Lab Worksheet

ชื่อ-นามสกุล นางสาวกรกนก พฤทธิพันธุ์ รหัสนักศึกษา 653380187-0 Section 3

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 C:\Lab8.1_6533801870> docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
my-app        latest    9656beb2174b   8 weeks ago      2.11GB
<none>        <none>    cf8ad210b881   8 weeks ago      2.07GB
<none>        <none>    d9c80a304561   8 weeks ago      2.07GB
<none>        <none>    5285d1c7c61c   8 weeks ago      1.89GB
ai-final      latest    a0708c9f306c   3 months ago     7.64GB
nginx         latest    60c8a892f36f   3 months ago     192MB
busybox       latest    af4709625109   3 months ago     4.27MB
hello-world   latest    d2c94e258dcb   21 months ago    13.3kB
PS C:\Lab8.1_6533801870>
```

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร ที่เก็บ image ของ Docker

(2) Tag ที่ใช้บ่งบอกถึงอะไร บอกเวอร์ชันหรือลักษณะเฉพาะของ Docker image ที่อยู่ใน repository เดียวกัน

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 C:\Lab8.1_6533801870> docker run busybox
PS C:\Lab8.1_6533801870> docker run -it busybox sh
/ # ls
bin    etc    lib    proc   sys    usr
dev    home   lib64  root   tmp    var
/ # ls -la
total 48
drwxr-xr-x    1 root     root          4096 Jan 23 01:43 .
drwxr-xr-x    1 root     root          4096 Jan 23 01:43 ..
-rwxr-xr-x    1 root     root             0 Jan 23 01:43 .dockerenv
drwxr-xr-x    2 root     root         12288 Sep 26 21:31 bin
drwxr-xr-x    5 root     root           360 Jan 23 01:43 dev
drwxr-xr-x    1 root     root          4096 Jan 23 01:43 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  348 root     root             0 Jan 23 01:43 proc
drwx------    1 root     root          4096 Jan 23 01:43 root
dr-xr-xr-x   11 root     root             0 Jan 23 01:43 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 C:\Lab8.1_6533801870> docker run busybox echo "Hello Konkanok Pruttipan from busybox"
Hello Konkanok Pruttipan from busybox
PS C:\Lab8.1_6533801870> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED              STATUS                          PORTS     NAMES
10a2bda5b9e7   busybox   "echo 'Hello Konkano…"   7 seconds ago        Exited (0) 7 seconds ago                  crazy_borg
df6edfcc8fdd   busybox   "sh"                     About a minute ago   Exited (0) 59 seconds ago                 affectionate_khayyam
4306bb0f2b39   busybox   "sh"                     About a minute ago   Exited (0) About a minute ago             kind_dhawan
PS C:\Lab8.1_6533801870> docker rm 4306bb0f2b39
4306bb0f2b39
PS C:\Lab8.1_6533801870> docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS                      PORTS     NAMES
10a2bda5b9e7   busybox   "echo 'Hello Konkano…"   11 minutes ago   Exited (0) 11 minutes ago             crazy_borg
df6edfcc8fdd   busybox   "sh"                     12 minutes ago   Exited (0) 11 minutes ago             affectionate_khayyam
PS C:\Lab8.1_6533801870>
```

(1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป

ทำให้สามารถเข้าไปใน container และใช้งาน shell เพื่อ interact กับ container ได้ เปิดโหมด interactive (ด้วย -i) และเชื่อมต่อกับ terminal (ด้วย -t)

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

สถานะของ Docker container ในปัจจุบัน โดยจะบอกว่า container นั้นๆ กำลังทำงานอยู่หรือถูกหยุด (stopped) หรือมีสถานะอื่นๆ

12. ป้อนคำสั่ง $ docker rm <container ID ที่ต้องการลบ>

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

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

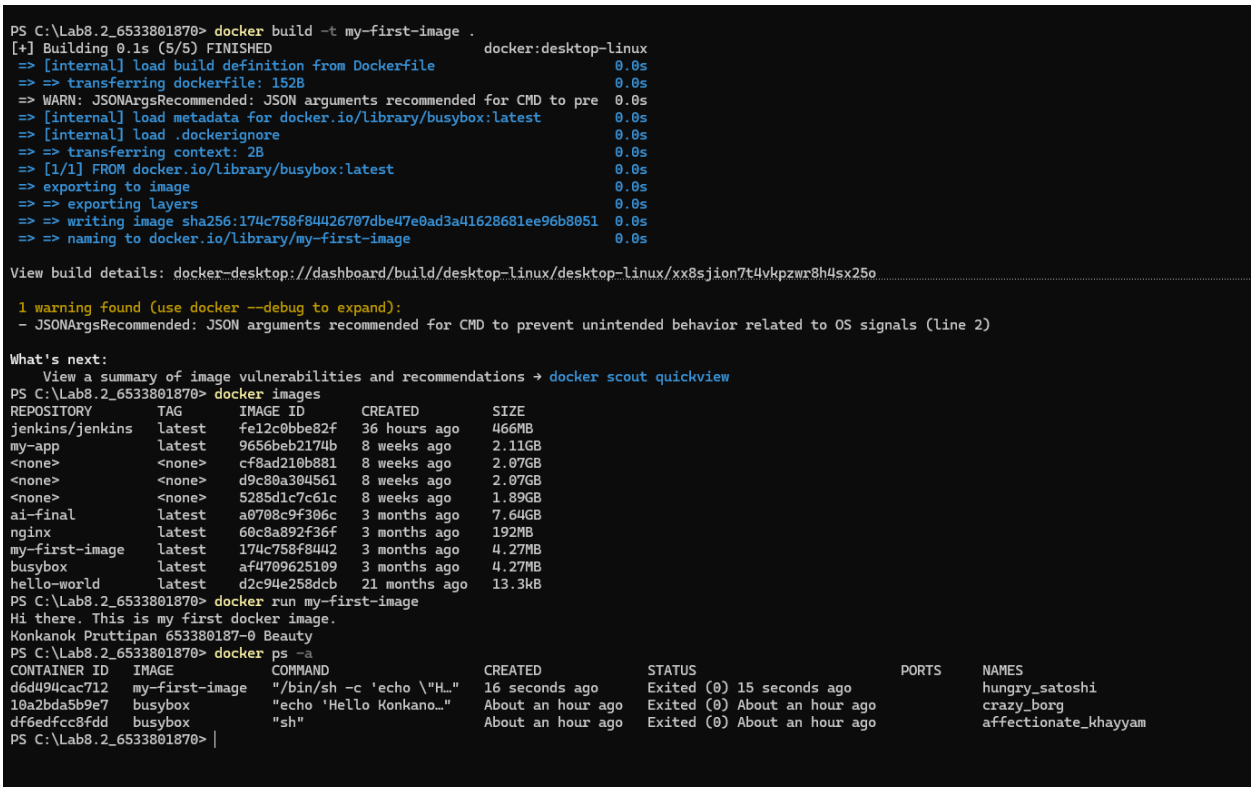
แล้วใช้ Text Editor ในการใส่เนื้อหาแทน

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:\Lab8.2_6533801870> docker build -t my-first-image .
[+] Building 0.1s (5/5) FINISHED                          docker:desktop-linux
 => [internal] load build definition from Dockerfile                      0.0s
 => => transferring dockerfile: 152B                                      0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to pre  0.0s
 => [internal] load metadata for docker.io/library/busybox:latest         0.0s
 => [internal] load .dockerignore                                         0.0s
 => => transferring context: 2B                                           0.0s
 => [1/1] FROM docker.io/library/busybox:latest                           0.0s
 => exporting to image                                                    0.0s
 => => exporting layers                                                   0.0s
 => => writing image sha256:174c758f84426707dbe47e0ad3a41628681ee96b8051  0.0s
 => => naming to docker.io/library/my-first-image                         0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/xx8sjion7t4vkpzwr8h4sx25o

 1 warning found (use docker --debug to expand):
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Lab8.2_6533801870> docker images
REPOSITORY        TAG       IMAGE ID       CREATED         SIZE
jenkins/jenkins   latest    fe12c0bbe82f   36 hours ago    466MB
my-app            latest    9656beb2174b   8 weeks ago     2.11GB
<none>            <none>    cf8ad210b881   8 weeks ago     2.07GB
<none>            <none>    d9c80a304561   8 weeks ago     2.07GB
<none>            <none>    5285d1c7c61c   8 weeks ago     1.89GB
ai-final          latest    a0708c9f306c   3 months ago    7.64GB
nginx             latest    60c8a892f36f   3 months ago    192MB
my-first-image    latest    174c758f8442   3 months ago    4.27MB
busybox           latest    af4709625109   3 months ago    4.27MB
hello-world       latest    d2c94e258dcb   21 months ago   13.3kB
PS C:\Lab8.2_6533801870> docker run my-first-image
Hi there. This is my first docker image.
Konkanok Pruttipan 653380187-0 Beauty
PS C:\Lab8.2_6533801870> docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED             STATUS                         PORTS     NAMES
d6d494cac712   my-first-image   "/bin/sh -c 'echo \"H…"   16 seconds ago      Exited (0) 15 seconds ago                hungry_satoshi
10a2bda5b9e7   busybox          "echo 'Hello Konkano…"   About an hour ago   Exited (0) About an hour ago             crazy_borg
df6edfcc8fdd   busybox          "sh"                     About an hour ago   Exited (0) About an hour ago             affectionate_khayyam
PS C:\Lab8.2_6533801870>
```

(1) คำสั่งที่ใช้ในการ run คือ

Docker run ตามด้วยชื่อ Image

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

-t ในคำสั่ง docker build ช่วยกำหนดชื่อและ tag ให้กับ Docker image ที่จะถูกสร้างขึ้น

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 C:\lab8.3> docker build -t kkukonkanok/lab8.3 .
[+] Building 0.1s (5/5) FINISHED                                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                0.0s
 => => transferring dockerfile: 179B                                                                                                0.0s
 => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals              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 sha256:3e8f6418b8655951e2db90393ac7b8c26ca87058671cd3133c7ea570dd82f927                                        0.0s
 => => naming to docker.io/kkukonkanok/lab8.3                                                                                       0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/fm42obrytiicrr18i9p2mvsrt

 1 warning found (use docker --debug to expand):
 - JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 2)

What's next:
    View a summary of image vulnerabilities and recommendations →docker scout quickview
PS C:\lab8.3> docker run kkukonkanok/lab8.3
Hi there. My work is done. You can run them from my Docker image.
Konkanok Pruttipan 653380187-0 Beauty
PS C:\lab8.3>
```

6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยการใช้คำสั่ง

   $ docker push <username ที่ลงทะเบียนกับ Docker Hub>/lab8

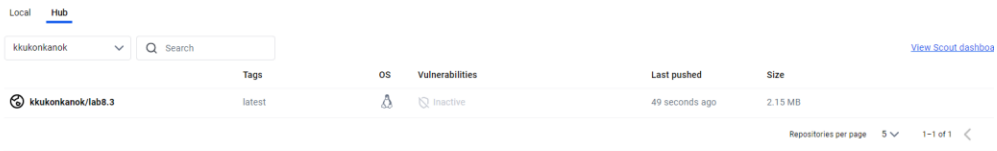
   ในกรณีที่ติดปัญหาไม่ได้ Login ไว้ก่อน ให้ใช้คำสั่งต่อไปนี้ เพื่อ Login ก่อนทำการ Push

   $ 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)

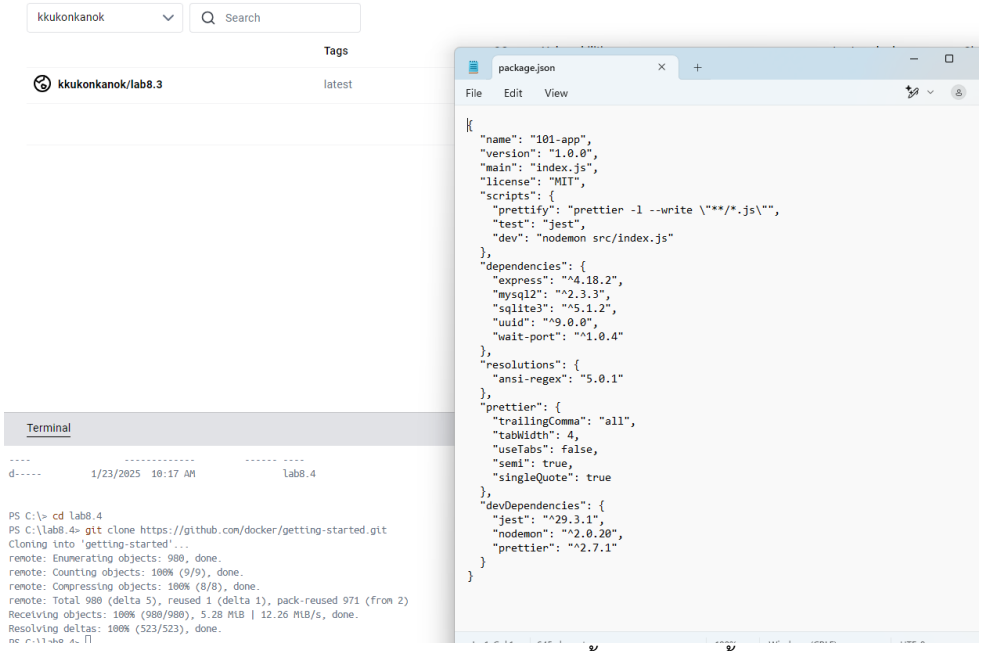


แบบฝึกปฏิบัติที่ 8.4: การ Build แอปพลิเคชันจาก Container image และการ Update แอปพลิเคชัน

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

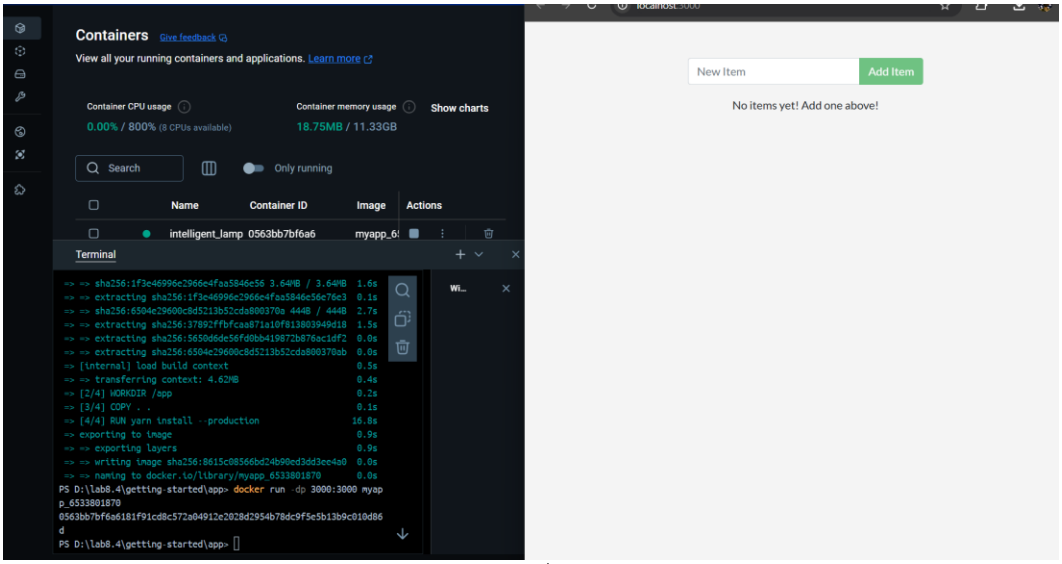
```
PS C:\lab8.4\getting-started\app> docker build -t myapp_6533801870:sec3 .
[+] Building 14.1s (10/10) FINISHED                                                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 156B                                                                                0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                                                   3.2s
 => [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@sha256:77e3b76b47148e28acc84f2e903f34bedc21385b3644c9967aa25b324bb0e6b4  2.2s
 => => resolve docker.io/library/node:18-alpine@sha256:77e3b76b47148e28acc84f2e903f34bedc21385b3644c9967aa25b324bb0e6b4  0.0s
 => => sha256:77e3b76b47148e28acc84f2e903f34bedc21385b3644c9967aa25b324bb0e6b4 7.67kB / 7.67kB                      0.0s
 => => sha256:6e804119c3884fc5782795bf0d2adc89201c63105aece8647b17a7bcebbc385e 1.72kB / 1.72kB                      0.0s
 => => sha256:dcbf7b337595be6f4d214e4eed84f230eefe0e4ac03a50380d573e289b9e5e40 6.18kB / 6.18kB                      0.0s
 => => sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0 3.64MB / 3.64MB                      0.4s
 => => sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c 40.01MB / 40.01MB                    1.4s
 => => sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1 1.26MB / 1.26MB                      1.0s
 => => extracting sha256:1f3e46996e2966e4faa5846e56e76e3748b7315e2ded61476c24403d592134f0                           0.1s
 => => sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771 444B / 444B                          0.9s
 => => extracting sha256:37892ffbfcaa871a10f813803949d18c3015a482051d51b7e0da02525e63167c                           0.7s
 => => extracting sha256:5650d6de56fd0bb419872b876ac1df28f577b39573c3b72fb0d15bf426d01bc1                           0.0s
 => => extracting sha256:6504e29600c8d5213b52cda800370abb3d12639802d06b46b6fce368990ca771                           0.0s
 => [internal] load build context                                                                                   0.3s
 => => exporting layers                                                                                             0.5s
 => => writing image sha256:2169cc66fb44c173a28468f9634e05f514fa9d23023157f039b82a594590c969                        0.0s
 => => naming to docker.io/library/myapp_6533801870:sec3                                                            0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/lnqf28dwld14ujbiwzqzvtjal

What's next:
    View a summary of image vulnerabilities and recommendations →docker scout quickview
PS C:\lab8.4\getting-started\app>
```

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

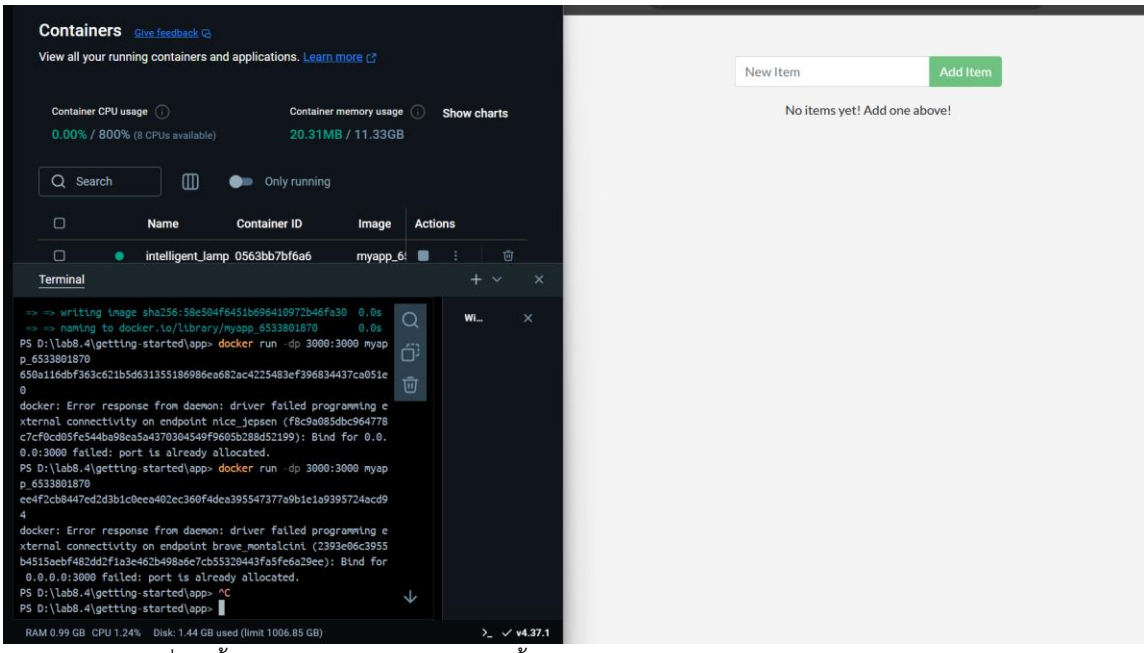
   No items yet! Add one above! เป็น

   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

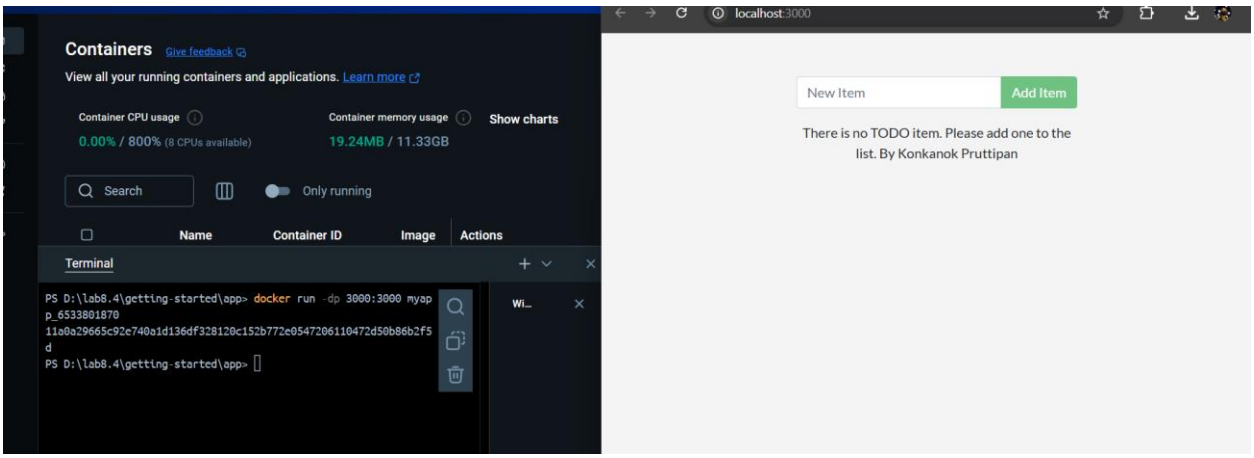
พอร์ต 3000 มีการใช้งานอยู่แล้วผ่าน Container อื่น ทำให้ Container ใหม่เชื่อมพอร์ตไม่สำเร็จ

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



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

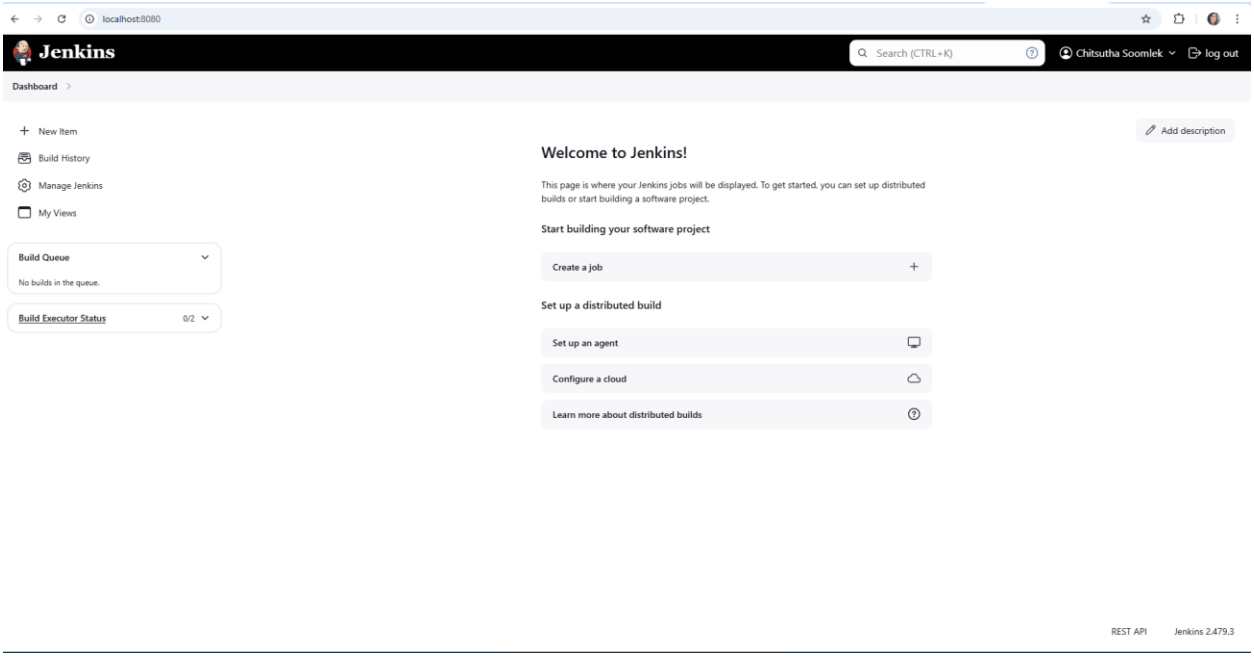
   $ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v jenkins_home:/var/jenkins_home jenkins/jenkins:lts-jdk17
3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

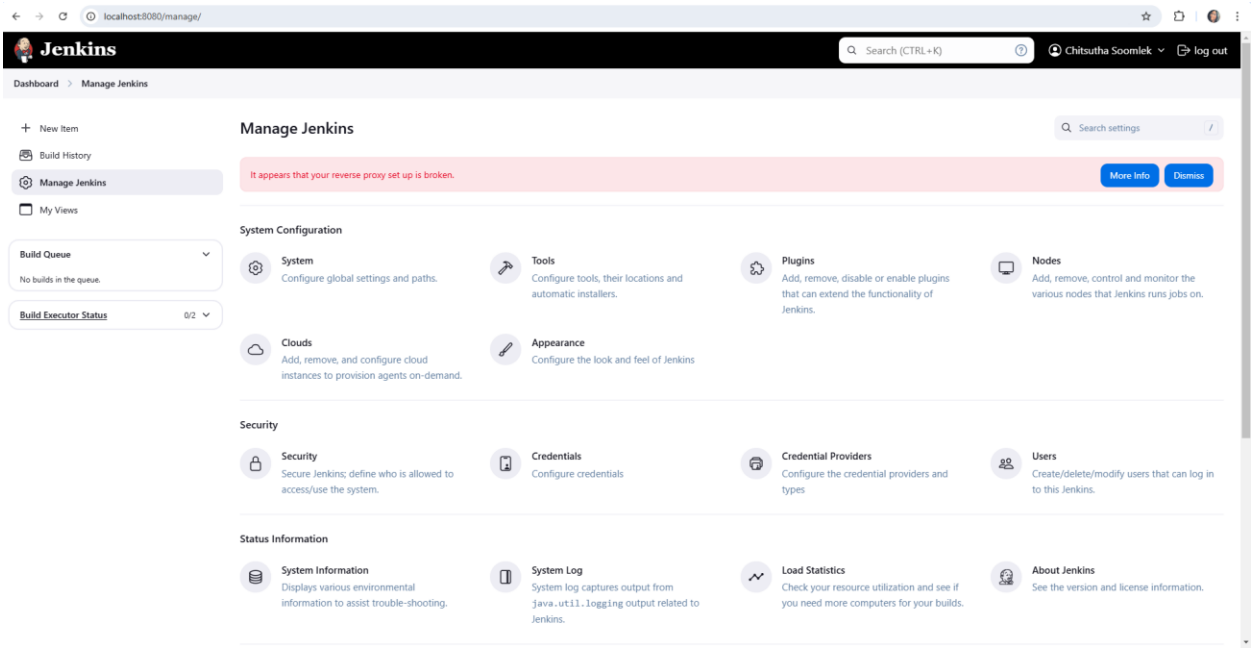
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดบราวเซอร์ และป้อนที่อยู่เป็น localhost:8080
5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

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

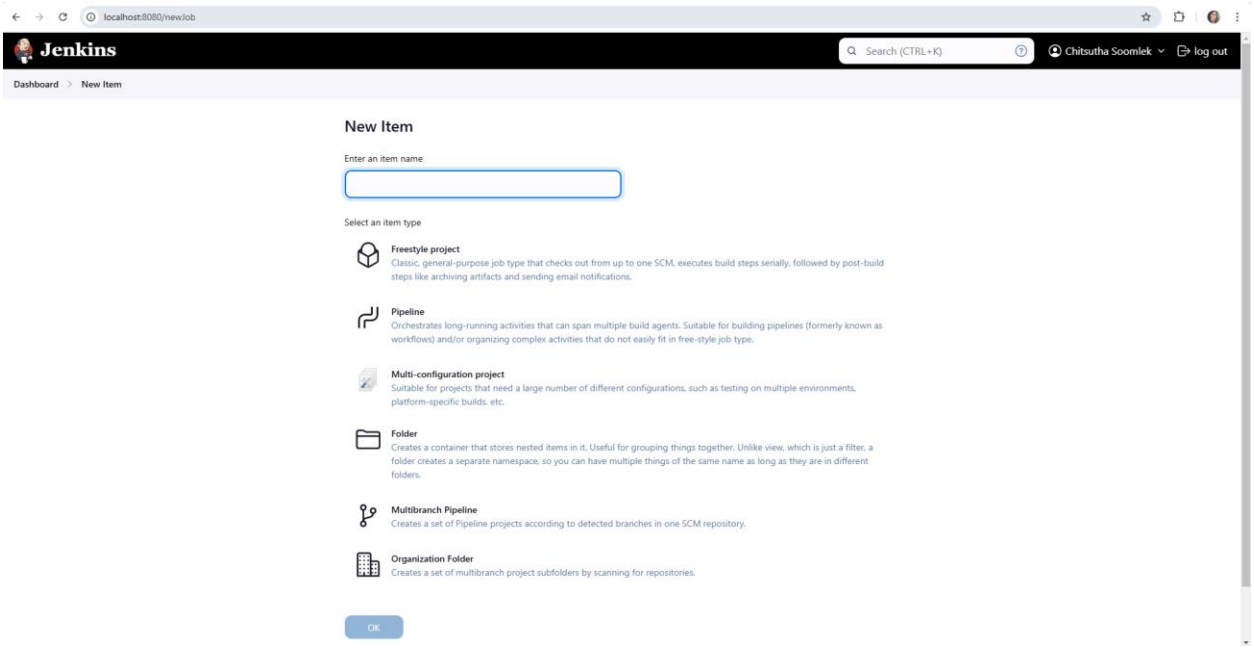


9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



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

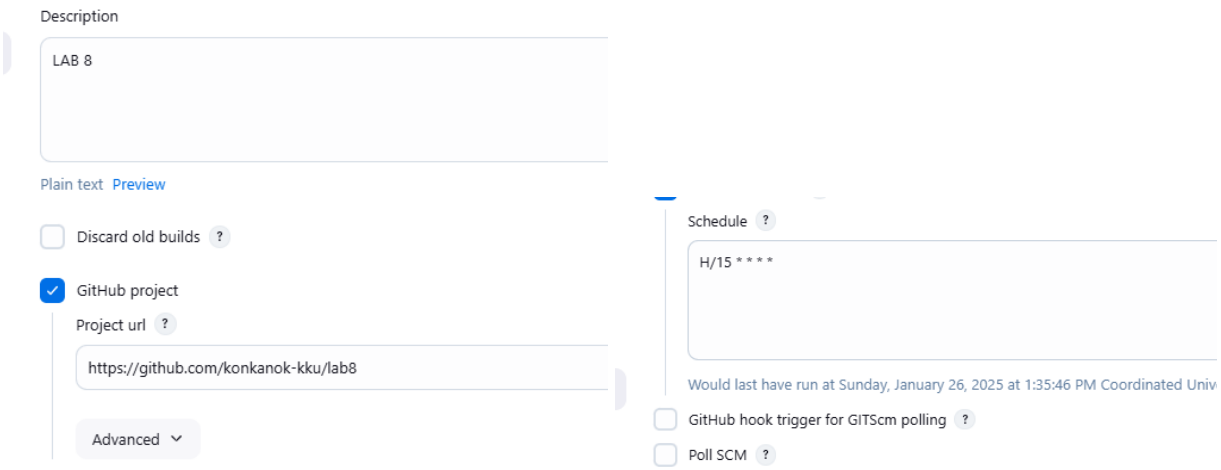
Description: Lab 8.5

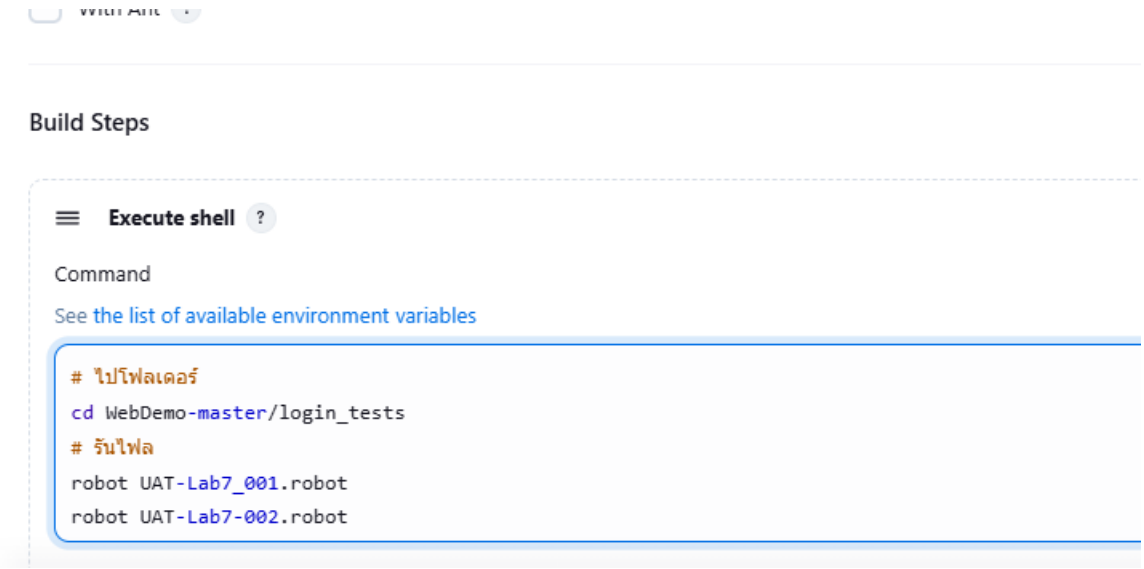
GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

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





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

ไปยังโฟลเดอร์

cd WebDemo-master/login_tests

รันไฟล์

robot UAT-Lab7_001.robot

robot UAT-Lab7_002.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

Lab Worksheet