

Lab Worksheet

ชื่อ-นามสกุล สร้อยทองแท้ อ้วนอก รหัสนักศึกษา 653380346-6 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

Lab Worksheet

```

Terminal
What's next:
  View a summary of image vulnerabilities and recommendations →docker scout quickview busybox
PS C:\Lab8_1> docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
ubuntu              latest          bid9df8ab815    2 months ago   78.1MB
busybox             latest          af4709625109    3 months ago   4.27MB
busybox             <none>          27a71e19c956    3 months ago   4.27MB
mysql               latest          c757d623b190    6 months ago   586MB
kormart/tf-jupyterlab latest          555931e56c99    3 years ago    1.58GB
PS C:\Lab8_1> docker run busybox

```

- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร คือชื่อ images ที่ติดตั้งไว้ทั้งหมด
- (2) Tag ที่ใช้บ่งบอกถึงอะไร ใช้บ่งบอกเวอร์ชัน หรือ ลักษณะเฉพาะ ของ Docker image เพราะอาจมี Docker image ที่ชื่อเหมือนกันเยอะ

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> docker run busybox echo "HELLO myname Sroithongthae Aunok from busybox"
HELLO myname Sroithongthae Aunok from busybox
PS C:\Lab8_1> docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS              PORTS          NAMES
c207e62175d9   busybox    "echo 'HELLO myname ..." 19 seconds ago Exited (0) 18 seconds ago           gracious_zhukovsky
458071be6eb9   busybox    "sh"                    2 minutes ago Exited (0) About a minute ago         sharp_pasteur
ebdc0e582bbd   busybox    "sh"                    2 minutes ago Exited (0) 2 minutes ago           nostalgic_payne
42e805f3d00f   ubuntu    "bash"                  11 minutes ago Exited (0) 10 minutes ago           quizzical_aryabhata
5a7aceb15b73   27a71e19c956 "sh"                    3 months ago Exited (0) 3 months ago           youthful_williamson
32477668c2f1   27a71e19c956 "sh"                    3 months ago Exited (0) 3 months ago           romantic_herschel

```

- (1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
- ช่วยให้การทำงานของ Docker container สามารถโต้ตอบกับผู้ใช้ได้ เหมาะสำหรับการพัฒนา การทดสอบ หรือการทำงานในโหมด interactive ที่ต้องการเข้าถึง terminal ภายใน container โดยตรง

Lab Worksheet

(2) คอลัมน์ STATUS จากการรันคำสั่ง `docker ps -a` แสดงถึงข้อมูลอะไร

แสดงถึง สถานะการทำงานของ container ในขณะนั้น โดยแสดงข้อมูลเกี่ยวกับการทำงานหรือสถานะล่าสุดของ container แต่ละตัว ซึ่งสามารถแสดงได้หลายรูปแบบ ขึ้นอยู่กับสถานะของ container นั้นๆ

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

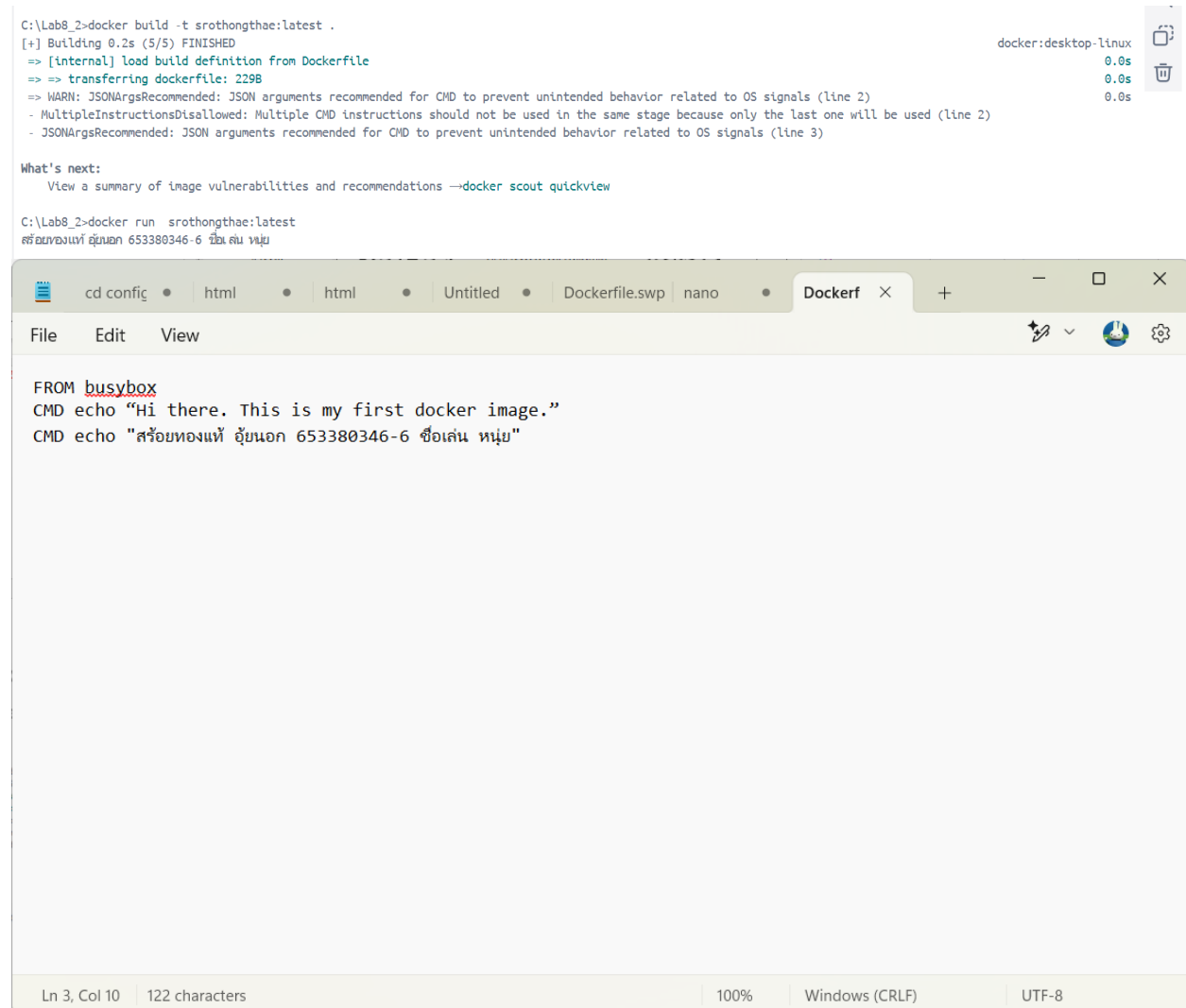
5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้

```
$ docker build -t <ชื่อ Image> .
```

6. เมื่อ Build สำเร็จแล้ว ให้ทำการรัน Docker image ที่สร้างขึ้นในขั้นตอนที่ 5

Lab Worksheet

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



The screenshot shows a terminal window at the top and a code editor (VS Code) at the bottom. The terminal window displays the output of the following commands:

```
C:\Lab8_2>docker build -t srothongthae:latest .
[+] Building 0.2s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 229B
=> WARN: 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 one will be used (line 2)
- JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavior related to OS signals (line 3)

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

C:\Lab8_2>docker run srothongthae:latest
สวัสดีของแท้ อู๋นอก 653380346-6 ชื่อเล่น หนุ่ย
```

The code editor shows the Dockerfile content:

```
FROM busybox
CMD echo "Hi there. This is my first docker image."
CMD echo "สวัสดีของแท้ อู๋นอก 653380346-6 ชื่อเล่น หนุ่ย"
```

(1) คำสั่งที่ใช้ในการ run คือ

`docker run srothongthae:latest`

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

ใช้สำหรับ กำหนด tag ให้กับ Docker image ที่ถูกสร้างขึ้นจาก Dockerfile ที่กำหนดไว้ โดย -t ทำให้เราสามารถตั้งชื่อและระบุเวอร์ชัน (หรือ tag) สำหรับ Docker image ที่สร้างขึ้น

Lab Worksheet

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

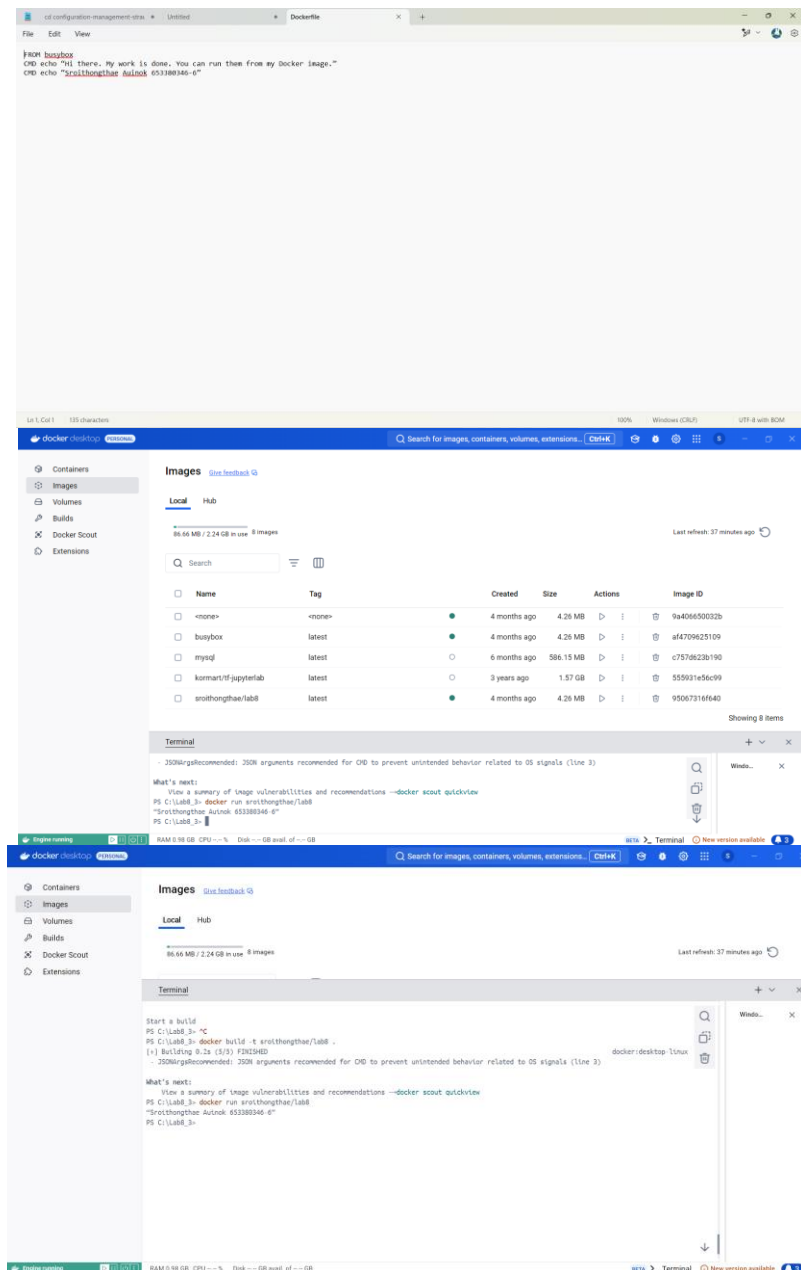
```
$ docker build -t <username ที่ลงทะเบียนกับ Docker Hub>/lab8
```

5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง

```
$ docker run <username ที่ลงทะเบียนกับ Docker Hub>/lab8
```

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

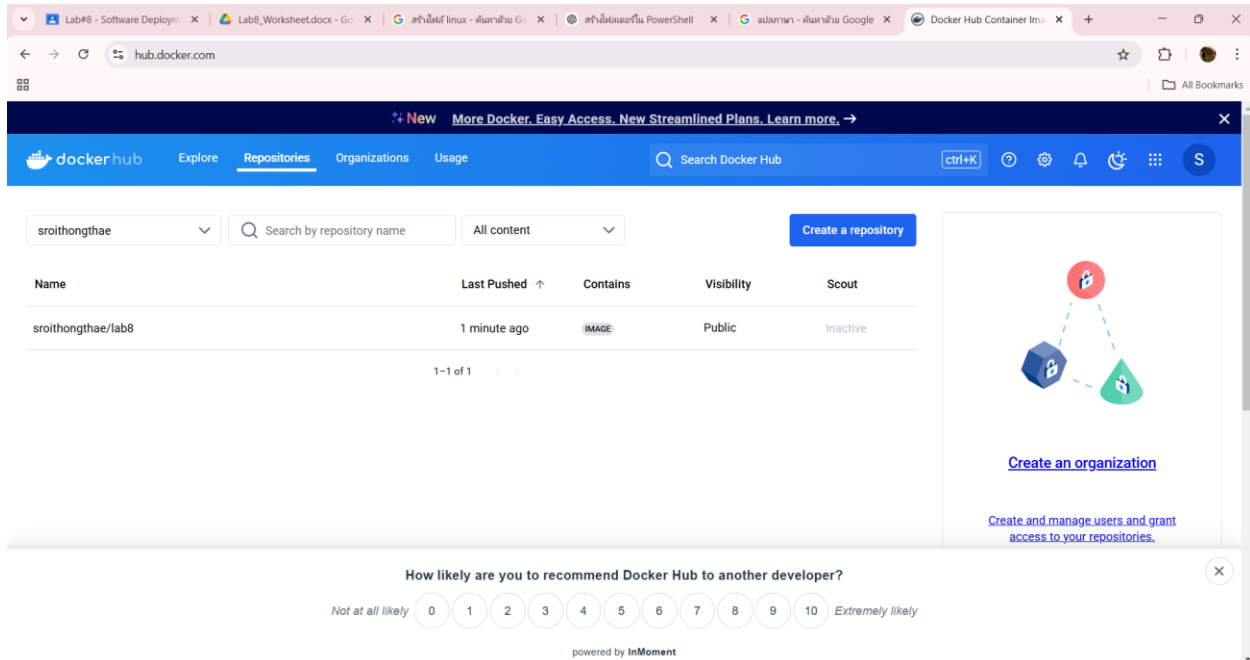
Lab Worksheet



6. ทำการ Push ตัว Docker image ไปไว้บน Docker Hub โดยใช้คำสั่ง
`$ 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

Lab Worksheet

[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

Lab Worksheet

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

The screenshot shows a terminal window and a file explorer. The terminal window displays the following commands and output:

```

59654b79daad: Mounted from library/busybox
latest: digest: sha256:755b83b863acd674bfff18033b5b755ebe4235428db842d56785510248ced8b size: 527
PS C:\Lab8_3> ^C
PS C:\Lab8_3> cd ..
PS C:\> mkdir Lab8_4

Directory: C:\

Mode                LastWriteTime         Length Name
----                -
d-----            1/29/2025   7:52 PM          Lab8_4

PS C:\> cd Lab8_4
PS C:\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 | 661.00 KiB/s, done.
Resolving deltas: 100% (523/523), done.
PS C:\Lab8_4>

```

The file explorer shows the directory structure of the cloned repository:

Name	Date modified	Type	Size
spec	29/1/2568 19:54	File folder	
src	29/1/2568 19:54	File folder	
package	29/1/2568 19:54	JSON Source File	1 KB
yarn.lock	29/1/2568 19:54	LOCK File	148 KB

The package.json file content is shown below:

```

{
  "name": "101-app",
  "version": "1.0.0",
  "main": "index.js",
  "license": "MIT",
  "scripts": {
    "prettify": "prettier -l --write \"**/*.js\"",
    "test": "jest",
    "dev": "nodemon src/index.js"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mysql2": "^2.3.3",
    "sqlite3": "^5.1.2",
    "uuid": "^9.0.0",
    "wait-port": "^1.0.4"
  },
  "resolutions": {
    "ansi-regex": "5.0.1"
  },
  "prettier": {
    "trailingComma": "all",
    "tabWidth": 4,
    "useTabs": false,
    "semi": true,
    "singleQuote": true
  },
  "devDependencies": {
    "jest": "^29.3.1",
    "nodemon": "^2.0.20",
    "prettier": "^2.7.1"
  }
}

```


Lab Worksheet

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไปไฟล์

```
FROM node:18-alpine
```

```
WORKDIR /app
```

```
COPY . .
```

```
RUN yarn install --production
```

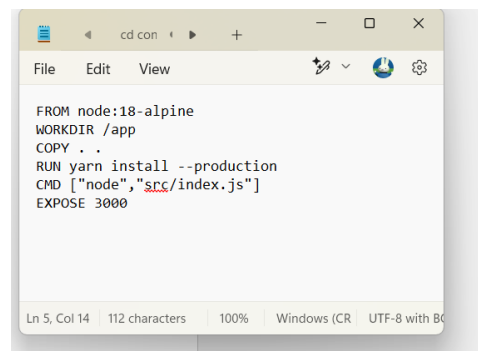
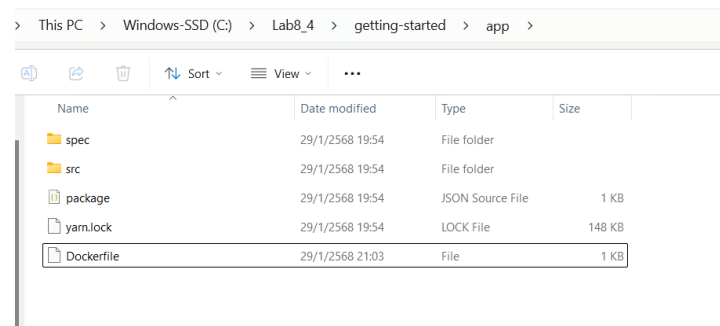
```
CMD ["node", "src/index.js"]
```

```
EXPOSE 3000
```

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดใช้ชื่อ image เป็น myapp_รหัสสนศ. ไม่มีขีด

```
$ docker build -t <myapp_รหัสสนศ. ไม่มีขีด> .
```

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



Lab Worksheet

```

PS C:\Lab8_4\getting-started\app> docker build -t myapp_6533803466 .
[+] Building 1.1s (9/9) FINISHED
=> [internal] Load build definition from Dockerfile
=> => transferring dockerfile: 158B
=> [internal] Load metadata for docker.io/library/node:18-alpine
=> [internal] Load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25
=> [internal] Load build context
=> => transferring context: 2.62kB
=> CACHED [2/4] WORKDIR /app
=> CACHED [3/4] COPY . .
=> CACHED [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> writing image sha256:dee6679ba3f668bbba13ecd2283ea0299c30be201a84f5d545654d2d1466f68
=> => naming to docker.io/library/myapp_6533803466

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/19zytmk1tsu1uy86mg6uq2dxo

```

6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

\$ docker run -dp 3000:3000 <myapp_รหัสสนศ. ไม่มีขีด>

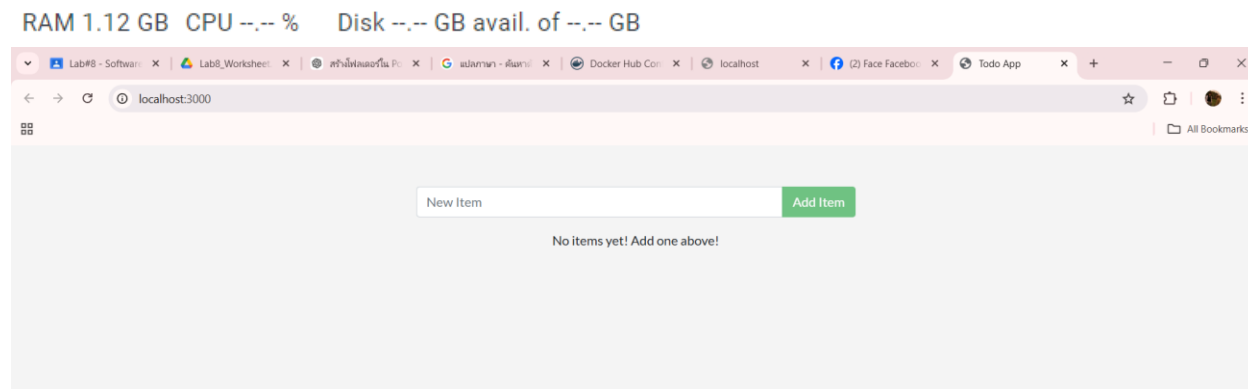
7. เปิด Browser ไปที่ URL = <http://localhost:3000>

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

```

PS C:\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533803466
07e5bb1068c169e46af6a8175972c10aab806d652182dc5c1acf3993dc05242b
PS C:\Lab8_4\getting-started\app>

```



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

<p className="text-center">No items yet! Add one above!</p> เป็น

<p className="text-center">**There is no TODO item. Please add one to the list.**

By ชื่อและนามสกุลของนักศึกษา</p>

b. Save ไฟล์ให้เรียบร้อย

Lab Worksheet

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

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

```
PS C:\Lab8_4\getting-started\app> docker build -t myapp_6533803466 .
[+] Building 2.5s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 158B
/!ew build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/ct5bwt50jbm5k1x735i70x68m
```

that's next:

```
View a summary of image vulnerabilities and recommendations →docker scout quickview
PS C:\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533803466
304b506143fc2bc7abf9fcd96546cf064bf1d930c6cf97165ad21246cb1d9bb0f
docker: Error response from daemon: driver failed programming external connectivity on endpoint focused_varahamihira (1cab3d995425eb61ebb35f2219
ic4a556522b6b4a58df6f63e1c829823c7178a): Bind for 0.0.0.0:3000 failed: port is already allocated.
PS C:\Lab8_4\getting-started\app> ^C
```

```
return (
  <React.Fragment>
    <AddItemForm onNewItem={onNewItem} />
    {items.length === 0 && (
      <p className="text-center">There is no TODO item. Please add one to the list. By สร้อยทองแท้ อุ่นนอก 653380346-6</p>
    )}
    {items.map(item => (
      <ItemDisplay
        item={item}
        key={item.id}
        onItemUpdate={onItemUpdate}
        onItemRemoval={onItemRemoval}
      />
    ))}
  </React.Fragment>
);
```

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

ตอบ หมายความว่า port 3000 มีการใช้งานอยู่ในขณะนี้ ทำให้ไม่สามารถเข้าใช้งานซ้ำได้

11. ลบ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้วิธีใดวิธีหนึ่งดังต่อไปนี้

a. ผ่าน Command line interface

- ใช้คำสั่ง `$ docker ps` เพื่อดู Container ID ที่ต้องการจะลบ
- Copy หรือบันทึก Container ID ไว้
- ใช้คำสั่ง `$ docker stop <Container ID ที่ต้องการจะลบ>` เพื่อหยุดการทำงานของ Container ดังกล่าว
- ใช้คำสั่ง `$ docker rm <Container ID ที่ต้องการจะลบ>` เพื่อทำการลบ

b. ผ่าน Docker desktop

- ไปที่หน้าต่าง Containers

Lab Worksheet

- 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

The screenshot shows a Windows PowerShell terminal window and the Docker Desktop application interface.

Windows PowerShell Terminal:

```

PS C:\Users\Lenovo> cd ..
PS C:\Users> cd ..
PS C:\> cd Lab8_4
PS C:\Lab8_4> cd getting-started
PS C:\Lab8_4\getting-started> cd app
PS C:\Lab8_4\getting-started\app> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
07e5bb1068c1   dee6679ba3f6   "docker-entrypoint.s..." 20 minutes ago Up 20 minutes   0.0.0.0:3000->3000/tcp   silly_b
ose
PS C:\Lab8_4\getting-started\app> docker stop 07e5bb1068c1
07e5bb1068c1
PS C:\Lab8_4\getting-started\app> docker rm 07e5bb1068c1
07e5bb1068c1
PS C:\Lab8_4\getting-started\app> |
  
```

Docker Desktop Interface:

- Images Tab:** Shows a list of local images. The table below summarizes the data shown in the interface.

Name	Tag	Created	Size	Actions	Image ID
<none>	<none>	1 hour ago	142.08 MB	[Play] [Stop] [Delete]	a95e44a4dd57
<none>	<none>	54 minutes ago	142.08 MB	[Play] [Stop] [Delete]	e71ae4d2df34
<none>	<none>	30 minutes ago	216.9 MB	[Play] [Stop] [Delete]	dee6679ba3f6
myapp_6533803466	latest	16 minutes ago	216.9 MB	[Play] [Stop] [Delete]	e5b83b94db7b

Terminal Window (Docker Desktop):

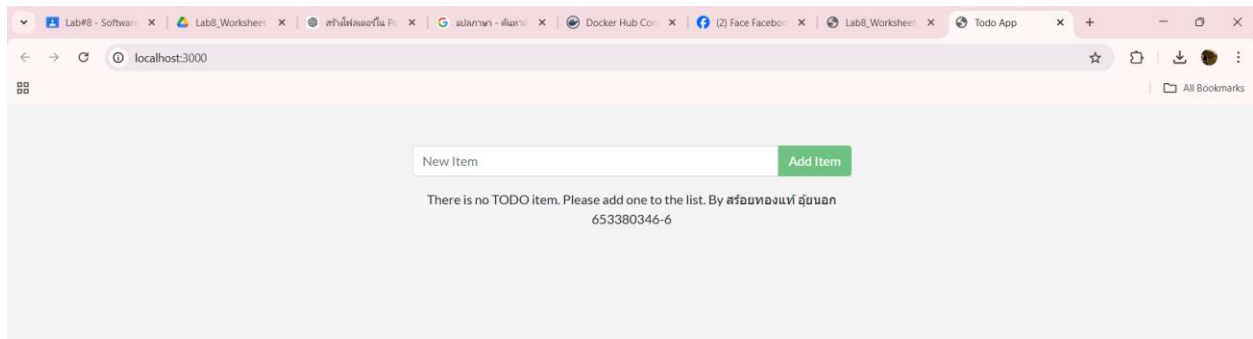
```

View a summary of image vulnerabilities and recommendations --docker scout quickview
PS C:\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533803466
804b506143fc2bc7abf9fcd96546cf064bf1d930c6cf97165ad21246cb1dbb0f
docker: Error response from daemon: driver failed programming external connectivity on endpoint focused_varahamihira (1cab3d995425eb61ebb35f22195c4a556522b6b4a58df6f63e1c829823c7178a): Bind for 0.0.0.0:3000 failed: port is already allocated.
PS C:\Lab8_4\getting-started\app> ^C
PS C:\Lab8_4\getting-started\app> docker run -dp 3000:3000 myapp_6533803466
dcb0e915e2a89f3d8ba5e70ad1eb75590967c49b884dce7d64b84f1c17259281
PS C:\Lab8_4\getting-started\app> |
  
```

System Status Bar:

- Engine running
- RAM 1.23 GB CPU -- % Disk -- GB avail. of -- GB
- BETA Terminal New version available

Lab Worksheet



แบบฝึกปฏิบัติที่ 8.5: เริ่มต้นสร้าง Pipeline อย่างง่ายสำหรับการ Deploy ด้วย Jenkins

1. เปิด Command line หรือ Terminal บน Docker Desktop

2. ป้อนคำสั่งและทำการรัน container โดยผูกพอร์ต

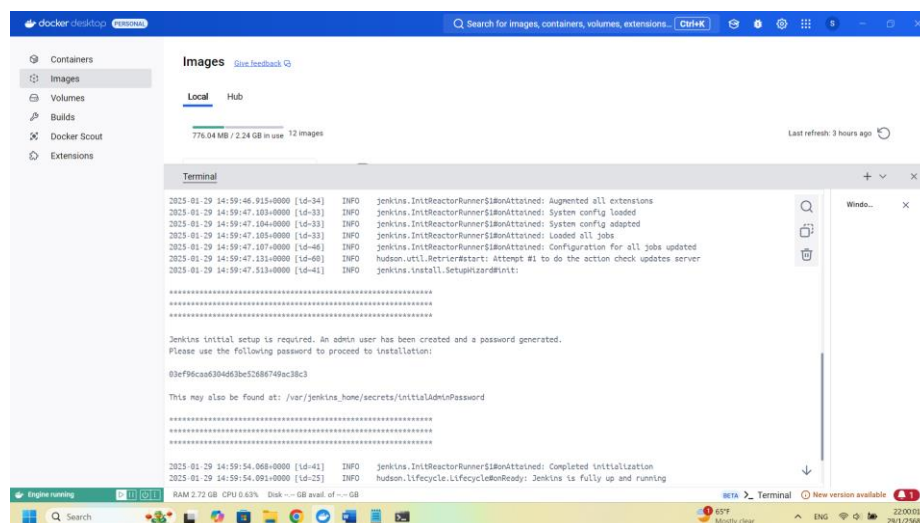
```
$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure jenkins/jenkins:ls-jdk17
```

หรือ

```
$ docker run -p 8080:8080 -p 50000:50000 --restart=on-failure -v  
jenkins_home:/var/jenkins_home jenkins/jenkins:ls-jdk17
```

3. บันทึกรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[Check point#12] Capture หน้าจอที่แสดงผล Admin password



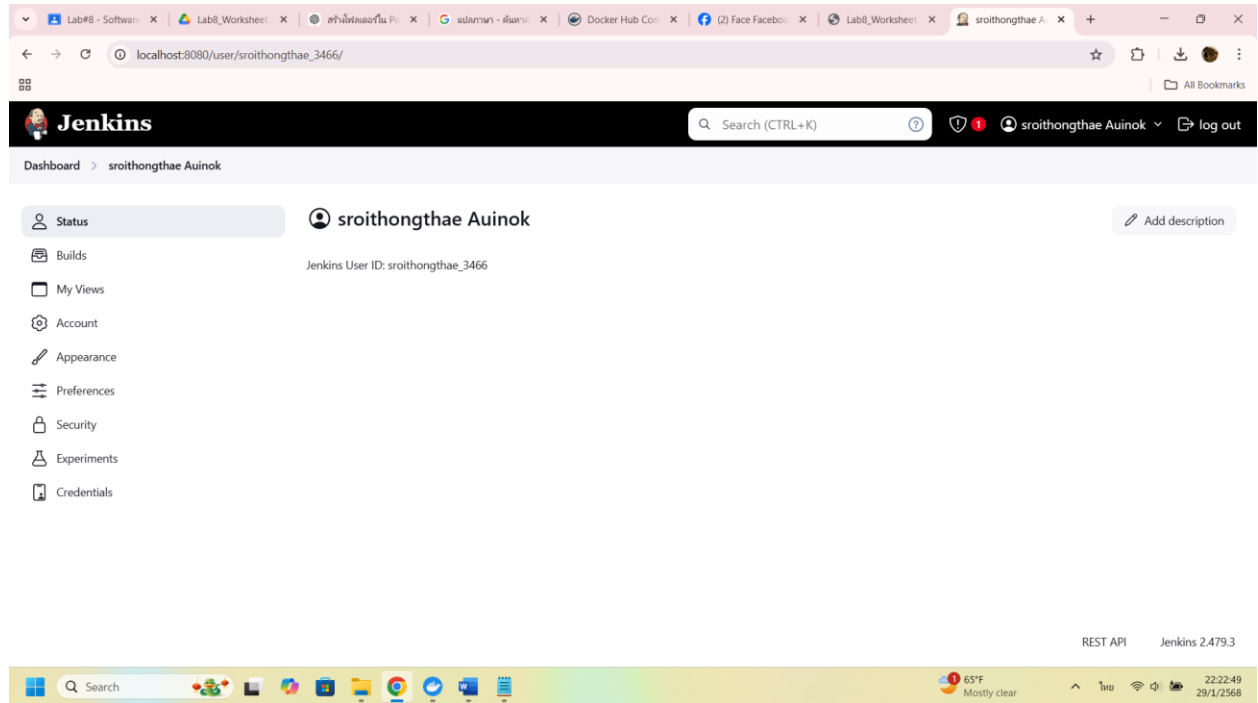
4. เมื่อได้รับการยืนยันว่า Jenkins is fully up and running ให้เปิดเบราว์เซอร์ และป้อนที่อยู่เป็น localhost:8080

5. ทำการ Unlock Jenkins ด้วยรหัสผ่านที่ได้ในข้อที่ 3

Lab Worksheet

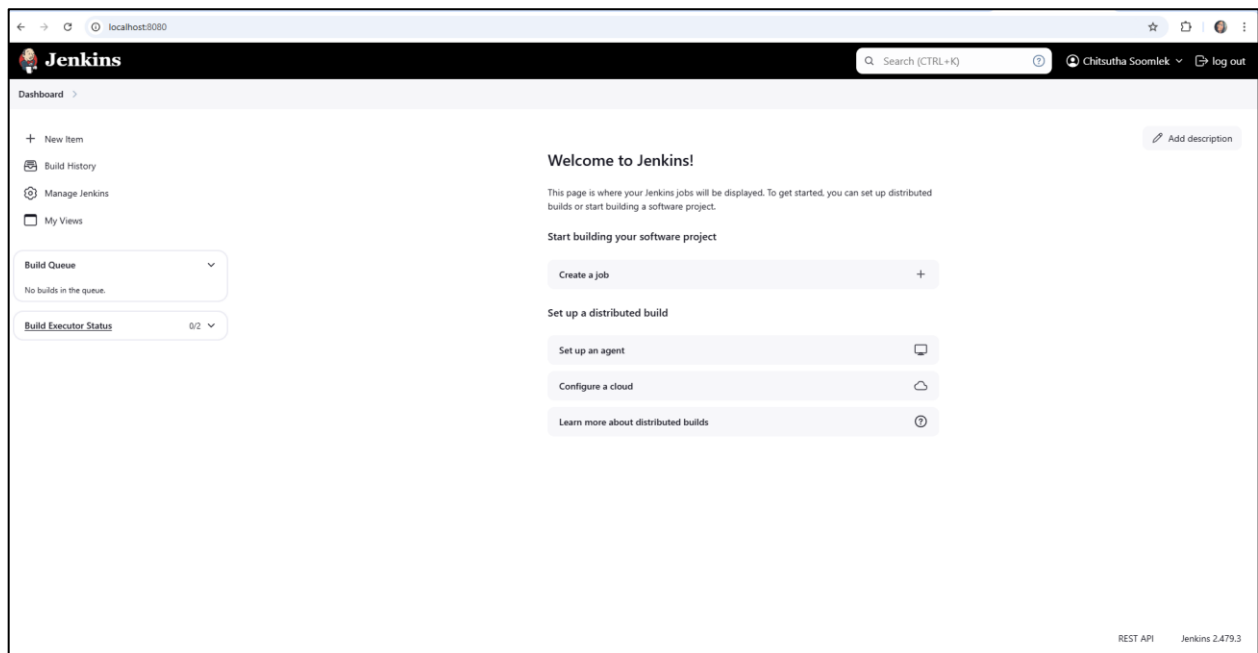
6. สร้าง Admin User โดยใช้ username เป็นชื่อจริงของนักศึกษาพร้อมรหัสสี่ตัวท้าย เช่น somsri_3062

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า



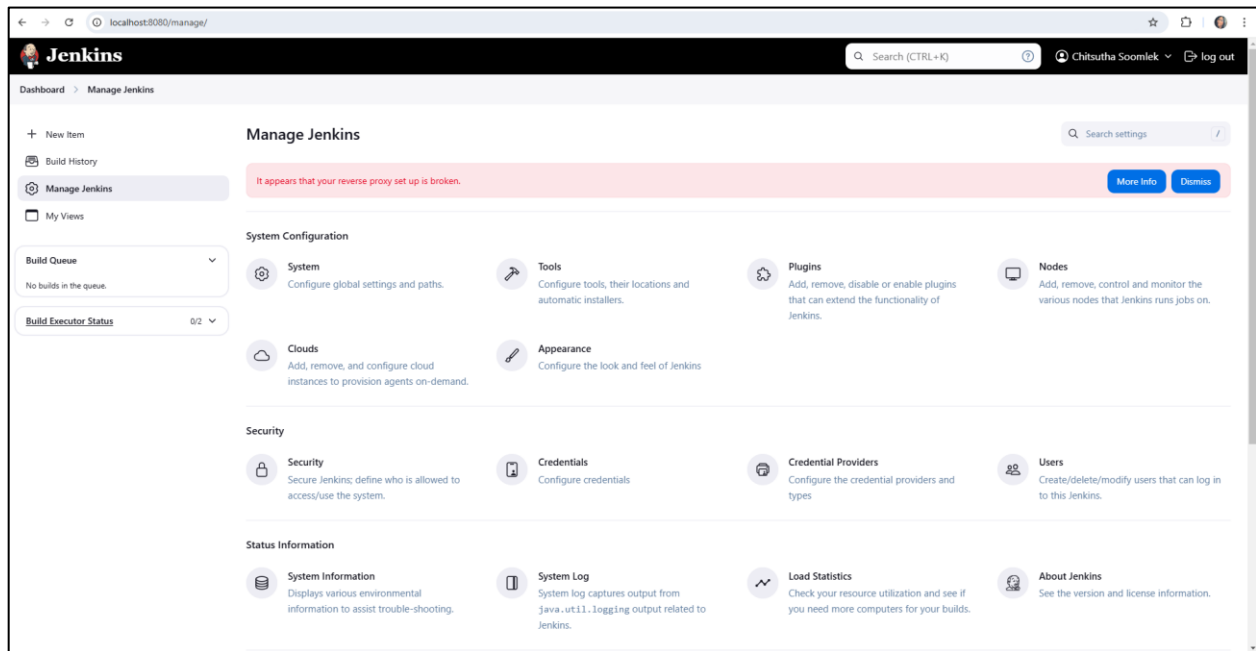
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>

8. เมื่อติดตั้งเรียบร้อยแล้วจะพบกันหน้า Dashboard ดังแสดงในภาพ

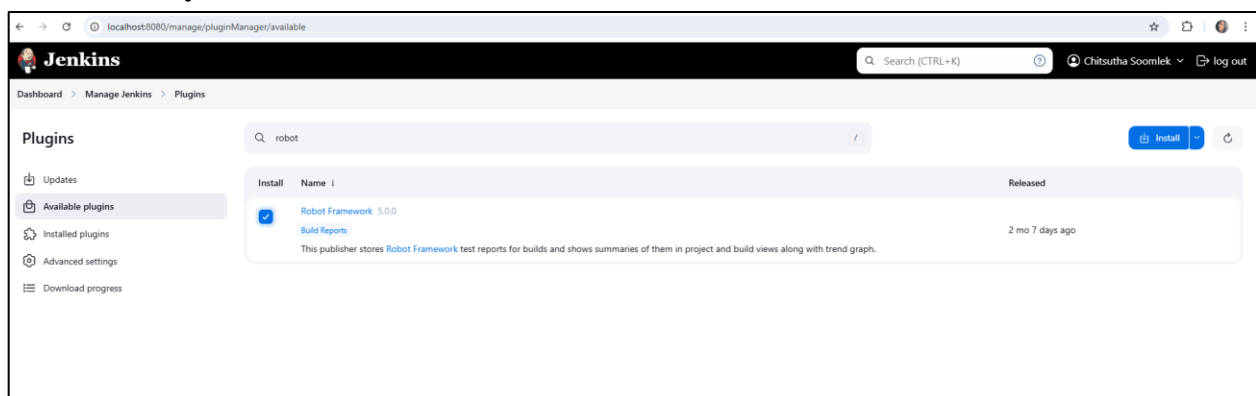


Lab Worksheet

9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins

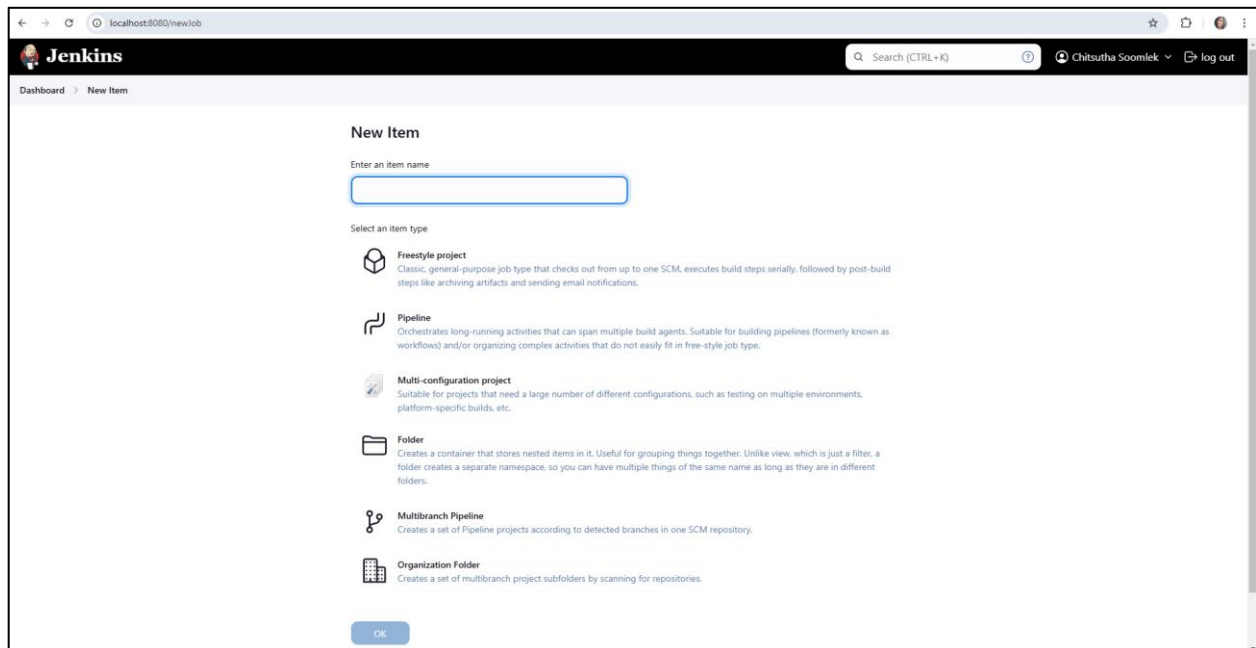


10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม



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

Lab Worksheet



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 ไปถึงไฟล์ให้เรียบร้อยแล้ว)

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้

Lab Worksheet

The image displays two screenshots of the Jenkins web interface, showing the configuration page for a job named 'Lab8.5'.

Top Screenshot: General Tab

- Configuration:** The 'General' tab is selected in the left sidebar.
- Description:** The job is named 'Lab8.5'.
- Plain text:** A 'Preview' link is visible.
- Discard old builds:** An unchecked checkbox.
- GitHub project:** This option is selected. The 'Project url' is set to 'https://github.com/sroithongthae346-6/Lab8/'.
- Advanced:** A dropdown menu is shown.
- Other options:** 'This project is parameterized', 'Throttle builds', and 'Execute concurrent builds if necessary' are all unchecked.
- Buttons:** 'Save' and 'Apply' buttons are at the bottom.

Bottom Screenshot: Source Code Management Tab

- Configuration:** The 'Source Code Management' tab is selected in the left sidebar.
- Source Code Management:** The 'Git' option is selected.
- Repositories:** A list of repositories is shown. The first repository has the URL 'https://github.com/sroithongthae346-6/Lab8.git' and 'Credentials' set to 'none'.
- Advanced:** A dropdown menu is shown.
- Add Repository:** A button to add a new repository.
- Branches to build:** A section for specifying branches. The 'Branch Specifier (blank for 'any')' is set to '*/main'.
- Buttons:** 'Save' and 'Apply' buttons are at the bottom.

Lab Worksheet

The screenshot displays the Jenkins configuration interface for a job named 'UAT'. The interface is divided into two main sections: 'Build Triggers' and 'Build Environment'.

Build Triggers:

- ☐ Trigger builds remotely (e.g., from scripts) ?
- ☐ Build after other projects are built ?
- ☒ Build periodically ?
- Schedule ?
H/15 * * * *
Would last have run at Wednesday, January 29, 2025 at 4:35:52 PM Coordinated Universal Time; would next run at Wednesday, January 29, 2025 at 4:50:52 PM Coordinated Universal Time.
- ☐ 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
- ☐ Terminate a build if it's stuck

Build Steps:

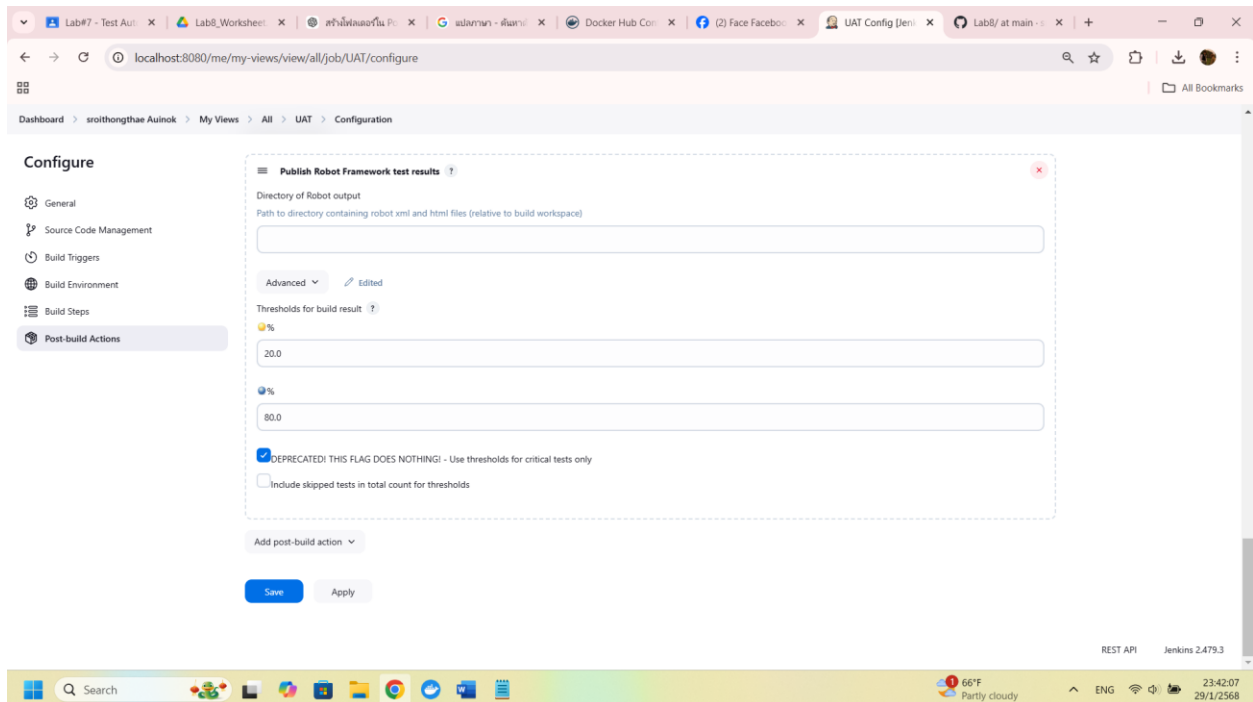
- Execute shell ?**
Command
See the list of available environment variables
Is -al
robot valid_login.robot

Post-build Actions:

- Publish Robot Framework test results ?**
Directory of Robot output
Path to directory containing robot.xml and html files (relative to build workspace)

The interface includes a sidebar with navigation links: General, Source Code Management, Build Triggers (selected), Build Environment, Build Steps, and Post-build Actions. The bottom of the page shows a Windows taskbar with various application icons and system information.

Lab Worksheet



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

ls -al

robot valid_login.robot

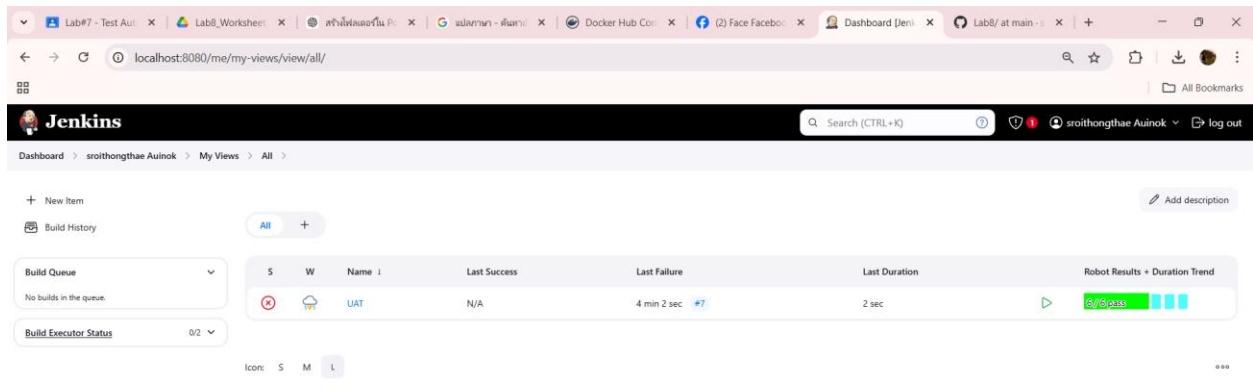
Post-build action: เพิ่ม Publish Robot Framework test results -> ระบุไดเรกทอรีที่เก็บไฟล์ผลการทดสอบโดย Robot framework ในรูป xml และ html -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ไม่ผ่าน แล้วนับว่าซอฟต์แวร์มีปัญหา -> ตั้งค่า Threshold เป็น % ของการทดสอบที่ผ่านแล้วนับว่าซอฟต์แวร์มีอยู่ในสถานะที่สามารถนำไปใช้งานได้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

13. กด Apply และ Save

14. สั่ง Build Now

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

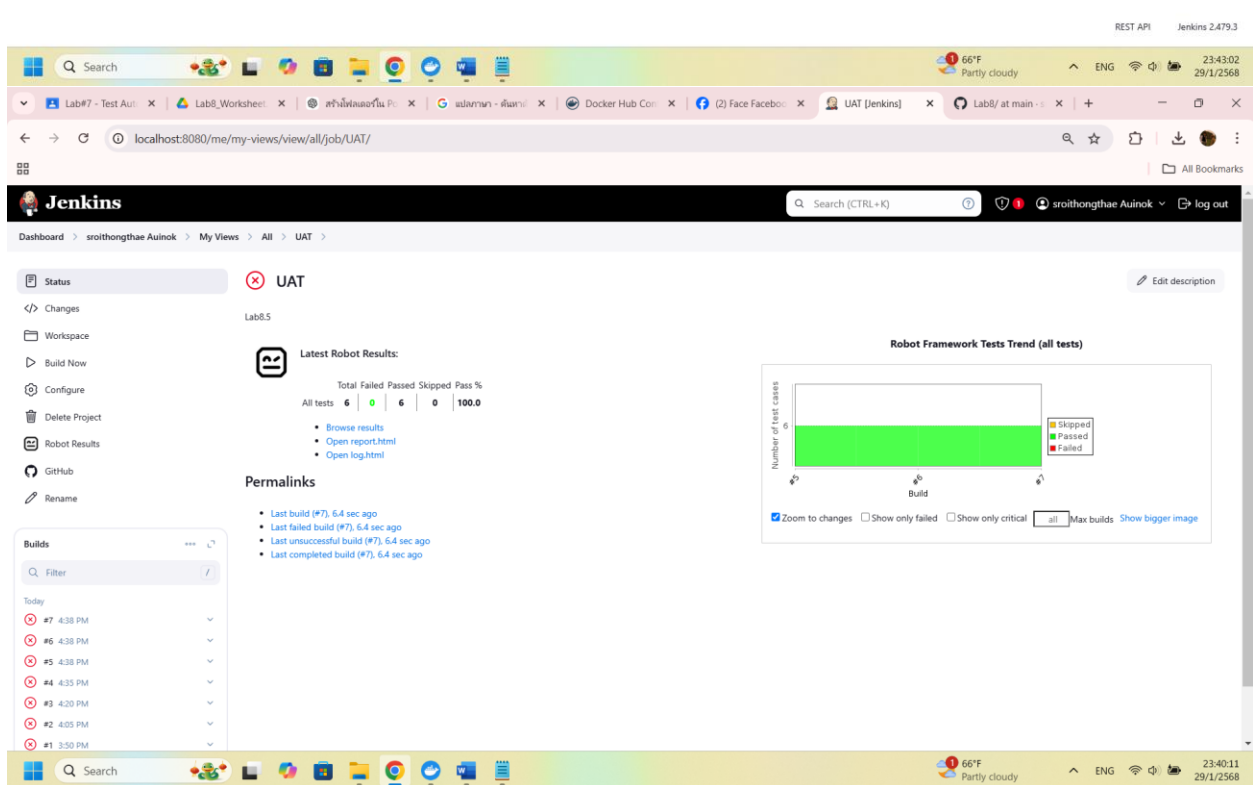
Lab Worksheet



The screenshot shows the Jenkins dashboard with the following details:

- Build Queue:** No builds in the queue.
- Build Executor Status:** 0/2
- Table:**

S	W	Name	Last Success	Last Failure	Last Duration	Robot Results	Duration Trend
⊘	☁	UAT	N/A	4 min 2 sec #7	2 sec	🟢🟢🟢🟢	



The screenshot shows the Jenkins job view for 'UAT' with the following details:

- Status:** UAT (Lab8.5)
- Latest Robot Results:**

Total	Failed	Passed	Skipped	Pass %
All tests	6	6	0	100.0
- Permalinks:**
 - Last build (#7), 6.4 sec ago
 - Last failed build (#7), 6.4 sec ago
 - Last unsuccessful build (#7), 6.4 sec ago
 - Last completed build (#7), 6.4 sec ago
- Robot Framework Tests Trend (all tests):**

Number of test cases	Passed	Failed	Skipped
6	6	0	0

Lab Worksheet

The screenshot shows the Jenkins web interface with the console output of a build job. The build is titled "Console Output" and is in a failed state. The output shows the following steps:

```

Started by user sroithongthae Aulnok
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UART
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/jenkins_home/workspace/UART/.git # timeout=10
Fetching changes from the remote git repository
> git config remote.origin.url https://github.com/sroithongthae346-6/Lab8.git # timeout=10
Fetching upstream changes from https://github.com/sroithongthae346-6/Lab8.git
> git --version # timeout=10
> git --version # 'git version 2.39.5'
> git fetch --tags --force --progress -- https://github.com/sroithongthae346-6/Lab8.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 78bbd79c6577a983461e41fb265a11c5caa0621 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 78bbd79c6577a983461e41fb265a11c5caa0621 # timeout=10
Commit message: "Delete output.html"
> git rev-list --no-walk 78bbd79c6577a983461e41fb265a11c5caa0621 # timeout=10
[UART] $ /bin/sh -xe /tmp/jenkins3191350612868972717.sh
+ ls -al
total 560
drwxr-xr-x 3 jenkins jenkins 4096 Jan 29 16:38 .
drwxr-xr-x 3 jenkins jenkins 4096 Jan 29 15:50 ..
drwxr-xr-x 8 jenkins jenkins 4096 Jan 29 16:38 .git
-rw-r--r-- 1 jenkins jenkins 1632 Jan 29 16:05 Lab7.robot
-rw-r--r-- 1 jenkins jenkins 6 Jan 29 16:05 README.md
-rw-r--r-- 1 jenkins jenkins 654 Jan 29 16:05 gherkin_login.robot
-rw-r--r-- 1 jenkins jenkins 4393 Jan 29 16:05 incomplete_robot
-rw-r--r-- 1 jenkins jenkins 2385 Jan 29 16:05 incomplete_res_robot
-rw-r--r-- 1 jenkins jenkins 1489 Jan 29 16:05 invalid_login_robot
-rw-r--r-- 1 jenkins jenkins 245207 Jan 29 16:36 log.html
-rw-r--r-- 1 jenkins jenkins 25291 Jan 29 16:38 output.xml
-rw-r--r-- 1 jenkins jenkins 247460 Jan 29 16:36 report.html
-rw-r--r-- 1 jenkins jenkins 1272 Jan 29 16:05 resource.robot
-rw-r--r-- 1 jenkins jenkins 448 Jan 29 16:05 valid_login_robot
+ robot valid_login.robot
/tmp/jenkins3191350612868972717.sh: 3: 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:
Done!
Copying log files to build dir:
Done!
Assigning results to build:
Done!
Checking thresholds:
Done!
Done publishing Robot results.
Finished: FAILURE
  
```

The build fails at the "Execute shell" step because the command "robot valid_login.robot" is not found. The error message is "robot: not found".