

Lab Worksheet

ชื่อ-นามสกุล นิธินันท์ อารยรุ่งโรจน์ รหัสนักศึกษา 653380204-6 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

```
Status: Image is up to date for busybox:latest
docker.io/library/busybox:latest
nokweed@Macintosh-4 Lab8_1 % docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
ubuntu          latest   20377134ad88  2 months ago  101MB
busybox         latest   fc0179a204e2  3 months ago  4.04MB
nokweed@Macintosh-4 Lab8_1 % docker run -it busybox
/ # ls
```

- (1) สิ่งที่อยู่ภายใต้คอลัมน์ Repository คืออะไร ubuntu คือ อิมเมจของระบบปฏิบัติการ ubuntu
Busybox อิมเมจขนาดเล็กที่มีเครื่องมือพื้นฐานของ Linux
- (2) Tag ที่ใช้บ่งบอกถึงอะไร TAG ใช้บ่งบอก เวอร์ชัน หรือ สถานะ ของอิมเมจ เช่น latest หมายถึงเวอร์ชันล่าสุด
หรือระบุเป็นเวอร์ชันเฉพาะ เช่น ubuntu:20.04 เพื่อใช้งานอิมเมจที่ตรงตามต้องการ

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

```
nokweed@Macintosh-4 lab8_1 % docker run busybox
nokweed@Macintosh-4 lab8_1 % docker run -it busybox sh
/ #
/ # ls
bin      dev      etc      home      lib      lib64      proc      root      sys      tmp      usr      var
```

Lab Worksheet

Terminal

```
/ # ls -la
total 48
drwxr-xr-x  1 root    root        4096 Jan 23 03:28 .
drwxr-xr-x  1 root    root        4096 Jan 23 03:28 ..
-rw-r--r--x  1 root    root        0 Jan 23 03:28 .dockerenv
drwxr-xr-x  2 root    root      12288 Sep 26 21:31 bin
drwxr-xr-x  5 root    root       360 Jan 23 03:28 dev
drwxr-xr-x  1 root    root        4096 Jan 23 03:28 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  219 root   root        0 Jan 23 03:28 proc
drwx----- 1 root    root        4096 Jan 23 03:28 root
dr-xr-xr-x  11 root   root        0 Jan 23 03:28 sys
drwxrwxrwt  2 root    root        4096 Sep 26 21:31 tmp
drwxr-xr-x  4 root    root        4096 Sep 26 21:31 usr
```

Terminal

```
/ # exit
nokweed@Macintosh-4 lab8_1 % docker run busybox echo "Hello NithinanArayarungroj from busybox"
Hello NithinanArayarungroj from busybox
nokweed@Macintosh-4 lab8_1 % docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
9f652eb49fe0 busybox "echo 'Hello Nithinan..." 9 seconds ago Exited (0) 8 seconds ago
    relaxed_mclean
0947d4e2444f busybox "sh" About a minute ago Exited (0) About a minute ago
    kind_knuth
c9c6519be42e busybox "sh" About a minute ago Exited (0) About a minute ago
    stoic_goldberg
cd0607908201 busybox "sh" 36 minutes ago Exited (0) 36 minutes ago
    romantic_kalam
4b6a5752dc20 busybox "sh" 36 minutes ago Exited (0) 36 minutes ago
    silly_blackwell
8b860007cb7d ubuntu  "bash" 46 minutes ago Up 45 minutes
    my_lab_container
```

Terminal

```
Hello NithinanArayarungroj from busybox
nokweed@Macintosh-4 lab8_1 % docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
9f652eb49fe0 busybox "echo 'Hello Nithinan..." 9 seconds ago Exited (0) 8 seconds ago
    relaxed_mclean
0947d4e2444f busybox "sh" About a minute ago Exited (0) About a minute ago
    kind_knuth
c9c6519be42e busybox "sh" About a minute ago Exited (0) About a minute ago
    stoic_goldberg
cd0607908201 busybox "sh" 36 minutes ago Exited (0) 36 minutes ago
    romantic_kalam
4b6a5752dc20 busybox "sh" 36 minutes ago Exited (0) 36 minutes ago
    silly_blackwell
8b860007cb7d ubuntu  "bash" 46 minutes ago Up 45 minutes
```

Lab Worksheet

- (1) เมื่อใช้ option -it ในคำสั่ง run ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสั้นๆ
สร้างและรันคอนเทนเนอร์จากอินเมจ Ubuntu พร้อมเปิด Bash shell ให้สามารถโต้ตอบได้ โดยพร้อมต์
root@e502889471b9:/# หมายถึงคุณกำลังอยู่ในคอนเทนเนอร์
- (2) คลอ้มั่น STATUS จากการรันคำสั่ง docker ps -a และดึงข้อมูลอะไร
คลอ้มั่น STATUS และสถานะของคอนเทนเนอร์

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

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

```
nokweed@Macintosh-4 lab8_1 % docker run -it ubuntu bash
root@e502889471b9:/# #
nokweed@Macintosh-4 lab8_1 % docker ps -a
CONTAINER ID        IMAGE       COMMAND           CREATED          STATUS          PORTS     NAMES
6c0dc466208a        ubuntu      "bash"            2 minutes ago   Exited (127) 44 seconds ago   intelligent
_bhaskara
9f652eb49fe0        busybox     "echo 'Hello Nithina..'"  15 minutes ago  Exited (0) 15 minutes ago   relaxed_mcl
ean
0947d4e2444f        busybox     "sh"              17 minutes ago  Exited (0) 16 minutes ago   kind_knuth
c9c6519be42e        busybox     "sh"              17 minutes ago  Exited (0) 17 minutes ago   stoic_goldb
erg
cd0607908201        busybox     "sh"              52 minutes ago  Exited (0) 51 minutes ago   romantic_ka
lam
4b6a5752dc20        busybox     "sh"              52 minutes ago  Exited (0) 52 minutes ago   silly_black
well
8b860007cb7d        ubuntu      "bash"            About an hour ago  Up About an hour   my_lab_cont
_zoee
□ ○ kind_knuth          0947d4e2444f    busybox          0% 41 minutes ago
nokweed@Macintosh-4 lab8_1 % docker rm 0947d4e2444f1d0a53872f32e901b97aba8929eac8ede73aacbbf733aa82b80b
0947d4e2444f1d0a53872f32e901b97aba8929eac8ede73aacbbf733aa82b80b
nokweed@Macintosh-4 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. ย้ายตัวแฟ้มง่ายๆไปที่ 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”

Lab Worksheet

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

```
nokweed@Macintosh-4 Lab8_2 % docker build -t first_docker .
[+] Building 0.1s (5/5) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 198B
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavio 0.0s
=> WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the s 0.0s
=> WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavio 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 sha256:2acbc040ed1a64e7de31d5990ee310a08e6725f38c1ed95317c5704e63c48fb7 0.0s
=> => naming to docker.io/library/first_docker 0.0s

View build details: https://docker-desktop://dashboard/build/desktop-linux/desktop-linux/kbzxrsv3mrw5n9ca5btvjti4b
```

- (1) คำสั่งที่ใช้ในการ run คือ run container จาก image first_docker
- (2) Option -t ในคำสั่ง \$ docker build ส่งผลต่อการทำงานของคำสั่งอย่างไรบ้าง อธิบายมาพอสั้นๆ

Lab Worksheet

Option -t เมื่อใช้จะเป็นการตั้งชื่อ image เรียกลบผ่านชื่อ image ได้ หากไม่ใช้ docker สร้าง image เมื่อใช้งานต้องเรียกผ่าน image ID

แบบฝึกปฏิบัติที่ 8.3: การแชร์ Docker image ผ่าน Docker Hub

1. เปิดใช้งาน Docker desktop และ Login ด้วย Username และ Password ที่ลงทะเบียนกับ Docker Hub เօไว

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

สำหรับเครื่องที่ใช้ระบบปฏิบัติการwinโดว์ 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
```

Lab Worksheet

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

```
nokweed@Macintosh-4 Lab8_3 % docker build -t nithinan2046/lab8 .
[+] Building 0.1s (5/5) FINISHED
    => [internal] load build definition from Dockerfile                               docker:desktop-linux
    => => transferring dockerfile: 210B                                            0.0s
    => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavio 0.0s
    => WARN: MultipleInstructionsDisallowed: Multiple CMD instructions should not be used in the s 0.0s
    => WARN: JSONArgsRecommended: JSON arguments recommended for CMD to prevent unintended behavio 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 sha256:88c044e3e8281f40915f0544eee2903e362c814e18036fbba7dede92ce21e70c 0.0s
    => => naming to docker.io/nithinan2046/lab8                                    0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/pe2yu5320ete8ayoko4vwdiko

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)
nokweed@Macintosh-4 Lab8_3 % docker run nithinan2046/lab8
NithinanArayarungroj 653380204-6
```

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 และป้อน 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) และ Repository ที่มี Docker image (<username>/lab8)

Lab Worksheet

nithinan2046

Search by repository name

All content

Create a repository

Name	Last Pushed	Contains	Visibility	Scout
nithinan2046/lab8	8 minutes ago	IMAGE	Public	Inactive

1-1 of 1 < >

```

nokweed@Macintosh-4 Lab8_3 % docker run nithinan2046/lab8
NithinanArayarungroj 653380204-6
nokweed@Macintosh-4 Lab8_3 % docker push nithinan2046/lab8
Using default tag: latest
The push refers to repository [docker.io/nithinan2046/lab8]
613e5fc506b9: Mounted from library/busybox
latest: digest: sha256:19e306b9df7593af99622bad6f6dd307a206966080a335d10661e8d032fd232b size: 527

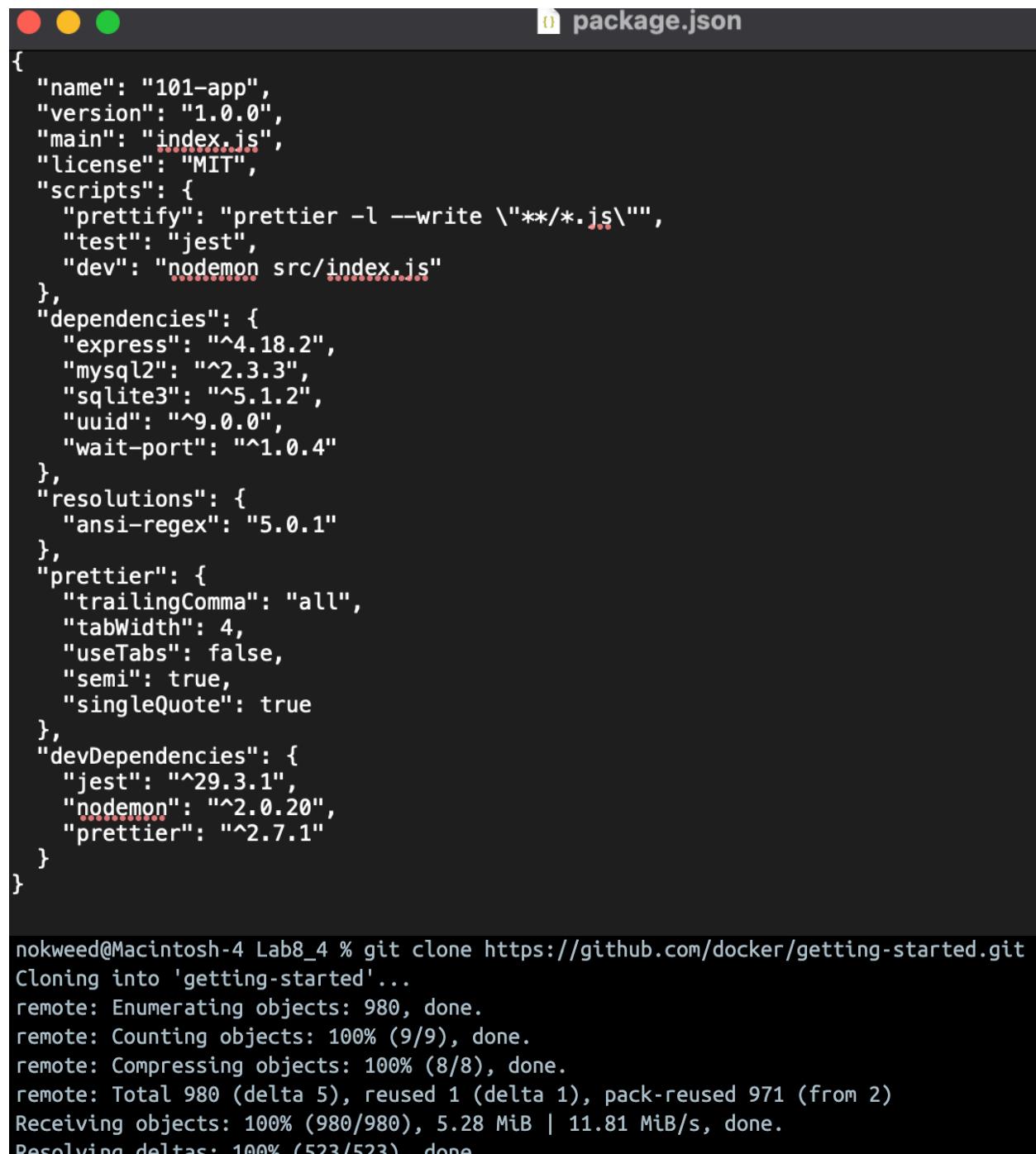
```

แบบฝึกปฏิบัติที่ 8.4: การ Build และ Update Container image และการ Update แอปพลิเคชัน

1. เปิด Command line หรือ Terminal จากนั้นสร้าง Directory ชื่อ Lab8_4
2. ทำการ Clone ซอฟต์แวร์โค้ดของเว็บแอปพลิเคชันจาก 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

Lab Worksheet



```

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"
  }
}

nokweed@Macintosh-4 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 | 11.81 MiB/s, done.
Resolving deltas: 100% (523/523), done.

```

4. ภายใต้ getting-started/app ให้สร้าง Dockerfile พร้อมกับใส่เนื้อหาดังต่อไปนี้ลงไว้ในไฟล์

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

```
WORKDIR /app
```

```
COPY ..
```

Lab Worksheet

RUN yarn install --production

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

EXPOSE 3000

5. ทำการ Build Docker image ที่สร้างขึ้นด้วยคำสั่งต่อไปนี้ โดยกำหนดให้ชื่อ image เป็น myapp_รหัสןศ. ไม่มีขีด

\$ docker build -t <myapp_รหัสนศ. ไม่มีขีด> .

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

```
nokweed@Macintosh-4 app % docker build -t myapp_6533802046 .
[+] Building 27.1s (10/10) FINISHED
=> [internal] load build definition from Dockerfile                               docker:desktop-linux
=> => transferring dockerfile: 187B                                         0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine               0.0s
=> [auth] library/node:pull token for registry-1.docker.io                      7.3s
=> [internal] load .dockerignore                                              0.0s
=> => transferring context: 2B                                              0.0s
=> [1/4] FROM docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d9  7.2s
=> => resolve docker.io/library/node:18-alpine@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d9  0.0s
=> => sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d975e9059dd066be3e274fbb25  7.67kB / 7.67kB  0.0s
=> => sha256:d59895120001b37ef5f75afc6342329f6037b1e2aea53a1055efa62b8bf6003  1.72kB / 1.72kB  0.0s
=> => sha256:7221f40791e5e6da0c9fa6b49b6238a51661222e4c58aba37e152e89914be09d  6.20kB / 6.20kB  0.0s
=> => sha256:52f827f723504aa3325bb5a54247f0dc4b92bb72569525bc951532c4ef679bd4  3.99MB / 3.99MB  0.6s
=> => sha256:4fe16fa8f46966191d59cfcabffff137a623b3cdda747d387bd85dcbf0feff3d  39.66MB / 39.66MB  5.7s
=> => sha256:fc4eb59aab89271acc5a8bd8e5e96fc17af489976964461471ea28fc8e0be459  1.26MB / 1.26MB  2.9s
=> => extracting sha256:52f827f723504aa3325bb5a54247f0dc4b92bb72569525bc951532c4ef679bd4  0.1s
=> => sha256:e668eba0f82bcfec9fb0cd787bd7f3d013a1266567b092e90ff8b3d3be41807c  4.43B / 443B   4.5s
=> => extracting sha256:4fe16fa8f46966191d59cfcabffff137a623b3cdda747d387bd85dcbf0feff3dd  1.3s
=> => extracting sha256:fc4eb59aab89271acc5a8bd8e5e96fc17af489976964461471ea28fc8e0be459  0.0s
=> => extracting sha256:e668eba0f82bcfec9fb0cd787bd7f3d013a1266567b092e90ff8b3d3be41807c  0.0s
=> [internal] load build context                                               0.1s
=> => transferring context: 4.60MB                                            0.0s
=> [2/4] WORKDIR /app                                                       0.1s
```

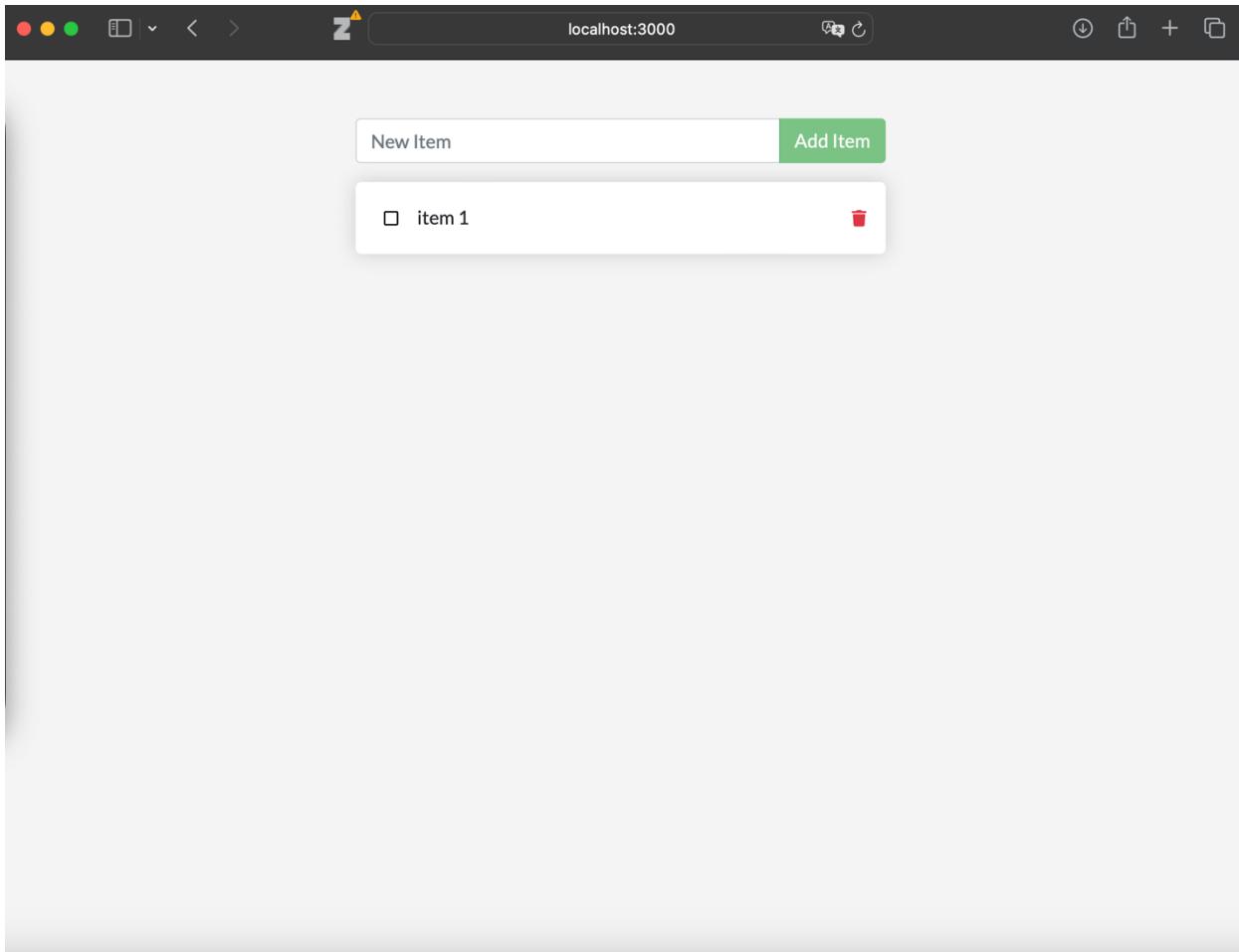
6. ทำการ Start ตัว Container ของแอปพลิเคชันที่สร้างขึ้น โดยใช้คำสั่ง

\$ docker run -dp 3000:3000 <myapp_รหัสนศ. 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

b. Save ไฟล์ให้เรียบร้อย

9. ทำการ Build Docker image โดยใช้คำสั่งเดียวกันกับข้อ 5

Lab Worksheet

10. Start และรัน Container ตัวใหม่ โดยใช้คำสั่งเดียวกันกับข้อ 6

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

```
nokweed@Macintosh-4 app % docker build -t myapp_6533802046 .
[+] Building 5.3s (10/10) FINISHED                                            docker:desktop-linux
=> [internal] load build definition from Dockerfile                      0.0s
=> => transferring dockerfile: 187B                                         0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine          5.3s
=> [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@sha256:974afb6cbc0314dc6502b14243b8a39fbb2d04d9  0.0s
=> [internal] load build context                                         0.0s
=> => transferring context: 4.34kB                                       0.0s
=> CACHED [2/4] WORKDIR /app                                              0.0s
=> CACHED [3/4] COPY . .                                                 0.0s
=> CACHED [4/4] RUN yarn install --production                           0.0s
=> exporting to image                                                    0.0s
=> => exporting layers                                                   0.0s
=> => writing image sha256:adf704d28d4272d95f71ba3e8645c0b7cb0490c29ae64ec6d82583b362b034F  0.0s
=> => naming to docker.io/library/myapp_6533802046                      0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/n34vaj4wa1xsl3edb8eh51lq
nokweed@Macintosh-4 app % docker run -dp 3000:3000 myapp_6533802046
38cf72102a00c78bf2582a31916e345cb1e3ee9c75e60c4cf45d722088b45d78
docker: Error response from daemon: driver failed programming external connectivity on endpoint vigilant_spence (0e691c2d
a4a85d48ac8a3979372cdff4d26928a30c612a7025d53f62d4102ab8): Bind for 0.0.0.0:3000 failed: port is already allocated.
```

(1) Error ที่เกิดขึ้นหมายความอย่างไร และเกิดขึ้นเพราะอะไร

มีการเชื่อมกับ port 3000 ในเครื่องมีการใช้งานอยู่

11. ลับ Container ของ Web application เวอร์ชันก่อนแก้ไขออกจากระบบ โดยใช้รีดิวิรีหนึ่งดังต่อไปนี้

a. ฝ่าย 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

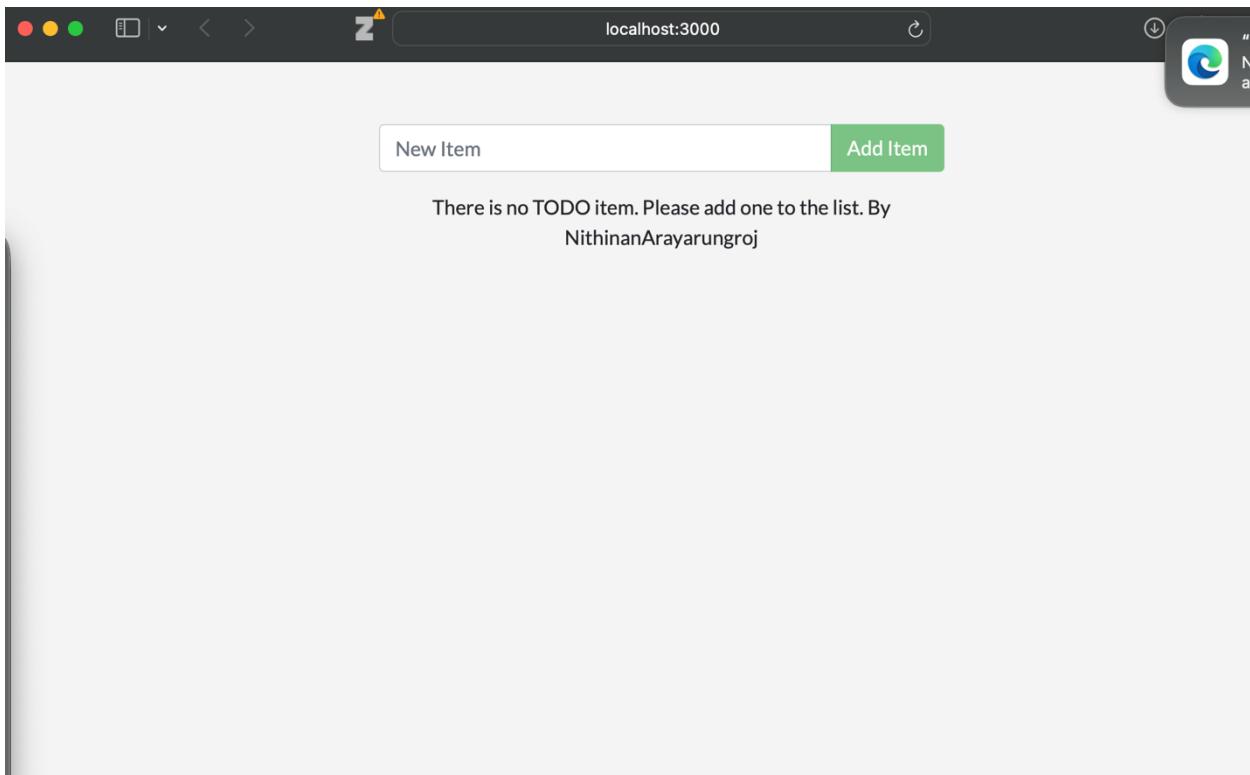
- i. ไปที่หน้าต่าง Containers
 - ii. เลือกไอคอนถังขยะในແຄວຂອງ Container ທີ່ຕ້ອງກາຈະລບບ
 - iii. ຍືນຍັນໂດຍກາຈຳ Delete forever

12. Start และรัน Container ตัวใหม่อีกครั้ง โดยในคำสั่งเดียวกันกับข้อ 6

13. ໄກສາ Browser ໂອດທີ່ URL = <http://localhost:3000>

Lab Worksheet

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



Lab Worksheet

Containers / nifty_archimedes

nifty_archimedes

Status: Running (42 minutes ago)

Logs Inspect Bind mounts Exec Files Stats

2025-01-29 09:52:27 Using sqlite database at /etc/todos/todo.db
2025-01-29 09:52:27 Listening on port 3000

Containers Give feedback

View all your running containers and applications. Learn more

0.00% / 800% (8 CPUs available) 21.62MB / 3.74GB

Search Only show running containers Delete

Name	Container ID	Image	Port(s)	CPU (%)	Last stat	Actions
ubuntu_lab	97149d15df5f	ubuntu		0%	6 days ago	⋮ ⌂
eloquent_mccar	18593cee6439	ubuntu		0%	6 days ago	⋮ ⌂
adoring_kirch	d5484e6ce0a9	nithinan202		0%	12 hours	⋮ ⌂
naughty_rhodes	f80fed95c838	myapp_6533802046:3000	3000:3000	0%	1 minute ago	⋮ ⌂

Selected 1 of 10

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

Lab Worksheet

[Check point#12] Capture หน้าจอที่แสดงผล Admin password

The screenshot shows a Jenkins setup interface. At the top, there's a navigation bar with 'Todo App' and 'Sign in [Jenkins]' links. Below the navigation bar, the title 'Getting Started' is displayed. The main content area has a heading 'Unlock Jenkins'. A text block explains that a password has been written to the log and a specific file path is provided: `/var/jenkins_home/secrets/initialAdminPassword`. Below this, instructions say to copy the password from either location and paste it below. A text input field contains several masked dots ('.....'). At the bottom right of the main content area is a blue 'Continue' button.

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

9e0401a86270485c98497482506615be

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

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

Lab Worksheet

[Check point#13] Capture หน้าจอที่แสดงผลการตั้งค่า

The screenshot shows the Jenkins user profile interface for 'Nithinan Arayarungroj'. The top navigation bar includes a search bar, a help icon, a notifications icon (1), and a log out button. Below the header, the user's name and Jenkins User ID are displayed. A sidebar on the left lists various account management options: Status, Builds, My Views, Account, Appearance, Preferences, Security, Experiments, and Credentials. The 'Status' option is currently selected.

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

The screenshot shows the Jenkins dashboard at the URL <http://localhost:8080>. The top navigation bar includes a search bar, a help icon, a notifications icon (0), and a log out button. The main content area features a 'Welcome to Jenkins!' message and a 'Start building your software project' section. It includes links for 'Create a job', 'Set up a distributed build', 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds'. On the left, there is a sidebar with links for 'New Item', 'Build History', 'Manage Jenkins', and 'My Views'. Below the sidebar are two status boxes: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (0/2).

9. เลือก Manage Jenkins และไปที่เมนู Plugins

Lab Worksheet

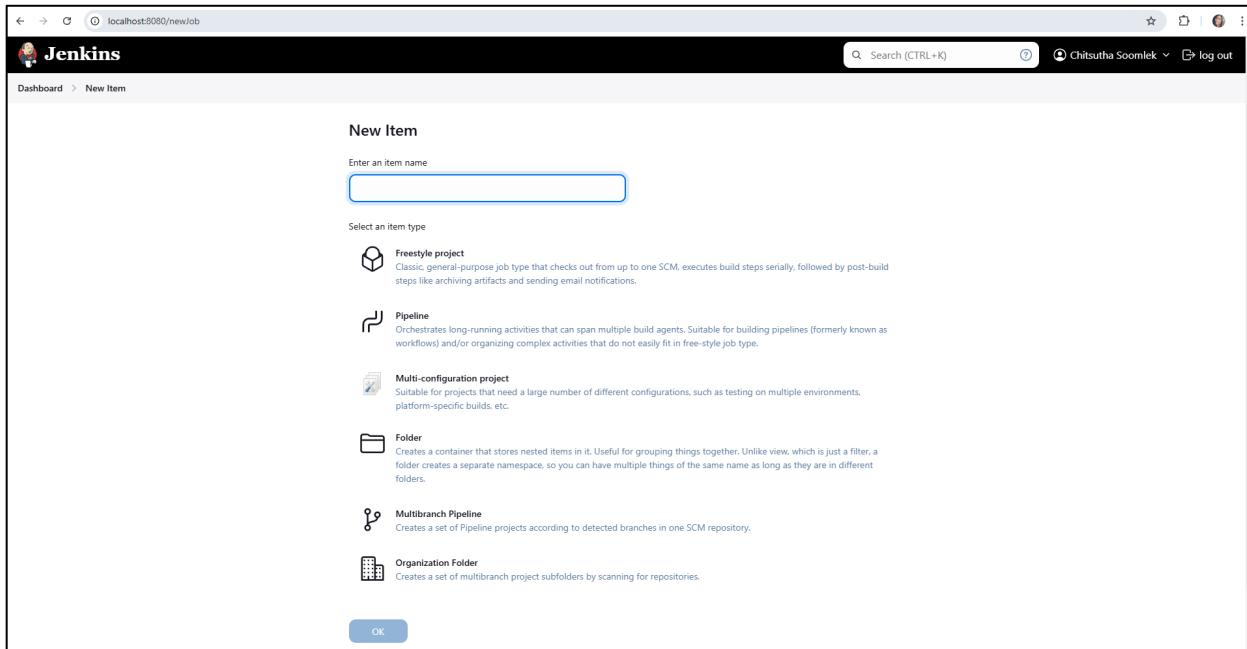
The screenshot shows the Jenkins Manage Jenkins page. In the top left, there are links for 'New Item', 'Build History', and 'Manage Jenkins'. A search bar at the top right contains the placeholder 'Search (CTRL+K)'. The user 'Chitsutha Soomlek' is logged in, and a 'log out' button is visible. A message box at the top center says 'It appears that your reverse proxy set up is broken.' with 'More Info' and 'Dismiss' buttons. The main area is titled 'System Configuration' and includes sections for 'Build Queue' (0 builds in the queue), 'Build Executor Status' (0/2), 'System' (Configure global settings and paths), 'Tools' (Configure tools, their locations and automatic installers), 'Clouds' (Add, remove, and configure cloud instances to provision agents on-demand), 'Appearance' (Configure the look and feel of Jenkins), 'Plugins' (Add, remove, disable or enable plugins that can extend the functionality of Jenkins), 'Nodes' (Add, remove, control and monitor the various nodes that Jenkins runs jobs on), 'Security' (Secure Jenkins; define who is allowed to access/use the system), 'Credentials' (Configure credentials), 'Credential Providers' (Configure the credential providers and types), 'Users' (Create/delete/modify users that can log in to this Jenkins), 'System Information' (Displays various environmental information to assist trouble-shooting), 'System Log' (System log captures output from java.util.logging output related to Jenkins), 'Load Statistics' (Check your resource utilization and see if you need more computers for your builds), and 'About Jenkins' (See the version and license information).

10. ไปที่เมนู Available plugins แล้วเลือกติดตั้ง Robotframework เพิ่มเติม

The screenshot shows the Jenkins Plugin Manager page under the 'Available plugins' tab. A search bar at the top right contains the placeholder 'Search (CTRL+K)'. The user 'Chitsutha Soomlek' is logged in, and a 'log out' button is visible. A message box at the top center says 'This publisher stores Robot Framework test reports for builds and shows summaries of them in project and build views along with trend graph.' with 'Install' and 'Cancel' buttons. The main area lists the 'Robot Framework' plugin, which is version 5.0.0 and was released 2 months ago. Other available plugins listed include 'Build Reports'.

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

Lab Worksheet



12. นำไฟล์ .robot ที่ทำให้แบบฝึกปฏิบัติที่ 7 (Lab#7) ไปวิ่งบน Repository ของนักศึกษา จนนั่นตั้งค่าที่
จำเป็นในหน้านี้ทั้งหมด 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

General

Enabled



Description

Lab 8.5

[Plain text](#) [Preview](#) Discard old builds [?](#) GitHub projectProject url [?](#)Advanced [▼](#) This project is parameterised [?](#) Throttle builds [?](#) Execute concurrent builds if necessary [?](#)Advanced [▼](#)

Lab Worksheet

Git ?

Repositories ?

Repository URL ? X

`https://github.com/653380204-6/UAT.git`

Credentials ? ▼

`653380204-6/*****`

+ Add

Advanced ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ? X

`*/main`

Add Branch

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 Thursday, January 30, 2025 at 4:35:11 AM Coordinated Universal Time; would next run at Thursday, January 30, 2025 at 4:50:11 AM Coordinated Universal Time.

- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

Lab Worksheet

Build Steps

≡ Execute shell ? X

Command
See the [list of available environment variables](#)

```
export PATH=$PATH:/usr/bin
. /myenv/bin/activate
robot test.robot
```

Advanced ▾

Add build step ▾

Post-build Actions

≡ Publish Robot Framework test results ? X

Directory of Robot output
Path to directory containing robot xml and html files (relative to build workspace)

Advanced ▾ Edited

Thresholds for build result ?

🟡 %

🔵 %

DEPRECATED! THIS FLAG DOES NOTHING! - Use thresholds for critical tests only

Include skipped tests in total count for thresholds

Add post-build action ▾

Lab Worksheet

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

```
export PATH=$PATH:/usr/bin  
. /myenv/bin/activate  
robot test.robot
```

Post-build action: เลือก 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. ลาก Build Now

Lab Worksheet

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

Console Output

[Download](#)
[Copy](#)
[View as plain text](#)

```

Started by timer
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/UAT
The recommended git tool is: NONE
using credential 5da474a1-1ffa-4366-b4ed-e10a1307036b
> git rev-parse --resolve-git-dir /var/jenkins_home/workspace/UAT/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/653380204-6/UAT.git # timeout=10
Fetching upstream changes from https://github.com/653380204-6/UAT.git
> git --version # timeout=10
> git --version # 'git version 2.39.5'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/653380204-6/UAT.git +refs/heads/*:refs/remotes/origin/*
# timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision b994f6a5b212609f6b2e980bbd92ec98964e8caf (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f b994f6a5b212609f6b2e980bbd92ec98964e8caf # timeout=10
Commit message: "Update test.robot"
> git rev-list --no-walk b994f6a5b212609f6b2e980bbd92ec98964e8caf # timeout=10
[UAT] $ /bin/sh -xe /tmp/jenkins3140005816538114909.sh
+ export PATH=/opt/java/openjdk/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/usr/bin
+ . /myenv/bin/activate
+ deactivate nondestructive
+ [ -n ]

```