

Lab Worksheet

ชื่อ-นามสกุล นายณรงศ์ฤทธิ์ พาทามาศ รหัสนักศึกษา 653380265-6 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
2. ย้ายตำแหน่งปัจจุบันไปที่ Lab8_1 เพื่อใช้เป็น Working directory
3. ป้อนคำสั่ง \$ docker pull busybox หรือ \$ sudo docker pull busybox สำหรับกรณีที่เกิดปัญหา Permission denied
(หมายเหตุ: BusyBox เป็น software suite ที่รองรับคำสั่งบางอย่างบน Unix - <https://busybox.net>)
4. ป้อนคำสั่ง \$ docker images

Lab Worksheet

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

Terminal

```

fan@MacBook-Pro--FAN Desktop % mkdir Lab8_1
fan@MacBook-Pro--FAN Desktop % cd Lab8_1
fan@MacBook-Pro--FAN Lab8_1 % docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
559c60843878: 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
fan@MacBook-Pro--FAN Lab8_1 % docker images

```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
jenkins/jenkins	latest	baa9d0bd2897	16 hours ago	495MB
busybox	latest	fc0179a204e2	3 months ago	4.04MB
synthesizedio/whalesay	latest	07da125a0bc8	6 months ago	45.2MB

```

fan@MacBook-Pro--FAN Lab8_1 % 

```

(1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร ชื่อ image

(2) Tag ที่ใช้บ่งบอกถึงอะไร เวอร์ชันของ Docker image

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

Lab Worksheet

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

```

Terminal
fan@MacBook-Pro--FAN Lab8_1 % docker run busybox
fan@MacBook-Pro--FAN Lab8_1 % 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 22 07:33 .
drwxr-xr-x  1 root    root          4096 Jan 22 07:33 ..
-rwxr-xr-x  1 root    root           0 Jan 22 07:33 .dockerenv
drwxr-xr-x  2 root    root        12288 Sep 26 21:31 bin
drwxr-xr-x  5 root    root         360 Jan 22 07:33 dev
drwxr-xr-x  1 root    root          4096 Jan 22 07:33 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 252 root    root           0 Jan 22 07:33 proc
drwx----- 1 root    root          4096 Jan 22 07:33 root
dr-xr-xr-x 11 root    root           0 Jan 22 07:33 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
fan@MacBook-Pro--FAN Lab8_1 % docker run busybox echo "Hello Narongrit P
ararmard from busybox"
Hello Narongrit Pararmard from busybox
fan@MacBook-Pro--FAN Lab8_1 % docker ps -a
CONTAINER ID   IMAGE      COMMAND
CREATED        STATUS      PORTS      NAMES
f966d34674aa   busybox    "echo 'Hello Narongr..."
8 seconds ago   Exited (0) 8 seconds ago      awesome_b
assi
f527cf1d5a91   busybox    "sh"
About a minute ago   Exited (0) About a minute ago      hardcore_
noyce
d602cee5ffac   busybox    "sh"
About a minute ago   Exited (0) About a minute ago      dazzling_
gauss
e1d9c268cfe1   synthesiz... "/usr/local/bin/cows..."
52 minutes ago     Exited (0) 52 minutes ago      happy_tes
la
fan@MacBook-Pro--FAN Lab8_1 %

```

- (1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
 รันคอนเทนเนอร์แบบ Interactive พร้อมเปิด shell เพื่อโต้ตอบ ทำให้สามารถตรวจสอบไฟล์หรือ
 ทดสอบคำสั่งใน BusyBox

Lab Worksheet

- i : โหมดอินพุตแบบโต้ตอบ (Interactive Input)
- t : เปิด pseudo-TTY เพื่อให้สามารถแสดงผลแบบ terminal ได้

(2) คอลัมน์ STATUS จากการรันคำสั่ง docker ps -a แสดงถึงข้อมูลอะไร

แสดงสถานะการทำงานของ container

Exited คอนเทนเนอร์หยุดทำงานแล้ว อาจเกิดจากการรันคำสั่งเสร็จสมบูรณ์หรือเกิดข้อผิดพลาด
“หยุดทำงาน”

Up คอนเทนเนอร์กำลังทำงานอยู่ และยังคงรันอยู่ในระบบ (เช่นเว็บเซิร์ฟเวอร์หรือฐานข้อมูล)
“กำลังทำงาน”

12. ป้อนคำสั่ง \$ docker rm <container ID ที่ต้องการลบ>

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

Terminal						
fan@MacBook-Pro--FAN ~ % docker ps -a						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f966d34674aa	busybox	"echo 'Hello Narongr..."	5 days ago	Exited (0) 5 days ago		awesome_bassi
f527cf1d5a91	busybox	"sh"	5 days ago	Exited (0) 5 days ago		hardcore_noyce
d602cee5ffac	busybox	"sh"	5 days ago	Exited (0) 5 days ago		dazzling_gauss
e1d9c268cfe1	synthesizedio/whalesay:latest	"/usr/local/bin/cows..."	5 days ago	Exited (0) 5 days ago		happy_tesla
fan@MacBook-Pro--FAN ~ % docker rm f966d34674aa						
f966d34674aa						
fan@MacBook-Pro--FAN ~ %						

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."

Lab Worksheet

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

```

Terminal

fan@MacBook-Pro--FAN Desktop % mkdir Lab8_2
fan@MacBook-Pro--FAN Desktop % ls

fan@MacBook-Pro--FAN Desktop % cd Lab8_2
fan@MacBook-Pro--FAN Lab8_2 % nano Dockerfile.swp
fan@MacBook-Pro--FAN Lab8_2 % ls
Dockerfile.swp
UW PICO 5.09 File: Dockerfile.swp

FROM busybox
CMD echo "Hi there. This is my first docker image."
CMD echo "Name:Narongrit Pararmard Student_Id:653380265-6 Nick_Name:fan"

fan@MacBook-Pro--FAN Lab8_2 % docker build -t firstdocker -f Dockerfile.swp .
[+] Building 0.0s (5/5) FINISHED docker:desktop-linux
=> [internal] load build definition from Dockerfile.swp 0.0s
=> => transferring dockerfile: 181B 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructio 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD 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 sha256:9eace285502efdba5869e957af1f3b893a368 0.0s
=> => naming to docker.io/library/firstdocker 0.0s

3 warnings found (use docker --debug to expand):
- 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
fan@MacBook-Pro--FAN Lab8_2 % docker run firstdocker
Name:Narongrit Pararmard Student_Id:653380265-6 Nick_Name:fan

```

- (1) คำสั่งที่ใช้ในการ run คือ

Docker run firstdocker

Lab Worksheet

- (2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสังเขป
ใช้ตั้งชื่อ Docker image

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 และบันทึกคำสั่งต่อไปนี้

```
$ 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>/lab8do
5. ทำการรัน Docker image บน Container ในเครื่องของตัวเองเพื่อทดสอบผลลัพธ์ ด้วยคำสั่ง
\$ docker run <username ที่ลงทะเบียนกับ Docker Hub>/lab8

Lab Worksheet

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

```

Terminal
fan@MacBook-Pro--FAN Desktop % mkdir Lab8_3
fan@MacBook-Pro--FAN Desktop % cd Lab8_3
fan@MacBook-Pro--FAN Lab8_3 % ls
fan@MacBook-Pro--FAN Lab8_3 % nano Dockerfile.swp
File: Dockerfile.swp

FROM busybox
CMD echo "Hi there. My work is done. You can run them from my Docker image."
CMD echo "Name:Narongrit Paramard Student_Id:653380265-6"

fan@MacBook-Pro--FAN Lab8_3 % docker build -t narongritparamard/lab8
-f Dockerfile.swp .
[+] Building 0.0s (5/5) FINISHED          docker:desktop-linux
=> [internal] load build definition from Dockerfile.swp      0.0s
=> => transferring dockerfile: 193B                          0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instruc 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for 0.0s
=> [internal] load metadata for docker.io/library/busybox:lat 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:dad6648fae2dfd4b92fcb2283af7bea8f6 0.0s
=> => naming to docker.io/narongritparamard/lab8            0.0s

3 warnings found (use docker --debug to expand):
- 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
fan@MacBook-Pro--FAN Lab8_3 % docker run narongritparamard/lab8
Name:Narongrit Paramard Student_Id:653380265-6

```

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)

```

fan@MacBook-Pro--FAN Lab8_3 % docker push narongritparamard/lab8
Using default tag: latest
The push refers to repository [docker.io/narongritparamard/lab8]
613e5fc506b9: Mounted from library/busybox
latest: digest: sha256:c1ac4f5da1ba8a51f79b87e7da10018427d53076b720ea85c71f93db54508a20 size: 527

```

Name	Last Pushed ↑	Contains	Visibility	Scout
narongritparamard/lab8	17 minutes ago	IMAGE	Public	Inactive

Lab Worksheet

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

```

Terminal
fan@MacBook-Pro--FAN Desktop % mkdir Lab8_4
fan@MacBook-Pro--FAN Desktop % cd Lab8_4
fan@MacBook-Pro--FAN Lab8_4 % ls
fan@MacBook-Pro--FAN 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 | 15.24 MiB/s, done.
Resolving deltas: 100% (523/523), done.
fan@MacBook-Pro--FAN Lab8_4 % ls
getting-started
fan@MacBook-Pro--FAN Lab8_4 % cd getting-started
fan@MacBook-Pro--FAN getting-started % ls
Dockerfile          docker-compose.yml
LICENSE             docs
README.md           mkdocs.yml
app                 requirements.txt
build.sh
fan@MacBook-Pro--FAN getting-started % cd app
fan@MacBook-Pro--FAN app % ls
package.json  spec          src          yarn.lock
fan@MacBook-Pro--FAN app % cat package.json
{
  "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"
  }
}
fan@MacBook-Pro--FAN app %

```


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

```

Terminal
fan@MacBook-Pro--FAN app % docker build -t myapp_6533802656 -f Dockerfile.swp .
[+] Building 14.3s (10/10) FINISHED          docker:desktop-linux
=> [internal] load build definition from Dockerfile.swp      0.0s
=> => transferring dockerfile: 245B                          0.0s
=> [internal] load metadata for docker.io/library/node:18-alp 4.1s
=> [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@sha256:974afb6 2.5s
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6 0.0s
=> => sha256:4fe16fa8f46966191d59cfcabfff13 39.66MB / 39.66MB 1.4s
=> => sha256:fc4eb59aab89271acc5a8bd8e5e96fc1 1.26MB / 1.26MB 1.1s
=> => sha256:974afb6cbc0314dc6502b14243b8a39f 7.67kB / 7.67kB 0.0s
=> => sha256:d59895120001b37ef5f75afc6342329f 1.72kB / 1.72kB 0.0s
=> => sha256:7221f40791e5e6da0c9fa6b49b6238a5 6.20kB / 6.20kB 0.0s
=> => sha256:52f827f723504aa3325bb5a54247f0dc 3.99MB / 3.99MB 1.4s
=> => sha256:e668eba0f82bcfec9fb0cd787bd7f3d013a1 443B / 443B 1.4s
=> => extracting sha256:52f827f723504aa3325bb5a54247f0dc4b92b 0.1s
=> => extracting sha256:4fe16fa8f46966191d59cfcabfff137a623b3 0.8s
=> => extracting sha256:fc4eb59aab89271acc5a8bd8e5e96fc17af48 0.0s
=> => extracting sha256:e668eba0f82bcfec9fb0cd787bd7f3d013a12 0.0s
=> [internal] load build context                              0.1s
=> => transferring context: 4.60MB                             0.0s
=> [2/4] WORKDIR /app                                         0.2s
=> [3/4] COPY . .                                             0.0s
=> [4/4] RUN yarn install --production                        7.1s
=> => exporting to image                                         0.4s
=> => exporting layers                                           0.4s
=> => writing image sha256:952b84c2ecbadd54fdb6bec648e3e8099 0.0s
=> => naming to docker.io/library/myapp_6533802656            0.0s

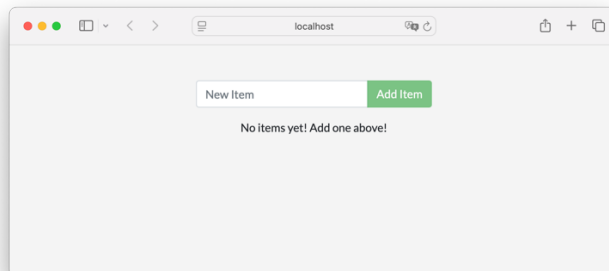
What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview
fan@MacBook-Pro--FAN app % docker run -dp 3000:3000 myapp_6533802656
bb5b01f566fb50ac71a3c2b4e754c930125ef00306cbb72034d3c124fd452bbe

```

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

Lab Worksheet

[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

`<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

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

[Check point#10] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง)

แสดงคำสั่งและผลลัพธ์ที่ได้ทางหน้าจอ พร้อมกับตอบคำถามต่อไปนี้

```
fan@MacBook-Pro--FAN app % docker build -t myapp_6533802656 -f Dockerfile.swp .
```

```
[+] Building 10.6s (10/10) FINISHED
=> [internal] load build definition from Dockerfile.swp
=> => transferring dockerfile: 245B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25
=> [internal] load build context
=> => transferring context: 9.92kB
=> CACHED [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> => writing image sha256:f45c42d7b7d3beb105312ad1036911db820dc18cc3c561f7b80e4f524de4cb38
=> => naming to docker.io/library/myapp_6533802656
```

```
docker:desktop-linux
0.0s
0.0s
3.0s
0.0s
0.0s
0.0s
0.0s
0.0s
0.0s
0.0s
0.0s
0.0s
0.0s
0.0s
0.0s
0.0s
0.0s
0.0s
```

What's next:

View a summary of image vulnerabilities and recommendations → `docker scout quickview`

```
fan@MacBook-Pro--FAN app % docker run -dp 3000:3000 myapp_6533802656
```

```
bdd82b86295f1c15e46546772ae362d149ebe3637e940bf2a7b1dc0133ba3342
```

```
docker: Error response from daemon: driver failed programming external connectivity on endpoint agitated_herschel (bd0a73216cb6f6242c7a0a93d0747d07185b5cc8dd3e15d4a541f5a3bd3ab13b): Bind for 0.0.0.0:3000 failed: port is already allocated.
```

```
fan@MacBook-Pro--FAN app %
```

Lab Worksheet

(1) Error ที่เกิดขึ้นหมายความว่าอย่างไร และเกิดขึ้นเพราะอะไร

หมายความว่า Docker พยายามใช้ พอร์ต 3000 จาก container เข้ากับเครื่องนี้ แต่พอร์ตนี้มีการใช้งานอยู่แล้ว (ถูก "allocated" ไปแล้ว) จึงไม่สามารถผูกพอร์ตนี้ได้อีก สาเหตุที่เกิดขึ้นคือ พอร์ต 3000 บน host machine ถูกใช้งานอยู่

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
- เลือกไอคอนถังขยะในแถวของ Container ที่ต้องการจะลบ
- ยืนยันโดยการกด Delete forever

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยใช้คำสั่งเดียวกันกับข้อ 6

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

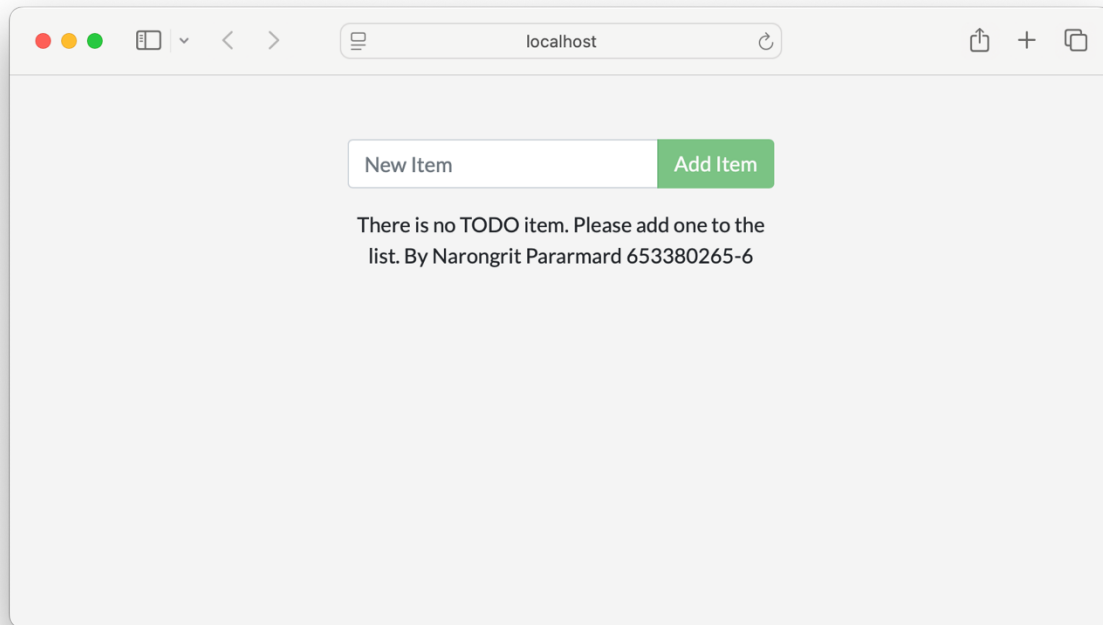
[Check point#11] Capture หน้าจอ (ทั้งหน้าต่างและทุกหน้าต่างที่เกี่ยวข้อง) แสดงผลลัพธ์ที่ได้บน Browser และ Dashboard ของ Docker desktop

```

fan@MacBook-Pro:~$ docker rm -f bb5b01f566fb
bb5b01f566fb
fan@MacBook-Pro:~$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS        NAMES
88a67f3a79cd   narongritparamard/lab8             "/bin/sh -c 'echo \"N... 4 hours ago    Exited (0) 4 hours ago
0f84f9d7f8a5   firstdocker                         "/bin/sh -c 'echo \"N... 5 hours ago    Exited (0) 5 hours ago
f527cf1d5a91   busybox                             "sh"                   5 days ago    Exited (0) 5 days ago  hardcore_noyce
d602cee5ffac   busybox                             "sh"                   5 days ago    Exited (0) 5 days ago  dazling_gauss
e1d9c268cfe1   synthesizedio/whalesay:latest       "/usr/local/bin/cows... 5 days ago    Exited (0) 5 days ago  happy_tesla
fan@MacBook-Pro:~$ docker build -t myapp_6533802656 -f Dockerfile.swp .
[+] Building 2.2s (18/18) FINISHED
=> [internal] load build definition from Dockerfile.swp
=> => transferring dockerfile: 245B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fb2d040975e9059dd066be3e274fbb25
=> [internal] load build context
=> => transferring context: 4.49kB
=> CACHED [2/4] WORKDIR /app
=> CACHED [3/4] COPY . .
=> CACHED [4/4] RUN yarn install --production
=> exporting to image
=> => exporting layers
=> writing image sha256:f45c42d7b7d3beb105312ad1036911db820dc18cc3c561f7b80e4f524de4cb38
=> naming to docker.io/library/myapp_6533802656

What's next:
View a summary of image vulnerabilities and recommendations -> docker scout quickview
fan@MacBook-Pro:~$ docker run -dp 3000:3000 myapp_6533802656
7600df3564ed9aaeb11e09823cf1abdf8c386a59a74184c38e257e4e52f9027
fan@MacBook-Pro:~$

```



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. บันทึกการรหัสผ่านของ Admin user ไว้สำหรับ log-in ในครั้งแรก

[Check point#12] Capture หน้าจอที่แสดงผล Admin

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

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

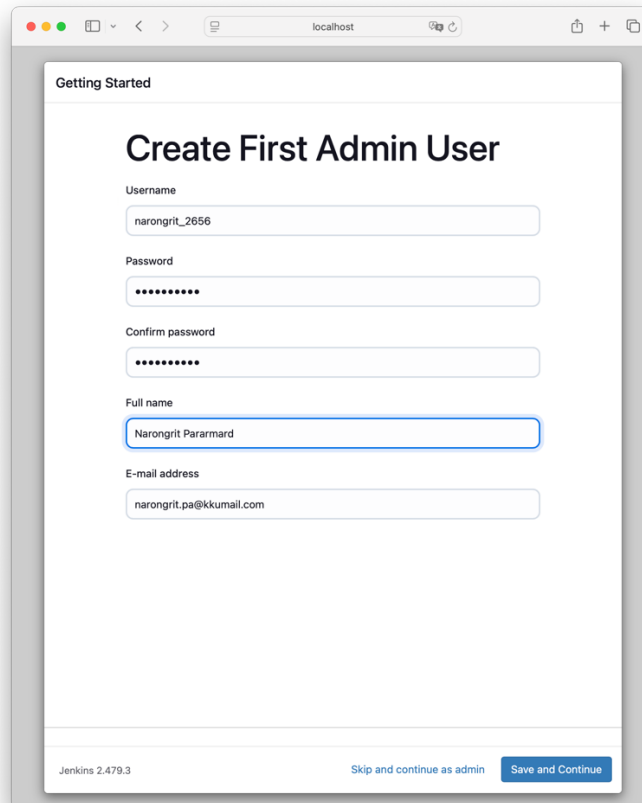
165c3a58559e4b1bb12d40db7f9300ad

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

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

Lab Worksheet

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



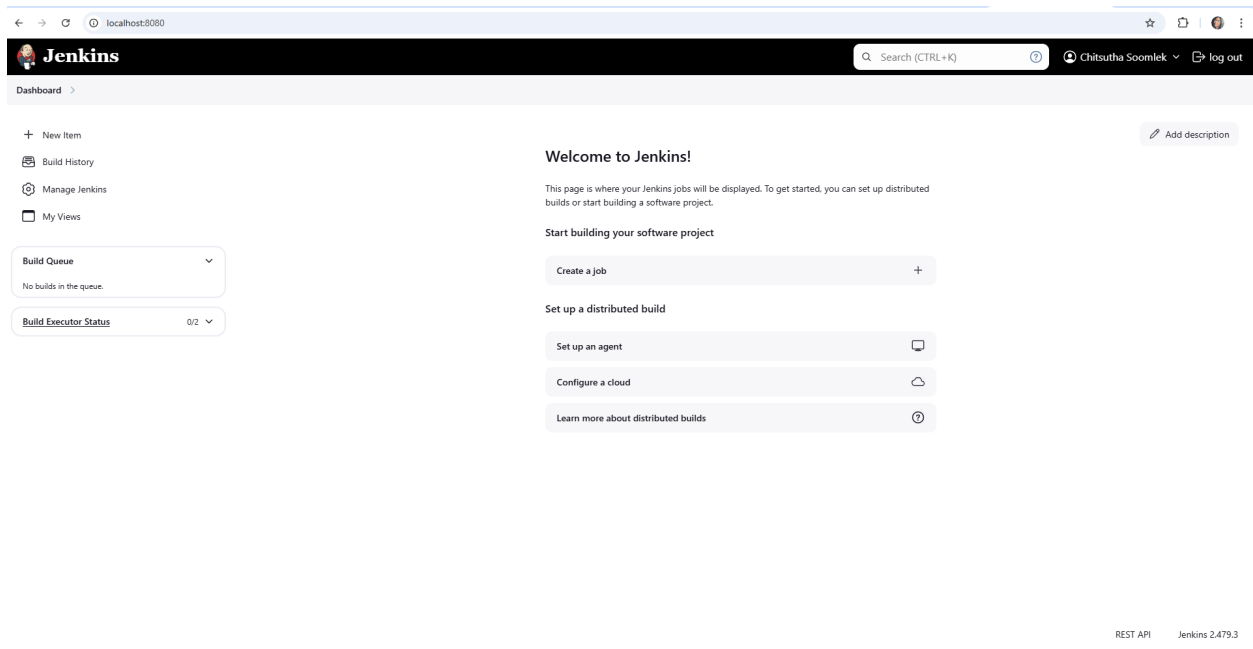
The screenshot shows the Jenkins 'Getting Started' page with the 'Create First Admin User' form. The form fields are filled with the following information:

- Username: narongrit_2656
- Password: (masked with dots)
- Confirm password: (masked with dots)
- Full name: Narongrit Paramard
- E-mail address: narongrit.pa@kkumail.com

At the bottom of the form, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'. The Jenkins version 'Jenkins 2.479.3' is displayed in the bottom left corner.

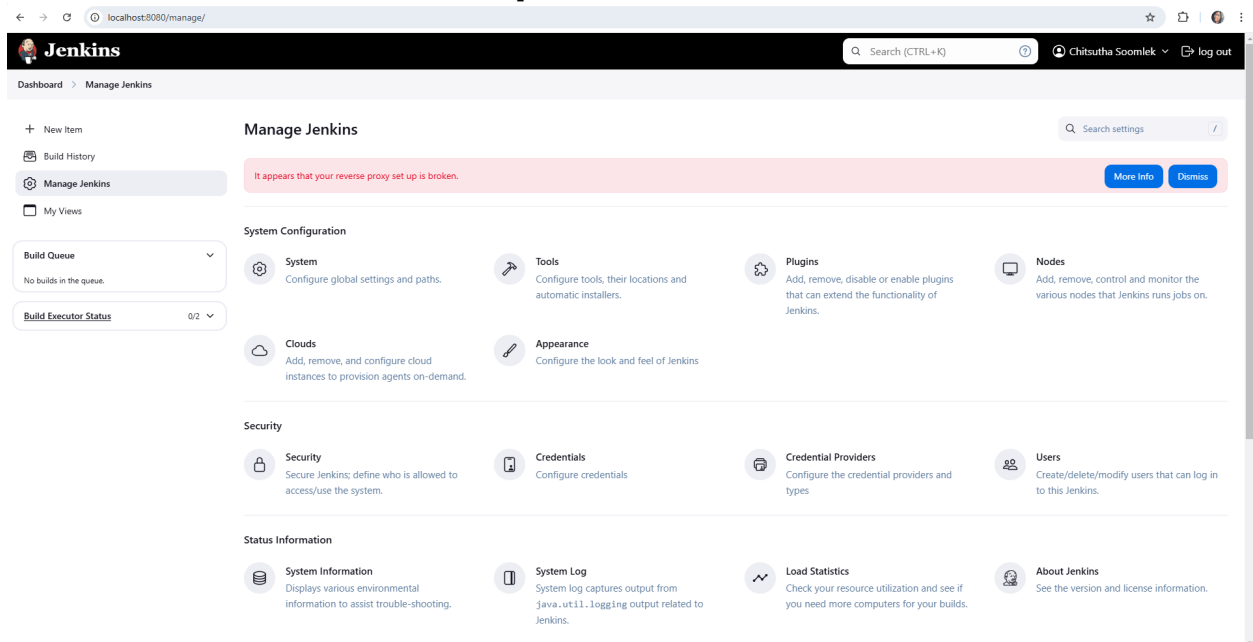
7. กำหนด Jenkins URL เป็น <http://localhost:8080/lab8>
8. เมื่อติดตั้งเรียบร้อยแล้วจะพบหน้าจอ Dashboard ดังแสดงในภาพ

Lab Worksheet



The screenshot shows the Jenkins Dashboard at localhost:8080. The top navigation bar includes the Jenkins logo, a search bar, and user information (Chitsutha Soomlek) with a log out button. The left sidebar contains links for New Item, Build History, Manage Jenkins, and My Views. The main content area displays a 'Welcome to Jenkins!' message and a 'Start building your software project' section with a 'Create a job' button. Below this, the 'Set up a distributed build' section offers options to 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. The bottom right corner shows 'REST API' and 'Jenkins 2.479.3'.

9. เลือก Manage Jenkins แล้วไปที่เมนู Plugins



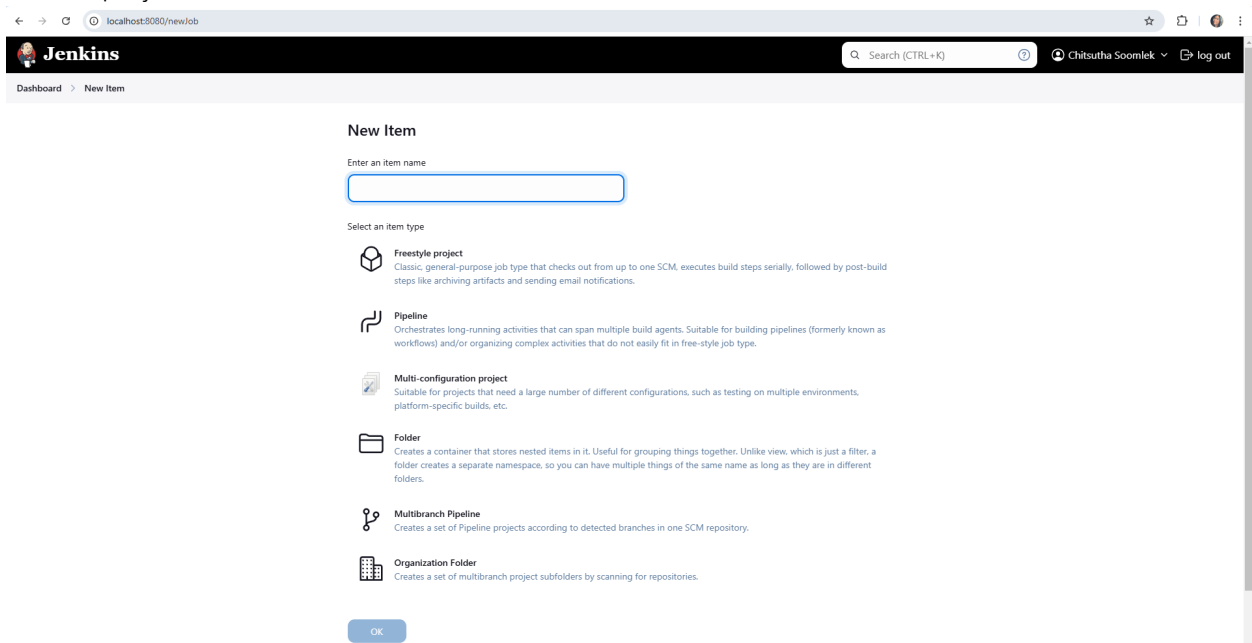
The screenshot shows the 'Manage Jenkins' page at localhost:8080/manage/. A red warning banner at the top states 'It appears that your reverse proxy set up is broken.' with 'More info' and 'Dismiss' buttons. The page is organized into several sections: 'System Configuration' (System, Tools, Plugins, Nodes, Clouds, Appearance), 'Security' (Security, Credentials, Credential Providers, Users), and 'Status Information' (System Information, System Log, Load Statistics, About Jenkins). The 'Plugins' section is highlighted, indicating the next step in the lab.

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม

Lab Worksheet



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

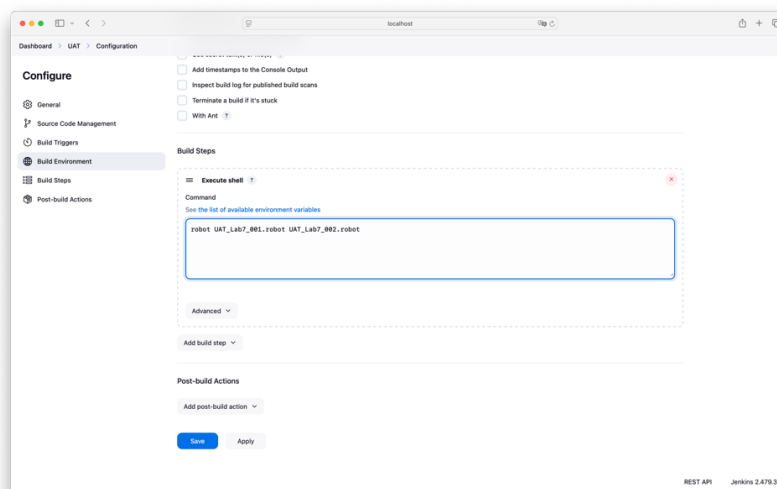
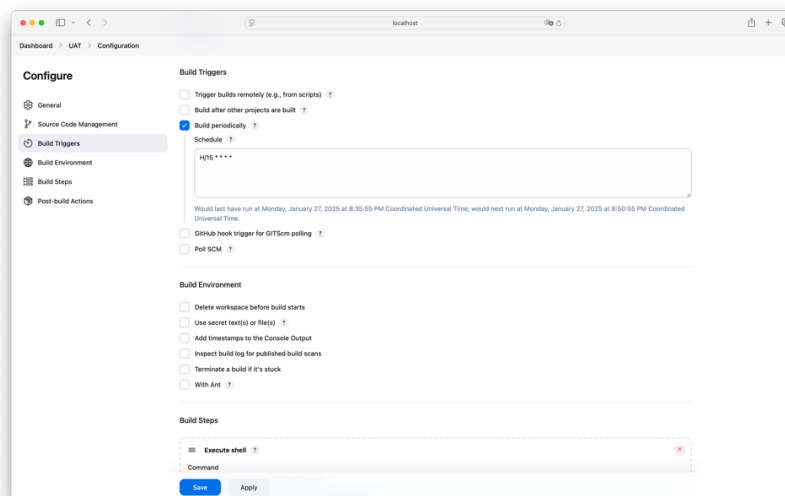
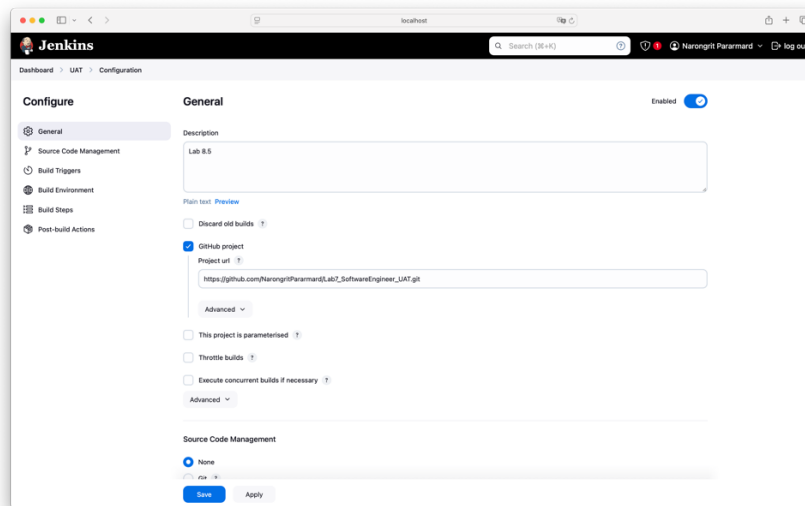
GitHub project: กดเลือก แล้วใส่ Project URL เป็น repository ที่เก็บโค้ด .robot (ดูขั้นตอนที่ 12)

Build Trigger: เลือกแบบ Build periodically แล้วกำหนดให้ build ทุก 15 นาที

Build Steps: เลือก Execute shell แล้วใส่คำสั่งในการรันไฟล์ .robot (หากไฟล์ไม่ได้อยู่ในหน้าแรกของ repository ให้ใส่ Path ไปถึงไฟล์ให้เรียบร้อยด้วย)

Lab Worksheet

[Check point#14] Capture หน้าจอแสดงการตั้งค่า พร้อมกับตอบคำถามต่อไปนี้



Lab Worksheet

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

robot UAT_Lab7_001.robot UAT_Lab7_002.robot

Post-build action: เพิ่ม Publish Robot Framework test results ->

ระบุไดเรกทอรีที่เก็บไฟล์ผลการทดสอบโดย 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

