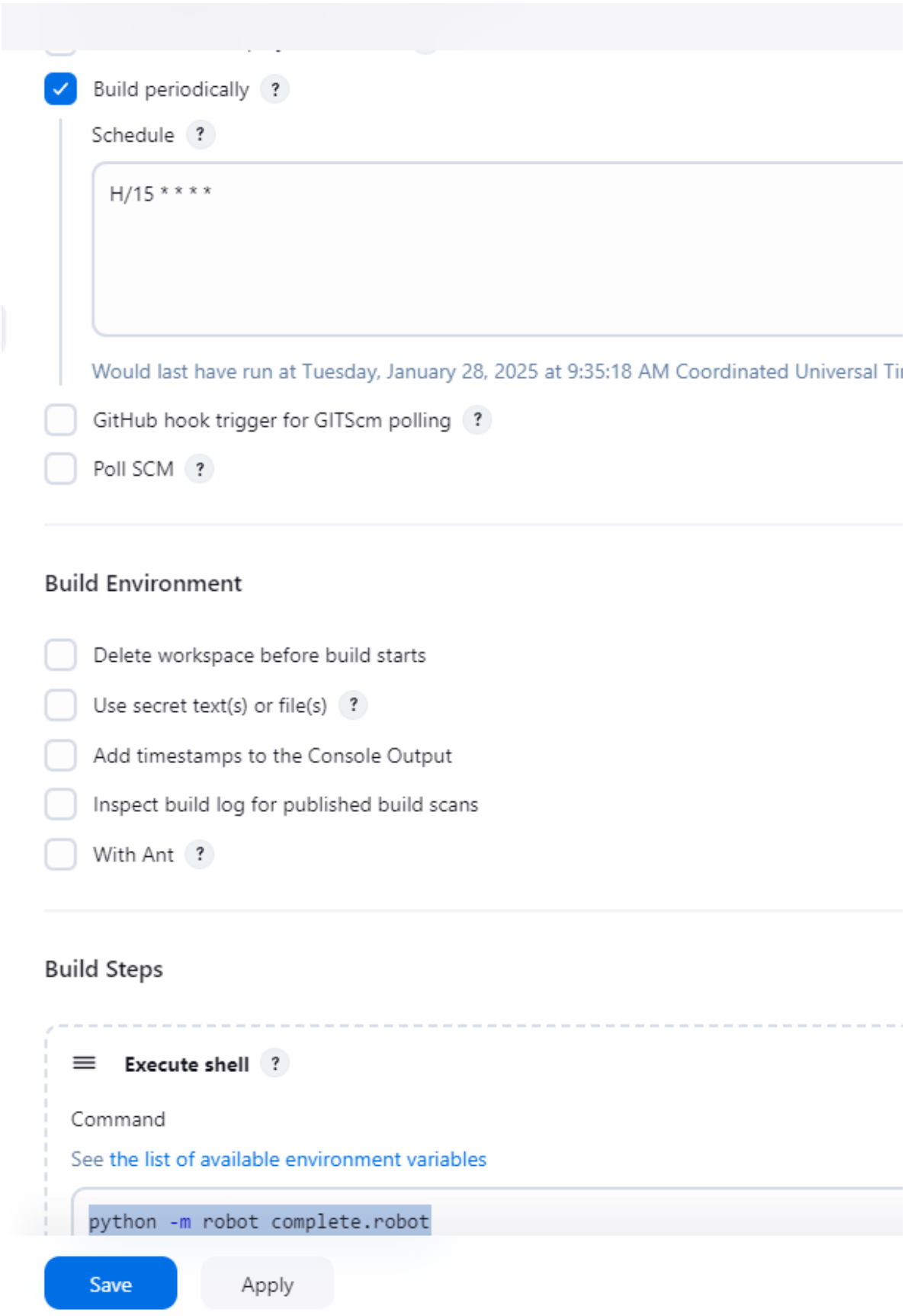
Description: Lab 8.5

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


Build periodically
Schedule
H/15 * * * *
Would last have run at Tuesday, January 28, 2025 at 9:35:18 AM Coordinated Universal Ti
GitHub hook trigger for GITScm polling
Poll SCM
Build Environment
Delete workspace before build starts
Use secret text(s) or file(s)
Add timestamps to the Console Output
Inspect build log for published build scans
With Ant
Build Steps
Execute shell
Command
See the list of available environment variables
python -m robot complete.robot
Save
Apply

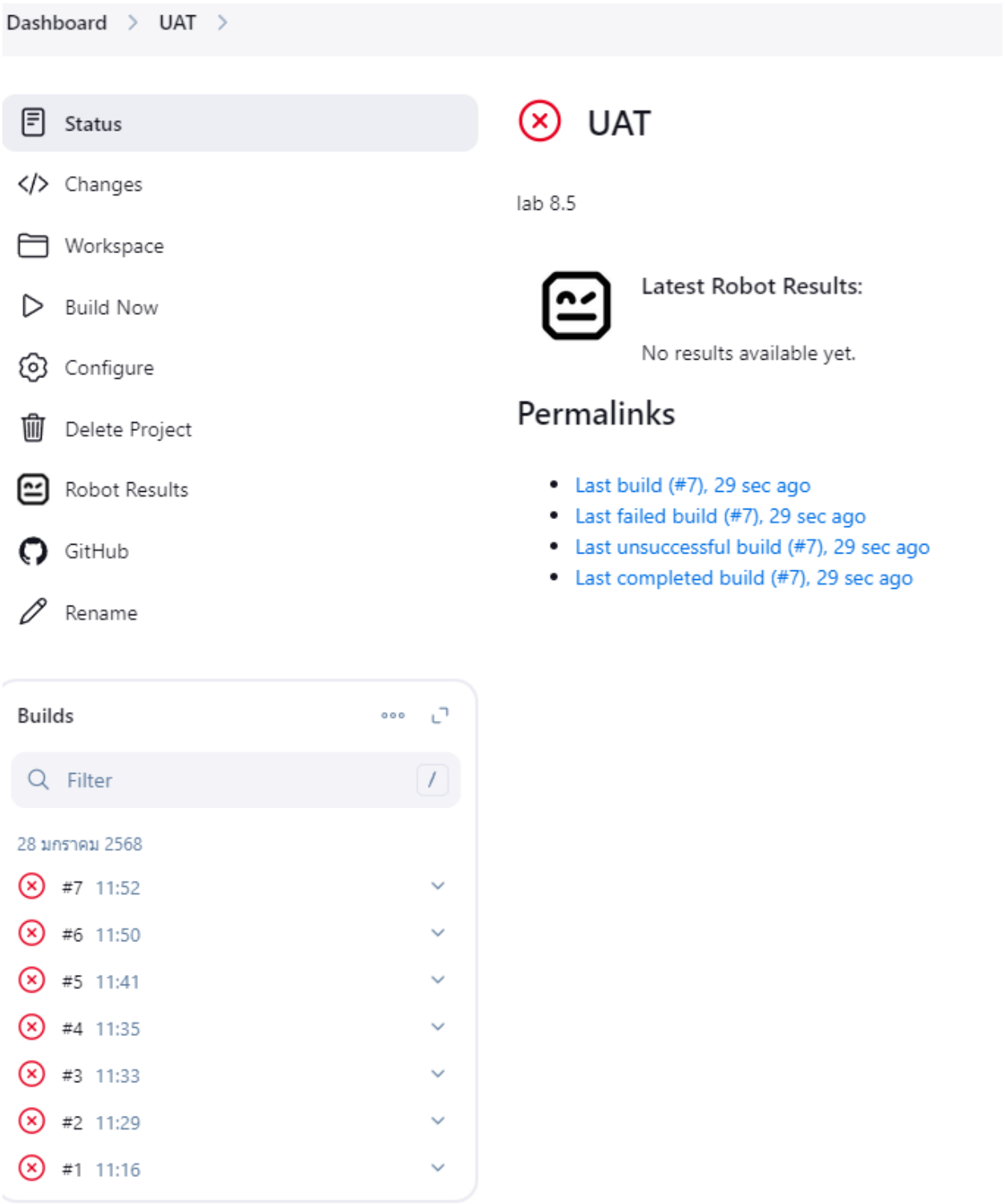
(1) คำสั่งที่ใช้ในการ Execute ไฟล์ .robot ใน Build Steps คือ

robot complete.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


Dashboard > UAT >
Status
Changes
Workspace
Build Now
Configure
Delete Project
Robot Results
GitHub
Rename
Builds
Filter /
28 มกราคม 2568
#7 11:52
#6 11:50
#5 11:41
#4 11:35
#3 11:33
#2 11:29
#1 11:16
UAT
lab 8.5
Latest Robot Results:
No results available yet.
Permalinks
Last build (#7), 29 sec ago
Last failed build (#7), 29 sec ago
Last unsuccessful build (#7), 29 sec ago
Last completed build (#7), 29 sec ago

Console Output

```
Started by timer
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
[UAT] $ /bin/sh -xe /tmp/jenkins12909361431540196975.sh
+ robot complete.robot
/tmp/jenkins12909361431540196975.sh: 2: robot: not found
Build step 'Execute shell' marked build as failure
Robot results publisher started...
INFO: Checking test criticality is deprecated and will be dropped in a future release!
-Parsing output xml:
Failed!
hudson.AbortException: No files found in path /var/jenkins_home/workspace/UAT with configured filemask: output.xml
        at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:81)
        at PluginClassLoader for robot//hudson.plugins.robot.RobotParser$RobotParserCallable.invoke(RobotParser.java:52)
        at hudson.FilePath.act(FilePath.java:1234)
        at hudson.FilePath.act(FilePath.java:1217)
        at PluginClassLoader for robot//hudson.plugins.robot.RobotParser.parse(RobotParser.java:48)
        at PluginClassLoader for robot//hudson.plugins.robot.RobotPublisher.parse(RobotPublisher.java:262)
        at PluginClassLoader for robot//hudson.plugins.robot.RobotPublisher.perform(RobotPublisher.java:286)
        at hudson.tasks.BuildStepCompatibilityLayer.perform(BuildStepCompatibilityLayer.java:80)
        at hudson.tasks.BuildStepMonitor$1.perform(BuildStepMonitor.java:20)
        at hudson.model.AbstractBuild$AbstractBuildExecution.perform(AbstractBuild.java:818)
        at hudson.model.AbstractBuild$AbstractBuildExecution.performAllBuildSteps(AbstractBuild.java:767)
        at hudson.model.Build$BuildExecution.post2(Build.java:179)
        at hudson.model.AbstractBuild$AbstractBuildExecution.post(AbstractBuild.java:711)
        at hudson.model.Run.execute(Run.java:1854)
        at hudson.model.FreeStyleBuild.run(FreeStyleBuild.java:44)
        at hudson.model.ResourceController.execute(ResourceController.java:101)
        at hudson.model.Executor.run(Executor.java:445)
Finished: FAILURE
```